Instituto de Computación (INCO)
Facultad de Ingeniería
Universidad de la República

Dispositivos móviles como front-end de sistemas interactuando mediante Servicios Web

Usuario Responsable: Ing. Leonardo Mena (Swordfish)

Supervisor: Ing. Alejandro Gutiérrez (INCO)

Autores

Alejandra De León	**2.981.267-6**
Natalia Espíndola	**3.323.160-6**
Gonzalo Miños	**2.843.023-3**

Agosto 2003

Dispositivos móviles como front-end de sistemas interactuando mediante Servicios Web

Resumen

La necesidad de obtener la información pertinente en el momento y lugar preciso, ha extendido el uso de los dispositivos móviles inalámbricos. Sin embargo, las limitaciones de recursos en los dispositivos móviles es una característica que se debe tener en cuenta al trabajar con ellos. Por otro lado, a medida que la interacción entre distintas aplicaciones es más necesaria, el uso de los Servicios Web también se ha expandido. En el caso de los Servicios Web las funciones lógicas y de procesamiento de datos residen en el servidor, mientras que los clientes se encargan de un procesamiento mínimo y generalmente requieren poca memoria para almacenar los resultados, por lo que parece adecuado aplicar el modelo de los Servicios Web a los dispositivos móviles inalámbricos.

En este documento, se presenta un relevamiento de tecnologías disponibles para la interacción entre Servicios Web y dispositivos móviles inalámbricos, y un relevamiento de las posibles interfases gráficas soportadas por dichos dispositivos. Finalmente, se aplican los conocimientos adquiridos en la construcción de un prototipo que permite consumir desde dispositivos móviles, los Servicios Web de un Sistema Administrador de Bienes Digitales desarrollado por la empresa *Swordfish*.

Índice

Historial

Todo cambio realizado a este documento debe ser documentado en esta sección. No olvidarse de cambiar la versión en la carátula del documento.

Fecha	Versión	Descripción	Autor
2002-09-21	1.0	Creación del Documento	Todos
2002-10-03	1.1	Modificaciones del índice Correcciones	Todos
2002-10-19	1.2	Correcciones	Natalia Espíndola
2002-10-30	1.3	Resumen de KSOAP y agregué en la intro Info sobre Microsoft y Sun	Gonzalo Miños
2002-11-01	1.4	J2ME	Alejandra de León
2002-11-03	1.5	Disp.móviles en introd..	Natalia Espíndola
2002-11-26	1.6	Organización	Gonzalo Miños
2003-03-06	1.7	Organización. Ampliación de KSOAP, WingSoap, GUI.	Natalia Espíndola
2003-03-08	1.8	Agregado MIDP 2, correcciones	Alejandra de León
2003-03-19	1.8	Agregado diseño e intro Implementación	Gonzalo Miños
2003-03-29	1.9	Correcciones y conclusiones de contexto	Gonzalo Miños
2003-04-12	2.0	Reestructuración. Correcciones en Introducción, WS.	Natalia Espíndola
2003-04-16	2.1	Agregar información sobre *Engine* DAMS, correcciones	Alejandra de león
2003-05-08	2.2	Arreglos y agregados en el Contexto, Servicios Web	Alejandra de León
2003-05-09	2.3	Arreglos intro, Def del problema y Capítulo 4	Gonzalo Miños
2003-05-09	2.4	Arreglos en J2ME y NETCF	Natalia Espíndola
2003-05-25	2.4	Arreglos en Plataformas	Gonzalo Miños
2003-05-26	2.4	Agregados en Caso de Uso, Diagramas	Alejandra de León
2003-05-27	2.5	Correcciones de Caso de Uso, y diagramas	Todos
2003-06-10	2.5	Correcciones de Caso de Uso	Alejandra de León, Gonzalo Miños
2003-06-12	2.6	Correcciones de Relevamiento de Interfase Gráfica	Natalia Espíndola
2003-06-15	2.7	Conclusiones de los Casos de Uso	Todos
2003-06-24	2.8	Conclusiones de los Casos de Uso	Todos
2003-07-30	2.9	Conclusiones	Todos
2003-08-06	2.10	Ultima revisión	Todos

1 Introducción

Actualmente, la necesidad de obtener la información pertinente, en el momento y lugar preciso, ha extendido el uso de los dispositivos móviles inalámbricos. Sin embargo, en el desarrollo de aplicaciones para estos dispositivos, los desafíos que se deben afrontar, son totalmente diferentes a aquellos relacionados con las computadoras de escritorio.
En primer lugar, están los recursos limitados de que se dispone, como ser el pequeño tamaño de su pantalla, su memoria, y su procesador. También existen limitaciones relacionadas con las redes inalámbricas, como por ejemplo un ancho de banda más limitado, inestabilidad y menor confiabilidad que el resto de las redes. Todas estas restricciones requieren encarar de forma especial temas como la comunicación, la interfase gráfica de usuario y la administración de la memoria.
Con respecto a las interfases gráficas, existe una relación directa entre la interfase gráfica que se puede desarrollar y los recursos del dispositivo. Por ejemplo, si se consideran dispositivos móviles con gran poder de procesamiento y memoria (los hay de 200 MHz y 64 MB) es posible lograr interfases gráficas ricas.

Por otro lado, a medida que la interacción entre distintas aplicaciones empresariales es más y más necesaria, el uso de los Servicios Web se está expandiendo debido a que proveen un método para lograr esta interacción basado en XML y protocolos estándares. Además, en los Servicios Web las funciones lógicas y de procesamiento de datos residen en el servidor, mientras que los clientes se encargan de un procesamiento mínimo y generalmente requieren poca memoria para almacenar los resultados.

Teniendo en cuenta lo anterior, parece adecuado aplicar el modelo de los Servicios Web a los dispositivos móviles inalámbricos.

En cuanto al estado actual de las tecnologías disponibles para desarrollo de aplicaciones móviles, las empresas, Microsoft y Sun Microsystems, son las que han tomado la iniciativa en tal sentido.
De parte de Microsoft, se dispone del *.Net Compact Framework* (.Net CF) que tiene integrado soporte para Servicios Web. Por otro lado, Sun Microsystems ofrece la plataforma *Java 2 Micro Edition* (*J2ME*) para dispositivos móviles y junto con un equipo de partners, actualmente está desarrollando una especificación estándar de acceso a Servicios Web desde su plataforma.

Como se verá a lo largo de este documento, se relevaron las tecnologías disponibles para la interacción entre Servicios Web y dispositivos móviles inalámbricos, y las herramientas existentes para el desarrollo de interfases gráficas para dichos dispositivos. Como aplicación de los conocimientos adquiridos a partir de estos relevamientos, se realizó un prototipo que a través de una interfase gráfica, permite consumir Servicios Web expuestos por la empresa *Swordfish* desde dispositivos móviles como *PDA*s y celulares.

1.1 Definición del Problema

Técnicamente, un bien digital es todo aquello que es transformado en código binario y que tiene valor en término de productividad y evaluación para la empresa que lo posee. Entre bienes digitales se pueden incluir, por ejemplo, logotipos, fotos, e-mails, documentos de texto y archivos de multimedia. A medida que se acrecienta el volumen de bienes digitales manejado por las empresas, aumenta la necesidad de administrarlos.
Un Sistema Administrador de Bienes Digitales (*DAMS* siglas en inglés de Digital Asset Management System) [1] es una herramienta para organizar, almacenar y recuperar bienes digitales.

La empresa *Swordfish* ha desarrollado un Sistema Administrador de Bienes Digitales llamado *Engine*, el cual posee una *API* que expone sus funcionalidades mediante Servicios Web.
El interés de la empresa en este proyecto es la necesidad de investigar la interacción de Servicios Web con dispositivos móviles de forma inalámbrica, para potenciar el DAMS *Engine*, evaluando la viabilidad del sistema en ese entorno.
Complementando lo anterior se investigarán las diferentes posibilidades de interfases gráficas para estos dispositivos.

1.2 Objetivos

El objetivo general de este proyecto es realizar un relevamiento en el tema de acceso a Servicios Web (SW de ahora en adelante) desde dispositivos móviles investigando las posibles interfases gráficas para este tipo de clientes. Se analizan las tecnologías actuales disponibles y las tendencias a futuro desde un punto de vista empresarial.

Más en particular, los objetivos del proyecto son:

A. Identificación de mecanismos y relevamiento de tecnologías para la interacción entre dispositivos móviles y SW.
B. Identificación de mecanismos y relevamiento de tecnologías para la creación de interfases gráficas orientadas a distintos tipos de dispositivos móviles, en particular *PDA*s y teléfonos celulares.

   Ambos objetivos son independientes, y se pretenderán estudios y conclusiones en forma no solo específica sino también integrada.

C. Aplicación de lo anterior en un caso de estudio. En líneas generales consistirá en crear una aplicación cliente para acceder a los SW exportados por *Engine* desde un PDA o un celular.

1.3 Organización del Informe

De aquí en adelante, este informe está organizado de la siguiente manera:

Capítulo 2: Contexto
Describe conceptos, tecnologías y herramientas necesarias para el desarrollo del proyecto.

Capítulo 3: Relevamiento sobre la interacción de Servicios Web y dispositivos móviles
Presenta las distintas tecnologías disponibles actualmente para la interacción de SW y dispositivos móviles. Se las compara, brindando por último una conclusión.

Capítulo 4: Relevamiento sobre interfases gráficas para dispositivos móviles
Presenta las distintas herramientas para creación de interfases gráficas. Finalmente se explican los motivos de las herramientas elegidas.

Capítulo 5: Caso de estudio
Presentación, análisis, diseño e implementación del caso de estudio elegido.

Capítulo 6: Conclusiones
Resumen del proyecto, conclusiones de los resultados obtenidos y posibles trabajos futuros.

Capítulo 7: Referencias
Una lista de libros y links a artículos en los que se basó el desarrollo del proyecto.

Capítulo 8: Apéndices
Incluye información que profundiza en las distintas herramientas usadas. También contiene ejemplos y resultados de testeos realizados.

Capítulo 9: Glosario
Glosario de los términos utilizados.

2 Contexto

En este capítulo se da una definición de Servicios Web, se habla de sus características, tecnologías relacionadas y se muestra un ejemplo. Enseguida se explican distintos tipos de dispositivos móviles y se presenta el Sistema Administrador de Bienes Digitales (DAMS) *Engine* y sus funcionalidades. Para terminar se describen y comparan las plataformas de desarrollo para dispositivos móviles J2ME y .Net.

2.1 Servicios Web

"Servicio Web" es un concepto genérico, lo que implica que existen múltiples definiciones, algunas más aceptadas o conocidas que otras.

Desde nuestro punto de vista y para nuestra realidad, nos parecieron más adecuadas las siguientes:

Definición 1

"Un Servicio Web es un sistema de software identificado mediante una *URI*, cuyas interfases públicas y las asociaciones entre un protocolo concreto, un formato de datos y dichas interfases, se definen y describen usando XML. Otros sistemas pueden descubrir los detalles de la descripción del SW y así interactuar con él de la forma establecida por dicha descripción, usando mensajes XML sobre protocolos de Internet."

Web Services Architecture (W3C). Febrero 2003 [2]

Analizando los términos por separado:

- Servicio: indica que es una aplicación que expone sus funcionalidades a través de algún tipo de API, la que se describe mediante algún lenguaje de descripción de interfases. La infraestructura del servicio (*SOA* – Service-Oriented Architecture) provee algún mecanismo para permitir a otras aplicaciones descubrir el servicio y su descripción.
- Web: indica que es un recurso Web. Un recurso Web es todo lo que se identifica por una URI y puede ser accedido a través de protocolos de Internet (por ejemplo *HTTP*).

Por lo tanto, un Servicio Web es una aplicación que se identifica mediante una URI y expone sus funcionalidades a través de una API que se describe en XML. Existe un método que permite descubrir el servicio y la descripción de su interfase, y que debe usar XML para representar los datos del mensaje.

Definición 2

"Un Servicio Web es un componente de software reusable, débilmente acoplado, que encapsula funcionalidades semánticamente, es accesible sin interacción humana y distribuido sobre protocolos de Internet estándares."

Brent Sleeper and Bill Robins (The Stencil Group). Junio 2001[3]

Examinando esta definición según [3]:

- Componente de software reusable: Los desarrolladores pueden reutilizar y extender SW ya existentes para sus nuevas aplicaciones.
- Débilmente acoplados: Las aplicaciones pueden ser implementadas en diferentes plataformas y sistemas operativos, y también pueden cambiar sus implementaciones sin afectar la interfase de los SW, permitiendo así la integración de aplicaciones sin aumentar el acoplamiento entre ellas [4].
- Encapsula funcionalidades semánticamente: un SW es una aplicación con un conjunto de funcionalidades enfocadas a un propósito común. El SW describe sus propias entradas y salidas de forma que otras aplicaciones puedan determinar lo que hace, como invocar sus funcionalidades y que resultados esperar.
- Accesible sin interacción humana: los SW no están diseñados para una directa interacción humana, y no tienen una interfase gráfica de usuario. Operan a nivel de

código, son llamados por otras aplicaciones e intercambian datos entre ellas. Sin embargo, pueden incorporarse dentro de una aplicación diseñada para interacción humana.
- Distribuido a través de Internet: hacen uso de los protocolos de transporte existentes como HTTP.

2.1.1 Características de los Servicios Web

Los SW se caracterizan [CJ02] por estar basados en XML, ser débilmente acoplados y utilizar mensajes. También se presenta como otra característica el uso de estándares, ya que usa SOAP para la comunicación, *UDDI* para registrar el servicio y *WSDL* para describir la interfase del SW.

No dependen específicamente de ningún protocolo de transporte, pero si se construyen sobre una infraestructura estándar de Internet, aseguran un mayor soporte y alcance. En particular, toman ventaja del HTTP, el mismo protocolo de conexión usado por servidores web y browsers.

Usan XML como lenguaje de representación de datos, por lo que las tecnologías y protocolos de SW creadas pueden interoperar y se permite el intercambio de documentos.

Permiten a los clientes invocar procedimientos remotos (*RPC*, siglas en inglés de Remote Procedure Call), usando un protocolo basado en XML.

Un cliente de SW no está ligado a ellos directamente; la implementación del SW puede cambiar con el tiempo sin comprometer la habilidad del cliente para interactuar con el servicio, esto se debe a que los SW adoptan una arquitectura débilmente acoplada.

Otra característica de los SW es la posibilidad de ser sincrónicos o asincrónicos. La sincronización se refiere al vínculo del cliente con la ejecución de un servicio. En invocaciones sincrónicas, el cliente se bloquea y espera que el servicio complete su operación antes de continuar. Las invocaciones asincrónicas permiten a un cliente invocar un servicio y entonces ejecutar otras funciones. Los clientes asincrónicos recuperan su resultado posteriormente en el tiempo, mientras que los clientes sincrónicos reciben su resultado cuando el servicio termina.

2.1.2 Tecnologías de los Servicios Web

Los SW [3] no se implementan en forma monolítica, sino que representan una colección de diferentes tecnologías relacionadas. Como mínimo, establece una conexión entre dos aplicaciones, lo que se llama RPC, en la que consultas y respuestas se intercambian en XML sobre HTTP. Sin embargo, una definición más completa implica una implementación de una pila de estándares como se muestra en la figura 2.1. En la capa básica (capas inferiores de la pila) tenemos Protocolos Comunes de Internet (como TCP/IP, HTTP o SMTP), XML y SOAP. Aunque los SW no están atados a ningún protocolo de transporte específico por lo general utilizan HTTP, que es el mismo protocolo de conexión usado por los servidores web y los browsers. XML, por su parte, provee un metalenguaje usado para escribir lenguajes especializados para expresar interacciones complejas entre clientes y servicios o entre componentes de un servicio compuesto.
En la capa de Alto Nivel tenemos WSDL y UDDI, que se explicarán posteriormente.

Universal Description Discovery and Integration (UDDI)
Web Services Description Language (WSDL)
Simple Object Access Protocol (SOAP)
Extensible Markup Language (XML)
Protocolos comunes de Internet (TCP/IP, HTTP, etc.)

Figura 2.1 Pila de Tecnologías usadas en la Implementación de Servicios Web

Simple Object Access Protocol (SOAP)

SOAP [5] define un protocolo liviano para intercambio de información. Es un protocolo basado en XML que consiste en 3 partes:

- un sobre, que define un mecanismo para describir que es un mensaje y como se procesa
- un conjunto de reglas para representar tipos de datos
- una convención para representar invocaciones y repuestas RPC

Los mensajes SOAP pueden intercambiarse usando otros protocolos, como SMTP o *FTP*, pero la especificación actual del W3C solamente define las relaciones para HTTP.

Web Service Description Language (WSDL)

WSDL [6] describe, usando XML, la interfase de un SW y cómo otras aplicaciones pueden consumir los servicios que este ofrece. WSDL estandariza cómo un SW representa externamente los parámetros de entrada y salida de una invocación, la estructura de las operaciones y la naturaleza de la invocación (In, In / out, o out). También define un mecanismo de *binding* que es usado para relacionar un protocolo específico (como SOAP o HTTP) o un formato de datos (por ejemplo *MIME*) a un mensaje u operación. Se podría pensar en un WSDL como en un manual de instrucciones en donde se explica como un usuario puede obtener el servicio brindado.

Universal Description, Discovery, and Integration (UDDI)

UDDI [7] provee un método estandarizado para la publicación, el descubrimiento e integración de SW. Los desarrolladores de aplicaciones usan el UDDI para publicar servicios (poner información acerca del servicio en los registros del UDDI) y para descubrir información sobre los servicios disponibles haciendo consultas por nombre, identificadores o categorías.

2.1.3 Interacción simple de un Servicio Web

Como muestra la figura 2.2, la relación entre SOAP, WSDL y UDDI [CJ02] puede describirse como sigue: una aplicación actuando en el rol de un cliente de SW, lo que se conoce en el área de los SW como *consumidor*, necesita localizar otra aplicación en algún lugar de la red. Este cliente consulta un registro UDDI buscando el servicio por nombre, categoría, identificador o especificaciones soportadas. Una vez localizado en el registro UDDI, el cliente obtiene información sobre la ubicación de un documento WSDL. El documento WSDL contiene información sobre cómo contactar al SW y el formato de solicitud de mensajes en esquema XML. El cliente crea un mensaje SOAP de acuerdo con el esquema XML encontrado en el WSDL y envía una solicitud al host (donde está el servicio).

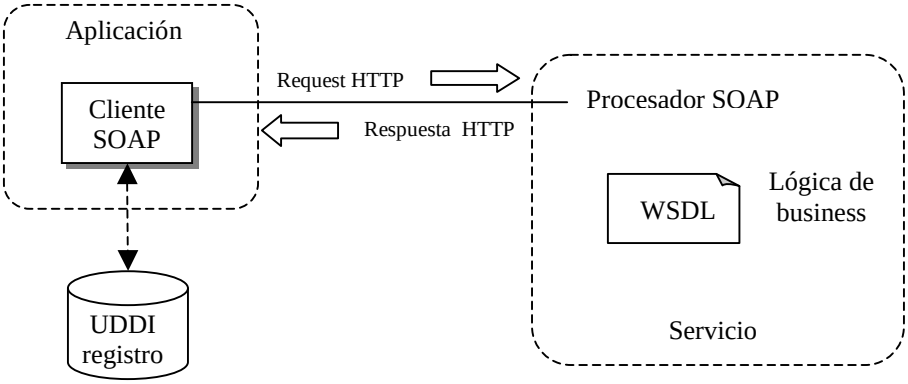

Figura 2. 2 Interacción simple de un Servicio Web

2.1.4 Ejemplo

En el ejemplo extraído de [8 b], se presenta un SW con una sola operación: getLastTradePrice. Se basa en SOAP 1.1 sobre HTTP. La operación recibe como parámetro un identificador (tickerSymbol) de tipo String y devuelve un parámetro de tipo Float que es el precio (price). Los elementos del WSDL que se presentan en el ejemplo son:

- <definitions> : contiene la descripción del servicio
- <types> : definición de tipos de datos. Se usan para describir el tipo de los datos usados en <message>.

```
<definitions name="StockQuote" targetNamespace="
http://example.com/stockquote.wsdl" xmlns:tns
=”http://example.com/stockquote.wsdl”
xmlns:xsd1 ="http://example.com/stockquote.xsd"
xmlns:soap ="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/”>

<types>
      <schema targetNamespace="http://example.com/stockquote.xsd"
      xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TradePriceRequest">
            <complexType>
                  <all>
                        <element name=”tickerSymbol" type="string"/>
                  </all>
            </complexType>
      </element>
      <element name="TradePrice">
            <complexType>
                  <all>
                        <element name="price" type="float"/>
                  </all>
            </complexType>
      </element>
      </schema>
</types>
```

- <message> : Definición de datos que van a ser intercambiados.

```
<message name="GetLastTradePriceInput">
      <part name="body" element="xsd1:TradePriceRequest"/>
```

```
</message>
<message name="GetLastTradePriceOutput">
      <part name="body" element="xsd1:TradePrice"/>
</message>
```

- <operation> : Descripción abstracta de una acción (nombre y parámetros de la operación).
- <portType> : Colección de operaciones.

```
<portType name="StockQuotePortType">
      <operation name="GetLastTradePrice">
            <input message="tns:GetLastTradePriceInput"/>
            <output message="tns:GetLastTradePriceOutput"/>
      </operation>
      <!-- Mas operaciones -->
</portType>
```

- <binding> : definición de protocolo a utilizar y formato de datos para un <portType>, como por ejemplo SOAP 1.1, HTTP, MIME

```
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
      <soap: binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="GetLastTradePrice">
            <soap: operation
            soapAction="http://example.com/GetLastTradePrice"/>
            <input><soap: body use="literal" />
            </input>
            <output><soap: body use=" literal" />
            </output>
      </operation>
</binding>
```

- <service> : Descripción del servicio. Puede contener uno o más elementos <port>.
- <port> : Define un Puerto de comunicación, como por ejemplo una URL para HTTP, o una dirección de correo electrónico para SMTP.

```
<service name=" StockQuoteService">
      <documentation> Mi primer servicio</ documentation>
      <port name="StockQuotePort" binding="tns:StockQuoteBinding">
            <soap:address location=" http://example.com/stockquote"/>
      </port>
</service>
```

Se detalla en el Apéndice 7.1 un ejemplo de código XML completo y de paquetes SOAP de pedido y respuesta.

2.1.5 Resumen

En resumen, podemos decir que los SW son un grupo de tecnologías que describen la arquitectura de una aplicación basada en componentes, orientada a servicios, con una infraestructura abierta, centrada en Internet y basada en estándares como XML, WSDL, UDDI y SOAP. Los componentes de los SW pueden ser reusados por otras empresas para satisfacer las necesidades de sus propias aplicaciones, es decir, que una empresa no necesita construir todas las componentes de su sistema, solo tiene que localizar los servicios necesarios y consumirlos. Como consecuencia se baja el costo de mantenimiento del sistema al reutilizar estos servicios.
Cuando se habla de SW, frecuentemente se resalta su característica de ofrecer bajo acoplamiento comparándolo con tecnologías un poco más viejas como *CORBA*, RMI y *COM*. Estas tres tecnologías están basadas en un modelo RPC y dependen de convenciones de lenguajes de programación o plataformas particulares. Aunque los SW también pueden ser usados como aplicaciones tipo RPC ofrecen, además, otra interfase basada en mensajes o documentos.

2.2 El Sistema *Engine*

Engine es un Sistema Administrador de Bienes Digitales (DAMS) creado por la empresa Swordfish con la idea de proteger, recuperar, organizar, transformar y distribuir bienes digitales. Como se mencionó antes un bien digital es todo aquello que sea transformado en código binario y que tiene valor en término de productividad y evaluación de la empresa. Entre bienes digitales se pueden incluir, por ejemplo, logotipos, fotos, videos, e-mails y documentos de texto.

2.2.1 Objetivos del Producto

Un Administrador de Bienes Digitales debe sobresalir en las siguientes áreas:

Orientación Digital
Esto incluye infraestructura, lógica y procesos para capturar y extraer el valor agregado de un contenedor de bienes digitales incluyendo derechos y permisos, rastreo de uso, y confiabilidad en las versiones.

