UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

Plataforma de Gestión de Contenidos para Dispositivos Móviles.
Servicio de Subscripción Horóscopo.

Trabajo de Diploma

para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Igor Gerardo Morffi Chechulina

Tutor: Ing. Lex Karel Zayas Hernández

Ciudad de la Habana, junio de 2009.

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis, reconociendo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de esta de manera exclusiva.

Para que así conste, firmo la presente a los ___ días del mes de _____ de _____.

Igor Gerardo Morffi Chechulina	Ing. Lex Karel Zayas Hernández
Firma del Autor	Firma del Tutor

"...Milito en el grupo de los impacientes, y milito en el bando de los apurados, y de los que siempre presionan para que las cosas se hagan, y de los que muchas veces tratan de hacer más de lo que se puede..."

Dr. Fidel Castro Ruz

DEDICATORIA

… a Fidel, por la oportunidad y el camino.

… a mis abuelos, principio de lo que he sido, soy y seré.

… a mis padres, presentes en cada ocasión.

… a mis profes, amigos y amigas de todos estos años de estudio.

AGRADECIMIENTOS

"Cumple con la gratitud del peregrino, no olvidar nunca
la fuente que apagó su sed, la palmera que le brindó frescor y sombra,
y el dulce oasís donde vio abrirse un horizonte a su esperanza"

Dedico esta tesis, colofón de mis estudios de pregrado, a todas aquellas personas e instituciones que han aportado a la realización de la misma; en mi formación como Revolucionario, como ingeniero y como persona, haciendo posibles, llevaderos y profundos todos estos años de estudio comenzados en septiembre del '90.

Agradezco A Fidel, por hacer posible la oportunidad de convertirnos en hombres de ciencia y realizar nuestros sueños más preciados.

Agradezco A la UCI, a sus profes, a su gente, por ser la casa donde vivimos y nutrimos la mente, por darnos la oportunidad de probarnos como hombres de bien.

Agradezco infinitamente A la Vocacional Máximo Gómez, por mostrarnos cómo seguir el camino correcto, por cultivar el espíritu y templar el alma.

Agradezco A mis abuelos, Elsa y Elmo, por hacer con su cariño y ejemplo que todo haya sido posible.

Agradezco A mis padres, por confiarme las riendas, por saber escuchar y hablar, por estar siempre presentes.

Agradezco A mis tíos, Nelson y Ana, por consentirme y ser mis hermanos.

Agradezco A mis amigos de todos estos años, a los de verdad, a los que han quedado, presentes siempre, a los del pre, a los de casa, a los de la UCI, gracias por aguantarme.

Agradezco a mi tutor Lex Karel por el interés y el apoyo, a Rammel, Eliéser y Franky por el cómo, el dónde y la paciencia.

El Autor

RESUMEN

En la actualidad las tecnologías de mensajería inalámbrica tienen un amplio alcance en el mercado mundial, generando su utilización un gran impacto en el mismo. Mediante su uso se hace posible la creación de un gran número de servicios que ofrecen cuantiosas ganancias a las empresas que se desempeñan en este sector; tal es el caso de Cubacel, vicepresidencia de servicios móviles de ETECSA, que tiene entre sus líneas actuales el desarrollo de aplicaciones encaminadas a la telefonía móvil, entre ellas una plataforma de envío de contenidos para dispositivos móviles. En su versión actual, esta plataforma cuenta con algunos servicios de valor agregado complementarios, además de una serie de servicios básicos para una plataforma de este tipo, los cuales, sin embargo, no cubren la totalidad de los servicios que esta plataforma pudiese ofrecer, aumentando así las ganancias obtenidas por esta empresa. Teniendo en cuenta la situación anteriormente expuesta, se propone el desarrollo de un sistema que haga posible la subscripción, confirmación, publicación y gestión de acceso a un Servicio de Horóscopos a través de teléfonos celulares, con el fin de aumentar el número de servicios que posee dicha plataforma y aumentando paralelamente las prestaciones recibidas por parte de clientes nacionales y extranjeros, el cual fue desarrollado satisfactoriamente utilizando técnicas y herramientas afines con este propósito, supliendo las necesidades promedio presentadas por proveedores y clientes.

ABSTRACT

Nowadays, the wireless messenger technologies have a huge reach on the market all over the world, creating a great impact on it. Due to its use, it becomes possible the creation of such a great number of services, offering amazing earnings to the companies involved on it. That´s the case of Cubacel, ETECSA's Vice-presidency for mobile services, having among its present main production lines the development of applications oriented to the mobile telephony, specifically a platform to manage cell-phones contents' dispatch. On its current version, this platform has some value-added complementary services, aside from a series of basic services for a suchlike platform, which, however, do not cover up the totality of services a platform like this could offer, increasing the profits obtained by this company. Taking into account the previously exposed situation, has been proposed the development of a system to provide a horoscopes´ send out service, with the purpose of increasing the number of services provided by the abovementioned platform and the amount of services received by national and foreign costumers, this system was successfully developed using some tools in accordance to this purpose, supplying the average requirements of customers and providers.

TABLA DE CONTENIDOS

introducción

INTRODUCCIÓN

Desde su surgimiento las tecnologías de telefonía móvil se manifestaron como un innegable paso de avance en la esfera de las comunicaciones, acogido su uso por un gran número de usuarios y respaldado su desarrollo por varias empresas existentes y otras surgidas con este propósito.

Actualmente la telefonía móvil continúa incrementando sus potencialidades, llegando incluso estas a diferir un tanto del objetivo principal de su creación: la transmisión de voz, pero manteniendo el concepto de servir como medio de comunicación. Entre estas potencialidades cuenta hoy con la posibilidad de brindar servicios de amplio acceso a las redes de computadoras, incluyendo la red de Internet, lo cual implica una gran cadena de productores de contenidos, proveedores de servicio y fabricantes de equipos, así como una nueva percepción de la rentabilidad apoyada en la gestión e inversión de la infraestructura, obligando a seguir una serie de estándares de calidad y seguridad que derivan en una mejor relación productor-consumidor.

De esta forma, estas tecnologías han llegado a convertirse en una prestación básica para los usuarios, a consecuencia de las mejorías que brinda en términos de conectividad, movilidad, disminución de costo, rapidez, facilidad de despliegue, aumento de los servicios y calidad de los mismos.

Además de los motivos anteriores, existen otros que de manera paralela han posibilitado el auge alcanzado por la telefonía móvil en años recientes, entre ellos la modernización tecnológica continua y el crecimiento de las redes de operadoras, la propagación de nuevos servicios asociados y la disminución de los costos de las comunicaciones a consecuencia de la competitividad entre compañías proveedoras que tiene lugar en un gran número de países, dando lugar a la apertura de este mercado a esferas sociales que años atrás no hubiesen podido costear el uso de estos productos.

Todos estos factores tienen como consecuencia que debido a su esparcimiento las líneas móviles hayan ido suplantando las líneas fijas tradicionales, a la vez que los productores de terminales van incorporando a los teléfonos celulares tecnologías que permiten la transmisión de datos, dando a su vez a los proveedores la posibilidad del uso de datos móviles para brindar nuevos servicios.

Entre estos servicios se encuentran los conocidos como servicios de mensajería, reportando altos ingresos para las Empresas de Servicios de Valor Agregado. Haciendo uso de estos se hace posible la implementación de nuevos servicios que disfrutan del aplauso de los consumidores, entre los cuales se encuentran servicios de chat, envío de postales, consulta de noticias y partes climatológicos, acceso a

correo electrónico, verificación del estado de cuentas bancarias, realización de transacciones de compra y venta, entre otros. De ahí que un gran número de empresas haya puesto sus esfuerzos en dirección hacia la creación de servicios para usuarios de telefonía móvil, con el fin de ampliar el número de actividades que estos pueden ejecutar a través de sus celulares.

En el caso específico de nuestro país, Cubacel, Vicepresidencia de servicios móviles del operador telefónico ETECSA, trabaja en dirección al desarrollo de una plataforma para brindar servicios de telefonía móvil, con el objetivo de insertarse en el mercado de estos productos. Esta plataforma debe contar con los servicios básicos para este tipo de aplicaciones y con otros servicios avanzados que serán puestos a disposición de los clientes, entre los cuales se ha propuesto desarrollar un sistema que brinde el servicio de envío de horóscopos mediante subscripción.

A partir de la **situación problémica** anterior, se deriva el siguiente **problema de investigación**:

¿Cómo brindar servicio de horóscopos mediante subscripción a dispositivos móviles en el escenario actual de la red móvil cubana?

El **objeto de estudio** del presente trabajo son los sistemas que posibilitan el servicio de solicitud y recepción de horóscopos a través de teléfonos móviles.

El problema anterior arrojó para su resolución como **objetivo general** la implementación de un sistema que posibilite la subscripción, confirmación, publicación y gestión de acceso al servicio de horóscopos a través de teléfonos celulares, dando lugar a los siguientes **objetivos específicos**:

- Implementación de una aplicación de tipo servidora que maneje el flujo de actividades relacionado con la solicitud (subscripción) por parte de los usuarios de recibir el servicio de horóscopo, permitiendo a éstos personalizar el mismo y encargándose además del acceso a dicho contenido de parte de los primeros según sus particularidades.
- Implementación de una interfaz Web que posibilite llevar a cabo el proceso de solicitud de servicio a los usuarios.
- Desarrollo de una interfaz *Wap*[1] amigable que permita consultar el servicio de horóscopos por parte de los usuarios del sistema.

El **campo de acción** se enmarca en la plataforma de Cubacel que brindará este servicio.

Las **tareas de investigación** para cumplimentar los objetivos propuestos son:

1 Interfaz de Usuario accedida haciendo uso del protocolo del mismo nombre.

- Estudio de las potencialidades y limitaciones actuales presentadas por el protocolo WAP[2].
- Estudio de tendencias nacionales e internacionales en la creación de portales y servicios WAP.
- Estudio y selección de las herramientas de desarrollo más afines con el propósito trazado.
- Estudio sobre comunicación entre aplicaciones y elección de un método viable.

Para un mejor entendimiento de los temas a tratar en este trabajo de diploma se establece a continuación la Visión General del Trabajo:

Esta tesis se compone de las siguientes partes a través de 5 capítulos:

- Introducción
- Estado del Arte
- Descripción del problema
- Solución
- Propuesta de Resultados
- Bibliografía
- Anexos

En el Capítulo 1 se abordan los principales conceptos y términos abordados en la investigación, centrándonos en el estado del arte y adentrándonos en el problema al que se busca dar solución, se realiza un análisis de los servicios de valor agregado para móviles y se abordan tecnologías, herramientas y metodología utilizadas en el desarrollo de la solución.

En el Capítulo 2 se plantean los objetivos estratégicos de la vicepresidencia de servicios móviles de ETECSA, se explican los procesos que serán objetos de automatización, así como la propuesta de sistema, enfatizando en la arquitectura y el flujo de procesos de la aplicación.

En el Capítulo 3 son descritas las fases de exploración y planificación propias de la metodología de desarrollo utilizada para la implementación del sistema propuesto.

En el Capítulo 4 son referidas las fases de diseño, implementación y prueba propias de la metodología de desarrollo utilizada.

En el Capítulo 5 se realiza el estudio de factibilidad para la ejecución del sistema propuesto.

2 *Wireless Application Protocol*, traducido como Protocolo de Aplicaciones Inalámbricas.

capítulo 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se tratarán los principales conceptos y términos abordados en la investigación, se realizará un análisis de los Servicios de Valor Agregado (*Value Added Service* o VAS) para móviles y se abordarán tecnologías, herramientas y metodología utilizadas en el desarrollo de la solución.

1.1 Estado del Arte

En el mundo existen numerosas empresas dedicadas a prestar servicios de valor agregado para la telefonía móvil. Muchas de estas facilitan un servicio de subscripción de horóscopos, utilizando diferentes medios de acceso a los mismos. En nuestro país se ha dado los primeros pasos en pos de concretar prestaciones de este tipo, sin embargo producto de la escasa competitividad entre proveedores dichos servicios no han sido totalmente explotados, lo cual imposibilita realizar un estudio profundo de los mismos a nivel nacional.

A continuación, se detallan servicios de subscripción de horóscopos para dispositivos móviles de diferentes proveedores y nacionalidades:

1.1.1 Servicio de Horóscopos Interacel

Interacel: Brinda a sus clientes la posibilidad de recibir un servicio sencillo de horóscopos cuya subscripción consiste sólo en enviar mediante un mensaje de texto el signo zodiacal del que se desea recibir información diaria a un número telefónico determinado, comenzando a recibir el horóscopo diario a través de otro mensaje de texto. Este servicio no posee la posibilidad de configurar la forma en que es recibida la información por parte del cliente (Interacel, 2009).

Fig. 1 Experiencia del Usuario Interacel

1.1.2 Servicio de Horóscopos T-mobile

T-mobile: Permite escoger a sus clientes entre dos tipos de horóscopos, llamados Horóscopo General y Horóscopo Romántico, además de ofrecer al cliente recibirlo de manera diaria o solo una vez. Para establecer cual de estos servicios desea recibir, el usuario de T-mobile deberá enviar un mensaje de texto al número correspondiente siguiendo la sintaxis: Rom/Horoscope, para definir qué horóscopo desea recibir, seguido de las cinco primeras letras de su signo zodiacal, y la palabra "on" para activar una subscripción diaria por tiempo indefinido a este servicio. De querer sólo la información para un día, el cliente deberá omitir la palabra "on" al final del mensaje de texto (T-Mobile, 2009).

Fig. 2 Experiencia del Usuario T-mobile

1.1.3 Servicio de Horóscopos Movistar

Telefónica Movistar: Posee tres formas de acceso a la subscripción de su servicio de horóscopos, ya sea a través de voz, mensajería o mediante navegación, desde cualquier tipo de terminal independientemente de la tecnología del terminal y sin necesidad de alta o subscripción mensual al servicio.

En el primer caso, acceso a través de voz, el usuario realiza una llamada a un número específico luego de lo cual el sistema indica qué palabras se debe decir o qué teclas deben ser pulsadas para llegar a las distintas opciones de menú. En caso de que el sistema no entienda al usuario (debido a ruido o voces de fondo, errores de transmisión, etc.), propone la utilización de las teclas del teléfono para continuar usando el servicio.

Fig. 3 Experiencia del Usuario Movistar (Servicio de Voz)

La segunda opción brindada por esta empresa proveedora consiste en el envío de un mensaje de texto por parte del cliente a un número específico con el código del servicio de horóscopos, recibiendo en pocos instantes la información solicitada a través de otro mensaje.

Fig. 4 Experiencia del Usuario Movistar (Servicio de Texto)

La manera más vertical ofrecida por Movistar consiste en acceder directamente a la página Web del servicio e ingresar los datos de la subscripción por parte del usuario (Telefónica I+D, 2005)

Fig. 5 Experiencia del Usuario Movistar (Servicio de Navegación)

En cualquier opción, el sistema proveerá al cliente de la información necesaria para configurar su servicio de horóscopo desde la página Web del mismo, así como un código de activación a través de un mensaje (Telefónica, 2009)

Fig. 6 Experiencia del Usuario Movistar (Configuración y recibo)

1.2 Introducción a la telefonía móvil

La telefonía celular fue concebida estrictamente para la transmisión de voz, sin embargo este concepto ha evolucionado a tal punto que en la actualidad es capaz de ofrecer otros servicios, así por ejemplo hoy puede transmitirse datos, audio y video haciendo uso de esta tecnología. En un principio estas posibilidades presentaban algunas limitaciones; sin embargo, la telefonía inalámbrica actual hace

posible el desarrollo de aplicaciones con más prestaciones a consecuencia del aumento de la capacidad de transmisión de las redes y la evolución de los dispositivos móviles.

A nivel mundial, uno de los estándares más propagados es el Sistema Global para Comunicaciones Móviles (*Global System for Mobile Communications* o GSM). Este estándar soporta múltiples servicios, destacando entre ellos el Servicio de Mensajería Corta (*Short Message System* o SMS) el cual es un servicio de telefonía móvil que cuenta con aceptación universal, posibilitando la transmisión de mensajes cortos (SM) alfanuméricos[3] entre usuarios móviles y sistemas externos como correo electrónico, *paging*[4] y sistemas de correos de voz (IEC, 2007).

Un Mensaje Corto (*Short Message* o SM) es un mensaje de texto breve enviado desde un dispositivo móvil MO (*Mobile Originated*), hacia un dispositivo móvil MT (*Mobile Terminated*), a través de un SMS. Un SM estándar consiste en 160 caracteres alfanuméricos máximo, aunque se pueden enviar mensajes con mayor cantidad de caracteres utilizando compresión de datos.

Los SMS no son enviados punto a punto, de manera que se hace necesaria una entidad en la red inalámbrica encargada del proceso de envío. Esta entidad es conocida como Centro de Servicio de Mensajes Cortos (*Short Message Service Center* o SMSC) y es responsable de la transmisión, almacenamiento y reenvío de SM entre Entidades de Mensajería Corta (*Short Message Entity* o SME) (TechTarget, 2005).

Como parte del proceso de evolución del estándar GSM, la versión 97 del mismo, desarrollada por 3GPP (3rd *Generation Partnership Project*) hace posible el surgimiento del Servicio General de Radio-Paquetes (*General Packet Radio Service* o GPRS), en el cual fueron adicionadas capacidades de empaquetamiento de datos. Con el surgimiento de GPRS también surgen nuevos servicios, pudiéndose mencionar entre los más conocidos el Servicio de Mensajes Multimedia (*Multimedia Message Service* o MMS), diseñado específicamente para GPRS (IEC, 2007).

Un mensaje multimedia es enviado a través del Servicio de Mensajes Multimedia, tecnología de comunicaciones que permite a los usuarios intercambiar mensajes multimedia entre teléfonos y otros dispositivos móviles. MMS es una extensión de SMS, que define una manera de enviar y recibir, casi instantáneamente, mensajes que incluyen imágenes, audio y videos clip además de texto (TechTarget, 2005)

3 Conjunto de letras, números y otros símbolos, como signos de puntuación o símbolos matemáticos.

4 *Paging* o localizador, servicio de comunicaciones móviles que permite recibir mensajes de texto de manera unidireccional.

El servicio MMS puede replicar el comportamiento de un correo electrónico con archivos adjuntos, y como valor añadido y diferenciador permite utilizar presentaciones animadas. Esto es posible mediante la inclusión en el mensaje de un elemento de presentación llamado Lenguaje de Integración Multimedia Sincronizada (*Synchronized Multimedia Integration Language* o SMIL) el cual es un simple pero poderoso lenguaje de etiquetas para especificar cómo y cuándo mostrar clips (TechTarget, 2005).

Este lenguaje especifica de qué forma, en qué momento y en qué lugar las diversas partes del mensaje deben ser presentadas al usuario. Es un lenguaje basado en XML (*eXtensible Markup Language*), 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 (*slides*), 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 (TechTarget, 2005).

Otro de los servicios relevantes que soporta el estándar GSM y por consiguiente GPRS son las Notificaciones Wap (*Wap-Push*). Estas típicamente hacen referencia a un MMS, a un correo electrónico o a una 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, donde el teléfono móvil receptor del mensaje muestra el texto, en caso de que exista, y provee al usuario la opción de conectarse al URL con solo presionar una tecla (Ericsson, 2006).

