Universidad de las Ciencias Informáticas

BLUEEYE. UNA PLATAFORMA PARA LA GESTIÓN DE SERVICIOS DE VALOR AGREGADO PARA DISPOSITIVOS MÓVILES

Tesis para optar por el título académico de Máster en Informática Aplicada

Autor: Lic. Noel García Guimeras

Tutores: MsC. Pedro Ángel Romero
Dr. Abel Alba Alfonso

Ciudad de La Habana
Mayo de 2010

-No me preocupa mayormente el lugar... -dijo Alicia.
-En tal caso poco importa el camino -declaró el Gato.
-... con tal de llegar a alguna parte -añadió Alicia
a modo de explicación.
-¡Oh! -dijo el Gato-: puede usted estar segura de llegar,
con tal de que camine durante un tiempo bastante largo.

Lewis Carroll
"Alicia en el País de Maravillas"

A modo de agradecimientos

I. Las influencias

A Matilde Bringas, quien dijo una vez que “el jugador es mal programador”, aunque no tenía razón.

A Juan Miguel Gómez, por prestarme mi primer libro de programación, en Turbo Pascal.

A Alexei Rodríguez, quien interrumpió una tarde su trabajo para explicarle el significado del galimatías

```
type
      editorptr = ^editor ;
      editor = object
            constructor init () ;
            destructor done () ;
            ...
      end ;
```

a un chama de trece o catorce años (entiéndase: a mí).

A Daniel de Bittencourt, el Viejo, el Uruguayo, quien me guió en aprender el arte de la programación. “Lo perfecto es enemigo de lo bueno”, dijo una vez. Aparentemente él sí tenía razón.

A la pipa, el paquete de mate y el termo con agua caliente que nunca se separaban del Uruguayo.

Al libro de programación de Miguel Katrib y Eduardo Quesada, donde se desentrañaba el problema de entender la recursividad (que casualmente consiste en entender la recursividad).

A Miguel Cepero y su Fortaleza en las Entrañas de la Bestia: escribiendo un clon de la Fortaleza logré comprender ese asunto enrevesado del polimorfismo.

A Kubilai, por aquel problema de las N parejas y los N preservativos, y por el 4 que se convirtió en 3; a Idania y Hedrie por los 4 que se convirtieron en 5.

A Marta Montes, quien un día me preguntó si “yo era alumno de su grupo”. Sí profe, yo era su alumno, usted me había dado clases durante todo el semestre anterior.

A los incunables de Jorge Lemagne; a los juegos de pelota durante los turnos de Eugenio.

A FIFA y Grand Theft Auto, que me robaban los días y las noches frente a la PC (¿y decían que el jugador era mal programador?).

A Edgardo Felipe Riverón y Roberto Rodríguez, por la “Segmentación de la red vascular en imágenes de retina humana”.

A Dijkstra, por denigrar del uso del **`goto`**; a Linus Torvalds por utilizarlo sin pudor.

A Andy Tannenbaum, por continuar defendiendo la arquitectura de microkernel, y por algún que otro libro que ha escrito.

A Richard “rms” Stallman, santo de la iglesia de Emacs, quien sin quererlo introdujo el socialismo al mundo del software; por mantener la extraña creencia de que el asunto del software libre es ante todo una cuestión ética.

```
def a_guido_van_rossum():
      '''
      porque además de éticas,
      también hay cuestiones estéticas
      '''
      por_los_bloques()
      indentados()
      de_python()
```

II. Las anti-influencias

No a Bill Gate$, y a Micro$oft Corporation.

No al MS-DOS, si tengo el `bash`.

No a GWH28-DGCMP-P6RC4-6J4MT-3HFDY.

No a los virus y los troyanos: no a los antivirus y los antitroyanos.

No a los End User License Agreement, ni a los “certifica que no es ciudadano de Cuba, Irán, Libia, Corea del Norte, Sudán ni Siria ni de cualquier otro país al que los Estados Unidos embarga bienes”.
No a los que bloquean `code.google.com`, `sourceforge.net` y `dev.java.net`, por las razones anteriores (lo cual me hace sentir enormemente feliz).
No a los que bloquean los mecanismos para acceder a los sitos arriba mencionados, por razones distintas a las anteriores (lo cual lleva mi felicidad al éxtasis).
No al Digital Restrictions Management.
No al RUP en exceso, a la documentación en exceso, a la ingeniería en exceso. ¿Y quién escribe el código? Los indios, claro.
No a aquellos que suelen pensar que el recurso del método lo es todo. “Escribir software no es un arte”, sentenció uno de ellos. Quizá los indios que escriben el código crean que sí.

III. Los que se fueron, los que se quedaron, los que faltan

A todos los que están y a todos los que no están: Abel Hernández, Alejandro Almeida, Amaury Suárez, Carlos Cueto, Cira Margarita, Cristina López, Dania Souchay, Denis Raúl Méndez, Dianela Zayas, Emilio Henríquez, Ernesto Rodríguez Reina, Iyima Casañas, Isachi Abreu, Karina Marrón, Llinersy Uranga, Luis Casadesus, Mariela Amaro y Mariela León, Michel Palenzuela, Mijail del Toro, Norbelis Leyva, Rayda Zaldívar, Rosa Muñoz, Tania Gual, Tomás Lay, et al.
A Alejo Carpentier, quien citó Lewis Carroll al inicio de “La consagración de la primavera”, y a quien el autor de estas líneas “hubo de hurtarle dicha cita, siendo las 18:43 del día 15 de agosto de 2009, utilizándose la misma al inicio de la tesis de maestría del acusado”, según consta en acta.
A Abel Alba y Pedro Ángel Romero, por sus sugerencias durante la escritura de este trabajo, y por hacer las veces de tutores. A Dania Souchay por la revisión de último momento.
A Javier Ramón García, quien definió la arquitectura general de la plataforma (él fue el de la idea de destrozar el huerto).
A Alejandro Almeida y Arián Zulueta, quienes trabajaron en la implementación del protocolo SMPP y en la integración de la plataforma con el operador móvil Cubacel.
A Mariela Amaro, Mariela León, York Figueroa y Pedro Rodríguez (ojo: no el delantero del Barça), quienes participaron en la implementación de la mayoría de las aplicaciones de BlueEye.
A Ariadna Piñeiro, que diseñó el logo.

Al resto del piquete de Procyon Soluciones.
A Noel García, por la paciencia que se tiene.
A mis padres, por haberme soportado durante los últimos treinta años.
A todos los amigos, que a veces molestan en los momentos más inoportunos, pero para eso son los amigos; a los que no son tan amigos, pero no por eso dejan de molestar.
A todos los que faltan o deseen incluirse.

***

Declaración de autoría

Declaro que yo, Noel García Guimeras, soy autor del presente trabajo, y autorizo a la Universidad de las Ciencias Informáticas (UCI) a utilizar el mismo como material de consulta o estudio para futuras investigaciones o desarrollo de software. Para que así conste se firma el presente documento a los _____ días del mes de _____ de _____.

_____	_____	_____
Lic. Noel García Guimeras	MsC. Pedro Ángel Romero	Dr. Abel Alba Alfonso
Autor	Tutor	Tutor

Resumen

En los últimos años la telefonía móvil ha ganado gran importancia y popularidad a nivel mundial, aunque en nuestro país se están dando los primeros pasos en este sentido. Entre los servicios que brinda la telefonía móvil se encuentra el servicio de mensajes de texto, el cual ha dado lugar al surgimiento de una gran cantidad de servicios de valor agregado. En la actualidad no existe una plataforma para la gestión de estos servicios de valor agregado desarrollada en el país.

La plataforma BlueEye es un software desarrollado utilizando tecnología Java con el objetivo de gestionar servicios de valor agregado basados en mensajes de texto, basándose en las funcionalidades fundamentales de otras plataformas similares a nivel mundial.

BlueEye da la posibilidad de comunicación con uno o varios operadores móviles por medio de distintos protocolos al mismo tiempo, y brinda un SDK que facilita y agiliza el desarrollo de los nuevos servicios. Estos servicios se desarrollan en forma de módulos independientes que se pueden desplegar en la plataforma sin afectar al resto de los servicios que se están ejecutando en un momento determinado. Además ofrece funciones de monitoreo y administración, reportes de tráfico, y una interfaz para brindar servicios de mensajería a terceros, implementada como un servicio más de la plataforma.

En el presente trabajo se exponen además las primeras aplicaciones implementadas sobre la plataforma BlueEye, así como las diversas pruebas a que ha sido sometida, incluyendo su despliegue en fase de pruebas en el operador móvil Cubacel.

Palabras clave: plataforma para servicios de valor agregado, servicios de valor agregado, servicios basados en mensajería móvil

Tabla de contenidos

Índice de figuras

INTRODUCCIÓN

La tecnología móvil ha crecido rápidamente en alcance e importancia a nivel mundial. En el año 2006 un artículo del periódico español El País reflejaba que, según la Asociación de Empresas de Servicios a Móviles (AESAM), en el año 2005 habían circulado 600 000 mensajes de texto por las redes españolas, lo que suponía una facturación de 350 millones de euros[Molist, 2006]. A eso había que añadir, según la consultora Gaptel, 525 millones más en concepto de navegación, descargas y votaciones, en un negocio que apenas había emergido en 2002. Asimismo señalaba que el 99.98% de estas ganancias se habían obtenido a partir de servicios de valor agregado.

Ya en 2008, la firma de análisis Gartner anunciaba que entre 2002 y 2006 los beneficios por concepto de mensajes cortos habían crecido un 29.8% a nivel mundial, al tiempo que preveía para 2011 un incremento de un 9.9%[Collazos, 2007].

No obstante este crecimiento a nivel mundial, en Cuba aun se están dando los primeros pasos en este sentido. La reciente autorización de la venta masiva de líneas y teléfonos móviles a toda la población hace pensar que en un período de tiempo relativamente corto pueda surgir un mercado para todo tipo de contenido para teléfonos móviles (entiéndase imágenes, tonos, videos, animaciones, juegos) así como para otros servicios de valor agregado basados en la telefonía celular.

Por esta razón, en el año 2005 surgió el proyecto Procyon Soluciones, centrado en el desarrollo de software para telecomunicaciones. En sus inicios el proyecto estuvo enmarcado en la Infraestructura Productiva de la Universidad de las Ciencias Informáticas (UCI), en cooperación con la empresa Softel del Ministerio de la Informática y las Comunicaciones (MIC). En la actualidad este proyecto se ha convertido en la División de Telecomunicaciones de Desoft, también perteneciente al MIC.

La situación que se le planteó a Procyon Soluciones fue la comercialización de servicios y contenidos específicamente destinados a teléfonos celulares, mediante plataformas diseñadas para la gestión de estos servicios y contenido, que también se podían comercializar a otras entidades interesadas en desempeñarse como proveedores de servicios o contenido. En el país no se había tenido una experiencia anterior en este sentido, y por tanto no existían plataformas para la gestión de servicios y contenido. La otra opción, comprar este software a empresas extranjeras, podía resultar costosa y podía conllevar a la dependencia de esa empresa a la hora de dar soporte a las plataformas o para la

implementación de nuevos servicios,

Además, el estado cubano ha venido fomentado en los últimos tiempos una política de soberanía tecnológica, como vía de desarrollo del país en las condiciones actuales del mundo. De esta forma se pretende elevar la seguridad y vitalidad de las comunicaciones, las tecnologías abiertas (software libre), y el aumento de las investigaciones y su vínculo con sectores productivos, sociales y de servicios[Tejera, 2009]. Por estas razones, se determinó la necesidad de desarrollar las plataformas para la gestión de servicios y contenido para dispositivos móviles dentro del país.

Los servicios de valor agregado destinados a dispositivos móviles son todos aquellos servicios que se basan en los servicios básicos de envío de mensajes de texto, multimedia y notificaciones, para crear nuevos servicios que pueden resultar de interés para los clientes y que van más allá de los servicios básicos. De esta forma los dispositivos móviles se convierten en herramientas capaces de brindar al usuario mayores beneficios, ya que pueden obtener todo tipo de información y realizar diversas operaciones desde su teléfono. Asimismo beneficia a los operadores móviles, a los desarrolladores de los servicios y a los proveedores de información, a partir del tráfico adicional que se genera gracias a estos. Estos servicios son muy populares a nivel mundial y entre los más conocidos se cuentan los servicios de votaciones, concursos, noticias, subscripciones, carteleras y muchos otros.

En este contexto, teniendo en cuenta el auge y la importancia que ha ganado en los últimos tiempos la telefonía móvil, se plantea el siguiente **problema**:

- Es necesario desarrollar en el país servicios de valor agregado destinados a dispositivos móviles.

A partir de este problema, surge la siguiente **hipótesis**:

- Si se desarrolla una plataforma capaz de comunicarse con los centros de mensajería y que brinde un conjunto de funcionalidades reusables para la implementación de servicios de valor agregado para dispositivos móviles, se puede estandarizar el desarrollo de servicios de valor agregado.

El problema planteado tiene como objeto de estudio el desarrollo de aplicaciones empresariales, y se enmarca en el campo del desarrollo de aplicaciones orientadas al manejo de servicios destinados a

dispositivos móviles.

Como **objetivo general** de este trabajo se plantea:

- Sentar las bases tecnológicas para el desarrollo de servicios de valor agregado para dispositivos móviles, por medio de una plataforma capaz de comunicarse con los distintos operadores móviles, y que permita gestionar y estandarizar el desarrollo de servicios de valor agregado.

Para ello se han definido los siguientes **objetivos específicos**:

- Realizar un estudio de las plataformas para la gestión de servicios de valor agregado que existen a nivel mundial.
- Realizar un estudio de las tecnologías a utilizar en el desarrollo de una plataforma portable.
- Implementar un SDK que permita desarrollar de una forma estándar las aplicaciones que implementan los servicios de valor agregado.
- Implementar el núcleo de la plataforma que gestione las distintas aplicaciones o servicios, y la comunicación con distintos operadores móviles a través de distintos protocolos.
- Evaluar el funcionamiento de la plataforma en un entorno de prueba.
- Integrar la plataforma con el operador móvil del país (Cubacel).

Para cumplir los objetivos específicos se definieron las siguientes tareas:

- Estudio de las principales características de las plataformas implementadas a nivel mundial, así como la factibilidad de la implementación de estas características como parte de la plataforma a desarrollar.
- Identificación de las tecnologías que más se adecuan para la implementación de la plataforma, teniendo en cuenta el uso de plataformas libres o de código abierto, de acuerdo con la política de soberanía tecnológica del país; así como que faciliten la portabilidad de la plataforma hacia diferentes hardwares y sistemas operativos.

- Definición de la comunicación entre la plataforma y el centro de mensajería, teniendo en cuenta de que podría utilizarse en otros entornos en donde deba comunicarse con varios centros de mensajería al mismo tiempo.

- Elaboración del esquema según el cual la plataforma debe enrutar los mensajes entrantes a las diferentes aplicaciones.

- Implementación de un SDK de funcionalidades comunes que debe ofrecer la plataforma a sus aplicaciones, con el objetivo de facilitar y agilizar el desarrollo de nuevas aplicaciones.

- Realización de pruebas de estrés para comprobar el funcionamiento de la plataforma ante una avalancha de mensajes, en un ambiente de prueba.

- Realización de pruebas de la plataforma y sus servicios en un ambiente real, para comprobar el funcionamiento de la plataforma una vez integrada con el centro de mensajería.

- Implementación de otras aplicaciones adicionales.

Como valor práctico de este trabajo se puede señalar que por primera vez en el país se desarrolla una plataforma para la gestión de servicios de valor agregado destinados a dispositivos móviles, una tecnología que, si bien es conocida y utilizada a nivel mundial, dentro del país aún está dando sus primeros pasos. De esta forma se brinda al país una plataforma de factura nacional, lo cual posibilita sustituir la importación de un producto similar para los centros de mensajería nacional. Al mismo tiempo se brinda un software de alta calidad, capaz de interactuar con los centros de mensajería y a su vez ofrecer gran variedad de servicios, los cuales pueden ser de beneficio para todos los usuarios de dispositivos móviles del país, así como para empresas y organismos estatales. También sirve como impulso para que se extienda en el país el uso de la tecnología móvil, dándole un nuevo sentido al uso de los teléfonos celulares, más allá que el realizar simples llamadas telefónicas.

División por capítulos

Este trabajo está dividido en 3 capítulos. En el Capítulo 1 se introducirán algunos conceptos referentes

a la telefonía móvil, así como su aplicación en la creación de servicios de valor agregado. Se analizará el estado del arte con respecto a las plataformas para la gestión de servicios destinados a dispositivos móviles existentes a nivel mundial y se expondrán algunos factores que inciden en la calidad del software, así como la importancia de la aplicación de distintos tipos de pruebas para obtener un producto de mayor calidad.

En el Capítulo 2 se tratará el tema de lo patrones de diseño, haciendo énfasis en los patrones fundamentales utilizados en la construcción de la plataforma y se expondrán las tecnologías utilizadas para la implementación de la plataforma, basadas principalmente en la plataforma Java Enterprise Edition.

En el Capítulo 3 se presentará la plataforma BlueEye, para el manejo de servicios destinados a dispositivos móviles. En este capítulo se expondrán los aspectos fundamentales de la implementación y funcionamiento de la plataforma.

I. PLATAFORMAS PARA LA GESTIÓN DE SERVICIOS PARA DISPOSITIVOS MÓVILES

En el presente capítulo se introducirán algunos conceptos referentes a la telefonía móvil, así como su aplicación en la creación de servicios de valor agregado. Se analizará el estado del arte con respecto a las plataformas para la gestión de servicios destinados a dispositivos móviles existentes a nivel mundial, dado que en el país este tipo de tecnología es incipiente, haciendo énfasis en las características fundamentales de estas plataformas y su modelo de negocio, las cuales son esenciales para determinar las características y el modelo de negocio de la plataforma que se va a desarrollar.

Asimismo se expondrán algunos factores que inciden en la calidad del software como son la corrección, la robustez, la extensibilidad, la portabilidad y la reusabilidad; así como la importancia de la aplicación de distintos tipos de pruebas para obtener un producto de mayor calidad.

I.1. Introducción a la telefonía móvil

La telefonía móvil o telefonía celular consiste en la transmisión asistida de señales a lo largo de una distancia determinada, con el propósito de comunicar desde y hacia un dispositivo móvil o usuario. Es una tecnología que permite que los usuarios se conecten mientras se encuentran en movimiento, en su casa, en la oficina, un auto o incluso en el mar[Ishii+, 1991]. La telefonía móvil básicamente está formada por dos grandes partes: una red de comunicaciones o red de telefonía móvil, y los terminales o teléfonos móviles que permiten el acceso a dicha red.

La telefonía móvil emplea ondas de radio, y las señales se transmiten a través del aire. Las ondas emitidas por la terminal son recibidas por la antena más próxima, que, a su vez, las envía a la estación base, la cual se comunica con la red telefónica conmutada, tal y como se muestra en la Figura 1.

A pesar de que la telefonía celular fue concebida estrictamente para la voz, hoy es capaz de ofrecer otro tipo de servicios de acceso a datos tales como imágenes, audio y video. De esta forma, gracias a la evolución de los dispositivos móviles, unida al aumento del ancho de banda de las redes, es posible

desarrollar aplicaciones con más prestaciones orientadas a la telefonía inalámbrica.

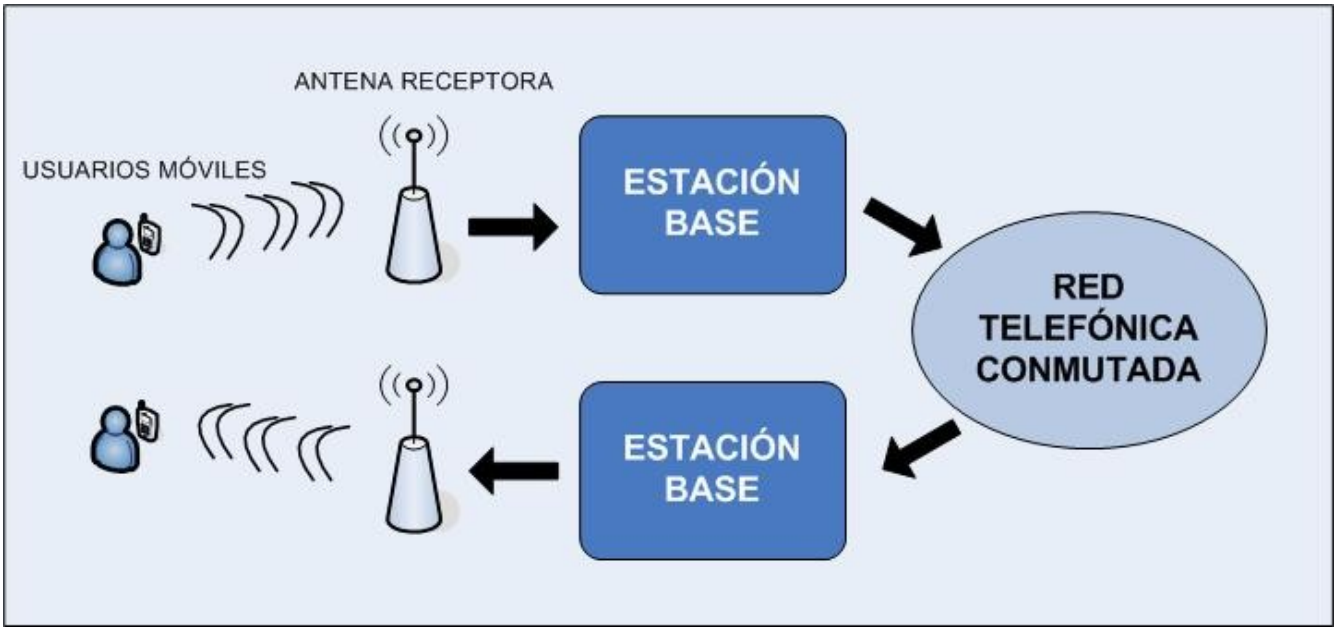

Figura 1: Transmisión de señales en la telefonía móvil.

I.1.1. Evolución de la telefonía móvil

La evolución de la telefonía móvil se puede ver a lo largo del tiempo a través de sus distintas generaciones. La telefonía móvil de primera generación (1G) se basaba en tecnología analógica y fue lanzada en la década del 80 del siglo pasado. Estos teléfonos continuaron después del lanzamiento comercial de los teléfonos móviles de segunda generación (2G). La mayor diferencia entre el 1G y el 2G es que el 1G es analógico y el 2G es digital. Aunque ambos usan sistemas digitales para conectar las radio-bases al resto del sistema telefónico, la llamada es cifrada cuando se usa 2G[Farley, 2007].

La telefonía móvil 2G no constituye un estándar o un protocolo, sino que es una forma de marcar el cambio de protocolos de telefonía móvil analógica a digital. La llegada de la segunda generación de telefonía móvil fue alrededor de 1990 y su desarrollo deriva de la necesidad de poder tener un mayor manejo de llamadas en prácticamente los mismos espectros de radiofrecuencia asignados a la telefonía móvil. Para esto se introdujeron protocolos de telefonía digital que además de permitir el uso de más

enlaces simultáneos en un mismo ancho de banda, permitían integrar otros servicios, que anteriormente eran independientes, en la misma señal, como es el caso del envío de mensajes de texto en un Servicio de Mensajería Corta, que se explicará más adelante, y una mayor capacidad de envío de datos desde dispositivos de fax y módem. 2G abarca varios protocolos distintos desarrollados por varias compañías e incompatibles entre sí, lo que limitaba el área de uso de los teléfonos móviles a las regiones con compañías que les dieran soporte[Farley, 2007].

La siguiente generación es conocida como generación 2.5 (2.5G), y constituye un puente entre la segunda y la tercera generación (3G). Como tal no existe ningún estándar ni tecnología a la que se pueda llamar 2.5G, pero suelen ser denominados así a algunos teléfonos móviles 2G que incorporan algunas de las mejoras y tecnologías del estándar 3G como es el caso del Servicio General de Radio-Paquetes en redes 2G y con tasas de transferencia de datos superiores a los teléfonos 2G regulares pero inferiores a 3G. En la actualidad, la red de telefonía celular de nuestro país puede ser considerada de generación 2.5[Farley, 2007].

La tercera generación (3G) constituye una familia de estándares para comunicaciones inalámbricas definidos por la International Telecommunication Union (ITU). Los servicios asociados con la tercera generación proporcionan la posibilidad de transferir tanto voz y datos (en el caso de una llamada telefónica) y datos que no contienen voz (como la descarga de programas, intercambio de correos y mensajería instantánea). Inicialmente la instalación de redes 3G fue demasiado lenta. Esto se debió a que los operadores necesitaban adquirir una licencia adicional para un espectro de frecuencias diferente al que era utilizado por las tecnologías anteriores 2G. El primer país en implementar una red comercial 3G a gran escala fue Japón y en la actualidad existen 164 redes comerciales en 73 países usando esta tecnología[ITU, 2000][ITU, 2007].

Aunque la cuarta generación (4G) hasta el momento no se encuentra definida, se puede augurar en qué consistirá en base a lo ya establecido. La 4G estará basada totalmente en IP siendo un sistema de sistemas y una red de redes. Se alcanzará después de la convergencia entre las redes de cables e inalámbricas así como en ordenadores, dispositivos eléctricos y en tecnologías de la información y con otras convergencias para proveer velocidades de acceso entre 100 Mbps en movimiento y 1 Gbps en

reposo. Mantendrá una calidad de servicio de punta a punta de alta seguridad para permitir ofrecer servicios de cualquier clase en cualquier momento, en cualquier lugar, con el mínimo coste posible. El Wireless World Research Forum (WWRF) define 4G como una red que funciona en la tecnología de Internet, combinándola con otros usos y tecnologías tales como Wi-Fi y WiMAX. La 4G no es una tecnología o estándar definido, sino una colección de tecnologías y protocolos para permitir el máximo rendimiento de procesamiento con la red inalámbrica más barata. El IEEE aún no se ha pronunciado, designando a la 4G como "más allá de la 3G"[Mohr, 2002][Ibrahim, 2002] .

I.1.2. Tipos de mensajes

Uno de los estándares más difundidos a nivel mundial es el Sistema Global para Comunicaciones Móviles (*Global System for Mobile Communications*, o GSM). GSM soporta múltiples servicios, entre ellos se destaca el Servicio de Mensajería Corta (*Short Message Service*, o SMS) el cual es un servicio de telefonía móvil universalmente aceptado, que posibilita la transmisión de mensajes cortos alfanuméricos entre usuarios móviles y sistemas externos como correo electrónico, o sistemas de correos de voz[IEC, 2007a].

Un mensaje corto es un mensaje de texto breve enviado desde un dispositivo móvil a otro, a través de un SMS. Un mensaje corto estándar consiste en 160 caracteres alfanuméricos como máximo, aunque se pueden enviar mensajes con mayor cantidad de caracteres utilizando compresión de datos.

Los SMS no se envían punto a punto, por lo cual, se necesita una entidad en la red inalámbrica encargada del proceso. Esta entidad conocida, como Centro de Servicio de Mensajes Cortos (*Short Message Service Center*, o SMSC), que es el responsable de la transmisión, almacenamiento y reenvío de mensajes cortos entre Entidades de Mensajería Corta (*Short Message Entity*, o SME)[TT, 2005a].

El estándar GSM continuó su desarrollo y con la versión 97 de este estándar desarrollado por 3rd Generation Partnership Project (3GPP) surgió el Servicio General de Radio-Paquetes (*General Packet Radio Service*, o GPRS), donde se adicionaron capacidades de empaquetamiento de datos. Con el surgimiento de GPRS se adicionaron nuevos servicios, como el Servicio de Mensajes Multimedia, diseñado específicamente para GPRS[IEC, 2007b].

Un mensaje multimedia se envía a través del Servicio de Mensajes Multimedia (*Multimedia Messaging Service*, o MMS), que es una tecnología de comunicaciones que permite a los usuarios intercambiar mensajes multimedia entre teléfonos móviles y otros dispositivos. MMS es una extensión de SMS, que define una manera de enviar y recibir, casi instantáneamente, mensajes inalámbricos que pueden incluir imágenes, audio y video, además de texto[TT, 2005b].