Soporte de formatos multimedia
Un *DAM* necesita soportar todo tipo de formatos multimedia, incluyendo funcionalidades de almacenamiento, categorización, seguridad, navegación, interfase de usuario e integración con otras aplicaciones.

Arquitectura escalable
Un DAM tiene que crecer paralelamente con la empresa, aún cuando el tamaño y cantidad de los bienes digitales aumente exponencialmente.

Alta facilidad de integración
Un DAM es un componente del sistema de componentes de una empresa, por lo que integrarlo con otros sistemas debe ser fácil. Además, debe ser configurable a las necesidades de cada empresa.

Fácil de usar
Un DAM debe ser lo suficientemente sencillo de navegar como para que cualquiera lo pueda usar, y lo suficientemente potente como para soportar la mayoría de los requerimientos de cualquier usuario.

2.2.2 Descripción de Funcionalidades

A continuación se describirán los módulos que integran *Engine*.

Protección de los Bienes Digitales
Permite la transferencia e inserción de diferentes formatos de bienes digitales en el sistema.

Recuperación
Usa la información que un bien digital posee para buscar y recuperar bienes digitales.

Organización
Permite definir criterios básicos y avanzados de búsqueda usando carpetas, palabras clave, descriptores y otros metadatos importantes, dependiendo de las necesidades particulares del caso.

Distribución
Permite distribuir bienes digitales para hacerlos accesibles tanto a usuarios de la aplicación como a usuarios externos.

Control de Acceso
Define usuarios y roles, asigna roles a cada usuario permitiendo personalizar el esquema de seguridad según las necesidades de cada cliente, para obtener un almacenamiento confiable y seguro de los bienes digitales.

Monitoreo / Estadística
El estado general del sistema se audita continuamente. Presenta información estadística acerca de las acciones realizadas y los bienes digitales.

Conversión Automática
Transforma y convierte material digital de acuerdo a las necesidades y estándares del usuario, y de acuerdo a cada formato.

Workflow
Facilita interacciones básicas entre los usuarios responsables de los bienes digitales y otros usuarios.

Para una descripción más detallada de las funcionalidades de *Engine*, ver el Apéndice 7.2**.** No todas estas funcionalidades se exportan como SW. Las funcionalidades exportadas se detallaran en la sección Capítulo 5.2.4.
Los tipos complejos que integran la API de *Engine* se detallan en la sección 5.2.3.

2.3 Dispositivos Móviles

Entre los tipos de dispositivos móviles más populares tenemos los *PDA* (Personal Digital Assistant) y los teléfonos celulares.
Los PDA[10] son computadoras de mano que pueden conectarse a PCs e intercambiar información con ellos mediante cable o puerto infrarrojo. Además, pueden tener incluido módem inalámbrico.
Entre sus características principales podemos destacar que no incluyen teclado, solamente botones de acceso directo. Su principal interfase con el usuario es mediante el lápiz o stylus. Esto implica que deban incorporar reconocimiento de escritura manuscrita para lo que usan un alfabeto de símbolos que representan las letras (sistema Graffiti). Debido a su pequeño tamaño, la mayoría tampoco incluyen disco duro. Sin embargo, a muchos PDAs se les puede adherir o conectar discos, memoria extra, teclado, módem inalámbrico y otros dispositivos.
Entre las múltiples utilidades de los PDA, estos pueden funcionar como un celular, un fax, un browser o también como un organizador digital.

Por su parte, los celulares son teléfonos móviles que utilizan tecnología celular digital o análoga. Dependiendo de la tecnología de la red celular local, los celulares ofrecen varios servicios tales como Internet, e-mail, y envío de mensajes.

Los dispositivos móviles se suelen clasificar según sus recursos, en dispositivos de bajos y altos recursos. Cuando nos referimos a dispositivos móviles de bajos recursos se habla de dispositivos de hasta 2MB de memoria RAM y un procesador menor de 100MHz. Por otro lado, cuando se habla de dispositivos móviles de altos recursos se considera una RAM de entre 2 y 64MB y un procesador de al menos 100MHz.

Con relación a la conectividad de los dispositivos, en el caso de las PDA, se requiere equipamiento auxiliar para conectarse a una red inalámbrica, como ser un módem inalámbrico, o un cable de conexión entre la PDA y un celular con capacidad de transmisión de datos.

Los sistemas operativos más usados en dispositivos móviles son Windows CE de Microsoft y Palm OS de Palm Inc. para las PDAs, y Symbian OS de Symbian para los teléfonos celulares.
Windows CE es una versión del sistema operativo Windows diseñada para dispositivos móviles. La interfase de usuario de Windows CE es muy similar a la de Windows 95. Actualmente las

PDAs con Windows CE se conocen como *Pocket PC*. Estrictamente hablando, Pocket PC[20] es un software para PDAs que incluye componentes de aplicaciones específicas de un modelo de PDA y el sistema operativo Windows CE. Los modelos más conocidos son Compaq y Hewlet Packard.
Palm OS es un sistema operativo para PDAs desarrollado por la empresa Palm Inc., muy popular en estos últimos años. Incluye capacidades de multimedia y seguridad que han mejorado en la última versión. En lo relacionado a la seguridad, los desarrolladores pueden utilizar los servicios de encriptado para proteger los datos del usuario, y en lo relacionado a multimedia duplicó la resolución de pantalla y aumentó las capacidades de audio.
Symbian OS es un sistema operativo diseñado para teléfonos-computadora pequeños y móviles, con acceso inalámbrico a diferentes servicios de información. Incluye soporte telefónico integrado(como por ejemplo libreta de direcciones, datos del teléfono), protocolos de comunicación, manejo de datos, soporte de gráficos avanzado y una variedad de aplicaciones.

Con respecto al mercado de dispositivos móviles, un estudio realizado en Europa, Medio Oriente y Asia [36] con datos del primer trimestre del 2003, indica que la situación actual es opuesta a la del año pasado. El mercado Symbian [37] ha crecido cerca del 800%, llegando al 53% de mercado total de dispositivos móviles. Los dispositivos con sistema operativo Windows CE se han colocado por encima de los que usan Palm OS, manteniendo una relación de 24% a 19%, mientras que el año pasado la situación era al revés. Si dividimos el mercado entre PDAs y celulares, en el caso de PDAs, Windows CE es el líder con un 48% frente a un 42% de Palm, mientras que el año pasado Palm tenía un 49% y Windows un 39%. Por el lado de los celulares, Symbian está en el 91% de los teléfonos móviles.

2.4 Plataformas de Desarrollo

Según su definición [24], una plataforma define un estándar, en cuanto a componentes de hardware y software, sobre el que puede ser desarrollado un sistema de computación. Como primer paso del proyecto se comenzó investigando las plataformas para desarrollo de aplicaciones móviles disponibles en la actualidad, teniendo en cuenta los objetivos planteados. En base a esto, en las 2 secciones siguientes se presenta una visión general de J2ME de Sun Microsystems y de .NET CF de Microsoft.

2.4.1 Java 2 Platform Micro Edition

Java 2 Platform Micro Edition (J2ME) creada por Sun Microsytems [9] es una plataforma de desarrollo de aplicaciones Java para dispositivos móviles. J2ME es una colección de tecnologías y especificaciones que están diseñadas para la amplia gama de dispositivos móviles existentes en la actualidad.

Los dispositivos a los que apunta la plataforma J2ME se pueden clasificar en dos categorías:

Dispositivos móviles personales
Esta categoría describe dispositivos que tienen un propósito especial o que tienen funciones limitadas, y que soportan conexión de red intermitente (como celulares, pagers y PDAs de bajos recursos).

Dispositivos móviles con conexión compartida
Esta categoría describe dispositivos que tienen una gran capacidad de interfase gráfica y mayor poder de procesamiento que la categoría anterior. Son dispositivos con una conexión de red fija e ininterrumpida, como por ejemplo *set-top boxes*, teléfonos móviles con acceso a Internet, sistemas de navegación de automóviles y PDAs de alto procesamiento.

Básicamente, lo que distingue a estas dos categorías es el poder de procesamiento de que disponen los dispositivos móviles, por lo que volviendo a la sección 2.2, la primera categoría se corresponde con dispositivos de bajos recursos y la segunda con dispositivos de altos recursos.

J2ME utiliza un diseño modular para lograr un soporte para múltiples tipos de dispositivos. Esto lo hace mediante *configuraciones* y *perfiles*. [35]
Una *configuración*[25] define una plataforma Java mínima para una familia particular de dispositivos. Está compuesta por una máquina virtual Java y un conjunto mínimo de bibliotecas.
Las configuraciones se pueden combinar con un conjunto de APIs de alto nivel o *perfiles,* para lograr un entorno de ejecución Java completo, y cubrir los requerimientos de un amplio rango de dispositivos. Las distintas combinaciones de estos elementos se optimizan según las características de memoria, de procesador, la interfase de usuario y las capacidades de entrada-salida relacionadas a cada categoría de dispositivos.
La Figura 2.3 muestra la plataforma J2ME formada por un conjunto de capas. Sobre el sistema operativo está la capa que constituye un entorno de ejecución básico compuesto por bibliotecas Java y una máquina virtual (VM) que forman parte de una configuración y un conjunto de APIs a nivel del sistema. En la capa siguiente hay un conjunto de APIs de alto nivel que forman un perfil.

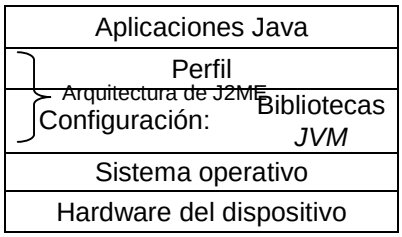

Figura 2.3 Plataforma J2ME

Todas las especificaciones de las configuraciones y los perfiles son desarrolladas por el *Java Community Process* (*JCP*)[34]. Una especificación comienza su ciclo de vida como una *Java Specification Request* (*JSR*) y pasa por varias etapas en el JCP antes de darse por finalizada. A todos los JSR se les asigna un número con el cual se les identifica comúnmente.

Configuraciones

Todos los miembros de una familia de dispositivos en particular tienen requerimientos similares de memoria y procesador. Una configuración [25] de J2ME define una interfase de programación (API) adecuada a dispositivos que pertenecen a cierta familia, así como también las disponibilidades a nivel del sistema y las características de la máquina virtual presente.

Una configuración especifica tres elementos básicos:

- Un subconjunto del lenguaje de programación Java.
- Una máquina virtual. Una máquina virtual permite a las aplicaciones escritas en el lenguaje de programación Java, ser portables entre entornos de hardware y sistemas operativos.
- Un subconjunto de bibliotecas y de APIs.

Hasta el momento hay definidas dos configuraciones:

Connected, Limited Device Configuration (CLDC)

Esta configuración está diseñada para dispositivos móviles personales, que como ya se mencionó tienen conexión de red intermitente y restricciones de procesador y memoria. Estos dispositivos generalmente tienen entre 16 y 32 bits de CPU, y de 128 a 512 KB de memoria disponible para la implementación de la plataforma Java y aplicaciones asociadas.
Esta configuración especifica la máquina virtual *KVM* (Kilobyte Virtual Machine)[29]. La KVM fue diseñada de forma de reducir el tamaño de la máquina virtual y bibliotecas de clases de Java 2 Standar Edition (*J2SE*), reducir la memoria que utiliza durante la ejecución, y permitir que sus componentes sean configurados para adecuarse a dispositivos particulares. Tiene un tamaño

aproximado de 70KB y está incluida en la implementación de referencia (RI) de *CLDC* 1.0 y CLDC 1.1.
Además, existe la implementación CLDC HotSpot, que es una máquina virtual que optimiza la performance de la KVM. Está enfocada a dispositivos de 16 a 32 bits de CPU, y como mínimo 192 KB de memoria disponible para implementación de la plataforma Java. Es necesaria una licencia comercial de Sun MicroSystems para poder utilizarla.
CLDC y sus perfiles incluyen los paquetes java.io, java.lang.* y java.util.* de *J2SE,* y agrega el paquete javax.microedition.io.* que brinda funcionalidad para conexión [38].

Connected Device Configuration (CDC)

Esta configuración está diseñada para dispositivos de altos recursos con conexión de red fija e ininterrumpida. Estos dispositivos generalmente tienen 32 bits de CPU, y un mínimo de 2MB de memoria disponible para la implementación de la plataforma Java y aplicaciones asociadas.
CDC especifica la máquina virtual *CVM* (Compact Virtual Machine)[39], que actualmente se conoce como implementación CDC HotSpot.
CDC y sus perfiles incluyen los paquetes java.io, java.lang.*, java.net, java.security.*, java.text y java.util.*. También agregan el paquete javax.microedition.io.* y javax.microedition.xlet.* [40].

La Figura 2.4 muestra la relación entre las dos configuraciones y la plataforma *J2SE*. CLDC, a pesar de ser diferente, es un subconjunto de CDC. Pero, ninguna de las dos está completamente incluida en J2SE, ya que ambas agregan nuevas clases necesarias para desarrollar servicios para sus respectivas categorías de dispositivos. Por otro lado, aunque las dos configuraciones son independientes, no deberían usarse juntas para definir una plataforma. Una implementación de la plataforma J2ME, puede contener solo una de estas dos configuraciones.

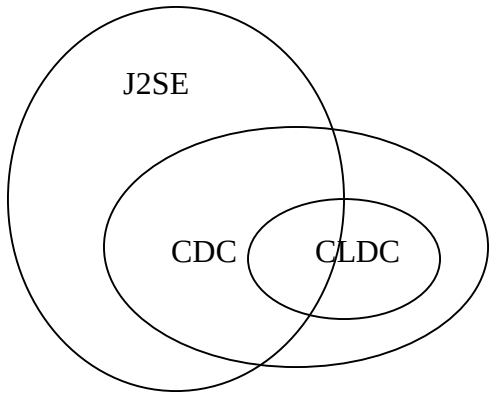

Figura 2.4 Relación entre CLDC, CDC y J2SE

Perfiles

Un perfil [25] especifica una interfase a nivel de aplicación para una clase particular de dispositivo. La implementación de un perfil se construye por encima de una configuración y consiste en un conjunto de bibliotecas Java que proveen esta interfase a nivel de aplicación.
Las configuraciones deben combinarse con perfiles para proveer un ambiente de ejecución de aplicaciones Java para la categoría de dispositivos que representan.

Las aplicaciones se construyen sobre las configuraciones y los perfiles, y pueden usar solo las bibliotecas que proveen estos dos niveles más bajos. Los perfiles pueden implementarse uno sobre el otro.

Perfiles del CLDC

En la Figura 2.5 se muestra un esquema del CLDC y sus tecnologías relacionadas.

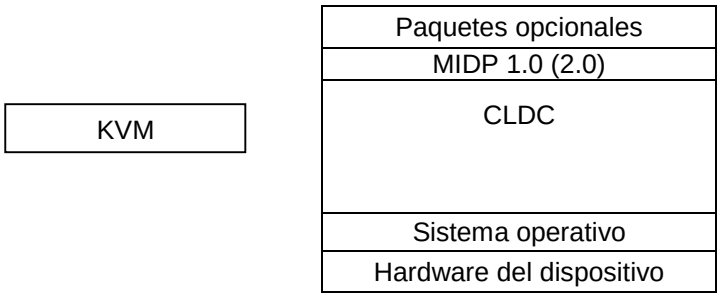

Figura 2.5 J2ME CLDC y tecnologías relacionadas

Mobile Information Device Profile (MIDP)
El *MIDP* provee una plataforma y un conjunto de APIs Java para dispositivos móviles de recursos limitados tales como teléfonos celulares y PDAs de bajos recursos. Brinda funcionalidades básicas como interfase de usuario, almacenamiento de datos local y persistente, conectividad de red y administración del ciclo de vida de las aplicaciones.
Las aplicaciones del MIDP se llaman *MIDlet* y extienden la clase MIDlet. Básicamente son aplicaciones Java que usan el perfil MIDP y la configuración CLDC. La Suite MIDlet es el nombre que se le da a la colección de todos los archivos y recursos necesarios que se necesitan como parte de un MIDlet. La *Suite MIDlet* consiste en: Un archivo *JAR* (Archivo de clases Java), un archivo manifiesto que describe el contenido del archivo JAR, recursos como por ejemplo imágenes y un archivo *JAD*(Archivo Descriptor de Aplicación Java).

Los paquetes que agrega el MIDP son javax.microedition.*lcdui*, javax.microedition.rms y javax.microedition.midlet.
Actualmente, es el único perfil disponible para CLDC.
Ver Apéndice 7.3 para detalles sobre MIDP.

Personal Digital Assistant Profile (PDAP)
Al inicio del proyecto este perfil estaba en desarrollo por el grupo JSR75[26] del JCP. Se enfocaba a dispositivos como PDA potenciando muchas de sus capacidades. Por este motivo, se consideró como una posible opción a utilizar. Sin embargo, en Enero del 2003 [51] el grupo del JSR75 llegó a un consenso de que el *PDAP* no era una solución viable para J2ME debido a sus condiciones actuales. Algunos de los factores en esta decisión incluyen el costo de este perfil, su fuerte relación y solapamiento con el MIDP y el Personal Profile, y la evolución constante de las necesidades del mercado. Entonces cambiaron la especificación original a un conjunto de paquetes opcionales para PDAs y otros dispositivos móviles, basados en CLDC y J2ME, que brindarían acceso a Personal Information Management (*PIM*), y acceso a sistemas de archivos a través de Generic Connection Framework (*GCF*).

Perfiles del CDC

La Figura 2.6 muestra el CDC y sus tecnologías relacionadas.

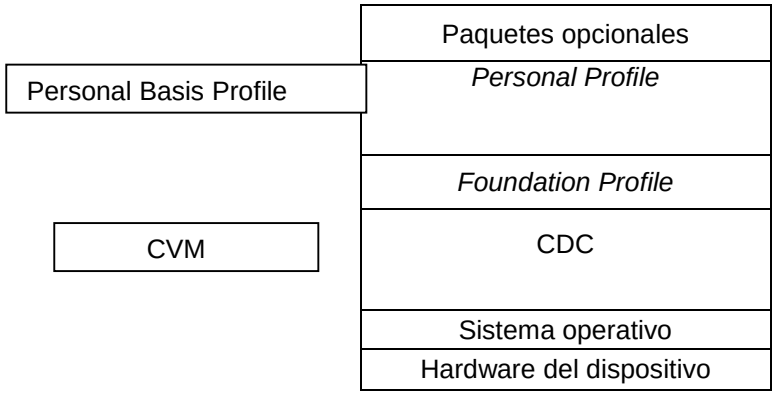

Figura 2.6 CDC y sus tecnologías relacionadas

Foundation Profile (FP)
El Fundation Profile es el perfil de nivel más bajo para CDC. En general se usa en implementaciones que no necesitan interfase de usuario, pero puede combinarse con el Personal Profile para dar soporte *GUI* a dispositivos que lo requieran.

Personal Profile (PP)
El Personal Profile extiende al FP para proveer soporte para interfases gráficas de usuario. Para esto incluye la biblioteca Java Abstract Window Toolkit (AWT) completa.

En el capítulo 4 se hablará de las capacidades para desarrollo de interfases gráficas que brinda el CDC y sus perfiles, para dispositivos móviles de altos recursos.

2.4.2 .NET Compact Framework

.NET CF [27] es un subconjunto del Microsoft .NET Framework, diseñado para desarrollar aplicaciones para dispositivos móviles. Aunque está diseñado para correr en múltiples plataformas y sistemas operativos, actualmente está disponible solo para dispositivos con el sistema operativo Windows CE, como dispositivos Pocket PC.

.NET CF consta de dos componentes principales: el Common Language Runtime (CLR) y una biblioteca de clases de .NET CF.

Common Language Runtime
Es responsable de la administración del código en tiempo de ejecución, brindando servicios del núcleo como administración de memoria y administración de hilos, mientras implementa la seguridad y exactitud del código. Puede verse como análoga a la máquina virtual de Java.

.NET Compact Framework class library
Es una colección de clases reutilizables para el desarrollo de aplicaciones, estrechamente integrada con el CLR. Es portable tanto en plataformas de Microsoft como de terceros, siempre que exista una versión de .NET CF para dicha plataforma.

Entre algunas de las funcionalidades que ofrece, .NET CF implementa un subconjunto de las clases encontradas en el .NET Framework, incluyendo soporte para formularios, la mayoría de sus controles, soporte para controles de terceros, mapas de bits y menús (clases System.Windows.Forms y System.Drawing). Implementa también un subconjunto de las clases

System.Data y System.XML para incorporar datos, ya sea de fuentes de datos relacionales como no relaciones, incluyendo contenido XML, en las aplicaciones móviles.
Además provee un conjunto de clases para crear aplicaciones multi-hilos, potenciar recursos de red, y trabajar con archivos (clases System.Threading, System.Net y System.IO).
Los SW están integrados en Visual Studio .Net, el ambiente de desarrollo de .NET CF, mediante bibliotecas de clases que ofrecen servicios de descubrimiento, descripción y soporte para protocolos como SOAP. La clase System.Web es la que brinda soporte para facilitar el consumo de SW (crear clientes de SW) de forma sincrónica o asincrónica. En general, los desarrolladores no tienen que escribir mucho código y pueden tratar al SW como un objeto local, invocándolo como si fuera una función, pero en realidad están haciendo una llamada remota al SW usando SOAP.

2.4.3 Resumen

J2ME y .Net CF son plataformas para desarrollar aplicaciones para dispositivos móviles y abarcan prácticamente el 100% del mercado actual.

La tabla 2.7 muestra un cuadro comparativo de estas plataformas.[28]

	.Net Compact Framework	**J2ME Connected Device**	**J2ME Connected Limited Device**
Requerimientos del dispositivo	Potente, caro	Potente, caro	Barato, extendido
Costo	Alto	Alto	Medio
Foco del mercado	Empresa	Empresa	Consumidores y empresas
Soporte de lenguaje	C#, VB.Net	Java	Java
Plataformas	Pocket PC, Windows CE	Todas las plataformas móviles excepto Palm OS	Todas las plataformas
Compatibilidad de código byte	Estándar .Net CLR	J2SE	No es compatible con J2SE o CDC
Compatibilidad de la API	Subconjunto de .Net	Subconjunto de J2SE más paquetes opcionales estándares	Compatibilidad parcial con CDC, con paquetes adicionales estándares
Herramientas de desarrollo	VS.Net 2003	Línea de comando, vendedores de SDKs, CodeWarrior, y WebSphere	Línea de comando, vendedores de SDKs, e *IDE*s de Java
Proceso de especificación	Microsoft	Comunidad (JCP)	Comunidad (JCP)

Tabla 2.7 .NET CF, J2ME CDC y CLDC, comparación de características principales

.Net CF está orientado a aplicaciones empresariales con interfases de usuario ricas, manejo de bases de datos y soporte para SW. .Net CF brinda un conjunto homogéneo de herramientas y

APIs para desarrollo pero solo corre sobre dispositivos que soporten la plataforma Windows CE de Microsoft. Cuando aparece una nueva tecnología, Microsoft la hace disponible rápidamente en .Net CF. Sin embargo, es Microsoft y no los clientes los que deciden cuales son las características importantes para agregar en .Net CF.

J2ME brinda un diseño modular y es portable entre una amplia variedad de dispositivos. Provee un soporte balanceado tanto para aplicaciones empresariales como para aplicaciones de consumidores. Sus APIs siguen un proceso de estandarización riguroso para asegurar un amplio soporte de las empresas y partners involucrados (mediante el JCP) y facilitar el aprendizaje de los desarrolladores.
En conclusión, si se trabaja en un mercado donde predomina Microsoft, .Net CF y las herramientas de Visual Studio .Net ayudarán definitivamente a portar aplicaciones empresariales a dispositivos móviles y a bajar los costos de desarrollo de la comunidad de desarrolladores Windows. Sin embargo, si se está en un ambiente heterogéneo o se necesita una solución que funcione en dispositivos de bajos recursos o que sea portable entre una amplia variedad de dispositivos, se debería optar por J2ME.

