Instituto de Computación - Facultad de Ingeniería - UdelaR

InTrack

Software Geográfico AVL para Ómnibus

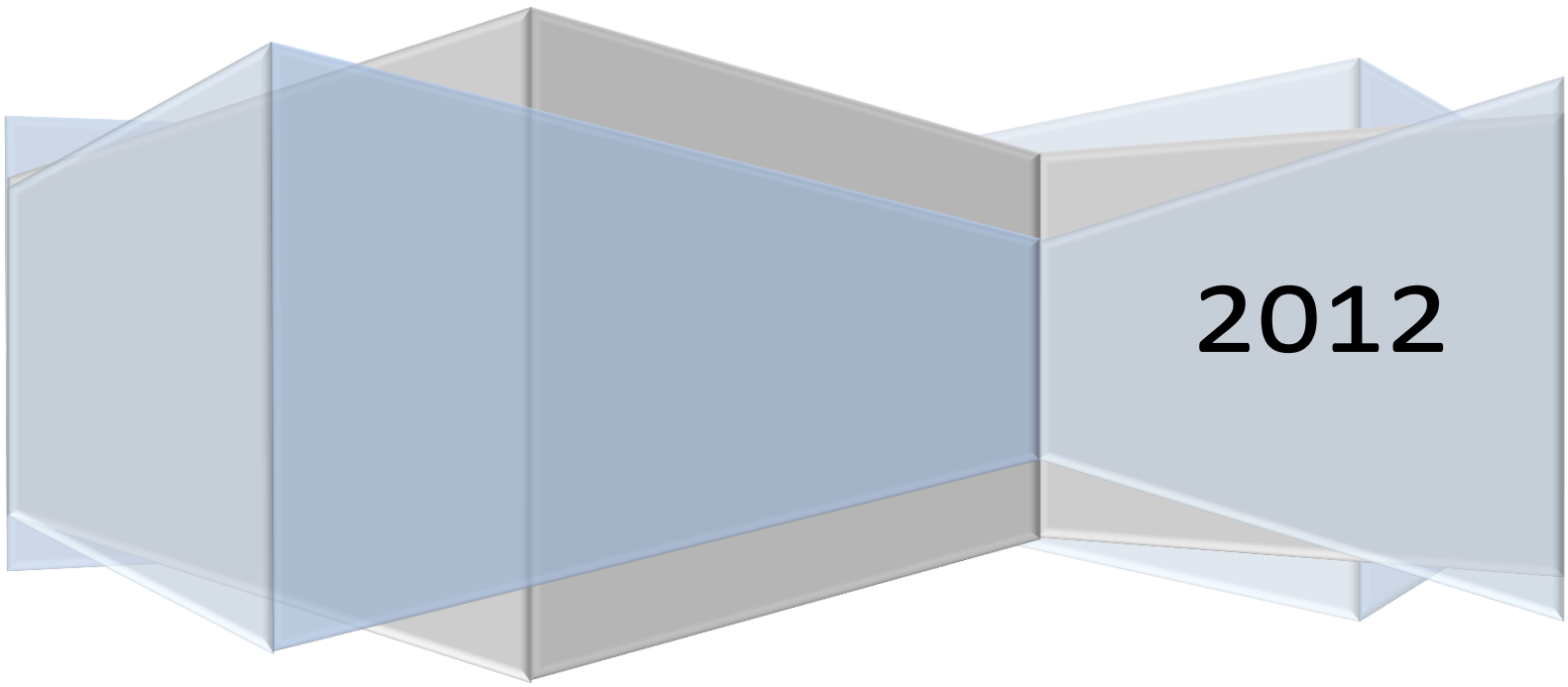
Integrantes

Bruno González, Máximo Mussini

Tutores

Pablo Rebufello, Sandro Moscatelli

INFORME DE PROYECTO DE GRADO



Resumen

El proyecto se enmarca en el ámbito de los Sistemas de Información Geográfica, del cual se utilizarán los conceptos espaciales para el posicionamiento geográfico, análisis espacial de datos en tiempo real, visualización y transmisión de información espacial entre dispositivos móviles. Surge como la combinación de tres puntos clave: evaluar las opciones de desarrollo de aplicaciones móviles, investigar la comunicación en tiempo real entre dispositivos móviles, y plasmar este aprendizaje en el desarrollo de un sistema con características de red social y funcionalidad geográfica, que esté optimizado en torno al transporte público y permita realizar seguimiento en tiempo real de usuarios y ómnibus.

Respecto al primer punto, resultaban de particular interés aquellas alternativas que permiten desarrollar para múltiples plataformas, por lo que se planteó como objetivo investigar GeneXus y HTML5 como opciones de desarrollo multiplataforma, evaluando sus virtudes y defectos en base a la experiencia de usuario del producto obtenido.

Como segundo objetivo, se decidió realizar una investigación de los protocolos y tecnologías de comunicación en tiempo real que se adecuaban a las redes móviles actuales, teniendo en cuenta el soporte existente de dichas tecnologías en las diferentes plataformas móviles.

Por último, una vez evaluadas las alternativas de desarrollo móvil y los métodos de comunicación disponibles, se comenzó el desarrollo del sistema, con la finalidad de que el mismo estuviera disponible para las plataformas móviles que cuentan con mayor presencia en el mercado.

Palabras clave

Sistemas de Información Geográfica, AVL, comunicación en tiempo real, red social.

Tabla de contenido

1 Introducción

El presente proyecto de grado involucra el diseño y construcción de una plataforma de servicios y una aplicación de software con el objetivo de compartir información geográfica entre dispositivos móviles en tiempo real. La plataforma presentará funcionalidades de un sistema AVL[1] permitiendo localizar ómnibus, y consultar tanto su ubicación, como información relacionada (recorrido, paradas) en tiempo real.

De forma similar la plataforma permitirá determinar la posición geográfica de los usuarios registrados en el sistema, brindando un mecanismo para la transmisión de datos punto a punto entre los usuarios del sistema, posibilitando la definición de un punto de encuentro, y permitiendo el seguimiento mutuo entre usuarios.

La plataforma incorpora características de las redes sociales; el acceso a la información de un usuario se regula mediante niveles de privacidad. Cada usuario tendrá una lista de usuarios asociados con los cuales puede compartir su posición geográfica y otros datos.

Finalmente, por tratarse de una aplicación orientada al público en general, será necesario prestar especial atención a la interfaz gráfica de la aplicación, apuntando a un diseño amigable e intuitivo sin comprometer su funcionalidad.

1.1 Motivación

Los dispositivos móviles se han vuelto parte del estilo de vida de la mayoría de las personas. Hay más de mil millones de dispositivos móviles en el mundo, y se estima que para fines de 2013 van a entrar al mercado casi mil millones de dispositivos más [1]. Este gran mercado potencial es lo que en primera instancia incentivó a realizar un proyecto de grado relacionado con el desarrollo de aplicaciones móviles.

A grandes rasgos, al momento de desarrollar una aplicación móvil hay dos opciones disponibles: desarrollo nativo y desarrollo web para móviles. En el contexto del proyecto resulta interesante analizar las ventajas y desventajas de estos dos enfoques.

Se destaca la necesidad de evaluar el uso de bibliotecas de software en ambos entornos, en particular ArcGIS [2], de modo de obtener una noción del porcentaje de reutilización y del costo de portar una aplicación móvil a otra plataforma.

A su vez, la comunicación en tiempo real es un área que ha cobrado gran interés en los últimos años debido a la proliferación de las redes móviles [3]. Por este motivo se propone investigar diferentes formas de realizar comunicación punto a punto en tiempo real entre dispositivos móviles.

1.2 Objetivos

Si bien el objetivo principal del proyecto es el desarrollo del sistema antes descrito, se pretende cubrir otros aspectos tales como determinar las dificultades y limitaciones en el desarrollo para dispositivos móviles, estudiar diferentes alternativas de comunicación en tiempo real, e investigar formas de reducir el consumo de batería de los dispositivos.

[1] AVL (Automatic Vehicle Locator)

En primer lugar se deberán analizar las características de los dispositivos móviles actuales, tanto a nivel de hardware como de software. En lo que refiere a software se estudiarán los sistemas operativos existentes, realizando una revisión de sus características principales y un breve análisis comparativo, evaluando además las distintas herramientas y ambientes de desarrollo disponibles para los mismos. En cuanto al hardware se tendrán en cuenta la capacidad de procesamiento y memoria de los dispositivos más representativos en la actualidad, junto con otras características como la capacidad gráfica, presencia de GPS, y funciones de red.

Además, se pretende evaluar hasta qué punto es posible desarrollar una aplicación portable con las herramientas y plataformas disponibles. En particular, se evaluará la posibilidad de desarrollar una aplicación multiplataforma mediante GeneXus X Ev2 o HTML5. En el caso de HTML5 interesa comparar sus prestaciones en relación a las de una aplicación nativa, tanto en rendimiento como en acceso a las características físicas del dispositivo.

Será necesario evaluar diferentes alternativas que permitan lograr una comunicación punto a punto entre dispositivos con una latencia mínima, por lo cual se investigarán protocolos y middleware de comunicación que pudieran ser utilizados con el propósito de transmitir información en tiempo real. Resultarán de especial interés aquellos métodos que tomen en cuenta la escasa autonomía de los dispositivos móviles actuales. A su vez, habrá que considerar la infraestructura de red subyacente, en particular la existente en Uruguay.

Por último, nos embarcaremos en el estudio de aquellas aplicaciones disponibles en el mercado que estén relacionadas con el proyecto, más concretamente, aquellas que intenten proveer funcionalidades similares.

Considerando la amplia gama de los temas anteriormente identificados, se hará énfasis en los que creemos son los más relevantes para la comprensión del proyecto, presentando brevemente el resto.

1.3 Contenido del documento

Este informe está organizado en siete capítulos de la siguiente forma:

Capítulo 1: Introducción

En el presente capítulo se plantea y define el problema, dando una breve introducción a los puntos más relevantes a tratar, dejando clara la motivación del trabajo y los objetivos planteados, así como los resultados esperados.

Capítulo 2: Estado del Arte

Se presenta un resumen del trabajo de investigación realizado sobre las tecnologías, lenguajes de programación y entornos de desarrollo a ser evaluados para la posterior implementación, mostrando las componentes esenciales para comprender las diferentes temáticas involucradas, dejando una versión más detallada con todas las alternativas estudiadas en el *Anexo I - Estado del Arte*.

Capítulo 3: Análisis y especificación de requerimientos

El objetivo principal de este capítulo es presentar las principales funcionalidades y características que deberá brindar el producto a desarrollar. El lector que esté interesado en

profundizar en los detalles de este análisis, podrá consultar el *Anexo II - Análisis y Descripción de Requerimientos*.

Capítulo 4: Diseño del sistema

En este capítulo se presenta una descripción general de la arquitectura del sistema, se muestran los aspectos más relevantes del diseño de las bases de datos utilizadas, así como la descripción de los servicios utilizados para su consulta, explicando y justificando las decisiones de diseño. Los aspectos detallados de esta etapa se pueden encontrar en el *Anexo III - Diseño del Sistema*.

Capítulo 5: Implementación

Se describen de forma general las decisiones tomadas acerca de la implementación del proyecto, las dificultades que se presentaron y cómo se resolvieron. En particular se cubren en detalle la comunicación en tiempo real entre dispositivos móviles, algoritmos geográficos utilizados, y el almacenamiento de datos.

Capítulo 6: Pruebas realizadas y resultados obtenidos

En este capítulo se resumen las pruebas realizadas sobre el sistema y los resultados obtenidos en las mismas. Se describe el plan de verificación implementado, las herramientas y técnicas utilizadas, y cómo se llevó a cabo la verificación a lo largo del proceso de desarrollo.

Capítulo 7: Conclusiones

El objetivo de este capítulo es el de presentar las conclusiones generales del proyecto, evaluando los resultados obtenidos y las dificultades encontradas. Se hace un resumen de lo que se planteó y finalmente se muestra lo que se pudo realizar.

Capítulo 8: Trabajo futuro

En este capítulo se muestra cómo puede extenderse o complementarse el trabajo realizado, haciendo un análisis autocrítico de lo que se logró y evaluando las diferentes oportunidades de mejora.

2 Estado del Arte

Para poder abordar el diseño del sistema antes descrito es necesario conocer las características de los dispositivos móviles actuales y las posibilidades existentes para realizar comunicación en tiempo real entre móviles.

A su vez, para realizar el desarrollo del sistema es fundamental conocer las plataformas y bibliotecas de software disponibles, en particular aquellas que brindan servicios o componentes de carácter geográfico.

En este capítulo se presentan aquellos conceptos y tecnologías que son importantes para comprender el trabajo realizado en el proyecto. Estos temas y el resto de la investigación realizada se presentan con mayor detalle en el *Anexo I - Estado del Arte*.

2.1 Dispositivos Móviles

Los dispositivos móviles se han convertido en una parte esencial de la vida diaria como herramientas de comunicación eficaces y cómodas, accesibles en todo momento y lugar. A su vez, la gran variedad de aplicaciones disponibles en mercados y la diversidad de servicios que estas proveen permiten que los usuarios obtengan una experiencia muy completa [4].

La evolución del teléfono móvil ocurrió a lo largo de cinco generaciones distintas, convirtiéndose en uno de los principales medios de comunicación a fines de la década del ’90 [5]. En alrededor de 20 años, esta tecnología conquistó a las masas y se convirtió en una necesidad básica para la mayor parte de la gente.

Originalmente, los teléfonos móviles estaban pensados para comunicar a la gente, sin importar el tiempo y lugar, permitiendo la transmisión simple de voz. Con el avance de la tecnología fueron evolucionando en sus características y funciones, convirtiéndose en dispositivos de entretenimiento que permiten a los usuarios almacenar su información personal, acceder a Internet, e incluso tener video-conferencias; acercándose cada vez más a las computadoras de escritorio en cuanto rendimiento [6], y dando lugar a grandes cambios.

En primer lugar, desde una perspectiva humana, la movilidad que brindan estos dispositivos permite a las personas acceder a sistemas computacionales en cualquier momento y en cualquier lugar, lo cual abre las puertas a un gran conjunto de nuevas aplicaciones. Estas aplicaciones admiten una gama de actividades diferente a la de las aplicaciones de escritorio, y pueden tomar provecho de la información que los dispositivos móviles poseen sobre el entorno del usuario, como ser su ubicación geográfica, para proporcionar mejores funcionalidades.

En segundo lugar, desde una perspectiva tecnológica, la movilidad logra un cambio global en la infraestructura de los sistemas computacionales; desde computadoras de escritorio, estáticas, homogéneas, y de gran potencia; hacia dispositivos de mano altamente dinámicos, heterogéneos, y con recursos limitados.

Si bien existe una poderosa tendencia en el desarrollo de la informática móvil, los dispositivos móviles aún se enfrentan a muchos desafíos en lo que respecta al uso eficiente de sus recursos (como ser batería y procesamiento), así como a la comunicación (específicamente la movilidad y la seguridad). Sumado a esto, las aplicaciones diseñadas para dispositivos móviles deben

hacer frente a las restricciones de ancho de banda de la red de datos en la que se encuentran. Estas limitaciones presentan un obstáculo significativo a la hora de asegurar la calidad de los servicios brindados.

A continuación, se presentan las características de hardware y software de los dispositivos móviles actuales, así como un breve análisis del mercado de sistemas operativos.

2.1.1 Hardware de Dispositivos Móviles

Un aspecto relevante en el desarrollo de aplicaciones para dispositivos móviles, es el hardware disponible en los dispositivos objetivo. Ya sea por las funcionalidades que provea la aplicación o por el rendimiento de la misma, es necesario tener en cuenta este factor que en muchos casos es determinante.

En lo que refiere al mercado de dispositivos móviles, las empresas que lo lideran son Samsung, Nokia y Apple [7]. A continuación se presenta una gráfica basada en un estudio de la empresa Gartner acerca de la venta de dispositivos móviles a usuarios finales hasta el segundo trimestre de 2012.

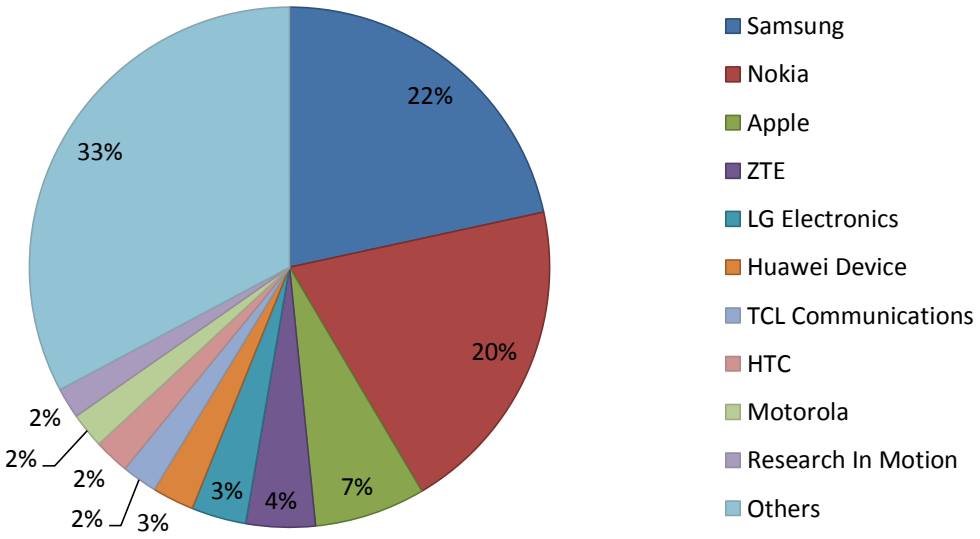


Figura 2.1: Proporción de ventas del mercado de dispositivos móviles (segundo trimestre 2012)

En la presente sección se analizará la capacidad de procesamiento y la capacidad gráfica de los dispositivos móviles para poder determinar el potencial de cómputo que estos poseen. Luego se realiza un estudio del sensor de GPS[2], ya que conocer sus características será de gran importancia para el proyecto debido a su naturaleza geográfica, culminando con un análisis de las tecnologías de batería, lo cual es de vital importancia ya que esta determina la autonomía del dispositivo móvil.

[2] GPS (Global Positioning System)

2.1.1.1 Procesador

Como se mencionó anteriormente, el procesador del dispositivo móvil es el que le dará a este la capacidad de procesamiento, lo que hace que forme parte del hardware determinante en lo que respecta a la ejecución de aplicaciones de usuario.

Cuando hablamos de procesadores en el ámbito de los teléfonos móviles, en realidad debemos referirnos a los llamados SoC[3], que son dispositivos que integran en un mismo chip varios componentes de un sistema [8] [9]. Los componentes que este integra son: núcleos del procesador, unidad de procesamiento de gráficos, memoria RAM[4] y ROM[5], e interfaces externas (por ejemplo: USB o tecnología inalámbrica, entre otros).

Concentrar los componentes esenciales de un dispositivo en un espacio relativamente pequeño, mejora la performance general y hace un uso más eficiente de la batería, además de abaratar los costos de ensamblado total del hardware del dispositivo.

Como los requerimientos de consumo y tamaño en un dispositivo móviles son mucho más acotados que en una computadora de escritorio, se han desarrollado arquitecturas pensadas específicamente para este tipo de dispositivos [8]. Hoy en día, ARM[6] es la arquitectura que lidera en lo que refiere a dispositivos móviles [10].

2.1.1.2 Capacidad gráfica

Otro de los dispositivos que se pueden encontrar dentro del SoC es la unidad de procesamiento de gráficos o GPU[7], la cual está muy próxima al procesador [11]. El uso que se le da a la GPU depende del sistema operativo y la estructura del SoC, pero básicamente se encarga de procesar tareas relacionadas con el procesamiento de gráficos, lo cual permite liberar a la CPU[8].

En lo que refiere a las tareas propias del sistema operativo, la GPU es utilizada para renderización 3D en juegos y aplicaciones. Los núcleos ARM no están diseñados para procesar este tipo de tareas, por lo que las mismas son delegadas a la GPU, la cual las procesará de forma más eficiente.

El sistema operativo Android tiene algunas particularidades con respecto al procesamiento de tareas de renderización. En un principio, como los dispositivos que corrían Android no poseían GPUs lo suficientemente potentes para encargarse completamente de este tipo de tareas, Google decidió delegarlas completamente a la CPU. Esto fue corregido en la versión 4.0 de este sistema operativo, ya que los SoCs actuales poseen GPUs con mejor capacidad de procesamiento. Por otro lado, iOS utiliza la GPU para la mayoría de las tareas de renderización [11].

---

[3] SoC (System-on-a-chip)
[4] RAM (Random-Access Memory)
[5] ROM (Read-Only Memory)
[6] ARM (Advanced RISC Machines)
[7] GPU (Graphics Processing Unit)
[8] CPU (Central Processing Unit)

2.1.1.3 GPS

Habiendo visto la capacidad de procesamiento general y específico para gráficos en un dispositivo móvil y cómo impacta dicha capacidad en el funcionamiento general del mismo, se continuará realizando un estudio acerca de los sensores de localización geográfica.

Con respecto a las tecnologías utilizadas por dichos sensores, el GPS[9] es uno de los métodos utilizados para obtener la posición del dispositivo. Este es un sistema de navegación satelital conformado por 32 satélites puestos en órbita por el Departamento de Defensa de los Estados Unidos, los cuales en un principio fueron utilizados con propósito militar y posteriormente habilitados para uso civil [12].

Vale la pena aclarar que el nombre correcto para este tipo de tecnologías es GNSS[10], que es una combinación de todos los sistemas de navegación satelital entre los cuales se encuentran GPS, GLONASS[11], WAAS[12], EGNOS[13], MSAS[14]. En el momento hay 32 satélites pertenecientes a la constelación GPS y 24 pertenecientes a GLONASS [13].

El sensor con el que cuentan los últimos dispositivos móviles del mercado no sólo procesa señales de los satélites GPS, sino que también procesa las de los satélites GLONASS. Al utilizar ambas señales, la cantidad de satélites que proveen información aumenta, lo cual mejora los resultados en tiempo real bajo circunstancias poco favorables (áreas urbanas, terreno montañoso o bosque) [13]. Según estudios de la empresa Qualcomm, utilizando la información provista por el servicio de GLONASS, combinado con el GPS, se puede obtener una precisión de hasta 2 metros [14].

Como ya se mencionó, el GPS es una de las formas con las que el sensor de localización puede obtener la posición geográfica del dispositivo móvil. También puede obtenerse mediante triangulación por torres celulares o redes WiFi, mediante una base de datos que contenga la posición geográfica de cada una de estas [15].

Los dispositivos móviles pueden seleccionar diferentes alternativas para obtener su posición geográfica (GPS, torres celulares o WiFi), dependiendo de la precisión que se desee obtener y de las circunstancias del momento.

2.1.1.4 Batería

Una de las características más importantes de un dispositivo móvil, más allá de la capacidad de procesamiento o sensores disponibles, es la autonomía. Un dispositivo móvil puede tener las mejores prestaciones del mercado a nivel de hardware, pero si cuenta con pocas horas de autonomía, sus capacidades se verán totalmente limitadas.

En el contexto del proyecto la batería es un factor muy importante a tener en cuenta, ya que se utiliza el sensor GPS y la red de datos para la trasmisión de información (mediante 3G o 4G), que son funciones que realizan un gran consumo de batería [16].

---

[9] GPS (Global Positioning System)
[10] GNSS (Global Navigation Satellite System)
[11] GLONASS (Global'naya Navigatsionnaya Sputnikovaya Sistema)
[12] WAAS (Wide Area Augmentation System)
[13] EGNOS (European Geostationary Navigation Overlay Service)
[14] MSAS (Multi-functional Satellite Augmentation System)

La batería de ion de Litio es el tipo de batería que se usa actualmente en dispositivos móviles. Este tipo de batería es usada en estos dispositivos, ya que el Litio es el más liviano de los metales para este tipo de fines y posee la mayor densidad de potencial. También tiene uno de los ciclos de vida más altos y la ventaja de no tener el conocido efecto memoria de las baterías [17].

En cuanto al uso de la batería en las aplicaciones, existen múltiples estudios que tratan el tema [18] [19] y muestran en dónde se consume mayor cantidad de energía. Haciendo un buen uso de las técnicas propuestas y de las recomendaciones de los expertos, se puede reducir el consumo de una aplicación entre un 20% y un 65% [19].

Dado que cada sistema operativo hace diferente uso del hardware disponible, existen recomendaciones propias de cada compañía en relación al desarrollo de aplicaciones, para que estas puedan utilizar este recurso de forma adecuada. Por ejemplo, Google presentó una charla [20] en la que hacía recomendaciones específicas de qué clases usar, qué funciones no eran recomendables para determinados entornos y cómo podía hacerse un uso eficiente de los recursos de software para evitar el consumo innecesario de batería.

2.1.2 Sistemas Operativos para Móviles

El sistema operativo (OS) es el software que se comunica con el hardware de una computadora, maneja los recursos, y provee una interfaz de usuario. Todos los teléfonos móviles utilizan algún tipo de sistema operativo, pero en los últimos años con la aparición de los dispositivos inteligentes el sistema operativo de un teléfono se ha convertido en un factor más que determinante.

El crecimiento de la demanda de dispositivos móviles llevó a que varios grandes fabricantes intentaran ganar una porción del mercado desarrollando su propio sistema operativo para móviles. Esto último provocó una fragmentación del mercado [21], acarreando tanto consecuencias positivas como negativas.

Por un lado, fomentó una situación de libre competencia donde cada fabricante intentaba mejorar y pulir sus productos para poder diferenciarse del resto. Esta carrera por atraer consumidores resultó en mejoras a la experiencia de usuario.

Por otro lado, dificultó el desarrollo de aplicaciones móviles, ya que cada plataforma tiene su propio lenguaje nativo, herramientas de desarrollo específicas, y un conjunto de peculiaridades. Esto quiere decir que para apuntar a la totalidad de los usuarios móviles es necesario desarrollar la aplicación en cada una de las plataformas, aumentando el costo del desarrollo.

A continuación se presenta una gráfica basada en un estudio de la empresa Gartner acerca de la venta de dispositivos móviles por sistema operativo en el primer trimestre de 2012 [21].

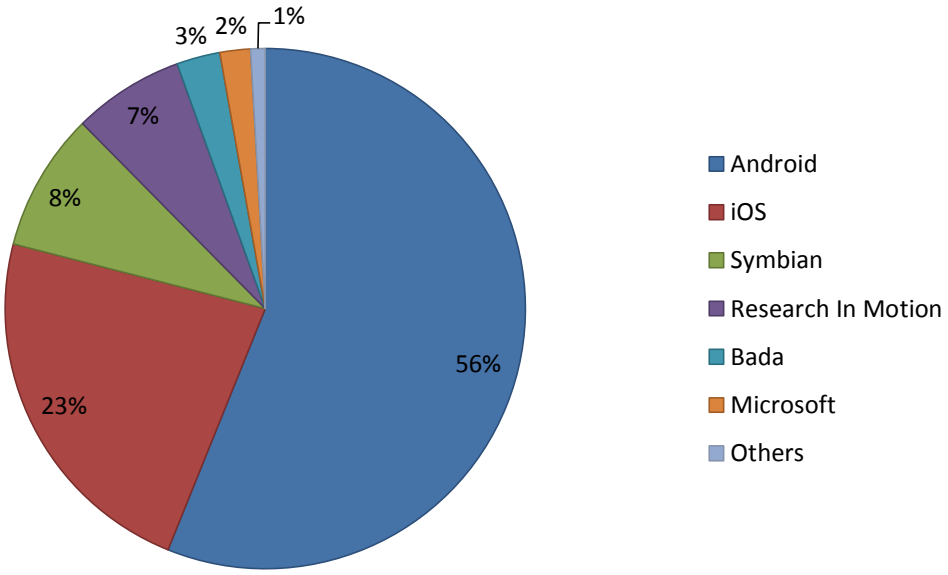


Figura 2.2: Proporción de ventas por sistema operativo (primer trimestre 2012)

La mayor parte del mercado se divide entre dos sistemas operativos para móviles, estos son Android e iOS. Más allá de que estos sean los más populares, se puede percibir un crecimiento lento y continuo de Windows Phone.

A continuación se presentan los conceptos más relevantes de los sistemas operativos Android e iOS. No se hace mención de los demás sistemas operativos por el hecho de que no aportan información necesaria para la comprensión general del proyecto.

2.1.2.1 Android

Android es una plataforma integral y de código abierto diseñada para dispositivos móviles. Está respaldado por Google, y pertenece a la *Open Handset Alliance* [22], un consorcio de más de 80 compañías líderes en las industrias del hardware y del software creado para "acelerar la innovación en móviles y ofrecer a los consumidores una experiencia móvil mejor, más completa, y menos costosa".

A la hora de diseñar Android se tomaron en cuenta aquellas restricciones que afectaban a los dispositivos móviles, y que probablemente no iban a cambiar en un futuro cercano [23]. En primer lugar, asumieron que el rendimiento de la batería no iba a mejorar demasiado en los próximos años, y en segundo lugar, que el tamaño reducido de los dispositivos móviles limitaba su memoria y capacidad de procesamiento.

El no haber realizado suposiciones con respecto al tamaño de la pantalla, resolución, procesador, y demás componentes del dispositivo, esto es, separar el hardware del software que corre sobre él, permitió que Android pudiera correr sobre una mayor variedad de dispositivos físicos.

Además, el hecho de ser gratuito permitió recortar gastos a las compañías de hardware, lo cual resultó en dispositivos de menor costo. Al ser un sistema operativo de código abierto permitió

que miles de desarrolladores en todo el mundo contribuyeran a mejorarlo, y que las compañías de hardware pudieran portarlo fácilmente a su hardware específico. Debido a su licencia Apache/MIT [24] [25] también es posible extenderlo y usarlo para una variedad de propósitos, sin obligación de liberar el código modificado.

Actualmente domina el mercado global de dispositivos móviles con más del 50% [26].

Java y Dalvik

Java es un lenguaje de programación de alto nivel basado en clases utilizado para propósito general, que soporta el paradigma de orientación a objetos. Este lenguaje toma varios conceptos de C y C++, aunque varía en su organización, ya que toma conceptos de otros lenguajes de programación. Uno de los aspectos en los que varía es en el del manejo implícito de la memoria, ya que Java utiliza un *garbage collector* para liberar la memoria no utilizada o inaccesible.

También podemos decir que es un lenguaje fuertemente tipado, lo que permite distinguir claramente entre errores en tiempo de compilación y errores en tiempo de ejecución. En tiempo de compilación, el código del programa es traducido a código de máquina, y en tiempo de ejecución, se cargan y se vinculan las clases necesarias para ejecutar el programa [27].

A diferencia de otros lenguajes de programación, para los cuales el código del programa es compilado para una arquitectura y un sistema operativo específicos, la ejecución de los programas Java se hace mediante la máquina virtual Java (JVM[15]). Este modo de ejecución permite que los programas puedan ser escritos y compilados una única vez, permitiendo que los mismos se ejecuten en diferentes sistemas operativos y arquitecturas. También presenta ventajas de seguridad, ya que tanto el código potencialmente malicioso, como los errores graves de un programa de una fuente externa, son ejecutados en un entorno seguro (*sandbox*) [28]. La máquina virtual Java sido utilizada para entornos de escritorio (JSE[16]), servidores (JEE[17]) y ecosistemas móviles (JME[18]), con diferentes configuraciones que permitieran soportar múltiples dispositivo.

A pesar de que Google eligió a Java como el lenguaje de programación para Android, descartó el uso de JME y la máquina virtual Java, eligiendo una alternativa diferente llamada Dalvik. También eligió una implementación alternativa y limitada de las bibliotecas estándar de Java, lo cual difiere de la tendencia que seguían otros lenguajes de programación que utilizaban la máquina virtual Java como plataforma de ejecución [29].

Un aspecto relevante a tener en cuenta es que Android fue pensado para trabajar sobre múltiples dispositivos, con diferentes limitantes en cuanto al poder de cómputo, cantidad de memoria RAM, volumen de almacenamiento y autonomía, y sus aplicaciones deben correr en un entorno seguro. Estos requerimientos podrían ser un buen indicio para elegir una máquina virtual como solución, si no fuera por las limitantes anteriormente mencionadas. Justamente por esta razón fue que Google decidió implementar su propia máquina virtual Java, la cual

[15] JVM (Java Virtual Machine)
[16] JSE (Java Standard Edition)
[17] JEE (Java Enterprise Edition)
[18] JME (Java Micro Edition)

podría brindar solución a los requerimientos de un entorno seguro, teniendo en cuenta el hardware anfitrión [29].

Cada aplicación Android corre su propio proceso en su propia instancia de la máquina virtual Dalvik, la cual está diseñada para correr múltiples instancias de forma eficiente. El hecho de que cada proceso corra su propia máquina virtual ayuda a prevenir que múltiples aplicaciones terminen su ejecución por la falla de una.

Android SDK

Android provee todas las herramientas necesarias para desarrollar aplicaciones móviles sobre la plataforma a través del *Android SDK* [30]. Entre ellas se encuentra un emulador de dispositivos que permite probar las aplicaciones sin necesidad de tener un dispositivo físico.

El módulo *SDK Tools* [31] incluye el conjunto completo de las herramientas de desarrollo y depuración, así como otras utilidades que son necesarias para desarrollar una aplicación. También cuenta con el módulo *SDK Platform-tools*, que contiene herramientas dependientes de la plataforma para desarrollar y depurar aplicaciones. Estas herramientas soportan las últimas funcionalidades de la plataforma Android y solo se actualizan cuando hay una nueva plataforma disponible. Vale la pena destacar que las nuevas actualizaciones son compatibles con las plataformas anteriores.

Para cada nueva versión de Android se libera un *SDK Platform*, que incluye una biblioteca totalmente compatible con esa versión del sistema operativo. Para construir una aplicación es necesario especificar una plataforma SDK como objetivo. Cada plataforma ofrece una o más imágenes de sistema (tanto para ARM, como para x86), las cuales son requeridas por el emulador de Android.

2.1.2.2 iOS

iOS es un sistema operativo para dispositivos móviles desarrollado y distribuido por Apple, el cual fue presentado junto con el lanzamiento del primer iPhone [32].

Este sistema operativo fue diseñado específicamente para la arquitectura de hardware de Apple, específicamente para dispositivos móviles como el iPhone, el iPod y el iPad. Si bien esta decisión puede parecer un tanto restrictiva, ya que limita la cantidad de dispositivos sobre las que puede correr el sistema operativo, la motivación detrás de la misma es asegurar el rendimiento y mejorar la interacción entre el hardware y el software.

Aplicaciones de terceros

Desde la primer versión del iPhone, iOS soporta aplicaciones creadas con el estándar Web 2.0, las cuales pueden acceder a los servicios del teléfono sin afectar la seguridad y confiabilidad del mismo [33].

A partir de la versión 2.0 del sistema operativo para dispositivos móviles, liberada en Junio del 2008, Apple liberó un SDK para el desarrollo de aplicaciones nativas para iPhone y iPod Touch, el cual provee al desarrollador un buen conjunto de APIs con las cuales interactuar con el hardware del dispositivo. Junto con la versión del software, se agregó una nueva aplicación llamada *Apple Store*, con la que los usuarios pueden navegar, buscar y comprar aplicaciones de terceros [34].

iOS SDK

El SDK que provee Apple contiene todas las herramientas necesarias para el desarrollo de aplicaciones nativas para iOS. Este SDK también brinda las herramientas para la instalación y prueba de las aplicaciones, para lo que se incluyen varios componentes clave para cumplir con las necesidades del desarrollador.

Provee un IDE llamado *XCode*, con el que se desarrollan las aplicaciones con todo lo que esto implica: edición, compilación, ejecución y pruebas sobre el código escrito [35]. También brinda una herramienta llamada *Instruments*, con la que se pueden hacer análisis de performance y pruebas sobre la aplicación, pudiendo colectar información acerca del comportamiento de la misma en tiempo de ejecución [36]. Para la prueba de las aplicaciones se utiliza el *iOS Simulator*, que simula el sistema operativo iOS y sus funcionalidades. Este simulador no sustituye las pruebas en un dispositivo real, por lo que también es posible configurar un dispositivo para su uso en el desarrollo de aplicaciones [37].

Objective-C y Cocoa Touch

Objective-C es un lenguaje de programación orientado a objetos, utilizado como lenguaje principal de desarrollo de aplicaciones para OS X e iOS [38]. Es un *superset* del lenguaje C y hereda la sintaxis, los tipos primitivos y el flujo de control de las sentencias, agregando nueva sintaxis para la definición de métodos y clases.

Este es un lenguaje sumamente dinámico, ya que delega la mayor cantidad de definiciones posibles del tiempo de compilación (*compile time*) y de enlace (*link time*) al tiempo de ejecución. Esto hace que no sólo sea necesario un compilador, sino que también un sistema de tiempo de ejecución en el que se ejecute el código compilado [39].

En el ámbito del paradigma de orientación a objetos, podemos decir que una aplicación es desarrollada en base a una “red de objetos”. En este caso, los objetos son instancias de clases programadas en el lenguaje Objective-C, entre las cuales hay un subconjunto que tendremos que desarrollar y otras serán provistas por los *frameworks* Cocoa o Cocoa Touch [38].

Cocoa Touch es un conjunto de *frameworks* que proveen una capa de abstracción para el desarrollo de aplicaciones sobre el sistema operativo iOS. Implementan varios patrones de diseño, haciendo un foco especial en aquellas aplicaciones que se ejecutan en interfaces con pantalla táctil. Estos son los que permiten el acceso a los controles de la interfaz gráfica, botones y vistas de pantalla completa [40].

La mayor parte de este *framework* está escrito en el lenguaje Objective-C e incluye todo lo necesario para el desarrollo de todo tipo de aplicaciones, proveyendo la API para acceder al hardware del dispositivo y dejando abierta la posibilidad de acceder directamente al sistema operativo si es necesario.

Las aplicaciones nativas son desarrolladas con este conjunto de *frameworks* con el lenguaje Objective-C y son instaladas físicamente en el dispositivo (a diferencia de las aplicaciones Web), lo que hace que estén siempre disponibles (también en modo de vuelo), sin necesidad de conexión a Internet [41].

2.2 Comunicación y Protocolos

Gran parte del atractivo de los dispositivos móviles proviene de su capacidad de comunicación inalámbrica. Si bien esta capacidad ha ido mejorando con el tiempo, a la hora de construir un sistema y evaluar la viabilidad del mismo es de gran importancia tener en cuenta si la infraestructura de red subyacente es capaz de cumplir sus requerimientos de conectividad.

El sistema planteado en este proyecto tiene requerimientos de *comunicación en tiempo real* [42], lo cual quiere decir que es necesario que todos los usuarios intercambien información de forma instantánea, o con muy baja latencia. En este contexto el término "tiempo real" es un sinónimo de "en vivo".

Este requerimiento entra en conflicto con las limitaciones de batería de los dispositivos móviles actuales ya que incluso un uso moderado de las funciones de red consume mucha energía [43], drenando rápidamente la batería del dispositivo.

En la presente sección se realiza un breve repaso sobre la evolución de la telefonía celular, y se analizan algunos métodos de comunicación en tiempo real que reducen el consumo de batería, permitiendo mejorar la autonomía de los dispositivos al utilizar la aplicación. Por último, se investigarán *frameworks* y *middleware* de comunicación disponibles, como una forma de reducir el tiempo de desarrollo del sistema.

2.2.1 Telefonía Celular

Desde sus inicios a finales de los años '70 [44], la telefonía celular ha evolucionado muy rápidamente, revolucionando drásticamente las actividades que se realizan diariamente. Los teléfonos celulares se han convertido en una herramienta primordial de uso cotidiano.

A pesar que la telefonía celular fue originalmente concebida para transmitir datos de voz debido a las limitaciones tecnológicas de esa época, la tecnología celular de hoy en día es capaz de brindar servicios tales como transmisión de audio, video, y datos en general, convirtiéndose en una de las formas de comunicación más utilizadas.

Las redes celulares cubren grandes áreas geográficas a través de múltiples estaciones base, utilizando métodos de división de la señal para evitar interferencia, tales como CDMA[19] [45]. Cada dispositivo móvil se conecta con una única estación base a la vez, pudiendo cambiar de estación a medida que se mueve. Una vez conectado a una estación base, el dispositivo móvil puede conectarse a otros dispositivos (móviles o estáticos) utilizando la estación base como intermediario.

El núcleo de la red puede ofrecer servicios al dispositivo móvil, dependiendo dichos servicios de la implementación del sistema, según la generación de telefonía móvil.

2.2.1.1 Generaciones de Telefonía Celular

La telefonía celular, al igual que muchas otras tecnologías, está en continua evolución y son varias las mejoras que se han constatado en pocos años, las cuales han sido potenciadas con la aparición de los teléfonos inteligentes [46]. Estos cambios vertiginosos que esta tecnología experimenta son agrupados por generaciones, cada una con mejoras en la velocidad de

[19] CDMA (Code Division Multiple Access)

transmisión de datos, mejoras de seguridad y privacidad, y optimizaciones para transmisión de datos multimedia.

Situación actual en Uruguay

En Uruguay hace más de 10 años que se cuenta con tecnología GSM (2G) que incorporó la tarjeta SIM[20], lo que permitía cambiar el aparato celular conservando el mismo número telefónico. Luego llegó la tecnología GPRS (2.5G) que permitía acceder a internet y la tecnología EDGE (2.75G) que duplicaba las velocidades de la anteriormente mencionadas [47].

Actualmente las tres empresas de telefonía (Antel, Movistar y Claro) proveen (hace ya algunos años) servicios de tercera generación (3G y 3.5G) y a fines del año 2011 la empresa Claro lanzó su producto 4G.

Lo que Claro presentó como 4G es en realidad tecnología HSDPA+ [48] y hay varias controversias acerca de las afirmaciones de la empresa acerca de la clasificación de esta tecnología como 4G. Fernando Leis, director de Marketing de Movistar, afirmó que esto era una confusión y que la tecnología HSDPA+ es considerada de tercera generación [49]. El gerente de Claro, Horacio Alvarellos, afirma que según resolvió la ITU[21], la tecnología HSPA+ se definió como 4G [50], por lo que Claro sigue en su posición de promocionar dicha tecnología como 4G.

Movistar también lanzó su servicio basado en HSPA+, llamado Internet Fácil Turbo [51]. Este servicio no es presentado como 4G, porque Movistar no considera esta tecnología dentro de la cuarta generación [47].

Antel, por su parte, el 12 de diciembre de 2011 presentó Internet Vera, basada en tecnología de cuarta generación LTE. En Antel se desarrollaron planes piloto y de prueba en centrales de Montevideo y Punta del Este [52]. Antel afirma que dependiendo de las condiciones operativas, se pueden alcanzar velocidades de 20 Mbits/s, 50 Mbits/s, 100 Mbits/s e incluso superarlas [53].

Hasta el momento, en Uruguay no está operativa esta tecnología ya que la URSEC[22] debe liberar y asignar la frecuencia de 700 MHz a las empresas de telefonía, la cual se utiliza en la implementación de LTE [49].

2.2.2 Message-Oriented Middleware

Se define *message-oriented middleware (MOM)* como una infraestructura de software o hardware que provee comunicación distribuida en base a un modelo de interacción asíncrona. Los participantes de un sistema basado en MOM [54] interactúan sin tener que bloquearse y esperar al enviar un mensaje, pueden continuar su ejecución luego del envío.

Esto permite que se envíen mensajes cuando el emisor o el receptor no están activos o disponibles durante el tiempo de la ejecución. A raíz de esto, una aplicación que envía un mensaje no tiene garantía de que el mensaje será leído por otra aplicación, ni recibe garantía

[20] SIM (Subscriber Identity Module)
[21] ITU (International Telecommunication Union)
[22] URSEC (Unidad Reguladora de Servicios de Comunicaciones)

alguna sobre el tiempo que demorará en ser entregado. Estos aspectos dependen principalmente de la aplicación que recibe.

Por lo general, los MOM desarrollados para interactuar con múltiples plataformas proveen una API común a todos los sistemas que soportan. El resultado es una capa distribuida de comunicación que hace posible distribuir los módulos de aplicación sobre plataformas heterogéneas, pero permitiéndole al desarrollador abstraerse de los detalles de las mismas.

Estos sistemas presentan cuatro características muy importantes; bajo acoplamiento, alta confiabilidad, escalabilidad, y alta disponibilidad. Debido a los requerimientos del proyecto, resulta sumamente útil adoptar una arquitectura de software similar a los MOM, de forma de poder construir una capa de comunicación entre dispositivos móviles.

2.2.3 Tecnología *Push*

El término se refiere al estilo de comunicación a través de Internet donde el servidor es quien realiza la petición para iniciar una transacción. Esto se contrasta con la forma típica, en la que el cliente es quien realiza el pedido para iniciar la transmisión de información [55] [56].

A menudo los servicios *push* están basados en un modelo publicar-suscribir, por lo que se conocen las preferencias del cliente en cuanto a la información que desea recibir. Un cliente puede *suscribirse* a diferentes *canales* de información. Cuando un canal tiene nuevo contenido disponible, el servidor envía la información al usuario.

Se pueden encontrar ejemplos de servicios *push* en aplicaciones de mensajería, conferencias de audio y video en tiempo real, y en aplicaciones web que tienen requerimientos de tiempo real, como sitios de juegos y apuestas en línea, subastas, resultados de deporte y monitoreo de sensores, entre otros.

2.2.3.1 Implementación

Para implementar la comunicación *push* existen diferentes opciones, estando basadas la mayoría en el protocolo HTTP[23]. Algunas de las implementaciones más utilizadas son *HTTP streaming*, *pushlets, BOSH, long polling,* y más recientemente, *WebSockets*. A continuación se presentan estas dos últimas técnicas en más detalle.

Long polling (XHR)

Es una técnica que permite emular la tecnología *push*. Al igual que en la técnica de *polling* tradicional, el cliente realiza un pedido de información. La diferencia radica en que el servidor aguarda a tener información disponible antes de enviar una respuesta al cliente. Por lo general luego de recibir una respuesta el cliente volverá a realizar un pedido de información, de forma tal que la mayor parte del tiempo el servidor tenga una petición pendiente que pueda usar para enviar información en respuesta a un evento [57].

Web Sockets

WebSockets es una técnica de comunicación bidireccional sobre un socket TCP, permite el envío de mensajes con baja latencia y alta frecuencia. Esto facilita la comunicación en tiempo

[23] HTTP (Hypertext Transfer Protocol)

real entre cliente y servidor. Al ser bidireccional provee soporte directo para *push,* lo cual combinado con la baja latencia en el envío de mensajes haría que desplace a otras técnicas como *long polling* y *BOSH*. Aún no es un estándar, pero varios de los grandes navegadores soportan *WebSockets* [58].

2.2.3.2 Push en Dispositivos Móviles

Las dos grandes plataformas para móviles de la actualidad, Android e iOS, cuentan con soporte para realizar *push* a estos dispositivos, provistos respectivamente por Google y Apple. La utilización de estos servicios puede ayudar a reducir drásticamente el consumo de batería ya que se reutiliza una misma conexión para todas las aplicaciones del dispositivo. Estos servicios se pueden catalogar como MOM, ya que manejan todos los aspectos de encolamiento y entrega del mensaje.

Push en Android

Google Cloud Messaging for Android (GCM) es un servicio que permite que los desarrolladores envíen datos desde servidores a sus aplicaciones de Android. Esto podría ser un mensaje liviano indicando que hay una nueva actualización disponible, o un mensaje con hasta 4kb de contenido, para que aplicaciones como las de mensajería instantánea puedan consumirlo directamente [59].

Push en iOS

Apple Push Notification service (APNs) es un servicio que provee una forma eficiente de propagar información a dispositivos de Apple (iPhone, iPad, iPod). Cada dispositivo establece una conexión encriptada con el servicio y recibe notificaciones a través de esta conexión persistente. Si llega una notificación para una aplicación cuando ésta se encuentra inactiva, el dispositivo alerta al usuario de que hay datos disponibles para la misma. Los desarrolladores originan las notificaciones desde el software de sus servidores [60].

2.2.4 Node.js

Node.js es una plataforma que pretende facilitar el desarrollo de software, específicamente el orientado a resolver problemas de redes de computadoras, que necesiten escalar con facilidad en cortos períodos de tiempo [61]. Utiliza JavaScript como lenguaje de scripting[24], utilizando un modelo basado en eventos no bloqueantes.

Las características antedichas hacen que Node.js sea adecuado para entornos distribuidos y aplicaciones que intercambian grandes volúmenes de datos en tiempo real [62]. Es utilizado frecuentemente para el desarrollo de servidores web, en el que múltiples clientes pueden ser atendidos de forma concurrente, ya que la infraestructura basada en eventos de Node.js resulta apropiada para este tipo de sistemas, que presentan una alta tasa de operaciones de entrada/salida.

[24] Lenguaje de Scripting: Lenguaje de programación que soporta la escritura de scripts, los cuales son escritos para un entorno particular y son utilizados para automatizar determinadas tareas.

2.2.4.1 Socket.IO

Node.js resulta de especial interés para el proyecto debido a Socket.IO, una biblioteca para esta plataforma que hace posible la implementación de *WebSockets* en todos los navegadores [63]. Es una API de *WebSockets* desarrollada en JavaScript, utilizada para el desarrollo de aplicaciones de tiempo real, la cual permite abstraerse del mecanismo de transporte utilizado por cada aplicación.

Esto quiere decir que el desarrollador no debe preocuparse de si la conexión será establecida mediante *WebSockets*, *XHR polling*, Flash u otro mecanismo, dado que Socket.IO detecta los mecanismos soportados por el entorno y selecciona el más conveniente, o el de mayor preferencia según los especificados por el desarrollador [64] [65].

Socket.IO va más allá de lo que sería un *polyfill*[25] de *WebSockets*, ya que provee ciertas funcionalidades que no se encuentran en la API de *WebSockets*, como la verificación de conexión (*heartbeat messages*), soporte de desconexión y *timeouts* [66].

Los mecanismos soportados por Socket.IO son los siguientes: *WebSockets*, *Adobe Flash Socket*, *XHR polling, XHR multipart streaming*, *forever iframe*, y *JSONP polling* [67].

2.3 Desarrollo de Aplicaciones

Como se mencionó anteriormente, el mercado de dispositivos móviles se encuentra fragmentado, cada plataforma tiene su propio lenguaje nativo, herramientas de desarrollo específicas, y adicionalmente, un paradigma de diseño particular.

Esto implica que para desarrollar una aplicación móvil que pueda ser utilizada por la totalidad de los usuarios móviles es necesario desarrollar una aplicación para cada una de las plataformas, lo cual hace que el costo del desarrollo sea más elevado.

Por este motivo, a la hora de construir una aplicación a menudo se debe decidir para qué subconjunto de plataformas se desarrollará la aplicación, lo cual reduce el costo de desarrollo pero también reduce el mercado potencial de la misma. Es conveniente, entonces, encontrar alternativas al desarrollo nativo de aplicaciones, que permitan desarrollar aplicaciones multiplataforma, utilizando lenguajes y tecnologías estándar.

A continuación se presentan las opciones de desarrollo móvil disponibles en la actualidad, mencionando también tecnologías de desarrollo web, y finalmente, tecnologías y herramientas de desarrollo híbrido que permiten crear aplicaciones que funcionan en múltiples plataformas.

2.3.1 Desarrollo Nativo

Una aplicación móvil nativa es un programa para teléfonos inteligentes que ha sido desarrollado para un ambiente de ejecución específico, por ejemplo, el ambiente Objective-C en iOS. Al correr directamente sobre el sistema operativo, las aplicaciones nativas pueden hacer uso de características de hardware y software específicas del dispositivo.

Entre otras cosas, las aplicaciones nativas pueden utilizar los componentes de interfaz gráfica que provee cada plataforma, interactuar con otras aplicaciones del dispositivo como el

[25] Polyfill: Una biblioteca que imita una API existente, y cuenta con diferentes implementaciones para brindar la misma funcionalidad cuando no es posible utilizar alguna de ellas.

calendario y la agenda de contactos, y permiten tener un control más preciso sobre las funcionalidades de bajo nivel del dispositivo, como utilización del GPS, acceso a red, gestos, y control del consumo de batería. De esta forma es posible obtener un mejor rendimiento, y brindar una mejor experiencia de usuario.

Actualmente las plataformas de dispositivos móviles más populares para desarrollo nativo son Android e iOS. Cada plataforma tiene sus propias convenciones de usabilidad en los controles, la navegación, y todo lo que involucra el diseño de interacción [68] [69]. Esto es muy bueno para los usuarios, ya que esa consistencia entre las aplicaciones de la plataforma ayuda a que tengan menor dificultad a la hora de interactuar con una nueva aplicación; los gestos utilizados, las opciones de menú, y otros componentes de la interfaz se comportan de forma predecible.

Debido a que una aplicación nativa está desarrollada para un ambiente particular, para que funcione en más de una plataforma es necesario portar la implementación. Esto significa que al desarrollar aplicaciones nativas, el costo varía en función del número de plataformas soportadas, por lo cual es necesario implementar las funcionalidades de forma específica para cada una de ellas.

Todas las plataformas móviles cuentan con una tienda de aplicaciones, las cuales permiten que los usuarios descubran y adquieran nuevas aplicaciones para sus dispositivos. Estas resultan de gran importancia ya que además de distribuir las aplicaciones, permiten monetizarlas mediante mecanismos de venta o suscripción. Muchas de las tiendas cuentan con un proceso de selección de aplicaciones [70] [71], mediante el cual controlan que estas cumplan con los estándares de calidad, y las políticas impuestas por la tienda.

Existen ciertas herramientas que permiten reutilizar parte de la lógica entre los portes de la aplicación, como el acceso a datos y conexiones de red. Esto se logra mediante la utilización de un lenguaje común para definir las operaciones, que luego es traducido a código nativo para cada plataforma. Dentro de esta categoría se encuentra la herramienta GeneXus, que se describe a continuación.

2.3.1.1 GeneXus

GeneXus es una herramienta desarrollada por la empresa Artech [72], la cual automatiza algunos procesos del desarrollo de software, lo que permite que el analista pueda enfocarse en comprender la realidad a modelar y los problemas del usuario, delegando gran parte de la codificación a la herramienta.

GeneXus automatiza aquellas tareas rutinarias, como son: normalización de los datos, mantenimiento y generación de la estructura de las bases de datos y de los programas de aplicación [73]. En cuanto a lo que a programas de aplicación se refiere, GeneXus genera automáticamente los que implementan las operaciones de ABM[26] y permite aplicar algunos patrones de diseño que facilitan la construcción de programas de consulta.

[26] ABM (Alta, baja y modificación)

Desarrollo para Dispositivos Móviles

Como se mencionó anteriormente, GeneXus parte del conocimiento del usuario y de la representación del mismo en la base de conocimiento, con lo que luego generará los programas. Debido al creciente uso de los dispositivos móviles, Artech ha desarrollado un motor que permite generar programas para plataformas móviles, a saberse Android, iOS y BlackBerry [74].

Entre las características que brinda GeneXus para el desarrollo de aplicaciones para dispositivos móviles se encuentran:

- Desarrollo de aplicaciones en múltiples plataformas, creándolas de forma genérica y pudiendo instalarlas en cualquiera de las plataformas mencionadas.
- Creación de aplicaciones nativas en los lenguajes de programación de cada plataforma: Java para Android, Objective-C en iOS y Java para BlackBerry.
- Integración con proyectos GeneXus existentes.
- Publicación de la aplicación en la nube.
- Pruebas de la aplicación con simuladores nativos.
- Publicación de la aplicación en los Marketplace de cada compañía (AppStore, Google Play y BlackBerry AppWorld).
- Interacción con el hardware del dispositivo, como son la cámara, GPS, teléfono, entre otros.
- Integración con redes sociales, como Facebook o Twitter [75].

2.3.2 Desarrollo Web

Otra opción para el desarrollo de aplicaciones móviles es el desarrollo web. A diferencia de las aplicaciones nativas, estas aplicaciones son ejecutadas dentro de un navegador en el dispositivo móvil, y se desarrollan utilizando tecnologías web, tales como HTML[27], CSS[28], y JavaScript.

HTML es uno de los lenguajes utilizados para el desarrollo de aplicaciones móviles multiplataforma. La revisión más reciente es HTML5 y apunta a proveer herramientas útiles para crear aplicaciones web complejas, y manejar contenido multimedia, así como crear documentos con contenido semántico. A la fecha es aún un *working draft de la W3C* [76], pero en los últimos años ha ido ganando soporte de los navegadores y se está utilizando tanto para desarrollo web como de aplicaciones móviles.

Toma HTML 4.01 como punto de partida, y tiene en cuenta que los documentos HTML presentes en la web son una mezcla de varias especificaciones, como las introducidas por los navegadores y por prácticas comunes, y que contienen muchos errores de sintaxis. Por ello define el procesamiento requerido para los documentos inválidos, de forma tal que los errores de sintaxis sean tratados uniformemente por todos los navegadores que se apeguen al estándar. Introduce varias APIs[29] y nuevas etiquetas que permiten desarrollar aplicaciones web

[27] HTML (Hypertext Markup Language)
[28] CSS (Cascading Style Sheets)
[29] API (Application Programming Interface)

complejas. Por este motivo se está perfilando como un candidato potencial para aplicaciones móviles multiplataforma [77]. Algunas de estas APIs son:

- Dibujo en dos dimensiones
- Reproducción de archivos de audio y video
- Aplicaciones web que funcionen cuando no se tiene conexión
- Registro de aplicaciones para ciertos protocolos
- Edición de documentos
- Funcionalidad *drag and drop*
- Facilitar la navegación hacia atrás, exponiendo el historial
- Mensajería entre documentos

En muchos casos se utiliza el término HTML5 para agrupar nuevas tecnologías de la web, muchas de las cuales no forman parte de la especificación, generalmente relacionadas con una experiencia web superior [78]. Algunos ejemplos de estas tecnologías son CSS3 [79], la *Geolocation API* [80], y *WebSockets* [58].

2.3.2.1 Sencha Touch

Dentro de los *frameworks* más utilizados para el desarrollo de aplicaciones móviles en HTML5 encontramos a Sencha Touch [81], que es una biblioteca JavaScript que permite desarrollar aplicaciones compatibles con los sistemas operativos iOS, Android, BlackBerry y Windows Phone, entre otros. Está desarrollado con los últimos estándares de HTML5 y CSS3, y las aplicaciones pueden ser accedidas como aplicaciones web o portarse para los diferentes sistemas operativos mencionados anteriormente mediante el uso de la herramienta Sencha Cmd [82].

Está construido sobre el sistema de clases del *framework* ExtJS 4, el cual brinda todo el potencial de la carga dinámica, manejo de herencia, carga de dependencias y *mixins* (permite incluir el código de una clase en otra) [83] [84]. Este sistema de clases facilita el desarrollo orientado a objetos sobre JavaScript, utilizando una arquitectura MVCS[30].

En lo que refiere a la interfaz de usuario, Sencha Touch soporta nativamente los eventos táctiles *tap*[31], *swipe*[32] y *pinch*[33]. Cuenta con animaciones fluidas y soporte de *scroll*, el cual es implementado mediante diferentes mecanismos en cada dispositivo, intentando lograr la mejor experiencia de uso en cada plataforma. También provee un motor de renderizado que permite cargar las aplicaciones de forma instantánea, el cual detecta rápidamente la posición del dispositivo, adaptando la interfaz a la misma.

Sencha Touch provee un componente básico llamado *Navigation View*, que es utilizado para mostrar las diferentes vistas desarrolladas, las cuales se apilan en un *stack* [85]. Mediante esta estructura se implementa el historial y la creación de nuevas instancias de las vistas.

---

[30] MVCS (Model-View-Controller-Store)
[31] Evento capturado cuando se hace contacto en un único punto entre el usuario y el dispositivo con interfaz sensible al tacto [171].
[32] Evento capturado cuando se hace un *tap*, para luego arrastrar el objeto con el cual se establece el contacto con la superficie sensible al tacto.
[33] Evento capturado cuando el usuario “pellizca” la superficie sensible al tacto.

2.3.3 Desarrollo Híbrido

Como se vio anteriormente, es posible desarrollar una aplicación nativa por cada plataforma, o crear una aplicación web con una experiencia de usuario más limitada o minimalista que funcione en varias plataformas. Existe una tercera opción, que consiste en combinar ambos enfoques en una aplicación híbrida.

Las aplicaciones móviles híbridas son una combinación de vistas web y componentes nativos, por lo que al igual que las aplicaciones web permiten reutilizar la interfaz en los portes de la aplicación para cada plataforma, pero pudiendo acceder a características del dispositivo que no se encuentran disponibles en el navegador, como lo haría una aplicación nativa.

Si bien esto permite reducir los costos de desarrollo, como las convenciones de usabilidad y diseño son diferentes para cada plataforma es imposible reutilizar toda la interfaz con buenos resultados, ya que el usuario espera que cada aplicación se comporte de forma similar a las aplicaciones nativas de su dispositivo.

Si se utilizan patrones de diseño ajenos a una plataforma la aplicación puede resultar difícil de utilizar o visualmente inconsistente con respecto a la plataforma, por lo cual a la hora de desarrollar aplicaciones híbridas es importante que las vistas reutilizadas no diverjan del paradigma de diseño de interfaz de usuario de las plataformas soportadas.

Las aplicaciones híbridas se instalan en el dispositivo como aplicaciones nativas, por lo cual permiten aprovechar los servicios que brindan las tiendas de aplicaciones, facilitando su descubrimiento y monetización.

La mayoría de las aplicaciones híbridas se desarrollan utilizando *frameworks* específicos que permiten combinar componentes nativos y vistas web con facilidad. Por lo general, estos *frameworks* proveen una API de JavaScript que permite acceder a las características del dispositivo, o proveen un mecanismo de comunicación entre JavaScript y los componentes nativos.

2.3.3.1 PhoneGap

PhoneGap [86] es un *framework* de código abierto que permite construir aplicaciones móviles multiplataforma utilizando tecnologías web como HTML, JavaScript y CSS. Soporta un gran número de sistemas operativos, entre los que se encuentran Android, iOS, y Windows Phone.

Las aplicaciones construidas con PhoneGap son híbridas, ya que se realiza el *renderizado* gráfico a través de un motor web en lugar de utilizar las funcionalidades de interfaz gráfica nativas del sistema operativo, pero se compilan e instalan como aplicaciones nativas.

PhoneGap provee un conjunto de APIs que forman una capa de abstracción sobre las características físicas del dispositivo y su sistema operativo, permitiendo acceder a características específicas del hardware y la plataforma. Esto permite desarrollar aplicaciones nativas para todos los sistemas operativos soportados utilizando el mismo código de base.

Algunas de las funciones soportadas mediante la API de PhoneGap son: geolocalización, agenda de contactos, acceso al sistema de archivos, acelerómetro, cámara, brújula, reproducción y grabación multimedia, notificaciones, y almacenamiento.

2.4 Software Geográfico

El sistema planteado requiere consultar y analizar información geográfica a partir de la ubicación de los usuarios y los ómnibus, por lo que es necesario evaluar software que provea funcionalidades como servicios de mapas y análisis de ruta.

Es necesario contar con un componente de mapas que permita mostrar ubicaciones, recorridos, y otros elementos como paradas, de forma dinámica y visualmente agradable. También se requiere un servicio de análisis de rutas que permita obtener el mejor camino para ir a un destino y que realice estimaciones de tiempo.

A continuación se presentan los productos y servicios geográficos evaluados, haciendo énfasis en los requerimientos específicos del proyecto.

2.4.1 Sistemas de Información Geográfica

Un Sistema de Información Geográfica es un tipo especial de sistemas de información utilizado para administrar, analizar y desplegar información geográfica [87]. La información geográfica es representada mediante estructuras simples que modelan la geografía y el SIG[34] provee las herramientas necesarias para manipular dicha información.

Puede verse a un SIG como la combinación de cinco componentes: datos espaciales descriptivos, métodos analíticos, personas especializadas, hardware, y software que den soporte a los demás componentes [88].

En el contexto del proyecto será interesante aplicar los métodos analíticos provistos por los SIG, utilizando los datos espaciales recolectados por los sensores del dispositivo móvil, los cuales serán procesados por el *Servidor Central* para ser utilizados en las diferentes funcionalidades provistas por la plataforma.

2.4.2 ESRI ArcGIS

ArcGIS es una integración de productos y componentes de software SIG desarrollada por ESRI [89] que permite representar datos geográficos, y brinda las herramientas necesarias para diseñar, crear y trabajar con ellos. La información geográfica es representada por conjuntos de datos geográficos que el SIG permitirá manipular, utilizando herramientas desarrolladas para este fin.

2.4.2.1 Características generales

La suite de ArcGIS provee soluciones para la Web, dispositivos móviles y computadoras de escritorio, así como servicios online en la nube o para empresas [90]. Hay varias prestaciones que esta provee, de las cuales enumeramos las siguientes:

- Consulta y edición de información geográfica.
- Creación de aplicaciones web de cartografía.
- Creación de base de datos geográfica.

[34] SIG (Sistemas de Información Geográfica)

- Publicación de datos y mapas en la nube, lo cual permite que los usuarios finales accedan a los mismos desde una gran variedad de dispositivos, obteniendo información actualizada en cualquier momento.
- Acceso a aplicaciones y mapas creados por terceros, las cuales podrán ser incorporados a los mapas propios.
- Entorno de trabajo colaborativo, con el cual se podrá trabajar conjuntamente con otros usuarios.
- Creación de aplicaciones personalizadas utilizando la API y el SDK de ArcGIS.

2.4.2.2 ArcGIS Server

Luego de crear los mapas y herramientas del SIG, es posible publicarlos mediante ArcGIS Server, lo que permite que el usuario pueda utilizarlas en cualquier lugar mediante servicios Web. Esta herramienta permite centralizar la información en un único lugar y brindar acceso multitudinario a la misma [91].

ArcGIS Server provee varios servicios [92] que pueden ser utilizados a la hora de desarrollar aplicaciones. Algunos de estos servicios son:

- Representación cartográfica. Es un servicio dinámico de mapas que será publicado en la Web.
- Geocodificación. Permite buscar una dirección y obtener sus coordenadas.
- Geodatos. Administración de base de datos espaciales mediante consultas, edición y acceso a los datos.
- Geoprocesamiento. Análisis espacial y procesamiento de datos.
- Análisis de red. Resuelve enrutamiento, instalación más cercana y área de servicio.
- Geometría. Resuelve cálculos geométricos espaciales.

2.4.3 ArcGIS para Dispositivos Móviles

Dado el creciente uso de dispositivos móviles y los beneficios que estos pueden brindar a una organización que trabaja con SIG, poder contar con una aplicación que soporte dichas funcionalidades en un dispositivo móvil conlleva grandes beneficios [2].

ArcGIS cuenta con un SDK[35] para plataformas móviles que facilita la creación de aplicaciones móviles que permitan al usuario navegar en los mapas, recolectar, editar y subir información geográfica y ejecutar análisis de SIG, accediendo directamente al SIG de la empresa mediante ArcGIS Online o ArcGIS Server [93]. El SDK es compatible con los sistemas operativos Android, iOS y Windows Phone [93]. También cabe la posibilidad de desarrollar aplicaciones web para dispositivos móviles mediante el uso de una API desarrollada en JavaScript [94].

2.4.3.1 ArcGIS para Android

El SDK de ArcGIS para Android permite agregar el potencial de las funcionalidades de ArcGIS en aplicaciones nativas del sistema operativo Android [95]. Este SDK permite acceder a ArcGIS Online y ArcGIS Server, desde donde podrán agregarse mapas y consultar servicios. Entre las funcionalidades a destacar se encuentran:

[35] SDK (Software Development Kit)

- Uso de mapas y capas, permitiendo la carga de mapas almacenados localmente, los cuales podrán ser desplegados utilizando diferentes proyecciones.
- Obtención y edición de información en el mismo dispositivo, permitiendo la interacción con los sensores (GPS, acelerómetro y brújula, entre otros).
- Ejecución de tareas de geoprocesamiento en el dispositivo, liberando la carga de los servidores y permitiendo explotar la capacidad de procesamiento local.
- Uso de funcionalidades nativas (*pop-ups* y *push notifications*, entre otras) y de ArcGIS (análisis de redes, manejo de capas y servicios de geocodificación, entre otros) para mejorar la experiencia del usuario.
- Integración con herramientas de desarrollo (Eclipse), facilitando el uso de mapas en aplicaciones nativas.