El servicio MMS puede replicar el comportamiento de un correo electrónico con archivos adjuntos, y como valor añadido permite utilizar presentaciones animadas. Esto es posible mediante la inclusión en el mensaje de un elemento de presentación llamado Lenguaje de Integración de Multimedia Sincronizada, el cual es un lenguaje basado en XML, que permite a los desarrolladores mezclar distintos medios para ser presentados y sincronizados unos con otros. El mensaje puede estar compuesto por varias páginas consecutivas, denominadas diapositivas, con un tiempo de presentación de cada página que se determina al componer el mensaje. Cada una de ellas puede contener uno o varios de los siguientes campos: texto, imágenes (simples o animadas), sonido e incluso video en los terminales que lo soporten[TT, 2005b].

Otro de los servicios que soporta el estándar GSM y por consiguiente GPRS son las Notificaciones WAP (conocidas como WAP Push). Estas notificaciones por lo general hacen referencia a un MMS, a un correo electrónico o a un URL para la descarga de contenidos como imágenes, juegos, entre otros. Una notificación WAP puede estar formada por un mensaje de texto y un URL, o solamente por un URL. El teléfono móvil receptor de la notificación debe mostrar el texto, en caso de que exista, y provee al usuario la opción de conectarse al URL con solo presionar una tecla[Ericsson, 2006] .

I.1.3. Servicios de valor agregado

El concepto de servicio de valor agregado para móviles es muy amplio. Todos aquellos servicios que vayan más allá del rol fundamental de comunicación por voz, pueden ser considerados servicios de valor agregado. Como no son parte del servicio básico de voz, pueden ser ofrecidos de forma separada a los usuarios finales, y los operadores pueden obtener mayores ganancias por este concepto. De forma general, pueden ser considerados como servicios de valor agregado, todos los servicios que incorporan

mensajes de texto y multimedia, descarga de contenido como tonos, imágenes y juegos, y otros[Sharma, 2007].

Los servicios de valor agregado para móviles ofrecen la posibilidad a los operadores de cobrar un precio diferente por cada mensaje de texto, comúnmente llamado SMS Premium. De forma general, los servicios de valor agregado añaden valor a servicios básicos que se estén ofreciendo, estimulan el incremento en el uso de los servicios básicos, pueden en ocasiones funcionar de forma autónoma y pueden ser ofrecidos a un precio extra[MobileIN, 2004].

El mundo de los servicios de valor agregado es amplio y heterogéneo, abarcando distintas necesidades y diversas porciones del mercado de telefonía móvil. En tal sentido, la gama de servicios es grande y puede estar orientada al entretenimiento o a otros temas más relevantes para la actividad laboral y cotidiana.

Entre todos estos, los servicios de valor agregado que se basan en mensajes de texto ocupan un lugar importante en el mercado. Los mensajes de texto fueron utilizados de forma tradicional hasta el instante que diferentes empresas desarrollaron los primeros servicios de valor agregado en este campo. A partir de este momento, los servicios basados en mensajes de texto se convirtieron en una industria de aplicaciones con diversos servicios que incluían noticias, votaciones, horóscopo, chat, reporte del tiempo, entre muchos otros. Con el surgimiento de los mensajes multimedia, los usuarios se beneficiaron de nuevas aplicaciones que combinan imágenes, sonidos, videos y texto, donde experimentaron una experiencia más agradable[Sharma, 2007].

I.2. Plataformas para la gestión de servicios de valor agregado

Numerosas empresas a nivel mundial cuentan con plataformas para la gestión de servicios de valor agregado destinados a dispositivos móviles. Cada una de estas plataformas tiene sus peculiaridades, tanto desde el punto de vista de sus funcionalidades como de sus arquitecturas. Seguidamente se describirán las funcionalidades principales y la arquitectura de algunas de estas plataformas, así como el modelo de negocio que siguen las empresas que se dedican al negocio de los servicios de valor agregado para dispositivos móviles.

I.2.1. Descripción de las principales funcionalidades

A continuación se describirán las principales funcionalidades ofrecidas por las plataformas de diversas empresas a nivel mundial. Entre estas se encuentran la norteamericana Google; Une Internet y empresaSMS.com de España; Vodafone y Esendex, de Gran Bretaña y Mercacel y Nuestro Site de México.

I.2.1.1. empresaSMS.com

Tal y como ellos mismos publican en su sitio web, empresaSMS.com es una compañía española de nuevas tecnologías especializada en la creación, desarrollo e implementación de nuevas alternativas y tecnologías para el envío de mensajes de texto a teléfonos móviles, cuyo objetivo es ofrecer una plataforma y soporte de comunicación por SMS simple, rápido, seguro, económico y efectivo al alcance de todo tipo de empresas y organizaciones públicas y privadas.

empresaSMS.com ofrece la posibilidad de enviar uno o más mensajes de texto a uno o más teléfonos móviles a través de una página que se encuentra en su sitio web y por un costo determinado por cada mensaje que se envíe. El acceso a la página de envío requiere que el usuario posea un identificador de usuario y una contraseña. En lo adelante, deberá indicar el país y los destinatarios del mensaje, el texto del mensaje y otros datos de personalización como si se van a utilizar caracteres especiales, o si se va a programar el envío del mensaje a una hora determinada; y podrá enviar el mensaje a su destino[EmpresaSMS, 2009].

La solución que da empresaSMS.com es sencilla y consiste en dar la posibilidad de que los usuarios subscritos al sitio puedan enviar mensajes de texto a móviles en varios países. Para poder enviar mensajes, cada usuario debe solicitar a través del sitio web de la empresa que se le cargue su crédito de SMS, en paquetes que pueden ir de 500 a 50 000 SMS, cada paquete por un precio determinado.

I.2.1.2. Google

Google es una de las más conocidas compañías en el mundo de la informática, fundada por Larry Page y Sergey Brin. Google nació en la residencia de estudiantes de la Universidad de Stanford y se expandió rápidamente a los buscadores de información de todo el mundo. Goza de prestigio como el motor de

búsqueda más importante del mundo: un servicio gratuito y fácil de utilizar que, por lo general, presenta resultados relevantes en una fracción de segundo. Google no solo se limita a brindar servicio de búsqueda. Actualmente brinda servicios de correo electrónico, mapas y localización, bitácora digital (*blog*), noticias, almacenamiento de fotos, red social y traducción de textos entre otros; además de contar con un paquete ofimático en línea, un explorador de Internet llamado Chrome, un sistema operativo basado en una distribución de GNU/Linux, y un sistema operativo para dispositivos móviles llamado Android, también basado en GNU/Linux.

Igualmente, Google cuenta con servicios para móviles. Este servicio puede ser probado desde una página en su sitio web, y también puede ser accedido a través de un número de teléfono. El servicio consiste en realizar consultas por medio de mensajes con un texto predeterminado a este número de teléfono, tras lo cual se recibe de vuelta un mensaje de texto en el móvil con el resultado de la consulta realizada. Los mensajes de texto que se envían deben tener una palabra clave, que es la primera palabra del mensaje de texto, mediante la cual se conoce qué tipo de consulta se está realizando. Entre las consultas que se pueden realizar están el parte del tiempo, glosario de palabras, deportes, películas, direcciones postales, vuelos, aeropuertos y líneas aéreas, calculadora y conversión de unidades entre otros[Google, 2009].

De forma general, el servicio de Google permite realizar distintos tipos de consultas por medio de mensajes de texto, basándose en la palabra clave del mensaje enviado por el usuario. El servicio está disponible lo mismo a través de un número de teléfono que desde una página web. Para obtener la información para estos servicios, Google aprovecha la enorme cantidad de información que posee en sus bases de datos.

I.2.1.3. Vodafone

Vodafone es una de las mayores compañías de redes de telecomunicaciones móviles del mundo, con intereses en redes móviles de todo el mundo y que cuenta con más de 147 millones de clientes alrededor del mundo.

Vodafone ofrece un servicio llamado Dicta SMS, que en esencia se encarga de convertir los mensajes de voz en mensajes de texto. Este servicio funciona de dos formas diferentes. La primera es mediante

activación. Todos los usuarios que tengan activado este servicio, si su teléfono está fuera de servicio o no contesta a la llamada, la persona que los llama les puede dejar un mensaje de voz que el usuario recibirá en forma de mensaje de texto. Con la segunda variante de este servicio, llamada SMS Fácil, no es necesario que la persona que llame o reciba la llamada tenga activado el servicio. Esta permite dejar un mensaje de voz que el interlocutor recibirá en forma de mensaje de texto, aunque el servicio no esté activado[Vodafone, 2009].

En general, este servicio de mensajería permite convertir mensajes de voz en mensajes de texto, sustituyendo de esta forma el contestador tradicional.

I.2.1.4. Mercacel

Mercacel es una empresa mexicana cuyo objetivo es ofrecer el servicio de mensajería para empresas, basándose fundamentalmente en servicios de mensajería de texto.

Mercacel cuenta con un servicio de mensajería que permite enviar mensajes de texto a cualquier móvil a través de una página web, además de una aplicación de escritorio para enviar mensajes. Esta aplicación permite enviar mensajes desde una computadora en cuestión de minutos y constituye una herramienta de gran utilidad para las empresas que necesitan comunicarse con empleados y clientes o con muchas personas. Esta aplicación permite además el envío de mensajes masivos, posee plantillas para mensajes estándar, informes de entrega en tiempo real, y la posibilidad de hacer encuestas a los clientes. Además Mercacel ofrecen un SDK SMS, un paquete que permite la integración de estos servicios de mensajería en las aplicaciones empresariales de sus clientes[Mercacel, 2009].

La solución de Mercacel difiere de las anteriores por su enfoque empresarial. Su propósito es permitir que distintas empresas utilicen sus servicios de mensajería desde las propias aplicaciones de las empresas. Para ello Mercacel ofrece una aplicación de escritorio con un conjunto de funcionalidades estándar, y ofrece un SDK para que otras aplicaciones puedan utilizar sus servicios.

I.2.1.5. Une Internet

Une Internet es otra empresa española que propone un sistema de envío de mensajes de texto masivos.

Los usuarios o empresas que deseen este servicio, deben pagar previamente uno de los paquetes ofertados, que pueden ser de 1000 a 50 000 mensajes de texto. Los envíos de mensajes se pueden realizar a través de una aplicación en línea, o bien mediante el programa que se conecta con la aplicación del cliente. La fecha y hora de los envíos de mensajes se pueden programar con antelación para asegurar una recepción óptima del mensaje, y la información sobre el estado de los envíos esta permanentemente disponible en tiempo real gracias al sistema que se pone a disposición del cliente. El servicio de Une Internet es multi-operador, es decir, permite conectarse con distintos proveedores como son Movistar, Amena y Vodafone[UneInternet, 2009] .

La solución de Une Internet es similar a la anterior: da la posibilidad de envíos de mensajes de texto a través de una página web, y también a través de una aplicación de escritorio que se instala localmente. Como característica distintiva, esta aplicación permite conocer el estado de los mensajes enviados por los clientes. Además permite conectarse con distintos proveedores de servicios.

I.2.1.6. Esendex

Esendex es una empresa británica que ofrece servicios de mensajería de texto a empresas en Gran Bretaña y otros países europeos. En la actualidad tienen como clientes a más de 7 000 compañías, tanto pequeñas y medianas empresas como grandes compañías, a nivel mundial.

Su gama de servicios para empresas comprende diferentes herramientas que permiten enviar y recibir mensajes de texto por vía web y desde la cuenta de correo electrónico, además de una API que permite la integración de sus servicios de mensajería con sitios web o aplicaciones de terceros. Permiten además el envío de mensajes de texto que serán recibidos como mensajes de voz[Esendex, 2009].

La solución de Esendex añade a las características vistas anteriormente, el envío de mensajes de texto desde la cuenta de correo electrónico.

I.2.1.7. Nuestro Site

Nuestro Site es una empresa mexicana dedicada al desarrollo de software para terceras empresas. Actualmente atiendo a clientes de México en el sector privado, financiero, educativo y de gobierno, y

brinda soluciones de infraestructura tecnológica. Entre sus productos se encuentra una plataforma para el envío de SMS y MMS.

Esta plataforma brinda una herramienta de gestión y entrega de SMS, la cual permite integrar de una manera sencilla a distintas empresas al mercado de los dispositivos móviles. La plataforma de envío de mensajes consiste de tres módulos principales: conector, enrutador y administrador de contenido.

- El conector se encarga de establecer las comunicaciones con los centros de mensajes, mediante diferentes protocolos como SMPP 3.4, Web Services, HTTP y otros. Permite además gestionar la conexión con diferentes operadores de manera simultánea.
- El enrutador es el encargado de administrar las marcaciones y las aplicaciones, con el fin de poder gestionar mejor los servicios con una o más marcaciones. Una marcación puede administrar diversos tipos de servicios como contenido, servicios en demanda, encuestas y otros.
- El administrador de contenidos gestiona la entrega del mismo, el proceso de las estadísticas y reportes.

La gestión de las estadísticas permite conocer qué medio o publicidad es la que mejor está funcionando, por lo que la plataforma maneja canales de venta con el fin de poder administrar las marcaciones y los canales de venta por donde están cursando tráfico.

La plataforma permite gestionar diferentes palabras clave, de tal forma que cada servicio tenga una o más de una palabra clave. Permite también la gestión de clientes y campañas. Para cada cliente se genera un prefijo, con el fin de distinguir su tráfico así como generar reportes a través de los cuales se pueda consultar solo sus servicios[NuestroSite, 2009] .

En resumen, esta plataforma permite la gestión de diferentes servicios basados en mensajes de texto, se puede conectar con varios operadores, se encarga de enrutar los pedidos de los usuarios a las aplicaciones. Además permite el registro de clientes y ofrece un módulo de estadísticas y reportes. Introduce, de forma adicional, el manejo de contenido para móviles como pueden ser juegos, tonos, animaciones, fondos de pantalla y otros.

I.2.1.8. Teleios Systems

Teleios Systems es una empresa fundada en 1997 que desarrolla y comercializa soluciones informáticas enfocadas en la telefonía inalámbrica y las redes, para un amplio rango de clientes que incluye instituciones gubernamentales, corporaciones y operadores telefónicos. Uno de sus productos es una plataforma para la gestión de mensajes, llamada Teleios MessageCentral.

Teleios MessageCentral es una plataforma de entrega de servicios de mensajería móvil que permite a los operadores de telefonía móvil múltiples servicios y aplicaciones de mensajería de valor agregado, al tiempo que mantiene control sobre los accesos de red y el tráfico.

Entre las características de esta plataforma se encuentra el soporte para múltiples protocolos, herramientas de administración que brindan información sobre las cuentas de mensajería, los privilegios de acceso, tamaños de los mensajes y tiempo de entrega; permite la administración de las aplicaciones a través de herramientas de configuración, gestión del contenido, reportes de tráfico, integración y rastreo de facturas, e inteligencia empresarial, entre otras[Teleios, 2009].

En específico, las herramientas de inteligencia empresarial permiten:

- Capturar la información del subscriptor para conocer el contenido y los servicios que más se utilizan.
- Generar estadísticas para identificar horas de mayor consumo y generar reportes por hora y diarios.
- Toma de decisiones basadas en las tendencias del mercado, realizando perfiles demográficos de los usuarios.
- Ayudar a los objetivos estratégicos, ya que los reportes de tráfico pueden resultar útiles para implementar actividades buscando aumentar el tráfico de mensajes.

I.2.1.9. Resumen de las principales funcionalidades

Tras estudiar las distintas soluciones que brindan las empresas dedicadas a los servicios de mensajería de texto a nivel mundial, se puede enumerar un grupo de funcionalidades que podrían formar parte de

una solución al problema de desarrollar una plataforma para la gestión de mensajes de texto. Estas son:

- Comunicación con uno o varios operadores móviles a través de distintos protocolos.
- Envío de mensajes a uno o más teléfonos a través de una interfaz web. Este envío puede ser gratuito o por medio de tarifas que se cobran a los clientes de acuerdo a la cantidad de mensajes que pueden enviar.
- Envío de mensajes a uno o más teléfonos a través de una aplicación de escritorio.
- Envío de mensajes de texto desde el correo electrónico.
- Soporte para distintos tipos de servicios basados en mensajes de texto, teniendo en cuenta en la palabra clave del mensaje. Estos servicios pueden dar distintos tipos de información a los usuarios.
- Conversión de mensajes de voz en mensajes de texto y viceversa.
- Generación de estadísticas y reportes, así como herramientas de inteligencia empresarial.
- Gestión de clientes que deseen ofrecer tanto servicios como contenido.
- SDK o API de funciones que permitan a aplicaciones de terceros el envío de mensajes de texto utilizando las facilidades de la plataforma.
- Extensión de los servicios de mensajes de texto a otros servicios de descarga de contenido para móviles.

A partir de estas funcionalidades, se identificaron las más importantes para el desarrollo de una nueva plataforma. Estas son la comunicación con varios operadores por medio de distintos protocolos, el soporte para varios servicios al mismo tiempo, la implementación de un SDK o API para el desarrollo de nuevos servicios y la generación de estadísticas y reportes. El resto de las funcionalidades se identificaron como secundarias, ya que se pueden implementar a partir de servicios específicos.

I.2.2. Arquitectura de las plataformas estudiadas

A continuación se describirá la arquitectura interna de alguna de las plataformas estudiadas. Para ello

se tomarán los casos de la plataforma Kannel, desarrollada por Wapit Ltd, y la plataforma Ozeki, desarrollada por Ozeki Informatics.

I.2.2.1. Kannel

Kannel es una pasarela para WAP y SMS, desarrollada en lenguaje C por Wapit Ltd y publicada bajo una licencia de código abierto. Kannel es capaz de dar soporte tanto a mensajes de texto como a notificaciones WAP. Además está preparado para ejecutarse en varios servidores de forma tal que si alguno de ellos falla, no se afecte el servicio.

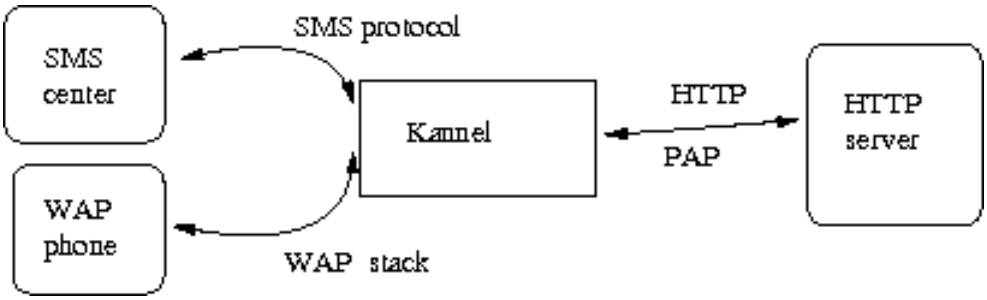

Figura 2: Interfaces de Kannel con el exterior.

Kannel brinda interfaces para conectarse con tres tipos de componentes externos, como se muestra en la Figura 2. Estas interfaces son las siguientes[Wirzenius, 2004]:

- SMSC, utilizando varios protocolos.
- Servidores HTTP, para obtener contenido WAP y SMS de aplicaciones de terceros, y para enviar contenido WAP.
- Teléfonos móviles que implementen el protocolo WAP e incluyan clientes de notificaciones WAP.

Los protocolos para la comunicación con los SMSC implementan una interfaz común, por lo que es posible agregar nuevos protocolos a Kannel, aunque actualmente solo soporta el protocolo SMPP. La comunicación con las aplicaciones se realiza por medio de los protocolos HTTP, y PAP para las notificaciones WAP.

Las tareas relacionadas con estas interfaces son divididas en tres tipos de procesos distintos, a los

cuales Kannel llama *smsbox*, *wapbox* y *bearerbox*, como se muestra en la Figura 3. El *bearerbox* se encarga de la comunicación con los SMSC. Kannel utiliza la implementación de los protocolos de SMS sólo para el envío de notificaciones WAP a través de SMS. Kannel puede usar la conexión al SMSC tanto para el envío y recepción de SMS de texto, como para utilizar los SMS como vía para el envío de notificaciones WAP. El *smsbox* se encarga de recibir los SMS del *bearerbox*, interpretarlos como pedidos de servicios y dar una respuesta apropiada. El *wapbox* implementa el protocolo WAP y las notificaciones WAP. Cuando el *wapbox* se utiliza para enviar notificaciones, se conoce como un *Push Proxy Gateway* [Wirzenius, 2004].

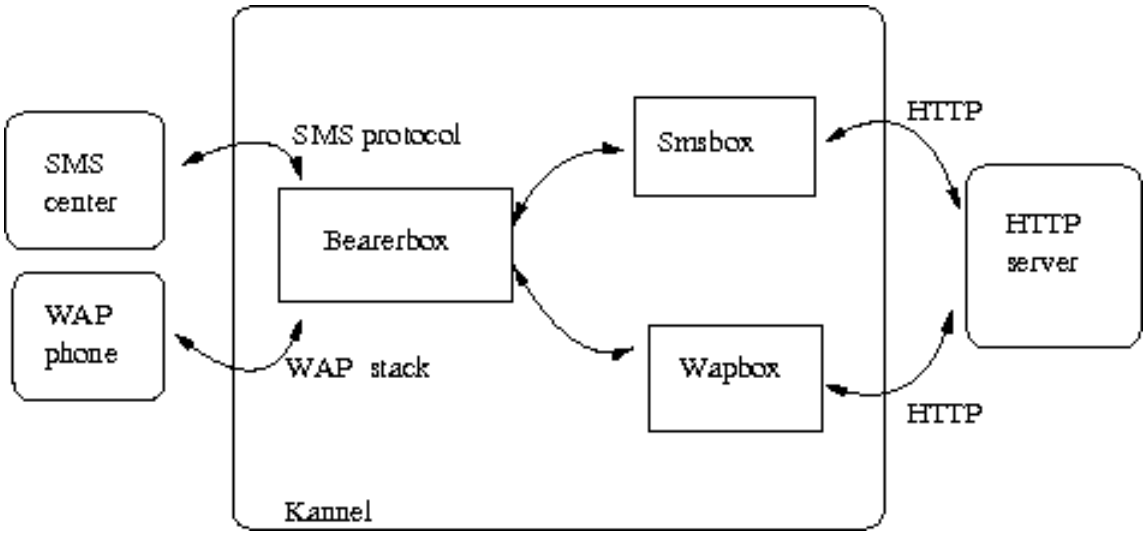

Figura 3: Principales procesos de Kannel.

Solo existe una única instancia del *bearerbox,* ya que debe exponer una única dirección IP al SMSC, en tanto que pueden existir múltiples instancias de *wapboxes* y *smsboxes*, implementadas en forma de hilos. El *bearerbox* además realiza un balanceo de carga entre las distintas instancias de los *wapboxes* y *smsboxes*[Wirzenius, 2004].

La solución de Kannel consiste en ofrecer una pasarela entre los SMSC o teléfonos con soporte para navegación WAP por un lado, y cualquier aplicación que desee utilizar los servicios de mensajería, tanto SMS como WAP, por el otro lado; por medio de la interfaz HTTP que ofrece Kannel.

Aunque Kannel parece una solución aceptable, en la práctica agregar nuevos protocolos de comunicación en cualquiera de las interfaces, tanto la interfaz al SMSC o a las aplicaciones; o desarrollar distintas funcionalidades que respondan a necesidades específicas, como puede ser un SDK

para aplicaciones, o la gestión de trazas para realizar reportes, o adaptar el protocolo SMPP a la implementación específica del SMSC, implicaría realizar un estudio a fondo de la plataforma y realizar cambios dentro de su código para adaptarlo a las nuevas necesidades.

I.2.2.2. Ozeki

El servidor de mensajes Ozeki es una pasarela SMS que permite enviar y recibir mensajes SMS desde una computadora hacia teléfonos móviles a través de una red GSM. Para esto puede utilizar un módem GSM conectado a la computadora, o puede utilizar una conexión IP para enviar mensajes directamente a un SMSC usando protocolos como SMPP o UCP, como se muestra en la Figura 4.

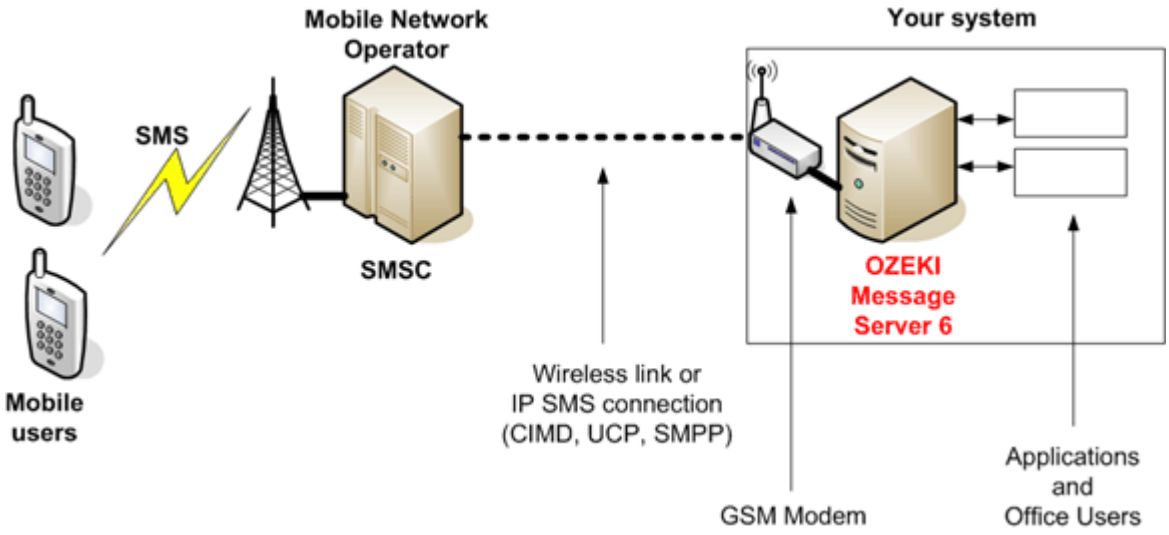

Figura 4: Arquitectura de Ozeki.

Ozeki se instala como un servicio sobre sistemas operativos Windows 2000, XP y 2003. Es capaz de gestionar grandes volúmenes de tráfico, y puede ser configurado para enviar y recibir mensajes a través de distintos dispositivos, soporte para múltiples usuarios, balanceo de cargas y gestión de las colas de mensajes [Ozeki, 2006].

Ozeki puede ser utilizado para enviar y recibir mensajes desde las herramientas de Microsoft Office, utilizando una macro diseñada para ello; o desde aplicaciones escritas en diferentes lenguajes de programación.

Puede ser utilizado por distintos usuarios al mismo tiempo, cada cual conectado con su nombre de

usuario y contraseña, y con una cola de mensajes para cada uno de ellos. Para procesar las colas de mensajes de los distintos usuarios, Ozeki utiliza un algoritmo *round robin,* el cual permite que ningún usuario tenga mayor prioridad que otros.

Ozeki ofrece dos interfaces para interactuar con las aplicaciones. La primera es a través de dos tablas que crea por cada usuario en un sistema SQL Server. Una tabla es para mensajes enviados y otra para mensajes recibidos. Si la aplicación desea enviar un mensaje basta con insertarlo en la tabla de mensajes enviados. De igual forma, al consultar la tabla de mensajes recibidos puede saber si han entrado nuevos mensajes. La segunda es una interfaz HTTP desde la cual se pueden enviar mensajes, y que puede ser utilizada para el envío de notificaciones.

Ozeki permite definir el enrutamiento de los mensajes desde una interfaz de escritorio. Los mensajes de salida son enrutados a un dispositivo o conexión determinada, en tanto los mensajes de entrada son enrutados a un usuario determinado. Para el enrutamiento se pueden tener en cuenta parámetros como el proveedor, el remitente, el destinatario, el usuario o la primera palabra del mensaje o palabra clave [Ozeki, 2006].

La solución ofrecida por Ozeki permite la conexión con centros de mensajería o módems GSM para el envío y recepción de mensajes por una parte, en tanto que por la otra ofrece una interfaz en forma de tablas en un servidor de datos SQL Server, las cuales pueden ser consultadas por cualquier aplicación desarrollada sobre cualquier lenguaje o plataforma de programación. No obstante, la adaptación de Ozeki a necesidades específicas sería mucho más complicada, ya que se trata de una aplicación comercial, cuyo código fuente no es ofrecido a los usuarios y que requiere de un pago por su licencia.

