Facultad de Ingeniería
Universidad de la República

Extensión Espacial de Middleware orientado a mensajería para dar soporte a servicios basados en geolocalización

Maestría en Sistemas de Información y Tecnologías de Gestión de Datos

Tesis de maestría
Diciembre 2018

Ing. Miguel Merlino
Tutor: Msc. Ing. Raquel Sosa

Resumen

La evolución sostenida y vertiginosa de los dispositivos móviles (celulares, tablets, agendas digitales, etc.) y de las redes inalámbricas (WiFi y las redes móviles) en el mundo ha fomentado el desarrollo de múltiples aplicaciones para diferentes usos en las plataformas de los dispositivos. En particular, muchas de las aplicaciones actuales aprovechan la función de georreferenciación del dispositivo, que permite obtener su localización geográfica de algún sistema de posicionamiento (típicamente, GPS). Como consecuencia directa, han surgido los llamados servicios basados en localización (LBS), que permite brindar al usuario del dispositivo determinado servicio relacionado a su ubicación actual.

Se desea proveer un LBS enmarcado en la plataforma de gobierno electrónico (PGE) de Uruguay, mantenida y gestionada por la AGESIC (Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento), que ayude al Estado uruguayo a satisfacer mejor las necesidades de la población, brindando información a partir de la ubicación de los ciudadanos mediante el envío de notificaciones utilizando algún mecanismo de mensajería confiable. Las notificaciones pueden abarcar diferentes categorías: emergencias, meteorología, cursos temáticos, accidentes de tránsito, locales comerciales, comida, inseguridad, cultura, medio ambiente, vivienda, etc. Actualmente, existen numerosos casos de notificaciones del Estado que se realizan sobre diferentes canales. Ejemplos de esto son avisos del Ministerio del Interior sobre un incendio en el barrio de Cordón a través de la red social Twitter o avisos de alertas meteorológicas que cuelga INUMET en su sitio web oficial. Por este motivo, surge la necesidad de definir un único canal de notificaciones oficial y que sea utilizado por todos los organismos del Estado y sus ciudadanos con el fin de comunicar eventos.

En el proceso de desarrollo de la tesis, se planteó el problema y se identificó a los diferentes actores involucrados. Luego, se especificó a alto nivel los requerimientos funcionales y no funcionales de la plataforma a construir. Se propuso una arquitectura de solución para el problema, identificando las diferentes partes que componen la solución y sus responsabilidades y, finalmente, se diseñó un prototipo que implemento un subconjunto de las funcionalidades de la plataforma.

En relación a los trabajos relacionados encontrados en el área de tecnologías de la información, la propuesta cuenta con algunas ventajas comparativas. Por un lado, permite a los ciudadanos conocer diversa información oficial publicada por múltiples organismos de la esfera estatal y relacionada a su localización actual. Por otro lado, utilizando a la plataforma de gobierno electrónico como canal de comunicación entre Estado y ciudadanos, el Estado puede publicar información de interés general y los usuarios pueden suscribirse a la información de su preferencia en forma explícita, recibiendo notificaciones de manera confiable y segura.

El principal aporte de la tesis es lograr una mejora de los servicios públicos del Uruguay, proveyendo nuevas formas de notificación a los usuarios del público general utilizando los mecanismos y herramientas de la PGE de AGESIC.

Palabras clave: LBS, MOM, PGE, AGESIC, GeoMOM, GIS.

Contenido

1. Introducción

El presente documento es la tesis de Maestría de Sistemas de Información y Tecnologías de Gestión de Datos propuesto por el LINS (Laboratorio de Integración de Sistemas) del Instituto de Computación (InCo) de la Universidad de la República (UDELAR) enmarcado en las áreas de sistemas de información geográficos empresariales (GIS su sigla en inglés) y tecnologías de Middleware.

1.1. Contexto y motivación

Actualmente, se ha masificado el uso de dispositivos móviles en el Uruguay. Hoy en día, muchas de las aplicaciones que fueron utilizadas en la primera década del siglo XXI en computadores personales, podrían ejecutarse adecuadamente en cualquiera de los dispositivos móviles que el mercado ofrece (tabletas, celulares, etc.) ya que han alcanzado grandes capacidades de procesamiento y memoria. En particular, el uso de sistemas de geolocalización (GPS) en dispositivos móviles ha permitido el desarrollo de múltiples servicios que utilizan la localización (en forma de coordenadas geográficas) del dispositivo para proveer variedad de información al usuario. Con dichos servicios, el usuario no solo puede consultar en qué lugar está ubicado, sino también consultar otro tipo de información relevante y relativa a esa ubicación, como encontrar ciertos lugares de interés (restaurantes, hoteles, etc.), personas, rutas, eventos o zonas que cumplan con ciertas restricciones [1].

En este contexto, surgen los servicios basados en localización (Location-Based Services o LBS) que se definen como los servicios que proporcionan datos e información sujetos a la localización actual (real o proyectada) y al contexto de un usuario que utiliza un dispositivo móvil [2]. Un LBS basado en notificaciones depende de algún mecanismo de integración de aplicaciones que resuelva la interacción entre los usuarios finales del servicio (quienes son notificados) y los publicadores de los datos (que proveen datos a los usuarios).
Para un sistema basado en notificaciones, se puede pensar en un sistema de mensajería que permita recibir, almacenar, distribuir y entregar los mensajes de forma fiable y segura. Dicho mecanismo puede ser un middleware orientado a mensajería o MOM (Message Oriented Middleware), definido como una infraestructura de middleware que soporta envío y recepción de paquetes de datos de forma sincrónica o asincrónica entre diferentes participantes [3]. El mismo permite establecer los actores que interactúan, los formatos de los mensajes y patrones de mensajería que definen mecanismos de distribución y entrega de los mensajes. En la propuesta, se hará foco en el patrón publish-subscribe, idóneo para la construcción de un sistema de notificaciones. En este esquema, existe una entidad que produce un mensaje y lo envía a un canal. En este canal, se suscriben o registran muchos consumidores [4] [5]. Por otra parte, el uso de un MOM ofrece diversos beneficios, como ser entrega confiable, asincronismo y la posibilidad de contar con un gran número de suscriptores.
Un LBS basado en notificaciones puede desplegarse en una plataforma de gobierno electrónico. Gobierno electrónico (e-gov) se puede definir como el uso que hace un gobierno de las tecnologías de la información para brindar a los ciudadanos y las empresas la oportunidad de concretar asuntos de gobierno utilizando diferentes medios electrónicos [6]. De esta forma, mediante e-gov se puede publicar información de interés general desde la órbita del Estado para notificar a los ciudadanos de ciertos eventos de relevancia local o nacional. Actualmente, en el Estado uruguayo existe un LBS orientado a notificaciones del SINAE (Sistema Nacional de Emergencias) a través de e-gov. Si bien es un caso de referencia para la propuesta, por ahora es solo un sistema hecho a medida y ajustado para

SINAE, con lo cual no podría reutilizarse para otro organismo. Además, solo permite realizar notificaciones puntuales y relacionadas a emergencias.

Como motivación, se puede pensar en un LBS que permita brindar información a usuarios de diversa índole, como ser de meteorología, crimen, tránsito, catástrofes naturales, etc., utilizando geolocalización. A través de dicho LBS un usuario puede ser notificado mediante alertas o eventos desde diferentes proveedores del Estado (INUMET [7] y SINAE [8] para meteorología, Dirección Nacional de Policía Caminera [9] para tránsito, etc.) en los diferentes departamentos del Uruguay. El servicio debe generar las notificaciones en base a la localización del usuario, la localización del evento o de la alerta y las restricciones establecidas por el usuario. En la Figura 1.1 se puede visualizar un ejemplo de una alerta meteorológica con su respectivo alcance geográfico. La alerta es disparada por el INUMET y corresponde a un reporte de fuertes tormentas en la zona centro-occidental del país.

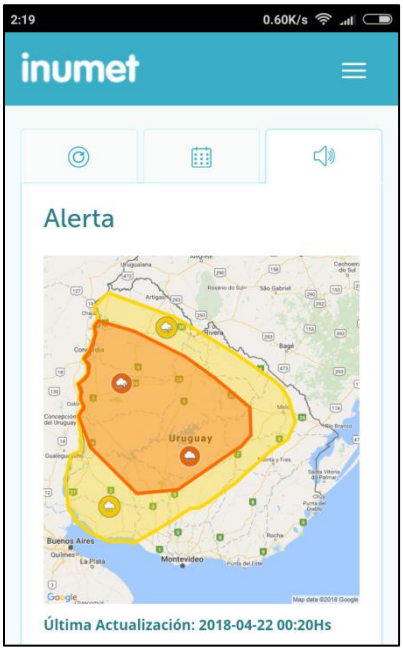

Figura 1.1: Ejemplo de alerta

La información geográfica puede representarse mediante un modelo de la información espacial. El OGC ha sido pionero en la definición de un modelo vectorial y estándar que propone una arquitectura con los aspectos simples de las geometrías (Simple Feature Access - SFA) [10]. El SFA se enfoca en la representación abstracta de las geometrías y sus posibles operaciones. Algunas operaciones básicas a utilizar por la lógica del sistema a construir, en orden de importancia, son: intersección, distancia, buffer y unión.

En la actualidad, existen LBSs con cada vez más proveedores y ofreciendo notificaciones a los usuarios de todo tipo. Además, existen MOMs especializados en Integración de Aplicaciones a través de mensajería (y notificaciones) y plataformas de gobierno electrónico orientadas a comunicar el Estado con sus ciudadanos. Por ello, surge como necesidad de que el Estado pueda proveer notificaciones a través de e-gov a los usuarios del público general, en función de sus preferencias y su localización actual, con las garantías que ofrecen los sistemas de MOMs.

1.2. Objetivos

El objetivo general de la tesis es proponer una plataforma tipo MOM extendida con capacidades geoespaciales que brinde soporte al desarrollo de LBS en un contexto de e-gov.

Los objetivos específicos de la tesis son:

- Relevar y analizar el conocimiento existente en middleware orientado a mensajería y modelos de representación espacial de GIS.
- Estudiar trabajos relacionados a esta tesis, que propongan alguna solución en el marco de tecnologías de middleware y GIS.
- Analizar e identificar requerimientos funcionales y no funcionales para una plataforma MOM con capacidades geoespaciales.
- Proponer arquitectura y diseño de la plataforma tipo MOM con capacidades geoespaciales, en base a los requerimientos.
- Diseñar e implementar un prototipo de la plataforma para evaluar su factibilidad técnica.
- Desarrollar un Caso de Estudio de interés para el contexto uruguayo que demuestre el uso de la plataforma.

La tesis apunta a proponer una extensión de los MOMs utilizando el modelo de datos espaciales SFA, con el objetivo de mejorar las capacidades de los LBSs basados en notificaciones y utilizando las capacidades de e-gov.

1.3. Resultados esperados

Los resultados esperados de esta tesis son:

- Relevamiento y análisis del conocimiento existente en el área.
- Documento de Análisis de Requerimientos
- Propuesta de una arquitectura de una plataforma tipo MOM geoespacial.
- Desarrollo de un prototipo de la arquitectura propuesta.
- Desarrollo de un Caso de Estudio de LBS en base a la plataforma propuesta.

1.4. Cumplimiento de objetivos y aportes de la tesis

Se logró el cumplimiento de los objetivos tanto generales como específicos. Se relevó el conocimiento existente en las áreas de GIS, LBS, e-gov y MOM y se buscaron trabajos relacionados. En segunda instancia, se propuso el problema de manera clara y consistente, definiendo los diferentes requerimientos de la plataforma a construir y sus actores. En base a esos requerimientos, se propuso una arquitectura solución del problema, con la especificación de las interfaces y las responsabilidades de cada componente. Finalmente, se diseñó e implemento un prototipo que satisfizo un subconjunto de las funcionalidades de la solución propuesta. Para validar el funcionamiento de prototipo, se ideó un caso de estudio conciso y aplicado a la realidad uruguaya que permite validar las interacciones entre los diferentes actores. Finalmente, se elaboraron las conclusiones del trabajo.

1.5. Estructura del documento

El presente documento se organiza de la siguiente manera: en el capítulo 2 se introduce el marco conceptual; en el capítulo 3 se establecen los requisitos para la plataforma; en el capítulo 4 se describe

la solución propuesta; en el capítulo 5 se especifica el prototipo y se presenta un caso de estudio aplicado a la realidad uruguaya. Finalmente, en el capítulo 6 se presentan las conclusiones y líneas de trabajo futuro.

2. Marco conceptual

Este capítulo define aspectos descriptivos y técnicos de gobierno electrónico, modelos de datos espaciales, un lenguaje de consulta aplicado a un modelo particular, servicios basados en localización y sistemas de mensajería. Respecto a gobierno electrónico, se hará foco sobre la plataforma de gobierno electrónico de Uruguay. Por último, se incluye una sección de trabajos relacionados que servirán de base a la propuesta de la tesis.

2.1. Gobierno electrónico

El concepto de gobierno electrónico (e-gov) surge en 1993 como una idea del ex vicepresidente norteamericano Al Gore de conectar a los ciudadanos con varios organismos y agencias del gobierno de forma de obtener diversos servicios de forma automática, así como ayudar al propio gobierno a organizar y agilizar el trabajo interno [11].

2.1.1. Definición y niveles de maduración

En [6], se define a e-gov como el uso que hace un gobierno de las tecnologías de la información para brindar a los ciudadanos y las empresas la oportunidad de concretar asuntos de gobierno utilizando diferentes medios electrónicos. Es un elemento fundamental para modernizar el sector público a través de la identificación de la estructura organizacional, las formas de comunicación entre los ciudadanos y las empresas, y reduciendo costos y complejidad en los procesos de negocio. E-gov permite lograr conexiones estratégicas entre las instituciones públicas y sus departamentos, así como la comunicación a diferentes niveles de gobierno (zona, ciudad, localidad, nación, etc.) [12].
En el concepto de e-gov se pueden distinguir tres elementos importantes: ciudadanos (G2C), empresas y servicios (G2B), y gobiernos departamentales de un país (G2G). La mayoría de los gobiernos comienzan publicando la información para acceso online, lo cual no siempre va aparejado de la definición y funcionamiento de servicios apropiados para uso del público general. Según el panorama general de e-gov que tenga un país, se pueden identificar diversos estadíos:

1. Uso de una red interna que incluye un sistema de envío y recepción de emails.
2. Habilitación del acceso a la información al público y a otras organizaciones.
3. Se permite la comunicación bidireccional desde y hacia el gobierno.
4. Se logra el intercambio de valor desde y hacia el gobierno
5. Se consolida la democracia digital – todos los ciudadanos adultos pueden participar en el desarrollo, propuesta y creación de leyes a través de e-gov.

Por otra parte, la implementación de e-gov es un proceso continuo que evoluciona en diferentes etapas. En [13], se define el modelo de maduración de Layne y Lee, que describe diferentes etapas de maduración de e-gov. Dichas etapas, ilustradas en la Figura 2.1, son:
1) Catalogación. En esta etapa, el gobierno crea una "página del Estado" en respuesta a la presión de la prensa, los empleados de IT, demandas de los ciudadanos y otros stakeholders. Dada la escasez de experiencia en IT, se realiza en un proyecto de pequeña escala en el cual se publica información no transaccional. Esto resulta ventajoso para el gobierno que ahorra una gran cantidad de tiempo en responder preguntas básicas sobre servicios, trámites y procedimientos institucionales. A medida que esta etapa avanza, mejora la calidad de la información publicada y deviene la necesidad de interrelacionar la información a través de links.
2) Transacción. A medida que los sitios web del gobierno progresan, tanto los funcionarios como los ciudadanos comprenden el valor de Internet como otro canal de comunicación en común. En esta

etapa, los ciudadanos requieren principalmente poder iniciar los trámites de gobierno online en lugar de asistir a determinado lugar a llenar formularios manualmente. En esta fase transaccional el gobierno ocupa un rol activo: el gobierno a través de los funcionarios provee respuestas, recepciones de mensajes, confirmaciones de resultados, etc.

3) Integración vertical. En esta etapa el objetivo es la evolución de los servicios de gobierno más que automatizar y digitalizar los procesos. Luego de que los servicios transaccionales se vuelven prevalentes y maduros, las expectativas de los ciudadanos conducen a la integración de los sistemas a diferentes niveles en los servicios. A este punto, las instituciones mantienen bases de datos separadas que no están interconectadas con las de otros organismos. La integración vertical consiste en unificar el acceso a la información entre instituciones a diferentes niveles de jerarquía y operando en el mismo rubro (por ejemplo, mutualistas y Ministerio de Salud Pública).

4) Integración horizontal. Es el mejor escenario de evolución de e-gov para el punto de vista del ciudadano, ya que permite integrar servicios a diferentes niveles funcionales, alcanzando los sectores público y privado. Se logra la integración y compartimiento de la información entre las bases de datos de las instituciones. La integración horizontal implica que una transacción puede ser llevada a cabo por múltiples organismos con diferentes roles, funciones y responsabilidades.

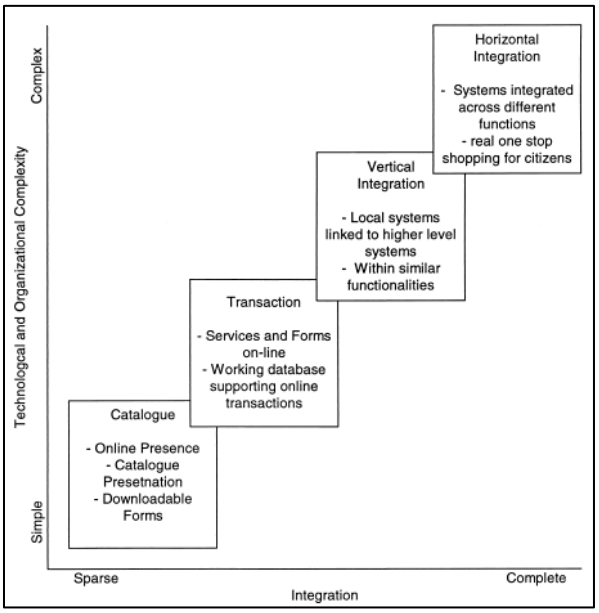

Figura 2.1: Modelo de maduración de Layne y Lee[1]

2.1.2. Desafíos

En [14], se mencionan los desafíos más comunes que debe afrontar la implementación de e-gov, así como recomendaciones para solventarlos. Entre esos desafíos de e-gov se pueden destacar:

- Desarrollo de infraestructura. El montaje de una arquitectura de e-gov requiere una infraestructura que involucre las tecnologías más avanzadas, de forma de que la comunicación entre los stakeholders y el gobierno sea lo más performante posible.
- Leyes y políticas públicas. La legislación de los países debe promover y apoyar el uso, actualización y reconocimiento de documentos electrónicos y transacciones.
- Brecha digital. Los ciudadanos (o la mayoría de ellos) deberían contar con acceso a Internet.

[1] Tomado de *Open Government Data: A Stage Model* [13]

- Accesibilidad. Los servicios deben ser accesibles a todos los individuos de la sociedad, independientemente de sus capacidades físicas. Por ello, se deben diseñar interfaces de uso apropiadas.
- Transparencia. El uso de e-gov debería favorecer la transparencia entre el gobierno y los ciudadanos a través de la participación activa de ambos.
- Interoperabilidad. E-gov debe poder adaptarse a las interfaces y formatos de mensajes de las diferentes organizaciones, tanto del sector público o privado y a través de diferentes organismos del estado en diferentes funciones (integración horizontal).
- Privacidad. Los gobiernos deben ser responsables de preservar la información personal de los ciudadanos, garantizando la confidencialidad e integridad de sus datos.
- Autoevaluación. Los gobiernos deben regularmente evaluar el progreso y eficacia de e-gov y determinar si los objetivos y las metas están siendo cumplidas.
- Disponibilidad. Debe ser 24/7.

2.1.3. Arquitectura general de E-Gov

La arquitectura de e-gov define estándares, componentes de infraestructura, aplicaciones, tecnologías, modelos de negocio y lineamientos para intercambio de información entre organizaciones, lo cual facilita la comunicación y promueve la productividad colaborativa [15]. En esta sección se propone un framework para e-gov que pretende conjugar la infraestructura de IT con la gestión de procesos de negocio en las organizaciones del sector público. El framework, que se manifiesta en la Figura 2.2, consiste en diversas capas:

- Capa de acceso. Engloba todos los canales de comunicación que emplean los usuarios para acceder a los servicios gubernamentales. Los usuarios pueden ser ciudadanos, empresas, empleados, cooperativas u otros gobiernos. Dichos canales pueden ser online (dispositivos celulares, sitios de internet) u offline (televisión digital). El objetivo de esta capa es proveer una forma común de acceso a la información y los servicios, mantener coordinación entre los diferentes canales y crear un look and feel común para todas las interfaces.
- Capa E-Gov. Esta capa se encarga de integrar la información digital de varias organizaciones en un portal web de servicios de gobierno. Los portales permiten la integración del gobierno con el resto de los actores (G2C, G2B y G2G). Permite al usuario acceder a la información a través de un browser. En definitiva, el portal constituye una aplicación web de frontend que ayuda a unificar la presentación de la información procedente de diversas fuentes.
- Capa de negocio. Esta capa se enfoca en el uso de aplicaciones, herramientas y plataformas para fomentar el compartimiento y procesamiento de la información, y generar redes de trabajo (VPNs) entre las organizaciones. La integración de bases de datos, procesos y aplicaciones de gobierno desempeña un rol critico en esta capa dado que e-gov depende en gran medida de datos, sistemas y procesos de la matriz organizacional existente, lo cual hace asequible la intercomunicación entre sistemas modernos y emergentes con los sistemas legados. El desarrollo sostenido de ICTs en las dos últimas décadas ha visto una vasta variedad de aplicaciones y tecnologías para integración de la infraestructura. Los principales mecanismos de integración han sido los siguientes:
    - ERP (Enterprise Resource Planning). Facilita la integración pero no permite hacer cambios significativos en los sistemas.
    - EAI (Enterprise Application Integration). Permite controlar, distribuir y gestionar la información que se propaga a través de la organización.

- Web Services. Son de baja complejidad y costo. Permite comunicar a los sistemas sin tener en cuenta la infraestructura subyacente.
- CRM (Client Resource Management). Es un sistema que mantiene la información de los ciudadanos y puede brindar toda clase de información personal.
- Datawarehouse. Permite almacenar e integrar datos históricos y agregados desde fuentes de datos heterogéneas y distribuidas.

- Capa de infraestructura. Esta capa se enfoca en el uso de tecnologías que deberían ser empleadas en las plataformas de e-gov, incluyendo estándares y protocolos de comunicación. Estas tecnologías deben soportar la adquisición, almacenamiento y transformación de datos, sean de fuentes internas o externas.

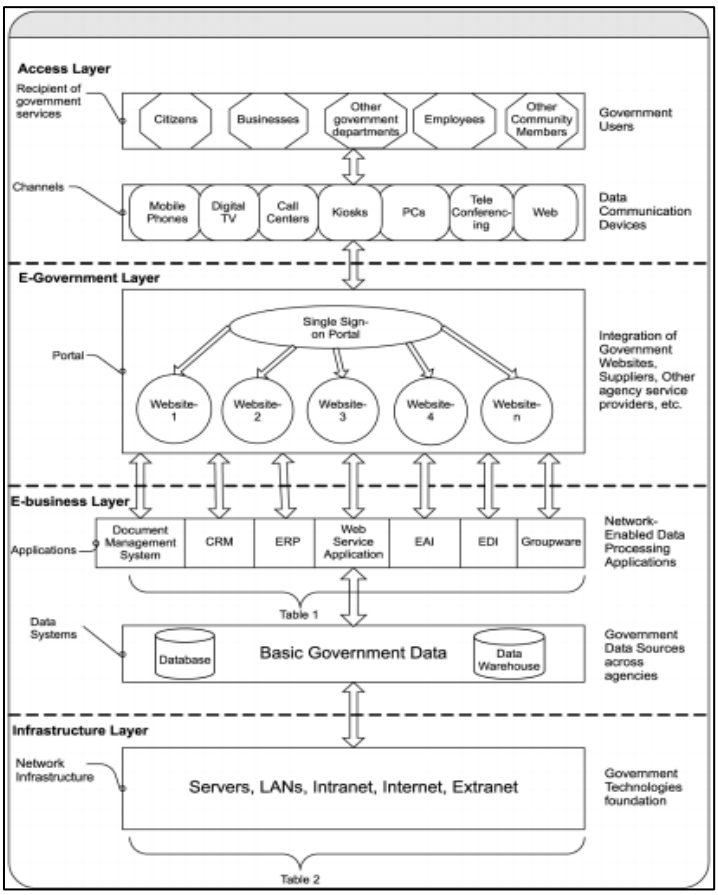

Figura 2.2: Arquitectura general de e-gov [2]

2.1.4. Caso uruguayo: AGESIC

AGESIC (Agencia para el Desarrollo del Gobierno de Gestión Electrónica y la Sociedad de la Información y del Conocimiento) es el organismo que lidera la estrategia de implementación de Gobierno Electrónico en Uruguay como base de un Estado centrado en el ciudadano. Promueve la inclusión, la apropiación y el buen uso de las Tecnologías de la Información y de las Comunicaciones en el Estado uruguayo [16].

Entre sus actividades permanentes se encuentran:

- Definir y difundir la normativa informática.

[2] Tomado de *E-government adoption: architecture and barriers* [15]

- Analizar las tendencias tecnológicas.
- Desarrollar proyectos en tecnologías de la información y comunicaciones.
- Asesorar en materia informática a las instituciones públicas del Estado.
- Capacitar y difundir en materia de e-gov, apoyando la transformación y transparencia del Estado.

2.1.4.1. Plataforma de Interoperabilidad

La Plataforma de Interoperabilidad (PDI) forma parte de la Plataforma de Gobierno Electrónico (PGE) de AGESIC y tiene como objetivo general facilitar y promover la implementación de servicios de e-gov en Uruguay. Para esto, la PDI brinda mecanismos que apuntan a simplificar la integración entre los organismos del Estado y a posibilitar un mejor aprovechamiento de sus activos. Provee infraestructura (hardware y software) y servicios utilitarios, que reducen la complejidad de implementar servicios al público y/o accesibles dentro del Estado. Por otra parte, la PDI aporta los mecanismos técnicos para implementar servicios compuestos, basados en los ofrecidos por diferentes organismos, normalizando e integrando la información proveniente de éstos. A nivel tecnológico, la PDI posibilita que los organismos provean sus funcionalidades de negocio a través de servicios de forma independiente a la plataforma en la que fueron implementados. Esto corresponde a la implementación de una SOA a nivel del Estado, en la cual los servicios ofrecidos por los organismos son descriptos, publicados y descubiertos, invocados y combinados a través de interfaces y protocolos estandarizados.
La Plataforma de Interoperabilidad está basada en una arquitectura orientada a servicios (SOA) e integrada por un sistema de control de acceso, un sistema de gestión de metadatos y una plataforma de middleware. Estos componentes facilitan la provisión, búsqueda e invocación de los servicios web que son brindados por los organismos, así como la interoperabilidad e interacción segura entre los mismos [17].
El **sistema de control de acceso**, integrado por una suite de productos que permiten la autenticación y autorización para el consumo de servicios basados en XML, oficia de puerta de entrada a la Plataforma. El **sistema de gestión de metadatos** provee una especificación de alto nivel de los conceptos relativos a servicios públicos, de forma de evitar, o eventualmente resolver, ambigüedades en el manejo de estos conceptos por parte de los organismos. El conocimiento en este sistema se maneja a través de ontologías, utilizando OWL como lenguaje de especificación, y Protege como herramienta de modelado. Finalmente, el **componente de middleware** de la PDI cuenta con mecanismos para facilitar el desarrollo, despliegue e integración de servicios y aplicaciones. Los organismos pueden utilizar esta plataforma para publicar y descubrir servicios. En la subsección siguiente se analizara este componente con mayor nivel de detalle.
En la Figura 2.3, se ilustra la arquitectura de la plataforma y cada uno de sus componentes.

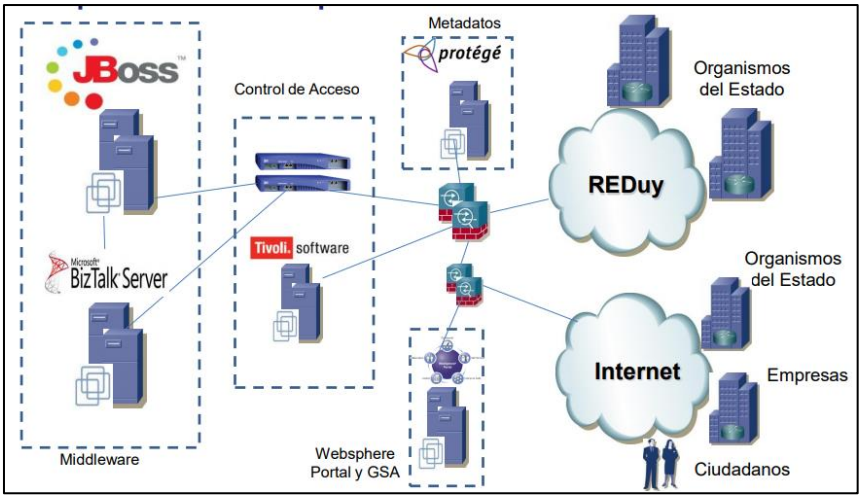

Figura 2.3: Arquitectura de la PDI [3]

2.1.4.2. Componentes de la Plataforma de Middleware

A grandes rasgos, en el middleware se pueden apreciar tres grandes bloques: entornos de ejecución para aplicaciones y servicios, un registro de servicios y productos de tipo ESB.
La Plataforma de Middleware proporciona **entornos de ejecución** a través de dos de las principales plataformas para el desarrollo de aplicaciones empresariales: la plataforma .NET de Microsoft y la plataforma Java Enterprise Edition (Java EE).
El **Registro de Servicios** de la PGE provee funcionalidades para que los organismos publiquen, describan, busquen y descubran servicios en la PGE. Además de los nombres, proveedores y categorías de los servicios, es posible acceder a la descripción de los mismos, especificada en WSDL (Web Service Description Language), que brinda los datos necesarios para que una aplicación cliente pueda invocarlos.
Los productos de ESB de la Plataforma de Middleware constituyen un amplio sistema de mensajería (MOM) para WS que provee mecanismos que pueden ser utilizados por los organismos para el consumo y provisión de servicios. Algunos de estos mecanismos son:

- Transparencia de ubicación. Los productos de ESB proveen mecanismos que permiten a la PGE brindar transparencia en la ubicación de los servicios que se acceden a través de ella. Cuando una aplicación cliente quiere invocar un servicio, debe enviar un pedido a la PGE especificando, a través de una dirección lógica, el servicio que se quiere invocar. Esta dirección lógica identifica al servicio en la plataforma. El mapeo entre direcciones lógicas y físicas es gestionado en los ESBs. Para especificar la dirección lógica del servicio que se quiere invocar, la aplicación cliente debe utilizar WS-Addressing.
- Mecanismos de mensajería confiable. Los productos de ESB de la PGE permiten brindar mecanismos de mensajería confiable utilizando los modelos point-to-point y publish-subscribe (PS).
- Transformación y Enriquecimiento de Mensajes. Los productos de ESB de la PGE proveen varios mecanismos para transformar y enriquecer mensajes que fluyen entre clientes y servicios, por ejemplo, utilizando XSLT. Estos mecanismos de transformación pueden utilizarse para abordar distintos requerimientos, como la resolución de discrepancias entre los formatos de datos intercambiados (XML, CSV, etc.).

[3] Tomado de https://www.agesic.gub.uy/innovaportal/file/1619/1/plataforma_de_interoperabilidad.pdf [17]

- Ruteo basado en contenido. Otra funcionalidad que proveen los productos de ESB es la posibilidad de direccionar mensajes de acuerdo a su contenido. Se analiza el contenido de los mensajes y en base a este, se hace redirección a determinado servicio

2.1.4.3. Otros servicios de la PGE

Los Servicios Transversales de la PGE son actualmente el Portal y el Buscador del Estado Uruguayo, el Sistema de Expediente Electrónico y el Geoportal.
El Portal del Estado Uruguayo es uno de los principales puntos de entrada al Gobierno Electrónico, permitiendo la interacción de los ciudadanos con contenidos, servicios y trámites de interés público.
El Buscador del Estado Uruguayo tiene como objetivo instrumentar una herramienta de búsqueda orientada a las necesidades del gobierno electrónico en Uruguay. La principal ventaja del buscador, con respecto a otros como Google, es que está específicamente optimizado para realizar búsquedas de información del Estado Uruguayo.
El Sistema de Expediente Electrónico tiene como objetivo principal informatizar el manejo de expedientes a nivel del Estado uruguayo y facilitar la interoperabilidad de los mismos a través de los diferentes organismos. Permite hacer trazabilidad de los expedientes.
Por último, el Geoportal es un portal de información geográfica que permite la consulta y análisis vía Web de la información geográfica proveniente de los organismos.

2.1.4.4. Sistema de notificaciones de la PGE

El sistema de Notificaciones y comunicaciones electrónicas (e-notificaciones), es una nueva herramienta que AGESIC pone a disposición del Estado de forma de lograr un vínculo eficiente entre los organismos y el ciudadano [18]. Permite enviar notificaciones de forma electrónica a empresas, personas y organismos en general optimizando tiempo y evitando costos a los destinatarios.
El objetivo de e-notificaciones es que los organismos de la Administración Central utilicen medios electrónicos en sus comunicaciones con los interesados, siempre que estos las hayan solicitado o consentido previamente.
En 2013 se implantaron pilotos del sistema de Notificaciones en la URCDP (Unidad de Protección y Control de Datos Personales), en la URSEA (Unidad Reguladora de Servicios de Energía y Agua), en El Correo Uruguayo y en el MIEM (Ministerio de Industria, Energía y Minería).
Para ser usuario de este sistema, el usuario debe concurrir personalmente al organismo que deba ser notificado, a los efectos de solicitar un usuario y la creación de un domicilio electrónico. Para ser destinatario de las notificaciones, el usuario debe suscribir su domicilio electrónico a los organismos de los cuales desea vincularse. Las notificaciones son puntuales (point-to-point) y dirigidas a un solo usuario suscripto a los canales de su interés.

2.2. Modelos de datos espaciales

Diferentes modelos de datos pueden ser utilizados para representar información geográfica. Se verán algunos modelos adoptados como estándares. Se destacan el modelo vectorial y el modelo ráster entre los modelos de la representación geográfica [19].

2.2.1. SFA

Simple Feature Access o SFA, también denominado ISO 19125, es el estándar propuesto por el OGC (Open Geospatial Consortium) para describir geometrías y sus operaciones utilizando notación UML [10]. Este puede constituir un modelo de representación vectorial de la información geográfica de cualquier sistema GIS. También se describen los atributos de las geometrías, se definen parámetros de las operaciones entre ellas (distancia, intersección, etc.), la semántica de dichas operaciones y las primitivas que debe soportar cada geometría.

2.2.1.1. Arquitectura de SFA

Las geometrías del SFA se representan en el diagrama de clases UML de la Figura 2.4, donde allí se manifiestan las relaciones de asociación, herencia y composición entre ellas.

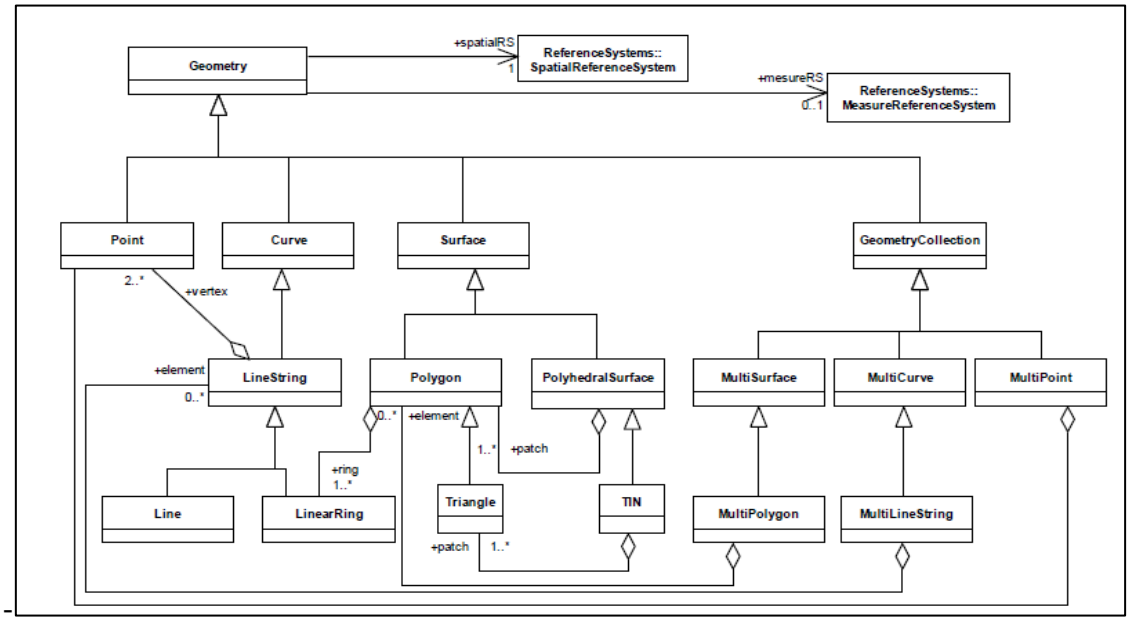

Figura 2.4: Diagrama de clases UML de SFA[4]

La clase abstracta Geometry es la raíz de toda la jerarquía. Las subclases de Geometry definidas en el estándar son instanciables y cuyos objetos geométricos están en los espacios de coordenadas R^2, R^3 y R^4. Los objetos en R^2 son parejas de valores reales (x,y); los de R^3 son tuplas de valores reales (x,y,z) o (x,y,m); los objetos de R^4 tienen puntos con coordenadas (x,y,z,m). La coordenada z representa altitud o elevación y no siempre es necesaria. La coordenada m permite a una aplicación asociar determinada medida a cierto punto, usualmente para medir distancias desde un punto de referencia. Por ejemplo, si una ruta se representa con una polilínea y las estaciones de servicio con puntos en la ruta, el valor m puede utilizarse para indicar la distancia desde el origen de la ruta a cada estación.

[4] Tomado de *Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture* [10]

2.2.1.2. Clase Geometry

En la Figura 2.5, se pueden visualizar todas las operaciones genéricas de la clase Geometry. Por un lado, cuenta con métodos que permiten obtener información básica de la geometría (tipo de geometría, identificador, sistema de referencia, exportaciones a diferentes formatos, etc.). Luego, se listan operaciones que permiten evaluar la relación espacial entre objetos de dicha geometría. Las principales son:

- Equals. Evalúa TRUE si dos geometrías del mismo tipo tienen exactamente los mismos conjuntos de puntos. Se debe cumplir el siguiente enunciado:

  $a,b \in Geometry,\ a.Equals(b) \Leftrightarrow a \subseteq b \wedge b \subseteq a$

- Disjoint. Evalúa TRUE si una geometría es espacialmente disjunta a otra geometría (intersección vacía). Se debe cumplir el siguiente enunciado:

  $a,b \in Geometry,\ a.Disjoint(b) \Leftrightarrow a \cap b = \emptyset$

- Intersects. Evalúa TRUE si una geometría intersecta a otra geometría, o sea, si la intersección no es vacía. Es el resultado opuesto a Disjoint. Cumple el siguiente enunciado:

  $a,b \in Geometry,\ a.Intersects(b) \Leftrightarrow !\ a.Disjoint(b)$

- Touches. Evalúa TRUE si ninguno de los puntos comunes pertenecientes a ambas geometrías interseca el interior de alguna de las geometrías. Al menos, una geometría debería ser una línea, polígono, multilínea o multipolígono. Cumple el siguiente enunciado:

  $a,b \in Geometry,\ a.Touch(b) \Leftrightarrow (I(a) \cap I(b) = \emptyset) \wedge (a \cap b \neq \emptyset)$

- Crosses. Evalúa TRUE si el resultado de la intersección de dos geometrías, es otra geometría cuyo número de dimensiones es una menos que el máximo de los números de dimensiones de las geometrías origen y, además, los puntos de la intersección pertenecen al interior de ambas geometrías. Cumple con el siguiente enunciado:

  $a,b \in Geometry,\ a.Cross(b) \Leftrightarrow$

  $[I(a) \cap I(b) \neq \emptyset \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)]$

- Within. Evalúa TRUE si, dadas dos geometrías a y b origen, a esta completamente contenida en b. Esto es equivalente a decir que la intersección de ambas es a y que el interior de a y el exterior de b son disjuntos. Cumple con el siguiente enunciado:

  $a,b \in Geometry,\ a.Within(b) \Leftrightarrow (a \cap b = a) \wedge (I(a) \cap E(b) = \emptyset)$

- Contains. Evalúa TRUE si, dadas dos geometrías origen, la primera contiene completamente a la segunda. Es el resultado opuesto a Within. Cumple con el siguiente enunciado:

  $a,b \in Geometry,\ a.Contains(b) \Leftrightarrow b.Within(a)$

- LocateAlong. Recibe como parámetro un valor numérico m y devuelve una geometría que se constituye a distancia m.

Por otro lado, hay operaciones que brindan análisis geométrico. Dada una geometría de origen O, las principales son:

- Distance(g:Geometry):Double. Devuelve la distancia mínima entre dos puntos pertenecientes a las dos geometrías, calculada en el sistema de referencia espacial de O.

- Buffer (m:Double):Geometry. Devuelve una geometría cuyo conjunto de puntos están a m de la geometría origen. El cálculo se realiza en el sistema de referencia espacial de O.

- ConvexHull(): Geometry. Devuelve una geometría que resulta de la envoltura convexa de O.
- Intersection(g:Geometry):Geometry. Devuelve la geometría intersección de g y O, o sea, el conjunto de puntos que pertenecen a g y que también pertenecen a O
- Union(g:Geometry): Geometry. Devuelve la geometría unión de g con O, o sea, el conjunto de puntos que pertenecen a g o a O.
- Difference(g:Geometry):Geometry. Devuelve la geometría conformada por el conjunto de puntos que pertenecen a O y no pertenecen a g.
- SymDifference(g:Geometry):Geometry. Devuelve la geometría conformada por el conjunto de puntos que pertenecen a Union(g) y no pertenecen a Intersection(g). También se le conoce como unión disyuntiva, según [20].