El SDK es compatible con versiones de Android superiores a 2.3.3 y de ArcGIS Server superiores a 9.3.1. Este utiliza la tecnología OpenGL ES 2.0 [96] para renderizar los mapas, la cual permite aumentar la velocidad de renderizado utilizando menos recursos de memoria y autonomía [97].

2.4.3.2 ArcGIS para Javascript

ArcGIS provee una API para JavaScript, con la cual permite desarrollar aplicaciones web que podrán utilizar las funcionalidades de ArcGIS Online y ArcGIS Server [98]. Esta API es compatible con los navegadores Chrome, Firefox, Safari e Internet Explorer, además de ser compatible con otras bibliotecas JavaScript como ExtJS y jQuery. La misma está disponible en los servidores de ArcGIS para uso libre, aunque también puede descargarse el código fuente y utilizarla desde un servidor propio [99]. Brinda soporte para HTML5 y CSS3 y provee una API específica para desarrollo de aplicaciones móviles.

La API específica para desarrollo de aplicaciones móviles es una versión compactada, la cual está optimizada para su uso en conexiones lentas o donde es necesario bajar la latencia de la red [100]. En esta versión se incluyen 32 de los 80 módulos utilizados en la versión estándar, permitiendo incluir los demás módulos en caso de ser necesario.

Entre las características y funcionalidades destacables se encuentran:

- Uso de mapas y capas, con detección de navegación para aplicaciones web y web móviles.
- Consumo de datos a través de servicios web, los cuales pueden ser estáticos o dinámicos, y pueden estar en diferentes formatos (CSV, KML, GeoRSS, XML, JSON, etc.).
- Consumo de servicios de ArcGIS Server, como son: ruteo, geoprocesamiento y operaciones geométricas, entre otros.
- Manejo de eventos de forma asíncrona, lo cual colabora con una mejor experiencia de usuario, ya que no experimenta bloqueos de interfaz entre tanto se procesa una consulta al servidor.
- Edición de datos geográficos, lo que permite que el usuario edite sus atributos (forma, tamaño, metadatos) y pueda impactar esos cambios en una base de datos.
- Desarrollo de nuevos *widgets* utilizando Dijit.
- Integración con otros *frameworks* de desarrollo escritos en JavaScript.

2.4.4 Bing Maps

Bing Maps es un servicio online de mapas desarrollado por Microsoft, el cual provee múltiples APIs para la interacción con sitios web y aplicaciones de diferentes plataformas [101]. Estas API permiten utilizar la cartografía de Bing, proveyendo mapas estáticos e interactivos.

Para enviar solicitudes a las diferentes API de Bing es necesario utilizar una clave proporcionada por Microsoft. Esta se utiliza para realizar reportes de uso, para los que se guardan determinados accesos a Bing Maps REST Services y Bing Spatial Data Services [102].

ArcGIS soporta integración con los mapas base de Bing en sus SDKs para Android y Javascript.

2.5 Bases de Datos

Una base de datos es un conjunto de datos relacionados entre sí. Si bien este concepto es muy amplio, por lo general se suele tratar dentro del ámbito de Sistemas de Información. Un Sistema de Información es un conjunto de componentes de hardware y software que interactúan entre sí con el objetivo de almacenar, recuperar y procesar datos e información [103].

El tamaño, las funcionalidades y el desempeño de las bases de datos actuales han crecido enormemente debido a los avances tecnológicos, específicamente en la capacidad de procesamiento. Este avance no sería posible de no ser por las nuevas generaciones de procesadores, memorias y arquitecturas; la disponibilidad de hardware que soporta mayores volúmenes de almacenamiento; y la gran capacidad de transferencia de datos que brindan las redes de computadoras actuales.

2.5.1 Bases de Datos Espaciales

Existen diversas áreas de aplicación en lo que refiere a las bases de datos. En algunas de ellas es necesario almacenar información geométrica, geográfica y/o espacial, lo que tiene como objetivo ubicar un elemento en un entorno, utilizando un sistema definido.

No hay una única definición para el concepto de base de datos espacial, pero es razonable utilizar la definición que da Ralf Hartmut Güting en el artículo "*An Introduction to Spatial Database Systems*" [104]. Define un sistema de base de datos espacial como aquel que provee tipos de datos espaciales a nivel de modelo y lenguaje de consulta, en donde estos tipos de datos son soportados a nivel de implementación, proveyendo índices espaciales y algoritmos eficientes de *join*[36] de estos datos.

Lo que se quiere representar son los objetos del espacio y el espacio mismo. Cuando hablamos de los objetos del espacio nos referimos, por ejemplo, a ciudades, ríos, bosques; y cuando pensamos en el espacio mismo queremos enfocarnos en características que describen a los objetos del espacio, por ejemplo, mapas temáticos. Estas dos consideraciones pueden unificarse si se provee un mecanismo para modelar objetos simples y colecciones de objetos espaciales relacionados.

[36] Join: Operación que permite combinar registros de dos o más tablas de una base de datos.

Entre los tipos de datos espaciales básicos que debe soportar una base de datos de este tipo están: punto, línea y polígono. Estos proveen el nivel de abstracción necesario para modelar cualquier estructura geométrica del espacio, la relación entre ellas, sus propiedades y las operaciones que se pueden definirse.

Un punto representa un objeto del espacio que sólo su posición puntual es relevante, descartando su forma o extensión. Si pensamos en el foco del proyecto, un ómnibus puede representarse como un punto, ya que no es relevante su forma o de qué tamaño es específicamente, sino que nos interesa representarlo como un todo. Una línea, que en la mayoría de los casos es utilizada dentro de un conjunto de líneas conectadas entre sí, formando una polilínea, es utilizada para representar objetos con los que uno se mueve a través del espacio. En el ámbito del proyecto se utilizan para modelar las rutas de los ómnibus o el trayecto que describe una persona al caminar. Un polígono es utilizado para representar extensiones de dos dimensiones, y puede contener agujeros y ser formado por partes disjuntas.

Cuando se habla de las posibles relaciones entre estos tipos de datos básicos, se pueden considerar diferentes tipos:

- Relaciones topológicas. Las relaciones de adyacencia, pertenencia (un objeto está dentro de otro) y disyunción son invariantes a las operaciones de traslación, rotación y escala.
- Relaciones direccionales. Algunos ejemplos son “arriba de”, “debajo de”, “al sur de”, entre otras.
- Relaciones métricas. Por ejemplo, que la distancia entre un punto y otro sea mayor a cien metros.

También es necesario definir propiedades sobre estos objetos, las cuales son representadas de forma tradicional en varios manejadores de bases de datos, almacenando las mismas en tablas relacionadas que hacen referencia a los objetos del espacio; y operaciones sobre los tipos de datos, para lo cual debe definirse un álgebra que las defina.

2.5.2 Bases de Datos Relacionales

El concepto de bases de datos relacionales fue introducido por E. F. Codd en 1970, en un artículo llamado “*A Relational Model of Data for Large Shared Data Banks*” [105]. En esta publicación se marca un claro contraste en lo que hasta ese momento eran los modelos de datos utilizados (modelo de red y jerárquico) y el modelo relacional. Específicamente trata el tema de la independencia de los datos, enfocándose en la necesidad de independencia entre la implementación, representación, y el uso de los mismos. Menciona que la representación tabular fue uno de los mayores avances en relación a la independencia de los datos, ya que posibilitan el cambio de la representación interna de los datos sin afectar el uso.

Luego introduce la visión relacional de los datos que deriva de la teoría matemática de conjuntos. También presenta el concepto de tuplas pertenecientes a una relación y muestra sus principales características. A lo largo de la publicación sigue presentando varios conceptos relacionados a esta representación: formas normales, operaciones, redundancia y consistencia.

En la publicación mencionada anteriormente se introdujo el modelo relacional, se presentó un lenguaje de acceso a datos basado en el cálculo de predicados. De esta idea, y luego de otras definiciones previas, surgió lo que hoy conocemos como el lenguaje SQL[37].

SQL es un lenguaje que permite realizar consultas a bases de datos a través de un manejador de bases de datos relacional (RDBMS). Puede dividirse en dos sublenguajes: DDL[38] y DML[39]. El lenguaje de definición (DDL) es utilizado para crear y eliminar bases de datos y sus objetos, mientras que el lenguaje de manipulación (DML) es utilizado para insertar, actualizar y obtener datos de las bases de datos [106].

2.5.2.1 SQLite

SQLite es una librería que implementa un motor de base de datos transaccional [107]. A diferencia de otros motores de bases de datos, SQLite no implementa el servidor como un servicio separado y tampoco utiliza mecanismos de comunicación específicos para el acceso a los datos, sino que los procesos acceden directamente a los archivos en disco para su lectura y escritura [108].

Esta librería mantiene múltiples tablas, índices, triggers y vistas en un único archivo. Este archivo tiene un formato que es independiente de la plataforma y de la arquitectura en la que se aloja.

Tal como se puede deducir de su nombre, es una librería compacta, que de omitirse algunas funcionalidades puede llegar a reducirse a 300 KiB. Está hecha para correr con un mínimo de espacio de *stack* (4 KiB) y con muy poco espacio de *heap* (100 KiB), lo que hace que sea la librería más utilizada en dispositivos con memoria limitada, como es el caso de los teléfonos inteligentes [107].

2.5.3 Bases de Datos No Relacionales

Las bases de datos relacionales son una pieza fundamental de los grandes sistemas de información empresariales, e históricamente se han utilizado de forma casi exclusiva en la mayoría de los sistemas de información. Sin embargo, el uso masivo de internet y la proliferación de dispositivos y aplicaciones que en forma conjunta generan grandes volúmenes de datos, potenciaron la necesidad de contar con otros paradigmas de almacenamiento y manipulación de datos que se adecúen a los requerimientos específicos de los sistemas actuales [109].

El concepto de bases de datos no relacionales, popularizadas como bases de datos “NoSQL”, se ha expandido rápidamente en los últimos tiempos [110]. Algunos autores atribuyen la etimología del nombre a que este tipo de bases de datos no soportan el lenguaje de consulta SQL, y otros a *not only* SQL [111]. Según esta definición, son muchas las bases de datos que entran en esta categoría, entre los que encontramos a las bases de datos basadas en documentos, grafos, clave-valor y columnas.

[37] SQL (Structured Query Language)
[38] DDL (Data Definition Language)
[39] DML (Data Manipulation Language)

Estos paradigmas de bases de datos intentan solucionar algunos de los problemas que tiene el modelo relacional, como por ejemplo, la escalabilidad. Los sistemas actuales cada vez manejan mayor cantidad de información y esta debe ser manejada de manera eficiente y rápida. Si bien el desempeño de las bases de datos relacionales suele mejorar con una mayor cantidad de recursos, no se adecúan al modelo *cloud* [112].

En el ámbito del proyecto tendremos especial interés en investigar las bases de datos basadas en documentos y aquellas basadas en clave-valor.

2.5.3.1 Bases de Datos basadas en Documentos

Las bases de datos basadas en documentos [109] se diferencian en que los datos no son almacenados en tablas, como se hace en el modelo relacional, sino que se almacenan en documentos.

Es una representación libre de esquemas, ya que no se obliga a definir una estructura particular para el contenido de un documento. Si un nuevo campo debe agregarse a ese documento, simplemente se agrega y esto no afecta a los documentos almacenados actualmente. Otra ventaja con respecto al modelo relacional, es que los documentos no tienen que almacenar campos vacíos si no se cuenta con el valor del mismo.

Los documentos pueden ser complejos, permitiendo la lectura y escritura concurrente de objetos enteros del modelo, son independientes entre sí y son descriptos en formato abierto, utilizando JSON, XML o similares.

MongoDB

MongoDB es un sistema de base de datos *open-source* de propósito general basado en documentos [113]. Los documentos son almacenados en una codificación binaria de JSON[40] llamada BSON[41]. BSON extiende el modelo de JSON y provee tipos de datos adicionales, así como una codificación y decodificación eficiente en diferentes lenguajes [114].

Provee un completo soporte de índices y búsquedas avanzadas, búsqueda de texto, seguridad avanzada y varios mecanismos que lo hacen escalable y altamente disponible. Esto se logra mediante *auto-sharding* [42] y replicación en varios servidores. También brinda algunas funcionalidades para el manejo de grandes volúmenes de datos, como es el *Aggregation Framework* y *MapReduce* [115].

2.5.3.2 Bases de Datos Clave-Valor

Las bases de datos basadas en clave-valor también tienen una representación libre de esquemas y usualmente consisten en una cadena de caracteres que representa la clave y el dato que representa el valor de la relación clave-valor [116]. El dato es en general un tipo de dato primitivo o un objeto de un determinado lenguaje de programación.

Este modelo permite que se almacenen datos arbitrarios utilizando una única clave, lo que hace que en determinados contextos se experimente un desempeño mucho mayor que con el

[40] JSON (JavaScript Object Notation)
[41] BSON (Binary JSON)
[42] Sharding: Particionamiento horizontal de la base de datos.

modelo tradicional. Otra ventaja significativa es que el código resultante suele ser mucho más breve y claro que en el caso de sentencias SQL embebidas.

Redis

Redis es un sistema de base de datos *open-source* escrito en C basado en clave-valor [117]. Lo interesante de Redis radica en que las claves no están limitadas a cadenas de caracteres, sino que pueden ser hashes, listas, conjuntos o conjuntos ordenados. Además, permite realizar operaciones básicas sobre estas claves, como pueden ser: incrementar valor del hash, concatenar una cadena de caracteres, agregar datos a una lista, intersectar dos conjuntos, entre otras.

Con el fin de obtener el desempeño requerido, Redis maneja los datos en memoria, aunque estos se persisten periódicamente de forma asíncrona en disco. Puede ser configurado para persistir los datos luego de un lapso de tiempo o una cantidad determinada de cambios.

Una de las características más significativas es el soporte de múltiples lenguajes de programación. Actualmente hay clientes de Redis para JavaScript, Ruby, Python, PHP, Erlang, Tcl, Perl, Lua y Java, entre otros [118].

2.6 Aplicaciones Relacionadas

Durante el curso del proyecto se estudiaron una multitud de sistemas y aplicaciones que tuvieran algún paralelismo con la propuesta del sistema a construir. A continuación presentamos dos sistemas que resultaban de particular interés, el primero por manejar información de ómnibus basada en la ubicación geográfica del usuario, y el segundo por brindar la posibilidad de compartir una coordenada geográfica con determinado grupo de usuarios.

2.6.1 iBus

La empresa uruguaya C.U.T.C.S.A. (Compañía Uruguaya de Transporte Colectivo S.A.) y la multinacional Movistar se unieron para proveer un servicio a clientes de Movistar, usuarios de C.U.T.C.S.A. llamado iBus [119].

El servicio es utilizado a través del envío y recepción de mensajes de texto. El envío debe realizarse con determinado formato y el servicio tiene disponible dos funcionalidades: recibir el número de la parada más cercana al punto en donde se encuentra el usuario y/o recibir el tiempo estimado en el que arribarán los próximos dos ómnibus de la línea consultada [119].

MOVISTAR utiliza la tecnología LBS[43] para la detección de la posición del usuario. Esta tecnología está basada en múltiples técnicas de geolocalización para móviles (como pueden ser *Time Difference of Arrival* o *Enhanced Observed Time Difference*, entre otras) [120], con la que obtiene una ubicación aproximada del celular del usuario que luego se utilizará para procesar la solicitud. En la especificación del servicio se indica que la posición del usuario y el tiempo de arribo que se obtienen son aproximados [121].

Se efectuó una prueba ubicando al usuario en la puerta de Facultad de Ingeniería (UdelaR) y se realizó la consulta para la línea de ómnibus 199. El texto del mensaje enviado desde el móvil fue el siguiente (el número de parada se había obtenido previamente): "*BUS 199 2114*".

El mensaje respuesta fue el siguiente: "Línea 199 CRIO DEL NORTE en camino, arriba a la parada 2114 sobre BENITO NARDONE esquina PATRIA en 4 minutos. Siguiente unidad en terminal, arriba en 12 minutos."

2.6.2 Google Latitude

Google Latitude era un servicio que proveía Google Maps para Android y que también contaba con una aplicación para iOS [122].

Básicamente brindaba la posibilidad de compartir la ubicación de un usuario y que este pudiera ver la ubicación de sus amigos desplegada en un mapa, pudiendo contactarse con ellos mediante distintos servicios de mensajería o mediante su mensaje de estado [122].

Dentro de las funcionalidades que proveía, se encontraban [123]:

- Compartir la ubicación propia con usuarios de Google Latitude (móviles o iGoogle).
- Ver la ubicación de sus amigos desplegada en un mapa.

[43] LBS (Location Based Service)

- Controlar la ubicación, controlando quién y qué ve cada usuario (diferentes niveles de detalle).
- Configuración de privacidad, permitiendo la edición manual, pudiendo presentar diferentes ubicaciones a cada usuario amigo (incluyendo la opción de no compartir la ubicación).
- Detección de visita a diferentes sitios. El dispositivo detecta automáticamente que se llegó a determinado lugar y realiza la petición de compartir dicha información.

Google Latitude podía ser utilizado desde cualquier navegador de internet, así como en las aplicaciones específicas para cada móvil. Respecto a la compatibilidad con los sistemas operativos de los diferentes dispositivos móviles, era compatible con Android, BlackBerry, iPhone, Symbian y Windows Mobile [123]. El servicio fue dado de baja el 9 de agosto de 2013.

3 Análisis y Especificación de Requerimientos

El objetivo de este capítulo es la de presentar las principales funcionalidades y características que deberá brindar el producto a desarrollar, que surgen como resultado de la recopilación de requerimientos planteados por los integrantes del proyecto en colaboración con los tutores.

Este capítulo se encuentra dividido en tres secciones. Inicialmente se presenta brevemente el proyecto y su alcance, una segunda sección en la que se presentan los requerimientos funcionales del sistema, y una última sección donde se indican los requerimientos no funcionales que se encontraron durante la etapa de análisis.

3.1 Objetivos y Alcance del Sistema

El principal objetivo del sistema a desarrollar es brindar una plataforma de base que permita que un dispositivo comparta su posición geográfica con otros dispositivos en tiempo real, así como cualquier tipo de datos. Esta plataforma servirá de base para diversas aplicaciones de naturaleza geográfica, dentro de las cuales consideraremos dos grupos funcionales.

Por un lado, un grupo funcional nuclea una red social de usuarios que permite que los usuarios compartan y consulten la posición geográfica de aquellos usuarios del sistema a los que están vinculados. Esta aplicación brindará la posibilidad de que los usuarios establezcan un punto de encuentro, y que se pueda realizar seguimiento mutuo entre usuarios en tiempo real.

En segundo lugar, la misma aplicación permitirá consultar información de ómnibus en tiempo real, con la finalidad de que el usuario pueda obtener tanto la ubicación de un ómnibus como una estimación certera del momento en que va a llegar a una parada determinada.

3.2 Descripción General

En este capítulo se presenta la descripción general de los requerimientos del proyecto, introduciendo a grandes rasgos lo que se espera de la aplicación, cuáles son las características del público objetivo y qué se puede asumir en relación al hardware y software en el que se ejecutará la misma.

3.2.1 Funcionalidades

El sistema a construir apunta al intercambio de información geográfica en tiempo real, ya sea para consultar la ubicación exacta de un ómnibus de una determinada línea, o la ubicación de personas en la red de usuarios de la misma. De este sistema se desprenderán dos grupos funcionales que compartirán el ambiente de ejecución, en el que se tendrá en cuenta la información geográfica del usuario que utiliza la aplicación, lo que permitirá brindarle información de utilidad, tanto al momento de tomarse un ómnibus, como para ubicar amigos o encontrarse con alguno de ellos, habiendo establecido previamente un punto de encuentro.

3.2.1.1 Información de Línea de Ómnibus en Tiempo Real

Una de las funcionalidades que este sistema presentará, será la de poder consultar información de líneas de ómnibus en tiempo real. Básicamente, un usuario deberá poder consultar el tiempo estimado de arribo de cualquier ómnibus de cualquier línea que esté próximo a la ubicación donde este se encuentra. También podrá hacer un seguimiento en

tiempo real del ómnibus que desee para obtener información exacta acerca de la ubicación del mismo y podrá obtener información de cómo llegar a la parada más cercana para arribar a su destino en tiempo y forma.

3.2.1.2 Establecer Punto de Encuentro con Usuarios de la Red

Otra funcionalidad que el sistema deberá brindar, es la de establecer puntos de encuentro con usuarios de la red. Esta funcionalidad permitirá a un usuario definir un punto de encuentro con otro, agregando información multimedia que permita ubicar fácilmente el lugar (por ejemplo: fotos, audio o video). Además, el sistema permitirá establecer alarmas o recordatorios, agendando el encuentro en el dispositivo del usuario y también permitirá hacer un seguimiento en tiempo real de un usuario.

3.2.2 Características de los Usuarios

Los usuarios a los que apunta el sistema pueden ser de cualquier tipo. Deben tener nociones muy básicas de utilización de mapas y aplicaciones geográficas.

3.2.3 Restricciones

La aplicación a desarrollar debe estar disponible para las plataformas Android e iOS, investigando si es posible utilizar GeneXus X Evolution 2 para realizar el desarrollo. Adicionalmente deberá evaluarse la posibilidad de desarrollar una aplicación HTML5 que pueda ser accedida desde la mayoría de las plataformas móviles.

En cualquiera de los casos anteriormente citados, la aplicación apunta a dispositivos móviles que cuenten con módulo GPS/GLONASS, lo que brindará una mejor interacción con el sistema, pudiendo acceder a información geográfica del usuario con una precisión aceptable.

3.2.4 Supuestos y Dependencias

La herramienta GeneXus X Evolution 2 tiene la capacidad de integrarse con el SDK de ArcGIS para dispositivos móviles, permitiendo el desarrollo de una aplicación multiplataforma sin necesidad de escribir código nativo.

Este producto depende del buen funcionamiento de la red de datos 3G/4G, que se utilizará para el intercambio de información. Tanto así es la dependencia que la mayor parte de la experiencia del usuario con la aplicación tendrá que ver con este aspecto.

Si bien se considerarán restricciones y problemas de seguridad, este no será el foco principal al momento de desarrollar la aplicación.

Se cuenta con datos geográficos de la ciudad como calles, paradas y recorridos de ómnibus, así como de la información asociada a las líneas de ómnibus, como son los números de línea, variantes del recorrido de una misma línea, y horarios de salida.

3.3 Requerimientos Funcionales

En esta sección se presentan los requerimientos funcionales que deberá cumplir el sistema a desarrollar. Los requerimientos se agrupan en categorías para permitir una mejor comprensión de las funcionalidades del sistema y el comportamiento del mismo.

3.3.1 Requerimientos Funcionales de FindMe!

El primer grupo funcional de la aplicación permitirá que un usuario conforme una red de usuarios o amigos, conformando una red social en la que podrá compartir información geográfica y seguir a sus amigos en tiempo real. La aplicación contará con la administración básica de usuarios y perfiles como la que brinda cualquier red social.

También permitirá que algunas de sus funcionalidades interactúen con el dispositivo, como puede ser, el envío de mensajes de texto, cámara, agenda y libreta de contactos, entre otros.

Dos usuarios que sean amigos podrán establecer un punto de encuentro, permitiendo agendar la cita en la agenda del teléfono, asociando información multimedia relacionada con el punto de encuentro que ayude a identificar el mismo o simplemente brindar información de contexto que sea de utilidad para los usuarios.

3.3.1.1 Administración de Usuarios

Uno de los objetivos principales del sistema es que cada usuario pueda compartir su posición geográfica con otros usuarios del sistema, pudiendo establecer un vínculo de amistad, conformando así una red social. Por este motivo será fundamental contar con el manejo de cuentas de usuario, con la que el usuario podrá acceder a todos los servicios del sistema.

Los requerimientos asociados a la administración de usuarios están presentes en la mayoría de las redes sociales, por lo que no se incluyeron en este documento. Los detalles pueden encontrarse en el *Anexo II - Análisis y Descripción de Requerimientos*.

3.3.1.2 Red de Usuarios

Como se mencionó anteriormente, los usuarios del sistema conforman una red social. Cada usuario tiene un conjunto de usuarios asociados con los cuales comparte su información, en particular su posición geográfica. A este conjunto de usuarios se les denomina la *red* de un usuario, o simplemente su conjunto de amigos.

Será necesario que el sistema provea un listado de usuarios, de forma que cada usuario pueda encontrar a otros usuarios y agregarlos a su red. El usuario podrá enviar y recibir solicitudes de amistad otros usuarios, permitiéndole aceptar o rechazar las mismas. También se proveerá un mecanismo para eliminar usuarios previamente agregados como amigos. Los casos de uso mencionados anteriormente son: Agregar Amigos, Enviar Solicitud de Amistad, Ver Solicitudes de Amistad, Aceptar Solicitud de Amistad, Rechazar Solicitud de Amistad y Eliminar Amigo.

Los detalles asociados a estos casos de uso se especifican en el *Anexo II - Análisis y Descripción de Requerimientos.*

En este documento se presenta únicamente los casos de uso *Ubicar Amigos* y *Establecer Punto de Encuentro*, ya que son los más interesantes en relación a los objetivos del sistema a desarrollar.

Ubicar Amigos

Este caso de uso permite que un usuario reciba la última posición geográfica conocida de los usuarios de su red, pudiendo filtrar por datos del usuario, o área geográfica.

El usuario indica los criterios de búsqueda para los usuarios de su red y el sistema devolverá aquellos usuarios que concuerden con los criterios de búsqueda, y para cada uno de estos, la última posición geográfica registrada en el sistema.

Establecer Punto de Encuentro

Este caso de uso permite que un usuario envíe una invitación a uno de sus amigos para encontrarse en determinada posición geográfica, fecha y hora. También podrá enviar datos adicionales, como pueden ser imágenes o videos.

3.3.1.3 Configuración

El sistema cuenta con algunas opciones configurables que están relacionadas con la experiencia de usuario y el consumo de batería de la aplicación. Se permite configurar el intervalo de tiempo entre el que se actualiza la posición geográfica y la forma de obtener la coordenada geográfica, en la que se podrá elegir entre utilizar el GPS, la red de telefonía celular (triangulación por antenas) o ambas.

3.3.1.4 Transmisión de Datos Geográficos en Tiempo Real

La plataforma a construir deberá brindar la posibilidad de transmitir datos geográficos de un usuario a otro en tiempo real.

Una vez establecida la conexión entre usuarios, el dispositivo del usuario a seguir obtendrá su posición geográfica mediante GPS u otro tipo de geolocalización y la enviará al otro usuario.

Seguir Amigo en Tiempo Real

Este caso de uso permite que un usuario siga a uno de sus amigos en tiempo real. El usuario a seguir es notificado de que uno de sus amigos quiere comenzar a seguirlo y solicita su confirmación para establecer el seguimiento. Una vez que el usuario a seguir confirma, el usuario seguidor comienza a recibir la posición geográfica del mismo, la cual va siendo actualizada según el intervalo de tiempo configurado.

3.3.2 Requerimientos Funcionales de ¿Cuándo llega?

El segundo grupo funcional de la aplicación permitirá que un usuario pueda consultar información de líneas de ómnibus en tiempo real. Básicamente, un usuario podrá consultar el tiempo estimado de arribo de un ómnibus de una línea determinada, el cual se calculará a partir de las posiciones geográficas actuales de los ómnibus, obteniendo así una estimación mucho más acertada que una basada en estadísticas de recorridos pasados.

Para realizar una consulta el usuario solo deberá indicar la parada en la cual desea tomar el ómnibus, y seleccionar una de las líneas que pasa por allí. Interesa que la aplicación sea sencilla de utilizar, por lo que deberá ser posible utilizar la posición geográfica del usuario para mostrarle aquellas paradas de ómnibus que se encuentran próximas a él.

También interesa que se pueda acceder rápidamente a la información de ómnibus, por lo que se ofrecerá la posibilidad de recordar paradas como favoritas, y de ver las últimas consultas de líneas de ómnibus realizadas.

Para realizar las estimaciones de llegada de los ómnibus será necesario que cada ómnibus transmita su posición geográfica en tiempo real al sistema. Se asume que cada ómnibus tendrá un dispositivo con sensor geográfico y acceso a Internet, y la capacidad de correr una aplicación que se comunica con el sistema.

Se considerará que un ómnibus está *activo* desde el momento en que comienza a realizar su recorrido hasta el momento en que llega a destino, y solo transmitirá su posición durante este intervalo de tiempo. Esto es necesario para que los ómnibus que están fuera de servicio no interfieran con las estimaciones.

3.3.2.1 Requerimientos Funcionales de la Aplicación Móvil

En esta sección se presentan los requerimientos de la aplicación móvil que utilizará el usuario para consultar información de ómnibus, y las funcionalidades que deberá proveer el sistema para soportar los casos de uso.

¿Cuándo llega?

Este caso de uso permite que un usuario obtenga una estimación del momento en el que un ómnibus de una línea particular llega a una parada determinada. El usuario selecciona una parada en el mapa y una línea de ómnibus, y el sistema establece una conexión entre el usuario y los dos próximos ómnibus de la línea que van a llegar a la parada.

Dejar de Seguir Ómnibus

Este caso de uso permite que un usuario cierre una conexión establecida con un ómnibus devuelto en el caso de uso ¿Cuándo llega?

Determinar Ruta hacia Parada

Este caso de uso permite que un usuario obtenga la ruta óptima a pie hacia la parada en la que desea tomar el ómnibus. El usuario indica la parada en la que tomará el ómnibus y el sistema calcula la ruta óptima desde el lugar en donde se encuentra el usuario hasta la parada seleccionada, mostrando el resultado en pantalla.

Agregar Parada a Favoritos

Este caso de uso permite que un usuario agregue una parada a sus favoritos, para acceder a la información de las líneas de ómnibus que paran en la misma de forma más rápida y conveniente.

Eliminar Parada de Favoritos

Este caso de uso permite que un usuario elimine una parada de sus favoritos.

Ver Favoritos

Este caso de uso permite que un usuario administre sus paradas de ómnibus favoritas, pudiendo eliminarlas, y pasar a consultar información sobre las líneas de ómnibus correspondientes a las mismas. Los favoritos se almacenan de forma local al dispositivo del usuario.

Ver Consultas Recientes

Este caso de uso permite que un usuario pueda ver las últimas consultas realizadas en el caso de uso ¿Cuándo llega?, las cuales consisten en una parada y una línea de ómnibus particulares.

3.3.2.2 Requerimientos Funcionales de la Aplicación de Ómnibus

En esta sección se presentan los requerimientos de la aplicación que estará presente en todos los ómnibus a ser rastreados por el sistema. Antes de comenzar el recorrido el guarda o chofer del ómnibus deberá indicar la línea, sub-línea y variante a la que pertenece el ómnibus.

El chofer o guarda le indicará al sistema cuando comienza el recorrido mediante un simple botón en la aplicación, activando así la transmisión de datos, y cuando llegue a destino indicará que desea detener la transmisión de forma similar.

El sistema contará con una lista de dispositivos habilitados para utilizar la aplicación de ómnibus como una medida de seguridad para reducir las chances de que algún impostor pueda actualizar información de ómnibus de forma incorrecta.

Configuración de Línea

Este caso de uso permite que el chofer o guarda de un ómnibus seleccione la línea de ómnibus de la cual va a realizar el recorrido. Esto permitirá que al comenzar el recorrido la información se actualice correctamente en el sistema.

Comenzar Recorrido

Este caso de uso permite que el chofer o guarda de un ómnibus indique que ha comenzado a realizar el recorrido, iniciando la transmisión de actualizaciones de la posición geográfica del vehículo.

Finalizar Recorrido

Este caso de uso permite que el chofer o guarda de un ómnibus indique que ha finalizado el recorrido de ómnibus a realizar.

3.4 Requerimientos No Funcionales

En esta sección se presentan los requerimientos no funcionales que deberá cumplir el sistema a desarrollar. Estos surgen a partir del tipo de aplicación y entorno en donde esta es utilizada, haciendo un fuerte hincapié en la usabilidad, movilidad y uso eficiente de los recursos disponibles.

3.4.1 Uso de Simbología Adecuada

En el contexto de una aplicación geográfica que nuclea varias funcionalidades, es de esperar que se utilice simbología que ayude al usuario a identificar qué tipos de objetos móviles está siguiendo, así como sus diferentes estados (a través de diferentes formas o colores, por ejemplo).

3.4.2 Baja Latencia en la Transmisión de Datos

La latencia se define como la cantidad de tiempo de espera que se experimenta en un sistema [124]. Es de esperarse que el sistema a construir tenga una baja latencia en la transmisión de datos, ya que de otra forma no se alcanzaría la experiencia del usuario deseada en el seguimiento en tiempo real.

3.4.3 Interfaz Intuitiva y Amigable

En el mercado actual coexisten varias aplicaciones de un mismo tipo, o que tienen funcionalidades similares, por lo que se necesita destacar sobre las demás de alguna forma.

Una forma de destacar es tener una interfaz intuitiva. Esto quiere decir que la aplicación no requiere que el usuario tenga que leer un manual para poder utilizarla, sino que él mismo puede descubrir fácilmente cómo utilizar y explotar todas sus funcionalidades. De otra forma, el usuario podría llegar a descartar la aplicación y utilizar otra, buscando otra manera de suplir su necesidad.

Además se requiere que la aplicación sea amigable, de acuerdo a principios básicos establecidos por las compañías desarrolladoras de sistemas operativos para dispositivos móviles y de uso general de aplicaciones móviles, teniendo en cuenta las limitaciones que posee un dispositivo de este tipo (por ejemplo, el tamaño de la pantalla).

3.4.4 Portabilidad

La cantidad y variedad de dispositivos móviles del mercado, y los diferentes sistemas operativos que estos utilizan, hacen que al momento de desarrollar una aplicación sea esperable que la misma pueda portarse en la mayoría de las plataformas existentes, con el objetivo de llegar a la mayor cantidad de usuarios.

Como se mencionó anteriormente, cada sistema operativo utiliza un lenguaje de programación diferente, por lo que se buscará la forma de cubrir la mayor cantidad de sistemas operativos con el menor esfuerzo a la hora de reescribir código.

3.4.5 Uso Eficiente de la Batería

Una de las problemáticas que tienen las aplicaciones que hacen uso de sensores y redes de telefonía celular, es el excesivo consumo de batería que tienen estas funciones. Algunas cosas son inevitables, pero será deseable que la aplicación haga un uso eficiente de la batería, dejando asimismo configurar algunos parámetros que mejoren el uso la batería en pos de la calidad del servicio.

3.4.6 Seguridad

En el ámbito de una aplicación con información sensible, es de esperar que se establezcan protocolos de seguridad adecuados para que la información se intercambie en un entorno confiable. De no ser así, la información podría utilizarse de forma inadecuada, lo que incluso podría llegar a comprometer la integridad física del usuario.

4 Diseño del Sistema

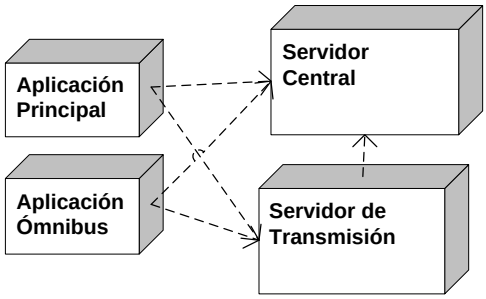
En esta sección se presenta una descripción general de la arquitectura del sistema. Se detalla el diseño de las bases de datos utilizadas, así como la descripción de los servicios utilizados para su consulta.

Además se explican y justifican algunas decisiones de diseño, como el mecanismo utilizado para la transmisión de datos en tiempo real y el diseño de la interfaz gráfica.

Una presentación más detallada de estos puntos se puede consultar en el *Anexo III: Diseño del Sistema*.

4.1 Arquitectura general del sistema

A nivel general, el sistema está compuesto por cuatro subsistemas bien definidos, el **Servidor Central**, el **Servidor de Transmisión**, la **Aplicación Principal** para dispositivos móviles, y la **Aplicación para Ómnibus**.



Cada aplicación se comunica con ambos servidores utilizando las interfaces de servicios que estos proveen. Los servicios que provee el *Servidor Central* pueden ser accedidos mediante la invocación de servicios web, mientras que toda la comunicación con el *Servidor de Transmisión* se realiza mediante una conexión de datos persistente.

Debido al carácter geográfico del sistema es necesario contar con servicios geográficos de mapas, geo-codificación, y otros. El *Servidor de Mapas* se encarga de brindar estos servicios a través de una interfaz de servicios REST, pero no se incluye aquí por ser una dependencia externa al proyecto.

A continuación se presenta el diagrama de arquitectura del sistema, y en las siguientes secciones se describe en más profundidad cada uno de los subsistemas.

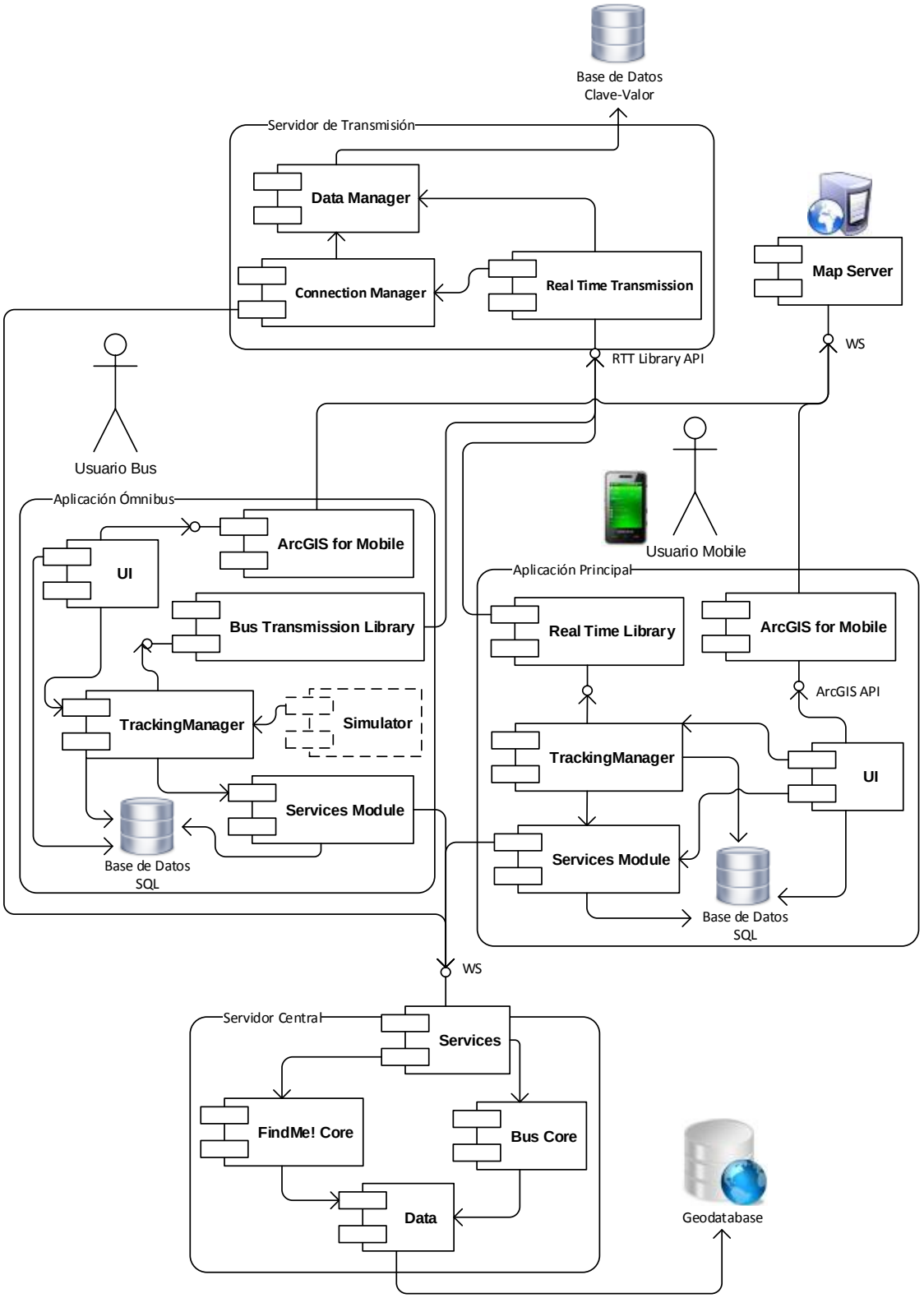


Fig. 4.1 Diagrama de Arquitectura del Sistema

4.2 Servidor Central

Está encargado de proveer funcionalidades a la *Aplicación Principal* y a la *Aplicación de Ómnibus* a través de una capa de servicios REST, tales como gestión de usuarios, creación de puntos de encuentro, y datos de ómnibus como líneas y paradas.

A su vez provee algunos servicios web al *Servidor de Transmisión* para poder realizar controles de autorización de usuarios.

4.2.1 Componentes de Software

El *Servidor Central* presenta una arquitectura de software de tres capas: lógica de negocio, persistencia, y servicios. A su vez, la capa de negocio se divide en dos componentes, respectivos a los requerimientos de *¿Cuándo llega?* y *FindMe.* A continuación se presentan los componentes que conforman el servidor.

FindMe! Core

Es el componente que contiene la lógica de negocio de la aplicación *FindMe!*, como autenticación, gestión de usuarios, manejo del flujo de solicitudes de amistad, puntos de encuentro, y demás funcionalidades de la aplicación.

Bus Core

Contiene servicios de datos de ómnibus utilizados por *¿Cuándo llega?*, así como la *Aplicación de Ómnibus*. No realiza ningún tipo de procesamiento.

Data

Módulo de acceso a datos, su objetivo es realizar el mapeo del esquema de datos a objetos de forma conveniente para cada entidad persistente del sistema, como paradas, líneas de ómnibus, usuarios, puntos de encuentro, etc.

Services

Expone las funcionalidades de los otros módulos mediante una API de servicios. Esta API puede ser consumida de forma remota a través de servicios REST.

4.2.2 Base de Datos basada en Documentos

La base de datos del servidor almacena la información que utilizan las aplicaciones móviles. Se utiliza un esquema de datos basado en documentos para almacenar las compañías, líneas, y paradas de ómnibus. También se persisten los usuarios, amigos, solicitudes de amistad, y demás datos de *FindMe.*

A continuación se presentan las colecciones de documentos que conforman la base de datos.

BusCompanies

Colección con las compañías de ómnibus que forman parte del STM.

BusStops

Colección que almacena las paradas de ómnibus de la ciudad, y que cuenta con la calle y esquina de la parada, y su coordenada geográfica.

BusLines

Esta colección almacena un documento por línea específica[44] de ómnibus, junto con sus datos asociados como los identificadores de la compañía de ómnibus a la que pertenece, los identificadores de las paradas que realiza, y la ruta geográfica que conforma el recorrido.

FriendshipRequests

Esta colección almacena las solicitudes de amistad enviadas por los usuarios, en particular, interesa el estado de las solicitudes que puede ser: pendiente, aceptada, rechazada, o bloqueada.

MeetingPoints

Colección con las citas o punto de encuentro entre usuarios, y que incluye título, descripción, coordenada geográfica del punto de encuentro, fecha y hora, e identificadores de los usuarios participantes. Las invitaciones al encuentro también se almacenan dentro del documento.

Users

Esta colección almacena la información de los usuarios de la *Aplicación Principal*, como datos personales, última ubicación conocida del usuario, los identificadores de sus amigos y puntos de encuentros a los que está invitado, historial de búsquedas recientes, y preferencias de la aplicación.

UserHistory

Este sub-documento está embebido en el usuario, y representa una búsqueda realizada por el usuario que puede ser de carácter geográfico o de búsqueda simple. Se utiliza para facilitar un historial de búsqueda al usuario.

UserPreferences

Este sub-documento también está embebido en el usuario, y almacena las preferencias de usuario de *FindMe!* Las preferencias están principalmente relacionadas con el seguimiento de usuario y tienen como fin que el usuario pueda controlar la utilización del GPS y el intervalo de actualización geográfica durante los seguimientos.

4.2.3 Servicios Web

El Servidor Central expone una serie de servicios web que permiten acceder a los datos del mismo de forma remota, así como acceder a ciertas funcionalidades avanzadas como búsqueda geográfica.

4.2.3.1 Servicios de Ómnibus

Este conjunto de servicios es utilizado por la *Aplicación Principal* y la *Aplicación para Ómnibus*, y cuyo fin es obtener los datos de ómnibus necesarios, que incluyen compañías, líneas, y paradas de ómnibus.

[44] Las líneas de ómnibus a menudo presentan variantes en su recorrido y destino, se entiende línea específica como la variante de una línea, por ejemplo, 117 CIUDAD VIEJA o 117 PLAZA INDEPENDENCIA.

4.2.3.2 *Servicios de* FindMe!

Este conjunto de servicios es utilizado exclusivamente por la *Aplicación Principal* y permite acceder a toda la información del usuario, como amigos, puntos de encuentro, búsqueda e historial. A su vez permiten manejar algunos *workflows* como el de solicitud de amistad y las invitaciones a puntos de encuentro, y permite realizar búsquedas de amigos en función a su ubicación geográfica.

A diferencia de los servicios de ómnibus, es necesario contar con un *token* de autenticación para poder acceder a los servicios. Cada *token* de autenticación está asociado a un usuario particular, lo cual permite manejar restricciones de acceso, permitiendo acceder únicamente a la información que pertenece al usuario.

4.2.3.3 *Servicios para el* Servidor de Transmisión

El Servidor de Transmisión tiene requerimientos de alto rendimiento, por lo que utiliza servicios que están optimizados en cuanto al volumen de datos transmitidos y el tiempo de respuesta, y cumplen funciones especializadas como autorización de un usuario para seguir a otro usuario.

4.3 Servidor de Transmisión

Su objetivo es proveer un mecanismo de transmisión de datos en tiempo real entre dispositivos móviles. Implementa una API que sigue el patrón publicar/suscribir, lo cual sumado a la utilización de un protocolo de comunicación liviano y de baja latencia resulta en un mecanismo flexible y eficiente para enviar datos geográficos a uno o más dispositivos móviles.

Cada usuario del sistema se corresponde con un canal de datos geográficos, al cual sus amigos se pueden suscribir luego de que el usuario accede al seguimiento. A su vez, se manejan canales dinámicos para los ómnibus que están activos, en base a la línea del ómnibus y el momento en que comenzó a realizar el recorrido.

Para realizar controles de autorización y acceso, el *Servidor de Transmisión* se comunica con el *Servidor Central* mediante la interfaz de servicios web que este ofrece.

4.3.1 Componentes de Software

El servidor está compuesto por tres componentes de software que giran en torno a la información de sesión de los clientes conectados; el primero se encarga de distribuir las actualizaciones geográficas, el segundo se encarga de mantener la información de sesión actualizada, y por último, el tercer componente se encarga de persistir esta información para mejorar la tolerancia a fallos.

Real Time Transmission

Este componente se encarga de manejar las actualizaciones de información geográfica. Al recibir una actualización geográfica desde un móvil se encarga de transmitir la información a todos los clientes que se hayan suscrito a la misma.

Connection Manager

Gestiona las conexiones de clientes, y responde a las peticiones de suscripción implementando controles de autorización en el seguimiento de usuarios. Su función principal es la de mantener la información de sesión para cada usuario conectado. También se encarga de multiplexar los pedidos de suscripción, encontrando el canal de transmisión adecuado.

Data Manager

Encapsula la persistencia de la información relacionada a las sesiones y canales disponibles en el servidor, y expone estos datos mediante una interfaz sencilla a los otros componentes del servidor.

4.3.2 Base de Datos Clave-Valor

La base de datos almacena la información necesaria para la transmisión de datos, esto es, usuarios conectados, recorridos de ómnibus activos, usuarios que están realizando seguimiento, y *tokens* de autorización.

Para lograr un mayor rendimiento se utiliza una base de datos no relacional del tipo clave-valor. En el caso de la información de usuario las claves están compuestas con el identificador de usuario, mientras que para la información de ómnibus se utiliza el identificador de viaje.

4.3.3 Protocolo de Transmisión

La comunicación que se realiza entre la *Aplicación Principal* y el *Servidor de Transmisión*, y entre la *Aplicación para Ómnibus* y el *Servidor de Transmisión,* está regida por dos respectivos protocolos de alto nivel.

La comunicación se realiza de forma completamente asíncrona, lo cual permite aprovechar mejor los recursos del servidor y atender múltiples pedidos sin necesidad de esperar por acceso a datos o respuestas del servidor central.

El servidor implementa dos APIs, una para el seguimiento de usuarios y otra para el seguimiento de ómnibus. A su vez, el protocolo también define contratos para la *Aplicación Principal* y a la *Aplicación de Ómnibus*, donde se establecen los eventos o mensajes que el servidor enviará a las aplicaciones, y el comportamiento esperado de las mismas.

En la Fig. 4.2 se presenta un flujo típico del seguimiento de usuarios en función del protocolo de comunicación definido.

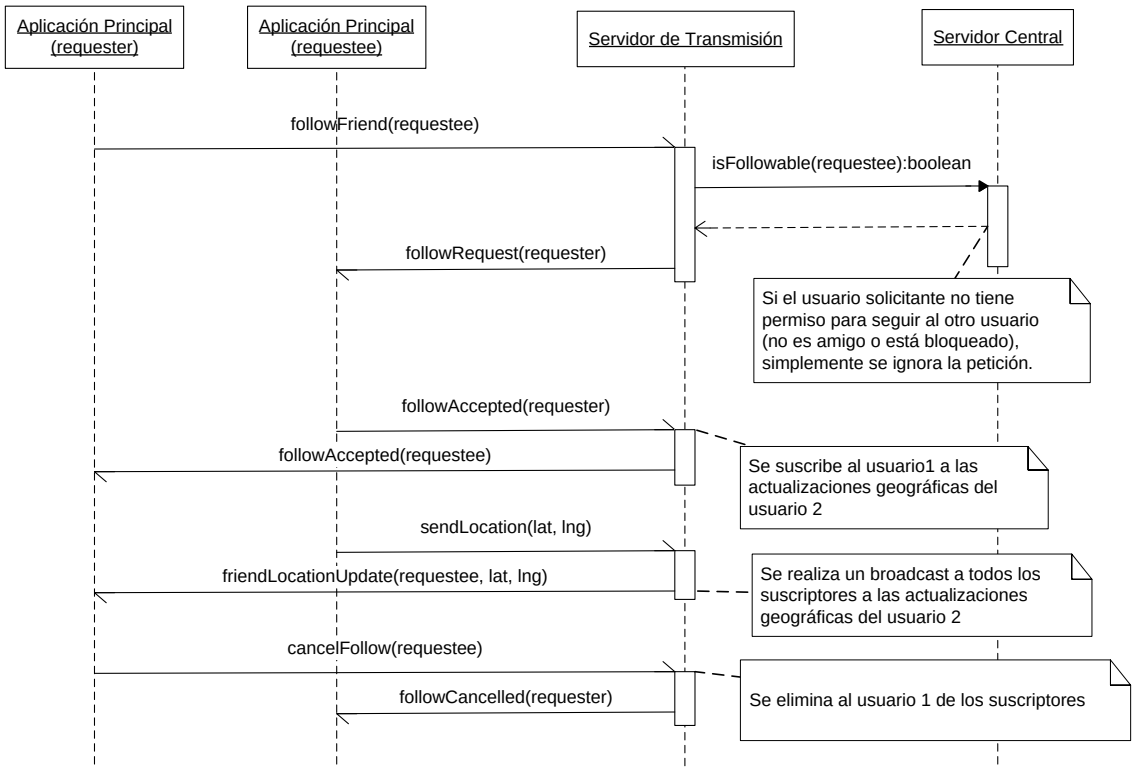


Fig. 4.2 Flujo típico del seguimiento de usuario

4.4 Aplicación Principal

Esta aplicación para dispositivos móviles engloba las funcionalidades de *FindMe* y *Cuándo llega* de forma sencilla e intuitiva, y puede ser utilizada por el usuario final a través de diferentes tipos de dispositivos móviles.

Sus funcionalidades principales son el seguimiento de ómnibus y usuarios en tiempo real, para lo cual hace uso intensivo de la conexión de datos del dispositivo, así como del GPS del mismo si está presente.

Para poder realizar el seguimiento es necesario enviar y recibir datos en tiempo real, lo cual se realiza a través de una conexión persistente con el *Servidor de Transmisión*. El resto de las interacciones, menos críticas en cuanto a tiempo de respuesta, se realizan mediante servicios web provistos por el *Servidor Central*.

El diseño toma en cuenta restricciones típicas de los dispositivos móviles tales como la autonomía, y velocidad de la conexión de datos.

4.4.1 Componentes de Software

La aplicación móvil es compleja ya que además de involucrar la transmisión de datos geográficos al *Servidor de Transmisión* y la visualización de posiciones y recorridos de ómnibus en un mapa, necesita sincronizar la información obtenida desde el *Servidor Central*, y está compuesta por dos juegos de requerimientos que bien podrían conformar aplicaciones

distintas. A continuación se describe la funcionalidad de cada uno de sus componentes de software.

User Interface

Es la interfaz gráfica de la aplicación móvil, y está diseñada para agilizar los casos de uso más frecuentes del usuario, en particular el seguimiento de ómnibus y el seguimiento de amigos. Si bien son funcionalidades parecidas, las situaciones en las que el usuario interactúa con dichos casos de uso pueden ser muy dispares, por lo que es importante que las funcionalidades no resulten invasivas, logrando un balance entre accesibilidad y facilidad de uso.

Debido a que el principal atractivo de la aplicación es la información en tiempo real, es importante que la información exhibida esté actualizada en todo momento. Esto combinado a que la mayoría de las interacciones con el *Servidor de Transmisión* son asíncronas, hace que el componente de interfaz de usuario tenga una complejidad elevada.

Services Module

Este componente se encarga de comunicarse con el *Servidor Central* a través de los servicios REST que este expone. También está encargado de persistir localmente la información obtenida desde el servidor, como paradas y líneas de ómnibus, en una base de datos SQL local, de forma de poder optimizar el acceso a red y mejorar el tiempo de respuesta de la aplicación.

A su vez hay datos que necesitan sincronización bidireccional, como es el caso de los amigos y las solicitudes de amistad de *FindMe!,* por lo que también debe encargarse de mantener la información actualizada, y enviar los cambios al servidor.

Por último, tiene como responsabilidad notificar al componente de interfaz de usuario cuando la información ha cambiado, de forma de actualizar la interfaz.

Tracking Manager

Este módulo está encargado de llevar un registro de los seguimientos en los que está participando el usuario, y estados de conexión de los amigos del usuario. Para esto utiliza las funcionalidades brindadas por la *Real Time Transmission Library*. Una de sus responsabilidades principales es la de enviar actualizaciones geográficas, para lo cual controla el intervalo de tiempo entre las actualizaciones y consulta la información de GPS y red para obtener la ubicación del usuario.

También tiene como responsabilidad notificar al componente de interfaz de usuario cuando se recibe nueva información. Debido a que las actualizaciones pueden llegar a ser muy frecuentes y toda la comunicación con el *Servidor de Transmisión* es asíncrona, el mecanismo de comunicación con el componente *User Interface* se diseñó para liviano y eficiente.

Por último, implementa el *Algoritmo de Ómnibus más Cercanos,* que calcula los ómnibus más cercanos a una parada de ómnibus particular que se encuentran realizando el recorrido activamente y aún no han pasado la parada.

Real Time Transmission Library

Este componente implementa el lado cliente de la comunicación con el *Servidor de Transmisión*, en particular cumple con el *Contrato para Seguimiento de Ómnibus* y con el *Contrato para Seguimiento de Usuarios*, y utiliza las API de seguimiento.

ArcGIS for Mobile

Para poder mostrar la información geográfica al usuario y permitir que interactúe con la misma, la aplicación hace uso extensivo de mapas. Debido a que una de las restricciones del proyecto es utilizar tecnologías de ESRI, para ello se utiliza la biblioteca ArcGIS, que se encuentra disponible para la gran mayoría de las plataformas móviles, y permite mostrar mapas y realizar operaciones geográficas sencillas.

4.4.2 Base de Datos SQL

Para reducir la utilización de red y optimizar los tiempos de respuesta, la gran mayoría de los datos utilizados en la aplicación se persisten de forma local, sincronizando mediante servicios web con el *Servidor Central* para mantener la información actualizada.

Los servicios de sincronización solo envían la información que ha cambiado desde la última sincronización, que está determinada por un *timestamp* que se envía en la llamada a los servicios.

Se utiliza una base de datos relacional principalmente por su disponibilidad en las plataformas móviles, ya que la mayoría ofrecen una base de datos relacional, ya sea mediante SQLite en Android e iOS, LINQ to SQL en Windows Phone, o a través de WebSQL en HTML5.

A continuación se detallan las entidades de la base de datos local.

BusCompanies

Tabla con las compañías de ómnibus que forman parte del STM.

BusLines

Esta tabla contiene las líneas de ómnibus del sistema.

BusStops

Tabla que almacena las paradas de ómnibus de la ciudad.

StopLines

Tabla que representa la relación entre paradas de ómnibus y líneas de ómnibus.

FriendshipRequests

Esta tabla contiene las solicitudes de amistad enviadas y recibidas por el usuario.

MeetingPoints

Tabla con las citas de punto de encuentro que el usuario creó o a las que fue invitado.

MeetingPointInvitations

Tabla que representa la relación entre un punto de encuentro y los usuarios invitados al mismo.

Friends

Esta tabla almacena la información de los amigos del usuario.

FriendSearchHistory

Esta tabla almacena las búsquedas de amigos realizadas por el usuario que pueden ser de carácter geográfico o de búsqueda simple. Se utiliza para facilitar un historial de búsqueda al usuario.

BusTrackHistory

Esta tabla almacena los últimos seguimientos de ómnibus realizados por el usuario, se utiliza para poder brindarle acceso rápido para seguimientos frecuentes

FavouriteBusStops

Tabla que almacena las paradas de ómnibus favoritas del usuario.

4.4.3 Algoritmo de Ómnibus más Cercanos

Este algoritmo se utiliza para calcular los ómnibus más cercanos a una parada de ómnibus particular que se encuentran realizando el recorrido activamente y aún no han pasado la parada. Se utiliza para automatizar el seguimiento de ómnibus, de forma de simplificar la interacción del usuario.

Al momento de consultar la ubicación en tiempo real para una línea de ómnibus, la *Aplicación Principal* realiza un pedido al *Servidor de Transmisión*, recibiendo los ómnibus activos para esa línea junto con su última ubicación conocida. A partir de esa información, el recorrido del ómnibus, y la ubicación geográfica de la parada de ómnibus se ejecuta el algoritmo, obteniendo los ómnibus o *viajes* más cercanos a la parada, que son los que resultan de mayor interés al usuario en un momento dado.

4.4.4 Diseño de la Aplicación

Uno de los retos del desarrollo de aplicaciones móviles es el diseño de la aplicación. El diseño de una aplicación está conformado en su mayor parte por dos componentes que van de la mano: el diseño de la experiencia de usuario, y el diseño de la interfaz gráfica [125].

El diseño de experiencia de usuario se trata de definir los objetivos de la aplicación, como qué funcionalidades incluir, y cómo permitir que el usuario realice estos objetivos. En cambio, el diseño de interfaz gráfica define como es esa experiencia visualmente. Esto incluye los colores, texturas, y fuentes que se utilizan para construir el estilo visual de la aplicación.

Realizar el diseño de la *Aplicación Principal* fue un proceso arduo y extenso; a continuación se presentan ciertos aspectos a modo de resumen. El proceso se encuentra plenamente documentado en la sección 4.4 del *Anexo III: Diseño del Sistema*.

4.4.4.1 Diseño de Interacción

Para empezar, se dibujaron bosquejos de la interfaz gráfica o *wireframes*. En principio se utilizó la herramienta web Balsamiq [126] para realizar los primeros bocetos de la aplicación, manteniendo un fuerte foco en el contenido de la aplicación y las interacciones y dejando de lado los detalles menores. De esta forma se pudieron validar rápidamente los flujos principales de la aplicación, y obtener una primera noción de los componentes gráficos a utilizar y de la navegación de la aplicación.

Fig. 4.3 Comienzo del flujo del seguimiento de ómnibus

4.4.4.2 Diseño Gráfico

Una vez que los bocetos alcanzaron el nivel esperado de detalle y usabilidad, se pasó a la etapa de diseño gráfico. Esta vez se decidió realizar los bocetos de forma manual, lo cual si bien no era del todo fiel al modelo real, permitía identificar rápidamente los componentes gráficos a utilizar, por lo que resultaron de gran utilidad durante el desarrollo.

Fig. 4.4 Boceto de 'Perfil de Usuario'

4.4.4.3 Diseñar para cada Plataforma

Debido a que hay tantas plataformas y dispositivos, al diseñar para móviles es importante saber cuál será la plataforma específica para la cual se desarrollará el producto final, ya que cada una tiene su propio paradigma y convenciones de diseño.

Para el diseño de la aplicación HTML5 se siguieron las convenciones de iOS6 [127], debido a que se utilizó el *framework* Sencha Touch, el cual tiene arraigadas las convenciones de estilo, navegación, e iconografía de esa plataforma. Esto es una desventaja cuando la aplicación se accede desde dispositivos de otras plataformas, ya que poseen convenciones diferentes y la usabilidad dela aplicación puede verse gravemente deteriorada.

En cambio, para la aplicación Android se respetaron los paradigmas de diseño propios de la plataforma [69], diferenciándose sobre todo en las diferencias de navegación debido al *back button* presente en los dispositivos Android que permite volver a la pantalla anterior, y la categorización de los dispositivos en base a tamaño y resolución de pantalla.

Para darle identidad a la aplicación se diseñaron versiones personalizadas de los controles utilizando una paleta de colores cálidos, con una fuerte presencia de amarillos y anaranjados. Esto ayuda al usuario identificar la aplicación en la que se encuentra en todo momento.

Fig. 4.5 Algunos componentes hechos a medida para la aplicación Android

En cada aplicación se utilizó una iconografía acorde a las convenciones descritas en la sección anterior, de forma de representar los conceptos de forma intuitiva y consistente en todas las pantallas de la aplicación.

Fig. 4.6 Íconos de menú de la aplicación Android

Fig. 4.7 Íconos de menú de la aplicación Sencha

Para permitir que el usuario pueda distinguir las líneas de ómnibus a las que está siguiendo con mayor facilidad, se utilizaron colores diferentes según la compañía de ómnibus a la cual pertenece la línea, tanto para los marcadores como para los recorridos que se muestran en el mapa. En lo posible se trataron de usar los colores característicos de cada compañía, de modo que la asociación entre los colores y la empresa resulten familiares.

Figura 4.8 Colores asociados a las distintas compañías del STM

En las aplicaciones móviles cada pantalla debe tener un propósito bien definido, su diseño debe girar en torno al objetivo principal que quiere lograr el usuario. Esto se logró enfatizando las acciones más importantes (a través de color, ubicación, o tamaño), y reduciendo la importancia visual de los elementos secundarios.

Un ejemplo de esto se encuentra en el perfil de usuario, donde la acción más importante es comenzar el seguimiento. Si bien hay varias acciones disponibles, como invitar al usuario a un punto de encuentro, ninguna compite con la acción principal.

Debido a que la vista principal de la aplicación es donde el usuario pasa la mayor parte del tiempo, ya que permite seguir amigos y ómnibus a través de un mapa, se utilizaron transparencias en la barra de menú y los botones que se despliegan sobre el mapa, de forma de sacarle el mayor provecho posible al tamaño de pantalla del dispositivo.

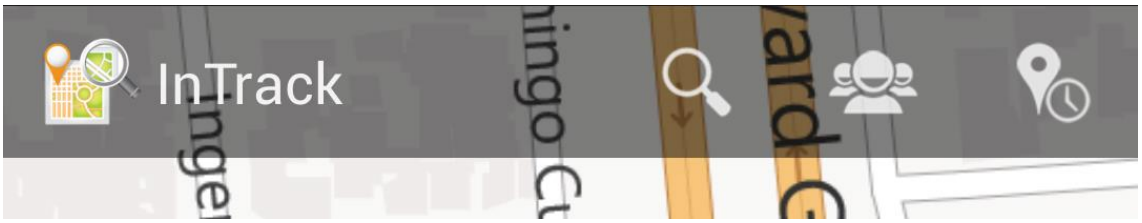


Fig. 4.9 Barra de menú de la aplicación con transparencia

4.5 Aplicación para Ómnibus

La *Aplicación para Ómnibus* es una aplicación para dispositivos móviles pensada para estar correr en dispositivos que estén a bordo de los ómnibus, y hace uso intensivo del GPS y la conexión de datos del dispositivo, con el fin de enviar la posición geográfica del ómnibus en tiempo real al *Servidor de Transmisión*, el cual luego hará difusión de la misma a todos los usuarios que estén realizando seguimiento al ómnibus.

Presenta una interfaz de usuario muy simple que permite seleccionar la línea de ómnibus, visualizar el recorrido de la línea y la posición actual del ómnibus, y empezar o detener la transmisión durante el recorrido. Los datos de ómnibus son obtenidos a través del *Servidor Central.*

A su vez brinda la posibilidad de simular recorridos, para lo cual se brinda una interfaz muy similar, pero con opciones variadas como control de velocidad, pausado de recorrido, y con la posibilidad de simular múltiples recorridos al mismo tiempo. Esto es crucial para poder realizar pruebas en el ámbito del proyecto, ya que no es necesario que el dispositivo móvil se encuentre a bordo de un ómnibus en movimiento.

4.5.1 Componentes de Software

Esta aplicación móvil tiene comparte la arquitectura con la *Aplicación Principal*, con la diferencia de que se agrega un componente de simulación de recorridos de ómnibus que interactúa con el módulo de seguimiento.

User Interface

Es la interfaz gráfica de la aplicación móvil, y está diseñada con foco en la facilidad de uso. La visualización de la posición geográfica se ofrece como una funcionalidad secundaria, pero es un componente de complejidad baja.

Services Module

Al igual que su contraparte en la aplicación principal, este componente se encarga de comunicarse con el *Servidor Central* a través de los servicios REST que este expone, persistiendo localmente la información de ómnibus en una base de datos relacional. Como los datos de ómnibus son muy estables y la aplicación no realiza manipulación de datos, la única responsabilidad del componente es mantener la información de ómnibus actualizada.

Tracking Manager

Este módulo está encargado de controlar el ciclo de vida del viaje al iniciar y finalizar el recorrido el ómnibus, enviando actualizaciones geográficas de la posición del ómnibus al *Servidor de Transmisión*. Para ello utiliza la *Bus Transmission Library.*

El componente tiene una interfaz sencilla y bien definida que permite ser utilizado tanto por el simulador como por el usuario a través de la interfaz gráfica.

Simulator

Este componente se encarga de simular recorridos de ómnibus, utilizando el *Tracking Manager* para manejar el estado de cada recorrido, pero simulando la posición geográfica de cada móvil de forma similar a como lo haría un ómnibus que hace el recorrido de una línea particular. Permite variar la velocidad y el estado de los recorridos simulados, y simular múltiples recorridos de forma simultánea.

Bus Transmission Library

Este componente implementa el lado cliente de la API de comunicación con el *Servidor de Transmisión*, en particular cumple con el *Contrato para la Aplicación de Ómnibus*, y utiliza la API de producción de datos de ómnibus.

ArcGIS for Mobile

Como funcionalidad secundaria se muestra el recorrido de la línea y la posición geográfica del ómnibus en un mapa, para lo cual también se utiliza la biblioteca ArcGIS.

4.5.2 Base de Datos SQL

Los datos de ómnibus que necesita la aplicación se almacenan localmente en una base de datos relacional, para que se disponga de los datos aún si el *Servidor Central* no estuviese disponible. A continuación se detallan las entidades de la base de datos local.

BusCompanies

Tabla con las compañías de ómnibus que forman parte del STM.

BusLines

Esta tabla contiene las líneas de ómnibus del sistema.

Rides

Esta tabla contiene las simulaciones de ómnibus activos, no se utiliza cuando la aplicación opera en el modo normal. Un viaje o *ride* está identificado por una línea de ómnibus, y el momento en que comienza el recorrido.

5 Implementación

En este capítulo de describen en forma general las decisiones tomadas sobre la implementación del sistema, las dificultades que se presentaron, y cómo se resolvieron. La implementación del sistema consiste en tres etapas: desarrollo de los componentes de software, obtención de datos, y puesta en producción.