I.2.3. Modelo del negocio

El servicio básico de envío de mensajes involucra a los usuarios de dispositivos móviles, y al operador. El operador es una empresa telefónica que provee servicios de telefonía para los clientes de teléfonos móviles. El operador puede instalar un SMSC, que es un elemento de la red de telefonía móvil que cumple la función de enviar y recibir mensajes, de forma tal que sus usuarios tienen la posibilidad de enviar y recibir mensajes[TT, 2007] .

Así, en el momento que un usuario envía un mensaje de texto a otro usuario, su teléfono envía dicho

mensaje al SMSC correspondiente al operador del usuario remitente. El SMSC guarda el mensaje y lo entrega a su destinatario cuando este se encuentre en cobertura. Por lo general el SMSC dispone de un tiempo máximo durante el cual el mensaje es guardado, y si en ese tiempo el destinatario no es localizado el mensaje es desestimado.

En el caso de los servicios de valor agregado, además de los usuarios y operadores, existe otro actor llamado Entidad Externa de Mensajería (*External Short Messaging Entity*, o ESME). ESME es un término que describe a una aplicación externa que se conecta a un SMSC con el propósito de enviar y recibir mensajes. Como ejemplos típicos de ESME están los sistemas que envían de forma automática mensajes de propaganda a los usuarios, los sistemas de votación y muchos más. Básicamente, cada vez que un usuario envía o recibe mensajes donde del otro lado no exista un usuario real, se está en presencia de un ESME[Aldiscon, 1996].

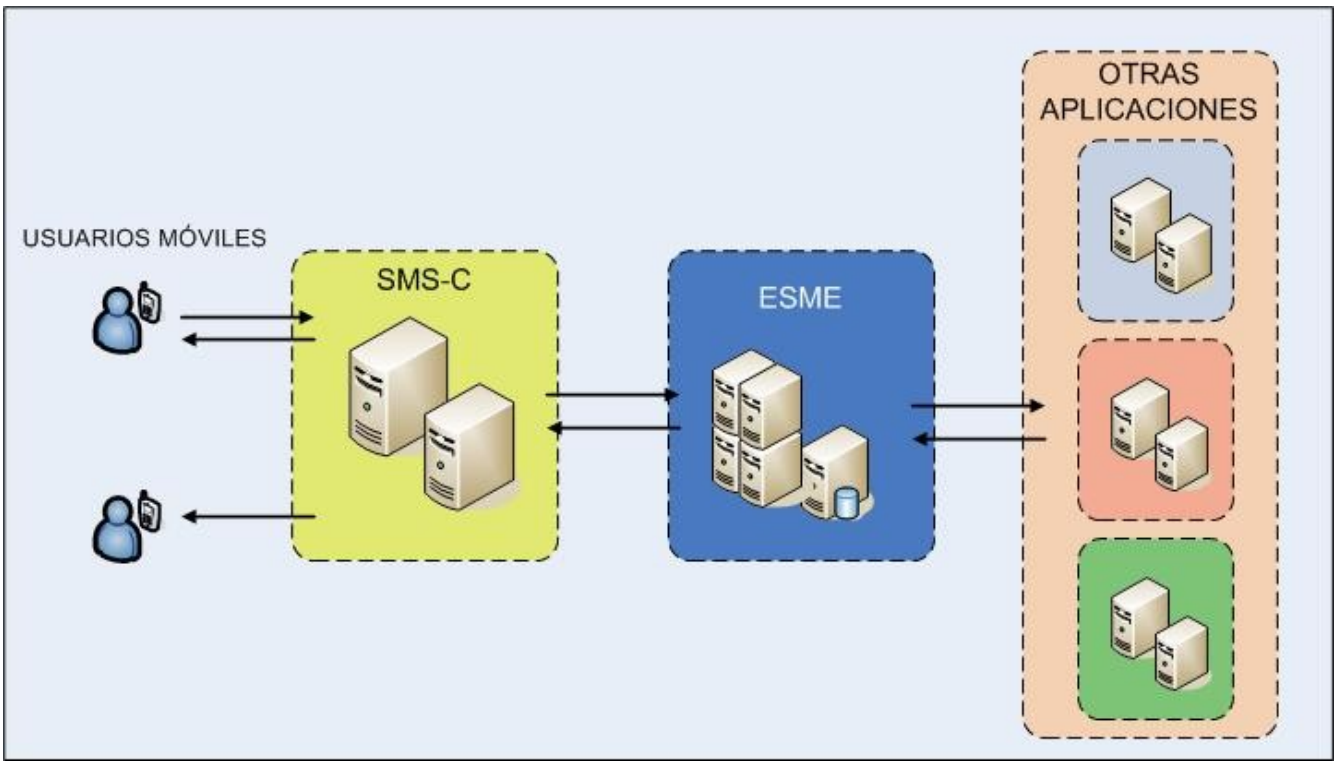

Figura 5: Modelo de negocio.

Por tanto, una plataforma para la gestión de mensajes no es más que un ESME que se dedica a enrutar los mensajes que recibe a otras aplicaciones (que pueden ser consideradas también como parte del ESME) tal y como se muestra en la Figura 5. Estas aplicaciones pueden procesar mensajes, realizar

acciones y dar una respuesta al usuario; o redirigir los mensajes a otras aplicaciones, de tal manera que se forma una cadena hasta llegar a alguna aplicación final que de la respuesta necesaria al usuario. A su vez, la plataforma es también la encargada de recibir las respuestas que son enviadas por las aplicaciones, y de enrutar estas respuestas a través del SMSC hacia el usuario o los usuarios finales.

La plataforma puede ser propiedad del operador móvil, y por tanto estar conectada a un único SMSC; o puede ser propiedad de terceros y tener la capacidad de conectarse a distintos SMSC, utilizando para ello distintos protocolos, como pueden ser SMPP, IPX y otros.

A su vez, la plataforma puede exportar una API o proporcionar un medio para que aplicaciones de terceros puedan implementar sus propios servicios de envío y recepción de mensajes. De esta forma la plataforma serviría como intermediaria entre uno o más centros de mensajería y uno o más clientes que implementarían sus servicios de valor agregado.

Así, el modelo inicial según el cual un usuario móvil envía un mensaje de texto a otro usuario móvil, se transforma en un modelo en el cual un usuario móvil puede solicitar distintos tipos de servicios haciendo uso de la mensajería móvil, los cuales están implementados en un ESME que se comunica directamente con el SMSC, o en una aplicación de terceros que se comunica con el ESME.

I.3. Factores que inciden en la calidad del software

El objetivo primordial de la ingeniería de software es desarrollar un software de calidad. La calidad del software mide cuán bien el software ha sido diseñado, y cuán bien la implementación se ajusta al diseño previo[Pressman, 2005].

La calidad del software se puede definir como la conjugación de diversos factores, tanto externos como internos. Los factores externos, como pueden ser velocidad de respuesta, o facilidad de uso del software, son factores fácilmente identificables por el usuario final. Sin embargo, otras características tales como si el software es modular, o el código legible, son factores internos, que solo pueden ser detectados por los desarrolladores que tienen que interactuar con el código del software. Para el usuario final, solamente cuentan los factores externos. Pero la clave para lograr estos factores externos está en los factores internos: para que los usuarios puedan disfrutar de las cualidades visibles, los

desarrolladores tienen que haber aplicado técnicas internas que aseguren las cualidades no visibles.

En su libro "Object-Oriented Software Construction", Bertrand Meyer expone un grupo de factores internos que inciden en la calidad del software. A continuación se expondrán algunos de los que se han tenido en cuenta para el desarrollo de la plataforma de gestión de mensajes para móviles.

- La corrección es la habilidad de un software de ejecutar las tareas que se supone que debe realizar, tal y como están descritas en las especificaciones. La corrección es una cualidad primaria. Si el sistema no hace lo que se supone que debe hacer, lo demás carece de sentido. Por supuesto, lograr la corrección del software no es un proceso sencillo, pues para ello se necesita primeramente especificar los requerimientos de una forma precisa, lo cual es de por sí una tarea difícil.

- La robustez es la propiedad del software de manejar apropiadamente situaciones anormales. La robustez complementa la corrección, ya que esta última tiene en cuenta el comportamiento del software en los casos que han sido cubiertos por la especificación, en tanto la robustez caracteriza lo que ocurre fuera de la especificación.

- La extensibilidad es la capacidad del software de adaptarse a los cambios en la especificación. En ocasiones no se tienen en cuenta los cambios al realizar el diseño de una aplicación, y parece que los requerimientos iniciales nunca van a cambiar a lo largo del proceso de desarrollo. Sin embargo, hay que tener en cuenta el cambio como un factor esencial en el desarrollo de software; tanto el cambio de requerimientos, de algoritmos, de representación de datos o de técnicas de implementación. Existen dos principios fundamentales a tener en cuenta para lograr la extensibilidad: la simplicidad en el diseño, ya que una arquitectura más simple siempre será más fácil de cambiar que una arquitectura compleja; y la descentralización, ya que cuanto más autónomos sean los módulos, menos repercusión tendrán los cambios sobre el sistema en su conjunto.

- La reusabilidad es la habilidad de los componentes del software de servir para la construcción de otras aplicaciones. La necesidad de la reusabilidad viene de la observación de que muchos sistemas siguen patrones similares, por lo cual debería ser posible utilizar esto para evitar

reinventar las mismas soluciones a problemas que ya han sido resueltos antes. Por medio de estos patrones, un elemento de software reusable puede ser aplicado en múltiples sistemas.

- La portabilidad es la capacidad de que el software pueda ser utilizado sobre distintos sistemas operativos y hardware. La portabilidad tiene en cuanta las variaciones tanto en el hardware como en el software (sistema operativo, bibliotecas y otros)[Meyer, 1997].

Todos estos factores son de gran importancia, aunque entre estos destacan cuatro. Los dos primeros son la corrección y la robustez. Es difícil producir software sin que aparezcan defectos (*bugs*), y siempre resulta difícil corregir estos defectos. Las técnicas para mejorar la corrección y robustez son las mismas: mayor formalidad en las especificaciones, chequeos a lo largo de todo el proceso de construcción del software y no solo realizar pruebas después de terminado, mejores mecanismos por parte de los lenguajes para el chequeo estático de tipos y aserciones, manejo automático de memoria y una gestión disciplinada de las excepciones; así como el uso de herramientas para detectar inconsistencias antes de que se conviertan en defectos. Debido a la similitud de estos dos factores, se puede utilizar el término fiabilidad para englobarlos.

Los otros dos factores fundamentales son la extensibilidad y la reusabilidad. El software debe ser fácil de cambiar al tiempo que cambian las especificaciones. Por tanto los elementos de software deben ser lo más generales posible, y se debería tener a disposición la mayor cantidad de componentes de propósito general posible a la hora de construir un software. Aquí también aparecen ideas similares para mejorar ambos factores: usar arquitecturas descentralizadas, en la cual los componentes sean autocontenidos y se comuniquen a través de interfaces restringidas y claramente definidas. El término modularidad se puede utilizar para reunir ambos factores[Meyer, 1997].

La modularidad en el software deja abierta la posibilidad de enriquecer el sistema a medida que sea necesario, agregándole nuevas funciones e integrándolo con otras plataformas. Un diseño modular efectivo reduce la complejidad del sistema, facilita los cambios y produce como resultado una implementación más sencilla, y permite el desarrollo en paralelo de diferentes partes del sistema[Meyer, 1997].

En este sentido, la arquitectura modular se contrapone a las arquitecturas monolíticas, en donde el

software se encuentra muy acoplado. Esto, si bien es bueno porque evita el riesgo de necesitar alguna funcionalidad que puede no estar disponible en cierto momento, por otra parte aumenta la complejidad del sistema, hace engorroso el proceso de detección de errores y de cambios, ya que al estar todos los módulos fuertemente conectados, un cambio en uno podría repercutir en el sistema en su conjunto.

I.3.1. Pruebas de software

Las pruebas de software permiten investigar la calidad de un producto o servicio, con respecto al contexto en el cual este va a operar. Constituyen además una forma objetiva de apreciar y entender los riesgos en la implementación del software. Las técnicas de pruebas incluyen, aunque no solo se limitan a, los procesos de ejecutar un programa o aplicación con el objetivo de encontrar errores en el software. También puede ser visto como un proceso para validar que el software cumple con los requerimientos técnicos que guiaron su diseño e implementación y que funciona de la forma esperada.

En dependencia de los modelos te prueba adoptados, las pruebas pueden ser implementadas en cualquier momento del proceso de desarrollo, aunque la mayor cantidad de pruebas se realizan por lo general después que han sido definidos los requerimientos y que el proceso de implementación ha terminado.

Aunque el proceso de prueba no puede identificar completamente todos los defectos del software, si puede realizar una comparación del estado y del comportamiento del producto contra un conjunto de principios o mecanismos mediante los cuales se pueden reconocer los problemas. Estos mecanismos pueden ser las especificaciones, diseño por contratos, otros productos que puedan servir como comparación, versiones anteriores del mismo producto, inferencias que se realizan a partir del propósito para el cual se realizó el software, expectativas de los clientes, estándares, u otros criterios[LCOMF, 2007].

I.3.1.1. Pruebas de unidad

Las pruebas de unidad constituyen métodos de verificación y validación en los cuales el programador realiza pruebas sobre unidades individuales dentro del código fuente. Una unidad es considerada la parte más pequeña que puede ser objeto de prueba dentro de la aplicación. En la programación basada

en procedimientos, una unidad puede ser un programa, una función o un procedimiento. En la programación orientada a objetos, la unidad más pequeña es la clase.

Cada caso de prueba debe ser independiente de los demás. Las pruebas de unidad son por lo general escritas y ejecutadas por los propios programadores, para asegurarse que el código cumple con el diseño y se comporta de la forma esperada.

El objetivo fundamental de las pruebas de unidad es aislar cada parte del programa y probar que cada una de las partes individuales funciona de forma correcta. De esta forma se pueden detectar errores en etapas tempranas del desarrollo, y asegurarse que en lo adelante el módulo probado continúe funcionando correctamente[Beck, 2003].

Algunas metodologías de desarrollo ágil, como es el caso de Programación Extrema (*Extreme Programming*, o XP) hacen uso extensivo de las pruebas de unidad. XP usa una metodología de desarrollo basado en pruebas de unidad. Los desarrolladores escriben sus pruebas de unidad que exponen lo mismo un requerimiento del software o un defecto. Las pruebas en un inicio deben fallar debido a que el requerimiento aun no ha sido implementado, o porque intencionalmente expone un defecto que existe en el código fuente[Wells, 1999] .

I.3.1.2. Pruebas de recuperación

Las pruebas de recuperación comprueban de qué forma una aplicación es capaz de recuperarse de los errores, fallas de hardware y otros problemas similares. Consisten en forzar las fallas, de forma tal que se pueda comprobar que el mecanismo de recuperación del software funciona de forma correcta[Lewis, 2000].

Entre las pruebas de recuperación se pueden enumerar las siguientes:

- Mientras la aplicación se encuentra ejecutándose, reiniciar la computadora de forma y más adelante chequear la integridad de los datos.
- Mientas la aplicación está recibiendo datos por la red, desconectar el cable de red. Después de un tiempo determinado, se vuelve a conectar y se analiza la habilidad de la aplicación para continuar la comunicación a través de la red.

I.3.1.3. Pruebas de estrés

Las pruebas de estrés son una forma de probar la estabilidad de un sistema en condiciones extremas, que van más allá de la capacidad operacional normal del sistema, con el objetivo de observar el comportamiento del sistema en estas condiciones.

Las pruebas de estrés ponen énfasis en la robustez, disponibilidad y la recuperación ante errores del software en condiciones de incremento de cargas, en lugar de lo que hubiera sido considerado un comportamiento correcto en condiciones normales. En particular, los objetivos de este tipo de test son asegurar que el software continúa funcionando en condiciones extremas de insuficiencia de recursos, como memoria, espacio en disco, alta concurrencia, y otras[Lewis, 2000].

I.3.1.4. Pruebas de integración

Las pruebas de integración son aquellas en las cuales los distintos módulos que conforman el software se combinan y son probados en grupos. Las pruebas de integración se realizan después de las pruebas de unidad y antes de las pruebas de sistema. Las pruebas de integración toman todos los módulos que han sido sometidos a pruebas de unidad, los agrupa en componentes más grandes, y somete estos componentes a pruebas. En este tipo de pruebas se verifican aspectos tales como el comportamiento de áreas de datos compartidos y comunicación entre procesos entre otros.

Las pruebas de integración se pueden realizar teniendo en cuenta el sistema completo, o integrando las partes que componen el sistema de forma ascendente (*bottom-up*) o descendente (*top-down*)[Lewis, 2000].

I.3.1.5. Pruebas de sistema

Las pruebas de sistema se realizan teniendo en cuenta un sistema completo, para evaluar si el sistema cumple con los requerimientos especificados. Las pruebas de sistema son pruebas de caja negra, y por tanto no requieren un conocimiento específico del diseño interno o la lógica del código de la aplicación. Como regla, para las pruebas de sistema se toman los componentes que han pasado las pruebas de integración.

Este tipo de pruebas se realizan desde una perspectiva externa del sistema, desde la cual se construyen los casos de prueba. Se seleccionan entradas de datos válidas en inválidas y se determina si la salida del sistema es la esperada para cada uno de los casos[Lewis, 2000].

I.4. Conclusiones del capítulo

En este capítulo se introdujeron conceptos referentes a la telefonía móvil. Esta consiste en la transmisión asistida de señales a lo largo de una distancia determinada, con el propósito de comunicar desde y hacia un dispositivo móvil o usuario. Actualmente es capaz de ofrecer otro tipo de servicios como el de acceso a datos y envío de mensajes. Existen tres tipos de mensajes: los mensajes de texto, los mensajes multimedia, y las notificaciones WAP. Los servicios de valor agregado se basan en estos servicios básicos y les agregan funcionalidades extra, de forma tal que añaden valor a servicios básicos de mensajería.

Numerosas compañías a nivel mundial poseen plataformas para la gestión de servicios basados en mensajes de texto, varias de las cuales fueron objeto de estudio para determinar las funcionalidades que pueden presentar las plataformas de este tipo. Las principales funcionalidades identificadas fueron la comunicación con varios operadores por medio de distintos protocolos, el soporte para varios servicios al mismo tiempo, la implementación de un SDK o API para el desarrollo de nuevos servicios y la generación de estadísticas y reportes. El resto de las funcionalidades se identificaron como secundarias, ya que se pueden implementar a partir de servicios específicos.

Se analizó además la arquitectura de algunas plataformas, las cuales de forma general permiten interconectar a múltiples operadores móviles por una parte, con múltiples aplicaciones, implementando en ocasiones mecanismos de enrutamiento de los mensajes entre operadores y aplicaciones. La interfaz que se ofrece a las aplicaciones varía de acuerdo a la plataforma: puede ser en forma de tablas en servidores de bases de datos, o como interfaces HTTP, o también en forma de API o SDK.

En la segunda parte del capítulo, se trataron algunos factores que inciden en la calidad del software, como son la corrección, la robustez, la extensibilidad, la portabilidad, la reusabilidad y la modularidad; así como en la importancia de las pruebas de software.

II. TECNOLOGÍA JAVA Y PATRONES DE DISEÑO

En la primera parte del presente capítulo se tratará el tema de lo patrones de diseño, haciendo énfasis en los patrones fundamentales utilizados en la construcción de la plataforma para el manejo de servicios basados en mensajería móvil, entre los cuales se encuentran los Patrones Generales de Asignación de Responsabilidades, los patrones de la Banda de los Cuatro, y los patrones J2EE. Más adelante se expondrán las tecnologías utilizadas para la implementación de la plataforma. Estas tecnologías están basadas en la plataforma Java, específicamente en la plataforma Java Platform Enterprise Edition (Java EE), la cual se utiliza para la implementación y el desarrollo de aplicaciones empresariales con una arquitectura n-capas distribuida.

II.1. Patrones de diseño

Los patrones de diseño constituyen soluciones reusables a problemas que ocurren comúnmente en el diseño del software. Un patrón de diseño no constituye un diseño terminado que puede ser transformado directamente en código, sino es una descripción de cómo se puede resolver un problema, y puede ser utilizado en diferentes situaciones. Los patrones orientados a objeto típicamente muestran las relaciones e interacciones entre clases u objetos, sin llegar a especificar las clases u objetos finales que se encuentran involucrados.

II.1.1. Patrones utilizados en el diseño de la plataforma

Los patrones de diseño, popularizados tras la publicación del libro “Design Patterns: Elements of Reusable Object-Oriented Software” por parte de la Banda de los Cuatro (*Gang of Four*, o GoF), se dividieron originalmente en las categorías de patrones de creación, patrones estructurales y patrones de comportamiento.

Los patrones de creación manejan los mecanismos de creación de los objetos, intentando que la creación de los objetos ocurra de una forma adecuada de acuerdo a la situación específica, ya que un mecanismo incorrecto de creación de objetos podría redundar en problemas de diseño y mayor

complejidad. Por su parte, los patrones estructurales facilitan el diseño al identificar la forma más sencilla de realizar las relaciones entre las distintas entidades. Finalmente los patrones de comportamiento identifican patrones comunes en la comunicación entre los objetos, y de esta forma incrementan la flexibilidad a la hora de llevar a cabo esta comunicación[GoF, 1995].

Por su parte Craig Larman en "Applying UML and Patterns" definió un conjunto de patrones de diseño conocidos como Patrones Generales para la Asignación de Responsabilidades (*General Responsibility Assignment Software Patterns*, o GRASP). Los patrones definidos por GRASP tienen como objetivo asignar correctamente las responsabilidades a cada una de las clases que intervienen en el modelo.

Según GRASP, uno de los principales objetivos de los patrones de diseño, es la protección frente al cambio, es decir, dado que los cambios son inevitables, en necesario realizar un diseño que pueda ser cambiado de forma fácil. Uno de los principios para la protección frente al cambio es mantener bajo el acoplamiento entre clases y la alta cohesión, es decir, minimizar las dependencias que tiene una clase con otras, ya que, cuanto menor sea el acoplamiento entre las clases, menor influencia tendrán los cambios; y a su vez asignar responsabilidades a las clases de manera que todos sus métodos tengan un comportamiento bien definido[Larman, 2001].

Por otra parte, existen los patrones de diseño descritos por Sun Microsystems en "Core J2EE Patterns", un catálogo de patrones que se pueden utilizar en el contexto del diseño de aplicaciones empresariales sobre la plataforma Java EE[ACM, 2003].

Otras clasificaciones han introducido la noción de patrones de arquitectura, que pueden ser aplicados al nivel de la arquitectura del sistema. Tal es el caso de la arquitectura multicapas, o el modelo-vista-controlador (MVC).

A continuación se exponen algunos de los patrones de diseño utilizados en el diseño de la plataforma BE.

- El patrón Bajo Acoplamiento es un patrón GRASP que define que cada clase está acoplada o relacionada a las clases estrictamente necesarias, garantizando que, de producirse cambios en una clase, estos tengan un bajo impacto en las clases que se relacionan con esta.

- El patrón de Alta Cohesión es un patrón GRASP que permite asignar responsabilidades a las clases de manera que todos sus métodos tengan un comportamiento bien definido.

- El patrón Experto es un patrón GRASP que constituye el principio básico de asignación de responsabilidades. Este patrón indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo.

- El patrón Creador es un patrón GRASP que ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases.

- El patrón Polimorfismo es un patrón GRASP que expresa que siempre que se tenga que llevar a cabo una responsabilidad que dependa del tipo, se tiene que hacer uso del polimorfismo; cuando las alternativas o comportamientos relacionados varían según el tipo, se debe asignar la responsabilidad para el comportamiento, utilizando operaciones polimórficas, a los tipos para los cuales varía el comportamiento.

- El patrón Localización de Servicios (*Service Locator*) es un patrón J2EE que se utiliza para centralizar la búsqueda de objetos distribuidos, por medio de un punto de control centralizado que actúa como caché para eliminar búsquedas redundantes. El uso de un contenedor como caché permite mantener una instancia de los objetos buscados, con lo cual las aplicaciones sólo tienen que buscar un objeto en una sola ocasión, y así mejora el rendimiento de las aplicaciones. Este patrón se utiliza en combinación con el patrón Singleton, para asegurarse que sólo existe una única instancia de cada objeto.

- El patrón Objeto de Transferencia *(Transfer Object* o *Value Object)* es un patrón J2EE que es utilizado para transportar datos, agrupar atributos relacionados entre sí formando un valor compuesto y funcionando como objeto de retorno para métodos remotos. También es conocido como Value Object. En el entorno de Java EE, es utilizado en los llamados remotos entre la capa de presentación y la capa de negocios para transportar los datos. De esta forma se retorna un objeto que contiene todos los atributos relacionados entre sí en un solo llamado, en lugar de hacer varios llamados para retornar los atributos por separado. Esta estrategia permite reducir el

tráfico por la red.

- El patrón Instancia Única (*Singleton*) es un patrón GoF que permite que solamente una instancia de una clase sea accesible de forma global a nivel de aplicación. De esta forma se puede controlar la forma en que una clase es instanciada, asegurando que sólo exista una única instancia de ella a nivel global.

- El patrón Método de Fabricación (*Factory Method*) es un patrón GoF que permite decidir qué clase en una jerarquía de clases será instanciada, basándose en las condiciones o parámetros dados. Este patrón se puede utilizar en combinación con el patrón Instancia Única, y se utiliza cuando se desea conectar jerarquías de clases paralelas, cuando una clase no puede anticipar quiénes serán sus subclases, cuando se trabaja utilizando interfaces y no clases implementadas de forma tal que se esconden al cliente las clases concretas.

- El patrón Fachada (*Facade*) es un patrón GoF que permite simplificar la complejidad de un sistema, ya que permite tener una interfaz general que funciona como una capa entre los subsistemas. Este patrón permite el desacoplamiento de los subsistemas al minimizar la comunicación entre estos; reduce su interdependecia y facilita la portabilidad del sistema.

- El patrón Decorador (*Decorator* o *Wrapper*) es un patrón GoF que agrega responsabilidades o funcionalidades adicionales a un objeto, de forma dinámica o estática.

II.1.2. Arquitectura multicapas

La arquitectura multicapas o n-capas es una arquitectura cliente-servidor en la cual la presentación, la lógica del negocio y la gestión de los datos constituyen procesos lógicamente separados. El diseño más comúnmente usado de la arquitectura n-capas es el de 3 capas, compuesto por capa de presentación, capa de negocios y capa de datos.

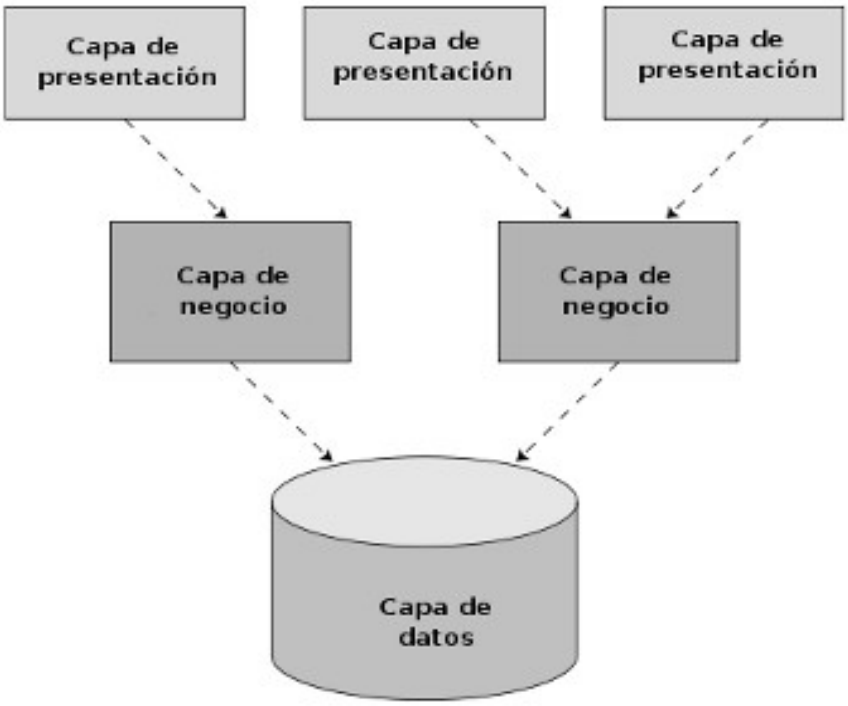

Figura 6: Arquitectura de tres capas.