1.2.1 Servicios de Valor Agregado

En el mundo de la telefonía celular los mensajes SMS, MMS y *Wap-Push*, fueron consumidos de manera tradicional hasta el momento en que diferentes empresas desarrollaron los primeros servicios de valor agregado en este campo. Desde entonces, un simple mensaje SMS que se enviaba desde un teléfono celular a otro se convirtió en toda una industria de aplicaciones con diversos servicios que incluían noticias, votaciones, horóscopos, chat, reporte del estado del tiempo, entre muchos otros. A su vez, con el surgimiento de los MMS, los usuarios se beneficiaron de nuevas aplicaciones que incluyen una combinación de imágenes, sonidos, videos y texto, experimentando una experiencia más agradable.

El mundo de los servicios de valor agregado es basto y diverso, englobando distintas necesidades y amplias porciones del mercado de telefonía móvil. En tal sentido, la gama de servicios es grande y brinda potencialidades que pueden ser explotadas en áreas de prioridad baja tales como el entretenimiento o en

temas más relevantes para la actividad laboral y cotidiana.

Una aplicación o servicio de valor agregado basado en MMS, es la instrumentación que se hace del servicio básico MMS para llevar a cabo un modelo de negocio habilitado por las capacidades de esta tecnología. La variedad y utilidad de las aplicaciones y servicios disponibles redundarán en el incremento del ingreso medio por usuario de los operadores móviles (Telefónica I+D, 2005).

Los distintos tipos de servicios se pueden agrupar en cuatro bloques:

1. Servicios basados en el contenido. No poseen ningún tipo de procesamiento, encapsulado o tratamiento; en otras palabras, son los mensajes intercambiados entre los usuarios.

2. Servicios básicos. Pueden ser de dos tipos:
   a) Aplicaciones de entretenimiento. Son la evolución natural de los servicios actuales de descarga de los elementos de entretenimiento para personalizar los terminales de los usuarios (logos, melodías, refrescadores de pantalla).
   b) Aplicaciones de almacenamiento de imágenes fijas. En este caso existen dos vertientes: la publicación de fotografías u otros elementos en álbumes Web, o el almacenamiento en un espacio privado del usuario para evitar el consumo de recursos en el terminal.

3. Servicios complementarios. Permiten la creación de contenidos por el usuario como el envío de tarjetas postales o felicitaciones, y los compositores de mensajes completos, ofreciendo al usuario repositorios de contenidos elementales como imágenes y sonidos.

4. Servicios de información multimedia. Son similares a los servicios de alertas SMS, pero explotan las capacidades ofrecidas por la tecnología MMS. Entre ellos se encuentran los servicios de suscripción, los servicios de noticias, los servicios financieros, de información meteorológica, los anuncios musicales o videos promocionales, los servicios de envío de fotografías capturadas por webcams y los juegos interactivos.

1.3 Metodología de Desarrollo utilizada

En el desarrollo de aplicaciones, es sin dudas de vital importancia la selección de una metodología de desarrollo. Actualmente existe un sinnúmero de éstas, siendo imposible determinar una que satisfaga en su totalidad las necesidades de un proyecto de desarrollo en específico; de esta forma, la selección de

una metodología de desarrollo debe llevarse a cabo teniendo en cuenta diferentes puntos que influyen en el desarrollo del sistema, entre estos: recursos técnicos, humanos y tiempo de desarrollo esperado o disponible.

Con el fin de llevar a cabo la implementación de esta aplicación se decidió tomar *Extreme Programming* (XP) como metodología de desarrollo a utilizar. Básicamente se podría decir que la Programación Extrema es una "metodología ligera o ágil para el desarrollo de software eficiente altamente efectivo" que busca clara y fundamentalmente dos objetivos: hacer software bien (con calidad) y de la forma más rápida posible (Martin, 2001).

1.3.1 ¿Por qué Extreme Programming?

La metodología de desarrollo *Extreme Programming* suple varias de las condiciones de desarrollo de este sistema, por lo que fue seleccionada para ser utilizada en esta aplicación, a continuación se exponen algunas ideas que clarifican esta decisión:

- Es una metodología ágil pensada para proyectos cortos con requerimientos muy cambiantes y pequeños grupos de desarrollo, siendo ambas características de este sistema.
- Apunta a una alta integración con el usuario.
- Su idea es ofrecer muchas entregas del sistema agregando funcionalidad paulatinamente en lapsos cortos de tiempo, esto permite que el usuario tenga más claro si el sistema hace lo que él quiere, además de comprometerlo en el desarrollo.
- Todo el trabajo se orienta hacia las pruebas, se realizan pruebas de unidad de los módulos que incluso se diseñan antes del software.
- Posibilita la realización de reuniones informales todos los días por la mañana para intercambiar experiencias del día anterior.
- Se realizan pruebas de integración con frecuencia.
- Es una metodología orientada fuertemente hacia la codificación.

Además para este sistema es muy factible que XP presenta sólo tres roles principales (Programador, Jefe de Proyecto, Cliente) permitiendo al programador desarrollar indistintamente las tareas de análisis, diseño, implementación y pruebas, de manera que el sistema puede ser completamente desarrollado por

una sola persona, con cierta asesoría del cliente y un tutor que haría las veces de Entrenador, Encargado de Proyecto y Rastreador (*Coach, Tester y Tracker*)[5].

1.4 Plataforma de Desarrollo utilizada

Una plataforma de desarrollo es el entorno de software común en el cual se desenvuelve la programación de un grupo definido de aplicaciones. Comúnmente se encuentra relacionada directamente a un sistema operativo; sin embargo, también es posible encontrarla ligada a una familia de lenguajes de programación o a una Interfaz de programación de aplicaciones (Wikipedia, 2008).

1.4.1 ¿Por qué Java EE?

Entre las muchas posibilidades que existen hoy en día para el desarrollo de aplicaciones para telefonía celular, la plataforma Java EE (*Java Enterprise Edition*) de *Sun Microsystems*, cuenta con una magnifica aceptación por parte de la comunidad internacional dedicada a esos menesteres.

Para desarrollar la aplicación fue seleccionada como plataforma Java EE, producto de su extensa utilización para el desarrollo de aplicaciones de telefonía móvil, siendo la más usada para este propósito a nivel mundial. Además para esta selección se tuvo en cuenta que la utilización de Java EE es una política establecida de Cubacel.

La Plataforma Java EE ofrece tecnología para el desarrollo de aplicaciones empresariales con distribución multicapa basadas en Web. Además, define una infraestructura común básica para el acceso a bases de datos, gestión de la persistencia, control de seguridad, y gestión de transacciones, ofreciendo una separación clara entre presentación (interfaz), modelo (lógica) de negocio y datos.

Java EE provee de grandes ventajas a todos los involucrados en el proceso de desarrollo de software; así por ejemplo, los desarrolladores cuentan con gran cantidad de recursos de la comunidad de desarrolladores, entre los cuales figuran libros, artículos, tutoriales, guías de mejores prácticas y patrones de diseño entre otros, además de poder usar o consultar componentes desarrollados por terceros; por otra parte los proveedores de aplicaciones trabajan juntos en las especificaciones y compiten en las implementaciones, teniendo libertad para innovar, a la vez que se mantiene la portabilidad de las aplicaciones, mientras los usuarios gozan de portabilidad garantizada respecto al sistema operativo, motor de bases de datos, servidor o contenedor de aplicaciones, entre otros, así como

5 Roles de *Extreme Programming* considerados “secundarios”.

múltiples opciones de implementación de acuerdo a diferentes requerimientos como pueden ser precio, capacidad de crecimiento, confiabilidad o rendimiento, y cuentan con un número mayor de desarrolladores disponibles.

Además, Java ofrece numerosas especificaciones de Interfaz de Programación de Aplicaciones (*Application Program Interface o* API), entre las cuales pueden mencionarse JDBC, RMI[6], e-mail, JMS, Servicios Web y XML, además de definir cómo coordinarlos. Java EE también configura algunas especificaciones únicas para componentes. Éstas incluyen *Enterprise Java Beans* (EJB), *Servlets*, *Portlets* (siguiendo la especificación de *Portlets* Java), Java *Server Pages* y varias tecnologías de servicios Web. Esto permite al desarrollador crear una aplicación empresarial escalable, portable entre plataformas e integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor o contenedor de aplicaciones puede manejar transacciones, seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, lo cual significa que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes, en lugar de tareas de mantenimiento de bajo nivel (Jendrock, 2006).

1.5 Tecnologías utilizadas

En el presente epígrafe serán detalladas las diferentes tecnologías utilizadas en el proceso de desarrollo de este sistema, seleccionadas en correspondencia con la plataforma de desarrollo así como las ventajas que estas brindan.

1.5.1 Wireless Markup Language

Para el desarrollo de interfaces *Wap* estándares, se utiliza el Lenguaje de Marcado Inalámbrico (*Wireless Markup Language* o WML) cuyo origen es el XML. Este lenguaje se utiliza para construir las páginas que aparecen en las pantallas de los teléfonos móviles y los asistentes personales digitales (PDA, por sus siglas en inglés) dotados de tecnología *Wap*. Es una versión reducida del lenguaje HTML que facilita la conexión a Internet de dichos dispositivos y que además permite la visualización de páginas web en dispositivos inalámbricos que incluyan la tecnología *Wap*. La visualización de la página dependerá del dispositivo que se use y de la forma en que este interprete el código, ya que varían entre sí.

WML es un metalenguaje, lo que implica que además de usar etiquetas predefinidas se pueden crear componentes propios y tiene ciertas similitudes con otro lenguaje de etiquetas bastante conocido, el

6 *Remote Method Invocation*, traducido como Invocación de Método Remoto.

HTML (*Hypertext Markup Language*), utilizado para la creación de páginas web convencionales. Un consorcio formado por Nokia, Phone.com, Motorola y Ericsson, conocido como el *WAP Forum*, define la sintaxis, variables y elementos utilizados en WML. Algunos fabricantes han desarrollado capacidades adicionales a este estándar. Al igual que el HTML se sirve de un lenguaje de script como JavaScript para dotar de cierto dinamismo a sus documentos, WML dispone del WMLS que es un lenguaje bastante similar al JavaScript, pero con algunas diferencias (Routt, 2003).

1.5.2 WURFL-WALL

Como consecuencia de la gran diversidad de dispositivos móviles que existen hoy en día debido a las distintas dimensiones de pantallas, formato de imágenes y audio que soportan, en el mundo se genera una gran polémica para los desarrolladores en la creación de páginas o sitios *Wap*. Para dar solución a este problema surge la combinación de WURFL-WALL.

WURFL (*Wireless Universal Resource File*) y WALL (*Wireless Abstraction Library*) son un conjunto de ficheros y librerías orientadas a facilitar el desarrollo de aplicaciones *Wap* para clientes que consten de poco procesamiento, con una gran portabilidad entre distintos tipos de navegadores embebidos dentro de clientes móviles (navegadores *Wap* que soporten WML, cHTML, iMode, XHTML) (Passani, 2002).

WURFL en realidad es un fichero XML donde se expresan, de manera compacta, las capacidades de diversos dispositivos. WALL es la combinación de una API Java, y una serie de librerías de etiquetas (*taglibs* JSP[7]) que permiten construir páginas JSP portables entre dispositivos.

Mediante su uso, se puede construir aplicaciones que pueden ser visualizadas en diferentes entornos, adaptadas a las capacidades disponibles en cada dispositivo. Así, el mismo contenido, ya sea imagen, animación o sonido, puede emplearse en distintos dispositivos sin la necesidad de tener varias versiones de la aplicación o codificar usando muchos bloques condicionales en función del elemento.

El uso típico de WALL es la utilización de la librería de etiquetas incluida para la creación de vistas JSP, asociadas al resto del modelo de la arquitectura. Su uso es compatible con cualquier otra librería de etiquetas, como las que proporciona JSTL (Guemes, 2006).

Otro uso de WALL es el de la API que nos informa de las características de los dispositivos directamente. Así, dentro del código Java, se puede analizar si un dispositivo dispone de una determinada

[7] Véase 1.5.4

característica (tamaño de pantalla, color, y demás) y en función de ello adaptar la funcionalidad de la aplicación a los dispositivos (Passani, 2002).

Las capacidades de WALL dependen del fichero de WURFL, que se actualiza constantemente. Conviene disponer de la última versión que cubra los dispositivos que se utilizarán en el proyecto.

1.5.3 Service Oriented Architecture

Los sistemas implementados mediante varias aplicaciones descomponiendo las funcionalidades del mismo, o que usan funcionalidades ofrecidas por otros sistemas, llamados Aplicaciones Distribuidas o de Arquitectura Orientada a Servicios (*Service Oriented Architecture* o SOA), permiten intercambiar información entre clientes en diversos niveles de detalle, contenido y presentación según varios parámetros que dependen del mismo cliente y del ambiente, proporcionando diversos servicios a diferentes clientes, o al mismo cliente en disímiles situaciones. De esta manera se ha desarrollado una noción diferente de contexto centrado principalmente en la adaptación de aplicaciones a diferentes necesidades y entornos con características que pueden ser explotadas de manera sencilla y eficiente, haciendo que estas tecnologías se hayan convertido en un paradigma de la programación actual.

Las aplicaciones distribuidas, se ejecutan en múltiples *hosts* (servidores), de manera que los objetos que se están ejecutando en un *host* invocan a los métodos de los objetos de otros *host* para ayudarles a realizar su procesamiento y pueden devolver valores que son utilizados por los objetos locales.

Se ha desarrollado una serie de perspectivas para la implantación de sistemas distribuidos. Internet y la web constituyen ejemplos de sistemas distribuidos que han sido desarrollados por medio del enfoque TCP/IP cliente/servidor donde clientes y servidores se comunican a través de sockets, lo cual requiere que haya protocolos a nivel de aplicación para que se establezca la comunicación entre ellos.

Para comunicar entre sí los extremos de las aplicaciones distribuidas en una arquitectura orientada a servicios, se hace necesaria la utilización de técnicas de implementación de servicios, entre las cuales están las soluciones *Web-Service* (WS), CORBA (*Common Object Request Broker Architecture*), COM (*Component Object Model)* de *Microsoft,* SOAP *(Simple Object Access Protocol)* y Java RMI (*Véase 1.5.3.1*), mecanismos que permiten invocar o ejecutar métodos remotos aislando al programador de todos los detalles que deben ser contemplados al diseñar un procedimiento o aplicación en Red.

1.5.3.1 ¿Por qué Java RMI?

La meta principal del diseño de RMI es permitir a los programadores el desarrollo de programas distribuidos en JAVA, usando la misma sintaxis y semántica de los programas no distribuidos, de manera que se haga más natural y fácil su uso. Por otra parte RMI es potente, simple y fácil de usar, siendo un magnífico mecanismo para ampliar las aplicaciones Java al trabajo en red (Oberg, 2001).

En la actualidad existe un marcado interés en usar las características únicas de RMI en la producción de sistemas, de forma que podemos encontrar pequeñas y grandes aplicaciones confeccionadas haciendo uso de RMI, lo cual da fe de su alto valor y operatividad (*Véase Anexo 1*)

En el caso de nuestra aplicación, RMI se muestra como la opción ideal para la integración entre aplicaciones, teniendo en cuenta, además de las razones anteriormente expuestas, que Java RMI se hace parte del estándar del lenguaje Java, utilizado por nuestro equipo de desarrollo para implementar la solución informática propuesta, aprovechando de esta forma todas las ventajas ofrecidas por dicho lenguaje de programación.

Desde el punto de vista de la lógica utilizada en su implementación, Java RMI es considerada una plataforma amigable para empezar en el área de las aplicaciones distribuidas, cuyos detalles de comunicación son transparentes para el programador, permitiendo el desarrollo rápido y fácil de objetos distribuidos, de fácil uso y sin requerir conocimientos de un Lenguaje de Definición de Interfaces, (*Interface's Definition Language*).

Esta tecnología logra ampliamente optimizar recursos y tiempo de trabajo, puesto que consigue lo que se denomina "escribir una vez, ejecutar en cualquier parte". Otra característica de interés es que un cliente RMI puede llamar a un objeto remoto en un servidor, y este servidor, a su vez, puede también ser un cliente de otros objetos remotos.

Clientes, servidores e interfaces remotos se escriben íntegramente en JAVA, a la vez que usa el protocolo JRMP (*JAVA Remote Method Protocol*) para comunicar los objetos JAVA remotos. Lo cual minimiza los riesgos en la transmisión de datos, desentendiendo al desarrollador de esa cuestión.

La tecnología RMI posee ventajas en cuanto a seguridad, puesto que es utilizado en redes cerradas y realiza la transmisión de datos a través de *sockets*, directamente entre un procesador y otro, lo cual además acorta el tiempo de espera entre solicitudes y respuestas.

1.5.4 JSP y Servlet

Un *Servlet* es un objeto que se ejecuta dentro del contexto de un servidor o contenedor Java EE. Fue especialmente diseñado para ofrecer contenido dinámico desde un servidor web, generalmente es HTML. Los *servlets* forman parte de la plataforma Java EE. Un *servlet* es un objeto Java que implementa la interfaz javax.servlet.Servlet o hereda alguna de las clases más convenientes para un protocolo específico (ejemplo: javax.servlet.HttpServlet). Al implementar esta interfaz el *servlet* es capaz de interpretar los objetos de tipo HttpServletRequest y HttpServletResponse quienes contienen la información de la página que invocó al *servlet* (Hall y Brown, 2003).

Entre el servidor de aplicaciones (o contenedor web) y el *servlet* existe un contrato que determina cómo han de interactuar. La especificación de éste se encuentra en los JSR (*Java Specification Requests*) del JCP (*Java Community Process*) (Perry, 1999).

JSP (*Java Server Pages*) por su parte, es una tecnología Java que permite a los programadores generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo. Las páginas JSP permiten que tanto el código Java como algunas acciones predefinidas puedan ser incrustadas en el contenido estático del documento web (Bayern, 2001).

1.5.5 SAMS-M

La existencia de múltiples protocolos de comunicación y diferentes servicios de mensajería como SMS, MMS y *Wap-Push,* representa un problema para los desarrolladores. La industria necesitaba un estándar que regulara las interfaces para el desarrollo de aplicaciones con servicios de SMS, MMS y *Wap-Push* con una arquitectura que permitiera agregar implementaciones para cada uno de los protocolos de comunicación con Centros de Mensajería. Este estándar fue llamado API Servidora para Servicios Móviles de Mensajería (*Server API for Mobile Services-Messaging* o SAMS-M) y no fue desarrollado hasta el 2004, como una alternativa a la variedad de soluciones propietarias que existían en el mercado donde cada una de estas definía cuáles protocolos implementar (Sun Microsystems, 2004).

SAMS-M representa el primer estándar que se define para homogenizar las interfaces de envío y recepción para las aplicaciones desarrolladas por los VASP.

El fluijo básico de mensajería de SAMS comienza con un cliente móvil enviando una petición de mensaje; este mensaje es enrutado al servido de mensajería correspondiente basándose en el tipo de

mensaje (SMS o MMS). En el caso de los mensajes SMS, el servidor de mensajes entrega directamente el mensaje al dispositivo receptor, mientras para entregar un MMS, se envía primeramente una notificación al receptor, siendo entregado el mensaje sólo si es aceptado por este.

Las API de SAMS están diseñadas para trabajar con las plataformas J2EE y J2SE, separadas en dos API, una llamada de Servicio (*SAMS Service API*) y otra llamada de Recurso (*SAMS Resource API*). El API de Servicio usa los patrones de diseño comunes de J2EE y provee a los desarrolladores de una manera fácil de programar aplicaciones para móviles, mientras que el API de Recurso descansa debajo del API de Servicio y actúa como puente entre este último y las Interfaces Proveedoras de Servicio (*Service Provider Interfaces* o SPI). En la mayor parte de los casos la implementación de SAMS es transparente al usuario, quien solo debe implementar los servicios que desea brindar, sin embargo también es posible que se haga necesaria la implementación de un driver propio, siendo necesario en estos casos usar el API de Recursos. SAMS pone a disposición la estructura base de sus API, dejando en hombros de cada proveedor de servicios la responsabilidad de su implementación (Java Boutique, (TM), 2004).