En primer lugar, el desarrollo del sistema involucra la construcción del *Servidor Central, el Servidor de Transmisión,* la *Aplicación Principal,* y la *Aplicación para Ómnibus*. Cabe destacar que uno de los objetivos del proyecto es que la *Aplicación Principal* se encuentre disponible para múltiples plataformas, por lo cual fue necesario evaluar qué herramientas utilizar para realizar el desarrollo, y qué plataformas iban a ser soportadas.

En segundo lugar, para poder llevar a cabo los requerimientos de *¿Cuándo llega?* de forma que se sienten las bases del trabajo futuro resultaba conveniente contar con información de ómnibus real, de modo que fue necesario investigar las fuentes de datos existentes y tener en consideración cómo adaptarlas al análisis realizado anteriormente.

Por último, para poder realizar pruebas realistas del sistema fue necesario ponerlo en producción, lo cual debido al carácter distribuido del mismo presentó varios desafíos. En particular, se cubren en detalle los problemas que presenta la comunicación en tiempo real entre dispositivos móviles.

5.1 Servidor de Transmisión

El seguimiento de ómnibus y amigos en tiempo real es parte fundamental de la aplicación, por lo que es crítico que el mecanismo de comunicación sea de muy baja latencia. Sin embargo, debido las condiciones de red a las que se enfrentan los dispositivos móviles es importante que sea eficiente.

Habiendo investigado una gran variedad de protocolos y bibliotecas (ver Anexo I), se decidió implementar la comunicación entre el *Servidor de Transmisión* y las aplicaciones móviles mediante Socket.IO, una biblioteca que corre sobre el *framework* Node.js.

5.1.1 Socket.IO

Implementar el servidor a través de Socket.IO tiene varios beneficios en comparación con realizar la implementación directamente a través de *WebSockets*.

En primer lugar, permite utilizar varios tipos de transporte además de *WebSockets*, como *FlashSockets*, *XHR* y *JSONP polling*, lo cual brinda mayor flexibilidad y permite que la aplicación funcione en caso de que no se pueda establecer una conexión exitosa mediante *WebSockets*.

Además, al tener el concepto de eventos, una cadena de caracteres con un paquete de datos asociados, implementar los mensajes del *Protocolo de Transmisión*[45] se realiza de forma directa.

[45] Ver Anexo III: Diseño del Sistema, sección 3.3

Finalmente, permite realizar *broadcasts* a todos los clientes conectados, y provee soporte para publicar y suscribir a través del concepto de *rooms* o salas. Una sala de Socket.IO es un conjunto de clientes conectados, a la cual se pueden unir sockets, y permite enviar mensajes a todos los clientes presentes en la sala de forma muy simple.

5.1.2 Difusión de Eventos

Para realizar la difusión de ciertos eventos fue necesario crear *salas*, el servidor se encarga de que cada cliente se una a las salas correspondientes.

Salas de Amigos del Usuario

Cada usuario del sistema tiene una sala a la cual pertenecen los amigos del usuario. Esto permite notificar a los amigos del usuario cuando este se conecta o desconecta, y es el mecanismo que se utiliza para mostrar el estado de cada usuario en la *Aplicación Principal*.

Una vez que la aplicación está conectada al *Servidor de Transmisión*, se agrega al usuario a la sala de cada uno de sus amigos notificando que el usuario se ha conectado. Los amigos del usuario se obtienen a través de un servicio de baja latencia que provee el *Servidor* Central.

Al desconectarse la aplicación del *Servidor de Transmisión*, el usuario abandona la sala y notifica a sus amigos que se ha desconectado.

Salas de Seguidores del Usuario

Asimismo, para poder enviar el mensaje de actualización geográfica a los usuarios que están realizando seguimiento a un amigo, el sistema tiene una sala con los seguidores de un usuario.

Cuando un usuario acepta la petición de seguimiento de uno de sus amigos, el sistema agrega al amigo a la sala de seguidores del usuario. De esta forma, cuando el usuario envía una actualización de su posición geográfica el servidor realiza una difusión de la actualización a todos sus seguidores.

Cuando una de las dos partes (seguido o seguidor) cancela el seguimiento, se quita al seguidor de la sala de seguidores del usuario seguido.

Salas de Ómnibus

El seguimiento de ómnibus funciona de forma similar al seguimiento de usuarios, pero no hay necesidad de realizar el flujo de aprobación del seguimiento, cuando la aplicación solicita seguimiento a un ómnibus, se la agrega a la sala de seguidores del ómnibus en el acto.

Cabe aquí destacar que a diferencia de los usuarios, los *viajes*[46] tienen un ciclo de vida mucho más corto por lo que se puede considerar que las salas de seguidores de ómnibus son dinámicas en el sentido de que la sala solo es utilizada por el período de tiempo en el que un ómnibus realiza el recorrido de una línea.

[46] Recordamos que un viaje representa a un ómnibus realizando el recorrido de una línea particular en un momento dado.

5.1.3 Redis

El sistema sería muy poco robusto si la información que maneja el *Servidor de Transmisión* (usuarios conectados, seguidores, recorridos de ómnibus activos) fuera volátil. Por este motivo la información se persiste utilizando Redis, un sistema de almacenamiento clave-valor de alto rendimiento.

Una consecuencia muy favorable de utilizar Redis para almacenar y consultar esta información es que es posible tener múltiples instancias del *Servidor de Transmisión* que accedan al mismo servidor Redis, permitiendo así escalar el sistema de forma horizontal con facilidad ya que el número de conexiones activas simultáneas aumenta proporcionalmente al número de servidores Node.js disponibles.

5.1.4 Transmisión de Datos Binarios

Si bien el punto fuerte de Socket.IO es la transmisión de texto plano, es posible transmitir datos binarios codificados en Base64 [128] ya que los datos binarios son convertidos a código ASCII.

De esta forma, es posible utilizar la arquitectura actual para enviar cualquier tipo de dato entre dos dispositivos móviles, sin embargo se estima que una solución a medida proveería un mejor rendimiento.

5.2 Servidor Central

Al momento de implementar el Servidor Central había un cierto grado de libertad respecto a las tecnologías a utilizar, ya que a nivel de base de datos se requería de búsqueda geográfica para uno de los servicios, pero el resto de la implementación no tenía requerimientos especiales ni restricciones de performance.

5.2.1 MongoDb

Se decidió utilizar MongoDb como sistema de base de datos ya que posee las capacidades geográficas requeridas por el servidor, en particular, consultas por cercanía geográfica. Una de las grandes ventajas de utilizar MongoDb es que al estar basado en documentos permite almacenar estructuras no planas, como líneas de ómnibus y su recorrido, lo cual simplifica enormemente el mapeo de datos a objetos. Por este mismo motivo, serializar a JSON los datos requeridos por los servicios de la API resulta extremadamente sencillo, ya que tienen la misma estructura.

5.2.2 Sinatra

Para implementar los servicios se eligió Sinatra, ya que este *framework* web basado en el lenguaje Ruby. Una de las características más interesantes de Sinatra es que tiene un DSL[47] para las rutas que permite definirlas a partir de los verbos HTTP, por lo cual resulta muy práctico para definir una API de servicios REST.

[47] DSL: Domain-Specific Language

Otra característica muy conveniente de Sinatra es que al modificar el código de la aplicación se recargan automáticamente los archivos modificados [129], lo cual permite iterar rápidamente durante el desarrollo sin necesidad de recompilar o reiniciar el servidor.

Un punto fuerte para utilizar Ruby es que se integra muy bien con MongoDb a través del ODM[48] Mongoid [130], el cual mapea automáticamente los documentos a objetos sin necesidad de escribir código a medida.

La elección de Sinatra fue arbitraria, y estuvo motivada principalmente porque ya se tenía experiencia con el *framework* y se consideró que esto aumentaría la productividad del equipo.

5.2.3 Estructura del Servidor

Se puede considerar que la arquitectura de la solución está distribuida en tres capas: rutas, que son la interfaz de los servicios web; modelos, que son los objetos correspondientes a la lógica de negocio; y el acceso a datos, que se encarga de la persistencia y el mapeo entre objetos y documentos de la base de datos.

Figura 5.1: Capas del Servidor Central

Rutas

En esta capa se definen las rutas, que son las interfaces de los servicios web. Cada ruta se define a partir del verbo HTTP correspondiente y de los parámetros que acepta.

Las responsabilidades de la capa consisten en validación y procesamiento de parámetros, y multiplexado a los modelos según la operación correspondiente.

Modelos

Cada modelo representa a un objeto del dominio, e implementa la lógica de negocio relacionada al mismo.

En el caso de los servicios de usuario esto incluye el manejo del flujo de solicitudes de amistad e invitaciones a puntos de encuentro, búsqueda de amigos de amigos, y búsqueda geográfica de amigos.

Por otro lado, los servicios de ómnibus ofrecidos se mapean directamente a operaciones básicas de acceso a datos, por lo cual la lógica resulta muy sencilla.

Persistencia

La capa de persistencia está implícita en la definición de los modelos, debido a que se utiliza Mongoid como ODM. Esto quiere decir que se define el esquema del documento, incluyendo

[48] ODM: Object-Document Mapper

campos y relaciones con otros documentos, directamente en el modelo, pudiendo luego interactuar con la base de datos a través de métodos que brinda Mongoid.

5.3 Aplicación Principal

Uno de los objetivos del proyecto es comprobar la factibilidad de un desarrollo multiplataforma a través de HTML5 de una aplicación móvil relativamente compleja, en particular, con funcionalidades de visualización de datos geográficos.

Para poder realizar dicha evaluación se consideró relevante comparar la aplicación HTML5 con una aplicación nativa que implementara las mismas funcionalidades. Para ello, se realizó el desarrollo de una versión nativa de la *Aplicación Principal* para la plataforma Android.

5.3.1 Evaluación de GeneXus X Ev2

Si bien en la propuesta original del proyecto se planteaba el desarrollo de la aplicación tanto para Android como para iOS mediante la herramienta GeneXus X Ev2 [75], al investigar la herramienta se encontraron varias limitaciones severas que hicieron que se descartara la idea de utilizarla.

A continuación se comentan algunas características y ventajas de la herramienta, las limitaciones de la misma, y los motivos por los cuales se decidió que no cumplía con los requerimientos mínimos para ser utilizada en el proyecto.

Ventajas

GeneXus provee una interfaz de localización para interactuar con el hardware del dispositivo, como el GPS. La interfaz brinda varias operaciones con las que se puede, entre otras cosas, obtener la posición geográfica actual del dispositivo, obtener la distancia entre dos puntos y obtener la dirección de una posición geográfica [131]. Esta información puede ser procesada o desplegada en un mapa que se provee como un control nativo en el *framework* [132].

A su vez permite generar automáticamente los programas de ABM, así como generar el código nativo para múltiples plataformas con el fin de facilitar el desarrollo de la aplicación para varios dispositivos. [75]

Limitaciones

Una de las principales limitaciones que se encontró al momento de evaluar la herramienta fue la falta de soporte nativo para la API de ArcGIS. Cualquier API o biblioteca específica para iOS o Android que no esté soportada por GeneXus de forma nativa debe ser implementada mediante *External Objects* [133] [134] o *User Controls* [135] [136].

Debido a esta limitación y a que el uso de ArcGIS para Mobile es uno de los requerimientos del proyecto, el software deja de ser portable, ya que se debe codificar lo mismo en Objective-C y Java, siguiendo las restricciones que impone GeneXus.

Otra limitación importante es que los *User Controls* deben ser escritos y compilados con los SDK provistos por las empresas que desarrollan dicho sistema operativo [135] [136], por lo que en el caso de iOS esto significa que es necesario contar con una computadora con sistema operativo de Apple.

Por último, para la aplicación móvil GeneXus fuerza una arquitectura Cliente-Servidor que utiliza servicios REST[49], lo cual también es una limitante ya que debido a los requerimientos de comunicación en tiempo real del sistema, es fundamental combinar este enfoque con una arquitectura orientada a eventos.

Conclusión

Uno de los objetivos a la hora de utilizar GeneXus era la reutilización de la implementación para las distintas plataformas, tomando ventaja de las abstracciones de los lenguajes y *frameworks* particulares que GeneXus provee.

Al ser necesario el uso de *External Objects* y *User Controls*, este objetivo se ve frustrado, haciendo que no sea posible realizar una implementación genérica para todas las plataformas. Sumado esto a que restringe la arquitectura de la aplicación y no provee una ventaja clara, se decidió descartar GeneXus Ev2 en función de una implementación nativa en Android, y manteniendo la idea de desarrollar también la aplicación mediante HTML5.

5.3.2 Desarrollo Nativo para Android

La decisión de desarrollar la versión nativa de la aplicación para Android se vio influenciada en gran parte por el hecho de que es una plataforma *open-source* que brinda herramientas de desarrollo para la mayoría de los sistemas operativos; para desarrollar aplicaciones para iOS es necesario contar con un equipo con un sistema operativo de Apple, del cual no se disponía. Es por esta razón que se resolvió no desarrollar una versión nativa para iOS, quedando esta plataforma dentro del grupo de aquellas que usarán la versión HTML5 de la aplicación.

5.3.2.1 Sincronización y Consumo de Batería

Para optimizar el tiempo de respuesta de la aplicación se decidió tener la gran mayoría de los datos que necesita la aplicación replicados en una base de datos SQLite local. A su vez, para reducir el impacto en el consumo de batería la sincronización se realiza siguiendo las mejores prácticas de acceso a red en dispositivos móviles.

La utilización de la conexión de datos inalámbrica es una fuente importante del consumo de batería del dispositivo. Cada vez que se realiza un acceso a red es necesario encender la radio inalámbrica del dispositivo, la cual permanece encendida por un cierto período de tiempo hasta que se detecta un cierto intervalo de inactividad. En promedio, cada sesión de acceso a red implican 20 segundos de consumo de batería [137], sin importar el tamaño de la transferencia de datos asociada.

[49] REST (Representational State Transfer)

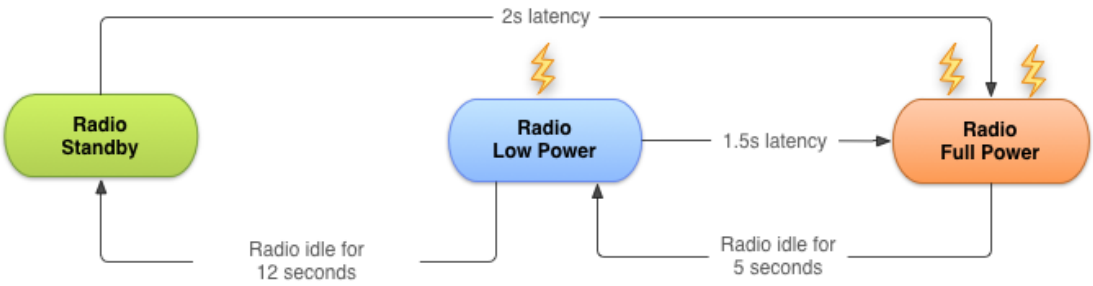


Figura 5.2: Máquina de estado de una radio 3G típica

Por este motivo resulta más eficiente traer todos los datos necesarios una vez que se ha encendido el radio, que realizar varios accesos para traer la misma información. Al agrupar las descargas no sólo se reduce el consumo de batería al disminuir el número de veces que se activa el radio, sino que además se reduce la latencia porque se utiliza la radio cuando está utilizando toda su capacidad.

Al momento de sincronizar los datos con el *Servidor Central* se decidió tomar un enfoque de caché agresivo basado en *timestamps*, de forma que sólo se transfieran aquellos datos que han sido creados, modificados, o eliminados desde la sincronización anterior. Esto reduce las descargas necesarias, optimizando el acceso a red y el consumo de batería.

Una buena práctica de acceso a red es intentar reutilizar las conexiones, por lo cual la sincronización de todos los datos del sistema se realiza mediante un único pedido HTTP, que contiene todos los datos que necesita ser actualizados, como amigos del usuario, puntos de encuentro, etc.

Esto quiere decir que cuando se activa una actualización, ya sea por una acción del usuario como refrescar las solicitudes de amistad, o automáticamente por la aplicación, se aprovecha la comunicación con el *Servidor Central* para traer todos los datos.

5.3.2.2 Comunicación en Tiempo Real

Debido a que la comunicación con el *Servidor de Transmisión* se realiza a través de Socket.IO, la recepción de mensajes o eventos es asíncrona. Debido a esto, fue necesario implementar un servicio en *background* que reciba y procese los eventos.

Si bien es una buena práctica realizar las operaciones de red en *background* en lugar de realizarlas en el hilo de ejecución principal, que está encargado de dibujar la interfaz gráfica, es necesario tener un mecanismo de comunicación bien definido entre el servicio en *background* y la vista que está corriendo en primer plano.

Cada vista de la aplicación responde a los eventos de manera instantánea, lo cual es muy importante para poder brindar una buena experiencia de usuario y dar la sensación de tiempo real.

Notificaciones mediante el LocalBroadcastManager

Como a priori no se conoce en qué pantalla se encuentra el usuario en un momento dado, es necesario desacoplar las vistas particulares de la notificación de eventos. Esto se resolvió mediante la emisión de eventos locales a través del *LocalBroadcastManager* [138], cada

evento que se recibe en el servicio a través de Socket.IO es procesado, y emitido como un evento local.

Cada vista de la aplicación se registra para recibir los eventos que afectan su interfaz gráfica, por ejemplo, la vista de amigos se registra para los eventos de conexión y desconexión de amigos.

Este mecanismo de comunicación es muy eficiente, ya que sólo se procesan los eventos relevantes para la vista que se encuentra en primer plano.

Actualizaciones del contenido a través de ContentObservers

Otro mecanismo de actualización utilizado con menor frecuencia está basado en *ContentObservers* [139], que permiten recibir llamadas cuando se actualizan ciertas entidades en la base de datos SQLite local.

De este modo cuando se sincroniza la base de datos con el *Servidor Central*, las vistas reciben una notificación de que la información que consultaron ha cambiado y que deben realizar una nueva consulta.

Si bien la mayoría de las actualizaciones de la interfaz gráfica se llevan a cabo a través de eventos locales, en ciertos casos, como las actualizaciones geográficas en el perfil de usuario, se realizan a través de *ContentObservers* ya que permiten reutilizar la lógica para ambos escenarios; actualización provocada por la sincronización, o actualización geográfica recibida a través del *Servidor de Transmisión.*

5.3.2.3 Algoritmo de Ómnibus más Cercano

Al momento de comenzar el seguimiento de una línea particular es necesario realizar el cálculo de los ómnibus activos de la línea que se encuentran más cerca de la parada seleccionada pero aún no han pasado por allí.

En lugar de realizar este cálculo en el servidor, se decidió que sería interesante ver qué herramientas proveía la API de ArcGIS para implementar este algoritmo en las aplicaciones móviles, sumado al hecho de que esto permite reducir la carga del servidor lo cual aumenta la cantidad de clientes que puede atender.

La API provee una función *GeometryEngine.getNearestVertex* que permite proyectar la ubicación de los ómnibus a los vértices de la poli-línea que forma el recorrido de los ómnibus, haciendo que sea posible implementar el algoritmo de forma sencilla a partir del pseudocódigo tal cual se describe en la sección 4.3 del *Anexo III: Diseño del Sistema*.

Al implementarlo en la aplicación móvil fue posible obtener un excelente tiempo de respuesta. Esto se debe a que al momento de ejecutar el algoritmo la aplicación cuenta con el recorrido del ómnibus y la posición de la parada seleccionada, por lo que solo es necesario obtener las posiciones de los ómnibus activos para la línea seleccionada, que se obtienen del *Servidor de Transmisión* con muy baja latencia.

5.3.2.4 Bibliotecas de Software

A continuación se mencionan las bibliotecas de software utilizadas para el desarrollo de la *Aplicación Principal* en Android, haciendo foco en su utilización en el proyecto.

ArcGIS para Android

Si bien en Android es muy común utilizar la API de Google Maps, uno de los requerimientos no funcionales del proyecto es la utilización de ArcGIS para móviles, por lo cual todos los componentes de mapas, y los servicios de geo-codificación son provistos por el SDK de ArcGIS para Android.

Los mapas se utilizan extensivamente en la aplicación en casos de uso variados como son: seleccionar la ubicación de un encuentro, mostrar la última ubicación de un amigo, visualizar las paradas de ómnibus cercanas, y realizar seguimiento en tiempo real de amigos y ómnibus.

ArcGIS maneja el concepto de capas, por lo cual los gráficos utilizados para representar las distintas entidades en el mapa se dividieron en las siguientes capas:

- **MapLayer**: Capa que contiene el mapa de base
- **RoutesLayer**: Muestra los recorridos de los ómnibus que están en pantalla.
- **BusStopsLayer**: Muestra las paradas de ómnibus cercanas al usuario.
- **BusLayer**: Muestra los ómnibus que el usuario está siguiendo.
- **FriendsLayer**: Muestra los amigos del usuario.

Para poder mostrar información relevante al usuario y que el mapa no esté atestado de información, se consideran los niveles de zoom y de escala para mostrar u ocultar los elementos. Por ejemplo, las paradas se muestran cuando la porción del mapa que se encuentra en pantalla no supera las 400 m^2.

AndroidAsync

Esta biblioteca de software es uno de los componentes más importantes de la aplicación. La implementación utilizada [140] no sólo implementa un cliente del protocolo de *WebSockets* para Android, sino que además implementa un cliente de Socket.IO, lo cual es la base de la comunicación en tiempo real con el *Servidor de Transmisión*.

Uno de los problemas encontrados fue la inmadurez del cliente de *WebSockets* en Android, lo cual dificultó seriamente la etapa de desarrollo, ya que fue necesario realizar correcciones, e implementar lógica de reconexión más robusta.

Otras bibliotecas utilizadas

- **Support Library**: Está biblioteca desarrollada por Google brinda versiones de las APIs de Android que son compatibles hacia atrás [141], lo cual permite funcionalidades y componentes de las últimas versiones del sistema operativo sin dejar de soportar versiones anteriores de Android.
- **Jackson**: Biblioteca escrita en Java para el procesamiento de JSON [142]. Se utiliza fundamentalmente para realizar el mapeo de JSON a objeto cuando se obtiene la respuesta de los servicios REST.

- **Spring for Android**: Biblioteca para Android que simplifica el desarrollo al proveer un cliente REST [143], que se encarga de serializar los objetos en JSON y viceversa de forma automática, permitiendo realizar pedidos HTTP con mucha facilidad.
- **ActionBarSherlock**: Permite utilizar el patrón de diseño ActionBar [144] que se introdujo en la versión 3.0 de Android, en versiones anteriores del sistema operativo [145].
- **Crouton**: Biblioteca que permite mostrar notificaciones sensibles al contexto [146], se utiliza en el proyecto para mostrar mensajes de validación y errores como pérdida de conexión.
- **RefreshActionItem**: Un elemento de la ActionBar que puede actuar tanto como botón de actualización como de indicador de progreso [147]. También provee soporte para emblemas. Se utiliza principalmente para los indicadores de notificaciones dentro de la aplicación, como invitaciones de punto de encuentro y solicitudes de amistad pendientes.
- **SlidingMenu**: Permite mostrar una vista que se desliza desde un costado de la pantalla [148]. Este patrón de diseño permite que se pueda acceder rápidamente a una vista específica sin sacrificar espacio en la pantalla. Se utiliza para mostrar la lista de amigos conectados, y seguimientos en progreso.

5.3.3 Desarrollo HTML5 con Sencha Touch

A la hora de realizar la implementación de la aplicación en HTML5 se evaluaron diferentes *frameworks*, quedando como principales opciones jQuery Mobile y Sencha Touch.

Se decidió implementar la aplicación utilizando Sencha Touch 2 debido a que resulta más conveniente para una aplicación compleja. Algunas de los puntos fuertes de Sencha son su arquitectura MVC, que tiene mejor performance que jQuery Mobile, que cuenta con su propio sistema de *plugins* nativos, y que está muy bien documentado.

5.3.3.1 Servicios REST

Sencha Touch provee un *plugin* para el consumo de servicios web implementado con *Ajax*, lo que permite consumir servicios sin necesidad de refrescar la interfaz gráfica. Esta abstracción provee los mecanismos para especificar la dirección del servicio, los parámetros asociados y las funciones que atenderán la respuesta del servidor.

Si bien en principio se contaba con la posibilidad de utilizar WebSQL en la aplicación HTML5 para poder sincronizar los datos con el *Servidor Central*, la falta de soporte para esta API, y lo engorroso que resultaba programar manualmente la persistencia en JavaScript, se decidió utilizar un enfoque basado en servicios REST y cachés volátiles en memoria.

Los resultados de los servicios invocados se almacenan en memoria, y el *framework* brinda diferentes funcionalidades para su posterior modificación y consulta. Estas funcionalidades mencionadas son accesibles desde cualquier parte del código, lo que facilita el uso de los datos obtenidos.

5.3.3.2 Comunicación en Tiempo Real

Una de las diferencias más grandes con la implementación nativa en Android es que Socket.IO fue pensado y desarrollado para JavaScript, tanto en el servidor como en el cliente.

Como resultado, implementar el contrato de comunicación con el *Servidor de Transmisión* resultó significativamente más sencillo en JavaScript, ya que facilita la recepción de eventos mediante la utilización de *callbacks*, y Sencha Touch provee una forma sencilla de actualizar la interfaz gráfica de forma desacoplada mediante selectores.

5.3.3.3 Algoritmo de Ómnibus más Cercano

El algoritmo de ómnibus más cercano se implementó de forma similar al de la aplicación para Android, con la única salvedad de que la API de ArcGIS para JavaScript no implementa todos los servicios que tiene el SDK de Android. Dadas estas restricciones, hubo que implementar en JavaScript aquellas funciones necesarias para el algoritmo que no estaban disponibles en la API.

5.3.3.4 Servicios Geográficos

Al igual que en el caso de la aplicación de Android, uno de los requerimientos no funcionales del proyecto es la utilización de ArcGIS para el consumo de servicios de índole geográfica, por lo que se utilizó la API de ArcGIS para JavaScript.

Si bien la API de ArcGIS para JavaScript se integra con Sencha Touch, este *framework* no provee un *wrapper* específico para esto, como sí lo hace para Google Maps. Esto dificultó la integración con la API, específicamente para el uso de los mapas base y cómo estos son desplegados por el navegador. Para solucionar este tema, se desarrolló una clase genérica que permitía utilizar los mapas base de forma sencilla instanciando la misma.

Como la API está desarrollada en JavaScript, la interacción con los servicios es buena y resulta muy sencilla. La única desventaja que se encontró está relacionada a las respuestas asíncronas de los mismos. Al no tener la posibilidad de realizar llamadas sincrónicas, resulta sumamente complejo implementar un algoritmo con llamadas sucesivas a los servicios como en el caso del *Algoritmo de Ómnibus más* Cercano, ya que en cada respuesta se debe retomar el algoritmo en el estado anterior.

El uso de mapas base en los diferentes casos de uso y el manejo de capas se implementó de forma análoga que en la aplicación para Android, por lo que no se vuelven a mencionar los detalles asociados.

5.3.4 Integración de ArcGIS en plataformas diferentes

Aunque se logró integrar ArcGIS en ambas plataformas, fue notorio que el SDK de ArcGIS para Android aún tiene que mejorar. Comparándolo con la experiencia que se obtiene con Google Maps, es más lento, pesado, y difícil de verificar. El hecho de que sea una biblioteca propietaria y que no se pueda inspeccionar directamente el código fuente para buscar las causas de las fallas, sumado a la escasez de documentación e información en línea, hace que sea mucho más difícil identificarlas. La implementación mediante Google Maps resultaba muy superior en cuanto a rendimiento, aunque no se contaba con muchos de los servicios geográficos provistos por ArcGIS, y resultaba más complicado el manejo de los marcadores.

También cabe destacar que no es sencillo integrar ArcGIS en proyectos Android que no se desarrollan con el IDE Eclipse, ya que no se proveen instructivos claros y mecanismos sencillos para incluir las bibliotecas necesarias para su uso.

En lo que respecta a la implementación en HTML5 utilizando la API de ArcGIS para JavaScript, si bien la documentación está completa y hay muchos ejemplos que pueden ajustarse a diversas realidades, fue notoria la diferencia con el SDK de ArcGIS para Android. Existen varias funcionalidades no están implementadas en JavaScript y tuvieron que buscarse alternativas que permitieran llegar al mismo resultado sin perder precisión, como en el caso del algoritmo de los ómnibus más cercanos.

Esto hace que a la hora de portar una aplicación a otra plataforma no siempre sea posible reutilizar el código relacionado con ArcGIS, ya que hay diferencias sustanciales en la API que provee según la plataforma.

5.4 Aplicación para Ómnibus

Esta aplicación tiene como fin permitir que los ómnibus transmitan su ubicación en tiempo real al *Servidor de Transmisión* para permitir su seguimiento. En la propuesta del proyecto no se tenía como objetivo desarrollar dicha aplicación, ni mucho menos incorporarla a un ómnibus real, por lo que no había requerimiento alguno sobre la misma.

Sin embargo, para poder probar el seguimiento de ómnibus de forma adecuada se decidió implementar un simulador de recorridos de ómnibus configurable, y una vez implementado se decidió agregar la posibilidad de transmitir la posición geográfica real ya que resultaba muy sencillo.

La aplicación móvil está desarrollada enteramente sobre la plataforma Android, ya que se podría reutilizar algunos de los componentes desarrollados para la *Aplicación Principal*. En particular se reutilizan los componentes que utilizan el SDK de ArcGIS, incluido el manejo de capas para el mapa.

5.4.1 Simulación

Si bien es posible utilizar la aplicación para que utilice el GPS para enviar la ubicación real del dispositivo, uno de sus componentes fundamentales es la simulación de recorridos de ómnibus.

Esto consiste en simular el movimiento del ómnibus y enviar las coordenadas geográficas simuladas al *Servidor de Transmisión* de la misma forma que en el flujo normal. La simulación de coordenadas se realiza en torno al recorrido de la línea de ómnibus elegida, avanzando con velocidad variable a través de los puntos del recorrido hasta llegar al final.

Las coordenadas geográficas se alteran levemente para no quedar encima de unos de los puntos del recorrido. Esto permite simular un ambiente más realista, y verificar que el *Algoritmo de Ómnibus más Cercano* está realizando las proyecciones al recorrido correctamente.

El simulador permite pausar y retomar recorridos, configurar la velocidad media, y realizar simulaciones de varias líneas al mismo tiempo, siendo esto último crítico para testear el seguimiento de ómnibus en la *Aplicación Principal*.

5.4.2 Comunicación en Tiempo Real

La comunicación con el *Servidor de Transmisión* se resolvió de forma similar que en la *Aplicación Principal* para Android, con la diferencia de que el contrato de eventos recibidos corresponde a las acciones relacionadas con ómnibus.

5.4.3 Persistencia

Para obtener los datos de ómnibus la aplicación utiliza los servicios web del *Servidor Central*, y al igual que la *Aplicación Principal*, los almacena de forma local para lograr una mejor performance y reducir el acceso a la red.

Es importante tener en cuenta que la base de datos del *Servidor Central* está basada en documentos, mientras que en la aplicación móvil se utiliza una base de datos relacional SQLite, por lo que es necesario realizar una conversión entre el esquema de documentos y el esquema relacional.

En particular, es necesario crear una tabla para almacenar la información de paradas de cada línea, mientras que en MongoDb esta información está embebida en cada línea y parada.

5.5 Servicios de Datos Geográficos

Utilizar mapas resulta indispensable a la hora de visualizar información geográfica, por lo que para el funcionamiento de las aplicaciones móviles es indispensable contar con un servicio que provea los mapas de base. En esta sección se describen los servicios de mapa utilizados, así como otros servicios geográficos utilizados en la implementación del sistema.

5.5.1 Mapa Base

En un principio se utilizó Google Maps para proveer los mapas de base, ya que se estaba utilizando Google Maps como biblioteca de componentes de mapas. Una vez que se comenzó a utilizar ArcGIS como biblioteca geográfica para las aplicaciones, se evaluaron diferentes alternativas para los mapas base.

En una primera instancia se utilizaron los mapas base por defecto que provee ESRI, pero el nivel de detalle presentado para algunos casos no cumplía con el mínimo requerido para la aplicación. Si bien el mapa topográfico y de calles tenía toda la información necesaria, el mapa satelital no llegaba al nivel de detalle requerido.

Se evaluaron otras alternativas para el mapa base de calles y satelital, entre las que se encontraban Google Maps y OpenStreetMaps, y finalmente se optó por utilizar Bing, que se integra fácilmente con la API de ESRI y su representación cartográfica resulta más adecuada para dispositivos móviles con pantalla de tamaño reducido.

5.5.2 ESRI ArcGIS Online Geocoding

Los servicios de geo-codificación se utilizan para transformar direcciones a coordenadas geográficas, coordenadas a direcciones, o para ubicar puntos de interés.

En la *Aplicación Principal* se utiliza el servicio cuando el usuario selecciona un lugar para un nuevo punto de encuentro, para transformar coordenadas geográficas a una dirección de forma de presentar las ubicaciones de forma agradable al usuario.

En la implementación del sistema se utiliza el ArcGIS Online Geocoding Service [149].

5.6 Extracción, Transformación, y Carga de Datos

Los datos de ómnibus utilizados por el *Servidor* Central, tales como líneas y paradas de ómnibus, se obtuvieron a partir de los datos abiertos de la Intendencia de Montevideo [150]. Para adaptarlos a los requerimientos del sistema fue necesario implementar un proceso de extracción, transformación, y carga (ETL[50]).

5.6.1 Datos Abiertos

Los datos abiertos de la Intendencia de Montevideo se rigen bajo los ocho principios para el manejo de datos abiertos en el gobierno [151]. La Intendencia de Montevideo provee un conjunto interesante de datos abiertos los cuales están disponibles en formato Shapefile, al que se anexa un archivo de texto con metadatos. En el contexto del proyecto se utilizaron los datos asociados a las paradas, las líneas de ómnibus y las rutas de cada variante de esa línea [152] [153].

5.6.2 Implementación del Proceso

Si bien los datos de la Intendencia de Montevideo fueron muy útiles, no todos los campos asociados a las paradas y las líneas eran de utilidad. Además, estos datos están publicados en la proyección SIRGAS 2000 / UTM zone 21S [154] y la proyección utilizada en el proyecto es la WGS 84 [155], ya que es utilizada por los sistema de navegación GPS. Para la transformación entre estas proyecciones se utilizó un SIG Open Source llamado Quantum GIS [156], el cual permitía exportar los datos geográficos con la proyección adecuada en formato GeoJSON.

Inicialmente se cargaron las compañías de ómnibus de Montevideo con sus respectivos nombres, a las que luego se le asociarían las diferentes líneas.

Como sólo se utilizó un subconjunto de los datos proporcionados por la Intendencia de Montevideo, se desarrolló un *parser* en el lenguaje Perl para extraer únicamente aquellos campos que eran de interés.

Una vez que se obtuvo el subconjunto de datos a utilizar, se utilizó la herramienta Kettle de la suite Pentaho [157] para realizar la extracción, transformación y carga en la base de datos. Se utilizaron cinco procesos, ya que se debía cumplir con determinadas restricciones en cuanto a la dependencia de los datos. El siguiente diagrama muestra el proceso de carga en su conjunto:

Figura 5.3: Proceso de carga de datos

[50] ETL (Extract, Transform and Load)

Algunos de estos procesos son muy escuetos, por lo que se presenta únicamente el diagrama del proceso de “Asociación entre líneas y paradas”:

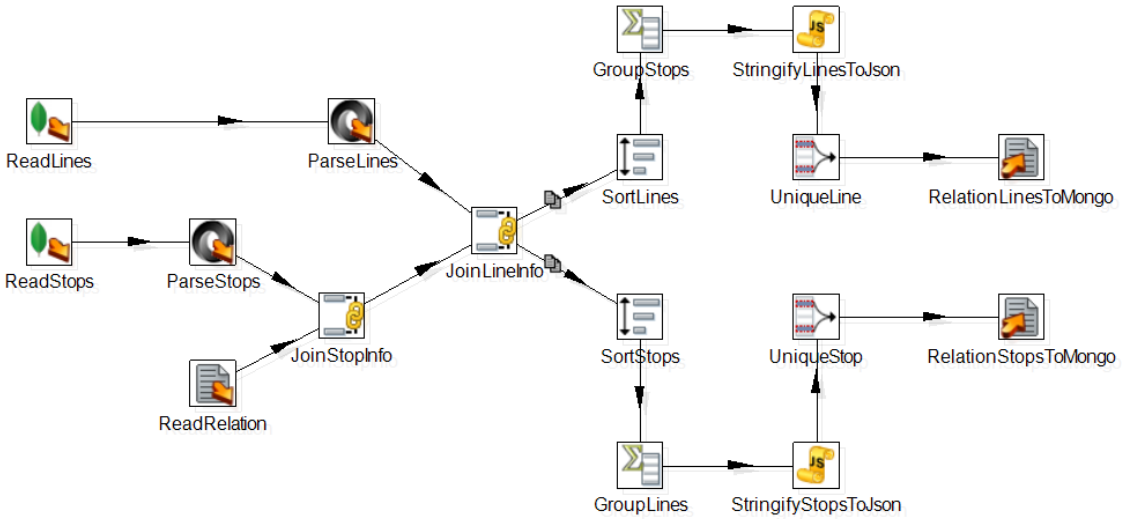


Figura 5.4: Proceso de carga de “Asociación entre líneas y paradas”

A continuación se plantea un resumen del proceso general según el orden en el que se realizó:

1. Obtención de datos de las compañías de ómnibus de Montevideo.
2. Obtención de los datos de las líneas en formato GeoJSON a partir del Shapefile de la Intendencia de Montevideo utilizando Quantum GIS.
3. Extracción de datos del GeoJSON de líneas con Perl.
4. Obtención de información de compañías de cada línea de ómnibus [170].
5. Cruzamiento de datos de números de línea y compañía con el número de variante de cada línea.
6. Carga de las compañías en la base de datos.
7. Carga de las líneas con su ruta correspondiente en la base de datos.
8. Obtención de los datos de las paradas en formato GeoJSON a partir del Shapefile de la Intendencia de Montevideo utilizando Quantum GIS.
9. Extracción de datos del GeoJSON de paradas con Perl.
10. Carga de paradas en la base de datos.
11. Obtención y carga de la relación entre líneas y paradas en la base de datos.
12. Obtención y carga de la relación entre líneas y compañías en la base de datos.

5.7 Puesta en Producción

Para poder probar el sistema en un ambiente realista donde los dispositivos móviles dispusieran de una conexión de datos inalámbrica fue necesario hacer *deploy* de los servidores. En esta sección se menciona brevemente el proceso, se mencionan las dificultades encontradas con la transmisión de datos en tiempo real, y cómo se resolvieron.

5.7.1 Servidor Central en Heroku

Para subir el *Servidor Central* se decidió utilizar Heroku [158], una solución de hosting basada en la nube que permite hacer deploy de una aplicación de forma muy sencilla que brinda soporte para aplicaciones Ruby, Node.js, entre otros lenguajes y plataformas.

MongoHQ

Debido a que Heroku no soporta MongoDB como sistema de base de datos, fue necesario configurar la instancia del servidor para que se conectara a un servidor de base de datos de MongoHQ [159], un servicio de hosting para MongoDB.

Una vez configurada la aplicación, se importaron los datos de ómnibus al servidor de MongoDB.

5.7.2 Servidor de Transmisión

En un principio se empezó a utilizar Heroku para alojar el servidor Node.js, pero al probar la *Aplicación Principal* para Android la comunicación parecía no funcionar.

Heroku y WebSockets

Al investigar el problema, se descubrió que el *stack* de Heroku no soporta el protocolo de WebSockets [160] por lo que Socket.IO intenta utilizar XHR Polling para mantener la conexión.

El problema ocurría porque la implementación utilizada como cliente de Socket.IO en Android sólo implementa el protocolo de WebSockets, y no el resto de los transportes alternativos.

Hosting Alternativo

Debido a que existe una gran diferencia entre el rendimiento del protocolo de WebSockets si se lo compara con técnicas de *polling* [161], y a que no se contaba con el tiempo que involucraría implementar el protocolo de Socket.IO mediante XHR Polling, se decidió buscar una alternativa a Heroku para alojar el *Servidor de Transmisión* que sí soportara el protocolo de WebSockets.

Debido a que no se pudo encontrar un servicio que soportara WebSockets y fuera gratuito, se decidió utilizar una máquina particular como servidor.

6 Pruebas Realizadas y Resultados Obtenidos

La etapa de verificación y validación es importante por el simple hecho de que se cometen errores en el desarrollo de software [162]. En general, estos errores dan lugar a fallas externas, lo que quiere decir que el software no hace lo requerido, o que hace algo que no debería [163].

6.1 Objetivos

Cuando hablamos de pruebas o *testing*, esto no solo implica probar que el software haga lo especificado, sino que cumpla con los requerimientos no funcionales y con las expectativas del usuario. Esto último es de vital importancia en el ecosistema móvil, ya que existe una amplia variedad de aplicaciones que cumplen con los mismos requerimientos funcionales, y las que el usuario preferirá son aquellas que le dan una experiencia más completa. Esta experiencia dependerá del diseño de interacción, fluidez de la interfaz y sensación de tiempo de respuesta.

6.2 Plan de Verificación

Si bien no se llegó a elaborar formalmente, se siguió un plan de verificación y validación en el que se identificaron los componentes críticos a ser verificados, se estudiaron las diferentes alternativas de software para realizar la verificación de cada componente, y se elaboraron las estrategias a utilizar.

Los componentes críticos que se identificaron fueron el Servidor Central, el Servidor de Transmisión y las aplicaciones móviles desarrolladas para Android y HTML5.

Una vez identificados los componentes, se realizó un estudio acerca de las herramientas de las que se dispone para realizar pruebas sobre los diferentes entornos.

En el caso del Servidor Central, dado que está desarrollado sobre el *framework* Sinatra, se optó por utilizar MiniTest [164] y FactoryGirl [165]. MiniTest es un *framework* que permite desarrollar pruebas unitarias en Ruby de forma rápida y sencilla, y FactoryGirl permite crear objetos Ruby que serán utilizados en las pruebas.

Para el Servidor de Transmisión se optó por Mocha [166], un *framework* que facilita las pruebas en entornos JavaScript, que permite probar la interacción entre los diferentes usuarios que realizan el seguimiento en tiempo real.

Para las pruebas sobre la aplicación desarrollada en HTML5 se optó por Siesta [167]. Este *framework* permite desarrollar pruebas unitarias y funcionales sobre JavaScript, incluso simulando la interacción con la interfaz gráfica.

6.3 Alternativas de Implementación

Habiendo estudiado las diferentes herramientas, al comenzar el desarrollo se evaluaron las ventajas y desventajas de implementar pruebas automáticas frente a la verificación manual, sobre todo teniendo en cuenta el tiempo que insume cada opción.

La verificación manual tiene la ventaja de adaptarse al cambio, permite detectar varios problemas a la vez, introduciendo variantes de uso que no se tendrían por qué dar de otra forma. Las pruebas automáticas son más rigurosas y específicas, permiten encontrar errores

que se introdujeron por interacción con nuevos componentes y en general, cubren una gran cantidad de pruebas en un tiempo mucho menor; sin embargo, este tipo de pruebas tienen un costo asociado al aprendizaje de la herramienta, el diseño e implementación de los casos de prueba, y el mantenimiento y actualización de los mismos en caso de cambios en los requerimientos. [168]

Como se mencionó anteriormente, la automatización de los casos de pruebas tiene un costo asociado que se recupera luego de la ejecución de varios ciclos de prueba. En el caso del proyecto, algunas funcionalidades no iban a cumplir con la cantidad de ciclos de prueba necesarios para recuperar la inversión de tiempo, por lo que directamente se optó por realizar las pruebas de forma manual.

6.4 Implementación de la Verificación

A continuación se presenta la implementación de la verificación en los diferentes componentes identificados como críticos.

6.4.1 Servidor Central

En el caso del Servidor Central se desarrollaron pruebas automáticas para verificar el comportamiento de los servicios asociados a la búsqueda y ubicación de amigos, el historial de búsqueda de amigos, el registro de nuevos usuarios y la obtención de las paradas favoritas (por más información, ver Anexo II: Análisis de Requerimientos). En total se corrieron dieciocho pruebas unitarias, las cuales sirvieron para detectar errores en la forma en la que se devolvía la información y en los controles asociados a la lógica de la aplicación.

Contar con este tipo de pruebas permitió corroborar el buen funcionamiento de los servicios mencionados a lo largo de la fase de implementación. Luego de agregar nuevas funcionalidades o hacer cambios en la lógica del Servidor Central, se corrían nuevamente los casos de prueba y se podía tener un primer *feedback* de los posibles errores introducidos en la integración.

Dado que el Servidor Central ofrece una API basada en REST, se realizaron múltiples pruebas de interacción utilizando la consola JavaScript de los navegadores web. Mediante llamadas a los servicios podía observarse la respuesta en la interfaz provista por los navegadores, lo que fue de gran utilidad a la hora de verificar el comportamiento de los servicios con diferentes parámetros.

6.4.2 Servidor de Transmisión

Las funcionalidades del Servidor de Transmisión se probaron de forma manual, utilizando directamente la aplicación, por lo que no se llegaron a desarrollar pruebas en Mocha.

Se realizaron pruebas relacionadas a la conexión y desconexión de amigos, y seguimiento en tiempo real con amigos y ómnibus. Para los casos que se presentan a continuación, se utilizaron las aplicaciones de Android y HTML5 conectadas a un mismo servidor, con las que se iniciaba sesión con usuarios diferentes que eran amigos.

En el caso de la conexión y desconexión de amigos, la aplicación mantiene informado al usuario acerca de los amigos que se conectan y desconectan, además de mostrar una lista con el estado actual de cada uno. Visto del punto de vista del usuario que se conecta o desconecta,

el Servidor de Transmisión debe ser capaz de notificar a sus amigos que éste inició o cerró sesión en el sistema.

Utilizando ambas aplicaciones en diferentes dispositivos, se desconectaba a uno de los amigos y se verificaba el cambio de estado en cada una de las aplicaciones que estaban utilizando sus amigos. Luego volvía a iniciar la sesión y se probaba desconectar a otro usuario utilizando otro método (forzando el cierre de la aplicación, cambiando de página en el navegador, etc.).

Para el caso del seguimiento en tiempo real de amigos, se realizaba la invitación desde uno de los dispositivos, en donde se probó cancelar la invitación, desconectarse antes de que se aceptara la misma y finalmente, aceptar la invitación (como pasaría en el caso normal). Luego de comenzar el seguimiento, se utilizó un *plugin* del navegador Google Chrome que permite modificar de forma manual la posición geográfica del usuario, emulando el movimiento del mismo.

Otra de las pruebas realizadas fue la de seguimiento mutuo, en donde dos usuarios amigos realizaban la invitación de seguimiento y actualizaban sus posiciones geográficas, debiendo actualizar su posición y la de su amigo en cada dispositivo. Finalmente, se probó el desconectar al amigo que se está siguiendo, lo que permitió verificar la robustez del Servidor de Transmisión.

Tanto para el Servidor Central, como para el Servidor de Transmisión se utilizaron logs que se mostraban en la consola del servidor cada vez que se invocaba un servicio. Esto ayudó a determinar la fuente de algunos errores, que a priori no podía saberse si era problema de la invocación del servicio, o de un problema interno de la lógica del servidor. En el caso particular del Servidor de Transmisión ayudaba a visualizar en tiempo real lo que estaba sucediendo, y eso permitió que se detectaran problemas en las aplicaciones móviles, que en un principio tenían errores en la comunicación con este servidor.

6.4.3 Aplicaciones Móviles

Para las pruebas en las aplicaciones móviles desarrolladas en Sencha y Android, se utilizó una estrategia similar a la del Servidor de Transmisión, ya que dada la cantidad de iteraciones que se iban a realizar sobre el producto y el costo asociado a desarrollar las pruebas, se prefirió realizar las pruebas de forma manual.

Al utilizarse una metodología iterativa e incremental, una vez que se completaba un ciclo y se agregaba un paquete de nuevas funcionalidades, se hacían pruebas de humo para verificar el buen funcionamiento de lo que ya se había implementado, y se procedían con las pruebas específicas para esas funcionalidades.

En forma conjunta se definían algunos casos de prueba básicos que se ejecutaban directamente utilizando la aplicación, y no sólo se verificaba que se cumpliera con la especificación del requerimiento implementado, sino que se evaluaba la interacción y la forma en la que se presentaban los resultados, teniendo en cuenta las limitaciones del tamaño de la pantalla.

Uno de los casos evaluados en relación a las limitaciones del tamaño de pantalla, fue la visibilidad de los objetos del mapa a diferentes escalas. Dado que en el mapa no solo se muestra la posición actual del usuario, sino que también están las paradas, los amigos y los

ómnibus a los que se está siguiendo, es necesario determinar cuál es la mejor forma de desplegar toda esta información y que el usuario pueda interpretar todo lo que se ve en el mapa. A raíz de algunas pruebas se determinó la escala a la que era visible cada capa, y se agregaron mecanismos para mostrar u ocultar algunas de ellas.

Como ya se mencionó anteriormente, se hizo uso de la consola de JavaScript de los navegadores para hacer pruebas sobre diferentes partes de la aplicación. En el caso de la aplicación móvil desarrollada en Sencha se utilizó para hacer *debugging* de las clases JavaScript, pudiendo acceder a las funcionalidades del *framework* usando la clase Ext, lo que permitía probar los métodos que daban errores, facilitando la corrección de los mismos.

6.4.4 Bases de Datos

Algo que no se comentó hasta el momento, fueron las pruebas sobre los datos cargados, y los generados por la aplicación. Si bien no se implementaron pruebas estrictas sobre los datos, sí se verificó que los datos ingresados por la aplicación fueran consistentes y correspondieran con lo que se estaba realizando en ese momento. En paralelo con las pruebas sobre la aplicación, se utilizó la consola de MongoDb para correr consultas que permitieran verificar los datos guardados. También se utilizaron consultas para verificar que los datos de las paradas y recorridos obtenidos de los datos abiertos de la Intendencia de Montevideo se correspondieran con lo que se esperaba cargar.

7 Conclusiones

El proyecto surge como la combinación de tres puntos clave: evaluar las opciones de desarrollo de aplicaciones móviles, investigar la comunicación en tiempo real entre dispositivos móviles, y plasmar este aprendizaje en el desarrollo de un sistema con características de red social y funcionalidad geográfica, que esté optimizado en torno al transporte público y permita realizar seguimiento en tiempo real de usuarios y ómnibus.

Respecto al primer punto, resultaban de particular interés aquellas alternativas que permiten desarrollar para múltiples plataformas, por lo que se planteó como objetivo investigar GeneXus y HTML5 como opciones de desarrollo multiplataforma, evaluando sus virtudes y defectos en base a la experiencia de usuario del producto obtenido.

Como segundo objetivo, se decidió realizar una investigación de los protocolos y tecnologías de comunicación en tiempo real que se adecuaban a las redes móviles actuales, teniendo en cuenta el soporte existente de dichas tecnologías en las diferentes plataformas móviles.

Por último, una vez evaluadas las alternativas de desarrollo móvil y los métodos de comunicación disponibles, se comenzó el desarrollo del sistema, con la finalidad de que el mismo estuviera disponible para las plataformas móviles que cuentan con mayor presencia en el mercado.

7.1 Opciones de Desarrollo para Aplicaciones Móviles

Al investigar la herramienta GeneXus X Evolution 2 para el desarrollo móvil, se comprobó que no era posible utilizarla para realizar una implementación genérica para todas las plataformas. Debido a que para desarrollar la aplicación era necesario escribir e integrar componentes nativos para cada plataforma, los beneficios de reutilización que proveía GeneXus se veían reducidos, por lo cual se la descartó como opción de desarrollo para las aplicaciones nativas.

Una vez descartado GeneXus como herramienta de desarrollo, la investigación se concentró en las herramientas que potenciaban el desarrollo de aplicaciones web para móviles a través de HTML5, entre las que se destacaban Sencha Touch 2 y jQuery Mobile. Finalmente, se escogió a Sencha Touch para desarrollar una aplicación multiplataforma que pudiera ser utilizada por la mayoría de los navegadores móviles.

Adicionalmente, se decidió que sería interesante desarrollar aplicaciones para Android e iOS de forma nativa, lo cual permitiría compararlas con la aplicación desarrollada en HTML5. Debido a que para desarrollar en iOS es necesario contar con hardware de Apple del cual no se disponía, se decidió no implementar una aplicación nativa para iOS, pero sí llevar a cabo el desarrollo de una aplicación nativa para Android.

7.2 Comunicación en Tiempo Real

La investigación de tecnologías disponibles para realizar comunicación en tiempo real punto a punto entre dispositivos móviles resultó ser un desafío, dado que aún no existen estándares multiplataforma. Se escogió Socket.IO como base del mecanismo de transmisión en tiempo real, ya que existen bibliotecas cliente para una gran variedad de plataformas.

Uno de los contratiempos que se sufrió durante la etapa de integración entre las aplicaciones móviles y el *Servidor de Transmisión*, fue que la biblioteca de Socket.IO para Android estaba inmadura, por lo que fue necesario trabajar en la misma para mejorar su estabilidad, lográndose así cumplir con los requerimientos de tiempo de respuesta y robustez esperados.

7.3 Desarrollo del Sistema

Una vez finalizada la etapa de evaluación de las tecnologías, se comenzó el desarrollo del sistema. Los componentes se desarrollaron de forma paralela e incremental, esto es, se comenzaban a desarrollar ciertas funcionalidades en ambas aplicaciones móviles a medida que se implementaban sus contrapartes en los servidores.

Una de las mayores dificultades del desarrollo para Android fue lograr que la interfaz de usuario presentara de forma instantánea y consistente las actualizaciones geográficas, la conexión y desconexión de los usuarios, y las solicitudes de seguimiento. Esto se resolvió mediante un diseño orientado a eventos, utilizando técnicas de suscripción y notificación dentro de la aplicación. La aplicación cumple con las convenciones típicas de la plataforma, logrando una excelente usabilidad.

Por otro lado, un contratiempo al cual hubo que enfrentarse durante el desarrollo de la aplicación Sencha fue la integración con ArcGIS for Javascript, ya que esta biblioteca no está soportada por defecto. Fue necesario crear un componente Sencha que permitiera utilizar las funcionalidades de ArcGIS, lo cual fue un proceso complicado y plagado de errores. Cabe destacar que en una aplicación de gran tamaño como esta resulta sumamente complejo solucionar errores de JavaScript.

7.4 Evaluación de HTML5

Habiendo desarrollado la misma aplicación tanto de forma nativa para Android, como para HTML5 a través de Sencha, se pudo comprobar la dificultad de diseñar una interfaz que se adapte a la interacción esperada por los usuarios de diferentes plataformas. Esto se debe a que cada plataforma tiene sus convenciones de usabilidad, las cuales determinan desde la ubicación de las acciones, hasta los patrones de navegación.

Por este motivo, dado que las aplicaciones implementadas con Sencha tienden a seguir las convenciones de iOS, para los usuarios de la plataforma Android la aplicación HTML5 resulta menos intuitiva que la aplicación nativa. Como resultado de este fenómeno, no es posible que una única aplicación HTML5 multiplataforma iguale la experiencia de usuario de una aplicación nativa en cada una de las plataformas objetivo.

Si bien la funcionalidad de seguir ómnibus en tiempo real se comporta de forma muy similar en ambas aplicaciones implementadas, la funcionalidad de seguir usuarios se ve muy limitada en la aplicación HTML5 frente a la nativa, ya que la carencia de servicios en *background* de las aplicaciones web no permite que la aplicación se mantenga conectada y continúe enviando la posición geográfica del usuario cuando se encuentra en segundo plano.

Esto causa un gran impacto en la utilidad de la aplicación, ya que para poder establecer un seguimiento entre usuarios utilizando la aplicación HTML5 es necesario que ambos mantengan

la aplicación abierta durante todo el proceso, donde se solicita permiso, se confirma, y se comienza a enviar la ubicación.

En cambio, los usuarios de la aplicación Android reciben una notificación cuando un amigo les solicita permiso, la cual pueden aceptar y comenzar a enviar la ubicación, sin importar si tienen la aplicación abierta o no, lo cual resulta mucho más simple y flexible.

Como punto final de la evaluación, se concluye que la conveniencia de desarrollar en HTML5 depende fuertemente del tipo de aplicación que se desee implementar, el acceso a las características físicas del dispositivo que requieran sus funcionalidades, y la cantidad de plataformas que se quieran cubrir con una misma aplicación.

Si bien el estándar HTML5 y las tecnologías asociadas al mismo continúan mejorando, aún tienen un largo camino por recorrer antes de poder igualar lo que es posible obtener a través del desarrollo nativo en la actualidad.

7.5 Resultados Obtenidos

En definitiva, se logró construir una plataforma de comunicación en tiempo real entre dispositivos móviles, que permite la interacción de múltiples dispositivos con diferentes sistemas operativos. Queda pendiente verificar el desempeño del sistema en el caso que la cantidad de dispositivos creciera de forma significativa, aunque el uso de la plataforma Heroku y el almacenamiento de la información del *Servidor de Transmisión* utilizando Redis permiten escalar horizontalmente las instancias del servidor de transmisión sin mayores dificultades.

A su vez, se desarrolló un sistema sobre la plataforma que permite realizar seguimiento de ómnibus y usuarios en tiempo real. Cabe destacar que se utiliza información real de ómnibus extraída a partir de los datos abiertos de la Intendencia de Montevideo.

Se desarrollaron tres aplicaciones móviles como parte de este sistema: una aplicación para ómnibus que permite enviar la información asociada a su posición geográfica a la plataforma (real o simulada), y dos aplicaciones para realizar seguimiento de usuarios y ómnibus, una nativa para Android y otra desarrollada a través de HTML5.

La aplicación HTML5 obtenida puede ser utilizada en iOS, Android, Windows Phone 8, y BlackBerry 10, por lo que se logró una vasta cobertura de plataformas. Es importante destacar que como cualquier aplicación web, es necesario que la aplicación esté en primer plano para que siga recibiendo o enviando información, lo cual limita seriamente la utilidad de la aplicación.

Resulta interesante el hecho de que se utilizaron tecnologías de última generación, lográndose una buena integración de los componentes desarrollados a pesar de la diversidad de las plataformas involucradas. El sistema desarrollado brinda funcionalidades que involucran aspectos sociales y cotidianos, y son aptas para el público masivo, y presenta una arquitectura escalable, por lo cual es viable su incorporación en un ambiente de producción.

8 Trabajo Futuro

Si bien los objetivos del proyecto fueron alcanzados en forma satisfactoria, se presentan ciertas oportunidades de trabajo a futuro, como llevar el sistema a producción, evaluar otras bibliotecas, realizar pruebas adicionales, mejorar las funcionalidades existentes, y desarrollar otras versiones nativas de la aplicación.

8.1 Integración con el Sistema de Transporte Metropolitano

Uno de los trabajos a futuro más interesantes es el de llevar el proyecto a producción. Sería importante contactarse con las diferentes empresas que brindan servicios de transporte público, evaluar si la plataforma cumple con los requerimientos necesarios para su uso en el Sistema de Transporte Metropolitano (STM) y ver cómo podría integrarse con los sistemas que están actualmente en uso, con el fin de detectar las oportunidades de negocio que existen.

De llevar a cabo esta iniciativa sería clave investigar el hardware del que disponen los ómnibus del STM actualmente, de modo de poder evaluar las características de procesamiento, los sensores disponibles como GPS, y el acceso a redes inalámbricas. De esta forma, se podría reutilizar la infraestructura existente y reducir los costos de integración con el sistema.

8.2 Desarrollo Nativo en otras Plataformas

Como se mencionó en el capítulo de implementación, se descartó el desarrollo en la plataforma iOS por las restricciones que ésta impone, y tampoco se tuvo en consideración el sistema operativo Windows Phone. Al extender el desarrollo nativo a estas plataformas se lograría brindar una experiencia de usuario superior, y permitiría que la aplicación esté disponible para un público más amplio.

8.3 Pruebas Automatizadas en el Servidor de Transmisión

Como se comentó en el capítulo de pruebas realizadas, se elaboraron pruebas de regresión en el Servidor Central, pero no así en el Servidor de Transmisión. Si bien se probaron las funcionalidades brindadas por éste mediante el uso de la aplicación, no se desarrollaron pruebas automatizadas. Esto podría hacerse utilizando Mocha [169], un *framework* que permite realizar pruebas en entornos asíncronos, como lo es NodeJS. Tener pruebas de integración para la comunicación con el Servidor de Transmisión permitiría continuar el desarrollo sin comprometer la funcionalidad existente.

8.4 Portal Web

Muchas aplicaciones para dispositivos móviles actuales cuentan con sitios web en donde se proveen extensiones a funcionalidades que por las limitaciones de autonomía, capacidad de procesamiento o tamaño de pantalla no se implementan en la aplicación móvil. En este caso, podría mostrarse toda la información asociada a las citas, amigos y paradas, y como extensiones, desplegar las rutas de los recorridos efectuados por el usuario mientras utilizaba la aplicación, mostrar un historial detallado de los seguimientos de amigos y ómnibus, y estadísticas de seguimiento (cuántos usuarios siguió, cuántas veces fue seguido, líneas más seguidas, gráficos, entre otros).

8.5 Alternativas de Implementación

Si bien se utilizaron los servicios de ArcGIS ya existentes brindados por las diferentes APIs, podría implementarse un servicio en ArcGIS Server para la obtención de los ómnibus más cercanos a una parada. Actualmente el procesamiento que realiza este algoritmo se realiza en el dispositivo, por lo cual sería interesante comparar el tiempo de respuesta de este enfoque con la alternativa de tener un servicio web que lo calcule en un servidor.

8.6 Extensión de las Bibliotecas Utilizadas

Una vez adquirida la experiencia con las bibliotecas utilizadas se vio la oportunidad de extenderlas, con el fin de facilitar las tareas de desarrollo futuro, o mejorar su funcionalidad.

8.6.1 Componente ArcGIS Maps para Sencha Touch

Como se mencionó en el capítulo del Estado del Arte, Sencha Touch provee un componente que facilita el uso de los mapas de Google, pero no existe uno similar para ArcGIS. Podría desarrollarse un componente que encapsule el uso de mapas y funcionalidades brindadas por la API de JavaScript de ArcGIS, lo cual facilitaría la incorporación de los mismos en diferentes proyectos.

8.6.2 Transporte alternativo para Android

La biblioteca utilizada para el uso de *WebSockets* en Android funciona únicamente cuando el entorno en el que ejecuta el Servidor de Transmisión soporta el protocolo de *WebSockets*. Como una alternativa para que la comunicación con el Servidor de Transmisión sea más robusta, podría extenderse el cliente de Socket.IO para Android utilizando *XHR Polling* como transporte, con lo que se podría establecer una conexión con el Servidor de Transmisión en caso de que no se pudiera utilizar *WebSockets*.

8.7 Extensión de Funcionalidades

Si bien las aplicaciones desarrolladas cuentan con un buen conjunto de funcionalidades, sería posible mejorar la usabilidad incorporando algunas de las funcionalidades que quedaron fuera del alcance.

En lo que respecta a las citas, podrían agregarse datos multimedia a la cita, como fotos, videos o audio, que ayudarían a ubicar el lugar o dar una idea más clara del punto de encuentro. Dado que en la aplicación web no se cuenta con la funcionalidad de agregar las citas a la agenda, podría desarrollarse un módulo nativo de notificaciones en las que se daría aviso al usuario de todos los eventos asociados al seguimiento de dispositivos y puntos de encuentro.

Para el módulo de ómnibus resultaría sumamente útil contar con el tiempo estimado de arribo del ómnibus a la parada, de modo que el usuario pueda disponer de toda la información a la hora de decidir entre varias líneas de ómnibus.

8.7.1 Realidad Aumentada

Si bien el uso de esta tecnología en aplicaciones móviles es reciente, brinda un gran valor agregado para ciertas aplicaciones geográficas. Antes de comenzar el proyecto se consideró la

posibilidad de incluir un componente de Realidad Aumentada, aunque luego se descartó la idea ya que se desviaba de los objetivos principales.

Utilizando la posición del usuario y los datos asociados a las paradas, podría proveerse información de utilidad simplemente "apuntando" con la cámara del dispositivo hacia la parada en donde se quiere tomar el ómnibus, u obtener información de una línea que está circulando.

8.7.2 Chat entre Usuarios

La plataforma y las tecnologías utilizadas para la implementación de los servicios y aplicaciones, hacen que la incorporación de un chat entre usuarios sea muy sencilla. Sería de utilidad a la hora de establecer un punto de encuentro, o comunicarse con amigos a los que se está siguiendo, y brindaría una experiencia más completa.

8.7.3 Integración con Redes Sociales

Dadas las características de red social de la aplicación, sería interesante integrarla con redes sociales como Facebook o Twitter, por ejemplo, a la hora de agregar amigos.

Como una alternativa a la agenda del dispositivo móvil, el usuario podría agendar sus puntos de encuentro en Facebook utilizando Eventos, lo cual le permitiría compartir el punto de encuentro con amigos que no están registrados en el sistema.

En el caso de Twitter, resultaría interesante permitir la publicación de Tweets o fotos asociadas a la posición actual del usuario o un punto de encuentro, proveer mensajes predeterminados que mostraran que se está utilizando alguna funcionalidad, y el uso de *hashtags* para lugares o personas.

9 Glosario

ABM	Acrónimo utilizado para referirse a las operaciones de alta, baja y modificación de datos o entidades de un sistema de información.
API	Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
ARM	Arquitectura de procesadores RISC utilizada en microprocesadores y microcontroladores de bajo consumo, ampliamente utilizados en telefonía móvil.
AVL	Acrónimo de Automatic Vehicle Location (Rastreo Vehicular Automatizado en español), que refiere a los sistemas de localización remota en tiempo real, utilizando una combinación de dispositivos de localización (GPS, GLONASS) con dispositivos de transmisión inalámbrica de datos (módems).
Background	Término utilizado para referirse al modo en que se ejecutan determinados servicios u operaciones para los que el usuario no percibe su ejecución, creados con el objetivo de que la usabilidad de la aplicación no se vea afectada.
BOSH	Protocolo de transporte que emula la comunicación bidireccional entre dos sistemas utilizando el protocolo HTTP. Hace un uso más eficiente del ancho de banda y mejora el tiempo de respuesta en aplicaciones que utilizan las operaciones pull y push.
BSON	Formato de intercambio de datos serializados en codificación binaria, utilizado para transferir y almacenar documentos tipo JSON.
Callback	Código ejecutable pasado como parámetro en la invocación de una función, el cual será ejecutado (de forma sincrónica o asincrónica) como respuesta de la misma.
CDMA	Refiere a los métodos de multiplexación o control de acceso al medio utilizados en algunas redes de telefonía celular, para que varios usuarios puedan transmitir información por un mismo canal, sin interferir entre ellos.
CPU	Acrónimo de Central Processing Unit (Unidad Central de Procesamiento). Es el componente encargado de ejecutar las instrucciones de máquina.
CSS	Lenguaje de hojas de estilo utilizadas para definir la presentación semántica de un documento escrito en lenguaje de marcas. Comúnmente utilizadas para dar estilo a páginas web escritas en lenguaje HTML.
¿Cuándo Llega?	Agrupación de funcionalidades de la aplicación relacionadas al seguimiento en tiempo real de ómnibus.

DDL	Lenguaje proporcionado por un manejador de base de datos que permite definir las estructuras en donde serán almacenados los datos, así como las funciones o procedimientos que permitan consultarlos.
Debug	Término utilizado para referirse al proceso en el que se realizan pruebas sobre una pieza de software o hardware, con el objetivo de encontrar defectos que afecten el correcto funcionamiento del mismo.
Deploy	Conjunto de actividades que permiten que un sistema de software quede disponible para ser utilizado.
DML	Lenguaje proporcionado por un manejador de base de datos que permite que los usuarios de las bases de datos lleven a cabo consultas o modificaciones sobre los datos contenidos en las mismas.
DSL	Lenguaje de programación especificado para resolver problemas de un dominio particular, representar un problema específico y proveer técnicas para determinar la solución del mismo.
EGNOS	Sistema de corrección de señales desarrollado por la Agencia Espacial Europea, la Comisión Europea y Eurocontrol. Desarrollado con el fin de complementar las redes GPS y GLONASS para mejorar la precisión y seguridad de señales, permitiendo una precisión inferior a dos metros.
Escalabilidad	Propiedad de un sistema que indica su habilidad para reaccionar y adaptarse al crecimiento continuo de la carga de trabajo, o que está preparado para soportar volúmenes mayores de trabajo sin perder la calidad del servicio ofrecido.
ETL	Acrónimo para referirse al proceso de extracción, transformación y carga de datos. Los datos son extraídos de una o varias fuentes de datos, pasan por un proceso de transformación en el que se pueden o no realizar cambios, para finalmente ser cargados en una nueva fuente de datos.
FindMe!	Agrupación de funcionalidades de la aplicación relacionadas a la red social de usuarios y seguimiento en tiempo real de amigos.
Flash Sockets	Implementación de Sockets de Flash, permite establecer conexiones TCP para enviar y recibir datos binarios.
Framework	Estructura conceptual y tecnológica de soporte definido, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.
Garbage Collector	Mecanismo de manejo automático de la memoria, intenta liberar memoria que ya no está siendo utilizada por los objetos de un programa.