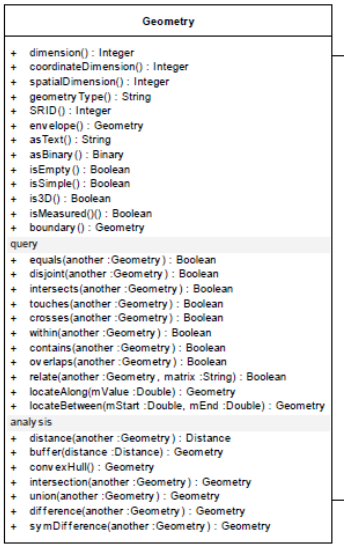

Figura 2.5: Operaciones de la clase Geometry[5]

Respecto a la arquitectura de la Figura 2.4, se describirán un subconjunto de las geometrías allí presentes y que serán de utilidad para la propuesta de esta tesis.

2.2.1.3. Geometrías

Se analizaran con más detalle: punto (Point), línea (Line) y polilínea (LineString), y polígono (Polygon).

Punto

El punto es una geometría de dimensión 0 y representa una localización en el espacio de coordenadas. Tiene coordenadas x e y, y, según ya se mencionó previamente, puede incluir coordenadas z o m.
Una variación más general del punto es el multipunto, que se considera una geometría que cuenta con un conjunto de puntos. También es de dimensión 0, ya que los puntos no están ni ordenados ni conectados entre sí.

Línea

[5] Tomado de *Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture* [10]

En SFA, una línea se define como una curva (clase Curve) con interpolación lineal entre dos puntos. Cada par de puntos consecutivos en una línea define un segmento. Una polilínea es una geometría con varios segmentos (o varias líneas), por lo que posee más de dos puntos. Las operaciones básicas de LineString (y las heredadas de Curve) se muestran en la Figura 2.6.

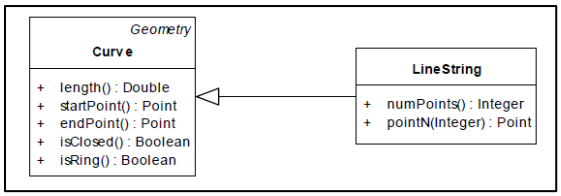

Figura 2.6: Clase LineString de SFA[6]

La primitiva isClosed devuelve TRUE si la polilínea es cerrada; isRing devuelve TRUE si la polilínea es un anillo (no hay intersección de segmentos en la polilínea).

Polígono

Un polígono es una superficie (la clase Polygon hereda de la clase Surface en la Figura 2.19) definida por un límite exterior y 0 o más limites interiores.

Existen 5 reglas que definen polígonos validos:

1. Los polígonos son cerrados
2. La frontera de un polígono está definida únicamente por un conjunto de anillos que definen los límites de su exterior e interior.
3. Los anillos que conforman la frontera del polígono no se pueden cruzar y, en caso de intersección, solo pueden hacerlo en forma tangencial.
4. Un polígono no puede tener líneas discontinuas o "cortadas".
5. El interior de un polígono debe conformar una única componente del polígono.

En la Figura 2.7 muestra algunos ejemplos de polígonos con diferente cantidad de anillos: 1 anillo en (a), 2 en (b) y 3 en (c).

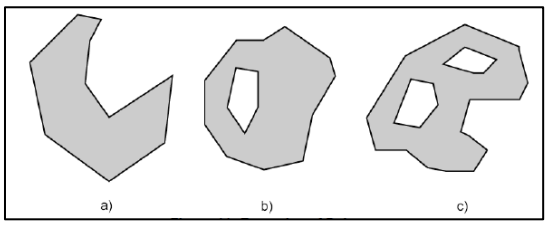

Figura 2.7: Ejemplos de polígonos válidos[7]

En la Figura 2.8, se muestran ejemplos de objetos que no son polígonos válidos. En (a), no se cumple la regla 3, ya que el anillo interior interseca el anillo exterior en 2 puntos. En (b), tampoco se cumple la regla 3, ya que el anillo interior interseca el anillo exterior en una línea. En (c), no se cumple la regla 2, ya que la frontera del polígono no se conforma por un anillo ya que existe un segmento adicional. En (d), no se cumple la regla 5, ya que el interior del polígono se divide en dos componentes.

[6,7] Tomado de *Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture* [10]

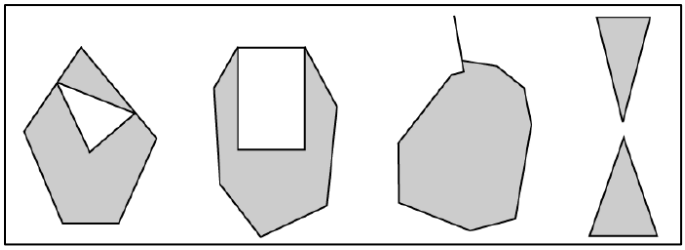

Figura 2.8: Ejemplos de polígonos inválidos[8]

Por último, en la Figura 2.9, se muestran las operaciones asociadas a un polígono. ExteriorRing devuelve la polilínea que conforma el anillo externo del polígono; NumInteriorRing devuelve el número de anillos interiores del polígono.

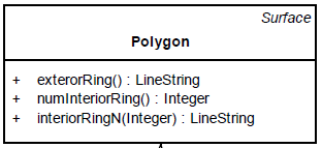

Figura 2.9: Clase Polígono[9]

2.2.1.4. SFS

El OGC ofrece un estándar que provee un esquema SQL para almacenamiento, obtención, consulta y actualización de los elementos geoespaciales definidos en el SFA [21]. Además, ofrece la especificación de una implementación de las geometrías detalladas en el SFA utilizando elementos de una base de datos: tablas, columnas, relaciones entre tablas, etc. El estándar tiene diferentes propósitos:

- Definir la arquitectura de un framework para la representación de elementos geométricos.
- Definir un conjunto de definiciones para términos utilizados en este framework.
- Definir un perfil geométrico para la definición de atributos geométricos en el framework.
- Describir un conjunto de tipos geométricos SQL y un conjunto de funciones sobre esos tipos.

La Figura 2.10 ilustra el esquema SQL para soportar las geometrías del SFA basado en tipos predefinidos. En dicho esquema se tienen: 1) La tabla GEOMETRY_COLUMNS describe las tablas con las diferentes geometrías y las propiedades de la clase Geometry, 2) la tabla SPATIAL_REF_SYS describe el sistema de coordenadas y transformaciones para Geometry, 3) la tabla FEATURE_TABLE almacena una colección de geometrías. Cada fila representa cada tipo de geometría y las columnas los atributos de ese tipo. Y 4) La tabla GEOMETRY_TABLE almacena los objetos geométricos.

El estándar constituye una convención para: 1) Nombres y definiciones de geometrías para los tipos SQL de la clase Geometry y 2) Nombres, firmas y definiciones para los procedimientos y funciones SQL para la clase Geometry, en conformidad con la especificación de SFA.

[8,9] Tomado de *Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture* [10]

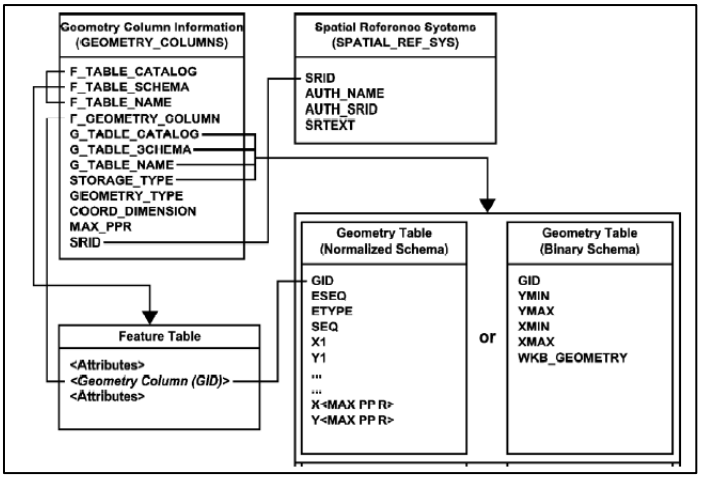

Figura 2.10: Esquema SQL para SFA[10]

Dependiendo del tipo de almacenamiento especificado en la tabla GEOMETRY_COLUMNS, un objeto geométrico es almacenado como un arreglo de coordenadas o como un único valor binario. En el primer caso, se utilizan tipos numéricos predefinidos de SQL y estos valores se obtienen de la tabla de geometrías hasta que el objeto haya sido construido. En el segundo caso, el objeto se obtiene utilizando la representación Well-Known Binary Representation (WKB) [22].

2.2.2. Modelo ráster

El ráster es su forma más simple, consta de una matriz de celdas (o pixeles) organizados en una cuadricula, en la que cada celda contiene un valor que representa información, como ser temperatura [23]. Los rásteres son fotografías aéreas digitales, imágenes de satélite, imágenes digitales o mapas escaneados.
Si bien la estructura de datos ráster es simple, es excepcionalmente útil para una amplia variedad de aplicaciones. En un GIS, los usos de los datos ráster se pueden dividir en cuatro categorías principales:

- Rásteres en forma de mapas base. Un uso común de los datos ráster en un GIS es en forma de visualización de fondo para otras capas de entidades. Se utilizan sobre las capas de un mapa para representar objetos o información, alineándose espacialmente.
- Rásteres en forma de mapas de superficie. Los rásteres son apropiados para representar datos que cambian continuamente en una superficie. Ofrecen un método efectivo para mostrar la continuidad de un fenómeno. También proporcionan una representación de superficies con espacios regulares. Los valores de elevación que se miden desde la superficie de la Tierra son la aplicación más común de los mapas de superficie.
- Rásteres en forma de mapas temáticos. Los rásteres que representan datos temáticos se pueden derivar al analizar otros datos. Una aplicación de análisis común consiste en clasificar una imagen de satélite por categorías de cobertura de suelo.
- Rásteres en forma de atributos de una entidad. Los rásteres utilizados como atributos de una entidad pueden ser fotografías digitales, documentos escaneados o dibujos escaneados

[10] Tomado de *Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture* [10]

relacionados con un objeto o ubicación geográfica. Una capa de parcela podría tener documentos legales escaneados que identifiquen la transacción más reciente de dicha parcela.

Las ventajas de almacenar los datos en forma de ráster son las siguientes:

- Estructura de datos simple. El ráster es una matriz de celdas con valores que representan una coordenada y que, en ocasiones, se encuentra vinculada a una tabla de atributos.
- Capacidad de representar superficies continuas.
- Capacidad de almacenar puntos, líneas, polígonos y superficies de manera uniforme.

La elección de un modelo basado en ráster o un modelo vectorial dependerá de si las propiedades topológicas son importantes para el análisis. Sí es así, el modelo de datos vectorial es la mejor opción, pero su estructura de datos, aunque muy precisa, es mucho más compleja y puede ralentizar el proceso. Por ello, si el análisis que interesa no requiere acudir a propiedades topológicas, es mucho más rápido, sencillo y eficaz el uso del formato ráster. También es más fácil inclinarse por una estructura de datos vectorial cuando hay que reflejar más de un atributo en un mismo espacio. Usar un formato ráster obligaría a crear una capa distinta para cada atributo.

2.3. Location Based Services

2.3.1. Surgimiento y definición

Los servicios basados en localización (Location-Based Services o LBS) se definen como servicios de información accesibles desde dispositivos móviles a través de la red y que hacen uso de la localización del dispositivo para otorgar cierta funcionalidad [24]. El OGC define LBS como un servicio wireless que utiliza información geográfica para brindar información a usuarios móviles, en base a la posición del terminal.

LBS emerge en la conjunción de tres tecnologías: NITCS (New Information and Communication Technologies) como la rama de sistemas de telecomunicación móvil y dispositivos móviles; Internet y GIS que opera sobre bases de datos espaciales. Los sistemas GIS han sido desarrollados durante las últimas décadas sobre la base de aplicaciones de datos geográficos profesionales, dirigidas a usuarios avezados en el área. No obstante, los LBS son desarrollados para dispositivos con bajos niveles de recursos y usuarios no profesionales.

LBS requiere de diversos elementos para su utilización:

- dispositivos móviles, que permiten al usuario solicitar la información.
- comunicación por red móvil, para transferir datos del dispositivo al proveedor del servicio.
- componente de posicionamiento, utilizado para suministrar la posición actual del usuario al servicio. Existen varios métodos para determinar la posición: red de telefonía celular o GPS (Global Positioning System). Si la posición no puede ser determinada en forma automática, el usuario la puede suministrar en forma manual.
- proveedor del servicio, el cual expone servicios a ser consumidos y realiza el procesamiento de los pedidos del usuario.
- proveedores de datos y de contenido, que almacenan información de carácter geográfico y que son consumidos por el proveedor del servicio.

2.3.2. Clasificación

Existen innumerables LBSs en la actualidad creados para diferentes propósitos. Ellos se clasifican, básicamente, en pull services o push services [25] [26]:

- Pull services son aquellos que entregan información a partir de un pedido específico que realiza el usuario. Pueden clasificarse como servicios funcionales y servicios de información. Los primeros permiten brindar al usuario un servicio o producto específico en base a la invocación (ejemplo: pedir un taxi o ambulancia); los segundos son solamente para consulta informativa (ejemplo: buscar un restaurante cercano).
- Push services son aquellos que envían información al usuario sin una solicitud previa, como respuesta a ciertos eventos o para notificaciones. En esta categoría, se pueden destacar aquellos servicios que se basan en notificaciones de acuerdo a la ubicación. Las notificaciones se pueden realizar en base a cierto evento (el usuario ingresa a cierta zona o al fin de un timer). Un ejemplo de push service es aquel que envía mensajes de publicidad cuando el usuario ingresa en un centro comercial.

En el trabajo [27], se ofrece otro criterio de clasificación de mayor granularidad, en función de las funciones básicas del middleware que soporta LBS. Dicho criterio se constituye de:

- LBS Reactivo o Proactivo. Un LBS es reactivo si la información basada en localización es entregada bajo un pedido del usuario. Si es proactivo, la entrega se produce en respuesta a cierto evento en determinado lugar y el usuario es automáticamente notificado. Este criterio es análogo a la clasificación pull y push.
- LBS con Referenciación Propia o Cruzada. Se distingue si la posición del usuario es procesada por el mismo (referenciación propia) o por otros usuarios (referenciación cruzada).
- LBS Centralizado o Peer-to-Peer. Los primeros son provistos y gestionados por un servidor centralizado quien provee la información de localización, mientras que en el segundo los datos de posición son intercambiados entre los usuarios sin actores intermedios.
- LBS Al Aire Libre o Interior. Los LBS exteriores abarcan áreas muy grandes y hacen de uso de satélites (GPS) o tecnologías de posicionamiento celular (Cell-ID). Los LBS interiores asisten al usuario en el interior de edificaciones y utilizan tecnologías de posicionamiento local.

La primera generación de LBSs fue reactiva con referenciación propia y al aire libre. Los servicios más avanzados y de última generación suelen ser reactivos, con referenciación cruzada y al aire libre o interiores. La tendencia más grande en la actualidad es hacia los LBS proactivos y con referenciación cruzada [28] [29] [30].

2.3.3. Usos de los LBS

Los servicios basados en localización pueden ser utilizados en muchos contextos y para diferentes propósitos. En particular, la mayor parte de los servicios son invocados mientras el usuario se mueve con el terminal móvil y, por lo tanto, realiza diferentes acciones mientras va trasladándose. Las acciones más básicas pueden ser:

- Localización: el usuario desea saber dónde se encuentra con respecto a otra persona u otro lugar.
- Búsqueda: el usuario desea buscar personas, objetos o eventos.
- Navegación: el usuario solicita una ruta desde su posición actual al objeto de búsqueda deseado.
- Identificación: el usuario desea conocer ciertas propiedades de una localización específica (horario de atención de un local, películas estrenadas en determinado cine, etc.)
- Chequeo: el usuario pretende conocer eventos circundantes a cierta localización.

Las aplicaciones para LBS pueden clasificarse según el tipo de uso, el tipo de información manipulada, el propósito de su utilización, el rol del usuario que la utiliza, entre otros criterios. La Figura 2.11 representa las categorías más comunes de LBSs según sus aplicaciones. Entre ellas, se tienen:

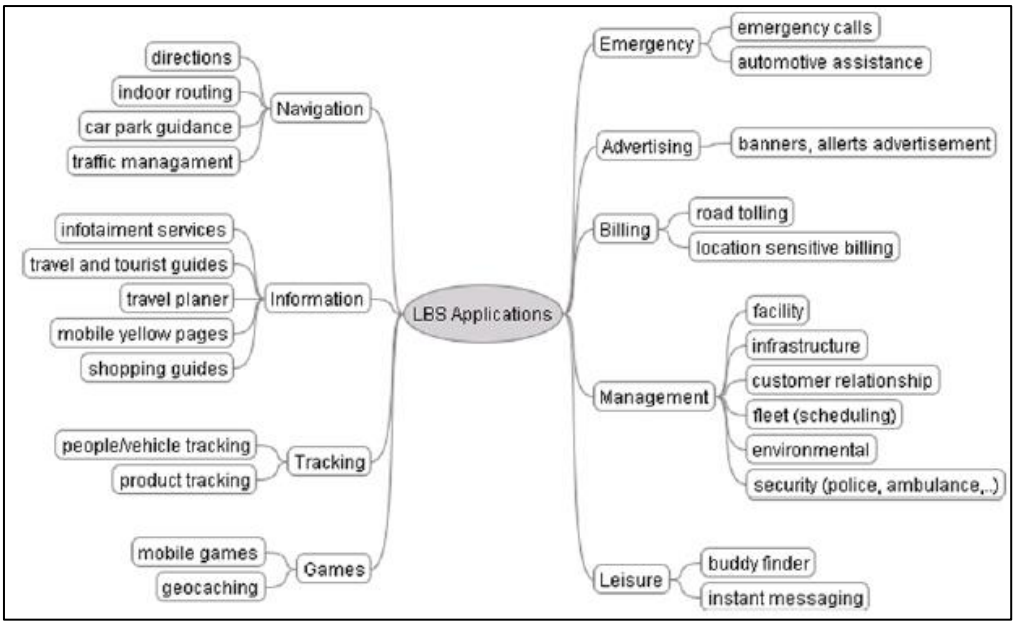

Figura 2.11: Clasificación de aplicaciones LBS[11]

- Servicios de emergencia. La aplicación LBS asiste en localizar a un individuo que puede estar inconsciente o incapacitado para suministrar su ubicación exacta debido a una situación de emergencia (daño, ataque criminal, etc.). La localización es automáticamente transferida a algún servicio de emergencia para que este pueda despachar el personal médico necesario hacia ese lugar.
- Servicios de navegación. Surgen de la necesidad de obtener direcciones desde la localización actual de un móvil para arribar a cierto destino.
- Servicios de información. Refieren a la distribución de la información basada en localización del dispositivo móvil, tiempo y comportamiento del usuario.
- Servicios de rastreo. Utilizados para rastrear y mantener registro del movimiento de un móvil, de forma de verificar cierta condición (ejemplo: enviar paquetes a ciertos puntos en tiempo pautado).

2.3.4. Contexto

Los LBS deben reaccionar al contexto en el cual están siendo utilizados, a los efectos de poder adaptar los contenidos de las aplicaciones adecuadamente. Existen diferentes tipos de contexto dependientes de la localización, tiempo y acciones del usuario, los cuales apuntan a responder dónde está el usuario, cuándo utiliza el servicio y para qué lo utiliza.
El contexto se define como cualquier información que puede ser utilizada para caracterizar el estado o situación de una entidad. La entidad puede ser una persona, lugar u objeto considerado relevante para la interacción entre un usuario y una aplicación [31]. Existen 9 tipos de elementos que influyen sobre el contexto:

[11] Tomado de *CartouCHe1 - Lecture Notes on LBS - Foundations of Location Based Service* [93]

- Identidad del usuario. Para un LBS, es importante conocer datos del usuario que lo utiliza, tales como edad, genero, preferencias, contactos, etc.
- Localización. Es el elemento fundamental que forma parte del contexto. Puede ser absoluta, utilizando un sistema de coordenadas, o relativa, por ejemplo una habitación dentro de un edificio.
- Tiempo. Utilizado para determinar si un evento es válido o no dependiendo de la variación de tiempo desde que fue creado.
- Orientación. Permite determinar en qué dirección el usuario se está moviendo, de forma de que el servicio alcance a identificar qué tiene delante, detrás o en sus aledaños.
- Historial de navegación. El usuario puede consultar los lugares en los que ha estado o qué cosas ha visto o hecho. Se puede utilizar para brindarle sugerencias o darle orientación en base a la información recabada previamente.
- Propósito de uso. Definido por las actividades, objetivos, tareas y roles de los usuarios. El conjunto de ellos puede establecer tipos de información y de presentación (mapas, texto, etc.).
- Situación social. Caracterizado por proximidad a otros usuarios, tareas en común, etc.
- Ambiente cercano. Las condiciones ambientales del lugar donde reside el usuario pueden ser una variable de contexto (iluminación, nivel de ruido, etc.)
- Propiedades del sistema. Refiere a la infraestructura de hardware que el usuario está empleando: tipo de dispositivo, tipo y calidad de la conexión a Internet, etc.

En base a estos parámetros, las aplicaciones que utilizan los LBSs pueden cambiar su comportamiento. Esta adaptación se puede lograr a diferentes niveles:

1. Información. Se modifica el contenido de la información (cambio de nivel de detalle, filtrados, etc.)
2. Tecnología. La información es codificada de acuerdo a las características del dispositivo.
3. Interfaz de usuario. Por ejemplo, reorientar un mapa mientras el usuario se mueve.
4. Presentación. Se adapta la visualización de la información (destacar unos ítems con diferente color y tamaño respecto a otros de acuerdo a relevancia).

2.3.5. Arquitectura general de un LBS

Un LBS debe contar con un modelo que permita integrar las aplicaciones con algún GIS y el sistema de posicionamiento [25]. La Figura 2.12 muestra el modelo compuesto por tres capas: posicionamiento, middleware y aplicación.

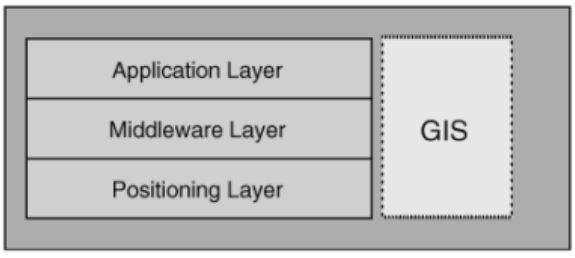

Figura 2.12: Modelo de LBS [12]

La capa de posicionamiento es responsable del cálculo de la posición del dispositivo móvil. Eso se lleva a cabo mediante algún sistema de posicionamiento (Cell-ID, GPS, etc.) y datos geoespaciales que residen en algún GIS. Mientras que el primero determina la posición y la devuelve en términos

[12] Tomado de *Location-based services* [25]

de bajo nivel, el segundo la transforma a información geográfica más adecuada (latitud y longitud). Este resultado se devuelve a la capa de aplicación o a la de middleware.
La capa de aplicación aglomera todos los servicios que solicitan los datos de localización. Con el surgimiento explosivo de aplicaciones orientadas a LBS, se ha incluido una capa de middleware entre la de posicionamiento y de aplicación. Esto es debido a que la capa de posicionamiento suma mucha complejidad en las operaciones, lo cual dificulta el desarrollo e integración de nuevos servicios. El middleware contribuye en encapsular dicha complejidad, lo cual permite reducir los costos y los tiempos de integración y mejorar el mantenimiento de los servicios. La Figura 2.13 ilustra este concepto.

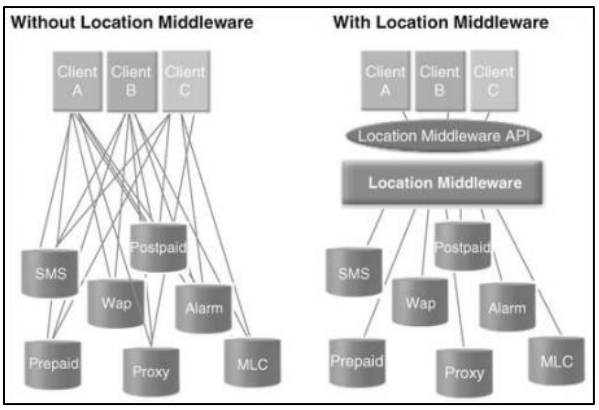

Figura 2.13: Rol que juega la capa de middleware en LBS [13]

El middleware debe poder mitigar los efectos de una serie de aspectos inherentes a LBS:

- Espacio. El middleware debe procesar la información de la posición de los dispositivos fijos y móviles y asociarlos a ubicaciones en mapas físicos o lógicos.
- Temporalidad. La información de localización incluye una dimensión temporal que debe ser incluida en las consultas de datos.
- Inexactitud, Imprecisión e Incertidumbre. El middleware debe resolverlos ya que las tecnologías de posicionamiento suelen tener algún margen de error.
- Grandes volúmenes de datos. LBS debe manipular grandes volúmenes de datos. Por ende, la escalabilidad es un aspecto de vital importancia.
- Consultas continuas. El middleware debe ser eficiente en términos de procesamiento ya que se requiere un gran número de consultas en lapsos de tiempo cortos.

2.4. Message Oriented Middleware

2.4.1. Motivación y definición

Durante las últimas décadas, los sistemas de software tuvieron que adaptarse a los cambios producidos por la evolución de las organizaciones y las tecnologías del mundo. Los sistemas comenzaron a operar en ambientes más complejos, con múltiples lenguajes de programación, hardware, sistemas operativos, y con requerimientos de alta disponibilidad, seguridad y brindando QoS (Quality-Of-Service), con lo cual los mecanismos de integración tradicionales comenzaron a no ser apropiados, ya que la lógica de integración se implementaba en las aplicaciones. Por otra parte,

[13] Tomado de *Location-based services* [25]

estos mecanismos no resultaban flexibles para integrar nuevos sistemas, ya que para llevar a cabo la integración era necesario: crear nuevos componentes en los sistemas de software que la implementen; o escalar y extender los componentes existentes para resolver la comunicación de esos sistemas con nuevas interfaces. Ambas estrategias llevaban a cambios profundos y costosos en los sistemas. Por ende, la mejor forma de minimizar el impacto en los sistemas se resolvió a través del middleware.

Middleware es una clase de tecnologías de software diseñada para lidiar con la complejidad y heterogeneidad asociada a los sistemas distribuidos. Es definido como una capa de software sobre los sistemas operativos y bajo las aplicaciones, y a las cuales les provee una API para operar en sistemas distribuidos. Permite reducir en gran medida la complejidad de las aplicaciones ya que se puede delegar al middleware gran parte de las operaciones y funciones vinculadas a la interoperabilidad [32]. Por otra parte, permite integrar componentes con diversos atributos: heterogeneidad, interoperabilidad, seguridad y dependabilidad [3].
Los frameworks de middleware buscan encapsular la heterogeneidad de las redes y el hardware entre los sistemas, abstrayéndose de sistemas operativos y lenguajes de programación. Algunos como CORBA también encapsulan diferentes implementaciones de middleware de proveedores. Los frameworks permiten lograr transparencia respecto a las siguientes dimensiones: localización, concurrencia, replicación, fallas y movilidad [32].
Existen diferentes categorías de middleware:

- Tuplas distribuidas. Permite almacenar información en estructura de tuplas en bases de datos relacionales distribuidas.
- Objetos de middleware distribuidos. Provee abstracción de un objeto remoto cuyos métodos pueden ser invocados de la misma forma que si estuviera en el espacio de direccionamiento del invocador. Brinda los beneficios del paradigma de orientación a objetos: herencia, polimorfismo y encapsulamiento.
- RPC (Remote Procedure Call). Permite invocar un procedimiento cuyo código y ejecución se distribuye a través de la red [33]. Es el mecanismo de integración de aplicaciones ampliamente utilizado por sistemas legados y que provee comunicación bajo modalidad cliente-servidor. El cliente realiza una llamada a un procedimiento remoto que envía un pedido al servidor, el cual ejecuta una cierta rutina en la que se procesa el pedido, y finalmente se retorna una respuesta al cliente. RPC sigue el modelo de comunicación sincrónica, ya que requiere una conexión y el cliente se bloquea. Este modelo se representa en la Figura 2.14.

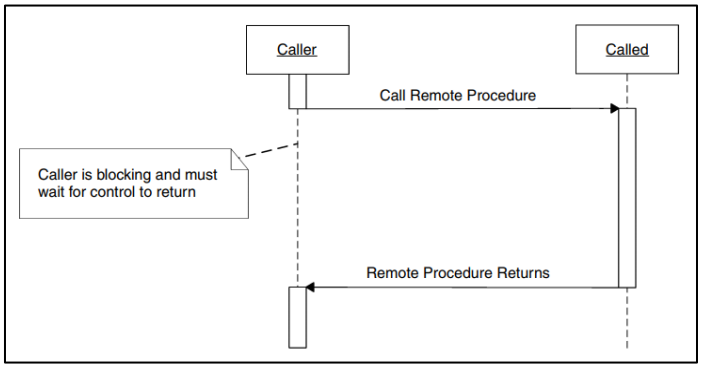

Figura 2.14: Modelo de comunicación sincrónica [14]

[14] Tomada de *Middleware* [32]

El cliente transfiere temporalmente el control al servidor que ejecuta el procedimiento remoto y espera hasta que aquel retorne el resultado de la ejecución. El cliente continúa la ejecución después que el servidor retorna la respuesta del procedimiento.

- Message-oriented Middleware. Provee una abstracción de cola de mensajes que puede ser accedida y referenciada remotamente desde cualquier aplicación.

Un middleware orientado a mensajería (Message-Oriented-Middleware o MOM) se puede definir como un tipo de middleware que soporta envío y recepción de paquetes de datos de forma sincrónica o asincrónica entre diferentes participantes [5]. Un MOM soporta los modelos de comunicación sincrónica (de RPC) y asincrónica. En el último caso, el cliente no debe esperar a que esté finalizado el pedido. El cliente puede continuar la ejecución sin tener en cuenta el estado del servicio o procedimiento invocado. Puede eventualmente obtener el mensaje de respuesta en cualquier momento de la ejecución (y si está listo el pedido) mediante una operación pull o push. En la Figura 2.15 se ilustra el modelo descrito.

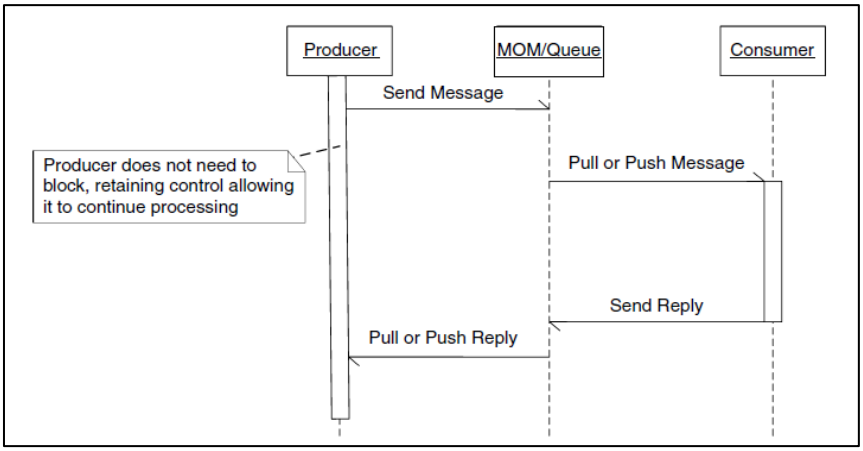

Figura 2.15: Modelo de comunicación asincrónica [15]

En rasgos generales, provee diversas características [34]:

- Transformación de formatos de datos
- Bajo acoplamiento de las aplicaciones. La capacidad de conectar aplicaciones sin tener que adaptar los sistemas origen ni destino.
- Procesamiento en paralelo de los mensajes
- Soporte para niveles de prioridad de los mensajes.
- Alta cohesión. Es consecuencia del bajo acoplamiento.
- Confiabilidad
- Alta disponibilidad. MOM no requiere que haya disponibilidad en todos los subsistemas al mismo tiempo. La falla en uno de ellos no hace que se caiga todo el sistema.
- Alta velocidad

2.4.2. Servicios básicos que brinda un MOM

Los servicios básicos que debe brindar un MOM son los siguientes [5]:

[15] Tomada de *Middleware* [32]

- Filtrado de mensajes. Permite a la aplicación o subsistema receptor seleccionar los mensajes que le llegan a su bandeja. El filtrado puede realizarse a diferentes niveles. Se utilizan expresiones booleanas para determinar si el mensaje es deseado por el cliente. En general, el filtro de basa en parejas clave-valor en el mensaje o en el cuerpo del mensaje. Se clasifican diferentes tipos de filtros:
    - Basado en canal. Los clientes se suscriben a los grupos de interés donde recibir mensajes.
    - Basado en asunto. Los mensajes son etiquetados mediante tags que describen el tema de contenido. El cliente especifica los tags de mensajes que desea consumir.
    - Basado en contenido. Se filtra según el contenido del mensaje utilizando algún lenguaje de consulta, por ejemplo SQL-2.
- Transaccionalidad. Una transacción es un mecanismo que permite agrupar varias tareas en una unidad de trabajo. Cuando un cliente desea enviar o recuperar mensajes en una transacción, se realizan múltiples tareas de mensajes (ejemplo: enviar 1..N mensajes) de forma de que todas sean exitosas o todas fallen. La aplicación emisora o receptora del mensaje tiene la posibilidad de confirmar o abortar la transacción. Los mensajes entregados al receptor en una transacción no son reenviados hasta que el cliente emisor haya confirmado la transacción.
- Garantía de entrega de mensajes. La plataforma MOM debe almacenar temporalmente los mensajes para garantizar la entrega. El MOM envía el mensaje al consumidor del mismo y espera por la confirmación de entrega. Si la aplicación no reconoce el mensaje en cierto periodo de tiempo, el servidor vuelve a reintentar el envió.

La garantía de entrega se manifiesta en el concepto Store-and-Forward, detallado en la sección 2.4.3. En la Figura 2.16, se muestra la noción básica.

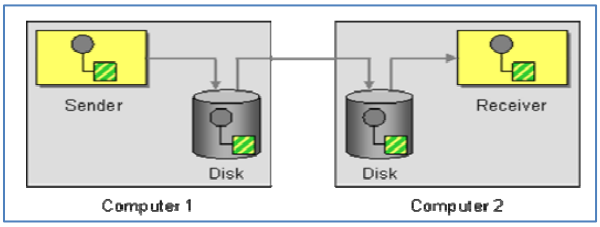

Figura 2.16: Modelo Store-and-Forward [16]

El MOM utiliza algún mecanismo de almacenamiento de datos persistente de los mensajes (base de datos, archivos, etc.), de forma de evitar que se pierdan en caso de falla del sistema. Cada aplicación en el sistema de mensajería realiza la persistencia local de los mensajes. Cuando el emisor envía un mensaje, la operación de envío no está finalizada hasta que se haya guardado exitosamente el mensaje. El mensaje no es eliminado hasta que es exitosamente direccionado y almacenado en la computadora destino del mensaje o en el siguiente paso.
La persistencia temporal incrementa la confiabilidad a expensas de la performance. Si el sistema es tolerable a la perdida de mensajes, se puede prescindir de realizar persistencia.

- Formatos de mensajes. Dependiendo de la implementación de MOM, diversos formatos pueden ser intercambiados. Algunos son: XML, HashMaps, Stream, etc.
- Balanceo de carga. Permite distribuir dinámicamente el trabajo de carga del MOM en un conjunto de servidores. El balanceo se logra de dos formas: "push" y "pull". En el modelo "push", se utiliza un algoritmo para balancear el trabajo sobre los servidores. Numerosos algoritmos existen para predecir el servidor menos "cargado" y pushear el pedido hacia él. En el modelo "pull", la carga es balanceada utilizando una cola point-to-point donde los pedidos se

[16] Tomada de *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* [3]

almacenan. De esta forma, los servidores van tomando los mensajes de la cola en la medida que se liberen de trabajo.

- Clustering. Permite replicar el estado del servidor del MOM en múltiples servidores, de forma de recuperar el estado en caso de falla. Esto permite que un cliente pueda ser migrado a un servidor alternativo si pierde conexión con el servidor con el que se está comunicando.

2.4.3. Mensajería

Un MOM se puede ver como un amplio sistema de mensajería con múltiples capacidades. Un sistema de mensajería es necesario para "llevar" mensajes de una computadora a la otra que se conectan en una red poco confiable. Además, la aplicación receptora no siempre está disponible para recibir el mensaje; el sistema de mensajería tiene la responsabilidad de realizar la entrega hasta que la misma este operable.

En un MOM, un mensaje se transmite en cinco pasos:

1. Crear. El emisor crea un mensaje y coloca en él los datos que desea transmitir.
2. Enviar. El emisor coloca el mensaje en el canal.
3. Entregar. El sistema de mensajería transporta el mensaje logrando que esté disponible para el receptor.
4. Recibir. El receptor lee el mensaje desde el canal.
5. Procesar. El receptor extrae los datos del mensaje.

En la Figura 2.17, se ilustran los cinco pasos descritos. Se puede ver que en ambas computadoras se hace un almacenamiento temporal del mensaje, en el primer caso antes de enviar y en el segundo antes de recibir.

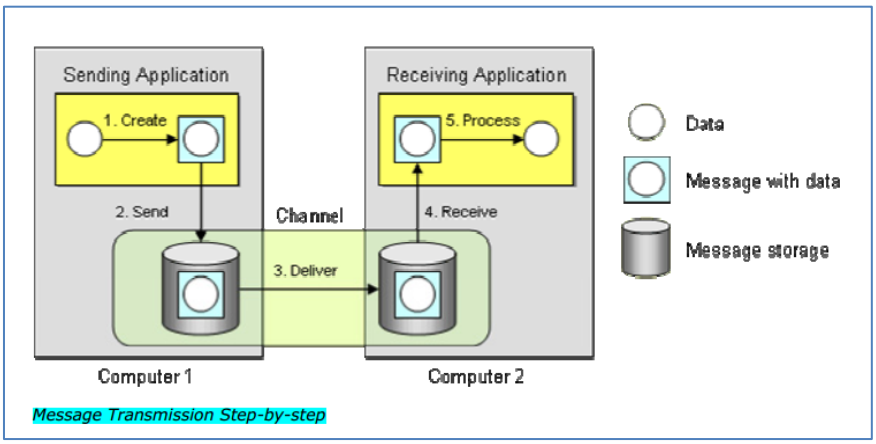

Figura 2.17: Pasos de transmisión de un mensaje en un MOM[17]

De la figura, también se desprenden dos conceptos importantes:

- Send-and-forget. En el paso 2, la aplicación emisora envía el mensaje a un canal. Una vez que el envío es completado, la aplicación puede realizar otras tareas mientras que el MOM transmite el mensaje en paralelo. La aplicación confiara en que el receptor recibirá el mensaje, y puede no esperar hasta que esto ocurra.
- Store-and-forward. En el paso 2, cuando la aplicación emisora envía el mensaje al canal, el sistema de mensajería lo almacena en la computadora del emisor, en memoria o disco. En el

[17] Tomada de *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* [3]

paso 3, el sistema de mensajería entrega el mensaje direccionándolo hasta la computadora del receptor donde se persiste allí una vez más.

Usar mensajería acarrea una serie de beneficios:

- Comunicación remota. Permite la comunicación de aplicaciones conectadas a través de la red.
- Integración de plataformas y lenguajes. Las aplicaciones que se comunican pueden estar desarrolladas en diferentes lenguajes, tecnologías y plataformas. El MOM se abstrae de estas diferencias, operando solo con formatos de mensajes y protocolos de comunicación específicos.
- Comunicación asincrónica. Utilizando un enfoque send-and-forget. El emisor del mensaje no tiene que esperar que el sistema de mensajería entregue el mensaje; solo debe esperar que el mensaje sea almacenado en el canal de mensajes del sistema de mensajería.
- Control sobre el ritmo de procesamiento. Un problema que se da en RPC es que muchas llamadas sobre un solo receptor pueden llevar a sobrecargarlo. Por esta razón, la comunicación asincrónica permite al receptor controlar la frecuencia a la cual recibir los pedidos, de forma de no quedarse sin capacidad de procesamiento.
- Comunicación confiable. La mensajería es más confiable que el RPC ya que utiliza el enfoque store-and-forward para transmitir los mensajes.
- Mediación. El sistema de mensajería actúa como mediador. Si una aplicación se desconecta de otras, solo debe reconectarse al MOM, no al resto de las aplicaciones; esto es posible gracias a que un MOM provee un pool de conexiones desde y hacia las aplicaciones.
- Gestión de threads. En el modelo de comunicación asincrónica, la aplicación emisora del mensaje no se bloquea hasta recibir la respuesta; en lugar de hacer esto, puede ejecutar otras tareas y dejar pendiente una función de callback en un thread que procese el mensaje recibido y notifique cuando el mensaje haya sido recibido. En este contexto, múltiples threads habilitan que muchos sub-procesos ejecuten concurrentemente, lo cual permite mejorar la performance. Estos subprocesos pueden ejecutar en cualquier orden, lo cual es responsabilidad de la aplicación ordenar y combinar los resultados.

2.4.4. Componentes de un MOM

Múltiples y diversos tipos de componentes integran una solución de MOM

Mensajes

Un mensaje se define como una unidad discreta de datos que puede ser transmitida por un canal. Para transmitir un mensaje, la aplicación debe fragmentarlo en uno o más paquetes de datos. La aplicación receptora del mensaje debe extraer los datos del mensaje para su posterior procesamiento.
Un mensaje cuenta con la siguiente estructura:

- Header. Contiene información sobre el mensaje: identificador, direcciones de origen y destino y propiedades (Timestamp, TTL, sequenceNumber, correlationID, etc.).
- Cuerpo. Contiene los datos transmitidos, generalmente ignorados por el sistema de mensajería.

En un MOM viven diferentes tipos de mensajes:

- Mensaje de comando: invoca un procedimiento en otra aplicación.
- Mensaje de documento: pasa un conjunto de datos a otra aplicación.
- Mensaje de evento: notifica a otra aplicación de un cambio en la aplicación emisora.
- Mensaje de pedido-respuesta: se utiliza para enviar una respuesta a un mensaje previo.
- Mensaje de secuencia: indica que la información que se envía está contenida en una secuencia de mensajes.