La API de Recursos de SAMS, puede ser desarrollada con cualquier tecnología, pero para reducir duplicaciones, pude ser utilizada la arquitectura de conexiones de Java, puesto que esta ofrece gestión de recursos y transacciones, seguridad y administración de *result-sets*[8].

1.6 Frameworks de Desarrollo

Los *Frameworks*[9] son estructuras de soporte definidas, en las cuales otro proyecto de software puede ser organizado y desarrollado. Estos proveen una estructura y una metodología de trabajo que extiende o utiliza las aplicaciones del dominio, puede ser considerado como el conjunto de procesos y tecnologías usados para resolver un problema complejo o como el esqueleto sobre el cual varios objetos son integrados para una solución dada (Wikipedia, 2009).

1.6.1 Spring Framework

Spring es un *framework* de código abierto que facilita la creación y desarrollo de aplicaciones para la plataforma Java. Ofrece muchas libertades a los desarrolladores, además de que cuenta con una

8 Termino técnico proveniente del inglés, utilizado para hacer alusión a conjuntos de datos devueltos como resultado de una operación.

9 Marcos de Trabajo

abundante documentación. Se basa y promueve el empleo de las mejores prácticas de programación aceptadas en la actualidad.

Está compuesto por siete módulos principales que colaboran y brindan una amplia gama de funcionalidades fáciles de emplear, que garantizan una arquitectura robusta para cualquier proyecto que tenga su basamento sobre Spring. Entre sus módulos se encuentran:

- Módulo Núcleo (*Core Container and Supperting Utilities*): Contiene la Fábrica de Clases (*BeanFactory*) que es el corazón de toda aplicación sobre Spring. *BeanFactory* aplica la Inversión de Controles (*Invertion of Control o IoC*), específicamente la Inyección de Dependencias (*Dependency Injection*), para separar la configuración de la aplicación y sus especificaciones de dependencias entre objetos de la lógica de negocio de la aplicación.
- Módulo AOP (*Aspect-Orient Programming*): Brinda soporte a la Programación Orientada a Aspectos, con la que es posible manejar y solucionar problemáticas como las auditorías y las transacciones de datos con la Base de Dato o entre capas de abstracción dentro de la aplicación, manteniendo la legibilidad y limpieza en el código que define la lógica de negocio de la aplicación en cuestión.
- Módulos ORM y JDBC abstraction and DAO: Brindan soporte a la gama de funcionalidades y problemáticas comunes en el trabajo con Bases de Datos y abstraen al programador de las acciones básicas de intercambio con la Base de Dato permitiendo que sea posible mantener el código de acceso a datos limpio y simple. Además, ORM (*Object Relational Mapping*), implementa mecanismos de enganches para lograr compatibilidad e integración con otros *frameworks* que soportan el mapeo de objetos relacionales como Hibernate e iBatis.
- Módulo MVC (*Model – View – Controller*): Brinda soporte al desarrollo de aplicaciones web basando el intercambio de los clientes (dígase navegadores web) con el servidor, en la filosofía que sigue el patrón arquitectónico del mismo nombre. Esta facilidad también emplea la IoC para lograr separar limpiamente la lógica de controlador de los objetos de negocio, también permite unir declarativamente, parámetros de una petición a objetos de negocio. Además incorpora un mecanismo de validación de formularios muy útil en la comprobación de los datos provenientes del cliente (Walls, y otros, 2005).

Spring permite el uso de POJO (*Plain Old Java Object*) revalorando la simplicidad de las clases Java y aportando manejo de transacciones de forma no intrusiva (Wikipedia, 2009).

Spring puede ser utilizado en toda clase de escenarios, mediante su uso se pueden lograr desde pequeñas aplicaciones hasta aplicaciones empresariales completamente flexibles, usando sus funcionalidades de gestión de transacciones de conjunto con una amplia integración con su *framework* para web.

Spring *Framework* no obliga a utilizar todo lo que contiene; no es una solución de "todo o nada". Aplicaciones construidas con WebWork, Struts, Tapestry, u otros *frameworks* de interfaz de usuario pueden integrarse perfectamente bien con Spring, permitiendo que se utilicen las características ofrecidas por él.

Por otra parte cuando se hace necesario tener acceso a código existente vía servicios, Spring ofrece las clases Hessian, Burlap, o Rmi de manera que el acceso a objetos remotos ya existentes es repentinamente más fácil de lograr.

Para muchos proyectos se hace necesario establecer convenciones y tener configuraciones por defecto razonables, lo cual tiene soporte explícito a través de Spring MVC. Esto significa que puedan establecerse un grupo de convenciones de nombre para los Controladores (*Controllers*) y Vistas, minimizando sustancialmente la cantidad de configuraciones XML requeridas para manejar los "*handler mappings*" y "*view resolvers*", instancias ambas del *ModelAndView* que provee MVC.

Spring cuenta con soporte para un set de configuración completo para trabajar con anotaciones, en combinación con soporte para las anotaciones de JSR-250, además mediante su uso se hace realmente fácil el trabajo con aplicaciones Orientadas a Aspectos, a través del AOP Framework de Spring, de configuración sencilla utilizando archivos XML.

A partir de su versión 2.0, Spring ofrece el *TaskExecutor,* una abstracción similar a un programador de tareas utilizado y manejado por el mismo *framework*, que posibilita diversas funcionalidades de ejecución de todo tipo de tareas que pueda necesitar una aplicación específica.

El Framework Spring provee integración con Hibernate, JDO, Oracle TopLink, iBATIS SQL Maps lo cual en términos de gestión de recursos se traduce en soporte para implementación de Acceso a Datos, y estrategias de transacciones.

1.6.2 MyMobileWeb

Como consecuencia de la gran diversidad de dispositivos móviles que existen hoy en día debido a las

distintas dimensiones de pantallas, formato de imágenes y audio que soportan, en el mundo se genera una gran polémica para los desarrolladores en la creación de páginas o sitios *Wap*.

Para dar solución a la situación expuesta anteriormente, se decide la utilización de la plataforma *Open Source* MyMobileWeb, basada en estándares abiertos, modular y de bajo coste que simplifica el desarrollo de aplicaciones y portales .mobi de calidad, proporcionando un entorno avanzado de adaptación de contenidos.

Incluye diferentes módulos que cubren todos los requisitos básicos que debe cumplir un sitio web móvil completo e integrado, ocultando a las aplicaciones toda la complejidad relacionada con la gestión de múltiples contextos de entrega. MyMobileWeb usa WURFL (*Véase* 1.5.2) para reconocer y obtener las capacidades de los dispositivos. Como característica de valor añadido, MyMobileWeb incorpora algunos módulos experimentales capaces de explotar la semántica en un entorno móvil, implementando el concepto de "Web Móvil Semántica".

MyMobileWeb está destinado al desarrollo de aplicaciones que deben ser entregadas a diferentes canales teniendo entre sus características principales ser un producto 100% Java, que sólo requiere un mínimo contenedor de Servlets/JSP, Apache Tomcat (*Véase 1.7.1*) en nuestro caso, y no está basado en la transcodificación del lenguaje de marcado, proporcionando mejor rendimiento a través del uso de técnicas avanzadas de pre generación del lenguaje de marcado para las diferentes tecnologías.

Además sus controles visuales se representan de diferentes maneras dependiendo del contexto de entrega o de las políticas de adaptación (reglas) que el programador especifica, proporcionando una avanzada interacción con el usuario y resolviendo problemas comunes del entorno móvil, como la paginación de contenidos muy largos. Por último, MyMobileWeb facilita el desarrollo de portales cuyos contenidos estén almacenados en un Sistema de Gestión de Contenidos (*Content Management System* o CMS) (InternetWebServeis, 2008).

1.6.3 iBatis Framework

Un *framework* de persistencia, es una librería de clases que facilita la tareas del programador, permitiéndole guardar objetos en bases de datos relacionales de manera lógica y eficiente, de otra manera este proceso sería realizado de forma manual, siendo tedioso, repetitivo y propenso a errores.

Para el desarrollo de nuestra aplicación optamos por la utilización de iBatis Framework, especializado en

dar soporte a las operaciones y problemáticas más comunes referentes a las interacciones entre el acceso a datos de una aplicación y la base de datos correspondiente.

Resulta un componente de software encargado de traducir entre objetos y registros, basado en capas, situado entre la lógica de negocio y la capa de origen de datos. Compuesto de dos paquetes complementarios pero independientes: iBatis *Data Access Objects* (DAO), que implementa la capa de abstracción (ibatis-dao.jar) e iBatis SQL MAPS, que implementa la capa de persistencia (ibatis-sqlmap.jar). iBatis DAO oculta los detalles de la capa de persistencia y proporciona un API común para todas las aplicaciones, mientras iBatis Data Mapper proporciona un modo simple y flexible de mover los datos entre los objetos Java y la base de datos relacional, teniendo toda la potencia de SQL[10] sin una línea de código JDBC (Begin, et al., 2007).

Para su uso no requiere de grandes esfuerzos de aprendizaje, consta de buena documentación y es un proyecto *Open Source*. Ofrece un sistema de mapeo directo en ficheros XML. Proporciona mecanismos para el acceso a procedimientos almacenados, con la debida consideración por parte del programador de no implementar lógica de negocio en los mismos.

Dada las condiciones en la que se desarrolla la aplicación, así como las características y facilidades antes mencionadas, se propones este *framework* de persistencia como una opción factible a ser utilizada en esta solución.

1.7 Contenedor de Aplicaciones utilizado

Los contenedores de aplicaciones son programas que se encuentran en un ordenador esperando de manera permanente a que algún otro ordenador realice una solicitud de conexión. En un mismo ordenador es posible tener simultáneamente servidores de diferentes tipos de servicio, cuando este ordenador recibe una petición de servicio desde otro ordenador en la red, se interpreta el tipo de llamada y se responde pasando el control de la conexión al servidor de aplicaciones correspondiente (Garcia de León, y otros, 1999).

1.7.1 ¿Por qué Apache Tomcat?

Apache Tomcat es un contenedor de *Servlets* desarrollado por la *Apache Software Foundation* (ASF, por sus siglas en inglés). Tomcat implementa las especificaciones de Java *Servlet* y Java *Server Page* de

10 Structured Query Language, traducido como Lenguaje de Consulta Estructurado.

Sun Microsystems, y proporciona un ambiente de Web Server HTTP en "Java puro" para el código de Java que se ejecuta sobre él. Puede utilizarse como un contenedor solitario (principalmente para desarrollo y depuración) o como *plug-in* para un servidor web existente (actualmente se soporta los servidores Apache, Internet Information Service y Netscape).

Apache Tomcat está compuesto por diferentes componentes que se encargan de las distintas funcionalidades que debe suplir, los mismos son relacionados a continuación:

- Catalina, contenedor de *servlets* del Tomcat, implementa las especificaciones de *Servlets* y *Server Pages* de Java.
- Coyote: es el componente Conector de HTTP de Tomcat, con soporte para el protocolo HTTP 1.1 para servidores web o contenedores de aplicación. Coyote escucha las conexiones entrantes en un puerto TCP específico del servidor y transmite la petición al Motor del Tomcat de manera que la petición pueda ser procesada y devolver una respuesta al cliente que realiza la petición.
- Jasper: es el motor de JSP de Tomcat, su trabajo es la conversión de ficheros JSP para compilarlos en código Java como *Servlets* que pueden luego ser manejados por Catalina. En tiempo de ejecución, Jasper puede detectar cambios en los JSP y recompilarlos (Brittain, y otros, 2003).

Nuestra aplicación será desarrollada usando Java como lenguaje de programación, de manera que Tomcat, una herramienta poderosa y flexible para estos fines, reconocida como la Referencia Oficial de Implementación de las tecnologías *Servlets* y *Server Pages* de Java, es nuestra mejor opción.

1.8 Gestor de Base de Datos utilizado

Los Sistemas de Gestión de Base de Datos (SGBD) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Un sistema de este tipo debe brindar diferentes funcionalidades, entre las cuales están: Abstracción de la Información, Independencia de Datos, Consistencia, Seguridad, Integridad, Respaldo, Control de Concurrencia, Manejo de Transacciones y Tiempo de Respuesta Mínimo. Estas facilidades ayudan a bajar considerablemente los tiempos de desarrollo, aumentando la calidad del sistema desarrollado si son bien explotados por los desarrolladores. Usualmente, proveen interfaces y lenguajes de consulta que simplifican la recuperación de los datos, haciendo más fácil la interacción entre los usuarios

(desarrolladores) y los datos en sí (Codd, 1990).

1.8.1 ¿Por qué PostgreSQL?

PostgreSQL es uno de los Sistemas Generadores de Bases de Datos Relacionales *Open-Source* con más usuarios en todo el mundo, incluyendo entre sus ventajas soportabilidad para casi todas las construcciones SQL, incluso extensiones orientadas a objetos (como por ejemplo la posibilidad de definir tablas mediante herencia), siendo en este sentido el motor de Bases de Datos más avanzado en código abierto.

Además posee la capacidad de permitir un amplio cúmulo de construcciones destacando entre estas las transacciones, vistas y *triggers*.

PostgreSQL cuenta con una amplia conectividad a través de JDBC y ODBC, poseyendo una amplia variedad de tipos de datos nativos, lo cual lo hace de nuestra preferencia para salvaguardar toda la información relacionada con nuestro sistema.

Otro punto a favor de su selección entre los diferentes Gestores de Bases de Datos existentes es la diversidad de herramientas de administración y conexión disponibles con que cuenta así como los clientes que provee para el desarrollo. Así por ejemplo podemos citar entre las herramientas de administración: copias de seguridad (pg_dump) y restauración (pg_restore), conexión segura de acceso a datos sobre los protocolos SSL y SSH, y clientes como pgaccess o pgmanager, una interface gráfica en TCL/TK[11] para la administración.

Su velocidad en aplicaciones de acceso múltiple concurrente es significativa, la misma es resuelta mediante un sistema denominado Control de Concurrencia Multiversión (*MultiVersion Concurrency Control* o MVCC*),* de esta forma PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos (PostgreSQL, 2008)

1.9 Herramientas de Desarrollo utilizadas

En el presente epígrafe serán detalladas las diferentes herramientas de desarrollo utilizadas en el proceso de desarrollo de este sistema, seleccionadas en correspondencia con la plataforma de desarrollo, la experiencia de los desarrolladores así como las ventajas que estas brindan.

11Tool Command Language/Tool Kit, lenguaje de herramientas de comando, utilizado para crear interfaces gráficas

1.9.1 Entorno Integrado de Desarrollo

Un IDE (*Integrated Development Environment*) es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica, que provee un marco de trabajo amigable para la mayoría de los lenguajes de programación (Wikipedia, 2008).

1.9.1.1 ¿Por qué Eclipse?

Eclipse es una poderosa plataforma de código abierto que permite integrar diferentes aplicaciones para construir un entorno integrado de desarrollo. Su principal aplicación es JDT (*Java Development Tool* o Herramienta de Desarrollo para Java), además de una serie de *plug-ins* que se encuentran acompañando a la plataforma, tales como: Ant, Compare, Core, CVS, Debug, Help, Jface, Releng, Scripting, Search, SWT, Text, UI, Update, Team y WebDAV (IBM, 2007).

El usuario se comunica con Eclipse por intermedio del *framework* que se inicia al ejecutar Eclipse (*workbench*). El *workbench* es la interfaz de usuario de la plataforma que está compuesto de un conjunto de vistas, editores y perspectivas. Las herramientas integradas a Eclipse operan en archivos del espacio de trabajo (*workspace*) del usuario. El *workspace* consta de uno o más proyectos donde cada uno se mapea a un directorio especificado por el usuario en el sistema de ficheros (IBM, 2007).

Eclipse se considera como uno de los IDE más poderosos para el desarrollo de aplicaciones en Java, provee soporte mediante *plug-ins* para trabajar con entornos J2SE (*Java 2 Platform, Standard Edition*) y J2EE (*Java 2 Platform, Enterprise Edition*) utilizados en el desarrollo de este trabajo. Con la adición de estos *plug-ins* se puede facilitar la automatización de diferentes tareas para los desarrolladores, tales son los casos del *plug-in* JBoss IDE y de un *plug-in* que permite la administrar el control de versiones de un proyecto. Seguidamente abordaremos más sobre el mismo.

1.9.2 Control de Versiones

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. Estos, pueden ser clasificados de acuerdo a la manera en que se realiza el almacenamiento del código de las siguientes formas:

Centralizados: cuando existe un repositorio centralizado de todo el código, del cual es responsable un

único usuario (o conjunto de ellos). Se facilitan las tareas administrativas a cambio de reducir la potencia y flexibilidad, pues todas las decisiones fuertes (como crear una nueva rama) necesitan la aprobación del responsable.

Distribuidos: si se aumenta la capacidad de decisión distribuida. Esto da más flexibilidad pero puede dificultar bastante la sincronización (Wikipedia, 2007).

1.9.2.1 SVN

Subversion es un sistema de control de versiones de código abierto. Subversion maneja ficheros y directorios, así como los cambios hechos sobre ellos a través del tiempo, esta capacidad permite la recuperación de versiones anteriores de la información que se trata o examinar el historial de cambios de dicha información. En este aspecto puede pensarse en subversion como una especie de “máquina del tiempo” para ficheros.

Subversion puede operar a través de la red, lo cual permite su uso desde diferentes puestos de trabajo en un equipo de desarrollo, incluso en algunos niveles permite manejar y modificar los mismos datos desde diferentes ubicaciones, lo cual sin dudas incrementa la colaboración laboral, teniendo en cuenta que las modificaciones progresivas no son hechas a través de un único conducto y que no se pone en riesgo la integridad de la información, la cual siempre puede ser recuperada en caso de cambios errados.

Algunos sistemas de control de versiones son además sistemas de control o administración de configuraciones (*Source Code Management* o SCM), estos sistemas son modificados específicamente para manejar grandes cantidades de código fuente y cuentan con diversas funcionalidades especificas del desarrollo de software, así por ejemplo, comprensión de lenguajes de programación, o suministro de herramientas de desarrollo de software. Subversion no forma parte de estos sistemas, en su lugar, es un sistema general que puede ser usado para manejar cualquier colección de ficheros, desde código fuente, archivos de audio, video, imágenes, documentos u hojas de cálculo (Pilato, y otros, 2004).

1.9.3 Herramienta de Modelado

Las herramientas de modelado, son herramientas que se emplean para la creación de modelos de sistemas que ya existen o que se desarrollarán. Estas permiten crear una representación del sistema, a bajo costo y riesgo mínimo. A bajo costo porque es un conjunto de gráficos y textos que representan el sistema, pero no son el sistema físico real, el cual por supuesto es más costoso. Además minimizan los riesgos, porque los cambios que se deban realizar, por errores o cambios en los requerimientos, se

pueden realizar más fácil y rápidamente sobre el modelo que sobre el sistema ya implementado.

1.9.3.1 ¿Por qué Visual Paradigm?

Visual Paradigm para UML (VP-UML) es una herramienta UML (*Unified Modeling Language*) profesional, que soporta todo el ciclo de vida de desarrollo de software. Utilizar UML, ayuda a una rápida construcción de aplicaciones de calidad, mejores y con menor coste.