3 Relevamiento sobre la interacción de Servicios Web y dispositivos móviles

El objetivo de esta sección es presentar los servicios web inalámbricos, sus limitaciones y ventajas y los tipos de tecnologías que se utilizan en la interacción entre éstos y los dispositivos móviles. Finalmente, veremos algunos ejemplos de estas tecnologías y concluiremos cual herramienta usar en el prototipo.

3.1 Servicios Web Inalámbricos

La elección de tecnología inalámbrica [44] se debe a la conveniencia de acceder a la información precisa, desde cualquier lugar, en cualquier momento, y a la necesidad de los empresarios de bajar costos y lograr mayor productividad, dando valor agregado a sus sistemas y extendiendo sus funcionalidades existentes, o automatizando procesos de negocios.
La naturaleza dinámica de la Web requiere una nueva especie de servicios dinámicos que involucran a muchos proveedores de servicios. Cada servicio móvil simple requiere, a su vez, de otros servicios.

Por ejemplo, una aplicación de manejo de stock necesita servicios como autenticación y autorización de usuario, administración de clientes, cotización y ventas. Estos servicios se construyen sobre otros servicios básicos, como conectividad y seguridad en la Web.
El mejor modelo para esto es tener varios proveedores, uno para cada servicio básico. Si consideramos cada servicio básico como un SW, se tendría un proveedor por cada SW. Para minimizar el uso de ancho de banda y CPU, limitaciones comunes en los dispositivos móviles, los usuarios pueden utilizar sólo los servicios básicos que realmente necesitan. A estas aplicaciones que utilizan una interfase inalámbrica con el usuario y por detrás utilizan SW para potenciar los recursos de información de la Web, se les llama SW inalámbricos.

El uso de los SW [30] en los dispositivos inalámbricos abre nuevas posibilidades:

- Integrar de forma transparente aplicaciones y datos: Los SW se exponen a través de una API de usuario. De esta forma las aplicaciones y datos se pueden portar a través de plataformas y ambientes.

- Eliminar la arquitectura de cliente pesado para dispositivos móviles: Un cliente liviano es una arquitectura que permite a un dispositivo intercambiar datos con el servidor realizando el mínimo procesamiento a nivel del cliente. En cambio un cliente pesado permite procesamiento local y almacenamiento de datos locales, lo que posibilita a los usuarios móviles de trabajar, aún sin mantener una conexión con el servidor.
  Como los SW pueden usarse para construir clientes livianos, las empresas han pensado reemplazar con ellos las arquitecturas de clientes pesados, pues son de mayor costo. Sin embargo, la mayoría de las arquitecturas de los SW asumen que los dispositivos están siempre conectados, lo cual no es cierto en la realidad, por lo que las aplicaciones móviles necesitan ser una combinación de ambas arquitecturas.
  Por ejemplo, un empleado va a una tienda para hacer un inventario con un PDA y no puede usar el dispositivo dentro de la tienda porque el sistema de seguridad y la estructura del edificio interfieren con la señal inalámbrica. Con una combinación de arquitecturas de clientes liviano y pesado, el empleado puede conectarse al servidor antes de entrar y bajar la sección de la base de datos que necesita. Luego ingresar a la tienda y realizar el inventario. Después de salir de la tienda, puede conectarse nuevamente y transmitir los datos. Si el PDA del ejemplo fuera estrictamente un cliente liviano, no se hubiera podido utilizar dentro de la tienda.

- Preparar y extender los datos empresariales a los dispositivos móviles: Los SW actúan como interfases a través de las que un servidor de sincronización puede intercambiar datos entre una base de datos móvil y una base de datos empresarial. De esta manera

se minimiza el trabajo de integración, lo que baja los costos de producir aplicaciones móviles efectivas.
Además, al hacer disponibles vía SW los datos requeridos por un dispositivo móvil, el trabajo de integración tradicional hecho por APIs propietarias se minimiza o directamente se elimina, reduciendo dramáticamente el costo de desarrollo de aplicaciones móviles efectivas. En otras palabras, una vez que se crea un SW para un portal por ejemplo, el mismo SW puede ser usado por una aplicación móvil que necesite los mismos datos, sin mayores esfuerzos aprovechando además la sencilla integración de los SW.

3.2 Relevamiento de las tecnologías

En principio se analizaron las soluciones que proveen las plataformas J2ME y .NET CF, descriptas en la sección 2.4.
J2ME no provee ningún componente nativo ni especificación estándar para clientes de SW móviles. Sin embargo, uno de los grupos de trabajo del JCP que Sun coordina, está trabajando en una especificación para SW de J2ME [32]. Se espera para mediados del 2003 una primera versión de esta especificación.
De igual forma, Microsoft con su .NET Compact Framework, busca alcanzar los mismos objetivos y ya tiene integrado el soporte para SW, pero solo para dispositivos con sistemas operativos propietarios de Microsoft como Windows CE.

En la ausencia de especificaciones estándares para la plataforma J2ME, se orientó la investigación al análisis de herramientas, como parsers XML o implementaciones de alguno de los protocolos estándares de SW que facilitaran la comunicación.
En la Figura 3.1 se puede observar un esquema básico de las diferentes herramientas necesarias para lograr la interacción entre un cliente en un dispositivo móvil y un servidor de SW.

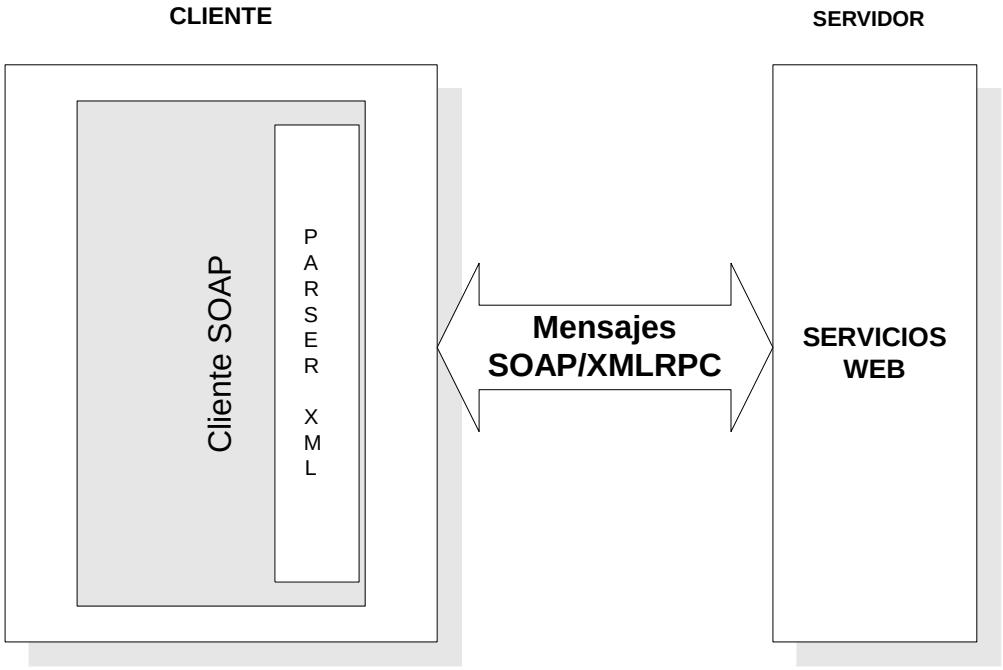

Figura 3.1 Esquema de interacción entre tecnologías

En esta figura se muestra como un servidor y un cliente se comunican mediante SW, intercambiando mensajes XML con cierto formato estándar, según lo especifica el protocolo de envío e intercambio de información utilizado. Los protocolos más aceptados son XMLRPC y SOAP.
Además existen clientes de estos protocolos que pueden recibir, interpretar y responder dichos mensajes. Estos clientes utilizan un parser XML que interpreta el código XML y chequea que éste siga algún patrón en particular.

A continuación detallamos los protocolos y las implementaciones de los mismos analizadas, enfocadas al entorno de los dispositivos móviles.

3.2.1 Parsers XML

Un parser XML [48]es un programa que lee un documento XML y chequea que siga algún patrón determinado. Intenta identificar que funciones cumple cada una de las partes del documento y lo deja accesible en memoria para algún lenguaje de programación que después lo interprete. Los patrones utilizados para interpretar el documento y validarlo son conocidos como DTDs.
Un DTD es una descripción formal para un tipo particular de documento, en una sintaxis de declaración XML. Define que nombres deben ser usados para determinados tipos de elementos, donde podrían aparecer y como se complementan estos elementos.

La mayoría de los parsers XML implementan una o más APIs para trabajar con XML. Entre ellas consideramos:

- SAX (Simple API for XML) [47] es una API basada en eventos. La aplicación le da el control al parser, pasándole el documento como parámetro. El parser lee el documento, lo divide en componentes, y llama a un manejador de eventos por cada uno. Estos manejadores de eventos son implementados por la misma aplicación. Un parser XML que soporta la API SAX se dice que cumple un modelo de parseo “push” siendo el SAX considerado un estándar para este modelo de parseo [46].
- XMLPULL [48] es una API similar a SAX. Sin embargo, en XMLPULL se usa el modelo de parseo “pull”, donde en vez de que el parser llame al manejador de eventos para cada componente del documento, la aplicación invoca al parser, el que retorna el próximo componente.
- DOM (Document Object Model), desarrollado por el W3C, es una API basada en una estructura de árbol. Los parsers XML que implementan el DOM [45] crean un modelo de objetos genérico en memoria que representa el contenido del documento XML. Una vez que el parser XML completa el parseo, se obtiene como resultado un árbol de objetos DOM que tiene la información del documento, y la aplicación puede navegarlo.

Básicamente, parsear un documento XML es como recorrer un flujo de datos. En el modelo push, el parser controla el recorrido y la aplicación es un cliente del parser. En el modelo pull, la aplicación misma controla el recorrido, y el parser es una herramienta de ayuda.

En particular se analizó *kXML* que es un parser de estilo pull desarrollado por tecnologías Lutris. Debido a su pequeño tamaño (21KB), es conveniente para aplicaciones que corran en dispositivos móviles. Este parser tiene las siguientes características:

- Soporte de namespaces para XML [49]
- Modos flexibles para parsear *HTML* y otros formatos
- Requiere muy poco espacio de memoria

El siguiente ejemplo [50] muestra como parsear un documento con kXML:

```
try {
  c =
  (StreamConnection)Connector.open("http://www.yourserver.com/file.xml")
  ;
  s = c.openInputStream();

  DefaultParser parser = new DefaultParser(new InputStreamReader(s));
  Document document = new Document();
  document.parse(parser);

  Element root = document.getRootElement();
  int count = root.getChildCount();

  for (int i=0; i < count; i++) {
     if (root.getType(i) == Xml.ELEMENT) {
        Element l = root.getElement(i);
        System.out.println("Element Name=" + l.getName());
     }
  }
}
catch (IOException err) {
  System.out.println(err);
  err.printStackTrace();
}
```

3.2.2 Cliente XML for Remote Procedure Calls (XML-RPC)

XML-RPC es una especificación [41] y un conjunto de implementaciones que permiten a aplicaciones que se ejecutan en diversos sistemas operativos y corren en diferentes entornos, realizar invocaciones a procedimientos remotos vía Internet. Estas invocaciones utilizan HTTP como protocolo de transporte y XML como formato de codificación.
XML-RPC provee la mínima funcionalidad necesaria para especificar tipos de datos, pasar parámetros e invocar procedimientos remotos en una forma neutral e independiente, utilizando XML. Requiere poca cantidad de memoria para procesar, parsear y construir los mensajes a ser intercambiados entre cliente y servidor.
XML-RPC fue desarrollado con el objetivo de ser rápido, liviano y eficiente. Sólo define seis tipos de datos simples (int, string, date, base64, boolean y double) y dos complejos (estructurados y arreglos), los cuales proveen en la gran mayoría de los casos, toda la funcionalidad requerida.

kXML-RPC

Es un proyecto [18] de código abierto de Tecnologías Lutris que implementa el protocolo XML-RPC y está orientado a la plataforma J2ME de Sun Microsystems.
kXML-RPC utiliza el parser kXML, que se mencionó anteriormente, para manejar los detalles de bajo nivel del parseo de XML. El tamaño de esta implementación es muy pequeño, de apenas 25 KB con el parser XML incluido.

3.2.3 Clientes SOAP

SOAP[5], como se mencionó anteriormente, es un protocolo de mensajes basado en XML utilizado para codificar la información de los mensajes de solicitudes y respuestas de los SW, antes de enviarlos por una red. Los mensajes SOAP son independientes de cualquier sistema operativo o protocolo y pueden ser transportados utilizando una amplia variedad de protocolos de Internet, incluidos SMTP y HTTP.
Un cliente SOAP o parser SOAP está construido sobre un parser XML con tipos de mapeos especiales y mecanismos de procesamiento y empaquetado de los datos. Un parser de SOAP

interpreta la información de los tipos de datos en los mensajes SOAP y automáticamente los convierte a tipos de datos del lenguaje de programación destino. El valor de este tipo de parsers es que proveen independencia de programación entre el mensaje SOAP y el programa.
A continuación se detallan dos clientes SOAP particulares para Java en un entorno inalámbrico.

WingSOAP

Es una implementación liviana de SOAP 1.1 [21] desarrollado por la Wingfoot Software, que incluye un parser XML para los detalles de parseo. Puede ser usado en plataformas MIDP/CLDC, *Personal Java*/CDC, J2SE y *J2EE*, pero por su pequeño tamaño (35 KB) está específicamente orientado a la plataforma J2ME.

kSOAP

kSOAP [22] es un cliente SOAP para la plataforma J2ME desarrollado por las Tecnologías Lutris[16].
Brinda un subconjunto de las características de SOAP, estándar de la W3C. La razón de no incluir todas las características de SOAP, es para contemplar las restricciones de memoria de los dispositivos móviles. Por ejemplo, no soporta los arreglos multidimensionales. Sin embargo, es posible agregarle soporte para tipos de datos tales como base64 o decimales.
La última versión estable disponible en este momento, es la versión kSOAP 1.2 (adopta SOAP 1.2), que ha sido incluida en los ambientes de desarrollo para dispositivos móviles de IBM y Sun por ejemplo. Producto de la aprobación que ha tenido kSOAP en la comunidad Java, es que ya existe una nueva versión mejorada del mismo, kSOAP 2.0 basada en kXML2 (versión 2.0 de kXML), pero todavía no se considera una versión estable.

En la tabla siguiente se resumen las características de las herramientas que implementan los protocolos analizados.

Características	Cliente XML-RPC	Cliente SOAP
Performance	Protocolo liviano, rápido y con bajo procesamiento	Baja performance por su complejidad y robustez
Tipos de datos	Simples	Simples y definidos por el usuario
Especificación	Basado en el protocolo XML-RPC, que aunque es muy usado, no está estandarizado	Basado en el protocolo SOAP, estándar de intercambio de SW según la W3C

Tabla 3.2 Comparación entre XML-RPC y SOAP

3.3 Barreras tecnológicas de los Servicios Web Inalámbricos

Parte de las ventajas de la adopción de los SW móviles es el cambio de clientes pesados hacia clientes livianos. Construir estos clientes puede ser una buena opción para sistemas sobre una *LAN*. Sin embargo, aunque es teóricamente posible en dispositivos móviles, estos dispositivos presentan restricciones únicas no encontradas en las LAN [31]:

- Pérdida de conexión: en la realidad, los dispositivos móviles no tienen cobertura en todo lugar en todo momento. Por ejemplo, dentro de ciertos edificios, bajo puentes o en lugares remotos, no se puede acceder a los servicios debido a que el dispositivo no se puede conectar. Usualmente las áreas urbanas tienen mejor cobertura.

- Tiempo de espera para la recuperación de datos: las conexiones inalámbricas son lentas, lo que resulta en largas esperas entre el instante en que el usuario solicita información y el instante en que la recibe.

- Altos costos de conexión: los ratios de uso de las redes inalámbricas han bajado con el avance de esta tecnología pero para empresas grandes, pueden seguir siendo muy costosos.

Para superar estas barreras mencionadas, las empresas de tecnología inalámbrica tienen grandes esperanzas puestas en las redes de tercera generación (*3G)*. A pesar de que esta nueva tecnología va a mejorar mucho la velocidad y la performance, quizás no sea tanto como se espera [43]. Debido a que el ancho de banda es compartido y sumado al aumento vertiginoso de la cantidad de dispositivos móviles, seguramente el crecimiento de la velocidad de transmisión va a ser mucho más bajo de lo esperado.
Sumado a lo anterior, el proceso de disponibilidad de las tecnologías 3G, iniciado ya hace algunos años en el hemisferio norte, ha sufrido grandes demoras por el colapso del mercado de las telecomunicaciones en el 2001 [52]. Esto ha provocado también que los estándares de las tecnologías 3G sigan siendo inciertos y hayan variado bastante desde el 2001 bajando los requerimientos, lo que hace por ejemplo que tecnologías previamente encasilladas como 2G, ahora se consideren en esta categoría. Esto sin duda provoca que los operadores demoren sus decisiones sobre la adopción de las tecnologías 3G.

Por otro lado los SW utilizan un diseño sincrónico (o al menos deben acusar recibo antes de un proceso asincrónico) en el cual se asume que una conexión de red está constantemente disponible como sucede en una típica LAN. Por más que las redes 3G sean lo suficientemente rápidas para soportar procesos sincrónicos, aún así deben establecer conexión lo que puede ser complicado debido a la inactividad o las limitaciones de procesamiento de los dispositivos.
En definitiva las redes 3G tendrán que seguir haciendo frente a estas limitaciones, lo que implica que los desarrolladores de aplicaciones móviles deberán seguir diseñando sus aplicaciones para trabajar de manera asincrónica y manejando posibles interrupciones de servicio.
Aún si la tecnología y las velocidades de las redes conectadas por cable y de las inalámbricas continúan aumentando a lo largo de los años, va a existir siempre una brecha de velocidades entre ambas tecnologías.

3.4 Conclusión

El hecho de consumir SW desde la API de *Engine* e implementar funcionalidades propias de un sistema como éste, implica el intercambio de tipos de datos complejos como veremos en la sección 5.2. Como kXML-RPC solo manipula tipos simples y no es capaz de soportar tipos de datos creados por el usuario, no es posible utilizarlo para la implementación de nuestro caso de uso. Además, los SW que expone la API de Engine utilizan el protocolo SOAP. Por estas razones, kXML-RPC se descarta.
En consecuencia se realizó una evaluación exhaustiva de los clientes SOAP: kSOAP y WingSOAP.
Se usaron procedimientos de testeo para probar la comunicación cliente / servidor usando SW. Estos procedimientos se implementaron como funciones “eco” cuyo parámetro de entrada era el retornado en la salida. Los tipos de datos testeados fueron boolean, int, string, array de bytes, date, float, array de string y tipos complejos con atributos de tipo int, string, float, date, array de bytes y array de string.
Se invirtieron muchas horas de desarrollo en estas pruebas debido a la complejidad de estos clientes y a la escasa documentación de los mismos, pero los resultados fueron satisfactorios (Ver Apéndice 7.5).

Finalmente, se optó por utilizar kSOAP, debido a que es una implementación de código libre a diferencia de WingSOAP, y posee buenas referencias en la comunidad Java.
A pesar de que una de sus carencias más notorias es la falta de documentación (prácticamente nula), existen foros y numerosos artículos técnicos sobre el mismo.

Sin embargo, no existen actualmente herramientas que cubran la totalidad de los aspectos de los SW inalámbricos. El descubrimiento de los SW (UDDI) desde cliente móvil está ausente al menos por ahora.

El tema de consumir SW desde dispositivos inalámbricos, al tratarse de una tecnología innovadora, ha tenido una gran evolución durante el transcurso del proyecto. Por esta razón es de destacar que ante la demora en lograr una especificación estándar de SW para J2ME de parte del JCP [34], empresas como IBM y Sun han incorporado en los últimos meses a sus ambientes de desarrollo la implementación de kSOAP. Esto demuestra la aprobación de este cliente SOAP y la necesidad de las empresas de enfrentar la nueva frontera tecnológica de los SW móviles.
De todas formas el soporte inalámbrico no es el adecuado actualmente y habrá que esperar una mayor evolución de las tecnologías inalámbricas en este sentido.

4 Relevamiento sobre interfases gráficas para dispositivos móviles

Como uno de los objetivos del proyecto se abordó el estudio de interfases gráficas de usuario (GUI). Esto se hizo de forma independiente al desarrollo mismo de la aplicación, con la idea de implementar interfases gráficas que aprovechen de la mejor forma posible las capacidades de cada tipo de dispositivo destino, ya sea un celular o un PDA.

Se debe tener en cuenta que las restricciones y particularidades al desarrollar una GUI para un celular, no son las mismas que para un PDA de alto procesamiento, que como se mencionó en el capítulo 2.2 cuenta con un procesador de al menos 100 Mhz y 16 MB de memoria.

Por estas razones, en este capítulo se presentan tecnologías existentes en la actualidad que se evaluaron y las que se utilizaron finalmente en la etapa de desarrollo.

4.1 Interfase Gráfica de Usuario del Mobile Information Device Profile

Según puede derivarse de la sección 2.4.1, CLDC no define ninguna funcionalidad para el desarrollo de GUIs, pero esto se soluciona utilizando perfiles como el MIDP sobre CLDC, cuyas clases se incluyen en el paquete javax.microedition de J2ME.

A pesar de que el MIDP se basa en la plataforma Java, su GUI es diferente del Abstract Windows Toolkit (AWT) [23]. El paquete java.awt contiene todas las clases para crear GUIs y para trabajar con colores de gráficos e imágenes, pero está diseñado para computadoras de escritorio. Incluye funcionalidades que son imprácticas o inviables en los dispositivos móviles, debido más que nada, a las restricciones que estos presentan. Por ejemplo, si se consideran modelos de interacción con el usuario utilizados por AWT, como trabajar con ventanas y mouse, resulta que es impráctico usarlos por el pequeño tamaño de la pantalla de los dispositivos o porque en general no incluyen mouse para entrada de datos. Está también el manejo de memoria que utiliza AWT, que no se adapta a las limitaciones de memoria y CPU de los dispositivos móviles.

APIs de la GUI del MIDP
La GUI del MIDP consiste de APIs de alto y bajo nivel, cada una con su propio conjunto de eventos.

La API de alto nivel esta diseñada para aplicaciones donde la portabilidad entre los dispositivos móviles es importante. Para alcanzar la portabilidad, la API permite muy poco control sobre como se ve y como se comporta. Por ejemplo, no se puede definir la apariencia visual (forma, color o fuente) de los componentes de alto nivel, como pantallas predefinidas y formularios. La implementación se encarga de la mayoría de las interacciones con los componentes de alto nivel y de adaptar la GUI al hardware del dispositivo, mientras que para la aplicación esto es transparente. Las clases que implementan ésta API heredan de javax.microedition.lcdui.Screen.

La API de bajo nivel está diseñada para aplicaciones que necesitan control preciso de los elementos gráficos y acceso a eventos de entrada de bajo nivel. Ésta API da a la aplicación control total sobre lo que aparece sobre la pantalla. Sin embargo, no se garantiza que las aplicaciones (midlets) que acceden a la API de bajo nivel sean portables, porque esta API provee mecanismos para acceder a detalles que son específicos de un dispositivo particular. Las clases que implementan ésta API son javax.microedition.lcdui.Graphics y javax.microedition.lcdui.Canvas.

Modelo GUI del MIDP
El paquete javax.microedition.lcdui contiene todas las clases GUI del MIDP que incluye 3 interfases y 21 clases. El listado de las mismas se puede ver en el Apéndice 7.3.2.
La Figura 4.1 muestra las principales clases del modelo y las relaciones entre ellas [23].