Canales

Las aplicaciones transmiten información a través de un canal de mensajes, un conducto virtual que conecta al emisor con el receptor. Generalmente son estáticos, por lo que resultan del acuerdo de distintas aplicaciones que se conectan al MOM. Un MOM no viene preconfigurado con todos los canales que sus aplicaciones van a utilizar; los desarrolladores que diseñan las aplicaciones determinan qué canales son necesarios para la comunicación entre ellas. Luego, el administrador del MOM debe configurarlo con los canales apropiados.
Los canales son direcciones lógicas en el sistema de mensajería; como están implementados depende del producto que implementa el MOM. Dependiendo del tipo de canal, puede requerir memoria o disco para alojar los mensajes.
En soluciones de integración de aplicaciones basadas en MOM, se suele utilizar el estilo arquitectónico Pipes and Filters, que divide una tarea de procesamiento compleja en múltiples pasos de procesamiento llamados Filters y que son interconectados mediante canales o Pipes. En la Figura 2.18 se representa dicha arquitectura con un ejemplo sencillo. El sistema emisor genera una orden de compra que esta encriptada y firmada electrónicamente. Para poder ser procesada por el sistema destino, debe atravesar a un conjunto de filtros. En el primer paso, se aplica un algoritmo de desencriptado para obtener la orden en claro; en el segundo paso, se valida que la orden provenga del emisor correspondiente autenticando la firma; en el tercer paso, se eliminan las órdenes de compra duplicadas.

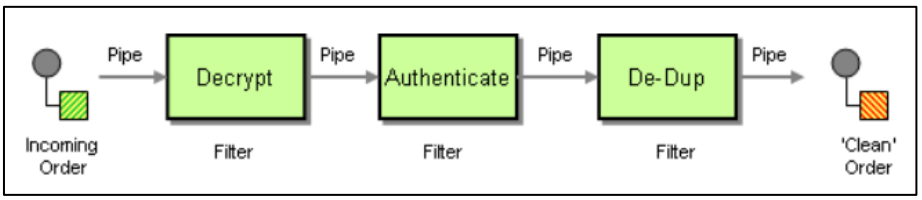

Figura 2.18: Ejemplo de envío de orden de compra. Se utiliza Pipes and Filters.[18]

Cada filtro expone una interfaz muy simple: recibe mensajes del canal entrante, procesa el mensaje y publica el mensaje en el canal saliente. El canal conecta el filtro con el siguiente. Debido a que los filtros utilizan la misma interfaz, es posible organizar los filtros en diferentes composiciones. Esto es posible gracias a que los filtros están desacoplados entre sí.

Routing

En una gran plataforma con numerosas aplicaciones y canales que las conectan, un mensaje puede fluir por varios canales hasta alcanzar su destino final. La ruta que construye el mensaje puede llegar a ser muy compleja y, por el contrario, ignorada por el emisor del mensaje. Los enrutadores en el MOM son los encargados de definir la ruta que seguirá el mensaje. El router no modifica el contenido de los mensajes; solo obtiene y lee el destino del mensaje.
La principal ventaja de utilizar enrutadores es que el criterio de decisión del destino de un mensaje se localiza en el router. Esto significa que si se definieran nuevos tipos de mensajes o si se agregan nuevos componentes o cambiaran las reglas de enrutamiento, solo habría que modificar la lógica del router de mensajería y sin impactar cambios en las aplicaciones.

[18] Tomada de *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* [3]

Un router debe conocer todo el conjunto posible de destinos de un mensaje (de forma de enviarlo al canal correcto). Dado que la lista de posibles destinos puede variar con cierta frecuencia, el mantenimiento del router puede tornarse un cuello de botella. En este escenario, es preferible delegar a los receptores la tarea de aceptar o rechazar los mensajes que llegan a su canal, aliviando o eliminando la tarea de enrutamiento que hace el MOM.
Algunos sistemas de mensajería deben decodificar el mensaje de un canal antes de insertarlo en otro canal, lo cual puede causar un importante overhead. Proveyendo múltiples routers en paralelo y hardware adicional, este efecto puede ser minimizado. Como resultado, el throughput (número de mensajes procesados por unidad de tiempo) puede no ser afectado, pero la latencia (tiempo del mensaje mientras viaja por el MOM) se incrementará.
Existen diversos tipos de routers:

- Basados en Contenido. Determinan el destino del mensaje basado en sus propiedades (tipo del mensaje, valores en los campos del mensaje, etc.)
- Basados en Contexto. Determinan el destino del mensaje en base a las condiciones del entorno. Estos usualmente son utilizados para realizar balanceo de carga, pruebas o reacción ante fallas. Por ejemplo, si un componente de procesamiento falla, este router puede redirigir el mensaje a otro componente del mismo tipo para que lo reemplace y pueda realizar la tarea del anterior.
- Routers stateless. Sólo inspeccionan el mensaje para tomar la decisión de ruteo.
- Routers stateful. Basan la decisión de ruteo en el contenido de los mensajes previos.

Traductores

Diversas aplicaciones pueden representar la misma información con diferente estructura y formato. El emisor del mensaje le da cierto formato y el receptor lo puede esperar en un formato diferente. Por ello, los traductores se encargan de convertir el formato original al de destino.
La traducción de mensaje puede realizarse a diferentes niveles. Por ejemplo, los datos de una entidad pueden compartir el mismo nombre y el mismo tipo de dato, pero pueden tener diferentes representaciones (archivo XML o CSV). O pueden tener formato XML y usar diferente conjunto de tags. Por esta razón, existen cuatro niveles (o capas) grandes de transformación:

- Capa de transporte. Provee el transporte de datos entre diferentes sistemas a través de la red.
- Capa de representación de datos (o de “sintaxis”). Define como se representan los datos y establece como transformar complejas estructuras de datos a secuencias de caracteres. Formatos comunes son XML, campos de largo fijo (registros EDI) o formatos propietarios.
- Capa de tipos de datos. Define los tipos de datos que conforman el dominio de una aplicación. Ejemplo: una fecha con tiempo se representa simplemente con un string, o se representa con una estructura compleja donde se almacena la fecha, la hora del día y la zona horaria.
- Capa de estructuras de datos. Describe las entidades lógicas que se almacenan a nivel de capa de aplicación, tales como Cliente, Cuenta o Dirección; y las relaciones entre ellas.

Las transformaciones en un sistema de mensajería pueden ocurrir a cualquiera de estos niveles. En el ejemplo de la Figura 2.19, se puede apreciar cómo se va modificando una orden de compra hasta que llega al sistema de gestión de órdenes. La traducción va de formato EDI (campos de largo fijo) a XML, en un nivel de representación de datos. Luego, la traducción se realiza desde XML a una orden de compra esperada por el sistema receptor, en un nivel de tipos de datos.

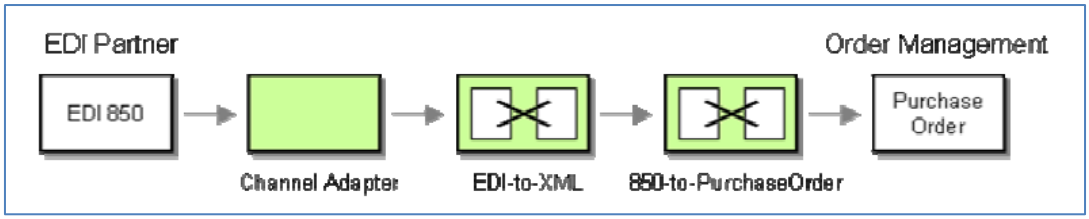

Figura 2.19: Ejemplo de transformaciones de formatos de datos[19]

Endpoints

En general, las aplicaciones no cuentan con interfaces para interactuar con el sistema de mensajería. Por esta razón, debe existir una capa de código entre la aplicación y el sistema de mensajería que interconecte a ambos. Esta capa recibe el nombre de Endpoint y permite a la aplicación enviar y recibir mensajes. Ejemplos: JMS para Java, MSMQ para .NET.
Un endpoint sólo se puede utilizar para conectarse a un canal y, por ende, una aplicación debe usar múltiples endpoints para conectarse a múltiples canales. O también puede ocurrir que use más de un endpoint para interfacear con un único canal en caso de soportar múltiples threads concurrentes.
La forma de recepción de mensajes puede variar según el tipo de Endpoint: Polling Consumer (recibe por consulta constante) o Event-Driven Consumer (recibe en base a un evento). Los mensajes almacenados en un canal se pueden aceptar o rechazar utilizando un Selective Endpoint. Cada tipo de Endpoint debe implementar la interfaz de software adecuada.

Gestión de mensajes

Es un componente especial encargado de proveer herramientas para llevar a cabo las tareas de monitoreo, administración, testeo y análisis de los mensajes en tránsito del sistema de mensajería.

2.4.5. Patrones de mensajes

Los patrones de mensajería identifican flujos de mensajes comunes que se pueden utilizar en soluciones de mensajería y que están basados en los componentes del MOM.
En este documento, solo se hará foco en los patrones de mensajes relacionados a canales, que son los de interés para la propuesta de la solución. Ellos son el patrón Point-to-Point y el PS.

2.4.5.1. Point-to-Point

En la Figura 2.20 se esquematiza este patrón.

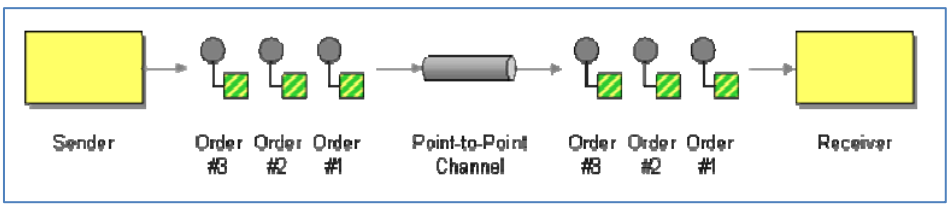

Figura 2.20: Patrón Point-to-Point [20]

[19] Tomado de *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* [3]
[20] Tomado de *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* [3]

Un canal Point-to-Point permite la comunicación punto a punto y asegura que si se envía un mensaje a un canal, solo un consumidor podrá recibir el mensaje. Si el canal tiene múltiples receptores, solo uno de ellos podrá consumir el mensaje y sin necesidad de coordinarse entre ellos para obtenerlo. En este caso, se habla de Competing Consumers ya que "compiten" por la recepción del mensaje. Otra característica es que se mantiene el orden de entrega, tal como se ilustra en la figura.

2.4.5.2. Publish-Subscribe

En la Figura 2.21 se esquematiza este patrón.

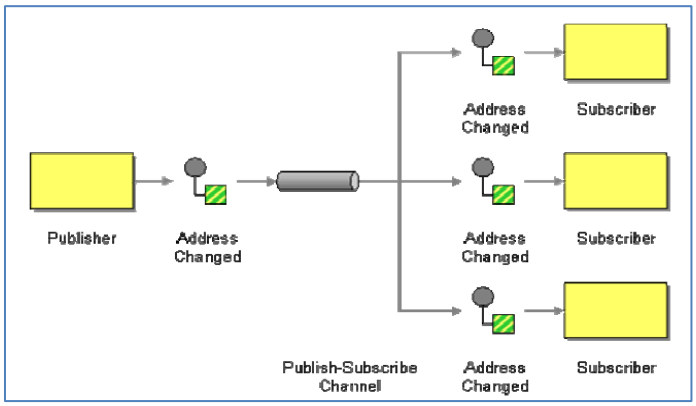

Figura 2.21: Patrón PS [21]

El patrón de diseño observer permite a un suscriptor (observer) registrarse con y recibir notificaciones de un proveedor específico (subject) [35]. Siempre y cuando ocurra un cambio de estado, un evento o se dé una condición, el proveedor automáticamente notifica a todos los suscriptores. Permite desacoplar a los suscriptores del proveedor. El patrón PS extiende el patrón Observer agregando la noción de canal de eventos para enviar las notificaciones.
El canal PS funciona de la siguiente forma: tiene un canal de entrada que se conecta a múltiples canales de salida, uno por cada subscriptor. Cuando un evento se publica en el canal, este envía una copia del mensaje a cada uno de los canales de salida. Cada canal de salida tiene un solo suscriptor, el cual está habilitado a consumir el mensaje exactamente una vez. De esta forma, el suscriptor solo obtiene una copia del mensaje. En consecuencia, el mensaje no se considera consumido hasta que todos los suscriptores hayan sido notificados.

Se pueden encontrar tres diferentes variantes de PS [4]:

- Topic-based Publish Subscribe. Está basado en topics o subjects y esta implementado en múltiples soluciones empresariales. Extiende la noción de canales, usados para comunicar pares a través de los endpoints. Los participantes pueden publicar eventos y subscribirse a múltiples topics identificados mediante palabras clave. Los tópicos se pueden ver como grupos de comunicación, por lo tanto, la subscripción a un tópico T se puede ver como la pertenencia al grupo T. La publicación de un evento en T resulta en la diseminación del evento entre todos los miembros del grupo T. En la práctica, topic-based PS mapea cada topic a varios canales.
  Por otra parte, se han realizado mejoras en este esquema basándose en jerarquías de topics, organizándolos según relaciones de contenido de los eventos. Por ende, una subscripción a

[21] Tomado de *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* [3]

cierto nodo en la jerarquía implica hacer subscripciones a todos los subtópicos de ese nodo. Los motores que implementan esta noción utilizan direcciones de URL para nombrar los tópicos y la subscripción múltiple se hace mediante wildcards que machean a los topics por su nombre.

- Content-Based Publish-Subscribe. Constituye una mejora del esquema de tópicos ya que se basa en el contenido de los eventos. Los eventos no se clasifican bajo los nombres de los tópicos sino a través de las propiedades que contengan los eventos. Dichas propiedades pueden ser atributos de estructuras de datos o metadatos asociados a los eventos.
  Los consumidores de subscriben a ciertos eventos especificando filtros mediante un lenguaje de subscripción. Los filtros definen restricciones en la forma de parejas clave-valor que mapeen a las propiedades y operadores de comparación básicos. Una operación de subscripción, a diferencia de la modalidad Topic-based, recibe un argumento adicional indicando el patrón de subscripción. Hay diversas formas para representar tales patrones:
  - String. Se provee un string que representa algún lenguaje de consulta, tales como SQL, XPath o algún lenguaje propietario.
  - Objeto template. Al subscribirse, el participante provee un objeto t, que indica que está interesado en cualquier evento que conforme el tipo t y cuyos atributos macheen los correspondientes a t, excepto aquellos que estén indefinidos (null).
  - Código ejecutable. Los subscriptores proveen una rutina para filtrar los eventos en tiempo de ejecución. No es un patrón muy utilizado ya que es costosa la optimización del filtro y debe aplicarse a cada evento en forma secuencial.
- Type-based Publish-Subscribe. Extiende la noción de topic-based para hacer filtrado de eventos en base al tipo de evento. Dichos tipos pueden ser organizados de forma jerárquica (herencia).

2.4.6. Web Services

Los web services se pueden definir como aplicaciones de negocio auto-contenidas que exponen interfaces de comunicación bien definidas y basadas en estándares a través de Internet [33]. El W3C lo define como un sistema de software diseñado para soportar interoperabilidad entre dos computadoras a través de internet [36]. Tiene una interfaz descrita en lenguaje WSDL [37] (Web Service Descriptor Language) y las maquinas que se comunican lo hacen a partir de mensajes SOAP (Simple Object Access Protocol) típicamente utilizando una serialización en formato XML sobre transporte HTTP en conjunción con otros estándares web. En un WSDL, se describe la interfaz del web service en lenguaje XML. Establece el protocolo de comunicación, formatos y tipos de mensajes de los servicios y sus operaciones en forma abstracta.
La segunda definición apunta a web services de tipo SOAP, donde los mensajes intercambiados por las aplicaciones son de formato SOAP. SOAP es un formato XML con dos partes bien definidas: encabezado y cuerpo. En el encabezado (Soap:Header) se agregan los atributos que representan la metadata del mensaje, como por ejemplo información de control, de contexto o relacionada al procesamiento o parseo del mensaje. Dentro del cuerpo (Soap:Body), van los datos propios del mensaje y se encapsulan los mensajes de solicitud y los de respuesta del servicio web. También se incluye una seccion para agregar posibles errores que ocurran en el procesamiento del mensaje (Soap:Faults). En la Figura 2.22 se muestra un ejemplo de un mensaje SOAP de respuesta con datos de una reserva de viajes.

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary
    xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid-morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging
   xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

Figura 2.22: Mensaje SOAP de ejemplo[22]

Existe un segundo tipo de web services creados más recientes: los web services RESTFul. Estos web services se adhieren al estilo arquitectónico RESTFul que define un conjunto de restricciones [38]. Estas restricciones comprenden bajo acoplamiento cliente-servidor, no existencia de sesiones de usuario ni de estado entre varias solicitudes http, caching a nivel de red y uso de una única interfaz para simplificar las interacciones entre cliente-servidor, lo cual implica tener una única dirección para cada recurso referenciable.

Los servicios REST utilizan el protocolo HTTP y permiten acceder a recursos identificados por una URL específica. A su vez, las diferentes acciones sobre esos recursos se mapean con los métodos HTTP: get, post, delete, head, put. La ventaja principal sobre los servicios SOAP es que reduce drásticamente el tamaño de los mensajes intercambiados, ya que, como ya se vio, SOAP debe respetar cierta estructura (encabezado, cuerpo, fallos, etc). Por otra parte, el acceso a un servicio REST tiene menos overhead que a uno SOAP, ya que en SOAP el cliente debe consumir primero un archivo WSDL con la especificación del contrato que define la interfaz del servicio web. En contrapartida, el descubrimiento de nuevos servicios en REST no está respaldado por un estándar de registro único (UDDI) para REST.

2.4.6.1. Web Services avanzados

Los estándares básicos no abordan problemáticas comunes en contextos empresariales, como por ejemplo, confiabilidad, seguridad, transaccionalidad, etc. Por ello, surgen un conjunto de nuevas especificaciones, conocidas como Web services avanzados o WS-*, destinadas a intentar resolver algunos de los problemas transversales a todas las organizaciones. Cada estándar intenta resolver una problemática específica y están orientados a la composición entre ellos. Algunos de estos estándares son: WS-Eventing, WS-Notification, WS-Security, WS-Addressing, WS-ReliableMessaging, WS-Reliability, WS-Trust, etc. En esta sección se hará foco en algunos de ellos.

[22] Tomado de https://www.w3.org/TR/2000/NOTE-SOAP-20000508/

2.4.6.1.1. WS-Addressing

WS-Addressing provee un mecanismo para referenciar web services y direccionar mensajes. Utiliza referencias endpoint (EPR) SOAP y headers en los mensajes para transmitir los mensajes. Los endpoints son utilizados en la ruta del mensaje para proveer las direcciones a otros endpoints, desde el WS origen al destino. Incluye dirección, parámetros de referencia y metadata.
Los headers definen información que incluye las direcciones de los endpoints de origen y destino, así como la identidad del mensaje [39] [40]. El mensaje SOAP puede contener diversas propiedades de direccionamiento:

- ReplyTo. Dirección de la EPR al que se debe enviar la respuesta de la solicitud.
- FaultTo. Especifica la dirección de un EPR al que se debe invocar cuando ocurra una situacion anómala y no se pueda terminar de procesar la solicitud.
- MessageID. String que identifica de forma univoca el mensaje.
- RelatesTo. String que permite correlacionar mensajes

En la Figura 2.23, se muestra un ejemplo de forwarding entre WSs. El cliente hace un pedido a un servidor y el servidor redirige el pedido hacia otro servidor para su procesamiento.

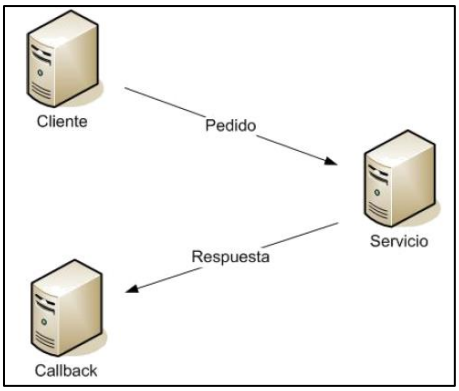

Figura 2.23:Forwarding en WS-Addressing

2.4.6.1.2. WS-ReliableMessaging

WS-ReliableMessaging (WS-RM) provee un protocolo interoperable para mensajería confiable de aplicaciones en un ambiente distribuido, garantizando la entrega de los mensajes para web services. Es responsabilidad del emisor y receptor del mensaje asegurar la entrega del mensaje o devolver un error. Hay varias formas de asegurar la entrega del mensaje [41]:

- Como máximo una vez. Los mensajes son entregados como máximo una vez sin duplicación o un error será generado en al menos un endpoint. Es posible que algunos mensajes en una secuencia no puedan ser entregados.
- Al menos una vez. Cada mensaje será entregado o un error será producido en al menos un endpoint. Algunos mensajes puede ser entregados más de una vez, no evitando duplicados.
- Exactamente una vez. Cada mensaje será entregado sin duplicación o un error será producido en al menos un endpoint. Es la conjunción de las dos formas de entrega anteriores.
- En orden. Los mensajes son entregados en el orden en que fueron enviados. La entrega en orden se utiliza con alguno de los esquemas anteriores. Se utilizan números de secuencia para garantizar la entrega en orden.

En WS-RM, el emisor envía un mensaje y lo transmite una o más veces. Luego de que el receptor recibe el mensaje, envía un reconocimiento al emisor y lo entrega a la aplicación. El envío de mensajes se traduce en el establecimiento de un protocolo entre el emisor y el receptor que implica

negociar las condiciones del protocolo, crear identificadores y números de secuencia, asignarlos a los mensajes, enviar los mensajes, crear y devolver los acks al emisor y finalizar la secuencia de mensajes. En la Figura 2.24 se ilustra un escenario típico de flujo de mensajes entre cliente y servidor en un escenario WS-RM.

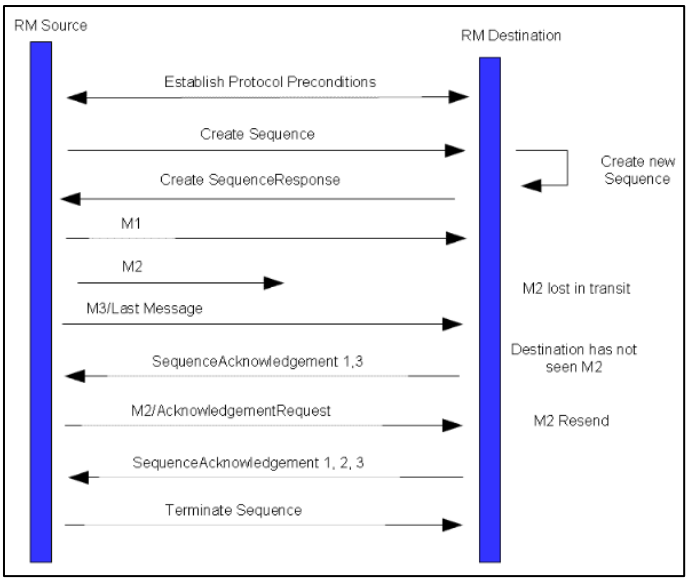

Figura 2.24: Flujo de mensajes en WS-RM [23]

En el estándar se define un binding del formato SOAP para WS-RM, definiendo propiedades con semántica apropiada en los headers de los mensajes.

2.4.6.1.3. WS-Security

WS-Security es una especificación de la OASIS [42], cuyo objetivo es dar un conjunto de herramientas para implementar seguridad en Web Services. Consiste en un conjunto de extensiones de SOAP que pueden utilizarse para construir Web Services seguros. Esta provee tres mecanismos principales:

- Posibilidad de enviar tokens de seguridad como parte de un mensaje
- Integridad de datos
- Confidencialidad de datos

A continuación se aclaran algunos términos utilizados, tomando las definiciones de [42]:

- Token de seguridad: Un token de seguridad representa una colección de (uno o más) claims.
- Claim: Un claimes una declaración hecha por una entidad, (por ejemplo: nombre, identidad, claves, grupos, privilegios, etc.).
- Token de seguridad firmado: Un token de seguridad firmado es un token de seguridad que contiene afirmaciones de una autoridad específica, y es firmado criptográficamente por ésta (ejemplo: un certificado X.509 o un ticket Kerberos).

2.4.6.1.4. WS-Trust

WS-Trust es una especificación de la OASIS [43] que describe una interfaz estándar para un servicio de seguridad que puede emitir, validar, renovar y cancelar tokens de seguridad. Se describe un

[23] Tomada de http://specs.xmlsoap.org/ws/2005/02/rm/WS-RMPolicy.pdf [41]

servicio llamado Security Token Service (STS) el cual brinda security tokens, los cuales podrían ser por ejemplo SAML assertions, tickets Kerberos, o claves. Para que pueda adaptarse a un conjunto variado de situaciones, se plantea una interfaz genérica de pedidos (RequestSecurityToken) y respuestas (RequestSecurityTokenResponse).
En un pedido, el solicitante indicará el tipo de security token que necesita, y también deberá dar pruebas de su identidad (por ejemplo agregando en el pedido un header WS-Security para autenticarse), ante lo cual el STS responderá con el token solicitado o responderá con un SOAP Fault en caso de no autenticar al solicitante.
Algunos escenarios habituales en el acceso a un STS son [44]:

1. El cliente solicita un token al STS para acceder a un servicio.
2. Un intermediario invoca al STS para realizar autenticación/autorización.

2.4.6.1.5. WS-Notification

WS-Notification es una familia de estándares avanzados de WS* creada por OSASIS en 2006 para especificar el patrón content-based PS para Web Services. Estandariza los conceptos, terminología e intercambios de mensajes utilizados en el patrón. En definitiva, permite diseminar información a un conjunto de web services sin tener previo conocimiento de la existencia de estos.
WS-Notification se compone de tres estándares principales [45] [46] [47]:

- WS-BaseNotification, define los roles de las aplicaciones productoras y consumidoras de los mensajes y el filtrado de mensajes a partir desde el contenido utilizando de selectores.
- WS-BrokeredNotification, extiende BaseNotification y define el rol del broker.
- WS-Topics define una sintaxis específica para topics que pueden ser implementados desde BaseNotification o BrokeredNotification

WS-BaseNotification brinda una especificación de interfaces de web services para el NotificationProducer (productor de mensajes) y NotificationConsumer (consumidor de mensajes). El NotificationProducer genera un conjunto de mensajes de notificación y aceptar pedidos de suscripción de tipo Subscribe. Cada pedido Subscribe contiene una ER (Endpoint Reference) a un NotificationConsumer y un conjunto de filtros booleanos sobre las notificaciones, incluyendo filtrado por Topics. El NotificationProducer acuerda en generar mensajes de tipo Notification. Un NotificationProducer es un web service que implementa la interfaz definida por el estándar de NotificationProducer. Un NotificationConsumer es un endpoint representado por una referencia de WS-Addressing y esta designado para recibir notificaciones. En la Figura 2.25 se puede ver un ejemplo de un mensaje SOAP de notificación de WS-BaseNotification. Se incluye el ER al NotificationProducer.

```
<s:Envelope ... >
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/Notify
    </wsa:Action>
    ...
  </s:Header>
  <s:Body>
    <wsnt:Notify>
      <wsnt:NotificationMessage>
        <wsnt:SubscriptionReference>
          <wsa:Address>
            http://www.example.org/SubscriptionManager
          </wsa:Address>
        </wsnt:SubscriptionReference>
        <wsnt:Topic Dialect=
     "http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple">
          npex:SomeTopic
        </wsnt:Topic>
        <wsnt:ProducerReference>
          <wsa:Address>
            http://www.example.org/NotificationProducer
          </wsa:Address>
        </wsnt:ProducerReference>
        <wsnt:Message>
          <npex:NotifyContent>exampleNotifyContent</npex:NotifyContent>
        </wsnt:Message>
      <wsnt:NotificationMessage>
    </wsnt:Notify>
  </s:Body>
</s:Envelope>
```

Figura 2.25: Ejemplo de mensaje SOAP-Notify[24]

WS-BrokeredNotification define una interfaz de web services para el rol NotificationBroker. Este es un intermediario entre los publicadores y los suscriptores, permitiendo desacoplar los NotificationProducers de los NotificationConsumers y brinda características de publicación a demanda y balanceo de carga. En cuanto a la publicación, se destacan dos escenarios:

1. Publicación simple. Un publicador envía un mensaje encapsulado en un mensaje Notify al NotificationBroker y el broker realiza la distribución. Se puede apreciar en la Figura 2.26.

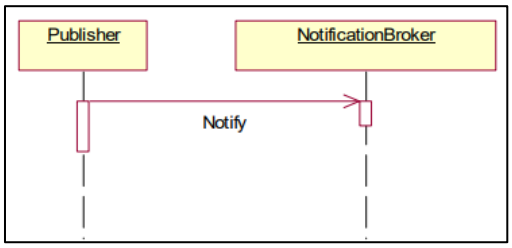

Figura 2.26: Publicación simple[25]

2. Publicación a demanda (Figura 2.27). El publicador se registra como un NotificationProducer en el Notificationbroker. El broker envía mensajes de suscripción al Producer para recibir notificaciones de este. El Producer pública utilizando mensajes Notify al broker quien se encarga de hacer la distribución en los NotificationConsumers. Este esquema se utiliza para cancelar la suscripción del broker cuando el publicador no tiene suscripciones de consumidores activas.

[24] Tomada de http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf [45]
[25] Tomada de http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf [46]

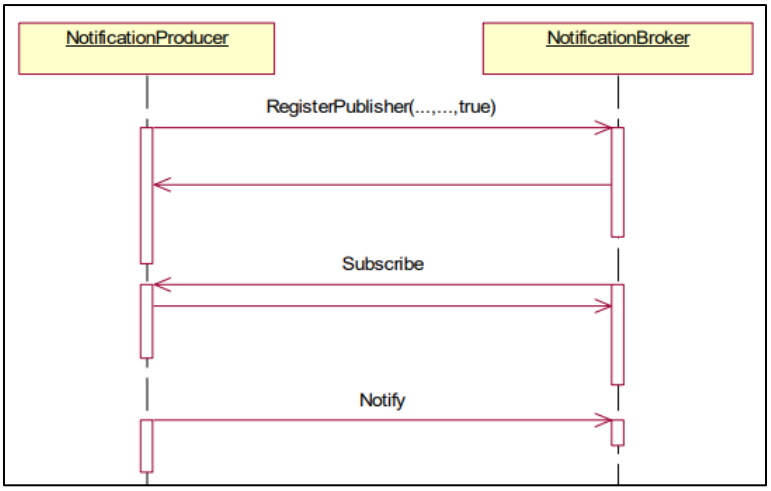

Figura 2.27: Publicación a demanda[26]

WS-Topics define un mecanismo para categorizar elementos de suscripción denominados topicos. Define 4 tipos de dialectos que pueden ser utilizados como filtros de una suscripción en los mensajes Subscribe de WS-BaseNotification. Además, define un modelo XML para describir la metadata de los tópicos.

2.4.7. JMS

JMS (Java Message Service) provee una API para Java que permite a las aplicaciones crear, enviar, recibir y leer mensajes [48]. La API provee un conjunto de interfaces que permite la comunicación por mensajes en diferentes formatos entre aplicaciones escritas en lenguaje Java.
La API de JMS minimiza el conjunto de conceptos que el desarrollador debe utilizar para productos de mensajería y brinda suficientes capacidades para soportar mensajería en aplicaciones. Favorece la portabilidad de aplicaciones JMS entre diferentes proveedores JMS. Cuenta con singulares características:

- Bajo acoplamiento entre aplicaciones
- Asincronismo. El proveedor JMS realiza la entrega de mensajes tan pronto como los clientes receptores estén disponibles; dicha entrega se realiza sin un previo pedido del receptor.
- Confiabilidad. Permite que un mensaje sea entregado solo una vez a cada receptor. El nivel de confiabilidad puede depender de los requerimientos de las aplicaciones; es el caso de aplicaciones que pueden permitir perdida de mensajes o mensajes duplicados.

JMS surgió como una especificación de Java para funcionar dentro de sistemas de MOM, por lo cual ha sido adoptado por múltiples proveedores de MOM. Un proveedor JMS debe capturar los siguientes conceptos relacionados a MOMs:

- Proveedor del sistema de mensajería que hace enrutamiento y entrega de mensajes
- Soporte para entrega confiable de mensajes.
- Implementación de patrones de mensajes point-to-point y PS.
- Entrega de mensajes en diversas formas: recepción asincrónica, mediante pushing de los mensajes a los receptores; y recepción sincrónica, por pedido del consumidor.
- Formatos comunes de mensajes como stream, byte, texto, etc.

[26] Tomada de http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf

Un sistema proveedor de JMS debe brindar un JMS provider o un conjunto de librerías que implementan las interfaces JMS con las funcionalidades del MOM, incluyendo herramientas de gestión, monitoreo y manipulación de servicios de mensajería [48]. Por otra parte, un JMS provider puede ser creado como un producto standalone o embebido dentro de un ambiente EAI (por ejemplo, JEE usa un JMS provider que implementa los message-driven beans que permite a los componentes EJB enviar y recibir mensajes de forma asincrónica).
Para enviar y recibir mensajes, un cliente JMS debe conectarse a un servidor de mensajes JMS o broker. La conexión abre un canal de comunicación dentro el cliente y el broker. Luego, el cliente debe establecer una sesión para crear, producir y consumir mensajes. Un cliente puede ser productor y consumidor de mensajes al mismo tiempo. El cliente envía un mensaje con destino a una dirección interpretada por el broker; dicha dirección es utilizada por el consumidor para obtener el mensaje.
En la Figura 2.28, se ilustran los patrones de mensajería en un MOM para JMS. El cliente A envía mensajes utilizando el patrón point-to-point a una cola de mensajes en el broker y solo un receptor podrá obtener el mensaje. El cliente B envía mensajes utilizando el patrón PS a un topic en el broker al cual se suscriben múltiples consumidores. Cada subscriptor obtiene una copia del mensaje.

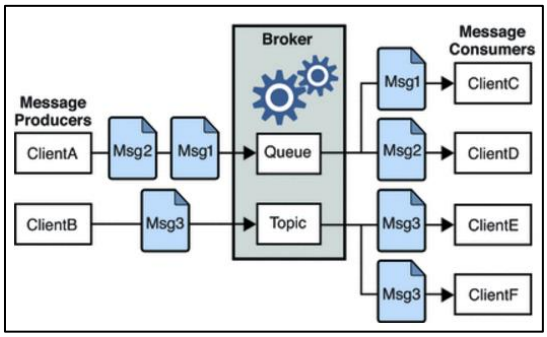

Figura 2.28: Patrones de mensajes en JMS[27]

En una plataforma JMS, existen dos tipos de objetos administrados por el proveedor para que los productores y consumidores puedan enviar y recibir mensajes: connection factory objects y destination objects. Los primeros son empleados por los clientes para generar conexiones al broker. Los segundos son utilizados para referenciar direcciones físicas en el broker, utilizando convenciones de nombramiento y dominio.
Los siguientes pasos (Figura 2.29) denotan las acciones que se deben realizar en una plataforma MOM con JMS para enviar y recibir mensajes:

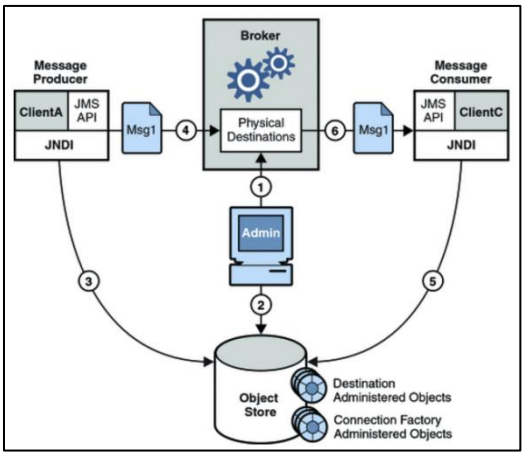

Figura 2.29: Pasos para enviar y recibir mensajes en un MOM JMS [28]

[27] Tomada de https://docs.oracle.com/cd/E19340-01/820-6424/aerar/index.html

1. El administrador del sistema crea una dirección de destino física en el broker
2. El administrador crea un destination object y le configura la dirección física de destino, cuyo nombre dependerá del tipo (cola o topic).
3. El productor del mensaje realiza una llamada JNDI para encontrar el destination object al cual apunta a la dirección de destino del mensaje.
4. El productor del mensaje envía el mensaje a dicha dirección.
5. El consumidor del mensaje utiliza una llamada JNDI para encontrar el destination object que apunta a la dirección de la cual recibir el mensaje.
6. El consumidor obtiene el mensaje utilizando dicha dirección.

2.4.8. Productos integradores de MOM

La siguiente lista representa algunos productos donde MOM es la piedra angular de la solución.

WebSphere IBM MQ.

MQ [49] provee un message-oriented middleware que permite la interacción entre aplicaciones, sistemas y servicios. En MQ, una aplicación utiliza una API simple para enviar mensajes. La aplicación receptora utiliza la misma API para obtener los datos del MOM.
IBM MQ brinda a las aplicaciones la posibilidad de comunicarse bajo la modalidad PS que ofrece flexibilidad y bajo acoplamiento. Por otra parte, constituye una plataforma para aplicaciones escritas en Java. Siendo JMS la API estándar de facto, MQ implementa como un JMS provider que puede ser utilizado aún en ambientes sujetos a otros proveedores de mensajería, con implementaciones en otras APIs. Esto se traduce en que una aplicación puede enviar y recibir mensajes de otras aplicaciones en MQ sin importar el lenguaje de programación o la plataforma de ejecución, lo cual permite a los desarrolladores abstraerse de la complejidad e interoperabilidad entre las aplicaciones.
Otro servicio con el que cuenta MQ es transaccionalidad, que realiza la gestión de transacciones (transaction manager). Se puede configurar para definir el nivel de transaccionalidad en el intercambio de mensajes, es decir, se puede especificar qué operaciones ejecutar en transacciones y cuáles no, lo cual puede favorecer menor overhead en ciertos escenarios de ejecución.

Rabbit MQ

RabbitMQ [50] provee un broker de mensajes que implementa un MOM con múltiples funcionalidades, incluyendo protocolos de intercambio de mensajes, protocolos de seguridad, tipos y formatos de mensajes, mensajería confiable, reconocimiento de mensajes, etc.
El empleo de protocolos de mensajes en las interacciones se hace factible a través del uso de plugins que se instalan en el motor. Entre los protocolos soportados por el broker, se tienen:

- AMQP (Advanced Message Queuing Protocol). Es el protocolo central del broker. AMQP es un protocolo que permite comunicar aplicaciones a través de brokers de mensajería. Es programable, ya que las aplicaciones pueden definir las entidades y los esquemas de enrutamiento de los mensajes, sin necesidad de un administrador.
- STOMP [51] (Simple Text Oriented Messaging Protocol). Stomp es un protocolo que provee un formato interoperable entre diferentes lenguajes de programación, plataformas y brokers. Los clientes pueden comunicarse mediante cualquier broker que implemente STOMP. Es un protocolo fácil de implementar. RabbitMQ lo soporta a través del uso de un plugin.

[28] Tomada de https://docs.oracle.com/cd/E19340-01/820-6424/aerar/index.html

- MQTT (MQ Telemetry Transport or Message Queue Telemetry Transport) es un estándar ISO (ISO/IEC 20922) basado en un protocolo de PS que corre sobre TCP/IP y orientado a dispositivos móviles. RabbitMQ lo soporta a través del uso de un plugin.
- HTTP. RabbitMQ transmite mensajes en HTTP en dos diferentes formas: plugin de administración para diagnóstico de incidentes y con Web sockets utilizando el protocolo STOMP para conexiones persistentes desde un browser.

Otra característica que ofrece RabbitMQ es clustering. En un clúster todas las definiciones (entidades, usuarios, bindings) son replicadas en todos los nodos. Las colas de mensajes solo residen en un solo nodo (por defecto) y son accesibles desde todos los nodos. Esto permite otorgar tolerancia a fallas, evitando perdida de mensajes.

Apache ActiveMQ

Provee un middleware para mensajería y patrones de mensajes. Soporta gran variedad de lenguajes de programación (Java, C++, C#, Ruby, etc.) y protocolos de mensajería [52]:

- OpenWire. Es un protocolo que permite comunicarse con un servidor ActiveMQ en diferentes lenguajes y plataformas, a través de la ejecución de comandos o del uso de APIs.
- STOMP
- AMQP
- MQTT

Respecto a mensajería, permite utilizar grupos de mensajes, direcciones virtuales y compuestas. Además, utiliza el estándar JMS para soporte de mensajería persistente y transaccional.

2.5. Trabajos relacionados

En el proceso de búsqueda y relevamiento del conocimiento existente en las áreas de MOM, GIS, LBS y E-Gov, se han encontrado varias publicaciones, conferencias, artículos y revistas relacionados a los conceptos mencionados. Algunas de ellas proponen algún enfoque alternativo a alguno o a una combinación de los conceptos. Otras proponen algún framework o arquitectura que vincule alguno de los conceptos. Otras brindan un marco conceptual de dichos conceptos.
La meta de dicho relevamiento consiste en encontrar trabajos relacionados que vinculen alguna combinación de los temas centrales de la propuesta (MOM, GIS, LBS, E-Gov) de forma de recabar y extraer ideas, lineamientos, conceptos, procedimientos o pautas de valor para el trabajo actual, consultando la bibliografía más actualizada posible. Se utilizó como fuentes de referencias bibliográficas los sitios en línea:

- Portal Timbo (http://www.timbo.org.uy/): de origen uruguayo, enmarcado dentro de la ANII (Agencia Nacional de Investigación e Innovación), provee referencias bibliográficas, así como links de acceso directo a publicaciones en diferentes áreas de investigación: ciencias agrarias, ciencias naturales, humanidades, ingeniería y ciencias sociales. Permite realizar búsquedas de publicaciones por varios parámetros: palabra clave, titulo o autor.
- ScienceDirect (http://www.sciencedirect.com/): de origen holandés, enmarcado dentro de la editorial Elsevier. Provee referencias y links de acceso (para descarga o compra) de publicaciones en diversas áreas: informática, energía, matemáticas, física, astronomía, medicina, etc. Permite realizar búsquedas con diferentes parámetros: palabra clave, autor, título de revista o libro, volumen, tema y página.

