Universidad de las Ciencias Informáticas

Facultad 2

Interfaz de usuario para la administración de los servicios para la plataforma de telefonía móvil, COMCEL

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autora: Neimary Pedraza Martínez

Tutor: Ing. Oigres Alvarez Pérez

Co-tutor: Ing. Elieser Bello Ross

Ciudad de La Habana, Cuba

"Año 52 de la Revolución"

Junio, 2010

"O nosotros somos capaces de destruir con argumentos las ideas contrarias, o debemos dejar que se expresen. No es posible destruir ideas por la fuerza, porque esto bloquea cualquier desarrollo libre de la inteligencia."

Che Guevara

Declaración de Autoría

Declaro que soy la única autora de este trabajo y autorizo a la Facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2010.

Autor:	Tutor:	Cotutor:
Neimary Pedraza Martínez	Ing. Oigres Alvarez Pérez	Ing. Elieser Bello Ross
_____	_____	_____

Les dedico este trabajo a mis padres por todo su amor, sin ellos no estaría aquí hoy.

A mis hermanitos pues son mis dos soles.

A Elítico por ser junto a mi familia el mejor regalo que me ha dado la vida.

Agradecimientos

Agradecer a todas las personas que de una forma u otra hicieron posible la realización de este trabajo. Especialmente a:

Oigres, mi tutor, por su ayuda, por todas las peleas que permitieron el éxito de este trabajo, por su amistad.

A mi novio y cotutor, mi niño, por su Amor, por estar siempre para mi, por creer y confiar en mi, por ser mi ángel de la guarda, por hacerme feliz, por cuidarme, por su apoyo, por TODO.

A mi mamá por estar en todo momento para mí, por su apoyo y amor incondicional, por todo su esfuerzo por hacerme una persona de bien, por estar orgullosa de mí, por ser la mejor madre.

A mi papá por todo lo que representa para mí.

A mi abuelo Chucho y mi abuelita linda Onelia por todo su cariño. A mis abuelos Nilo y Hirdeliza, por el orgullo que estarían sintiendo hoy, los extraño mucho.

A mis hermanos Ivancito y Yenly por ser mis alegrías, por existir.

A mi familia en general, en especial a mi tía Milagros, mi tío Ismael y mis primas Yudy y Yula.

A Estrellita por ser tan especial, por ser como una madre para mí. A Zenaida, por su cariño, por estar en todo momento.

A Todos mis amigos. Robe, Alian, Nelito, Diego, Lisbet, Lisandra, Yudita. Al 8108. A todos mis amigos del pre.

A la Revolución, a la Universidad de Ciencias Informáticas, al Comandante el Jefe Fidel, por contribuir en mi formación. A todos MUCHAS GRACIAS.

Resumen

El uso de la telefonía móvil ha evolucionado hasta lograr crecimientos considerables en pocos años. Lo que ha provocado que los usuarios de teléfonos móviles sean cada vez más exigentes en cuanto a diversidad y gusto de los contenidos que necesitan.

En Cuba, como en el resto del mundo, existe un incremento de la telefonía móvil, provocando que la Unidad de Negocios Móvil de ETECSA, Cubacel, tenga la necesidad de contar con una plataforma que permita la administración y descarga de contenidos hacia el teléfono móvil. Como parte del desarrollo de la plataforma es necesario implementar un mecanismo que les permita a los proveedores de contenidos y administradores poder interactuar con la misma de una forma sencilla y amigable.

Con el desarrollo de este trabajo de diploma con título "Interfaz de usuario para la administración de los servicios para la plataforma de telefonía móvil, COMCEL" se presenta una propuesta de solución informática que facilita el acceso a las funcionalidades que provee la plataforma telefónica para la gestión de los contenidos. En el mismo se abordan los fundamentos teóricos, así como la descripción de la metodología de desarrollo, las herramientas y tecnologías seleccionadas para el desarrollo del sistema. Se específica la arquitectura sobre la cual se basa el desarrollo y los diagramas que ilustran los elementos del diseño e implementación de la aplicación. El resultado final del trabajo es un sistema que sirve como interfaz de usuario a las funcionalidades que brinda la plataforma para la gestión de los contenidos.