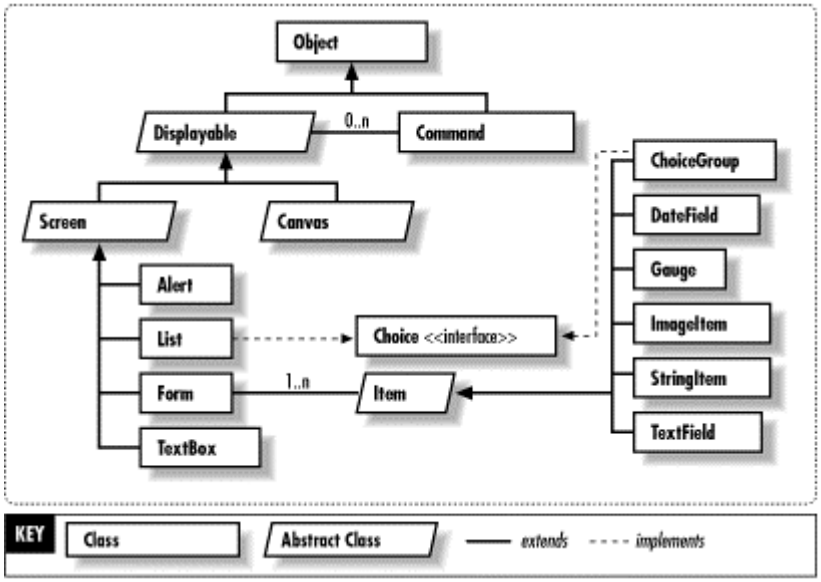

Figura 4.1 – Clases del MIDP-GUI

Entre estas clases, las que brindan funcionalidad básica son:

- La clase javax.microedition.lcdui.Display, que representa el display de un dispositivo que utiliza MIDP. La clase Display es instanciada para cada MIDlet activo y provee métodos para devolver información sobre las capacidades del display del dispositivo.
- La clase javax.microedition.lcdui.Screen, que representa las pantallas que son mostradas por el objeto Display llamando al método setCurrent( ). Como muestra la Figura 4.2, puede haber varias pantallas (objeto Screen) en una aplicación, pero solamente una pantalla puede ser visible (o current) por vez en un display y el usuario puede moverse solamente sobre esa pantalla.

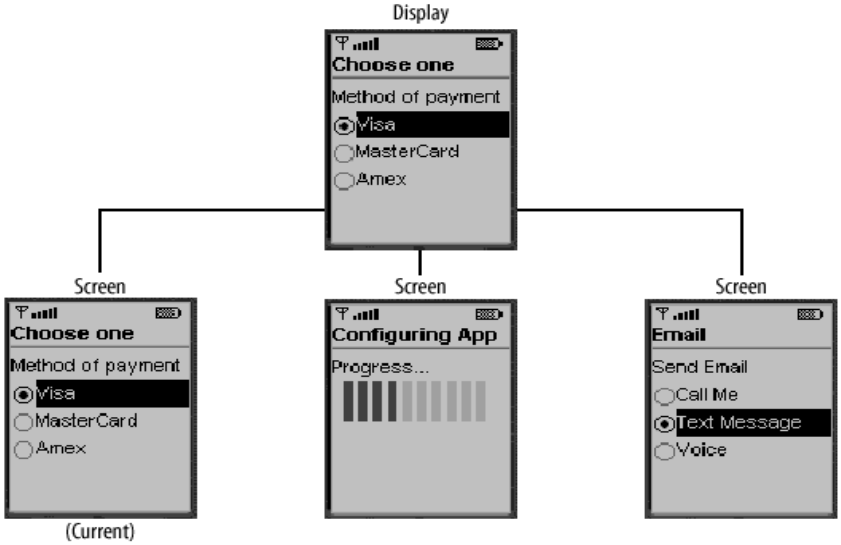

Figura 4.2 – Relación entre Display y Screens

Hay 3 tipos de pantallas en la GUI del MIDP:

1. Pantalla que encapsula totalmente un componente de interfase de usuario, tal como un componente List o TextBox. La estructura de estas pantallas está predefinida y la aplicación no puede agregarle otros componentes.
2. Pantallas genéricas que usan un componente Form. La aplicación puede agregar un conjunto simple de componentes UI relacionados con el Form, como los que se muestran en la Figura 4.1. En este caso el Form actúa como un contenedor.
3. Pantalla usada dentro del contexto de la API de bajo nivel, tal como una subclase de la clase Canvas o Graphics. La clase Canvas permite al Midlet dibujar formas simples sobre una pantalla [42] que luego puede ser utilizada por el método SetCurrent( ).

El MIDP 1.0 era la versión estable con la que se contaba al principio de la investigación. Esta versión tiene carencias de elementos de GUI en comparación con otras bibliotecas gráficas como AWT, lo que disminuiría la calidad de la interfase gráfica final. Además, MIDP para Palm OS solo puede desplegar imágenes de formato PNG de 16 colores.
Luego se liberó la versión 2.0, que presenta mejoras con respecto a la anterior (por más detalles ver Apéndice 7.3.3), sin embargo, no existe soporte para PDAs con sistema operativo Palm OS, hasta el momento.

4.2 Interfase Gráfica de Usuario del Personal Profile

En la sección 2.4.1 se vio que el CDC de J2ME está orientado a dispositivos con mayores recursos y mejor poder de procesamiento, en comparación con los dispositivos a los que apunta el CLDC.
El Personal Profile (PP) extiende al Foundation Profile (FP) que va sobre el CDC, para proveer interfases de usuario gráficas, lo que el FP no brinda. El PP es un conjunto de APIs gráficas basadas en la bibioteca *AWT* de Java 2.

4.3 Proyecto kAWT

El objetivo del proyecto kAWT [19], es proveer un subconjunto funcional de la biblioteca AWT de Java (mencionada en la sección anterior), que pueda ser utilizado para la *KVM* incluida en J2ME/CLDC. J2ME tiene componentes de GUI muy limitados en lo que respecta a los perfiles del CLDC, ya que se dejó de lado este aspecto en pos de lograr aplicaciones portables y multiplataformas, por lo que la biblioteca kAWT puede ser utilizada para suplir esta carencia. Por más información sobre esta biblioteca ver Apéndice 7.4.

Al contrario de la AWT original, el diseño de kAWT no depende de componentes nativos, lo que hace simple portar kAWT a otras plataformas. Por otro lado, no es posible que todas las aplicaciones que utilicen AWT corran sin conversión en dispositivos móviles, pero lo inverso funciona. Es decir, aplicaciones diseñadas para kAWT deben correr sobre plataformas Java estándares sin necesidad de transformaciones adicionales.

Para utilizar kAWT debe tenerse en cuenta que requiere mayor capacidad de almacenamiento y la carga de aplicaciones con kAWT demora más, debido a que es más grande que la KVM básica que tiene sólo 70KB.

A pesar de lo anterior, la biblioteca kAWT es una opción válida para potenciar la capacidad de desarrollo de interfases gráficas en J2ME. Además, se le puede agregar la biblioteca xkVM (inicialmente Color-KVM) que le da a la máquina virtual la capacidad de soportar colores y aumenta la velocidad de carga de las aplicaciones con kAWT.
Es recomendable para dispositivos móviles y es un proyecto reconocido y aceptado en la comunidad Java.

4.4 Macromedia Flash (.SWF)

La empresa Macromedia [17] posee una versión liviana de *Flash* para dispositivos móviles, tanto para desarrollo como para desplegar contenido Flash.
La principal característica de Flash es que está basado en un formato gráfico de vectores, es decir, que utiliza un lenguaje de programación para describir un gráfico en términos geométricos. Esto lo hace comparable en cierta forma, con otros lenguajes usados en dispositivos como *HTML*, *WML*, *cHTML* y J2ME. Aunque existen diferencias, por ejemplo, HTML es un lenguaje estático, mientras que Flash es dinámico. Por otro lado, J2ME soporta una GUI muy limitada en comparación con la que se puede desarrollar con las herramientas de Flash.

En particular Flash Player 6, la versión actual de Flash para dispositivos móviles, incluye un kit de desarrollo de contenido Flash, un kit para desarrollo de interfases de usuario efectivas para los recursos limitados del dispositivo, y componentes predefinidos que simplifican el desarrollo de las GUI. Entre estos componentes existen CheckBox, ComboBox, ListBox, RadioButton y ScrollBar que han sido optimizados y pueden usarse tanto para agregar una interacción simple con el usuario a una aplicación Flash, como para crear una interfase de usuario completa para aplicaciones o formularios Web. Entre las ventajas que presenta esta versión, incluye un parser XML, utiliza interpretación de gráficos basada en vectores, incluye *Drag & Drop* y se descarga rápidamente debido a que sus archivos son pequeños (200KB).
Sin embargo, actualmente para PDAs, solo está disponible para dispositivos Pocket PC 2002 de Microsoft.

4.5 Conclusión

Como se pudo ver después de este análisis de las distintas alternativas de interfases gráficas, la mayoría están estrictamente ligadas a la plataforma de desarrollo y también al poder de procesamiento del dispositivo de destino. No podemos desentendernos del hecho de que este proyecto utiliza las funcionalidades de un sistema comercial que como tal, debe tratar de brindar una solución compatible con la mayor variedad posible de dispositivos móviles. Por tal motivo, en nuestra decisión de cual usar, no podemos atarnos a tecnologías o hardware particulares.
De todas maneras es tanta la variedad de dispositivos, que uno debe fijar ciertos parámetros, por lo que se decidió que el desarrollo que se haga debe poder correr en los PDAs con sistema operativo Palm y en celulares.

kAWT resultó ser una opción válida comparado con la GUI del perfil MIDP. Creemos que es una buena alternativa a futuro, ya que es un paquete de código libre y un subconjunto bastante completo de AWT, pero no resultó ser viable para los alcances de este proyecto. No es recomendable para dispositivos como celulares o PDAs con Palm OS por su bajo poder de procesamiento. *kAWT* es bastante “pesado” en lo que refiere a performance.

Macromedia Flash es un lenguaje potente si hablamos de desarrollo de GUI. Sin embargo, las aplicaciones desarrolladas con él no son portables y solamente está disponible para algunos dispositivos como set-top boxes y PDAs que corran el sistema operativo de Microsoft PocketPC 2002. Esto deja de lado a las PDAs con Palm OS.

Por el lado de CLDC, en la actualidad existe solamente un *convertidor* de MIDP 1.0 para el Palm OS, pero no uno para MIDP 2.0. Quizás porque Palm Inc. apuntaba al perfil para PDAs (PDAP) que estaba en desarrollo hasta hace poco, dentro del JCP, pero que al final se discontinuó como se explica en el capítulo 2.4.1.

Por otro lado, a medida que fue avanzando el proyecto, se fue abriendo una brecha cada vez más grande entre los avances de la tecnología de los dispositivos y la evolución de las tecnologías Java. Como consecuencia de esto, los PDAs con Palm OS que entraban dentro de las características del CLDC, hoy en día se adecuarían más al CDC, teniendo en cuenta la memoria, el procesador y recursos disponibles para los últimos modelos de estos dispositivos, como por ejemplo la serie Clié de la empresa Sony. De ahí surgió la necesidad de investigar

sobre las posibilidades que brindaba el CDC. Pero no se logró encontrar un ambiente Java gratis donde poder probar las aplicaciones desarrolladas sobre el CDC.

En definitiva, la implementación de la interfase gráfica del caso de uso se basó en el perfil MIDP Este perfil y en particular su versión 1.0, nos asegura la portabilidad entre los distintos tipos de PDA y también entre los celulares, aunque perdemos calidad en la interfase gráfica.

5 Caso de Estudio

El caso de estudio a desarrollar se plantea como una aplicación cliente dirigida a dispositivos inalámbricos (celulares y PDAs de bajos recursos), que consuma los SW exportados por la API de *Engine* ya descripta. Se considera a este cliente como liviano según se explicó en la sección 3.1. Más en detalle, el caso de estudio consiste en ingresar al sistema, poder realizar búsquedas sobre los bienes digitales del sistema, navegar los resultados, ver las imágenes correspondientes a los bienes digitales recuperados y salir del sistema. [1]

Del análisis de los capítulos anteriores se desprende el uso de las siguientes tecnologías que se aplicaron en el desarrollo de este prototipo.
La elección de J2ME como plataforma de desarrollo se debió a razones de portabilidad y disponibilidad de recursos, como se describe posteriormente en la sección 5.4.1. La interfase gráfica se desarrolló en base a las características que brinda el perfil MIDP de J2ME, según se desprende del capítulo 4. Las limitaciones en los recursos de los dispositivos móviles inalámbricos a los cuales apunta el prototipo, es decir teléfonos celulares y PDAs de bajos recursos, fueron determinantes para esta elección. En cuanto a la comunicación entre los SW y los dispositivos, se utilizó kSOAP como cliente SOAP, de acuerdo a lo que se concluye en el capítulo 3. En este caso se tuvieron en cuenta los tipos de datos complejos que maneja *Engine.*

Para el desarrollo de los casos de uso tomamos ideas del libro “Applying UML and Patterns” de Larman [LAR98].

5.1 Definición de los Casos de Uso

A continuación se explican los casos de uso que se muestran en el diagrama de la figura 5.1. Estos casos son los que se aplicarán en la implementación del caso de estudio.

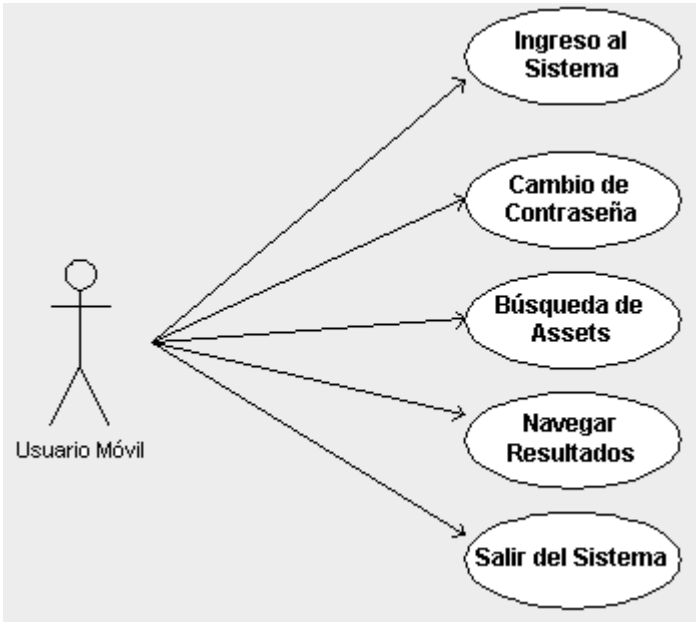

Figura 5.1– Diagrama de Casos de Uso

[1] Se quiere aclarar que para la interfase con el usuario del prototipo se usará el lenguaje Inglés. El motivo de esto es que la empresa donde se desarrolla el proyecto usa este idioma como lenguaje por defecto para la API de *Engine*.

5.1.1 Escenario Típico de Ingreso al Sistema

Acciones del Actor	Respuesta del Sistema
	1. Despliega el formulario de Ingreso solicitando "Login" y "Password"
2. Ingresa "Login" y "Password".	3. Valida los datos ingresados por el usuario. 4. Despliega el Menú Principal con las siguientes funcionalidades: "Search" (Buscar), "Search Results" (Ver resultados de la última búsqueda), "Change Password" (Cambiar la contraseña), "Log Out" (Salir del sistema).

Escenarios Alternativos:
Línea 4: El sistema no valida los datos ingresados por el usuario. Vuelve a desplegar el formulario de entrada al sistema con un mensaje que indica el error.

5.1.2 Escenario Típico de Búsqueda

Acciones del Actor	Respuesta del Sistema
1. Este caso de uso se inicia cuando el usuario presiona la opción "Search" del Menú Principal de la aplicación.	
	2. Despliega dos opciones "Format Results" (elegir el formato de los resultados) y "Set Search Parameters" (elegir los parámetros o criterios de búsqueda).
3. Elige la opción "Format Results"	4. Despliega un formulario con una lista de propiedades: "Size" (tamaño de archivo), "File Name" (nombre de archivo), "File Format" (formato de archivo), "Inserted by" (insertado por), "Status" (estado), "View Image" (ver imagen).
5. Selecciona las propiedades que serán desplegadas para cada asset en el resultado de la búsqueda. Si se selecciona la opción "View Image", se habilita ver las imágenes correspondientes a los assets resultado.	
6. Elige "Set Search Parameters"	7. Despliega el formulario de ingreso de datos del "Search" con las opciones: "Asset Id", "File Name", "Status" y los comandos de "Cancel" y "Search".
8. Selecciona las propiedades para la búsqueda. Las opciones posibles son: "File Name, "Asset Id" y "Status".	
9. Presiona el comando "Search".	10. Despliega un formulario con la lista de resultados por páginas y los comandos "Previous", "Next", "View Image" y "Menu". Cada elemento de la lista contendrá los valores correspondientes a las propiedades seleccionadas en el punto 5.

11. Selecciona un elemento de la lista y presiona el comando “View Image”.	12. Despliega un formulario con todos los valores de las propiedades elegidas en 5, la imagen correspondiente y un comando “Back”.
13. Presiona el comando “Back”.	14. Despliega nuevamente la lista de resultados.

Escenarios Alternativos:
Línea 3: Si no se elige la opción “Format Results” antes de realizar la búsqueda se utilizará el formato por defecto que consiste en solamente mostrar el indentificador del asset resultado.
Línea 8: Si la única opción seleccionada para la búsqueda es “Asset Id”, el resultado constará a lo sumo de un asset.
Línea 9: El usuario presiona el comando “Cancel”, en vez de “Search”. El sistema despliega el Menú Principal.

5.1.3 Escenario Típico de Navegación

Acciones del Actor	**Respuesta del Sistema**
	1. Este caso de uso se inicia cuando el sistema despliega la lista con los resultados de la búsqueda y los comandos “Previous”, “Next”, “View Image” y “Menu”.
2. Presiona el comando “Next”.	3. Despliega la siguiente página de resultados.
4. Presiona el comando “Previous”.	5. Despliega la página previa de resultados.

Escenario Alternativo:
Línea 2: El usuario presiona el comando “Menu”, el sistema despliega el Menú Principal.

5.1.4 Escenario Típico de Salida del Sistema

Acciones del Actor	**Respuesta del Sistema**
	1. Este caso de uso se inicia cuando el sistema despliega el Menú Principal
2. Selecciona la opción “Log Out”.	3. Despliega un formulario con el mensaje de confirmación de salida del sistema y los comandos “OK” y “Cancel”.
4. Presiona el comando “OK”.	5. Termina la aplicación.

Escenario Alternativo:
Línea 4: El usuario selecciona el comando “Cancel”. El sistema despliega el Menú Principal.

5.1.5 Escenario Típico de Cambio de Contraseña

Acciones del Actor	**Respuesta del Sistema**
	1. Este caso de uso se inicia cuando el sistema despliega el Menú Principal
2.Selecciona la opción “Change Password”.	3.Despliega un formulario solicitando “Password”, “New Password” y “Confirmation” y los comandos “OK” y “Cancel”.
4. Presiona el comando “OK”.	5. Valida la información ingresada y realiza el cambio de contraseña. 6. Despliega el Menú Principal.

Escenario Alternativo:
Línea 4: El usuario presiona el comando “Cancel”, el sistema despliega nuevamente el Menú Principal.
Línea 5: La información ingresada no es válida, el sistema vuelve a desplegar el formulario de cambio de contraseña con un mensaje que indica el error.

5.2 Diseño

Como se mencionó en la sección anterior, el caso de estudio consiste en consumir desde dispositivos móviles las funcionalidades que *Engine* exporta como SW.
Para utilizar estas funcionalidades, la aplicación cliente debe ser capaz de enviar y recibir los objetos que *Engine* maneja, entre el servidor y el cliente móvil.

Debido a que J2ME carece de APIs para procesar XML es necesario utilizar herramientas de terceros que nos ayuden a implementar esta comunicación. Un cliente SOAP brinda las herramientas para poder serializar y des-serializar tipos simples y complejos. Para nuestro prototipo se utilizó kSOAP, que permite transmitir automáticamente los tipos simples: int, String, boolean, y arrays. Además provee los mecanismos para serializar y des-serializar tipos complejos propios de un sistema. Los tipos complejos que la API de Engine expone son: DigitalAsset, DigitalAssetSet, SearchCriteria, Criteria, AssetTemplate y AssetProperty, los cuales se explicarán más adelante. Para transmitir estos tipos complejos, se debe implementar la *interfase* KVMSerializable de kSOAP (por más detalles ver apéndice 7.5.2).

Para no atarnos a la implementación de kSOAP, decidimos utilizar la noción de interfases. Por cada objeto complejo de *Engine* en el servidor creamos una interfase en el Cliente. Así implementamos objetos que cumplen tanto con la interfase kSOAP como con las interfases de *Engine*. De esta manera aislamos el resto de la implementación del cliente SOAP utilizado, logrando más flexibilidad si en el futuro cambiamos esta implementación por otra.

Por lo explicado anteriormente, optamos por una arquitectura en capas, donde la capa más cercana al servidor se encarga de los detalles de comunicación. Por encima de ésta, existe otra capa que encapsula la lógica del cliente y brinda una API de abstracción que provee la funcionalidad necesaria para el desarrollo de las diferentes componentes gráficas.
A continuación se explica más en detalle la arquitectura en capas del prototipo.

5.2.1 Arquitectura en Capas del Prototipo

Engine expone gran parte de su funcionalidad como SW para lograr la integración con otros sistemas que así lo requieran. En nuestro caso en particular, el que consume estos servicios es un cliente desde un dispositivo móvil**.** La figura 5.2 presenta un esquema general del prototipo a desarrollar. El cliente consiste de un núcleo compuesto por 3 capas que nombramos Capa de Comunicación, Capa Lógica del Cliente y Capa Gráfica.

Capa de Comunicación
Esta capa se encarga de todo lo referente a la comunicación con los SW expuestos de *Engine*, y la manipulación de los objetos para enviar y recibir los datos. En esta capa se encuentra el soporte para el cliente SOAP, encargado de serializar y des-serializar los datos, armar los mensajes SOAP o generar los tipos simples o complejos al recibirlos. Para esto se utilizan los paquetes de kSOAP.

Capa de Lógica del Cliente
Esta capa tiene el propósito de aislar la implementación del cliente SOAP del resto del núcleo, para que la adopción de una u otra implementación de SOAP no impacte en el resto de la aplicación. La componen los objetos EngineMD, Search, Navigation, Information y DomainInfo, de los que posteriormente hablaremos más en detalle.

Capa Gráfica
En esta capa se centraliza el manejo de la interacción del usuario con el sistema. Se interpretan los comandos y se desarrollan las acciones. Se considera el desarrollo de interfases gráficas para distintos tipos de dispositivos móviles inalámbricos.