Visual Paradigm para UML está diseñado para una amplia gama de usuarios, incluyendo herramientas de modelado y documentación que cubren desde los procesos del negocio hasta las especificaciones detalladas de un sistema, soportando más de 20 tipos de diagramas, minimizando los tiempos invertidos y aumentando la eficacia de modelar.

Visual Paradigm puede ejecutase de diferentes formas, el mismo pude ser instalado, o portable, es multiplataforma y posee variados ambientes, es capaz de generar código para Java y .NET, de manera que pueden desarrollarse fácilmente proyectos escritos en estos lenguajes o hacer ingeniería inversa a partir de su código fuente para generar diagramas. Además, para la selección de esta herramienta, se tomó en cuenta la experiencia del grupo de desarrollo y sus conocimientos sobre la misma.

1.10 Conclusiones

En este capitulo fueron tratados los principales conceptos abordados en la investigación, se realizó un análisis del estado de los servicios de valor agregado, se hizo un estudio sobre algunos de los servicios de envío de horóscopos a dispositivos móviles en el mundo y fue detallada la metodología, plataforma, tecnologías y herramientas de desarrollo utilizadas en la construcción del presente sistema.

capítulo 2

CAPÍTULO 2: CARACTERÍSITICAS DEL SISTEMA

En el presente capítulo se plantean los objetivos estratégicos de Cubacel, se explican los procesos que serán objetos de automatización, así como la propuesta de sistema para el servicio de subscripción de horóscopos a través de dispositivos móviles, haciendo énfasis en el flujo de procesos del mismo y su arquitectura. También se hace un análisis comparativo de las soluciones existentes con la nueva propuesta planteada en el presente trabajo.

2.1 Objetivos estratégicos de la organización

Cubacel provee de soluciones y servicios a sus clientes dentro del mundo de la telefonía móvil, prestando especial atención al empleo de las nuevas tecnologías de acceso a contenidos, nuevos modelos de dispositivos móviles, así como nuevos formatos multimedia. Constantemente actualiza su infraestructura tecnológica garantizando capacidad inmediata de respuesta a los distintitos modelos de negocios que puedan presentar operadores, clientes y proveedores de contenidos. Actualmente el principal objetivo de la entidad es adentrarse en el mercado de las aplicaciones y servicios para dispositivos móviles.

2.2 Objeto de Automatización

Serán objeto de automatización en esta aplicación los procesos de subscripción, confirmación, publicación y gestión de acceso que tienen lugar en el servicio de subscripción de horóscopos mediante dispositivos móviles. Los mismos pueden ser desglosados de la siguiente forma:

- Procesamiento de solicitud: proceso mediante el cual se crean las pre-órdenes que contienen todos los datos referentes a la solicitud de servicio por parte del cliente realizada desde una página Web.
- Envío y Recepción de mensajes SMS: proceso mediante el cual se envían y recogen los mensajes de confirmación de servicio enviados desde y hacia los terminales de usuario.
- Personalización de Servicio: proceso mediante el cual los usuarios pueden personalizar la manera en que recibirán la información de su horóscopo de manera simple mediante una interfaz Web.
- Conformación de Horóscopos, proceso mediante el cual se crea la información que será consultada por los usuarios luego de realizar su subscripción.

- Publicación de Contenido: proceso encargado de la publicación en una interfaz *Wap* del contenido a consultar por el cliente, previo envío de la dirección donde será publicado el mismo a través de un mensaje *Wap-Push*.
- Acceso al Contenido: proceso responsable de permitir el acceso por parte de los clientes a los contenidos específicos que le corresponden.

2.3 Propuesta de Sistema

Se propone la puesta en marcha de un sistema que añada un nuevo servicio de valor agregado a la plataforma que se desarrolla actualmente para Cubacel, a través de un sistema multicapa distribuido. De esta manera se decidió la implementación de una aplicación que permita a los usuarios de dicha plataforma subscribirse a un servicio de horóscopos y recibir el mismo a través de dispositivos móviles.

2.3.1 Flujo de procesos del sistema propuesto

El Usuario accede a una página Web que le permite realizar una solicitud de subscripción al servicio de horóscopos, el módulo Web se encarga de procesar la petición originada y crea una pre-orden de subscripción de servicio que es almacenada en la base de datos, haciendo uso del módulo *Core* (núcleo), registrando este último los datos del Usuario de interés para la aplicación, así como su número de teléfono y frecuencia de recibo del servicio. Esta pre-orden contiene además los campos fecha de creación y estado (el estado es una variable lógica que tiene valor “falso” cuando el usuario todavía no ha activado el servicio mediante confirmación y “verdadero” en caso contrario). Luego de realizar este proceso el sistema comienza la lógica de verificación de autenticidad de los datos provistos por parte del cliente que solicita el servicio de horóscopo usando los datos adquiridos a través del módulo de subscripciones, creando un mensaje que es enviado al usuario como petición de confirmación de servicio, al cual deberá responder según su intención de recibir o no el servicio, mediante este proceso se confirma la existencia y legitimidad del número de teléfono provisto en el proceso de subscripción . Una vez el usuario notifica su intención, el sistema activa la subscripción de dicho usuario (cambia el estado inicial de la pre-orden) en caso afirmativo, y envía al usuario la información de acceso a la información correspondiente. En este punto comienza el proceso de publicación de la información de horóscopo del usuario en el módulo correspondiente al sitio *Wap*, a la cual puede acceder a través de una URL que le es enviada por el sistema mediante un SMS según la frecuencia de recibo seleccionada.

2.3.2 Arquitectura Propuesta

Para la organización fundamental del sistema se definió una arquitectura Java EE que tiene como objetivo principal aportar elementos que ayuden a la toma de decisiones. La arquitectura Java EE define un modelo de programación encaminado a la creación de n-capas (Jendrock, 2006). En nuestro caso, tomamos dicho modelo para implementar un sistema de tres capas. La Capa de Cliente representa la interfaz de usuario que maneja el cliente, la Capa de Presentación representa el conjunto de componentes que generan la información que se representa en la interfaz de usuario del cliente, la Capa de Servicio contiene los componentes de negocio reutilizables y componentes que permiten hacer más transparente el acceso a la Capa de Datos. Esta última engloba nuestro sistema de información. La ventaja de utilizar un modelo como este es que permite hacer modificaciones en cualquiera de las capas sin tener que interferir en las demás, además de obtener mejoras en cuanto a mantenimiento, extensibilidad y reutilización de componentes. La arquitectura del sistema se detalla a continuación en la Fig. 7.

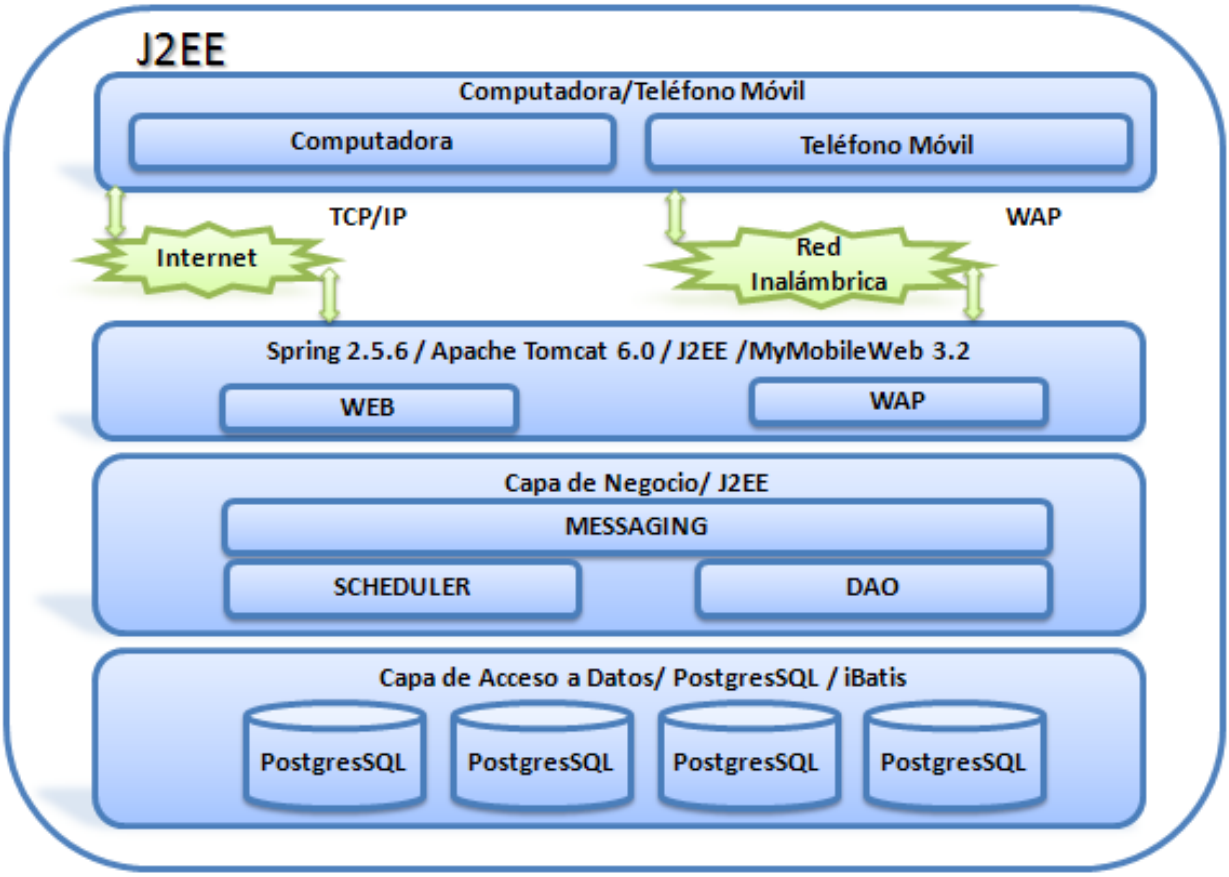


Fig. 7 Arquitectura del Sistema

Para maximizar las posibilidades de integración así como minimizar las dependencias funcionales del sistema, se decidió confeccionar tres módulos principales que serán desplegados como aplicaciones diferentes, manteniendo los niveles de operatividad y rendimiento. Estos módulos fueron confeccionados a partir de las funcionalidades que debían tenerse en cuenta en esta aplicación y en principio establecerán comunicación haciendo uso de las tecnologías Java RMI. Para su mejor entendimiento, esta idea es detallada a continuación en la Fig. 8.

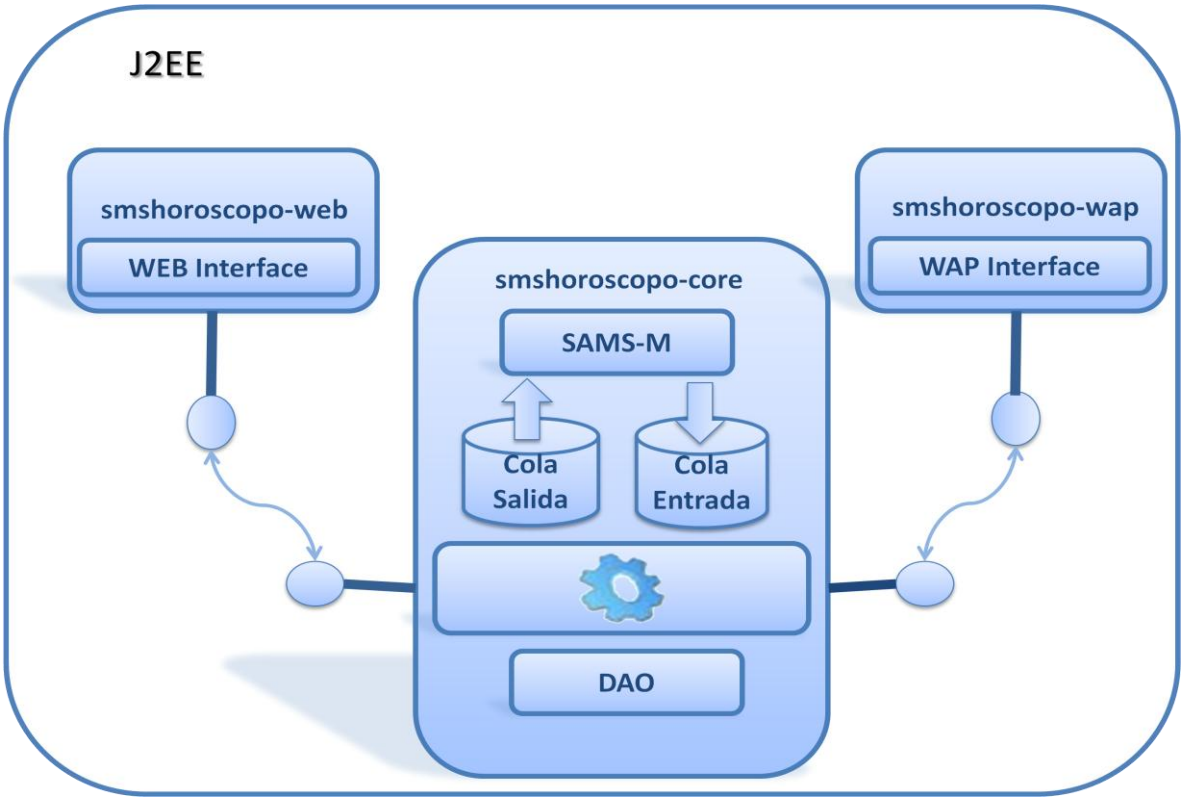


Fig. 8 Aplicaciones del Sistema (Distribución)

De esta manera dos aplicaciones diferentes se encargarán de la lógica concerniente a las interfaces de usuario y su interacción con el mismo, mientras una tercera aplicación hará las veces de núcleo, siendo responsable de la lógica de negocio de la aplicación así como de la interacción con la base de datos.

2.3.3 Patrones de Diseño aplicados

Un patrón de diseño es una buena práctica documentada, o solución, que se ha aplicado con éxito en ambientes múltiples para solucionar un problema que se repite en los sistemas de dichos ambientes y en situaciones específicas (Alexander, y otros, 1997).

2.3.3.1 Patrón Interfaz (Interface)

Aplicando el patrón interfaz, los servicios comunes ofrecidos por diversas clases del proveedor de servicio se pueden abstraer hacia fuera y declarar como un interfaz independiente. Cada uno de las clases del proveedor de servicio se puede diseñar luego como ejecutores de este interfaz común.

El patrón interfaz es de vital importancia en la implementación de aplicaciones de varios módulos, máxime si la comunicación entre estas se lleva a cabo a través de Java RMI, puesto que dicha comunicación se lleva a cabo a través de interfaces en ambos lados del canal de transmisión.

2.3.3.2 Patrón Solitario (Singleton)

Este patrón se encarga de garantizar que una clase sólo tiene una única instancia, proporcionando un punto de acceso global a la misma (Gamma, y otros, 1995).

En nuestra aplicación se realiza el mapeo de datos utilizando iBatis, de esta forma se hace sumamente importante el manejo de este patrón, puesto que la clase *IbatisUtil*, debe ser instanciada una sola vez de manera global.

2.3.3.3 Patrón Fachada (Facade)

Con este patrón se proporciona una interfaz unificada de alto nivel que, representando a todo un subsistema, facilita su uso. La “fachada” satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas (Gamma, y otros, 1995).

El patrón fachada es usado en diferentes situaciones, entre ellas cuando se pretende proporcionar una interfaz simple para un subsistema complejo, o cuando es necesario separar al cliente de los componentes de un subsistema, reduciendo así el número de objetos con los que el cliente trata, y facilitando el uso del subsistema.

2.3.3.4 Patrón Proxy

En este patrón un objeto en un contexto es representado por otro (el proxy) en un contexto diferente. El proxy sabe cómo redirigir las invocaciones a métodos entre los distintos objetos participantes (Gamma, y otros, 1995).

De manera general su lógica establece que un “apoderado remoto “proporcione un “representante local” para un objeto en un espacio de direcciones diferente, estos representantes también son conocidos como “embajadores”. Mediante este mecanismo se introduce un nivel de “*indirección*” en el acceso al objeto, que permite al apoderado remoto ocultar el hecho de que el objeto reside en un espacio de

direcciones distinto, siendo esta precisamente la fórmula utilizada por Java RMI para su implementación.

2.3.3.5 Patrón DAO

El patrón DAO[12] posibilita trabajar con objetos de acceso a datos para abstraer y encapsular los accesos a las distintas fuentes de datos (base de datos, archivos o servicios externos) que pueda tener una aplicación. El DAO maneja las conexiones con las fuentes de datos para obtener y almacenar datos persistentes o temporales. Mediante su uso se hace posible ocultar la complejidad del uso de JDBC a las capas superiores.

2.4 Análisis comparativo de otras soluciones existentes con la propuesta

En la etapa inicial del desarrollo del sistema fueron analizados algunos servicios de envío de horóscopos ofrecidos por otras plataformas de entrega de contenidos. Luego de realizar un profundo estudio sobre estos, se eligieron las características más afines, con el objetivo de incorporarlas a la solución que se plantea en el presente trabajo. Del servicio de Telefónica Movistar, se seleccionó la característica de personalizar el servicio, ya que para el usuario es más fácil acceder a una página Web y mediante un asistente insertar los datos necesarios para la personalización de su contenido. Además se eligió la característica de que el servicio sea inicializado por un mensaje enviado por el cliente, que en nuestro caso sirve como confirmación de la solicitud de servicio y de autenticidad del cliente.

Se definieron nuevas características que no fueron encontradas en las soluciones estudiadas y que eran de interés para la nueva propuesta. Estas características se exponen a continuación:

- La información del horóscopo será publicada en una interfaz *Wap*, brindando al usuario la posibilidad de consultar la misma en el momento que lo desee.
- Los usuarios pueden seleccionar la frecuencia de recibo de su horóscopo.
- Se enviará la dirección donde está publicada la información a través de una notificación *Wap-Push.*

Se considera que las características tomadas de otros sistemas así como las añadidas en la nueva propuesta, son las que más se ajustan al entorno en que nuestro sistema se va a desenvolver. De igual forma, consideramos que la nueva propuesta mejora en algunos aspectos la solución del sistema de envío de horóscopos de *Movistar*, donde el usuario realiza su solicitud a través de la Web, pero recibe su información a través de un mensaje de texto, al igual que en el resto de las soluciones estudiadas, lo cual

12 *Data Access Object*, traducido como Objetos de Acceso a Datos.

no es aplicable a nuestras condiciones teniendo en cuenta que uno de los objetivos de la nueva propuesta era que el usuario consultara su información a través de una interfaz *Wap* en el momento que lo deseara.

2.5 Conclusiones

En este capítulo fueron explicados los diferentes objetivos estratégicos de Cubacel, se expuso la propuesta de sistema para el servicio de subscripción, envío y recepción de horóscopos, se definió la arquitectura a seguir y el flujo de procesos del mismo. Asimismo fue realizado un análisis comparativo con otras soluciones existentes definiendo de una manera crítica las diferencias y similitudes que existen entre estas soluciones y la propuesta de este trabajo.

capítulo 3

CAPÍTULO 3: EXPLORACIÓN Y PLANIFICACIÓN

En el presente capítulo se describen las fases de exploración y planificación pertenecientes a la metodología de desarrollo utilizada para la implementación del sistema que se propone, haciendo énfasis en los artefactos generados durante el transcurso de las mismas.

3.1 Fase de Exploración

Durante esta fase se realiza el proceso de identificación de las historias de usuario, durante la misma los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo.