Todas las capas pueden residir en un solo servidor, aunque lo más común es que la capa de presentación resida en uno o varios servidores, independientemente de las capas de negocio y datos. Estas dos últimas pueden residir en un solo servidor o, de acuerdo a las necesidades y la complejidad del sistema, separase en servidores distintos.

Aparte de las ventajas de desarrollar un software modular con interfaces bien definidas, la arquitectura n-capas permite que cualquiera de las capas pueda ser remplazada de forma independiente, sin afectar el resto de las capas[ACM, 2003].

Como se muestra en la Figura 6, las tres capas fundamentales de esta arquitectura son las siguientes:

- Capa de presentación, es la que presenta sistema al usuario, muestra y recoge información del usuario, y puede realizar un filtrado previo para comprobar que no existan errores. Esta capa debe ser entendible y fácil de usar para el usuario. Se comunica únicamente con la capa de negocio.

- Capa de negocio, es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (o lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al sistema gestor de bases de datos el almacenamiento o recuperación de datos.

- Capa de datos, es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más sistemas de gestión de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

II.2. La plataforma Java Platform Enterprise Edition

La plataforma Java Platform Enterprise Edition (Java EE, y conocida hasta su versión 1.4 como J2EE), es una plataforma de programación basada en la plataforma Java para desarrollar y ejecutar aplicaciones empresariales con una arquitectura de n-capas distribuida, basándose en componentes de software modulares que se ejecutan sobre un servidor de aplicaciones.

II.2.1. Java Platform Enterprise Edition

La plataforma Java es el nombre de un entorno o plataforma de computación originaria de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java u otros lenguajes que compilen a *bytecode* Java, y un conjunto de herramientas de desarrollo. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino más bien una máquina virtual encargada de la ejecución de aplicaciones, y un conjunto de librerías estándar que ofrecen funcionalidades comunes. Java es usado en una gran variedad de plataformas que pueden ir desde dispositivos móviles hasta grandes servidores[Sun, 2008].

Una edición de la plataforma Java consiste en un grupo de programas relacionados entre sí, o plataforma, que permiten le ejecución y desarrollo de programas escritos en esta plataforma. Las

ediciones existentes para Java son las siguientes:

- Java SE (Edición Estándar o *Standard Edition*), utilizada para propósitos generales, en PCs de escritorio y servidores.

- Java ME (*Micro Edition*), es una edición limitada, compuesta por un subconjunto de las librerías de la Edición Estándar, y utilizada fundamentalmente para desarrollar aplicaciones que deben ejecutarse en dispositivos móviles.

- Java EE (Edición Empresarial o *Enterprise Edition*), está compuesta por las librerías de la Edición Estándar, más una amplia gama de librerías destinadas al desarrollo de aplicaciones empresariales basadas en arquitectura multicapas.

Java EE incluye varias especificaciones de API, tales como JDBC, RMI, JMS, servicios web, XML, y otros; y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para componentes Java EE. Estas incluyen Enterprise JavaBeans (EJB), Servlets, Portlets, JavaServer Pages (JSP) y varias tecnologías de servicios web. Esto permite a los desarrolladores aplicaciones empresariales portables entre plataformas y escalables, a la vez que se pueden integrar con tecnologías anteriores[Sun, 2006a].

Java EE está conformada por dos elementos de arquitectura fundamentales: los componentes y los contenedores. Los contenedores constituyen ambientes de ejecución Java EE sobre los cuales se despliegan los componentes, de forma tal que los contenedores proporcionan todos los recursos que estos necesitan para su ejecución. Como se observa en la Figura 7, la especificación de Java EE define cuatro tipos de componentes, que son los siguientes:

- Aplicaciones cliente escritas en lenguaje Java, que pueden ser aplicaciones de consola o escritorio.

- Applets, es decir, aplicaciones con una interfaz gráfica de usuario semejante a la de una aplicación de escritorio, pero que se ejecutan en un navegador web.

- Servlets, páginas JSP, aplicaciones JSF y otros que se ejecutan dentro de un contenedor web y

responden a pedidos HTTP efectuados por los clientes. En este punto también se pueden incluir los servicios web, que utilizan el protocolo SOAP/HTTP.

- Componentes distribuidos o Enterprise JavaBeans (EJB), que se despliegan sobre un contenedor EJB, el cual da soporte para seguridad, transacciones y otros. La lógica de negocio de las aplicaciones Java EE suele estar implementada sobre EJB.

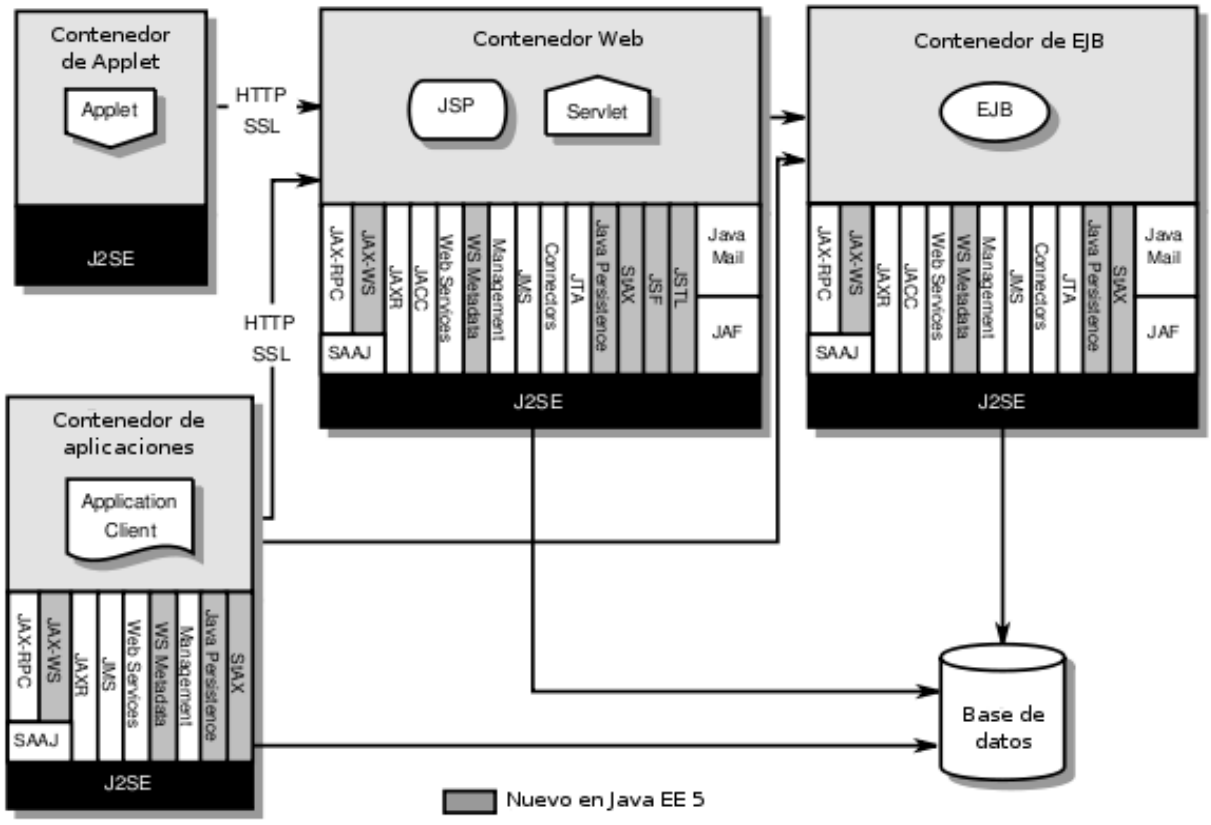

Figura 7: Arquitectura de la plataforma Java EE versión 5.

Los componentes Java EE nunca interactúan directamente con otros componentes Java EE, sino que utilizan los protocolos y métodos del contenedor para interactuar entre ellos y con los servicios de la plataforma. Al interponer el contenedor entre los distintos componentes, se permite que el contenedor, de forma transparente, pueda inyectar a los componentes los servicios que estos necesiten, entre los cuales se incluye el manejo de transacciones, chequeos de seguridad, asignación de recursos y gestión del estado de los componentes[Sun, 2006a].

II.2.1.1. Otras plataformas

Por supuesto, existen otras plataformas para el desarrollo de aplicaciones empresariales, como es el caso de la plataforma Microsoft.NET (en lo adelante .NET), desarrollada por Microsoft.

Las plataformas .NET y Java presentan muchas similitudes. Ambos se basan en el modelo de máquina virtual, es decir, las aplicaciones son compiladas a un código intermedio que se abstrae de las peculiaridades del hardware. En .NET este código intermedio se llama Common Intermediate Language (CIL, y primeramente llamado MSIL)[Microsoft, 2009a][Microsoft, 2009b]; y Sun llama al suyo *bytecode* Java. Ambas plataformas cuentan con un conjunto de librerías de clases que implementan gran cantidad de funcionalidades.

La plataforma .NET, en su implementación original desarrollada por Microsoft, solo puede ser instalada sobre el sistema operativo Microsoft Windows, mientras que la plataforma Java puede ser instalada sobre gran variedad de sistemas operativos como Microsoft Windows, GNU/Linux, Solaris y MacOS X. No obstante existe una implementación libre de la plataforma .NET, conocida como Mono y desarrollada por Novell, que sí está pensada para ejecutarse sobre cualquier sistema operativo; aunque las librerías de clases que provee Mono no siempre están completamente implementadas o actualizadas, por lo cual una aplicación compilada sobre la plataforma .NET podría no ejecutarse correctamente sobre Mono.

La plataforma .NET fue diseñada desde un principio para soportar múltiples lenguajes como C++.NET, VisualBasic.NET, J# (una implementación de Java para .NET) y C#, el lenguaje emblemático de la plataforma[Microsoft, 2009a][Microsoft, 2009b]. Por su parte la plataforma Java fue inicialmente pensada para soportar únicamente el lenguaje Java. No obstante, se han implementado otros lenguajes que tributan a ambas plataformas, como IronPython y IronRuby (implementaciones de Ruby y Python para .NET); JRuby y Jython (implementaciones de Ruby y Python para la plataforma Java) y Groovy, también para la plataforma Java[IsResearch, 2009].

Finalmente, la plataforma Java, incluyendo las librerías de clases, el compilador, la Máquina Virtual de Java y otras herramientas asociadas a la plataforma Java, tienen licencia de código abierto GNU

General Public License (con excepción del Classpath). Por su parte, la plataforma .NET es software propietario, el código de las librerías de clases solo está disponible como referencia, bajo la licencia Microsoft Reference Source License[Olamendi, 2005].

II.2.2. Enterprise JavaBeans

Los Enterprise JavaBeans (EJB) son componentes distribuidos definidos como parte de la plataforma Java EE, que se despliegan del lado del servidor para la construcción de aplicaciones distribuidas. Las especificaciones de EJB fueron desarrolladas originalmente por IBM en 1997, adoptadas dos años más tarde por Sun Microsystems (EJB 1.0 y 1.1) y mejoradas por el Java Community Process (JCP) bajo los estándares JSR-19 para EJB 2.0, JSR-153 para EJB 2.1 y JSR-220 para EJB 3.0[Sun, 2006b]. Actualmente se encuentra en fase de desarrollo la versión 3.1 de EJB[Sun, 2009].

La especificación de los EJB provee una forma estándar de implementar la lógica de negocio de las aplicaciones empresariales, dado que se constató que en la implementación de este tipo de aplicaciones aparecían problemas comunes y se solían reimplementar las mismas soluciones. Así, EJB está diseñado para llevar a cabo tareas como en manejo de la persistencia, la integridad de las transacciones, la seguridad, entre otras, de una forma estándar, dejando que los programadores se puedan concentrar en los problemas particulares del software que están desarrollando[SBS, 2006].

II.2.2.1. Componentes distribuidos

EJB permite el desarrollo y despliegue de componentes distribuidos. Estos componentes distribuidos, también conocidos como objetos distribuidos, pueden ser invocados de forma remota. Es decir, además de poder ser invocados desde cliente que se encuentre en el mismo proceso, pueden ser invocados por clientes que se encuentren en diferentes sistemas a lo largo de la red.

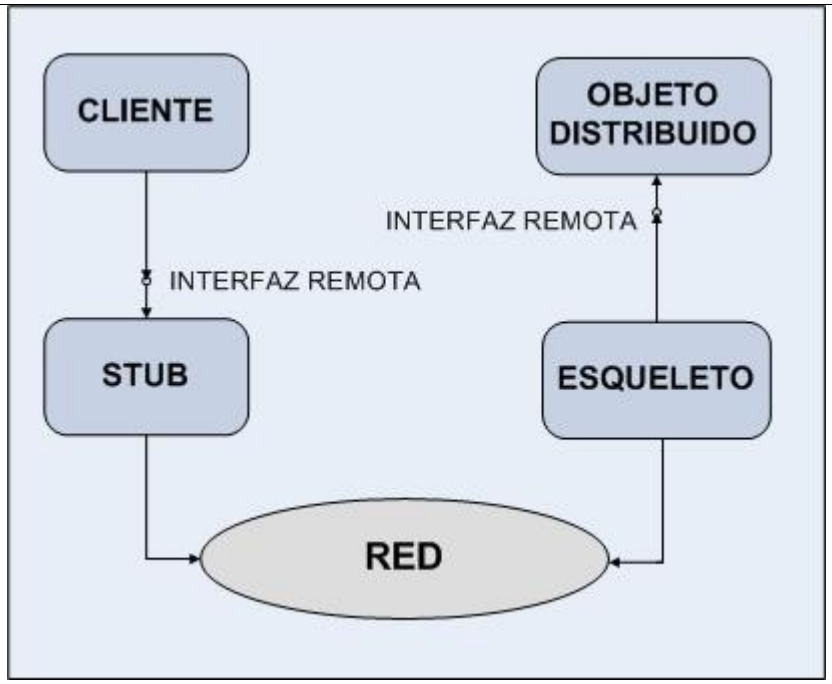

Figura 8: Invocación a un método remoto.

La invocación de un método remoto en un componente distribuido se realiza como se muestra en la Figura 8:

- El cliente llama a un objeto que actúa como proxy del lado del cliente. Este es el responsable de hacer transparente para el cliente todo el proceso de comunicación a través de la red.

- El proxy del lado del cliente llama a través de la red a un esqueleto, que es un proxy del lado del servidor. Este otro proxy enmascara las comunicaciones a través de la red del lado del servidor.

- El proxy del lado del servidor delega el control al objeto que contiene la implementación del método llamado. Este realiza sus operaciones, y al terminar retorna el control al proxy del lado del servidor, quien a su vez retornar al proxy del lado del cliente, y este por último retorna al cliente.

Tanto el proxy del lado del cliente como el proxy del lado del servidor implementan la misma interfaz, llamada interfaz remota. Esto significa que clonan la definición de los métodos del componente distribuido, de forma tal que el cliente cree que está llamando directamente al componente distribuido,

cuando en realidad está llamando al proxy[SBS, 2006].

Entre las tecnologías existentes para la programación de componentes distribuidos, se encuentran Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM) de Microsoft; e Remote Method Invocation sobre Internet Inter-Orb Protocol (RMI-IIOP) de Sun Microsystems[SBS, 2006].

II.2.2.2. Contenedor de EJB

El contenedor de EJB es el responsable de la gestión de los EJB. Entre sus responsabilidades se encuentran ofrecer un ambiente distribuido, seguro y transaccional, en el cual los EJB se puedan ejecutar. Por tanto, ni los EJB ni los clientes necesitan realizar llamadas de forma explícita a ninguna API del contenedor para poder acceder a sus servicios; sino que le hacen saber al contenedor sus necesidades a través de un fichero de descripción de despliegue en formato XML, o dentro del código del EJB por medio de anotaciones Java. De acuerdo a las especificaciones, los contenedores de EJB deben llevar a cabo las siguientes tareas[Jboss, 2006]:

- Persistencia, por medio del API de Persistencia de Java (*Java Persistence API* o JPA).
- Procesamiento de transacciones, a través de la API de Transacciones de Java (*Java Transaction API* o JTA).
- Control de concurrencia.
- Clusterización y balanceo de cargas.
- Gestión de recursos, como es el caso de hilos, sockets y conexiones a bases de datos.
- Gestión de eventos, usando el Servicio de Mensajería de Java (*Java Message Service*, o JMS).
- Gestión de nombres y servicio de directorios, por medio de la Interfaz de Nombres y Directorios de Java (*Java Naming and Directory Interface*, o JNDI).
- Seguridad, por medio de la Extensión de Criptografía de Java, y del Servicio de Autorización y

Autenticación de Java.

- Despliegue de componentes de software en el contenedor de EJB, situado en el servidor de aplicaciones.
- Gestión del ciclo de vida de los EJB, que incluyen su creación, destrucción o serialización a dispositivos de almacenamiento si es necesario.
- Llamado a métodos remotos utilizando el mecanismo RMI-IIOP.
- Exposición de métodos de negocio por medio de Servicios Web.

II.2.2.3. Tipos de EJB

Los EJB están formados por tres tipos de componentes, también llamados *Beans*, que son los EJB de Entidad (*Entity Beans*), que representan datos persistentes en una base de datos; los EJB de Sesión (*Session Beans*), que representan la lógica del negocio; y los EJB orientados a Mensajes (*Message Driven Beans*), diseñados para manejar mensajes JMS de forma asíncrona.

Los EJB de entidad representan datos persistentes en una base de datos. Utilizan la técnica del Mapeo Objeto-Relacional (*Object-Relational Mapping*, u ORM), que consiste en convertir los datos entre los tipos utilizados por un lenguaje de programación orientado a objetos, y los utilizados por una base de datos relacional, de forma tal que el desarrollador se abstraiga del sistema de gestión de bases de datos y solamente vea objetos relacionados entre sí en el lenguaje de programación que esté utilizando.

Un EJB de Entidad puede gestionar su propia persistencia, es decir, definir sus propias consultas SQL para todas las operaciones básicas como inserción, actualización, borrado o recuperación; o puede dejar su persistencia en manos del contenedor de EJB.

A partir de la especificación de EJB 3.0, los EJB de Entidad fueron sustituidos por JPA. JPA es un estándar definido por Sun Microsystems, tomando ideas de otras plataformas ORM, como es el caso de Hibernate y Java Data Objects. Para JPA, siguiendo la filosofía ORM, una entidad persistente es una clase en Java que representa una tabla en una base de datos relacional. Cada instancia de esta entidad

representa una fila en la tabla, en tanto que cada propiedad definida en la clase representa una columna en la tabla. Las relaciones entre las entidades son representadas como relaciones entre las tablas de la base de datos, y son expresadas en JPA por medio de metadatos, los cuales pueden ser especificados directamente en la clase Java utilizando anotaciones Java, o en un fichero descriptor en formato XML que se distribuye junto con la aplicación. Las consultas a la base de datos se realizan utilizando el Java Persistence Query Language (JPQL). Este lenguaje utiliza una sintaxis semejante a la de las consultas SQL, pero opera sobre las entidades en lugar de hacerlo sobre las tablas directamente[SBS, 2006].

Si bien los EJB de Entidad representan datos persistentes en una base de datos, los EJB de Sesión implementan la lógica del negocio sobre la capa de datos. Una instancia de un EJB de Sesión es creada por medio de un pedido de un cliente, y dura únicamente mientras se mantenga activa la sesión cliente-servidor. Aunque estas sesiones son transaccionales, no son recuperables si existiese un error durante la ejecución. Los objetos de los EJB de Sesión pueden mantener su estado a lo largo de varios métodos o transacciones (*statefull*), o pueden no hacerlo (*stateless*).

Los EJB de sesión exportan al cliente una interfaz, con el objetivo de que el cliente conozca los métodos del EJB a los que puede acceder. Esta interfaz puede ser local, si el cliente se encuentra desplegado en el mismo contenedor de EJB que el EJB de Sesión, o remota si se encuentra desplegado fuera del contenedor. Con esta interfaz y el nombre del EJB de Sesión, el cliente debe buscar el EJB en el JNDI, para obtener una instancia del EJB.

Los EJB de Sesión con estado se utilizan cuando se necesita realizar operaciones que involucran varios pasos, y por lo tanto es necesario guardar el estado del objeto entre cada una de las invocaciones a los distintos métodos. Por su parte, los EJB de Sesión sin estado no necesitan guardar el contenido de las variables, por lo cual son más livianos y necesitan menos gasto de recursos[SBS, 2006].

Los EJB orientados a Mensajes integran la tecnología EJB con JMS, para crear un nuevo tipo de EJB diseñado para manejar mensajes JMS de forma asíncrona. Es decir, que se encargan de realizar tareas que no necesitan una respuesta inmediata y por tanto pueden ser realizadas en paralelo con la tarea que la invocó. Estos EJB se subscriben a las colas y a los tópicos definidos por JMS, y solamente definen un método que se encarga de escuchar qué mensajes llegan a través de la cola o tópico JMS, y

procesarlos[SBS, 2006].

II.2.3. Servidores de aplicaciones Java EE

Como se ha explicado anteriormente, los EJB son desplegados sobre un contenedor de EJB. Este contenedor, a su vez, se encuentra implementado como parte de un servidor de aplicaciones Java EE. Un servidor de aplicaciones Java EE funciona como un contenedor para los componentes que conforman dichas aplicaciones, como pueden ser los ya mencionados Servlets, JSP, Applets y EJB. La portabilidad de la plataforma Java ha permitido que estos servidores de aplicaciones estén disponibles para una gran variedad de plataformas, como Unix, Microsoft Windows y GNU/Linux[Sun, 2006a].

Los servidores de aplicaciones son certificados oficialmente por Sun Microsystems como compatibles con determinada versión de Java EE. Actualmente existe una gran cantidad de servidores de aplicaciones compatibles con Java EE. Entre ellos se puede mencionar a Jboss AS, desarrollado por Red Hat, que cuenta con una versión código abierto desarrollada por la comunidad, y otra versión empresarial; el Sun Java System Application Server, versión comercial del servidor de aplicaciones GlassFish, de Sun Microsystems, pero de código abierto; el WebLogic Application Server de BEA Systems; el Apache Geronimo, de Apache Foundation y el IBM WebSphere Application Server, desarrollado por IBM y basado en Apache Geronimo, entre otros.

El término de servidor de aplicaciones se ha utilizado no solo para productos compatibles con Java EE, sino también para servidores que tributan a otras plataformas. Tal es el caso de Internet Information Service, un servidor de aplicaciones desarrollado por Microsoft y que funciona sobre la plataforma .NET[Microsoft, 2009c]; y de Zope, un servidor para aplicaciones web desarrollado en Python, y sobre el cual se pueden ejecutar aplicaciones desarrolladas sobre plataformas de código abierto como Django y TurboGears, las cuales han ido ganando popularidad dentro del entorno Python[Zope, 2009].

II.2.3.1. Servidor de aplicaciones Jboss AS

Como se ha mencionado anteriormente, Jboss AS (a partir de ahora Jboss) es un servidor de aplicaciones Java EE de código abierto. Al estar basado en Java, JBoss puede ser utilizado en cualquier

sistema operativo que lo soporte. Inicialmente Jboss fue desarrollado por Jboss Inc, la cual fue posteriormente adquirida por Red Hat en abril del 2006. En la actualidad el proyecto es apoyado por una red mundial de colaboradores.

Como todo servidor de aplicaciones Java EE, Jboss debe soportar todas las características de un contenedor EJB que se han expuesto anteriormente, además de otras. Entre estas se puede mencionar la clusterización, balanceo de cargas, caché (utilizando un producto llamado Jboss Cache), despliegue distribuido de componentes, soporte para programación orientada a aspectos, soporte para Servlets, JSP y JavaServer Faces, soporte para EJB, integración con Hibernate para persistencia por medio de JPA, soporte para acceso a bases de datos a través de Java Database Connectivity (JDBC), soporte para transacciones con JTA, servicios web Java EE utilizando tecnología Java API for XML Web Services (JAX-WS), integración con JMS, soporte para seguridad a través de JAAS, y manejo y monitoreo de aplicaciones, objetos del sistema y dispositivos utilizando Java Management Extensions (JMX), tecnología a la que se hará referencia más adelante; entre otras[Jboss, 2006].

II.2.3.2. JMX y MBeans

El servidor de aplicaciones Jboss presenta una arquitectura modular, basada en una infraestructura JMX. JMX brinda a las aplicaciones herramientas de monitoreo tanto locales como remotas.

La arquitectura de JMX está formada por tres capas. La primera es la capa de instrumentación, que contiene la instrumentación que da el usuario a un recurso dado, por medio de un componente llamado Managed Bean (MBean). Los MBean son objetos Java que representan los recursos que van a ser manejados de forma remota. Un MBean contiene una interfaz, consistente en:

- Atributos que pueden ser leídos o escritos.
- Operaciones que se puede invocar.
- Notificaciones emitidas por el MBean.

Por ejemplo, un MBean para la configuración de una aplicación puede definir atributos que representen

los parámetros de configuración de la aplicación, así como operaciones que permitan persistir la configuración actual, y notificaciones en el momento en que una configuración dada cambie[Jboss, 2006].

La segunda es la capa agente, también conocida como MBeanServer, que constituye el núcleo de JMX y funciona como intermediara entre los MBean y las aplicaciones. La tercera es la capa de manejo remoto, que permite que aplicaciones remotas puedan acceder al MBeanServer por medio de adaptadores o conectores.

En el momento del arranque del servidor de aplicaciones, se crea una instancia del MBeanServer, de forma tal que todos los recursos manejables puedan ser registrados en este MBeanServer. Los MBean a ser cargados posteriormente se especifican en un fichero de configuración. El MBeanServer no realiza demasiadas tareas, solamente actúa como un microkernel encargado de agregar componentes que se interconectan con los MBean. Estos MBean son los que aportan las funcionalidades específicas[Jboss, 2006].

De esta forma, todos los componentes que soporta Jboss se agregan en forma de módulos manejados por medio de MBean y conectados al MBeanServer. Los principales módulos son los siguientes:

- Jboss EJB Container, el cual es el núcleo de la implementación de Jboss. Genera los proxy del lado del cliente y del servidor para los EJB en tiempo de ejecución, y soporta redespliegue de componentes en caliente.
- JbossNS, es el servicio de nombres que permite localizar los objetos y recursos. Implementa la especificación de JNDI.
- JbossMQ, una implementación de JMS.
- WebServers, con soporte para contenedores web.

Los MBean también se pueden utilizar para implementar el inicio y finalización de una aplicación empresarial, definiendo los métodos *start* para el inicio, y *stop* para la finalización. Por lo general las aplicaciones empresariales no tienen un punto de entrada definido, sino que son desplegadas en el

servidor de aplicaciones y desde ese momento las aplicaciones clientes son capaces de acceder a los métodos de negocio de forma remota.

No obstante, en ocasiones es necesario realizar algunas operaciones en el momento del despliegue de la aplicación, para lo cual se utiliza un MBean. De esta forma, se puede definir cómo aplicación cualquiera se inicia y finaliza, y realizar operaciones como registrar componentes en una base de datos, iniciar temporizadores (*timers*), cargar ficheros de propiedades y en general realizar cualquier acción necesaria para el inicio de una aplicación empresarial[Jboss, 2006].

II.2.4. Servicios destinados a móviles en Java

La API para servicios destinados a móviles de Java, llamada Server APIs for Mobile Services (SAMS), fue creada con el objetivo de acelerar el desarrollo de herramientas y aplicaciones a través de una interfaz estándar que fuera capaz de comunicarse con los proveedores de servicios para móviles.

La especificación de SAMS, conocida como JSR-212, fue aprobada en 2004, y en ella trabajó un grupo de expertos de Nokia Corporation, aunque también aportaron otras empresas como Apache, Cingular, BEA Systems, Siemens y Sun Microsystems[Nokia, 2004].

Las principales características que presenta SAMS son las siguientes:

- Es una API común que representa una capa de abstracción a los diferentes protocolos que ofrecen los proveedores de servicios para móviles.
- Utiliza interfaces simples y fáciles de usar por los programadores.
- Permite crear aplicaciones portables.
- Utiliza patrones de diseño establecidos y constituye un modelo confiable para la programación Java.
- Separa la implementación de la API, de forma tal que la API es independiente de la implementación, y por tanto pueden existir diferentes implementaciones de la misma

especificación.