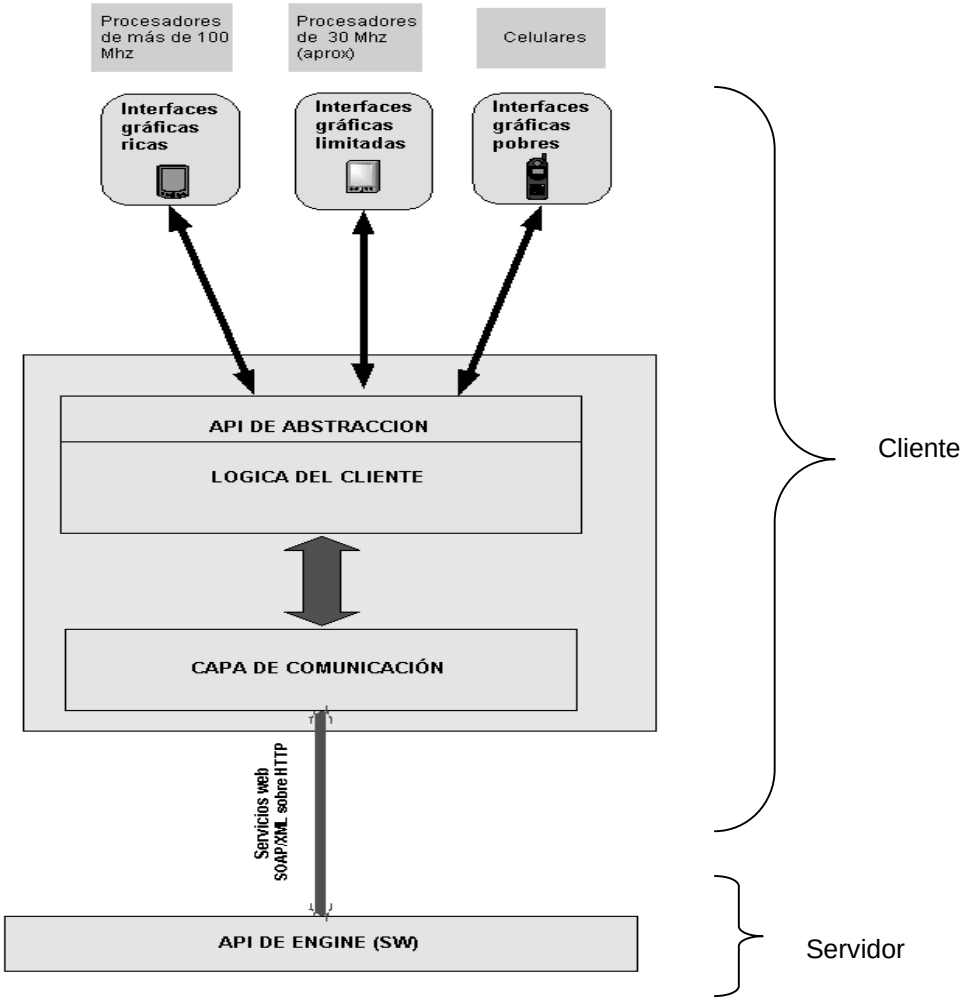

Figura 5.2– Arquitectura en Capas del Prototipo

5.2.2 Diagrama de Componentes

En la figura 5.3 se muestran los componentes del sistema y su interacción. Del lado del servidor tenemos a *Engine* que tiene una API que es utilizado para exportar funcionalidades como SW. En la parte del cliente, tenemos una componente de comunicación que es la que se encarga de consumir estos SW y brindar la interfase Provider que exporta las mismas operaciones que el servidor. En esta capa se encuentra la implementación del cliente kSOAP. La capa *Lógica del Cliente* utiliza las funcionalidades que brinda el Provider para crear toda la funcionalidad de nuestro cliente. En esta capa se encuentran las clases presentadas en la sección anterior, es decir EngineMD, Search, Navigation y DomainInfo. Por último, tenemos la componente gráfica que se basa en la API de *Abstracción* para construir las interfases de usuario. En esta capa están todos los objetos gráficos, como los formularios de todas las funcionalidades, y el MenuController, encargado de procesar la entrada del usuario.
En las secciones siguientes se explicarán cada una de estas componentes.

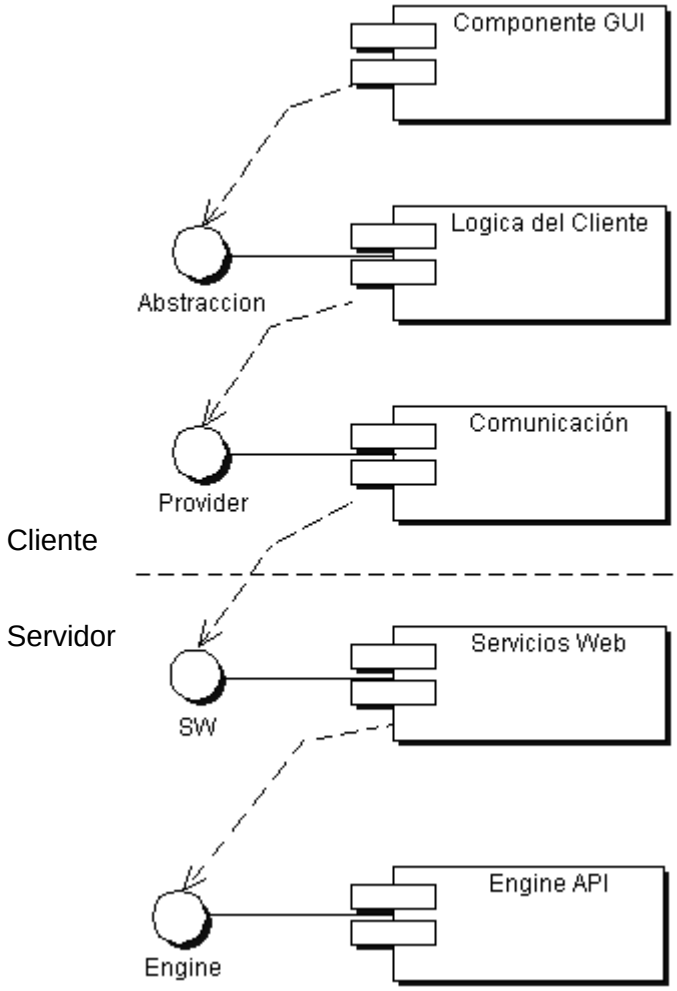

Figura 5.3 – Interacción entre componentes

5.2.3 Funcionalidades y Tipos Complejos de *Engine*

A continuación se explicarán objetos y funcionalidades de la componente Engine API utilizadas en este caso de estudio. Para una descripción más clara se presenta la figura 5.4 con el diagrama de alguna de las clases que componen la interfase de *Engine*.

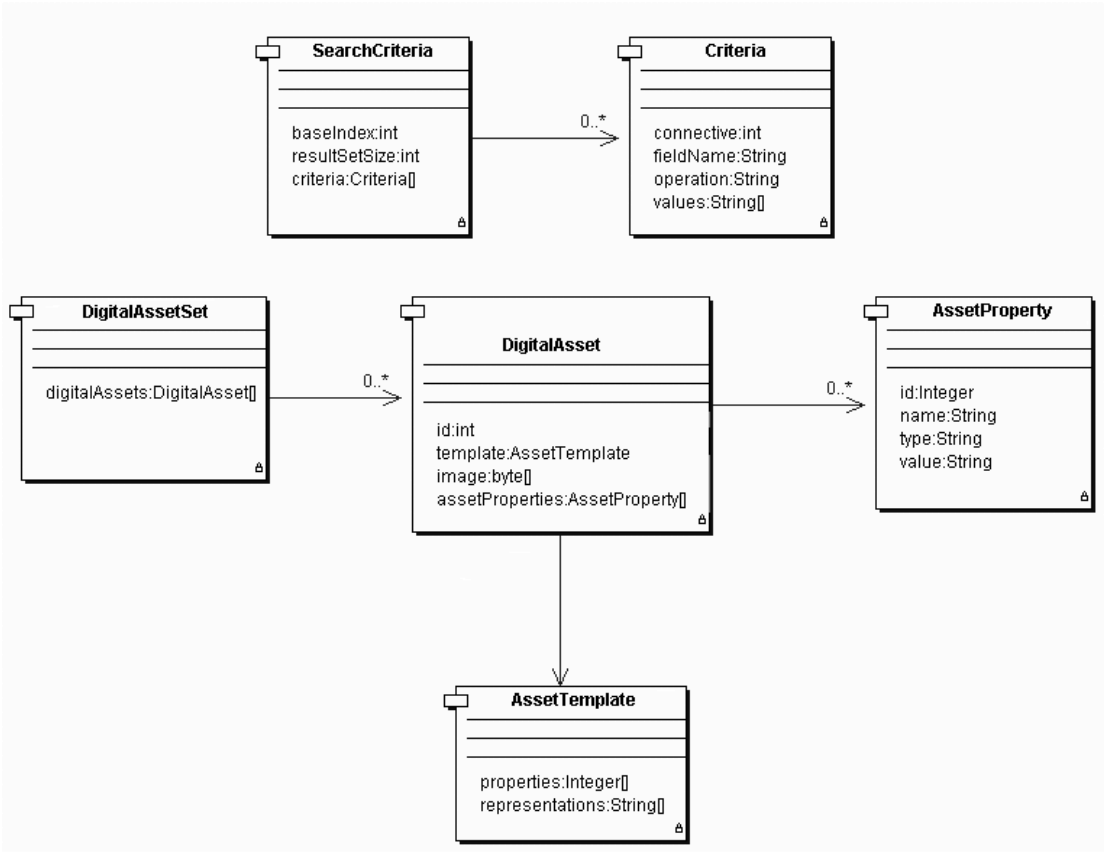

Figura 5.4 Interfase de Engine en el Servidor

Login a *Engine*

El primer paso es loguearse al sistema. Toda tarea a ejecutarse en *Engine* debe realizarse mientras se esté logueado. Al loguearse, se retorna un identificador de sesión (sessionId) que se utiliza como parámetro en toda invocación al sistema.

Búsqueda de Assets

Para realizar una búsqueda se deben especificar tres cosas: dominio (domain), la plantilla (template) y el criterio de búsqueda (criteria). Una vez seteados quedan activos hasta que se seteen nuevamente. No se deben especificar cada vez que se hace una búsqueda.

a) Especificar el dominio

   Toda acción en la API de *Engine* tiene un dominio asociado que especifica sobre que conjunto de assets se realizará la búsqueda. En nuestro caso de estudio solo se utilizará el dominio por defecto que es el total de assets en el sistema.

b) Especificar la plantilla

   Una plantilla (AssetTemplate) se utiliza en todas las acciones del *Engine* que devuelven assets como resultado, y especifica el conjunto de propiedades que un asset tendrá.
   El AssetTemplate posee dos atributos que son, propiedades (properties) y representaciones (representations). Las propiedades son una colección de características del asset, como por ejemplo nombre de archivo, tamaño o formato. Las representaciones se utilizan para setear el tamaño de la imagen que se va asociar al asset. Por ejemplo, imagen pequeña de 100x75 píxeles, mediana de 270x200 píxeles u original. En nuestro caso, y por las limitaciones de los dispositivos móviles mencionadas

en el capítulo anterior, solo trabajaremos con la representación pequeña. El AssetTemplate por defecto contiene el identificador del asset y el nombre del archivo.

c) Especificar el criterio de búsqueda

Para realizar una búsqueda es necesario especificar el criterio por el cual se desea buscar. Si el criterio de búsqueda es vacío, se obtendrá como resultado el total de assets del sistema, sino solo se obtendrán aquellos assets que cumplan con las restricciones (criterios) especificadas.
Un criterio de búsqueda (SearchCriteria**)** es en sí una agregación de criterios. Un criterio (Criteria) esta formado por un operador lógico (AND), un identificador de la propiedad, una condición y un valor a buscar para esa propiedad (ver apéndice por más detalles).
Al momento de realizar la búsqueda, el criterio a utilizar será la concatenación de los elementos del SearchCriteria según el operador lógico correspondiente a cada uno (AND).

Una búsqueda podría retornar una gran cantidad de assets como resultado. Esto, en un ambiente inalámbrico sería insostenible por una cuestión de performance. En ese sentido, el SearchCriteria permite navegar los resultados de a páginas con un número fijo de assets en cada una. De esta manera, cada vez que se setea un SearchCriteria, se deben proveer 2 parámetros que van a indicar el índice del primer asset del conjunto resultado o índice base y la cantidad de assets por página o tamaño de página. Al navegar el conjunto resultado, el índice base se actualiza automáticamente (índice base = índice base + tamaño de página).

Una vez seteado el dominio, el AssetTemplate y el SearchCriteria, se realiza la búsqueda y se obtiene el resultado como una colección de DigitalAsset, llamada DigitalAssetSet.

Un DigitalAsset contiene como atributos un identificador, un AssetTemplate, una colección de propiedades y la imagen (como un array de bytes). Cada propiedad (AssetProperty) contiene un identificador, un tipo, un valor y un nombre. Solo van a existir en la colección de AssetProperty, los valores para aquellas propiedades que se hayan especificado en el AssetTemplate.

5.2.4 API de *Engine* expuesta como SW

Esta sección explica la componente de Servicios Web, dando una breve descripción de las operaciones que *Engine* expone como SW.

Para loguearse al sistema:

String login(String client, String userName, String password)
Recibe como parámetros tres strings que son el nombre de la aplicación cliente, el nombre de usuario y la password.

String login(String client, String userName, String password, String language)
Recibe como parámetros cuatro strings que son el nombre de la aplicación cliente, el nombre de usuario, la password y el lenguaje de la aplicación.

Ambos métodos devuelven un string que es el identificador de una sesión.

Para cambiar password:

void changeLogin(String sessionId, String currentPassword, String newPassword, String passwordConfirmation)
Recibe la sessionId, la password original, la nueva password y la confimación de la nueva password. Todos los parámetros son strings.

Métodos para obtener información:

String getClient(String sessionId)
Recibe como parámetro un string que es el identificador de la sesión y devuelve el nombre del cliente.

String getUserName(String sessionId)
Recibe como parámetro un string que es el identificador de la sesión y devuelve el nombre del usuario logueado.

String getRole(String sessionId)
Recibe como parámetro un string que es el identificador de la sesión y devuelve el rol del usuario logueado.

String getLanguage(String sessionId)
Recibe como parámetro un string que es el identificador de la sesión y devuelve el lenguaje de la aplicación.

Métodos del contexto:

void setAssetTemplate(String sessionId, AssetTemplate template)
Recibe un string que es el identificador de la sesión y un assetTemplate que será usado en las futuras búsquedas.

AssetTemplate getAssetTemplate(String sessionId)
Devuelve el AssetTemplate que está siendo usado por la sesión pasada como parámetro.

void setSearchCriteria(String sessionId, SearchCriteria criteria)
Setea el criterio de búsqueda para la sesión pasada como parámetro.

SearchCriteria getSearchCriteria(String sessionId)
Devuelve el criterio de búsqueda actual para la sesión pasada como parámetro.

Métodos para buscar assets

DigitalAsset getAsset(String sessionId, int assetId)
Devuelve el digitalAsset correspondiente al assetId pasado como parámetro.

int getSearchSize(String sessionId)
Devuelve el total de assets del conjunto resultado de la búsqueda.

int getSearchResultSetSize(String sessionId)
Cada conjunto resultado de la búsqueda se divide en subconjuntos, este método devuelve el total de assets de cada subconjunto.

int getSearchResultSetLowerIndex(String sessionId)
Devuelve el índice inferior del subconjunto de assets.

int getSearchResultSetUpperIndex(String sessionId)
Devuelve el índice superior del subconjunto de assets.

boolean hasNextSearchResultSet(String sessionId)
Devuelve TRUE si existe un próximo subconjunto de assets.

DigitalAssetSet nextSearchResultSet(String sessionId)
Devuelve el conjunto de assets (DigitalAssetSet) del próximo subconjunto de assets.

boolean hasPreviousSearchResultSet(String sessionId)

Devuelve TRUE si existe un subconjunto de assets anterior.

DigitalAssetSet previousSearchResultSet(String sessionId)
Devuelve el conjunto de assets (DigitalAssetSet) del subconjunto de assets anterior.

5.2.5 Componente de Comunicación

En la figura 5.5 se presenta el diagrama de clase del objeto Provider, que se encuentra en la componente de Comunicación. Este objeto se encarga de la comunicación con los SW y utiliza para esto el paquete kSOAP. Es importante notar que todos los objetos obtenidos a través del Provider son Interfases. Esto nos asegura la independencia de la implementación kSOAP con el resto de la implementación del cliente.

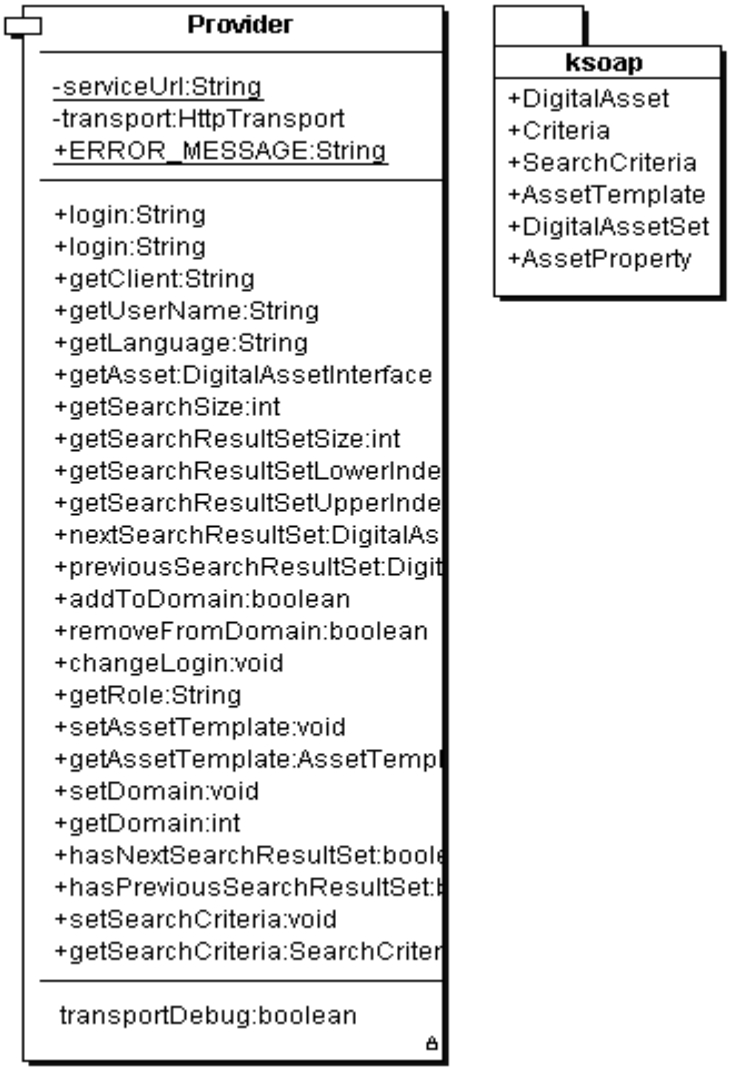

Figura 5.5 – Capa de comunicación

5.2.6 Componente de Lógica del Cliente

Esta componente tiene como objetivo aislar la implementación del cliente de SOAP del resto del núcleo. La figura 5.6 da una visión detallada de la capa de Lógica del Cliente. El objeto EngineMD es el encargado de la entrada al sistema y además de brindar los objetos Search, Information y DomainInfo.

EngineMD: Responsable del ingreso al sistema, validación de contraseña y cambio de contraseña. Además se encarga de mantener la sesión, obtenida cuando un usuario se loguea al sistema y necesaria para toda invocación al servidor.
Information: Mantiene información de la aplicación, como nombre de la aplicación y nombre del usuario logueado.
DomainInfo: brinda información del dominio y las funcionalidades necesarias, para setear un nuevo dominio o acceder al dominio actual.
Search: Responsable de todos los objetos relacionados a la búsqueda, del conjunto resultado, criterio de búsqueda, y de la plantilla utilizada en los assets resultado. Este objeto se puede obtener únicamente a través de EngineMD y solo si ya se tiene una sesión válida.
Navigation: brinda funcionalidades para realizar la navegación de los resultados a partir de una búsqueda dada.

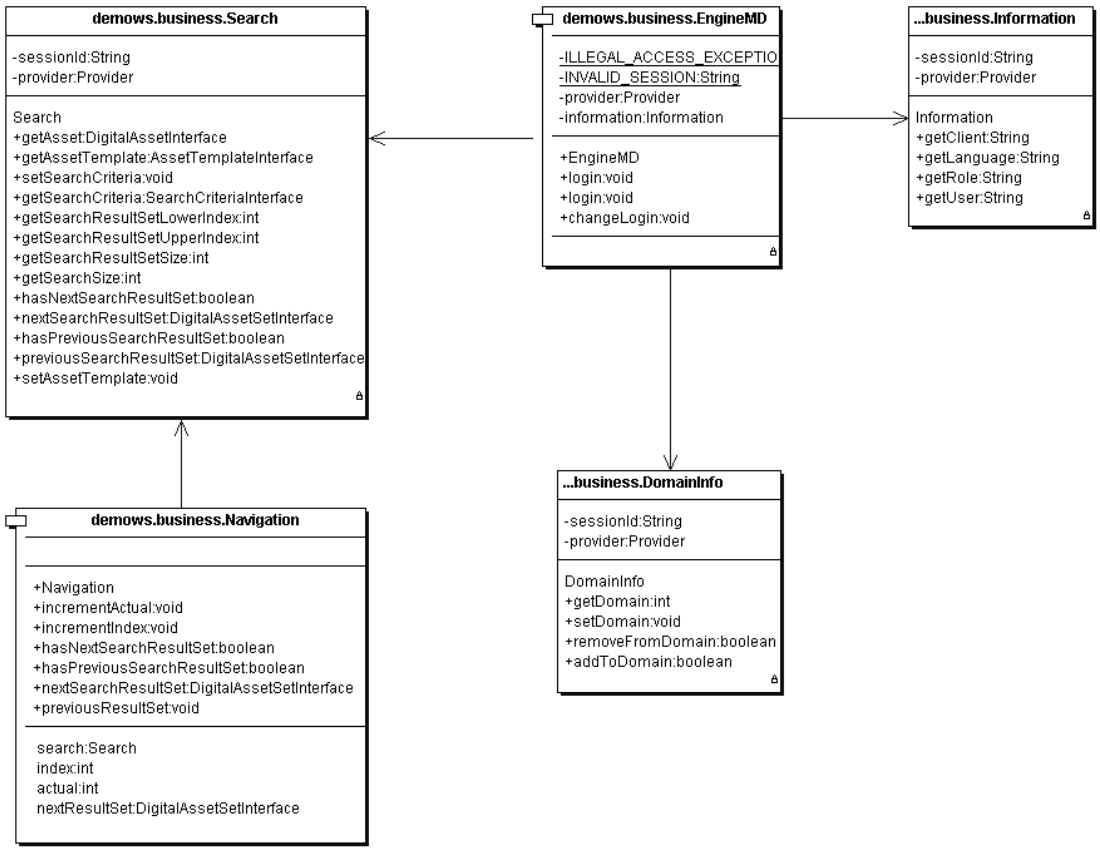

Figura 5.6 – Interfase de Abstracción

5.2.7 Componente GUI

Esta componente utiliza las funcionalidades que brinda la API de Abstracción para construir las interfases gráficas. En esta capa se utilizan los paquetes del perfil MIDP.
Además, se utilizaron los patrones de diseño Cascading Menu y Pagination List [53] para la construcción del menú de la aplicación y de la lista de resultados obtenidos después de una búsqueda, respectivamente.
Dentro de esta componente se encuentra el objeto MenuController que se encarga de procesar los pedidos del usuario y el manejo de eventos asociado a los mismos.

5.3 Diagramas de Secuencia

A continuación se presentan los diagramas de secuencia de los escenarios principales de los casos de uso de Ingreso al Sistema y Búsqueda de Assets.

Como se observa en la figura 5.7, el MenuController que es parte de la componente GUI, procesa los datos ingresados por el usuario, en este caso recibe un login y una password, e invoca el método login de EngineMD. Éste, a su vez invoca al método login del Provider en la Capa de Comunicación. Si los datos son válidos, se obtiene como respuesta un String que es el identificador de sesión.

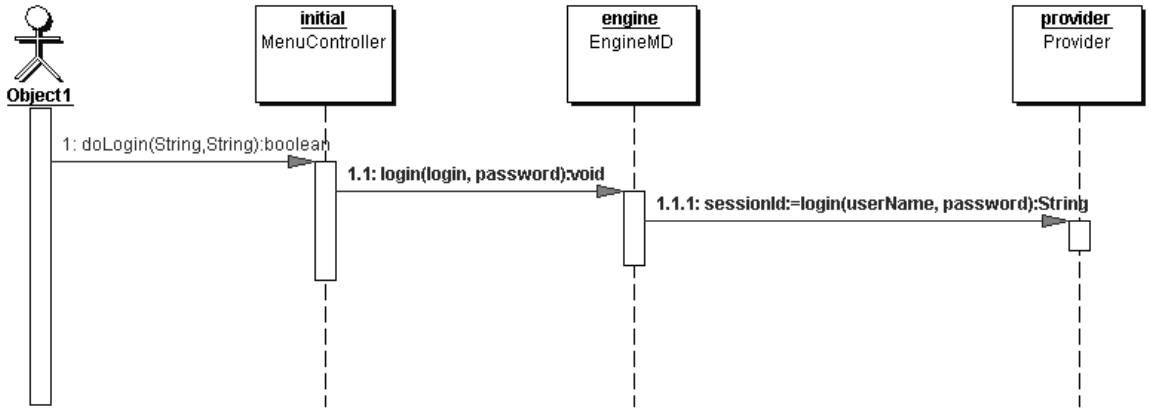

Figura 5.7 - Diagrama de Secuencia del Caso de Uso Ingreso al Sistema

Como se observa en la figura 5.8, el MenuController, se encarga de obtener el objeto Search de EngineMD. Esto es posible solo si el usuario se ha logueado al sistema previamente.
Si el assetId obtenido desde el formulario FrmSearchCriteria no es vacío, se realiza una búsqueda por assetId invocando al método getAsset del objeto Search y recuperando un DigitalAssetBean. Si por el contrario el assetId es vacío, se realiza una búsqueda por propiedades invocando al método nextSearchResultSet, que devuelve un DigitalAssetSetBean.
Los objetos DigitalAssetBean y DigitalAssetSetBean son las implementaciones en la Capa Lógica del Cliente, de las interfases DigitalAsset y DigitalAssetSet.