Palabras Claves: Contenidos, plataforma de telefonía móvil COMCEL, Interfaz de usuario, ETECSA, Cubacel.

Índice de Contenidos

Introducción **10**

Capítulo 1: Fundamentación Teórica **13**

1.1 Introducción 13

1.2 Gestión de Contenidos para Móviles 13

1.3 Interfaz de Usuario 16

1.4 Aplicación Web 17

1.5 Plataforma de Desarrollo 19

1.5.1 Plataforma J2EE 19

1.5.2 Plataforma .NET 21

1.6 Tecnología del lado del cliente 21

1.6.1 GWT 21

1.6.2 Dojo 22

1.6.3 Prototype 23

1.7 Tecnología del lado del Servidor 23

1.7.1 Spring framework 23

1.8 IDE 25

1.8.1 Eclipse Ganymede 25

1.9 Metodologías de Desarrollo de Software 26

1.9.1 Proceso Unificado de Desarrollo (Rational Unified Process, RUP) 26

1.9.2 Programación Extrema (Extreme Programming, XP) 27

1.10 Herramientas Case 28

1.10.1 Visual Paradigm 29

1.10.2 Rational Rose 30
1.11 Conclusiones del Capítulo 30
Capítulo 2: Descripción y Diseño del Sistema 31
2.1 Introducción 31
2.2 Descripción del sistema a desarrollar 31
2.2.1 Requerimientos del Sistema 31
2.2.2 Actores del Sistema 34
2.2.3 Casos de Uso del Sistema 35
2.2.4 Diagrama de Casos de Uso 37
2.3 Descripción de la Arquitectura 38
2.3.1 Estilos Arquitectónicos 38
2.3.2 Arquitectura Cliente-Servidor 39
2.3.3 Arquitectura en Capas 40
2.3.4 Modelo – Vista – Controlador 42
2.3.5 Diseño de la Arquitectura del Sistema 43
2.4 Patrones de Diseño 46
2.4.1 Patrones GRASP 46
2.4.2 Patrones GoF 47
2.5 Modelo de Diseño 48
2.5.1 Diagrama de Clases del Diseño 48
2.5.3 Diagrama de Paquetes 58
2.6 Conclusiones del Capítulo 60
Capítulo 3: Implementación del Sistema 61
3.1 Introducción 61

Índice de Contenidos

3.2 Modelo de Implementación 61

3.2.1 Diagrama de Componentes 61

3.2.2 Descripción de los Componentes 62

3.2.3 Diagrama de Despliegue 64

3.3 Conclusiones del Capítulo 65

Capítulo 4: Pruebas 66

4.1 Introducción 66

4.2 Modelo de Prueba 66

4.2.1 Métodos de Pruebas 66

4.2.2 Diseño de Casos de Prueba 66

4.3 Conclusiones del Capítulo 71

Conclusiones Generales 72

Recomendaciones 73

Referencias Bibliográficas 74

Bibliografía 77

Anexo 78

Anexo #1: Diagrama de Secuencia Insertar Promociones 78

Glosario de Términos 79

Introducción

Con el incremento en los últimos años de la telefonía móvil el intercambio de información usando esta tecnología es cada vez mayor, brindando a los usuarios la posibilidad de tener información más novedosa y personalizada a sus necesidades. Actualmente se integran en el teléfono móvil servicios como: consulta de información, interacción con otros sistemas informáticos, descarga de contenidos, etcétera.

Existen en el mundo empresas como China Mobile, Vodafone, Orange y Movistar [1], que brindan servicios de descarga de contenidos (imágenes, videos, música, etc.), consultas de información relacionada con el horóscopo, estado del tiempo, deporte, cultura, directorio telefónico, cartelera de cine, y más; con el objetivo de cubrir la mayor cantidad de gustos y necesidades de los usuarios. Estas empresas proveedoras de servicios para móviles, no solo brindan el servicio, sino que gestionan información relacionada a estos, por ejemplo: datos de los proveedores de contenidos, el costo de un contenido, cantidad de suscripciones o permisos sobre un servicio, entre otras funcionalidades.