- DBLP (http://dblp.uni-trier.de/): servicio de origen almenan de la universidad de Trier. Provee una vasta información bibliográfica en temas relacionados a ciencias de la computación. Permite realizar búsqueda por título de publicación, autor, lugar de publicación o una combinación de ellos. Los resultados aparecen ordenados por fecha de publicación de forma descendiente.
- Google Scholar (https://scholar.google.com): servicio provisto por Google Inc. para búsqueda de publicaciones académicas en una gran variedad de temas. Los parámetros de búsqueda pueden ser nombre, autor, rango de fechas, idioma y contenido de publicación. En los resultados de búsqueda, se incluye el título de la publicación, una breve descripción, puede incluir un link en formato PDF a la publicación (si está disponible en la Web) y la cita bibliográfica en varios formatos. A su vez, se puede ordenar por fecha o por relevancia.

2.5.1. Publicaciones interesantes

La mayoría de los trabajos encontrados plantean arquitecturas y diseños de sistemas MOM que utilizan el patrón PS sobre datos geográficos. En algunos de ellos los suscriptores son dispositivos móviles y utilizan LBS para reportar su ubicación, donde se realizan notificaciones en base a la ubicación actual o en restricciones geográficas con diversas topologías.

2.5.1.1. PS sobre datos geográficos

Se presentan trabajos de extensiones de MOMs para soportar restricciones geoespaciales.
En [53]**,** se presenta un modelo Content-Based PS para soportar suscripciones geométricas. Las publicaciones son mensajes que contienen parejas clave-valor donde un atributo define un polígono donde ocurre el evento. Las subscripciones son predicados sobre esas regiones utilizando operadores sobre regiones 2D irregulares, que incluyen overlap, touch y disjoint. Por otra parte, ofrece un algoritmo de indexación para organizar restricciones espaciales (para procesamiento eficiente de las subscripciones) y un método de simplificación para eliminar restricciones redundantes en las subscripciones.
Un enfoque similar se da en [54]**,** donde se propone un framework para sistemas distribuidos que opera con un middleware PS basado en datos geoespaciales para notificaciones de eventos en tiempo real. La propuesta implementa un prototipo que transmite eventos geoespaciales relacionados a incendios, incluyendo localizaciones de las unidades de emergencia por GPS, observaciones meteorológicas, incendios forestales, etc. Utiliza un modelo de datos basado en el SFA para las publicaciones. Las subscripciones son conjuntos de predicados de atributos y predicados espaciales. Los últimos consisten en una geometría base (punto, polígono, polilínea, etc.), un operador espacial (Contain, Disjoint, etc.) y valor de buffer opcional.
El GeoMQTT [55]**,** extensión de MQTT, utiliza un esquema PS basado en colas de mensajes con marca temporal y geometrías. Los suscriptores definen sus intereses a ciertos eventos geográficos, mediante suscripción a una cola de mensajes y definiendo filtros de tipo espacial y temporal. El filtro espacial puede adquirir diferentes formatos: GeoJSON, WKT, EWKT o GML.
Por último, en [56] se propone la arquitectura de un middleware PS basado en notificaciones espaciales para ser utilizado por LBSs. Se define un modelo de eventos espaciales, un modelo de subscripciones espaciales y un modelo de notificaciones espaciales. En el modelo de subscripciones se utilizan las operaciones Within y Distance. El último se utiliza para indicar que un usuario está ingresando o abandonando cierta área.

2.5.1.2. PS sobre LBS

Estos trabajos utilizan plataformas que utilizan diversos estándares específicos para publicar eventos y recibir mensajes.
En [57], se propone un MOM en la nube que integra un LBS para transferir información geoespacial hacia dispositivos móviles. Se expone una API de servicios web que permite a los usuarios obtener los datos geoespaciales de múltiples proveedores, utilizando diversos estándares: SOAP, WSDL, WMS, WCS, WFS y WPS. Las fuentes de datos incluyen los servidores de Google [58], ArcGIS [59], NASA [60], etc. El middleware se encarga de realizar operaciones de transformación y composición de los datos de origen y los presenta en formato JSON utilizando notificaciones push. La arquitectura también cuenta con servicios de cauterización y balanceadores de carga.
Existen un par de trabajos orientados a adaptar plataformas MOMs con estándar JMS a escenarios de clientes móviles, proveyendo servicios que difieren de los MOMs tradicionales. En [61] se presenta JOMS (Java Opportunistic Message Service), un MOM que implementa Content-Based and Subject-Based PS con un modelo liviano de servidor: colas de mensajes, topics y servicio de directorio están totalmente distribuidos entre los dispositivos móviles, que colaboran para compartir datos a través de una red ad hoc. Los nodos son productores, consumidores, y a su vez brokers de mensajes. Un tercer tipo de mensajes, mensajes de anuncio, se utilizan mensajes de anuncio para informar la lista de vecinos que tiene un nodo. Los nodos realizan la tarea de macheo de publicaciones con las subscripciones y, en tal caso, hacen forwarding de los mensajes hacia los nodos consumidores.

2.5.1.3. PS para LBS adaptivo y clientes móviles

Los trabajos de esta subsección proponen sistemas para ajustar los mecanismos de PS de acuerdo a las condiciones del entorno: número y densidad de usuarios, requerimientos de aplicaciones y de dispositivos, movilidad de los usuarios, conectividad y ancho de banda, etc. En un ambiente mobile, es necesidad adaptar los parámetros de configuración de un sistema para aseguran los niveles de calidad de servicio (QoS).
En [62] que presenta un MOM basado en PS en la nube para adquirir datos desde sensores en dispositivos móviles. Tanto los publicadores como consumidores de datos son móviles que se comunican con un motor de procesamiento en la nube. Una publicación puede ser un dato crudo capturado por un sensor o un dato procesado y/o agregado. Una suscripción es una conjunción de predicados. El CPSP es responsable de la adquisición de datos externos, procesamiento y diseminación de datos hacia los consumidores utilizando PS. Una particularidad es que se introduce el concepto de broker móvil, nodo encargado de recibir los datos de los sensores conectados y de controlar el consumo de recursos (batería, ancho de banda, etc.).
En [63] se presenta un framework que permite ejecutar transiciones entre diferentes protocolos de PS en redes móviles ad hoc (MANET) basados en condiciones específicas o en eventos externos, como ser movilidad de los clientes esperada, número de usuarios, carga de aplicaciones, requerimientos de las aplicaciones, etc. El patrón de mensajería se ajusta a un Topic-Based PS de forma de lograr sencillez en los protocolos derivados. En el trabajo se desarrolla un prototipo con cambio de un esquema basado en multicast de mensajes a uno basado en brokers, donde un subconjunto de nodos son brokers y mantienen subscripciones en nombre de otros nodos.
En [28], se introduce el concepto de Location-based PS (LPS), donde los mensajes publicados son ruteados basados en contenido y en la localización de publicadores y subscriptores. El trabajo presenta un algoritmo para LPS (Adaptive LPS) cuya estrategia de diseminación de publicaciones (mensajes) y

subscripciones (consultas) se adapta en tiempo de ejecución en función de las condiciones del entorno (conectividad, procesador, batería, distancia a otros nodos, etc.).
En [64]**,** se propone un mecanismo de adaptación a QoS de forma automática en un MOM para asegurar notificaciones en tiempo real. Se basa en tres políticas: historia, límite de recursos y filtros de tiempo. La primera determina si un nodo productor (data writer) debe priorizar la publicación de mensajes a realizar otras tareas; la segunda especifica la máxima cantidad de recursos que un productor puede consumir; la tercera controla que un subscriptor reciba cierta cantidad de mensajes por intervalo de tiempo.
En el trabajo [30]**,** se describe un MOM que implementa PS admitiendo retardo en la entrega de mensajes, de forma de salvaguardar recursos en los dispositivos móviles que publican datos y realizan subscripciones. Existe garantía de entrega de mensajes utilizando mecanismos de store-and-forward entre el móvil y el broker de mensajes. Se emplean notificaciones push a los dispositivos para entregar los mensajes bajo el protocolo AMQP. En la medida que los clientes se movilizan, se van registrando con el broker más cercano disponible.

2.5.1.4. PS utilizando alguna complejidad en el matching

Estos trabajos utilizan alguna complejidad en el algoritmo de matching de PS para dispositivos móviles. La complejidad puede residir en la utilización de algún otro algoritmo eficiente y conocido o en el uso de estructuras de datos intermedias.
En [65]**,** se presenta un MOM-PS distribuido y orientado a un ambiente mobile. El MOM está compuesto por un grafo de brokers (fijos) los cuales mantienen las subscripciones activas de los clientes móviles y realizan el macheo contra las publicaciones. Cuando un móvil publica un evento, este lo hace sobre el broker más cercano. A medida que ocurre el macheo, el evento se propaga por toda la red de brokers siguiendo la estructura de un árbol (spanning tree), hasta alcanzar los nodos consumidores. Por otro lado, los nodos realizan el macheo de eventos con una copia local de las subscripciones activas (quenching), de forma de alivianar el procesamiento en el MOM. El framework busca garantizar las siguientes propiedades: orden, consistencia y completitud de los mensajes.
Por otra parte, el trabajo [29] se basa en el diseño de un MOM LPS que incluye el tiempo como parámetro en la publicación, en un contexto de redes de automóviles (VANETs) con topología variable. Define espacios de publicación de los vehículos, donde una publicación se notifica y propaga mediante multicast a aquellos que estén en el mismo rango espacial, hasta que expire el tiempo de notificación. Este middleware permite definir diversos tipos de publicaciones, acordes a una categorización jerarquizada en forma de árbol, con atributos comunes en los niveles intermedios. Como en [63], existen nodos (vehículos) que funcionan como brokers para otros nodos en cierta área, que realizan el matching de eventos y subscripciones y se encargan de notificar los mensajes.

2.5.1.5. Comparación de algunos trabajos

En la tabla siguiente se aprecia una comparación general de las características comunes de algunos de los trabajos encontrados.

Tabla 1: Comparación de los trabajos relacionados

Característica	**[54]**	**[56]**	**[55]**	**[53]**	**[57]**	**[65]**	**[61]**
Variedad de proveedores de datos	X		X		X		X
Capacidad de agregar nuevos proveedores					X		X

Utiliza patrón PS	X	X		X		X	X
Mensajes estructurados	X	X		X		X	X
Operaciones espaciales SFA	X	X	X	X		X	
Restricciones de buffer		X	X				
Algoritmo de matching eficiente	X	X		X		X	
Utiliza diversos canales temáticos							
Geometrías de las publicaciones	Punto, polígono	Punto	Buffer, polígono, polilínea	Polígono		Punto	
Geometrías de las suscripciones	Buffer, polígono, polilínea	Buffer	Buffer, polígono, polilínea	Polígono		Punto, polígono	
Se expone algún LBS		X			X		
Utiliza plataforma de gobierno electrónico							
Utiliza base de datos geográfica	X		X				
Notificaciones push	X	X	X		X	X	X
Notificaciones pull					X		X
Flexibilidad sobre parámetros de suscripciones	X				X		
Utiliza diversos estándares de WS			X		X	X	
Autenticación de usuarios		X					X
Mensajería sincrónica				X	X		X
Mensajería asincrónica	X	X	X	X	X	X	X

3. Planteo de la solución

En este capítulo se brindara un análisis de una plataforma que soporte una extensión espacial de MOMs orientada a servicios basados en localización (LBS). Dicho análisis comprende requerimientos generales, actores, y escenarios posibles de uso de la plataforma. A su vez, se detallaran las posibles operaciones que pueden ser invocadas por cada uno de los actores y los mecanismos se utilizaran para recibir, recuperar y enviar los datos.

En la sección 3.1 se describirá un marco de trabajo del problema. En la sección 3.2 se detallara el planteo del problema a resolver. En la sección 3.3 se analizaran los posibles actores que publicaran datos hacia la plataforma y los que consumirán de los datos de la plataforma. En la sección 3.4, se dilucidaran la naturaleza de los datos que se publiquen, que información contengan y que actor los va a generar. Por último, en la sección 3.5 se abordaran las diferentes operaciones que pueden invocar cada uno de los actores en el sistema.

3.1. Marco de trabajo

Para la solución del problema que se verá en profundidad en las siguientes secciones, diversos elementos del marco teórico se verán involucrados. Específicamente, se trabajara sobre la Plataforma de Gobierno Electrónico (PGE) de AGESIC y, sobre dicha plataforma, se instalara un MOM extendido con capacidades geoespaciales, el cual conformara el core de la solución. Los diferentes actores de la solución se comunicaran con la plataforma para la satisfacción de sus requerimientos a través de diversos mecanismos. Se facilitara a los desarrolladores de la solución a habilitar servicios basados en localización (LBS) que permitirán obtener la información necesaria para ser procesada en el MOM.

3.2. Planteo general del problema

Se desea construir una plataforma a través de la cual se realice la publicación de eventos con datos de carácter geográfico por parte de organismos del Estado uruguayo, que poseen datos de interés para los usuarios del problema. Los datos pueden tener asociada una geometría especifica. Hay datos que pueden tener una localización geográfica puntual con coordenadas específicas (ejemplo: latitud y longitud); es el caso de un siniestro vial. Algunos datos pueden tener un área de influencia o de incidencia, representada por un polígono con puntos en el espacio geográfico. Es el caso de un incendio forestal, que abarca un área amplia y con límites específicos. Y, por último, otros datos pueden representarse mediante una polilínea en el espacio, por ejemplo, si se refiere a una ruta o camino de una ciudad. Por otra parte, el evento puede contener un timeout o tiempo de vencimiento según el cual, pasado dicho tiempo desde la emisión del evento, pierda validez y deba ser eliminado.

Estos organismos publican eventos de forma periódica en la plataforma accedida a través de la red, con una ubicación particular o área de influencia, y marca temporal. Además, los eventos tienen un tipo asociado y un nivel de prioridad, para indicar si corresponde a una alerta o un mensaje normal. Opcionalmente, podría incluir un tiempo de vida del evento, de forma que el mismo pierda validez pasado un tiempo desde el instante en que se generó. Utilizando algún mecanismo de persistencia, estos datos se almacenan en la plataforma para su posterior disponibilidad y uso de los usuarios.

Por otra parte, existen usuarios que desean recibir notificaciones o eventos, que pueden depender de su localización actual o que se produzcan en cierta área geográfica de interés. En el primer caso, dicha

área puede comprender un buffer de distancia a la ubicación del usuario (ejemplo: recibir incidentes de inseguridad que ocurran en un rango de 300 metros de mi posición actual) o un área con limites fijos y donde su posición pertenezca al interior del área (ejemplo: mi departamento actual o mi barrio actual). En el segundo caso, dicha área es arbitraria, permitiéndole al usuario definir un polígono sobre un mapa y recibir notificaciones que procedan de esa área independientemente de si su ubicación se encuentra en ella (ejemplo: deseo recibir eventos que ocurran en el LATU). El usuario puede dar de alta estas restricciones utilizando una aplicación a medida para cualquier dispositivo móvil corriente y se lleva a cabo a través de un servicio diseñado para tal propósito, el cual recibe las restricciones y la localización del usuario de forma periódica. En base a estas restricciones, el usuario recibe las notificaciones ya sea a pedido o en tiempo real. Además, los usuarios pueden indicar que tipo de eventos recibir y con qué nivel de prioridad discriminando entre varios posibles tipos de alertas.

Entre los usuarios y los publicadores de datos, existe una plataforma que recibe tanto las restricciones de los usuarios como los eventos generados por los organismos. Esta plataforma debe contar con las capacidades de un MOM, es decir, un sistema de mensajería que permita entregar de forma fiable y segura los mensajes de los productores a los usuarios, bajo diversos mecanismos de comunicación. La plataforma debe recibir los eventos de los organismos y alojarlos temporalmente hasta que los clientes estén disponibles para ser notificados, utilizando el patrón PS descrito en el capítulo 2. En este esquema, las subscripciones de los usuarios comprenden las restricciones espaciales que imponen a las notificaciones que pretenden recibir. El sistema de mensajería debe cruzar la ubicación puntual o área geográfica de los eventos y las zonas de interés de los usuarios, y si la intersección es diferente a vacío, realizar las notificaciones correspondientes.

Las notificaciones se pueden realizar bajo un pedido explícito del usuario o sin un previo pedido y en tiempo real. Bajo la primera modalidad, el usuario ingresa a la aplicación en su dispositivo móvil y requiere actualizar su bandeja de entrada con las últimas notificaciones. La plataforma envía los mensajes que el usuario aún no ha sido notificado. En la segunda modalidad, cuando el usuario está conectado a la plataforma, se fuerza el envío de mensajes hacia su bandeja de entrada, refrescándosele su listado. En este contexto, el usuario podría indicar la modalidad de notificación que desee tener en una suscripción particular.

3.3. Actores

Se cuenta con dos tipos de actores: los proveedores de datos geográficos publican sobre la plataforma, y los usuarios que son notificados de los eventos en relación a sus restricciones. Además, un usuario intermedio, el desarrollador, se encarga de realizar la implementación de la aplicación para el usuario final.

3.3.1. Proveedores de datos

En el Estado uruguayo existen múltiples organismos que generan datos de diversa índole, para ser consumidos por otros organismos o por el público en general. A continuación, se describirán algunos de los organismos que pueden ser interesantes para el dominio del problema:

- SINAE. El SINAE (Sistema Nacional de Emergencias) es el organismo cuya responsabilidad es la gestión integral de riesgo de desastres en el Uruguay. Su objetivo es proteger a las personas, los bienes de importancia y el medio ambiente de fenómenos adversos que deriven, o puedan

derivar, en situaciones de emergencia o desastre [8]. Se encarga de monitorear, informar y adoptar medidas para la reducción de riesgos y manejo de situaciones de emergencia para las siguientes clases de amenazas: concentración masiva de personas, incendios, inundaciones, eventos meteorológicos adversos, derrames de materiales peligrosos, siniestros y accidentes, olas de frio y de calor, etc. Ofrece un sistema de información geográfica en el cual se proyecta la cartografía del Uruguay y diversas capas temáticas [66]. Es un organismo dependiente directo de la Presidencia de la Republica así como de varios ministerios.

- INUMET. El INUMET (Instituto Nacional de Meteorología) es la autoridad meteorológica de la República Oriental del Uruguay, y la autoridad aeronáutica en aplicación de la Convención de Aviación Civil Internacional (OACI) [7]. Su misión es prestar servicios públicos meteorológicos y climatológicos, con el objeto de contribuir a la seguridad de las personas y sus bienes, actuando como autoridad meteorológica en el territorio nacional. Entre los diversos servicios que brinda, se destacan: mapas de precipitación, mapas de temperatura, datos de variabilidad climática, balance hídrico del suelo, etc. Depende del Ministerio de Vivienda, Ordenamiento Territorial y Medio Ambiente (MVOTMA).
- Dirección Nacional de Policía Caminera. La Dirección de Policía Caminera es la institución policial del país encargada de organizar, sistematizar y controlar el tránsito en la Red Nacional de rutas [9]. Tiene como objetivos controlar el cumplimiento de las normas de tránsito en todas las rutas y caminos de jurisdicción nacional, prestar auxilio a las víctimas de los accidentes de tránsito, procesar datos estadísticos de los servicios de las rutas, etc. Depende del Ministerio del Interior.
- Intendencias departamentales. Son los organismos responsables de la coordinación de políticas públicas de los gobiernos departamentales. Se encargan de la organización y la prestación de servicios y actividades propias o comunes, tanto en sus respectivos territorios como en forma regional o interdepartamental [67]. Actúan junto al Poder Ejecutivo, Entes Autónomos y servicios descentralizados. En particular, la Intendencia de Montevideo ofrece un sistema de información geográfica [68] que permite plasmar en un mapa datos geográficos de padrones, manzanas, barrios, categorías de suelo, calles, estacionamiento tarifado, centros culturales, complejos habitacionales, ciclovías, líneas de ómnibus, etc. Además, provee archivos en formato shapefile sobre esos features de la cartografía de Montevideo.
- AGESIC. AGESIC [16] es la Agencia Nacional de gobierno electrónico. Provee un catálogo de datos abiertos a través de una API de web services y a través de archivos descargables en diferentes formatos. Entre los datos que ofrece a través del catálogo se cuenta con: líneas de ómnibus, accidentes de tránsito, paradas de ómnibus y puntos de control, municipios de la ciudad de Montevideo, asentamientos, etc.
- OAN. El OAN (Observatorio Ambiental Nacional) es una plataforma de información ambiental, de libre acceso, que aporta insumos para la toma de decisiones institucionales y al mismo tiempo acerca esa información a la gente [69]. Depende del Ministerio de Vivienda, Ordenamiento Territorial y Medio Ambiente. Permite acceder a indicadores ambientales que permiten visualizar la evolución y tendencias del estado del ambiente, ecosistemas, emisiones contaminantes, residuos, afectaciones a la calidad del agua, aire, suelo y biodiversidad. Ofrece variedad de datos abiertos de carácter geográfico y medioambiental. El sitio web publica datos abiertos sobre calidad de agua a nivel de cuenca, subcuenca y departamento, calidad de aire para estaciones fijas y móviles y por periodo de tiempo, y áreas protegidas en todo el territorio.
- Cámara de Comercio. La Cámara de Comercio y Servicios es una organización formada por empresarios o dueños de pequeños, medianos o grandes comercios con el fin de elevar la

productividad, empleados y competitividad de sus negocios. Su misión es velar por el interés general del comercio y los servicios y del sector privado de la Economía Nacional [70].

- UNASEV. La UNASEV (Unidad Nacional de Seguridad Vial) es un organismo dependiente de la Presidencia de la Republica creado para unificar criterios para encarar la problemática del tránsito y la seguridad vial [71]. Los objetivos de la UNASEV, son la construcción de una política nacional en Seguridad Vial, y promoverla para que todos los usuarios de las vías de tránsito circulen de forma segura. Procura contribuir a generar un cambio de la "cultura vial" del país y responsabilidad social. Provee una variedad de datos abiertos que incluyen datos sobre características de los siniestros viales por fecha y departamento y los fallecidos en estos por rango de edad.
- Proveedores privados. Además de los organismos del Estado, podrían agregarse otros proveedores de datos de la órbita privada que publiquen datos de interés público y de naturaleza geoespacial para generar notificaciones oficiales. Entre ellos, se pueden distinguir:
    - PedidosYa. PedidosYa es una compañía de delivery de comida online de América Latina de 15.000 restaurantes con entrega a domicilio en Argentina, Brasil, Chile, Panamá, Paraguay y Uruguay. El servicio consiste en brindar una plataforma online simple, práctica y gratuita que permite a los usuarios elegir un plato y realizar el pedido a través de su sitio web o las aplicaciones para móviles [72].
    - CityCop [73]. Es una aplicación que permite registrar incidentes de inseguridad, tales como robos y diferentes tipos de crímenes. También permite recibir alertas en tiempo real de los incidentes registrados por otros usuarios. Dado que los datos publicados en CityCop pueden carecer de relevancia o de consistencia para el resto de los usuarios, debería existir un fuerte control de calidad de los mensajes publicados sobre la plataforma.
    - Socio Espectacular. Es un servicio que permite obtener acceso a una amplia oferta de expresiones culturales a través del pago de una cuota mensual: danza, cine, teatro, futbol. Fue cofundado por el Teatro el Galpón y el Teatro Circular de Montevideo [74].

3.3.2. Usuarios finales

Los usuarios finales de la plataforma pueden ser cualquier usuario del público general. El único requisito del usuario es registrarse en el sistema e ingresar sus preferencias personales de las notificaciones que desea recibir. Se requiere como mínimo que el usuario este registrado a nivel del Estado con su cedula de identidad.

3.3.3. Desarrollador de aplicaciones

Es un usuario intermedio entre la plataforma y los usuarios finales y es el encargado de implementar la aplicación para el cliente móvil, de forma de permitir la entrada y lectura de datos utilizando interfaces acordes y bien definidas. La aplicación desarrollada puede ser desde muy genérica para consumir eventos de cualquier organismo/prioridad/tipo de evento, hasta ser circunscripta a cierto o ciertos proveedores de datos o a procesar y desplegar cierto tipo o tipos de eventos por prioridad, o alguna combinación de estos. Dicha aplicación está sujeta a las necesidades de un tipo particular de usuario y a los requerimientos establecidos por un organismo o grupos de organismos. Independientemente de estas variables, la aplicación debe contar con las funciones descritas más adelante y con la posibilidad de ejecutar las operaciones necesarias para cada caso de uso, haciendo uso de la API de servicios de la plataforma.

3.4. Eventos generados

Múltiples tipos de eventos de carácter geográfico podrían generarse y ser transmitidos al GeoMOM para notificar a los usuarios subscriptos. En esta sección se categorizaran dichos eventos y se mencionara cuáles de los organismos expresados anteriormente podrían ser sus emisores. Los eventos pueden clasificarse según diferentes dimensiones: por tipo de evento, por prioridad y por proveedor de datos. En la siguiente subsección se analizaran los diferentes tipos de eventos y luego se presentara una tabla comparativa donde se cruzaran las tres dimensiones.

3.4.1. Tipo de evento

Se pueden discernir diferentes categorías de eventos de ser generados por los proveedores de datos.

3.4.1.1. Eventos de tipo meteorológico

Se podría notificar sobre tormentas y vientos fuertes, granizo o heladas en determinadas zonas, barrios o regiones del país, incluyendo las inundaciones que ocurran por fuertes precipitaciones de lluvias. Este tipo de eventos acarrean un área geográfica o polígono de influencia. El usuario podría subscribirse a los eventos de tipo meteorológico recibiendo las alertas meteorológicas en tiempo real y los pronósticos informativos mediante consulta. Dichas zonas podrían ser el departamento o barrio donde actualmente está localizado el usuario.

3.4.1.2. Eventos de criminalidad e inseguridad

En esta categoría, se podría notificar de eventos asociados a actos criminales, como robos, rapiñas y asesinatos. Este tipo de eventos tienen una ubicación puntual en el espacio. De esta forma, el usuario puede recibir en tiempo real aquellos mensajes de más alta prioridad, que se encuentren en un buffer de distancia a su localización o en determinada zona (barrio o municipio). Esta variante de datos podría ser suministrada por el Ministerio del Interior en base a las denuncias formales registradas en las seccionales o por datos generados por los vecinos a través de la aplicación CityCop
Por otra parte, sería interesante advertirle al usuario sobre aquellas zonas o barrios evaluadas con mayor riesgo en base al número de episodios de inseguridad, representadas como un polígono en el espacio. En esta variante, el Ministerio del Interior podría suministrar la capa de seguridad por zonas, discerniendo entre zonas de bajo riesgo (verdes), de mediano riesgo (amarillas), de alto riesgo (naranjas), y de muy alto riesgo (rojas).

3.4.2. Eventos asociados a rutas y caminos

Se podría notificar de eventos que representen alguna incidencia que ocurra en rutas o caminos de vialidad nacional. Estas incidencias pueden referirse a cortes o exceso de tráfico en las rutas. Este tipo de eventos pueden tener prioridad media o alta y tienen asociada una polilínea que represente el tramo de la ruta afectado por el suceso. Una vez subscripto el usuario, cuando va circulando por la ruta y en un buffer de kilómetros relativo a su localización actual, se le notifica en tiempo real sobre una aglomeración de vehículos o un desvió de ruta. De esta forma, el usuario puede tomar una ruta alternativa para prevenir un embotellamiento.

Otro tipo de evento interesante seria obtener los fragmentos de velocidades máximas de circulación de rutas y caminos. En esta variante, el usuario subscripto recibe alertas cuando supera la velocidad máxima del tramo en el que está circulando, según su localización. Esta información puede ser generada por alguna capa de rutas de las intendencias con los límites de velocidad en las rutas y caminos. Este evento tiene prioridad alta y corresponde a una representación de polilínea ya que se traduce en segmentos de ruta.

3.4.2.1. Eventos de incendios

Otro tipo de evento seria los incendios producidos en algún lugar. La representación podría ser puntual o un área de influencia (polígono) afectada por el incendio si se produce una concentración masiva de personas. El evento tendría cualquier prioridad, dependiendo de la magnitud del incendio. Adicionalmente, en el evento se podría indicar el tipo de inmueble afectado (edificio, casa, fábrica, vehículo, etc.) y la gravedad del mismo.

3.4.2.2. Eventos al público

Los eventos al público refiere a aquellos eventos donde se realiza alguna participación multitudinaria, diferenciando entre aquellos que están sucediendo actualmente, como ser: marchas de protesta, partidos de futbol, carreras a pie (ejemplo: running 5K) o de bicicletas, exposiciones artísticas; o que se producirán en el futuro, como ser cursos temáticos, obras de teatro, conciertos musicales, etc. Para el caso de marchas o carreras, la representación geométrica es una polilínea donde tiene curso la marcha. La prioridad podría ser alta, ya que el usuario que está conduciendo un vehículo debería desviarse para tomar un camino alternativo hacia su destino por estar las calles cortadas. Para los casos de exposiciones artísticas o partidos de futbol, la representación puede ser un polígono. Los eventos que ocurrirán en el futuro son solo de carácter informativo. El usuario podría subscribirse a un subconjunto de estos subtipos de eventos y definir qué alertas recibir. La mayoría de estos eventos son relevados por las intendencias departamentales.
En caso de un usuario que es miembro del servicio Socio Espectacular, puede ser notificado de actividades culturales que se desarrollen en ciertos puntos del departamento en función de su localización. Se puede filtrar por tipo de actividad.

3.4.2.3. Eventos de accidentes viales

Los eventos que responden a accidentes viales se deberían desencadenar por siniestros de tránsito en calles, rutas o caminos de vialidad nacional. Pueden contener información de las características de los vehículos que participan en el siniestro, con imágenes del estado de los daños ocasionados y reportar si existieron fallecidos en el acto. La representación geométrica habitual para estos eventos es puntual en el espacio. El evento puede tener prioridad baja o alta. Para la prioridad baja, el usuario desea informarse de los siniestros que ocurren en un área particular (como un buffer relativo a la ubicación). Para la prioridad alta, el usuario tiene como objetivo tomar un camino alternativo para evitar un potencial embotellamiento causado por el bloqueo de los vehículos impactados y se es notificado en forma inmediata.

3.4.2.4. Eventos asociados a locomoción

Son eventos que están vinculados al transporte urbano. Un ejemplo son las líneas de ómnibus. Un evento de esta categoría podría contener la ubicación actual de las unidades vehiculares de ómnibus. Un usuario típico que desea utilizar el transporte público le interesaría conocer cuál es la localización actual de cierta línea de ómnibus en el departamento, de forma de minimizar el tiempo de espera del ómnibus en la parada. La representación geométrica es claramente puntal y el evento podría incluir las horas en que el móvil pasara por cada parada. De esta forma, el usuario se subscribe a recibir mensajes de los móviles de ómnibus de cierto departamento que estén próximos por pasar en una parada especifica. Esta información la pueden brindar las intendencias departamentales.
Otro ejemplo de eventos serían los relacionados a taxis. En este caso, el usuario se subscribe a recibir la ubicación de los móviles que estén en un buffer de distancia.

3.4.2.5. Eventos asociados a establecimientos comerciales

Estos eventos se disparan en función de novedades relacionadas a ciertos tipos de comercios, o en base a brindar la ubicación más próxima a un local comercial en particular. El primer caso resultaría en notificar, por ejemplo, ante la apertura o cierre de cierto local comercial, donde se indicaría en el mensaje la dirección exacta del local y a que firma y rubro pertenece. A modo de ejemplo, apertura de una sucursal de la farmacia Farmashop en esquina 26 de marzo y José Ellauri. Esta información puede ser útil para el usuario ya que, si se subscribe a este tipo de evento, puede ser notificado y dirigirse a la nueva sucursal más cercana a su domicilio. La suscripción podría incluso realizarse por el tipo de comercio o el nombre de la firma.
En el segundo caso, el evento corresponde a la ubicación más cercana de un local comercial de interés, por ejemplo: agentes de cobranza (Abitab) o estaciones de servicio. El usuario suscrito desea conocer la sucursal más cercana del comercio elegido.
En cualquiera de los dos casos, la localización de las sucursales de cada comercio y el tipo de comercio puede ser consultada con los registros de la Cámara de Comercio y Servicios. La Cámara posee el listado de todos los locales habilitados a ejercer actividad comercial en las zonas declaradas como comerciales.

3.4.2.6. Eventos relacionados a vivienda

Estos eventos están vinculados a ventas de inmuebles (casa, terreno, apartamento, etc.) o el surgimiento de planes de vivienda del Estado en determinada zona. Un usuario interesado puede ser notificado de ventas o planes de vivienda que existan en determinada zona, municipio, barrio o departamento. La representación geométrica puede ser puntual si se trata de una venta de un inmueble, ya que indicara la localización del mismo; o un polígono si es un plan del Estado ya que puede abarcar a un conjunto de viviendas. Este tipo de evento es solo informativo.

3.4.2.7. Eventos de estacionamiento disponible

Estos eventos permiten brindar datos de estacionamiento céntrico vial de la ciudad. La intendencia departamental podría hacerse cargo de relevar periódicamente estos datos a horas pico en la capital y publicar aquellas franjas donde exista estacionamiento disponible. Dichas franjas deben tener representación geométrica de polilínea, ya que pueden conformar segmentos contiguos de calles. Un usuario subscripto desearía encontrar un lugar disponible a cierta hora y calle específicos, y podría ser notificado con el lugar más próximo en tiempo real de ello.

3.4.2.8. Eventos de medio ambiente

Los eventos orientados a impacto medio ambiental están dirigidos a usuarios del ámbito rural. El OAN podría brindarle a los usuarios propietarios de tierras rurales, a través de sus estaciones de control, información de variables medioambientales, como ser calidad del aire, balance hídrico del suelo, fauna rural y emisiones de contaminantes sobre los suelos. Con estos datos, un usuario subscripto puede recibir datos a pedido de los padrones que son de su interés, representados por un polígono en el espacio, de forma de tomar medidas en caso de que se violen los parámetros medioambientales. Otro tipo de usuario interesado pueden ser las intendencias departamentales que deben regular el uso apropiado y sostenible de los suelos, fiscalizando el no tratamiento adecuado de desechos, debido al uso de modalidades de explotación que produzcan desgaste. De esta forma, las intendencias son notificadas de los datos medioambientales que ocurran en los departamentos de su jurisdicción y analizan dichos datos para generar alertas o multas sobre los propietarios rurales. Estas alertas también podrían ser publicadas a través de la plataforma, y ser decepcionadas a través de notificaciones a los padrones del departamento que corresponda.

3.4.2.9. Eventos de comidas

Seria de interés notificar al usuario de la ubicación de los restaurantes o locales de comida cercanos con la capacidad de brindar disponibilidad de platos preferidos y precios preferenciales. Esta información puede ser provista por el servicio de PedidosYa a través de su red de restaurantes y locales inscriptos.

3.4.3. Prioridades y proveedores de datos

En la siguiente tabla se puede ver, para cada tipo de evento, cuáles serían los proveedores de datos posibles y los niveles de prioridad según diferentes circunstancias.

Tabla 2: Clasificación de eventos

Tipo de evento	Proveedor de datos	Prioridad
Meteorológico	INUMET, SINAE, intendencias departamentales	Baja o Media: consulta de pronostico Alta: alertas meteorológicas
Criminalidad de inseguridad	Ministerio del Interior, CityCop	Alta: episodio cercano de inseguridad, circulación en zona de riesgo
Rutas y caminos	Dirección Nacional de Policía Caminera, intendencias departamentales	Alta: desvió de ruta, alta densidad de tráfico, exceso de velocidad
Incendio	SINAE	Media o alta según las dimensiones del incendio
Publico	Intendencias departamentales, SocioEspectacular	Baja: información futura de cursos, obras teatrales, conciertos, etc. Media: evento ocurriendo en cierto lugar y con alta densidad de personas (ejemplo: partido de futbol) Alta: Cortes importantes de caminos (carreras o marchas de manifestación)
Accidente vial	Dirección Nacional de Policía Caminera, UNASEV, AGESIC	Media: accidentes en zona especifica Alta: accidentes a corta distancia del usuario

Locomoción	Intendencias departamentales	Baja: horarios de líneas de ómnibus por parada Media: móviles de taxis según buffer de usuario
Locales comerciales	Intendencias departamentales, Cámara de Comercio	Media: apertura o cierre de local comercial, ubicación más cercana a sucursal.
Vivienda	Intendencias departamentales, Cámara de Comercio	Baja: venta de inmuebles, planes de vivienda estatales
Estacionamiento	Intendencia departamental	Media o alta: búsqueda de un lugar disponible cercano al usuario
Medio ambiente	OAN	Media: notificación de variables medioambientales Alta: violación de parámetros medioambientales en cierto padrón rural
Comidas	PedidosYa	Baja: nuevos platos o restaurants

3.5. Operaciones

Existen diferentes tipos de operaciones que se pueden realizar sobre la plataforma, tanto de parte de los publicadores de datos como de los usuarios. La plataforma debe ser capaz de ofrecer interfaces bien definidas, estableciendo estructura y semántica de los parámetros y de los resultados. En la Figura 3.1 se pueden apreciar las operaciones que se pueden invocar sobre la plataforma. Se darán detalles de las operaciones por actores.

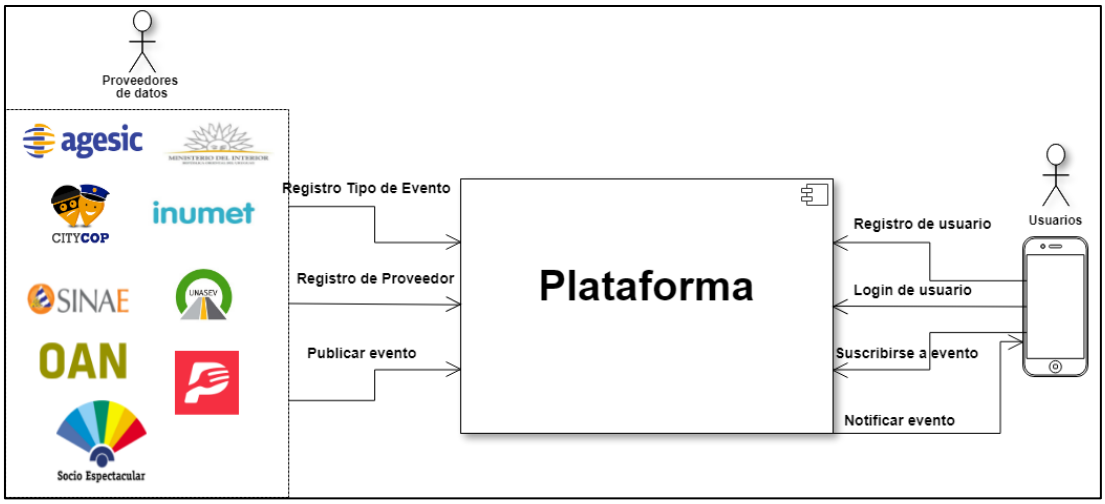

Figura 3.1: Operaciones sobre la plataforma

3.5.1. Operaciones de proveedores

Las operaciones que puede ejecutar un publicador de datos son: registro de proveedor y publicación de evento.

3.5.1.1. Registrar proveedor

Un proveedor (SINAE, INUMET, OAN, etc.) desea registrarse como proveedor en la plataforma. Para ello, debe brindar una serie de datos. En primer lugar, el proveedor debe brindar su identidad. La identidad puede ser otorgada por un certificado emitido y firmado por alguna autoridad certificadora que represente al organismo; puede ser un token de seguridad o simplemente un usuario y contraseña. La plataforma debe verificar los datos para autenticar al proveedor y validar que provengan de una institución del Estado. La operación indica si el registro fue exitoso o no. En el segundo caso debe imprimir el error asociado. En el primer caso y como postcondición, el proveedor puede empezar a registrar tipos de eventos y publicar datos sobre la plataforma.

3.5.1.2. Registrar tipo de evento

En esta operación, el proveedor registrado asocia una estructura a cierta clase de eventos que va a emitir. Esta estructura será descrita en el capítulo 4. La operación recibe como parámetro la estructura del evento que va a publicar y la identificación del proveedor, de forma que la plataforma pueda hacer la asociación correspondiente.

3.5.1.3. Publicar evento

El proveedor, una vez registrado y autenticado, realiza la publicación de un evento en la plataforma de acuerdo a la estructura de algún tipo de evento dado de alta en la operación Registrar tipo de evento. Para ello, debe enviar el mensaje estructurado según la metadata que proporciono en el registro. Los tipos de los eventos son los detallados en la sección 3.3.

3.5.2. Operaciones de usuarios

Las operaciones que puede ejecutar un publicador de datos son: registro y login de usuario y suscripción a eventos.

3.5.2.1. Registrar usuario

Se ejecuta cuando un usuario cualquiera desea registrarse en la plataforma para recibir notificaciones de eventos desde su dispositivo móvil. Para ello, provee un nombre de usuario, contraseña y mail de contacto. Puede incluir su documento de identidad opcionalmente. La operación debe devolver un mensaje de éxito si el registro es correcto, o un mensaje de error por el contrario.

3.5.2.2. Login de usuario

El usuario se loguea desde su cliente móvil utilizando las credenciales provistas en el registro. Un usuario logueado puede tener acceso a las operaciones siguientes.

3.5.2.3. Subscribirse a eventos

Se ejecuta cuando un usuario registrado y logueado desea subscribirse a los tipos de eventos de la plataforma. Esto puede realizarse desde una aplicación cliente para dispositivos móviles o desde la web según el organismo interesado que publica el tipo del evento de la suscripción. Para ello, debe

ingresar el tipo de evento al cual quiere subscribirse. Los eventos sugeridos son: accidente, incendio, meteorología, estacionamiento, vivienda, línea de ómnibus, taxis, incendios, vialidad, inseguridad, comercios y medio ambiente. Según cada tipo de evento, el usuario podría agregar filtros adicionales para refinar su criterio de búsqueda. Estos parámetros son aquellos dados de alta en Registrar Tipo de Evento del proveedor donde se indica qué parámetros pueden ser filtrables.
Una vez definida las características del evento sobre el cual quiere ser notificado, es necesario fijar en que zona del espacio geográfico donde debe suceder y de que prioridad. Esto lo puede realizar de dos formas:

- Definiendo un buffer de distancia. El usuario ingresa la distancia desde su posición actual y la unidad de medida (metros o kilómetros). El radio del buffer no debe ser inferior a los 100 metros ni superar los 50 kilómetros.
- Definiendo una zona libre en un mapa. Para ello, abre un mapa donde traza un polígono que contenga la zona de interés como lo hace Google Maps [75]. En la Figura 3.2 se ilustra un ejemplo de definición del polígono. La zona de interés abarca los barrios Barrio Sur, Ciudad Vieja, Centro, Cordón y Parque Rodo.