La arquitectura de SAMS puede ser dividida en tres partes fundamentales: la API de SAMS, la implementación de la API, y los controladores (*drivers*) para cada uno de los protocolos, como se observa en la Figura 9. Las aplicaciones interactúan con la API de SAMS, aunque al final se ejecuta el código subyacente en la implementación de la API. Esta implementación a su vez contiene el código necesario para manejar los distintos protocolos implementados en los controladores, a través de ficheros de configuración.

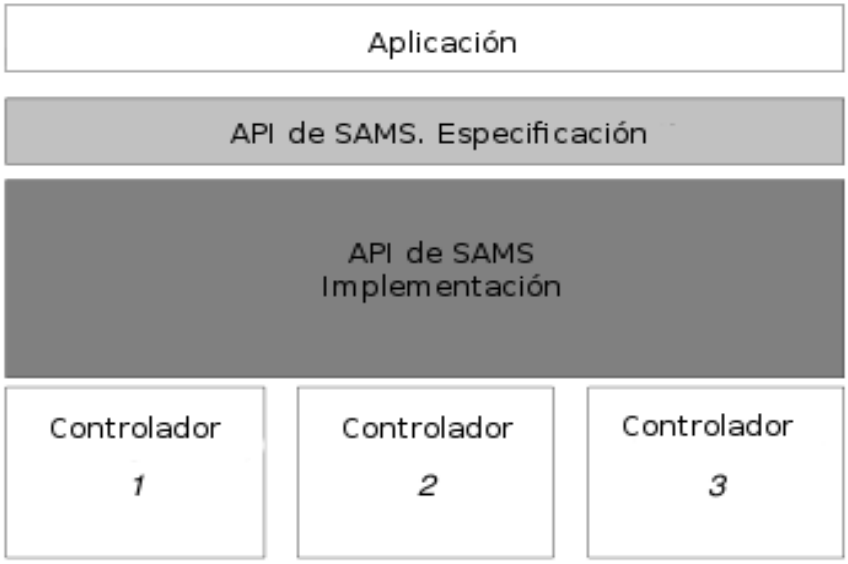

Figura 9: Arquitectura de SAMS.

De esta forma, utilizando una implementación de SAMS, en conjunto con los controladores para cada los protocolos que ofrecen los proveedores de servicios, cualquier aplicación puede enviar y recibir mensajes de texto (SMS), mensajes multimedia (MMS) y notificaciones (WAP Push)[Nokia, 2004].

II.3. Conclusiones del capítulo

En la primera parte del capítulo se trató el tema de los patrones de diseño. Los patrones de diseño constituyen soluciones reusables a problemas que ocurren comúnmente en el diseño del software. Entre

los patrones de diseño más populares se encuentran los definidos por la Banda de los Cuatro y los patrones GRASP. Asimismo, para el caso específico de la plataforma Java EE se encuentran los patrones J2EE, definidos por Sun Microsystems. Existen también patrones de arquitectura, que pueden ser aplicados al nivel de la arquitectura del sistema, como el caso de la arquitectura multicapas, en la cual la presentación, la lógica del negocio y la gestión de los datos constituyen procesos lógicamente separados.

La segunda parte del capítulo aborda la plataforma Java, específicamente la plataforma para aplicaciones empresariales Java EE. Esta plataforma permite el desarrollo de aplicaciones empresariales basadas en una arquitectura multicapas. Java EE incluye varias especificaciones de API, como JMS para la gestión asíncrona de mensajes y SAMS. Este último ofrece una interfaz estándar para comunicarse con los proveedores de servicios para móviles. Java EE además incluye especificaciones para componentes Java EE. Estas últimas incluyen los EJB, Servlets y JSP.

Específicamente los EJB están formados por tres tipos de componentes: los EJB de Entidad, que representan datos persistentes en una base de datos; los EJB de Sesión, que representan la lógica del negocio; y los EJB orientados a Mensajes, diseñados para manejar mensajes JMS de forma asíncrona.

Finalmente se abordaron los servidores de aplicaciones Java EE, haciendo énfasis en el servidor Jboss AS. Estos servidores de aplicaciones funcionan como contenedores para los componentes que conforman dichas aplicaciones. Los servidores de aplicaciones Java son capaces de manejar transacciones, seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados.

III. LA PLATAFORMA BLUEEYE

En el presente capítulo se expondrán los aspectos fundamentales de la implementación y funcionamiento de la plataforma para la gestión de servicios destinados a dispositivos móviles BlueEye (BE); las funciones del SDK para la implementación de nuevas aplicaciones, y de la interfaz para el monitoreo y la administración. Además se expondrán las primeras aplicaciones implementadas y desplegadas sobre la plataforma, así como las pruebas realizadas a esta.

III.1. Características generales de la plataforma

La plataforma fue diseñada teniendo en cuenta las principales funcionalidades de las plataformas existentes a nivel mundial que fueron objeto de estudio. De todas estas, se tuvieron en cuenta un grupo de funcionalidades que resultaban fundamentales para la implementación de BE, como son la posibilidad de comunicarse con uno o varios operadores móviles por medio de distintos protocolos, el soporte para distintos servicios basados en mensajería móvil, la implementación de mecanismos de monitoreo y reportes, la posibilidad de ofrecer un SDK para facilitar la implementación de nuevas aplicaciones, así como la posibilidad de ofrecer servicio de mensajería a terceros. Estas funcionalidades se exponen a continuación.

- Comunicación con uno o varios operadores móviles por medio de distintos protocolos. A pesar de que en el país solamente existe un solo operador móvil (Cubacel) y que la comunicación con este se realiza por medio del protocolo SMPP, se considera importante que la plataforma esté preparada para comunicarse con varios operadores y a través de varios protocolos al mismo tiempo. Además, esta característica se puede aprovechar para implementar un SMSC virtual que simule el envío y recepción de mensajes, lo cual facilita la realización de pruebas a la plataforma de forma local, sin necesidad de estar conectados a un SMSC real.

- Soporte para distintos tipos de servicios basados en mensajería móvil, teniendo en cuenta la palabra clave del mensaje para distinguirlos. La plataforma debe ser capaz de gestionar múltiples aplicaciones basadas en mensajes de texto. Para ello se utilizará la primera palabra del

mensaje de texto como palabra clave para enrutar los mensajes entrantes a las distintas aplicaciones. Esto conlleva a que la plataforma debe controlar qué aplicaciones se encuentran desplegadas en todo momento. Por su parte, las aplicaciones deben utilizar las funcionalidades que brinda la plataforma para enviar sus mensajes de respuesta a los usuarios.

- Monitoreo y reportes. La plataforma debe brindar funciones que permitan su monitoreo, administración y generación de reportes. Entre los parámetros a monitorear pueden mencionarse el estado de los mensajes de entrada y de salida, el listado de las aplicaciones desplegadas, y las trazas generadas por la plataforma y las aplicaciones, entre otros. También debe generar reportes del tráfico de mensajes, que permitan contrastar la información de la plataforma contra la del operador móvil.

- SDK para la implementación de nuevas aplicaciones sobre la plataforma. Mediante este SDK se facilita y acelera el desarrollo de aplicaciones que brinden nuevos servicios a través de la plataforma.

- Servicio de mensajería a terceros. Si bien el SDK facilita el desarrollo de aplicaciones que se van a desplegar sobre la plataforma, es conveniente ofrecer un mecanismo para que las aplicaciones de terceros puedan utilizar las facilidades que da la plataforma para el envío y recepción de mensajes. Por ello se habilita la posibilidad de realizar el envío y recepción de mensajes por vía HTTP/HTTPS a clientes registrados, por medio de una aplicación que se despliega sobre la plataforma y se encarga de brindar este servicio.

Otras funcionalidades que presentan los sistemas a nivel mundial, como son la conversión de mensajes de voz a mensajes de texto y viceversa, envío de mensajes desde correo electrónico, no se tuvieron en cuenta para esta primera versión del producto, aunque podrían ser agregados también en forma de aplicaciones. Por su parte, la descarga de contenidos podría ser vista como otra aplicación a desplegar sobre la plataforma, aunque diseñar e implementar una aplicación para la gestión de contenidos para móviles es un tema de mayor complejidad que no se pretende abordar en este trabajo.

Además, se tuvieron en cuenta una serie de factores como son la robustez, extensibilidad, reusabilidad y

portabilidad. Por ello, se definieron una serie de requisitos que la plataforma debía cumplir, como los siguientes:

- Como la plataforma es la encargada de enrutar los mensajes entrantes a la aplicación correspondiente, y los mensajes salientes al operador, debe tener en cuenta los mensajes que fallan al ser enrutados. Estos fallos pueden ocurrir por problemas en la conexión con las aplicaciones o el SMSC, o porque el mensaje contiene una palabra clave que no está registrada. En todos los casos la plataforma debe enviar estos mensajes a una cola de reintentos, y tratar de enviarlos una cantidad predeterminada de veces.

- La plataforma debe estar preparada para hacer frente a fallos en la conexión con el SMSC. La conexión con el SMSC puede perderse por múltiples razones, y la plataforma debe ser capaz de detectar estos eventos y reintentar reconectarse cada cierto tiempo.

- La plataforma debe tener una arquitectura modular, de forma tal que exista alta cohesión y el bajo acoplamiento a nivel de cada uno de los módulos. La plataforma debe consistir en un núcleo que se encargue del enrutamiento de los mensajes y la gestión de las aplicaciones desplegadas; y de aplicaciones independientes que se registran en la plataforma, utilizan funcionalidades comunes que la plataforma provee en forma de un SDK, y que implementan cada una servicios independientes y bien definidos.

- La plataforma debe proveer una serie de funcionalidades comunes en forma de SDK, así como un conjunto de clases abstractas, que faciliten y aceleren el desarrollo de aplicaciones. El propio núcleo de la plataforma puede desarrollarse sobre las funcionalidades comunes que ofrece el SDK.

- La plataforma debe ser portable desde el punto de vista del hardware y del sistema operativo. Por ello se desarrolló sobre la plataforma Java Enterprise Edition, que es una plataforma de código abierto para el desarrollo de aplicaciones empresariales, y que cumple con los requisitos de portabilidad deseados. Asimismo la plataforma debe ser independiente del servidor de aplicaciones sobre el cual se vaya a desplegar, para facilitar la migración no solo de sistema

operativo o hardware, sino también de servidor de aplicaciones, dada la gran cantidad de servidores de aplicaciones Java que existen. Por ello, debe evitarse el uso de funciones o propiedades que sean propias de un servidor de aplicaciones en específico.

III.2. Funcionamiento general de la plataforma

La plataforma BE tiene como objetivo comunicarse con los SMSC de los operadores, y ofrecer una serie de servicios basados en mensajes para móviles, a través de estos SMSC. En la Figura 10 se muestra un esquema de la arquitectura de la plataforma. La comunicación con los SMSC de los operadores móviles se realiza a través de controladores (o *drivers*) implementados como parte de una capa de comunicaciones en el nivel más bajo de la plataforma. Esta capa de comunicaciones implementa el estándar SAMS (JSR-212) definido por Sun Microsystems. De esta forma, el funcionamiento de la plataforma se vuelve independiente del SMSC al que esté conectado y del protocolo que este utilice, y en teoría podría ser conectado a varios SMSC al mismo tiempo, utilizando el concepto de rutas, que será explicado en detalle más adelante. Por tanto, en esta capa de comunicaciones, la plataforma se encarga esencialmente de garantizar la conexión con los SMSC, reintentar conexiones cerradas, y enviar y recibir mensajes que pueden ser mensajes de texto (SMS), mensajes multimedia (MMS) o notificaciones (WAP Push).

En la siguiente capa se encuentra el núcleo de la plataforma, el cual es el encargado de gestionar las rutas, los mensajes que entran o salen de la plataforma, los mensajes que fallan al ser enrutados, así como gestionar las aplicaciones desplegadas sobre la plataforma, las trazas de las aplicaciones, entre otras.

El núcleo de la plataforma recibe los mensajes que le envía SAMS en forma de eventos. Estos mensajes son procesados y enrutados hacia las distintas aplicaciones. El enrutamiento se realiza utilizando la primera palabra del mensaje de texto como palabra clave y el número corto al cual se envió el mensaje. Los temas relacionados con el enrutamiento se explicarán en detalle más adelante.

Los mensajes de salida que son enviados por las aplicaciones, se procesan a través de una cola que funciona de forma asíncrona. Esta cola está desplegada en el núcleo de la plataforma y es la encargada

de enrutar todos los mensajes de salida hacia el SMSC, a través de las rutas definidas en la plataforma.

De igual forma, el núcleo de la plataforma gestiona las aplicaciones que se encuentran ofreciendo servicios. Como la arquitectura de la plataforma es distribuida, estas aplicaciones pueden estar desplegadas en el mismo servidor de aplicaciones que el núcleo, o en servidores diferentes. Por tanto, cada aplicación debe encargarse de notificar al núcleo en el momento en que se inicia o se finaliza, a fin de que éste conozca cuándo comenzar o dejar de enrutarle mensajes a determinada aplicación.

El núcleo también se encarga de la gestión de las trazas. Estas pueden ser generadas por el mismo núcleo o por las aplicaciones. Las trazas generadas por las aplicaciones son enviadas al núcleo a través de una cola que funciona de forma asíncrona. De esta forma las aplicaciones no tienen que esperar que el núcleo procese la traza para continuar realizando sus operaciones. Las aplicaciones sólo pueden generar trazas de negocio, en tanto el núcleo es el encargado de generar las trazas de mensaje, que describen el tráfico de mensajes desde y hacia la plataforma, como se explicará más adelante.

La plataforma también cuenta con una interfaz para la administración y el monitoreo. Esta interfaz exporta las funciones necesarias para conocer el estado de las trazas, las aplicaciones desplegadas, los mensajes que se envían y se reciben y el estado de las rutas. Además permite conectar o desconectar las rutas manualmente, y generar ficheros de reportes, entre otros. Mediante esta interfaz cualquier aplicación cliente desarrollada sobre Java, ya sea de escritorio o web, puede conocer el estado de la plataforma y hacer uso de las funcionalidades que se exportan.

La plataforma cuenta con un SDK, consistente en un conjunto de clases y funciones comunes que se utilizan para desarrollar y desplegar nuevas aplicaciones, incluyendo el núcleo de la plataforma. Así, el desarrollo de aplicaciones se vuelve un proceso más rápido, en donde sólo es necesario implementar ciertas interfaces y utilizar las funciones ya definidas en la API, además de implementar la funcionalidad propia de cada aplicación. De igual forma, el SDK de BE debe desplegarse en cada servidor en donde se vayan a desplegar las aplicaciones de BE, incluyendo el servidor en donde está desplegado el núcleo de la plataforma. Por tanto el SDK de BE es la base tanto para el desarrollo como para el despliegue de aplicaciones basadas en la plataforma BE.

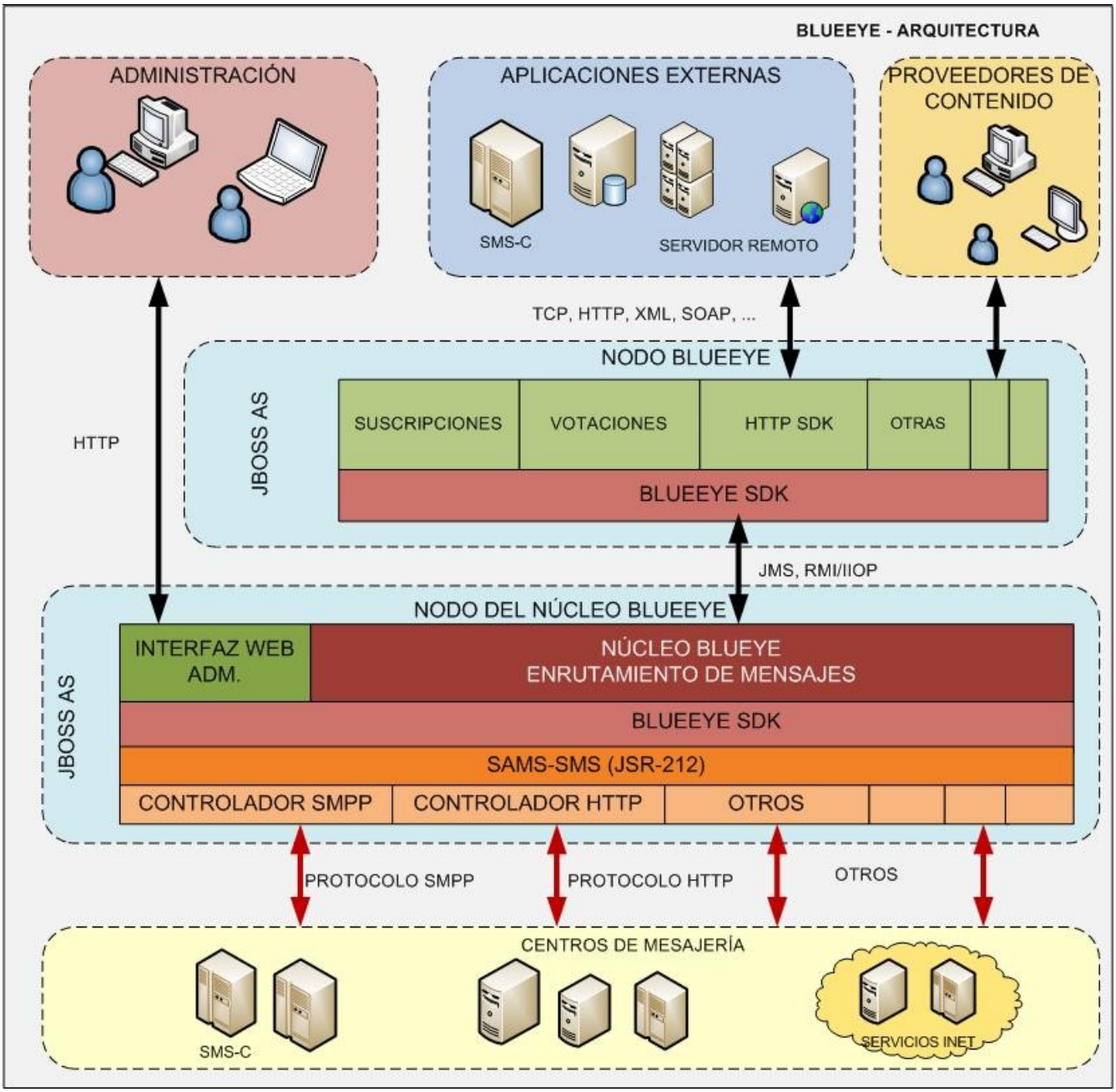

Figura 10: Arquitectura de la Plataforma BlueEye.

Las aplicaciones no forman parte del núcleo de la plataforma, debido a la arquitectura modular de esta. El núcleo de la plataforma puede existir y ejecutarse sin ninguna aplicación, y a lo largo del tiempo estas se pueden ir agregando o quitando en dependencia de las necesidades, incluso sin necesidad de

detener la ejecución del núcleo o del resto de las aplicaciones. Para desplegar las aplicaciones por separado, es necesario desplegar en cada servidor el SDK de BE, como se muestra en la Figura 10. Cada aplicación cuenta con una cola que funciona de forma asíncrona, y que tiene la función de recibir los mensajes que le son enviados a la aplicación desde el núcleo de la plataforma.

Cada aplicación, de acuerdo a sus características, puede ofrecer interfaces para la actualización de su información, o puede comunicarse con proveedores de información o contenido, tal y como se muestra en la Figura 10. De la misma forma, se pueden implementar aplicaciones de pasarela (en la figura representada como HTTP-SDK) cuyo objetivo es dar servicio de mensajería a aplicaciones de terceros, teniendo en cuenta la seguridad y la autenticación de los diferentes clientes.

A continuación se analizarán algunos de los requisitos que se tuvieron en cuenta en el diseño e implementación de la plataforma.

III.2.1. Modularidad

La arquitectura modular de BE permite que se implementen las aplicaciones de forma independiente, y que estas se puedan desplegar sobre la plataforma de acuerdo a las necesidades o intereses en un determinado momento, de tal forma que no tienen por qué conocer qué otras aplicaciones están desplegadas en ese momento. Si fuese necesario que existiera interacción entre distintas aplicaciones, se debería realizar enviando mensajes predefinidos al núcleo de la plataforma, que se encargaría más tarde de reenviarlos a la aplicación adecuada.

Por su parte, el núcleo de la plataforma se encarga solamente del envío y recepción de mensajes desde y hacia las aplicaciones; el enrutamiento de los mensajes que entran en la plataforma, el registro de las aplicaciones desplegadas y la gestión de las trazas.

Cada aplicación está implementada para brindar un servicio específico, y no tiene por qué interferir con el funcionamiento del resto de las aplicaciones desplegadas en un momento determinado. Es más, cada aplicación debe existir con independencia del resto de las aplicaciones, y debería estar preparada para ser desplegada ella sola, o con un conjunto de aplicaciones al mismo tiempo.

Pueden existir meta-aplicaciones, que implementan funcionalidades comunes a un grupo de aplicaciones y se encargan de interactuar con estas, pero ni siquiera en este caso la aplicación tiene por qué conocer cuáles son las demás aplicaciones desplegadas en un momento dado. Tal es el caso de la aplicación que gestiona las suscripciones. Ella solamente se encarga de notificar a una aplicación determinada que debe ejecutar un método definido en la interfaz de subscripciones, pero no sabe de qué forma cada aplicación implementa este método, ni siquiera si la aplicación se encuentra desplegada o no. Este proceso de notificación se realiza por medio de un tipo de mensaje destinado únicamente a las suscripciones.

De forma general, para el funcionamiento correcto de este tipo de meta-aplicaciones, es necesario agregar algún tipo de funcionalidad en el núcleo de la plataforma, pues, como se explicó anteriormente, el manejo de mensajes entre las aplicaciones se realiza por medio del núcleo. Estos cambios resultan transparentes al resto de las aplicaciones y no afectan las funcionalidades anteriores.

Las aplicaciones de la plataforma BE se pueden encontrar distribuidas en distintos servidores, o en un solo servidor, de acuerdo a las necesidades que puedan surgir en un momento determinado, tal y como se muestra en la Figura 10. La plataforma está preparada para lidiar con este tipo de situaciones sin que esto afecte las aplicaciones desplegadas en un momento dado. Agregar o quitar un nuevo servidor de aplicaciones con nuevas aplicaciones desplegadas debe ser un proceso transparente al núcleo de la plataforma, y se debe poder realizar sin afectar el servicio que se está brindando en un momento dado.

III.2.2. Independencia del servidor de aplicaciones

Como se explicó anteriormente, la plataforma BE es una aplicación distribuida que utiliza la tecnología Java EE. Por tanto, esta plataforma debe desplegarse sobre un servidor de aplicaciones Java. Para dar mayor portabilidad a la plataforma, es buena idea escribir la mayor cantidad posible de código fuente que sea independiente del servidor de aplicaciones. El código que sea específico de un servidor de aplicaciones determinado debe encontrarse localizado, lo cual facilitaría una posible migración a otro servidor de aplicaciones, ya que habría que cambiar poco o ningún código en el caso ideal, y solamente agregar las clases que implementen las características del nuevo servidor de aplicaciones. Otros factores

que inciden en la portabilidad como son el sistema operativo y el hardware, están asegurados ya que la plataforma Java es independiente de ambos.

Cuando una aplicación Java EE es desplegada en un servidor de aplicaciones, entre otras cosas, se registran los EJBs que forman parte de la aplicación, se crean entradas en el JNDI para cada uno de los Beans de Sesión y los Beans orientados a Mensajes, y se mapean los Beans de Entidad en el sistema de gestor de bases de datos que está usando el servidor de aplicaciones.

Cada servidor de aplicaciones tiene características específicas, como pueden ser los directorios donde se despliegan las aplicaciones, los directorios para los ficheros de configuración de las aplicaciones, los sistemas de *logs*, las propiedades para acceder al JNDI. Además, las aplicaciones Java EE pueden definir servicios dentro del servidor de aplicaciones Jboss utilizando MBeans. Esta característica permite realizar secuencias de inicio y finalización para cada aplicación, al definir en el MBean los métodos *start* y *stop*. Estos métodos se ejecutan en el momento en que el MBean se despliega en el servidor de aplicaciones, por lo cual el MBean puede ser utilizado para iniciar o finalizar las aplicaciones dentro de la plataforma.

En el caso de escribir el código de la plataforma orientado a un servidor de aplicaciones específico (por ejemplo, Jboss), si se quisiera desplegar en otro servidor de aplicaciones (por ejemplo, a pedido de algún cliente que utilice GlassFish u otro) sería necesario rescribir una parte del código, y prácticamente hacer una nueva rama del proyecto que cumpla con los nuevos requerimientos. Si por el contrario, se escribe el código independiente del servidor de aplicaciones, solamente sería menester implementar una nueva clase con las características del nuevo servidor de aplicaciones, sin afectar el resto del código. En el diagrama de clases de la Figura 11 se muestra la forma en que se implementó esta característica usando el patrón de diseño Método de Fabricación.

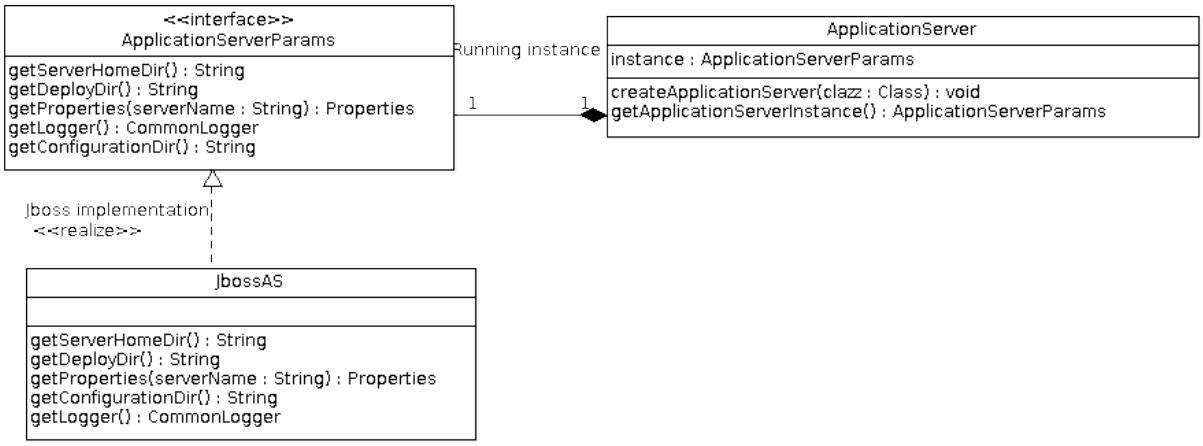

Figura 11: Diagrama de clases del patrón Método de Fabricación utilizado.

Las características que se desean conocer acerca del servidor de aplicaciones son:

- Directorio de configuración de las aplicaciones. Es necesario ya que tanto el núcleo de la plataforma como las aplicaciones desplegadas sobre esta, cuentan con ficheros de configuración que deben ser leídos por el núcleo de la plataforma cada vez que una nueva aplicación se despliega.

- Propiedades para el acceso al JNDI de forma remota. A través de estas propiedades la plataforma puede acceder a un servidor de aplicaciones remoto y utilizar componentes desplegadas en este otro servidor de aplicaciones.

- Sistema de *logs* para imprimir mensajes en la consola. Este sistema de *logs* es importante para imprimir mensajes en la pantalla, y a su vez guardarlos en el fichero de *logs* del servidor de aplicaciones y de esta forma dejar una traza detallada del funcionamiento de la plataforma. También puede ser utilizado con fines de depuración.

Sin embargo, el solo hecho de tener en cuenta estas características no es suficiente. Aunque se logró separar y localizar el código propio del servidor de aplicaciones Jboss, y escribir código independiente de las características de éste, falta experiencia en el uso de otros servidores de aplicaciones tales como GlassFish o WebLogic, entre otros, por lo cual no se puede asegurar que la posible migración a otro

servidor de aplicaciones pueda ocurrir sin afectar el código ya existente. No obstante lo hecho hasta ahora constituye una primera aproximación a una solución completa de este problema.