3.1.1 Historias de Usuario

Las historias de usuario son la forma en que se especifican en *Extreme Programming* los requisitos del sistema. Éstas se escriben desde la perspectiva del cliente aunque los desarrolladores pueden brindar ayuda en su identificación. El contenido de las mismas debe ser concreto y sencillo. El tiempo de desarrollo ideal para una historia de usuario es entre una y tres semanas (Beck, 1999).

Las historias de usuario también conducen el proceso de los *test* de aceptación, que son empleados para verificar que las historias han sido implementadas correctamente. Existen diferencias entre las historias de usuario y la tradicional especificación de requisitos, siendo su principal diferencia el nivel de detalle. Las historias de usuario solamente proporcionan los detalles sobre la estimación de riesgo y cuánto tiempo conllevará la implementación de dicha historia (Fowler, 2004).

Durante esta fase se identificaron seis historias de usuario, las cuales se exponen a continuación.

Tabla 1. Historia de Usuario Gestionar Solicitud de Servicio

Historia de Usuario	
Número: 1	**Nombre:** Gestionar Solicitud de Servicio
Usuario: Cliente	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 2	**Iteración Asignada:** 1
Descripción: El cliente accede a una página Web donde está publicado el servicio de subscripción e inserta sus datos a través de un formulario, se genera una pre-orden con los datos recibidos y se le envía un mensaje solicitando que confirme su intención de recibir el servicio.	
Observaciones:	

Tabla 2. Historia de Usuario Gestionar Orden

Historia de Usuario	
Número: 2	**Nombre:** Gestionar Orden
Usuario: Cliente	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 2	**Iteración Asignada:** 2
Descripción: Se recibe un mensaje de respuesta de confirmación de parte del cliente, especificando su intención de recibir el servicio, en caso afirmativo se le envía la dirección donde está publicado su contenido dependiendo de la frecuencia de recibo seleccionada.	
Observaciones: Las informaciones publicadas tendrán un tiempo de vida de 24 horas, luego serán actualizadas.	

Tabla 3. Historia de Usuario Chequear Existencia de pre-orden

Historia de Usuario	
Número: 3	**Nombre:** Chequear Existencia de pre-orden
Usuario: Cliente	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo
Puntos Estimados: 1	**Iteración Asignada:** 2
Descripción: Una vez la aplicación recibe un mensaje de parte de un cliente especificando que desea recibir el servicio, debe verificarse que exista una pre-orden a nombre de dicho cliente. Asimismo cuando realiza una solicitud de subscripción se chequea que no exista una subscripción previa.	
Observaciones:	

Tabla 4. Historia de Usuario Gestionar Servicio

Historia de Usuario	
Número: 4	**Nombre:** Gestionar Servicio
Usuario: Cliente	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 2	**Iteración Asignada:** 2
Descripción: Si el cliente decide continuar con el servicio, comienza el proceso de publicación y envío a partir de ese momento de la dirección donde se encuentra la información de su horóscopo.	
Observaciones:	

Tabla 5. Historia de Usuario Crear Orden

Historia de Usuario	
Número: 5	**Nombre:** Crear Orden
Usuario: Cliente	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo
Puntos Estimados: 1	**Iteración Asignada:** 2
Descripción: Con los datos introducidos en la historia de usuario Gestionar Orden se completa la pre-orden y se crea una nueva orden.	
Observaciones: El estado de la pre-orden cambia**.**	

Tabla 6. Historia de Usuario Publicar Servicio de Horóscopo

Historia de Usuario	
Número: 6	**Nombre:** Publicar Servicio de Horóscopo
Usuario: Cliente	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Baja
Puntos Estimados: 2	**Iteración Asignada:** 2
Descripción: El cliente accede al contenido de su horóscopo publicado en una interfaz *Wap* a través de un teléfono celular dependiendo de la frecuencia de recibo.	
Observaciones:	

3.2 Fase de Planificación

Durante la fase de planificación se establece por parte del cliente la prioridad de las diferentes historias de usuario, y en correspondencia los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Las estimaciones de esfuerzo asociado a la implementación de las historias son establecidas por los programadores utilizando como medida la cantidad de puntos. Un punto equivale a una semana ideal de programación.

3.2.1 Estimación de esfuerzo por historias de usuario

Para el desarrollo del sistema propuesto se realizó una estimación del esfuerzo para cada una de las historias de usuario identificadas, los resultados de este estudio se muestran en la siguiente tabla.

Tabla 7. Estimación de Esfuerzo por Historias de Usuario

Historias de usuario	Puntos estimados
Gestionar Solicitud de Servicio	2
Gestionar Orden	2
Chequear Existencia de pre-orden	1
Gestionar Servicio	2
Crear Orden	1
Publicar Servicio de Horóscopo	2

3.2.2 Plan de Iteraciones

Una vez identificadas las Historias de Usuario del sistema y estimado el esfuerzo dedicado a la realización de cada una de estas, se procede a la planificación de la etapa de implementación del proyecto. Para dar cumplimiento a dicha tarea se decidió realizar la misma en tres iteraciones, las cuales son explicadas a continuación.

3.2.2.1 Primera iteración

Esta iteración tiene como objetivo fundamental la implementación de las historias de usuario de mayor prioridad en el sistema. Durante ésta será creada la mayor parte de la arquitectura del sistema, encapsulada en el módulo **smshoroscope-core**. Al final de esta iteración se contará con un prototipo del sistema, capaz de gestionar el servicio de horóscopos. Este prototipo será presentado al cliente, con la finalidad de lograr una retroalimentación para el grupo de trabajo (*Véase Fig. 9*).

3.2.2.2 Segunda iteración

El objetivo de esta iteración es la implementación de las funcionalidades del módulo **smshoroscope-web**. Durante ésta se gestiona el acceso por parte de los clientes a la página de subscripción brindada por la aplicación del servicio a través de un navegador mediante una computadora (*Véase Fig. 10*).

3.2.2.3 Tercera iteración

El objetivo de esta iteración es la implementación de las últimas funcionalidades del módulo **smshoroscope-core** y del módulo **smshoroscope-wap**. Durante la primera parte de la iteración se gestionará todo lo referente al envío y recepción de mensajes de texto SMS entre el cliente y la aplicación. En la segunda parte de la iteración se implementará el completamiento del servicio, específicamente la publicación y acceso del cliente al contenido del horóscopo en la interfaz *Wap*. Al final de esta iteración se contará con una aplicación empresarial capaz de realizar la subscripción, confirmación, publicación y gestión de acceso al servicio de horóscopos (*Véase Fig. 11*)

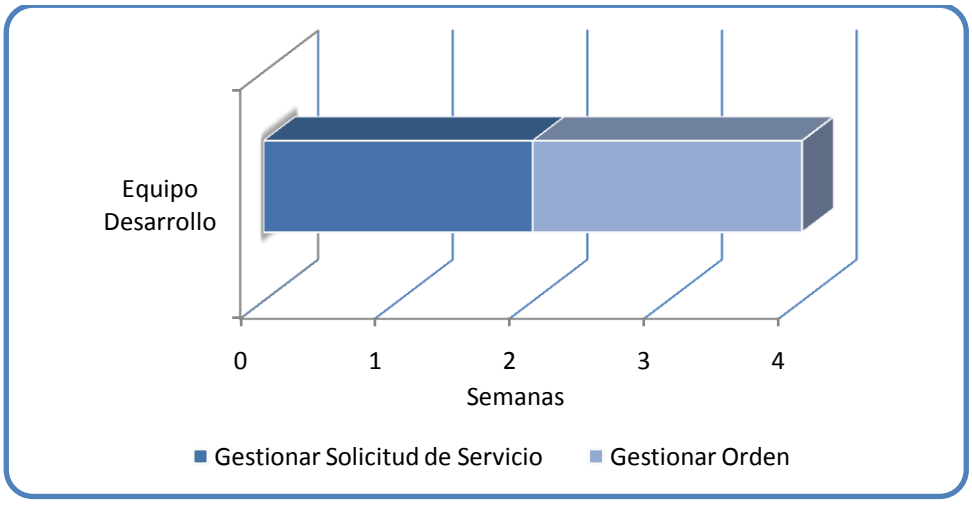


Fig. 9 Plan de iteración 1

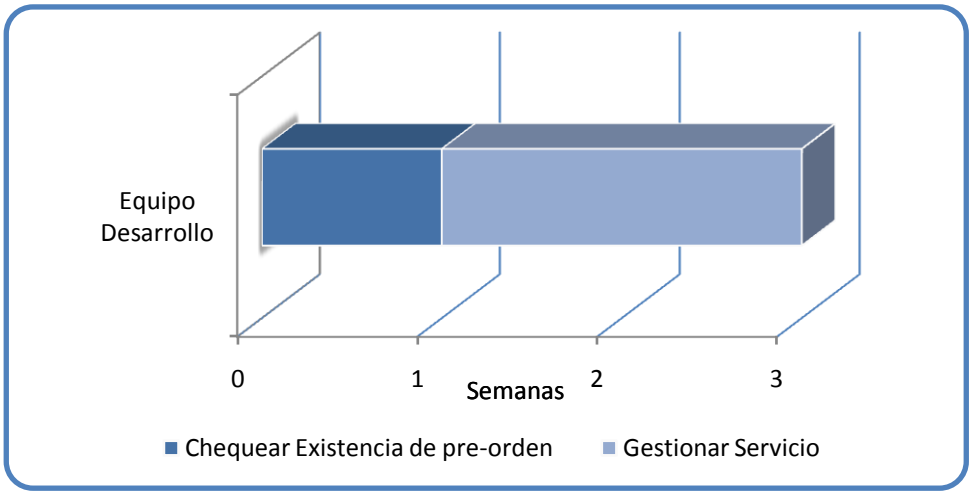


Fig. 10 Plan de iteración 2

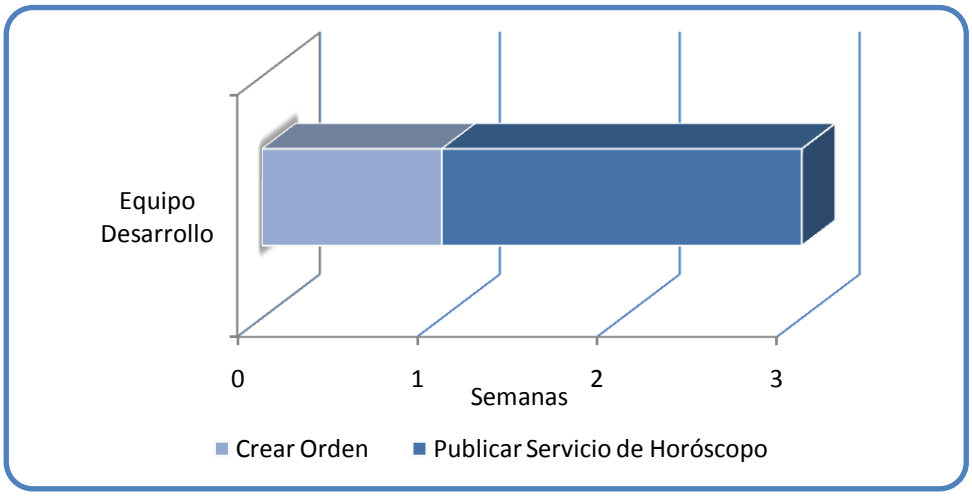


Fig. 11 Plan de iteración 3

3.2.3 Plan de Entrega

A continuación se presenta el plan de entregas para la fase de implementación. Como resultado del mismo se pueden ver las fechas reales en que cada versión será entregada. El motivo por el que se crean nuevas versiones de algunos de los módulos puede ser por cambios desde la última versión o simplemente actualización de las dependencias en otros módulos cuando una nueva versión de éstos últimos es entregada.

Tabla 8. Plan de Entrega por Fechas

Módulo	2da semana de abril	1ra semana de mayo	1ra semana de junio
smshoroscope-core	v0.1	v0.2	v0.3
smshoroscope-wap		v0.1	v0.2
smshoroscope-web	v0.1		v0.2

3.3 Conclusiones

En el presente capítulo fueron abordados los contenidos de las fases de exploración y planificación de la metodología utilizada para el desarrollo del sistema, siendo generados los artefactos de las mismas, se explicaron las historias de usuario y se definió el trabajo a realizar en cada una de las iteraciones del proyecto, así como el plan de entrega de cada uno de los módulos.

capítulo 4

CAPÍTULO 4: DISEÑO, IMPLEMENTACIÓN Y PRUEBAS

En el presente capítulo se describen las fases de diseño, implementación y pruebas propias de la metodología de desarrollo XP. Se detallan las tres iteraciones realizadas durante la etapa de codificación del proyecto, así como las tareas de programación desarrolladas en cada una de estas. También se hace referencia a las pruebas de aceptación realizadas sobre el sistema.

4.1 Fase de Diseño

Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra deben ser evitados. En todo momento, el diseño adecuado para el software es aquel que supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos (Fowler, 2004).

Para el diseño de las aplicaciones, la metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando notación UML. En su lugar se usan otras técnicas como las tarjetas CRC[13] como una extensión informal a UML. La técnica de las tarjetas CRC se puede usar para guiar el sistema a través de análisis guiados por la responsabilidad. Las clases se examinan, se filtran y se refinan en base a sus responsabilidades con respecto al sistema, y las clases con las que necesitan colaborar para completar sus responsabilidades (Wake, 1999). No obstante el uso de los diagramas de clases UML puede aplicarse siempre y cuando mejore la comunicación, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante (Fowler, 2004).

Los diagramas utilizados durante el desarrollo del sistema de subscripción de horóscopos pueden ser consultados en el *Anexo 2.*

13 Tarjetas CRC (Clase, Responsabilidad y Colaboración) son una metodología para el diseño de software orientado por objetos creada por Kent Beck y Ward Cunningham en 1989, publicada en "*A Laboratory For Teaching Object-Oriented Thinking*".

4.2 Fase de Implementación

Durante el inicio de cada iteración, se lleva a cabo una revisión del plan de iteraciones y se modifica de ser necesario (Fowler, 1999).Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable. Estas tareas, pueden escribirse utilizando un lenguaje técnico y no necesariamente deben ser entendibles para el cliente. (Beck, 1999).

A continuación se detalla cada una de las iteraciones del sistema, desarrolladas según la planificación propuesta.

4.2.1 Primera iteración

Durante el transcurso de esta iteración se implementaron las historias de usuario de mayor prioridad en el sistema, construyéndose la base de la arquitectura. Se logró crear un prototipo del sistema capaz de gestionar el servicio de horóscopo y enviar el mismo a los clientes.

Tabla 9. Historias de Usuario desarrolladas en la primera iteración

Historia de Usuario	Estimación	Real
Gestionar Solicitud de Servicio	2	2
Gestionar Orden	2	2
Total	**4**	**4**

4.2.1.1 Tareas de las historias de usuario desarrolladas en la primera iteración

Historia de usuario Gestionar Solicitud de Servicio

Tabla 10. Tarea 1 de la historia de usuario Gestionar Solicitud de Servicio

Tarea	
Número tarea: 1	**Número historia:** 1
Nombre tarea: Recibir Petición de solicitud de servicio	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0,4
Fecha inicio: 16/03/09	**Fecha fin:** 20/03/09
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Se implementará la manera en que serán registradas las solicitudes de subscripción de parte de los clientes en la aplicación, a través de una computadora mediante un navegador Web.	

Tabla 11. Tarea 2 de la historia de usuario Gestionar Solicitud de Servicio

Tarea	
Número tarea: 2	**Número historia:** 1
Nombre tarea: Procesar solicitud de servicio	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 22/03/2009	**Fecha fin:** 30/03/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: A partir de la solicitud, se crea una orden de servicio que es almacenada el la base de datos. Como el usuario no ha verificado su existencia, a la orden se le llama pre-orden. Esta pre-orden esta compuesta por la fecha de creación, datos de usuario, frecuencia de recibo, estado (“falso” si la orden no esta completa y “verdadero” en caso contrario) y código de la orden, entre otros.	

Tabla 12. Tarea 3 de la historia de usuario Gestionar Solicitud de Servicio

Tarea	
Número tarea: 3	**Número historia:** 1
Nombre tarea: Enviar mensaje de servicio (SMS)	
Tipo de tarea : Desarrollo	**Puntos estimados:** 0,6
Fecha inicio: 30/03/2009	**Fecha fin:** 02/04/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: A partir de la creación de la orden, se compone un mensaje SMS que lleva consigo una petición de verificación al cliente para establecer si de verdad desea recibir el servicio así como su autenticidad. Este mensaje será enviado a través de la cola de salida del sistema, por donde saldrán todos los mensajes que genere la aplicación.	

Historia de usuario Gestionar Orden

Tabla 13. Tarea 1 de la historia de usuario Gestionar Orden

Tarea	
Número tarea: 1	**Número historia:** 2
Nombre tarea: Eliminar órdenes expiradas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0,4
Fecha inicio: 02/04/2009	**Fecha fin:** 05/04/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: A partir de que una orden se crea tiene un tiempo de vida de 24 horas. Si el usuario en este tiempo no personaliza su horóscopo la orden expira.	

Tabla 14. Tarea 2 de la historia de usuario Gestionar Orden

Tarea	
Número tarea: 2	**Número historia:** 2
Nombre tarea: Enviar mensaje *Wap-Push*.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 0,8
Fecha inicio: 04/04/2009	**Fecha fin:** 09/04/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Se crea un mensaje a partir de los datos introducidos por el usuario para recibir su servicio de horóscopos, en el cual le es enviada la URL donde está publicada su información. El mensaje es enviado a través de la cola de salida del sistema.	

Tabla 15. Tarea 3 de la historia de usuario Gestionar Orden

Tarea	
Número tarea: 3	**Número historia:** 2
Nombre tarea: Planificador encargado de gestionar las órdenes	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0,8
Fecha inicio: 10/04/2009	**Fecha fin:** 15/04/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Se implementará un servicio que se active cada vez que se introduzca una pre-orden, que será el encargado de cambiar el estado de las pre-órdenes, decidir luego los destinatarios según frecuencia de recibo, y de enviar los mensajes de texto a los destinatarios correspondientes.	

4.2.2 Segunda iteración

Durante el transcurso de la iteración fundamentalmente se implementaron las funcionalidades del módulo **smshoroscope-web**. Se logró crear una primera vista de la interfaz *Web* donde el usuario puede subscribirse al servicio.

Tabla 16. Historias de usuario desarrolladas en la segunda iteración

Historia de Usuario	Estimación	Real
Chequear Existencia de pre-orden	1	1
Gestionar Servicio	2	2
Total	**3**	**3**

4.2.2.1 Tareas de las historias de usuario desarrolladas en la segunda iteración

Historia de usuario Chequear Existencia de pre-orden

Tabla 17. Tarea 1 de la historia de usuario Chequear Existencia de pre-orden

Tarea	
Número tarea: 1	**Número historia:** 3
Nombre tarea: Crear servlet de escucha.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 16/04/2009	**Fecha fin:** 20/04/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Se implementará un *servlet* encargado de estar a la escucha de las peticiones de solicitud de subscripción que se realicen por los usuarios, verificando si existen solicitudes previas.	

Historia de usuario Gestionar Servicio

Tabla 18. Tarea 1 de la historia de usuario Gestionar Servicio

Tarea	
Número tarea: 1	**Número historia:** 4
Nombre tarea: Crear interfaz *Wap* para consultar la información publicada.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0,6
Fecha inicio: 23/04/2009	**Fecha fin:** 25/04/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Se creará una interfaz *Wap* que permita al usuario consultar su horóscopo.	