En Cuba también existe un incremento de la telefonía móvil, lo que trae consigo que la Unidad de Negocios Móvil de ETECSA, Cubacel, tenga como necesidad aumentar sus servicios de cara al cliente, para acercarse así a la tendencia actual de la telefonía móvil, con el objetivo de ganar en audiencia y ofrecer facilidades a los usuarios afiliados a la misma.

Como parte de su crecimiento, Cubacel promueve el desarrollo de sus propias herramientas para brindar sus servicios. Una de estas herramientas es la plataforma de servicios para telefonía móvil, COMCEL.

COMCEL es una plataforma de software que tiene como objetivo, integrar los servicios de telefonía móvil existentes en la actualidad en Cuba. Dentro de los servicios con que cuenta la plataforma están: la descarga de contenidos (imágenes, tonos y videos), consulta del estado del tiempo, directorio telefónico, noticias, entre otros. En la fase actual del desarrollo de la plataforma, no se cuenta con una herramienta que les permita a los usuarios, gestionar y administrar los servicios que oferta.

Se ha identificado como **problema a resolver** ¿Cómo desarrollar una interfaz de usuario para administrar los servicios que se brindan en la plataforma de telefonía móvil, COMCEL?

El **objeto de estudio** se enmarca en los sistemas de gestión para telefonía móvil, y el **campo de acción** en la gestión y administración de los servicios para teléfonos móviles en la plataforma COMCEL. El **objetivo general** de la investigación es desarrollar una interfaz de usuario para la administración de los servicios disponibles en la plataforma de telefonía móvil, COMCEL.

Con el fin de alcanzar el objetivo planteado se definen de los siguientes **objetivos específicos**:

- Evaluar y definir las herramientas, y metodologías de desarrollo de software más adecuadas para la construcción del sistema.
- Diseñar el sistema cumpliendo con los requerimientos y casos de usos definidos en el análisis.
- Implementar un software que permita la gestión y administración de los servicios de COMCEL, a partir del resultado obtenido en la fase de diseño.
- Realizar pruebas de calidad al sistema desarrollado.

Con el propósito de cumplir con todos los objetivos planteados se definen las siguientes tareas:

- Estudio de las tendencias actuales de los sistemas de gestión y administración de servicios para la telefonía móvil.
- Estudio de los frameworks existentes que facilitan el desarrollo de la aplicación y selección del más adecuado.
- Definición de la metodología de desarrollo de software que va a regir el proceso de desarrollo.
- Estudio de los patrones de diseño de software existentes y las pautas de programación establecidas para el desarrollo de COMCEL.
- Estudio del modelo de casos de uso y la especificación de los requerimientos del sistema, para conocer las funcionalidades que necesita la aplicación.
- Realización del diseño del sistema.
- Implementación de los componentes de la aplicación para la gestión y administración de los servicios, de forma tal que se integre con COMCEL.
- Realización de pruebas al sistema.

La implementación del sistema para la gestión y administración de los servicios que brinda la plataforma COMCEL, permitirá que Cubacel cuente con una herramienta que gestione y administre los servicios.

Estructura capitular

Capítulo 1: Fundamentación Teórica

En este capítulo se analizan aspectos teóricos que serán necesarios investigar para la correcta realización del trabajo. Se describen las metodologías de desarrollo de software, herramientas a utilizar y características de los sistemas de gestión y administración de servicios para telefonía móvil. Se definen las herramientas, tecnologías y metodología a usar para la construcción del sistema.

Capítulo 2: Descripción y Diseño del Sistema

Se modelan y describen los diagramas que representan las funcionalidades del sistema, aplicando los patrones de arquitectura y diseño seleccionados.

Capítulo 3: Implementación del Sistema

Se representa el diagrama de componentes que detalla la forma en que está estructurado el sistema, reflejando la transformación de los elementos del modelo del diseño en términos de componentes, así como las dependencias entre ellos.