III.3. El núcleo de BlueEye

Las principales tareas del núcleo de la plataforma BE son la gestión de rutas, la gestión de aplicaciones y la gestión de trazas, como se muestra en la Figura 12. El gestor de rutas se encarga de la comunicación de la plataforma con los SMSC, en tanto que el gestor de aplicaciones se encargar del enrutamiento de los mensajes hacia las distintas aplicaciones desplegadas, teniendo en cuenta para el enrutamiento la primera palabra del mensaje y el número corto al cual fue enviado. La plataforma también debe controlar el registro de las aplicaciones que se encuentran desplegadas en un momento determinado, ya que estas se pueden agregar o quitar "en caliente" del servidor de aplicaciones. Además, el núcleo de la plataforma se encarga de gestionar las trazas que generan tanto las aplicaciones como el propio núcleo de la plataforma. Estos aspectos serán abordados en detalle a continuación.

III.3.1. Gestión de rutas

La plataforma BE se comunica con los operadores por medio de SAMS, la API definida por Sun Microsystems para la mensajería móvil. Gracias a esta API, que permite desarrollar controladores para distintos protocolos sin afectar al resto de la aplicación, la plataforma BE está preparada para recibir y enviar mensajes a través de distintos operadores móviles y protocolos al mismo tiempo. Para ello se define el concepto de ruta.

Una ruta está conformada por un controlador para SMS, un controlador para MMS, y un controlador para Notificaciones WAP, como se observa en la Figura 12. Estos controladores pueden implementar diferentes protocolos de comunicación con los SMSC, como IPX, SMPP, MM7 u otros. Para que la ruta sea válida, al menos se debe definir uno de estos tres controladores. Esto significa que podría existir una ruta que se encargue únicamente de enviar y recibir SMS, pero que no sea capaz de enviar MMS o Notificaciones WAP. Si la plataforma necesitara conectarse con distintos SMSC, sería suficiente con definir una ruta para cada SMSC, y en cada una de ellas definir los controladores que implementan los

protocolos con los cuales se va a comunicar con cada uno de los SMSC.

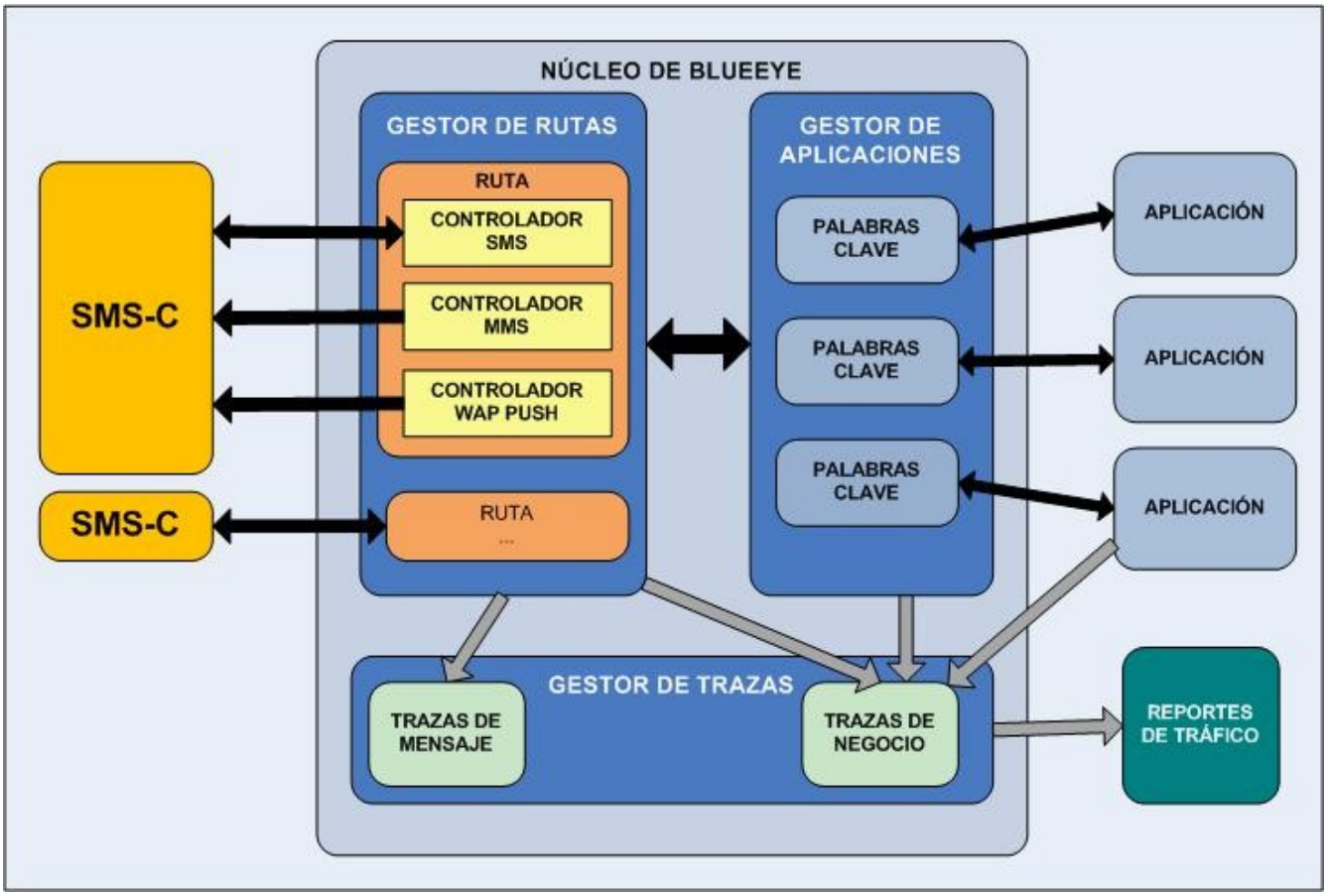

Figura 12: Principales tareas del núcleo de la plataforma.

La plataforma BE, al iniciarse, debe leer el fichero *sams-routes.xml*, en el cual se definen las rutas que puede utilizar la plataforma, y debe intentar conectarse con todas ellas. De fallar la conexión con alguna ruta, la plataforma la marcará como inactiva y reintentará conectarse con ella cada cierto tiempo, de forma tal que pueda recuperarse ante posibles fallos en la conexión. Estos fallos en la conexión deben ser detectados primeramente por el controlador, que es el responsable de notificarle a la plataforma la ocurrencia de este tipo de errores.

De esta forma, la plataforma BE puede gestionar varias rutas al mismo tiempo, por lo que puede estar conectada con varios centros de mensajería a través de distintos protocolos al mismo tiempo. Esto permite también implementar simuladores para realizar pruebas a la plataforma en ausencia de un

centro de mensajería real.

III.3.2. Gestión de aplicaciones

Una vez que la plataforma BE recibe un mensaje por alguna de sus rutas, se realiza un proceso de enrutamiento del mensaje a la aplicación que lo debe procesar y darle respuesta. Como se explicó anteriormente, cada aplicación, al registrarse en la plataforma, debe dar un conjunto de palabras clave y números cortos, que se van a usar para realizar el enrutamiento.

Se toma como palabra clave a la primera palabra del SMS. Si esta palabra está definida por alguna aplicación, el mensaje se enruta hacia esta. En caso contrario, el mensaje pasa a una cola de reintentos, y se tratará en reenviar cada cierto intervalo de tiempo, hasta que el mensaje caduca.

Existen aplicaciones que por su complejidad o características especiales necesitan un número corto para ellas solas. En este caso, el mensaje puede ser enrutado a partir del número corto al que fue enviado, y no a partir de la palabra clave. También puede existir una mezcla de número corto con palabra clave. Los pasos para realizar el enrutamiento son los siguientes:

- Se busca si existe alguna regla donde coincide la palabra clave y el número corto
- Se busca si existe alguna regla donde coincide la palabra clave para cualquier número corto.
- Se busca si existe alguna regla donde coincide el número corto para cualquier palabra clave.
- Se busca si existe alguna regla donde coincide cualquier palabra clave y cualquier número corto.
- En caso contrario, el mensaje pasa a la cola de reintentos.

Las palabras claves definidas por una aplicación se puede agrupar. Esto viene dado por el hecho de que una aplicación puede utilizar una o más palabras claves para realizar cierta acción (por ejemplo, para realizar la suscripción a un servicio se pueden usar las palabras “subs” y “subscribe”); y una o más palabras claves para realizar otra acción (por ejemplo, para borrar una suscripción se pueden usar las palabras “unsubs” y “unsubscribe”).

Sin las agrupaciones, la aplicación necesitaría preguntar cuál es la palabra clave para decidir cuál acción debe realizar. Si en algún momento se decidiera añadir una palabra clave nueva en el fichero de configuración de la aplicación, esto podría implicar cambios en el código fuente de la aplicación. Mediante las agrupaciones, la aplicación no necesita saber cuál fue la palabra clave, sino a qué grupo pertenece. Si en algún momento se añade una nueva palabra clave, basta con asociarla a uno de los grupos existentes, y no hay necesidad de hacer cambios en el código.

Por su parte, los comandos son la palabra que sigue a la palabra clave dentro del mensaje de texto. Los comandos no se utilizan para enrutamiento, sino para uso interno de la aplicación, y su definición es opcional. También se puede agrupar, como mismo se hace con las palabras clave, y se utilizan en aplicaciones más complejas, para facilitar y simplificar la implementación. Por ejemplo, una aplicación para el parte del tiempo puede definir la palabra clave “tiempo” y los comandos “occ” y “occidental” para el parte del tiempo de la región occidental; los comandos “ori” y “oriental” para el parte del tiempo de la región oriental; agrupando cada conjunto de comandos en grupos distintos. Así la aplicación no se tiene que preocupar por analizar la cadena en busca de las palabras que indiquen la región del país, porque al definir los comandos el núcleo de la plataforma se encarga de hacerlo.

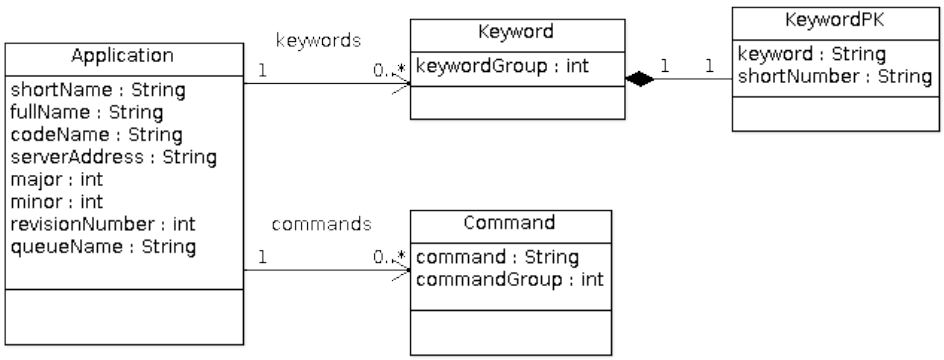

Figura 13: Diagrama de clases de la aplicación con sus palabras clave y comandos.

En la Figura 13 se muestra el diagrama de clases que representa a cada aplicación con sus palabras clave y sus comandos. Las clases, al ser utilizadas por JPA para realizar la persistencia en base de datos, deben seguir las reglas que impone esta API. Por ello es necesario definir la clase *KeywordPK* para representar la llave primaria compuesta de la clase *Keyword*. Como el enrutamiento se realiza utilizando la primera palabra del mensaje más el número corto al cual fue enviado, esta combinación tiene que ser única para cada aplicación.

Cada aplicación puede tener varias o ninguna palabra clave, y varios o ningún comando. A una aplicación que no defina ninguna palabra clave, la plataforma no le podrá enviar mensajes entrantes, aunque la aplicación sí podría enviar mensajes a los usuarios haciendo uso de las facilidades de la plataforma.

Una vez que la plataforma decide a qué aplicación le será enrutado el mensaje, este se almacena en una cola de mensajes en espera de que la aplicación lo procese. Cada aplicación define su propia cola de mensajes, y este proceso se realiza de forma asíncrona, por lo cual la plataforma no tiene que esperar que el mensaje sea procesado por la aplicación, sino que después de dejarlo en la cola de mensajes adecuada, está lista para procesar el siguiente mensaje.

Las aplicaciones pueden enviar mensajes de salida al usuario. Para esto existe en el núcleo de la plataforma una cola que procesa los mensajes de salida de forma asíncrona. Estos mensajes de salida son enviados a través de la misma ruta por la que llegó el mensaje que generó la respuesta, o a través de una ruta predeterminada si el mensaje no fue generado como respuesta a un mensaje entrante.

Los mensajes que no se pueden procesar, tanto los de entrada como los de salida, se quedan en una cola especial definida para estos propósitos. Un mensaje puede caer en esta cola producto de que la aplicación a la que se enrutó no se encuentra desplegada, aunque las palabras clave están definidas en el núcleo, o porque la palabra clave no existe, o porque existió algún fallo en la comunicación con el núcleo o la conexión con el centro de mensajería. Estos mensajes que no se han podido procesar tienen un tiempo de espiración definido en un fichero de configuración de la plataforma, después del cual se eliminan completamente.

III.3.3. Registro de aplicaciones

Como se explicó anteriormente, en el momento en que la aplicación se despliega en el servidor de aplicaciones, se ejecuta la secuencia de arranque asociada a la aplicación, que está implementada dentro del MBean que se ejecuta al desplegarse la aplicación. Es en este momento que la aplicación se registra en la plataforma BE.

En la Figura 14 se muestra un diagrama con las principales clases envueltas en la secuencia de arranque de la plataforma y sus aplicaciones. En ellas se encuentran la clase *SystemProperties*, que define las propiedades de cada aplicación a partir de un fichero de propiedades que se lee al inicio, y la clase *StartupSequence*, que define la secuencia de inicio y finalización de cada aplicación, incluyendo su registro en la plataforma.

Entre los parámetros que registra cada aplicación en la plataforma se encuentran su nombre, el número de versión y una descripción de la aplicación. Estos parámetros dan información a los desarrolladores y administradores de la plataforma BE acerca de las aplicaciones desplegadas.

El número de versión de la aplicación contiene un número de versión mayor, asociado a los hitos (o *milestones*) fundamentales del proyecto, un número de versión menor asociado a pequeños cambios y arreglo de errores menores dentro del hito, y el número de revisión dentro del repositorio. Además, a cada hito está asociado un sobrenombre del producto (o *code-name*).

Otros parámetros que se registran, están orientados al funcionamiento interno de la plataforma BE. Son ellos el nombre corto, el nombre de la cola de entrada, la dirección del servidor donde se encuentra desplegada y el nombre del archivo JAR que contiene la aplicación.

En específico, el nombre corto sirve como identificador de la aplicación y tiene que ser único. En cualquier lugar de la plataforma donde se quiera hacer referencia a una aplicación, se deberá hacer usando este nombre corto. También en el caso de aplicaciones que manejan otras aplicaciones, como lo hace la aplicación de subscripciones, la forma de referirse a estas otras aplicaciones es mediante el nombre corto.

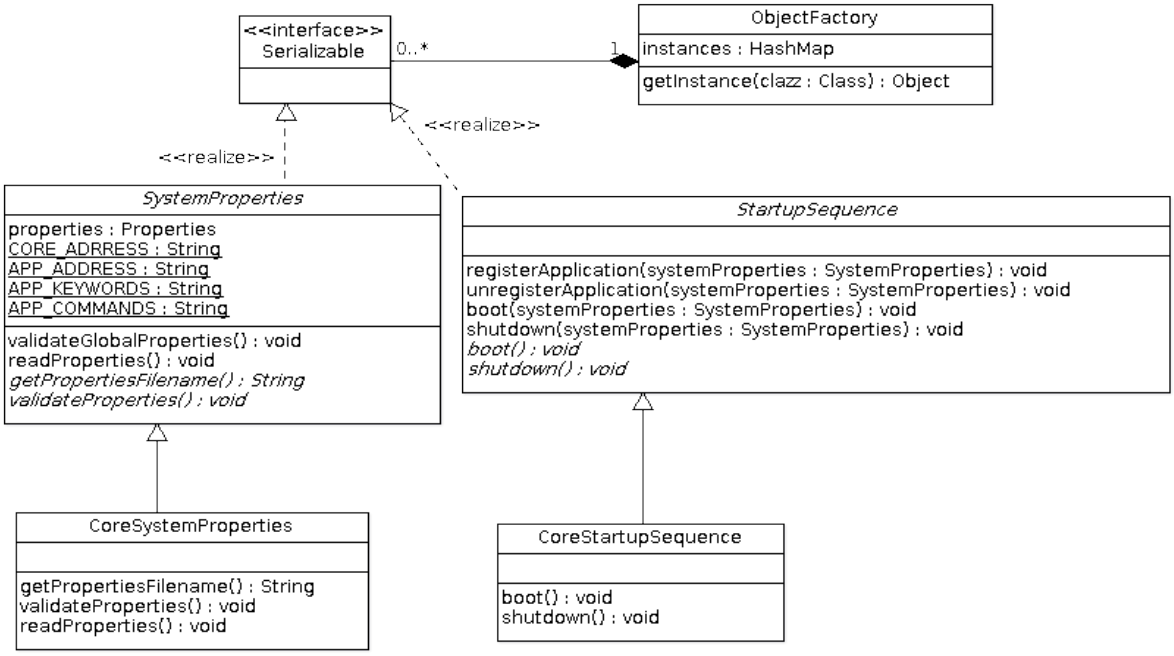

Figura 14: Diagrama de las clases involucradas en la secuencia de inicio de la plataforma.

El nombre de la cola de entrada es el parámetro mediante el cual la aplicación registra la cola por donde va a estar escuchando los mensajes entrantes. Este parámetro, en conjunto con la dirección del servidor donde está desplegada la aplicación, son utilizados por el núcleo de la plataforma para enrutar los mensajes entrantes a las distintas aplicaciones.

Finalmente, parámetro con el nombre del archivo JAR que contiene la aplicación actualmente no se utiliza, pero se mantiene con el objetivo de agregar nuevas funcionalidades en el futuro.

Hay que señalar que el núcleo de la plataforma se implementó usando las mismas funcionalidades comunes al resto de las aplicaciones de la plataforma, por tanto las secuencias de inicio y finalización son iguales. Cada aplicación, incluyendo al núcleo, define sus propiedades específicas, así como métodos de inicio y finalización específicos que son llamados después que los métodos comunes se ejecutan.

De la misma forma que las aplicaciones se registran en la plataforma, deben eliminarse en el momento en que la aplicación es quitada del servidor de aplicaciones. En este caso se ejecuta una secuencia de apagado, en donde la aplicación le indica al núcleo de la plataforma que debe borrarla de la base de datos. Asimismo, las secuencias de inicio o apagado de las aplicaciones se pueden utilizar para usos específicos, como por ejemplo iniciar o apagar un temporizador (o *timer*).

Después que la aplicación se registra en la plataforma, se lee el fichero de configuración de la aplicación. Cada aplicación debe definir un fichero de propiedades en donde se definen propiedades a nivel de la plataforma, y propiedades a nivel de la aplicación. Las propiedades a nivel de la plataforma son obligatorias para todas las aplicaciones, aunque de no definirse la plataforma asumiría valores por defecto para cada una de ellas. Por su parte, cada aplicación define sus propiedades de acuerdo a sus necesidades, y define los valores por defecto para cada una de ellas. El núcleo de la plataforma, implementado utilizando el mismo API común para el resto de las aplicaciones, puede verse a sí mismo como una aplicación y tiene sus propiedades específicas a nivel de aplicación.

Las propiedades a nivel de la plataforma son las siguientes:

- Dirección IP donde se encuentra desplegado el núcleo de la plataforma.
- Dirección IP donde se encuentra desplegada la aplicación.
- Palabras clave, números cortos y comandos definidos por la aplicación (este punto se explicará más adelante en detalle).

Las propiedades a nivel de aplicación para el caso del núcleo son las siguientes:

- Intervalo de reintento de mensajes que han fallado al enviarse.
- Cantidad máxima de reintentos antes de desechar estos mensajes.
- Intervalo de reconfiguración de conexiones con los SMSC que han fallado.
- Tiempo de vida de las trazas.
- Cantidad de mensajes que pueden salir a través del centro de mensajería por segundo.

III.3.4. Gestión de trazas

Por medio de las trazas se pueden acumular datos acerca del funcionamiento de la plataforma, o de errores que hayan aparecido durante la ejecución de algún proceso.

La gestión de trazas es realizada por el núcleo de la plataforma, aunque cualquier aplicación puede dejar trazas de su funcionamiento. El envío de trazas al núcleo se realiza como un proceso asíncrono a través de una cola. Se realiza de esta forma para evitar que la traza se pierda si existiera algún fallo en la comunicación con el núcleo, ya que se guardaría en una base de datos manejada por el servidor de aplicaciones como parte del servicio de manejo de mensajes de J2EE conocido como JMS. Además, de esta forma la aplicación no pierde tiempo esperando por que el núcleo procese la traza, y puede continuar ejecutándose.

Las trazas se dividen en trazas de negocio y trazas de mensaje. Las trazas de negocio tienen como objetivo guardar todos los acontecimientos que las aplicaciones consideren importantes, como pueden ser entrada y salida de mensajes, conexión o desconexión con el centro de mensajería, inicio o apagado de una aplicación, u ocurrencia de errores. Cada aplicación es responsable de generar sus propias trazas. Las trazas de negocio constan de:

- Fecha
- Nombre de la aplicación que la genera.
- Tipo de traza (de entrada o salida de mensajes, de conexión o desconexión, u otros).
- Texto de la traza.

Por otra parte, las trazas de mensaje son gestionadas únicamente por el núcleo de la plataforma. Estas trazas difieren además de las trazas de negocio en que su único objetivo es el de dejar un registro único por cada mensaje que entra y sale de la plataforma. Las trazas de mensaje constan de:

- Fecha.
- Remitente del mensaje.

- Destinatario del mensaje.
- Identificador del mensaje.
- Texto del mensaje.
- Tipo de traza (de entrada de mensaje o de salida de mensaje).

En las últimas versiones de la plataforma también se han agregado los siguientes campos:

- Nombre de la aplicación que recibió o envió el mensaje.
- Precio del mensaje.

A partir de las trazas de mensaje se pueden realizar reportes que permitan conocer la cantidad de mensajes que se recibieron y enviaron, conocer si algún cliente se quedó sin respuesta a alguna solicitud, y comparar los datos de la plataforma con los del centro de mensajería, con el objetivo de buscar errores y definir las ganancias de cada parte por concepto de tráfico. Asimismo se pueden utilizar para implementar mecanismos de cobro revertido, ya que por medio de estos reportes el SMSC puede conocer qué usuarios han utilizado los servicios de la plataforma y por tanto descontarle del saldo el precio del mensaje.

Los reportes se pueden obtener manualmente desde la interfaz de administración y monitoreo, o de forma automática a través de ficheros en formato CSV (valores separados por comas) que la plataforma genera periódicamente.

III.4. SDK de BlueEye

La plataforma BE define un SDK que puede ser utilizado por las aplicaciones para facilitar el acceso a las distintas funcionalidades de la plataforma. El propio núcleo de la plataforma fue implementado utilizando las funciones comunes del SDK, y su comportamiento y ciclo de vida es similar al de cualquier otra aplicación.

Este SDK está dividido en funciones para aplicaciones, funciones para mensajería, funciones para trazas, y otras funciones.

Las funciones para aplicaciones son las siguientes:

- Obtener una aplicación a partir de su nombre corto.
- Obtener el número de versión de una aplicación, conformado por el número de versión mayor, el número de versión menor, y el número de revisión.
- Reconfigurar las conexiones con los SMSC. Con esta funcionalidad se cierran todas las conexiones existentes y se vuelve a intentar conectar, tanto las que estaban activas como las inactivas.
- Obtener las rutas activas e inactivas. Con esta funcionalidad se obtienen listados de rutas activas e inactivas de acuerdo a si existen o no conexiones activas con algún centro de mensajería.

Las funciones para mensajería son las siguientes:

- Obtener la palabra clave y el resto del texto a partir de un mensaje de texto. Estas funciones se encuentran en desuso ya que actualmente la palabra clave es separada como parte del pre-procesamiento realizado en el núcleo de la plataforma. No obstante la función se mantiene por compatibilidad con versiones anteriores.
- Obtener un arreglo de todas las palabras que conforman el texto del mensaje.
- Obtener una nueva instancia de un mensaje tanto de texto, como multimedia o de notificación. Estas clases relacionadas con mensajes no se pueden instanciar directamente, sino que se debe pedir una instancia a la sesión creada en la plataforma.
- Enviar un mensaje de salida. Este mensaje será insertado en la cola de salida y procesado de forma asíncrona por le núcleo de la plataforma.

Las funciones asociadas con trazas son las siguientes:

- Agregar traza de negocio.
- Agregar traza de mensaje.

Las otras funciones de utilidades son las siguientes:

- Obtener la instancia del servidor de aplicaciones. De esta forma se puede acceder a las características propias del servidor de aplicaciones que se está utilizando, como son las propiedades para acceder al JNDI, los directorios de despliegue y configuración, que ya fueron explicadas anteriormente.
- Obtener las propiedades de la aplicación. Estas propiedades son las que se leen al iniciarse la aplicación del fichero de configuración que define cada aplicación.

En la Figura 15 se muestra el diagrama de las clases del SDK de BE.

No obstante, estas funciones no suficientes. El SDK de BE brinda un conjunto de interfaces y clases abstractas que cada nueva aplicación debe implementar, ya que resultan imprescindibles para el funcionamiento correcto de la aplicación.

Estas clases abstractas definidas en el núcleo de la plataforma definen funcionalidades que serán más tarde implementadas en cada aplicación, y al mismo tiempo implementan funcionalidades comunes a todas las aplicaciones, de forma tal que no exista necesidad de repetir el mismo código en cada una de las aplicaciones.

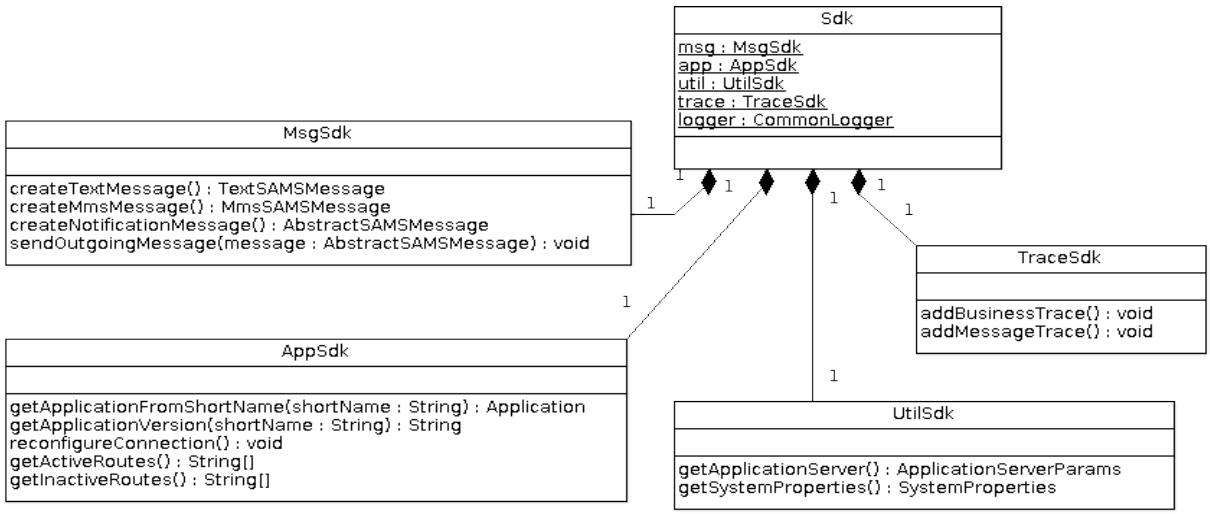

Figura 15: Diagrama de clases del SDK.