Geolocalización	Posicionamiento con el que se define la localización de un objeto espacial (representado mediante punto, vector, área, volumen) en un sistema de coordenadas y datum determinado. Este proceso es utilizado frecuentemente en los SIG.
GLONASS	Constelación de 31 satélites desarrollada por la Unión Soviética, y siendo administrada actualmente por la Federación Rusa, representa la contrapartida al GPS estadounidense.
GNSS	Constelación de satélites que transmite rangos de señales utilizados para el posicionamiento y localización en cualquier parte del globo terrestre, ya sea en tierra, mar o aire. Estos permiten determinar las coordenadas geográficas y la altitud de un punto dado como resultado de la recepción de señales provenientes de constelaciones de satélites artificiales de la Tierra para diversos fines.
GPS	Sistema de satélites perteneciente al Departamento de Defensa de los Estados Unidos, permite triangular la ubicación de un objeto en la tierra, como una persona o un vehículo con una precisión en el ámbito de los metros.
GPU	Procesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos y o aplicaciones 3D interactivas.
GSM	Sistema de telefonía móvil digital estándar, y libre de regalías, de segunda generación.
Heap	Estructura de datos del tipo árbol con información perteneciente a un conjunto ordenado.
Hosting	Servicio que provee a los usuarios de Internet un sistema para poder almacenar información, imágenes, vídeo, o cualquier contenido accesible vía web. Se utiliza también para referirse al alojamiento del código y ejecución de un servidor.
HSPA	Combinación de tecnologías posteriores y complementarias a la tercera generación de telefonía móvil (3G).
HTML	Es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.
HTTP	Protocolo de aplicación para sistemas de información distribuidos, colaborativos, o de hypermedia. Es la base de la comunicación de datos de la World Wide Web.
JSON	Formato ligero para el intercambio de datos, es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

JSONP	Técnica de comunicación utilizada por los programas JavaScript que ejecutan en los navegadores web, permite realizar peticiones a un servidor que se encuentra en un dominio diferente que la aplicación.
JVM	Máquina virtual de proceso nativo, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial que es generado por el compilador del lenguaje Java.
Location Based System (LBS)	Sistemas que utilizan servicios de software basados en información de ubicación geográfica y tiempo, para brindar funcionalidades que sean relevantes al contexto en el que se utiliza el sistema.
Log	Registro oficial de eventos durante un rango de tiempo en particular, es utilizado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre para un dispositivo en particular o aplicación.
MapReduce	Programa que está compuesto de una operación (map) que realiza filtrado y ordenamiento, y otra operación (reduce) que realiza una operación de agregación.
Model-View-Controller-Store (MVCS)	Paradigma de arquitectura de software, que parte del patrón MVC, e intenta minimizar el código redundante y simplificar la forma en que se reciben y almacenan los datos, al encapsular esto último en un objeto llamado Store.
MOM	Infraestructura de software o hardware que provee comunicación distribuida en base a un modelo de interacción asíncrona.
MQTT	Protocolo de mensajería simple y liviano que ofrece funcionalidad de publicar/suscribir y ha sido diseñado específicamente para dispositivos con restricciones energéticas y redes poco confiables, con alta latencia, o ancho de banda mínimo
.NET Framework	Plataforma de desarrollo de software de Microsoft, con énfasis en transparencia de redes e independencia de plataforma de hardware, que permite un rápido desarrollo de aplicaciones.
NoSQL	Bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales en aspectos importantes, por ejemplo, no utilizar SQL como el principal lenguaje de consultas.
Object-Document Mapper (ODM)	Herramienta que mapea los documentos de una base de datos orientada a documentos, a objetos. También se encarga de abstraer las operaciones comunes como consultas, actualizaciones y borrado.
Open Source	Software distribuido y desarrollado libremente. El movimiento open-source se concentra en los beneficios prácticos, como acceso al código fuente, así como en cuestiones éticas o de libertad.

Parser	Programa cuya finalidad es analizar una cadena de caracteres, de acuerdo a las reglas de una gramática formal.
Pinch	Evento capturado cuando el usuario “pellizca” la superficie sensible al tacto.
Pixelado	Efecto causado por visualizar una imagen a un tamaño en el que los pixels individuales son visibles al ojo.
Plugin	Componente de software que agrega una funcionalidad específica a una aplicación de software existente.
Polling	Operación de consulta constante, por ejemplo, con el objetivo de presentar información actualizada en una interfaz web.
Polyfill	Biblioteca que imita una API existente, y cuenta con diferentes implementaciones para brindar la misma funcionalidad cuando no es posible utilizar alguna de ellas.
Progressive Disclosure	Técnica de diseño de interacción que permite retener la atención del usuario al reducir la confusión y la carga cognitiva de la tarea a realizar.
Push	Estilo de comunicación basado en Internet donde la petición para una transacción dada es iniciada por el servidor, en contraste con el mecanismo habitual donde la petición es realizada por el cliente.
RAM	Medio de almacenamiento temporal donde se cargan todas las instrucciones que ejecutan el procesador y otras unidades de cómputo.
RDBMS	Manejador de Bases de Datos Relacionales. Es un conjunto de programas que proveen los mecanismos necesarios para almacenar, modificar y extraer datos de bases de datos relacionales.
Renderización	Proceso de generar una imagen o vídeo a partir de un modelo mediante un programa de computadora.
REST	El término se aplica en la actualidad a cualquier interfaz web simple que utiliza HTTP, con un protocolo cliente/servidor sin estado y operaciones bien definidas, donde cada recurso puede ser accedido utilizando un identificador global único.
ROM	Medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite sólo la lectura de la información y no su escritura, independientemente de la presencia o no de una fuente de energía.
Sans-serif	También llamadas fuentes de palo seco o sin remates, son fuentes que en cada carácter no tienen unas pequeñas terminaciones llamadas remates o serifas.
Scripting (lenguaje)	Lenguaje de programación que soporta la escritura de scripts, los cuales son escritos para un entorno particular y son utilizados para automatizar determinadas tareas.

SDK	Conjunto de herramientas de desarrollo que permite la creación de aplicaciones para un cierto paquete o plataforma de software, plataforma de hardware, sistema de computadora, sistema operativo, o plataforma de desarrollo similar.
Sharding	Particionamiento horizontal de una base de datos, por ejemplo, donde las filas de una misma tabla se almacenan de forma separada.
SIG	Tipo especial de sistemas de información utilizado para administrar, analizar y desplegar información geográfica. La información geográfica es representada mediante estructuras simples que modelan la geografía y el SIG provee las herramientas necesarias para manipular dicha información.
SIM	Tarjeta inteligente desmontable usada en teléfonos móviles y módems HSPA o LTE, almacena de forma segura la clave de servicio del suscriptor usada para identificarse ante la red.
Smartphone	Teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos, y que posibilita realizar actividades semejantes a una minicomputadora, brindando mayor conectividad que un teléfono móvil convencional.
SoC (System on a Chip)	Tecnologías de fabricación que integran todos o gran parte de los módulos componentes de una computadora, o cualquier otro sistema informático o electrónico, en un único circuito integrado o chip.
SQL	Lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas, como consultas y actualizaciones.
Stack (estructura de datos)	Lista ordenada o estructura de datos en la que el modo de acceso a sus elementos es de tipo LIFO (último en entrar, primero en salir) que permite almacenar y recuperar datos.
Stack (software)	Ambiente completo de deployment, incluyendo el sistema operativo, el ambiente de ejecución del lenguaje, y las bibliotecas asociadas.
Streaming	Proceso de enviar los datos de forma continua desde el servidor al cliente, a través de una conexión persistente.
Swipe	Evento capturado cuando se hace un tap, para luego arrastrar el objeto con el cual se establece el contacto con la superficie sensible al tacto.
Tap	Evento capturado cuando se hace contacto en un único punto entre el usuario y el dispositivo con interfaz sensible al tacto.
TCP	Protocolo de comunicación de transmisión fiable orientado a conexión, opera al nivel de capa de transporte.
Timestamp	Identificador basado en el tiempo, puede ser la fecha de creación o actualización de una entidad.

Token	Credencial o llave que se utiliza para dar autorización en un sistema informático.
Viaje	Ómnibus que está realizando el recorrido de una línea particular en un instante dado.
WebSockets (WSS)	Tecnología que proporciona un canal de comunicación bidireccional sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede ser por cualquier aplicación cliente/servidor.
Wide Area Augmentation System (WAAS)	Sistema de satélites y estaciones base que proveen correcciones de la señal de GPS, brindando mayor precisión en las ubicaciones.
Wireframe	Representación esquemática de una aplicación, es una guía visual básica que intenta sugerir la ubicación de los elementos de diseño clave en la interfaz gráfica. Sirven como herramienta de comunicación y discusión entre desarrolladores, diseñadores y clientes.
Workflow	Flujo de trabajo de una aplicación, define cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas.
Wrapper	Es un programa que hace posible que se pueda ejecutar otro programa, más importante, en un entorno en el que no podría ejecutar.
x86	Conjunto de instrucciones utilizada en la microarquitectura de CPU, siendo también una denominación genérica dada a ciertos microprocesadores.
XMLHttpRequest (XHR)	Interfaz empleada para realizar peticiones HTTP a servidores Web, se utiliza comúnmente para proporcionar contenido dinámico y actualizaciones asíncronas en páginas Web a través de tecnologías como AJAX.

10 Bibliografía

[1] Gartner. (2012, Noviembre) Gartner Says 821 Million Smart Devices Will Be Purchased Worldwide in 2012; Sales to Rise to 1.2 Billion in 2013. [Online]. http://www.gartner.com/newsroom/id/2227215

[2] ESRI. (2012) ArcGIS for Mobile. [Online]. http://www.esri.com/~/media/Files/Pdfs/library/brochures/pdfs/arcgis-for-mobile.pdf

[3] Ontology Engineering Group. Internet of Things. [Online]. http://mayor2.dia.fi.upm.es/oeg-upm/index.php/es/researchareas/5-internetofthings

[4] Gross, Doug. (2011, Octubre) 10 ways mobile gadgets have changed our lives. [Online]. http://edition.cnn.com/2011/10/07/tech/mobile/smartphones-change-lives

[5] International Telecommunication Union. (2010, Julio) MDG - ITU Story Line. [Online]. http://www.itu.int/ITU-D/ict/mdg/storyline/index.html

[6] Samsung. (2012, Abril) Samsung's New Quad-core Application Processor. [Online]. http://www.samsung.com/global/business/semiconductor/minisite/Exynos/news_12.html

[7] Raphael Schrader. (2012, Agosto) Gartner - Android Rules the Smartphone Market Share in Q2 2012 [Report] | FriendCaller Blog. [Online]. http://blog.friendcaller.com/gartner-android-rules-the-smartphone-market-share-in-q2-2012-report/

[8] Tim Schiesser. (2012, Febrero) Guide to smartphone hardware (1/7): Processors - Neowin. [Online]. http://www.neowin.net/news/guide-to-smartphone-hardware-17-processors

[9] Angel Luis Sanchez Iglesias. ¿Qué significa SOC? System on a chip. [Online]. http://computadoras.about.com/od/preguntas-frecuentes/a/Que-Significa-Soc-System-On-A-Chip.htm

[10] ARM. Smartphones - ARM. [Online]. http://www.arm.com/markets/mobile/smartphones.php

[11] Tim Schiesser. (2012, Febrero) Guide to smartphone hardware (2/7): Graphics - Neowin. [Online]. http://www.neowin.net/news/guide-to-smartphone-hardware-27-graphics-2

[12] GARMIN. Garmin | What is GPS? [Online]. http://www8.garmin.com/aboutGPS/

[13] GeoNet Ltd. What is GNSS?:GeoNet. [Online]. http://geonet.bg/answers/items/what-is-gnss.42.html

[14] Qualcomm. (2011, Diciembre) GPS and GLONASS: "Dual-core" Location For Your Phone. [Online]. http://www.qualcomm.com/media/blog/2011/12/15/gps-and-glonass-dual-core-location-your-phone

[15] M.Sc. Maximilian Schirmer and Jun.-Prof. Dr.-Ing. Hagen Höpfner. Smartphone Hardware Sensors. [Online]. http://www.uni-weimar.de/medien/wiki/images/Zeitmaschinen-smartphonesensors.pdf

[16] Jordi Sabaté Martí. (2011, Marzo) Trucos para alargar la batería de un móvil smartphone | EROSKI CONSUMER. [Online]. http://www.consumer.es/web/es/tecnologia/hardware/2011/03/22/199332.php

[17] ThermoAnalytics : HEV Vehicle Battery Types. [Online]. http://www.thermoanalytics.com/support/publications/batterytypesdoc.html

[18] Aaron Carroll and Gernot Heiser. (2010) An Analysis of Power Consumption in a Smartphone. [Online]. http://www.nicta.com.au/pub?doc=3587

[19] Abhinav Pathak, Y. Charlie Hu, and Ming Zhang. (2012) Where is the energy spent inside my app? [Online]. http://research.microsoft.com/en-us/people/mzh/eurosys-2012.pdf

[20] Jeff Sharkey. (2009, Mayo) Coding for Life--Battery Life, That Is. [Online]. https://dl.google.com/io/2009/pres/W_0300_CodingforLife-BatteryLifeThatIs.pdf

[21] Gartner. (2012, Mayo) Gartner Says Worldwide Sales of Mobile Phones Declined 2 Percent in First Quarter of 2012. [Online]. http://www.gartner.com/it/page.jsp?id=2017015

[22] Open Handset Alliance FAQ. [Online]. http://www.openhandsetalliance.com/oha_faq.html

[23] Marko Gargenta, "Android Overview," in *Learning Android*., 2011, ch. 1.

[24] Apache License, Version 2.0. [Online]. http://www.apache.org/licenses/LICENSE-2.0

[25] The MIT License (MIT). [Online]. http://opensource.org/licenses/mit-license.php

[26] IDC. (2012, Junio) Android Expected to Reach Its Peak This Year as Mobile Phone Shipments Slow, According to IDC. [Online]. http://www.idc.com/getdoc.jsp?containerId=prUS23523812

[27] Oracle. (2013, Agosto) Chapter 1. Introduction. [Online]. http://docs.oracle.com/javase/specs/jls/se7/html/jls-1.html

[28] Amber D. Walker. (2013, Agosto) Advantages & Disadvantages of Java Virtual Machine Interpreter | eHow. [Online]. http://www.ehow.com/facts_5558791_advantages-java-virtual-machine-interpreter.html

[29] David Ehringer. (2010, Marzo) Adventures in Software Development | Mindful Mischief. [Online]. http://davidehringer.com/software/android/The_Dalvik_Virtual_Machine.pdf

[30] Android. Exploring the SDK. [Online]. http://developer.android.com/sdk/exploring.html

[31] Android. (2013, Agosto) SDK Tools | Android Developers. [Online]. http://developer.android.com/tools/sdk/tools-notes.html

[32] Kerris, Natalie. (2007, Enero) Apple - Press Info - Apple Reinvents the Phone with iPhone. [Online]. http://www.apple.com/pr/library/2007/01/09Apple-Reinvents-the-Phone-with-iPhone.html

[33] Dowling, Steve. (2007, Enero) Apple - Press Info - iPhone to Support Third-Party Web 2.0 Applications. [Online]. http://www.apple.com/pr/library/2007/06/11iPhone-to-Support-Third-Party-Web-2-0-Applications.html

[34] Kerris, Natalie. (2008, Marzo) Apple - Press Info - Apple Announces iPhone 2.0 Software Beta. [Online]. http://www.apple.com/pr/library/2008/03/06Apple-Announces-iPhone-2-0-Software-Beta.html

[35] Apple Inc. XCode 4 - Apple Developer. [Online]. https://developer.apple.com/xcode/

[36] Apple Inc. Instruments User Guide: About Instruments. [Online]. https://developer.apple.com/library/mac/#documentation/developertools/conceptual/InstrumentsUserGuide/Introduction/Introduction.html

[37] Apple Inc. Tools Workflow Guide for iOS. [Online]. http://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/ios_development_workflow/25-Using_iOS_Simulator/ios_simulator_application.html#//apple_ref/doc/uid/TP40007959-CH9-SW1

[38] Apple Inc. (2013, Agosto) Programming with Objective-C. [Online]. https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html

[39] Apple Inc. (2009, Octubre) Objective-C Runtime Programming Guide: Introduction. [Online]. https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ObjCRuntimeGuide/Introduction/Introduction.html

[40] Apple Inc. (2013, Agosto) Cocoa Touch Frameworks. [Online]. https://developer.apple.com/technologies/ios/cocoa-touch.html

[41] Apple Inc. (2011, Octubre) iOS Technology Overview. [Online]. http://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf

[42] Margaret Rouse. (2008) TechTarget. [Online]. http://searchunifiedcommunications.techtarget.com/definition/real-time-communications

[43] Ericsson Labs. Push lasts longer - Save your mobile battery. [Online]. https://labs.ericsson.com/developer-community/blog/push-lasts-longer---save-your-mobile-battery

[44] Abdulkadir Özcan, Dhinaharan Nagamalai, and Jan Zizka, *Recent Trends in Wireless and*

Mobile Networks: Third International Conferences, WiMo 2011 and CoNeCo 2011.: Springer, 2011.

[45] Frank H. P. Fitzek and Hassan Charaf, *Mobile Peer to Peer (P2P): A Tutorial Guide*.: Wiley, 2009.

[46] TICbeat. El crecimiento de la telefonía móvil - infografía | TICbeat. [Online]. http://www.ticbeat.com/sim/crecimiento-telefonia-movil-infografia/

[47] Cellular Center. (2011, Noviembre) 4G en Uruguay: Verdades a medias tintas - Cellular Center. [Online]. http://www.celulares.com.uy/blog/11-general/130-4g-en-uruguay-verdades-a-medias-tintas

[48] Emiliano Cotelo. (2011, Noviembre) ¿La tecnología 4G llegó al Uruguay? Informes. [Online]. http://www.espectador.com/1v4_contenido.php?id=226521&sts=1

[49] El País. (2011, Noviembre) Operadores chocan por debut del 4G - Diario EL PAIS - Montevideo - Uruguay. [Online]. http://www.elpais.com.uy/suplemento/empresario/operadores-chocan-por-debut-del-4g/elempre_605460_111111.html

[50] NETUruguay. (2011, Noviembre) Claro lanza tecnología 4G. [Online]. http://neturuguay.com/index.php/noticias/item/922-claro-lanza-tecnolog%C3%ADa-4g

[51] Movistar. Internet Fácil Turbo 5GB. [Online]. http://www.movistar.com.uy/Default.aspx?tabid=462

[52] Antel. (2011, Diciembre) Antel - Institucional - Eventos - Internet Vera: Antel primera en lanzar tecnología LTE en Latinoamérica. [Online]. http://www.antel.com.uy/antel/institucional/sala-de-prensa/eventos/2011/internet-vera-antel-primera-en-lanzar-tecnologia-lte-en-latinoamerica

[53] Antel. Internet Vera: Antel primera en lanzar tecnología LTE en Latinoamérica. [Online]. http://www.antel.com.uy/wps/wcm/connect/e48121804966085896cd9e7c46b5f36a/Gacetilla-lanzamiento-LTE.pdf?MOD=AJPERES&CACHEID=e48121804966085896cd9e7c46b5f36a

[54] Edward Curry, "Message-Oriented Middleware," in *Middleware for Communications*, Qusay Mahmoud, Ed., 2004, ch. 1.

[55] Jorge Escobar. (2009) Push Technology is the Core of the Real Time Web. [Online]. http://jungleg.com/2009/07/07/push-technology-is-the-core-of-the-real-time-web/

[56] Alessandro Alinone. (2011) 10 Years of Push Tecnology, Comet, and WebSockets. [Online]. http://cometdaily.com/2011/07/06/push-technology-comet-and-websockets-10-years-of-history-from-lightstreamers-perspective/

[57] Greg Wilkins. (2007) Comet is Always Better Than Polling. [Online]. http://cometdaily.com/2007/11/06/comet-is-always-better-than-polling/

[58] W3C. (2011) The WebSocket API. [Online]. http://www.w3.org/TR/websockets/

[59] Google Inc. (2013, Agosto) Google Cloud Messaging for Android | Android Developers. [Online]. http://developer.android.com/google/gcm/index.html

[60] Apple Inc. (2013, Abril) Local and Push Notification Programming Guide: Apple Push Notification Service. [Online]. https://developer.apple.com/library/mac/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html#//apple_ref/doc/uid/TP40008194-CH100-SW9

[61] Joyent Inc. (2013, Setiembre) About Node.js. [Online]. http://nodejs.org/about/

[62] Joyent Inc. (2013, Setiembre) Node.JS. [Online]. http://nodejs.org/

[63] Joyent Inc. (2013, Setiembre) NPM socket.io. [Online]. https://npmjs.org/package/socket.io

[64] David Walsh. (2010, Noviembre) WebSocket and Socket.IO. [Online]. http://davidwalsh.name/websocket

[65] Guillermo Rauch. (2013, Agosto) Socket.IO: the cross-browser WebSocket for realtime apps. [Online]. http://socket.io/

[66] Guillermo Rauch. (2013, Agosto) Socket.IO: the cross-browser WebSocket for realtime apps. [Online]. http://socket.io/#faq

[67] Guillermo Rauch. (2013, Agosto) Socket.IO: the cross-browser WebSocket for realtime apps. [Online]. http://socket.io/#browser-support

[68] Apple. iOS Human Interface Guidelines: Designing for iOS7. [Online]. https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/

[69] Android Developers. Android Design Guidelines. [Online]. http://developer.android.com/design/index.html

[70] Google. Google Play Developer Program Policies. [Online]. https://play.google.com/about/developer-content-policy.html

[71] Apple. App Review Guidelines. [Online]. https://developer.apple.com/appstore/guidelines.html

[72] Artech Consultores S.R.L. (2013, Agosto) About Artech. [Online]. http://www.genexus.com/company/about-artech?en

[73] Artech Consultores S.R.L. (2012) GeneXus Overview. [Online]. http://www.genexus.com/files/wp-genexus-overview?en

[74] GeneXus. GeneXus, Cross-platform software development. [Online]. http://www.genexus.com/technologies

[75] GeneXus. (2012) Mobile and Smart Devices Development Solution. [Online]. http://www.genexus.com/files/wp-mobile-application-development-genexus.pdf?en

[76] W3C. HTML Current Status. [Online]. http://www.w3.org/standards/techs/html#drafts

[77] Jennifer Niederst Robbins, *HTML & XHTML Pocket Reference*.: O'Reilly, 2009.

[78] Google. HTML5 Rocks. [Online]. http://www.html5rocks.com/en/why

[79] W3C. Cascading Style Sheets (CSS3). [Online]. http://www.w3.org/TR/CSS/#css3

[80] W3C. Geolocation API Specification. [Online]. http://dev.w3.org/geo/api/spec-source.html

[81] Sencha Inc. (2013, Agosto) Mobile HTML5 Framework - Features of Sencha Touch | Features | Sencha Touch | Products | Sencha. [Online]. http://www.sencha.com/products/touch/features/

[82] Sencha Inc. (2013, Agosto) Native Packaging for Mobile Devices. [Online]. http://docs-origin.sencha.com/cmd/3.1.2/#!/guide/native_packaging

[83] Sencha Inc. (2013, Agosto) How to Use Classes in Sencha Touch. [Online]. http://docs.sencha.com/touch/2.2.1/#!/guide/class_system

[84] Sencha Inc. (2013, Agosto) The Sencha Class System | Learn | Sencha. [Online]. http://www.sencha.com/learn/sencha-class-system

[85] Sencha Inc. (2013, Agosto) Using Navigation View in Sencha Touch. [Online]. http://docs.sencha.com/touch/2.2.1/#!/guide/navigation_view

[86] Adobe. PhoneGap. [Online]. http://phonegap.com

[87] ESRI. (2004, Enero) What is ArcGIS. [Online]. http://downloads.esri.com/support/documentation/ao_/698What_is_ArcGIS.pdf

[88] Curso Sistemas de Información Geográfica. (2012, Agosto) Introducción a los SIG. [Online]. http://www.fing.edu.uy/inco/cursos/sig/clases/Generalidades160812.pdf

[89] ESRI. Esri Products | Alphabetical Index of All Products. [Online]. http://www.esri.com/products/products-alpha

[90] ESRI. ArcGIS - Features. [Online]. http://www.esri.com/software/arcgis/features

[91] ESRI. ArcGIS for Server | GIS Web Server Software | Web Mapping Server. [Online]. http://www.esri.com/software/arcgis/arcgisserver

[92] ESRI. Servicios SIG | ArcGIS Resource Center. [Online]. http://resources.arcgis.com/es/content/arcgisserver/10.0/gis-services

[93] ESRI. ArcGIS for Mobile | Mobile Solutions that Fit Your Enterprise. [Online]. http://www.esri.com/software/arcgis/about/mobile-gis-for-you

[94] Esri. (2013, Setiembre) ArcGIS API for JavaScript. [Online]. https://developers.arcgis.com/en/javascript/

[95] Esri. (2013, Setiembre) ArcGIS Runtime SDK for Android - Guide. [Online]. https://developers.arcgis.com/en/android/guide/welcome-to-the-help-for-arcgis-runtime-sdk-for-android.htm

[96] Khronos Group. (2013, Setiembre) OpenGL ES - The Standard for Embedded Accelerated 3D Graphics. [Online]. http://www.khronos.org/opengles/

[97] (2013, Setiembre) ArcGIS Runtime SDK for Android - System Requirements. [Online]. https://developers.arcgis.com/en/android/system-reqs.html

[98] Esri. (2013, Setiembre) ArcGIS API for JavaScript Overview. [Online]. https://developers.arcgis.com/en/javascript/jshelp/

[99] Esri. (2013, Setiembre) Get the ArcGIS API for JavaScript. [Online]. https://developers.arcgis.com/en/javascript/jshelp/intro_accessapi.html

[100] Esri. (2013, Setiembre) ArcGIS API for JavaScript - Compact Build. [Online]. https://developers.arcgis.com/en/javascript/jshelp/inside_compactbuild.html

[101] Microsoft. (2013, Agosto) Bing Maps. [Online]. http://msdn.microsoft.com/en-us/library/dd877180.aspx

[102] (2013, Setiembre) Viewing Bing Maps Usage. [Online]. http://msdn.microsoft.com/en-us/library/ff859477.aspx

[103] Elmasri-Navathe,., ch. 1.

[104] Ralf Hartmut Güting. (1994, Octubre) An Introduction to Spatial Database Systems. [Online]. http://www.cise.ufl.edu/~mschneid/Research/thesis_papers/Gue94VLDBJ.pdf

[105] E. F. Codd. (1970, Junio) A Relational Model of Data for Large Shared Data Banks. [Online]. http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

[106] Mike Chapple. (2013, Setiembre) SQL Fundamentals. [Online]. http://databases.about.com/od/sql/a/sqlfundamentals.htm

[107] SQLite. (2013, Setiembre) About SQLite. [Online]. http://www.sqlite.org/about.html

[108] SQLite. (2013, Setiembre) SQLite Is Serverless. [Online]. http://www.sqlite.org/serverless.html

[109] Brian Ritchie. (2010, Agosto) An Introduction to Document Databases. [Online]. http://weblogs.asp.net/britchie/archive/2010/08/12/document-databases.aspx

[110] Google Inc. (2013, Setiembre) Interés en Búsqueda web: nosql. En todo el mundo, 2004 - hoy. [Online]. http://www.google.com/trends/explore?q=nosql#q=nosql&cmpt=q

[111] Brian Ritchie. (2010, Agosto) An Introduction to Document Databases. [Online].

http://weblogs.asp.net/britchie/archive/2010/08/12/document-databases.aspx

[112] Michael Morin. (2013, Setiembre) NoSQL and Document Oriented Databases. [Online]. http://ruby.about.com/od/nosqldatabases/a/nosql1.htm

[113] MongoDB Inc. (2013, Setiembre) MongoDB Overview. [Online]. https://www.mongodb.com/products/mongodb

[114] MongoDB Inc. (2013, Setiembre) JSON and BSON. [Online]. https://www.mongodb.com/json-and-bson

[115] MongoDB Inc. (2013, Setiembre) Aggregation Concepts. [Online]. http://docs.mongodb.org/manual/core/aggregation/

[116] Marc Seeger. (2009, Setiembre) Key-Value stores: a practical overview. [Online]. http://blog.marc-seeger.de/assets/papers/Ultra_Large_Sites_SS09-Seeger_Key_Value_Stores.pdf

[117] Citrusbyte. (2013, Setiembre) Redis. [Online]. http://redis.io/

[118] Citrusbyte. (2013, Setiembre) Redis Clients. [Online]. http://redis.io/clients

[119] C.U.T.C.S.A. (2012) iBus // Ya no lo esperás,ahora sabés cuándo te pasa a buscar. [Online]. http://www.ibus.com.uy/terminos_y_condiciones.html

[120] LG Electronics MobileComm. (2008) IEEE ICC 2008 - Beijing. [Online]. http://to.swang.googlepages.com/ICC2008LBSforMobilessimplifiedR2.pdf

[121] C.U.T.C.S.A. (2012) iBus // Ya no lo esperás,ahora sabés cuándo te pasa a buscar. [Online]. http://www.ibus.com.uy/como_se_usa.html

[122] Vic Gundotra. (2009, Febrero) See where your friends are with Google Latitude. [Online]. http://googleblog.blogspot.com/2009/02/see-where-your-friends-are-with-google.html

[123] Google. (2010) Google Latitude. [Online]. http://www.google.com/intl/es_ALL/mobile/latitude/

[124] Bellevue Linux. (2005, Setiembre) Latency Definition. [Online]. http://www.linfo.org/latency.html

[125] Charles B. Kreitzberg and Ambrose Little. Agile Ux Development. [Online]. http://msdn.microsoft.com/en-us/magazine/dd882523.aspx

[126] Balsamiq Studio. Balsamiq. [Online]. http://balsamiq.com/

[127] Apple. iOS Human Interface Guidelines: Designing for iOS. [Online]. https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/

[128] IETF. RFC 4648 - The Base 64 Encoding. [Online].

https://tools.ietf.org/html/rfc4648#section-4

[129] Sinatra. Sinatra:Reloader. [Online]. http://www.sinatrarb.com/contrib/reloader.html

[130] Durran Jordan. Mongoid. [Online]. http://mongoid.org/en/mongoid/

[131] J Larrosa. (2012, Febrero) GeoLocationAPI - Smart Device Location Api. [Online]. http://wiki.gxtechnical.com/commwiki/servlet/hwiki?GeoLocationAPI+-+Smart+Device+Location+Api,

[132] Armin Bachmann. (2011, Mayo) Smart Devices: How to Show a Map with points. [Online]. http://wiki.gxtechnical.com/commwiki/servlet/hwiki?Smart+Devices%3A+How+to+Show+a+Map+with+points,

[133] L Silveira. (2012, Febrero) External Object for Android Smart Devices. [Online]. http://wiki.gxtechnical.com/commwiki/servlet/hwiki?External+Object+for+Android+Smart+Devices,

[134] L Silveira. (2012, Marzo) External Object for iOS Devices. [Online]. http://wiki.gxtechnical.com/commwiki/servlet/hwiki?External+Object+for+iOS+Devices,

[135] D Marquez. (2012, Julio) Creating User Controls for Android. [Online]. http://wiki.gxtechnical.com/commwiki/servlet/hwiki?Creating+User+Controls+for+Android

[136] D Marquez. (Julio, 2012) Creating User Controls for iOS. [Online]. http://wiki.gxtechnical.com/commwiki/servlet/hwiki?Creating+User+Controls+for+iOS,

[137] Android Developers. Transferring Data Without Draining the Battery. [Online]. http://developer.android.com/training/efficient-downloads/index.html

[138] Android Developers. LocalBroadcastManager. [Online]. http://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html

[139] Android Developers. ContentObserver. [Online]. http://developer.android.com/reference/android/database/ContentObserver.html

[140] Koushik Dutta. AndroidAsync. [Online]. https://github.com/koush/AndroidAsync

[141] Android Developers. Support Library. [Online]. http://developer.android.com/tools/support-library/index.html

[142] FasterXML. Jackson Mapper. [Online]. https://github.com/FasterXML/jackson-core

[143] SpringSource. Spring for Android. [Online]. http://www.springsource.org/spring-android

[144] AndroidDevelopers. Action Bar Design Pattern. [Online]. http://developer.android.com/design/patterns/actionbar.html

[145] Jake Wharton. ActionBarSherlock. [Online]. http://actionbarsherlock.com/

[146] Benjamin Weiss. Crouton. [Online]. https://github.com/keyboardsurfer/Crouton

[147] Manuel Peinado. RefreshActionItem. [Online]. https://github.com/ManuelPeinado/RefreshActionItem

[148] Jeremy Feinstein. Sliding Menu. [Online]. https://github.com/jfeinstein10/SlidingMenu

[149] ESRI. ArcGIS Online Geocoding Service. [Online]. http://geocode.arcgis.com/

[150] IMM. Datos abiertos. [Online]. http://www.montevideo.gub.uy/node/13757

[151] Intendencia de Montevideo. (2013, Setiembre) Principios para el manejo de datos abiertos en el gobierno | Intendencia de Montevideo. [Online]. http://www.montevideo.gub.uy/institucional/datos-abiertos/articulos-de-interes/principios-para-el-manejo-de-datos-abiertos-en-el-

[152] Intendencia de Montevideo. Líneas de ómnibus, origen y destino. [Online]. http://www.montevideo.gub.uy/datos/23415

[153] Intendencia de Montevideo. Transporte colectivo (paradas, líneas de ómnibus y puntos de control). [Online]. http://www.montevideo.gub.uy/datos/15066

[154] Spatial Reference. (2013, Setiembre) SIRGAS 2000 / UTM zone 21S: EPSG Projection -- Spatial Reference. [Online]. http://spatialreference.org/ref/epsg/31981/

[155] Spatial Reference. (2013, Setiembre) WGS 84: EPSG Projection -- Spatial Reference. [Online]. http://spatialreference.org/ref/epsg/4326/

[156] QuantumGIS. [Online]. http://qgis.org/

[157] Pentaho Corporation. (2013, Setiembre) Pentaho Data Integration (Kettle). [Online]. http://kettle.pentaho.com/

[158] Salesforce. Heroku. [Online]. https://www.heroku.com/

[159] MongoHQ. MongoHQ. [Online]. https://www.mongohq.com

[160] Heroku dev center. Using Socket.IO with Node.js on Heroku. [Online]. https://devcenter.heroku.com/articles/using-socket-io-with-node-js-on-heroku

[161] Kaazing. HTML5 Web Sockets: A Quantum Leap in Scalability for the Web. [Online]. http://www.websocket.org/quantum.html

[162] ISTQB GUIDE. (2013, Octubre) Why is testing necessary? [Online]. http://istqbexamcertification.com/why-is-testing-necessary/

[163] IIS. (2008, Marzo) Verificación y Validación. [Online]. http://www.fing.edu.uy/inco/cursos/iis/wikiIIS/uploads/Material/10-Verificacion-Validacion.pdf

[164] Seattlerb. (2013, Octubre) bfts's minitest-5.0.8 Documentation. [Online]. http://docs.seattlerb.org/minitest/

[165] Joe Ferris. (2013, Octubre) factory_girl. [Online]. https://github.com/thoughtbot/factory_girl

[166] Visionmedia. (2013, Octubre) Mocha - the fun, simple, flexible JavaScript test framework. [Online]. http://visionmedia.github.io/mocha/

[167] Bryntum. (2013, Octubre) Siesta. [Online]. http://www.bryntum.com/products/siesta/

[168] Taller de Verificación de Software. (2013, Octubre) Automatización en el Testing Funcional: Introducción y Metodología. [Online]. http://www.fing.edu.uy/inco/cursos/tvs/pmwiki/uploads/Material/Automatizaci%f3n%20del%20Testing%20Funcional%20-%20Parte%20I.pdf

[169] Visionmedia. Mocha. [Online]. http://visionmedia.github.io/mocha/

[170] (2013, Abril) Lineas de Omnibus de Montevideo. [Online]. http://www.geocoder.com.uy/geocoder_buses.php

[171] Eric Shepherd. (2013, Abril) Touch - Web API reference | MDN. [Online]. https://developer.mozilla.org/en-US/docs/Web/API/Touch

Instituto de Computación - Facultad de Ingeniería - UdelaR

InTrack

Software Geográfico AVL para Ómnibus

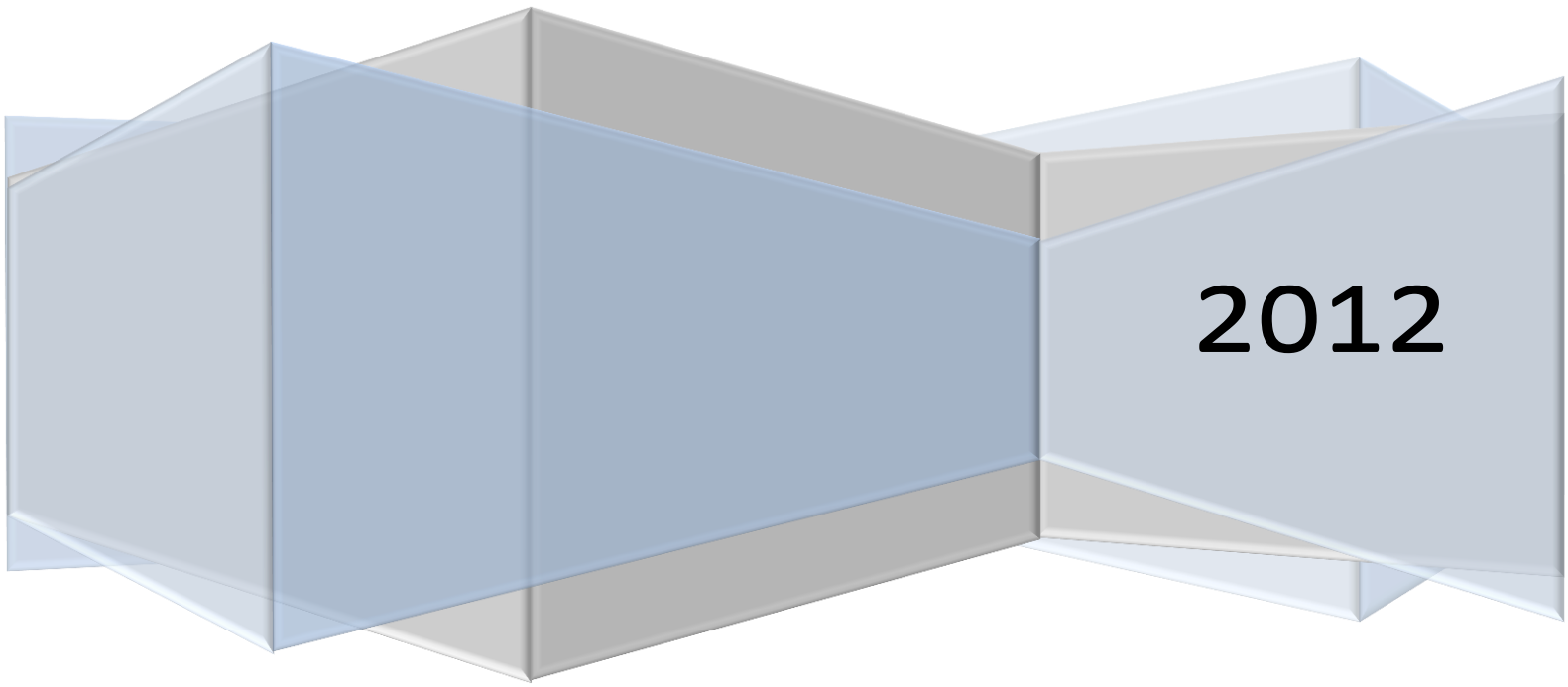
Integrantes

Bruno González, Máximo Mussini

Tutores

Pablo Rebufello, Sandro Moscatelli

ANEXO I – ESTADO DEL ARTE



Contenido

1 Introducción

El presente proyecto de grado involucra el diseño y construcción de una plataforma de servicios y una aplicación de software con el objetivo de compartir información geográfica entre dispositivos móviles en tiempo real. La plataforma presentará funcionalidades de un sistema AVL[1] permitiendo localizar ómnibus, y consultar tanto su ubicación como información relacionada (recorrido, paradas, etc.) en tiempo real.

De forma similar la plataforma permitirá determinar la posición geográfica de los usuarios registrados en el sistema, brindando un mecanismo para la transmisión de datos punto a punto entre los usuarios del sistema, brindando la posibilidad de que establezcan un punto de encuentro, y permitiendo el seguimiento mutuo entre usuarios.

La plataforma incorpora características de las redes sociales; el acceso a la información de un usuario se regula mediante niveles de privacidad. Cada usuario tendrá una lista de usuarios asociados con los cuales puede compartir su posición geográfica y otros datos.

Finalmente, por tratarse de una aplicación orientada al público en general, será necesario prestar especial atención a la interfaz gráfica de la aplicación, apuntando a un diseño amigable e intuitivo sin comprometer su funcionalidad.

Si bien el objetivo principal del proyecto es el desarrollo del sistema descrito, se pretende cubrir otros aspectos tales como determinar las dificultades y limitaciones en el desarrollo para dispositivos móviles, estudiar diferentes alternativas de comunicación en tiempo real, e investigar formas de reducir el consumo de batería de los dispositivos.

En primer lugar se deberán analizar las características de los dispositivos móviles actuales, tanto a nivel de hardware como de software. En lo que refiere a software se estudiarán los sistemas operativos existentes, realizando una revisión de sus características principales y un breve análisis comparativo, evaluando además las distintas herramientas y ambientes de desarrollo disponibles para los mismos. En cuanto al hardware se tendrán en cuenta la capacidad de procesamiento y memoria de los dispositivos más representativos en la actualidad, junto con otras características como la capacidad gráfica, presencia de GPS, y conexión inalámbrica a la red de datos.

Además, se pretende evaluar hasta qué punto es posible desarrollar una aplicación portable con las herramientas y plataformas disponibles. En particular, se evaluará la posibilidad de desarrollar una aplicación multiplataforma mediante Genexus X EV2 o HTML5. En el caso de HTML5 interesa comparar sus prestaciones contra las de una aplicación nativa, tanto en rendimiento como en acceso a las características físicas del dispositivo.

Será necesario evaluar diferentes alternativas que permitan lograr una comunicación punto a punto entre dispositivos con una latencia mínima, por lo cual se investigarán protocolos y middleware de comunicación que pudieran ser utilizados con el propósito de transmitir información en tiempo real. Resultarán de especial interés aquellos métodos que tomen en

[1] AVL (Automatic Vehicle Locator)

cuenta la escasa autonomía de los dispositivos móviles actuales. A su vez, habrá que considerar la infraestructura de red subyacente, en particular la existente en Uruguay.

Por último, nos embarcaremos en el estudio de aquellas aplicaciones disponibles en el mercado que estén relacionadas con el proyecto, más concretamente, aquellas que intenten proveer funcionalidades similares.

Considerando la amplia gama de los temas anteriormente identificados, se hará énfasis en los que creemos son los más relevantes para la comprensión del proyecto, presentando brevemente el resto.

2 Dispositivos Móviles

Los dispositivos móviles se han convertido en una parte esencial de la vida diaria como herramientas de comunicación eficaces y cómodas, accesibles en todo momento y lugar. A su vez, la gran variedad de aplicaciones disponibles en mercados y la diversidad de servicios que estas proveen permiten que los usuarios obtengan una experiencia muy completa [1].

La evolución del teléfono móvil ocurrió a lo largo de cinco generaciones distintas, convirtiéndose en uno de los principales medios de comunicación a fines de la década del '90 [2]. En alrededor de 20 años, esta tecnología conquistó a las masas y se convirtió en una necesidad básica para la mayor parte de la gente.

Originalmente, los teléfonos móviles estaban pensados para comunicar a la gente, sin importar el tiempo y lugar, permitiendo la transmisión simple de voz. Con el avance de la tecnología fueron evolucionando en sus características y funciones, convirtiéndose en dispositivos de entretenimiento que permiten a los usuarios almacenar su información personal, acceder a Internet, e incluso tener video-conferencias; acercándose cada vez más a las computadoras de escritorio en cuanto rendimiento [3], y dando lugar a grandes cambios.

En primer lugar, desde una perspectiva humana, la movilidad que brindan estos dispositivos permite a las personas acceder a sistemas computacionales en cualquier momento y en cualquier lugar, lo cual abre las puertas a un gran conjunto de nuevas aplicaciones. Estas aplicaciones admiten una gama de actividades diferente a la de las aplicaciones de escritorio, y pueden tomar provecho de la información que los dispositivos móviles poseen sobre el entorno del usuario, como ser su ubicación geográfica, para proporcionar mejores funcionalidades.

En segundo lugar, desde una perspectiva tecnológica, la movilidad logra un cambio global en la infraestructura de los sistemas computacionales; desde computadoras de escritorio, estáticas, homogéneas, y de gran potencia; hacia dispositivos de mano altamente dinámicos, heterogéneos, y con recursos limitados.

Si bien existe una poderosa tendencia en el desarrollo de la informática móvil, los dispositivos móviles aún se enfrentan a muchos desafíos en lo que respecta al uso eficiente de sus recursos (como ser batería y procesamiento), así como a la comunicación (específicamente la movilidad y la seguridad). Sumado a esto, las aplicaciones diseñadas para dispositivos móviles deben hacer frente a las restricciones de ancho de banda de la red de datos en la que se encuentran. Estas limitaciones presentan un obstáculo significativo a la hora de asegurar la calidad de los servicios brindados.

A continuación, se presentan las características de hardware y software de los dispositivos móviles actuales, así como un breve análisis del mercado de sistemas operativos y la distribución de aplicaciones.

2.1 Hardware de Dispositivos Móviles

Un aspecto relevante en el desarrollo de aplicaciones para dispositivos móviles, es el hardware disponible en los dispositivos objetivo. Ya sea por las funcionalidades que provea la aplicación

o por el rendimiento de la misma, es necesario tener en cuenta este factor que en muchos casos es determinante.

En lo que refiere al mercado de dispositivos móviles, las empresas que lo lideran son Samsung, Nokia y Apple [4]. A continuación se presenta una gráfica basada en un estudio de la empresa Gartner acerca de la venta de dispositivos móviles a usuarios finales hasta el segundo trimestre de 2012.

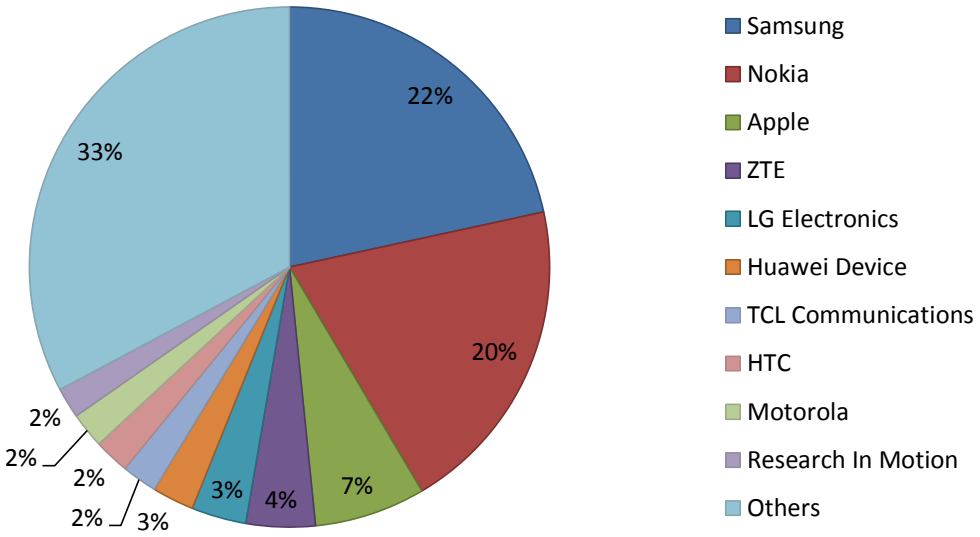


Figura 2.1: Proporción de ventas del mercado de dispositivos móviles (segundo trimestre 2012)

En la presente sección se analizará la capacidad de procesamiento de los dispositivos móviles para poder determinar el potencial que estos despliegan. Luego se estudiará la capacidad gráfica, las diferentes alternativas que hay en el mercado, y las ventajas y desventajas que presenta cada una. Se continúa con un estudio del sensor de GPS[2], ya que conocer sus características será de gran importancia para el proyecto debido a su naturaleza geográfica, culminando con un análisis de las tecnologías de batería, lo cual es de vital importancia ya que esta determina la autonomía del dispositivo móvil.

2.1.1 Procesador

Como se mencionó anteriormente, el procesador del dispositivo móvil es el que le dará a este la capacidad de procesamiento, lo que hace que forme parte del hardware determinante en lo que respecta a la ejecución de aplicaciones de usuario.

Cuando hablamos de procesadores en el ámbito de los teléfonos móviles, en realidad debemos referirnos a los llamados SoC[3], que son dispositivos que integran en un mismo chip varios componentes de un sistema [5] [6]. Los componentes que este integra son: núcleos del

[2] GPS (Global Positioning System)
[3] SoC (System-on-a-chip)

procesador, unidad de procesamiento de gráficos, memoria RAM[4] y ROM[5], e interfaces externas (por ejemplo: USB o tecnología inalámbrica, entre otros).

Concentrar los componentes esenciales de un dispositivo en un espacio relativamente pequeño, mejora la performance general y hace un uso más eficiente de la batería, además de abaratar los costos de ensamblado total del hardware del dispositivo.

Como los requerimientos de consumo y tamaño en un dispositivo móviles son mucho más acotados que en una computadora de escritorio, se han desarrollado arquitecturas pensadas específicamente para este tipo de dispositivos [5].

Hoy en día, ARM[6] es la arquitectura que lidera en lo que refiere a dispositivos móviles [7]. ARM es tres cosas: una empresa (ARM Holdings PLC), una arquitectura de microprocesadores y el núcleo del procesador [5]. Algunas compañías como NVidia, Samsung o Texas Instruments compran la licencia del microprocesador ARM, para luego producir sus propios SoCs, a diferencia de Qualcomm o Marvell que únicamente compran la licencia para utilizar el conjunto de instrucciones del ARM, para luego producir sus propios procesadores [8].

Actualmente existen dos procesadores ARM ampliamente utilizados en el mercado [7]: el ARM Cortex-A8 y el ARM Cortex-A9 (ambos utilizan la arquitectura ARMv7). El Cortex-A8 es single-core, trabaja bajo frecuencias entre 600 MHz y 1 GHz (o más), consume menos de 300 mW y ejecuta 2.0 DMIPS[7]/MHz [9]. El Cortex-A9 es multi-core (hasta 4 núcleos), puede trabajar bajo frecuencias de hasta 2 GHz, consume 250 mW y ejecuta 2.50 DMIPS/MHz por núcleo [10].

2.1.2 Capacidad gráfica

Otro de los dispositivos que se pueden encontrar dentro del SoC es la unidad de procesamiento de gráficos o GPU[8], la cual está muy próxima al procesador [11]. El uso que se le da a la GPU depende del sistema operativo y la estructura del SoC, pero básicamente se encarga de procesar tareas relacionadas con el procesamiento de gráficos, lo cual permite liberar a la CPU[9].

En lo que refiere a las tareas propias del sistema operativo, la GPU es utilizada para renderización 3D en juegos y aplicaciones. Los núcleos Cortex no están diseñados para procesar este tipo de tareas, por lo que las mismas son delegadas a la GPU, la cual las procesará de forma más eficiente.

El sistema operativo Android tiene algunas particularidades con respecto al procesamiento de tareas de renderización. En un principio, como los dispositivos que corrían Android no poseían GPUs lo suficientemente potentes para encargarse completamente de este tipo de tareas, Google decidió delegarlas completamente a la CPU. Esto fue corregido en la versión 4.0 de este sistema operativo, ya que los SoCs actuales poseen GPUs con mejor capacidad de

[4] RAM (Random-Access Memory)
[5] ROM (Read-Only Memory)
[6] ARM (Advanced RISC Machines)
[7] DMIPS (Dhrystone Million Instructions Per Second)
[8] GPU (Graphics Processing Unit)
[9] CPU (Central Processing Unit)

procesamiento. Por otro lado, iOS utiliza la GPU para la mayoría de las tareas de renderización [11].

En la actualidad hay varias compañías que fabrican GPUs, entre las que podemos encontrar a las series Andreno de Qualcomm [12], POWERVR de Imagination Technologies [13], Mali de ARM [14] y Tegra de NVidia [15]. A continuación se presenta una tabla comparativa entre las distintas GPU que hay en el mercado [11]:

GPU	Ejemplo de SoC	Ejemplo de Dispositivo	GFLOPS[10] a 200 MHz	GFLOPS en SoC
PowerVR SGX543MP4+	PSVita	PlayStation Vita	25.6	25.6+
PowerVR SGX543MP2	Apple A5	Apple iPhone 4S	12.8	16 at 250 MHz*
Mali-400 MP4	Exynos 4210	Samsung Galaxy S II	7.2	9.9 at 275 MHz
"Kal-El" GeForce	Tegra 3	ASUS Transformer Prime	4.8	9.6 at 400 MHz*
PowerVR SGX540	OMAP4460	Galaxy Nexus	3.2	6.1 at 384 MHz
Adreno 220	MSM8260	HTC Sensation	N/A	N/A
ULP GeForce	Tegra 2	Motorola Xoom	3.2	5.3 at 333 MHz
PowerVR SGX540	OMAP4430	Motorola Droid Razr	3.2	4.8 at 304 MHz
ULP GeForce	Tegra 2	LG Optimus 2X	3.2	4.8 at 300 MHz
PowerVR SGX540	Hummingbird	Samsung Galaxy S	3.2	3.2 at 200 MHz
Adreno 205	MSM8255	HTC Titan	N/A	N/A
PowerVR SGX535	Apple A4	iPhone 4	1.6	1.6 at 200 MHz*
PowerVR SGX530	OMAP3630	Motorola Droid X	1.6	1.6 at 200 MHz
Adreno 200	QSD8250	HTC HD7	N/A	N/A

(*) Las cifras de estos GFLOPS se basan en velocidades de reloj de SoC estimadas

Tabla 2.1: Comparación entre GPUs

2.1.3 GPS

Habiendo visto la capacidad de procesamiento general y específico para gráficos en un dispositivo móvil y cómo impacta dicha capacidad en el funcionamiento general del mismo, se continuará realizando un estudio acerca de los sensores de localización geográfica.

Con respecto a las tecnologías utilizadas por dichos sensores, el GPS[11] es uno de los métodos utilizados para obtener la posición del dispositivo. Este es un sistema de navegación satelital conformado por 32 satélites puestos en órbita por el Departamento de Defensa de los Estados Unidos, los cuales en un principio fueron utilizados con propósito militar (en un principio había menor cantidad de satélites) y posteriormente (1980) habilitados para uso civil [16].

Vale la pena aclarar que el nombre correcto para este tipo de tecnologías es GNSS[12], que es una combinación de todos los sistemas de navegación satelital entre los cuales se encuentran

[10] GFLOPS (Giga Floating point Operations Per Second)
[11] GPS (Global Positioning System)
[12] GNSS (Global Navigation Satellite System)

GPS, GLONASS[13], WAAS[14], EGNOS[15], MSAS[16]. En el momento hay 32 satélites pertenecientes a la constelación GPS y 24 pertenecientes a GLONASS [17].

El sensor con el que cuentan los últimos dispositivos móviles del mercado no sólo procesa señales de los satélites GPS, sino que también procesa las de los satélites GLONASS. Al utilizar ambas señales, la cantidad de satélites que proveen información aumenta, lo cual mejora los resultados en tiempo real bajo circunstancias poco favorables (áreas urbanas, terreno montañoso o bosque) [17]. Según estudios de la empresa Qualcomm, utilizando la información provista por el servicio de GLONASS, combinado con el GPS, se puede obtener una precisión de hasta 2 metros [18].

Como ya se mencionó, el GPS es una de las formas con las que el sensor de localización puede obtener la posición geográfica del dispositivo móvil. También puede obtenerse mediante triangulación por torres celulares o redes WiFi, mediante una base de datos que contenga la posición geográfica de cada una de estas [19].

Los dispositivos móviles pueden seleccionar diferentes alternativas para obtener su posición geográfica (GPS, torres celulares o WiFi), dependiendo de la precisión que se desee obtener y de las circunstancias del momento [19].

Básicamente existen dos variantes de tipo de GPS: el S-GPS[17] y el A-GPS[18]. El S-GPS brinda la posibilidad de utilizar la señal de teléfono y GPS al mismo tiempo, ya que comúnmente se utiliza un esquema de multiplexado de tiempo. Este sistema permite mejorar la sensibilidad en 4-dB aproximadamente y habilita el LBS[19] a los proveedores de servicio [20]. El A-GPS es una mejora de los GPS que se encuentran en dispositivos móviles conectados a una red de datos. Este obtiene la información de ubicación de los satélites a través de la red de datos y la guarda para su posterior utilización. También puede llegar a utilizar la triangulación por antenas si la señal del GPS es débil o no está disponible (como en el interior de un edificio por ejemplo) [21].

2.1.4 Batería

Una de las características más importantes de un dispositivo móvil, más allá de la capacidad de procesamiento o sensores disponibles, es la autonomía. Un dispositivo móvil puede tener las mejores prestaciones del mercado a nivel de hardware, pero si cuenta con pocas horas de autonomía, sus capacidades se verán totalmente limitadas.

En el contexto del proyecto la batería es un factor muy importante a tener en cuenta, ya que se utiliza el sensor GPS y la red de datos para la trasmisión de información (mediante 3G o 4G), que son funciones que realizan un gran consumo de batería [22].

La batería de ión de Litio es el tipo de batería que se usa actualmente en dispositivos móviles. Este tipo de batería es usada en estos dispositivos, ya que el Litio es el metal más liviano (entre

[13] GLONASS (Global'naya Navigatsionnaya Sputnikovaya Sistema)
[14] WAAS (Wide Area Augmentation System)
[15] EGNOS (European Geostationary Navigation Overlay Service)
[16] MSAS (Multi-functional Satellite Augmentation System)
[17] S-GPS (Simultaneous GPS)
[18] A-GPS (Assisted GPS)
[19] LBS (Location Based Service)

los utilizados para este tipo de fines) y posee la mayor densidad de potencial (300 W/kg). También tiene uno de los ciclos de vida más altos (entre 400 y 1200 ciclos) y la ventaja de no tener el conocido efecto memoria [23].

Existen varias investigaciones abiertas en esta área, destacándose el mini generador del MIT [24], el cual se basa en células fotovoltaicas que generan energía a través del calor que toman del ambiente, del generado por el mismo dispositivo o de alguna otra fuente de calor próxima. También hay una investigación de Northwestern University [25] acerca del *sándwich* de carbono y silicio, que permitiría aumentar la densidad de iones de Litio, brindando mayor capacidad de autonomía, y por último, el polímero de University of Leeds [26] que plantearía mejoras a las baterías de Litio actuales simplemente cambiando el electrolito líquido por un polímero con consistencia de gel.

En lo que respecta a la duración de la batería en diferentes dispositivos móviles, presentamos un estudio realizado por la empresa CNET de Julio de 2012 [27] en donde se muestran los veinte celulares con mayor tiempo de conversación en el mercado, lo que dará algunas pautas acerca de este tema.

Puesto	Modelo	Tiempo de conversación (en horas)
1	Motorola Droid Razr Maxx	19,78
2	LG Optimus Vu (unlocked)	16,08
3	Apple iPhone 4 (with 3G off)	14,55
4	RIM BlackBerry Curve 9360 (T-Mobile)	12
5	Kyocera DuraPlus (Sprint)	11,47
6	Samsung Captivate Glide (AT&T)	10,3
7	ZTE Fury (Sprint)	10,02
8	Samsung Galaxy S II (U.S. Cellular)	9,47
9	LG Optimus 3D Max (unlocked)	9,35
10	Huawei Mercury (Cricket Wireless)	9,32
11	Apple iPhone 4S (Sprint)	9,2
12	Kyocera Milano (Sprint)	9,2
13	Samsung Focus S (AT&T)	9,01
14	Kyocera DuraCore (Sprint)	8,95
15	Samsung Rugby Smart (AT&T)	8,8
16	LG Lucid (Verizon Wireless)	8,47
17	T-Mobile Prism (T-Mobile)	8,32
18	RIM BlackBerry Torch 9810 (AT&T)	8,1
19	Pantech Swift (AT&T)	7,27
20	Huawei Ascend II (U.S. Cellular)	7,25

Tabla 2.2: Veinte celulares con mayor tiempo de conversación del mercado

En cuanto al uso de la batería en las aplicaciones, existen múltiples estudios que tratan el tema [28] [29] y muestran en dónde se consume mayor cantidad de energía. Haciendo un buen uso de las técnicas propuestas y de las recomendaciones de los expertos, se puede reducir el consumo de una aplicación entre un 20% y un 65% [29].

Dado que cada sistema operativo hace diferente uso del hardware disponible, existen recomendaciones propias de cada compañía en relación al desarrollo de aplicaciones, para que

estas puedan utilizar este recurso de forma adecuada. Por ejemplo, Google presentó una charla [30] en la que hacía recomendaciones específicas de qué clases usar, qué funciones no eran recomendables para determinados entornos y cómo podía hacerse un uso eficiente de los recursos de software para evitar el consumo innecesario de batería.

2.2 Sistemas Operativos para Móviles

El sistema operativo (OS) es el software que se comunica con el hardware de una computadora, maneja los recursos, y provee una interfaz de usuario. Todos los teléfonos móviles utilizan algún tipo de sistema operativo, pero en los últimos años con la aparición de los dispositivos inteligentes el sistema operativo de un teléfono se ha convertido en un factor más que determinante.

El crecimiento de la demanda de dispositivos móviles llevó a que varios grandes fabricantes intentaran ganar una porción del mercado desarrollando su propio sistema operativo para móviles. Esto último provocó una fragmentación del mercado [31], acarreando tanto consecuencias positivas como negativas.

Por un lado, fomentó una situación de libre competencia donde cada fabricante intentaba mejorar y pulir sus productos para poder diferenciarse del resto. Esta carrera por atraer consumidores resultó en mejoras a la experiencia de usuario.

Por otro lado, dificultó el desarrollo de aplicaciones móviles, ya que cada plataforma tiene su propio lenguaje nativo, herramientas de desarrollo específicas, y un conjunto de peculiaridades. Esto quiere decir que para apuntar a la totalidad de los usuarios móviles es necesario desarrollar la aplicación en cada una de las plataformas, aumentando el costo del desarrollo.

Actualmente la mayor parte del mercado se divide entre dos sistemas operativos para móviles; estos son Android e iOS. A continuación se presenta una gráfica basada en un estudio de la empresa Gartner acerca de la venta de dispositivos móviles por sistema operativo en el primer trimestre de 2012 [31].

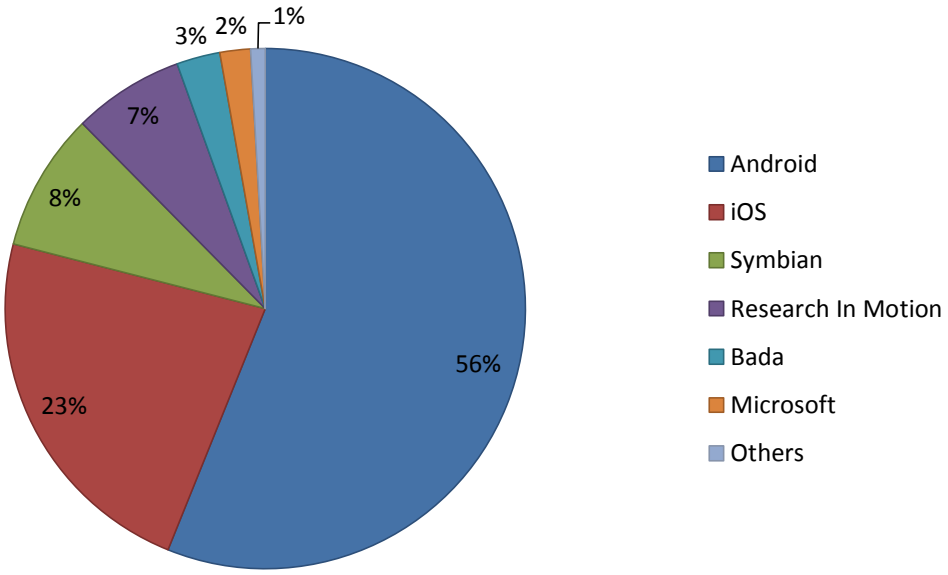


Figura 2.2: Proporción de ventas por sistema operativo (primer trimestre 2012)

2.2.1 Android

Android es una plataforma integral y de código abierto diseñada para dispositivos móviles. Está respaldado por Google, y pertenece a la *Open Handset Alliance* [32], un consorcio de más de 80 compañías líderes en las industrias del hardware y del software creado para "acelerar la innovación en móviles y ofrecer a los consumidores una experiencia móvil más rica, menos costosa, y mejor".

A la hora de diseñar Android se tomaron en cuenta aquellas restricciones que afectaban a los dispositivos móviles, y que probablemente no iban a cambiar en un futuro cercano [33]. En primer lugar, asumieron que el rendimiento de la batería no iba a mejorar demasiado en los próximos años, y en segundo lugar, que el tamaño reducido de los dispositivos móviles limitaba su memoria y capacidad de procesamiento.

El no haber realizado suposiciones con respecto al tamaño de la pantalla, resolución, procesador, y demás componentes del dispositivo, esto es, separar el hardware del software que corre sobre él, permitió que Android pudiera correr sobre una mayor variedad de dispositivos físicos.

Además, el hecho de ser gratuito permitió recortar gastos a las compañías de hardware, lo cual resultó en dispositivos de menor costo. Al ser un sistema operativo de código abierto permitió que miles de desarrolladores en todo el mundo contribuyeran a mejorarlo, y que las compañías de hardware pudieran portarlo fácilmente a su hardware específico. Debido a su licencia Apache/MIT [34] [35] también es posible extenderlo y usarlo para una variedad de propósitos, sin obligación de liberar el código modificado.

Actualmente domina el mercado global de dispositivos móviles con más del 50% [36].

2.2.1.1 *Java y Dalvik*

Java es un lenguaje de programación de alto nivel basado en clases utilizado para propósito general, que soporta el paradigma de orientación a objetos. Este lenguaje toma varios conceptos de C y C++, aunque varía en su organización, ya que toma conceptos de otros lenguajes de programación. Uno de los aspectos en los que varía es en el del manejo implícito de la memoria, ya que Java utiliza un *garbage collector* para liberar la memoria no utilizada o inaccesible.

También podemos decir que es un lenguaje fuertemente tipado, lo que permite distinguir claramente entre errores en tiempo de compilación y errores en tiempo de ejecución. En tiempo de compilación, el código del programa es traducido a código de máquina, y en tiempo de ejecución, se cargan y se vinculan las clases necesarias para ejecutar el programa [37].

A diferencia de otros lenguajes de programación, para los cuales el código del programa es compilado para una arquitectura y un sistema operativo específicos, la ejecución de los programas Java se hace mediante la máquina virtual Java (JVM[20]). Este modo de ejecución permite que los programas puedan ser escritos y compilados una única vez, permitiendo que los mismos se ejecuten en diferentes sistemas operativos y arquitecturas. También presenta ventajas de seguridad, ya que tanto el código potencialmente malicioso, como los errores graves de un programa de una fuente externa, son ejecutados en un entorno seguro (*sandbox*) [38]. La máquina virtual Java sido utilizada para entornos de escritorio (JSE[21]), servidores (JEE[22]) y ecosistemas móviles (JME[23]), con diferentes configuraciones que permitieran soportar múltiples dispositivo.