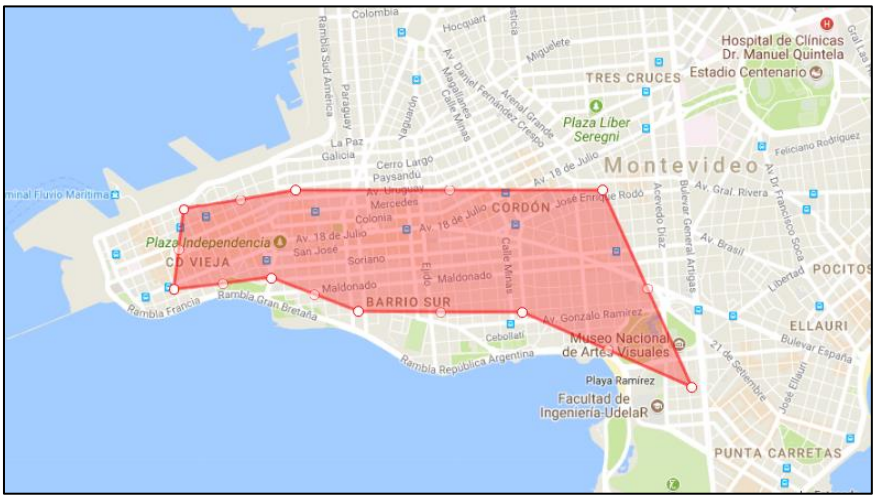

Figura 3.2: Dibujado de un área definida por el usuario

- Seleccionando una división administrativa especifica. Por ejemplo: departamento de Montevideo, departamento de Canelones, ciudad de Punta del Este, localidad de Balizas, etc. La aplicación muestra al usuario los límites del área seleccionada, tal como se muestra en la Figura 3.3. En el ejemplo, se demarca el balneario de Punta del Este.

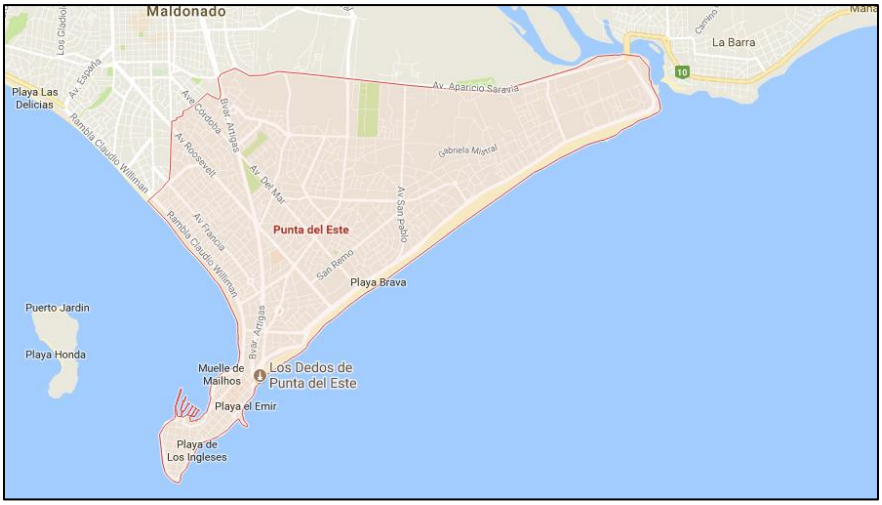

Figura 3.3: Área geográfica administrativa

Por último, los niveles de prioridad posibles para los eventos son: bajo, medio, alto y muy alto, que se relacionan respectivamente con las siguientes descripciones de prioridad: dato informativo, advertencia, alerta y alerta urgente.

3.5.2.4. Notificar evento

En última instancia, el usuario registrado recibe las notificaciones a su bandeja de entrada. Para ello, ingresa a la aplicación cliente desde su dispositivo móvil y solicita la recepción de los nuevos eventos. La recepción se hace de forma explícita sobre eventos de baja prioridad y de forma implícita en los de mediana y alta prioridad, de acuerdo a sus subscripciones. La plataforma realiza el macheo entre los eventos y las subscripciones del usuario, y efectiviza el envío si se cumplen las siguientes condiciones:

1. El tipo de evento de la suscripción coincide con el tipo de evento en cuestión.
2. Los filtros adicionales de la suscripción coinciden con los parámetros adicionales del evento en cuestión.
3. Existe intersección en la representación geométrica del evento y la geometría asociada a la suscripción.

En la Figuras 3.4 y 3.5, se ilustra tal como se mostraría al usuario el resultado de la intersección, cotejando su área de suscripción y el evento notificado. En el primer caso contra un área definida contra un evento puntual y en el segundo caso se ilustra un buffer con un evento de polígono.

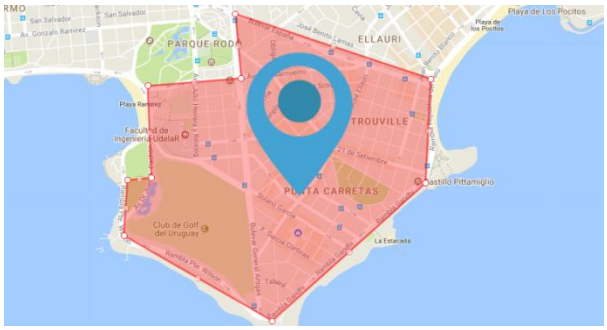

Figura 3.4: Notificación en zona definida

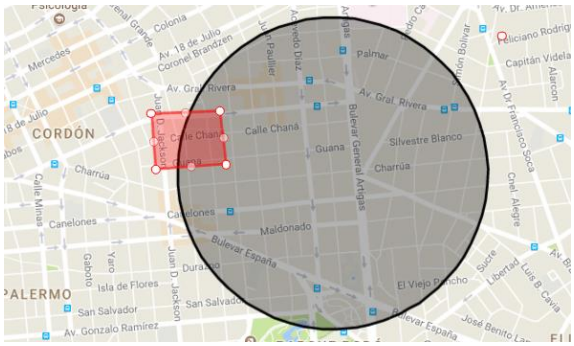

Figura 3.5: Notificación en buffer de usuario

Además, debe de mostrarse toda la información relativa al evento, incluyendo el tipo del evento, el organismo publicador, prioridad, título y la descripción.

3.5.3. Secuencia de operaciones

En la Figura 3.6, se puede apreciar un diagrama de secuencia UML que permite visualizar la secuencia de operaciones que se invocan en el sistema según cada actor.

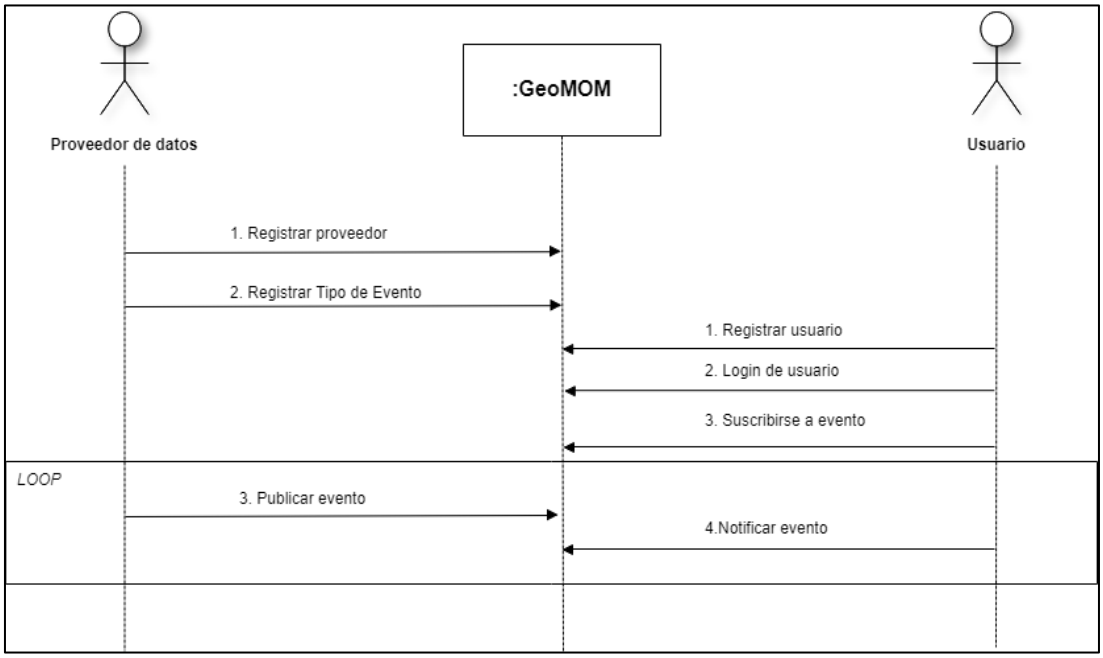

Figura 3.6: Operaciones del sistema

4. Propuesta de la solución

En este capítulo se presentan encares de solución al problema presentado en el capítulo anterior. Se enfocará en el análisis para los flujos de datos en la plataforma y especialmente en el uso de parte de los organismos y de los usuarios. También se prioriza el criterio de generar el menor impacto posible a los organismos proveedores.
Se propondrá una arquitectura en base a los requerimientos establecidos en el capítulo anterior. Se definirán los diferentes componentes de dicha arquitectura, protocolos, estándares y mecanismos de comunicación entre ellos, y responsabilidades individuales. A su vez, se detallaran los diversos mecanismos y artefactos utilizados para realizar las notificaciones a los usuarios y la forma de persistencia de datos en la plataforma. Por otra parte, se definirán los protocolos y estándares utilizados para las interacciones entre los actores (organismos y usuarios) con la plataforma, en particular para ejecutar las operaciones detalladas en el capítulo 3.

4.1. Arquitectura

Como primera instancia, cabe delimitar la porción o frontera del sistema sobre el cual se centrara la arquitectura. En la Figura 4.1 se muestra una representación del sistema en un diagrama de subsistemas UML. Como se verá en mayor detalle en este capítulo, la propuesta hará foco en los componentes GeoMOM y broker de mensajería.

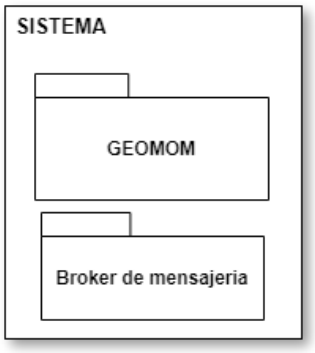

Figura 4.1: Frontera del sistema

Si bien la plataforma de la solución puede ser general, se toma como referencia la Plataforma de Gobierno Electrónico (PGE) de AGESIC Uruguay debido a las características avanzadas de la PGE que la convierten también en una plataforma modelo. Se utilizara la PGE de AGESIC y sus componentes internos. En el contexto de una PGE, existen entidades que producen y publican datos en la plataforma con diferente estructura y privacidad y existen entidades que consumen los datos.
Se propone el esquema arquitectónico ilustrados en la Figura 4.2. En el modelo la PGE constituye el principal componente de la arquitectura. Recibe los pedidos de los organismos y de los clientes a través del broker de mensajería. A la izquierda, los proveedores de datos publican eventos sobre la PGE. El subcomponente principal de estudio de la PGE es el GeoMOM que recibe los eventos. El broker de mensajería hace de interfaz entre el GeoMOM y los clientes móviles, recibiendo las subscripciones y notificando a estos.

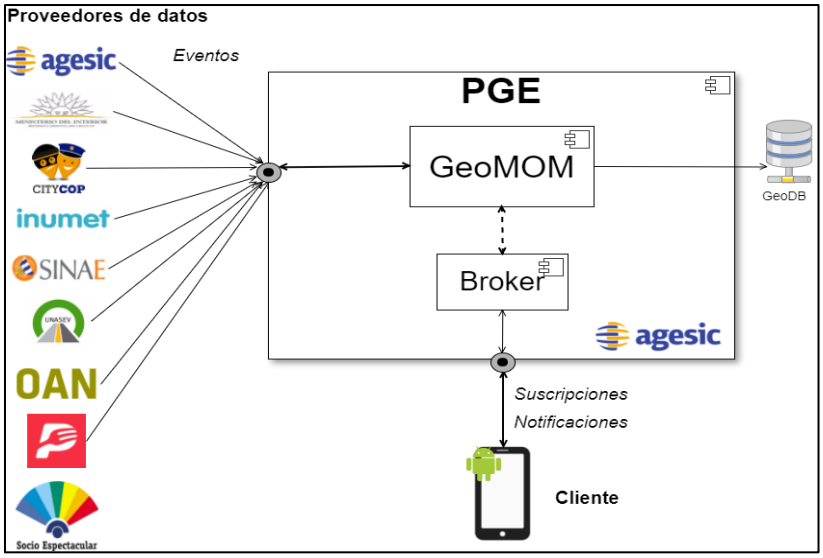

Figura 4.2: Arquitectura general

Los componentes de estudio en la arquitectura son: PGE, GeoMOM y broker de mensajería. La GeoDB forma parte del estudio del GeoMOM. Los proveedores de datos publican eventos sobre el GeoMOM a través de la PGE de AGESIC. El GeoMOM recibe los eventos de los proveedores de datos y los pedidos de los clientes móviles a través del broker de mensajería. Por otra parte, el GeoMOM genera los mensajes de notificación y son enviados al broker, el cual es responsable de hacerlos llegar a sus destinatarios respectivos. El GeoMOM se comunica con una base de datos que aloja los datos de usuarios, suscripciones, topics y proveedores.

4.2. PGE

En la Figura 4.3 se representan los componentes de la PGE modelo a utilizar. Además, se representa la secuencia de pasos para invocar un servicio utilizando la PGE, los cuales se detallaran en las secciones 4.2.1.1 y 4.2.1.2.

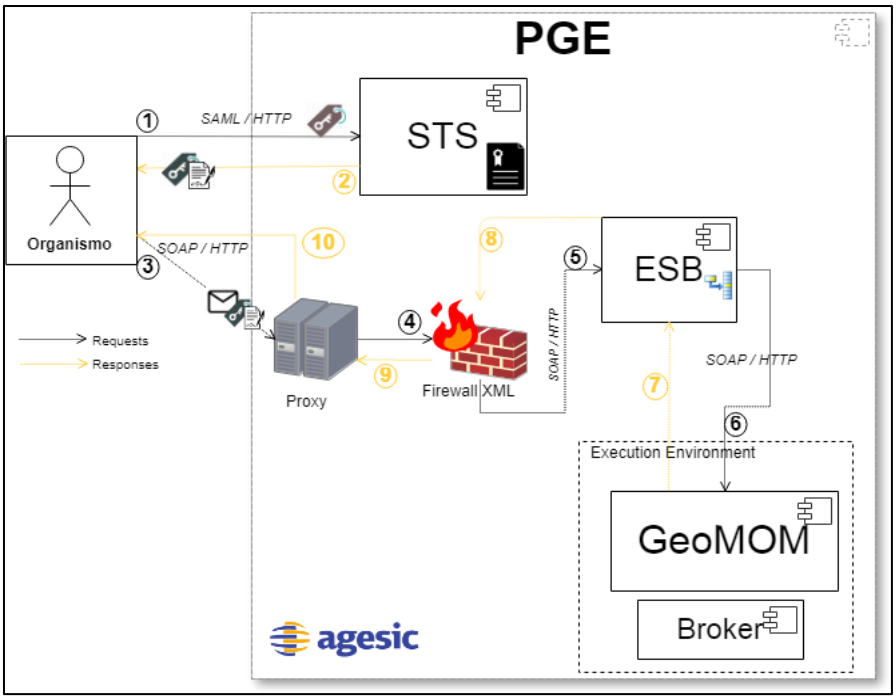

Figura 4.3: Subcomponentes de la PGE y flujo de mensajes para invocar servicio en GeoMOM

La PGE del problema está integrada por el sistema de autenticación y la plataforma de middleware, dos de los componentes fundamentales de la PGE de AGESIC. La plataforma componente que se encarga de procesar todos los pedidos y controlar que sean correctos así como también maneja los mecanismos de seguridad y de balanceo de carga. Cuenta con diversos componentes embebidos:

- STS (Security Token Service). Encargado de la autenticación. Provee tokens de seguridad que luego la PGE se encarga de verificar.
- Proxy. Recibe los pedidos de los organismos y los redirige hacia el ESB.
- Firewall XML. Intercepta los pedidos de los organismos, recibidos del proxy, y verifica la validez del token y que el token sea válido para el servicio invocado. En caso de que se permita el acceso, el pedido es redirigido a la Plataforma de Middleware, la cual envía la solicitud al servicio real. Es responsable de tomar la decisión de autorizar los pedidos de invocación a los de la plataforma.
- ESB. Es el principal componente de la plataforma de Middleware. Cuando un organismo quiere invocar un servicio, debe enviar un pedido a la PGE especificando, a través de una dirección lógica, el servicio que se quiere invocar. Esta dirección lógica identifica al servicio. El mapeo entre direcciones lógicas y físicas es gestionado en los ESBs [17]. El ESB implementa el MOM de la PGE, ya que permite realizar ruteo basado en contenido y transformación de los mensajes. Además, realiza nuevamente chequeos de seguridad de los pedidos, igual que el Firewall XML.
- GeoMOM. Es el componente central de la solución del problema. Recibe los mensajes de los organismos, las subscripciones de los usuarios y los eventos de los organismos. En la sección 4.3 se abordara con mayor profundidad.
- Broker de mensajería. En la sección 4.4. se abordara con mayor profundidad.

4.2.1. Invocación de servicios

4.2.1.1. Organismos – PGE – GeoMOM

La invocación a un servicio del GeoMOM utilizando la PGE se puede apreciar en la Figura 4.3. Como se puede deducir de la figura, todos los pedidos a la PGE deben realizarse invocando web services SOAP y, por ende, los mensajes intercambiados (pedidos y respuestas) deben tener formato SOAP-XML.

A continuación, se describen los pasos necesarios para consumar unas invocaciones relacionadas a la Figura 4.3, incluyendo protocolos y estándares de comunicación:

1. Un organismo que desee realizar un pedido a la PGE, debe obtener un token de seguridad SAML de la PGE. Para ello, se utilizan los estándares avanzados WS-Security y WS-Trust y la comunicación utiliza el protocolo SAML. Bajo el protocolo, el organismo emite un token SAML que utiliza para solicitar otro token al STS de la PGE (1). El token del organismo indica el servicio al que se quiere acceder y el retornado por el STS de la PGE, está firmado por este y contiene información de algún rol y usuario asociado al organismo de invocación (2).
2. Una vez que el organismo obtuvo un token de seguridad del STS de la PGE, debe invocar por servicio web SOAP al componente proxy de la PGE agregando el token al pedido (3), utilizando WS-Security, y encabezados WS-Addressing para indicar el servicio y método a invocar.
3. Los llamados recibidos por el Proxy son interceptados por el componente Firewall XML quien se encarga de validar y autorizar el pedido leyendo toda la metadata anterior (4).

4. Si la autorización es exitosa, el firewall reenvía el pedido al componente ESB, el cual realiza el mapeo entre dirección física y lógica (5).
5. Finalmente, el ESB reenvía el mensaje SOAP al servicio destino, en este caso en el GeoMOM, en base al mapeo anterior (6), utilizando el protocolo SOAP-XML.
6. Cuando el GeoMOM devuelve una respuesta, esta se propaga al ESB (quien invoco el servicio en la dirección real del GeoMOM) (7) y la reenvía al proxy service quien realizo el pedido original (8 y 9). El proxy entrega la respuesta a la entidad que realizo la petición (el forwarding se hace utilizando los encabezados de WS-Addressing (WS-ReplyTo).
7. El proxy service retorna la respuesta al organismo que realizo la invocación (10).

4.2.1.2. Clientes – PGE – Broker

El caso de la comunicación Clientes contra el Broker es análogo a la comunicación Organismos – PGE GeoMOM. Se ilustra en la Figura 4.4. La secuencia de pasos es idéntica a la comunicación Organismos- GeoMOM.

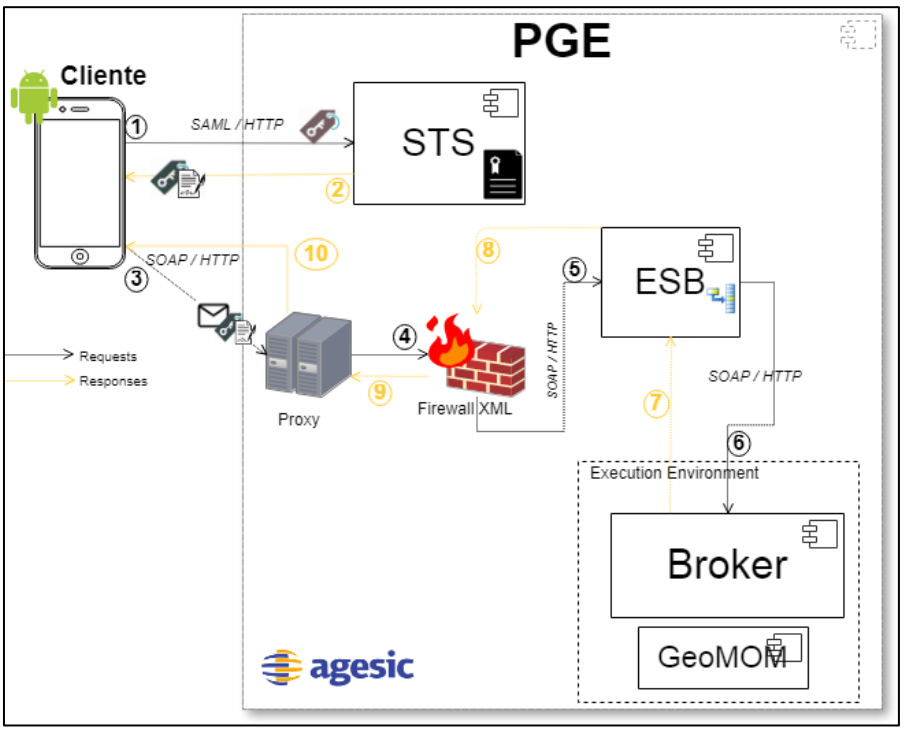

Figura 4.4: Pasos para ejecutar servicio del Broker

4.2.2. Servicios invocados sobre la PGE

Las funciones primordiales de la PGE para esta solución son:

- recibir los pedidos de Registro de Proveedor.
- Recibir los pedidos de Registro de Tipos de Evento de los proveedores a través del método "Registro de Topic" y redirigirlos al GeoMOM (ver sección 4.3.2.2).
- Recibir los pedidos de listado de tipos de eventos de los proveedores a través del método "Listado de Topics" y redirigirlos al GeoMOM (ver sección 4.3.2.3)
- recibir las publicaciones de eventos de los proveedores a través del método "Publicar Evento" ,y redirigirlas al GeoMOM (ver sección 4.3.2.1)

- Recibe pedidos de registro de usuario del sistema a través del método "Registro de Usuario" y redirigirlos al broker de mensajería (ver sección 4.4.2.1)
- recibe las suscripciones a eventos de los usuarios y redirigirlas al broker de mensajería, a través del método "Suscribirse a Topic" (ver sección 4.4.2.3)
- Recibir los pedidos de listado de topics de los usuarios a través del método "Listado de Topics" y redirigirlos al broker.
- Enviar las notificaciones de los eventos que se produzcan en el GeoMOM hacia el broker de mensajería y luego hacia los clientes finales (ver sección 4.4.2.4)

Los mensajes de entrada y salida de la PGE se pueden visualizar en la Figura 4.5.

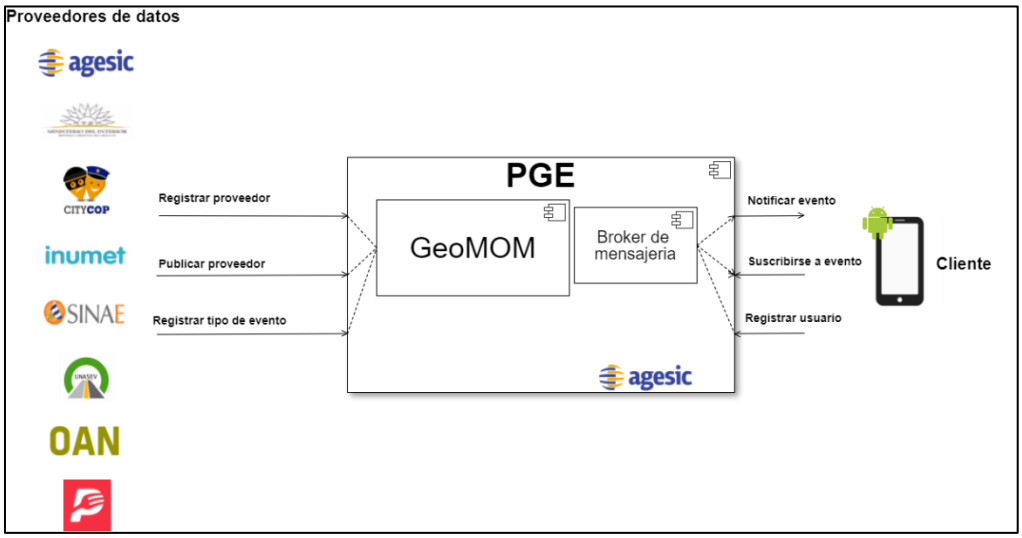

Figura 4.5: Métodos invocados sobre la PGE

4.2.3. Usuarios, roles y permisos

El sistema de autenticación de la PGE funciona a partir de mapeos entre servicios y métodos disponibles para invocar servicios internos o externos a ella, y usuarios, roles y permisos gestionados en la plataforma. Cuando una nueva entidad se da de alta en la PGE y desea exponer servicios, el servicio a publicar debe especificar las políticas de acceso que define. Las mismas incluyen los perfiles de usuarios que admite, las operaciones a las que cada perfil tiene permitido invocar y cómo se relacionan los perfiles con los roles definidos por los organismos que consumirán dicho servicio. De esta forma, es necesario mantener una correlación entre los perfiles de usuarios de los organismos con los servicios que se publican en la plataforma.

En el proyecto de grado de [76] de la Facultad de Ingeniería de la UDELAR, se propone un modelo conceptual de una simulación de la PGE de usuarios, roles y permisos que puede ser útil para este escenario. En dicho modelo (Figura 4.6) se manifiestan los principios de gestión de los conceptos anteriores.

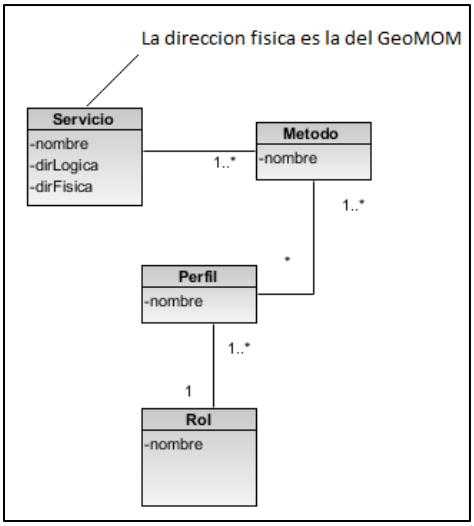

Figura 4.6: Modelo de datos de simulación de la PGE

Los servicios son la lista de servicios publicados en la PGE y en los cuales se hace el mapeo de direcciones lógica-física. Como ya se vio, la plataforma de middleware es quien utiliza este mapeo para redirigir los pedidos a los servidores destino utilizando las direcciones lógicas y WS-Addressing. Cada servicio se conforma de un nombre, una dirección lógica y una dirección física. Cada servicio puede tener asociado una colección de métodos que invocan cierta funcionalidad.
Los roles son colecciones de perfiles y agrupan permisos de usuarios, y cada perfil puede acceder a cierto conjunto de métodos de diferentes servicios. De esta manera, cuando se da de alta un usuario de un organismo, se le debe asignar el rol que permita tener algún perfil al cual este autorizado a la invocación de métodos asociados a servicios de la plataforma.

4.2.4. Métodos invocados sobre la PGE

Según lo que se vio en el capítulo 3, los actores que participan en las interacciones son los usuarios finales y los proveedores de datos. En las Figura 4.4, los actores pueden invocar varios métodos sobre la PGE: Registro de Proveedor, Publicar Evento, Registrar Topic, Listar Topics, Registrar Usuario, Suscribirse a Topic y Notificar evento. Salvo el método Registro de Proveedor, los otros pedidos de los otros métodos son direccionados al GeoMOM o al Broker de mensajería utilizando el mecanismo descrito en la sección 4.2.1.1. Estos métodos se abordaran en detalle en las secciones referentes a GeoMOM y Broker de Mensajería.

4.2.4.1. Registro de proveedor

Los organismos o servicios que quieran darse de alta como proveedores de datos de la plataforma deben invocar el método "Registro de Proveedor". La entidad brinda un identificador de proveedor, un mail de contacto y una lista de los usuarios y los roles internos a los que desea otorgar permiso de publicación y/o registro de tipos de eventos. El método genera una solicitud al administrador de la PGE de AGESIC que es revisada y aprobada o rechazada. En el primer caso, se envía una notificación por mail al proveedor indicando que el registro ha sido exitoso y, en caso contrario, se establecen los motivos por los cuales se denegó la solicitud.
En caso de la solicitud aprobada, se generan las entradas adecuadas en el modelo de base de datos de simulación de la PGE haciendo el mapeo de los usuarios del proveedor a los perfiles existentes.

4.3. GeoMOM

Es el principal componente que concierne a la solución del problema ya que implementa el middleware de mensajería de la solución. Es el encargado de recibir las publicaciones de eventos geográficos de los organismos a través de la PGE, y de las subscripciones y las localizaciones actuales de los usuarios a través del broker de mensajería. Además, es el responsable de despachar las notificaciones espaciales a los usuarios.

4.3.1. Arquitectura

En la Figura 4.7, se pueden apreciar las estructuras del sistema de mensajería que forman parte del GeoMOM.

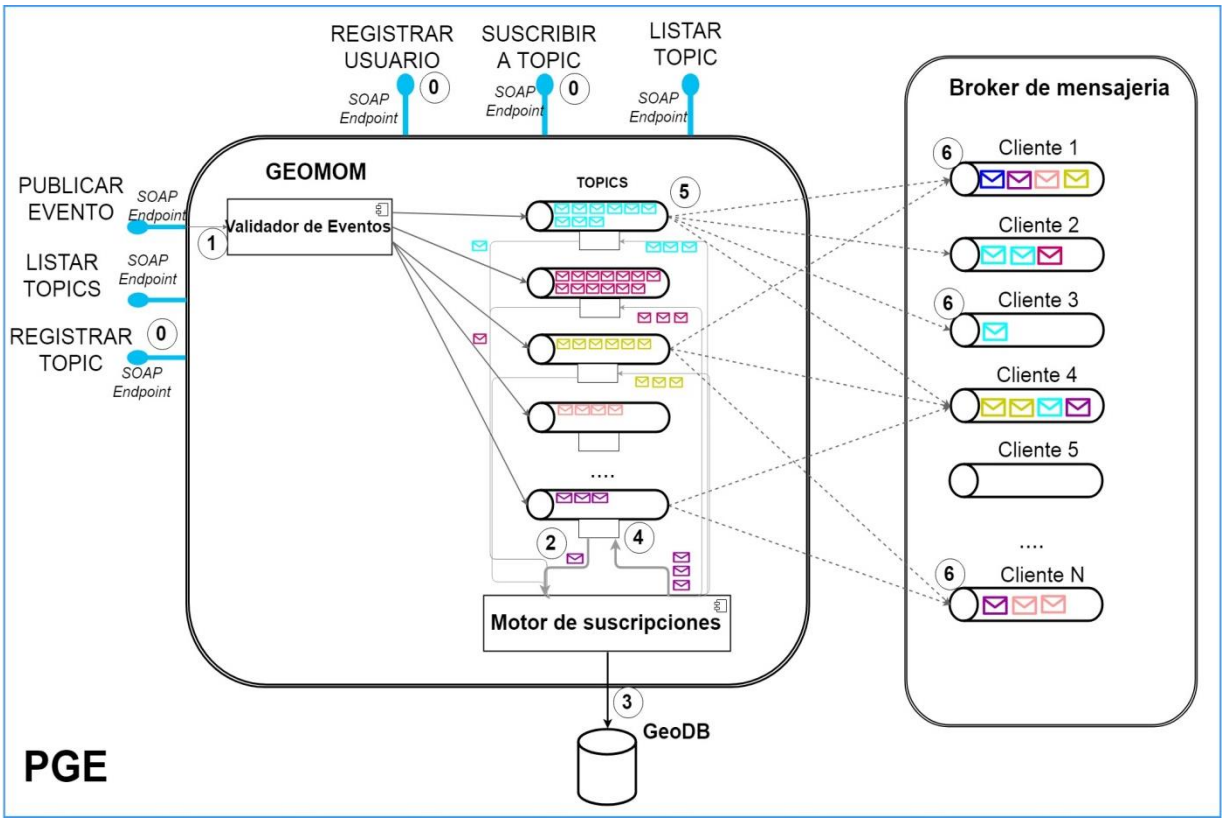

Figura 4.7: Arquitectura del GeoMOM

Como se puede observar, el GeoMOM, se compone de varios topics que reciben los mensajes entrantes de eventos de los organismos que viajan a través de la PGE y un motor de suscripciones que procesa los mensajes de los topics. Cada mensaje de diferente color en la figura alude a diferentes tipos de eventos. Cada topic se mapea a un tipo de evento dado de alta por un proveedor de datos. Por otra parte, el componente Validador de Eventos se encarga de validar la correctitud y correspondencia del mensaje.

El procesamiento, lectura y reenvío de los eventos se hace en una secuencia de pasos:

- En los pasos (0), el GeoMOM recibe los pedidos de nuevas suscripciones del broker de mensajería y registros de los tipos de eventos (secciones 4.4.2.1 y 4.4.2.3) de la PGE y organismos. Estos pedidos son obtenidos a través de endpoints SOAP que exponen web services apropiados para ello.

- En (1), un topic recibe los mensajes correspondientes a las publicaciones de eventos de su tipo. El topic es un canal de mensajes de tipo PS.
- El GeoMOM se encarga de procesar cada mensaje en el Motor de Suscripciones (2), quien manda a almacenar en la base de datos GeoDB las nuevas suscripciones y los tipos nuevos de eventos posibles. La estructura del mensaje se conforma de acuerdo a la estructura de los tipos de eventos que el organismo registra en la plataforma, según se verá en la sección 4.3.2.2.
- El motor de suscripciones se encarga de tomar el evento leído, obtener las suscripciones de los usuarios de la base de datos GeoDB (3) y realizar el macheo de los atributos del evento y la zona geográfica contra los filtros y el área geográfica de la suscripción. El mecanismo de matching se verá en detalle en la sección 4.3.3.
- En (4), para cada mensaje, se retorna al GeoMOM una instancia del evento por usuario para cada suscripción macheada. Estos mensajes se utilizaran posteriormente para notificar a los usuarios.
- En (5), se realiza la distribución de los mensajes utilizando el patrón PS. Cada topic hace broadcast de los eventos a cada uno de los suscriptores. Los suscriptores son colas de mensajes (queues) que residen en el broker de mensajería y que representan el conjunto de los mensajes a notificar de cada usuario final (6). Como se aprecia en la figura del lado derecho, cada cola almacena diferente cantidad y tipos de eventos en un momento dado.

El GeoMOM esta hosteado en un Execution Environment provisto por la PGE. Se tomó esta decisión de forma de encapsular el acceso al GeoMOM, de forma que tanto organismos como clientes no puedan acceder directamente al servidor del GeoMOM y delegando la seguridad del acceso al sistema de seguridad de la PGE.
El GeoMOM expone SOAP endpoints que permiten recibir los pedidos de registro de topics y listado de topics de la PGE procedentes de los organismos y otros endpoints para recibir los pedidos de registro de usuario, listado de topics y suscripciones de los usuarios del broker de mensajería, procedentes de los usuarios.

4.3.2. Métodos invocados sobre el GeoMOM

Los métodos invocados sobre el GeoMOM son: Registro de Topic, Listas Tópicos y Publicar Evento, todas ellas originados en los proveedores (Figura 4.7).

4.3.2.1. Publicar evento

Los proveedores de datos son los que publican los eventos con geometría asociada sobre la plataforma a través de este método. Estos eventos son redirigidos al GeoMOM para su procesamiento.
Este método recibe el evento en cuestión como parámetro que debe corresponder a una estructura base tipo JSON que se puede apreciar en la Figura 4.8. Esta estructura base puede ser extensible. En el método "Registrar Topic", cada proveedor publica que estructuras de eventos va a publicar, respetando la estructura base y pudiendo agregar campos adicionales. Estos campos adicionales pueden llegar a ser utilizados por la UI de la aplicación cliente para renderizar otros datos al usuario, ya sea en formato texto o en un mapa.

```
{
        tipo: String,
        fecha: Date,
        geo_localizacion:{
                punto: String
                poligono: List of String,
                polilinea: List of String,
        },
        prioridad: Enumerated,
        titulo: String,
        descripcion: String,
        timeout: Integer,
        ... // campos de extension de E.B.
}
```

Figura 4.8: Estructura base de evento

En el campo geo_localizacion se almacena la geometría del evento, que puede ser un punto o un polígono. En la Figura 4.9 se detallan el tipo de dato y descripción de cada campo.

Campo	Tipo de dato	Requerido?	Descripción
tipo	String	Si	Indica el tipo del evento. Los tipos de eventos sugeridos son Incendio, Accidente Vial, Estacionamiento, Vivienda, Local Comercial, Restaurante, Rutas, Meteorología, Crimen, Locomoción, etc.
fecha	Date	Si	Fecha que ocurre el evento
geo_localizacion. punto	String	No*	Representa las coordenadas de un punto en el espacio geográfico
geo_localizacion.polígono	Lista de String	No*	Representa una lista de coordenadas de un polígono en el espacio geográfico
geo_localizacion.polilínea	Lista de String	No*	Representa una lista de coordenadas de una polilínea en el espacio geográfico
prioridad	Enumerado	Si	Define los tipos posibles de prioridad de un evento: Dato Informativo, Advertencia, Alerta y Alerta Urgente
titulo	String	Si	Define título o nombre del evento.
descripción	String	Si	Breve descripción del evento
timeout	Integer	No	Indica un tiempo en segundos en el cual tendrá validez el evento desde su creación

Figura 4.9: Descripción de los campos de la estructura base

El GeoMOM es responsable de validar la correctitud de los mensajes contra la estructura base o alguna de las extensiones de esta. Si el mensaje está mal formado, entonces el evento es rechazado. En caso contrario, se verifica que exista un topic para el tipo del evento publicado. Dicha tarea es responsabilidad del componente Validador de Eventos (Figura 4.6).
Como poscondición de este método, el GeoMOM almacena el evento en el topic del tipo del evento.

4.3.2.2. Registrar Nuevo Topic

En la operación descrita en la subsección 3.4.1.2, se mencionó la posibilidad del proveedor de registrar una estructura de evento (metadatos) de algún tipo de evento a publicar. Todos los eventos

que luego correspondan a este tipo, deberán respetar esa estructura base y serán almacenados en el topic de ese tipo. Este método recibe como parámetros los atributos especificados en la Figura 4.10.

```
{
        proveedor_id: String,
        nombre_evento: String,
        canal: Enumerado,
        atributos: Lista de {nombre:String, tipo:String, requerido: Boolean, filtrable:Boolean, valores_enum:Lista de String}
}
```

Figura 4.10: Estructura de mensaje de registro de topic

El método recibe tres parámetros bien diferenciados:

- ProveedorId: identificador del proveedor. Es requerido.
- Nombre_evento: nombre del evento a publicar (ejemplo: "Inundación"). Es requerido.
- Canal. Parámetro que identifica la forma de notificación del evento: SMS, Mail o Web Socket (ver sección 4.4.2.4.2)
- Atributos: lista de atributos que extienden la estructura base. Es opcional. Para cada atributo, se indica el nombre, tipo de dato, si es requerido y si es filtrable por suscripción.

En caso de que el campo "atributos" venga vacío, se tomara la estructura base como estructura del evento. Por ejemplo, si se desea registrar el topic "Inundación" y agregar, por ejemplo, el dato de cantidad de metros del sobre caudal de agua, como dato informativo adicional, se podría invocar el método de esta forma:

```
ProveedorId="INUMET"
Canal= TipoCanal.Mail
Nombre_evento="Inundación"
Atributos=[{nombre:"Metros_Cauce", tipo: "Integer"}]
```

Para el tipo de evento orientado al público, podemos definir varios subtipos asociados (obra teatro, carrera, marcha, etc.) y hacer filtrable por suscripción este subtipo. Un registro posible de este tipo de evento es el siguiente:

```
ProveedorId="IMM"
Canal= TipoCanal.WebSocket
Nombre_evento="Publico"
Atributos=[{nombre:"Subtipo", tipo: "Enumerado", valores: ["ObraTeatro",
"Carrera", "Curso", "Marcha", ..., filtrable: true]}]
```

Los pedidos de registro de tipos de eventos son recibidos por la PGE y redirigidos al GeoMOM. Los atributos adicionales que se pueden agregar al tipo del evento permiten realizar filtros en las suscripciones sobre esos campos de forma de refinar el conjunto de notificaciones deseadas por el usuario.

4.3.2.2.1. Propuesta de extensiones para algunos tipos de eventos

Se pueden diferenciar múltiples estructuras de evento de publicación que extienden la estructura base de la Figura 4.8 agregando nuevos campos. Cada estructura depende de los tipos de eventos discutidos en la sección 3.2. En esta sección, se proponen extensiones de la estructura base para algunos tipos de evento.

Eventos de criminalidad

En los sucesos de criminalidad, podría ser interesante para el usuario visualizar la zona de riesgo (barrio o municipio) en que se produjo el evento en un mapa, que coincida con alguna zona de inseguridad. De esta forma, la localización puntual del evento debe pertenecer a la zona de inseguridad registrada por el proveedor correspondiente (Intendencias por ejemplo) y en el evento se puede adjuntar los límites de dicha zona. De esta forma, la extensión se manifiesta en la Figura 4.11.