De esta forma, resulta imprescindible que sean implementadas para cada aplicación:

- Una cola para recibir los mensajes entrantes, definida a través de un Bean orientado a Mensajes; aunque las funcionalidades fundamentales de estas colas se encuentran definidas en el núcleo de la plataforma.
- Un Bean de Sesión que implemente el método principal de la aplicación mediante el cual se procesan los mensajes.
- Una clase que defina secuencias de inicio y apagado de la aplicación, en caso de que sea necesario realizar estas operaciones. Las secuencias de inicio y apagado por defecto, que incluyen el registro de las aplicaciones en la plataforma, se encuentran definidas a nivel del núcleo.
- Una clase que defina las propiedades específicas de cada aplicación, tal y como serán leídas del fichero de configuración. Las propiedades comunes se leen en una clase definida en el núcleo de la plataforma.
- El nombre del fichero de configuración de la aplicación.
- Los datos específicos de la aplicación como nombre corto, descripción, número de versión.

III.4.1. Desarrollo y despliegue de una aplicación BlueEye

A continuación se describirá el desarrollo y despliegue de una aplicación utilizando el SDK de BE.

III.4.1.1. Desarrollo de la aplicación

Para desarrollar una aplicación usando el SDK de BE basta con implementar las funcionalidades básicas de una aplicación BE, utilizando las clases e interfaces que ofrece el SDK. Estas funcionalidades básicas son: definir un conjunto de propiedades en un fichero de configuración que será leído en el momento en que la aplicación se inicia, y validar estas propiedades; definir las

operaciones que realizará la aplicación cuando se inicia o se finaliza, si es que va a realizar alguna; definir la cola asíncrona para la recepción de los mensajes que le envía el núcleo de BE, y definir el método para procesar estos mensajes.

Propiedades de la aplicación.

Como se explicó anteriormente, cada aplicación debe definir un fichero de configuración, en el cual se definen tanto las propiedades comunes de todas las aplicaciones de la plataforma, como las propiedades específicas de la aplicación.

En el momento en que la aplicación se inicia, se lee este fichero y se cargan en memoria las propiedades definidas en él. Este proceso no es necesario realizarlo explícitamente, ya que se realiza como parte de las funcionalidades comunes definidas en la clase *SystemProperties* del SDK, por lo que la nueva aplicación solamente deberá extender esta clase y especificar el nombre de su fichero de configuración redefiniendo el método *getPropertiesFilename*.

Además, como parte de las funcionalidades del SDK, se validarán las propiedades comunes a todas las aplicaciones, como pueden ser la dirección IP del servidor donde está desplegado el núcleo de BE, la dirección IP del servidor donde está desplegada la aplicación, las palabras clave y los comandos de la aplicación. Cada aplicación, por su parte, es responsable de validar sus propiedades específicas. Esto se hace redefiniendo el método *checkDefaults*. El proceso de validación es importante no solo porque garantiza que los valores de las propiedades sean válidos, sino porque permite también definir valores por defecto en caso de que estén definidas en el fichero de configuración.

Inicio y apagado de la aplicación.

Todas las aplicaciones cuentan con secuencias de inicio y apagado, definidas en el SDK de BE en la clase *StartupSequence*. Las secuencias comunes a todas las aplicaciones tienen como objetivo registrar la aplicación en la base de datos, así como sus palabras clave y sus comandos. Cuando la aplicación finaliza, estos datos son borrados de la base de datos, como parte de la secuencia de apagado.

Cada aplicación debe extender de la clase *StartupSequence* y puede implementar en los métodos *boot* y

shutdown sus acciones específicas para el inicio y apagado, como podrían ser inicializar un temporizador (*timer*), o inicializar alguna tabla de la base de datos con valores predefinidos, u otros.

Además, cada clase debe redefinir el método *getApplication*, donde se especifican los datos propios de la aplicación como el nombre, el número de versión y otros. Este método es invocado desde el SDK a la hora de tomar los datos de la aplicación que se guardarán en la base de datos.

Cola de entrada de mensajes.

Cada aplicación debe definir su propia cola asíncrona para la recepción de los mensajes que le son enrutados desde el núcleo de la aplicación. Aunque la funcionalidad de esta cola es común a todas las aplicaciones, cada una debe definir su propia clase ya que esta cola está implementada como un Message-Driven Bean y necesita metainformación en forma de anotaciones Java que deben ser escritas explícitamente en la nueva clase. No obstante, las funcionalidades comunes están escritas en la clase *ApplicationMessageListener* del SDK de BE, por lo cual basta con extender de esta clase y escribir la metainformación propia de la nueva clase.

Procesamiento de mensajes.

Finalmente, la aplicación debe definir una clase para el procesamiento de los mensajes. Esto se realiza implementando la interfaz EntryPoint del SDK de BE y definiendo el método *processMessage*. Este método recibe como parámetro el mensaje de texto que le fue enrutado a la aplicación. La aplicación es la encargada de procesar el mensaje y dar una respuesta, utilizando las funcionalidades que se brindan en la clase *Sdk* y sus propias funcionalidades, con el fin de dar una respuesta apropiada al usuario.

Vale aclarar que la aplicación no devuelve ningún objeto, y que es su responsabilidad enviar un mensaje de respuesta al usuario. Esto da la libertad suficiente para que se implementen aplicaciones que no den mensajes de respuesta, o que generen mensajes para otros usuarios si fuera necesario. El envío de mensajes desde la aplicación se realiza de forma asíncrona, utilizando una función definida en la clase *Sdk*, como se explicó anteriormente.

De forma general, los pasos a seguir para procesar un mensaje serían los siguientes:

- Agregar una traza de negocio indicando que se reibió un mensaje, con la función *Sdk.trace.addBusinessTrace*.

- Obtener el texto del mensaje, utilizando la función getText de la clase *TextSAMSMessage*; o el texto sin la palabra clave, utilizando la función *getData* de la clase *TextSAMSMessage*.

- Procesar el texto obtenido y generar un mensaje de respuesta.

- Crear un mensaje de texto de respuesta, utilizando la función *Sdk.msg.createTextMessage*.

- Enviar el mensaje de respuesta, utilizando la función *Sdk.msg.sendOutgoingMessage*.

- Agregar una traza de negocio indicando que se envió un mensaje, con la función *Sdk.trace.addBusinessTrace*.

III.4.1.2. Despliegue de la aplicación

El despliegue de una aplicación BE se realiza sobre un servidor de aplicaciones Java, como puede ser el Jboss AS. Como la plataforma BE es distribuida, el despliegue de las aplicaciones se puede realizar lo mismo en el mismo nodo del núcleo de BE, o en un nodo diferente.

Los requerimientos mínimos de hardware utilizados para el despliegue en modo de prueba son un único servidor, con procesador Intel Pentium 4 a 2.80 GHz y 1 Gb de memoria RAM. Los requerimientos de software son los siguientes:

- Sistema operativo GNU/Linux kernel 2.6.x o Microsoft Windows XP.

- Plataforma Java versión 5.

- Servidor de aplicaciones Jboss AS 4.2.1

- Servidor de bases de datos MySQL 5.

Antes de realizar el despliegue es necesario escribir los ficheros de configuración, tanto de los

controladores de SAMS, como del núcleo de la plataforma y de cada una de las aplicaciones que se van a desplegar. Para el caso particular del servidor de aplicaciones Jboss AS, los ficheros de configuración se deberán copiar en la carpeta *conf/* del servidor de aplicaciones. Los ficheros de configuración necesarios son los siguientes:

- El fichero *sams-routes.xml*, utilizado por el núcleo de la plataforma para definir las rutas. Cada ruta debe especificar un controlador para SMS, MMS y notificaciones WAP.
- Un fichero de configuración por cada uno de los controladores especificados en *sams-routes.xml*.
- El fichero *blueeye.core.properties*, con la configuración inicial del núcleo.
- Un fichero *.properties*, similar al del núcleo, por cada aplicación que se despliegue, con la configuración específica de cada aplicación.

Para realizar el despliegue, es necesario tener en cuenta las librerías de las cuales depende la plataforma y que no se encuentran en el servidor de aplicaciones. Tal es el caso de SAMS, que pese a ser un estándar definido por Sun Microsystems como parte de Java EE, aun no es soportado oficialmente por los servidores de aplicaciones.

Para realizar el despliegue del núcleo de la plataforma, es necesario desplegar los siguientes paquetes, tal y como se muestra en la Figura 16:

- El paquete *sams-jsr212.jar*, que es la base de la implementación de SAMS.
- El paquete *sams-spi-common.jar*, con las funcionalidades comunes para los controladores SAMS.
- Los paquetes con los controladores SAMS para los distintos protocolos que se deseen utilizar, con todas sus dependencias. Por ejemplo, el paquete *sams-smpp.jar* con el controlador para el protocolo SMPP.

- El paquete *blueeye-core.jar*, que contiene tanto el SDK como el núcleo de la plataforma BE.
- El paquete *blueeye-webadmin.war*, que contiene la interfaz web de administración.

Para desplegar aplicaciones en un nodo diferente al del núcleo, basta con desplegar los siguientes paquetes, tal y como se muestra en la Figura 14:

- El paquete *sams-jsr212.jar*. Aunque las aplicaciones no se encargan de la gestión de mensajes, sí utilizan algunas clases definidas en este paquete.
- El paquete *blueeye-sdk.jar*, con las funcionalidades comunes a todas las aplicaciones.
- Las dependencias propias de cada aplicación, si existieran.
- Los paquetes específicos de cada una de las aplicaciones.

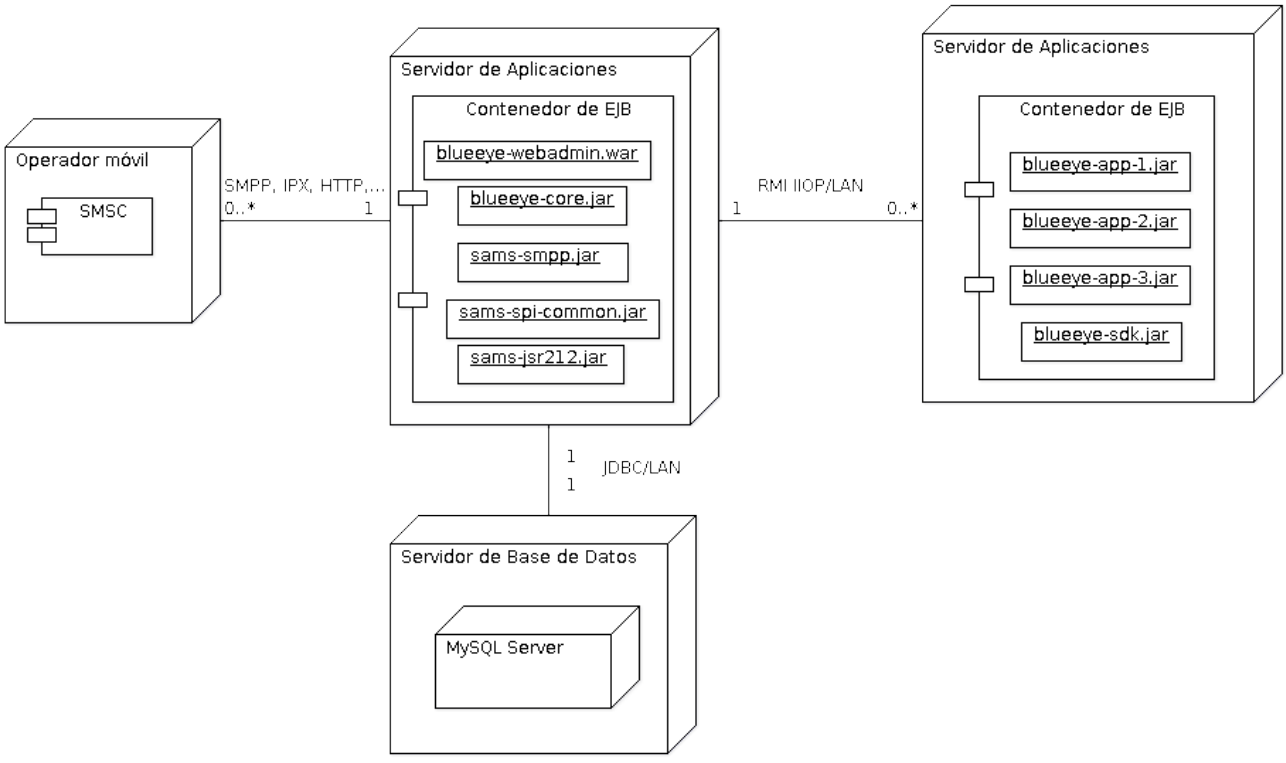

Figura 16: Diagrama de despliegue de la plataforma.

III.4.2. Primeras aplicaciones implementadas sobre la plataforma BlueEye

A continuación se describirán las primeras aplicaciones que han sido implementadas y puestas en funcionamiento sobre la plataforma BE. Entre estas, se encuentran la aplicación de eco, que se implementó a modo de prueba y como guía para implementar otras aplicaciones más complejas; la aplicación para el manejo de subscripciones, que funciona como una especie de meta-aplicación, ya que su acción incide sobre otras aplicaciones desplegadas; aplicaciones para eventos públicos como la aplicación para la cartelera del Festival de Cine, y la aplicación para el Clásico Mundial de Béisbol; y otras de interés general como la aplicación para el parte del tiempo, y la aplicación para conocer la ubicación de las embajadas y consulados.

III.4.2.1. Aplicación de eco

El objetivo fundamental de esta aplicación es servir de prueba para la plataforma, y como guía para el desarrollo de otras aplicaciones. De manera general, cada vez que se introducen cambios en el núcleo de la plataforma, que puedan incidir directamente en las aplicaciones, primeramente se prueban en la aplicación de eco.

Esta aplicación recibe un mensaje de texto que debe tener la siguiente estructura:

- **echo** [texto]

y devuelve como respuesta otro mensaje con el texto recibido.

También se ha adaptado para que funcione conjuntamente con la aplicación de subscripciones, de forma tal que se puede suscribir al eco (y también cancelar la suscripción) de la siguiente forma:

- **subscribe** echo
- **unsubscribe** echo

En el caso de las subscripciones, la aplicación de eco envía periódicamente a sus subscriptores un mensaje con un texto predeterminado, mediante el cual se puede probar el funcionamiento de la

aplicación de subscripciones.

III.4.2.2. Aplicación de subscripciones

La aplicación de subscripciones tiene como objetivo manejar las subscripciones a todas las aplicaciones que se encuentren desplegadas en un momento determinado, y que soportan esta funcionalidad. Una primera aproximación al tema de las subscripciones podría indicar que cada aplicación que lo necesite, implemente su propio sistema de subscripciones. Pero esto implicaría rescribir parte del código en cada una de ellas, e iría contra la política seguida en la plataforma, de que cada aplicación se encargue de hacer únicamente lo que le compete.

Por ello se decidió que el manejo de subscripciones debía realizarse en una aplicación aparte, destinada solo a este fin. Esta aplicación debe guardar una lista de subscriptores por cada una de las aplicaciones desplegadas, y es la encargada de levantar una notificación cuando alguna de las aplicaciones debe enviar algún mensaje a sus subscriptores.

Como se ha explicado, las aplicaciones son independientes y no tienen conocimiento del resto de las aplicaciones que se encuentran desplegadas en un momento determinado. Solamente el núcleo de la plataforma tiene conocimiento de esto. Por tanto, la aplicación de subscripciones envía las notificaciones primeramente al núcleo de la plataforma. En esta notificación se incluyen, entre otros parámetros, el subscriptor y el nombre de la aplicación a la que está subscrito. El núcleo de la plataforma, por su parte, es el encargado de notificar a la aplicación de que se ha levantado un evento de suscripción, y por tanto debe realizar alguna acción al respecto.

Por su parte, las aplicaciones deben implementar un método de una interfaz definida en el núcleo de la plataforma, para poder recibir los eventos de suscripción.

La sintaxis del mensaje de texto para solicitar o cancelar una suscripción es la siguiente:

- **subscribe** [nombre de la aplicación]
- **unsubscribe** [nombre de la aplicación]

en donde el nombre de la aplicación debe corresponder a alguna aplicación desplegada y que tenga soporte para las subscripciones. Como respuesta, la aplicación de subscripciones envía al usuario un mensaje de texto con una confirmación de la suscripción, o de la cancelación de la misma.

III.5. Compuerta para dar servicio de mensajería a terceros

El objetivo fundamental de la compuerta (o *gateway*) es dar servicio de mensajería a terceros, por ejemplo, empresas que necesiten enviar notificaciones por medio de mensajes de texto, usando para ello la vía Internet.

Para estos fines se implementó una aplicación cuyo objetivo es ofrecer una interfaz a terceros, usando el protocolo HTTP/HTTPS. Esta aplicación debe controlar los usuarios que se conectan a ella intentando enviar un mensaje. Cada usuario debe identificarse por medio de un nombre de usuario y una contraseña, unido a la dirección IP desde donde se está intentando realizar la conexión. Otra restricción a los usuarios es la cantidad de mensajes por hora, con lo cual se evita que un usuario sobrecargue la plataforma de mensajes.

En la Figura 17 se muestra la interacción de terceras aplicaciones con la aplicación de compuerta, por medio del protocolo HTTP/HTTPS, y la aplicación de compuerta con el núcleo de la plataforma.

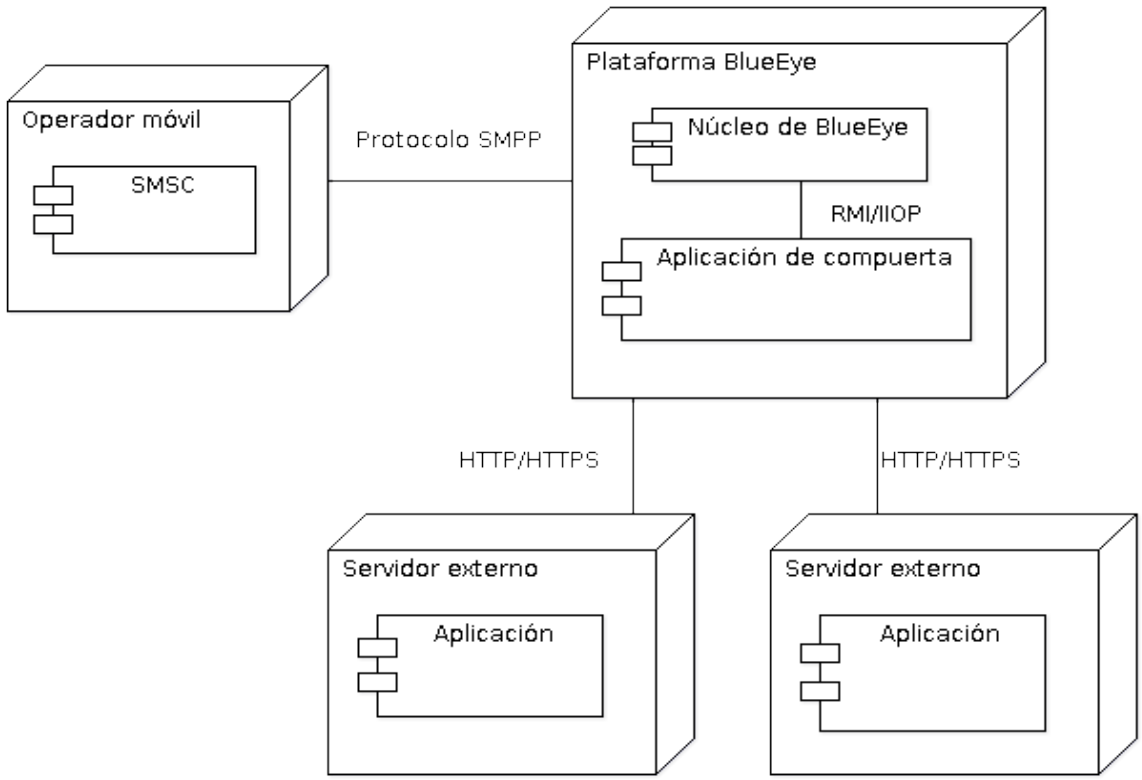

Figura 17: Interacción de terceras aplicaciones con la aplicación de compuerta.

A su vez, los usuarios pueden acceder también a los mensajes de entrada, en caso de que les sean enviados. Para esto existen dos estrategias fundamentales:

- Que cada usuario provea una dirección IP por la cual estará escuchando, y a donde se le enviarán los mensajes entrantes, o
- Que cada usuario acceda cada cierto tiempo a un buzón desde donde pueda descargar los mensajes que le han sido enviados.

Igualmente los usuarios pueden averiguar el estado de los mensajes que ha enviado, por lo cual la aplicación debe llevar trazas acerca del estado de los mensajes. El usuario puede cancelar los mensajes

que no se hayan podido enviar por motivo de algún fallo, aunque si no los cancela la aplicación intentará enviarlos mientras no se haya vencido su tiempo de expiración.

De esta forma terceras empresas pueden acceder al servicio de mensajes de texto de la plataforma, sin tener necesariamente que implementar un servicio siguiendo la API que brinda la plataforma.

III.6. Monitoreo y administración de BlueEye

La interfaz para el monitoreo y administración de la plataforma da la posibilidad de conocer al momento lo que está sucediendo en el sistema, cómo se comporta la entrada y salida de mensajes, cuántos mensajes hay en las colas, así como cuáles rutas con los SMSC se encuentran conectadas y cuáles no, realizar la reconexión con los SMSC, y obtener reportes mensuales, trimestrales y anuales del tráfico de mensajes por la plataforma.

La plataforma define una serie de funciones para conocer estos datos acerca de su funcionamiento. Estas funciones se pueden utilizar lo mismo desde una interfaz web que de una interfaz de escritorio; y actualmente se han implementado ambos tipos de interfaz para distintas versiones de la plataforma.

Las funciones definidas por la plataforma para el monitoreo y administración son las siguientes:

- Obtener las rutas con los SMSC que se encuentran activas e inactivas.
- Reconfigurar las conexiones con los SMSC.
- Obtener un listado de todas las colas (tanto de entrada como de salida) activas y de la cantidad de mensajes que se encuentran en ellas.
- Obtener un listado de las aplicaciones desplegadas.
- Obtención de trazas, tanto de mensajes como de negocio. Estas trazas se pueden obtener según la fecha, o el número del que envió o recibió el mensaje. También se pueden obtener con paginación, para ayudar a la visualización de los listados.
- Obtención de trazas en forma de fichero CSV. Con esta función se pueden obtener las trazas en

un formato que se puede importar desde cualquier aplicación ofimática (Microsoft Office, OpenOffice) con el objetivo de obtener estadísticas acerca del tráfico de la plataforma.

III.7. Pruebas realizadas a la plataforma

Con el objetivo de comprobar el funcionamiento correcto de la plataforma, se realizaron varios tipos de pruebas. Entre estas se pueden mencionar pruebas de unidad, de estrés, de recuperación, de integración y de sistema.

Las pruebas de unidad se definieron y realizaron basándose primeramente en los requerimientos de la plataforma, así como en los defectos detectados, tal y como se expone en la metodología de desarrollo XP. Estas pruebas se desarrollaron de forma paralela a la implementación de la plataforma, para asegurar que las funcionalidades fundamentales se comportaban de la forma esperada. Entre estas se encuentran las funcionalidades definidas para el registro de aplicaciones, gestión de trazas a bajo nivel, así como las funcionalidades de la interfaz de administración y monitoreo, y las funcionalidades definidas en el SDK.

Las pruebas de integración estuvieron divididas en dos partes. Primeramente se probó el funcionamiento del núcleo de la plataforma con varias aplicaciones desplegadas. Se comprobó el registro de las aplicaciones en la plataforma, la ejecución correcta de las secuencias de inicialización y finalización de las aplicaciones, y el registro correcto de las palabras clave y comandos de cada aplicación.

En su segunda parte, las pruebas de integración se centraron en comprobar la integración de la plataforma y sus aplicaciones con el SMSC. En este caso se comprobó la creación correcta de la ruta con el SMSC, la comunicación de la plataforma con el controlador que implementa el protocolo del SMSC por medio de SAMS, y el correcto funcionamiento del envío y recepción de mensajes a través del SMSC.

Durante las pruebas de estrés se sometió a la plataforma a un tráfico ininterrumpido de mensajes, con el objetivo de verificar su funcionamiento ante un gran flujo de mensajes. Estas pruebas de estrés se

realizaron utilizando un simulador de envío y recepción de mensajes, y no directamente con el SMSC, ya que podría saturar el tráfico del centro de mensajería. Como parte de las pruebas de estrés se realizaron los siguientes controles:

- Enviar de forma ininterrumpida, a través de un simulador, 10, 100, y 1000 mensajes.
- Realizar la prueba anterior, utilizando dos simuladores simultáneamente, cada uno en un servidor distinto.
- Enviar mensajes de forma ininterrumpida cada cierto intervalo de tiempo, y durante 2 días.

En todas estas pruebas la plataforma debía demostrar su capacidad de procesar los mensajes recibidos y responder a todas las solicitudes, lo cual realizó de forma satisfactoria.

Se realizaron pruebas de recuperación para comprobar la robustez de la plataforma ante fallas de hardware. Se comprobó que al apagar el sistema de forma imprevista algunas tuplas en la base de datos pueden quedar en un estado inconsistente. Estas tuplas no están relacionadas directamente con el modelo de datos definido por la plataforma, pero sí con el funcionamiento de algunos componentes de servidor de aplicaciones, como los temporizadores (o *timers*).

En otra prueba de recuperación se comprobó que, si se perdía la comunicación con el SMSC, la plataforma no tenía forma de intentar recuperar la conexión con este a través del controlador. Se comprobó además que la comunicación entre el controlador y la plataforma no era del todo correcta, ya que el controlador no enviaba ninguna notificación a la plataforma en caso de desconexión con el SMSC.

Finalmente, se sometió a la plataforma a pruebas de sistema, conectada directamente al SMSC y con varias aplicaciones de prueba desplegadas, con el objetivo de comprobar su comportamiento en un entorno real y muy cercano al entorno en el que deberá desplegarse, y detectar posibles dificultades. Como resultado de estas pruebas se detectaron problemas como el crecimiento de los archivos de *logs* del servidor de aplicaciones, así como la mala respuesta de la plataforma ante la entrada de mensajes con caracteres “extraños”, como es el caso de caracteres latinos (como letras con tildes, eñes), y con

caracteres en formato UTF-16, ya que eran tratados como caracteres en formato UTF-8.

A partir de estos problemas detectados se planificaron y ejecutaron un conjunto de acciones con el objetivo de mejorar en lo posible el funcionamiento de la plataforma. Estas son las siguientes:

- Implementación de un *script* que inicia el servidor de aplicaciones y que se encarga previamente de limpiar los *logs* antiguos, y arreglar posibles problemas que pudieran surgir en la base de datos a partir de un apagado incorrecto del servidor de aplicaciones.
- Preparar tanto la plataforma como los controladores que se conectan directamente con el centro de mensajería para detectar fallos en la conexión y recuperarse automáticamente de estos problemas.
- Sustituir dentro del texto de los mensajes entrantes los caracteres latinos por equivalentes dentro de la codificación US-ASCII (por ejemplo, á por a, ñ por n) con el objetivo de mejorar la respuesta de la plataforma ante estos casos.
- Mejorar en el controlador la conversión de caracteres codificados en formato GSM 7 bit y en formato UTF-16 a formato UTF-8.

III.8. Resultados

Como resultado de este trabajo se obtuvo una plataforma capaz de gestionar aplicaciones que implementan servicios de valor agregado para dispositivos móviles. Específicamente, se obtuvieron los siguientes resultados:

- La plataforma es extensible, presenta una arquitectura modular que permite que se le agreguen o quiten aplicaciones en tiempo de ejecución sin afectar el resto de los servicios desplegados. La plataforma consta de un núcleo que es el encargado de las comunicaciones con los centros de mensajería, y de gestionar las aplicaciones desplegadas. De esta forma las aplicaciones no necesitan implementar ningún protocolo de comunicación, ni preocuparse por la forma en que las conexiones son realizadas.