A pesar de que Google eligió a Java como el lenguaje de programación para Android, descartó el uso de JME y la máquina virtual Java, eligiendo una alternativa diferente llamada Dalvik. También eligió una implementación alternativa y limitada de las bibliotecas estándar de Java, lo cual difiere de la tendencia que seguían otros lenguajes de programación que utilizaban la máquina virtual Java como plataforma de ejecución [39].

Un aspecto relevante a tener en cuenta es que Android fue pensado para trabajar sobre múltiples dispositivos, con diferentes limitantes en cuanto al poder de cómputo, cantidad de memoria RAM, volumen de almacenamiento y autonomía, y sus aplicaciones deben correr en un entorno seguro. Estos requerimientos podrían ser un buen indicio para elegir una máquina virtual como solución, si no fuera por las limitantes anteriormente mencionadas. Justamente por esta razón fue que Google decidió implementar su propia máquina virtual Java, la cual podría brindar solución a los requerimientos de un entorno seguro, teniendo en cuenta el hardware anfitrión [39].

Cada aplicación Android corre su propio proceso en su propia instancia de la máquina virtual Dalvik, la cual está diseñada para correr múltiples instancias de forma eficiente. El hecho de que cada proceso corra su propia máquina virtual ayuda a prevenir que múltiples aplicaciones terminen su ejecución por la falla de una. La máquina virtual Dalvik utiliza un modelo basado

[20] JVM (Java Virtual Machine)
[21] JSE (Java Standard Edition)
[22] JEE (Java Enterprise Edition)
[23] JME (Java Micro Edition)

en registros con instrucciones de 16 bits [40] y ejecuta archivos *DEX*[24], que están optimizados para requerir poca memoria, y son generados a partir de clases Java compiladas (.class) utilizando la herramienta *dx* (incluida en el SDK de Android).

En entornos Java estándares, el código Java es compilado en *Java bytecode*, el cual se almacena en archivos .class que son leídos en tiempo de ejecución por la máquina virtual Java. Cada clase Java es compilada en un único archivo .class. En la plataforma Android, el código Java también es compilado en archivos .class, pero posteriormente se utiliza la herramienta *dx* para transformarlos en archivos .dex, que serán ejecutados en la máquina virtual Dalvik. En la siguiente figura podemos ver la diferencia entre estos dos tipos de archivos:

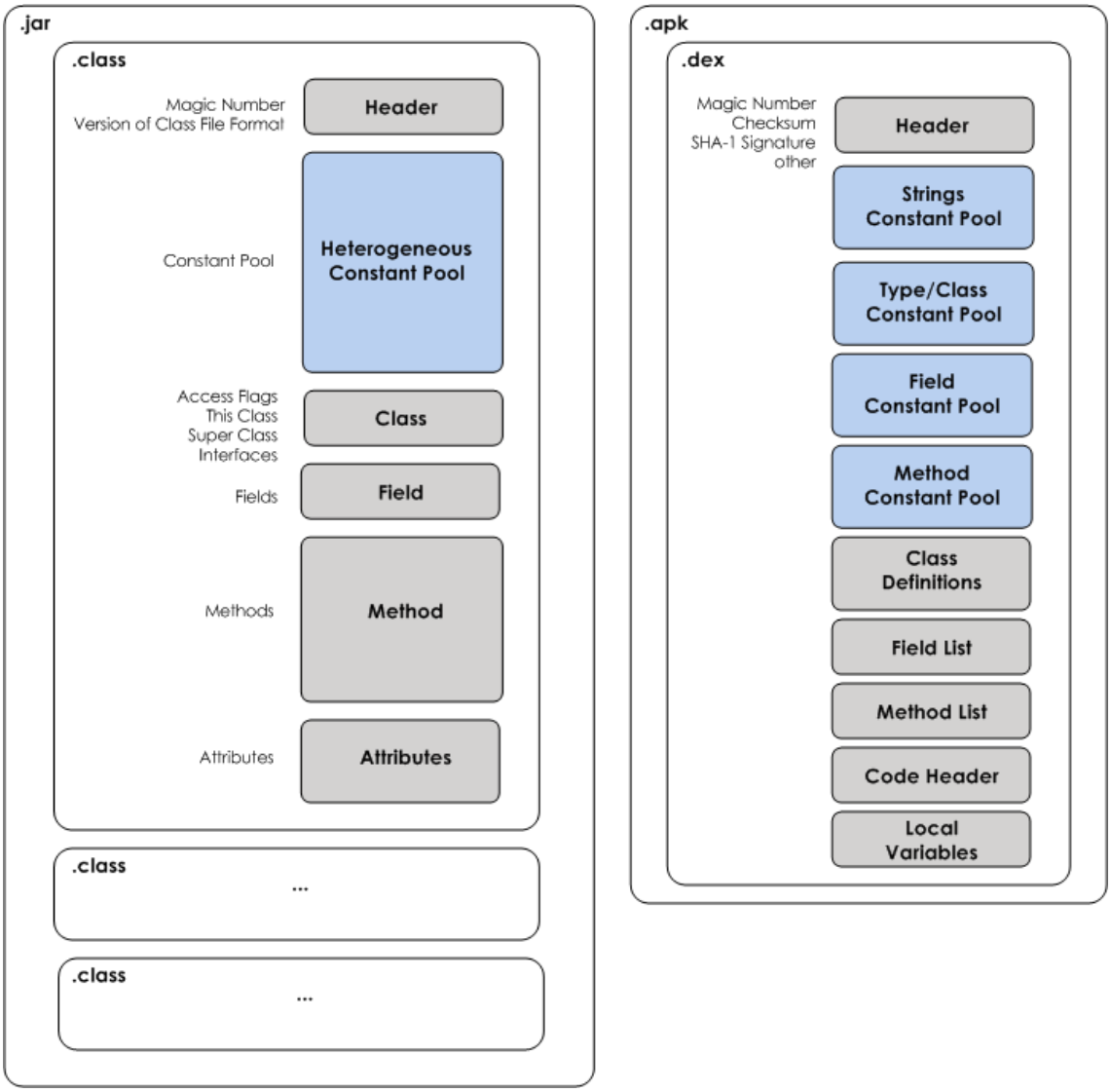


Figura 2.3: Diferencias entre los tipos de archivo .class y .dex

El formato .dex utiliza un espacio compartido para constantes de un tipo específico, en contraposición con el formato .class que tiene un único espacio para constantes. Llamamos constantes a todos los campos, variables, clases, interfaces y nombres de métodos utilizados a lo largo del código. En lugar de tener estas constantes desperdigadas a lo largo del código,

[24] DEX (Dalvik Executable)

estas son referenciadas por un índice relativo a este espacio para un tipo específico. El objetivo principal de discriminar por tipo es el de ahorrar tamaño en memoria, ya que el espacio de constantes representa un 61% del tamaño total de un archivo .class.

Partiendo de que cada proceso instancia su propia máquina virtual Dalvik, es necesario contar con un mecanismo para compartir código entre máquinas virtuales, así como proveer una rápida velocidad de respuesta a la hora de instanciar una nueva máquina virtual. Dalvik utiliza un proceso llamado Zygote que se instancia en el arranque del sistema operativo y permite compartir bibliotecas básicas entre procesos, así como mejorar la velocidad de arranque de una nueva instancia de la máquina virtual Dalvik.

2.2.1.2 *Android SDK*

Android provee todas las herramientas necesarias para desarrollar aplicaciones móviles sobre la plataforma a través del *Android SDK* [41]. Entre ellas se encuentra un emulador de dispositivos que permite probar las aplicaciones sin necesidad de tener un dispositivo físico.

El módulo *SDK Tools* [42] incluye el conjunto completo de las herramientas de desarrollo y depuración, así como otras utilidades que son necesarias para desarrollar una aplicación. También cuenta con el módulo *SDK Platform-tools*, que contiene herramientas dependientes de la plataforma para desarrollar y depurar aplicaciones. Estas herramientas soportan las últimas funcionalidades de la plataforma Android y solo se actualizan cuando hay una nueva plataforma disponible. Vale la pena destacar que las nuevas actualizaciones son compatibles con las plataformas anteriores.

Para cada nueva versión de Android se libera un *SDK Platform*, que incluye una biblioteca totalmente compatible con esa versión del sistema operativo. Para construir una aplicación es necesario especificar una plataforma SDK como objetivo. Cada plataforma ofrece una o más imágenes de sistema (tanto para ARM, como para x86), las cuales son requeridas por el emulador de Android.

2.2.1.3 *Android NDK*

Es un conjunto de herramientas que permite implementar ciertas partes de una aplicación de Android en lenguajes nativos como C y C++. Resulta útil para cierto tipo de aplicaciones que contienen operaciones aisladas que son intensivas en código, o para reutilizar bibliotecas de código ya existentes que están escritas en estos lenguajes [43]. De todas formas, no se recomienda utilizarlo, ya que no presenta grandes beneficios en la mayoría de las aplicaciones. Es necesario analizar los beneficios que pueden proveer, en detrimento de la facilidad de comprensión del código generado.

2.2.2 iOS

iOS es un sistema operativo para dispositivos móviles desarrollado y distribuido por Apple, el cual fue presentado junto con el lanzamiento del primer iPhone [44].

Este sistema operativo fue diseñado específicamente para la arquitectura de hardware de Apple, específicamente para dispositivos móviles como el iPhone, el iPod y el iPad. Si bien esta decisión puede parecer un tanto restrictiva, ya que limita la cantidad de dispositivos sobre las

que puede correr el sistema operativo, la motivación detrás de la misma es asegurar el rendimiento y mejorar la interacción entre el hardware y el software.

2.2.2.1 *Aplicaciones de terceros*

Desde la primer versión del iPhone, iOS soporta aplicaciones creadas con el estándar Web 2.0, las cuales pueden acceder a los servicios del teléfono sin afectar la seguridad y confiabilidad del mismo [45].

A partir de la versión 2.0 del sistema operativo para dispositivos móviles, liberada en Junio del 2008, Apple liberó un SDK para el desarrollo de aplicaciones nativas para iPhone y iPod Touch, el cual provee al desarrollador un buen conjunto de APIs con las cuales interactuar con el hardware del dispositivo. Junto con la versión del software, se agregó una nueva aplicación llamada *Apple Store*, con la que los usuarios pueden navegar, buscar y comprar aplicaciones de terceros [46].

2.2.2.2 *iOS SDK*

El SDK que provee Apple contiene todas las herramientas necesarias para el desarrollo de aplicaciones nativas para iOS. Este SDK también brinda las herramientas para la instalación y prueba de las aplicaciones, para lo que se incluyen varios componentes clave para cumplir con las necesidades del desarrollador.

Provee un IDE llamado *XCode*, con el que se desarrollan las aplicaciones con todo lo que esto implica: edición, compilación, ejecución y pruebas sobre el código escrito [47]. También brinda una herramienta llamada *Instruments*, con la que se pueden hacer análisis de performance y pruebas sobre la aplicación, pudiendo colectar información acerca del comportamiento de la misma en tiempo de ejecución [48]. Para la prueba de las aplicaciones se utiliza el *iOS Simulator*, que simula el sistema operativo iOS y sus funcionalidades. Este simulador no sustituye las pruebas en un dispositivo real, por lo que también es posible configurar un dispositivo para su uso en el desarrollo de aplicaciones [49].

2.2.2.3 *Objective-C y Cocoa Touch*

Objective-C es un lenguaje de programación orientado a objetos, utilizado como lenguaje principal de desarrollo de aplicaciones para OS X e iOS [50]. Es un *superset* del lenguaje C y hereda la sintaxis, los tipos primitivos y el flujo de control de las sentencias, agregando nueva sintaxis para la definición de métodos y clases.

Este es un lenguaje sumamente dinámico, ya que delega la mayor cantidad de definiciones posibles del tiempo de compilación (*compile time*) y de enlace (*link time*) al tiempo de ejecución. Esto hace que no sólo sea necesario un compilador, sino que también un sistema de tiempo de ejecución en el que se ejecute el código compilado [51].

En este momento existen dos versiones del entorno de ejecución: *modern* y *legacy* [52]. La versión *modern* fue lanzada con Objective-C 2.0, la cual incluye múltiples nuevas funcionalidades. Entre las funcionalidades más destacadas se encuentra el cambio en las variables de instancia, las cuales pasaron a ser "*non-fragile*". Esto quiere decir que si se hace un cambio en una variable de instancia de una clase, no es necesario recompilar las clases que heredan de esta.

En el ámbito del paradigma de orientación a objetos, podemos decir que una aplicación es desarrollada en base a una "red de objetos". En este caso, los objetos son instancias de clases programadas en el lenguaje Objective-C, entre las cuales hay un subconjunto que tendremos que desarrollar y otras serán provistas por los frameworks Cocoa o Cocoa Touch [50].

Cocoa Touch es un conjunto de frameworks que proveen una capa de abstracción para el desarrollo de aplicaciones sobre el sistema operativo iOS. Implementan varios patrones de diseño, haciendo un foco especial en aquellas aplicaciones que se ejecutan en interfaces con pantalla táctil. Estos son los que permiten el acceso a los controles de la interfaz gráfica, botones y vistas de pantalla completa [53].

La mayor parte de este framework está escrito en el lenguaje Objective-C e incluye todo lo necesario para el desarrollo de todo tipo de aplicaciones, proveyendo la API para acceder al hardware del dispositivo y dejando abierta la posibilidad de acceder directamente al sistema operativo si es necesario.

Las aplicaciones nativas son desarrolladas con este conjunto de frameworks con el lenguaje Objective-C y son instaladas físicamente en el dispositivo (a diferencia de las aplicaciones Web), lo que hace que estén siempre disponibles (también en modo de vuelo), sin necesidad de conexión a Internet [54].

3 Comunicación y Protocolos

Gran parte del atractivo de los dispositivos móviles proviene de su capacidad de comunicación inalámbrica. Si bien esta capacidad ha ido mejorando con el tiempo, a la hora de construir un sistema y evaluar la viabilidad del mismo es de gran importancia tener en cuenta si la infraestructura de red subyacente es capaz de cumplir sus requerimientos de conectividad.

El sistema planteado en este proyecto tiene requerimientos de *comunicación de tiempo real* [55], lo cual quiere decir que es necesario que todos los usuarios intercambien información de forma instantánea, o con muy baja latencia. En este contexto el término "tiempo real" es un sinónimo de "en vivo".

Este requerimiento entra en conflicto con las limitaciones de batería de los dispositivos móviles actuales ya que incluso un uso moderado de las funciones de red consume mucha energía [56], drenando rápidamente la batería del dispositivo.

En la presente sección se realiza un breve repaso sobre la evolución de la telefonía celular, y se analizan algunos métodos de comunicación en tiempo real que reducen el consumo de batería, permitiendo mejorar la autonomía de los dispositivos al utilizar la aplicación. Por último, se investigarán frameworks y middleware de comunicación disponibles, como una forma de reducir el tiempo de desarrollo del sistema.

3.1 Telefonía Celular

Desde sus inicios a finales de los años '70 [57], la telefonía celular ha evolucionado muy rápidamente, revolucionando drásticamente las actividades que se realizan diariamente. Los teléfonos celulares se han convertido en una herramienta primordial de uso cotidiano.

A pesar que la telefonía celular fue originalmente concebida para transmitir datos de voz debido a las limitaciones tecnológicas de esa época, la tecnología celular de hoy en día es capaz de brindar servicios tales como transmisión de audio, video, y datos en general, convirtiéndose en una de las formas de comunicación más utilizadas.

Las redes celulares cubren grandes áreas geográficas a través de múltiples estaciones base, utilizando métodos de división de la señal para evitar interferencia, tales como CDMA[25] [58]. Cada dispositivo móvil se conecta con una única estación base a la vez, pudiendo cambiar de estación a medida que se mueve. Una vez conectado a una estación base, el dispositivo móvil puede conectarse a otros dispositivos (móviles o estáticos) utilizando la estación base como intermediario.

El núcleo de la red puede ofrecer servicios al dispositivo móvil, dependiendo dichos servicios de la implementación del sistema, según la generación de telefonía móvil.

3.1.1 Generaciones de Telefonía Celular

La telefonía celular, al igual que muchas otras tecnologías, está en continua evolución y son varias las mejoras que se han constatado en pocos años, las cuales han sido potenciadas con la

[25] CDMA (Code Division Multiple Access)

aparición de los teléfonos inteligentes [59]. Estos cambios vertiginosos que esta tecnología experimenta son agrupados por generaciones.

La primera generación de dispositivos móviles refiere a los teléfonos analógicos, introducidos en la década de los '80 [60]. Estos fueron utilizados hasta ser posteriormente remplazados por la segunda generación, la cual fue concebida para el estándar GSM[26] y trabajaba con señales digitales. Con la segunda generación se introdujo el SMS[27], la encriptación digital de las comunicaciones de voz y posteriormente el GPRS[28], que brindaba servicios como el MMS[29] y WAP[30]. Esta generación también dio cabida al desarrollo de EDGE[31] o EGPRS[32] (2.5G) [61].

3.1.1.1 *3G*

Al igual que la segunda generación, la tercera generación o 3G, introduce varias mejoras a la generación anterior. Dentro de las mejoras que brinda 3G se encuentra la transmisión de datos a alta velocidad, con una velocidad mínima de 2 Mbit/s para usuarios a pie (o estáticos) y 384 Kbits/s para los usuarios en vehículos [61] [62] [63].

También brinda mejoras en la transmisión de audio y video, navegación Web y WAP a mayor velocidad y soporte para IPTV[33] [63]. Todos estos nuevos servicios están acompañados de una mejora en la seguridad y privacidad [61].

3.1.1.2 *3.5G/3.75G*

Así como surgieron mejoras menores en las anteriores generaciones, en 3G, estas mejoras resultaron en dos estándares llamados 3.5G y 3.75G. Estas mejoras se debieron a la aparición de HSDPA[34] para el caso de 3.5G, y HSDPA+ y HSUPA[35] para el caso de 3.75G. En el caso de HSDPA y HSDPA+ las velocidades de transferencia de datos alcanzan los 14 Mbits/s [60] [64] y hasta 5.74 Mbits/s para HSUPA [65].

3.1.1.3 *4G*

La creciente utilización de la red de datos móvil no es algo nuevo, ya que las necesidades de los usuarios de hoy son mayores en cuanto a las velocidades de transmisión de datos en dispositivos móviles [66]. La cuarta generación de tecnología de dispositivos móviles brinda mayor velocidad de acceso, pero aún está limitada a determinadas áreas [67].

Hay varios servicios que son llamados 4G, pero la tecnología que hay por detrás no es la misma en todos los casos. Cuando hablamos de 4G nos referimos a dos tecnologías: WiMAX[36] y LTE[37] [67]. Estas tecnologías deberían proveer velocidades de hasta 30-40Mbits/s para WiMAX y

[26] GSM (Global System for Mobile Communications)
[27] SMS (Short Message Service)
[28] GPRS (General Packet Radio Services)
[29] MMS (Multimedia Messaging System)
[30] WAP (Wireless Application Protocol)
[31] EDGE (Enhanced Data rates for GSM of Evolution)
[32] EGPRS (Enhanced GPRS)
[33] IPTV (Internet Protocol Television)
[34] HSDPA (High-Speed Downlink Packet Access)
[35] HSUPA (High-Speed Uplink Packet Access)
[36] WiMAX (Worldwide Interoperability for Microwave Access)
[37] LTE (Long Term Evolution)

hasta 100 Mbits/s para LTE. La realidad dista mucho de estos números, ya que las implementaciones realizadas por Sprint (WiMAX) llegan a un promedio de 3 Mbits/s de bajada y 1.5 Mbits/s de subida, y las realizadas por Verizon (LTE) llegan a un promedio de 9.46 Mbits/s de bajada y 1.35 Mbits/s de subida [68].

3.1.1.4 *Situación actual en Uruguay*

En Uruguay hace más de 10 años que se cuenta con tecnología GSM (2G) que incorporó la tarjeta SIM[38], lo que permitía cambiar el aparato celular conservando el mismo número telefónico. Luego llegó la tecnología GPRS (2.5G) que permitía acceder a internet y la tecnología EDGE (2.75G) que duplicaba las velocidades de la anteriormente mencionadas [69].

Actualmente las tres empresas de telefonía (Antel, Movistar y Claro) proveen (hace ya algunos años) servicios de tercera generación (3G y 3.5G) y a fines del año 2011 la empresa Claro lanzó su producto 4G.

Lo que Claro presentó como 4G es en realidad tecnología HSDPA+ [70] y hay varias controversias acerca de las afirmaciones de la empresa acerca de la clasificación de esta tecnología como 4G. Fernando Leis, director de Marketing de Movistar, afirmó que esto era una confusión y que la tecnología HSDPA+ es considerada de tercera generación [71]. El gerente de Claro, Horacio Alvarellos, afirma que según resolvió la ITU[39], la tecnología HSPA+ se definió como 4G [72], por lo que Claro sigue en su posición de promocionar dicha tecnología como 4G.

Movistar también lanzó su servicio basado en HSPA+, llamado Internet Fácil Turbo [73]. Este servicio no es presentado como 4G, porque Movistar no considera esta tecnología dentro de la cuarta generación [69].

Antel, por su parte, el 12 de diciembre de 2011 presentó Internet Vera, basada en tecnología de cuarta generación LTE. En Antel se desarrollaron planes piloto y de prueba en centrales de Montevideo y Punta del Este [74]. Antel afirma que dependiendo de las condiciones operativas, se pueden alcanzar velocidades de 20 Mbits/s, 50 Mbits/s, 100 Mbits/s e incluso superarlas [75].

Hasta el momento, en Uruguay no está operativa esta tecnología ya que la URSEC[40] debe liberar y asignar la frecuencia de 700 Mhz a las empresas de telefonía, la cual se utiliza en la implementación de LTE [71].

3.2 Message-Oriented Middleware (MOM)

Cuando se habla de *message-oriented middleware* se está refiriendo a una infraestructura de software o hardware que provee comunicación distribuida en base a un modelo de interacción asíncrona. Los participantes de un sistema basado en MOM [76] interactúan sin tener que bloquearse y esperar al enviar un mensaje, pueden continuar su ejecución luego del envío. Esto permite que se envíen mensajes cuando el emisor o el receptor no están activos o disponibles durante el tiempo de la ejecución. A raíz de esto, una aplicación que envía un

[38] SIM (Subscriber Identity Module)
[39] ITU (International Telecommunication Union)
[40] URSEC (Unidad Reguladora de Servicios de Comunicaciones)

mensaje no tiene garantía de que el mensaje será leído por otra aplicación, ni recibe garantía alguna sobre el tiempo que demorará en ser entregado. Estos aspectos dependen principalmente de la aplicación que recibe.

3.2.1 Características

Por lo general, los MOM desarrollados para interactuar con múltiples plataformas proveen una API común a todos los sistemas que soportan. El resultado es una capa distribuida de comunicación que hace posible distribuir los módulos de aplicación sobre plataformas heterogéneas, pero permitiéndole al desarrollador abstraerse de los detalles de las mismas. A continuación se presentan las cuatro características más importantes.

3.2.1.1 *Bajo acoplamiento*

MOM inyecta una capa entre emisores y receptores, lo cual permite que los mismos utilicen esta capa independiente como un intermediario para intercambiar mensajes. Uno de los beneficios más importantes de MOM es el bajo acoplamiento entre los participantes del sistema, esto es, la habilidad de conectar aplicaciones sin tener que adaptar una en base a la otra.

3.2.1.2 *Alta confiabilidad*

Se previene la pérdida de mensajes por fallas de la red o del sistema utilizando un mecanismo *store-and-forward* para la persistencia de los mensajes (al recibir un mensaje el intermediario almacena el mensaje para entregarlo al receptor cuando sea conveniente). Esta capacidad brinda un alto nivel de confiabilidad en el mecanismo de distribución, previniendo la pérdida de mensajes cuando partes del sistema están ocupadas o no disponibles. Generalmente el nivel de confiabilidad es configurable, pero los sistemas MOM son capaces de asegurar que cada mensaje será entregado exactamente una vez a cada uno de los receptores deseados.

3.2.1.3 *Escalabilidad*

Además de desacoplar la interacción de los subsistemas, también desacopla las características de rendimiento de cada subsistema del resto. Los subsistemas pueden escalar de forma independiente sin afectar al resto. También permite que el sistema pueda sobrellevar períodos de alta actividad en un subsistema sin afectar otras áreas del mismo. El modelo MOM tiene ciertos rasgos que ayudan a balancear la carga de forma simple y efectiva, como permitir que un subsistema acepte un mensaje cuando está listo en lugar de forzarlo a recibirlo.

3.2.1.4 *Alta disponibilidad*

Debido al modelo asíncrono, no requiere que todos los subsistemas estén disponibles al mismo tiempo. La falla de un subsistema no causará fallas en todo el sistema. Además el bajo acoplamiento entre los subsistemas puede ayudar a reducir el tiempo de finalización de un proceso y mejorar la capacidad de respuesta y la disponibilidad global del sistema.

3.3 Tecnología *Push*

El término se refiere al estilo de comunicación a través de Internet donde el servidor es quien realiza la petición para iniciar una transacción. Esto se contrasta con la forma típica, en la que el cliente es quien realiza el pedido para iniciar la transmisión de información [77] [78].

A menudo los servicios *Push* están basados en un modelo publicar-suscribir, por lo que se conocen las preferencias del cliente en cuanto a la información que desea recibir. Un cliente puede *suscribirse* a diferentes *canales* de información. Cuando un canal tiene nuevo contenido disponible, el servidor envía la información al usuario.

Algunos ejemplos de servicios *Push* son:

- Mensajería instantánea y *chat rooms* (XMPP[41], IRC[42])
- Conferencias de audio y video en tiempo real

Otros usos se dan en aplicaciones web que tienen requerimientos de tiempo real, como sitios de juegos y apuestas en línea, subastas, resultados de deporte y monitoreo de sensores, entre otros.

3.3.1 Implementación

Para implementar la comunicación *Push* existen diferentes opciones, estando basadas la mayoría en el protocolo HTTP[43]. A continuación se presentan las implementaciones más utilizadas.

3.3.1.1 *HTTP streaming*

Es un mecanismo que permite enviar datos desde un servidor web a un navegador. Esto se puede lograr a través de distintos mecanismos. Por lo general el servidor no finaliza la conexión luego de que se ha enviado la respuesta al cliente. De esta forma cuando ocurre un evento se puede notificar inmediatamente al cliente, en lugar de almacenar la información hasta que el cliente haga una nueva petición [79].

3.3.1.2 *Pushlet*

El servidor toma ventaja de las conexiones HTTP persistentes y deja la respuesta sin terminar, dejando al navegador en un continuo estado de "carga". Luego envía periódicamente rutinas de JavaScript para actualizar el contenido de la página, realizando así el *push*. Una importante desventaja de este método es la falta de control que tiene el servidor sobre el timeout del navegador, si ocurre un timeout es necesario recargar la página [80].

3.3.1.3 *Long polling*

Es una técnica que permite emular la tecnología *Push*. Al igual que en la técnica de polling tradicional, el cliente realiza un pedido de información. La diferencia radica en que el servidor aguarda a tener información disponible antes de enviar una respuesta al cliente. Por lo general luego de recibir una respuesta el cliente volverá a realizar un pedido de información, de forma tal que la mayor parte del tiempo el servidor tenga una petición pendiente que pueda usar para enviar información en respuesta a un evento [81].

[41] XMPP (Extensible Messaging and Presence Protocol)
[42] IRC (Internet Relay Chat)
[43] HTTP (Hypertext Transfer Protocol)

3.3.1.4 *BOSH (Bidirectional-streams Over Synchronous HTTP)*

Es un protocolo de transporte que emula la semántica de una conexión TCP bidireccional de larga duración entre dos entidades (como cliente y servidor). Esto se lleva a cabo utilizando eficientemente varios pares de petición/respuesta de HTTP sin requerir polling frecuente o respuestas parciales. Es más eficiente en ancho de banda que otras técnicas bidireccionales basadas en HTTP, y que *Ajax* [82].

3.3.1.5 *Web Sockets*

WebSockets es una técnica de comunicación bidireccional sobre un socket TCP, permite el envío de mensajes con baja latencia y alta frecuencia. Esto facilita la comunicación en tiempo real entre cliente y servidor. Al ser bidireccional provee soporte directo para *Push,* lo cual combinado con la baja latencia en el envío de mensajes haría que desplace a otras técnicas como *long polling* y *BOSH*. Aún no es un estándar, pero varios de los grandes navegadores soportan WebSockets [83].

3.3.2 Push en Dispositivos Móviles

Las dos grandes plataformas para móviles de la actualidad, Android e iOS, cuentan con soporte para realizar *Push* a estos dispositivos, provistos respectivamente por Google y Apple. La utilización de estos servicios puede ayudar a reducir drásticamente el consumo de batería ya que se reutiliza una misma conexión para todas las aplicaciones del dispositivo. Estos servicios se pueden catalogar como MOM, ya que manejan todos los aspectos de encolamiento y entrega del mensaje.

3.3.2.1 *Push en Android*

Google Cloud Messaging for Android (GCM) es un servicio que permite que los desarrolladores envíen datos desde servidores a sus aplicaciones de Android. Esto podría ser un mensaje liviano indicando que hay una nueva actualización disponible, o un mensaje con hasta 4kb de contenido, para que aplicaciones como las de mensajería instantánea puedan consumirlo directamente [84].

3.3.2.2 *Push en iOS*

Apple Push Notification service (APNs) es un servicio que provee una forma eficiente de propagar información a dispositivos de Apple (iPhone, iPad, iPod). Cada dispositivo establece una conexión encriptada con el servicio y recibe notificaciones a través de esta conexión persistente. Si llega una notificación para una aplicación cuando ésta se encuentra inactiva, el dispositivo alerta al usuario de que hay datos disponibles para la misma. Los desarrolladores originan las notificaciones desde el software de sus servidores [85].

3.3.3 Kaazing WebSocket Gateway – HTML5 Edition

Este producto de Kaazing [86] provee emulación de la capa de transporte de WebSocket, permitiendo a los desarrolladores trabajar de forma transparente escribiendo código para las API estándar de WebSocket, sin importar si la plataforma destino soporta WebSocket o no. De esta forma hace posible implementar protocolos estándar o propietarios que utilicen

WebSocket como capa de transporte. También provee soporte directo para protocolos como JMS[44], AMQP[45], y XMPP.

Soporta todos los navegadores más utilizados, utilizando WebSocket de forma nativa cuando está soportado, y brindando emulación de WebSocket en el caso contrario. Además de los navegadores soporta tecnologías adicionales, como Java, .NET y Flash.

La licencia de desarrollador permite mantener hasta 50 conexiones simultáneas de forma gratuita.

3.4 MQTT (Message Queue Telemetry Transport)

Otro protocolo que posibilita la comunicación entre dispositivos móviles es MQTT. Este es un protocolo de mensajería simple y liviano que ofrece funcionalidad de publicar/suscribir y ha sido diseñado específicamente para dispositivos con restricciones energéticas y redes poco confiables, con alta latencia, o ancho de banda mínimo. Estos criterios de diseño hacen que MQTT esté ganando popularidad para la comunicación máquina a máquina (M2M), o la interconexión de dispositivos (“Internet de las cosas”) [87].

El protocolo define tres niveles de calidad de servicio (QoS[46]), permitiendo así proveer características de mensajería tradicional cuando las condiciones lo permiten.

- **QoS 0 – A lo sumo una entrega:** El *broker* o cliente enviará el mensaje una vez, sin confirmación. Esto corresponde a enviarlo de acuerdo al mejor esfuerzo de la red.
- **QoS 1 – Al menos una entrega:** El *broker* o cliente intentará entregar el mensaje al menos una vez (hasta recibir una confirmación). Puede haber duplicados.
- **QoS 2 – Exactamente una entrega:** El *broker* o cliente enviará el mensaje exactamente una vez, utilizando un acuerdo de cuatro pasos.

Se lo utiliza a menudo en sistemas embebidos y aplicaciones móviles, donde se requiere bajo uso de memoria, y el acceso a la red es limitado. También es útil para conexiones con lugares remotos.

3.5 Node.JS

Node es una plataforma que pretende facilitar el desarrollo de software, específicamente el orientado a resolver problemas de redes de computadoras, que necesiten escalar con facilidad en cortos períodos de tiempo [88]. Utiliza JavaScript como lenguaje de scripting[47], utilizando un modelo basado en eventos no bloqueantes.

Esta plataforma toma conceptos del manejo de eventos de bibliotecas como Event Machine de Ruby [89], o Twisted de Python [90], aunque difiere levemente en el manejo de los mismos, ya que no utiliza llamadas bloqueantes a los procedimientos que atienden el evento. En general se declaran los procedimientos que atienden a los eventos para luego iniciar de forma explícita el ciclo de eventos. Esto no es así en el caso de Node, ya que no hace nada similar al inicio del

[44] JMS (Java Message Service)
[45] AMQP (Advanced Message Queuing Protocol)
[46] QoS (Quality of Service)
[47] Lenguaje de Scripting: Lenguaje de programación que soporta la escritura de scripts, los cuales son escritos para un entorno particular y son utilizados para automatizar determinadas tareas.

ciclo de eventos. Node entra en el ciclo de eventos luego de procesar la entrada y sale automáticamente cuando no hay más eventos para procesar.

Las características antedichas hacen que Node sea adecuado para entornos distribuidos y aplicaciones que intercambian grandes volúmenes de datos en tiempo real [91]. Frecuentemente es utilizado para el desarrollo de servidores web, en el que múltiples clientes pueden ser atendidos de forma concurrente. La infraestructura de Node basada en eventos es una buena base para el desarrollo de este tipo de sistemas.

3.5.1 Socket.IO

Node.JS resulta de especial interés para el proyecto debido a Socket.IO, una biblioteca para Node.JS que hace posible la implementación de WebSockets en todos los navegadores [92]. Es una API de WebSockets desarrollada en JavaScript, utilizada para el desarrollo de aplicaciones de tiempo real, la cual permite abstraerse del mecanismo de transporte utilizado por cada aplicación.

Esto quiere decir que el desarrollador no debe preocuparse de si la conexión será establecida mediante WebSocket, *XHR polling*, Flash u otro mecanismo, dado que Socket.IO detecta los mecanismos soportados por el entorno y selecciona el más conveniente, o el de mayor preferencia según los especificados por el desarrollador [93] [94].

Socket.IO va más allá de lo que sería un *polyfill*[48] de WebSockets, ya que provee ciertas funcionalidades que no se encuentran en la API de WebSockets, como la verificación de conexión (*heartbeat messages*), soporte de desconexión y timeouts [95].

Los mecanismos soportados por Socket.IO son los siguientes: WebSocket, Adobe Flash Socket, *XHR polling, XHR multipart streaming*, *forever iframe*, y *JSONP polling* [96].

[48] Polyfill: Una biblioteca que imita una API existente, y cuenta con diferentes implementaciones para brindar la misma funcionalidad cuando no es posible utilizar alguna de ellas.

4 Desarrollo de Aplicaciones

Como se mencionó anteriormente, el mercado de dispositivos móviles se encuentra fragmentado, cada plataforma tiene su propio lenguaje nativo, herramientas de desarrollo específicas, y un paradigma de diseño particular.

Esto implica que para desarrollar una aplicación móvil que pueda ser utilizada por la totalidad de los usuarios móviles es necesario desarrollar una aplicación para cada una de las plataformas, lo cual hace que el costo del desarrollo sea más elevado.

Por este motivo, a la hora de construir una aplicación a menudo se debe decidir para qué subconjunto de plataformas se desarrollará la aplicación, lo cual reduce el costo de desarrollo pero también reduce el mercado potencial de la misma. Es conveniente, entonces, encontrar alternativas al desarrollo nativo de aplicaciones, que permitan desarrollar aplicaciones multiplataforma, utilizando lenguajes y tecnologías estándar.

A continuación se presentan las opciones de desarrollo móvil disponibles en la actualidad, mencionando también tecnologías de desarrollo web, y finalmente, tecnologías y herramientas de desarrollo híbrido que permiten crear aplicaciones que funcionan en múltiples plataformas.

4.1 Desarrollo Nativo

Una aplicación móvil nativa es un programa para teléfonos inteligentes que ha sido desarrollado para un ambiente de ejecución específico, por ejemplo, el ambiente Objective-C en iOS. Al correr directamente sobre el sistema operativo, las aplicaciones nativas pueden hacer uso de características de hardware y software específicas del dispositivo.

Entre otras cosas, las aplicaciones nativas pueden utilizar los componentes de interfaz que provee cada plataforma, interactuar con otras aplicaciones del dispositivo como el calendario y la agenda de contactos, y permiten tener un control más preciso sobre las funcionalidades de bajo nivel del dispositivo, como utilización del GPS, acceso a red, gestos, y control del consumo de batería. De esta forma es posible obtener un mejor rendimiento, y brindar una mejor experiencia de usuario.

Actualmente las plataformas de dispositivos móviles más populares para desarrollo nativo son Android e iOS. Cada plataforma tiene sus propias convenciones de usabilidad en los controles, la navegación, y todo lo que involucra el diseño de interacción [97] [98]. Esto es muy bueno para los usuarios, ya que esa consistencia entre las aplicaciones de la plataforma ayuda a que tengan menor dificultad a la hora de interactuar con una nueva aplicación; los gestos utilizados, las opciones de menú, y otros componentes de la interfaz se comportan de forma predecible.

Debido a que una aplicación nativa está desarrollada para un ambiente particular, para que funcione en más de una plataforma es necesario portar la implementación. Esto significa que al desarrollar aplicaciones nativas, el costo varía en función del número de plataformas soportadas, por lo cual es necesario implementar cada funcionalidad de forma específica para cada una de ellas.

Todas las plataformas móviles cuentan con una tienda de aplicaciones, las cuales permiten que los usuarios descubran y adquieran nuevas aplicaciones para sus dispositivos. Estas resultan de

gran importancia ya que además de distribuir las aplicaciones, permiten monetizarlas mediante mecanismos de venta o suscripción. Muchas de las tiendas cuentan con un proceso de selección de aplicaciones [99] [100], mediante el cual controlan que estas cumplan con los estándares de calidad, y las políticas impuestas por la tienda.

Existen ciertas herramientas que permiten reutilizar parte de la lógica entre los portes de la aplicación, como el acceso a datos y conexiones de red. Esto se logra mediante la utilización de un lenguaje común para definir las operaciones, que luego es traducido a código nativo para cada plataforma. Dentro de esta categoría se encuentra la herramienta Genexus, que se describe a continuación.

4.1.1 GeneXus

GeneXus es una herramienta desarrollada por la empresa Artech [101], la cual automatiza algunos procesos del desarrollo de software, lo que permite que el analista pueda enfocarse en comprender la realidad a modelar y los problemas del usuario, delegando gran parte de la codificación a la herramienta.

GeneXus automatiza aquellas tareas rutinarias, como son: normalización de los datos, mantenimiento y generación de la estructura de las bases de datos y de los programas de aplicación [102]. En cuanto a lo que a programas de aplicación se refiere, GeneXus genera automáticamente los que implementan las operaciones de ABM[49] y permite aplicar algunos patrones de diseño que facilitan la construcción de programas de consulta.

4.1.1.1 *Características*

GeneXus utiliza una filosofía incremental y se basa en un desarrollo basado en el conocimiento. Hablando en cuestiones de desarrollo de software, el analista deberá “capturar el conocimiento” del usuario e incorporar el mismo en la base de conocimiento de GeneXus. La herramienta se encargará de realizar transformaciones del “conocimiento” representado con objetos GeneXus, lo cual permitirá generar los programas y la base de datos, e impactar los cambios de la visión de los usuarios en los mismos de forma automática.

Entre algunas de las principales ventajas de las características anteriormente mencionadas, se destaca la generación de la base de datos en tercera forma normal, la cual es realizada automáticamente por la herramienta. También permite especificar el diseño en forma abstracta, lo que nos brinda la posibilidad de generar programas para diferentes plataformas a partir de una misma base de conocimiento, utilizando el manejador de base de datos preferido por el analista [103].

4.1.1.2 *Desarrollo para Dispositivos Móviles*

Como se mencionó anteriormente, GeneXus parte del conocimiento del usuario y de la representación del mismo en la base de conocimiento, con lo que luego generará los programas. Debido al creciente uso de los dispositivos móviles, Artech ha desarrollado un motor que permite generar programas para plataformas móviles, a saberse Android, iOS y BlackBerry [103].

[49] ABM (Alta, baja y modificación)

Entre las características que brinda GeneXus para el desarrollo de aplicaciones para dispositivos móviles se encuentran:

- Desarrollo de aplicaciones en múltiples plataformas, creándolas de forma genérica y pudiendo instalarlas en cualquiera de las plataformas mencionadas.
- Creación de aplicaciones nativas en los lenguajes de programación de cada plataforma: Java para Android, Objective-C en iOS y Java para BlackBerry.
- Integración con proyectos GeneXus existentes.
- Publicación de la aplicación en la nube.
- Pruebas de la aplicación con simuladores nativos.
- Publicación de la aplicación en los Marketplace de cada compañía (AppStore, Google Play y BlackBerry AppWorld).
- Interacción con el hardware del dispositivo, como son la cámara, GPS, teléfono, entre otros.
- Integración con redes sociales, como Facebook o Twitter [104].

4.2 Desarrollo Web

Otra opción para el desarrollo de aplicaciones móviles es el desarrollo web. A diferencia de las aplicaciones nativas, estas aplicaciones son ejecutadas dentro de un navegador en el dispositivo móvil, y se desarrollan utilizando tecnologías web, tales como HTML[50], CSS[51], y JavaScript.

HTML es uno de los lenguajes utilizados para el desarrollo de aplicaciones móviles multiplataforma. La revisión más reciente es la cinco y apunta a proveer herramientas útiles para crear aplicaciones web complejas, y manejar contenido multimedia, así como crear documentos con contenido semántico. A la fecha es aún un *working draft de la W3C* [105], pero en los últimos años ha ido ganando soporte de los navegadores y se está utilizando tanto para desarrollo web como de aplicaciones móviles.

Toma HTML 4.01 como punto de partida, y tiene en cuenta que los documentos HTML presentes en la web son una mezcla de varias especificaciones, como las introducidas por los navegadores y por prácticas comunes, y que contienen muchos errores de sintaxis. Por ello define el procesamiento requerido para los documentos inválidos, de forma tal que los errores de sintaxis sean tratados uniformemente por todos los navegadores que se apeguen al estándar. Introduce varias APIs[52] y nuevas etiquetas que permiten desarrollar aplicaciones web complejas. Por este motivo se está perfilando como un candidato potencial para aplicaciones móviles multi-plataforma [106]. Algunas de estas APIs son:

- Dibujo en dos dimensiones
- Reproducción de archivos de audio y video
- Aplicaciones web que funcionen cuando no se tiene conexión
- Registro de aplicaciones para ciertos protocolos
- Edición de documentos
- Funcionalidad *drag and drop*
- Facilitar la navegación hacia atrás, exponiendo el historial
- Mensajería entre documentos

En muchos casos se utiliza el término HTML5 para agrupar nuevas tecnologías de la web, muchas de las cuales no forman parte de la especificación, generalmente relacionadas con una experiencia web superior [107]. Algunos ejemplos de estas tecnologías son CSS3 [108], la Geolocation API [109], y WebSockets [83].

4.2.1 jQuery Mobile

Dentro de los frameworks utilizados para el desarrollo de aplicaciones móviles en HTML5 encontramos a jQuery, que es una biblioteca JavaScript que propone simplificar la interacción con los documentos HTML y el objeto DOM, manejo de eventos, creación de efectos y animaciones, y el uso de Ajax, mediante una API soportada por múltiples navegadores [110].

[50] HTML (Hypertext Markup Language)
[51] CSS (Cascading Style Sheets)
[52] API (Application Programming Interface)

jQuery Mobile es un framework utilizado para el desarrollo de interfaces de usuario construido sobre jQuery, que es compatible con la mayoría de las plataformas para dispositivos móviles y de escritorio. Utiliza las marcas del lenguaje HTML5 para facilitar el aprendizaje, brindando la posibilidad de personalizar la biblioteca a través de la API proporcionada [111].

Las páginas son la unidad de interacción con el framework, las cuales agrupan conceptos lógicos en una única vista. Se introducen los conceptos de *page* y *dialog*, que son contenedores de páginas utilizados para el sistema de navegación entre páginas. La diferencia entre ambos elementos radica en que los de tipo *dialog* hacen que la página tenga un estilo de página modal, que se superpone a la vista de la página principal [112].

El sistema de navegación del que se habló anteriormente utiliza Ajax para cargar las páginas directamente en el objeto DOM, las que posteriormente son desplegadas utilizando animaciones. El uso de Ajax evita que se realicen consultas innecesarias al servidor, brindando una mejor experiencia de usuario y minimizando los tiempos de respuesta. El framework automáticamente reimplementa la navegación de los links y envío de formularios, transformándolos en consultas Ajax [113].

Para la implementación de las vistas se pueden utilizar elementos HTML estándar, aunque jQuery Mobile incluye varios *widgets* amigables para interfaces táctiles, entre los que se encuentran: botones, cuadros de diálogo, elementos de formulario, entre otros.

4.2.2 Sencha Touch

Sencha Touch [114] es un framework de desarrollo de aplicaciones móviles HTML5, que permite desarrollar aplicaciones compatibles con los sistemas operativos iOS, Android, BlackBerry y Windows Phone, entre otros. Está desarrollado con los últimos estándares de HTML5 y CSS3, y las aplicaciones pueden ser accedidas como aplicaciones web o portarse para los diferentes sistemas operativos mencionados anteriormente mediante el uso de la herramienta Sencha Cmd [115].

Está construido sobre el sistema de clases del framework ExtJS 4, el cual brinda todo el potencial de la carga dinámica, manejo de herencia, carga de dependencias y *mixins* (permite incluir el código de una clase en otra) [116] [117]. Este sistema de clases facilita el desarrollo orientado a objetos sobre JavaScript, utilizando una arquitectura MVCS[53].

En lo que refiere a la interfaz de usuario, Sencha Touch soporta nativamente los eventos táctiles *tap*[54], *swipe*[55] y *pinch*[56]. Cuenta con animaciones fluidas y soporte de scroll, el cual es implementado mediante diferentes mecanismos en cada dispositivo, intentando lograr la mejor experiencia de uso en cada plataforma. También provee un motor de renderizado que permite cargar las aplicaciones de forma instantánea, el cual detecta rápidamente la posición del dispositivo, adaptando la interfaz a la misma.

---

[53] MVCS (Model-View-Controller-Store)

[54] Evento capturado cuando se hace contacto en un único punto entre el usuario y el dispositivo con interfaz sensible al tacto [185].

[55] Evento capturado cuando se hace un *tap*, para luego arrastrar el objeto con el cual se establece el contacto con la superficie sensible al tacto.

[56] Evento capturado cuando el usuario “pellizca” la superficie sensible al tacto.

Sencha Touch provee un componente básico llamado *Navigation View*, que es utilizado para mostrar las diferentes vistas desarrolladas, las cuales se apilan en un *stack* [118]. Mediante esta estructura se implementa el historial y la creación de nuevas instancias de las vistas.

4.3 Desarrollo Híbrido

Como se vio anteriormente, es posible desarrollar una aplicación nativa por cada plataforma, o crear una aplicación web con una experiencia de usuario más limitada o minimalista que funcione en varias plataformas. Existe una tercera opción, que consiste en combinar ambos enfoques en una aplicación híbrida.

Las aplicaciones móviles híbridas son una combinación de vistas web y componentes nativos, por lo que al igual que las aplicaciones web permiten reutilizar la interfaz en los portes de la aplicación para cada plataforma, pero pudiendo acceder a características del dispositivo que no se encuentran disponibles en el navegador, como lo haría una aplicación nativa.

Si bien esto permite reducir los costos de desarrollo, como las convenciones de usabilidad y diseño son diferentes para cada plataforma, es imposible reutilizar toda la interfaz con buenos resultados, ya que el usuario espera que cada aplicación se comporte de forma similar a las aplicaciones nativas. Si se utilizan patrones de diseño ajenos a una plataforma, la aplicación podría resultar visualmente inconsistente o difícil de utilizar. A la hora de desarrollar aplicaciones híbridas, es importante que las vistas reutilizadas no diverjan del paradigma de diseño de las plataformas soportadas.

Las aplicaciones híbridas se instalan en el dispositivo como aplicaciones nativas, por lo cual permiten aprovechar los servicios que brindan las tiendas de aplicaciones, facilitando su descubrimiento y monetización.

La mayoría de las aplicaciones híbridas se desarrollan utilizando *frameworks* específicos que permiten combinar componentes nativos y vistas web con facilidad. Por lo general, estos *frameworks* proveen una API de JavaScript que permite acceder a las características del dispositivo.

4.3.1 PhoneGap

PhoneGap [119] es un *framework* de código abierto que permite construir aplicaciones móviles multi-plataforma utilizando tecnologías web como HTML, JavaScript y CSS. Soporta un gran número de sistemas operativos, entre los que se encuentran Android, iOS, y Windows Phone.

Las aplicaciones construidas con PhoneGap son híbridas, ya que se realiza el *renderizado* gráfico a través de un motor web en lugar de utilizar las funcionalidades de interfaz gráfica nativas del sistema operativo, pero se compilan e instalan como aplicaciones nativas.

PhoneGap provee un conjunto de APIs que forman una capa de abstracción sobre las características físicas del dispositivo y su sistema operativo, permitiendo acceder a características específicas del hardware y la plataforma. Esto permite desarrollar aplicaciones nativas para todos los sistemas operativos soportados utilizando el mismo código de base.

Algunas de las funciones soportadas mediante la API de PhoneGap son: geolocalización, agenda de contactos, acceso al sistema de archivos, acelerómetro, cámara, brújula, reproducción y grabación multimedia, notificaciones, y almacenamiento.

4.3.1.1 *PhoneGap Build*

PhoneGap dispone de un servicio online de compilación para múltiples plataformas llamado PhoneGap Build [120]. El servicio permite subir los archivos HTML5, CSS, y JavaScript que conforman la aplicación, encargándose de compilarlo para cada una de las plataformas soportadas.

Una de las ventajas de este servicio es que permite obtener los ejecutables sin necesidad de disponer de los SDKs de cada plataforma; una vez finalizada la compilación es posible descargar las aplicaciones obtenidas desde la página. También es posible acceder a la funcionalidad provista por la API de PhoneGap Build mediante Web Services.

Otra característica es que permite probar y modificar las aplicaciones en tiempo de ejecución, de forma similar a las funciones de *inspección* presentes en Firebug [121] y Google Chrome [122], las cuales resultan herramientas muy útiles para el desarrollo de aplicaciones web.

4.3.1.2 *Apache Cordova*

El código de PhoneGap fue donado a la Apache Software Foundation para empezar un nuevo proyecto, que se denominó Apache Cordova [123], por lo que continuará siendo gratuito y de código abierto, bajo la licencia Apache.

5 Software Geográfico

El sistema planteado requiere consultar y analizar información geográfica a partir de la ubicación de los usuarios y los ómnibus, por lo que será necesario evaluar software que provea funcionalidades como servicios de mapas y análisis de ruta.

Se requerirá un componente de mapas que permita mostrar ubicaciones, recorridos, y otros elementos como paradas, de forma dinámica y visualmente agradable. También será necesario contar con un servicio de análisis de rutas que permita obtener el mejor camino para ir a un destino y que realice estimaciones de tiempo.

Por último, será necesario evaluar diferentes opciones a la hora de almacenar los datos geográficos involucrados, tales como los recorridos del usuario, y las paradas y recorridos de ómnibus.

A continuación se presentan los productos y servicios geográficos evaluados, haciendo énfasis en los requerimientos específicos del proyecto.

5.1 Sistemas de Información Geográfica

Un Sistema de Información Geográfica es un sistema utilizado para administrar, analizar y desplegar información geográfica [124]. La información geográfica es representada mediante estructuras simples que modelan la geografía y el SIG[57] provee las herramientas necesarias para trabajar con esta información.

Puede utilizarse un SIG con tres enfoques diferentes, dependiendo de cómo se trabaje con la información. El primer enfoque está orientado a los datos, en donde estos se representan en una base de datos espacial, utilizando estructuras específicas. El segundo refiere a la visualización de estos datos en mapas inteligentes, los cuales permiten desplegar información geográfica, características de los diferentes elementos y las relaciones entre ellos. El tercer enfoque involucra al procesamiento de la información geográfica, en la que se aplican funciones de transformación al conjunto de datos existente en la base de datos espacial con el fin de generar datos nuevos.

5.2 ESRI ArcGIS

ArcGIS es una integración de productos SIG desarrollada por ESRI [125] que permite representar datos geográficos, y brinda las herramientas necesarias para diseñar, crear y trabajar con ellos. La información geográfica es representada por conjuntos de datos geográficos que el SIG permitirá manipular, utilizando herramientas desarrolladas para este fin.

5.2.1 Características generales

La suite de ArcGIS provee soluciones para la Web, dispositivos móviles y computadoras de escritorio, así como servicios online en la nube o para empresas [126]. Hay varias prestaciones que esta provee, de las cuales enumeramos las siguientes:

[57] SIG (Sistemas de Información Geográfica)

- Consulta y edición de información geográfica.
- Creación de aplicaciones web de cartografía.
- Creación de base de datos geográfica.
- Publicación de datos y mapas en la nube, lo cual permite que los usuarios finales accedan a los mismos desde una gran variedad de dispositivos, obteniendo información actualizada en cualquier momento.
- Acceso a aplicaciones y mapas creados por terceros, las cuales podrán ser incorporados a los mapas propios.
- Entorno de trabajo colaborativo, con el cual se podrá trabajar conjuntamente con otros usuarios.
- Creación de aplicaciones personalizadas utilizando la API y el SDK de ArcGIS.

5.2.2 ArcGIS Server

Luego de crear los mapas y herramientas del SIG, es posible publicarlos mediante ArcGIS Server, lo que permite que el usuario pueda utilizarlas en cualquier lugar mediante servicios Web. Esta herramienta permite centralizar la información en un único lugar y brindar acceso multitudinario a la misma [127].

ArcGIS Server provee varios servicios [128] que pueden ser utilizados a la hora de desarrollar aplicaciones. Algunos de estos servicios son:

- Representación cartográfica. Es un servicio dinámico de mapas que será publicado en la Web.
- Geocodificación. Permite buscar una dirección y obtener sus coordenadas.
- Geodatos. Administración de base de datos espaciales mediante consultas, edición y acceso a los datos.
- Geoprocesamiento. Análisis espacial y procesamiento de datos.
- Análisis de red. Resuelve enrutamiento, instalación más cercana y área de servicio.
- Geometría. Resuelve cálculos geométricos espaciales.

5.2.3 ArcSDE

La tecnología ArcSDE es la que se encarga de administrar la información geográfica dentro de una base de datos relacional (RDBMS), permitiendo el acceso a los usuarios de ArcGIS. Facilita el almacenamiento, acceso y administración de los datos espaciales y es compatible con los siguientes manejadores de base de datos: DB2, Informix, Oracle, PostgreSQL, SQL Server y SQL Server Express [129].

La tecnología ArcSDE soporta ambientes con edición multiusuario y edición distribuida, lo cual es crítico en aplicaciones geográficas [129]. Este producto de ESRI se vendía por separado hasta la versión 9.2 de ArcGIS y ahora está integrado en ArcGIS Desktop y ArcGIS Server [130].

5.2.4 ArcGIS para Análisis de Transporte

ArcGIS también provee soluciones a nivel de transporte mediante ArcGIS for Transportation Analytics [131]. Este software colabora con las operaciones de la flota de transporte, creando rutas, organizando las planificaciones y monitoreando los procesos realizados.

Algunas de las prestaciones que provee son [131]:

- Geocodificación de paradas.
- Optimización de rutas de transporte.
- Identificación de lugares de fácil acceso y definición de paradas.
- Obtención de matrices de costo de origen/destino.
- Seguimiento de vehículos.
- Navegación dentro del vehículo e información de tráfico en tiempo real.

5.3 ArcGIS para Dispositivos Móviles

Dado el creciente uso de dispositivos móviles y los beneficios que estos pueden brindar a una organización que trabaja con SIG, poder contar con una aplicación que soporte dichas funcionalidades en un dispositivo móvil conlleva grandes beneficios [132]. Algunas de las funcionalidades requeridas pueden ser la eficiencia y precisión del trabajo de campo, y rapidez a la hora de recolectar datos espaciales.

Con ArcGIS para dispositivos móviles, el usuario podrá navegar en los mapas, recolectar, editar y subir información geográfica y ejecutar análisis de SIG, accediendo directamente al SIG de la empresa mediante ArcGIS Online o ArcGIS Server [133].

Además de estas aplicaciones que brinda ArcGIS para móviles, también cuenta con un SDK[58] que permite extender su uso a dispositivos móviles como tabletas o teléfonos inteligentes, el cual es compatible con los sistemas operativos Android, iOS y Windows Phone [133]. También cabe la posibilidad de desarrollar aplicaciones web para dispositivos móviles mediante el uso de una API desarrollada en JavaScript [134].

5.3.1 ArcGIS para Android

El SDK de ArcGIS para Android permite agregar el potencial de las funcionalidades de ArcGIS en aplicaciones nativas del sistema operativo Android [135]. Este SDK permite acceder a ArcGIS Online y ArGIS Server, desde donde podrán agregarse mapas y consultar servicios. Entre las funcionalidades a destacar se encuentran:

- Uso de mapas y capas, permitiendo la carga de mapas almacenados localmente, los cuales podrán ser desplegados utilizando diferentes proyecciones.
- Obtención y edición de información en el mismo dispositivo, permitiendo la interacción con los sensores (GPS, acelerómetro y brújula, entre otros).
- Ejecución de tareas de geoprocesamiento en el dispositivo, liberando la carga de los servidores y permitiendo explotar la capacidad de procesamiento local.
- Uso de funcionalidades nativas (*pop-ups* y *push notifications*, entre otras) y de ArcGIS (análisis de redes, manejo de capas y servicios de geocodificación, entre otros) para mejorar la experiencia del usuario.
- Integración con herramientas de desarrollo (Eclipse), facilitando el uso de mapas en aplicaciones nativas.

El SDK es compatible con versiones de Android superiores a 2.3.3 y de ArcGIS Server superiores a 9.3.1. Este utiliza la tecnología OpenGL ES 2.0 [136] para renderizar los mapas, la cual

[58] SDK (Software Development Kit)

permite aumentar la velocidad de renderizado utilizando menos recursos de memoria y autonomía [137].

5.3.2 ArcGIS para Javascript

ArcGIS provee una API para JavaScript, con la cual permite desarrollar aplicaciones web que podrán utilizar las funcionalidades de ArcGIS Online y ArcGIS Server [138]. Esta API es compatible con los navegadores Chrome, Firefox, Safari e Internet Explorer, además de ser compatible con otras bibliotecas JavaScript como ExtJS y jQuery. La misma está disponible en los servidores de ArcGIS para uso libre, aunque también puede descargarse el código fuente y utilizarla desde un servidor propio [139]. Brinda soporte para HTML5 y CSS3 y provee una API específica para desarrollo de aplicaciones móviles.

La API específica para desarrollo de aplicaciones móviles es una versión compactada, la cual está optimizada para su uso en conexiones lentas o donde es necesario bajar la latencia de la red [140]. En esta versión se incluyen 32 de los 80 módulos utilizados en la versión estándar, permitiendo incluir los demás módulos en caso de ser necesario.

Entre las características y funcionalidades destacables se encuentran:

- Uso de mapas y capas, con detección de navegación para aplicaciones web y web móviles.
- Consumo de datos a través de servicios web, los cuales pueden ser estáticos o dinámicos, y pueden estar en diferentes formatos (CSV, KML, GeoRSS, XML, JSON, etc.).
- Consumo de servicios de ArcGIS Server, como son: ruteo, geoprocesamiento y operaciones geométricas, entre otros.
- Manejo de eventos de forma asíncrona, lo cual colabora con una mejor experiencia de usuario, ya que no experimenta bloqueos de interfaz entre tanto se procesa una consulta al servidor.
- Edición de datos geográficos, lo que permite que el usuario edite sus atributos (forma, tamaño, metadatos) y pueda impactar esos cambios en una base de datos.
- Desarrollo de nuevos *widgets* utilizando Dijit.
- Integración con otros frameworks de desarrollo escritos en JavaScript.

5.3.2.1 *Dojo*

La API de ArcGIS para JavaScript está desarrollada sobre Dojo. Esto surge de la necesidad de agilizar el proceso de desarrollo de la API y de poder asegurar que la misma se comporte de forma similar en todos los navegadores [141].

Dojo [142] es un framework de desarrollo que permite escribir código JavaScript de forma eficiente y robusta, reduciendo así la cantidad de código, abstrayéndose de las funciones específicas del lenguaje.

A pesar de que existen frameworks más populares Esri eligió Dojo por varias razones [143], entre las que se encuentran el uso sencillo de herencia basado en clases, abstracción de los gráficos vectoriales a través de dojo.gfx e integración con otros frameworks.

5.4 Google Maps

Google Maps es un servicio cartográfico web provisto por Google que consta de una aplicación web [144], y de una API [145] que proporciona a los desarrolladores diversas formas de embeber mapas en sus aplicaciones.

Entre otras características, brinda la posibilidad de visualizar diferentes tipos de mapas, como mapas de calles, mapas de imágenes satelitales, y mapas híbridos que combinan las opciones anteriores. En algunas ubicaciones también provee vistas fotográficas a nivel de calle, mapas 3D, y mapas de interiores, logrando así un increíble nivel de detalle.

La última versión de la API para JavaScript fue diseñada para el uso en dispositivos móviles, pero además Google Maps cuenta con SDKs nativos tanto para Android como para iOS.

Resulta una opción interesante ya que está disponible para la mayoría de las plataformas, la mayoría de los usuarios están acostumbrados a utilizarlo, es altamente configurable, y provee un conjunto completo de funcionalidades mediante Web Services, como geocodificación, ruteo, cálculo de distancias, y búsqueda de lugares.

5.4.1 Integración de ArcGIS y Google Maps

ArcGIS se puede integrar con Google Maps mediante la *ArcGIS Extension for the Google Maps API* [146], la cual permite extender la API de Google Maps para utilizar servicios de ArcGIS Server, mediante código JavaScript.

La integración es bastante flexible, permitiendo diferentes combinaciones. Es posible mostrar mapas del usuario en capas que se superpongan a un mapa base de Google Maps, ejecutar un modelo de SIG o buscar elementos en los datos geográficos del usuario y mostrar los resultados, o incluso mostrar atributos de los datos de SIG en el mapa mediante gráficas utilizando la Google Chart API [147], logrando un *mashup*[59] de SIG.

5.5 OpenStreetMaps

OpenStreetMaps [148] es un proyecto que brinda un mapa de calles a escala mundial de manera totalmente gratuita. La edición de la información del mismo está a cargo de la comunidad de usuarios de forma similar a un proyecto Wiki, permitiendo a cualquier persona el ingreso y edición de información de manera colaborativa.

5.6 Bing Maps

Bing Maps es un servicio online de mapas desarrollado por Microsoft, el cual provee múltiples APIs para la interacción con sitios web y aplicaciones de diferentes plataformas [149]. Estas API permiten utilizar la cartografía de Bing, proveyendo mapas estáticos e interactivos.

Cuenta con tres APIs para el desarrollo de aplicaciones móviles, las cuales son: AJAX V7, REST Services y Spatial Data Services. La API Bing Maps AJAX Control V7 [150] está desarrollada en JavaScript, pensada para ser utilizada con Bing Maps REST Services [151], las cuales proveen servicios de localización y búsqueda sobre mapas de Bing.

[59] Mashup: Aplicación resultante de la combinación de información de diversas fuentes web.

Bing Maps REST Services es una API que provee una interfaz REST para la ejecución de tareas, como pueden ser la creación de mapas estáticos, geocodificación de direcciones y obtención de datos de imágenes geográficas [152].

La API Bing Spatial Data Services provee una interfaz REST para la manipulación de información espacial. Esta interfaz utiliza los parámetros de la URL que posteriormente serán enviados mediante una solicitud HTTP, y en la respuesta de la misma se obtendrá el resultado.

Para enviar solicitudes a las diferentes API de Bing es necesario utilizar una clave proporcionada por Microsoft. Esta se utiliza para realizar reportes de uso, para los que se guardan determinados accesos a Bing Maps REST Services y Bing Spatial Data Services [153]. A estos accesos se los llama transacciones, las cuales se categorizan según si tienen costo o no. Los reportes anteriormente mencionados pueden accederse a través del sitio de Bing Maps Account Center.

6 Bases de Datos

Una base de datos es un conjunto de datos relacionados entre sí. Si bien este concepto es muy amplio, por lo general se suele tratar dentro del ámbito de Sistemas de Información. Un Sistema de Información es un conjunto de componentes de hardware y software que interactúan entre sí con el objetivo de almacenar, recuperar y procesar datos e información [154].

El tamaño, las funcionalidades y el desempeño de las bases de datos actuales han crecido enormemente debido a los avances tecnológicos, específicamente en la capacidad de procesamiento. Este avance no sería posible de no ser por las nuevas generaciones de procesadores, memorias y arquitecturas; la disponibilidad de hardware que soporta mayores volúmenes de almacenamiento; y la gran capacidad de transferencia de datos que brindan las redes de computadoras actuales.

6.1 Bases de Datos Espaciales

Existen diversas áreas de aplicación en lo que refiere a las bases de datos. En algunas de ellas es necesario almacenar información geométrica, geográfica y/o espacial, lo que tiene como objetivo ubicar un elemento en un entorno, utilizando un sistema definido. Esto no se limita únicamente a la geografía, sino que es utilizado en áreas como la medicina y la arquitectura, entre otras.

No hay una única definición para el concepto de base de datos espacial, pero es razonable utilizar la definición que da Ralf Hartmut Güting en el artículo "*An Introduction to Spatial Database Systems*" [155]. Define un sistema de base de datos espacial como aquel que provee tipos de datos espaciales a nivel de modelo y lenguaje de consulta, en donde estos tipos de datos son soportados a nivel de implementación, proveyendo índices espaciales y algoritmos eficientes de *join*[60] de estos datos.

Lo que se quiere representar son los objetos del espacio y el espacio mismo. Cuando hablamos de los objetos del espacio nos referimos, por ejemplo, a ciudades, ríos, bosques; y cuando pensamos en el espacio mismo queremos enfocarnos en características que describen a los objetos del espacio, por ejemplo, mapas temáticos. Estas dos consideraciones pueden unificarse si se provee un mecanismo para modelar objetos simples y colecciones de objetos espaciales relacionados.

Entre los tipos de datos espaciales básicos que debe soportar una base de datos de este tipo están: punto, línea y polígono. Estos proveen el nivel de abstracción necesario para modelar cualquier estructura geométrica del espacio, la relación entre ellas, sus propiedades y las operaciones que se pueden definirse.

Un punto representa un objeto del espacio que sólo su posición puntual es relevante, descartando su forma o extensión. Si pensamos en el foco del proyecto, un ómnibus puede representarse como un punto, ya que no es relevante su forma o de qué tamaño es específicamente, sino que nos interesa representarlo como un todo. Una línea, que en la

[60] Join: Operación que permite combinar registros de dos o más tablas de una base de datos.

mayoría de los casos es utilizada dentro de un conjunto de líneas conectadas entre sí, formando una polilínea, es utilizada para representar objetos con los que uno se mueve a través del espacio. En el ámbito del proyecto se utilizan para modelar las rutas de los ómnibus o el trayecto que describe una persona al caminar. Un polígono es utilizado para representar extensiones de dos dimensiones, y puede contener agujeros y ser formado por partes disjuntas.

Cuando se habla de las posibles relaciones entre estos tipos de datos básicos, se pueden considerar diferentes tipos:

- Relaciones topológicas. Las relaciones de adyacencia, pertenencia (un objeto está dentro de otro) y disyunción son invariantes a las operaciones de traslación, rotación y escala.
- Relaciones direccionales. Algunos ejemplos son “arriba de”, “debajo de”, “al sur de”, entre otras.
- Relaciones métricas. Por ejemplo, que la distancia entre un punto y otro sea mayor a cien metros.