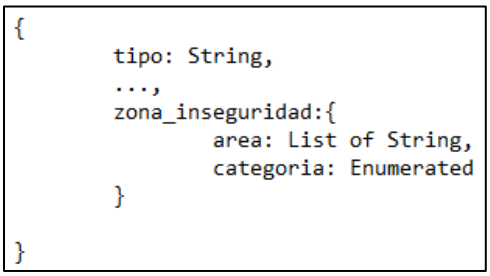

Figura 4.11: Estructura extendida del evento de criminalidad

En el campo zona_inseguridad (opcional), se añade el área de inseguridad representada por un polígono y la categoría de la zona. La categoriza puede ser zona alto riesgo, mediano riesgo o bajo riesgo. Dado el evento e, se debe cumplir que:

e.geo_location within e.zona_inseguridad.area

Eventos relacionados al público

Los eventos asociados al público refieren a aquellos eventos donde se realiza alguna participación multitudinaria, como por ejemplo: marchas de protesta, partidos de futbol, cursos temáticos, obras de teatro, carreras de bicicleta o de pie, exposiciones artísticas, conciertos musicales, etc. Todos estos subtipos del evento deberían aparecer como dato del evento. En base a este análisis, a la estructura base podrían agregarse los campos de la Figura 4.12. En el campo subtipo, se indica que tipo de evento al público se refiere específicamente.

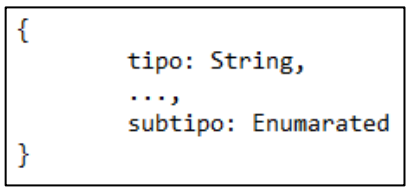

Figura 4.12: Estructura extendida del evento al público

Eventos de locales comerciales

Los eventos asociados a locales comerciales podrían tener asociada la identidad del mismo. Dicha identidad puede ser brindada por el nombre de la firma que abre el local y la dirección de domicilio de la sucursal. Ambos atributos pueden ser agregados en la extensión de la estructura base, como se visualiza en la Figura 4.13. De esta forma, ante la apertura de un nuevo local comercial o de proximidad a cierto comercio de interés, se informan estos dos nuevos datos.

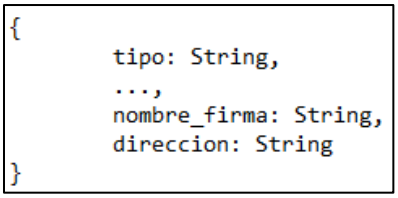

Figura 4.13: Estructura extendida del evento orientado a locales comerciales

4.3.2.2.2. Ejemplos de publicaciones de eventos

En la tabla siguiente se ilustran ejemplos para algunos eventos descritos con las estructuras previas. Los valores de los campos están expresados en alto nivel.

Tabla 3: Ejemplos de eventos

Tipo evento	**Mensaje**

Crimen	tipo: "Inseguridad"
	fuente: "Ministerio del Interior"
	geo_localizacion.punto: "-34.8114755,-56.1470241"
	prioridad: TipoPrioridad.Advertencia
	título: "Rapiña en Piedras Blancas"
	descripción: "Ocurrió a las 18:30 Hs del día viernes 16/10/2016. La víctima fue una mujer de 35 años. Le robaron su cartera y su celular y no sufrió ninguna agresión."
	timeout: 5*24*3600 segundos// segundos = 5 días
	zona_inseguridad: {
	categoría: "Zona Roja",
	área: ["(-34.8114755,-56.1470241)", "(-34.8084655,-56.1578855)", "(-34.8011655,-56.1571052)", "(-34.81021,-56.1799867)"
	}

Meteorología	tipo: "Meteorología" fuente: "INUMET" geo_localizacion.area: "Departamento de Montevideo" prioridad: TipoPrioridad.Alerta título: "Se pronostican chaparrones de granizo y rachas de viento "muy intensas y destructivas (con características de temporal) de corta duración y efectos muy localizados en las próximas 24 horas en las zonas del este y sur del país" timeout: 2*24*3600 segundos// = 2 días
Publico	tipo: "Publico" subtipo: "Carrera" fuente: Intendencia de Montevideo geo_localizacion.polilinea: ["(-34.8114755,-56.1470241)", "(-34.908451, -56.133583)", "(-34.908204, -56.133534)", "(-34.906995, -56.133295)", "(-34.906171, -56.132980)", "(-34.905592, -56.132783)", "(-34.904712, -56.132233)", "(-34.904251, -56.131049)"] prioridad: TipoPrioridad.Informacion título: "Carrera 5K en Rambla Armenia" descripción: "Carrera de 5 kilómetros por Rambla Armenia. Más de 500 personas inscriptas están corriendo a 30 minutos de su inicio." timeout: 3*3600 segundos// = 3 horas
Incendio	tipo: "Incendio" fuente: "SINAE" geo_localizacion.punto: "-34.818484, -56.1400232"

```
prioridad: TipoPrioridad.Advertencia
título: "Incendio en complejo en calle Camino Carrasco"
descripción: "Fuerte incendio registrado en un complejo de viviendas
en Camino Carrasco hace 20 minutos. Dos gatos murieron a causa del
siniestro, sin embargo, ninguno de los 10 residentes de las cinco
unidades del complejo estaban en la propiedad o resultaron
lesionadas, se informó."
timeout: 6*3600 segundos// segundos = 6 horas
```

4.3.2.3. Listar Topics

Este método permite a un nuevo proveedor de datos listar los topics disponibles y previamente dados de alta en el GeoMOM por otros proveedores. Para cada topic en el GeoMOM, se devuelve el nombre del tipo del evento asociado al topic y la estructura de metadatos del mensaje que debe satisfacer los eventos que se publiquen con este tipo. Este método permite al proveedor enviar mensajes a tópicos ya existentes, sin la necesidad de crear tópicos repetidos. De esta forma, se asegura el uso compartido de tópicos entre múltiples proveedores de datos.

4.3.3. Matching de suscripciones

Cuando llega un nuevo evento a un topic del GeoMOM, el componente motor de procesamiento se encarga de generar los mensajes de notificación de cada evento de ese topic por cada cliente. Este componente es el encargado de realizar el matching suscripción-evento.
Para cada caso, según la geometría del evento y la de la suscripción, el matching geométrico se puede realizar de diferentes formas. El mismo depende de la operación geométrica a utilizar según los tipos de geometrías de ambos. En la tabla siguiente se pueden ver las operaciones utilizadas para cada par de geometrías. Se toman como referencia las geometrías y operaciones del SFA descritas en el capítulo 2. Llámesele *LOC* a la localización del usuario, *Radio* al tamaño del buffer de suscripción y *matching_geométrico* a la operación de matching respectiva.

Tabla 4: Operaciones del matching geométrico

Geometría A: Evento	**Geometría B: Suscripción**	**Matching_geométrico**
Punto	Buffer	A Within Distance(*LOC*, *Radio*)
Punto	Polígono	A Within B
Polilínea	Buffer	A Crosses Distance(*LOC*, *Radio*)
Polilínea	Polígono	A Crosses B
Polígono	Buffer	A Intersects Distance(*LOC*, *Radio*)

Polígono	Polígono	A Intersects B

En segunda instancia, deben satisfacerse otras condiciones para que el usuario sea notificado del evento. Se deben satisfacer los filtros que se corresponden a los parámetros adicionales, según lo especificado en la sección 4.3.2.2. Estos están sujetos a la extensión de la estructura base del topic y sobre la cual se hubieran especificado propiedades "filtrables". Entonces, en conclusión, dado un evento E y una suscripción S, para que el macheo sea exitoso las tres afirmaciones siguientes deben ser afirmativas y en el siguiente orden:

1. El tipo del evento debe ser igual al tipo de evento de la suscripción:
   E.tipo = S.tipo
2. Dada la estructura de E, para cada propiedad p' adicional filtrable en ET, se debe cumplir que:
   E.p' = S.p'
3. Si las dos condiciones anteriores son verdaderas, debe resultar el macheo entre las geometrías, cumpliéndose que:
   matching_geometrico (E.geo_localizacion , S.area_subscripcion)

4.3.4. GeoDB

La GeoDB es la base de datos que utiliza el GeoMOM para gestionar sus datos. Es una base de datos relacional geográfica, encargada de la persistencia de suscripciones de los usuarios y tipos de eventos publicados por los organismos. En la Figura 4.14 se puede apreciar el modelo entidad-relación que se asume en la GeoDB.

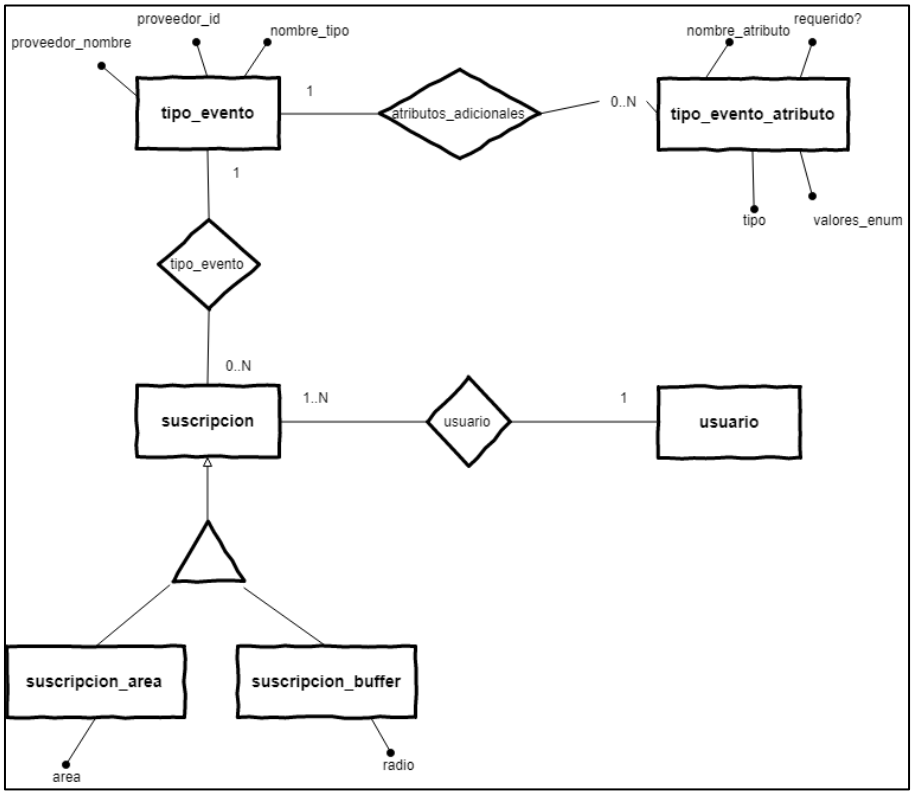

Figura 4.14: Modelo Entidad Relación de la GeoDB

En el modelo, se pueden observar las siguientes entidades:

- Suscripción. La entidad suscripción almacena las suscripciones dadas de alta. Las suscripciones tienen un usuario asociado y cada usuario puede tener múltiples suscripciones registradas. Las suscripciones se pueden clasificar en suscripción de área y en suscripción de buffer (categorización debajo de suscripción). La primera tiene asociado un polígono que representa un área geográfica y la segunda tiene un atributo numérico que indica el radio del buffer. Por otro lado, cada suscripción está asociada a un tipo de evento. Se da de alta en el método Suscribirse a Topic.
- Usuario. Son las entidades que registran las suscripciones. Un usuario pueden tener alguna suscripción registrada a su nombre. Se da de alta en el método Registrar Usuario.
- Tipo de evento. Cada tipo de evento corresponde a las categorías de eventos que pueden ser notificados. Tiene asociado un identificador y nombre de proveedor y un nombre del tipo. Si la relación atributos_adicionales es vacía, entonces el tipo de evento debe ser notificado utilizando la estructura base (sección 4.3.2.1) o, en caso contrario, deben agregársele los atributos adicionales especificados en la entidad tipo_entidad_atributo. Cada registro de esta entidad contiene metadata de dichos atributos, como el nombre del atributo, tipo de dato y si es requerido o no para los filtros de la suscripción. En caso de que tipo_entidad_atributo sea de tipo Enumerado, deben incluirse la cadena de valores en el campo valores_enum.

Por cada registro exitoso de tipo de evento, existirá un topic en el GeoMOM al cual lleguen todos los eventos publicados de ese tipo.
El atributo suscripcion_area.area es una geometría en formato GM que corresponde a un polígono en el espacio. En las consultas con operaciones geométricas se utilizó el estándar SFS (sección 2.2.1) que permite hacer consultas SQL sobre datos geográficos. De esta manera, se pueden ejecutar las operaciones geométricas acordes a los macheos geométricos dependiendo de cada par de geometrías evento-suscripción.

4.4. Broker de mensajería

El broker de mensajería es un componente intermedio entre el GeoMOM y los usuarios finales. Tiene diversas responsabilidades, actuando básicamente como un proxy entre los usuarios y el GeoMOM.

4.4.1. Componentes

El broker de mensajería cuenta con los componentes y estructuras de la Figura 4.15. Recibe las invocaciones de los métodos de los usuarios y los redirige al GeoMOM, pudiendo realizar alguna transformación de los mensajes para adecuar a los formatos que espera recibir el GeoMOM.
Expone diferentes SOAP endpoints para recibir los pedidos SOAP de los usuarios relativos a registros de usuario, suscripciones a topics y listado de topics.

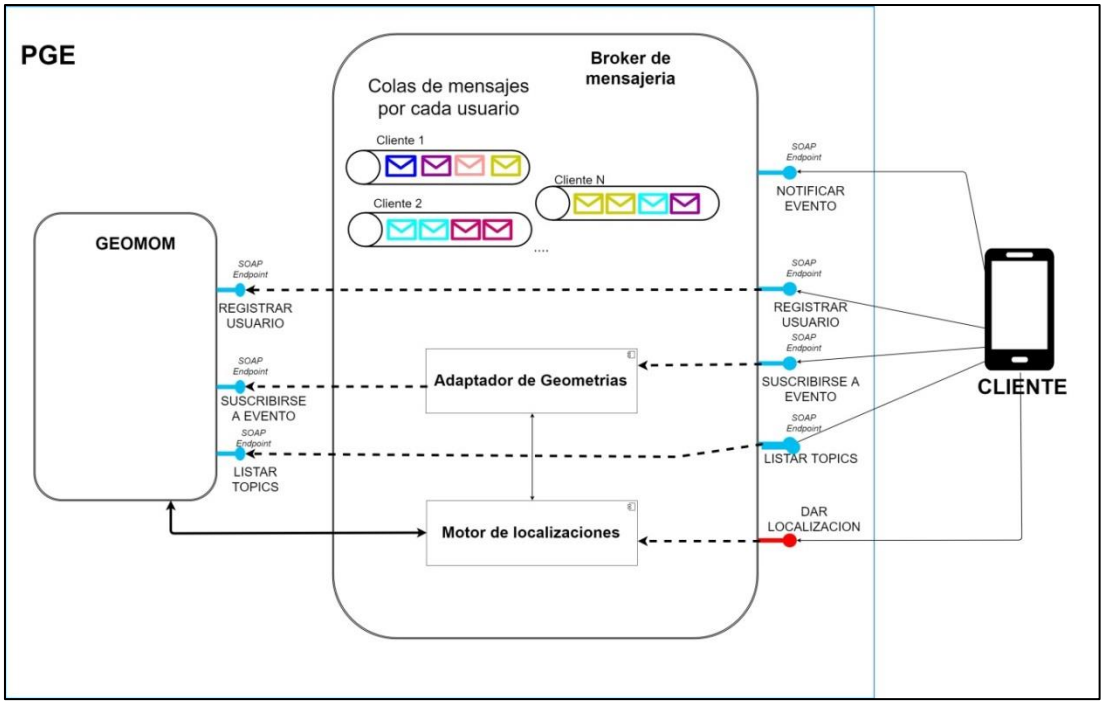

Figura 4.15: Arquitectura del Broker de Mensajería

4.4.2. Métodos invocados sobre el broker

Los métodos invocados sobre el broker son: registrar usuario, listar topics, suscribirse a topic y notificar evento.

4.4.2.1. Registrar usuario

Este método se invoca por la aplicación cliente cuando un nuevo usuario desea registrarse en la plataforma desde su dispositivo móvil. El pedido de registro de usuario se redirecciona al GeoMOM, invocando el SOAP endpoint correspondiente utilizando WS-Addressing. El método devuelve un mensaje de éxito si el registro es correcto, o un mensaje de error por el contrario.
En caso de éxito, se genera una entidad de nuevo usuario en la GeoDB que representa al nuevo usuario creado. Por otra parte, se genera una nueva cola de mensajes para el usuario creado en el broker, que permitirá recibir los mensajes de los topics del GeoMOM. A partir de este momento, el usuario podrá realizar suscripciones y recibir notificaciones del broker de mensajería.

4.4.2.2. Listar Topics

Este método permite al usuario listar los topics disponibles y previamente dados de alta en el GeoMOM por los proveedores, sobre los cuales se pueden hacer suscripciones. Para cada topic en el GeoMOM, se retorna el nombre del tipo del evento asociado al topic.
Los pedidos de listados son redirigidos al GeoMOM a través de WS-Addressing igual que en Registro de Usuario.

4.4.2.3. Suscribirse a Topic

Los usuarios realizan las subscripciones de sus preferencias en la PGE a través del método Suscribirse a Topic. Para ello, indican el tipo de evento (topic) del cual desean recibir mensajes y valores en los campos de extensión de la estructura base del evento que se hubieran definido como filtrables. Una vez confirmados los parámetros, la aplicación cliente envía un mensaje de suscripción de tipo JSON que se puede visualizar en la Figura 4.16. Junto al tipo de evento, se debe agregar el identificador del usuario que realiza la suscripción y opcionalmente el canal correspondiente. El tipo de canal indica el modo en que los mensajes de esta suscripción serán notificados. El canal es alguno de los indicados en la sección 4.3.2.2. De esta forma, se declara el canal de notificación sobre las notificaciones de la suscripción. El valor de este campo solo se tomara en cuenta para notificaciones push (ver sección 4.4.2.4.2).

```
{
        usuario_id: String,
        tipo_evento: String,
        canal: Enumerado,
        area_suscripcion: {
                buffer: Double
                area: List of String,
                identificacion_area: String
        }
}
```

Figura 4.16: Estructura de mensaje de una suscripción

En el campo area_suscripcion se almacena la geometría deseada de la suscripción geográfica, que puede ser un polígono, buffer o el identificador de una zona geográfica específica. En la Figura 4.17 se detallan el tipo de dato y descripción de cada campo.

Campo	Tipo de dato	Requerido ?	Descripción
usuario_id	String	Si	Identificador del usuario que hace la suscripción
canal	Enumerado	No	Forma de notificación de la suscripción: Web Socket, Mail o SMS
tipo_evento	String	Si	Tipo de evento
canal	Enumerado	No	Tipo de canal de notificación: pull, push
area_subscripcion.buffer	Double	No*	Tamaño del buffer sobre el cual recibir eventos desde la localización del usuario
area_subscripcion.área	Lista de Strings	No*	Cada entrada de la lista es un par de coordenadas en el espacio geográfico y representa un vértice del polígono de esa área.
area_subscripcion. identificación_area	String	No*	Identificación de una división administrativa particular: ciudad, barrio, departamento, etc.
* Uno y solo uno de estos tres campos debe estar definido mandatoriamente			

Figura 4.17: Tipos de datos de los campos de la suscripción

En el caso del campo identificacion_area se almacena un texto que se utilice para representar un área geográfica específica (barrio, ciudad, departamento, etc.). Ejemplo para representar la ciudad de Montevideo: "Montevideo, Departamento de Montevideo, Uruguay"; ejemplo para representar el barrio Malvín: "Malvín 11400, Montevideo, Departamento de Montevideo, Uruguay".

4.4.2.3.1. Parámetros adicionales

Como ya se vio, según el tipo de evento la estructura del mensaje puede variar, siempre extendiendo de la estructura base. Para las estructuras extendidas, el usuario podría hacer la suscripción sobre las propiedades adicionales filtrables (es decir, agregar filtros sobre esas variables). De las extensiones propuestas en la sección 4.3.2.2.1, cabe destacar:

- Eventos asociados a locales comerciales. Se puede incluir en la suscripción el nombre de la firma del local comercial de interés y el rubro al que pertenece, de acuerdo a 4.2.1.1.5. El usuario solo recibirá los eventos de locales comerciales que sean del rubro y nombre que se defina.
- Eventos al público. Se puede incluir el tipo del evento al público, según la estructura de 4.2.1.1.3. El usuario solo recibirá los eventos del subtipo definido.

4.4.2.3.2. Ejemplos

En la tabla siguiente se pueden ver ejemplos de suscripciones modelo a diferentes eventos. Los valores de los campos están expresados en alto nivel.

Tabla 5: Mensajes de suscripciones

Tipo de evento	Mensaje

Incendio	usuarioId: "user1@pge.com",
	tipo_evento: "IncendiO",
	canal: TipoCanal.Mail,
	area_subscripcion: {
	buffer: 300// metros
	}

Meteorología	usuarioId: user2@pge.com,
	tipo_evento: "Meteorología",
	canal: TipoCanal.SMS,
	area_subscripcion: {
	identificacion_area: "Departamento de Canelones"
	}

Publico	usuarioId: user2@pge.com, tipo_evento: "Publico", Subtipo: "ObraTeatro" area_subscripcion: { área: ["-34.8741351,-56.1761602","-34.883746, -56.188091", "-34.893392, -56.169122", "-34.884169, -56.163929"]

4.4.2.3.3. Poscondiciones

El broker se encarga de tomar los mensajes de nuevas suscripciones y procesar los campos geométricos, parseando su contenido y transformándolos en el componente Adaptador de Geometrías (Figura 4.15). Son los casos de:

- area_subscripcion.area y area_suscripcion.punto, que representan un polígono y punto respectivamente cuyo formato de origen es un lista de strings o un string. El Adaptador obtiene el valor de los campos y los convierte a formato GML (Geography Markup Language).

- area_subscripcion.identificacion_area que identifica una zona particular. El broker aplica alguna técnica de geo codificación para obtener la geometría geoespacial de esa zona. Finalmente, el Adaptador sustituye el valor de este campo por el de la geometría en formato GML.

El pedido de suscripción se redirecciona al GeoMOM, invocando el SOAP endpoint correspondiente. En el GeoMOM se genera una nueva entidad de suscripción en la GeoDB (Figura 4.14) de la categoría correspondiente (área o buffer).

4.4.2.4. Notificar Evento

En este método el usuario es notificado de los eventos de acuerdo a sus suscripciones previas y, dependiendo del caso, su localización actual. El mensaje de notificación se puede apreciar en la Figura 4.18.

```
{
        usuario_id: String,
        fecha_notificacion: Date,
        canal: Enumerado,
        area_suscripcion:{
                buffer: Double,
                identificacion_area: String,
                area: Lista de String
        },
        evento: ... // el evento en cuestión
}
```

Figura 4.18: Estructura de un mensaje de notificación

Los parámetros que recibe el usuario son:

- UsuarioId: identificador del usuario notificado
- Fecha_notificacion: fecha exacta en que se produce la notificación
- Area_suscripcion: área de suscripción en la que se produce el evento. Esta área es aquella que corresponde a alguna suscripción dada de alta por el usuario, ya sea un buffer relativo, área arbitraria o área geográfica predefinida.
- Canal. Canal definido para el mensaje. Si la suscripción tiene definido el canal, se copia este valor al parámetro. Si no está definido en la suscripción, se aplica el canal definido al crear el topic (sección 4.3.2.2).
- Evento. Todo el evento que fue publicado por el proveedor. En este campo se almacena la estructura completa del evento para el tipo de evento de notificación. Dicha estructura debe estar dada de alta en el método "Registrar Topic".

En la siguiente tabla se presentan un ejemplo de notificación para meteorología.

Tabla 6: Ejemplo de notificación

```
usuarioId: user1@pge.com,
fecha_notificacion: 27 Dic 2012,
canal: TipoCanal.Mail,
area_subscripcion: {
buffer: 300// metros
},
evento:{
tipo: "Meteorología"
fuente: "INUMET"
geo_localizacion: "Departamento de Montevideo"
prioridad: TipoPrioridad.Alerta
título: "Se pronostican chaparrones de granizo y rachas de viento "muy intensas y
```

```
destructivas (con características de temporal) de corta duración y efectos muy
localizados",
timeout: 2*24*3600 segundos// segundos = 2 días
}
```

Como se vió en la Figura 4.7, el broker tiene colas de mensajes que representan el conjunto de mensajes sin notificar para cada usuario. Estas colas de mensajes se cargan con los mensajes de los topics del GeoMOM bajo la modalidad PS. Estos mensajes deben ser entregados a la aplicación de los usuarios finales que se encargara de desplegarlos correctamente.
Los mensajes que tengan asociado un timeout (atributo timeout definido), serán descartados y eliminados de las colas de mensajes cuando la fecha actual sea mayor a la fecha de generación del evento sumado a la cantidad de segundos del timeout. En consecuencia, estos mensajes no serán notificados a los usuarios.
La entrega de los mensajes se puede realizar bajo las dos modalidades LBS: notificación pull o notificación push. En la Figura 4.19 se muestra los flujos de mensajes para ambos tipos de notificaciones.

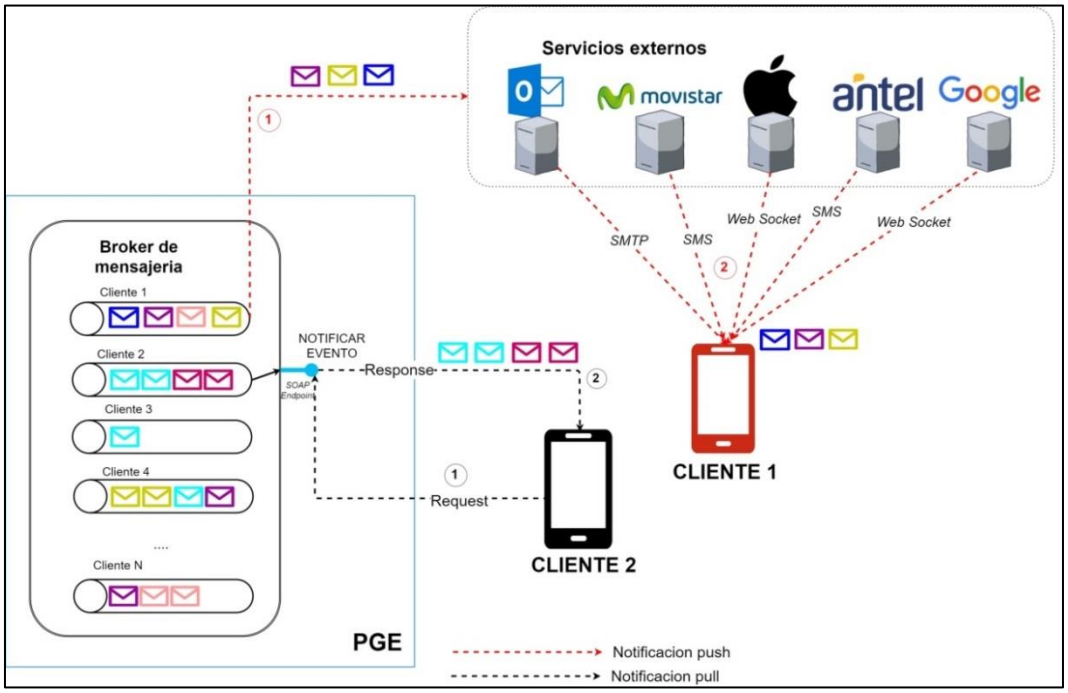

Figura 4.19: Mecanismos de notificaciones

4.4.2.4.1. Notificaciones pull

Según la Figura 4.18, la aplicación cliente se comunica con el broker solicitando el envío de nuevas notificaciones. Por defecto, los mensajes que se consuman de esta forma serán aquellos de baja prioridad y de carácter informativo, que responden a eventos de restaurantes cercanos, obras de teatro, exposiciones artísticas o planes de vivienda (cuando el canal es de tipo default).
Cuando el usuario desea recibir las notificaciones, la aplicación cliente invoca el método Notificar Evento del broker. El broker se encarga de tomar los mensajes de la cola de mensajes que representa al usuario y de enviarlos directamente a la aplicación. La aplicación se encarga de recibir los mensajes y listarlos en pantalla.

4.4.2.4.2. Notificaciones push

Bajo esta modalidad, el broker hace push de las notificaciones en las aplicaciones de los dispositivos móviles. Esto permite recibir notificaciones sin una solicitud previa y explicita del usuario. Por defecto, estas se utilizan para hacer notificaciones en tiempo real en relación a eventos de media y alta prioridad, como ser de alertas de emergencia asociados a meteorología, incendios, inundaciones, siniestros de tránsito, etc. En general, las notificaciones push se producirán sobre aquellos eventos con prioridad no informativa (es decir, alertas o advertencias).
El broker se encarga de tomar los mensajes de la cola de mensajes que representa al usuario y de enviarlos a algún servicio externo de notificaciones push. Los servicios externos (Figura 4.18) pueden ser:

- Google para notificaciones push a dispositivos Android y Apple para dispositivos IOs. El mecanismo de comunicación puede ser algún protocolo de web sockets.
- Antel y Movistar para SMSs. Para SMS se suele utilizar MAP (Mobile Application Part) [77] y TCP/IP como protocolos de comunicación.
- GMail y Outlook para Mails. El protocolo utilizado es SMTP.

Según lo mencionado anteriormente, el modo de notificación puede estar declarado como parámetro en la suscripción del usuario (atributo canal) o por defecto en el topic cuando este fue registrado en la plataforma. El broker lee la propiedad canal del mensaje de notificación y deriva el mensaje, mediante alguna regla de negocio, al servicio externo correspondiente que se encargue de implementar el mecanismo de notificación. Las formas de notificación son dadas de alta y configuradas por el administrador de la plataforma.
La aplicación cliente se encarga de recibir los mensajes y listarlos en pantalla.

4.4.3. Tracking de localización de usuarios

Otra de las funcionalidades del broker es obtener la localización actual de los usuarios activos en el sistema y enviarlas al GeoMOM. Cada cierto periodo de tiempo, el GeoMOM solicita las ubicaciones de los usuarios para hacer los matchings de suscripciones. El broker otorga estas ubicaciones utilizando una de las dos políticas siguientes:

- (1) El broker obtiene la lista de todos los usuarios conectados y, para cada uno, solicita a la aplicación del cliente la ubicación del dispositivo. La aplicación la obtiene mediante algún sistema de geolocalización ya sea GPS o Cell-ID y se la da al broker. Finalmente, el broker actualiza las posiciones actuales de los usuarios conectados en el sistema. Para los usuarios inactivos no se retorna ninguna localización.
- (2) El broker devuelve al GeoMOM la última ubicación conocida de cada usuario. Para ello, retorna al GeoMOM la lista de ubicaciones previamente guardadas de cada usuario. En este esquema, la aplicación cliente envía periódicamente su ubicación actual sin solicitud previa del broker.
- (3) Se podría adoptar un enfoque hibrido de las dos políticas anteriores, que comprende devolver la última ubicación conocida para las ubicaciones reportadas por los usuarios a un tiempo no mayor a 30 minutos de la fecha actual. En caso de que el tiempo sea mayor a los 30 minutos, el broker debe solicitar la ubicación a los usuarios que se encuentren activos.

La ventaja de utilizar el enfoque (3) frente a (1) es que reduce el riesgo que se produzca un cuello de botella entre el GeoMOM y el broker cuando este último solicita todas las ubicaciones de los usuarios. Si los usuarios son muchos (> 1 millón) el broker podría tardar un tiempo considerable en trackear todas las posiciones de los dispositivos, creando un delay importante en refrescar las posiciones en el GeoMOM para hacer el matching. Por otro lado, usar (2) evita el cuello de botella de (1) pero hace

que el broker pueda llegar a devolver localizaciones muy desactualizadas y para las cuales no tenga sentido realizar los macheos. Esto incurriría en realizar notificaciones incorrectas para la posición real de los usuarios si es grande la distancia a la posición guardada en el broker. Por ende, el enfoque (3) parece el más óptimo en cuanto a mejor consistencia de datos y a reducir tiempos de respuesta.
La gestión de localizaciones es realizada por el componente Motor de Localizaciones del broker (Figura 4.17). Este componente obtiene las localizaciones de los usuarios, las mantiene en memoria y las devuelve al GeoMOM. Las localizaciones pasan previamente por el componente Adaptador del broker de forma de convertirlas a formato GML.

4.5. Configuración de roles y permisos de la PGE

Para la realidad planteada, se deben registrar los servicios y métodos que pueden ser invocados en el GeoMOM y en el Broker de Mensajería. Para ello, sería conveniente agregar los métodos y servicios del GeoMOM que se pueden invocar y los perfiles y roles que utilicen esos métodos. En las tablas siguientes se propone un ejemplo de configuración del esquema de simulación de la PGE de la Figura 4.6 que se vio anteriormente.

Tabla 7: Servicios

Nombre	Dirección lógica	Dirección física
ApiGeoMOM	http://pge.red.uy:8080/geoApi	http://localhost:8255/geo
ApiBroker	http://pge.red.uy:8080/brokerApi	http://localhost:8499/broker

Tabla 8: Métodos

Servicio	Nombre método
ApiGeoMOM	publicarEvento
ApiGeoMOM	registrarTopic
ApiGeoMOM	listarTopics
ApiGeoMOM	suscribirseTopicGeoMOM
ApiGeoMOM	registrarUsuarioGeoMOM
ApiGeoMOM	notificarEventoGeoMOM
ApiBroker	listarTopics
ApiBroker	suscribirseTopic
ApiBroker	registrarUsuario
ApiBroker	notificarEvento

Tabla 9: Perfiles

Nombre	Métodos
Publicador	publicarEvento
AltaPublicador	publicarEvento, registrarTopic, listarTopics
SuscriptorPublicador	publicarEvento, suscribirseTopic, notificarEvento, registrarUsuario
Suscriptor	suscribirseTopic, notificarEvento, registrarUsuario, listarTopics

Tabla 10: Roles

Nombre	Perfiles
Admin	*
Organismo	AltaPublicador
ProveedorExterno	Publicador
Cliente	Suscriptor

Un organismo del estado puede tener el rol Organismo que le permite publicar eventos y registrar nuevos topics. Un proveedor de datos privado (CityCop, PedidosYa, etc.) podrían tener el rol ProveedorExterno al cual solo puede publicar eventos en los topics asignados. Por otra parte, el rol Cliente, correspondiente a los usuarios móviles que utilizan la aplicación de notificaciones, puede hacer suscripciones a topics, listar los topics disponibles y recibir notificaciones de eventos. Por último, existe un rol Admin que tiene acceso a todas las funcionalidades. Este podría ser un rol de prueba dado de alta por AGESIC para uso administrativo o de prueba de la plataforma.
Por otra parte, los métodos *GeoMOM son invocados por el broker de mensajería en el GeoMOM, de los que realiza el forwarding hacia este. No existe un rol asociado que permita a un agente externo invocar directamente esos métodos del GeoMOM sin pasar por intermedio del broker.

4.6. Aplicación cliente

Se dispone una aplicación para dispositivos móviles que permite al usuario realizar todas las operaciones previamente detalladas. Debe estar desarrollada en plataformas para clientes móviles de Android e IOs. Permite al usuario dar de alta suscripciones así como también recibir las notificaciones de los proveedores y rende rizar todos los datos de la notificación de forma apropiada, con la capacidad de poder visualizar en un mapa el área de interés (arbitraria, buffer o división administrativa) y la geometría del evento ocurrido (punto, polilínea o polígono). También puede desplegar las propiedades adicionales del evento que se definieron como extensión de la estructura base del evento (ver sección 4.4.2.4).
La aplicación tiene comunicación directa con el broker de mensajería, la cual se realiza a través de la PGE. Como se explicó en la sección 4.2.1.2, la aplicación debe realizar una serie de pasos para invocar un servicio en el broker de mensajería, lo cual incluye obtener un token de seguridad del STS enviarlo al proxy de la PGE utilizando encabezados de WS-Addressing. Toda esta lógica de comunicación es transparente al usuario, por lo tanto, no requiere de ninguna acción de parte de este. Además, se encarga de reportar la localización actual del usuario de forma periódica o a pedido del broker. Para que lo haga en forma periódica, el usuario debe autorizar el tracking en tiempo real.
La aplicación alberga una cola de mensajes donde se reciben los mensajes de las notificaciones, ya sea desde un servicio externo (push) o de forma directa y explicita del broker (pull). Esta cola de recepción constituye el conjunto de mensajes que forman parte de la bandeja de entrada del usuario. Por otra parte, la aplicación podría tener un histórico de los mensajes clasificados por tipo y fecha, de forma de poder utilizarlos para un posterior análisis.

4.7. Comparación con algunos trabajos relacionados

En esta sección, se hará una comparación entre algunos trabajos relacionados (sección 2.5.1) y la solución propuesta en la tesis (capítulos 3 y 4), vinculando elementos de la arquitectura y mecanismos de suscripción y de notificaciones.

El trabajo expuesto en [54] está muy relacionado a la solución propuesta en la tesis con algunas variaciones. Se propone una arquitectura donde se comunican clientes móviles y publicadores de datos a través de un MOM que implementa la capa lógica del sistema utilizando el patrón PS. Los datos publicados en la plataforma refieren a emergencias relacionadas a incendios.
Los clientes realizan suscripciones geográficas que comprenden tres parámetros: geometría base (punto, polilínea, polígono), operador espacial (Contain, Disjoint, Cross, Touch, Overlap, etc.) y un

valor de buffer numérico opcional. Por otra parte, se incluyen predicados en la suscripción, condiciones booleanas que se deben evaluar sobre los atributos del evento. El algoritmo de matching es más sofisticado que el propuesto ya que utiliza dos fases: una fase de preprocesamiento y una de matching. El objetivo es hacer más eficiente el matching previendo que el sistema escale a miles de usuarios y millones de suscripciones. En la primera fase, se lee el conjunto de suscripciones dadas de alta y se los divide en clases. Las clases se particionan por operador espacial y se le asignan índices espaciales. En la fase de matching propiamente, se ejecuta la consulta que conforman las geometrías de una misma clase, que representa el set de geometrías y el operador espacial contra la geometría del evento. La agrupación en clases permite ejecutar un máximo de 6 consultas.
El MOM se encarga de recibir las suscripciones de los clientes y utiliza un esquema de notificación push sobre los clientes móviles. En particular, el subcomponente Notification Service obtiene los eventos de los proveedores de datos y envía los mensajes de notificación. Cumple la función del broker de mensajería de la solución propuesta. El subcomponente Matching Engine se encarga de ejecutar el algoritmo de matching y se correspondería con el subcomponente GEOMOM de la solución.

El trabajo [56] propone un sistema de PS espacial (con restricciones-eventos espaciales) para servicios basados en geolocalización (LBS). Abarca algunos de los conceptos utilizados en la propuesta y, además, propone una arquitectura modelo, que relaciona clientes móviles (productores y consumidores de datos) con un MOM que se encarga de recibir los eventos y las suscripciones y ejecutar el matching geométrico.
Utiliza tres modelos: evento espacial, suscripción espacial y de notificación espacial. El primero indica que un evento conforma de propiedades clave valor, incluyendo un identificador OID, timestamp y localización puntual de un cliente móvil. En el modelo de suscripción espacial se utilizan dos operadores espaciales: *Within*, que se ejecuta sobre N regiones predefinidas en el sistema (no establecidas por el usuario); y *Distance*. Las suscripciones se realizan sobre otros clientes, en función de su localización puntual y atributos específicos. El modelo de notificación incluye dos componentes: información del evento original y de la suscripción derivada del proceso de matching.
Como variante sustancial a la propuesta, el trabajo propone la ejecución del algoritmo de matching no solo en el MOM sino también en los clientes móviles. Esto permite acelerar el proceso de matching, ya que en este caso el dispositivo no reporta su ubicación al servidor, sino que recibe un conjunto de suscripciones del servidor y ejecuta el matching localmente basándose en su ubicación actual. Si se satisface alguna suscripción, el mensaje de notificación se envía al servidor para que este lo distribuya sobre los dispositivos a ser notificados. El procesamiento local solo se circunscribe a las suscripciones Within. El resto de las suscripciones son procesadas en el MOM. Para ello, cada nodo cliente reporta su ubicación al servidor y el MOM realiza el matching en un motor de procesamiento espacial. El MOM envía las notificaciones a los clientes a través de un LBS.
Otra variación importante es que los clientes generan los eventos y no proveedores de datos externo.

En la tabla comparativa siguiente se aprecia una lista de las características comunes y diferentes de la solución propuesta a las soluciones desarrolladas en ambos artículos. Se indica si la característica se cumple o no para cada trabajo.

Tabla 11: Comparación con trabajos relacionados

Característica	Tesis	Trabajo [54]	Trabajo [56]
Variedad de proveedores de datos	X	X	

Capacidad de agregar nuevos proveedores	X		
Utiliza patrón PS	X	X	X
Mensajes estructurados	X	X	X
Operaciones espaciales SFA soportadas	Withn, Cross, Intersects, Distance	Contain, Disjoint, Cross, Touch, Overlap	Within, Distance
Eventos agrupados por topic	X		
Restricciones de buffer (operación distance)	X		X
Algoritmo de matching eficiente		X	X
Procesamiento dividido de la lógica del algoritmo en clientes y en MOM			X
Geometrías de las publicaciones	Punto, Polilínea, Polígono	Punto, Polígono	Punto
Geometrías de las suscripciones	Buffer, Polígono	Buffer, Polígono, Polilínea	Buffer
Se expone algún LBS	X		X
Utiliza plataforma de gobierno electrónico	X		
Utiliza base de datos geográfica	X	X	
Notificaciones push	X	X	X
Notificaciones push en diversos canales	X		
Notificaciones pull	X		
Flexibilidad sobre parámetros de suscripciones	X	X	
Utiliza diversos estándares de WS	X		
Autenticación de usuarios	X		X
GUI que despliega área suscripción y zona del evento	X		

5. Prototipo de la solución

En este capítulo se detallara en profundidad el diseño y aspectos de implementación de un prototipo funcional que evalué la factibilidad técnica de la solución de la tesis. Los aspectos de implementación abarcan el uso de diversas herramientas, ambientes, frameworks y elementos del área de las tecnologías de la información y que son necesarios para el desarrollo ágil, eficiente y sostenible de los componentes de software del prototipo.
El prototipo pretende brindar una prueba de concepto para todos los elementos arquitectónicos detallados en el capítulo 4 y de forma de otorgar un subconjunto de las funcionalidades listadas en el capítulo 3. Para ello, resulto importante deliberar cuales características funcionales y cuales componentes de la arquitectura incluir en el prototipo. En las subsecciones siguientes se definirán las características del prototipo y los requerimientos funcionales que abarca. Además, se hará foco en los componentes de la arquitectura que se contemplan en el diseño, las interacciones con los actores de la solución, las simplificaciones respecto a la solución general y las formas de procesamiento y persistencia de los datos.