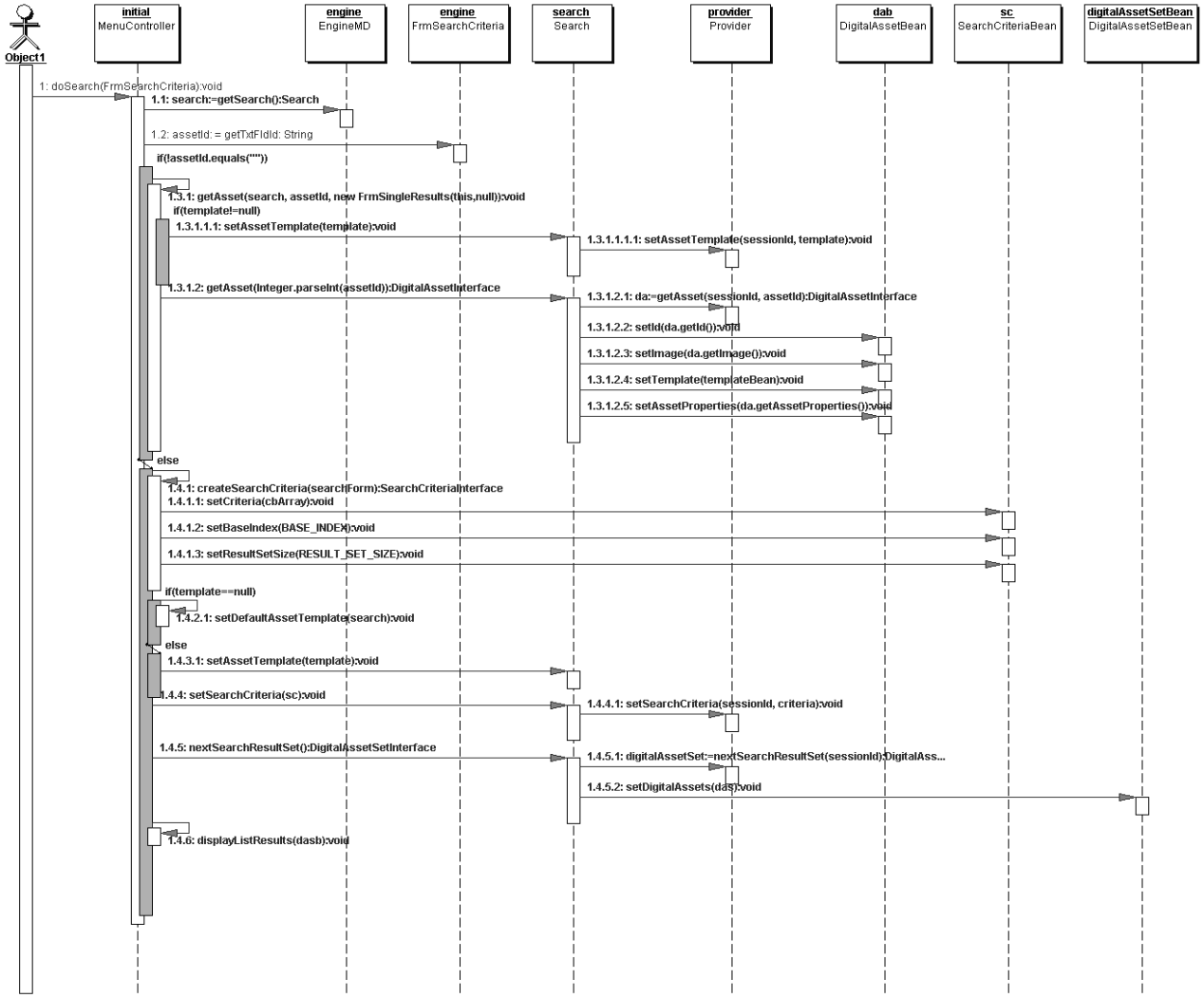

Figura 5.8 - Diagrama de Secuencia de la Búsqueda

5.4 Implementación del Caso de Estudio

En esta sección presentaremos las herramientas utilizadas para la implementación del caso de estudio. También se presentará un breve análisis de la plataforma elegida para el desarrollo y el porqué de la elección.

5.4.1 J2ME como plataforma de desarrollo

En el capítulo 2 se presentaron las dos plataformas de desarrollo para dispositivos móviles más importantes en la actualidad: J2ME de Sun Microsystems y .Net Compact Framework de Microsoft.
Al momento de implementar el prototipo se optó por J2ME como la plataforma de desarrollo. Se explican a continuación los factores predominantes para inclinarnos hacia J2ME.

Portabilidad
.NET CF está orientado a dispositivos con sistema operativo Windows CE y Pocket PC que corren en más de 200 dispositivos diferentes, que es una pequeña cantidad en relación a la gran diversidad de dispositivos móviles existentes en la actualidad. Las aplicaciones desarrolladas con .NET CF solamente son portables entre estos dispositivos.
Debido a que *Engine* es un sistema comercial, la portabilidad es importante de forma de poder abarcar la mayor variedad posible de dispositivos móviles.
Para el caso de los celulares, también es determinante este factor, ya que Symbian OS que posee la amplia mayoría del mercado, brinda el soporte necesario para desarrollar aplicaciones J2ME.

Recursos disponibles
Cuando se comenzó con el desarrollo del caso de uso, aún no estaba disponible el .NET CF, aunque había versiones beta disponibles. Esto nos llevó a inclinarnos por J2ME para no correr riesgos con una versión beta de .NET CF orientado a una tecnología nueva como es el caso del desarrollo para dispositivos móviles.

Soporte de los fabricantes de dispositivos
Como ya se mencionó en el Capítulo 2, las especificaciones de las APIs de J2ME siguen un proceso de estandarización riguroso para asegurar un amplio soporte y consenso de las industrias y empresas involucradas. De esta manera al desarrollar las especificaciones mediante este proceso, se asegura portabilidad y un extenso mercado aún antes de tener una implementación final de las mismas.
En este sentido, las grandes empresas implicadas en el desarrollo orientado a dispositivos móviles, de tecnología celular tales como Nokia y Motorola se han alineado tras J2ME y están dando gran empuje a la plataforma J2ME.

5.4.2 Software Utilizado

Ambiente de Desarrollo Integrado (IDE)

El propósito de un ambiente de desarrollo integrado [33] (IDE) es mejorar la productividad del desarrollador brindando un conjunto de herramientas de desarrollo a través de una interfase gráfica. En particular un IDE para J2ME debería proveer:

- Facilidades para Gerenciamiento de Proyectos, administrando los archivos fuentes y los atributos de las *suites MIDlets*.
- Un editor de código fuente y recursos.
- Compilación, *obfuscación* y *preverificación* del código fuente.
- Facilidades para empaquetar MIDlets en archivos *JAR* y *JAD*.
- Emulación, es decir permitir ejecutar un MIDlet en un *emulador*.
- Debugguear MIDlets.

Basándonos en los criterios anteriores investigamos las siguientes herramientas: Sun J2ME Wireless Toolkit 1.0.4 Beta (*J2ME WTK*), Borland JBuilder MobileSet 3.0 y Sun Forte for Java 4.0 Mobile Edition (FFJ4ME), obteniendo como resultado las siguientes conclusiones:

Sun J2ME Wireless Toolkit 1.0.4 Beta
WTK provee un ambiente básico para desarrollar MIDlets, pero carece de editor y debugger integrado, por lo que no puede competir con los otros IDEs disponibles. Sin embargo, WTK es fácil de usar para principiantes en J2ME debido a que tiene una estructura de directorios simple, permite al desarrollador comenzar inmediatamente sin preocuparse por la infraestructura. Además, WTK es gratuito y pequeño en tamaño. Actualmente esta disponible la versión 2.0 con soporte para MIDP 2.0

Borland JBuilder MobileSet 3.0
MobileSet provee una extensión de JBuilder, que se integra al ambiente existente. Su diseñador MIDP es una herramienta práctica para construir interfases de usuario para MIDPs. Sin embargo, no soporta todo el conjunto de componentes MIDP UI y en muchos casos el código que genera necesita ser corregido manualmente. La Personal Edition es gratuita, pero la licencia es sólo para uso personal. Actualmente esta disponible la versión MobileSet 3.0

Sun Forte for Java 4.0 Mobile Edition
En FFJ4ME todas las características de J2ME se integran con el ambiente Forte para Java existente. Por esto, desarrolladores con un entendimiento básico de aplicaciones MIDP y experiencia previa de Forte for Java, lo encuentran fácil de utilizar. FFJ4ME también es gratuito.

Se hicieron pruebas con los tres IDEs nombrados, pero finalmente al momento de decidir optamos por usar JBuilder con el MobileSet 3.0. La elección se hizo teniendo en cuenta la disponibilidad del mismo, que se adecua a nuestros objetivos y principalmente, porque estamos más familiarizados con el funcionamiento de la versión empresarial de JBuilder.

5.4.3 Desarrollo de una aplicación J2ME

Todas las aplicaciones para el perfil MIDP tienen que extender la clase MIDlet. Esta clase maneja el ciclo de vida de la aplicación y se encuentra en el paquete javax.microedition.midlet.
Como muestra la figura 5.9, un MIDlet puede existir en cuatro estados diferentes: cargado, activo, pausado, destruido. Cuando se instala un MIDlet en un dispositivo y se llama al constructor, el MIDlet pasa al estado cargado (loaded). Esto es antes que el manejador de programa inicie la aplicación invocando al método startApp(). Después de que se invoca este método, el MIDlet está en estado activo, hasta que el manejador de programa invoca a pauseApp() o destroyApp(). El método pauseApp() interrumpe el MIDlet, mientras que el método destroyApp() lo destruye.

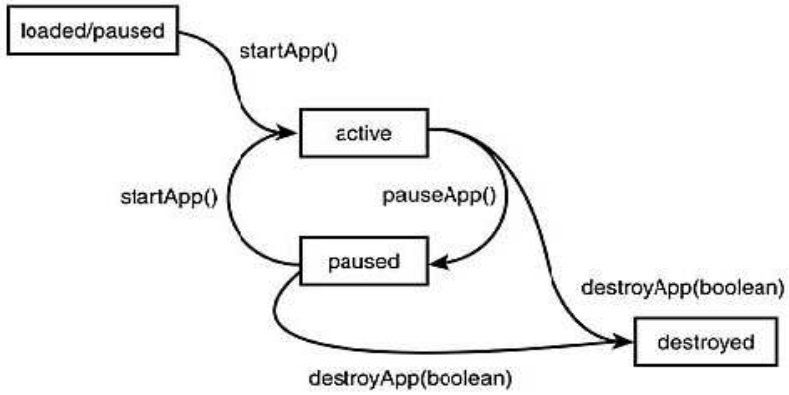

Figura 5.9 Ciclo de Vida de un MIDlet

A continuación se muestra un ejemplo de un MIDlet.

```
public class HelloMidp extends MIDlet {

  Display display;
  Form mainForm;

  public HelloMidp () {
    mainForm = new Form ("HelloMidp");
  }

  public void startApp() {
    display = Displayable.getDisplay (this);
    display.setCurrent (mainForm);
  }

  public void pauseApp() {
  }

  public void destroyApp(boolean unconditional) {
  }
}
```

En el constructor se obtiene el Display y se crea un Form. Cuando se inicia el Midlet por primera vez o cuando regresa de un estado de pausa, se invoca el método starApp(). Se setea el Display al Form, lo que provoca que el Form se muestre. En este ejemplo pauseApp() y destroyApp() no deben ejecutar ninguna acción pero al ser métodos de la interfase MIDlet, se deben proveer las implementaciones vacías.

El manejo de eventos en J2ME está basado en una sucesión de pantallas. Cada pantalla (screen) mantiene cierta cantidad de datos. Los comandos (command) encapsulan el nombre e información relacionado a la semántica de una acción. La principal utilidad es ofrecer diferentes acciones al usuario. El comportamiento final está definido en el CommandListener, asociado con la pantalla. Cada comando contiene una etiqueta, un tipo y una prioridad. La etiqueta se usa para la representación visual del comando; el tipo y la prioridad los usa el sistema para determinar como se mapea el comando a una interfase de usuario en concreto.
En nuestro prototipo el MIDlet es el MenuController, que procesa toda la entrada del usuario.

5.5 Testeo y Evaluación

5.5.1 Ambiente de Testeo

Emulador: DefaultColorPhone
WirelessToolkit versión 1.0.4
Servidor de Aplicación: Resin 2.1.3
Servidor de Base de Datos: SQL 7.0
Cliente / Servidor : PIII 500MHz
512MB RAM

5.5.2 Caso de testeos

A continuación se describen los casos de testeo que se llevaron a cabo.

Caso de Testeo 1
Ingreso al sistema ingresando login y password.

Caso de Testeo 2
Búsqueda por assetId usando el template por defecto (assetId y file name). El asset resultado incluye su imagen asociada, que para este caso es de 6KB.

Caso de Testeo 3
Búsqueda por assetId usando un template completo (size, file name, file format, inserted by y status). El asset resultado incluye su imagen asociada, que para este caso es de 6KB.

Caso de Testeo 4
Búsqueda por file name y status usando el template por defecto (assetId y file name). El resultado contiene una lista de 10 assets sin imágenes asociadas.

Caso de Testeo 5
Búsqueda por file name y status usando un template completo (size, file name, file format, inserted by y status). El resultado contiene una lista de 10 assets sin imágenes asociadas.

Se tomaron 10 muestras por cada caso y se calculó el promedio obteniéndose los resultados que se observan en la Tabla 5.10.

Caso de Testeo	Tiempo de comunicación	Resultado (en segundos)
1	4.868	5.034
2	40.142	42.393
3	45.710	47.598
4	37.414	42.053
5	74.342	78.901

Tabla 5.10 – Resultados obtenidos de los Casos de Testeo

Se puede observar que el mayor tiempo lo consume la comunicación con el servidor. Esta comunicación implica armar el paquete SOAP, transmitirlo y desplegar los resultados.

En los casos 2 y 3 la búsqueda devuelve un asset con su imagen. Aquí la diferencia de tiempos radica en que en el caso 3 se setea el template completo y por lo tanto el resultado incluye todos los valores correspondientes al template seteado.

En los casos 4 y 5 la búsqueda devuelve un conjunto de 10 assets pero sin imagen asociada. En el caso 5 para cada asset resultado se cargan todos los valores seteados en el template lo que explica la diferencia de tiempos con el caso 4.

5.6 Conclusión

El desarrollar aplicaciones para dispositivos móviles implica tener en cuenta varios aspectos importantes, como por ejemplo la gran diversidad de los dispositivos existentes, las limitaciones de los mismos y la evolución de las tecnologías inalámbricas. Ante el interés de la empresa de brindar para *Engine*, la funcionalidad de acceso inalámbrico desde dispositivos móviles, los factores mencionados son decisivos para lograr una aplicación viable y aplicable a la realidad de la empresa. Prestando atención a estas consideraciones, hicimos el intento de diseñar una aplicación modular, escalable, de no atarnos a tecnologías o softwares propietarios y de lograr una aplicación portable no solo entre distintos sistemas operativos, sino también entre la mayor variedad posible de dispositivos.

En la etapa de implementación, después de analizar la API de *Engine*, se comenzó por realizar un testeo exhaustivo de kSOAP para evaluar si era capaz de serializar y des-serializar todos los tipos de datos complejos que *Engine* expone.
Este testeo consistió en implementar funciones “eco” para los objetos de tipo: DigitalAsset, DigitalAssetSet, Criteria, SearchCriteria, AssetProperty y AssetTemplate. Cada una de estas funciones recibía como parámetro uno de estos objetos y lo retornaba. Con esto se logró probar la comunicación del cliente y el servidor, y se aseguró que la implementación de la interfase KVMSerializable de kSOAP de cada uno de estos objetos era correcta. Esta etapa insumió muchas horas de trabajo, constatándose que kSOAP requiere un tiempo extenso de aprendizaje y de familiarización con sus metodologías. Sus mayores limitaciones están por el lado de la baja performance y la ausencia de documentación que dificulta su utilización. Finalmente, los resultados fueron satisfactorios.

Por no disponer de la tecnología necesaria, el testeo de la aplicación se realizó en base a emuladores, en particular un modelo de celular color y un emulador de las PDAs con Palm OS, y la comunicación fue vía Internet. Pese a no poder recrear el ambiente real, se constató la lentitud del tiempo de respuesta del servidor como se pudo deducir de los testeos realizados en el capítulo 5.5. Las causas son por un lado el cliente kSOAP, cuyos procesos de parseo del XML y de armado de los paquetes SOAP son complejos, y por otro lado las limitaciones de procesamiento de los dispositivos.
El factor de los dispositivos no requiere mayor preocupación. A causa de su evolución constante, ya no será una limitante a corto plazo. Incluso ya existen dispositivos con un considerable poder de procesamiento tales como los Pocket PC. No fue posible testear la aplicación en uno de ellos, por lo que no evaluamos este factor.

Por la falta de recursos necesarios, no se ha podido evaluar el comportamiento de las redes inalámbricas. Además de no poseer los recursos de hardware necesarios, la tecnología inalámbrica actual del mercado uruguayo se basa en *CDPD* (Cellular Digital Packet Data), una tecnología de primera generación (1G).
Teniendo en cuenta lo analizado en el capítulo 3, donde incluso las tecnologías 3G mucho más avanzadas, tienen actualmente problemas de adopción en el mercado de las comunicaciones, y una performance real mucho menor de lo previsto, es evidente que aplicaciones como la desarrollada no son aun viables.

6 Conclusiones

En este capítulo se presentarán los resultados obtenidos a lo largo de este proyecto y las conclusiones acerca de los relevamientos realizados y del prototipo implementado. Para terminar se presentará los trabajos futuros.

6.1 Resultados Obtenidos

En este proyecto se analizó la situación actual de la industria en el campo de los dispositivos móviles y de las tecnologías para el consumo inalámbrico de Servicios Web.
Se estudió la evolución de dichas tecnologías a lo largo del proyecto, las perspectivas a futuro y las posibilidades presentes en un entorno inalámbrico.

Los dispositivos móviles brindan la posibilidad de acceder a información en el lugar y momento preciso de forma inalámbrica. Por esto son potenciales clientes para el consumo de los SW.
Para realizar la comunicación de dispositivos móviles con SW se estudiaron herramientas tales como parsers XML y clientes de los protocolos XML-RPC y SOAP. Se decidió profundizar en el estudio de los clientes porque son más completos funcionalmente que los parsers XML. En cuanto al cliente XML-RPC, se descartó debido a que no puede manipular los tipos complejos de *Engine*. Por el lado de los clientes SOAP se testearon WingSOAP y kSOAP. Los testeos fueron satisfactorios en ambos pero se optó por kSOAP por las razones mencionadas en el capítulo 3.
Aunque existen estas herramientas que facilitan la interacción con SW desde dispositivos móviles, al día de hoy no se cuenta con soluciones que lo hagan sin necesidad de programación adicional. Estas tecnologías se encuentran en una etapa de desarrollo, ya que su evolución está acotada por los recursos limitados de los dispositivos y las características no favorables de las redes inalámbricas citadas en el capítulo 3.

Como parte de nuestros objetivos, también se analizaron distintas tecnologías para el desarrollo de interfases gráficas para dispositivos móviles. En el capítulo 4 vimos que la calidad de la interfase gráfica está ligada a las características del dispositivo móvil, como ser el poder de procesamiento, la cantidad de memoria y el sistema operativo. Se analizaron las características gráficas que proveen los perfiles de la plataforma J2ME, la biblioteca kAWT que complementa el MIDP de CLDC y el lenguaje Macromedia Flash para el ambiente Pocket PC. En este aspecto, para el desarrollo del prototipo se utilizó el perfil MIDP de la configuración CLDC, debido a la disponibilidad de los recursos de hardware y software como se explicó en los capítulos 4 y 5, y a que nos brinda la mayor portabilidad entre dispositivos.

Para la implementación del prototipo, se estudiaron tres Ambientes de Desarrollo Integrado. Estos fueron Sun J2ME Wireless Toolkit, Borland JBuilder MobileSet 3.0 y Sun Forte for Java 4.0 Mobile Edition. En líneas generales, todos resultaron completos pero optamos por JBuilder MobileSet por estar más familiarizados con la versión empresarial.

Se implementó un prototipo que ejecuta en PDAs con Palm OS y en celulares logrando resultados satisfactorios. Se logró una interfase para un cliente móvil de *Engine* con las siguientes funcionalidades: Búsqueda de assets en el sistema según ciertos criterios y navegación de los resultados, además de poder visualizar las imágenes de los assets resultados.

No se pudo recrear el ambiente real de una red inalámbrica al no contar con los recursos de software y hardware necesarios. El testeo de la aplicación móvil se realizó con emuladores y la comunicación fue vía Internet. En base a las pruebas presentadas en el capítulo 5.5, se pudo observar la baja performance y la lentitud del tiempo de respuesta del servidor. A su vez, el escenario inalámbrico actual está marcado por la alta latencia de las redes y la lenta velocidad de conexión. No está claro durante cuanto tiempo se va a mantener esta situación.

6.2 Conclusión Final

En un principio, se tenía mayor confianza en la evolución y las características favorables de los procesos de definición de especificaciones del JCP para J2ME. Pero con el correr de los meses, al seguir más de cerca dichos procesos, nos topamos con retrasos y demoras, lo cual influyó en el proyecto e incluso en algunas decisiones de diseño e implementación.
Como ejemplo tenemos el PDAP (JSR 75) que podía llegar a ser el perfil más adecuado para la implementación del prototipo en relación a los objetivos del proyecto. Después de dos años y medio de investigación, el grupo de trabajo llegó a la conclusión de que no era viable un perfil de estas características.
Otro caso es la especificación de Servicios Web para dispositivos móviles (JSR 172) esperada en un principio para diciembre del 2002, que aún hoy sigue sin ser liberada.
Esto deja entrever la burocracia de los procesos del JCP, la dificultad de ponerse de acuerdo entre tantas partes involucradas, y la complejidad de definir estándares para tecnologías que evolucionan y cambian rápidamente en el tiempo. Esta situación complicó las decisiones a tomar para el desarrollo del prototipo.

Otra dificultad encontrada al trabajar con las herramientas analizadas, fue el escaso soporte y documentación que brindan. Esto se reflejó en el tiempo insumido en comprender el funcionamiento de cada una y en los testeos posteriores.

Pensando en una aplicación con fines comerciales, la portabilidad que brinda la plataforma J2ME restringe funcionalidades y características gráficas, que son importantes para este tipo de aplicación en particular. Es necesario tener claro el tipo de dispositivo al cual se apunta.
Sun es consciente de las dificultades para lograr esta portabilidad, por lo que un grupo del JCP (JSR 185) está trabajando con el objetivo de definir la especificación de una plataforma estándar de teléfonos celulares que apunte a desarrolladores de aplicaciones y fabricantes de estos dispositivos.

En definitiva, podemos concluir que la adopción de estas tecnologías no es viable en la actualidad. Sin embargo, por todo lo expuesto en este documento, las perspectivas no dejan de ser alentadoras y habrá que esperar la evolución de dichas tecnologías en el futuro.