Capítulo 4: Pruebas

Se diseñan las pruebas a realizarle al sistema. Se diseñan los casos de pruebas para comprobar el correcto funcionamiento de las principales funcionalidades de la aplicación.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se enuncian los principales conceptos teóricos que constituyen la base de la investigación realizada. Se realiza un estudio del arte de algunas soluciones existentes en el mundo y en Cuba para la gestión de contenidos, para dispositivos móviles. Se analizan las características fundamentales de las tecnologías, metodologías y herramientas de desarrollo que se utilizarán en el desarrollo del sistema.

1.2 Gestión de Contenidos para Móviles

En la actualidad existe gran demanda de contenidos para celulares: ringtones[1], fondos, animaciones, videos, programas y juegos, lo que provoca un incremento en las empresas encargadas de facilitar aplicaciones para la descarga de contenidos.

Con el objetivo de satisfacer los deseos y necesidades del cliente, los operadores[2] necesitan coordinar todos los recursos disponibles, para hacerlos llegar hasta el dispositivo móvil. Dicha coordinación implica fuertes interacciones entre el entorno (cliente móvil), las estructuras (aplicaciones para la gestión) y los productos (contenidos). Se conoce como proceso de gestión de contenidos, a las actividades que se desarrollan para cumplir este objetivo.

Existen empresas que se han dedicado a la gestión de contenidos para los dispositivos móviles. Algunas de las que podemos mencionar “Content Delivery Platform”, de la empresa Volatis, y “Mobile Web & Content Delivery Platform” creado por la empresa BeeWeb.

[1] Fichero con formato de sonido usado para indicar cuando se recibe una llamada o mensaje en el teléfono móvil.

[2] Un operador de telefonía móvil es una compañía telefónica que provee servicios de telefonía para clientes de teléfonos móviles.

Content Delivery Platform

Volantis Content Delivery Platform (V-CDP) proporciona la plataforma de software más completa e integrada para la entrega de contenido variado, innovador y personalizado, dentro de una experiencia de usuario sin fisuras y convincente, para el mayor número de abonados móviles. La Plataforma aplica el formato de presentación necesario a los contenidos según la interfaz de usuario y el dispositivo final. Permite a los operadores y proveedores de servicios de valor añadido la gestión y entrega de contenidos multimedia a escala general. Además presenta un completo juego de herramientas para el desarrollo, gestión y entrega de los contenidos. [2]

Mobile Web & Content Delivery Platform.

Mobile Web & Content Delivery Platform (MWCDP) surge como resultado de la experiencia y el grado de madurez alcanzado por BeeWeb en el suministro de soluciones para la administración de contenidos. La plataforma se basa en un sistema de centralización de funcionalidades y permite que los contenidos se puedan adaptar para cualquiera de los canales de distribución soportados. La integración, publicación de contenidos y prestación de servicios a los usuarios están agrupados en módulos de salida personalizables. Todos los módulos de salida son manejados por un módulo de administración de servicios central. Además cuenta con una librería para garantizar que todos los contenidos manejados por la plataforma (imágenes, textos, audio, videos, aplicaciones, etcétera.) sean almacenados con un adecuado formato, tamaño y nivel de compresión. [3]

En Cuba, a pesar de no contar con un desarrollo de punta en la gestión de contenidos para móviles, se están desarrollando algunos proyectos para responder al crecimiento de la telefonía celular.

Nuestro país ha experimentado un crecimiento acelerado de las comunicaciones móviles en los últimos años, y hoy existe un 70 por ciento de cobertura territorial a nivel nacional con tecnología GSM, la cual cubre al 77,5 por ciento de la población. [4]

Muestra de esta expansión es la cantidad de radio bases instaladas para asegurar la transmisión, que pasó de 60 en el año 2003 a 350 en la actualidad, mientras existe el firme propósito de terminar el año 2010 con al menos una instalada en todos los municipios del país, cubriendo así a los 23 que en estos momentos no tienen. [4]

No solo ha crecido la cobertura, sino también la cantidad de líneas activadas, con más de 600 000 nuevas desde el 2008; así como el tráfico o tiempo consumido en llamadas diarias, que pasó de 1,94 millones de minutos en el 2007 a 2,80 millones de minutos estimados para el 2010. [4]