5.1. Diseño del prototipo

El diseño del prototipo se puede apreciar en la Figura 5.1: Arquitectura general del prototipo. El prototipo aglutina las funcionalidades del GeoMOM descrito en la Figura 4.7 y en la sección 4.3.1. Se evitara el despliegue del GeoMOM en la simulación de la PGE del proyecto de grado de [76], ya que el foco principal de la tesis es sobre el sistema de mensajería con capacidades de geolocalización. Los clientes móviles se comunican con el prototipo, encargado de recibir las solicitudes de los clientes y direccionarlas al GeoMOM y de realizar la tarea de notificar los eventos. Para ello, existen diferentes subcomponentes del prototipo que sirven de interfaz para ejecutar los casos de uso de los proveedores y de los clientes.

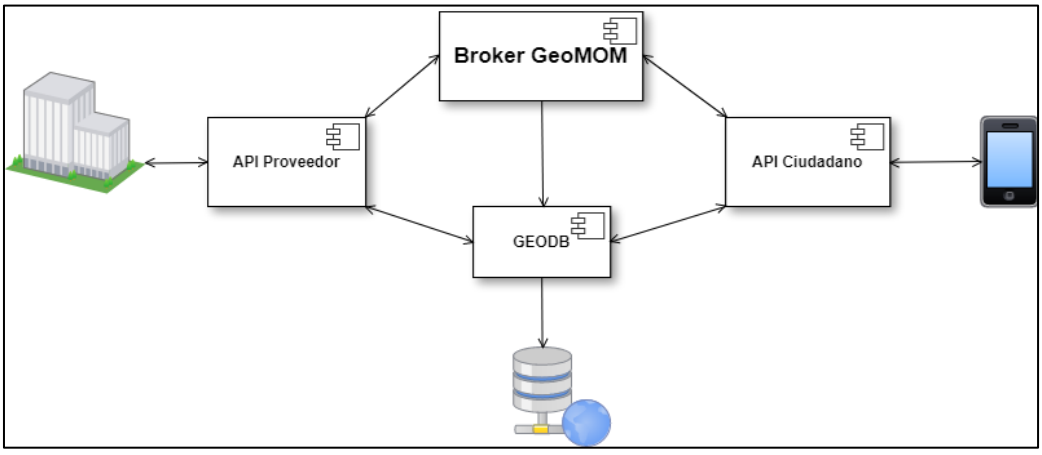

Figura 5.1: Arquitectura general del prototipo

Analizando la figura tal se desprende que hay tres subcomponentes principales que conforman el prototipo: componente API Proveedor, componente API Cliente y el GeoMOM. Todos ellos interactúan y realizan el acceso a datos a través de la GeoDB, base de datos relacional y geográfica del prototipo.

5.1.1. Componente API Proveedor

Recibe y procesa los pedidos de los proveedores de datos a través de endpoints HTTP.

5.1.1.1. API de Servicios

En este componente se exponen SOAP/HTTP endpoints para consumir diferentes servicios para ser consumidos por los proveedores de datos. Se pueden ejecutar diferentes métodos utilizando la API de servicios SOAP. En la siguiente tabla se presentan los diferentes métodos que se pueden ejecutar por el proveedor de datos.

Tabla 12: Métodos que invoca el proveedor

Método	**Parámetros**	**Salida**
registrarProveedor	OID y nombre del proveedor	Id. De proveedor
registrarTopic	Nombre del Topic Descripción del topic	
publicarEvento	Nombre del topic Id. De proveedor Descripción del evento Timeout del evento Geometría del evento: [Punto, Polígono] Coordenada/s del evento	
listarTopics		Lista de los topics creados

En todas las operaciones, la salida incluye un flag que indica si la operación fue exitosa o no y, en este caso, se devuelve un mensaje de error adecuado.

El proveedor de datos puede invocar los siguientes métodos:

- RegistrarProveedor. Recibe como parámetros el nombre y el OID (Object Identifier) del nuevo proveedor. Si la operación es exitosa, se devuelve un identificar univoco de este proveedor. La operación falla si ya existe el proveedor con la combinación OID-nombre correspondiente en el GeoMOM. Como postcondición, el proveedor puede empezar a publicar eventos en el GeoMOM.
- RegistrarTopic. Recibe como parámetros un nombre y descripción del topic a crear. Se valida que el nombre del topic no sea duplicado en el sistema.
- PublicarEvento. Recibe como parámetros el nombre del topic sobre el cual se va a publicar el evento, una descripción o cuerpo, el identificador del proveedor y opcionalmente un timeout de vida del evento. Además, recibe el tipo de geometría del evento si es puntual o abarca una zona definida. En el primer caso, se tiene que brindar las coordenadas del punto y en el segundo caso una lista de coordenadas del polígono que la representa. Se valida que el nombre del topic no sea duplicado en el sistema.
- ListarTopics. Devuelve la lista de topics creados en el sistema.

5.1.2. Componente API Cliente

En este componente se exponen SOAP/HTTP endpoints para consumir diferentes servicios para ser consumidos por los clientes móviles. Se pueden ejecutar diferentes métodos utilizando la API de servicios SOAP. En la siguiente tabla se presentan los diferentes métodos que se pueden ejecutar por el usuario ciudadano.

Tabla 13: Métodos que invoca el ciudadano

Método	**Parámetros**	**Salida**
registrarUsuario	Cedula de usuario Contraseña de usuario	
loginUsuario	Cedula de usuario Contraseña de usuario	Id. De usuario
suscribirTopic	Id. De usuario Nombre topic del evento Nombre de suscripción Tipo de suscripción [área, buffer] Si tipo=área, Lista de Coordenadas geográficas Si tipo=buffer, tamaño del buffer	
reportarUbicacion	Id de usuario Par de coordenadas geográficas	
listarTopics		Lista de los nombres de topics
notificarEvento	Id de usuario	Lista de notificaciones del usuario
listarSuscripciones	Id de usuario	Lista de las suscripciones dadas de alta por el usuario con todos sus datos.

En todas las operaciones, la salida incluye un flag que indica si la operación fue exitosa o no y, en este caso, se devuelve un mensaje de error adecuado.

El ciudadano puede invocar los siguientes métodos:

- RegistrarUsuario. Este método permite hacer el registro de un nuevo ciudadano a través de su cedula de identidad (secuencia de 6-8 dígitos sin guion) y una contraseña. Como poscondición, el usuario puede comenzar a crear suscripciones en el sistema
- LoginUsuario. Este método realiza el login de usuario. Para ello, debe proveer sus credenciales (número de cedula de identidad y contraseña) del registro de usuario. Si la operación es exitosa, el sistema le retorna un identificador del usuario que será el que se utilice para invocar el resto de las operaciones.
- SuscribirTopic. Crea una suscripción del usuario. Para ello, se debe suministrar el identificador devuelto en el login, el nombre del topic de la suscripción, un nombre particular de suscripción el cual debe ser único y el tipo geográfico de la suscripción, pudiendo elegir entre restricción de tipo buffer o zonal. Si el tipo es buffer, se debe ingresar un valor numérico que representa el radio del buffer en metros, no pudiendo ser mayor a los 1000 metros ni menor a los 100 metros. Si el tipo es zona, debe ingresarse una lista de coordenadas geográficas del Poligono asociado.

- ListarTopics. Devuelve la lista de topics creados en el sistema.
- ListarSuscripciones. Devuelve la lista de suscripciones del usuario creadas en el sistema a partir de su identificador univoco.
- NotificarEvento. Devuelve la lista de notificaciones pendientes del usuario. No es invocada directamente por el usuario: el componente se encarga de efectuar las notificaciones a los usuarios en la modalidad push
- ReportarUbicacion. Permite registrar una nueva ubicación del usuario. Esta ubicación se debe obtener del dispositivo del ciudadano utilizando algún sistema de georreferenciación (ejemplo: GPS) y se utiliza para el cálculo del matching geométrico en las restricciones de tipo buffer. Como parámetros, se otorga el identificador del usuario del login y las coordenadas geográficas de dicha posición.

5.1.3. GeoDB

Realiza el almacenamiento de datos del sistema, comprendiendo proveedores, topics, usuarios y suscripciones de los usuarios. Recibe solicitudes de acceso a datos de los componentes API y del broker GeoMOM. Ejecuta las consultas en la base de datos de las funcionalidades del sistema y las retorna al subcomponente respectivo. Es un componente interno del sistema.

5.1.4. Broker GeoMOM

Se integra por las colas de mensajes que permiten alojar los eventos publicados y los mensajes de notificación de los usuarios, así como también las estructuras que permiten cruzar las restricciones espaciales con las geometrías de los eventos. Es un componente interno del sistema.

5.2. Implementación del prototipo

En esta sección se detallaran las características técnicas y los flujos de comunicación de cada uno de los subcomponentes del prototipo, especificando tecnologías, estándares y protocolos de comunicación utilizados.
Para el desarrollo de los componentes API proveedor, Broker GEOMOM, API ciudadano y GeoDB se utilizó el entorno de programación Eclipse Mars [78]. En cada uno se utilizó un proyecto específico y con Maven como gestor de dependencias.

5.2.1. Componente API Proveedor

El componente API proveedor expone servicios SOAP para ser consumidos por los proveedores de datos.

5.2.1.1. Características técnicas

El componente API proveedor es un proyecto desarrollado en la plataforma Java 8 y al cual se lo enriqueció con el framework Spring Boot 1.5 [79].

Para la estructura lógica del proyecto, se diseñó una estructura en capas que se muestra en la Figura 5.2. La capa superior, de presentación, posee el endpoint que crea los servicios SOAP para el proveedor. La capa intermedia posee la capa lógica que recibe las solicitudes del endpoint, realiza validaciones e invoca operaciones en la GeoDB y en el broker. Esta capa está integrada por service beans de spring, componentes de negocio del framework (como los EJB de JEE). La capa inferior, que constituye la capa de acceso a datos y se implementa en proyectos separados (GeoDB y Broker GeoMOM).

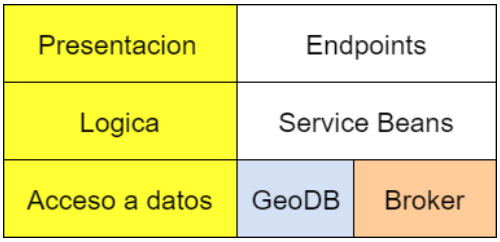

Figura 5.2: Estructura en capas del componente

5.2.1.2. Flujos de comunicación

En la Figura 5.3 se aprecian los protocolos de comunicación entre las diferentes entidades con el componente API Proveedor. Los proveedores de datos utilizan mensajes SOPA/HTTP para comunicarse con la API de servicios web del componente. A su vez, el componente se comunica con el componente GeoDB para hacer persistencia y consulta de datos. En la siguiente tabla se detalla, según cada servicio, la forma de comunicación con el resto de las entidades.

Tabla 14: Flujos de comunicación

Servicio	**Actor/componente**	**Protocolo**	**Acción**
registrarProveedor	GeoDB	TCP/IP	Se salvan los datos del proveedor en una tabla de la base de datos
registrarTopic	Broker GeoMOM	JMS	Se crea una cola de mensajes en el broker con el nombre del topic.
registrarTopic	GeoDB	TCP/IP	Se salvan los datos del topic en una tabla de la base de datos
publicarEvento	Broker GeoMOM	JMS	El componente establece una conexión con el broker para pushear el mensaje del evento en la cola de mensajes que tenga el nombre del topic especificado.
listarTopics	GeoDB	TCP/IP	Se ejecuta una consulta que devuelve los datos de los topics dados de alta
todos	Proveedor de datos	SOAP/HTTP	Interacción proveedor-componente API

La API provista por el Broker facilita la tarea de crear conexiones, sesiones y envío/recepción de mensajes del broker, así como la creación de nuevos destinos.

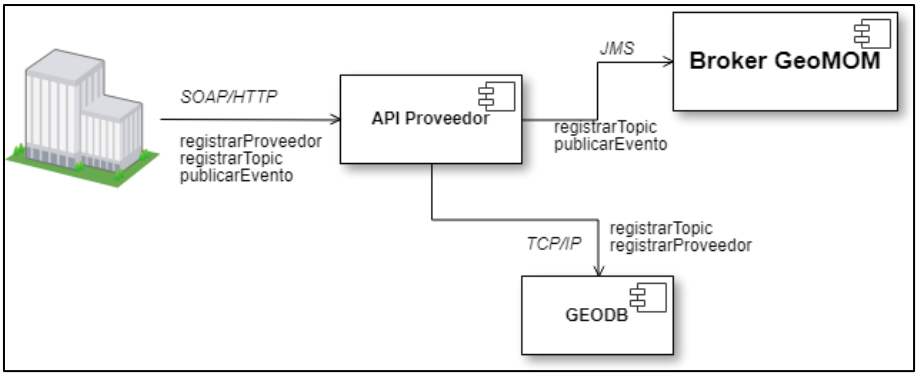

Figura 5.3: Protocolos de comunicación

5.2.2. Componente API Ciudadano

El componente API proveedor expone servicios SOAP para ser consumidos por la aplicación cliente del ciudadano.

5.2.2.1. Características técnicas

Al igual que el componente API proveedor, es un proyecto desarrollado en la plataforma Java 8 y al cual se lo enriqueció con el framework Spring Boot 1.5 [79].

Para la estructura lógica del proyecto, se diseñó una estructura en capas, de forma análoga a la del componente API proveedor que se ilustra en la Figura 5.2. La capa superior, de presentación, posee el endpoint que crea los servicios SOAP para el ciudadano. La capa intermedia posee la capa lógica que recibe las solicitudes del endpoint, realiza validaciones e invoca operaciones en la GeoDB y en el broker. La capa inferior, que constituye la capa de acceso a datos y se implementa en proyectos separados (GeoDB y Broker GeoMOM).

5.2.2.2. Flujos de comunicación

Como se describe en la Figura 5.4, se establecen los mismos protocolos de comunicación entre las diferentes entidades con el componente API Ciudadano. La aplicación ciudadano utiliza mensajes SOPA/HTTP para comunicarse con la API de servicios web del componente. A su vez, el componente se comunica con el componente GeoDB para hacer persistencia y consulta de datos y con el broker para leer los mensajes de notificaciones. En la siguiente tabla se detalla, según cada servicio, la forma de comunicación con el resto de las entidades.

Tabla 15: Servicios del prototipo

Servicio	**Actor/componente**	**Protocolo**	**Acción**
registrarUsuario	GeoDB	TCP/IP	Se salvan los datos del usuario en una tabla de la base de datos
registrarUsuario	Broker GeoMOM	JMS	Crea un canal para recibir los mensajes de notificación del nuevo usuario registrado.
loginUsuario	GeoDB	TCP/IP	Se realiza la consulta en la base de datos que

			verifica la existencia del usuario dadas las credenciales
suscribirTopic	GeoDB	TCP/IP	Verifica la existencia del topic pasado por parámetro en la base mediante consulta. Persiste la suscripción en las tablas adecuadas.
listarSuscripciones	GeoDB	TCP/IP	Lista las suscripciones del usuario en la base
listarTopics	GeoDB	TCP/IP	Se ejecuta una consulta que devuelve los datos de los topics dados de alta
todos	Ciudadano	SOAP/HTTP	Interacción ciudadano-componente API
notificarEvento	Broker GeoMOM	JMS	Devuelve las notificaciones pendientes del canal de mensajes del usuario pasado por parámetro
notificarEvento	Ciudadano	Web Socket/ HTPP	Por web socket, se envían las notificaciones por web socket al dispositivo móvil del cliente

Se utiliza la API provista por el Broker para facilitar la tarea de crear conexiones, sesiones y envío/recepción de mensajes del broker.

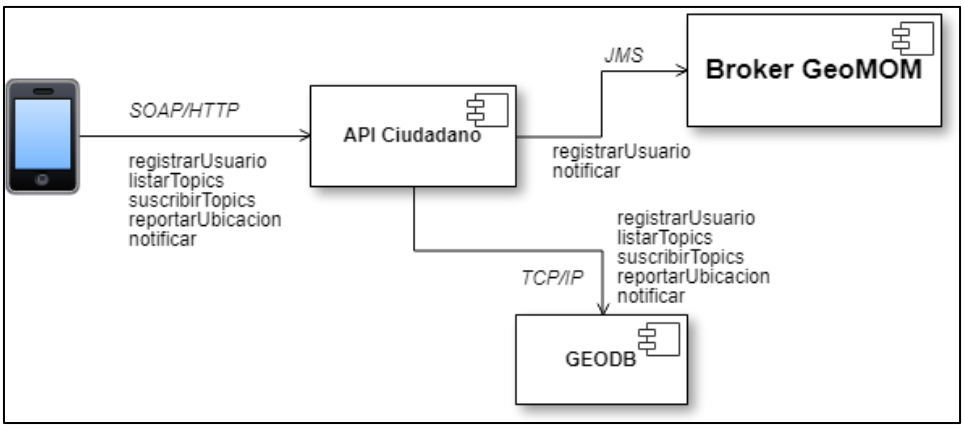

Figura 5.4: Protocolos de comunicación del componente API Ciudadano

5.2.3. GeoDB

Este componente realiza el acceso a la base de datos relacional geográfica del sistema.

5.2.3.1. Características técnicas

Se utilizó el motor PostgreSQL 9.3 [80] como base de datos, con la extensión PostGIS 2.4 [81]. A nivel de implementación, se utiliza la API de JPA de Java, que permite definir entidades y relaciones entre las entidades con anotaciones, así como definir el ORM con las tablas de la base. Para realizar las consultas y la persistencia de datos, se utilizó a nivel de API la librería de hibernate-spatial 1.1.5 [82], una extensión de hibernate que permite utilizar las funciones primitivas de PostGIS, de forma de hacer factible las operaciones de matching geométrico sobre la base de datos.

5.2.3.2. Modelo relacional

En la Figura 5.5 se aprecia el modelo relacional utilizado en el prototipo. Algunas observaciones del modelo:

- Se almacena en la tabla usuario la última localización conocida del mismo con su fecha.
- Las tablas suscripcion_area y suscripcion_buffer almacenan los datos según los tipos de suscripciones que puede hacer el usuario.

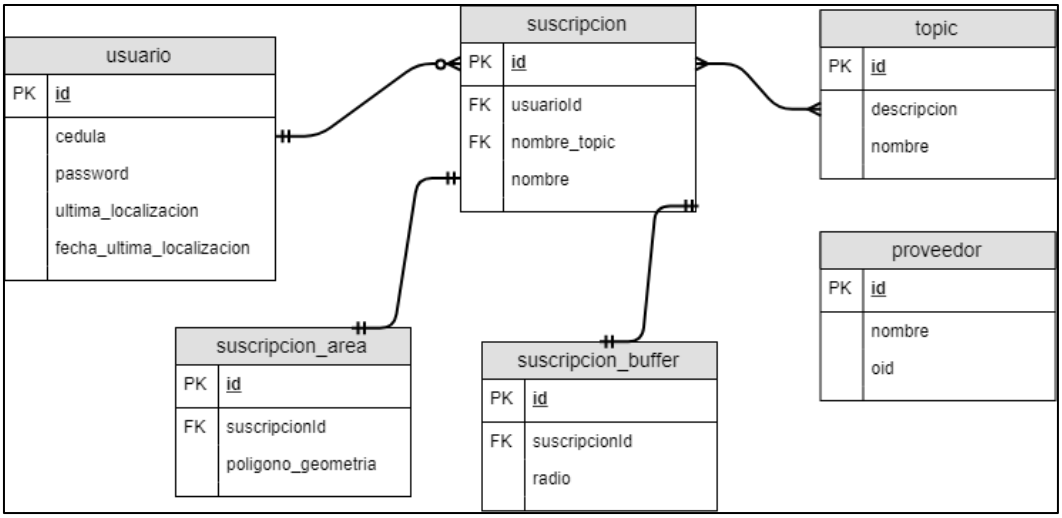

Figura 5.5: Modelo relacional de la GeoDB

5.2.4. Componente Broker GeoMOM

Este componente constituye el broker de mensajería que implementa el MOM con capacidades de geolocalización. Alberga los nuevos eventos que llegan a la plataforma a través del componente API proveedor y los mensajes pendientes de notificación de eventos de los usuarios. Su funcionalidad principal es el macheo geométrico (sección 4.3.3) de las geometrías evento-suscripciones.

5.2.4.1. Características técnicas y estructura

Es un proyecto desarrollado en la plataforma Java 8. El broker GeoMOM es una extensión del producto MOM Active MQ [52] en la versión 5.15.0 para soportar suscripciones geoespaciales.
La estructura interna del broker se puede apreciar en la Figura 5.6.

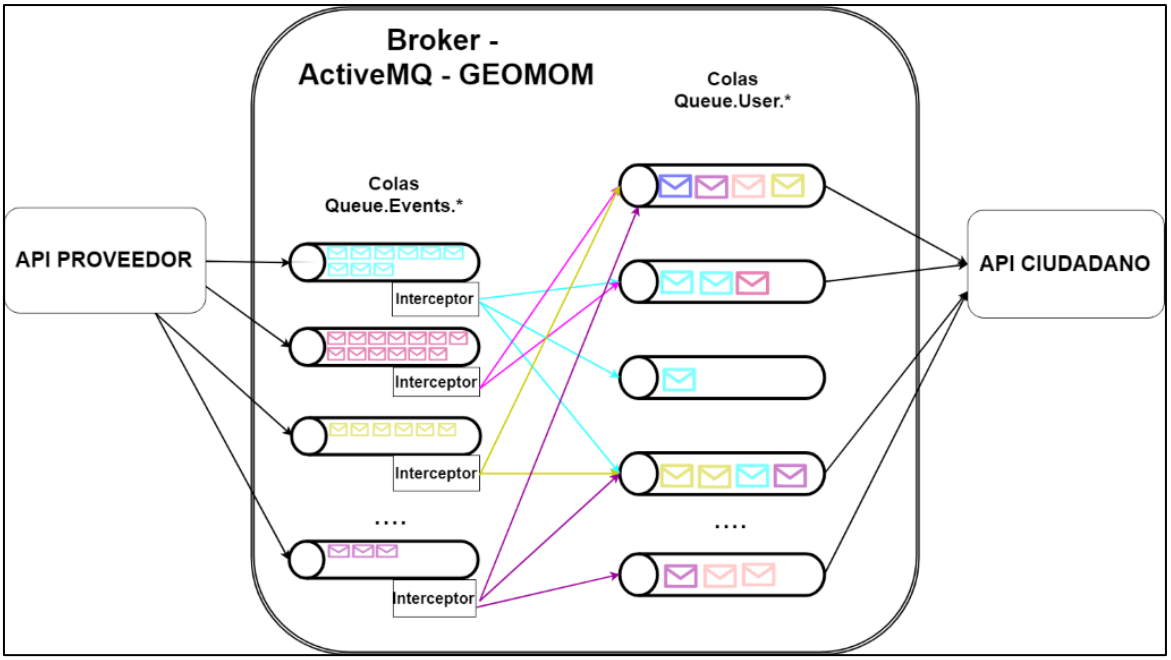

Figura 5.6: Estructura del GeoMOM- ActiveMQ

En la parte izquierda de la figura, se muestran colas de mensajes (queues de activemq) para cada topic creado con nombres del tipo Queue.Events.<Nombre_Topic>, donde Nombre_Topic refiere a un tipo de evento. Estas alojaran los nuevos eventos que vayan llegando al broker desde el componente API proveedor, de acuerdo al tipo de evento que corresponda. A la derecha de la figura, se presentan colas de mensajes que contienen el conjunto de mensajes de notificación pendientes de cada usuario. Los nombres de estas son de la forma Queue.User.<Cedula_Usuario>, donde Cedula_Usuario es la cedula de identidad del ciudadano provista al registrase con el componente API del ciudadano.
Para cada cola de eventos, se declara un interceptor de ActiveMq, que permite consumir cada mensaje que llega a la cola y procesarlo. El interceptor se encarga de leer el mensaje, extraer sus campos y geometría del evento, leer las suscripciones de los usuarios en el sistema, realizar el macheo geométrico y, para cada matching, generar un mensaje de notificación a las colas de usuario cuyas suscripciones tuvieron matching. La función de matching es una consulta SFS ejecutada en la base de datos, recibiendo como parámetro el nombre de un topic y la geometría del evento. Utiliza las funciones auxiliares st_buffer y st_intersects de PostGIS. Puede visualizarse en el anexo A.
Todo lo anterior se refleja en el pseudocódigo de la Figura 5.7.

```
interceptor(broker, topic){
        evento = topic.leerNuevoEvento()
        // obtiene todo los datos de las sucripciones donde hay maching. Ver anexo.
        machings= ejecutarMachingGeometrico(evento.geometria, evento.topic)
        para cada macheo en machings
                nuevo_mensaje = crearMensaje()
                nuevo_mensaje.destino = "Queue.User." + macheo.usuario.cedula
                nuevo_mensaje.contenido= {
                        macheo.suscripcion,
                        evento,
                        macheo.usuario,
                        fecha_actual
                }
                broker.enviar(nuevo_mensaje)
        fin para
        evento.destruir()

}
```

Figura 5.7: Pseudocódigo del matching de evento

5.2.5. Aplicación cliente

Para el prototipo, se desarrolló una aplicación cliente que permita al usuario realizar los casos de uso básicos de la solución: registrar usuario, suscribirse a topic, listar los topics, listar las suscripciones y notificar al usuario, invocando los servicios SOAP respectivos del componente API Ciudadano.
La aplicación se desarrolló en la librería React Native 0.51.0 [83] creada por Facebook Inc. Permite crear aplicaciones para dispositivos móviles de forma ágil y sencilla en lenguaje javascript ES6 y para plataformas Android e IOs. RN hace la traducción a código binario nativo en la plataforma destino, con lo cual la GUI posee el look and feel del sistema operativo en que se hace deploy de la aplicación.
Como gestor de dependencias, se utilizó npm (Node Package Manager) de Node.js [84]. Se instalaron algunas dependencias utilitarias, como por ejemplo para invocación de servicios SOAP del backend, para crear y renderizar mapas, para crear listas dinámicas de elementos y para persistencia de datos local.
Como entorno de programación, se utilizó JetBrains WebStorm [85]. Es un IDE especializado para desarrollar aplicaciones de frontend.

5.2.6. Aplicación proveedor

Para ejecutar los casos de uso del proveedor, se desarrolló una aplicación web acorde para tal propósito. Los casos de uso comprenden registrar proveedor, registrar topic y publicar un evento, invocando los servicios SOAP respectivos del componente API Proveedor.
La aplicación se desarrolló con la librería ReactJS 16.2.0 [86] creada por Facebook Inc. Se empleó JetBrains WebStorm como entorno de desarrollo.
Como gestor de dependencias, se utilizó npm (Node Package Manager) de Node.js. Se instalaron algunas dependencias utilitarias, como por ejemplo para invocación de servicios SOAP del backend, para crear y renderizar mapas y para estilización de elementos.
Además, se utilizó webpack [87] de npm de forma de cargar la aplicación sobre un contenedor web liviano e independiente, de forma de desacoplar el frontend del backend y logrando la comunicación a través de servicios web.

5.2.7. Despliegue de despliegue.

En la Figura 5.8 se ilustra un diagrama de despliegue UML del prototipo.

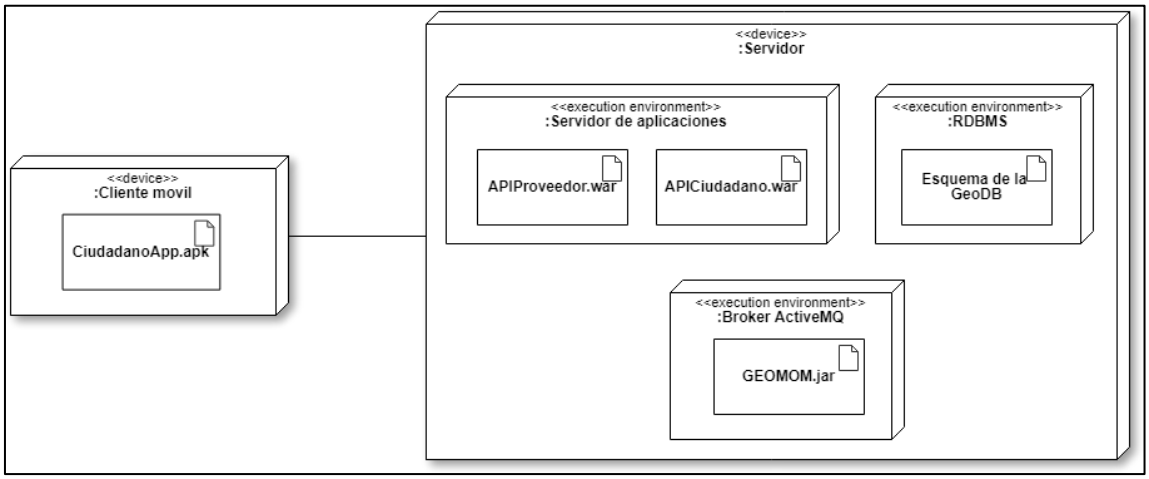

Figura 5.8: Diagrama de despliegue

El prototipo se despliega sobre dos nodos físicos: el cliente móvil y el servidor. El cliente lanza y ejecuta la aplicación CiudadanoApp.apk del ciudadano sobre la plataforma Android. El servidor contiene tres ambientes de ejecución: el servidor de aplicaciones, el manejador de la base de datos GeoDB y el broker de mensajería ActiveMQ. El servidor de aplicaciones realiza el deploy de los artefactos web APIProveedor.war y APICiudadano.war, correspondientes a las aplicaciones de los componentes API Proveedor y API Ciudadano respectivamente. El broker ActiveMQ realiza el deploy de la extensión del MOM en el artefacto GEOMOM.jar. Finalmente, el manejador RDBMS levanta una instancia de la base de datos geográfica del prototipo con el esquema de la GeoDB detallado en la sección 5.2.3.2.

5.2.8. Elección de algunas tecnologías

En esta sección se justifican algunas de las tecnologías utilizadas:

- Java 8. Se utilizó la plataforma Java 8 en los componentes APIs, en el componente GeoDB y en el broker GeoMOM. Java 8 permite, a diferencia de distribuciones anteriores, realizar operaciones en lenguaje funcional sobre colecciones genéricas.
- Spring boot. Spring Boot tiene la cualidad de definir todos los parámetros de configuración de la aplicación con anotaciones de Java. Para crear los web services SOAP, Spring Boot permite levantar un servidor Tomcat web embebido en la aplicación. Además, se dispone de la capacidad de generar un cliente SOAP a partir de la definición de los mensajes de las operaciones del servicio SOAP en un archivo xsd.
- React. Es una librería de javascript de vanguardia que permite el desarrollo ágil de aplicaciones web y móviles, con una amplia gama de dependencias disponibles en la web para integrar diversas funcionalidades. Si bien es una tecnología incipiente, existe una amplia comunidad que ayuda a encontrar soluciones a problemas comunes de la herramienta, en sitios como Stackoverflow [88] y GitHub [89].
- PostgreSQL+PostGIS. La extensión PostGIS brinda una amplia gama de operaciones SFA para consultar las relaciones entre las geometrías. Es sencilla la instalación y el uso de la extensión, y es muy directo el mapeo de hibernate-spatial sobre la herramienta.
- ActiveMQ. Se utiliza ActiveMQ como broker de mensajería. Existe amplia documentación de soporte en la web, no solo de sus características y elementos de mensajería, sino también de ejemplos de la API de ActiveMQ para crear productores y consumidores de las colas y topics de mensajes. Por otra parte, brinda muchos de los elementos de un MOM (traductores, adaptadores, enrutadores, etc.). Además, es sencillo lograr la extensibilidad del producto, solo realizando extensiones de clases Java y archivos de configuración. Por último, las versiones 4 y 5 utilizan el framework Spring, con lo cual es fácil la definición de beans de negocio e inyección de dependencias, así como acceso a templates de JMS que permiten interactuar con las estructuras del broker.

5.2.9. Simplificaciones respecto a la solución propuesta

El foco principal del prototipo es validar una prueba de concepto de las funcionalidades principales de la solución de la tesis y atacar los aspectos centrales de la propuesta. Por lo tanto, tiene algunas funcionalidades específicas que difieren de la propuesta completa del capítulo 4. En primer lugar, el mecanismo de notificación queda implementado por el componente API ciudadano, el cual ejecuta en forma parcial las funcionalidades detalladas en el broker de mensajería (sección 4.4). En segundo lugar, se evitó el despliegue en la simulación de la PGE. Con esto, se logra simplificar la implementación en gran medida y reducir la complejidad de los flujos de comunicación tanto entre

proveedores de datos y usuarios con la PGE, así como entre el GeoMOM y los subcomponentes internos de la PGE de su sistema de seguridad y de middleware.

5.2.10. Testing del prototipo

Se realizó el testing del prototipo en forma manual generando las entradas de datos en las aplicaciones cliente y la del proveedor con el fin de representar a los participantes del sistema. Por otra parte, se realizaron tests unitarios en la capa lógica de los diferentes componentes API Proveedor, API Ciudadano y broker de mensajería, de forma de verificar el correcto funcionamiento de ellos y de chequear la consistencia de los datos persistidos en la base de datos GeoDB. Para tal propósito, se utilizó la herramienta de testing JUnit extendida para proyectos en Spring [90].

5.3. Caso de studio

En esta sección se verá paso a paso el desarrollo un caso de estudio aplicado a la realidad uruguaya. Se darán de alta usuarios y suscripciones utilizando la aplicación cliente y se crearan nuevos proveedores y categorías de eventos junto a la publicación de eventos utilizando la aplicación del proveedor. Para cada caso de uso, se mostrarán capturas de pantalla para los ingresos de datos en el caso de las altas, así como para el despliegue de las salidas correspondientes, por ejemplo, en las notificaciones.

5.3.1. Descripción de la realidad del Caso de Estudio

Se pretende que algunos organismos del Estado uruguayo realicen la publicación de eventos con cierta localización geográfica y de interés para los usuarios utilizando credenciales registradas a nivel estatal. Los usuarios deben ser notificados en tiempo real en sus dispositivos móviles de los eventos publicados de acuerdo a ciertas restricciones. Dichas restricciones son por el tema del evento y por área geográfica. El evento debe ocurrir en la zona geográfica definida por el usuario.
Se desean publicar eventos relacionados a incendios ocurridos en un cierto lugar, accidentes viales, incidentes de tránsito y cursos de profesionalización académica en Montevideo. Los incendios son reportados por el SINAE (sección 3.2.1) especializado en notificar emergencias. Los accidentes de tránsito de la ciudad y los cursos serán informados por la Intendencia de Montevideo.
Como prueba de la plataforma, se registrará un usuario que realizará las suscripciones a los eventos deseados y recibirá las notificaciones de los eventos generados por SINAE e IMM. Se publicará un evento de incendio que ocurre en una zona forestal del barrio Cerro, un evento sobre accidente de tránsito entre dos vehículos en el barrio Buceo y por último un evento de un nuevo curso de preparación de chef profesional en pastelería en la zona de Parque Rodó.

5.3.2. Capturas y salidas de datos

En esta sección se verá la entrada de los datos de los eventos y de las suscripciones por los actores involucrados y descritos en la sección anterior. Se ilustran capturas de pantalla según las diferentes operaciones del sistema del prototipo. Además, se mostrarán las salidas de datos correspondientes a las notificaciones del ciudadano.

5.3.2.1. Alta de proveedores de datos y topics

En la pantalla de la Figura 5.9, se aprecia la estructura de la interfaz web de la aplicación proveedor. Como se puede visualizar, hay tres pestañas que corresponden a los tres casos de uso detallados anteriormente: Registrar Usuario (1), Registrar Categoría (2) y Publicar Evento (3). Ellos se señalan en rojo en la figura.

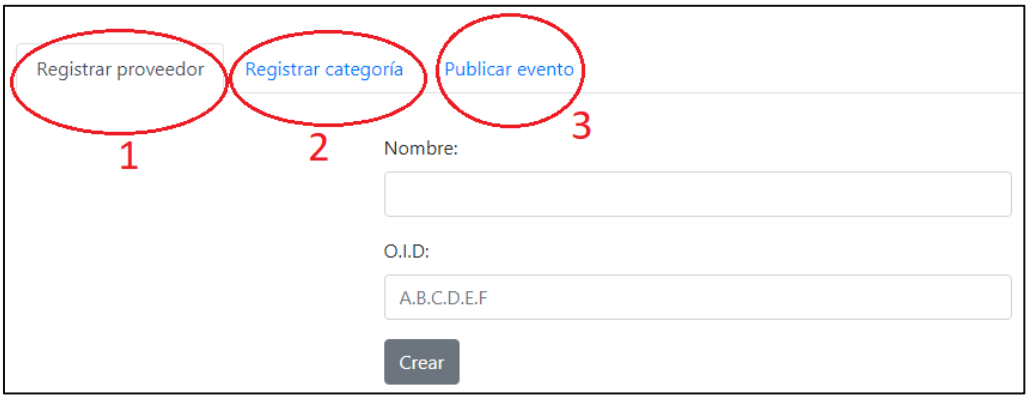

Figura 5.9: Interfaz web de proveedor

En primer lugar, se utilizara la aplicación para dar de alta proveedores de datos del Estado uruguayo. En este caso, se darán de alta SINAE (emergencias) e Intendencia de Montevideo. Para cada uno, se ingresa el nombre del proveedor y un OID (Object Identifier), identificador univoco del proveedor.
En la Figura 5.10, se muestra un ejemplo de alta de proveedor para el organismo SINAE.

Figura 5.10: Ejemplo de alta de proveedor

Luego, se crean las categorías de eventos (topics) que se utilizaran en el caso de estudio: incendio, cursos y accidente de tránsito. En la Figura 5.11, se muestra la creación del topic incendio y, de forma análoga, se realiza lo mismo para los restantes.

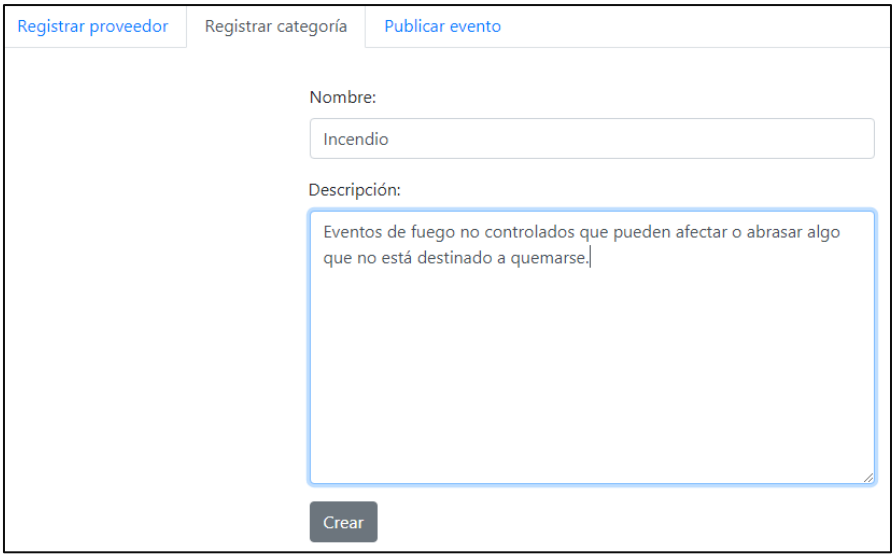

Figura 5.11: Ejemplo de alta de topic

5.3.2.2. Alta de usuario

El registro de usuarios solicita una cédula de identidad y una contraseña. Se crearan un usuario con número de documento 4123456-7. En la Figura 5.12 se aprecia, como ejemplo, el ingreso de datos.

Figura 5.12: Registro de usuario

Una vez logrado el registro del usuario, se procede a hacer el login en el sistema que se ilustra en la Figura 5.13. Una vez que se realiza el login, la sesión del usuario queda recordada en el sistema y el usuario no debe volver a loguearse en caso que cierre la aplicación. En este caso, cuando vuelve a abrir, la aplicación lo redirigirá al menú principal de opciones.

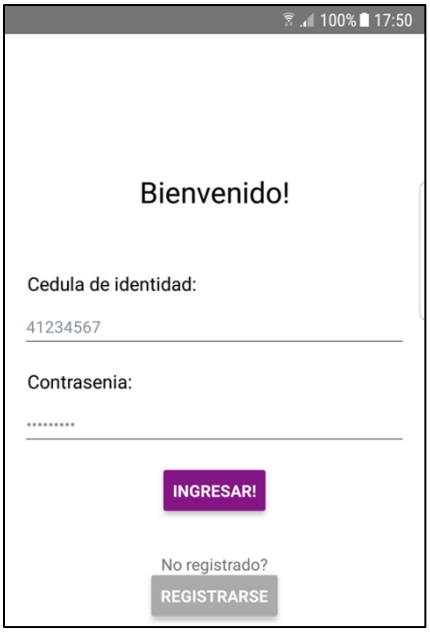

Figura 5.13: Login del usuario registrado

Si el login es correcto, se le redirige a la pantalla de menú principal (Figura 5.14) donde se muestran cuatro pestañas:

- Notificaciones: aquí se irán desplegando las notificaciones de eventos que vayan ocurriendo de acuerdo a las suscripciones del usuario.
- Restricciones zonales: aquí se mostrara un mapa que permitirá dar de alta las restricciones que dependen de un Polígono en el espacio geográfico. Dichos polígonos se irán renderizando en el mapa. Haciendo long-tap en el Polígono, se desplegaran los datos de la suscripción.
- Restricciones buffer: aquí se mostrara una lista de restricciones de tipo buffer en forma de lista. Depende de la localización del usuario. Por lo tanto, para que se realice una notificación de este tipo de restricción, debe estar activado el GPS del dispositivo móvil.
- Opciones. Solo se permite hacer logout de la aplicación, eliminando la sesión del usuario.