6.3 Trabajos Futuros

- Desarrollar una GUI para dispositivos de altos recursos, compatible con dispositivos Pocket PC.
- Agregar más funcionalidades de *Engine* al cliente móvil, como por ejemplo insertar un asset al sistema desde el dispositivo.
- Implementar un cliente más robusto. El cliente móvil actual es un cliente liviano. Sería bueno agregar controles de la conexión de red. Lograr descargar en el prototipo parte de la información para poder seguir con la aplicación cliente, aunque la conexión se interrumpa.
- Migrar la implementación hacia la nueva especificación de consumo de SW desde J2ME (JSR 172).
- Definir un marco de evaluación que permita decidir cuando un sistema de estas características es viable.

7 Apéndice

7.1 Código de ejemplo de WSDL y de paquetes SOAP

Estructura general de WSDL

En el siguiente código se muestran los elementos que aparecen en un documento WSDL. El asterisco cerca de un elemento significa que puede aparecer más de uno de estos elementos en el documento:

```
<definitions>
        <import>*
        <types>
        <schema></schema>*
        </types>
        <message>*
        <part></part>*
        </message>
        <PortType>*
        <operation>*
                <input></input>
                <output></output>
                <fault></fault>*
        </operation>
        </PortType>
        <binding>*
        <operation>*
                <input></input>
                <output></output>
        </operation>
        </binding>
        <service>*
        <port></port>*
        </service>
</definitions>
```

Aviso de Servicio

Otros elementos utilizados:

- <soap:binding>: Significa que el binding se limita al formato del protocolo SOAP. Los atributos de esta etiqueta son transport y style. El primer atributo especifica en que protocolo de transporte de red va a viajar el mensaje SOAP (por ejemplo HTTP o SMTP), y el segundo especifica el estilo por defecto de cada operación para este binding. Los posibles valores para el atributo style son rpc y document.

```
<?xml version="1.0"?>
<definitions name="StockQuote"
                targetNamespace="http://example.com/stockquote.wsdl"
                xmlns:tns="http://example.com/stockquote.wsdl"
                xmlns:xsd1="http://example.com/stockquote.xsd"
                xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
            xmlns="http://www.w3.org/2000/10/XMLSchema">
```

```
     <element name="TradePriceRequest">
      <complexType>
       <all>
        <element name="tickerSymbol" type="string"/>
       </all>
      </complexType>
     </element>
     <element name="TradePrice">
      <complexType>
       <all>
        <element name="price" type="float"/>
       </all>
      </complexType>
     </element>
    </schema>
   </types>

  <message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
  </message>
  <message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>
  </message>

  <portType name="StockQuotePortType">
       <operation name="GetLastTradePrice">
              <input message="tns:GetLastTradePriceInput"/>
              <output message="tns:GetLastTradePriceOutput"/>
       </operation>
  </portType>
  <binding name="StockQuoteSoapBinding"
type="tns:StockQuotePortType">
       <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
       <operation name="GetLastTradePrice">
              <soap:operation
              soapAction="http://example.com/GetLastTradePrice"/>
              <input>
              <soap:body use="literal"
              namespace="http://example.com/stockquote.xsd"
       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
       </input>
      <output>
       <soap:body use="literal"
         namespace="http://example.com/stockquote.xsd"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
     </operation>
  </binding>

  <service name="StockQuoteService">
   <documentation>My first service</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
     <soap:address location="http://example.com/stockquote"/>
    </port>
  </service>
</definitions>
```

A continuación se muestra un paquete de solicitud y respuesta del servicio StockQuote [8a].