- La plataforma es portable a diversos sistemas operativos y hardware, ya que se implementó utilizando tecnología Java. Asimismo, se dieron pasos para lograr portabilidad a diversos servidores de aplicaciones Java, aunque no se logró completamente.
- La plataforma es capaz de comunicarse con diferentes operadores móviles a través de distintos protocolos, para lo cual se utilizan las rutas.
- Se desarrolló un SDK con funcionalidades reusables, a partir del cual se implementó primeramente el núcleo de la plataforma, y más adelante el resto de las aplicaciones.
- La plataforma es capaz de ofrecer una interfaz a terceros para brindarles servicio de mensajería. Esta interfaz se implementa también como un servicio más de la plataforma.
- La plataforma permite desplegar diferentes servicios al mismo tiempo. Cada servicio se registra en la plataforma y define las palabras clave a partir de las cuales se van a enrutar los mensajes.
- Se implementó una interfaz de administración y monitoreo de la plataforma. Asimismo se ofrecen reportes de tráfico a partir de los cuales se puede conciliar la información con los operadores móviles.
- La plataforma es robusta, está preparada para gestionar los mensajes que no pueden ser enrutados a las aplicaciones en un momento determinado por fallos en la conexión u otros. Asimismo está preparada para recuperarse automática y manualmente ante fallos del centro de mensajería.
- El tiempo de implementación de las aplicaciones puede variar entre una semana como mínimo y un mes como máximo para las aplicaciones de mayor complejidad. Esto permite dar una respuesta rápida a cualquier necesidad que surja de nuevos servicios.
- En los primeros quince días en fase de producción, la plataforma fue capaz de procesar más de 104 000 mensajes de texto, registrando picos máximos de más de 2 000 mensajes de texto en una hora, y más de 20 000 mensajes de texto en un día.

De esta forma, fueron implementadas las principales funcionalidades estudiadas en otras plataformas a nivel internacional. Entre estas funcionalidades se encuentran el envío de mensajes a través de

interfaces web y de escritorio, el envío de mensajes a través de correo electrónico, y la conversión de mensajes de texto a mensajes de voz y viceversa.

Mientras que muchas de las plataformas estudiadas son de propósito específico, la plataforma BlueEye puede ser adaptada a distintos propósitos de acuerdo a las necesidades existentes, debido a su modularidad y extensibilidad, por lo cual las funcionalidades que no fueron implementadas en una primera iteración del desarrollo del producto, podrían ser incluidas si fuera necesario en forma de aplicaciones que extiendan las funcionalidades básicas de la plataforma.

Otras funcionalidades referentes a la ampliación de los servicios de la plataforma para la descarga de contenido digital (imágenes, tonos, animaciones, aplicaciones para móviles y otros), caen dentro del tema específico de una plataforma para la gestión de contenido para dispositivos móviles. En la actualidad también se dispone de una plataforma para este propósito, y se estudia la forma de integrar parte de sus funcionalidades como parte de una aplicación implementada sobre la plataforma BlueEye para la descarga de contenido.

III.9. Conclusiones del capítulo

La plataforma BlueEye fue diseñada teniendo en cuenta las funcionalidades estudiadas en las plataformas existentes a nivel mundial, como comunicarse con uno o varios operadores móviles por medio de distintos protocolos, el soporte para distintos servicios basados en mensajes para móviles, la implementación de mecanismos de monitoreo y reportes, la posibilidad de ofrecer un SDK para facilitar la implementación de nuevas aplicaciones, así como la posibilidad de ofrecer servicio de mensajería a terceros. Además en el diseño se tuvieron en cuenta una serie de factores como son la robustez, extensibilidad, reusabilidad y portabilidad.

La independencia del servidor de aplicaciones facilita una posible migración a otros servidores de aplicaciones, aunque el trabajo realizado en este sentido aún no es suficiente. La arquitectura modular de la plataforma permite que se implementen las aplicaciones de forma independiente y que se puedan desplegar sobre la plataforma de acuerdo a las necesidades o intereses en un determinado momento.

El núcleo de la plataforma se encarga del envío y recepción de mensajes desde y hacia las aplicaciones;

el enrutamiento de los mensajes entre los centros de mensajería y las aplicaciones, el registro de las aplicaciones desplegadas y el control de las trazas.

La plataforma además cuenta con un SDK que permite la implementación de nuevas aplicaciones. Estas, al iniciarse, registran sus palabras clave y sus comandos, y cargan su fichero de configuración. El enrutamiento de los mensajes a las aplicaciones se realiza a partir de la palabra clave y/o el número corto al cual fue enviado el mensaje de texto. Los mensajes que no se puedan enrutar van a una cola de reintentos, en la que se mantienen durante un tiempo determinado antes de ser desechados definitivamente, si no se han podido enviar.

Las trazas se dividen en trazas de negocio y de mensaje. Estas últimas son gestionadas únicamente por el núcleo de la plataforma y son útiles a la hora de generar reportes de tráfico. Estos reportes, así como otros datos de interés, se pueden consultar a través de la interfaz para el monitoreo y administración de la plataforma.

La primera versión de la plataforma fue sometida a una serie de pruebas en las que se detectaron un conjunto de fallas. Estas han sido solucionadas de forma total o parcial en las versiones más recientes de la plataforma.

CONCLUSIONES

En el presente trabajo se dio respuesta al problema planteado referente a la necesidad de desarrollar en el país servicios de valor agregado para dispositivos móviles, y se cumplió el objetivo general de este trabajo:

- Se sentaron las bases tecnológicas para el desarrollo de servicios de valor agregado para dispositivos móviles en el país, por medio de una plataforma para la gestión de servicios de valor agregado para dispositivos móviles, la cual es capaz de comunicarse con los distintos operadores móviles, así como gestionar y estandarizar el desarrollo de los distintos servicios de valor agregado.

Asimismo se cumplieron los objetivos específicos trazados para el trabajo:

- Se realizó un estudio de otras plataformas para la gestión de servicios de valor agregado que existan a nivel mundial. Como resultado de este estudio se identificaron una serie de funcionalidades que presentan las plataformas, las cuales se tuvieron en cuenta para el desarrollo de la plataforma.

- Se realizó un estudio de las tecnologías a utilizar en el desarrollo de la plataforma, el cual se centró en la plataforma Java Enterprise Edition y en los diferentes estándares definidos por el Java Community Process e implementados como parte de Java Enterprise Edition, para realizar diferentes tareas como gestión de mensajes para móviles e implementación de componentes distribuidos.

- Se desarrolló un SDK a partir del cual se implementó el núcleo de la plataforma, y que constituye un estándar para el desarrollo de las aplicaciones que implementan los servicios de valor agregado.

- Se implementó el núcleo de la plataforma, de tal forma que cumplió con los requisitos de comunicación con varios operadores a través de distintos protocolos al mismo tiempo, soporte para la implementación de aplicaciones, gestión de trazas, e interfaces para el monitoreo y reportes.

- Se evaluó el funcionamiento del núcleo de la plataforma a través de diferentes pruebas, como pruebas de unidad, de integración, de recuperación, estrés y de sistema. En estas pruebas se detectaron diferentes problemas que se resolvieron de forma completa o parcial.

- Se integró de forma satisfactoria la plataforma con el operador móvil del país, con buenos resultados hasta el momento. Actualmente se encuentra en funcionamiento en el operador móvil Cubacel.

Recomendaciones

Las siguientes recomendaciones se deben tener en cuenta para el posterior desarrollo y mejora de la plataforma:

- Mejorar la detección de errores ortográficos y no ortográficos durante el procesamiento de los mensajes de texto escritos por los usuarios, utilizando para ello algoritmos de distancia entre cadenas, de forma tal que mejore la capacidad de la plataforma de dar respuesta a los pedidos de los usuarios. Mejorar la sustitución de caracteres latinos por equivalentes en US-ASCII dentro del núcleo de la plataforma.

- Agregar nuevas funcionalidades a la interfaz de monitoreo y administración, de forma tal que mejore la administración de las rutas, las aplicaciones y las trazas.

- Mejorar la generación de reportes para dar soporte al mecanismo de cobro revertido.

- Simplificar el SDK de forma tal que se disminuya al mínimo la cantidad de clases y componentes distribuidos a implementar, para desarrollar una nueva aplicación.

- Implementar un *plugin* para el entorno de desarrollo Eclipse, que facilite el desarrollo de aplicaciones basadas en el SDK de BlueEye.

- Estudiar la forma de gestionar de una forma única las páginas web de administración de las distintas aplicaciones basadas en BlueEye.

REFERENCIAS BIBLIOGRÁFICAS

[ACM, 2003] Alur, Deepak; Crupi, John; Malks, Dan. *Core J2EE Patterns. Best Practices and Design Strategies*. Sun Microsystems Press, Prentice-Hall. 2003.

[Aldiscon, 1996] Aldiscon. *Short Message Peer to Peer (SMPP) Interface Specification*. [en línea] 14 enero 1996. <http://opensmpp.logica.com/CommonPart/Documentation/external/SMPP-IF-SPEC.v3.3.pdf> [Consultado: 13 septiembre 2009]

[Beck, 2003] Beck, Kent. *Test-Driven Development. By Example*. Addison-Wesley. 2003

[Collazos, 2007] Collazos, Marisol. *En 2008 enviaremos 23 millones de SMS* [en línea]. 19 diciembre 2007. <http://www.marisolcollazos.es/noticias-informatica/?p=240> [Consultado: 10 agosto 2009]

[EmpresaSMS, 2009] empresaSMS.com. *Servicios SMS de empresaSMS.com*. [en línea] 2009. <http://www.empresasms.com/en/servicios-sms.html> [Consultado: 13 septiembre 2009]

[Ericsson. 2006] Sony Ericsson. *WAP Push*. [en línea] 2 mayo 2006. <http://www.ericsson.com/mobilityworld/sub/open/technologies/open_development_tips/tools/wap_push_sdk> [Consultado: 12 septiembre 2009]

[Esendex, 2009] Esendex. *Envío de SMS de negocios con nuestros servicios SMS*. [en línea] 2009. <http://www.esendex.es/Envio-de-SMS/> [Consultado: 13 septiembre 2009]

[Farley, 2007] Farley, Tom. The Cell-Phone Revolution. [en línea]. *American heritage of invention & technology*. 2007. <http://www.americanheritage.com/events/articles/web/20070110-cell-phone-att-mobile-phone-motorola-federal-communications-commission-cdma-tdma-gsm.shtml> [Consultado: 13 septiembre 2009]

[GoF, 1995] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vissides, John. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley. 1995.

[Google, 2009] Google. *Google SMS. Overview*. [en línea] 2009. <http://www.google.ca/mobile/sms/index.html> [Consultado: 13 septiembre 2009]

[Ibrahim, 2002] Ibrahim, Jawad. *4G Features*. [en línea]. Bechtel Telecommunications Technical Journal, 2002. <http://www.bechtel.com/communications/assets/files/TechnicalJournals/December2002/Article2.pdf> [Consultado: 12 septiembre 2009]

[IEC, 2007a] International Engineering Consortium. *Global System for Mobile Communication (GSM)*. [en línea] 2007. <http://www.iec.org/online/tutorials/gsm/index.html> [Consultado: 12 septiembre 2009]

[IEC, 2007b] International Engineering Consortium. *Wireless Internet Network Communications Architecture*. [en línea] 2007. <http://www.iec.org/online/tutorials/winternet/topic04.html> [Consultado: 12 septiembre 2009]

[Ishii+, 1991] Ishii et al. *US Patent 5,058,201* [en linea]. 15 noviembre 1991. <http://www.google.com/patents?id=UkIoAAAAEBA> [Consultado: 9 febrero 2009]

[IsResearch, 2009] is-research. *Programming Languages for the Java Virtual Machine JVM*. [en línea] 2009. <http://www.is-research.de/info/vmlanguages/> [Consultado: 22 septiembre 2009]

[ITU, 2000] International Telecommunications Union. What really is a Third Generation (3G) Mobile Technology. [en línea] 2007. <http://www.itu.int/ITU-D/imt-2000/DocumentsIMT2000/What_really_3G.pdf> [Consultado: 12 septiembre 2009]

[ITU, 2007] International Telecommunications Union. ITU Radiocommunication Assembly approves new developments for its 3G standards. [en línea] 2007. <http://www.itu.int/newsroom/press_releases/2007/30.html> [Consultado: 12 septiembre 2009]

[Jboss, 2006] Jboss. *The Jboss 4 Application Server J2EE Reference*. Jboss, Inc. 2006.

[Larman, 2001] Larman, Craig. *Applying UML and Patterns*. Prentice Hall. 2001.

[LCOMF, 2007] Leitner, Andreas; Ciupa, Ilinca; Oriol, Manuel, Meyer, Bertrand; Fiva, Arno. *Contract Driven Developments = Test Driven Developments – Writing Test Cases*. Proceedings of ESEC/FSE'07: European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. Dubrovnik, Croatia. 2007.

[Lewis, 2000] Lewis, William. *Software Testing and Continuous Quality Improvement*. CRC Press. 2000.

[Mercacel, 2009] Mercacel. *Servicios*. [en línea] 2009. <http://www.mercacel.com/servicios_mercacel.php> [Consultado: 13 septiembre 2009]

[Meyer, 1997] Meyer, Bertrand. "Software quality". *Object-Oriented Software Construction*. 2da edición. Prentice Hall. 1997.

[Microsoft, 2009a] Microsoft. *.NET Framework Conceptual Overview*. [en línea] 2009. <http://msdn.microsoft.com/en-us/library/zw4w595w.aspx> [Consultado: 22 septiembre 2009]

[Microsoft, 2009b] Microsoft. *.NET Framework 3.5 Architecture*. [en línea] 2009. <http://msdn.microsoft.com/en-us/library//bb822049.aspx> [Consultado: 22 septiembre

2009]

[Microsoft, 2009c] Microsoft. *Introduction to IIS 7.0 Architecture*. [en línea]. 2009. <http://learn.iis.net/page.aspx/101/introduction-to-iis-70-architecture/> [Consultado: 22 septiembre 2009]

[MobileIN, 2004] Mobile in a Minute. *Value-Added Services*. [en línea] 2004. <http://www.mobilein.com/what_is_a_VAS.htm> [Consultado: 13 septiembre 2009]

[Mohr, 2002] Mohr, Werner. *Mobile Communications Beyond 3G in the Global Context*. [en línea]. Siemens mobile, 2002. <http://www.cu.ipv6tf.org/pdf/werner_mohr.pdf> [Consultado: 12 septiembre 2009]

[Molist, 2006] Molist, Mercè. "Las empresas de SMS facturaron 350 millones por sus servicios de pago". [en línea]. *El País digital*. 28 diciembre 2006. <http://www.elpais.com/articulo/tecnologia/empresas/SMS/facturaron/350/millones/servicios/pago/elpeputec/20061228elpcibtec_7/Tes> [Consultado: 10 agosto 2009]

[Nokia, 2004] Nokia Corporation. *JSR-212. SAMS Messaging API*. Nokia Corporation. 2004.

[NuestroSite, 2009] Nuestro Site. *Plataforma de envío de mensajes SMS/MMS*. [en línea] 2009. <http://www.nuestrosite.com.mx/productos/plataforma-de-envios-sms> [Consultado: 13 septiembre 2009]

[Olamendi, 2005] Olamendi, John. *J2EE and Microsoft .NET for Enterprise Applications*. [en línea]. 5 diciembre 2005. <http://www.c-sharpcorner.com/UploadFile/john_charles/12052005J2EEAndMSNET12052005104912AM/12052005J2EEAndMSNET.aspx> [Consultado: 22 septiembre 2009]

[Ozeki, 2006] Ozeki Informatics. *Ozeki Message Server 6. Product Guide.* Ozeki Informatics. 2006.

[Pressman, 2005] Pressman, Scott. *Software Engineering: A Practitioner's Approach*. 6ta edición. McGraw-Hill Education. 2005.

[SBS, 2006] Sriganesh, Rima; Brose, Gerald; Silverman, Micah. *Mastering Enterprise JavaBeans 3.0*. Wiley Publishing. 2006.

[Sharma, 2007] Sharma, Sonia. *Value Added Services: Reaching New Heights.* [en línea] 6 julio 2007. <http://voicendata.ciol.com/content/vNd100/2007vol-II/107070618.asp> [Consultado: 13 septiembre 2009]

[Sun, 2006a] Sun Microsystems. *Java Platform, Enterprise Edition (Java EE) Specification, v5*. [en línea] 8 mayo 2006. <http://jcp.org/aboutJava/communityprocess/pfd/jsr244/index.html> [Consultado: 19 septiembre 2009]

[Sun, 2006b] Sun Microsystems. *JSR 220: Enterprise JavaBeans, Version 3.0.* [en línea] 2 mayo 2006.

<http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html> [Consultado: 22 septiembre 2009]

[Sun, 2008] Sun Microsystems. *About the Java Technology*. [en línea]. 2008. <http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html> [Consultado: 19 septiembre 2009]

[Sun, 2009] Sun Microsystems. *JSR 318. Enterprise JavaBeans, Version 3.1*. [en línea] 2 mayo 2006. <http://jcp.org/aboutJava/communityprocess/edr/jsr318/index.html> [Consultado: 22 septiembre 2009]

[Tejera, 2009] Tejera, Lucilo. *Cuba busca soberanía en informática y comunicaciones* [en línea]. 5 febrero 2009. <http://embacu.cubaminrex.cu/Default.aspx?tabid=10604> [Consultado: 10 agosto 2009]

[Teleios, 2009] Teleios. *Plataforma para la entrega de servicios de mensajería móvil*. [en línea] 2009. <http://www.teleios-systems.com/Products/MessageCentral/Documents/MMSDP_Spanish.pdf> [Consultado: 13 septiembre 2009]

[TT, 2005a] TechTarget. *Short Message*. [en línea] 2 mayo 2005. <http://searchmobilecomputing.techtarget.com/sDefinition/0,290660,sid40_gci773769,00.html> [Consultado: 12 septiembre 2009]

[TT, 2005b] TechTarget. *Multimedia Message Service*. [en línea] 2 mayo 2005. <http://searchmobilecomputing.techtarget.com/sDefinition/0,290660,sid40_gci943702,00.html> [Consultado: 12 septiembre 2009]

[TT, 2007] TechTarget. *Short Message Service Center*. [en línea] 17 julio 2007. <http://searchmobilecomputing.techtarget.com/sDefinition/0,,sid40_gci1264572,00.html > [Consultado: 13 septiembre 2009]

[UneInternet, 2009] Une Internet. *Servicios SMS*. [en línea] 2009. <www.uneinternet.es/servicios_sms.html> [Consultado: 13 septiembre 2009]

[Vodafone, 2009] Vodafone. *DictaSMS*. [en línea] 2009. <http://www.vodafone.es/empresas/servicios/voz/basicos/DictaSMS/> [Consultado: 13 septiembre 2009]

[Wells, 1999] Wells, Don. *Unit Tests*. [en línea] 1999. <http://www.extremeprogramming.org/rules/unittests.html> [Consultado: 15 septiembre 2009]

[Wirzenius, 2004] Wirzenius, Lars. *Kannel Architecture and Design. Revision 1.19.* Wapit Ltd. 2004.

[Zope, 2009] Zope. *The Zope2 Book*. [en línea] 2009. <http://docs.zope.org/zope2/zope2book/source/> [Consultado: 22 septiembre 2009]

GLOSARIO DE SIGLAS Y TÉRMINOS

API: Ver Interfaz de programación de aplicaciones.

Arquitectura multicapas: La arquitectura multicapas o n-capas es una arquitectura cliente-servidor en la cual la presentación, la lógica del negocio y la gestión de los datos constituyen procesos lógicamente separados. El diseño más comúnmente usado de la arquitectura n-capas es el de 3 capas, compuesto por capa de presentación, capa de negocios y capa de datos.

BlueEye (aplicación): Una aplicación BlueEye es una aplicación desarrollada utilizando el SDK de BlueEye y en la cual se implementa un servicio basado en mensajes de texto. Cada aplicación debe definir en el núcleo de la plataforma su nombre, palabras clave y comandos que utilizará, entre otros parámetros.

BlueEye (núcleo): El núcleo de la plataforma BlueEye es el encargado de realizar la conexión con los operadores móviles a través de los distintos controladores, de gestionar los mensajes que entran y salen de la plataforma, y de gestionar las aplicaciones que se encuentran desplegadas en un momento determinado.

BlueEye (plataforma): La plataforma BlueEye es un software desarrollado utilizando tecnología Java, con el objetivo de gestionar servicios basados en mensajes para móviles.

Centro de mensajería: Un centro de mensajería o SMSC que es un elemento de la red de telefonía móvil que cumple la función de enviar y recibir mensajes, de forma tal que sus usuarios tienen la posibilidad de enviar y recibir mensajes.

Componente distribuido: Componentes distribuidos u objetos que pueden ser invocados de forma remota. Un caso específico de componente distribuido son los Enterprise JavaBeans.

Contenedor de EJB: El contenedor de EJB es el responsable de la gestión de los EJB, brinda un ambiente distribuido, seguro y transaccional, en el cual los EJB se puedan ejecutar, de forma tal que ni los EJB ni los clientes necesiten hacer llamados explícitos a ninguna API del contenedor para poder

acceder a sus servicios; sino que lo hacen a través del propio contenedor.

Dispositivo móvil: Un dispositivo móvil es un aparatos de tamaño pequeño, con algunas capacidades de procesamiento, con conexión permanente o intermitente a una red, con un capacidad de memoria limitada, y diseñados específicamente para una función determinada, como la comunicación telefónica en el caso de los teléfonos celulares, aunque también pueden llevar a cabo otras funciones más generales.

EJB: Ver Enterprise JavaBeans.

Enterprise JavaBeans: Los Enterprise JavaBeans (EJB) son componentes distribuidos definidos como parte de la plataforma Java EE, que se despliegan del lado del servidor para la construcción de aplicaciones distribuidas. Existen tres tipos de EJB: los EJB de entidad, los EJB de sesión y los EJB orientados a mensajes.

Entidad Externa de Mensajería: Una Entidad Externa de Mensajería es una aplicación externa que se conecta a un Centro de Mensajería con el propósito de enviar y recibir mensajes.

ESME: Ver Entidad Externa de Mensajería.

Interfaz de programación de aplicaciones: Conjunto de funciones y métodos que ofrece cierta biblioteca para ser usado por otro software como una capa de abstracción.

Java (plataforma): La plataforma Java es un entorno o plataforma de computación de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java u otros lenguajes que compilen a *bytecode* Java.

Java EE: Ver Java Platform Enterprise Edition.

Java Platform Enterprise Edition: Java Platform Enterprise Edition (Java EE) es una plataforma de programación basada en la plataforma Java para desarrollar y ejecutar aplicaciones empresariales con una arquitectura de n-capas distribuida, basándose en componentes de software modulares que se ejecutan sobre un servidor de aplicaciones.

Mensaje de texto: Un mensaje de texto es un mensaje breve enviado desde un dispositivo móvil a otro.

Mensaje multimedia: Un mensaje multimedia permite enviar mensajes que pueden incluir imágenes, audio y video, además de texto.

MMS: Ver Mensaje multimedia.

Notificación WAP: Una Notificación WAP es un mensaje que se envía a un dispositivo móvil con un enlace a Internet que puede hacer referencia a un MMS, un correo electrónico o a un URL para la descarga de contenidos como imágenes, juegos, entre otros.

Operador móvil: Un operador móvil es una compañía telefónica que brinda servicios de telefonía para clientes de teléfonos móviles.

PAP: Es un protocolo definido por la Open Mobile Alliance, que se utiliza para el envío de notificaciones WAP a través de los Push Proxy Gateway definidos en las pasarelas WAP. Ver Push Proxy Gateway.

Patrón de diseño: Los patrones de diseño constituyen soluciones reusables a problemas que ocurren comúnmente en el diseño del software. Un patrón de diseño no constituye un diseño terminado que puede ser transformado directamente en código, sino es una descripción de cómo se puede resolver un problema, y puede ser utilizado en diferentes situaciones.

Push Proxy Gateway: Es uno de los componentes de las pasarelas WAP, que se encarga de enviar notificaciones WAP a los dispositivos móviles.

Servicio de valor agregado: Servicios de valor agregado son todos los servicios que van más allá de los servicios básicos prestados por la telefonía móvil. Pueden incorporar mensajes de texto y multimedia, descarga de contenido como tonos, imágenes y juegos, y otros.

SMPP: Protocolo utilizado en la industria de las telecomunicaciones para el intercambio de SMS entre los Centros de Mensajería y las Entidades Externas de Mensajería.

SMS: Ver Mensaje de texto.

SMSC: Ver Centro de Mensajería.

Teléfono celular: Ver Dispositivo móvil.

UCP: Protocolo utilizado para enviar mensajes de texto a dispositivos móviles a través de los Centros de Mensajería.

WAP Push: Ver Notificación WAP.

XML: XML es un metalenguaje extensible de etiquetas que permite definir la gramática de lenguajes específicos. XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

ANEXO 1: OTRAS APLICACIONES IMPLEMENTADAS

A1.1. Aplicación para la cartelera del Festival de Cine

Esta aplicación se creó con el objetivo de informar acerca de la cartelera del Festival de Cine de La Habana. La aplicación presenta cierta complejidad ya que incluye varios tipos de consultas, cada uno con una palabra clave distinta, como las consultas por cine, nombre de película y país, además de poder especificar el día para el cual se deseaba conocer la cartelera. Debido al volumen de información a manejar, se implementaron varias formas de actualización, tanto por medio de una página web de administración, como a través de un fichero XML definido para esta función.

Un aspecto que hace más compleja esta aplicación es el tema de las búsquedas. Cuando el cliente pide la cartelera para un determinado cine, película o país, puede introducir errores al escribir el nombre. Por tanto la aplicación se preparó para lidiar con este tipo de situaciones y tratar de devolver la respuesta deseada. A cada uno de los cines, películas o países, les acompañó un listado de posibles variantes en que podía ser nombrado, incluidas faltas de ortografía, supresión de letras y abreviaturas. Este método es costoso a la hora de introducir los datos, dada la gran cantidad de películas, cines y países que tomaron parte en el Festival; y es propenso a errores, ya que la misma persona que introduce los datos tiene que imaginar las variantes para cada nombre. No obstante, su implementación es fácil y rápida, por lo cual se ha empleado en otras aplicaciones como la aplicación de las embajadas.

Para acceder a este servicio, se pueden enviar mensajes de texto con el siguiente formato:

- **cine** [nombre del cine] [día]
- **pelicula** [nombre de la película] [día]
- **muestra** [nombre del país] [día]

en donde día es el número del día, y si no se especifica se asume el número del día actual. Como respuesta, la aplicación devolverá un mensaje de texto con la cartelera según la consulta solicitada. En

el mensaje de respuesta se agregan tantas entradas de la cartelera sea posible, de acuerdo al tamaño máximo del mensaje de texto.

A1.2. Aplicación para el parte del tiempo

Esta es una aplicación sencilla, que tiene como objetivo ofrecer el parte del tiempo para el país en el día actual y los dos días siguientes, dividido por zonas (occidental, central y oriental). Entre otros datos se ofrecen las temperaturas máximas y mínimas, y el estado general del tiempo en la región, actualizado según los datos que se ofrecen en la página del Instituto de Meteorología.

Para acceder a este servicio, se pueden enviar mensajes de texto con el siguiente formato:

- **tiempo** [región]

en donde se obtiene como resultado un mensaje de texto con el parte del tiempo para la región especificada. De no especificarse la región, se devuelve el estado del tiempo de todo el país.

A1.3. Aplicación para las embajadas

Esta aplicación tiene como objetivo ofrecer datos acerca de la ubicación de las embajadas y consulados de las distintas naciones que tienen relaciones diplomáticas con nuestro país. En ella se ofrecen datos de interés como la dirección postal, teléfono y horario de atención al público de las distintas embajadas y consulados.

La búsqueda de los datos se realiza utilizando el mismo método de semejanza de nombres, esta vez teniendo en cuenta el nombre del país a que pertenece la sede diplomática.

Para acceder a este servicio, se pueden enviar mensajes de texto con el siguiente formato:

- **embajada** [nombre del país]

en donde es necesario especificar algún país. Se devuelve como resultado un mensaje de texto con los datos de la sede diplomática solicitada.

A1.4. Aplicación para el Clásico Mundial de Béisbol

Esta aplicación se concibió a raíz del II Clásico Mundial de Béisbol. El objetivo es ofrecer tanto los resultados de los juegos ya realizados, como la programación de los juegos por efectuarse.

De cada uno de los juegos se escogieron datos como la hora de comienzo, los equipos que tomaban parte; y lanzadores ganador y perdedor y resultado del juego en el caso de que ya se hubieran efectuado.

Para acceder a este servicio, se pueden enviar mensajes de texto con el siguiente formato:

- **resultados**
- **programacion**

en donde la consulta de resultados ofrece los resultados de los últimos juegos efectuados, y la consulta de programación los próximos juegos a realizarse.