También es necesario definir propiedades sobre estos objetos, las cuales son representadas de forma tradicional en varios manejadores de bases de datos, almacenando las mismas en tablas relacionadas que hacen referencia a los objetos del espacio.

El último aspecto a considerar en esta introducción a las bases de datos espaciales, son las operaciones que pueden definirse sobre los tipos de datos. Para esto es necesario definir un álgebra que defina las mismas. Estas podrán ser clasificadas como selección, *join*, aplicación de funciones y otras operaciones. La selección es una operación en la que se toman de un conjunto aquellos elementos que satisfacen una condición, por ejemplo, los ómnibus de determinada línea. Como ya se mencionó anteriormente, la operación de *join* permite combinar registros utilizando una condición de comparación. Esta operación hace que sea importante el soporte de índices espaciales. La aplicación de funciones refiere principalmente a los operadores de proyección y reemplazo.

6.2 Bases de Datos Relacionales

El concepto de bases de datos relacionales fue introducido por E. F. Codd en 1970, en un artículo llamado “*A Relational Model of Data for Large Shared Data Banks*” [156]. En esta publicación se marca un claro contraste en lo que hasta ese momento eran los modelos de datos utilizados (modelo de red y jerárquico) y el modelo relacional. Específicamente trata el tema de la independencia de los datos, enfocándose en la necesidad de independencia entre la implementación, representación, y el uso de los mismos. Menciona que la representación tabular fue uno de los mayores avances en relación a la independencia de los datos, ya que posibilitan el cambio de la representación interna de los datos sin afectar el uso.

Luego introduce la visión relacional de los datos que deriva de la teoría matemática de conjuntos. También presenta el concepto de tuplas pertenecientes a una relación y muestra sus principales características. A lo largo de la publicación sigue presentando varios conceptos relacionados a esta representación: formas normales, operaciones, redundancia y consistencia.

Más allá de los conceptos teóricos, el usuario necesita un mecanismo para interactuar con los datos contenidos en las bases de datos. De ahí surge la necesidad de contar con aplicaciones especialmente diseñadas para proveer mecanismos de definición, creación, administración, actualización y búsqueda sobre bases de datos [157]. A estas aplicaciones les llamamos manejadores de bases de datos.

En la publicación mencionada anteriormente se introdujo el modelo relacional, se presentó un lenguaje de acceso a datos basado en el cálculo de predicados. De esta idea, y luego de otras definiciones previas, surgió lo que hoy conocemos como el lenguaje SQL[61].

SQL es un lenguaje que permite realizar consultas a bases de datos a través de un manejador de bases de datos relacional (RDBMS). Puede dividirse en dos sublenguajes: DDL[62] y DML[63]. El lenguaje de definición (DDL) es utilizado para crear y eliminar bases de datos y sus objetos, mientras que el lenguaje de manipulación (DML) es utilizado para insertar, actualizar y obtener datos de las bases de datos [158].

6.2.1 SQLite

SQLite es una librería que implementa un motor de base de datos transaccional [159]. A diferencia de otros motores de bases de datos, SQLite no implementa el servidor como un servicio separado y tampoco utiliza mecanismos de comunicación específicos para el acceso a los datos, sino que los procesos acceden directamente a los archivos en disco para su lectura y escritura [160].

Esta librería mantiene múltiples tablas, índices, triggers y vistas en un único archivo. Este archivo tiene un formato que es independiente de la plataforma y de la arquitectura en la que se aloja.

Tal como se puede deducir de su nombre, es una librería compacta, que de omitirse algunas funcionalidades puede llegar a reducirse a 300 KiB. Está hecha para correr con un mínimo de espacio de *stack* (4 KiB) y con muy poco espacio de *heap* (100 KiB), lo que hace que sea la librería más utilizada en dispositivos con memoria limitada, como es el caso de los teléfonos inteligentes [159].

6.3 Bases de Datos No Relacionales

Las bases de datos relacionales son una pieza fundamental de los grandes sistemas de información empresariales, e históricamente se han utilizado de forma casi exclusiva en la mayoría de los sistemas de información. Sin embargo, el uso masivo de internet y la proliferación de dispositivos y aplicaciones que en forma conjunta generan grandes volúmenes de datos, potenciaron la necesidad de contar con otros paradigmas de almacenamiento y manipulación de datos que se adecúen a los requerimientos específicos de los sistemas actuales [161].

El concepto de bases de datos no relacionales, popularizadas como bases de datos “NoSQL”, se ha expandido rápidamente en los últimos tiempos [162]. Algunos autores atribuyen la

[61] SQL (Structured Query Language)
[62] DDL (Data Definition Language)
[63] DML (Data Manipulation Language)

etimología del nombre a que este tipo de bases de datos no soportan el lenguaje de consulta SQL, y otros a *not only* SQL [163]. Según esta definición, son muchas las bases de datos que entran en esta categoría, entre los que encontramos a las bases de datos basadas en documentos, grafos, clave-valor y columnas.

Estos paradigmas de bases de datos intentan solucionar algunos de los problemas que tiene el modelo relacional, como por ejemplo, la escalabilidad. Los sistemas actuales cada vez manejan mayor cantidad de información y esta debe ser manejada de manera eficiente y rápida. Si bien el desempeño de las bases de datos relacionales suele mejorar con una mayor cantidad de recursos, no se adecúan al modelo *cloud* [164].

En el ámbito del proyecto tendremos especial interés en investigar las bases de datos basadas en documentos y aquellas basadas en clave-valor.

6.3.1 Bases de Datos basadas en Documentos

Las bases de datos basadas en documentos [161] se diferencian en que los datos no son almacenados en tablas, como se hace en el modelo relacional, sino que se almacenan en documentos.

Es una representación libre de esquemas, ya que no se obliga a definir una estructura particular para el contenido de un documento. Si un nuevo campo debe agregarse a ese documento, simplemente se agrega y esto no afecta a los documentos almacenados actualmente. Otra ventaja con respecto al modelo relacional, es que los documentos no tienen que almacenar campos vacíos si no se cuenta con el valor del mismo.

Los documentos pueden ser complejos, permitiendo la lectura y escritura concurrente de objetos enteros del modelo, son independientes entre sí y son descriptos en formato abierto, utilizando JSON, XML o similares.

6.3.1.1 *MongoDB*

MongoDB es un sistema de base de datos *open-source* de propósito general basado en documentos [165]. Los documentos son almacenados en una codificación binaria de JSON[64] llamada BSON[65]. BSON extiende el modelo de JSON y provee tipos de datos adicionales, así como una codificación y decodificación eficiente en diferentes lenguajes [166].

Provee un completo soporte de índices y búsquedas avanzadas, búsqueda de texto, seguridad avanzada y varios mecanismos que lo hacen escalable y altamente disponible. Esto se logra mediante *auto-sharding* [66] y replicación en varios servidores. También brinda algunas funcionalidades para el manejo de grandes volúmenes de datos, como es el *Aggregation Framework* y *MapReduce* [167].

6.3.2 Bases de Datos Clave-Valor

Las bases de datos basadas en clave-valor también tienen una representación libre de esquemas y usualmente consisten en una cadena de caracteres que representa la clave y el

[64] JSON (JavaScript Object Notation)
[65] BSON (Binary JSON)
[66] Sharding: Particionamiento horizontal de la base de datos.

dato que representa el valor de la relación clave-valor [168]. El dato es en general un tipo de dato primitivo o un objeto de un determinado lenguaje de programación.

Este modelo permite que se almacenen datos arbitrarios utilizando una única clave, lo que hace que en determinados contextos se experimente un desempeño mucho mayor que con el modelo tradicional. Otra ventaja significativa es que el código resultante suele ser mucho más breve y claro que en el caso de sentencias SQL embebidas.

6.3.2.1 *Redis*

Redis es un sistema de base de datos *open-source* escrito en C basado en clave-valor [169]. Lo interesante de Redis radica en que las claves no están limitadas a cadenas de caracteres, sino que pueden ser hashes, listas, conjuntos o conjuntos ordenados. Además, permite realizar operaciones básicas sobre estas claves, como pueden ser: incrementar valor del hash, concatenar una cadena de caracteres, agregar datos a una lista, intersectar dos conjuntos, entre otras.

Con el fin de obtener el desempeño requerido, Redis maneja los datos en memoria, aunque estos se persisten periódicamente de forma asíncrona en disco. Puede ser configurado para persistir los datos luego de un lapso de tiempo o una cantidad determinada de cambios.

Una de las características más significativas es el soporte de múltiples lenguajes de programación. Actualmente hay clientes de Redis para JavaScript, Ruby, Python, PHP, Erlang, Tcl, Perl, Lua y Java, entre otros [170].

7 Aplicaciones Relacionadas

En la presente sección se documentan algunos de los sistemas y aplicaciones que están de alguna forma relacionadas con el proyecto, ya sea por manejar información de ómnibus de forma distribuida, o por enviar datos en tiempo real entre dispositivos móviles.

7.1 iBus

La empresa uruguaya C.U.T.C.S.A. (Compañía Uruguaya de Transporte Colectivo S.A.) y la multinacional Movistar se unieron para proveer un servicio a clientes de Movistar, usuarios de C.U.T.C.S.A. llamado iBus [171].

El servicio es utilizado a través del envío y recepción de mensajes de texto. El envío debe realizarse con determinado formato y el servicio tiene disponible dos funcionalidades: recibir el número de la parada más cercana al punto en donde se encuentra el usuario y/o recibir el tiempo estimado en el que arribarán los próximos dos ómnibus de la línea consultada [171].

MOVISTAR utiliza la tecnología LBS[67] para la detección de la posición del usuario. Esta tecnología está basada en múltiples técnicas de geolocalización para móviles (como pueden ser *Time Difference of Arrival* o *Enhanced Observed Time Difference*, entre otras) [172], con la que obtiene una ubicación aproximada del celular del usuario que luego se utilizará para procesar la solicitud. En la especificación del servicio se indica que la posición del usuario y el tiempo de arribo que se obtienen son aproximados [173].

Se efectuó una prueba ubicando al usuario en la puerta de Facultad de Ingeniería (UdelaR) y se realizó la consulta para la línea de ómnibus 199. El texto del mensaje enviado desde el móvil fue el siguiente (el número de parada se había obtenido previamente): "*BUS 199 2114*".

El mensaje respuesta fue el siguiente: "Linea 199 CRIO DEL NORTE en camino, arriba a la parada 2114 sobre BENITO NARDONE esquina PATRIA en 4 minutos. Siguiente unidad en terminal, arriba en 12 minutos."

7.2 Google Latitude

Google Latitude era un servicio que proveía Google Maps para Android y que también contaba con una aplicación para iOS [174].

Básicamente brindaba la posibilidad de compartir la ubicación de un usuario y que este pudiera ver la ubicación de sus amigos desplegada en un mapa, pudiendo contactarse con ellos mediante distintos servicios de mensajería o mediante su mensaje de estado [174].

Dentro de las funcionalidades que proveía, se encontraban [175]:

- Compartir la ubicación propia con usuarios de Google Latitude (móviles o iGoogle).
- Ver la ubicación de sus amigos desplegada en un mapa.
- Controlar la ubicación, controlando quién y qué ve cada usuario (diferentes niveles de detalle).

[67] LBS (Location Based Service)

- Configuración de privacidad, permitiendo la edición manual, pudiendo presentar diferentes ubicaciones a cada usuario amigo (incluyendo la opción de no compartir la ubicación).
- Detección de visita a diferentes sitios. El dispositivo detecta automáticamente que se llegó a determinado lugar y realiza la petición de compartir dicha información.

Google Latitude podía ser utilizado desde cualquier navegador de internet, así como en las aplicaciones específicas para cada móvil. Respecto a la compatibilidad con los sistemas operativos de los diferentes dispositivos móviles, era compatible con Android, BlackBerry, iPhone, Symbian y Windows Mobile [175]. El servicio fue dado de baja el 9 de agosto de 2013.

7.3 Syncromatics

Syncromatics es una empresa estadounidense con sede en Los Ángeles que brinda soluciones ITS (Intelligent Transportation System) a empresas de transporte y universidades [176]. Los servicios ITS son aplicaciones avanzadas que proveen servicios para la administración del tráfico y el transporte, brindando al usuario la información necesaria para que este pueda utilizar las vías de transporte en forma coordinada, inteligente y segura [177].

Syncromatics utiliza la tecnología AVL y automatiza diferentes procesos, con lo que brinda servicios de monitoreo, análisis, publicación y mejora de los sistemas de tránsito [178].

Dentro de los servicios que provee se encuentran los siguientes [178]:

- Seguimiento de unidades mediante AVL. La posición, velocidad y dirección (hacia donde está yendo) son actualizadas cada seis segundos.
- Sistema de información al pasajero. Brinda información en tiempo real de tiempo de arribo y cantidad de pasajeros de una determinada unidad, las cuales pueden ser accedidas mediante portal web (con interfaz alternativa para dispositivos móviles), señales en la vía pública o sistemas telefónicos interactivos [179].
- Administración y análisis de rutas. Permite crear, editar, analizar, monitorear y diagnosticar rutas.
- Conteo de pasajeros. Brinda información en tiempo real de la cantidad de pasajeros de una unidad y permite extraer reportes de uso de la misma en un determinado intervalo de tiempo.

7.4 GPS Tracking Pro

GPS Tracking Pro [180] es una aplicación para Android que brinda la posibilidad de realizar un seguimiento en tiempo real de otros usuarios que utilicen Android.

Mediante tecnología GPS, la aplicación rastrea la posición exacta del otro usuario (puede ser un miembro de la familia) y la despliega en un mapa.

Esta también puede avisar si el usuario está en problemas, utilizando el servicio de check-in que permite especificar cuándo llega a un determinado lugar y está a salvo.

Otra funcionalidad con la que cuenta es la de alertar si hay agresores sexuales en el área.

7.5 Real Time GPS Tracker

Real Time GPS Tracker [181] es una aplicación para Android que permite al usuario compartir su ubicación con sus amigos, la cual es desplegada en un mapa de Google Maps u OpenStreetMap.

La aplicación brinda información adicional de los dispositivos que estoy rastreando, como puede ser: velocidad, elevación, nivel de batería, etc. Cualquiera de los usuarios puede enviar mensajes que aparecerán en el mapa.

El rastreo puede comenzar remotamente mediante un mensaje de texto (SMS) al dispositivo que se desea rastrear o mediante un portal web. Este portal web también permite visualizar en un mapa los dispositivos que un usuario está rastreando en un determinado momento.

7.6 London Transport Pro

London Transport Pro [182] es una aplicación para Android que asiste al usuario que se encuentra de viaje en Londres, utilizando información de tiempo real de TfL (Transport for London), desplegando la misma en un mapa.

Algunos de los servicios que brinda esta aplicación son: información de arribo de ómnibus, información de arribo del subterráneo, actualización de rutas de ómnibus y subterráneo, acceso a cámaras CCTV de tráfico, visualización y filtro de rutas de ómnibus.

7.7 Bus Checker Londres

Bus Checker Londres [183] es una aplicación para Android que permite obtener información en tiempo real del arribo de ómnibus.

Esta aplicación permite ver en tiempo real, el tiempo que falta para que llegue un ómnibus, permite desplegar en un mapa las paradas más cercanas, obteniendo la información desde el GPS del dispositivo. También permite desplegar en un mapa los recorridos completos de los ómnibus, brinda información de desviaciones, cierres y cancelaciones y permite filtrar por líneas de ómnibus de interés.

7.8 WhatsApp Messenger

WhatsApp Messenger [184] es una aplicación de mensajería móvil multiplataforma que funciona a través del plan de datos de Internet existente de un dispositivo. A diferencia de los SMS, los mensajes son enviados a través de Internet, por lo cual no supone ningún coste adicional.

Presenta un mecanismo de confirmación doble, la primera confirmación cuando se entrega el mensaje al servidor de WhatsApp, y la segunda cuando el mensaje es entregado al destinatario.

La aplicación identifica un dispositivo mediante el número de teléfono, lo cual permite guiar los mensajes entre contactos. Utiliza la agenda de contactos del dispositivo para obtener los números, de modo que el usuario pueda saber con cuáles contactos se puede comunicar mediante WhatsApp. Al utilizar notificaciones *push*, el uso de la aplicación no aumenta demasiado el consumo de la batería. Está disponible para iPhone, BlackBerry, Android, Windows Phone y Nokia Symbian.

8 Glosario

ABM	Acrónimo utilizado para referirse a las operaciones de alta, baja y modificación de datos o entidades de un sistema de información.
API	Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
ARM	Arquitectura de procesadores RISC utilizada en microprocesadores y microcontroladores de bajo consumo, ampliamente utilizados en telefonía móvil.
AVL	Acrónimo de Automatic Vehicle Location (Rastreo Vehicular Automatizado en español), que refiere a los sistemas de localización remota en tiempo real, utilizando una combinación de dispositivos de localización (GPS, GLONASS) con dispositivos de transmisión inalámbrica de datos (módems).
Background	Término utilizado para referirse al modo en que se ejecutan determinados servicios u operaciones para los que el usuario no percibe su ejecución, creados con el objetivo de que la usabilidad de la aplicación no se vea afectada.
BOSH	Protocolo de transporte que emula la comunicación bidireccional entre dos sistemas utilizando el protocolo HTTP. Hace un uso más eficiente del ancho de banda y mejora el tiempo de respuesta en aplicaciones que utilizan las operaciones pull y push.
BSON	Formato de intercambio de datos serializados en codificación binaria, utilizado para transferir y almacenar documentos tipo JSON.
Callback	Código ejecutable pasado como parámetro en la invocación de una función, el cual será ejecutado (de forma sincrónica o asincrónica) como respuesta de la misma.
CDMA	Refiere a los métodos de multiplexación o control de acceso al medio utilizados en algunas redes de telefonía celular, para que varios usuarios puedan transmitir información por un mismo canal, sin interferir entre ellos.
CPU	Acrónimo de Central Processing Unit (Unidad Central de Procesamiento). Es el componente encargado de ejecutar las instrucciones de máquina.
CSS	Lenguaje de hojas de estilo utilizadas para definir la presentación semántica de un documento escrito en lenguaje de marcas. Comúnmente utilizadas para dar estilo a páginas web escritas en lenguaje HTML.
¿Cuándo Llega?	Agrupación de funcionalidades de la aplicación relacionadas al seguimiento en tiempo real de ómnibus.

DDL	Lenguaje proporcionado por un manejador de base de datos que permite definir las estructuras en donde serán almacenados los datos, así como las funciones o procedimientos que permitan consultarlos.
Debug	Término utilizado para referirse al proceso en el que se realizan pruebas sobre una pieza de software o hardware, con el objetivo de encontrar defectos que afecten el correcto funcionamiento del mismo.
Deploy	Conjunto de actividades que permiten que un sistema de software quede disponible para ser utilizado.
DML	Lenguaje proporcionado por un manejador de base de datos que permite que los usuarios de las bases de datos lleven a cabo consultas o modificaciones sobre los datos contenidos en las mismas.
DSL	Lenguaje de programación especificado para resolver problemas de un dominio particular, representar un problema específico y proveer técnicas para determinar la solución del mismo.
EGNOS	Sistema de corrección de señales desarrollado por la Agencia Espacial Europea, la Comisión Europea y Eurocontrol. Desarrollado con el fin de complementar las redes GPS y GLONASS para mejorar la precisión y seguridad de señales, permitiendo una precisión inferior a dos metros.
Escalabilidad	Propiedad de un sistema que indica su habilidad para reaccionar y adaptarse al crecimiento continuo de la carga de trabajo, o que está preparado para soportar volúmenes mayores de trabajo sin perder la calidad del servicio ofrecido.
ETL	Acrónimo para referirse al proceso de extracción, transformación y carga de datos. Los datos son extraídos de una o varias fuentes de datos, pasan por un proceso de transformación en el que se pueden o no realizar cambios, para finalmente ser cargados en una nueva fuente de datos.
FindMe!	Agrupación de funcionalidades de la aplicación relacionadas a la red social de usuarios y seguimiento en tiempo real de amigos.
Flash Sockets	Implementación de Sockets de Flash, permite establecer conexiones TCP para enviar y recibir datos binarios.
Framework	Estructura conceptual y tecnológica de soporte definido, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.
Garbage Collector	Mecanismo de manejo automático de la memoria, intenta liberar memoria que ya no está siendo utilizada por los objetos de un programa.

Geolocalización	Posicionamiento con el que se define la localización de un objeto espacial (representado mediante punto, vector, área, volumen) en un sistema de coordenadas y datum determinado. Este proceso es utilizado frecuentemente en los SIG.
GLONASS	Constelación de 31 satélites desarrollada por la Unión Soviética, y siendo administrada actualmente por la Federación Rusa, representa la contrapartida al GPS estadounidense.
GNSS	Constelación de satélites que transmite rangos de señales utilizados para el posicionamiento y localización en cualquier parte del globo terrestre, ya sea en tierra, mar o aire. Estos permiten determinar las coordenadas geográficas y la altitud de un punto dado como resultado de la recepción de señales provenientes de constelaciones de satélites artificiales de la Tierra para diversos fines.
GPS	Sistema de satélites perteneciente al Departamento de Defensa de los Estados Unidos, permite triangular la ubicación de un objeto en la tierra, como una persona o un vehículo con una precisión en el ámbito de los metros.
GPU	Procesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos y o aplicaciones 3D interactivas.
GSM	Sistema de telefonía móvil digital estándar, y libre de regalías, de segunda generación.
Heap	Estructura de datos del tipo árbol con información perteneciente a un conjunto ordenado.
Hosting	Servicio que provee a los usuarios de Internet un sistema para poder almacenar información, imágenes, vídeo, o cualquier contenido accesible vía web. Se utiliza también para referirse al alojamiento del código y ejecución de un servidor.
HSPA	Combinación de tecnologías posteriores y complementarias a la tercera generación de telefonía móvil (3G).
HTML	Es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.
HTTP	Protocolo de aplicación para sistemas de información distribuidos, colaborativos, o de hypermedia. Es la base de la comunicación de datos de la World Wide Web.
JSON	Formato ligero para el intercambio de datos, es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

JSONP	Técnica de comunicación utilizada por los programas JavaScript que ejecutan en los navegadores web, permite realizar peticiones a un servidor que se encuentra en un dominio diferente que la aplicación.
JVM	Máquina virtual de proceso nativo, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial que es generado por el compilador del lenguaje Java.
Location Based System (LBS)	Sistemas que utilizan servicios de software basados en información de ubicación geográfica y tiempo, para brindar funcionalidades que sean relevantes al contexto en el que se utiliza el sistema.
Log	Registro oficial de eventos durante un rango de tiempo en particular, es utilizado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre para un dispositivo en particular o aplicación.
MapReduce	Programa que está compuesto de una operación (map) que realiza filtrado y ordenamiento, y otra operación (reduce) que realiza una operación de agregación.
Model-View-Controller-Store (MVCS)	Paradigma de arquitectura de software, que parte del patrón MVC, e intenta minimizar el código redundante y simplificar la forma en que se reciben y almacenan los datos, al encapsular esto último en un objeto llamado Store.
MOM	Infraestructura de software o hardware que provee comunicación distribuida en base a un modelo de interacción asíncrona.
MQTT	Protocolo de mensajería simple y liviano que ofrece funcionalidad de publicar/suscribir y ha sido diseñado específicamente para dispositivos con restricciones energéticas y redes poco confiables, con alta latencia, o ancho de banda mínimo
.NET Framework	Plataforma de desarrollo de software de Microsoft, con énfasis en transparencia de redes e independencia de plataforma de hardware, que permite un rápido desarrollo de aplicaciones.
NoSQL	Bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales en aspectos importantes, por ejemplo, no utilizar SQL como el principal lenguaje de consultas.
Object-Document Mapper (ODM)	Herramienta que mapea los documentos de una base de datos orientada a documentos, a objetos. También se encarga de abstraer las operaciones comunes como consultas, actualizaciones y borrado.
Open Source	Software distribuido y desarrollado libremente. El movimiento open-source se concentra en los beneficios prácticos, como acceso al código fuente, así como en cuestiones éticas o de libertad.

Parser	Programa cuya finalidad es analizar una cadena de caracteres, de acuerdo a las reglas de una gramática formal.
Pinch	Evento capturado cuando el usuario “pellizca” la superficie sensible al tacto.
Pixelado	Efecto causado por visualizar una imagen a un tamaño en el que los pixels individuales son visibles al ojo.
Plugin	Componente de software que agrega una funcionalidad específica a una aplicación de software existente.
Polling	Operación de consulta constante, por ejemplo, con el objetivo de presentar información actualizada en una interfaz web.
Polyfill	Biblioteca que imita una API existente, y cuenta con diferentes implementaciones para brindar la misma funcionalidad cuando no es posible utilizar alguna de ellas.
Progressive Disclosure	Técnica de diseño de interacción que permite retener la atención del usuario al reducir la confusión y la carga cognitiva de la tarea a realizar.
Push	Estilo de comunicación basado en Internet donde la petición para una transacción dada es iniciada por el servidor, en contraste con el mecanismo habitual donde la petición es realizada por el cliente.
RAM	Medio de almacenamiento temporal donde se cargan todas las instrucciones que ejecutan el procesador y otras unidades de cómputo.
RDBMS	Manejador de Bases de Datos Relacionales. Es un conjunto de programas que proveen los mecanismos necesarios para almacenar, modificar y extraer datos de bases de datos relacionales.
Renderización	Proceso de generar una imagen o vídeo a partir de un modelo mediante un programa de computadora.
REST	El término se aplica en la actualidad a cualquier interfaz web simple que utiliza HTTP, con un protocolo cliente/servidor sin estado y operaciones bien definidas, donde cada recurso puede ser accedido utilizando un identificador global único.
ROM	Medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite sólo la lectura de la información y no su escritura, independientemente de la presencia o no de una fuente de energía.
Sans-serif	También llamadas fuentes de palo seco o sin remates, son fuentes que en cada carácter no tienen unas pequeñas terminaciones llamadas remates o serifas.
Scripting (lenguaje)	Lenguaje de programación que soporta la escritura de scripts, los cuales son escritos para un entorno particular y son utilizados para automatizar determinadas tareas.

SDK	Conjunto de herramientas de desarrollo que permite la creación de aplicaciones para un cierto paquete o plataforma de software, plataforma de hardware, sistema de computadora, sistema operativo, o plataforma de desarrollo similar.
Sharding	Particionamiento horizontal de una base de datos, por ejemplo, donde las filas de una misma tabla se almacenan de forma separada.
SIG	Tipo especial de sistemas de información utilizado para administrar, analizar y desplegar información geográfica. La información geográfica es representada mediante estructuras simples que modelan la geografía y el SIG provee las herramientas necesarias para manipular dicha información.
SIM	Tarjeta inteligente desmontable usada en teléfonos móviles y módems HSPA o LTE, almacena de forma segura la clave de servicio del suscriptor usada para identificarse ante la red.
Smartphone	Teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos, y que posibilita realizar actividades semejantes a una minicomputadora, brindando mayor conectividad que un teléfono móvil convencional.
SoC (System on a Chip)	Tecnologías de fabricación que integran todos o gran parte de los módulos componentes de una computadora, o cualquier otro sistema informático o electrónico, en un único circuito integrado o chip.
SQL	Lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas, como consultas y actualizaciones.
Stack (estructura de datos)	Lista ordenada o estructura de datos en la que el modo de acceso a sus elementos es de tipo LIFO (último en entrar, primero en salir) que permite almacenar y recuperar datos.
Stack (software)	Ambiente completo de deployment, incluyendo el sistema operativo, el ambiente de ejecución del lenguaje, y las bibliotecas asociadas.
Streaming	Proceso de enviar los datos de forma continua desde el servidor al cliente, a través de una conexión persistente.
Swipe	Evento capturado cuando se hace un tap, para luego arrastrar el objeto con el cual se establece el contacto con la superficie sensible al tacto.
Tap	Evento capturado cuando se hace contacto en un único punto entre el usuario y el dispositivo con interfaz sensible al tacto.
TCP	Protocolo de comunicación de transmisión fiable orientado a conexión, opera al nivel de capa de transporte.
Timestamp	Identificador basado en el tiempo, puede ser la fecha de creación o actualización de una entidad.

Token	Credencial o llave que se utiliza para dar autorización en un sistema informático.
Viaje	Ómnibus que está realizando el recorrido de una línea particular en un instante dado.
WebSockets (WSS)	Tecnología que proporciona un canal de comunicación bidireccional sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede ser por cualquier aplicación cliente/servidor.
Wide Area Augmentation System (WAAS)	Sistema de satélites y estaciones base que proveen correcciones de la señal de GPS, brindando mayor precisión en las ubicaciones.
Wireframe	Representación esquemática de una aplicación, es una guía visual básica que intenta sugerir la ubicación de los elementos de diseño clave en la interfaz gráfica. Sirven como herramienta de comunicación y discusión entre desarrolladores, diseñadores y clientes.
Workflow	Flujo de trabajo de una aplicación, define cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas.
Wrapper	Es un programa que hace posible que se pueda ejecutar otro programa, más importante, en un entorno en el que no podría ejecutar.
x86	Conjunto de instrucciones utilizada en la microarquitectura de CPU, siendo también una denominación genérica dada a ciertos microprocesadores.
XMLHttpRequest (XHR)	Interfaz empleada para realizar peticiones HTTP a servidores Web, se utiliza comúnmente para proporcionar contenido dinámico y actualizaciones asíncronas en páginas Web a través de tecnologías como AJAX.

9 Bibliografía

[1] Gross, Doug. (2011, Octubre) 10 ways mobile gadgets have changed our lives. [Online]. http://edition.cnn.com/2011/10/07/tech/mobile/smartphones-change-lives

[2] International Telecommunication Union. (2010, Julio) MDG - ITU Story Line. [Online]. http://www.itu.int/ITU-D/ict/mdg/storyline/index.html

[3] Samsung. (2012, Abril) Samsung's New Quad-core Application Processor. [Online]. http://www.samsung.com/global/business/semiconductor/minisite/Exynos/news_12.html

[4] Raphael Schrader. (2012, Agosto) Gartner - Android Rules the Smartphone Market Share in Q2 2012 [Report] | FriendCaller Blog. [Online]. http://blog.friendcaller.com/gartner-android-rules-the-smartphone-market-share-in-q2-2012-report/

[5] Tim Schiesser. (2012, Febrero) Guide to smartphone hardware (1/7): Processors - Neowin. [Online]. http://www.neowin.net/news/guide-to-smartphone-hardware-17-processors

[6] Angel Luis Sanchez Iglesias. ¿Qué significa SOC? System on a chip. [Online]. http://computadoras.about.com/od/preguntas-frecuentes/a/Que-Significa-Soc-System-On-A-Chip.htm

[7] ARM. Smartphones - ARM. [Online]. http://www.arm.com/markets/mobile/smartphones.php

[8] Techautos. (2010, Marzo) Making Sense of Smartphone Processors: The Mobile CPU/GPU Guide. [Online]. http://www.techautos.com/2010/03/14/smartphone-processor-guide/

[9] ARM. Cortex-A8 Processor - ARM. [Online]. http://www.arm.com/products/processors/cortex-a/cortex-a8.php

[10] ARM. Cortex-A9 Processor - ARM. [Online]. http://www.arm.com/products/processors/cortex-a/cortex-a9.php

[11] Tim Schiesser. (2012, Febrero) Guide to smartphone hardware (2/7): Graphics - Neowin. [Online]. http://www.neowin.net/news/guide-to-smartphone-hardware-27-graphics-2

[12] Qualcomm. Adreno Graphics Processing Units, 3D GPU - Qualcomm Developer Network. [Online]. https://developer.qualcomm.com/discover/chipsets-and-modems/adreno-gpu

[13] Imagination Technologies. POWERVR SGX Graphics IP Core Family. [Online]. http://www.imgtec.com/powervr/sgx.asp

[14] ARM. Mali Graphics Hardware - ARM. [Online]. http://www.arm.com/products/multimedia/mali-graphics-hardware/index.php

[15] NVidia. Tegra 3 Super Chip Processors | NVIDIA. [Online]. http://www.nvidia.com/object/tegra.html

[16] GARMIN. Garmin | What is GPS? [Online]. http://www8.garmin.com/aboutGPS/

[17] GeoNet Ltd. What is GNSS?:GeoNet. [Online]. http://geonet.bg/answers/items/what-is-gnss.42.html

[18] Qualcomm. (2011, Diciembre) GPS and GLONASS: “Dual-core" Location For Your Phone. [Online]. http://www.qualcomm.com/media/blog/2011/12/15/gps-and-glonass-dual-core-location-your-phone

[19] M.Sc. Maximilian Schirmer and Jun.-Prof. Dr.-Ing. Hagen Höpfner. Smartphone Hardware Sensors. [Online]. http://www.uni-weimar.de/medien/wiki/images/Zeitmaschinen-smartphonesensors.pdf

[20] Avago Technologies. (2008, Enero) Improving S-GPS sensitivity - Electronic Products. [Online]. http://www2.electronicproducts.com/Improving_S_GPS_sensitivity-article-farr_avago_jun2008-html.aspx

[21] Fred Zahradnik. A-GPS - How Assisted GPS Works in Cell Phones. [Online]. http://gps.about.com/od/glossary/g/A-GPS.htm

[22] Jordi Sabaté Martí. (2011, Marzo) Trucos para alargar la batería de un móvil smartphone | EROSKI CONSUMER. [Online]. http://www.consumer.es/web/es/tecnologia/hardware/2011/03/22/199332.php

[23] ThermoAnalytics : HEV Vehicle Battery Types. [Online]. http://www.thermoanalytics.com/support/publications/batterytypesdoc.html

[24] Justin Fritz. (2011, Agosto) Is Everlasting Smartphone Battery Life Finally Here? | Wall Street Daily. [Online]. http://www.wallstreetdaily.com/2011/08/29/everlasting-smartphone-battery-life-finally-here/

[25] Justin Fritz. (2011, Noviembre) Smartphone Batteries are About to Get a Massive Upgrade | Wall Street Daily. [Online]. http://www.wallstreetdaily.com/2011/11/18/lithium-ion-smartphone-battery-life-upgrade/

[26] Justin Fritz. (2011, Setiembre) Scientists Just Revolutionized the Battery… With a Jelly Recipe | Wall Street Daily. [Online]. http://www.wallstreetdaily.com/2011/09/14/scientists-just-revolutionized-the-battery-with-a-jelly-recipe/

[27] (2012, Julio) Cell phone battery life charts - CNET Reviews - CNET Reviews. [Online]. http://reviews.cnet.com/cell-phone-battery-life-charts/

[28] Aaron Carroll and Gernot Heiser. (2010) An Analysis of Power Consumption in a Smartphone. [Online]. http://www.nicta.com.au/pub?doc=3587

[29] Abhinav Pathak, Y. Charlie Hu, and Ming Zhang. (2012) Where is the energy spent inside my app? [Online]. http://research.microsoft.com/en-us/people/mzh/eurosys-2012.pdf

[30] Jeff Sharkey. (2009, Mayo) Coding for Life--Battery Life, That Is. [Online]. https://dl.google.com/io/2009/pres/W_0300_CodingforLife-BatteryLifeThatIs.pdf

[31] Gartner. (2012, Mayo) Gartner Says Worldwide Sales of Mobile Phones Declined 2 Percent in First Quarter of 2012. [Online]. http://www.gartner.com/it/page.jsp?id=2017015

[32] Open Handset Alliance FAQ. [Online]. http://www.openhandsetalliance.com/oha_faq.html

[33] Marko Gargenta, "Android Overview," in *Learning Android*., 2011, ch. 1.

[34] Apache License, Version 2.0. [Online]. http://www.apache.org/licenses/LICENSE-2.0

[35] The MIT License (MIT). [Online]. http://opensource.org/licenses/mit-license.php

[36] IDC. (2012, Junio) Android Expected to Reach Its Peak This Year as Mobile Phone Shipments Slow, According to IDC. [Online]. http://www.idc.com/getdoc.jsp?containerId=prUS23523812

[37] Oracle. (2013, Agosto) Chapter 1. Introduction. [Online]. http://docs.oracle.com/javase/specs/jls/se7/html/jls-1.html

[38] Amber D. Walker. (2013, Agosto) Advantages & Disadvantages of Java Virtual Machine Interpreter | eHow. [Online]. http://www.ehow.com/facts_5558791_advantages-java-virtual-machine-interpreter.html

[39] David Ehringer. (2010, Marzo) Adventures in Software Development | Mindful Mischief. [Online]. http://davidehringer.com/software/android/The_Dalvik_Virtual_Machine.pdf

[40] Mark Sinnathamby. (2012, Julio) Stack based vs Register based Virtual Machine Architecture, and the Dalvik VM. [Online]. http://markfaction.wordpress.com/2012/07/15/stack-based-vs-register-based-virtual-machine-architecture-and-the-dalvik-vm/

[41] Android. Exploring the SDK. [Online]. http://developer.android.com/sdk/exploring.html

[42] Android. (2013, Agosto) SDK Tools | Android Developers. [Online]. http://developer.android.com/tools/sdk/tools-notes.html

[43] Android. Android NDK. [Online]. http://developer.android.com/tools/sdk/ndk/index.html

[44] Kerris, Natalie. (2007, Enero) Apple - Press Info - Apple Reinvents the Phone with iPhone. [Online]. http://www.apple.com/pr/library/2007/01/09Apple-Reinvents-the-Phone-with-iPhone.html

[45] Dowling, Steve. (2007, Enero) Apple - Press Info - iPhone to Support Third-Party Web 2.0

Applications. [Online]. http://www.apple.com/pr/library/2007/06/11iPhone-to-Support-Third-Party-Web-2-0-Applications.html

[46] Kerris, Natalie. (2008, Marzo) Apple - Press Info - Apple Announces iPhone 2.0 Software Beta. [Online]. http://www.apple.com/pr/library/2008/03/06Apple-Announces-iPhone-2-0-Software-Beta.html

[47] Apple Inc. XCode 4 - Apple Developer. [Online]. https://developer.apple.com/xcode/

[48] Apple Inc. Instruments User Guide: About Instruments. [Online]. https://developer.apple.com/library/mac/#documentation/developertools/conceptual/InstrumentsUserGuide/Introduction/Introduction.html

[49] Apple Inc. Tools Workflow Guide for iOS. [Online]. http://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/ios_development_workflow/25-Using_iOS_Simulator/ios_simulator_application.html#//apple_ref/doc/uid/TP40007959-CH9-SW1

[50] Apple Inc. (2013, Agosto) Programming with Objective-C. [Online]. https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html

[51] Apple Inc. (2009, Octubre) Objective-C Runtime Programming Guide: Introduction. [Online]. https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ObjCRuntimeGuide/Introduction/Introduction.html

[52] Apple Inc. (2009, Octubre) Objective-C Runtime Programming Guide: Runtime Versions and Platforms. [Online]. https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ObjCRuntimeGuide/Articles/ocrtVersionsPlatforms.html

[53] Apple Inc. (2013, Agosto) Cocoa Touch Frameworks. [Online]. https://developer.apple.com/technologies/ios/cocoa-touch.html

[54] Apple Inc. (2011, Octubre) iOS Technology Overview. [Online]. http://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf

[55] Margaret Rouse. (2008) TechTarget. [Online]. http://searchunifiedcommunications.techtarget.com/definition/real-time-communications

[56] Ericsson Labs. Push lasts longer - Save your mobile battery. [Online]. https://labs.ericsson.com/developer-community/blog/push-lasts-longer---save-your-mobile-battery

[57] Abdulkadir Özcan, Dhinaharan Nagamalai, and Jan Zizka, *Recent Trends in Wireless and*

Mobile Networks: Third International Conferences, WiMo 2011 and CoNeCo 2011.: Springer, 2011.

[58] Frank H. P. Fitzek and Hassan Charaf, *Mobile Peer to Peer (P2P): A Tutorial Guide*.: Wiley, 2009.

[59] TICbeat. El crecimiento de la telefonía móvil - infografía | TICbeat. [Online]. http://www.ticbeat.com/sim/crecimiento-telefonia-movil-infografia/

[60] funSMS.NET. Mobile Phone Generations - funSMS.NET. [Online]. http://www.funsms.net/mobile_phone_generations.htm

[61] Rahul Gondane. (2011, Marzo) Mobile Generations - 1G to 4G. [Online]. http://www.coolpctips.com/2011/03/mobile-generations-1g-to-4g/

[62] International Telecommunication Union. (2011, Abril) All about the Technology. [Online]. http://www.itu.int/osg/spu/ni/3G/technology/index.html#Introduction___-Evolution

[63] Nadeem Unuth. 3G - What is 3G? [Online]. http://voip.about.com/od/mobilevoip/p/3G.htm

[64] Liane Cassavoy. What is HSDPA - An Explanation of HSDPA 3G Service for Your Smart Phone. [Online]. http://cellphones.about.com/od/glossary/g/hsdpa_definied.htm

[65] sayanthan. (2011, Mayo) Sayy's 365: What is 3.75G? [Online]. http://sayys365.blogspot.com/2011/05/what-is-375g.html

[66] Motorola, Inc. (2011) Driving 4G: WiMAX & LTE. [Online]. http://www.motorola.com/web/Business/Solutions/Industry%20Solutions/Service%20Providers/Wireless%20Operators/Wireless%20Broadband/wi4%20WiMAX/_Document/StaticFile/a%20Driving_4G__WiMAX_and_LTE.pdf

[67] Liane Cassavoy. What is 4G Wireless - What Is a 4G Network. [Online]. http://cellphones.about.com/od/frequentlyaskedquestions/f/what-is-4g-wireless.htm

[68] Sascha Segan. (2012, Mayo) WiMAX vs. LTE: Should You Switch? | PCMag.com. [Online]. http://www.pcmag.com/article2/0,2817,2403490,00.asp

[69] Cellular Center. (2011, Noviembre) 4G en Uruguay: Verdades a medias tintas - Cellular Center. [Online]. http://www.celulares.com.uy/blog/11-general/130-4g-en-uruguay-verdades-a-medias-tintas

[70] Emiliano Cotelo. (2011, Noviembre) ¿La tecnología 4G llegó al Uruguay? Informes. [Online]. http://www.espectador.com/1v4_contenido.php?id=226521&sts=1

[71] El País. (2011, Noviembre) Operadores chocan por debut del 4G - Diario EL PAIS - Montevideo - Uruguay. [Online]. http://www.elpais.com.uy/suplemento/empresario/operadores-chocan-por-debut-del-4g/elempre_605460_111111.html

[72] NETUruguay. (2011, Noviembre) Claro lanza tecnología 4G. [Online]. http://neturuguay.com/index.php/noticias/item/922-claro-lanza-tecnolog%C3%ADa-4g

[73] Movistar. Internet Fácil Turbo 5GB. [Online]. http://www.movistar.com.uy/Default.aspx?tabid=462

[74] Antel. (2011, Diciembre) Antel - Institucional - Eventos - Internet Vera: Antel primera en lanzar tecnología LTE en Latinoamérica. [Online]. http://www.antel.com.uy/antel/institucional/sala-de-prensa/eventos/2011/internet-vera-antel-primera-en-lanzar-tecnologia-lte-en-latinoamerica

[75] Antel. Internet Vera: Antel primera en lanzar tecnología LTE en Latinoamérica. [Online]. http://www.antel.com.uy/wps/wcm/connect/e48121804966085896cd9e7c46b5f36a/Gacetilla-lanzamiento-LTE.pdf?MOD=AJPERES&CACHEID=e48121804966085896cd9e7c46b5f36a

[76] Edward Curry, "Message-Oriented Middleware," in *Middleware for Communications*, Qusay Mahmoud, Ed., 2004, ch. 1.

[77] Jorge Escobar. (2009) Push Technology is the Core of the Real Time Web. [Online]. http://jungleg.com/2009/07/07/push-technology-is-the-core-of-the-real-time-web/

[78] Alessandro Alinone. (2011) 10 Years of Push Tecnology, Comet, and WebSockets. [Online]. http://cometdaily.com/2011/07/06/push-technology-comet-and-websockets-10-years-of-history-from-lightstreamers-perspective/

[79] Michael Mahemoff. (2006) HTTP Streaming - Ajax Patterns. [Online]. http://ajaxpatterns.org/HTTP_Streaming

[80] Just Van Den Broecke. (2002) Pushlets - Whitepaper. [Online]. http://www.pushlets.com/doc/whitepaper-all.html

[81] Greg Wilkins. (2007) Comet is Always Better Than Polling. [Online]. http://cometdaily.com/2007/11/06/comet-is-always-better-than-polling/

[82] Ian Paterson, Dave Smith, Peter Saint-Andre, and Jack Moffitt. (2010) Bidirectional-streams Over Synchronous HTTP (BOSH). [Online]. http://xmpp.org/extensions/xep-0124.html

[83] W3C. (2011) The WebSocket API. [Online]. http://www.w3.org/TR/websockets/

[84] Google Inc. (2013, Agosto) Google Cloud Messaging for Android | Android Developers. [Online]. http://developer.android.com/google/gcm/index.html

[85] Apple Inc. (2013, Abril) Local and Push Notification Programming Guide: Apple Push Notification Service. [Online]. https://developer.apple.com/library/mac/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html#//apple_ref/doc/uid/TP40008194-CH100-SW9

[86] Kaazing. Kaazing WebSocket Gateway - HTML5 Edition. [Online]. http://kaazing.com/products/html5-edition

[87] Andy Stanford-Clark and Arlen Nipper. (2011) MQ Telemetry Transport. [Online]. http://mqtt.org/

[88] Joyent Inc. (2013, Setiembre) About Node.js. [Online]. http://nodejs.org/about/

[89] Francis Cianfrocca. (2013, Setiembre) Event Machine. [Online]. http://rubyeventmachine.com/

[90] Twisted community. (2013, Setiembre) Twisted. [Online]. http://twistedmatrix.com/trac/

[91] Joyent Inc. (2013, Setiembre) Node.JS. [Online]. http://nodejs.org/

[92] Joyent Inc. (2013, Setiembre) NPM socket.io. [Online]. https://npmjs.org/package/socket.io

[93] David Walsh. (2010, Noviembre) WebSocket and Socket.IO. [Online]. http://davidwalsh.name/websocket

[94] Guillermo Rauch. (2013, Agosto) Socket.IO: the cross-browser WebSocket for realtime apps. [Online]. http://socket.io/

[95] Guillermo Rauch. (2013, Agosto) Socket.IO: the cross-browser WebSocket for realtime apps. [Online]. http://socket.io/#faq

[96] Guillermo Rauch. (2013, Agosto) Socket.IO: the cross-browser WebSocket for realtime apps. [Online]. http://socket.io/#browser-support

[97] Apple. iOS Human Interface Guidelines: Designing for iOS7. [Online]. https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/

[98] Android Developers. Android Design Guidelines. [Online]. http://developer.android.com/design/index.html

[99] Google. Google Play Developer Program Policies. [Online]. https://play.google.com/about/developer-content-policy.html

[100] Apple. App Review Guidelines. [Online]. https://developer.apple.com/appstore/guidelines.html

[101] Artech Consultores S.R.L. (2013, Agosto) About Artech. [Online]. http://www.genexus.com/company/about-artech?en

[102] Artech Consultores S.R.L. (2012) GeneXus Overview. [Online]. http://www.genexus.com/files/wp-genexus-overview?en

[103] GeneXus. GeneXus, Cross-platform software development. [Online].

http://www.genexus.com/technologies

[104] GeneXus. (2012) Mobile and Smart Devices Development Solution. [Online]. http://www.genexus.com/files/wp-mobile-application-development-genexus.pdf?en

[105] W3C. HTML Current Status. [Online]. http://www.w3.org/standards/techs/html#drafts

[106] Jennifer Niederst Robbins, *HTML & XHTML Pocket Reference*.: O'Reilly, 2009.

[107] Google. HTML5 Rocks. [Online]. http://www.html5rocks.com/en/why

[108] W3C. Cascading Style Sheets (CSS3). [Online]. http://www.w3.org/TR/CSS/#css3

[109] W3C. Geolocation API Specification. [Online]. http://dev.w3.org/geo/api/spec-source.html

[110] jQuery Foundation. (2013, Agosto) jQuery. [Online]. http://jquery.com/

[111] jQuery Foundation. (2013, Agosto) Introduction - jQuery Mobile Demos. [Online]. http://view.jquerymobile.com/1.3.2/dist/demos/intro/

[112] jQuery Foundation. (2013, Agosto) Pages - jQuery Mobile Demos. [Online]. http://view.jquerymobile.com/1.3.2/dist/demos/widgets/pages/

[113] jQuery Foundation. (2013, Agosto) Navigation - jQuery Mobile Demos. [Online]. http://view.jquerymobile.com/1.3.2/dist/demos/widgets/navigation/

[114] Sencha Inc. (2013, Agosto) Mobile HTML5 Framework - Features of Sencha Touch | Features | Sencha Touch | Products | Sencha. [Online]. http://www.sencha.com/products/touch/features/

[115] Sencha Inc. (2013, Agosto) Native Packaging for Mobile Devices. [Online]. http://docs-origin.sencha.com/cmd/3.1.2/#!/guide/native_packaging

[116] Sencha Inc. (2013, Agosto) How to Use Classes in Sencha Touch. [Online]. http://docs.sencha.com/touch/2.2.1/#!/guide/class_system

[117] Sencha Inc. (2013, Agosto) The Sencha Class System | Learn | Sencha. [Online]. http://www.sencha.com/learn/sencha-class-system

[118] Sencha Inc. (2013, Agosto) Using Navigation View in Sencha Touch. [Online]. http://docs.sencha.com/touch/2.2.1/#!/guide/navigation_view

[119] Adobe. PhoneGap. [Online]. http://phonegap.com

[120] Adobe. PhoneGap Build. [Online]. build.phonegap.com/apps

[121] Mozilla. Firebug. [Online]. http://getfirebug.com/

[122] Google. Chrome Developer Tools. [Online]. https://developers.google.com/chrome-developer-tools

[123] Apache Software Foundation. Apache Cordova. [Online]. http://incubator.apache.org/cordova/

[124] ESRI. (2004, Enero) What is ArcGIS. [Online]. http://downloads.esri.com/support/documentation/ao_/698What_is_ArcGIS.pdf

[125] ESRI. Esri Products | Alphabetical Index of All Products. [Online]. http://www.esri.com/products/products-alpha

[126] ESRI. ArcGIS - Features. [Online]. http://www.esri.com/software/arcgis/features

[127] ESRI. ArcGIS for Server | GIS Web Server Software | Web Mapping Server. [Online]. http://www.esri.com/software/arcgis/arcgisserver

[128] ESRI. Servicios SIG | ArcGIS Resource Center. [Online]. http://resources.arcgis.com/es/content/arcgisserver/10.0/gis-services

[129] ESRI. Geodatabase | Storage in a Relational Database Management System (RDBMS). [Online]. http://www.esri.com/software/arcgis/geodatabase/storage-in-an-rdbms

[130] ESRI. ArcSDE Technology. [Online]. http://www.esri.com/software/arcgis/arcsde

[131] ESRI. ArcGIS for Transportation Analytics. [Online]. http://www.esri.com/software/arcgis/arcgis-for-transportation-analytics

[132] ESRI. (2012) ArcGIS for Mobile. [Online]. http://www.esri.com/~/media/Files/Pdfs/library/brochures/pdfs/arcgis-for-mobile.pdf

[133] ESRI. ArcGIS for Mobile | Mobile Solutions that Fit Your Enterprise. [Online]. http://www.esri.com/software/arcgis/about/mobile-gis-for-you

[134] Esri. (2013, Setiembre) ArcGIS API for JavaScript. [Online]. https://developers.arcgis.com/en/javascript/

[135] Esri. (2013, Setiembre) ArcGIS Runtime SDK for Android - Guide. [Online]. https://developers.arcgis.com/en/android/guide/welcome-to-the-help-for-arcgis-runtime-sdk-for-android.htm

[136] Khronos Group. (2013, Setiembre) OpenGL ES - The Standard for Embedded Accelerated 3D Graphics. [Online]. http://www.khronos.org/opengles/

[137] (2013, Setiembre) ArcGIS Runtime SDK for Android - System Requirements. [Online]. https://developers.arcgis.com/en/android/system-reqs.html

[138] Esri. (2013, Setiembre) ArcGIS API for JavaScript Overview. [Online]. https://developers.arcgis.com/en/javascript/jshelp/

[139] Esri. (2013, Setiembre) Get the ArcGIS API for JavaScript. [Online]. https://developers.arcgis.com/en/javascript/jshelp/intro_accessapi.html

[140] Esri. (2013, Setiembre) ArcGIS API for JavaScript - Compact Build. [Online].

https://developers.arcgis.com/en/javascript/jshelp/inside_compactbuild.html

[141] Esri. (2013, Setiembre) ArcGIS API for JavaScript - Working with Dojo. [Online]. https://developers.arcgis.com/en/javascript/jshelp/inside_dojo.html

[142] Dojo Fundation. (2013, Setiembre) Unbeatable JavaScript Tools - The Dojo Toolkit. [Online]. http://dojotoolkit.org/

[143] Esri. (2013, Setiembre) ArcGIS API for JavaScript - Why Dojo? [Online]. https://developers.arcgis.com/en/javascript/jshelp/why_dojo.html

[144] Google. Google Maps. [Online]. https://maps.google.com/

[145] Google. Google Maps API. [Online]. https://developers.google.com/maps/

[146] ESRI. ArcGIS Extension for the Google Maps API. [Online]. http://help.arcgis.com/en/webapi/javascript/gmaps/index.html

[147] Google. Google Chart Tools. [Online]. https://developers.google.com/chart/

[148] OpenStreetMap. OpenStreetMap. [Online]. http://www.openstreetmap.org

[149] Microsoft. (2013, Agosto) Bing Maps. [Online]. http://msdn.microsoft.com/en-us/library/dd877180.aspx

[150] Microsoft. (2013, Setiembre) Bing Maps AJAX Control, Version 7.0. [Online]. http://msdn.microsoft.com/en-us/library/gg427610.aspx

[151] Microsoft. (2013, Setiembre) Bing Maps REST Services. [Online]. http://msdn.microsoft.com/en-us/library/ff701713.aspx

[152] Microsoft. (2013, Setiembre) Bing Maps REST Service. [Online]. http://msdn.microsoft.com/en-us/library/ff701713.aspx

[153] (2013, Setiembre) Viewing Bing Maps Usage. [Online]. http://msdn.microsoft.com/en-us/library/ff859477.aspx

[154] Elmasri-Navathe,., ch. 1.

[155] Ralf Hartmut Güting. (1994, Octubre) An Introduction to Spatial Database Systems. [Online]. http://www.cise.ufl.edu/~mschneid/Research/thesis_papers/Gue94VLDBJ.pdf

[156] E. F. Codd. (1970, Junio) A Relational Model of Data for Large Shared Data Banks. [Online]. http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

[157] The Pennsylvania State University. (2008, Enero) Lesson 4: An Intrduction to Database Management Systems. [Online]. http://www.personal.psu.edu/glh10/ist110/topic/topic07/topic07_05.html

[158] Mike Chapple. (2013, Setiembre) SQL Fundamentals. [Online]. http://databases.about.com/od/sql/a/sqlfundamentals.htm

[159] SQLite. (2013, Setiembre) About SQLite. [Online]. http://www.sqlite.org/about.html

[160] SQLite. (2013, Setiembre) SQLite Is Serverless. [Online]. http://www.sqlite.org/serverless.html

[161] Brian Ritchie. (2010, Agosto) An Introduction to Document Databases. [Online]. http://weblogs.asp.net/britchie/archive/2010/08/12/document-databases.aspx

[162] Google Inc. (2013, Setiembre) Interés en Búsqueda web: nosql. En todo el mundo, 2004 - hoy. [Online]. http://www.google.com/trends/explore?q=nosql#q=nosql&cmpt=q

[163] Brian Ritchie. (2010, Agosto) An Introduction to Document Databases. [Online]. http://weblogs.asp.net/britchie/archive/2010/08/12/document-databases.aspx

[164] Michael Morin. (2013, Setiembre) NoSQL and Document Oriented Databases. [Online]. http://ruby.about.com/od/nosqldatabases/a/nosql1.htm

[165] MongoDB Inc. (2013, Setiembre) MongoDB Overview. [Online]. https://www.mongodb.com/products/mongodb

[166] MongoDB Inc. (2013, Setiembre) JSON and BSON. [Online]. https://www.mongodb.com/json-and-bson

[167] MongoDB Inc. (2013, Setiembre) Aggregation Concepts. [Online]. http://docs.mongodb.org/manual/core/aggregation/

[168] Marc Seeger. (2009, Setiembre) Key-Value stores: a practical overview. [Online]. http://blog.marc-seeger.de/assets/papers/Ultra_Large_Sites_SS09-Seeger_Key_Value_Stores.pdf

[169] Citrusbyte. (2013, Setiembre) Redis. [Online]. http://redis.io/

[170] Citrusbyte. (2013, Setiembre) Redis Clients. [Online]. http://redis.io/clients

[171] C.U.T.C.S.A. (2012) iBus // Ya no lo esperás,ahora sabés cuándo te pasa a buscar. [Online]. http://www.ibus.com.uy/terminos_y_condiciones.html

[172] LG Electronics MobileComm. (2008) IEEE ICC 2008 - Beijing. [Online]. http://to.swang.googlepages.com/ICC2008LBSforMobilessimplifiedR2.pdf

[173] C.U.T.C.S.A. (2012) iBus // Ya no lo esperás,ahora sabés cuándo te pasa a buscar. [Online]. http://www.ibus.com.uy/como_se_usa.html

[174] Vic Gundotra. (2009, Febrero) See where your friends are with Google Latitude. [Online]. http://googleblog.blogspot.com/2009/02/see-where-your-friends-are-with-google.html

[175] Google. (2010) Google Latitude. [Online]. http://www.google.com/intl/es_ALL/mobile/latitude/

[176] Syncromatics. (2012) About Syncromatics. [Online].

http://www.syncromatics.com/syncromatics-bus-tracking.aspx

[177] Official Journal of the European Union. (2012, Julio) DIRECTIVE 2010/40/EU OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 7 July 2010. [Online]. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010:207:0001:0013:EN:PDF

[178] Syncromatics. (2012) Syncromatics: Services: GPS AVL Bus Tracking, Passenger Information Systems, Transit ITS Solutions. [Online]. http://www.syncromatics.com/gps-avl-bus-tracking.aspx

[179] Syncromatics. (2012) Passenger Information Systems. [Online]. http://www.syncromatics.com/services/passenger-information-systems.aspx

[180] Family Safety Production. GPS Tracking Pro - Aplicaciones de Android en Google Play. [Online]. https://play.google.com/store/apps/details?id=com.fsp.android.c#?t=W251bGwsMSwxLDIxMiwiY29tLmZzcC5hbmRyb2lkLmMiXQ.

[181] Greenalp. Real Time GPS Tracker - Aplicaciones de Android en Google Play. [Online]. https://play.google.com/store/apps/details?id=com.greenalp.RealtimeTracker&feature=also_installed

[182] RouteMaster. London Transport Pro - Aplicaciones de Android en Google Play. [Online]. https://play.google.com/store/apps/details?id=com.toson.londontransport#?t=W251bGwsMSwxLDIxMiwiY29tLnRvc29uLmxvbmRvbnRyYW5zcG9ydCJd

[183] FatAttitude Ltd. Bus Checker Londres - Aplicaciones de Android en Google Play. [Online]. https://play.google.com/store/apps/details?id=com.fatattitude.buschecker&feature=search_result

[184] WhatsApp. WhatsApp FAQ. [Online]. http://www.whatsapp.com/faq/

[185] Eric Shepherd. (2013, Abril) Touch - Web API reference | MDN. [Online]. https://developer.mozilla.org/en-US/docs/Web/API/Touch

Instituto de Computación - Facultad de Ingeniería - UdelaR

InTrack

Software Geográfico AVL para Ómnibus

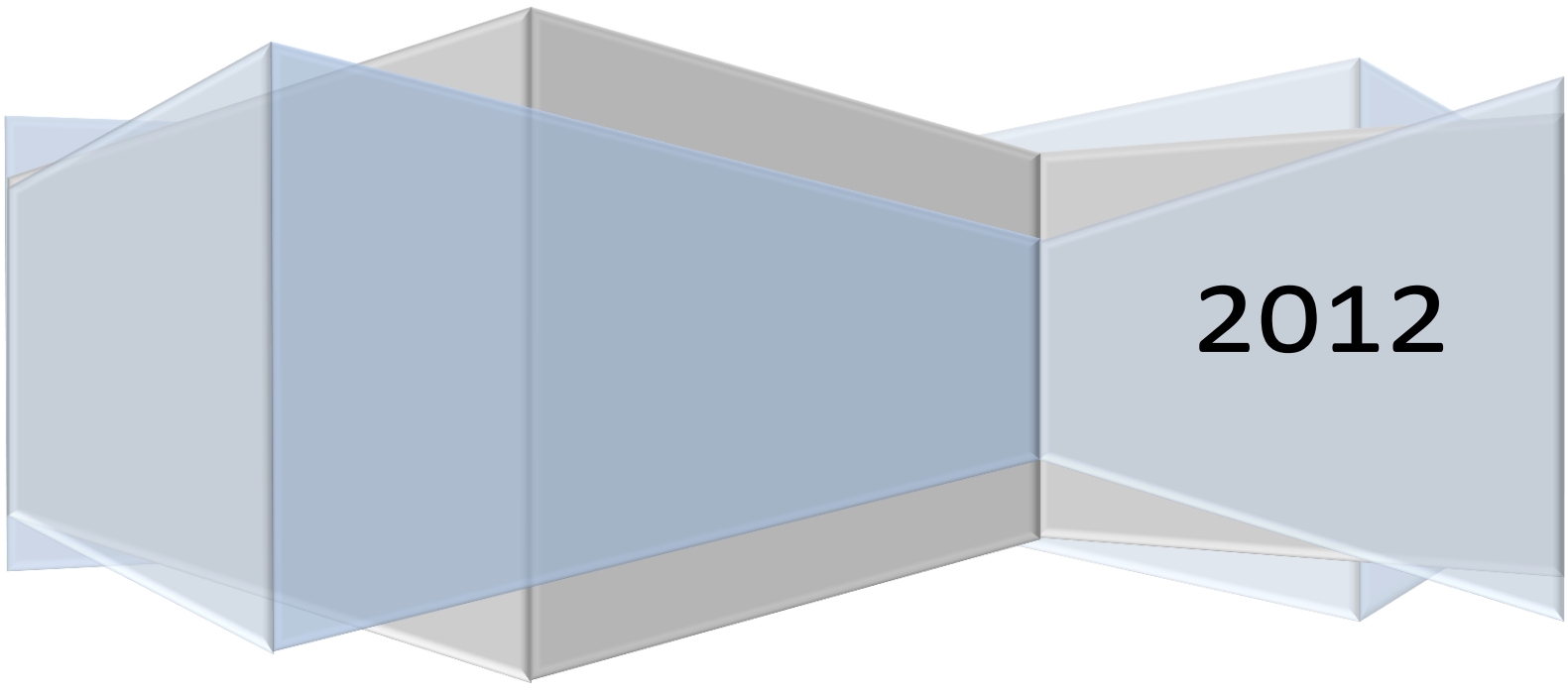
Integrantes

Bruno González, Máximo Mussini

Tutores

Pablo Rebufello, Sandro Moscatelli

Anexo II – Análisis y Especificación de Requerimientos



Contenido

1 Introducción

El objetivo de este documento es el de presentar las principales funcionalidades y características que deberá brindar el producto a desarrollar. Estas han sido identificadas en base a los datos aportados por el cliente en entrevistas realizadas con el mismo.

Este documento sigue los lineamientos de la norma ANSI/IEEE 830 [1] adecuando los mismos al tamaño del proyecto, entendiéndose éste como una aplicación de mediano porte.

El presente documento se encuentra dividido en tres capítulos. Inicialmente se presenta brevemente el proyecto y su alcance, un segundo capítulo en el que se presentan los requerimientos funcionales del sistema, y un último capítulo donde se indican los requerimientos no funcionales que se encontraron durante la etapa de análisis.

1.1 Objetivos y Alcance del Sistema

El principal objetivo del sistema a desarrollar es brindar una plataforma de base que permita que un dispositivo comparta su posición geográfica con otros dispositivos en tiempo real, así como cualquier tipo de datos. Esta plataforma servirá de base para diversas aplicaciones de naturaleza geográfica, dentro de las cuales consideraremos dos grupos funcionales.

Por un lado, un grupo funcional nuclea una red social de usuarios que permite que los usuarios compartan y consulten la posición geográfica de aquellos usuarios del sistema a los que están vinculados. Esta aplicación brindará la posibilidad de que los usuarios establezcan un punto de encuentro, y que se pueda realizar seguimiento mutuo entre usuarios en tiempo real.

En segundo lugar, la misma aplicación permitirá consultar información de ómnibus en tiempo real, con la finalidad de que el usuario pueda obtener tanto la ubicación de un ómnibus como una estimación certera del momento en que va a llegar a una parada determinada.

2 Descripción General

En este capítulo se presenta la descripción general de los requerimientos del proyecto, introduciendo a grandes rasgos lo que se espera de la aplicación, cuáles son las características del público objetivo y qué se puede asumir en relación al hardware y software en el que se ejecutará la misma.

2.1 Funcionalidades

El sistema a construir apunta al intercambio de información geográfica en tiempo real, ya sea para consultar la ubicación exacta de un ómnibus de una determinada línea, o la ubicación de personas en la red de usuarios de la misma. De este sistema se desprenderán dos grupos funcionales que compartirán el ambiente de ejecución, en el que se tendrá en cuenta la información geográfica del usuario que utiliza la aplicación, lo que permitirá brindarle información de utilidad, tanto al momento de tomarse un ómnibus, como para ubicar amigos o encontrarse con alguno de ellos, habiendo establecido previamente un punto de encuentro.

2.1.1 Información de Línea de Ómnibus en Tiempo Real

Una de las funcionalidades que este sistema presentará, será la de poder consultar información de líneas de ómnibus en tiempo real. Básicamente, un usuario deberá poder consultar el tiempo estimado de arribo de cualquier ómnibus de cualquier línea que esté próximo a la ubicación donde este se encuentra. También podrá hacer un seguimiento en tiempo real del ómnibus que desee para obtener información exacta acerca de la ubicación del mismo y podrá obtener información de cómo llegar a la parada más cercana para arribar a su destino en tiempo y forma.

2.1.2 Establecer Punto de Encuentro con Usuarios de la Red

Otra funcionalidad que el sistema deberá brindar, es la de establecer puntos de encuentro con usuarios de la red. Esta funcionalidad permitirá a un usuario definir un punto de encuentro con otro, agregando información multimedia que permita ubicar fácilmente el lugar (por ejemplo: fotos, audio o video). Además, el sistema permitirá establecer alarmas o recordatorios, agendando el encuentro en el dispositivo del usuario y también permitirá hacer un seguimiento en tiempo real de un usuario.

2.2 Características de los Usuarios

Los usuarios a los que apunta el sistema pueden ser de cualquier tipo. Deben tener nociones muy básicas de utilización de mapas y aplicaciones geográficas.

2.3 Restricciones

- La aplicación a desarrollar debe estar disponible para las plataformas Android e iOS, investigando si es posible utilizar GeneXus X Evolution 2 para realizar el desarrollo.
- Deberá evaluarse la posibilidad de desarrollar una aplicación HTML5 que pueda ser accedida desde la mayoría de las plataformas móviles.

- La aplicación apunta a dispositivos móviles que cuenten con módulo GPS/GLONASS, lo que brindará una mejor interacción con el sistema, pudiendo acceder a información geográfica del usuario con una precisión aceptable.

2.4 Supuestos y Dependencias

- La herramienta GeneXus X Evolution 2 tiene la capacidad de integrarse con el SDK de ArcGIS para dispositivos móviles, permitiendo el desarrollo de una aplicación multiplataforma sin necesidad de escribir código nativo.
- Este producto depende del buen funcionamiento de la red de datos 3G/4G, que se utilizará para el intercambio de información. Tanto así es la dependencia que la mayor parte de la experiencia del usuario con la aplicación tendrá que ver con este aspecto.
- Si bien se considerarán restricciones y problemas de seguridad, este no será el foco principal al momento de desarrollar la aplicación.
- Se cuenta con datos geográficos de la ciudad como calles, paradas y recorridos de ómnibus.
- Se cuenta con información de las líneas de ómnibus, como números de línea, variantes del recorrido de una misma línea, y horarios de salida.