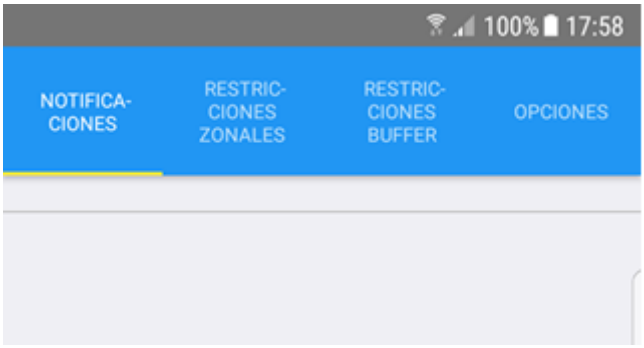

Figura 5.14: Menú principal de la aplicación cliente

5.3.2.3. Alta de suscripciones

Se darán de alta, en primera instancia, dos restricciones de tipo zonal y una de tipo buffer:

- Restricción zonal 1. Abarca los barrios de Parque Rodo y Punta Carretas. La suscripción se hace sobre la categoría cursos y tendrá el nombre "ZonalCursos".

- Restricción zonal 2. Abarca parte del barrio Cerro. La suscripción se hace sobre la categoría incendio y tendrá el nombre "ZonalIncendio".
- Restricción de buffer. Se definirá un buffer de tamaño 500 metros sobre la categoría accidente_transito y tendrá el nombre "BufferAccidenteTransito".

En las figuras 5.15 a 5.20 se ilustran capturas de pantalla de alta de suscripción.

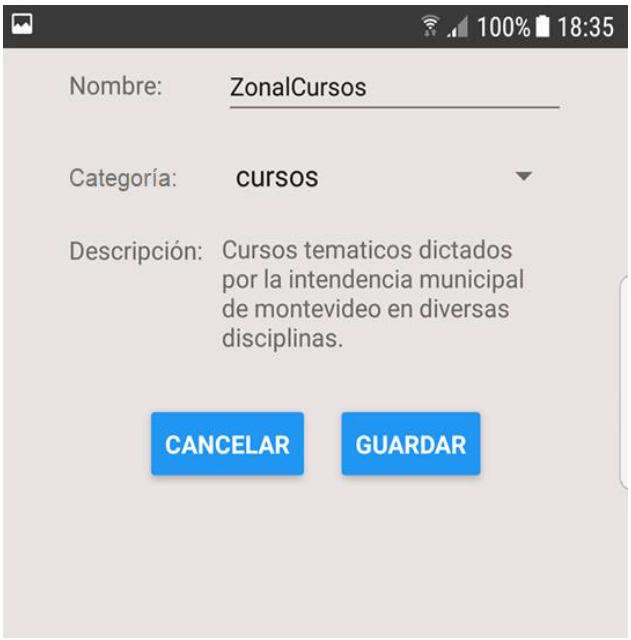

Figura 5.15: Alta de suscripción "ZonalCursos"

Figura 5.16: Área de suscripción ZonalCursos

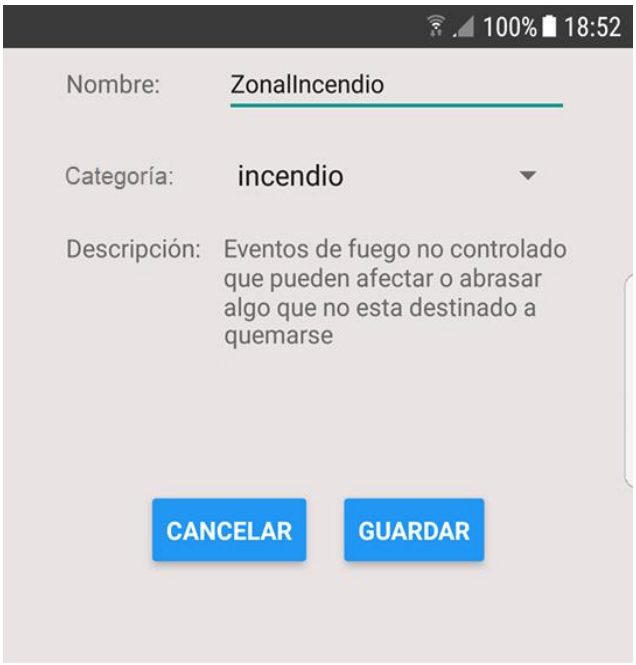

Figura 5.17: Alta de suscripción "ZonalIncendio"

Figura 5.18: Área de suscripción ZonalIncendio

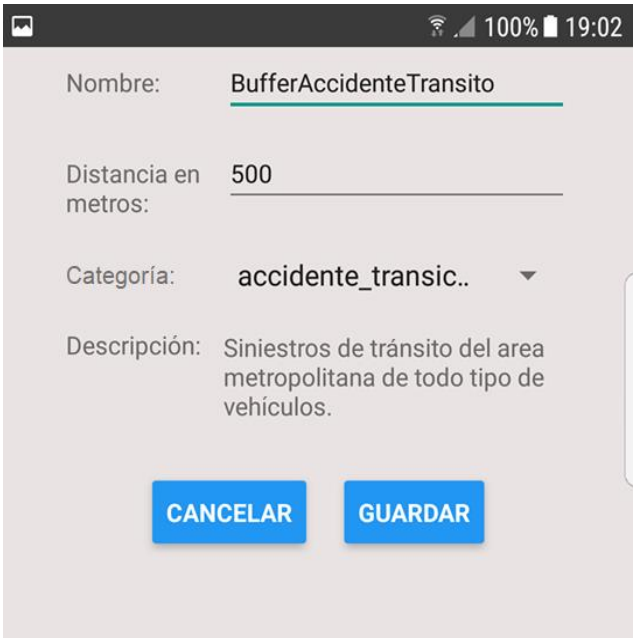

Figura 5.19: Alta de suscripción BufferAccidenteTransito

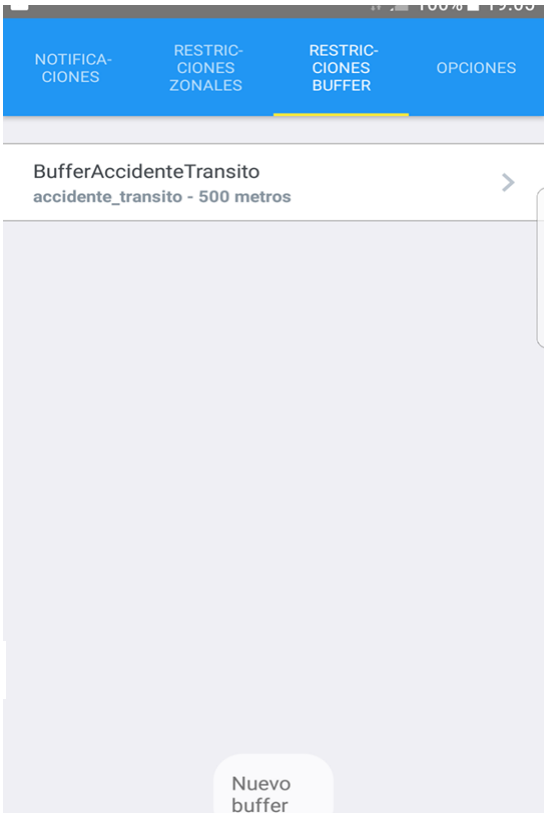

Figura 5.20: Suscripción creada

5.3.2.4. Publicación de eventos

Ahora se publicaran los eventos en la interfaz web. En primer lugar, se creara un evento de tipo *incendio* en la zona de barrio Cerro que abarcara un área, siendo el SINAE el organismo productor del mensaje. En la Figura 5.21 se ilustra el ingreso de datos del evento. Se define una descripción del evento ocurrido, un timeout de 1 día (1440 minutos) y la representación geográfica del evento que es un Polígono en el mapa.

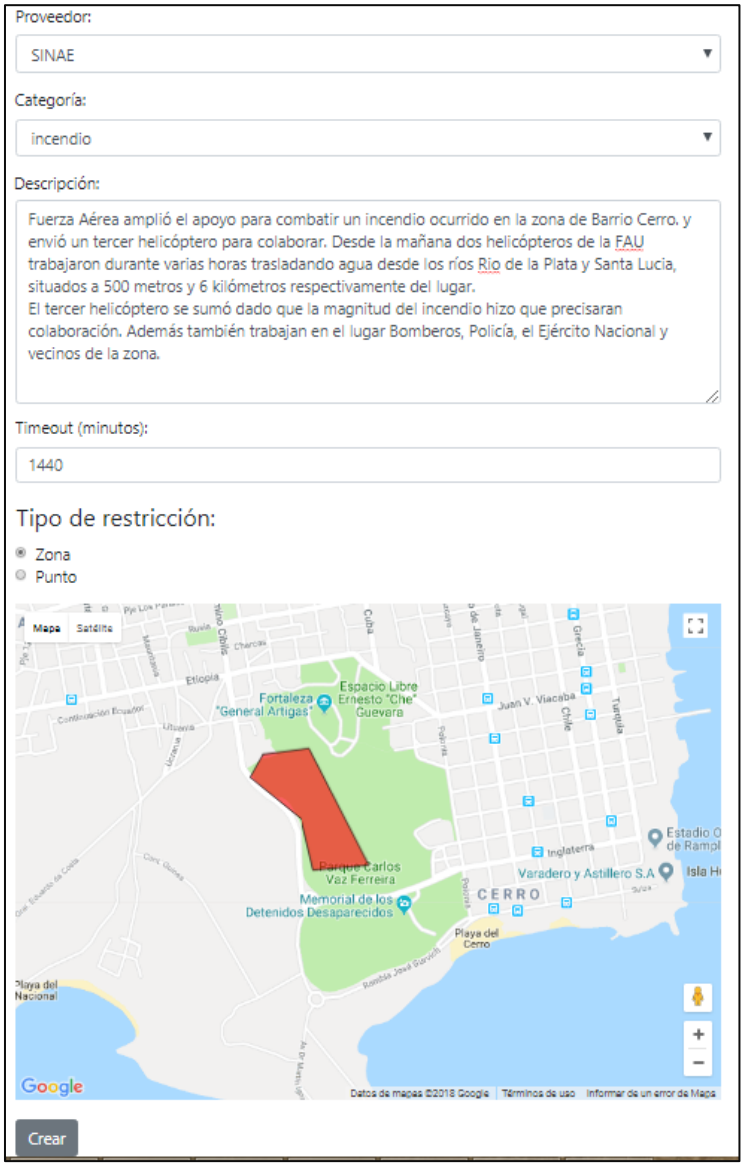

Figura 5.21: Creación de evento de Incendio en barrio Cerro

En segundo lugar, se creará un evento de tipo *curso* que se dictara en una dependencia de la IMM en el cruce de calles Av. Julio María Sosa y Av. Bvar Artigas (barrio Parque Rodo), por lo tanto tiene localización puntual. Este evento genera un matching con la suscripción "ZonalCursos", ya que hay coincidencia de topics y el punto del evento pertenece al Polígono de la suscripción. El evento se ilustra en la Figura 5.22.

Figura 5.22: Creación de evento de Curso en Parque Rodó

Por último, se creará un evento de tipo *accidente_transito* entre automóviles en el cruce de calles Bv. José Batlle y Ordoñez y Mentana, junto con las características del siniestro siendo reportando por la Intendencia de Montevideo. El evento es puntual ya que ocurre en una esquina (Figura 5.23).

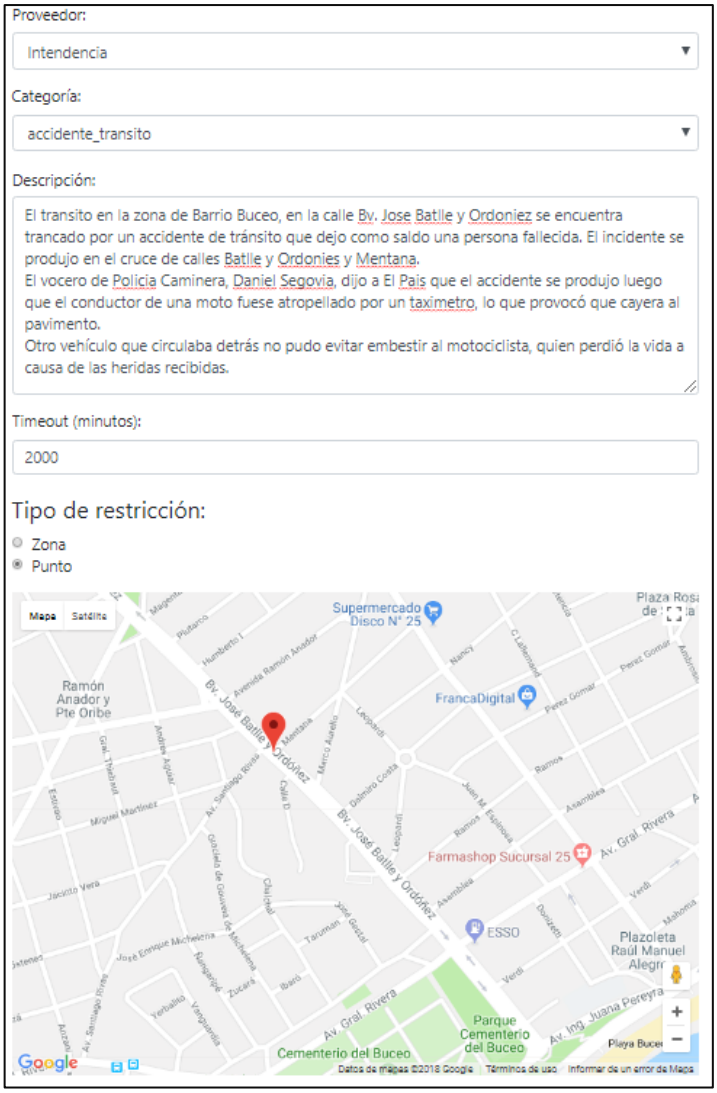

Figura 5.23: Creación de evento de Accidente de Tránsito en barrio Buceo

5.3.2.5. Notificaciones

De acuerdo a las suscripciones del usuario y sus suscripciones, existe matching con los eventos publicados y lo cual tiene como efecto el envío de notificaciones al usuario con la información de los eventos de la subsección anterior.
Respecto al evento de incendio (Figura 5.21), el área del evento se intersecta con la suscripción "ZonalIncendio" y hay coincidencia de eventos. Por ende, se despliega el mensaje de notificación en la aplicación cliente de la Figura 5.24. Allí se marca en rojo el área de la suscripción y en azul el área del evento.

Figura 5-24: Notificación de evento Incendio

Respecto al evento de curso (Figura 5.22), se genera un matching con la suscripción "ZonalCursos", ya que hay coincidencia de tipos de evento y el punto del evento pertenece al polígono de la suscripción. El mensaje de notificación en la aplicación cliente aparece en la Figura 5.25. En la notificación, aparecen los detalles del evento junto con las geometrías competentes, destacando en rojo el área de notificación y con marcador azul la del evento.

Figura 5-25: Notificación de evento curso

Por último, el evento de accidente vial (Figura 5.23) genera un matching con la suscripción "BufferAccidenteTransito", ya que hay coincidencia de tipos de eventos y el punto del evento pertenece al buffer de suscripción, suponiendo que el usuario se encuentre en las cercanías del incidente y a una distancia no mayor a los 500 metros. El mensaje de notificación en la aplicación cliente aparece en la Figura 5.26. En la notificación, aparecen los detalles del evento junto con las geometrías relacionadas, destacando en rojo el buffer y con marcador azul la del evento.

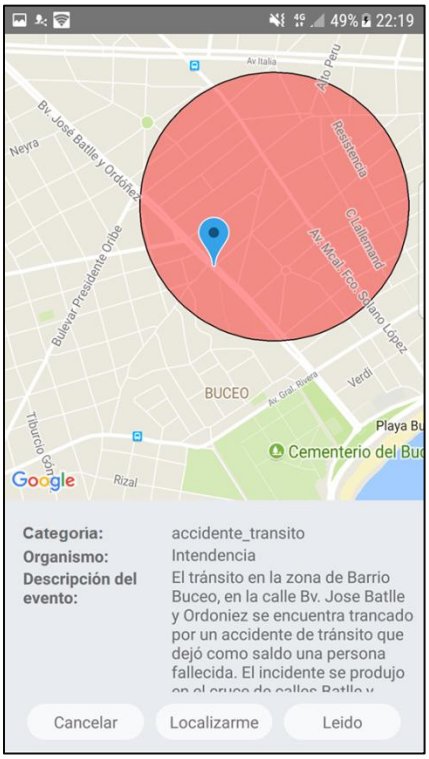

Figura 5-26: Notificación de evento accidente de tránsito

6. Conclusiones y trabajo futuro

En este capítulo, en la sección 6.1 se presenta un resumen del trabajo realizado y sus principales contribuciones, y en la sección 6.2, se esbozan posibles líneas de trabajo a futuro.

6.1. Resumen y contribuciones

La tesis presenta una extensión de la plataforma de gobierno electrónico de AGESIC. En dicha existen entidades que publican información geoespacial y otras que se suscriben a diferentes eventos utilizando restricciones de diferente tipo. Los suscriptores reciben notificaciones en base a las restricciones.

En primera instancia, se presenta el planteo del problema basado en la noción anterior. Para ello, se especifica, sin brindar mucho detalle, el comportamiento general de la plataforma y los actores involucrados. Se hizo un relevamiento de posibles actores que podrían ser potenciales publicadores de datos en la plataforma, brindando datos de naturaleza geográfica. La mayoría de dichos publicadores corresponden a proveedores del Estado uruguayo ya que la solución se enmarca en la plataforma de AGESIC, aunque también se incluyeron otros proveedores privados que podrían ser interesantes. Por otra parte, cualquier ciudadano del Estado registrado puede ser un actor en este sistema con el objetivo de recibir alertas y notificaciones en tiempo real de su interés con cierta y ocurrencia en determinado lugar en el espacio geográfico. Por otra parte, se otorga la flexibilidad de agregar dinámicamente nuevos proveedores que puedan aportar nuevos sets de datos geoespaciales potencialmente atractivos para los usuarios.

Especificado el problema, se propone una arquitectura de solución. Aquí se definen en detalle las piezas de software necesarias de la solución y los roles y responsabilidades de cada una de ellas. Finalmente, se realiza una comparación con algunos de los trabajos académicos encontrados en la etapa de investigación y de relevamiento, de forma de contraponer características comunes y de resaltar el valor agregado de la solución de la tesis. Por último, se construyó un prototipo que cubre una amplia gama de las funcionalidades descritas, el cual es una prueba de concepto fiel a la solución propuesta.

En la fase de relevamiento, no se encontraron trabajos académicos que vincularan middleware de mensajería (MOM) con capacidades de geolocalización en el contexto de gobierno electrónico, lo cual hace a la solución de la tesis una propuesta innovadora y original. Por este motivo, la propuesta logra unificar los conceptos de LBS, e-gov y MOM en un solo sistema.

En síntesis, este trabajo permite atacar algunas de las problemáticas actuales del Uruguay que conciernen a los mecanismos de notificación de gran variedad de información a la población en general. El ciudadano puede establecer órdenes de relevancia de la información y definir canales de su recepción de acuerdo a sus preferencias. Esta operativa puede consumarse utilizando las capacidades de la PGE de AGESIC, la cual se encuentra en un estado de maduración reconocido a nivel mundial. Además, la solución propuesta posee la suficiente flexibilidad para articular proveedores, categorías de eventos, suscripciones y canales de comunicación.

Una versión preliminar del análisis y la arquitectura de este trabajo fueron presentados a personal técnico de AGESIC y de la IDE [91] en diciembre de 2017 siendo validados y muy bien recibidos. La propuesta resulto de amplio interés para los representantes de los organismos. En segundo lugar, se escribió un artículo de la propuesta que fue evaluado y aprobado para su presentación en octubre de 2018 en el CLEI [92] como parte del Simposio Latinoamericano de Procesos de Negocio, Arquitecturas y Sistemas Organizacionales (SLPNASO).

6.2. Trabajo a futuro

Esta tesis se inscribe en un marco de actividades académicas sobre Sistemas de Información Geográficos, así como actividades en el contexto de la AGESIC para su aplicación en servicios públicos, por lo que constituye naturalmente un paso intermedio más que un resultado final.
El trabajo en curso consiste en la profundización sobre la aplicación de tecnologías de middleware avanzado para la mejora de los Servicios de Información Geográficos y de su integración con otros sistemas, especialmente en el marco del Gobierno Electrónico.
Los resultados de la tesis permiten identificar varios aspectos que resultan de candidatos a trabajos de futuro de la tesis:

1. Análisis y diseño de diferentes tipos de LBS aplicados a las necesidades de cada uno de los organismos publicadores de datos u orientados a ciertos grupos de usuarios. Los diferentes LBS deben ser implementados por los desarrolladores de aplicaciones de cada organismo o de un grupo de organismos.
2. Integración técnica de la PGE de AGESIC con la arquitectura de la solución, de forma de estudiar las interacciones de los componentes de la PGE real con los componentes GeoMOM y broker de mensajería.
3. Definición de mecanismos para el registro de nuevos proveedores en la PGE, indicando la información de identificación que debe suministrarse al administrador de la PGE para que pueda invocar las operaciones del GeoMOM. En este punto, cabe profundizar en la especificación de roles y permisos que realizan el control de acceso en la PGE y los procesos de negocio necesarios para aceptar un nuevo proveedor.
4. Abordar posibles convenios y las modalidades de integración con los servicios externos responsables de efectuar las notificaciones push en los diferentes canales. Se deben analizar las diferentes formas de notificación y los diferentes requisitos que el ciudadano debe satisfacer para el uso de determinado tipo de canal (sms, mail, etc.). A más bajo nivel, habría que estudiar las APIs y protocolos de comunicación con cada uno de los servicios.
5. Estudiar la normativa vigente relativa a las políticas de privacidad del ciudadano y de protección de datos personales, de forma que la aplicación cliente cumpla con las condiciones para reportar la ubicación actual del ciudadano y sin infringir ninguna cláusula.
6. Implementación completa de la propuesta. La misma se está llevando a cabo en 2018 por un grupo de tesis de grado de la Facultad de Ingeniería de la Universidad de la República, en coordinación con la docente Raquel Sosa del Instituto de Computación.

7. Bibliografía

[1] J. McNamara, GPS For Dummies, Second ed., Brownstown: John Wiley And Sons Ltd, 2008. ISBN: 978-0470156230.

[2] C. Li and A. Brimicombe, Location-Based Services and Geo-Information Engineering, 1st ed., Wiley, 2009, pp. 1-2. ISBN: 978-0470857373.

[3] G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley Professional, 2003. ISBN: 978-00133065107.

[4] P. Eugster, "The many faces of publish/subscribe," *ACM computing surveys (CSUR)*, vol. 35, no. 2, pp. 114-131, 2003. ISSN: 0360-0300.

[5] E. Curry, "Message-oriented middleware," in *Middleware for communications*, Wiley, Ed., John Wiley & Sons, 2004. ISBN: 978-0-470-86206-3, pp. 1-28.

[6] T. Almarabeh and A. Abuali, "A general framework for e-government: definition maturity challenges, opportunities, and success," *European Journal of Scientific Research*, vol. 39, no. 1, pp. 29-42, 2010. ISSN:1450-216X.

[7] "Instituto Nacional de Meteorología - Institucional," INUMET, [Online]. Available: http://www.meteorologia.com.uy/. [Accessed October 2017].

[8] "Sistema Nacional de Emergencias - Institucional," SINAE, [Online]. Available: http://sinae.gub.uy/. [Accessed August 2017].

[9] "Dirección Nacional de Policía Caminera," Ministerio del Interior, [Online]. Available: https://www.minterior.gub.uy/index.php/2-uncategorised/115-direccion-nacional-de-policia-caminera. [Accessed October 2017].

[10] "Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture," Open Geospatial Consortium Inc., version 1.2.1, 2011. Standard key: 06-103r4. [Online]. Available: http://www.opengeospatial.org/standards/sfa.

[11] T. Almarabeh, Y. K. Majdalawi and H. Mohammad, "Cloud Computing of E-Government," *Communications and Network*, vol. 8, no. 01, pp. 1-8, 2016. ISSN: 1949-2421.

[12] R. Heeks, "Benchmarking egovernment: improving the national and international measurement, evaluation and comparison of egovernment," in *Evaluating Information Systems: Public and Private Sector*, vol. 257, 2008. ISBN: 978-0750685870, pp. 257-301.

[13] E. Kalampokis, E. Tambouris and K. Tarabanis, "Open Government Data: A Stage Model," in *Electronic Government: 10th IFIP WG 8.5 International Conference, EGOV 2011 Proceedings*, vol. 6846, Netherlands, 2011. ISBN: 978-3-642-22878, pp. 235-246.

[14] R. Nripendra, Y. K. Dwivedi and M. D. Williams, "Analysing challenges, barriers and CSF of egov adoption," *Transforming Government: People, Process and Policy*, vol. 7, no. 2, pp. 177-198, 2013. ISSN: 1750-6166.

[15] Z. Ebrahim and Z. Irani, "E-government adoption: architecture and barriers," *Business process management journal*, vol. 11, no. 5, pp. 589-611, 2005. ISSN: 1463-7154.

[16] "Institucional," AGESIC, [Online]. Available: https://www.agesic.gub.uy/innovaportal/v/19/1/agesic/institucional.html. [Accessed August 2017].

[17] "Plataforma de Interoperabilidad," AGESIC, [Online]. Available: https://www.agesic.gub.uy/innovaportal/file/1619/1/plataforma_de_interoperabilidad.pdf. [Accessed August 2017].

[18] "Notificaciones y Comunicaciones Electrónicas," AGESIC, [Online]. Available: https://www.agesic.gub.uy/innovaportal/v/3937/1/agesic/notificaciones-y-comunicaciones-electronicas.html?idPadre=4047. [Accessed January 2018].

[19] C. López Vázquez and M. Á. Bernabé, Fundamentos de las Infraestructuras de Datos Espaciales (IDE),

Madrid: UPM Press, 2012, pp. 95-117. ISBN: 978-8493919665.

[20] S. Givant and P. Halmos, Introduction to Boolean algebras, Springer Science & Business Media, 2008. ISBN: 978-1441923240.

[21] "Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option," Open Geospatial Consortium Inc., version 1.2.1, 2010. Standard key: 06-104r4. [Online]. Available: http://www.opengeospatial.org/standards/sfs.

[22] A. Machanic and A. Aitchison, Expert SQL Server 2008 Development, 1st ed., Apress, 2010, pp. 296-301. ISBN: 978-1430269526.

[23] C. D. Tomlin, GIS and Cartographic Modeling, E. Press, Ed., 2012. ISBN: 978-1589483095, p. 182.

[24] K. Virrantaus, J. Veijalainen, J. Markkula, A. Garmash, A. Katasonov, V. Terziyan and H. Tirri, "Developing GIS-supported location-based services," in *Proceedings of the Second International Conference on Web Information Systems Engineering*, 2001.

[25] J. Schiller and A. Voisard, Location-based services, Elsevier, 2004, pp. 13-20. ISBN: 978-1558609297.

[26] A. Kushwaha and V. Kushwaha, "IJAET Location Based Services using Android Mobile," *International Journal of Advances in Engineering & Technology,* vol. 1, no. 1, pp. 14-20, 2011. ISSN: 2231-1963.

[27] A. Cupper, G. Treur and C. Linhoff, "TraX: A device-centric middleware framework for location-based services," *IEEE Communications Magazine,* vol. 44, no. 9, pp. 114-120, 2006. ISSN: 0163-6804.

[28] A. Holzer, P. Eugster and B. Garbinato, "Adaptive Location-based Publish/Subscribe," *Computer Networks: The International Journal of Computer and Telecommunications Networking,* vol. 56, no. 12, pp. 2949-2962, 2012. ISSN: 1389-1286.

[29] I. Leontadis, "Publish/subscribe notification middleware for vehicular networks," in *Middleware 2007 Proceedings of the 4th on Middleware doctoral symposium*, New York, 2007.

[30] P. Jiang, J. Bigham, E. Bodanese and E. Claudel, "Publish/subscribe delay-tolerant message-oriented middleware for resilient communication," *IEEE Communications Magazine,* vol. 49, no. 9, pp. 124-130, 2011. ISSN: 0163-6804.

[31] A. K. Dey, "Understanding and using context," *Personal and ubiquitous computing,* vol. 5, no. 1, pp. 4-7, 2001. ISSN: 1617-4909.

[32] D. Bakken, "Middleware," in *Encyclopedia of Distributed Computing*, vol. 11, Kluwer Academic Press, 2001.

[33] G. Alonso, F. Casati, H. Kuno and V. Machiraju, Web Services: Concepts, Architectures and Applications, Springer Science & Business Media, 2004. ISBN: 978-3540440086, p. 354.

[34] M. Albano, "Message-oriented middleware for smart grids," *Computer Standards & Interfaces,* vol. 38, pp. 133-143, 2015. ISSN: 0920-5489.

[35] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns, Addison-Wesley, 1995. ISBN: 0-201-63361-2.

[36] "Web Services Glossary," W3C, 2004. [Online]. Available: https://www.w3.org/TR/ws-gloss/. [Accessed May 2018].

[37] "Web Services Description Language (WSDL)," W3C, version 2.0, 2007. Standard key: 20070626.. [Online]. Available: https://www.w3.org/TR/2007/REC-wsdl20-20070626/.

[38] B. M. Balachandar, RESTful Java Web Services: A pragmatic guide to designing and building RESTful APIs using Java, 3rd ed., Packt, 2017, pp. 5-10. ISBN: 978-1788294041.

[39] "Web Service Addressing (WS-Addressing)," W3C, [Online]. Available: https://www.w3.org/Submission/ws-addressing/. [Accessed August 2017].

[40] "Web Service Standards and Notifications," INNOQ, [Online]. Available: https://www.innoq.com/resources/WebServicesStandardsOverview-2005-01.pdf. [Accessed August 2017].

[41] "Web Services reliable messaging protocol (WS-ReliableMessaging)," BEA Systems, IBM, Microsoft Corporation, Inc, TIBCO Software, 2005. [Online]. Available:

http://specs.xmlsoap.org/ws/2005/02/rm/WS-RMPolicy.pdf.

[42] "WS-Security Core Specification," OASIS Web Services Security TC, version 1.1, 2004.. [Online]. Available: https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-osSOAPMessageSecurity.pdf.

[43] T. van Lessen and S. Moser, *JAX 2009: SOA-basierte Business Integration mit Eclipse BPEL und Apache ODE,* Stuttgart, 2009.

[44] R. Kanneganti and P. A. Chodavarapu, SOA Security, Manning Publications, 2008. ISBN: 978-1932394689.

[45] "OASIS. Web Services Base Notification 1.3 (WS-BaseNotification)," version 1.3, 2006. [Online]. Available: Standard reference: http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf.

[46] "OASIS. Web Services Brokered Notification 1.3 (WS-BrokeredNotification)," version 1.3, 2006. [Online]. Available: Standard reference: http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf.

[47] "OASIS. Web Services Topics 1.3 (WS-Topics)," versión 1.3, 2006. [Online]. Available: Standard reference: https://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf.

[48] M. Richards, R. Monson-Haefel and . A Chappell, Java Message Service: Creating Distributed Enterprise Applications, 2nd ed., I. O'Reilly Media, Ed., Chappell, 2009. ISBN: 978-0596000684, p. 330.

[49] "IBM MQ: Integrated messaging to connect your enterprise, Technical White Paper," IBM, [Online]. Available: https://public.dhe.ibm.com/common/ssi/ecm/ws/en/wsw14195usen/WSW14195USEN.PDF. [Accessed July 2017].

[50] "Overview," RabbitMQ, [Online]. Available: https://www.rabbitmq.com/. [Accessed June 2017].

[51] "The Simple Text Oriented Messaging Protocol," Stomp, [Online]. Available: http://stomp.github.io/ . [Accessed July 2017].

[52] "Overview," Apache Active MQ, [Online]. Available: http://activemq.apache.org/. [Accessed July 2017].

[53] A. Konstantinidis, A. Carzaniga and A. L. Wolf, "A content-based publish/subscribe matching algorithm for 2d spatial objects," in *Middleware 2011 Proceedings of the 12th International Middleware Conference*, Lisbon, 2011. ISBN: 978-3-642-25820-6.

[54] S. Liang, "Real-time notification and improved situational awareness in fire emergencies using geospatial-based publish/subscribe," *International Journal of Applied Earth Observation and Geoinformation,* vol. 12, no. 6, pp. 431-438, 2010. ISSN: 0303-2434.

[55] "GeoMQTT," [Online]. Available: http://www.gia.rwth-aachen.de/GeoMQTT. [Accessed March 2017].

[56] X. Chen, Y. Chen and F. Rao, "An efficient spatial publish/subscribe system for intelligent location-based services," in *Middleware 2003, Proceedings of the 2nd international workshop on Distributed event-based systems*, 2003. ISBN: 3-540-40317-5.

[57] Z. Sun, L. Di, G. Heo, C. Zhang, H. Fang, P. Yue, L. Jiang, X. Tan, L. Guo and L. Lin, "GeoFairy: Towards a one-stop and location based Service for Geospatial Information Retrieval," *Computers, Environment and Urban Systems,* vol. 62, no. C, pp. 156-167 , 2017. ISSN: 0198-9715.

[58] "Google," Google Inc, [Online]. Available: https://www.google.com/. [Accessed April 2018].

[59] "Representación cartográfica sin límites," ArcGIS, [Online]. Available: https://www.arcgis.com/features/index.html. [Accessed May 2018].

[60] "International Space Nation," NASA, [Online]. Available: https://www.nasa.gov/mission_pages/station/main/index.html. [Accessed May 2018].

[61] A. Benchi, P. Launay and F. Guidec, "JMS for opportunistic networks," *Ad Hoc Networks,* vol. 25, no. 4, pp. 359-369, 2015. ISSN: 1570-8705.

[62] A. Antonic, "A mobile crowdsensing ecosystem enabled by a cloud-based publish/subscribe middleware," in *FICLOUD '14 Proceedings of the 2014 International Conference on Future Internet of Things and Cloud*, 2014. ISBN: 978-1-4799-4357-9 .

[63] B. Richerzhagen, "A Framework for Publish/Subscribe Protocol Transitions in Mobile Crowds," in *Proceedings of the 10th IFIP WG 6.6 International Conference on Management and Security in the Age of Hyperconnectivity*, Munich, 2016. ISBN: 978-3-319-39813-6.

[64] B. Almadani, S. Abudalfa and S.-H. Yang, "QoS adaptation for publish/subscribe middleware in real-time dynamic environments," *International Arab Journal of Information Technology,* vol. 14, no. 2, pp. 230-238, 2017. ISSN: 2309-4524.

[65] Y. Huang and H. Garcia-Molina, "Publish/Subscribe in a Mobile Environment," *Wireless Networks,* vol. 10, no. 6, p. 643–652, 2004. ISSN: 1022-0038.

[66] "Visualizador SIG del SINAE," SINAE, [Online]. Available: http://visualizador.sinae.gub.uy/sinaeViz/. [Accessed October 2017].

[67] "Congreso de Intendentes," Intendencia de Montevideo, [Online]. Available: http://www.montevideo.gub.uy/palabras-claves/congreso-de-intendentes. [Accessed October 2017].

[68] "Sistema de Información Geográfico," Intendencia de Montevideo, [Online]. Available: http://sig.montevideo.gub.uy/. [Accessed October 2017].

[69] "Observatorio Ambiental Nacional - Institucional," MVOTMA, [Online]. Available: https://www.dinama.gub.uy/oan/. [Accessed October 2017].

[70] "Cámara de Comercio y Servicios," CNCS, [Online]. Available: http://www.cncs.com.uy/. [Accessed October 2017].

[71] "Unidad Nacional de Seguridad Vial – Institucional," UNASEV, [Online]. Available: http://unasev.gub.uy/inicio/institucional/que_es_la_unasev/. [Accessed October 2017].

[72] "Delivery de Comida Online," PedidosYa, [Online]. Available: https://www.pedidosya.com.uy/blog/wp-content/uploads/sites/4/2015/03/Acerca_de_PedidosYa.pdf. [Accessed November 2017].

[73] "About CitiCop," CitiCop, [Online]. Available: https://www.citycop.org/. [Accessed October 2017].

[74] "La cultura al alcance de todos," SocioEspectacular, [Online]. Available: http://www2.socioespectacular.com.uy/. [Accessed November 2017].

[75] "Simple Polygon," Google Maps, [Online]. Available: https://developers.google.com/maps/documentation/javascript/examples/polygon-simple?hl=es-419. [Accessed October 2017].

[76] L. Canales, M. Félix and A. Ramiro, "Integración GIS-PGE," Instituto de Computación, Facultad de Ingeniería, UDELAR, Montevideo, 2015.

[77] "Mobile Application Part," International Telecommunication Union, [Online]. Available: http://www.itu.int/rec/T-REC-H.246-200011-S!AnnE2. [Accessed November 2017].

[78] "MARS Release," Eclipse, [Online]. Available: https://www.eclipse.org/mars/. [Accessed February 2018].

[79] "Spring Boot Project," Spring, [Online]. Available: https://projects.spring.io/spring-boot/. [Accessed January 2018].

[80] "PostgreSQL 9.3 Released!," PostgreSQL, [Online]. Available: https://www.postgresql.org/about/news/1481/. [Accessed January 2018].

[81] "Spatial and Geographic objects for PostgreSQL," PostGIS, [Online]. Available: https://postgis.net/. [Accessed February 2018].

[82] "Hibernate Spatial is now part of the Hibernate ORM project," Hibernate Spatial, [Online]. Available: http://www.hibernatespatial.org/. [Accessed January 2018].

[83] "React Native: Build native mobile apps using JavaScript and React," Facebook Inc, [Online]. Available: https://facebook.github.io/react-native/. [Accessed February 2018].

[84] "npm," npmjs, [Online]. Available: https://www.npmjs.com/. [Accessed February 2018].

[85] "WebStorm, The Smartest Javascript IDE," JetBrains, [Online]. Available: https://www.jetbrains.com/webstorm/. [Accessed December 2017].

[86] "React, A JavaScript library for building user interface," Facebook Inc, [Online]. Available:

https://reactjs.org/. [Accessed January 2018].

[87] "Webpack: Build your styles," Webpack, [Online]. Available: https://webpack.js.org/. [Accessed March 2018].

[88] "Stackoverflow," Stack Exchange Inc, [Online]. Available: https://stackoverflow.com/. [Accessed May 2018].

[89] "Learn Git and GitHub without any code!," GitHub Inc, [Online]. Available: https://github.com/. [Accessed May 2018].

[90] "Testing," Spring, [Online]. Available: https://docs.spring.io/spring/docs/current/spring-framework-reference/testing.html. [Accessed March 2018].

[91] "IDEuy: Infraestructura de Datos Espaciales," IDE, [Online]. Available: http://ide.uy/. [Accessed June 2018].

[92] "CLEI: Centro Latinoamericano de Estudios de Informática," Universidad EAFIT, [Online]. Available: https://www.clei.org/. [Accessed June 2018].

[93] S. Steiniger, M. Neun and A. Edwardes, "Foundations of Location Based Services," in *CartouCHe - Lecture Notes on LBS*, v. 1.0 ed., University of Zürich, 2006.

[94] L. David, J. V. Filho, M. Endler and S. D. Barbosa, "Middleware Support for Context-Aware Mobile Applications with Adaptive Multimodal User Interfaces," in *2011 Fourth International Conference on Ubi-Media Computing*, 2011.

[95] H. Flores and S. N. Srirasma, "Mobile Cloud Middleware," *Journal of Systems and Software*, vol. 92, pp. 82-94, 2014. ISSN: 0164-1212.

[96] C. M. Keet, " Structuring GIS information with types of granularity: a case study," in *Proceedings of the 6th International Conference on Geomatics [VI Congreso Internacional de Geomática]*, La Habana, 2009.

Anexo

A) Matching geométrico

En la Figura A1 se puede apreciar la consulta SQL+SFS que realiza el matching geométrico y la especificación de sus parámetros. Las tablas se corresponden con las del esquema de la Figura 5.5.

```
/*

    Parametros:
        :topic = evento.topic
        :punto = evento.geometria.punto // null si evento.geometria es poligono
        :poligono = evento.geometria.poligono // null si evento.geometria es punto
        ·convFactor = valor numerico real que permite convertir una distancia
                    en metros a grados terrestres

*/

SELECT suc
FROM suscripcion suc
JOIN usuario u ON suc.usuarioId=u.id
LEFT JOIN suscripcion_area sarea ON sarea.suscripcionId=suc.id
LEFT JOIN suscripcion_buffer sbuffer ON sbuffer.suscripcionId=suc.id
WHERE
    LOWER(suc.nombre_topic)=LOWER(:topic) AND -- (1)
    (
        (sarea IS NOT NULL AND :punto IS NOT NULL AND st_within(:punto, sarea.poligono_geometria)) OR --(2)
        (sarea IS NOT NULL AND :poligono IS NOT NULL AND st_intersects(:poligono, sarea.poligono_geometria)) OR -- (3)
        (sbuffer IS NOT NULL AND u.ultima_localizacion IS NOT NULL AND minute(getTimestamp()-u.fecha_ultima_localizacion) < 10 AND -- (4)
            (:punto IS NOT NULL AND st_within(:punto, st_buffer(u.ultima_localizacion, sbuffer.radio*:convFactor))) OR -- (5)
            (:poligono IS NOT NULL AND st_intersects(:poligono, st_buffer(u.ultima_localizacion, sbuffer.radio*:convFactor))) -- (6)
        )
    )
```

Figura A1: Consulta SQL Matching evento-suscripción

La consulta devuelve la lista de suscripciones de los usuarios en las cuales la intersección de la geometría de la suscripción y de la geometría del evento es distinta de vacía. En el WHERE de la consulta se distinguen las siguientes condiciones:

- En (1) se evalúa que el topic de la suscripción coincida con el topic del evento pasado por parámetro.
- De (2) a (6) se evalúa si existe matching geométrico.
    - En (2), la geometría del evento es puntual y la del a suscripción es un polígono. Por ende, se evalúa que la función st_within de PostGIS sea verdadera.
    - En (3), la geometría del evento es un Polígono y la de la suscripción es un polígono. Por ende, se evalúa que la función st_intersects de PostGIS sea verdadera.
    - En (4), la geometría de la suscripción es de buffer. Además, se verifica que se haya reportado alguna localización del usuario de la suscripción y que esta no tenga antigüedad mayor a los 10 minutos. En este caso, se evalúan (5) y (6).
        - La geometría del evento es puntual y se evalúa la función st_within del punto con el buffer st_buffer con centro la localización del usuario y el radio de la suscripción
        - La geometría del evento es un polígono y se evalúa la función st_intersects del Polígono con el buffer st_buffer con centro la localización del usuario y el radio de la suscripción.