Tabla 19. Tarea 2 de la historia de usuario Gestionar Servicio

Tarea	
Número tarea: 2	**Número historia:** 4
Nombre tarea: Cancelar servicio	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0,8
Fecha inicio: 26/04/2009	**Fecha fin:** 30/04/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Si el usuario selecciona la opción de cancelar servicio, o decide no confirmar que desea recibir el servicio, la orden se pone en estado de expirada, a partir de la fecha en que fue confeccionada y lista para luego ser eliminada.	

Tabla 20 . Tarea 3 de la historia de usuario Gestionar Servicio

Tarea	
Número tarea: 3	**Número historia:** 4
Nombre tarea: Continuar con el servicio	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0,6
Fecha inicio: 01/05/2009	**Fecha fin:** 04/05/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Si el usuario selecciona la opción de darle seguimiento al servicio, le será enviado un mensaje con la URL de publicación de su contenido para que pueda consultarlo, pasando su pre-orden a ser activada y convertirse en orden.	

4.2.3 Tercera iteración

Durante el transcurso de la tercera iteración se implementaron las funcionalidades del módulo **smshoroscope-wap,** que proporcionarán al usuario la forma de consultar la información de su horóscopo; así como las últimas funcionalidades del módulo **smshoroscope-core**, encargadas de la mensajería desde y hacia la aplicación. Al finalizar esta fase, se obtendrá una primera versión del producto. Antes de salir esta versión al mercado debe ser sometido a un periodo de pruebas.

4.2.3.1 Tareas de las historias de usuario desarrolladas en la tercera iteración

Tabla 21. Historias de usuario desarrolladas en la tercera iteración

Historia de Usuario	**Estimación**	**Real**
Crear Orden	1	1
Publicar Servicio Horóscopo	2	2
Total	**3**	**3**

Historia de usuario Crear Orden

Tabla 22. Tarea 1 de la historia de usuario Crear Orden

Tarea	
Número tarea: 1	**Número historia:** 5
Nombre tarea: Crear Mensaje de Acceso	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0,4
Fecha inicio: 06/05/2009	**Fecha fin:** 09/05/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Se creará el mensaje que será enviado al cliente según su frecuencia de recibo para que consulte su horóscopo, el mismo deberá contener la URL donde se encuentra publicado dicho contenido.	

Tabla 23. Tarea 2 de la historia de usuario Crear Orden

Tarea	
Número tarea: 2	**Número historia:** 5
Nombre tarea: Enviar Mensaje de Acceso	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0,6
Fecha inicio: 09/05/2009	**Fecha fin:** 13/05/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Se enviará el mensaje al cliente conteniendo la URL de su información. El mensaje será enviado a través de la cola de salida de la aplicación.	

Historia de usuario Publicar Servicio Horóscopo

Tabla 24. Tarea 1 de la historia de usuario Publicar Servicio Horóscopo

Tarea	
Número tarea: 1	**Número historia:** 6
Nombre tarea: Publicar interfaz *Wap* para acceder a la información	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 13/05/2009	**Fecha fin:** 18/05/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Se implementará la lógica de selección y creación de la interfaz *Wap* a la que un determinado usuario deberá acceder para consultar la información de su servicio de horóscopo.	

Tabla 25. Tarea 2 de la historia de usuario Publicar Servicio Horóscopo

Tarea	
Número tarea: 2	**Número historia:** 6
Nombre tarea: Buscar Información de Horóscopo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0,2
Fecha inicio: 18/05/2009	**Fecha fin:** 20/05/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Se buscarán y seleccionarán las informaciones que deberán conformar la página de horóscopo de un determinado usuario dependiendo de su signo zodiacal.	

Tabla 26. Tarea 3 de la historia de usuario Publicar Servicio Horóscopo

Tarea	
Número tarea: 3	**Número historia:** 6
Nombre tarea: Buscar Imágenes de Horóscopo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0,2
Fecha inicio: 20/05/2009	**Fecha fin:** 22/05/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Se buscarán y seleccionarán las imágenes que deberán conformar la página de horóscopo de un determinado usuario dependiendo de su signo zodiacal.	

Tabla 27. Tarea 4 de la historia de usuario Publicar Servicio Horóscopo

Tarea	
Número tarea: 4	**Número historia:** 6
Nombre tarea: Actualizar Contenidos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0,6
Fecha inicio: 22/05/2009	**Fecha fin:** 25/05/2009
Programador responsable: Igor G. Morffi Chechulina	
Descripción: Se implementará la lógica de actualización de los contenidos de las interfaces publicadas según el tiempo establecido por usuario, signo y frecuencia de recibo.	

4.3 Fase de Prueba

Uno de los pilares de XP es el proceso de pruebas (Beck, 1999). XP anima a probar tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones.

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguieron las funcionalidades requeridas diseñadas por el cliente final (Crispin, y otros, 2002).

4.3.1 Pruebas de Aceptación

Las pruebas de aceptación son pruebas de caja negra que se crean a partir de las historias de usuario (Crispin, y otros, 2002). Durante las iteraciones las historias de usuario seleccionadas serán traducidas a pruebas de aceptación. En ellas se especifican, desde la perspectiva del cliente, los escenarios para probar que una historia de usuario ha sido implementada correctamente. Una historia de usuario puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. Las pruebas de aceptación significan la satisfacción del cliente con el producto desarrollado, el final de una iteración y el comienzo de la siguiente (Miller, 2001). El objetivo final de estas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable (Wells, 2006).

Las pruebas de aceptación realizadas al sistema se encuentran en el *Anexo 3.*

4.4 Conclusiones

En este capítulo se describieron las fases de diseño, implementación y prueba. Se realizó una breve descripción del diseño del sistema, el cual se realizó basado en la simplicidad. Se detallaron cada una de las tres iteraciones del proyecto haciendo énfasis en los artefactos generados en la fase de implementación y se abordó acerca de la importancia de la pruebas en la metodología XP, en especial las pruebas de aceptación, ya que éstas miden la satisfacción del cliente con el producto desarrollado.

capítulo 5

CAPÍTULO 5: ESTUDIO DE FACTIBILIDAD

En la planificación del proceso de desarrollo de software es de vital importancia la estimación, la cual consiste en determinar con cierto grado de certeza los recursos necesarios para el desarrollo del mismo, ya sean recursos de hardware, software, esfuerzo, tiempo y costo. En el presente capítulo se realizará un estudio de factibilidad para la realización del sistema propuesto mediante una estimación de tamaño, esfuerzo y planificación necesaria para llevar a cabo el mismo.

5.1 Características del proyecto

Tabla 28. Entradas externas

Nombre de la entrada externa	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media y compleja)
Solicitar Servicio	1	6	Simple
Acceder Servicio de Horóscopo	1	1	Simple
Total		**7**	

Tabla 29. Salidas externas

Nombre de la salida externa	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media y compleja)
Publicación de Subscripción	1	5	Simple
Información de Horóscopo	1	5	Simple
Petición de Confirmación	1	1	Simple
Vista de Recibo de Contenido	1	3	Simple
Total		**14**	

Tabla 30. Ficheros internos

Nombre del fichero interno	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media y compleja)
Colas de mensajes	1	1	Simple
Temporalizadores de órdenes	1	1	Simple
Interacción con *Gateway*	1	1	Simple
Total		**3**	

Tabla 31. Peticiones

Nombre de la petición	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación
Total		**0**	

.

Tabla 32. Interfaces externas

Nombre de la entrada externa	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media y compleja)
SMS	1	1	Simple
Total		**1**	

Tabla 33. Puntos de función desajustados

Elementos	Simple		Medio		Complejo		Subtotal
	No.	Peso	No.	Peso	No.	Peso	
Entradas externas	7	3	0	4	0	6	21
Salidas externas	14	4	0	5	0	7	56
Ficheros internos	3	7	0	10	0	15	21
Peticiones	0	3	0	4	0	6	0
Interfaces externas	1	5	0	7	0	10	5
Total							**103**

5.1.1 Cálculos de instrucciones fuentes, esfuerzo, tiempo de desarrollo, cantidad de hombres y costo

El modelo COCOMO II usa Puntos Función (FP) y/o Líneas de Código Fuente (SLOC) como base para medir tamaño en los modelos de estimación de Diseño Temprano y Post-Arquitectura. Los Puntos de Función procuran cuantificar la funcionalidad de un sistema de software. La meta es obtener un número que caracterice completamente al sistema. Son útiles estimadores ya que están basados en información que está disponible en las etapas tempranas del ciclo de vida del desarrollo de software.

Tabla 34. Características

Características	Valor
Puntos de función desajustados	103
Lenguaje (Java)	53
Instrucciones fuentes por puntos de función	5459
Instrucciones fuentes	5.45 ksloc

Tabla 35. Factores de escala

Nombre	Valor	Justificación
Precedentes	1.24	Existen varios sistemas similares a nivel mundial, pero no a nivel nacional.
Flexibilidad de desarrollo	1.01	Cuenta con una flexibilidad de desarrollo muy alta.
Cohesión del equipo	1.10	El equipo de desarrollo presenta una muy alta cohesión.
Solución de riesgos	4.24	No se identificaron grandes riesgos.
Madurez del proceso	1.56	Se tiene una experiencia en el desarrollo de aplicaciones de este tipo.
Total(SF)	**9.15**	

Tabla 36. Multiplicadores de esfuerzo

Nombre	Valor	Justificación
RCPX	1.05	El sistema presenta un nivel alto de complejidad
RUSE	1.00	El nivel de reutilización del producto es alto.
PDIF	0.96	Uso de memoria y almacenamiento normal, plataforma estable.
PREX	0.90	Alto grado de experiencia en el área de la aplicación, plataforma, lenguaje y herramienta.
PERS	0.52	Alta capacidad del personal.
FCIL	0.82	Se utilizaron entornos de desarrollo integrados, herramientas de modelación y automatización que facilitan el trabajo.
SCED	1.00	Se empleó el tiempo planificado para el desarrollo del sistema.
Total(EM)	**0.89**	

Cálculos:

$A = 2.94 \qquad B = 0.91 \qquad C = 3.67 \qquad D = 0.48$

$E = B + 0.01 * SF = 0.91 + 0.01 * 9.15 = 1.0015$

$PM = A * Size\text{^}E * \prod EM = 2.94 * 5.45\text{^}1.0015 * 0.89 = 13.33$

$F = D + 0.2 * 0.01 * \sum SF = 0.48 + 0.2 * 0.01 * 9.15 \approx 0.50$

$TDEV = C * PM\text{^}F = 3.67 * 13.33\text{^}0.50 = 13.3$

$$CH = \frac{PM}{TDEV} = \frac{13.33}{13.3} \approx 1$$

$C = CH * Sal * PM = 1 * 150 * 13.33 = \$1995$

.

Tabla 37. Resultados

Cálculo de:	Valor
Esfuerzo	13.33 hombres/mes
Tiempo de desarrollo	6.6 meses
Cantidad de hombres	1 hombres
Salario medio	$150
Costo	$1999,5

5.2 Beneficios tangibles e intangibles

El Sistema de Servicio de Subscripción de Horóscopo es un producto con fines comerciales, puede ser añadido a la plataforma de Servicios de Cubacel como servicio adicional, aumentando su gama de productos. El beneficio fundamental está dado por la posibilidad de poder contar con un sistema que brinde a los clientes la prestación de recibir su horóscopo según la frecuencia que estos decidan, y con ello aumentar los ingresos de Cubacel por este concepto.

Como beneficio intangible aportado por el sistema desarrollado se puede mencionar que servirá de base para otros servicios de valor agregado que serán desarrollados por la entidad antes mencionada. Vale destacar que haciendo uso de la presente solución, pueden brindarse otros servicios similares, como recibir diariamente, por ejemplo, la primera noticia de un determinado periódico.

5.3 Análisis de costo

El desarrollo de un producto siempre presenta un costo de producción, el cual es justificado en base a los beneficios reportados por el mismo. Los productos informáticos no están exentos de ello. El sistema presentado en este trabajo no precisa grandes gastos, debido en gran medida a la utilización de plataformas, tecnologías y herramientas libres que no requieren del pago de licencia por su uso o comercialización, su costo está fundamentalmente determinado por el salario devengado por los desarrolladores que toman parte en su desarrollo.

5.4 Conclusiones

En este capítulo se realizó un análisis de factibilidad de la solución propuesta, obteniendo los estimados de tamaño, esfuerzo y planificación. Se arribó a la conclusión de que el desarrollo del sistema propuesto es viable, basándonos en la comparación de los costos de producción con los beneficios que reportará el sistema al incorporarse a la plataforma de Servicios de Cubacel.

CONCLUSIONES GENERALES

Con el desarrollo de este trabajo se profundizó en el conocimiento de la mensajería móvil y los servicios de valor agregado. Se aplicó la metodología XP para guiar el proceso de desarrollo de software, mediante el cual se obtuvo la implementación de un servicio de horóscopos para usuarios de teléfonos móviles con las funcionalidades previstas para el primer ciclo de desarrollo del software. El diseño y la implementación se rigieron por los estándares de la plataforma Java EE, aplicando diferentes patrones de diseño. Se definieron los parámetros de configuración e instalación con la documentación correspondiente. Se puede concluir que se ha cumplido satisfactoriamente el objetivo trazado para este trabajo, enfatizando en los siguientes puntos:

- Fue implementada una aplicación de tipo servidora que maneja el flujo de actividades relacionado con la solicitud por parte de los usuarios de recibir el servicio de horóscopo, permitiendo a éstos personalizar el mismo y encargándose además de la publicación de dicho contenido.

- Se desarrolló una interfaz *Wap* amigable que permite consultar el servicio de horóscopos por parte de los usuarios del sistema.

- Se implementó una interfaz Web que posibilita llevar a cabo el proceso de solicitud de servicio a los usuarios.

RECOMENDACIONES

A continuación son listadas las recomendaciones que en opinión del autor deben ser tenidas en cuenta con el objetivo de realizar un seguimiento de este trabajo:

- Integrar el sistema a la Plataforma de Servicios de Cubacel.
- Implementar un módulo que brinde interfaces de gestión, administración y auditoría con el objetivo de poder controlar el servicio de manera independiente.
- Utilizar este sistema como base para la implementación de otros servicios de valor agregado en la plataforma de servicios de Cubacel.
- Realizar pruebas de este sistema en condiciones reales.

REFERENCIAS BIBLIOGRÁFICAS

Alexander, Christopher, Ishikawa, S. y Silverstein, Murray. 1997. *A Pattern Language: Towns, Buildings,Construction.* Oxford, England : Oxford University Press, 1997.

Beck, Kent. 1999. *Extreme Programming Explained.* s.l. : Addison-Wesley, 1999. 0201616416.

Beck, Kent y Cunningham, Ward. 1989. *A Laboratory For Teaching Object-Oriented Thinking.* New Orleans : SIGPLAN Notices, 1989.

Begin, Clinton, Goodin, Brandon and Meadors, Larry. 2007. *iBatis in Action.* s.l. : Manning Publications, 2007.

Brittain, Jason y Darwin, Ian F. 2003. *Tomcat: The Definitive Guide.* s.l. : O'Reilly Books, 2003. 9780596003180.

Codd, E. F. 1990. *The relational model for database management: version 2.* Boston : Addison-Wesley, 1990. 0-201-14192-2.

Crispin, Lisa y House, Tip. 2002. *Testing Extreme Programming.* s.l. : Addison-Wesley, 2002. 0321113551.

Dubin, Ben. 1997. Real World Applications Using JavaTM RMI. [En línea] 16 de Junio de 1997. [Citado el: 12 de Mayo de 2009.] http://www.ecst.csuchico.edu/~amk/foo/csci611/notes/rmiexamples.html.

Ericsson. 2006. WAP-Push. *Ericsson.* [En línea] 2006. [Citado el: 4 de Marzo de 2009.] http://www.ericsson.com/mobilityworld/sub/open/technologies/open_development_tips/tools/wap_push_sdk.

Fowler, Martin. 2004. Is Design Dead. *martinfowler.com.* [En línea] Mayo de 2004. [Citado el: 18 de Abril de 2009.] http://martinfowler.com/articles/designDead.html.

—. 1999. *Planning Extreme Programming.* s.l. : Addison-Wesley, 1999. 0201710919.

Gamma, y otros. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Reading, MA: Addison-Wesley, 1995.

Garcia de León, Javier, Rodríguez, José Ignacio y Imaz, Aitor. 1999. *Aprenda servlets de Java como*

si estuviera en primero. San Sebastián : TECNUN, 1999.

Guemes, Celetino. 2006. Wurfl-Wall. *FRAMEWORK AME.* [En línea] 6 de Abril de 2006. [Citado el: 18 de Marzo de 2009.] http://ame.endesa.es/confluence/display/AMEBASE/wurlf-wall.

IBM. 2007. Eclipse Platform Technical Overview. *Eclipse.* [En línea] 2007. [Citado el: 12 de Abril de 2009.] http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.pdf.

IEC. 2007. International Engineering Consortium. *Global System for Mobile Communication (GSM). IEC.* [En línea] 2007. [Citado el: 26 de Febrero de 2009.] http://www.iec.org/online/tutorials/gsm/index.html..

—. 2007. International Engineering Consortium. *Wireless Internet Network Communications Architecture. IEC.* [En línea] 2007. [Citado el: 26 de Febrero de 2009.] http://www.iec.org/online/tutorials/winternet/topic04.html.

Interacel. 2009. Interacel. *Horoscopo Interacel.* [En línea] 2009. [Citado el: 19 de Febrero de 2009.] http://www.interacel.com/?page=article&c=GT&a=8&i=45_i_zodiaco_guatemala.png&s=1011.

InternetWebServeis. 2008. MyMobileWeb. *MyMobileWeb | IWS - Internet Web Serveis.* [En línea] 2008. [Citado el: 16 de Marzo de 2009.] http://www.iws.es/es/iwsinnovacion/mymobileweb.

Java Boutique, (TM). 2004. SAMS: Java's API For Mobile Services. *Java(TM) Boutique.* [En línea] 22 de Enero de 2004. [Citado el: 10 de Mayo de 2009.] http://javaboutique.internet.com/tutorials/SAMS/index-2.html.

Jendrock, Eric, y otros. 2006. *The Java™ EE 5.* Los Angeles : Addison-Wesley, 2006. 0–321–49029–0.

Martin, Robert C. 2001. *Extreme Programming in Practice.* Los Angeles : Addison-Wesley, 2001. 0201709376.

Miller, Roy. 2001. *Extreme Programming Applied.* s.l. : Addison-Wesley, 2001. 0201616408.

Oberg, Rickard. 2001. *Mastering RMI: Developing Enterprise Applications in Java and EJB.* s.l. : John Wiley & Sons, 2001. 0471389404.

Passani, Luca. 2002. WALL - The Wireless Abstraction Library. *Wireless Universal Resource File.* [En línea] 2002. [Citado el: 12 de Marzo de 2009.] http://wurfl.sourceforge.net/index.php.

Pilato, C. Michael, Collins-Sussman, Ben y Fitzpatrick, Brian W. 2004. *Version Control with Subversion.* s.l. : O'Reilly Media, Inc, 2004. ISBN 0596004486.

PostgreSQL. 2008. PostgreSQL. *About.* [En línea] 2008. [Citado el: 25 de Marzo de 2009.] http://www.postgresql.org/about/.

Raible, Matt. 2008. *The Spring Primer.* S.L : SourceBeat, 2008. 0974884375.

Routt, William. 2003. *Wireless Markup Language (WML) Scripting and Programming using WML, cHTML, and xHTML.* s.l. : Althos, 2003. 0974278750.