3 Requerimientos Funcionales

En este capítulo se presentan los requerimientos funcionales que deberá cumplir el sistema a desarrollar. Los requerimientos se agrupan en categorías para permitir una mejor comprensión de las funcionalidades del sistema y el comportamiento del mismo.

3.1 Requerimientos Funcionales de FindMe!

El primer grupo funcional de la aplicación permitirá que un usuario conforme una red de usuarios o amigos, conformando una red social en la que podrá compartir información geográfica y seguir a sus amigos en tiempo real. La aplicación contará con la administración básica de usuarios y perfiles como la que brinda cualquier red social.

También permitirá que algunas de sus funcionalidades interactúen con el dispositivo, como puede ser, el envío de mensajes de texto, cámara, agenda y libreta de contactos, entre otros.

Dos usuarios que sean amigos podrán establecer un punto de encuentro, permitiendo agendar la cita en la agenda del teléfono, asociando información multimedia relacionada con el punto de encuentro que ayude a identificar el mismo o simplemente brindar información de contexto que sea de utilidad para los usuarios.

3.1.1 Administración de Usuarios

Uno de los objetivos principales del sistema es que cada usuario pueda compartir su posición geográfica con otros usuarios del sistema, pudiendo establecer un vínculo de amistad, conformando así una red social. Por este motivo será fundamental contar con el manejo de cuentas de usuario, con la que el usuario podrá acceder a todos los servicios del sistema.

3.1.1.1 *Registro de Usuario*

Descripción

Este caso de uso permite que un usuario se registre en el sistema, lo cual le habilita a acceder a los servicios que este provee.

Entradas

- Nombre de usuario
- Contraseña
- Confirmación de contraseña

Pre-condiciones

El usuario no está registrado en el sistema.

Flujo Principal

1. El usuario indica al sistema que quiere registrarse.
2. El usuario ingresa nombre de usuario, contraseña, y confirmación de contraseña.
3. El sistema verifica que las contraseñas ingresadas coincidan.
4. El sistema verifica que el nombre de usuario esté disponible.

5. El sistema verifica que los datos ingresados sean correctos y crea una cuenta para el usuario.
6. El sistema notifica al usuario que su cuenta fue creada con éxito, dando la opción de comenzar el caso de uso Inicio de Sesión.
7. Fin de Caso de Uso.

Flujo Alternativo

1.A. Las contraseñas ingresadas no coinciden.
   1. El sistema notifica al usuario, pidiendo que ingrese nuevamente las contraseñas.
   2. Vuelve al punto 2 del flujo principal.

2.A. El nombre de usuario ingresado no está disponible.
   1. El sistema notifica al usuario, pidiendo que ingrese otro nombre de usuario.
   2. Vuelve al punto 2 del flujo principal.

5.A. Ocurrió un error al crear el usuario.
   1. Se notifica al usuario de lo ocurrido.
   2. Fin de caso de uso.

Post-condiciones

El usuario queda registrado en el sistema.

3.1.1.2 *Inicio de Sesión*

Descripción

Este caso de uso permite que un usuario inicie sesión en el sistema, lo cual le permite acceder al resto de los casos de uso del mismo.

Entradas

- Nombre de usuario
- Contraseña

Pre-condiciones

El usuario está registrado en el sistema.

Flujo Principal

1. El usuario indica al sistema que quiere iniciar sesión.
2. El usuario ingresa nombre de usuario y contraseña.
3. El sistema verifica que el nombre de usuario y contraseña sean correctos.
4. El usuario queda autenticado en el sistema.
5. Fin de Caso de Uso.

Flujo Alternativo

3.A. El nombre de usuario no existe.
   1. El sistema notifica al usuario que el nombre de usuario o la contraseña son incorrectos.
   2. Continúa en el punto 2 del flujo principal.

3.B. La contraseña ingresada es incorrecta.
   1. El sistema notifica al usuario que el nombre de usuario o la contraseña son incorrectos.
   2. Continúa en el punto 2 del flujo principal.

Post-condiciones

El usuario queda autenticado en el sistema hasta que cierre sesión o pierda conexión.

3.1.1.3 *Cerrar Sesión*

Descripción

Este caso de uso permite que un usuario cierre la sesión en el sistema.

Entradas

- Nombre de usuario

Pre-condiciones

El usuario inició sesión en el sistema.

Flujo Principal

1. El usuario indica que quiere cerrar sesión.
2. El sistema cierra la sesión del usuario.
3. Fin de Caso de Uso.

Post-condiciones

Se cierra la sesión del usuario.

3.1.1.4 *Editar Perfil de Usuario*

Descripción

Este caso de uso permite que un usuario edite los datos de su perfil. Los datos que pueden editarse son: contraseña, nombre para mostrar, mensaje para mostrar, número de celular y foto de perfil.

Entradas

- Nombre de usuario
- Datos a ser editados

Pre-condiciones

El usuario inició sesión en el sistema.

Flujo Principal

1. El usuario indica que quiere editar su perfil de usuario.
2. El usuario edita los datos que desea y confirma los mismos.
3. El sistema persiste la información actualizada.
4. Fin de Caso de Uso.

Post-condiciones

Se guardan los cambios en el perfil del usuario.

3.1.2 Red de Usuarios

Como se mencionó anteriormente, los usuarios del sistema conforman una red social. Cada usuario tiene un conjunto de usuarios asociados con los cuales comparte su información, en particular su posición geográfica. A este conjunto de usuarios se les denomina la *red* de un usuario, o simplemente su conjunto de amigos.

Será necesario que el sistema provea un listado de usuarios, de forma que cada usuario pueda encontrar a otros usuarios y agregarlos a su red.

3.1.2.1 *Agregar Amigos*

Descripción

Este caso de uso permite realizar una búsqueda de los usuarios registrados en el sistema con el fin de agregarlos a su red de usuarios. La búsqueda se podrá realizar a partir del nombre de usuario o su nick.

Entradas

- Palabra clave de búsqueda

Pre-condiciones

El usuario inició sesión en el sistema.

Flujo Principal

1. El usuario ingresa una o más palabras clave para la búsqueda.
2. El sistema devuelve un conjunto de usuarios que cumplen con el criterio de búsqueda ingresado.
3. El usuario selecciona un usuario para agregar a su red de amigos, iniciando el caso de uso Enviar Solicitud de Amistad**¡Error! La autoreferencia al marcador no es válida.**.
4. Fin de Caso de Uso.

Flujo Alternativo

2.A. No hay usuarios que cumplan con el criterio de búsqueda
   1. El sistema devuelve una lista vacía.
   2. Fin de Caso de Uso.

Post-condiciones

No hay.

3.1.2.2 *Enviar Solicitud de Amistad*

Descripción

Este caso de uso permite que un usuario envíe una solicitud a otro usuario para unirse a su red de amigos.

Entradas

- Nombre de usuario solicitante
- Nombre de usuario destinatario

Pre-condiciones

El usuario inició sesión en el sistema.

Flujo Principal

1. El usuario solicitante ingresa el nombre del usuario a cuya red desea unirse.
2. El sistema verifica que el nombre de usuario exista.
3. El sistema envía una solicitud al usuario especificado para unirse a la red de amigos del usuario solicitante.
4. Fin de Caso de Uso.

Flujo Alternativo

2.A. Nombre de usuario inexistente.

1. El sistema devuelve un error.
2. Fin de Caso de Uso.

Post-condiciones

Se agregó una solicitud de amistad a la lista de solicitudes del usuario seleccionado.

3.1.2.3 *Ver Solicitudes de Amistad*

Descripción

Este caso de uso permite que un usuario visualice todas las solicitudes de amistad recibidas, pudiendo aceptar o rechazar las mismas, y las solicitudes enviadas por él que están pendientes de aprobación.

Entradas

- Nombre de usuario

Pre-condiciones

El usuario inició sesión en el sistema.

Flujo Principal

1. El usuario indica al sistema que quiere ver las solicitudes de amistad.
2. El sistema devuelve una lista de solicitudes recibidas y de solicitudes enviadas pendientes de aprobación.
3. Fin de Caso de Uso.

Flujo Alternativo

2.A. No hay solicitudes recibidas y/o enviadas pendientes de aprobación

1. El sistema devuelve una lista vacía.
2. Fin de Caso de Uso.

Post-condiciones

El sistema muestra en pantalla una lista de las solicitudes recibidas, dando la posibilidad de aceptar o rechazar las mismas, y las solicitudes enviadas pendientes de aprobación.

3.1.2.4 *Aceptar Solicitud de Amistad*

Descripción

Este caso permite que un usuario acepte una solicitud de amistad realizada por otro usuario.

Entradas

- Nombre de usuario que recibió la solicitud
- Nombre de usuario solicitante

Pre-condiciones

El usuario inició sesión en el sistema.

El usuario recibió una solicitud de parte de otro usuario.

Flujo Principal

1. El usuario selecciona la solicitud deseada de su lista de solicitudes recibidas, y confirma la solicitud.
2. El sistema agrega al usuario a la red del usuario que envió la solicitud, y viceversa.
3. El sistema elimina la solicitud de la lista de solicitudes recibidas del usuario.
4. Fin de Caso de Uso.

Post-condiciones

Se estableció una relación de amistad entre los usuarios y se eliminó la solicitud de la lista de solicitudes del usuario.

3.1.2.5 *Rechazar Solicitud de Amistad*

Descripción

Este caso permite que un usuario rechace una solicitud de amistad de otro usuario.

Entradas

- Nombre de usuario que recibió la solicitud
- Nombre de usuario solicitante

Pre-condiciones

El usuario inició sesión en el sistema.

El usuario recibió una solicitud de parte de otro usuario.

Flujo Principal

1. El usuario selecciona la solicitud deseada de su lista de solicitudes recibidas, e indica que desea rechazar la solicitud.

2. El sistema elimina la solicitud de la lista de solicitudes recibidas del usuario.
3. Fin de Caso de Uso.

Post-condiciones

Se eliminó la solicitud de la lista de solicitudes del usuario.

3.1.2.6 *Eliminar Amigo*

Descripción

Este caso permite que un usuario elimine a otro usuario de su red de amigos.

Entradas

- Nombre de usuario
- Nombre de usuario a eliminar

Pre-condiciones

El usuario inició sesión en el sistema.

El usuario es amigo del usuario que desea eliminar.

Flujo Principal

1. El usuario indica que quiere eliminar a un amigo de su red de amigos.
2. El sistema despliega una lista con los amigos de su red.
3. El usuario selecciona el usuario que desea eliminar.
4. El sistema elimina al usuario seleccionado de la red de amigos del usuario y viceversa.
5. Fin de Caso de Uso.

Post-condiciones

Se eliminó el usuario seleccionado de la red de amigos del usuario y viceversa.

3.1.2.7 *Ubicar Amigos*

Descripción

Este caso de uso permite que un usuario reciba la última posición geográfica conocida de los usuarios de su red, pudiendo filtrar por nombre de usuario, etiqueta, o área geográfica.

Entradas

- Palabras clave de búsqueda

Pre-condiciones

El usuario inició sesión en el sistema.

El usuario tiene al menos un amigo en su red de usuarios.

Flujo Principal

1. El usuario indica los criterios de búsqueda para los usuarios de su red.

2. El sistema devolverá aquellos usuarios que concuerden con los criterios de búsqueda, y para cada uno de estos, la última posición geográfica registrada en el sistema.
3. Fin de Caso de Uso.

Flujo Alternativo

1.A. El usuario no ingresa criterios de búsqueda
   1. El sistema devuelve todos los usuarios de su red de usuarios, y para cada uno de estos, la última posición geográfica registrada en el sistema.
   2. Fin de Caso de Uso.

2.A. No hay usuarios que cumplan los criterios de búsqueda ingresados
   1. El sistema indica que no hay usuarios que cumplan con los criterios de búsqueda ingresados.
   2. Fin de Caso de Uso.

Post-condiciones

Se devuelven los usuarios que cumplen con los criterios de búsqueda.

3.1.2.8 *Establecer Punto de Encuentro*

Descripción

Este caso de uso permite que un usuario envíe una invitación a uno de sus amigos para encontrarse en determinada posición geográfica, fecha y hora. También podrá enviar datos adicionales, como pueden ser imágenes o videos.

Entradas

- Nombre de usuario
- Nombre de usuario del amigo
- Posición geográfica del punto de encuentro
- Fecha y hora del encuentro
- Datos adicionales

Pre-condiciones

El usuario inició sesión en el sistema.

Flujo Principal

1. El usuario indica que quiere establecer un punto de encuentro con uno de sus amigos.
2. El sistema le presenta un mapa para permitir que el usuario seleccione la posición geográfica del encuentro.
3. El usuario selecciona en el mapa la ubicación del encuentro, pudiendo determinar la dirección del lugar de forma opcional.
4. El usuario ingresa la fecha, hora, y otros datos adicionales que desee.
5. El sistema envía la invitación al usuario que eligió anteriormente.
6. Fin de Caso de Uso.

Post-condiciones

Se envía la invitación al usuario indicado con los datos ingresados.

3.1.3 Configuración

El sistema cuenta con algunas opciones configurables que están relacionadas con la experiencia de usuario y el consumo de batería de la aplicación. Estas son el intervalo de actualización geográfica y la forma de obtener la coordenada geográfica.

3.1.3.1 *Configuración del Intervalo de Actualización de Posición Geográfica*

Descripción

Este caso de uso permite que un usuario configure el intervalo de tiempo entre las actualizaciones de su posición geográfica. Se podrá configurar tanto el intervalo cuando lo están siguiendo, como cuando no, eligiendo entre los intervalos propuestos por el sistema.

En el caso de que se esté configurando el intervalo de tiempo para cuando no lo están siguiendo, podrá elegir el tiempo de forma manual, teniendo la opción de actualización a demanda.

Que el usuario tenga la posibilidad de configurar diferentes intervalos, le da la libertad de configurar un intervalo más pequeño para cuando lo están siguiendo y otro más grande para cuando no lo están haciendo, lo que disminuirá el consumo de batería de la aplicación.

Entradas

- Nombre de usuario
- Valor de intervalo de tiempo

Pre-condiciones

El usuario inició sesión en el sistema.

Flujo Principal

1. El usuario indica que desea configurar el intervalo de actualización de la posición geográfica.
2. El sistema actualiza el valor del intervalo de actualización.
3. Fin de Caso de Uso.

Flujo Alternativo

1.A. El usuario desea configurar el intervalo de actualización de la posición geográfica cuando lo están siguiendo
   1. El sistema despliega las opciones de intervalos de actualización.
   2. El usuario elige una y confirma.
   3. Continúa en el punto 2 del flujo principal.

1.B. El usuario desea configurar el intervalo de actualización de la posición geográfica cuando no lo están siguiendo
   1. El sistema despliega las opciones de intervalos de actualización.
   2. El usuario elige una o establece que la misma se hará a demanda.

3. Continúa en el punto 2 del flujo principal.

Post-condiciones

Se actualiza el valor del intervalo de tiempo.

3.1.3.2 *Configuración de Obtención de Localización*

Descripción

Este caso de uso permite que un usuario configure el método que el sistema utilizará para obtener su localización geográfica. El usuario podrá elegir entre utilizar el GPS, la red de telefonía celular (triangulación por antenas) o ambas, pudiendo establecer una como preferida.

Entrada

- Nombre de usuario
- Método de obtención de localización
- Método preferido (opcional)

Pre-condiciones

El usuario inició sesión en el sistema.

Flujo Principal

1. El usuario indica que desea configurar el método para obtener su localización.
2. El usuario indica la opción de su preferencia y confirma.
3. El sistema actualiza el método a utilizar para obtener su localización.
4. Fin de Caso de Uso.

Flujo Alternativo

2.A. El usuario elige ambos métodos para la obtención de su localización

1. El usuario elige cuál de ambos métodos será el preferido.
2. Continúa en el punto 3 del flujo principal.

Post-condiciones

Se actualizan las preferencias del usuario.

3.1.4 Transmisión de Datos Geográficos en Tiempo Real

La plataforma a construir deberá brindar la posibilidad de transmitir datos geográficos de un usuario a otro en tiempo real.

Una vez establecida la conexión entre usuarios, el dispositivo del usuario a seguir obtendrá su posición geográfica mediante GPS u otro tipo de geolocalización y la enviará al otro usuario.

3.1.4.1 *Seguir Amigo en Tiempo Real*

Descripción

Este caso de uso permite que un usuario siga a uno de sus amigos en tiempo real.

Entradas

- Nombre de usuario seguidor
- Nombre de usuario a seguir

Pre-condiciones

El usuario seguidor inició sesión en el sistema.

El usuario a seguir es amigo del usuario seguidor.

El usuario a seguir inició sesión en el sistema.

Flujo Principal

1. El usuario selecciona uno de sus amigos para hacer el seguimiento en tiempo real.
2. El usuario a seguir es notificado de que uno de sus amigos quiere comenzar a seguirlo y solicita su confirmación para establecer el seguimiento.
3. El usuario seguido confirma que permite que se establezca la conexión para el seguimiento en tiempo real.
4. El sistema crea una conexión entre el dispositivo del usuario a seguir y el usuario seguidor.
5. El sistema despliega en la pantalla del usuario seguidor la posición del usuario a seguir. Esta posición va siendo actualizada según los intervalos de tiempo configurados por el usuario a seguir en el caso de uso Configuración del Intervalo de Actualización de Posición Geográfica, hasta que el usuario seguidor indica que quiere dejar de seguir a su amigo.
6. Fin de Caso de Uso.

Flujo Alternativo

2.A. El usuario rechaza la solicitud de seguimiento en tiempo real
   1. El sistema notifica al usuario seguidor que su solicitud fue rechazada.
   2. Fin de Caso de Uso.

5.A. El usuario a seguir indica que quiere dejar de ser seguido
   1. El sistema notifica de este suceso al usuario seguidor y finaliza la conexión.
   2. Fin de Caso de Uso.

5.B. El usuario a seguir cierra sesión del sistema
   1. El sistema notifica de este suceso al usuario seguidor y finaliza la conexión.
   2. Fin de Caso de Uso.

Post-condiciones

No hay.

3.1.5 Asistencia al Usuario

La aplicación móvil deberá contemplar ciertas funcionalidades que faciliten la adopción del sistema. Una de las tareas más engorrosas a la que se enfrentan los usuarios al ingresar a una red social es encontrar a sus amigos y agregarlos a su red, por lo que se proveerá la posibilidad de agregar usuarios a partir de los contactos del dispositivo móvil.

Para simplificar la búsqueda de usuarios en la red, cada usuario podrá etiquetar a los usuarios de su red de forma de poder agruparlos como crea conveniente, pudiendo luego buscar filtrando por etiquetas.

Finalmente, dado que la aplicación está pensada para permitir conocer la ubicación de otros usuarios, cada usuario que establezca un punto de encuentro con usuarios de su red, podrá agregarlo a la agenda de su dispositivo móvil.

3.1.5.1 *Enviar Solicitudes a Contactos del Dispositivo*

Descripción

Este caso de uso facilita al usuario la tarea de agregar a sus amigos a su red, permitiéndole enviar una solicitud a los contactos de la agenda de su dispositivo móvil.

Se utilizará la agenda de contactos del dispositivo y se realizará una búsqueda en el sistema a partir del número de teléfono del contacto, presentándole al usuario una lista con aquellos contactos que ya están registrados en el sistema.

Entrada

- Nombre de usuario
- Lista de nombres de usuarios a los que se enviarán solicitudes

Pre-condiciones

El usuario inició sesión en el sistema.

Flujo Principal

1. El usuario indica que quiere enviar solicitudes de unión a la red de los contactos de la agenda de su dispositivo.
2. El sistema realiza la búsqueda por número de teléfono del contacto y muestra en pantalla aquellos usuarios que están registrados en el sistema.
3. El usuario indica los usuarios a los que desea enviarles solicitud de unión a la red.
4. El sistema envía las solicitudes a los usuarios indicados.
5. Fin de Caso de Uso.

Post-condiciones

Se envían las solicitudes a los usuarios indicados.

3.1.5.2 *Etiquetar Amigo*

Descripción

Este caso de uso permite que un usuario asigne una etiqueta a sus amigos lo cual podrá ser utilizado como criterio de búsqueda en el caso de uso Ubicar Amigos.

Entradas

- Nombre de usuario
- Nombre de usuario a etiquetar
- Nombre de la etiqueta

Pre-condiciones

El usuario inició sesión en el sistema.

Flujo Principal

1. El usuario selecciona al amigo que quiere etiquetar.
2. El sistema despliega las etiquetas guardadas por el usuario.
3. El usuario asigna la etiqueta deseada.
4. El sistema guarda las opciones del usuario.
5. Fin de Caso de Uso.

Flujo Alternativo

2.A. El usuario desea crear una nueva etiqueta
   1. El usuario indica el nombre de la nueva etiqueta.
   2. Se da de alta la etiqueta en el sistema.
   3. Se asigna dicha etiqueta al usuario.
   4. Vuelve al punto 3.

Post-condiciones

Se crean las etiquetas (si corresponde), y se asignan las etiquetas seleccionadas al amigo seleccionado.

3.1.5.3 *Agendar Punto de Encuentro*

Descripción

Este caso de uso permite que un usuario agregue una invitación de punto de encuentro a la agenda de su dispositivo.

Entradas

- Posición geográfica del punto de encuentro
- Fecha y hora del encuentro
- Datos adicionales

Pre-condiciones

El usuario inició sesión en el sistema.

El usuario recibió una invitación de establecimiento de punto de encuentro.

Flujo Principal

1. El usuario indica que quiere visualizar las invitaciones recibidas.
2. El sistema despliega en pantalla las invitaciones, dando la opción de agregar el encuentro a la agenda.
3. El usuario indica que quiere agregar el encuentro a la agenda.
4. El sistema agrega el encuentro a la agenda del dispositivo del usuario.
5. Fin de Caso de Uso.

Post-condiciones

Se agrega el encuentro a la agenda del dispositivo del usuario, en la fecha y hora del encuentro.

3.2 Requerimientos Funcionales de ¿Cuándo llega?

El segundo grupo funcional de la aplicación permitirá que un usuario pueda consultar información de líneas de ómnibus en tiempo real. Básicamente, un usuario podrá consultar el tiempo estimado de arribo de un ómnibus de una línea determinada, el cual se calculará a partir de las posiciones geográficas actuales de los ómnibus, obteniendo así una estimación mucho más acertada que una basada en estadísticas de recorridos pasados.

Para realizar una consulta el usuario solo deberá indicar la parada en la cual desea tomar el ómnibus, y seleccionar una de las líneas que pasa por allí. Interesa que la aplicación sea sencilla de utilizar, por lo que deberá ser posible utilizar la posición geográfica del usuario para mostrarle aquellas paradas de ómnibus que se encuentran próximas a él.

También interesa que se pueda acceder rápidamente a la información de ómnibus, por lo que se ofrecerá la posibilidad de recordar paradas como favoritas, y de ver las últimas consultas de líneas de ómnibus realizadas.

Para realizar las estimaciones de llegada de los ómnibus será necesario que cada ómnibus transmita su posición geográfica en tiempo real al sistema. Se asume que cada ómnibus tendrá un dispositivo con sensor geográfico y acceso a Internet, y la capacidad de correr una aplicación que se comunica con el sistema.

Se considerará que un ómnibus está *activo* desde el momento en que comienza a realizar su recorrido hasta el momento en que llega a destino, y solo transmitirá su posición durante este intervalo de tiempo. Esto es necesario para que los ómnibus que están fuera de servicio no interfieran con las estimaciones.

3.2.1 Requerimientos Funcionales de la Aplicación Móvil

En esta sección se presentan los requerimientos de la aplicación móvil que utilizará el usuario para consultar información de ómnibus, y las funcionalidades que deberá proveer el sistema para soportar los casos de uso.

3.2.1.1 *¿Cuándo llega?*

Descripción

Este caso de uso permite que un usuario obtenga una estimación del momento en el que un ómnibus de una línea particular llega a una parada determinada.

Entradas

- Identificador de la parada
- Identificador de la línea de ómnibus
- Posición geográfica del usuario (opcional)

Pre-condiciones

El identificador de la parada corresponde a una parada existente.

El identificador de la línea de ómnibus corresponde a una línea de ómnibus existente.

La parada pertenece al recorrido de la línea de ómnibus.

Flujo Principal

1. El usuario indica que desea consultar información sobre ómnibus.
2. El sistema le muestra al usuario un mapa que indica las paradas de ómnibus.
3. El usuario selecciona una parada en el mapa.
4. El sistema le devuelve una lista de líneas de ómnibus que paran en la parada seleccionada.
5. El usuario elige una línea de ómnibus.
6. El sistema agrega la parada y línea de ómnibus seleccionada a la lista de consultas recientes del usuario.
7. El sistema consulta las posiciones de los ómnibus pertenecientes a la línea seleccionada que están activos (actualmente realizando su recorrido) y descarta aquellos que ya han pasado la parada, obteniendo los dos más cercanos a la misma.
8. El sistema establece una conexión entre el usuario y los dos próximos ómnibus de la línea que van a llegar a la parada, permitiendo consultar el tiempo estimado de arribo en horas y minutos. La conexión con cada uno de los ómnibus persiste hasta que el usuario indica que quiere cerrar la misma o hasta que el ómnibus que está siendo seguido pasa por la parada que indicó el usuario.
9. Fin de caso de uso.

Flujo Alternativo

3.A. Es la primera vez que el usuario entra.

1. El sistema le pregunta al usuario si desea utilizar su posición geográfica para buscar la parada.
2. El usuario elige la opción que desea.

El sistema guarda la preferencia del usuario, la cual a partir de ese momento se utilizará utilizará por defecto. Esta podrá luego ser modificada en Descripción

Este caso de uso permite que un usuario agregue una parada a sus favoritos, para acceder a la información de las líneas de ómnibus que paran en la misma de forma más rápida y conveniente.

Entradas

- Identificador de la parada

Pre-condiciones

El identificador de la parada corresponde a una parada existente.

La parada no pertenece a los favoritos del usuario.

Flujo Principal

1. El usuario selecciona una parada en el mapa e indica que desea agregarla a favoritos.

2. El sistema agrega la parada a los favoritos del usuario, y le notifica al usuario.
3. Fin de caso de uso.

Post-condiciones

La parada pertenece a los favoritos del usuario.

3.2.1.2 *Eliminar Parada de Favoritos*

Descripción

Este caso de uso permite que un usuario elimine una parada de sus favoritos.

Entradas

- Identificador de la parada

Pre-condiciones

El identificador de la parada corresponde a una parada existente.
La parada pertenece a los favoritos del usuario.

Flujo Principal

1. El usuario selecciona una parada en el mapa, e indica que desea eliminarla de sus favoritos.
2. El sistema elimina la parada de los favoritos del usuario, y le notifica al usuario.
3. Fin de caso de uso.

Flujo Alternativo

1.A. El usuario selecciona una parada de su lista de favoritos, e indica que desea eliminarla.
   1. Continúa en el paso 2 del flujo principal.

Post-condiciones

La parada no pertenece a los favoritos del usuario.

3.2.1.3 *Ver Favoritos*

Descripción

Este caso de uso permite que un usuario administre sus paradas de ómnibus favoritas, pudiendo eliminarlas, y pasar a consultar información sobre las líneas de ómnibus correspondientes a las mismas. Los favoritos se almacenan de forma local al dispositivo del usuario.

Entradas

Ninguna.

Pre-condiciones

Ninguna.

Flujo Principal

1. El usuario indica que desea ver sus favoritos.
2. El sistema le presenta una lista con sus paradas de ómnibus favoritas.
3. El usuario selecciona uno de sus favoritos e indica que desea consultar información de ómnibus.
4. Continúa en el punto 4 del flujo principal del caso de uso ¿Cuándo llega?.
5. Fin de caso de uso.

Flujo Alternativo

3.A. El usuario selecciona una parada de su lista de favoritos, e indica que desea eliminarla.
   1. Comienza el caso de uso Eliminar Parada de Favoritos.
   2. Fin de caso de uso.

3.B. El usuario no desea consultar información de ómnibus.
   1. Fin de caso de uso.

Post-condiciones

No hay.

3.2.1.4 *Ver Consultas Recientes*

Descripción

Este caso de uso permite que un usuario pueda ver las últimas consultas realizadas en el caso de uso ¿Cuándo llega?, las cuales consisten en una parada y una línea de ómnibus particulares.

Entradas

Ninguna.

Pre-condiciones

Ninguna.

Flujo Principal

1. El usuario indica que desea ver sus consultas recientes.
2. El sistema le presenta una lista con las últimas consultas realizadas por el usuario, donde cada consulta está compuesta por una parada y una línea de ómnibus.
3. El usuario selecciona una de las consultas e indica que desea consultar información de ómnibus.
4. Continúa en el punto 7 del flujo principal del caso de uso ¿Cuándo llega?.
5. Fin de caso de uso.

Flujo Alternativo

3.A. El usuario no desea consultar información de ómnibus.
   1. Fin de caso de uso.

3.B. El usuario indica que desea eliminar la información de consultas recientes.
   1. El sistema elimina todas las consultas recientes del usuario.
   2. Fin de caso de uso.

Post-condiciones

No hay.

1. Configurar Preferencias de .

7.B. El usuario desea utilizar su posición geográfica para ver la parada.

1. El sistema obtiene la posición geográfica del usuario.
2. El sistema centra el mapa en la posición del usuario.

8.A. El usuario desea agregar la parada a Favoritos.

Comienza el caso de uso Descripción

Este caso de uso permite que un usuario obtenga la ruta óptima a pie hacia la parada en la que desea tomar el ómnibus.

Entradas

- Identificador de la parada
- Posición geográfica del usuario (opcional)

Pre-condiciones

El identificador de la parada corresponde a una parada existente.

Flujo Principal

1. El usuario indica la parada en la que tomará el ómnibus.
2. El sistema obtiene la posición geográfica del usuario.
3. El sistema calcula la ruta óptima desde el lugar en donde se encuentra el usuario hasta la parada seleccionada.
4. El sistema muestra en pantalla la ruta obtenida.
5. Fin de caso de uso.

Flujo Alternativo

3.A. El usuario desea indicar su posición geográfica en el mapa para realizar el cálculo.

1. El usuario indica su posición geográfica en el mapa.
2. Continúa en el paso 3 del flujo principal.

Post-condiciones

No hay.

1. Agregar Parada a Favoritos.
2. Continúa en el paso 4 del flujo principal.

10.B. El usuario desea eliminar la parada de Favoritos.

1. Comienza el caso de uso Eliminar Parada de Favoritos.
2. Continúa en el paso 4 del flujo principal.

12.A. El usuario indicó en sus preferencias que no desea que se recuerden sus recientes.

1. No se guarda la consulta y continúa en el paso 7 del flujo principal.

13.A. El usuario proporcionó su posición geográfica en un paso previo.

1. El sistema descarta aquellos ómnibus que el usuario no llegará a tomar suponiendo que camina desde el lugar en el que se encuentra hasta la parada que eligió.
2. Continúa en el paso 8 del flujo principal.

14.A. No hay ómnibus activos yendo hacia la parada del usuario

1. El sistema devuelve la hora de salida del próximo ómnibus de la línea y el tiempo estimado que le tomará llegar hasta la parada indicada.
2. Fin de caso de uso.

Post-condiciones

Se creó una conexión entre los dos ómnibus de la línea que están más próximos a la parada que indicó el usuario y el usuario que realizó la solicitud.

3.2.1.5 *Dejar de Seguir Ómnibus*

Descripción

Este caso de uso permite que un usuario cierre una conexión establecida con un ómnibus devuelto en el caso de uso ¿Cuándo llega?

Entradas

- Identificador del ómnibus

Pre-condiciones

El identificador del ómnibus corresponde a un ómnibus que el usuario está siguiendo.

Flujo Principal

1. El usuario indica el ómnibus que desea dejar de seguir.
2. El sistema cierra la conexión entre el usuario y el ómnibus.
3. Fin de caso de uso.

Post-condiciones

Se cerró la conexión entre el ómnibus indicado y el usuario que realizó la solicitud.

3.2.1.6 *Determinar Ruta hacia Parada*

Descripción

Este caso de uso permite que un usuario obtenga la ruta óptima a pie hacia la parada en la que desea tomar el ómnibus.

Entradas

- Identificador de la parada
- Posición geográfica del usuario (opcional)

Pre-condiciones

El identificador de la parada corresponde a una parada existente.

Flujo Principal

6. El usuario indica la parada en la que tomará el ómnibus.
7. El sistema obtiene la posición geográfica del usuario.
8. El sistema calcula la ruta óptima desde el lugar en donde se encuentra el usuario hasta la parada seleccionada.
9. El sistema muestra en pantalla la ruta obtenida.
10. Fin de caso de uso.

Flujo Alternativo

5.A. El usuario desea indicar su posición geográfica en el mapa para realizar el cálculo.

1. El usuario indica su posición geográfica en el mapa.
2. Continúa en el paso 3 del flujo principal.

Post-condiciones

No hay.

3.2.1.7 *Agregar Parada a Favoritos*

Descripción

Este caso de uso permite que un usuario agregue una parada a sus favoritos, para acceder a la información de las líneas de ómnibus que paran en la misma de forma más rápida y conveniente.

Entradas

- Identificador de la parada

Pre-condiciones

El identificador de la parada corresponde a una parada existente.

La parada no pertenece a los favoritos del usuario.

Flujo Principal

4. El usuario selecciona una parada en el mapa e indica que desea agregarla a favoritos.
5. El sistema agrega la parada a los favoritos del usuario, y le notifica al usuario.
6. Fin de caso de uso.

Post-condiciones

La parada pertenece a los favoritos del usuario.

3.2.1.8 *Eliminar Parada de Favoritos*

Descripción

Este caso de uso permite que un usuario elimine una parada de sus favoritos.

Entradas

- Identificador de la parada

Pre-condiciones

El identificador de la parada corresponde a una parada existente.
La parada pertenece a los favoritos del usuario.

Flujo Principal

4. El usuario selecciona una parada en el mapa, e indica que desea eliminarla de sus favoritos.
5. El sistema elimina la parada de los favoritos del usuario, y le notifica al usuario.
6. Fin de caso de uso.

Flujo Alternativo

5.B. El usuario selecciona una parada de su lista de favoritos, e indica que desea eliminarla.
   1. Continúa en el paso 2 del flujo principal.

Post-condiciones

La parada no pertenece a los favoritos del usuario.

3.2.1.9 *Ver Favoritos*

Descripción

Este caso de uso permite que un usuario administre sus paradas de ómnibus favoritas, pudiendo eliminarlas, y pasar a consultar información sobre las líneas de ómnibus correspondientes a las mismas. Los favoritos se almacenan de forma local al dispositivo del usuario.

Entradas

Ninguna.

Pre-condiciones

Ninguna.

Flujo Principal

6. El usuario indica que desea ver sus favoritos.
7. El sistema le presenta una lista con sus paradas de ómnibus favoritas.
8. El usuario selecciona uno de sus favoritos e indica que desea consultar información de ómnibus.
9. Continúa en el punto 4 del flujo principal del caso de uso ¿Cuándo llega?.
10. Fin de caso de uso.

Flujo Alternativo

6.A. El usuario selecciona una parada de su lista de favoritos, e indica que desea eliminarla.
   1. Comienza el caso de uso Eliminar Parada de Favoritos.

2. Fin de caso de uso.

6.B. El usuario no desea consultar información de ómnibus.

1. Fin de caso de uso.

Post-condiciones

No hay.

3.2.1.10 *Ver Consultas Recientes*

Descripción

Este caso de uso permite que un usuario pueda ver las últimas consultas realizadas en el caso de uso ¿Cuándo llega?, las cuales consisten en una parada y una línea de ómnibus particulares.

Entradas

Ninguna.

Pre-condiciones

Ninguna.

Flujo Principal

6. El usuario indica que desea ver sus consultas recientes.
7. El sistema le presenta una lista con las últimas consultas realizadas por el usuario, donde cada consulta está compuesta por una parada y una línea de ómnibus.
8. El usuario selecciona una de las consultas e indica que desea consultar información de ómnibus.
9. Continúa en el punto 7 del flujo principal del caso de uso ¿Cuándo llega?.
10. Fin de caso de uso.

Flujo Alternativo

6.A. El usuario no desea consultar información de ómnibus.

1. Fin de caso de uso.

6.B. El usuario indica que desea eliminar la información de consultas recientes.

1. El sistema elimina todas las consultas recientes del usuario.
2. Fin de caso de uso.

Post-condiciones

No hay.

3.2.1.11 *Configurar Preferencias de Usuario*

Descripción

Este caso de uso permite que un usuario configure sus preferencias, como si se debería utilizar su ubicación geográfica al realizar una búsqueda de paradas, o si se desea que la aplicación recuerde las últimas consultas de ómnibus realizadas.

Entradas

Ninguna.

Pre-condiciones

Ninguna.

Flujo Principal

1. El usuario indica que desea configurar sus preferencias.
2. El sistema le muestra al usuario dos casillas de verificación, una para que indique si desea utilizar su ubicación geográfica para localizar paradas cercanas, y otra para que indique si desea que la aplicación recuerde las últimas consultas de ómnibus realizadas en el caso de uso ¿Cuándo llega?.
3. El usuario modifica sus preferencias.
4. El usuario indica que desea guardar los cambios.
5. El sistema guarda las nuevas preferencias del usuario.
6. Fin de caso de uso.

Flujo Alternativo

4.A. El usuario indica que desea descartar los cambios.
   1. Fin de caso de uso.

Post-condiciones

Se actualizaron las preferencias del usuario.

3.2.2 Requerimientos Funcionales de la Aplicación de Ómnibus

En esta sección se presentan los requerimientos de la aplicación que estará presente en todos los ómnibus a ser rastreados por el sistema. Antes de comenzar el recorrido el guarda o chofer del ómnibus deberá indicar la línea, sub-línea y variante a la que pertenece el ómnibus.

El chofer o guarda le indicará al sistema cuando comienza el recorrido mediante un simple botón en la aplicación, activando así la transmisión de datos, y cuando llegue a destino indicará que desea detener la transmisión de forma similar.

El sistema contará con una lista de dispositivos habilitados para utilizar la aplicación de ómnibus como una medida de seguridad para reducir las chances de que algún impostor pueda actualizar información de ómnibus de forma incorrecta.

3.2.2.1 *Configuración de Línea*

Descripción

Este caso de uso permite que el chofer o guarda de un ómnibus seleccione la línea de ómnibus de la cual va a realizar el recorrido. Esto permitirá que al comenzar el recorrido la información se actualice correctamente en el sistema.

Entradas

- Número de línea
- Nombre de sub-línea

Pre-condiciones

Ninguna.

Flujo Principal

1. El usuario indica que desea configurar la línea del ómnibus.
2. El sistema le muestra al usuario un conjunto con los números de las líneas de ómnibus existentes en la ciudad.
3. El usuario selecciona la línea de ómnibus deseada.
4. El sistema le presenta al usuario un conjunto de variantes pertenecientes a la línea de ómnibus seleccionada.
5. El usuario selecciona una sub-línea.
6. El sistema recuerda la línea de ómnibus y el nombre de la sub-línea seleccionada a utilizar en el caso de uso Comenzar Recorrido.
7. Fin de caso de uso.

Flujo Alternativo

* El usuario desea cancelar.

1. Se deshabilita la posibilidad de comenzar el caso de uso Comenzar Recorrido.
2. Fin de caso de uso.

Post-condiciones

El sistema recuerda la línea y el nombre de la sub-línea.

3.2.2.2 *Comenzar Recorrido*

Descripción

Este caso de uso permite que el chofer o guarda de un ómnibus indique que ha comenzado a realizar el recorrido, iniciando la transmisión de actualizaciones de la posición geográfica del vehículo.

Entradas

- Número de línea
- Nombre de sub-línea
- Identificador interno del dispositivo

Pre-condiciones

Se ha seleccionado una línea en Configuración de Línea.

Flujo Principal

1. El usuario indica que desea comenzar el recorrido de ómnibus.
2. El sistema verifica que el identificador interno del dispositivo pertenece a uno de los dispositivos habilitados para ómnibus.
3. El sistema verifica que el número de línea y nombre de sub-línea previamente configurados sean correctos.
4. El sistema crea un nombre compuesto a partir del número de línea, el nombre de sub-línea, y el identificador interno del dispositivo, dispuestos de forma jerárquica (ej. "/bus/117/PUNTA CARRETAS/6546841684").
5. Comienza el caso de uso Abrir Canal de Transmisión con el nombre antes mencionado.
6. El sistema se suscribe a las actualizaciones del canal creado, sobre-escribiendo la información de cada actualización en la base de datos.
7. La aplicación de ómnibus comenzará el caso de uso Actualizar Ubicación Geográfica de forma regular cada dos segundos (u otro intervalo similar), hasta que el usuario decida finalizar el recorrido.
8. Fin de caso de uso.

Flujo Alternativo

2.A. El dispositivo no está habilitado a utilizar la aplicación.

1. El sistema notifica al usuario de que el dispositivo necesita ser registrado.
2. Fin de caso de uso.

3.A. La línea no es correcta.

1. El sistema notifica al usuario que la línea es incorrecta.
2. Comienza el caso de uso Configuración de Línea.
3. Fin de caso de uso.

Post-condiciones

El sistema abrió un canal y la aplicación comenzó a transmitir la posición geográfica del dispositivo en intervalos regulares.

3.2.2.3 *Finalizar Recorrido*

Descripción

Este caso de uso permite que el chofer o guarda de un ómnibus indique que ha finalizado el recorrido de ómnibus a realizar.

Entradas

- Número de línea
- Nombre de sub-línea
- Identificador interno del dispositivo

Pre-condiciones

Se ha comenzado el recorrido.

Hay un canal de transmisión abierto para la línea seleccionada y el dispositivo utilizado.

Flujo Principal

1. El usuario indica que desea finalizar el recorrido de ómnibus.
2. Comienza el caso de uso Cerrar Canal de Transmisión, utilizando como nombre del canal un nombre compuesto a partir del número de línea, el nombre de sub-línea, y el identificador interno del dispositivo, dispuestos de forma jerárquica (ej. "/bus/117/PUNTA CARRETAS/6546841684").
3. El sistema deja de recordar el nombre de la sub-línea, para forzar al usuario a que configure la línea nuevamente ya este es el caso más común.
4. Fin de caso de uso.

Post-condiciones

El sistema cerró el canal de transmisión, y ya no recuerda la sub-línea de ómnibus.

3.3 Requerimientos Funcionales de la Plataforma

La plataforma deberá proveer un mecanismo de transmisión de datos en tiempo real que permita soportar tanto el caso de uso 3.1.4.1 en el caso de *FindMe!*, así como la actualización de la posición geográfica de los ómnibus a ser utilizada para realizar los cálculos en el caso de uso *¿Cuándo llega?*

A pesar de que ambas funcionalidades necesitan transmitir los datos de forma rápida y continua, el ciclo de vida de las transmisiones es diferente en cada una. Una línea de ómnibus comenzará a transmitir actualizaciones de su posición geográfica cuando comienza el recorrido y lo seguirá haciendo hasta llegar a destino, mientras que un usuario comenzará a transmitir su posición sólo cuando uno de sus amigos decida seguirlo.

Consideraremos que para cada transmisión se crea un *canal* donde se publicarán las actualizaciones geográficas. La plataforma permitirá la creación dinámica de canales, brindando también funcionalidades de publicación y suscripción a dichos canales.

Cada canal tiene un nombre que se trata como una jerarquía, utilizando una barra (/) como separador, de forma similar a un sistema de archivos. Esto permite agrupar la información de varios canales relacionados (ej. De dos vehículos de la misma línea de ómnibus).

De esta forma, cada vez que comienza un seguimiento en tiempo real se creará un canal donde el usuario a seguir publicará actualizaciones geográficas, mientras que el usuario que lo sigue estará suscrito a dicho canal recibiendo así las actualizaciones geográficas. Esto ocurre de forma transparente para los usuarios de *FindMe!*, el sistema es el encargado de establecer esta conexión entre los usuarios.

En *¿Cuándo llega?* se crea un canal en el instante que un ómnibus empieza a realizar su recorrido, pero en este caso es el sistema el que está suscrito al mismo, almacenando la información geográfica en una base de datos sobre la cual luego se realizarán cálculos geográficos a demanda.

Al crear un canal se crean dos claves, una es requerida para publicar en el canal y la otra para poder suscribirse. Esto tiene como fin poder controlar que sólo quien crea el canal pueda publicar en el mismo, y que tenga control sobre quiénes pueden acceder a la información del canal otorgándoles la clave de suscripción. Estas claves son manejadas internamente por las aplicaciones.

3.3.1.1 *Abrir Canal de Transmisión*

Descripción

Este caso de uso permite que un usuario cree un canal en el cual puede compartir su ubicación geográfica y otros datos, lo cual permite que cualquiera que conozca el nombre del canal y conozca de clave acceso al mismo pueda recibir esa información.

Entradas

- Nombre del canal

Salidas

- Clave de publicación
- Clave de suscripción

Pre-condiciones

No existe un canal abierto con el mismo nombre.

Flujo Principal

1. El usuario indica al sistema que desea abrir un canal indicando el nombre del mismo.
2. El sistema genera una clave de publicación y una clave de suscripción, las que será necesario proveer para poder publicar y suscribirse al canal respectivamente.
3. El sistema crea un canal bajo el nombre indicado que responde a las claves generadas.
4. El sistema devuelve las claves al usuario.
5. Fin de Caso de Uso.

Post-condiciones

Se abrió un canal de transmisión que ahora puede recibir publicaciones y suscripciones.

3.3.1.2 *Actualizar Ubicación Geográfica*

Descripción

Este caso de uso permite que un usuario publique información geográfica o datos de otro tipo en un canal, de forma que pueda ser recibida por varios usuarios.

Entradas

- Nombre del canal
- Clave de publicación
- Coordenada geográfica u otros datos

Pre-condiciones

Ninguna.

Flujo Principal

1. El usuario indica al sistema que desea actualizar la información de un canal, indicando el nombre del canal, la clave de publicación del canal, y los datos que desea publicar.
2. El sistema verifica que exista un canal abierto bajo el nombre indicado.
3. El sistema verifica que la clave de publicación sea correcta.
4. El sistema envía la información publicada a todos los usuarios suscritos al canal.
5. Fin de Caso de Uso.

Flujo Alternativo

2.A. No existe un canal abierto con el nombre indicado.
   1. El sistema notifica al usuario.
   2. Fin de caso de uso.

3.A. La clave de publicación es inválida.
   1. El sistema notifica al usuario.
   2. Fin de caso de uso.

Post-condiciones

Se envió la actualización de la información a todos los usuarios suscritos al canal.

3.3.1.3 *Suscribirse a Canal de Transmisión*

Descripción

Este caso de uso permite que un usuario se suscriba a un canal para poder recibir las actualizaciones de la información allí publicada.

Entradas

- Nombre del canal
- Clave de suscripción

Pre-condiciones

Ninguna.

Flujo Principal

1. El usuario indica al sistema que desea suscribirse a un canal, indicando el nombre y la clave de suscripción al canal.
2. El sistema verifica que exista un canal abierto bajo el nombre indicado.
3. El sistema verifica que la clave de suscripción sea correcta.
4. El sistema suscribe al usuario al canal, y le enviará información cuando ocurra una actualización.
5. Fin de Caso de Uso.

Flujo Alternativo

2.A. No existe un canal abierto con el nombre indicado.
1. El sistema notifica al usuario.
2. Fin de caso de uso.

3.A. La clave de suscripción es inválida.
1. El sistema notifica al usuario.
2. Fin de caso de uso.

Post-condiciones

El usuario está suscrito al canal.

3.3.1.4 *Cerrar Canal de Transmisión*

Descripción

Este caso de uso permite que un usuario cierre un canal de transmisión.

Entradas

- Nombre del canal
- Clave de publicación

Pre-condiciones

Ninguna.

Flujo Principal

1. El usuario indica al sistema que desea cerrar un canal, indicando el nombre y la clave de publicación del canal.
2. El sistema verifica que exista un canal abierto bajo el nombre indicado.
3. El sistema verifica que la clave de publicación sea correcta.
4. El sistema notifica a todos los usuarios suscritos al canal que este ha dejado de transmitir.
5. El sistema cierra el canal y elimina la lista de suscriptores del mismo.
6. Fin de Caso de Uso.

Flujo Alternativo

2.A. No existe un canal abierto con el nombre indicado.
1. El sistema notifica al usuario.
2. Fin de caso de uso.

3.A. La clave de publicación es inválida.
1. El sistema notifica al usuario.
2. Fin de caso de uso.

Post-condiciones

Se cerró el canal de transmisión y se notificó a todos los usuarios suscritos al mismo.

4 Requerimientos No Funcionales

En este capítulo se presentan los requerimientos no funcionales que deberá cumplir el sistema a desarrollar. Estos surgen a partir del tipo de aplicación y entorno en donde esta se ejecuta, haciendo un fuerte hincapié en la usabilidad, movilidad y uso eficiente de los recursos disponibles.

4.1 Uso de Simbología Adecuada

En el contexto de una aplicación geográfica que nuclea varias funcionalidades, es de esperar que se utilice simbología que ayude al usuario a identificar qué tipos de objetos móviles está siguiendo, así como sus diferentes estados (a través de diferentes formas o colores, por ejemplo).

4.2 Baja Latencia en la Transmisión de Datos

La latencia se define como la cantidad de tiempo de espera que se experimenta en un sistema [2]. Es de esperarse que el sistema a construir tenga una baja latencia en la transmisión de datos, ya que de otra forma no se alcanzaría la experiencia del usuario deseada en el seguimiento en tiempo real.

4.3 Interfaz Intuitiva y Amigable

En el mercado actual coexisten varias aplicaciones de un mismo tipo, o que tienen funcionalidades similares, por lo que se necesita destacar sobre las demás de alguna forma.

Una forma de destacar es tener una interfaz intuitiva. Esto quiere decir que la aplicación no requiere que el usuario tenga que leer un manual para poder utilizarla, sino que él mismo puede descubrir fácilmente cómo utilizar y explotar todas sus funcionalidades. De otra forma, el usuario podría llegar a descartar la aplicación y utilizar otra, buscando otra manera de suplir su necesidad.

Además se requiere que la aplicación sea amigable, de acuerdo a principios básicos establecidos por las compañías desarrolladoras de sistemas operativos para dispositivos móviles y de uso general de aplicaciones móviles, teniendo en cuenta las limitaciones que posee un dispositivo de este tipo (por ejemplo, el tamaño de la pantalla).

4.4 Portabilidad

La cantidad y variedad de dispositivos móviles del mercado, y los diferentes sistemas operativos que estos utilizan, hacen que al momento de desarrollar una aplicación sea esperable que la misma pueda portarse en la mayoría de las plataformas existentes, con el objetivo de llegar a la mayor cantidad de usuarios.

Como se mencionó anteriormente, cada sistema operativo utiliza un lenguaje de programación diferente, por lo que se buscará la forma de cubrir la mayor cantidad de sistemas operativos con el menor esfuerzo a la hora de reescribir código.

4.5 Uso Eficiente de la Batería

Una de las problemáticas que tienen las aplicaciones que hacen uso de sensores y redes de telefonía celular, es el excesivo consumo de batería que tienen estas funciones. Algunas cosas son inevitables, pero será deseable que la aplicación haga un uso eficiente de la batería, dejando asimismo configurar algunos parámetros que mejoren el uso la batería en pos de la calidad del servicio.

4.6 Seguridad

En el ámbito de una aplicación con información sensible, es de esperar que se establezcan protocolos de seguridad adecuados para que la información se intercambie en un entorno confiable. De no ser así, la información podría utilizarse de forma inadecuada, lo que incluso podría llegar a comprometer la integridad física del usuario.

5 Glosario

API	Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
AVL	Acrónimo de Automatic Vehicle Location (Rastreo Vehicular Automatizado en español), que refiere a los sistemas de localización remota en tiempo real, utilizando una combinación de dispositivos de localización (GPS, GLONASS) con dispositivos de transmisión inalámbrica de datos (módems).
¿Cuándo Llega?	Agrupación de funcionalidades de la aplicación relacionadas al seguimiento en tiempo real de ómnibus.
Escalabilidad	Propiedad de un sistema que indica su habilidad para reaccionar y adaptarse al crecimiento continuo de la carga de trabajo, o que está preparado para soportar volúmenes mayores de trabajo sin perder la calidad del servicio ofrecido.
FindMe!	Agrupación de funcionalidades de la aplicación relacionadas a la red social de usuarios y seguimiento en tiempo real de amigos.
Geolocalización	Posicionamiento con el que se define la localización de un objeto espacial (representado mediante punto, vector, área, volumen) en un sistema de coordenadas y datum determinado. Este proceso es utilizado frecuentemente en los SIG.
GPS	Sistema de satélites perteneciente al Departamento de Defensa de los Estados Unidos, permite triangular la ubicación de un objeto en la tierra, como una persona o un vehículo con una precisión en el ámbito de los metros.
Viaje	Ómnibus que está realizando el recorrido de una línea particular en un instante dado.

Bibliografía

[1] IEEE. (1998) IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. [Online]. http://standards.ieee.org/findstds/standard/830-1998.html

[2] Bellevue Linux. (2005, Setiembre) Latency Definition. [Online]. http://www.linfo.org/latency.html

Instituto de Computación - Facultad de Ingeniería - UdelaR

InTrack

Software Geográfico AVL para Ómnibus

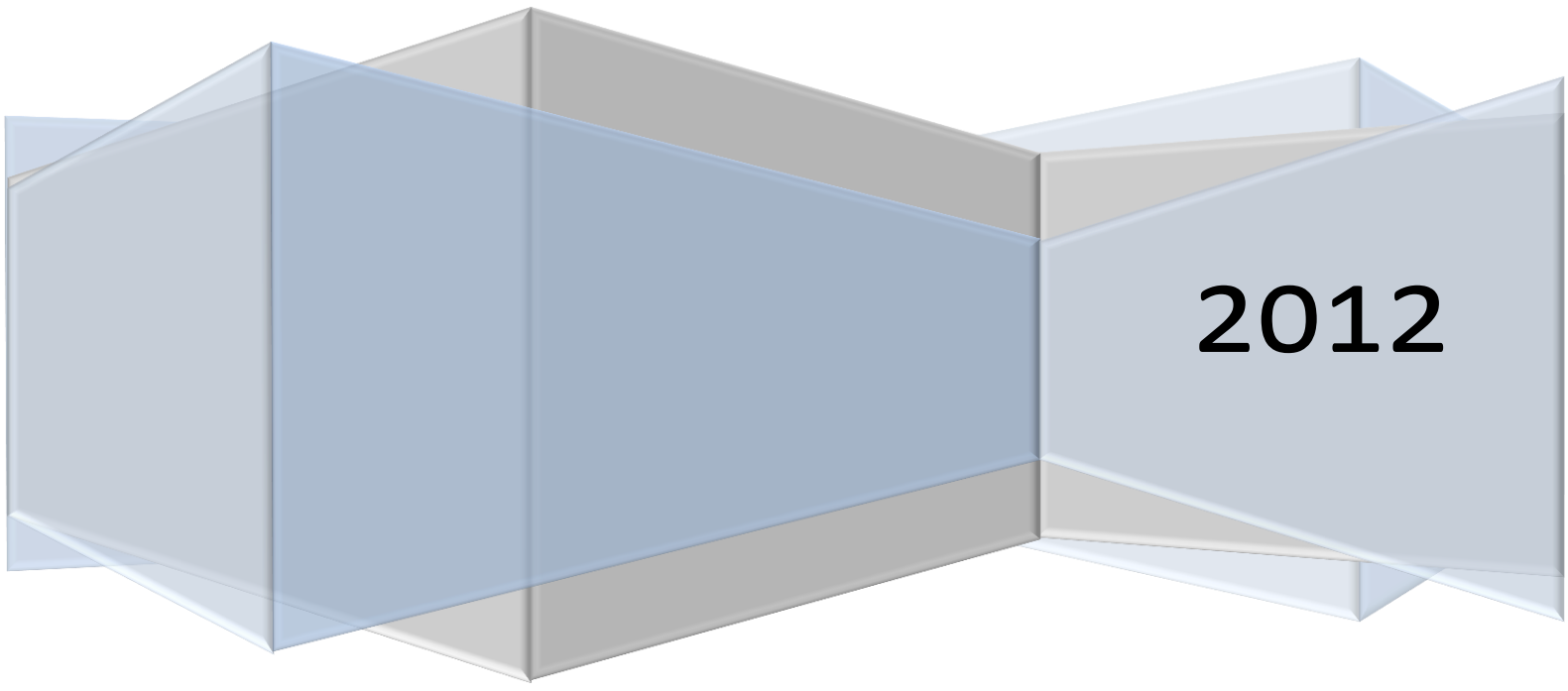
Integrantes

Bruno González, Máximo Mussini

Tutores

Pablo Rebufello, Sandro Moscatelli

ANEXO III – DISEÑO DEL SISTEMA



2012

Contenido

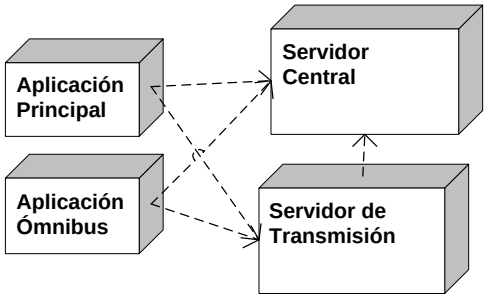
1 Diseño del Sistema

En este documento se presenta una descripción general de la arquitectura del sistema. Se detalla el diseño de las bases de datos utilizadas, así como la descripción de los servicios utilizados para su consulta.

Además se explican y justifican algunas decisiones de diseño, como el mecanismo utilizado para la transmisión de datos en tiempo real y el diseño de la interfaz gráfica.

1.1 Arquitectura general del sistema

A nivel general, el sistema está compuesto por cuatro subsistemas bien definidos, el **Servidor Central**, el **Servidor de Transmisión**, la **Aplicación Principal** para dispositivos móviles, y la **Aplicación para Ómnibus**.



Cada aplicación se comunica con ambos servidores utilizando las interfaces de servicios que estos proveen. Los servicios que provee el *Servidor Central* pueden ser accedidos mediante la invocación de servicios web, mientras que toda la comunicación con el *Servidor de Transmisión* se realiza mediante una conexión de datos persistente.

A continuación se presenta un breve resumen del propósito de cada subsistema.

1.1.1 Servidor Central

Está encargado de brindar servicios tanto a la *Aplicación Principal*, como a la *Aplicación de Ómnibus*, tales como gestión de usuarios, creación de puntos de encuentro, datos de ómnibus como líneas y paradas, etc. Esto se logra mediante una capa de servicios REST.

1.1.2 Servidor de Transmisión

Su objetivo es proveer un mecanismo de transmisión de datos en tiempo real entre dispositivos móviles. Utiliza un protocolo de comunicación liviano y de baja latencia para mejorar el tiempo de respuesta de la aplicación.

Para realizar controles de autorización y acceso, se comunica con el *Servidor Central* mediante una interfaz de servicios web que este ofrece.

1.1.3 Aplicación Principal

Es una aplicación para dispositivos móviles, y es utilizada por el usuario final para acceder a las funcionalidades del sistema a través de diferentes tipos de dispositivos móviles.

La aplicación engloba las funcionalidades de *¿Cuándo llega?* y *FindMe!*, y hace uso de la conexión de datos del dispositivo, así como del GPS del mismo si está presente.

Una de sus principales tareas es enviar y recibir datos en tiempo real a través de una conexión persistente con el *Servidor de Transmisión*. El resto de las interacciones, menos críticas en cuanto a tiempo de respuesta, se realizan mediante servicios web provistos por el *Servidor Central*.

1.1.4 Aplicación para Ómnibus

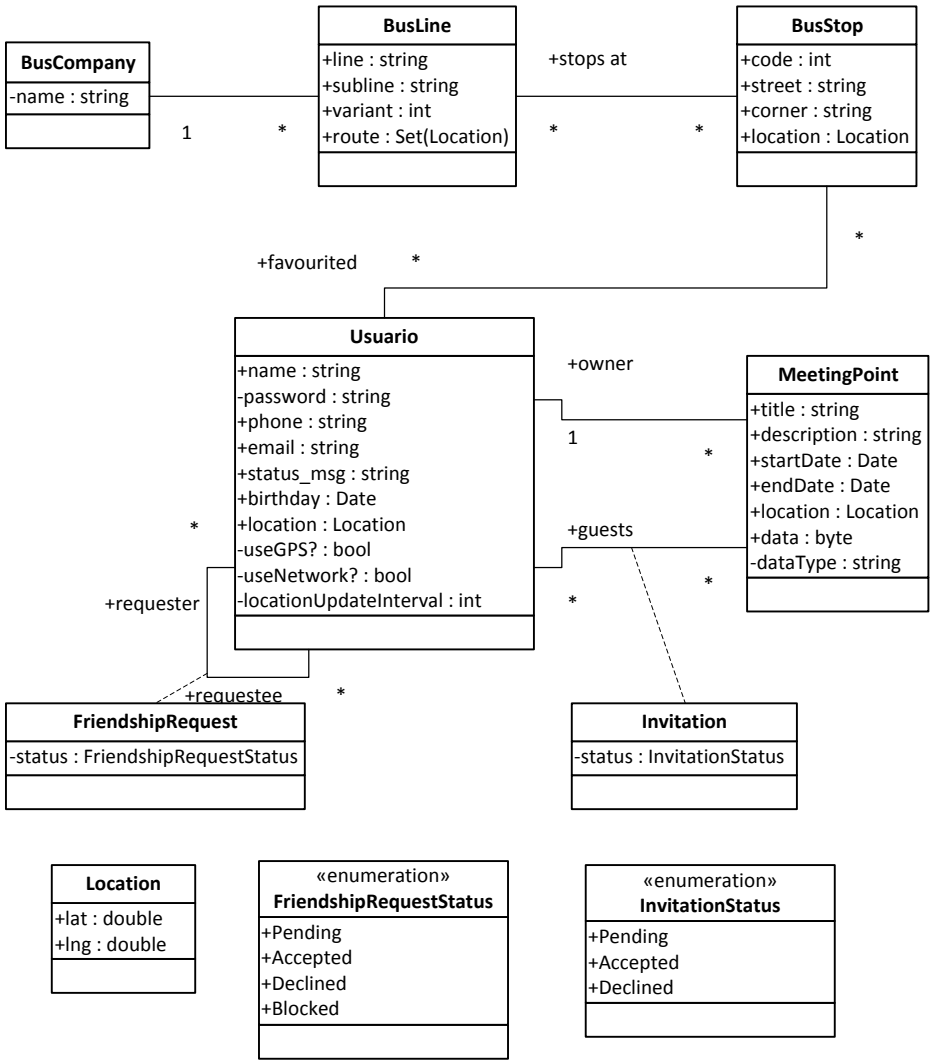
También una aplicación para dispositivos móviles, hace uso intensivo del GPS y la conexión de datos del dispositivo, el cual está a bordo de un ómnibus, de forma de enviar información geográfica en tiempo real al *Servidor de Transmisión*. Presenta una interfaz de usuario muy simple que permite seleccionar la línea de ómnibus, visualizar el recorrido de la línea y la posición actual del ómnibus, y empezar o detener la transmisión durante el recorrido. Los datos de ómnibus son obtenidos a través del *Servidor Central.*

Para poder realizar pruebas en el ámbito del proyecto, se agrega a la aplicación un componente de simulación que permite iniciar viajes de ómnibus para cualquier línea, lo cual permite probar el seguimiento de ómnibus de forma efectiva sin necesidad de tener el dispositivo presente en un ómnibus.

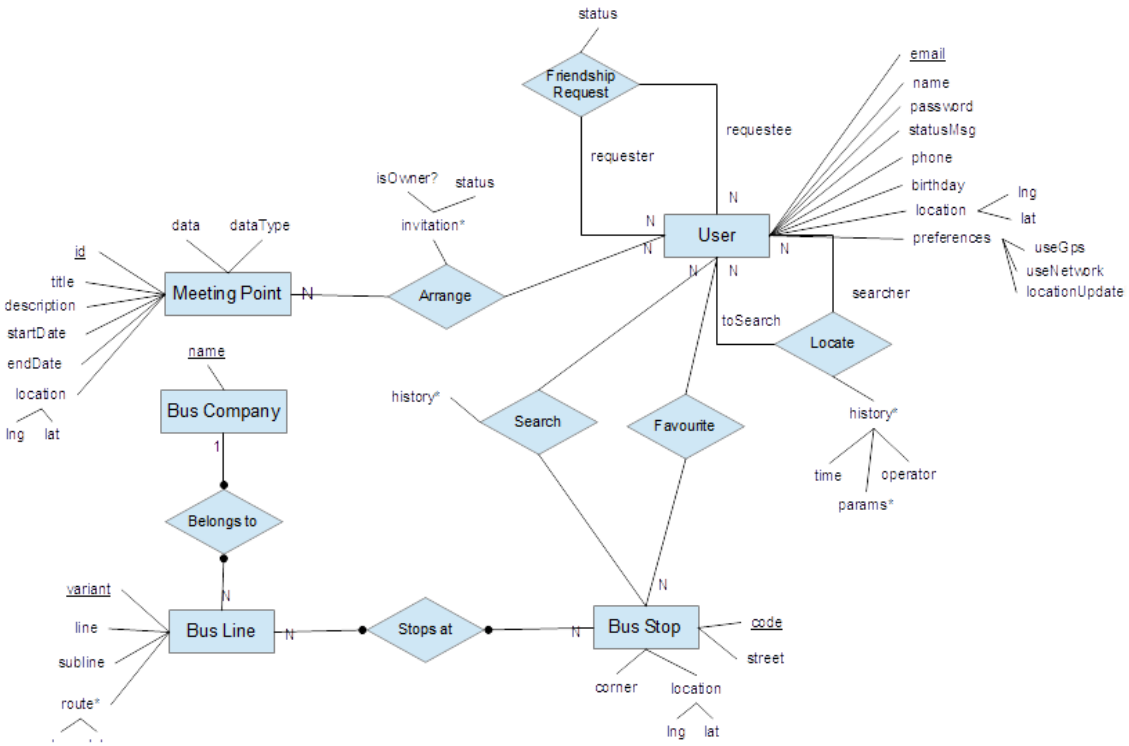
1.1.5 Servidor de Mapas

Debido al carácter geográfico del sistema es necesario contar con servicios geográficos de mapas, geo-codificación, y otros. El *Servidor de Mapas* se encarga de brindar estos servicios a través de una interfaz de servicios REST.