La Oficina Nacional de Estadísticas (ONE) había reportado 331 mil 270 usuarios de móviles antes de la liberación del servicio, en abril de 2008, mientras el diario Juventud Rebelde que el número de líneas saltó hasta 838 mil 370 en marzo pasado. [5]

Con todo este crecimiento las demandas de contenidos se han acelerado a la par de que los usuarios se hacen más exigentes y variados. Se hace necesario entonces desarrollar soluciones para la gestión de contenidos. Como referencia de desarrollos de plataformas de este tipo en nuestro país se encuentra Procyon Soluciones[3] que les ofrece a los usuarios de teléfonos móviles, servicios basados en mensajes de texto o SMS. [6]

Estos servicios pueden ser utilizados por cualquier persona, que cuenten con un dispositivo móvil, con una línea de Cubacel, y se activan cuando el usuario envía un SMS a un número corto predeterminado con un texto cuya primera palabra identifica el servicio solicitado. El usuario recibirá como respuesta un SMS con la información solicitada. [6]

Entre los servicios que actualmente se brindan a la población se encuentran: [6]

- Información acerca del estado del tiempo.
- Seguimiento de huracanes.
- Datos de embajadas.
- Presentaciones de artistas y grupos artísticos.
- Cartelera de centros culturales y recreativos.
- Seguimiento de eventos deportivos.

[3] División de Telecomunicaciones de DESOFT S.A y Cubacel.

1.3 Interfaz de Usuario

La interfaz de usuario se define como un entorno o herramienta que permite una conexión física y funcional entre dos aparatos, es decir, el medio por el cual los usuarios se pueden comunicar con la computadora. [7]

Las interfaces de usuarios han evolucionado logrando ser más amigables, intuitivas y prácticas; en los inicios se trabajaban con consolas, modo de texto. En este tipo de interfaces los usuarios necesitan conocer los comandos a introducir para ser procesados y obtener los resultados. La interfaz de consola tiene una limitada capacidad expresiva, las primeras solo contaban con un tipo de texto sin colores y una cantidad de caracteres limitado. Aunque con el desarrollo de la informática se han ido enriqueciendo las posibilidades de estas terminales, las ventajas expresivas que brinda la interfaz gráfica son muy superiores.

Las interfaces gráficas son una evolución natural de las interfaces modo consola y la facilidad de uso que se puede alcanzar dentro de una interfaz gráfica es mucho mayor. Por ejemplo, el uso de menús permite que el usuario pueda ejecutar comandos de forma guiada, sin necesidad de conocer la sintaxis concreta del mismo. Con un diseño adecuado de los menús de una aplicación, se pueden poner a disposición del usuario decenas de comandos fácilmente accesibles. Además que permiten diseños de mucha riqueza visual y más expresivos.

Buscando una homogeneidad entre los millones de páginas web que existen actualmente en Internet se han definido elementos y agrupaciones de estos que han demostrado su utilidad y su comprensión por los usuarios, entre los que se puede destacar los sistemas de navegación[4], los dinteles[5], los pies de página[6],

[4] Los sistemas de navegación constituyen elementos de una interfaz que permiten la navegación por las diferentes secciones y vistas que componen la aplicación Web. Generalmente se presentan como menús formados por diferentes opciones

[5] El Dintel constituye una zona de la interfaz Web situada en la parte superior, de anchura generalmente igual a la de la página y altura variable, en la que se ubica comúnmente el logotipo de la aplicación Web o la empresa propietaria

[6] El pie de página es un elemento opcional que suele contener información muy concreta. Si la página es muy alta, de tal forma que el usuario se vea obligado a utilizar la barra de desplazamiento vertical del navegador, el pie de página suele contener un menú auxiliar que permita al usuario continuar navegando por el sistema sin tener que volver a buscar el menú principal.

los formularios[7] de entrada de datos, etcétera, que normalmente se encuentran en todas las páginas web y cuyo diseño y funcionalidad son similares en todas ellas. [8]