TechTarget. 2005. Multimedia Message Service. *Search Tech Target.* [En línea] 2 de Mayo de 2005. [Citado el: 21 de Febrero de 2009.] http://searchmobilecomputing.techtarget.com/sDefinition/0,290660,sid40_gci943702,00.html.

—. 2005. Short Message. *Search Tech Target.* [En línea] 2 de Mayo de 2005. [Citado el: 15 de Febrero de 2009.] http://searchmobilecomputing.techtarget.com/sDefinition/0,290660,sid40_gci773769,00.html.

—. 2005. SMIL. *Search Tech Target.* [En línea] 2 de Mayo de 2005. [Citado el: 1 de Marzo de 2009.] http://searchwebservices.techtarget.com/sDefinition/0,290660,sid26_gci214217,00.html.

Telefónica I+D. 2005. *Las Telecomunicaciones y la Movilidad en la Sociedad de la Información.* Madrid : AHCIET, 2005. 84-89900-37-X..

Telefónica, Movistar. 2009. Telefónica Movistar. *E-mocion Movistar.* [En línea] 2009. [Citado el: 23 de Febrero de 2009.] http://www.movistar.es/accesible/movistar-emocion.html.

T-Mobile. 2009. T-Mobile. *Horoscope.* [En línea] 2009. [Citado el: 23 de Febrero de 2009.] http://www.t-zones.co.uk/en/More/40_horoscopes/text_horoscopes.html.

Wake, William C. 1999. *Extreme Programming Explored.* s.l. : Addison-Wesley, 1999. 0201733978.

Walls, C y Breidenbach, R. 2005. *Spring in Action.* s.l. : Manning Publications, 2005.

Wells, Don. 2006. Acceptance Tests. *Extreme Programming.* [En línea] 2006. [Citado el: 24 de Abril de 2009.]

Wikipedia. 2007. Control de versiones. *Wikipedia enciclopedia libre.* [Online] 2007. [Cited: Abril 11,

2009.] http://es.wikipedia.org/wiki/Control_de_versiones.

—. 2008. Entorno de desarrollo integrado. *Wikipedia enciclopedia libre.* [En línea] 2008. [Citado el: 19 de Marzo de 2009.] http://es.wikipedia.org/wiki/Ambiente_integrado_de_desarrollo.

—. 2009. Framework. *Wikipedia enciclopedia libre.* [En línea] 2009. [Citado el: 6 de Marzo de 2009.] http://es.wikipedia.org/wiki/Framework.

—. 2009. Interfaz de programación de aplicaciones. *Wikipedia enciclopedia libre.* [En línea] 2009. [Citado el: 14 de Marzo de 2009.] http://es.wikipedia.org/wiki/Application_Programming_Interface.

—. 2009. Plain Old Java Object. *Wikipedia enciclopedia libre.* [En línea] 2009. [Citado el: 9 de Marzo de 2009.] http://es.wikipedia.org/wiki/Plain_Old_Java_Object.

—. 2008. Plataforma de Desarrollo. *Wikipedia enciclopedia libre.* [En línea] 2008. [Citado el: 3 de Marzo de 2009.] http://es.wikipedia.org/wiki/Plataforma_de_desarrollo.

GLOSARIO DE TÉRMINOS

API: (*Application Programming Interface*) Interfaz de Programación de Aplicaciones.

CHTML: (*Compact HTML*) HTML compacto. Es un subconjunto de HTML para dispositivos pequeños de información.

Framework: estructura de soporte definida, en la cual otros proyectos de software pueden ser organizados y desarrollados.

GPRS: (*General Packet Radio Service*) Servicio General de Paquetes de Radio. Es un estándar de comunicación para teléfonos móviles que transmite la información por grupos significativos o paquetes.

GSM: (*Global System for Mobile Communications*) Sistema Global para comunicaciones Móviles. Sistema de telefonía celular digital para comunicaciones móviles de segunda generación.

HTML: (*Hyper Text Markup Language*) Lenguaje de Etiquetas de Hipertexto. Es un lenguaje de etiquetas diseñado para estructurar textos y presentarlos en forma de hipertexto, es el formato estándar de las páginas web.

IDE: (*Integrated Development Environment*) Entorno Integrado de Desarrollo.

iMode: Conjunto de tecnologías y protocolos diseñados para poder navegar a través de mini páginas diseñadas específicamente para dispositivos móviles.

JAVA: Lenguaje de programación multiplataforma desarrollado por Sun Microsystem.

JDBC: (*Java Database Connectivity*). Es un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

JMS: (*Java Message Service*) Servicios de Mensajería de Java. Es un estándar de mensajería que permite a los componentes de aplicaciones basados en la plataforma de Java EE crear, enviar, recibir y leer mensajes.

JSTL: *(Java Server Pages Standard Tag Libraries)* proporciona librerías de etiquetas con utilidades para el desarrollo de páginas web dinámicas.

MMS: (*Multimedia Messaging System*) Servicio de Mensajería Multimedia. Es un estándar de mensajería

que le permite a los teléfonos móviles enviar y recibir contenidos multimedia.

MMSC: (*Multimedia Messaging System Center*) Centro de Mensajería Multimedia. Es un elemento de la red de telefonía móvil cuya función es la de enviar y recibir mensajes MMS.

MVC: (*Model – View – Controller)* Modelo Vista Controlador, estilo o patrón arquitectónico muy difundido.

MVCC: (*MultiVersion Concurrency Control)* es una tecnología de los sistemas generadores de base de datos que posibilita el acceso de varios clientes a la misma información en un mismo instante de tiempo.

Plugin: Es un programa adicional que puede ser añadido a una aplicación para aumentar la funcionalidad de esta.

PostgreSQL: Sistema gestor de base de datos

PPG: (*Push Proxy Gateway*). Elemento encargado de soportar la funcionalidad Push de contenidos hacia los terminales móviles.

Portlets: Son componentes de interfaz de usuario que son gestionadas y visualizadas en un portal web.

RMI: (*Java Remote Method Invocation*). Es un mecanismo ofrecido en Java para invocar un método remotamente.

SLOC: (*Source lines of Code*). Líneas de Código Fuente.

SMS: (*Short Message Service*) Servicio de Mensajes Cortos. Es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos.

SMSC: (*Short Message Service Center*) Centro de Servicios de Mensajes Cortos. Es un elemento de la red de telefonía móvil cuya función es la de enviar y recibir mensajes SMS.

SVN: Sistema de Control de Versiones.

URL: (*Uniform Resource Locator*) Localizador Uniforme de Recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

VAS: (*Value Added Services*) Servicios de Valor Agregado.

VASP: (*Value Added Services Provider*) Proveedor de Servicios de Valor Agregado.

WALL: (*Wireless Abstraction Library*). Es la combinación de una API Java, y una serie de librerías de tags (taglibs JSP) que permiten construir páginas JSP portables entre dispositivos.

WAP: (*Wireless Application Protocol*) Protocolo de Aplicaciones Inalámbricas.

WURFL: (*Wireless Universal Resource File*). Es un fichero XML donde se expresan, de manera compacta, las características de diversos dispositivos móviles.

XHTML: (*eXtensible Hypertext Markup Language*) Lenguaje Extensible de Marcado de Hipertexto. XHTML es la versión XML de HTML.

XML: (*eXtensible Markup Language*) Lenguaje de Marcas Extensible. Es un metalenguaje extensible de etiquetas.

XP: (*Extreme Programming*) Programación Extrema. Es una metodología ágil de desarrollo de software.

anexos

ANEXOS

Anexo 1. Relación de Compañías que utilizan Java RMI como tecnología de comunicación en aplicaciones distribuidas (Dubin, 1997)**.**

Caso 1: *International Business Machines Corporation* (IBM)

Perfil de la compañía

IBM, la mayor compañía de software del mundo, crea, desarrolla y manufactura productos y servicios avanzados de tecnología de la información, incluyendo ordenadores, software, sistemas de red, dispositivos de almacenamiento y microelectrónica. IBM ha sido un líder en el desarrollo de Internet desde el principio de esta tecnología y está dedicada a ayudar a fabricantes y desarrolladores explotando el potencial de Java. Con los recursos de mas de 20 laboratorios de desarrollo en el mundo, IBM rápidamente monta sus propios productos -- incluyendo sistemas operativos -- así como desarrollando herramientas y tecnologías de Internet para soportar Java.

(Desarrolladores y fabricantes pueden encontrar más información acerca de los esfuerzos de IBM en Java en: http://www.ibm.com)

Aplicación

El tiempo de desarrollo en aplicaciones de negocios es actualmente lo más importante para las empresas actualmente. La necesidad es responder rápidamente a los cambios en los negocios, agregar componentes y desarrollar nuevas aplicaciones. Las compañías y sus departamentos de desarrollo necesitan una manera de acelerar este proceso para aumentar su productividad y ser más competitivos.

El Proyecto San Francisco de IBM muestra las necesidades de una pequeña o mediana empresa y provee soluciones independientes para un más rápido desarrollo de aplicaciones. Esta iniciativa de IBM, proveerá una serie de aplicaciones de alto nivel escritas en Java. Primero, la capa Base provee un conjunto unificado de servicios para construir con objetos distribuidos Java. Segundo, la capa *Common Business Object* (Objetos de Negocios Comunes) provee un conjunto de objetos de negocios comunes a una gran variedad de aplicaciones de negocios. Tercero, el marco de aplicación provee funciones específicas del dominio del negocio. El Proyecto San Francisco permite a las sociedades enfocarse menos en la tecnología y más en las soluciones únicas, para desarrollar mas fácilmente aplicaciones de

negocios que se adapten a los cambios y puedan interoperar con otras aplicaciones de otros desarrollos.

El Proyecto San Francisco Project usa *Remote Method Invocation* (RMI) de Java como la base de la infraestructura distribuida. Hemos elegido RMI porque es muy cercana a los requerimientos distribuidos de este proyecto. La distribución trae la necesidad de gestionar y conectar todos los servidores que realizaran el trabajo distribuido. Se han realizado extensiones a RMI las cuales hacen fácil gestionar y usar múltiples direcciones de servidores donde múltiples objetos distribuidos residen.

Estas extensiones proveen la forma de conectarse de los clientes con los objetos distribuidos y de arrancar y parar los servidores así como otras funciones básicas de gestión de servidores.

Ejemplos de áreas de las extensiones son:

- Manejo de excepciones
- Gestión de transacciones distribuidas
- Gestión de los procesos de servidor

Caso 2: *Swiss Federal Supreme Court (Corte Federal Suprema suiza)*

Perfil de la compañía

En la práctica la Corte Federal Suprema decide sobre 5500 casos al año. Todas las decisiones se almacenan en el sistema de bases de datos textual *BASISplus*. Los 30 jueces tienen acceso a esta base de datos textual.

De las 250 personas que trabajan en la Corte Federal Suprema, 15 trabajan en el departamento de computación. El equipo de desarrollo de software trabaja para desarrollar las aplicaciones necesarias para los juristas, en particular aplicaciones para gestionar las decisiones que están tratando.

En el futuro estamos pensando en cambiar a una plataforma servidor. Pero necesitamos escribir aplicaciones lo mas independientes posible del hardware. Esta es una de las razones para usar el lenguaje de programación Java.

Aplicación

Marco para acceder al sistema de bases de datos textual *BASISplus* desde aplicaciones Java.

Con el crecimiento de Internet, vino la idea de publicar una parte de nuestras decisiones a través de la

red. Como la seguridad es lo que más nos preocupaba, hicimos un pequeño análisis de las tecnologías existentes en ese momento para acceder a bases de datos a través de clientes WWW. El proyecto arranca y usaba la arquitectura CGI, luego echamos un vistazo a la tecnología Java. Entonces descubrimos la API RMI y decidimos evaluar si esta tecnología nos permitía acceder a nuestra base de datos *BASISplus* desde cualquier cliente.

Usando la API JNI, creamos un driver para acceder a la base de datos *BASISplus* desde el lenguaje Java. Nosotros llamamos a esta clase *OpenAPI* la cual es un "pegamento" que permite acceder al DBMS localmente en el servidor. Entonces creamos, con la API RMI una fábrica de objetos en el lado del servidor. Un cliente, el cual necesita acceder a la base de datos se conecta a la fábrica de objetos, la cual crea un objeto del servidor (cliente del DBMS). Una vez creado, el objeto del servidor devuelve su referencia a la fábrica de objetos, y esta lo envía al cliente. El cliente tiene ahora capacidad para dialogar directamente con la base de datos a través de un objeto dedicado del servidor. Como *BASISplus* no tiene seguridad multi-hilo, cada objeto del servidor se ejecuta en su propio subproceso.

Principales ventajas:

- Cada cliente debe autentificarse en la base de datos. Con otras soluciones, la autentificación se hace normalmente a través de UID & PASSWORD que están almacenados en el servidor HTTP.
- Las transacciones no se restringen a un solo documento HTML.
- Verdaderas aplicaciones cliente/servidor que pueden ponerse en Internet.

Hemos creado un pequeño prototipo de una parte de la aplicación. Estamos usando el lenguaje Ada83. Conclusión: Java es fácil, potente e independiente del hardware.

Caso 3: Avitek

Perfil de la compañía

Avitek es una compañía de rápido crecimiento centrada en la creación de soluciones Java de Boulder, Colorado. Fundamos la compañía en 1995 y éramos una de las compañías pioneras del mundo en desarrollo real en Java. Desde el principio, el 85% de rentas son resultado del trabajo de desarrollo en Java y esto ha continuado hasta ser nuestro núcleo y área de competencia. Actualmente, Avitek consta de cuatro arquitectos sénior cada uno capaz de llevar cualquier solución avanzada basada en Java, así como varios especialistas Java entre el personal.

Nuestro rango de soluciones Java van desde Internet, applets, y sistemas de punto de venta al servicio de intranet y sistemas reservados escritos completamente en Java. Proyectos actuales incluyen un sistema de ventas on-line con una configuración muy compleja del producto y un sistema de entrenamiento y aprendizaje para soportar el funcionamiento corporativo y ayuda en el trabajo sobre la intranet.

La primera aclamación de la crítica para Avitek fueron dos premios para nuestro primer applet llamado *WebContact*. Gamalan y JARS lo colocaron como uno de los mejores applets en la red. Como consecuencia, nosotros hemos desarrollado varios productos a medida y soluciones en Java. Nuestra mayor fuerza esta en nuestra habilidad en hacer correctamente aplicaciones y reunir requerimientos específicos técnicos y de negocios. Nosotros tenemos una profunda experiencia que incluye:

1. Java *front ends* para Bases de Datos en el lado del servidor,

2. Soluciones cliente y servidor en JDBC implementadas con Oracle, Sybase, SQL Server, o Access.

3. LDAP basados en preguntas y aplicaciones de administración,

4. Aplicaciones basadas en RMI que soportan integración con Bases de Datos propietarias,

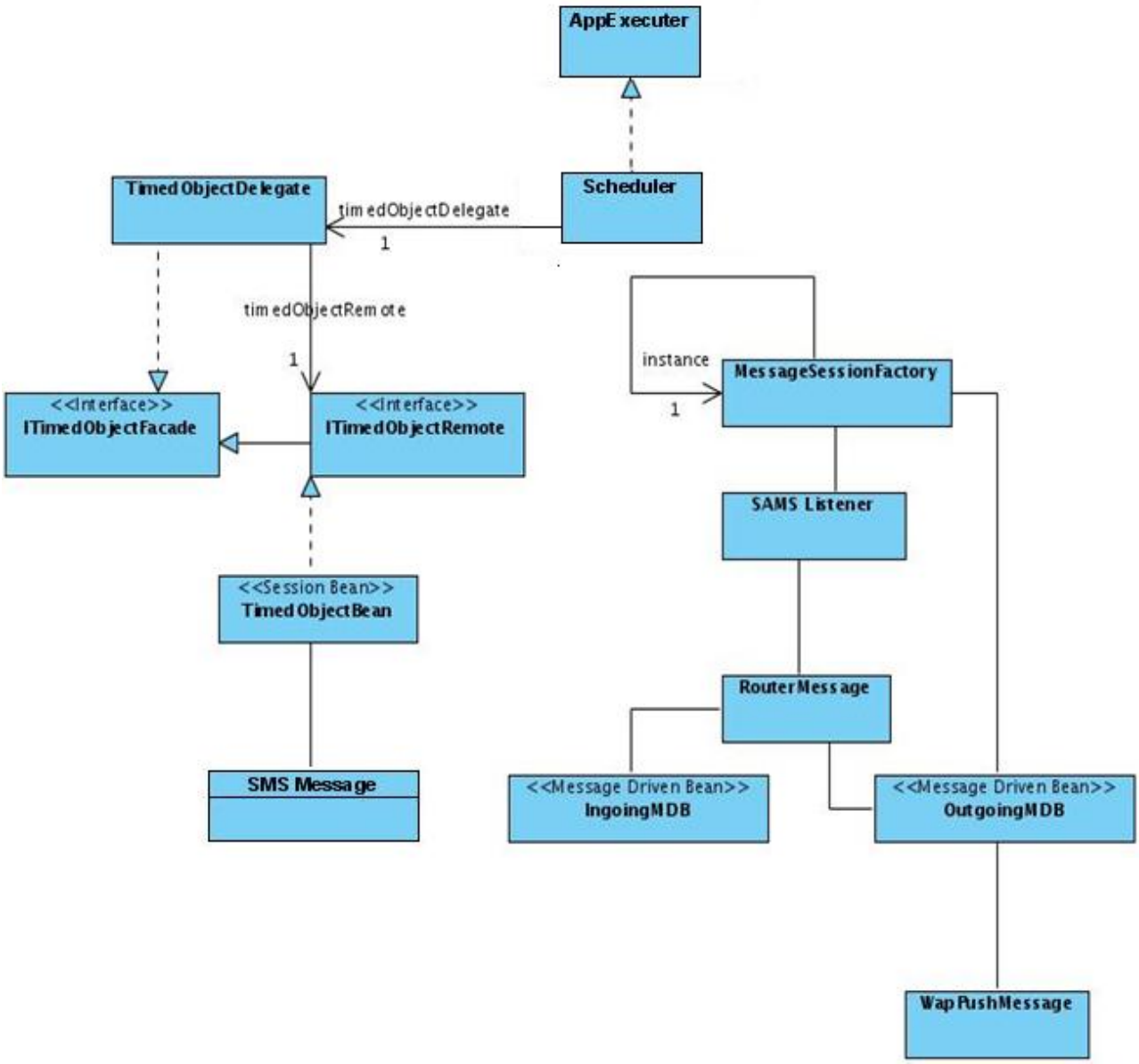
5. Servidores Java usando Java Servlet API y el servidor Netscape Java API, y