Un paquete SOAP – solicitud al servicio StockQuote

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml;
charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
    <m:GetLastTradePrice
     xmlns:m="Some-URI">
      <tickerSymbol>MOT</tickerSymbol>
    </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Un paquete SOAP - respuesta del servicio StockQuote

```
HTTP/1.1 200 OK Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
 <SOAP-ENV:Body>
  <m:GetLastTradePriceResponse
   xmlns:m="Some-URI">
   <price>14.5</price>
  </m:GetLastTradePriceResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

7.2 Detalle de los módulos que componen el DAM *Engine* 3.0

A continuación se presentan las funcionalidades de los módulos que componen *Engine.*

7.2.1 Protección de los Bienes Digitales

Inserción de Bienes Digitales en el Sistema
Los usuarios pueden insertar archivos en el sistema. Aunque se puede insertar cualquier archivo, no todos los tipos de archivo son reconocidos por el sistema para procesos futuros. Aquellos archivos no reconocidos son almacenados y los archivos originales están disponibles para ser recuperados cuando el usuario lo requiera.

Dependiendo de sus permisos en el sistema, un usuario puede insertar directamente bienes digitales en el sistema o pedir a alguien más que lo haga. En este último caso los archivos se dejaran en un lugar especial y se enviará un mensaje al usuario con los permisos indicados para insertar.

Se pueden insertar tanto archivos individuales como estructuras de carpetas conteniendo archivos y subcarpetas.

7.2.2 Recuperación

Búsqueda
Es posible buscar archivos mediante diferentes criterios (categorías o atributos). Si los archivos están correctamente categorizados la búsqueda será posible y bastante exacta. Los resultados de la búsqueda se presentan al usuario con la posibilidad de navegarlos y verlos de diferentes maneras (lista de datos o imágenes, por ejemplo).

Descarga del Bien Digital
El usuario puede descargar diferentes formatos de archivos.
Cuando se elige esta opción todos los archivos seleccionados se empaquetaran en un archivo zip, el cual será descargado inmediatamente.

Inspección del Archivo
Es posible ver una vista previa del archivo. Aquí se muestra toda la información asociada al mismo. También es posible inspeccionar esta información. La clase de inspección que pueda ser realizada depende del tipo de archivo.
Cada tipo de archivo es accedido en una nueva ventana con la herramienta asociada, si está configurada. Por ejemplo, Acrobat es usado para archivos PDF y MS Office, el navegador web es utilizado para abrir imágenes y archivos EPS, mientras que sonido y video utilizan un plugin de Microsoft Media Player.

7.2.3 Organización

Asignación de Descriptores
A medida que los archivos son insertados en el sistema, automáticamente se indexan con información básica tal como autor de la inserción, fecha de la misma y formato del archivo. Esta información no es editable por el usuario, que de todas maneras puede indexar archivos de acuerdo a sus necesidades específicas, llenando campos ya existentes en el sistema.

Carpetas
Los bienes digitales pueden ser almacenados y organizados en carpetas permitiendo una clasificación natural para los usuarios.

Eliminar
Al seleccionar los archivos que se desean eliminar del sistema, el usuario puede elegir las opciones de borrado desde la barra de opciones. El sistema controla si los archivos se encuentran en alguna selección de otro usuario o en una publicación. Si esto sucede, va a depender de los permisos del usuario, si puede o no eliminar los archivos en cuestión independientemente de sus estados.
Los archivos son eliminados lógicamente del sistema, en particular son movidos a una carpeta especial.

7.2.4 Distribución

Ordenar
La idea detrás de este módulo, es la de tener la posibilidad de pedir archivos para uso personal o para hacerlos disponibles para otras personas que no sean usuarios de *Engine*.
Pueden ser ordenados distintos tipos de formatos de archivos (originales o personalizados).
Los archivos ordenados son empaquetados en un archivo zip. El o los destinatarios recibirán un email con un link, usuario y contraseña. A partir de esto, se podrán bajar a disco los archivos ordenados, accediendo al sistema por el link incluido en el e-mail, utilizando el usuario y contraseña recibidos.

Publicar
Una publicación consiste en uno o varios archivos que son agrupados para ser accesibles por usuarios especiales, llamados “invitados”.
De esta manera podemos restringir el acceso a diferentes archivos para algunos usuarios que igual tendrán la funcionalidad básica tal como búsqueda, selección y órdenes de bienes digitales.

Para acceder a una publicación, se requiere un usuario y contraseña que estarán disponibles por un período de tiempo predefinido.
Mientras que una publicación esté disponible, el usuario invitado estará habilitado para buscar, ordenar y bajar a disco los archivos incluidos en la misma. Ésta es la principal diferencia en relación a las órdenes, donde solamente pueden bajar a disco los archivos incluidos.
Las publicaciones pueden ser administradas (modificadas o eliminadas) por sus creadores.

7.2.5 Control de Acceso

Administración de Usuarios
Solo los administradores del sistema pueden crear, modificar y eliminar usuarios.

7.2.6 Conversión Automática

Transformación de Archivos
Este módulo brinda la posibilidad de ordenar "transformaciones" especiales de un archivo original. Está dividido en dos versiones: simple y avanzada. Cada archivo ordenado puede tener su propia transformación.
Los archivos ordenados son comprimidos para mayor rapidez. El sistema empaqueta los archivos y envía un email al destinatario, incluyendo un link, nombre de usuario y contraseña. El destinatario debe loguearse al sistema para descargar los archivos.
Cuando el destinatario se loguea al sistema, podrá observar la lista de órdenes asignadas a él y tendrá la posibilidad de descargar los archivos comprimidos.

Engine Print
Engine Print ASP-pages brinda la posibilidad a usuarios finales de crear publicidades, avisos o afiches (en versión PDF) basados en plantillas predefinidas y en un conjunto preseleccionado de imágenes.
La administración de imágenes de Engine Print permite utilizar imágenes almacenadas en *Engine* para crear dichos afiches usando Engine Print ASP-pages.
Utilizando las funciones de administración de Engine Print en *Engine*, el administrador especifica que imágenes pueden ser introducidas en las distintas partes de la plantilla. También se especifican los campos que están relacionados con las imágenes seleccionadas. *Engine* realiza las transformaciones de color y tamaño necesarias, y el resultado se ubica en un lugar desde donde Engine Print y los usuarios finales lo pueden obtener.

7.2.7 Workflow

Revisiones
En algunas organizaciones, los documentos, textos, imágenes u otros archivos, tienen que ser revisados por varias personas en la compañía y fuera de ella. Para acelerar este workflow y hacerlo más controlable, se ha desarrollado este módulo "Revisiones", para ser integrado en *Engine*.
Una revisión es creada con un conjunto seleccionado de archivos. El usuario "editor" selecciona los usuarios que están invitados a hacer la revisión, llamados "revisores", y se envía un email a cada uno de ellos notificándoles de su nueva tarea.
Cuando el revisor se loguea al sistema, va a poder visualizar una lista de todas las revisiones asignadas al mismo. Al seleccionar una de ellas, podrá comenzar a trabajar con la misma. Además podrá crear sus propias revisiones y visualizar otras ya hechas al archivo.
Hay dos maneras de hacer revisiones; una es utilizando texto y la otra es utilizando el programa Acrobat. La última es válida solamente para archivos PDF.

Comentar
Esta característica permite a los usuarios hacer comentarios a un bien digital en particular. Los comentarios se guardan junto con la información de quién y cuando los realizó, permitiendo a otros usuarios pedir o verificar información relevante sobre el bien digital

7.2.8 Características auxiliares

Login
Todos los usuarios ingresan al sistema digitando usuario y contraseña. *Engine* posee las características de seguridad usuales para estos sistemas. La contraseña puede ser cambiada en cualquier momento.

Selecciones
Existen 4 selecciones para cada usuario. El propósito de las mismas es permitir al usuario agrupar bienes digitales y tenerlos a mano para un procesamiento futuro. Los bienes digitales de una selección pueden ser ordenados, descargados, publicados o eliminados. Estas selecciones persisten a lo largo de las diferentes sesiones de trabajo.

7.3 El Modelo MIDP

7.3.1 MIDP 1.0

El MIDP se enfoca a dispositivos móviles que se encuadren dentro de las siguientes características:

- Tamaño de pantalla de a lo sumo 96x54 pixels.
- Profundidad de display de 1 bit.
- Dispositivos de entrada con tecnología touch-screen.
- 128 KB de memoria no volátil para componentes MIDP.
- 8 KB de memoria no volátil para datos persistentes.
- 32 KB de memoria volátil para el heap de Java.
- Conectividad inalámbrica de dos vías.

En la tabla 7.1 se muestran los paquetes del MIDP junto con una breve descripción.

Paquetes del MIDP	Descripción del paquete
javax.microedition.lcdui	Clases e interfases de GUI
javax.microedition.rms	Record management system (RMS). Soporte para almacenamiento persistente en dispositivos
javax.microedition.midlet	Soporte para la definición de aplicaciones MIDP
javax.microedition.io	Clases e interfases de conexión del MIDP
java.io	Clases e interfases de entrada / salida
java.lang	Clases e interfases de la Maquina Virtual
java.util	Clases e interfases utilitarias

Tabla 7.1 Paquetes del MIDP

7.3.2 Interfases Gráficas en MIDP 1.0

El paquete *javax.microedition.lcdui* contiene todas las clases del MIDP-GUI que incluye 3 interfases y 21 clases que se muestran en la Tabla 7.2 y 7.3 respectivamente:

Interfase	**Descripción**
Choice	Define una API para el componente *interfase de usuario* que implementa una selección desde un número predefinido de opciones.
CommandListener	Usado por aplicaciones que necesitan recibir eventos de alto nivel desde implementaciones.
ItemStateListener	Usado por aplicaciones que necesitan recibir eventos que indican cambios en el estado interno de los ítems interactivos.

Tabla 7.2 Interfases del MIDP 1.0

Clase	**Descripción**
Alert	Una pantalla que muestra datos al usuario y espera por cierto período de tiempo antes de pasar a la pantalla siguiente.
AlertType	Indica la naturaleza del alert.
Canvas	Clase base para escribir aplicaciones que necesitan manejar eventos de bajo nivel y llamadas gráficas para dibujar en el display.
ChoiceGroup	Grupo de elementos seleccionables que se ubican en un form.
Command	Constructor que encapsula la información semántica de una acción.
DateField	Componente editable para presentar información de fecha y hora que puede ubicarse en un form.
Display	Utilidad que representa al administrador del display y dispositivos de entrada del sistema.
Displayable	Objeto que tiene la capacidad de ser ubicado en el display.
Font	Utilidad que representa fuentes y sus métricas.
Form	Pantalla que contiene una mezcla arbitraria de ítems (Image, TextField o ChoiceGroup).
Gauge	Utilidad que implementa un display de gráfico de barras de un valor que se usa en el form.
Graphics	Utilidad que provee una visión geométrica en 2 dimensiones.
Image	Utilidad que mantiene datos de imagen gráfica.
ImageItem	Utilidad que provee control del layout cuando objetos *Image* son agregados al form o alert.
Item	Una superclase para todos los componentes que pueden ser agregados a un form o alert.
List	Pantalla con lista de selección.
Screen	Superclase de todas las clases de interfase de usuario de alto nivel.
StringItem	Item que puede contener un string.
TextBox	Pantalla que permite al usuario ingresar y editar texto.
TextField	Componente de texto editable que puede ubicarse en un form.
Ticker	Pieza de texto de tipo ticker que ejecuta continuamente a través del display. Puede ser agregado a todo tipo de pantalla excepto Canvas.

Tabla 7.3 Clases del MIDP 1.0

7.3.3 Imágenes en MIDP

Al trabajar con imágenes en este ambiente, se debe tener en cuenta que las mismas tienen que salvarse en formato *PNG* (Portable Network Graphics) que es el único formato soportado por todas las implementaciones MIDP.
Existen varias características gráficas para las que MIDP 1.0 no provee soporte directo, sin embargo hay muchos artículos en Internet que dan soluciones a estas carencias. Por ejemplo, se resuelven temas como la animación de imágenes [12] y el dibujo de gráficos no parpadeantes [13]. Por otro lado, está en desarrollo una especificación (JSR-135) [15] de una API para Media móvil [14].
Hay dos formas de tener disponibles las imágenes para un MIDlet que va a utilizarlas. Una forma es colocarlas en el servidor de web y tener el MIDlet bajándolas usando el soporte propio para HTTP del MIDP. La forma más simple, sin embargo, es empaquetarlas con el MIDlet en su archivo JAR. Si se está usando el J2ME Wireless Toolkit, solo se colocan los archivos PNG en el directorio *res* del proyecto.

7.3.4 MIDP 2.0

La especificación de MIDP 2.0 hace numerosos refinamientos a la interfase de usuario para soportar mayor extensibilidad y sobre todo portabilidad de MIDlets entre dispositivos. Algunas de las mejoras son:

- La clase CustomItem puede ser extendida para desarrollar objetos que no están incluidos en la API estándar.
- Se agrega la clase ItemCommandListener para mejorar las interacciones entre elementos y la clase Spacer para mayor precisión en el layout.
- Se agregan atributos de “tamaño mínimo” y “tamaño recomendado” para que los elementos de un form puedan adaptarse a diferentes dispositivos.

7.4 El Proyecto kAWT

A continuación se muestra un lista de los paquetes de KAWT.

Paquetes kAWT (Especificos de J2ME)	
java.awt	Contiene todas las clases para crear interfase de usuario y para pintar gráficos e imágenes.
java,awt.event	Provee interfases y clases para tratar con diferentes tipos de eventos disparados por componentes AWT.
java.awt.image	Provee clases para crear y modificar imágenes.

Tabla 7.4 Paquetes kAWT básicos

Paquetes Adicionales kAWT (J2ME y J2SE)	
de.kawt	Provee componentes AWT adicionales que no son parte del AWT estándar para el uso en la Palm y en PCs.
de.kawt.shell	Provee intérpretes de archivos que no son parte del AWT estándar para usar en la Palm y en PCs.

Tabla 7.5 Paquetes kAWT adicionales

Paquetes adicionales para CLDC(específicos para J2ME)	
java.io	Para entrada y salida de sistemas mediante flujo de datos y pone al sistema

	de archivos al tope del Sistema de Archivos de Emulación (FSE).
java.net	Provee las clases para implementar aplicaciones de red.
java.util	Contiene modelo de eventos.

Tabla 7.6 Paquetes kAWT adicionales para CLDC

7.5 Testeo de WingSOAP y KSOAP

7.5.1 Wingfoot SOAP

Es un cliente SOAP liviano de Wingfoot, que se combina con un parser XML. Puede ser usado en plataformas MIDP/CLDC, Personal Java/CDC, J2SE y J2EE.

Consideraciones sobre Wingfoot SOAP 1.02

- Serializa y des-serializa automáticamente instancias de:
  - java.lang.String
  - java.lang.Byte
  - java.lang.Short
  - java.lang.Integer
  - java.lang.Long
  - java.lang.Boolean
  - java.util.Date
  - com.wingfoot.soap.encoding.Float
  - com.wingfoot.soap.encoding.Base64

- Serializa y des-serializa automáticamente arrays de tipos primitivos.
- Devuelve arrays enviados por el servidor como arrays de objetos (Object[]).
- Para utilizar un Bean, primero debe definirse el mismo y después implementar una instancia de WSerializable.

Resultados

En la siguiente tabla se muestran las funciones testeadas y los resultados obtenidos. Los resultados posibles son “Si” o “No”. “Si” significa que se logró la transmisión correcta del tipo de dato. “No” indica que no logra la transmisión del tipo de dato.

1) EchoBoolean: Si
2) EchoInt: Si
3) EchoString: Si
4) EchoByteArray: Si
5) EchoDate: Si
6) EchoFloat: Si
7) EchoStringArray: Si
8) EchoData - Bean con Int, String, Float, Date, ByteArray, StringArray : Si
9) EchoDataArray: Si
10) EchoException: Si
11) EchoCustomizedException: Si

7.5.2 kSOAP

kSOAP es una API SOAP adecuada para J2ME, que se basa en kXML. El conjunto de características de kSOAP es un subconjunto de SOAP 1.1. Es posible agregar soporte para tipos de datos tales como decimales o arrays de bytes (base64). No soporta arrays multidimensionales[11]. La razón por la cual no se incluyen todas las características de SOAP es por las restricciones de memoria de los dispositivos con J2ME.

Consideraciones sobre kSOAP 1.2

- Automáticamente mapea los siguientes tipos SOAP en tipos Java:
  - xsd:int → java.lang.Integer
  - xsd:long → java.lang.Long

- xsd:string → java.lang.String
- xsd:boolean → java.lang.Boolean

- Si el elemento SOAP es un elemento primitivo y de un tipo por defecto, lo convierte en un objeto SoapPrimitive.
- Si el elemento SOAP es un elemento complejo, lo convierte en un objeto KvmSerializable, llamado SoapObject.
- Los hijos de un elemento complejo son propiedades dentro del SoapObject padre, según las reglas anteriores. Cada propiedad, a su vez, tiene asociado un objeto PropertyInfo con información, como el nombre del elemento SOAP.
- Lee los arrays de SOAP como un objeto java.util.Vector de Java.
- Como todos los tipos SOAP se representan como strings, el parser utiliza objetos Marshal para convertirlos en los correspondientes objetos Java.

Resultados

En la siguiente tabla se muestran las funciones testeadas y los resultados obtenidos. Los resultados posibles son “Si” o “No”. “Si” significa que se logró la transmisión correcta del tipo de dato. “No” indica que no logra la transmisión del tipo de dato.

1) EchoBoolean: Si
2) EchoInt: Si
3) EchoString: Si
4) EchoByteArray: Si
5) EchoDate: Si
6) EchoFloat: Si
7) EchoStringArray: Si
8) EchoData - Bean con Int, String, Float, Date, ByteArray, StringArray : Si
9) EchoDataArray: Si
10) EchoException: Si
11) EchoCustomizedException: Si

7.6 Gerenciamiento del Proyecto

Durante el proyecto se reportaron las horas de trabajo utilizando la herramienta de Gerenciamiento de Proyectos Microsoft Project.
En la figura 7.7 se presenta la Tabla con la división de Tareas y las horas invertidas en cada Tarea.

Nombre de Tarea	Total
Estudio de Herramientas (Forte, JBuilder)	47.5
Estudio de Tecnologías (kSoap, JSR)	70.5
WS. Conceptos Básicos	39.75
WS. Comunicación de SW con Dispositivos Móviles	75
Dispositivos Móviles	35
Interfases Gráficas para Dispositivos Móviles	124
J2ME. Conceptos Básicos	51
J2ME. Programación Básica	31
Testeo KSOAP WingSOAP	144.75
Reuniones Grupo	41.4
Reuniones Tutor Facultad	49.5
Reuniones Tutor Swordfish	33
Informe Final	437.75
Documentación Otras	48

Diseño	58.5
Implementación	260
Total	**1559.65**

Figura 7.7 Tabla de Tareas

La figura 7.8 muestra una gráfica en donde se han reagrupado las tareas para lograr un mejor análisis. El tiempo invertido en Investigación corresponde al 30% del total de horas. Un 8% corresponde a las reuniones durante el proyecto incluyendo las reuniones con el tutor de Facultad, las de grupo y las reuniones con el tutor de la empresa. El 33% de las horas se dedicó a la documentación que incluye el tiempo dedicado al informe y parte del Gerenciamiento del proyecto. El 20% corresponde al caso de estudio, incluyendo horas de análisis, diseño e implementación. Por último la parte de Testeo de clientes SOAP (kSOAP y WingSOAP) insumió el 9%.
Como se puede observar la parte de Testeo de las herramientas a utilizar para la comunicación de dispositivos móviles con Servicios Web es prácticamente la mitad del total de horas invertidas en la implementación del prototipo. Como ya mencionamos la falta de documentación de estas herramientas hizo bastante costoso el testeo de las mismas.
Otro aspecto a resaltar es el tiempo invertido en el informe final. En el comienzo del proyecto solo se había estimado un mes para la documentación, observando la gráfica se puede comprobar que el tiempo estimado fue muy poco comparado con el tiempo real invertido.

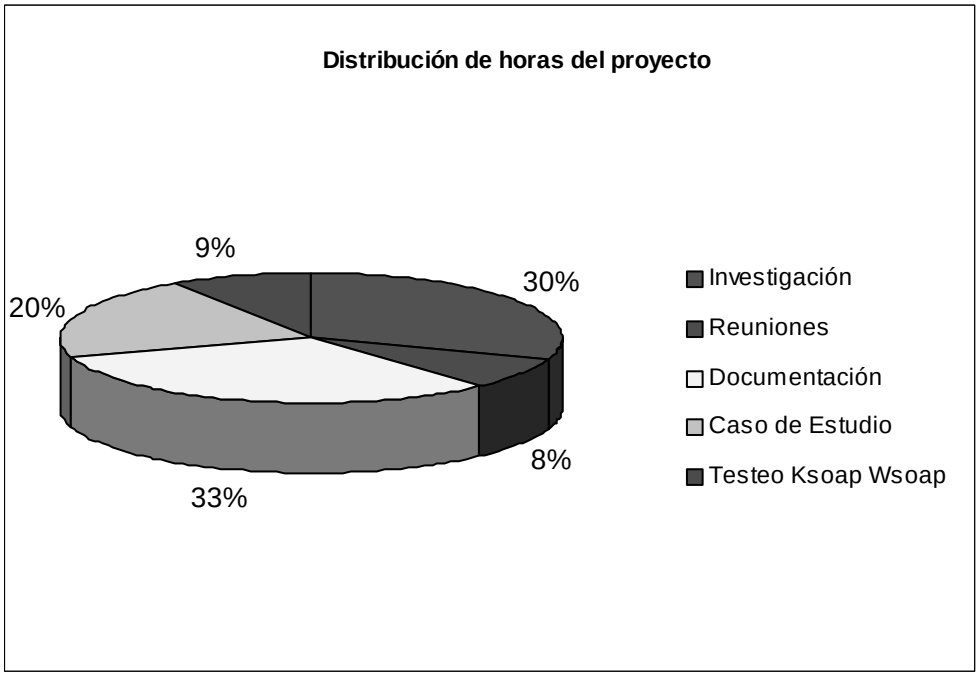

Figura 7.8 Distribución de las horas del Proyecto

8 Referencias

8.1 Bibliografía

[LAR98] "Applying UML and Patterns". Craig Larman. Edición 1998.
[CJ02] "Java Web Services". Chappell & Jewell. Edición 2002. Editorial O'Reilly.

8.2 Links

[1] Digital Asset Management: An Introduction to Key Issues
http://www.cit.cornell.edu/oit/Arch-Init/DigAssetMgmt.pdf
http://sitemaker.umich.edu/dams

[2] Glosario de SW de W3C
http://www.w3.org/TR/ws-gloss
http://www.w3.org/TR/wsa-reqs

[3] "Defining Web Services" Brent Sleeper & Bill Robins (The Stencil Group). Junio 2001.
http://www.stencilgroup.com/ideas_scope_200106wsdefined.html

[4] "Web Services Business Strategies and Architectures"
http://www.webservicesarchitect.com/content/extras/Chapter10.pdf

[5] Especificaciones de SOAP en W3C
http://www.w3.org/TR/soap12-part1

[6] Especificaciones de WSDL en W3C
http://www.w3.org/TR/wsdl

[7] Sitio oficial UDDI
http://www.uddi.org/

[8] "A Web Services Primer". Venu Vasudevan. Abril 2001.
a) www.xml.com/pub/a/2001/04/04/webservices/
Otros:
" Web Services, Sun and Java". Sun Tech Days. A developer Conference 2001-2002
b) http://fr.sun.com/developpeurs/ressources-developpeurs/pdf/web-services.pdf

[9] Sitio oficial de J2ME
http://java.sun.com/j2me/

[10] "How Personal Digital Assistant Works". Craig C. Freudenrich (Ph.D). 2001.
http://www.howstuffworks.com/pda.htm
Otros: http://www.pctechguide.com/25mob3.htm

[11] "Access Web services from wireless devices – Handle SOAP messages on MIDP devices using kSOAP" (23/Agosto/2002). Michael Juntao Yuan.
http://www.javaworld.com/javaworld/jw-08-2002/jw-0823-wireless_p.html

[12] Animating Images en MIDP (6/6/2002)
[13] Drawing Flicker-Free Graphics in MIDP (25/7/2001)
http://wireless.java.sun.com/midp/ttips/

[14] Mobile Media API Overview (Junio/2002)
http://wireless.java.sun.com/midp/ttips/

[15] JSR 135 – Mobile Media API
http://jsp.org/jsr/

[16] Enhydra on the Edge
http://www.enhydra.org/

[17] Macromedia – Flash Player
http://www.macromedia.com/software/flashplayer/resources/devices/

[18] "The Home of kXML-RPC at Enhydra.org"
http://kxmlrpc.enhydra.org/

[19] KAWT Project
http://www.trantor.de/kawt
http://www.j2meolympus.com/other/kawt/kawt.html
http://www.kawt.de/
http://www.microjava.com/articles/techtalk/kawt

[20] PocketPC
http://www.microsoft.com/mobile/pocketpc/software

[21] "Wingfoot SOAP 1.0 – User Guide"
http://www.wingfoot.com

[22] Sitio oficial de kSOAP
http://www.ksoap.org/

[23] MIDP GUI Programming
http://wireless.java.sun.com/midp/chapters/wjqusay/ch5.pdf

[24] Webopedia – The online dictionary and search engine for computer and Internet technology.
http://www.webopedia.com

[25] Java 2, Micro Edition – The Java Platform for consumer and embedded devices– SUN MicroSystems – Datasheet (Noviembre/2002)
http://java.sun.com/j2me/j2me-ds.pdf

[26] JSR 75 - PDA Optional Packages for the J2ME™ Platform
http://www.jcp.org/en/jsr/detail?id=75

[27] Getting Started with Visual Studio .NET and the Microsoft .NET Compact Framework – Microsoft SmartDevices Developer Community – Larry Roof (Marzo/2002)
http://smartdevices.microsoftdev.com/Learn/Getting+Started/508.aspx#netcfgetstarted_topic28

[28] Let the mobile games begin, Part 1 – A comparison of the philosophies, approaches, and features of J2ME and the upcoming .Net CF – Michael Juntao Yuan (21/Febrero/2002)
http://www.javaworld.com/javaworld/jw-02-2003/jw-0221-wireless.html?

[29] The K Virtual Machine
http://java.sun.com/products/cldc/ds/

[30] "The facts about mobile Web services" – Joseph Owen – 29/Enero/2003
http://zdnet.com.com/2102-1107-982536.html

[31] "Evolution of Mobile Web Services" – XML & Web Services Magazine – Jeff Jurvis- Setiembre/2002
http://www.fawcette.com/xmlmag/2002_08/magazine/columns/jjurvis/default_pf.asp

[32] JSR 172 – J2ME Web Services Specification
http://www.jcp.org/en/jsr/detail?id=172&showPrint

[33] "J2ME IDE Comparison" – Develop{net} –Michael Tylor- Junio/2002

http://www.microjava.com/articles/J2ME_IDE_Comparison.pdf

[34] Java Community Process
http://www.jcp.org/en/home/index

[35] "Introduction to the Java 2 Micro Edition (J2ME) Platform" – Vartan Piroumian – Octubre/2002
http://www.developer.com/ws/j2me/article.php/1475521

[36] "Mobile devices market in EMEA on its head" – Canalys.com – Abril/2003
http://www.geek.com/news/geeknews/2003Apr/bpd20030425019730.htm

[37] Symbian home page
http://www.symbianpages.com

[38] "J2ME CLDC API"
http://java.sun.com/j2me/docs/pdf/cldc11api.pdf

[39] The Compact Virtual Machine
http://java.sun.com/products/cdc-hi/

[40] "CDC API comparison"
http://java.sun.com/products/cdc/api.html

[41] XmlRpc home page
http://www.xmlrpc.org/

[42] MIDP Style Guide – MIDP 1.0a
http://java.sun.com/j2me/docs/alt-html/midp-style-guide7/canvas.html#wp1002192

[43] Can Web Services drive mobile applications?
http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2863482,00.html
3G:Don´t believe the hype - No killer apps?
http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2871929-2,00.html

[44] "Java readies itself for wireless Web services – Emerging Java platforms are well positioned for wireless Web services developers" -Michael Juntao Yuan and Ju Long – 21/Junio/2002
http://www.javaworld.com/javaworld/jw-06-2002/jw-0621-wireless_p.html

[45] "XML documents on the run" - Dennis M. Sosnoski
Part 1 – "SAX speeds through XML documents with parse-event streams" (febrero/2002)
http://www.javaworld.com/javaworld/jw-02-2002/jw-0208-xmljava.html
Part 2 – "Better SAX2 handling and the pull-parser alternative" (marzo/2002)
http://www.javaworld.com/javaworld/jw-03-2002/jw-0329-xmljava2.html
Part 3 – "How do SAX2 parsers perform compared to new XMLPull parsers?" (abril/2002)
http://www.javaworld.com/javaworld/jw-04-2002/jw-0426-xmljava3.html

[46] "Design of a Pull and Push Parser System for Streaming XML" - Aleksander Slominski - Indiana University Computer Science Department 1 - 2002-02-10
http://www.extreme.indiana.edu/xgws/papers/xml_push_pull/

[47] Sax 2.0: The simple api for xml - Megginson et al. - visited 07-01-00.
http://www.saxproject.org

[48] "Processing XML with Java" (Chapter 5: Reading XML) - Elliotte Rusty Harold
http://www.perfectxml.com/JavaXML.asp

[49] "Namespaces in XML" – World Wide Web Consortium – 14/Enero/1999
http://www.w3.org/TR/REC-xml-names/#ns-decl

[50] "Deliver Mobile Services Using XML And SOAP" – WirelessDevNet – 8/Febrero/2001
http://www.wirelessdevnet.com/columns/feb2001/editor14.html

[51] "Java on the PalmOS"
http://www.palmos.com/dev/tech/java/

[52] The CIO`s Guide To Wireless
http://www.synchrologic.com/images/whitepapers/cio_wireless.zip

[53] "Big designs for small devices" – Ben Hui - Diciembre 2002
http://www.javaworld.com/javaworld/jw-12-2002/jw-1213-j2medesign.html

9 Glosario

3G	Es una especificación de la ITU para la tercer generación de tecnología de comunicaciones móvil (celulares análogos son la primera, Personal Communication Service y celulares digitales son la segunda). 3G promete mejorar la velocidad de transmisión de datos mediante anchos de banda muy superiores(hasta 384 kbps) a las anteriores generaciones.
API	Application Programming Interface – Es un conjunto de clases que pueden usarse en una aplicación. A veces llamadas librerías o módulos, las APIs permiten escribir una aplicación sin reinventar segmentos comunes de código. Por ejemplo, una API para redes es algo que una aplicación puede usar para hacer conexiones de red, sin tener que entender el código subyacente.
Binding	Es una asociación entre dos entidades.
CDC	Connected Device Configuration – Es una especificación para una configuración J2ME. Conceptualmente, CDC trata con dispositivos con más memoria y procesador que CLDC; es para dispositivos con una conexión de red dedicada y un mínimo de 2 MB de memoria disponible para el sistema Java.
CDPD	Cellular Digital Packet Data - Tecnología de transmisión de datos desarrollada para usar sobre frecuencias de teléfonos celulares. CDPD usa canales celulares en el rango de 800 a 900 Mhz para transmitir datos en paquetes. Esta tecnología ofrece tasas de transferencia de datos de hasta 19.2 Kbps.
CHTML	Compact HyperText Markup Language – Un subconjunto de HTML para dispositivos móviles de recursos limitados.
CLDC	Connected, Limited Device Configuration – Es una especificación para una configuración de J2ME. El CLDC es para dispositivos con menos de 512 KB o RAM disponible para el sistema Java y una conexión de red intermitente (limitada). Especifica una máquina virtual Java reducida al mínimo llamada KVM, así como también varias APIs para servicios de aplicación. Tres de los paquetes son versiones minimizadas de los paquetes java.lang, java.io, y java.util de J2SE. Un cuarto paquete, javax.microedition.io, implementa el Generic Connection Framework, una API generalizada para conexiones de red.
COM	Component Object Model - Arquitectura desarrollada por Microsoft para construir aplicaciones basadas en componentes. Los objetos COM son componentes discretas, que exponen interfases que permiten a otras componentes y/o aplicaciones acceder a sus características.
Configuración	Las ramas fundamentales de J2ME son las configuraciones. Una configuración es una especificación que describe una máquina virtual Java (JVM) y cierto conjunto de APIs referidas a una clase específica de dispositivo.
Consumidor	Una aplicación web que usa uno o más SW.
Convertidor	A partir de la aplicación Java desarrollada, un convertidor genera una aplicación con el formato necesario (*.prc*) para poder ser instalado en un dispositivo con Palm OS.
CORBA	Common Object Request Broker Architecture - Es una arquitectura que permite que objetos se comuniquen unos con otros sin importar el lenguaje en el que fueron programados o el sistema operativo en el que corren.
CVM	Compact Virtual Machine – Es una máquina virtual Java optimizada usada por el CDC.
DAM	Digital Asset Management - Administrador de bienes digitales
DAMS	Sistema Administrador de bienes digitales
Drag & Drop	Término que indica la transferencia de datos desde un componente GUI a otro.

Emulador	Software que emula en el PC, el ambiente del dispositivo móvil y permite probar las aplicaciones móviles desarrolladas.
Engine	DAMS (Digital Asset Management System) de la empresa Swordfish.
Foundation Profile	Especificación de un perfil de J2ME basado en CDC. Agrega clases e interfaces adicionales a las APIs de CDC pero no llega a especificar APIs de internase de usuario (UI), almacenamiento persistente, o ciclo de vida de la aplicación. Otros perfiles de J2ME se basan en la combinación de CDC/Foundation: por ejemplo, el Personal Profile y el RMI Profile.
FTP	Protocolo básico de Internet de transferencia de archivos. FTP, que esta basado en TCP/IP, habilita la búsqueda y almacenamiento de archivos entre hosts en Internet.
GCF	Generic Connection Framework – Facilita las conexiones de red para dispositivos inalámbricos. Es parte del CLDC y CDC, y reside en el paquete javax.microedition.io.
GUI	Graphic User Interface – Se refiere a la interfase de usuario gráfica.
HTML	HyperText Markup Language – Lenguaje usado para crear documentos en la World Wide Web.
HTTP	HyperText Transfer Protocol. Protocolo de transferencia de HiperTexto. Protocolo de Internet, basado en TCP/IP usado para recuperar objetos de hipertexto de hosts remotos
IDE	Integrated Development Environment – Provee un ambiente de programación como una aplicación individual. Generalmente consiste de un compilador, un debugger, y un constructor GUI. Por ejemplo, el Forte para Java es un IDE Java de Sun.
IP	Internet Protocol. Protocolo básico de Internet. Habilita el envío no confiable de paquetes de datos de un host a otro. No da garantía de que el paquete sea repartido, en cuanto tiempo va a llevar el envío o si múltiples paquetes enviados van a llegar en el orden en que fueron enviados. Los protocolos que se construyen por encima de este agregan la noción de conexión y confiabilidad.
J2EE	Java 2 Platform, Enterprise Edition – Es un ambiente para desarrollo y ejecución de aplicaciones empresariales. La plataforma J2EE consiste de un conjunto de servicios, APIs y protocolos que proveen la funcionalidad para desarrollar aplicaciones web multicapas.
J2ME	Java 2, Micro Edition – Es un grupo de especificaciones y tecnologías que se refieren a Java en pequeños dispositivos. El término J2ME cubre un amplio rango de dispositivos, desde pagers (buscapersonas) y teléfonos móviles hasta decodificadores de señales y sistemas de navegación en automóviles. El mundo de J2ME se divide en configuraciones y perfiles, especificaciones que describen un ambiente Java para una clase específica de dispositivo.
J2ME WTK	J2ME Wireless Toolkit – Es un conjunto de herramientas que proveen a los desarrolladores con un ambiente de emulación, documentación y ejemplos para desarrollar aplicaciones Java para dispositivos pequeños. El J2ME WTK se basa en las implementaciones de referencia del CLDC y del MIDP, y puede ser integrado con Forte para Java.
J2SE	Java 2, Standar Edition – Núcleo de la plataforma de tecnología Java.
JAD	Java Application Descriptor File. Archivo que contiene información acerca de un MIDlet.
JAR	Java Archive File. Formato de Archivo que se usa para agrupar muchos archivos en uno solo.
Java Card	La especificación Java Card le permite a Java correr sobre tarjetas inteligentes y otros pequeños dispositivos. La API de Java Card es compatible con estándares internacionales formales, tales como ISO7816, y estándares industriales, tales como, Europay/Master Card/Visa (EMV).
JCP	Java Community Process – Es una organización abierta internacional de desarrolladores Java con licencia, quienes desarrollan y revisan

	especificaciones de Java, implementaciones de referencia, y kits de compatibilidad de tecnologías a través de un proceso formal.
JSR	Java Specification Request – Es la descripción actual de especificaciones, propuestas y finales, para la plataforma Java. Las JSRs son revisadas por el JCP y el público antes de que se haga la versión final de una especificación.
JVM	Java Virtual Machine – Es una máquina virtual para la plataforma Java.
KJava	Es un término desactualizado para J2ME. Apareció en un paquete inicial de software Java para PalmOS, liberado en el evento JavaOne 2000. Las clases para esa versión están en el paquete com.sun.kjava.
kSOAP	Es una API de SOAP adecuada para J2ME, basada en kXML.
kXML	Proyecto que provee un parser XML liviano que puede ser usado con J2ME.
KVM	Kilobyte Virtual Machine – La 'K' en KVM viene de kilobyte, lo que significa que la KVM corre en kilobytes de memoria, no en megabytes. Es una JVM compacta que está diseñada para dispositivos pequeños y soporta sólo un subconjunto de las características de la JVM. Por ejemplo, la KVM no soporta operaciones de punto flotante ni el método object.finalize(). El uso de la KVM lo especifica CLDC.
LAN	Local Area Network - Es un grupo de dispositivos conectados con varias tecnologías de comunicación en un área geográfica pequeña. La tecnología LAN más ampliamente usada es Ethernet. La comunicación sobre una LAN puede ser Peer-to-Peer o Client-Server.
LCDUI	Es una abreviatura para las APIs de interfase de usuario del MIDP, contenidas en el paquete javax.microedition.lcdui. Estrictamente, LCDUI viene de Liquid Crystal Display User Interface, que es un toolkit de interfaces de usuario para pantallas de dispositivos pequeños, las que comúnmente son pantallas LCD.
MIDlet	Es una aplicación escrita para el MIDP. Las aplicaciones MIDlet son subclases de la clase javax.microedition.midlet.MIDlet que está definida por el MIDP.
Suite MIDlet	Los MIDlets son empaquetados y distribuidos como MIDlet suites. Un MIDlet suite puede contener uno o más MIDlets. El MIDlet suite consiste de dos archivos, un archivo descriptor de aplicación con extensión .jad y un archivo de registro con extensión .jar. El descriptor lista el nombre del archivo de registro, los nombres y nombres de clase para cada MIDlet en el suite, y otra información. El archivo de registro contiene clases MIDlet y archivos de recursos.
MIDP	Mobile Information Device Profile – Es una especificación para un perfil de J2ME. Está sobre el CLDC y agrega APIs para el ciclo de vida de la aplicación, interfaces de usuario, redes, y almacenamiento persistente.
MIME	Estándar que permite que datos binarios sean publicados y leídos en Internet.
Obfuscación	Es una técnica utilizada para complicar código, es decir lo hace más difícil de entender cuando es descompilado, pero generalmente no afecta la funcionalidad del código. Los programas de obfuscación pueden ser usados para proteger programas Java haciéndolos más difíciles de descompilar.
PDA	Personal Digital Assistant – Dispositivo portátil, que combina características informáticas, teléfono/fax y redes. Los PDAs son más grandes que los teléfonos móviles pero más pequeños que los decodificadores de señales.
PDAP	Personal Digital Assistant Profile – Es una especificación de un perfil de J2ME diseñado para PDAs. El PDAP está sobre el CLDC y especificará las APIs de interfaces de usuario y almacenamiento persistente. El PDAP actualmente está en desarrollo con el proceso.
Personal Profile	Es una especificación de un perfil de J2ME. Se ubica sobre el Foundation Profile y el CDC, y será la próxima generación de la tecnología PersonalJava. La especificación actualmente está en desarrollo con el proceso JCP.

PersonalJava	En un ambiente Java basado en la JVM y un conjunto de APIs, similar al ambiente JDK 1.1. Incluye el Touchable Look and Feel (también conocido como Truffle), un toolkit gráfico que es optimizado para dispositivos de consumo con una pantalla sensible al tacto. PersonalJava será incluido en J2ME en el próximo Personal Profile, sobre CDC.
PIM	Personal Information Management - Tipo de aplicación diseñada para ayudar al usuario a organizar información. La mayoría de los PIMs permiten ingresar textos, incluyen calendario y calculadora.
PNG	Portable Network Graphics – Es un formato de imagen que ofrece compresión sin pérdida y flexibilidad de almacenamiento. La especificación del MIDP requiere que las implementaciones reconozcan ciertos tipos de imágenes PNG.
Pocket PC	Software para PDAs. Comprende el sistema operativo y componentes de aplicaciones específicos de cada modelo de PDA.
PRC	Palm Resource Code – Es el formato de archivo para las aplicaciones de Palm OS.
Preverificación	Debido a la poca memoria y procesador disponible en los dispositivos, el proceso de verificación de clases se divide en dos. El primer proceso es la preverificación que se hace fuera del dispositivo y usando una herramienta de preverificación. El segundo proceso es la verificación que se hace sobre el dispositivo.
RMI	Remote Method Invocation - Un modelo de objetos distribuido para Java, en el cual los métodos de objetos remotos escritos en el lenguaje de programación Java pueden ser invocados desde otras máquinas virtuales Java, posiblemente en diferentes hosts.
RMI Profile	Es una especificación de un perfil de J2ME diseñada para soportar el sistema de objetos distribuido RMI (Remote Method Invocation) de Java. Los dispositivos que implementen el RMI Profile serán capaces de interoperar mediante RMI con otros dispositivos Java, incluyendo J2SE. El RMI Profile se basa en el Foundation Profile, el que a su vez se basa en CDC.
RMS	Record Management System – Es una base de datos simple orientada a registros que permite a un MIDlet almacenar y devolver información de forma persistente. Diferentes MIDlets pueden usar también el RMS para compartir datos.
RPC	Remote Procedure Call – Una llamada a un procedimiento local que es invocado desde un programa o espacio de direcciones no-local. Habilita la lógica de aplicaciones para dividirla entre un cliente y un servidor de forma que el mejor use los recursos disponibles.
SDK	Software Development Kit – Es un conjunto de herramientas usado para desarrollar aplicaciones para una plataforma particular. Un SDK generalmente contiene un compilador, un linker, y un debugger. También puede contener librerías y documentación para APIs.
Set-Top box	Dispositivo electrónico usado para recibir y decodificar señales de televisión digital y para conectarse a Internet a través de la televisión del usuario en vez de hacerlo a través de un PC.
SGML	Standard Generalized Markup Language – Es un estándar complejo (ISO 8879) para mantener depósitos de documentación estructurada.
SMTP	Simple Mail Transfer Protocol – Es un protocolo TCP/IP usado para enviar y recibir e-mail.
SOA	Service Oriented-Architecture – Los SW se usan para construir arquitecturas orientadas a servicios. Cada servicio se diseña para ser integrado rápida y fácilmente en sistemas que pueden no compartir la misma semántica fundamental.
SOAP	Simple Object Access Protocol – Es un protocolo basado en XML que permite a objetos de cualquier tipo comunicarse en un ambiente distribuido. SOAP se usa en desarrollo de Servicios Web.
TCP/IP	Transmission Control Protocol – Protocolo de control de transmisión basado

	en IP. Este es un protocolo de Internet para brindar envío confiable de flujos de datos desde un host a otro.
UDDI	Universal Description, Discovery, and Integration – Es un estándar basado en XML para describir, publicar, y encontrar Servicios Web. UDDI es una especificación para un registro distribuido de Servicios Web.
URI	Uniform Resource Identifier - Cadena de caracteres compacta para identificar un recurso físico o abstracto. Una URI puede ser una URL o una URN. Las URLs y URNs son entidades concretas que existen en realidad; Una URI es algo abstracto.
WML	Wireless Markup Language – Es un lenguaje simple usado para crear aplicaciones para dispositivos pequeños inalámbricos como teléfonos móviles. WML es análogo a HTML en el World Wide Web.
WSDL	Web Services Description Language – Es un formato XML que describe la interfase de un Servicio Web de forma estandarizada.
XML	Extensible Markup Language – Es una forma más simple pero restringida de SGML. El markup describe el significado del texto. XML permite separar contenido de datos. Fue creado para que documentos ricos en estructura puedan ser usados sobre la Web.