Una interfaz Web cuenta con cuatro características fundamentales, que son: contenido, tecnología, aspectos visuales y económicos. El contenido es a través del cual se informa a los usuarios; la tecnología es quien ofrece la funcionalidad, interactividad y el dinamismo del sistema; los temas visuales influyen en el impacto visual de la aplicación hacia la audiencia; y las implicaciones económicas en la factibilidad de la construcción del sistema. [9]

1.4 Aplicación Web

Una aplicación Web es un software basado en tecnologías y estándares de W3C[8] que provee recursos específicos tales como contenidos y servicios a través de una interfaz de usuario a la que puede accederse utilizando un navegador Web [10]. Intenta portar las clásicas aplicaciones de escritorio hacia entornos Web, que son utilizadas a través de los distintos navegadores, agregando portabilidad y capacidad de acceder desde diferentes dispositivos.

Una de las ventajas más significativas de las aplicaciones Web, es su forma de instalación y distribución, ya que normalmente instalar una aplicación Web consiste en configurar los componentes del lado del servidor en la red y no necesariamente una instalación o configuración en el lado del cliente. Son independientes de las tecnologías de hardware y software (sistema operativo) que se utilicen. Una empresa podría realizar un proceso de migración y todo quedar funcionando correctamente.

El diseño de las aplicaciones Web ha evolucionado con el tiempo hacia un esquema general perfectamente definido, ofreciendo unas interfaces bien diseñadas, con un conjunto de componentes gráficos y funcionales similares que hacen posible que sea cual sea el usuario que accede a una aplicación Web, la comunicación entre ellos sea posible y efectiva.

[7] Los formularios de entrada de datos en los cuales los usuarios registran la información solicitada para que posteriormente persista en el sistema de almacenamiento de datos que esté acoplado a dicha aplicación Web.

[8] World Wide Web Consortium. Consorcio internacional de compañías y organizaciones involucradas en el desarrollo de Internet y en especial de la WWW. Su propósito es desarrollar estándares y "poner orden" en Internet.

La evolución de la Web ha sido continua; llegaron las imágenes a acompañar los textos, aparecieron las primeras animaciones y las primeras herramientas interactivas. Actualmente, y desde hace un tiempo, la Web está sufriendo una evolución importante y marcada que merece mucha atención especialmente por la gran explosión de contenidos y de aplicaciones realmente útiles y sorprendentes. Esta evolución no es solo tecnológica, sino que también es en una manera distinta de ver y hacer las cosas; a estos cambios es a lo que muchos llaman Web 2.0.

La Web 2.0 es la representación de la evolución de las aplicaciones tradicionales hacia aplicaciones web enfocadas al usuario final [11]. Se trata de aplicaciones que generen colaboración y servicios que reemplacen las aplicaciones de escritorio. El término surgió durante una conferencia impartida por Dale Dougherty donde se hablaba del renacimiento y evolución de la Web. Con la Web 2.0 se ha pasado de la complejidad técnica para publicar, a la facilidad para producir y compartir. Los usuarios están convirtiéndose en participantes activos de su desarrollo, creando y compartiendo contenidos.

La Web 2.0 no es precisamente una tecnología, sino la actitud con la que se debe trabajar para desarrollar aplicaciones. Ha estado sustentada en el uso de diferentes tecnologías y estándares, como XHTML[9], la separación del contenido y el diseño a través de hojas de estilo (CSS) y el uso de AJAX, agregando mucha más interactividad con los usuarios.

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Con el uso de esta técnica el intercambio de información con el servidor se realiza en segundo plano evitando así la recargas constantes de páginas, lo que permite mejorar totalmente la interacción del usuario con la aplicación.

[9] XHTML, Extensible Hypertext Markup Language (Lenguaje Extensible de Marcado de Hipertexto) extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos.

1.5 Plataforma de Desarrollo

1.5.1 Plataforma J2EE

Java es un lenguaje de programación orientado a objetos, seguro, portable y multitarea, creado por Sun Microsystems[10]. El lenguaje en sí mismo toma mucho de C y C++ pero elimina herramientas de bajo nivel, como la manipulación directa de punteros o memoria.