6. La organización Avitek desarrolló y construyó todos nuestros applets.

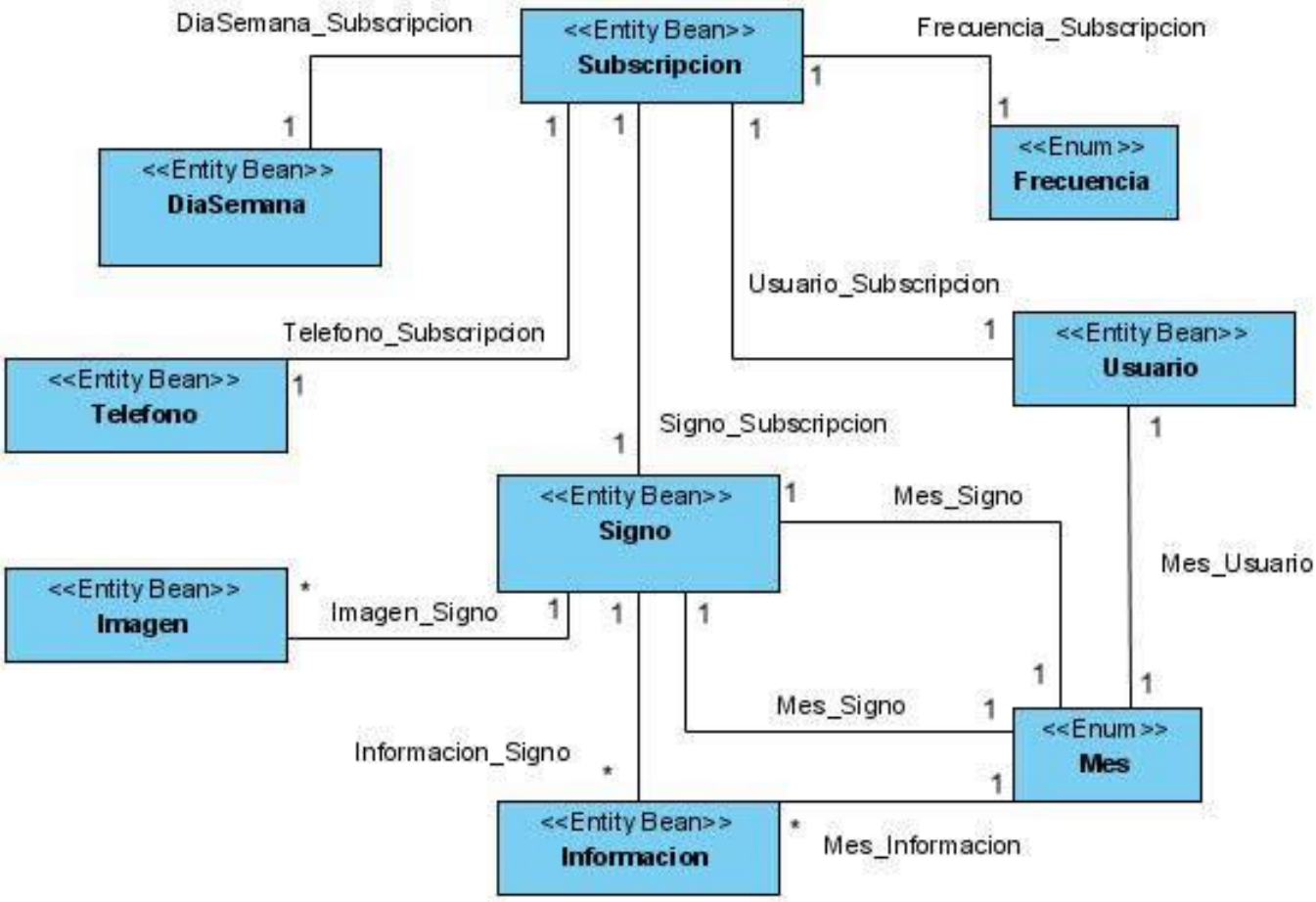
Aplicación

Nuestro proyecto RMI fue diseñado para construir unos estándares robustos y extensibles basados en aplicaciones software que nuestro cliente pudiera actualizar fácilmente, modificar, y ampliar en una intranet funcionando bajo Windows 95 o NT. La aplicación está siendo desarrollada como una reescritura de un completo sistema reservado de contabilidad con 10 personas/año en su desarrollo inicial. El objetivo de este proyecto era desarrollar y soportar la creación de una prueba completa del concepto de *front end* Java y un sistema de Bases de Datos. Esto fue logrado con Java en el cliente y en el servidor usando RMI y una API personalizada implementada en una DLL.

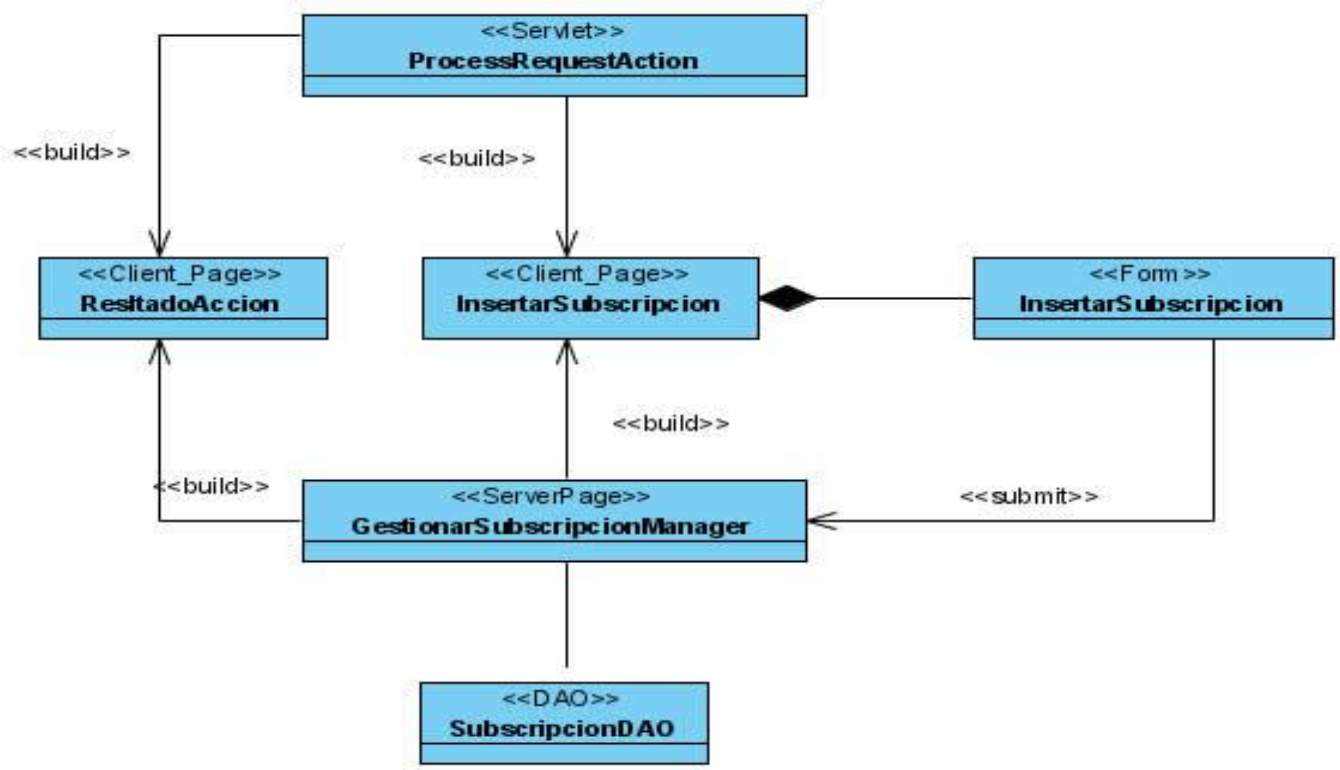
Anexo 2. Diagramas de Clase del Sistema



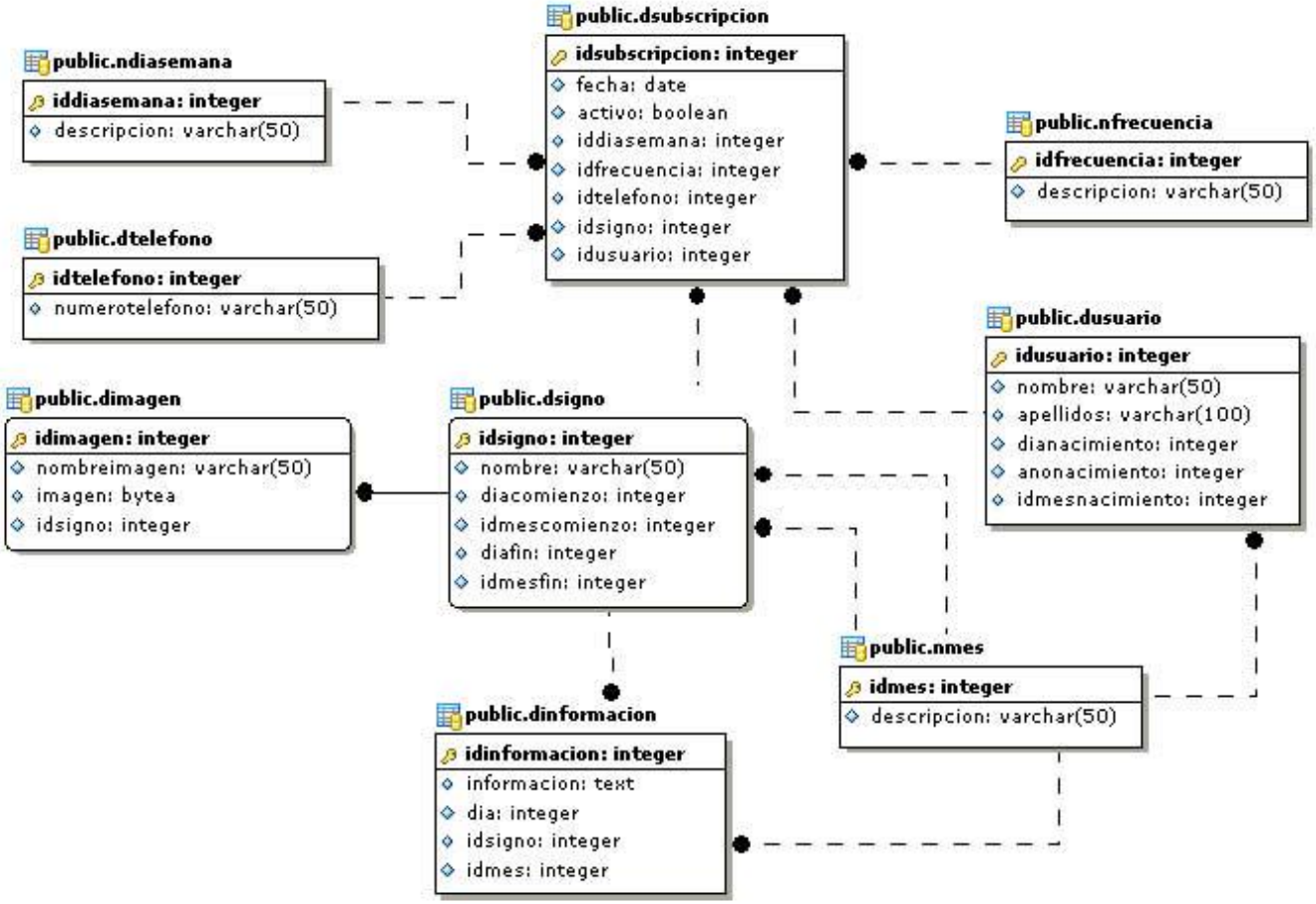
Anexo 2. 1 Diagrama de clases *Messaging*



Anexo 2. 2 Diagrama de Clases Persistentes



Anexo 2. 3 Diagrama de Clases Web



Anexo 2.4 Diagrama de Entidades Persistentes

Anexo 3. Pruebas de Aceptación

Pruebas de aceptación para la historia de usuario **Gestionar Solicitud de Servicio**

Anexo 3. 1Gestionar solicitud de servicio

Caso de Prueba de Aceptación	
Código: HU1_P1	**Historia de Usuario:** 1 – Gestionar solicitud de servicio.
Nombre: Gestionar solicitud de servicio.	
Descripción: Prueba para verificar la funcionalidad de recibir la solicitud de servicio.	
Condiciones de Ejecución: El servicio debe estar iniciado y ejecutándose. La sesión de SAMS debe estar creada. Se introducirá una solicitud de subscripción.	
Entrada / Pasos de ejecución: Se intenta recibir la solicitud insertada con los datos válidos.	
Resultado Esperado: La solicitud es insertada sin errores.	
Evaluación de la Prueba: Prueba satisfactoria	

Anexo 3. 2 Generar pre-orden

Caso de Prueba de Aceptación	
Código: HU1_P2	**Historia de Usuario:** 1 – Gestionar solicitud de servicio.
Nombre: Generar pre-orden.	
Descripción: Prueba para verificar la funcionalidad de generar la pre-orden.	
Condiciones de Ejecución: El servicio debe estar iniciado y ejecutándose. Se utilizará una solicitud de subscripción con datos válidos.	
Entrada / Pasos de ejecución: Se intenta generar una pre-orden con los datos recibidos.	
Resultado Esperado: La pre-orden es generada sin errores.	
Evaluación de la Prueba: Prueba satisfactoria	

Anexo 3. 3 Enviar Mensaje

Caso de Prueba de Aceptación	
Código: HU1_P3	**Historia de Usuario:** 1 – Gestionar solicitud de servicio.
Nombre: Enviar SMS.	
Descripción: Prueba para verificar la funcionalidad de enviar un *mensaje* de verificación para la activación del servicio.	
Condiciones de Ejecución: El servicio debe estar iniciado y ejecutándose. La sesión de SAMS debe estar creada. Se utilizará un mensaje con datos válidos.	
Entrada / Pasos de ejecución: Se intenta enviar un mensaje *de* verificación a un número válido.	
Resultado Esperado: El mensaje es enviado y recibido sin errores.	
Evaluación de la Prueba: Prueba satisfactoria	

Pruebas de aceptación para la historia de usuario **Gestionar Orden**

Anexo 3. 4 Chequear órdenes completadas

Caso de Prueba de Aceptación	
Código: HU2_P1	**Historia de Usuario:** 2 – Gestionar orden.
Nombre: Chequear ordenes completadas.	
Descripción: Prueba para verificar la funcionalidad de chequear las órdenes que han sido completadas.	
Condiciones de Ejecución: El servicio debe estar iniciado y ejecutándose. Se establece cada que tiempo se efectuará el chequeo. Se completarán órdenes.	
Entrada / Pasos de ejecución: Se intenta encontrar las órdenes que han sido completadas.	
Resultado Esperado: Se encuentra las órdenes completadas.	
Evaluación de la Prueba: Prueba satisfactoria	

Anexo 3. 5 Conformar SMS

Caso de Prueba de Aceptación	
Código: HU2_P2	**Historia de Usuario:** 2 – Gestionar orden.
Nombre: Conformar SMS.	
Descripción: Prueba para verificar la funcionalidad de conformar SMS con la URL de publicación.	
Condiciones de Ejecución: El servicio debe estar iniciado y ejecutándose. Se utilizarán datos de subscripción válidos.	
Entrada / Pasos de ejecución: Se intenta confeccionar un mensaje SMS.	
Resultado Esperado: Se genere correctamente el mensaje SMS.	
Evaluación de la Prueba: Prueba satisfactoria	

Anexo 3. 6 Enviar SMS

Caso de Prueba de Aceptación	
Código: HU2_P3	**Historia de Usuario:** 2 – Gestionar orden.
Nombre: Enviar SMS.	
Descripción: Prueba para verificar la funcionalidad de enviar el mensaje SMS.	
Condiciones de Ejecución: El servicio debe estar iniciado y ejecutándose. La sesión de SAMS debe estar creada. Se utilizará un mensaje SMS válido.	
Entrada / Pasos de ejecución: Se intenta enviar un mensaje SMS.	
Resultado Esperado: Se envíe correctamente el mensaje SMS.	
Evaluación de la Prueba: Prueba satisfactoria	

Pruebas de aceptación para la historia de usuario **Chequear existencia de pre-orden**

Anexo 3. 7 Chequear existencia de pre-orden

Caso de Prueba de Aceptación	
Código: HU3_P1	**Historia de Usuario:** 3 – Chequear existencia de pre-orden.
Nombre: Chequear existencia de pre-orden.	
Descripción: Prueba para verificar la funcionalidad de chequear que el código introducido por un usuario corresponde a una pre-orden válida.	
Condiciones de Ejecución: El servicio debe estar iniciado y ejecutándose. Se utilizará un código de pre-orden válido.	
Entrada / Pasos de ejecución: Se intenta encontrar la pre-orden solicitada.	
Resultado Esperado: Se encuentre la pre-orden solicitada sin errores	
Evaluación de la Prueba: Prueba satisfactoria	

Pruebas de aceptación para la historia de usuario **Gestionar servicio**

Anexo 3. 8 Continuar con el servicio

Caso de Prueba de Aceptación	
Código: HU4_P1	**Historia de Usuario:** 4 – Gestionar servicio.
Nombre: Continuar con el servicio	
Descripción: Prueba para verificar la funcionalidad de chequear cuando el usuario escoja continuar con el servicio.	
Condiciones de Ejecución: El servicio debe estar iniciado y ejecutándose. Se utilizará un código de pre-orden válido.	
Entrada / Pasos de ejecución: Se intenta cambiar el estado de la pre-orden	
Resultado Esperado: Se cambie satisfactoriamente el estado de la pre-orden.	
Evaluación de la Prueba: Prueba satisfactoria	

Anexo 3. 9 Eliminar pre-orden

Caso de Prueba de Aceptación	
Código: HU4_P2	**Historia de Usuario:** 4 – Gestionar servicio.
Nombre: Eliminar pre-orden.	
Descripción: Prueba para verificar la funcionalidad de eliminar una pre-orden por parte del usuario.	
Condiciones de Ejecución: El servicio debe estar iniciado y ejecutándose. Se utilizará un código de pre-orden válido.	
Entrada / Pasos de ejecución: Se intenta eliminar pre-orden por parte del usuario.	
Resultado Esperado: Se elimine correctamente la pre-orden.	
Evaluación de la Prueba: Prueba satisfactoria	

Pruebas de aceptación para la historia de usuario **Crear orden**

Anexo 3. 10 Crear Orden

Caso de Prueba de Aceptación	
Código: HU5_P1	**Historia de Usuario:** 5 – Crear orden.
Nombre: Crear orden.	
Descripción: Prueba para verificar la funcionalidad de confeccionar la orden con los datos introducidos por el usuario.	
Condiciones de Ejecución: El servicio debe estar iniciado y ejecutándose. Se utilizará datos válidos.	
Entrada / Pasos de ejecución: Se intenta confeccionar la orden con los datos introducidos.	
Resultado Esperado: Se confeccione la orden sin errores.	
Evaluación de la Prueba: Prueba satisfactoria	

Pruebas de aceptación para la historia de usuario **Publicar Servicio Horóscopo**

Anexo 3. 11 Cargar Datos a Publicar

Caso de Prueba de Aceptación	
Código: HU6_P1	**Historia de Usuario:** 6 – Publicar Servicio Horóscopo.
Nombre: Cargar Datos.	
Descripción: Prueba para verificar la funcionalidad de cargar los datos que serán mostrados al usuario.	
Condiciones de Ejecución: El servicio debe estar iniciado y ejecutándose. Se utilizará un número de teléfono subscrito.	
Entrada / Pasos de ejecución: Se intenta publicar la información con los datos obtenidos.	
Resultado Esperado: Se carguen los datos sin errores.	
Evaluación de la Prueba: Prueba satisfactoria	

Anexo 3. 12 Mostrar Información Publicada.

Caso de Prueba de Aceptación	
Código: HU6_P2	**Historia de Usuario:** 6 – Publicar Servicio Horóscopo.
Nombre: Mostrar Información Publicada.	
Descripción: Prueba para verificar la funcionalidad de mostrar al usuario su horóscopo.	
Condiciones de Ejecución: El servicio debe estar iniciado y ejecutándose. Se utilizará un número de teléfono subscrito.	
Entrada / Pasos de ejecución: Se intenta acceder a la información publicada en una URL.	
Resultado Esperado: Se muestren los datos sin errores.	
Evaluación de la Prueba: Prueba satisfactoria	