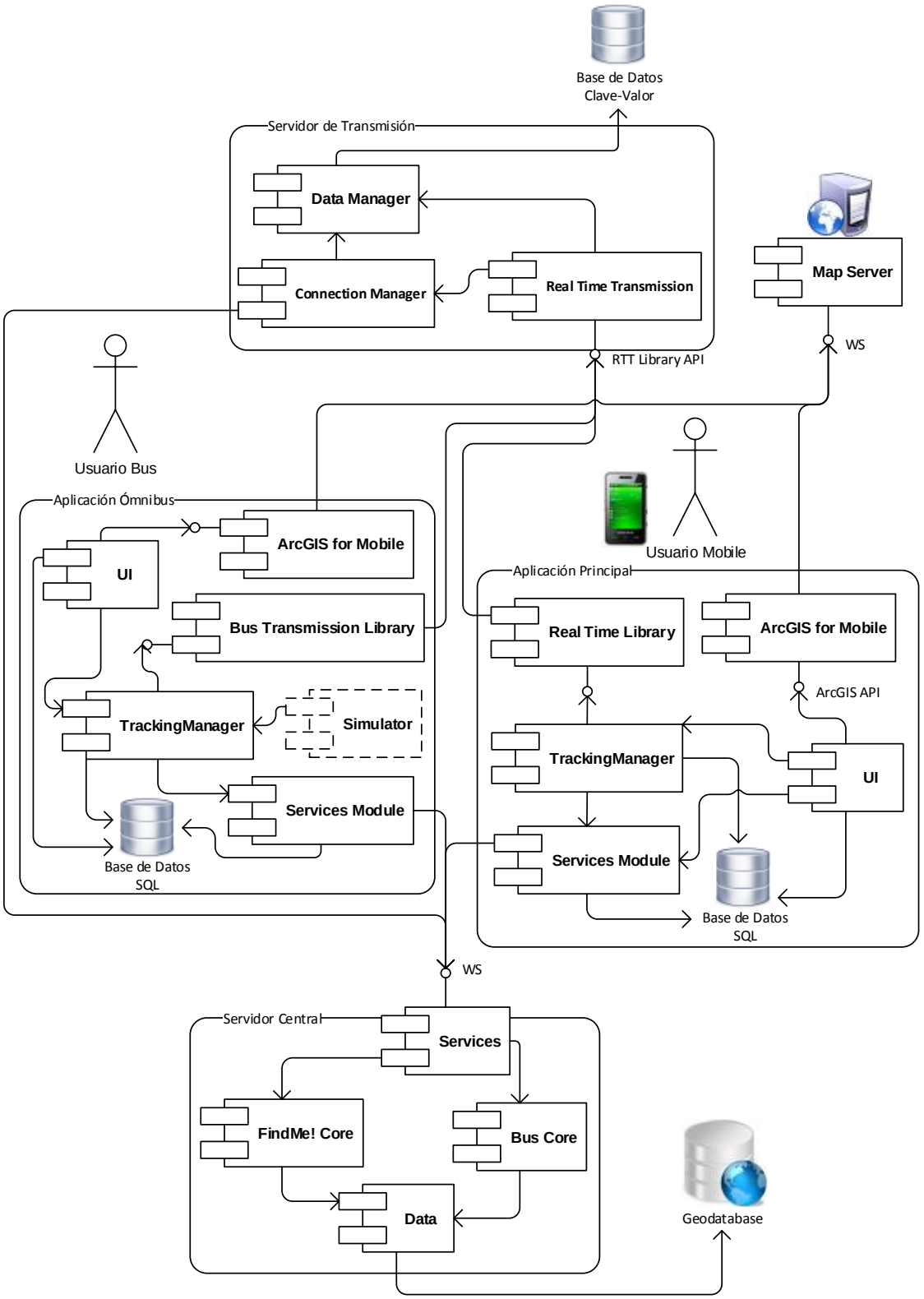
1.2 Modelo de Dominio del Sistema



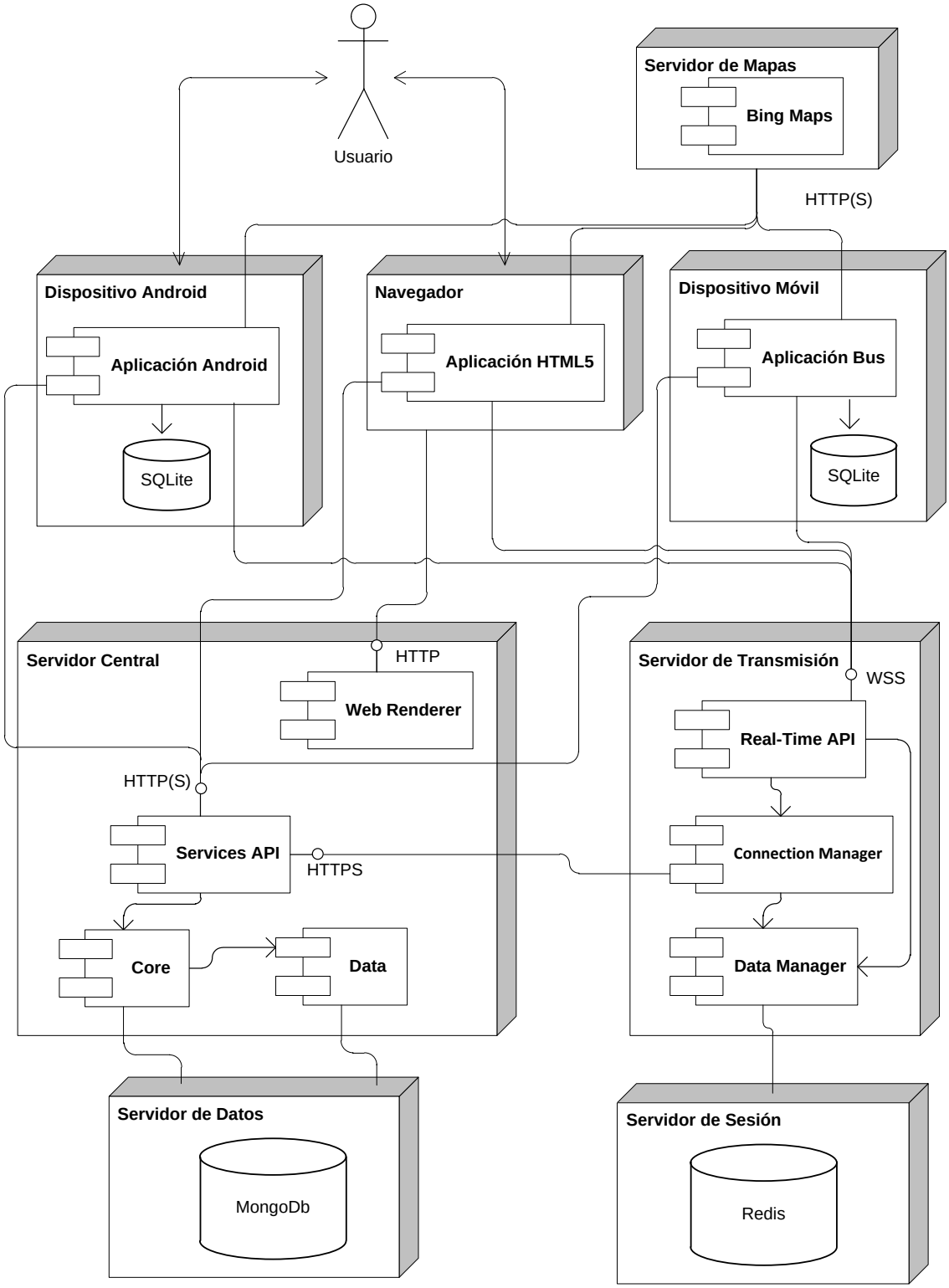
1.3 Modelo Entidad-Relación del Sistema



1.4 Diagrama de Arquitectura del Sistema



1.5 Diagrama de Distribución del Sistema



2 Servidor Central

Su función principal es ser el proveedor de los datos utilizados por la *Aplicación* Principal, y por la *Aplicación para* Ómnibus, así como proveer servicios de autenticación y de lógica de negocio para la *Aplicación Principal*. A su vez provee algunos servicios web al *Servidor de Transmisión* para poder realizar controles de autorización de usuarios.

2.1 Componentes de Software

El *Servidor Central* presenta una arquitectura de software de tres capas: lógica de negocio, persistencia, y servicios. A su vez, la capa de negocio se divide en dos componentes, respectivos a los requerimientos de *¿Cuándo llega?* y *FindMe.* A continuación se presentan los componentes que conforman el servidor.

FindMe! Core

Es el componente que contiene la lógica de negocio de la aplicación *FindMe!*, como autenticación, gestión de usuarios, manejo del flujo de solicitudes de amistad, puntos de encuentro, y demás funcionalidades de la aplicación.

Bus Core

Contiene servicios de datos de ómnibus utilizados por *¿Cuándo llega?*, así como la *Aplicación de Ómnibus*. No realiza ningún tipo de procesamiento.

Data

Módulo de acceso a datos, su objetivo es realizar el mapeo del esquema de datos a objetos de forma conveniente para cada entidad persistente del sistema, como paradas, líneas de ómnibus, usuarios, puntos de encuentro, etc.

Services

Expone las funcionalidades de los otros módulos mediante una API de servicios. Esta API puede ser consumida de forma remota a través de servicios REST.

2.2 Base de Datos basada en Documentos

La base de datos del servidor almacena la información que utilizan las aplicaciones móviles. Se utilizará un esquema de datos basado en documentos para almacenar las compañías, líneas, y paradas de ómnibus. También se persisten los usuarios, amigos, solicitudes de amistad, y demás datos de *FindMe!*

BusCompanies

Colección con las compañías de ómnibus que forman parte del STM.

Campo	Tipo	Descripción
id	ObjectId	Identificador único de una compañía de ómnibus el sistema
name	String	Nombre de la compañía de ómnibus

BusStops

Colección que almacena las paradas de ómnibus de la ciudad.

Campo	Tipo	Descripción
id	ObjectId	Identificador único de la parada en el sistema
code	Integer	Identificador de la línea en el STM
street	String	Nombre de la calle de la parada
corner	String	Nombre de la esquina más cercana a la parada
location	POINT	Ubicación geográfica de la parada de ómnibus
line_ids	Array(ObjectId)	Identificadores de las líneas que paran en la parada

BusLines

Esta colección almacena un documento por línea específica de ómnibus (ej: 117 PUNTA CARRETAS), junto con sus datos asociados.

Campo	Tipo	Descripción
id	ObjectId	Identificador único de la línea de ómnibus en el sistema
line	String	Nombre de línea
subline	String	Nombre de sublínea
variant	Integer	Código de la variante del recorrido
route	POLYLINE	Recorrido de la línea
stop_ids	Array(ObjectId)	Identificadores de las paradas del recorrido de la línea
company_id	ObjectId	Identificador de la compañía a la que pertenece la línea

FriendshipRequests

Esta colección almacena las solicitudes de amistad enviadas por los usuarios.

Campo	Tipo	Descripción
id	ObjectId	Identificador único de la solicitud de amistad
status	String	Estado de la solicitud de amistad
requester_id	ObjectId	Identificador del usuario que realizó la solicitud
requestee_id	ObjectId	Identificador del usuario que recibe la solicitud

MeetingPoints

Colección con las citas de punto de encuentro entre usuarios.

Campo	Tipo	Descripción
id	ObjectId	Identificador único de la cita o punto de encuentro
owner_id	ObjectId	Identificador del usuario creador del punto de encuentro

guest_ids	Array(ObjectId)	Identificadores de los usuarios que asistirán al encuentro
title	String	Nombre de la cita o punto de encuentro
description	String	Descripción del punto de encuentro
location	POINT	Posición geográfica del encuentro
start_time	Time	Fecha y hora de comienzo del encuentro
end_time	Time	Fecha y hora de fin del encuentro
data	Blob	Datos adicionales
data_type	String	Tipo de los datos adjuntos
invitations	Hash (ObjectId,String)	Invitaciones al encuentro almacenadas en pares *(identificador de usuario, estado de la invitación)*

Users

Esta colección almacena la información de los usuarios de *FindMe!*

Campo	Tipo	Descripción
id	ObjectId	Identificador único del usuario
name	String	Nombre público del usuario
email	String	Correo electrónico del usuario
password	EncryptedString	Contraseña del usuario
status_msg	String	Mensaje para mostrar
phone	String	Número celular del usuario
birthday	Date	Fecha de nacimiento del usuario
location	POINT	Última ubicación conocida del usuario
friend_ids	Array(ObjectId)	Identificadores de los amigos del usuario
friendship_requests_sent_ids	Array(ObjectId)	Identificadores de las solicitudes de amistad enviadas
friendship_requests_received_ids	Array(ObjectId)	Identificadores de las solicitudes de amistad recibidas
meeting_point_ids	Array(ObjectId)	Identificadores de los puntos de encuentro del usuario
favourite_stop_ids	Array(ObjectId)	Identificadores de las paradas de ómnibus favoritas del usuario
locate_friends_history	Array(UserHistory)	Historial de búsqueda geográfica de amigos
friends_search_history	Array(UserHistory)	Historial de búsqueda de nuevos amigos
bus_search_history	Array(UserHistory)	Historial de búsqueda de paradas de ómnibus

user_preferences	UserPreferences	Docum

UserHistory

Este sub-documento está embebido en el usuario, y representa una búsqueda realizada por el usuario que puede ser de carácter geográfico o de búsqueda simple. Se utiliza para facilitar un historial de búsqueda al usuario.

Campo	Tipo	Descripción
query	Hash	Búsqueda compleja, la clave determina el campo y el valor determina el patrón de la búsqueda a realizar
operator	String	Operador interno que determina si se realiza una búsqueda exhaustiva (OR), o específica (AND)

UserPreferences

Este sub-documento también está embebido en el usuario, y almacena las preferencias de usuario de *FindMe!* Las preferencias están principalmente relacionadas con el seguimiento de usuario y tienen como fin que el usuario pueda controlar el consumo de batería.

Campo	Tipo	Descripción
use_gps	Boolean	Determina si el usuario permite que la aplicación utilice el GPS para determinar su ubicación
use_network	Boolean	Determina si el usuario permite que la aplicación utilice información de la red para determinar su ubicación
location_update_interval	Integer	Intervalo mínimo entre envíos de la posición geográfica del usuario al servidor para seguimiento

2.3 Servicios Web

El Servidor Central expone una serie de servicios web que permiten acceder a los datos del mismo de forma remota, así como acceder a ciertas funcionalidades avanzadas como búsqueda geográfica.

2.3.1 Servicios de Ómnibus

Este conjunto de servicios es utilizado por la *Aplicación Principal* y la *Aplicación para Ómnibus*, y cuyo fin es obtener los datos de ómnibus necesarios, que incluyen compañías, líneas, y paradas de ómnibus.

URL	Método	Descripción	Parámetros
/companies	GET	Devuelve una lista con todas las compañías de ómnibus existentes en el sistema	-
/company/#	GET	Devuelve la información completa sobre una compañía específica	id
/lines	GET	Devuelve una lista con todas las líneas de ómnibus existentes en el sistema, sin los recorridos	-

/line/#	GET	Devuelve información de una línea específica, incluyendo el recorrido de la misma	id
/stops	GET	Devuelve una lista con todas las paradas del sistema	-
/stop/#/lines	GET	Devuelve una lista con todas las líneas y sublíneas cuyo recorrido incluye una parada específica	id
/stops/find	GET	Devuelve una lista con la información básica de las paradas más cercanas a la posición geográfica	lat, lng
/bus/all	GET	Devuelve toda la información de ómnibus del sistema sin redundancia, siempre y cuando los datos hayan cambiado luego del timestamp	timestamp

2.3.2 Servicios de *FindMe!*

Este conjunto de servicios es utilizado exclusivamente por la *Aplicación Principal* y permite acceder a toda la información del usuario, como historial de búsqueda, puntos de encuentro, etc. A su vez permiten manejar algunos flujos como el de solicitud de amistad, y las invitaciones a puntos de encuentro.

A diferencia de los servicios de ómnibus, es necesario contar con un *token* de autenticación para poder acceder a los servicios. Cada *token* de autenticación está asociado a un usuario particular, lo cual permite manejar restricciones de acceso, permitiendo acceder únicamente a la información que pertenece al usuario.

Servicios de Usuario

URL	Método	Descripción	Parámetros
/register	POST	En caso de ser válidos los datos se crea un nuevo usuario, devolviendo los datos del mismo	name, password, email, phone, birthday
/login	POST	En caso de ser válida la contraseña, se inicia la sesión y retorna un token que habilita el acceso al resto de los servicios de FindMe!	username, password
/logout	POST	Invalida el token de autenticación	-
/me/all	GET	Devuelve toda la información del usuario, como amigos, puntos de encuentro, etc, que hayan cambiado luego del timestamp. Se utiliza para optimizar la sincronización de datos	timestamp
/me/update	POST	Actualiza la información del usuario	attributes
/me/stops	GET	Devuelve una lista con las paradas de ómnibus favoritas del usuario, para brindar un acceso rápido a las paradas utilizadas frecuentemente	-
/me/stop/#	PUT	Agrega la parada a los favoritos del usuario	stop_id
/me/stop/#	DELETE	Elimina la parada de los favoritos del usuario	stop_id

Servicios de Amigos

URL	Método	Descripción	Parámetros

/me/friends	GET	Devuelve una lista con los amigos del usuario, incluyendo la información básica de cada uno	-
/me/friend/#	GET	Devuelve la información completa de un amigo del usuario	-
/me/friend/#	DELETE	Elimina a un amigo de la lista de amigos del usuario	-
/me/friends/requests	GET	Obtiene las solicitudes de amistad pendientes del usuario	-
/me/friends/requests	POST	Crea una solicitud de amistad	friend_id
/me/friends/requests/accept	PUT	Acepta una solicitud de amistad, convirtiendo a ambos usuarios en amigos	request_id
/me/friends/requests/decline	PUT	Marca una solicitud de amistad como rechazada	request_id
/me/friends/requests/block	PUT	Marca una solicitud de amistad como rechazada, y no permite que el usuario que envió la solicitud pueda enviar nuevas solicitudes	request_id

Servicios de Puntos de Encuentro

URL	Método	Descripción	Parámetros
/me/meetings	GET	Devuelve una lista con las invitaciones de encuentro pendientes o aceptadas del usuario	-
/me/meetings	POST	Crea un evento de punto de encuentro, e invita a los usuarios correspondientes	title, description, location, guest_ids, start_date, end_date, data, data_type
/me/meetings/#/accept	PUT	Acepta una invitación de punto de encuentro	meeting_point_id
/me/meetings/#/decline	PUT	Rechaza una invitación de punto de encuentro	meeting_point_id
/me/meeting/#	DELETE	Elimina un evento de punto de encuentro creado por el usuario	meeting_point_id

Servicios de Búsqueda

URL	Método	Descripción	Parámetros
/me/friends/locate	GET	Realiza una búsqueda de amigos del usuario a partir de su ubicación geográfica (búsqueda por cercanía), y algunos parámetros opcionales	name, email, phone, location_radius
/me/friends/search	GET	Realiza una búsqueda simple de usuarios que no son amigos, con el fin de facilitar el agregar nuevos amigos	name, email, pone
/me/friends/search/fof	GET	Realiza una búsqueda de contactos de	-

		segundo grado (amigos de amigos) del usuario, con el fin de agregar amigos	
/me/friends/locate/history	GET	Permite consultar el historial de búsqueda geográfica de amigos existentes	-
/me/friends/search/history	GET	Permite consultar el historial de búsqueda de nuevos amigos	-
/me/bus/search/history	GET	Permite consultar el historial de búsqueda de ómnibus	-

2.3.3 Servicios para el *Servidor de Transmisión*

El Servidor de Transmisión tiene requerimientos de alto rendimiento, por lo que utiliza servicios que están optimizados en cuanto al volumen de datos transmitidos y el tiempo de respuesta, y cumplen funciones muy específicas.

URL	Método	Descripción	Parámetros
/me/friends/ids	GET	Devuelve una lista con los identificadores de los amigos del usuario, que se utiliza para notificar al usuario del estado de conexión de sus amigos	-
/me/friend/#/followable	GET	Controla que el usuario pueda seguir a cierto usuario, para lo cual debe ser su amigo y no estar bloqueado	friend_id

3 Servidor de Transmisión

Su objetivo es proveer un mecanismo de transmisión de datos en tiempo real entre dispositivos móviles. Implementa una API que sigue el patrón publicar/suscribir, lo cual sumado a la utilización de un protocolo de comunicación liviano y de baja latencia resulta en un mecanismo flexible y eficiente para enviar datos geográficos a uno o más dispositivos móviles.

Cada usuario del sistema se corresponde con un canal de datos geográficos, al cual sus amigos se pueden suscribir luego de que el usuario accede al seguimiento. A su vez, se manejan canales dinámicos para los ómnibus que están activos, en base a la línea del ómnibus y el momento en que comenzó a realizar el recorrido.

Para realizar controles de autorización y acceso, el *Servidor de Transmisión* se comunica con el *Servidor Central* mediante la interfaz de servicios web que este ofrece.

3.1 Componentes de Software

El servidor está compuesto por tres componentes de software que giran en torno a la información de sesión de los clientes conectados; el primero se encarga de distribuir las actualizaciones geográficas, el segundo se encarga de mantener la información de sesión actualizada, y por último, el tercer componente se encarga de persistir esta información para mejorar la tolerancia a fallos.

3.1.1.1 Real Time Transmission

Este componente se encarga de manejar las actualizaciones de información geográfica. Al recibir una actualización geográfica desde un móvil se encarga de transmitir la información a todos los clientes que se hayan suscrito a la misma.

3.1.1.2 Connection Manager

Gestiona las conexiones de clientes, y responde a las peticiones de suscripción implementando controles de autorización en el seguimiento de usuarios. Su función principal es la de mantener la información de sesión para cada usuario conectado. También se encarga de multiplexar los pedidos de suscripción, encontrando el canal de transmisión adecuado.

3.1.1.3 Data Manager

Encapsula la persistencia de la información relacionada a las sesiones y canales disponibles en el servidor, y expone estos datos mediante una interfaz sencilla a los otros componentes del servidor.

3.2 Base de Datos Clave-Valor

La base de datos almacena la información necesaria para la transmisión de datos, esto es, usuarios conectados, recorridos de ómnibus activos, usuarios que están realizando seguimiento, y *tokens* de autorización.

Para lograr un mayor rendimiento se utiliza una base de datos no relacional del tipo clave-valor.

Clave	Tipo	Descripción
connected_users	Hash(String, JSON)	Indexado por identificador de usuario, contiene información básica del usuario como nombre y token de autenticación. Cuando un usuario se desconecta se elimina la clave.
/user/#/followers	Array(String)	Identificadores de los usuarios que están realizando seguimiento a un cierto usuario. La clave está compuesta por el id de dicho usuario. Se utiliza para notificar a los usuarios que están realizando seguimiento de una nueva actualización geográfica del usuario seguido.
/user/#/friends	Array(String)	Identificadores de los usuarios que son amigos de cierto usuario. La clave está compuesta por el id de dicho usuario. Sirve para notificar a los usuarios frente a la conexión o desconexión de un amigo.
active_rides	Hash(String, JSON)	Indexado por identificador de ómnibus, contiene información básica del ómnibus como última ubicación recibida, número de línea, nombre de compañía, etc. La entrada es válida hasta que se da por

		finalizado el recorrido.
/bus/#/followers	Array(String)	Identificadores de los usuarios que están realizando seguimiento a cierto ómnibus. La clave está compuesta por el id de dicho ómnibus. Se utiliza para notificar a los usuarios que están realizando seguimiento de un ómnibus de una actualización geográfica del ómnibus seguido.
line_rides	Hash (String,Array(String))	Indexado por identificador de una línea de ómnibus, contiene los identificadores de los ómnibus activos para esa línea. Es una optimización que permite consultar rápidamente los ómnibus activos de una línea.

3.3 Protocolo de Transmisión

En esta sección se describe a grandes rasgos el protocolo de alto nivel que utilizan la *Aplicación Principal* y el *Servidor de Transmisión* para comunicarse, así como también el que utiliza la *Aplicación para Ómnibus*.

La comunicación se realiza de forma completamente asíncrona, lo cual permite aprovechar mejor los recursos del servidor y atender múltiples pedidos sin necesidad de esperar por acceso a datos o respuestas del servidor central.

3.3.1 API del Servidor para el seguimiento de Amigos

A continuación se indican los mensajes que procesa el *Servidor de Transmisión* para permitir el seguimiento de amigos. Para poder interactuar correctamente, el cliente debe manejar los mensajes que el servidor devuelve.

Mensaje	Parámetros	Descripción
login	username, password	Necesario para establecer la sesión del usuario con el servidor de transmisión. Notifica a todos los amigos del usuario enviando el mensaje *friendConnected*. Valida las credenciales con el *Servidor Central*.
followFriend	friendId	Envía una solicitud de seguimiento (*followRequest*) al amigo del usuario, siempre y cuando el usuario esté habilitado a seguirlo, y el amigo se encuentre conectado al servidor.
acceptFollow	friendId	Inicia el seguimiento del amigo que envió la solicitud al usuario que la confirmó, agregando al amigo a la lista de seguidores del usuario. Se notifica al amigo que su solicitud de seguimiento fue aprobada mediante el mensaje *followAccepted.*
rejectFollow	friendId	Rechaza una solicitud de seguimiento.

		Se notifica al amigo del usuario que realizó una solicitud de seguimiento de que su solicitud fue rechazada, mediante el mensaje *followRejected*.
cancelFollow	friendId	Se cancela el seguimiento, quitando al amigo de la lista de seguidores del usuario. Se notifica al amigo que ya no recibirá más actualizaciones geográficas mediante el mensaje *requesteeCancelledFollow*
stopFollowing	friendId	Se cancela el seguimiento, quitando al usuario de la lista de seguidores del amigo. Se notifica al amigo que el usuario ya no está subscripto a sus actualizaciones geográficas mediante el mensaje *requesterCancelledFollow*
sendLocation	lat, lng	Se actualiza la posición geográfica del usuario en el caché del servidor, y se realiza un broadcast con la actualización geográfica a todos los amigos que estén en la lista de seguidores del usuario, enviando el mensaje friend*LocationUpdate*.

3.3.2 API del Servidor para realizar seguimiento de Ómnibus

A continuación se indican los mensajes que procesa el *Servidor de Transmisión* para permitir el seguimiento de ómnibus.

Productor

La API para publicar datos de ómnibus en tiempo real. Cuando el ómnibus inicia el recorrido, se crea un nuevo viaje o *ride*, que se mantiene activo hasta que el ómnibus termina su recorrido y el viaje es destruido. Esto permite que ante la pérdida de conexión se pueda retomar el envío de actualizaciones geográficas.

Mensaje	Parámetros	Descripción
createRide	lineId	Crea un nuevo viaje o *ride* en el servidor. Un viaje representa a un ómnibus realizando un recorrido para una línea particular. Se notifica el éxito de la operación con *rideCreated*.
destroyRide	rideId	Destruye un viaje, por ejemplo, cuando el ómnibus llega a destino. Se notifica a los seguidores del ómnibus mediante el mensaje *busDestroyed*, indicando que el viaje ya no enviará más actualizaciones geográficas.
sendBusLocation	lat, lng	Se actualiza la posición geográfica del ómnibus en el caché del servidor, y se realiza un broadcast con la actualización geográfica a todos los usuarios que estén en la lista de seguidores del ómnibus, enviando el mensaje *busLocationUpdate*.

Consumidor

La API para manejar el seguimiento de ómnibus en tiempo real. Esta API es utilizada por el *Algoritmo de Ómnibus más Cercanos*, para obtener los viajes activos de una línea y luego calcular los dos ómnibus más cercanos al usuario, y comenzar el seguimiento de los viajes correspondientes.

Mensaje	Parámetros	Descripción
fetchBusRides	lineId	Envía una lista con los viajes activos para una línea de ómnibus particular. La respuesta es asíncrona y se envía mediante el mensaje *busRidesForLine*.
startFollowingBus	rideId	Se inicia el seguimiento de un ómnibus, agregando al usuario a la lista de seguidores del viaje.
stopFollowingBus	rideId	Se cancela el seguimiento del ómnibus, quitando al usuario de la lista de seguidores.

3.3.3 Contrato para el Seguimiento de Usuarios

El *Servidor de Transmisión* envía los siguientes mensajes a la aplicación cliente, de forma de poder establecer una comunicación bidireccional. La *Aplicación Principal* cumple con los requisitos de cada mensaje.

Mensaje	Parámetros	Descripción
friendConnected	friendId	Indica que el amigo está conectado al *Servidor de Transmisión.*
friendDisconnected	friendId	Indica que el amigo se ha desconectado del *Servidor de Transmisión.*
followRequest	friendId	Indica que se ha recibido una solicitud de seguimiento. Se debe responder al servidor de forma de permitir o denegar el seguimiento, enviando los mensajes *acceptFollow* y *rejectFollow* respectivamente.
followAccepted	friendId	Indica que un amigo ha aprobado la solicitud de seguimiento realizada por el usuario, y que debería prepararse para comenzar a recibir las actualizaciones geográficas.
followRejected	friendId	Indica que un amigo ha rechazado la solicitud de seguimiento realizada por el usuario.
requestee CancelledFollow	friendId	Indica que un amigo del usuario ha finalizado el seguimiento, por lo cual no recibirá más actualizaciones geográficas de su parte.
requester CancelledFollow	friendId	Indica que un amigo del usuario ha dejado de seguirlo.

		Si no hay más amigos siguiendo al usuario puede dejar de enviar actualizaciones geográficas al servidor.
friendLocationUpdate	lat, lng, friendId	Actualización geográfica de uno de los amigos del usuario.

3.3.4 Contrato para el Seguimiento de Ómnibus

La aplicación cliente debe procesar los siguientes mensajes para poder desarrollar el seguimiento de ómnibus.

Mensaje	Parámetros	Descripción
busRidesForLine	Set(Rides)	Lista de viajes activos para una línea particular. Esta lista contiene la última ubicación conocida de cada ómnibus, lo cual le permite filtrar los ómnibus relevantes según la ubicación del usuario.
busLocationUpdate	lat, lng, rideId	Actualización geográfica de uno de los ómnibus que está siguiendo el usuario.
busDestroyed	*rideId*	Indica que el ómnibus ha finalizado el recorrido y ya no enviará más actualizaciones geográficas.

3.3.5 Contrato de la Aplicación para Ómnibus

La aplicación para ómnibus sólo necesita procesar el mensaje de validación del recorrido, que es similar al inicio de sesión del usuario.

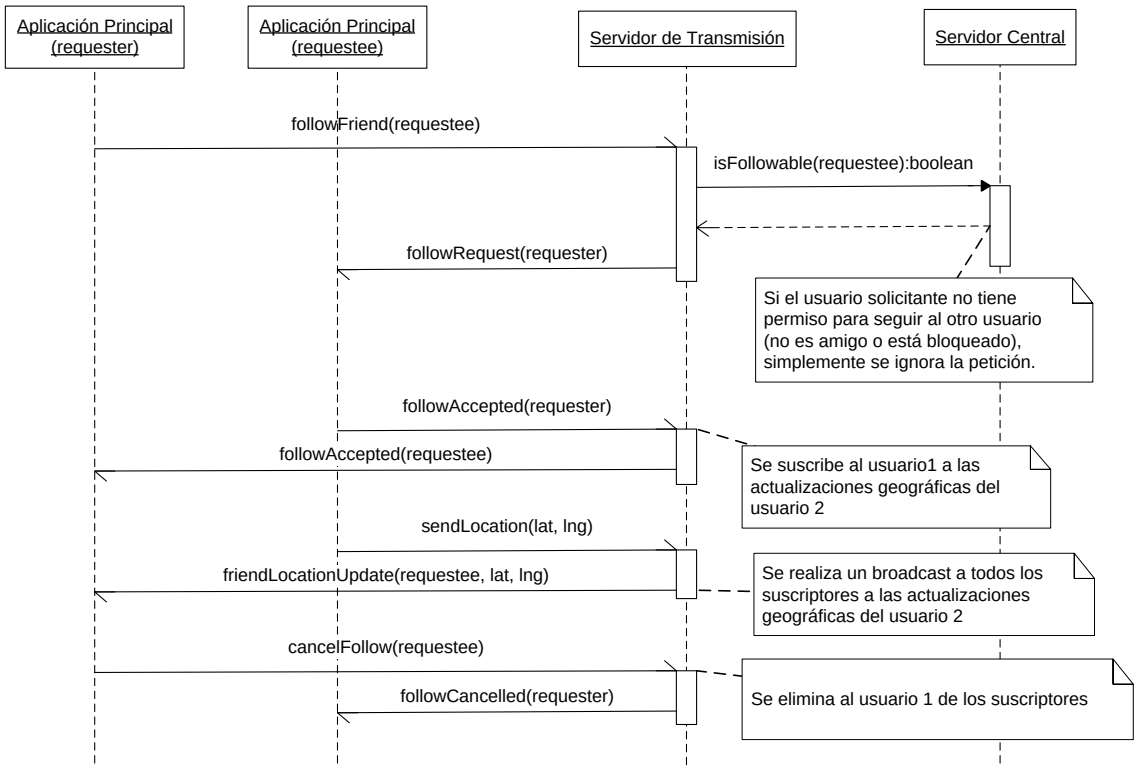
Mensaje	Parámetros	Descripción
rideCreated	Ride	Indica que se creó un nuevo viaje con éxito, junto con toda la información disponible del mismo, en particular el identificador del viaje.

3.4 Diagramas de Secuencia

La interacción entre las aplicaciones y los servidores puede resultar difícil de apreciar de la forma en que se plantea en la sección anterior, por lo que se facilitan diagramas de secuencia para los flujos de seguimiento de forma de poder visualizarlo mejor.

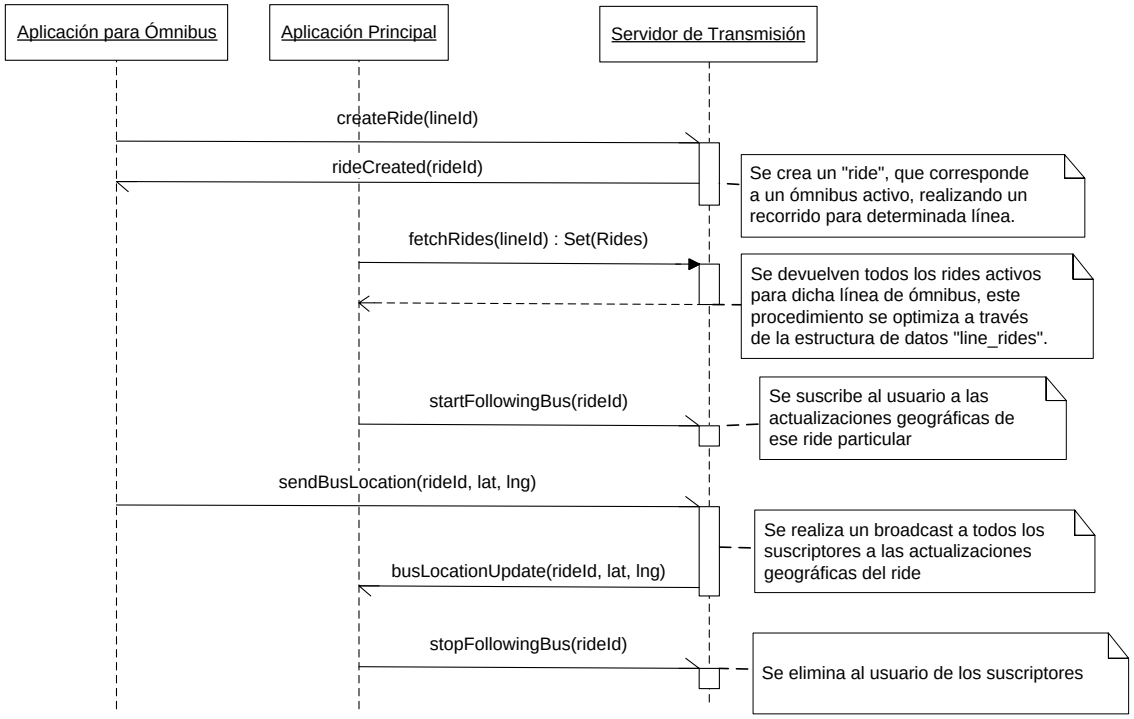
3.4.1 Seguimiento de Usuario

Este diagrama ejemplifica un flujo típico de seguimiento de usuario entre dos amigos.



3.4.2 Seguimiento de Ómnibus

Este diagrama de secuencia muestra un flujo típico de seguimiento de ómnibus, donde se muestra el *Productor* (Aplicación para Ómnibus) y el *Consumidor* (Aplicación Principal).



4 Aplicación Principal

Esta aplicación para dispositivos móviles integra las funcionalidades de *FindMe!* y *Cuándo llega?* de forma sencilla e intuitiva, y puede ser utilizada por el usuario final a través de diferentes tipos de dispositivos móviles.

Sus funcionalidades principales son el seguimiento de ómnibus y usuarios en tiempo real, para lo cual hace uso intensivo de la conexión de datos del dispositivo, así como del GPS del mismo si está presente.

Para poder realizar el seguimiento es necesario enviar y recibir datos en tiempo real, lo cual se realiza a través de una conexión persistente con el *Servidor de Transmisión*. El resto de las interacciones, menos críticas en cuanto a tiempo de respuesta, se realizan mediante servicios web provistos por el *Servidor Central*.

El diseño toma en cuenta restricciones típicas de los dispositivos móviles tales como la autonomía, y velocidad de la conexión de datos.

4.1 Componentes de Software

La aplicación móvil es compleja ya que además de involucrar la transmisión de datos geográficos al *Servidor de Transmisión* y la visualización de posiciones y recorridos de ómnibus en un mapa, necesita sincronizar la información obtenida desde el *Servidor Central*, y está compuesta por dos juegos de requerimientos que bien podrían conformar aplicaciones distintas. A continuación se describe la funcionalidad de cada uno de sus componentes de software.

User Interface

Es la interfaz gráfica de la aplicación móvil, y está diseñada para agilizar los casos de uso más frecuentes del usuario, en particular el seguimiento de ómnibus y el seguimiento de amigos. Si bien son funcionalidades parecidas, las situaciones en las que el usuario interactúa con dichos casos de uso pueden ser muy dispares, por lo que es importante que las funcionalidades no resulten invasivas, logrando un balance entre accesibilidad y facilidad de uso.

Debido a que el principal atractivo de la aplicación es la información en tiempo real, es importante que la información exhibida esté actualizada en todo momento. Esto combinado a que la mayoría de las interacciones con el *Servidor de Transmisión* son asíncronas, hace que el componente de interfaz de usuario tenga una complejidad elevada.

Services Module

Este componente se encarga de comunicarse con el *Servidor Central* a través de los servicios REST que este expone. También está encargado de persistir localmente la información obtenida desde el servidor, como paradas y líneas de ómnibus, en una base de datos SQL local, de forma de poder optimizar el acceso a red y mejorar el tiempo de respuesta de la aplicación.

A su vez hay datos que necesitan sincronización bidireccional, como es el caso de los amigos y las solicitudes de amistad de *FindMe!,* por lo que también debe encargarse de mantener la información actualizada, y enviar los cambios al servidor.

Por último, tiene como responsabilidad notificar al componente de interfaz de usuario cuando la información ha cambiado, de forma de actualizar la interfaz.

Tracking Manager

Este módulo está encargado de llevar un registro de los seguimientos en los que está participando el usuario, y estados de conexión de los amigos del usuario. Para esto utiliza las funcionalidades brindadas por la *Real Time Transmission Library*. Una de sus responsabilidades principales es la de enviar actualizaciones geográficas, para lo cual controla el intervalo de tiempo entre las actualizaciones y consulta la información de GPS y red para obtener la ubicación del usuario.

También tiene como responsabilidad notificar al componente de interfaz de usuario cuando se recibe nueva información. Debido a que las actualizaciones pueden llegar a ser muy frecuentes y toda la comunicación con el *Servidor de Transmisión* es asíncrona, el mecanismo de comunicación con el componente *User Interface* se diseñó para liviano y eficiente.

Por último, implementa el *Algoritmo de Ómnibus más Cercanos,* que calcula los ómnibus más cercanos a una parada de ómnibus particular que se encuentran realizando el recorrido activamente y aún no han pasado la parada.

Real Time Transmission Library

Este componente implementa el lado cliente de la comunicación con el *Servidor de Transmisión*, en particular cumple con el *Contrato para Seguimiento de Ómnibus* y con el *Contrato para Seguimiento de Usuarios*, y utiliza las API de seguimiento.

ArcGIS for Mobile

Para poder mostrar la información geográfica al usuario y permitir que interactúe con la misma, la aplicación hace uso extensivo de mapas. Debido a que una de las restricciones del proyecto es utilizar tecnologías de ESRI, para ello se utiliza la biblioteca ArcGIS, que se encuentra disponible para la gran mayoría de las plataformas móviles, y permite mostrar mapas y realizar operaciones geográficas sencillas.

4.2 Base de Datos SQL

Para reducir la utilización de red y optimizar los tiempos de respuesta, la gran mayoría de los datos utilizados en la aplicación se persisten de forma local, sincronizando mediante servicios web con el *Servidor Central* para mantener la información actualizada.

Los servicios de sincronización solo envían la información que ha cambiado desde la última sincronización, que está determinada por un *timestamp* que se envía en la llamada a los servicios.

Se utiliza una base de datos relacional principalmente por su disponibilidad en las plataformas móviles, ya que la mayoría ofrecen una base de datos relacional, ya sea mediante *SQLite* en Android e iOS, *LINQ to SQL* en Windows Phone, o a través de *WebSQL* en HTML5.

A continuación se detallan las entidades de la base de datos local.

BusCompanies

Tabla con las compañías de ómnibus que forman parte del STM.

Campo	Tipo	Descripción
id	TEXT PRIMARY KEY	Identificador único de una compañía de ómnibus el sistema
name	TEXT	Nombre de la compañía de ómnibus

BusLines

Esta tabla contiene las líneas de ómnibus del sistema.

Campo	Tipo	Descripción
id	TEXT PRIMARY KEY	Identificador único de la línea de ómnibus en el sistema
line	TEXT	Nombre de línea
subline	TEXT	Nombre de sublínea
variant	INTEGER	Código de la variante del recorrido
route	TEXT	Recorrido de la línea, en JSON (desnormalizado)
company_id	TEXT	Clave foránea de la compañía a la que pertenece la línea

BusStops

Tabla que almacena las paradas de ómnibus de la ciudad.

Campo	Tipo	Descripción

id	TEXT PRIMARY KEY	Identificador único de la parada en el sistema
code	INTEGER	Identificador de la línea en el STM
street	TEXT	Nombre de la calle de la parada
corner	TEXT	Nombre de la esquina más cercana a la parada
lat	NUMBER	Latitud geográfica de la parada de ómnibus
lng	NUMBER	Longitud geográfica de la parada de ómnibus

StopLines

Tabla que representa la relación entre paradas de ómnibus y líneas de ómnibus.

Campo	Tipo	Descripción
stop_id	TEXT	Clave foránea de la parada de ómnibus
line_id	TEXT	Clave foránea de la línea de ómnibus

FriendshipRequests

Esta tabla contiene las solicitudes de amistad enviadas y recibidas por el usuario.

Campo	Tipo	Descripción
id	TEXT PRIMARY KEY	Identificador único de la solicitud de amistad
status	TEXT	Estado de la solicitud de amistad
requestee_id	TEXT	Clave foránea del usuario que recibió la solicitud
requester_id	TEXT	Clave foránea del usuario que realizó la solicitud
requester_name	TEXT	Nombre del usuario que realizó la solicitud
requester_email	TEXT	Email del usuario que realizó la solicitud

MeetingPoints

Tabla con las citas de punto de encuentro entre usuarios.

Campo	Tipo	Descripción
id	TEXT PRIMARY KEY	Identificador único de la cita o punto de encuentro
owner_id	TEXT PRIMARY KEY	Identificador del usuario creador del punto de encuentro
title	TEXT	Nombre de la cita o punto de encuentro
description	TEXT	Descripción del punto de encuentro
lat	NUMBER	Latitud geográfica del punto de encuentro
lng	NUMBER	Longitud geográfica del punto de encuentro
start_time	INTEGER	Fecha y hora de comienzo del encuentro en milisegundos
end_time	INTEGER	Fecha y hora de fin del encuentro en milisegundos
data	BLOB	Datos adicionales

data_type	TEXT	Tipo de los datos adjuntos

MeetingPointInvitations

Tabla que representa la relación entre un punto de encuentro y los usuarios invitados al mismo.

Campo	Tipo	Descripción
meeting_point_id	TEXT PRIMARY KEY	Identificador único de la cita o punto de encuentro
user_id	TEXT PRIMARY KEY	Identificador único de un usuario invitado al punto de encuentro
status	TEXT	Estado de la invitación al punto de encuentro

Friends

Esta tabla almacena la información de los amigos del usuario.

Campo	Tipo	Descripción
id	TEXT PRIMARY KEY	Identificador único del amigo
name	TEXT	Nombre público del amigo
email	TEXT	Correo electrónico del amigo
status_msg	TEXT	Mensaje para mostrar
phone	TEXT	Número celular del amigo
birthday	INTEGER	Fecha de nacimiento del amigo en milisegundos
lat	NUMBER	Última latitud geográfica conocida del amigo
lng	NUMBER	Última longitud geográfica conocida del amigo

FriendSearchHistory

Esta tabla almacena las búsquedas de amigos realizadas por el usuario que pueden ser de carácter geográfico o de búsqueda simple. Se utiliza para facilitar un historial de búsqueda al usuario.

Campo	Tipo	Descripción
name	TEXT	Valor utilizado en la búsqueda por nombre de amigo
email	TEXT	Valor utilizado en la búsqueda por email de amigo
phone	TEXT	Valor utilizado en la búsqueda por teléfono de amigo
operator	TEXT	Operador interno que determina si se realiza una búsqueda exhaustiva (OR), o específica (AND)
location_radius	INTEGER	Radio de búsqueda en metros tomando como referencia la ubicación del usuario

BusTrackHistory

Esta tabla almacena los últimos seguimientos de ómnibus realizados por el usuario, se utiliza para poder brindarle acceso rápido para seguimientos frecuentes

Campo	Tipo	Descripción
stop_id	TEXT	Identificador de la parada de ómnibus de referencia
line_id	TEXT	Identificador de la línea de ómnibus del seguimiento

FavouriteBusStops

Se almacenan las paradas de ómnibus favoritas del usuario.

Campo	Tipo	Descripción
stop_id	TEXT	Clave foránea de la parada de ómnibus

4.3 Algoritmo de Ómnibus más Cercanos

Este algoritmo se utiliza para calcular los ómnibus más cercanos a una parada de ómnibus particular que se encuentran realizando el recorrido activamente y aún no han pasado la parada. Se utiliza para automatizar el seguimiento de ómnibus, de forma de simplificar la interacción del usuario.

Al momento de consultar la ubicación en tiempo real para una línea de ómnibus, la *Aplicación Principal* realiza un pedido al *Servidor de Transmisión*, recibiendo los ómnibus activos para esa línea junto con su última ubicación conocida. A partir de esa información, el recorrido del ómnibus, y la ubicación geográfica de la parada de ómnibus se ejecuta el algoritmo, obteniendo los ómnibus o *viajes* más cercanos a la parada, que son los que resultan de mayor interés al usuario en un momento dado.

A continuación se presenta el pseudo-código del algoritmo:

```
findClosestRides(Location busStopLocation, Route route, List<Ride> rides) {
  // Proyectar la ubicación de los omnibus al recorrido de la línea
  for (Ride ride : rides) {
    ride.location = ESRI.getNearestVertex(route, ride.location);
  }

  Stack<Ride> closestRides;

  // Recorrer cada punto del recorrido, verificando si algún recorrido está allí
  for (Point point : route) {

    // Finalizar cuando se llega a la parada de ómnibus
    if(point == busStopProjection) {
       break;
    }

    // Agregar los recorridos a medida que los encontramos
    Ride ride = getRideInLocation(point);
    if (ride) {
      closestRides.push(ride);
    }
  }

  // Los recorridos quedan ordenados según la distancia a la parada
  return closestRides;
}
```

4.4 Diseño de la Aplicación

Uno de los retos del desarrollo de aplicaciones móviles es el diseño de la aplicación. El diseño de una aplicación está conformado en su mayor parte por dos componentes que van de la mano: el diseño de la experiencia de usuario, y el diseño de la interfaz gráfica [1].

El diseño de experiencia de usuario se trata de definir los objetivos de la aplicación, como qué funcionalidades incluir, y cómo permitir que el usuario realice estos objetivos. En cambio, el diseño de interfaz gráfica define como es esa experiencia visualmente. Esto incluye los colores, texturas, y fuentes que se utilizan para construir el estilo visual de la aplicación.

Realizar el diseño de la *Aplicación Principal* fue todo un desafío, ya que fue necesario integrar dos conjuntos de requerimientos dispares: la consulta de información y seguimiento de ómnibus, y las funcionalidades de red social de la aplicación que involucran a los amigos del usuario. Por este motivo, el diseño de la interfaz gráfica se planteó en varias etapas.

4.4.1 Diseño de Interacción

Para empezar, se dibujaron bosquejos de la interfaz gráfica o *wireframes*. En principio se utilizó la herramienta web Balsamiq [2] para realizar los primeros bocetos de la aplicación, manteniendo un fuerte foco en el contenido de la aplicación y las interacciones y dejando de lado los detalles menores. De esta forma se pudieron validar rápidamente los flujos principales de la aplicación, y obtener una primera noción de los componentes gráficos a utilizar y de la navegación de la aplicación. Los dibujos se presentan en la sección

Bocetos.

4.4.2 Diseño Gráfico

Una vez que los bocetos alcanzaron el nivel esperado de detalle y usabilidad, se pasó a la etapa de diseño gráfico. Esta vez se decidió realizar los bocetos de forma manual, lo cual si bien no era del todo fiel al modelo real, permitía identificar rápidamente los componentes gráficos a utilizar, por lo que resultaron de gran utilidad durante el desarrollo.

Fig. 4.1 Boceto de 'Perfil de Usuario'

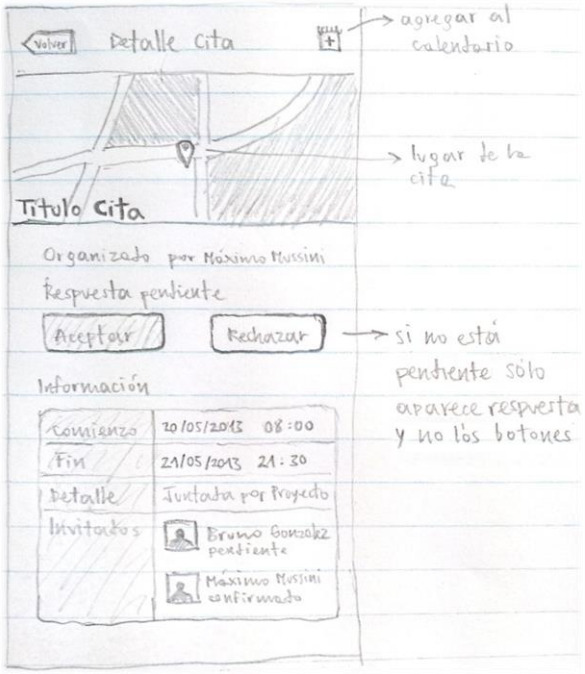


Fig. 4.2 Boceto de 'Detalle de Cita'

4.4.3 Diseñar para cada Plataforma

Debido a que hay tantas plataformas y dispositivos, al diseñar para móviles es importante saber cuál será la plataforma específica para la cual se desarrollará el producto final, ya que cada una tiene su propio paradigma y convenciones de diseño.

Para el diseño de la aplicación HTML5 se siguieron las convenciones de iOS6 [3], debido a que se utilizó el *framework* Sencha Touch, el cual tiene arraigadas las convenciones de estilo, navegación, e iconografía de esa plataforma. Esto es una desventaja cuando la aplicación se accede desde dispositivos de otras plataformas, ya que poseen convenciones diferentes y la usabilidad dela aplicación puede verse gravemente deteriorada.

En cambio, para la aplicación Android se respetaron los paradigmas de diseño propios de la plataforma [4], diferenciándose sobre todo en las diferencias de navegación debido al *back button* presente en los dispositivos Android que permite volver a la pantalla anterior, y la categorización de los dispositivos en base a tamaño y resolución de pantalla.

Una vez finalizados los bosquejos del diseño gráfico a grandes rasgos, comenzó el proceso de incorporación de elementos propios de cada plataforma, por lo que se puede decir que el diseño se dividió en dos partes: Sencha/iOS6 y Android. Las diferencias entre un estilo y otro se reflejan en las secciones correspondientes.

4.4.4 Consideraciones de Diseño

A continuación se presentan algunas de las consideraciones que se tuvieron durante el diseño de la aplicación, tanto el diseño de interacción como el diseño gráfico.

Controles

Para darle identidad a la aplicación se diseñaron versiones personalizadas de los controles utilizando una paleta de colores cálidos, con una fuerte presencia de amarillos y anaranjados. Esto ayuda al usuario identificar la aplicación en la que se encuentra en todo momento.

Fig. 4.3 Algunos componentes hechos a medida para la aplicación Android

Iconografía

En cada aplicación se utilizó una iconografía acorde a las convenciones descritas en la sección anterior, de forma de representar los conceptos de forma intuitiva y consistente en todas las pantallas de la aplicación.

Fig. 4.4 Iconos de la aplicación Android

Fig. 4.5 Iconos de menú de la aplicación Android

Fig. 4.7 Íconos de menú de la aplicación Sencha

Para permitir que el usuario pueda distinguir las líneas de ómnibus a las que está siguiendo con mayor facilidad, se utilizaron colores diferentes según la compañía de ómnibus a la cual pertenece la línea, tanto para los marcadores como para los recorridos que se muestran en el mapa. En lo posible se trataron de usar los colores característicos de cada compañía, de modo que la asociación entre los colores y la empresa resulten familiares.

Figura 4.1 Colores asociados a las distintas compañías del STM

Tipografía

Se decidió utilizar fuentes sans-serif para la tipografía de la aplicación, ya que el texto no es predominante. Se utilizaron mayormente fuentes de la familia Roboto [5], las cuales están

optimizadas para su legibilidad en dispositivos móviles y pantallas de alta resolución. Se utilizó la fuente en sus variedades *normal*, *light*, y *condensed*.

Menús

La barra de menú es uno de los elementos más prominentes de la interfaz gráfica, por lo cual es imprescindible reservar el espacio para las actividades cruciales de la aplicación. Para evaluar cuáles eran se utilizó el esquema *FIT* [6], que prioriza las siguientes acciones:

- *Frecuentes*: Aquellas acciones que se usan todo el tiempo, o que se utilizan varias veces seguidas. Casos donde un paso extra cada vez sería molesto.
- *Importantes*: Acciones que todo usuario debería descubrir, o que tiene que estar muy accesible cuando se necesita.
- *Típicas*: Las acciones a las que generalmente se les da importancia en aplicaciones similares.

Fig. 4.6 Iteraciones sobre el menú de la sección 'Amigos' de la aplicación HTML5

Luego de utilizar esta guía, se realizó un proceso iterativo para determinar la ubicación y el orden de las opciones de menú para lograr una experiencia más fluida y sin sorpresas, dejando las acciones más importantes a la vista.

Simple e Intuitivo

En las aplicaciones móviles cada pantalla debe tener un propósito bien definido, su diseño debe girar en torno al objetivo principal que quiere lograr el usuario. Esto se logró dándole énfasis a las acciones más importantes (a través de color, ubicación, o tamaño), y reduciendo la importancia visual de los elementos secundarios.

Un ejemplo de esto se encuentra en el perfil de usuario, donde la acción más importante es comenzar el seguimiento. Si bien hay varias acciones disponibles, como invitar al usuario a un punto de encuentro, ninguna compite con la acción principal.

Interfaces Táctiles

La gran mayoría de los dispositivos móviles poseen una pantalla táctil, por lo que para mejorar la usabilidad de la aplicación es importante asegurar que el área táctil de cada botón es suficientemente grande, ya que el usuario interactúa con el dispositivo a través de sus dedos en lugar del cursor de un mouse, perdiendo así precisión.

Por este motivo, se diseñó la aplicación de modo tal que el área de los elementos con los que se espera que el usuario interactúe fuera de al menos 1 cm^2, el tamaño promedio de la yema

de los dedos [7]. Esto no quiere decir que la parte visual del elemento sea tan grande, en algunos casos el área táctil se extendió más allá del tamaño visual, para lograr que la interacción resulte menos frustrante.

Resolución de Pantalla

La resolución de las pantallas de los dispositivos móviles varía en gran medida, por lo cual al desarrollar aplicaciones nativas es necesario crear versiones de cada gráfico que estén optimizadas para cada una de las distintas resoluciones a utilizar [8], al hacerlo se mejora la experiencia de usuario para diferentes tamaños y densidades de pantalla, evitando que los gráficos se vean pixelados o estirados.

En Android, los diferentes tamaños y densidades de pantalla están categorizados, de forma que solo es necesario crear un recurso gráfico para cada categoría, en lugar de tener que cubrir cada resolución. La aplicación para Android está optimizada adecuadamente para dispositivos de resolución mediana a muy alta, cubriendo alrededor del 90% de los dispositivos que están actualmente en funcionamiento [9].

Fig. 4.7 Un logo de la aplicación en distintas resoluciones

Para la vista principal de la aplicación, que permite seguir amigos y ómnibus a través de un mapa, se utilizaron transparencias en la barra de menú y los botones que se despliegan sobre el mapa, de forma de sacarle el mayor provecho posible al tamaño de pantalla del dispositivo.

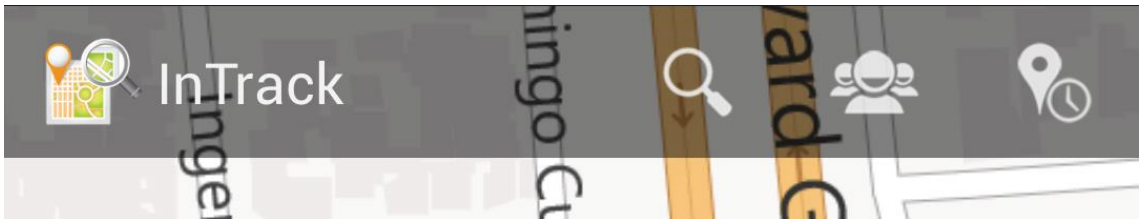


Fig. 4.8 Barra de menú de la aplicación con transparencia

Progressive Disclosure

En aquellos casos donde hay demasiada información como para mostrarla en pantalla, es conveniente utilizar pantallas adicionales. Al mostrar solo la información relevante, y mostrar más información a pedido del usuario, se logra reducir la confusión y la dificultad de la interacción. Esta técnica es conocida como *progressive disclosure* [10].

Por ejemplo, en la pantalla de *Crear Punto de Encuentro*, cuando el usuario desea seleccionar los amigos que están invitados al encuentro se lo lleva a una nueva pantalla de selección, que presenta una lista con sus amigos permitiéndole seleccionar o deseleccionar amigos. Algo

similar ocurre al seleccionar la ubicación de punto de encuentro, en la que se le presenta una pantalla con un mapa.

Fig. 4.9 Selección de amigos para un nuevo punto de encuentro

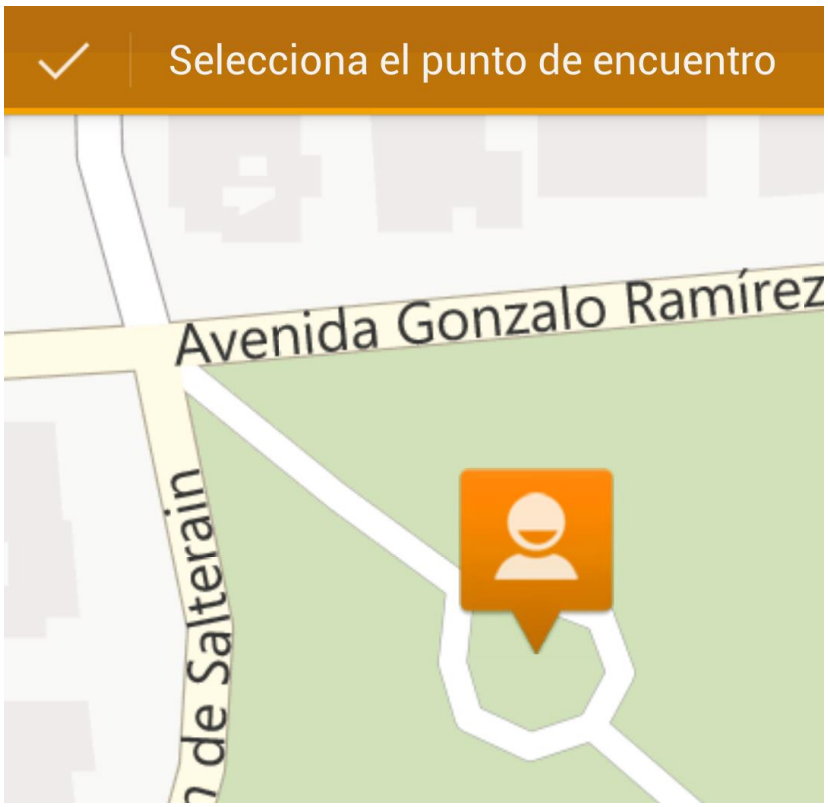


Fig. 4.10 Selección del punto de encuentro

4.4.5 Bocetos de Diseño de Interacción

En esta sección se presentan algunos de los bocetos realizados utilizando la herramienta Balsamiq, estos resultaron de mucha utilidad a la hora de definir las interacciones necesarias para que el usuario lleve a cabo un caso de uso.

Fig. 4.11 Diferentes *mockups* de *FindMe*

Fig. 4.12 Comienzo del flujo del seguimiento de ómnibus

Fig. 4.13 Continuación del flujo del seguimiento de ómnibus

5 Aplicación para Ómnibus

Esta aplicación para dispositivos móviles está pensada para estar correr en dispositivos que estén a bordo de los ómnibus, y hace uso intensivo del GPS y la conexión de datos del dispositivo, con el fin de enviar la posición geográfica del ómnibus en tiempo real al *Servidor de Transmisión*, el cual luego hará difusión de la misma a todos los usuarios que estén realizando seguimiento al ómnibus.

Presenta una interfaz de usuario muy simple que permite seleccionar la línea de ómnibus, visualizar el recorrido de la línea y la posición actual del ómnibus, y empezar o detener la transmisión durante el recorrido. Los datos de ómnibus son obtenidos a través del *Servidor Central.*

A su vez brinda la posibilidad de simular recorridos, para lo cual se brinda una interfaz muy similar, pero con opciones variadas como control de velocidad, pausado de recorrido, y con la posibilidad de simular múltiples recorridos al mismo tiempo. Esto es crucial para poder realizar pruebas en el ámbito del proyecto, ya que no es necesario que el dispositivo móvil se encuentre a bordo de un ómnibus en movimiento.

5.1 Componentes de Software

Esta aplicación móvil tiene comparte la arquitectura con la *Aplicación Principal*, con la diferencia de que se agrega un componente de simulación de recorridos de ómnibus que interactúa con el módulo de seguimiento.

User Interface

Es la interfaz gráfica de la aplicación móvil, y está diseñada con foco en la facilidad de uso. La visualización de la posición geográfica se ofrece como una funcionalidad secundaria, pero es un componente de complejidad baja.

Services Module

Al igual que su contraparte en la aplicación principal, este componente se encarga de comunicarse con el *Servidor Central* a través de los servicios REST que este expone, persistiendo localmente la información de ómnibus en una base de datos relacional. Como los datos de ómnibus son muy estables y la aplicación no realiza manipulación de datos, la única responsabilidad del componente es mantener la información de ómnibus actualizada.

Tracking Manager

Este módulo está encargado de controlar el ciclo de vida del viaje al iniciar y finalizar el recorrido el ómnibus, enviando actualizaciones geográficas de la posición del ómnibus al *Servidor de Transmisión*. Para ello utiliza la *Bus Transmission Library.*

El componente tiene una interfaz sencilla y bien definida que permite ser utilizado tanto por el simulador como por el usuario a través de la interfaz gráfica.

Simulator

Este componente se encarga de simular recorridos de ómnibus, utilizando el *Tracking Manager* para manejar el estado de cada recorrido, pero simulando la posición geográfica de cada móvil de forma similar a como lo haría un ómnibus que hace el recorrido de una línea particular. Permite variar la velocidad y el estado de los recorridos simulados, y simular múltiples recorridos de forma simultánea.

Bus Transmission Library

Este componente implementa el lado cliente de la API de comunicación con el *Servidor de Transmisión*, en particular cumple con el *Contrato para la Aplicación de Ómnibus*, y utiliza la API de producción de datos de ómnibus.

ArcGIS for Mobile

Como funcionalidad secundaria se muestra el recorrido de la línea y la posición geográfica del ómnibus en un mapa, para lo cual también se utiliza la biblioteca ArcGIS.

5.2 Base de Datos SQL

Los datos de ómnibus que necesita la aplicación se almacenan localmente en una base de datos relacional, para que se disponga de los datos aún si el *Servidor Central* no estuviese disponible.

A continuación se detallan las entidades de la base de datos local.

BusCompanies

Tabla con las compañías de ómnibus que forman parte del STM.

Campo	Tipo	Descripción
id	TEXT PRIMARY KEY	Identificador único de una compañía de ómnibus el sistema
name	TEXT	Nombre de la compañía de ómnibus

BusLines

Esta tabla contiene las líneas de ómnibus del sistema.

Campo	Tipo	Descripción
id	TEXT PRIMARY KEY	Identificador único de la línea de ómnibus en el sistema
line	TEXT	Nombre de línea
subline	TEXT	Nombre de sublínea
variant	INTEGER	Código de la variante del recorrido
route	TEXT	Recorrido de la línea, en JSON (desnormalizado)
company_id	TEXT	Clave foránea de la compañía a la que pertenece la línea

Rides

Esta tabla contiene las simulaciones de ómnibus activos, no se utiliza cuando la aplicación opera en el modo normal. Un viaje o *ride* está identificado por una línea de ómnibus, y el momento en que comienza el recorrido.

Campo	Tipo	Descripción
id	TEXT PRIMARY KEY	Identificador único del viaje en el sistema
line_id	TEXT	Clave foránea de la línea de ómnibus del viaje
status	TEXT	Estado del viaje (NotStarted, Active, OnHold)
timestamp	INTEGER	Momento en que comenzó el viaje (en milisegundos)
lat	NUMBER	Última latitud geográfica simulada del ómnibus
lng	NUMBER	Última longitud geográfica simulada del ómnibus
route_position	INTEGER	Índice del punto más cercano del recorrido
speed	NUMBER	Velocidad simulada del ómnibus (en km/h)

6 Glosario

API	Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
HTML	Es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.
REST	El término se aplica en la actualidad a cualquier interfaz web simple que utiliza HTTP, con un protocolo cliente/servidor sin estado y operaciones bien definidas, donde cada recurso puede ser accedido utilizando un identificador global único.
WebSockets (WSS)	Tecnología que proporciona un canal de comunicación bidireccional sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede ser por cualquier aplicación cliente/servidor.
TCP	Protocolo de comunicación de transmisión fiable orientado a conexión, opera al nivel de capa de transporte.
Viaje	Ómnibus que está realizando el recorrido de una línea particular en un instante dado.
Wireframe	Representación esquemática de una aplicación, es una guía visual básica que intenta sugerir la ubicación de los elementos de diseño clave en la interfaz gráfica. Sirven como herramienta de comunicación y discusión entre desarrolladores, diseñadores y clientes.
Pixelado	Efecto causado por visualizar una imagen a un tamaño en el que los pixels individuales son visibles al ojo.
Progressive Disclosure	Técnica de diseño de interacción que permite retener la atención del usuario al reducir la confusión y la carga cognitiva de la tarea a realizar.
Sans-serif	También llamadas fuentes de palo seco o sin remates, son fuentes que en cada carácter no tienen unas pequeñas terminaciones llamadas remates o serifas.

7 Bibliografía

[1] Charles B. Kreitzberg and Ambrose Little. Agile Ux Development. [Online]. http://msdn.microsoft.com/en-us/magazine/dd882523.aspx

[2] Balsamiq Studio. Balsamiq. [Online]. http://balsamiq.com/

[3] Apple. iOS Human Interface Guidelines: Designing for iOS. [Online]. https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/

[4] Android Developers. Android Design Guidelines. [Online]. http://developer.android.com/design/index.html

[5] Christian Robertson. Fonts - Roboto. [Online]. http://www.google.com/fonts/specimen/Roboto

[6] AndroidDevelopers. ActionBar. [Online]. http://developer.android.com/design/patterns/actionbar.html#ActionButtons

[7] Benjamin Bederson, Amy Karlson, and Pekka Parhi. (2006) Target Size Study for One-Handed Thumb Use on Small Touchscreen Devices. [Online]. http://research.microsoft.com/pubs/75812/parhi-mobilehci06.pdf

[8] AndroidDevelopers. Supporting Multiple Screens. [Online]. http://developer.android.com/guide/practices/screens_support.html

[9] Google. (2013, Octubre) Dashboards - Screen Sizes and Densities. [Online]. http://developer.android.com/about/dashboards/index.html#Screens

[10] Larry Constantine and Lucy Lockwood. Instructive Interaction: Making Innovative Interfaces Self-Teaching. [Online]. http://foruse.com/articles/instructive.pdf

[11] Allen Holub. Una referencia rápida de UML. [Online]. http://www.holub.com/goodies/uml/

[12] IBM. Tutorial de Diagramas de Secuencia. [Online]. http://www.ibm.com/developerworks/rational/library/3101.html

[13] Trace Modeler. Tutorial de Diagramas de Secuencia. [Online]. http://www.tracemodeler.com/articles/a_quick_introduction_to_uml_sequence_diagrams/index.html

[14] Microsoft. Diagramas de secuencia UML: Referencia. [Online]. http://msdn.microsoft.com/es-es/library/dd409377.aspx

[15] Craig Larman, *Applying UML and patterns: An introduction to object-oriented analysis and design*.: Prentice Hall, 1997.