La plataforma Java ha sido dividida en tres ediciones distintas según sus diferentes objetivos: Java 2 Standard Edition (J2SE), Java 2 Micro Edition (J2ME) y Java 2 Enterprise Edition (J2EE).

J2EE, es una propuesta de SUN para el desarrollo y la implementación de aplicaciones corporativas multinivel [12]. Se apoya completamente en el lenguaje Java beneficiándose, por tanto, de sus características. Esta plataforma incluye las funcionalidades de la Standard Edition y muchas más extensiones.

Su estructura está basada en J2SE y un conjunto de sus APIs[11], a las que J2EE les aporta un conjunto de componentes, contenedores (containers) y las APIs para los servicios de transacciones, mensajería, servicio de correo, y conectores de recursos externos. [13]

[10] Es una empresa informática de Silicon Valley, fabricante de semiconductores y software. Las siglas SUN se derivan de "Stanford University Network", proyecto que se había creado para interconectar en red las bibliotecas de la Universidad de Stanford.

[11] Application Programming Interface, es el conjunto de objetos y funciones que ofrece cierta biblioteca para ser utilizado por otro software.

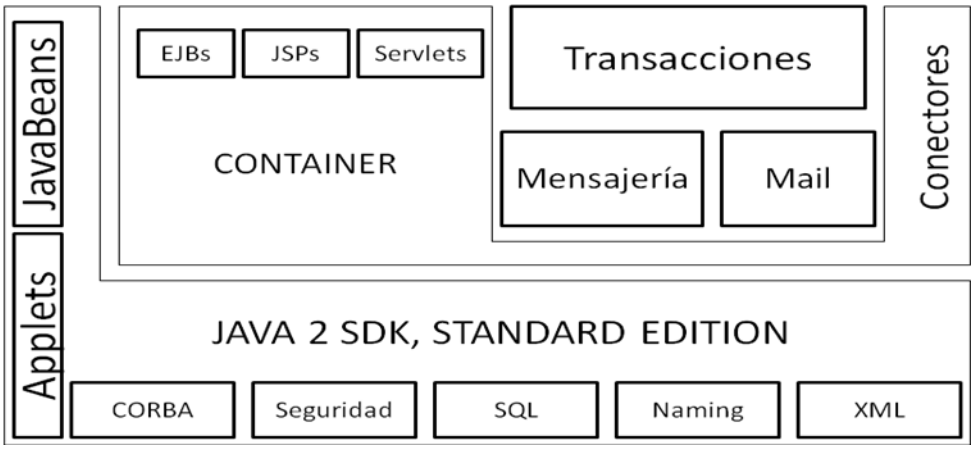


Figura 1 Estructura J2EE.

J2EE proporciona: [12]

- Un modelo de desarrollo de componentes Web (Servlet[12], JSP) y de componentes activos (EJB) bajo la forma de APIs de Java.
- Un conjunto de servicios (JDBC, JTA, JNDI, JMS, RMI/IIOP, JavaMail, XML), herramientas para los componentes, bajo la forma APIs de Java.
- Un modelo de creación de módulos Web (.war), de módulos EJB (. jar) y de módulos corporativos (.ear), asociados a descriptores de despliegue en formato XML, herramientas para el desarrollo de aplicaciones de empresa.
- Contenedores (web y EJB), para la realización de los componentes.

Uno de los beneficios de J2EE es que se puede empezar con poco o ningún coste, además, la implementación J2EE de Sun Microsystems puede ser descargada gratuitamente, y hay muchas herramientas de código abierto, disponibles para extender la plataforma o para simplificar el desarrollo.

[12] Un servlet es un objeto que se ejecuta en un servidor o contenedor JEE (ej. Tomcat), especialmente diseñado para ofrecer contenido dinámico desde un servidor web, generalmente HTML.

1.5.2 Plataforma .NET

La plataforma .Net es una implementación de Microsoft basada en estándares abiertos. Soporta múltiples lenguajes de programación, desde los más conocidos como C# y Visual Basic hasta otros no tan conocidos como Cobol y Perl. Esta va más allá de soportar estos lenguajes, sino que también ofrece la interoperabilidad entre ellos. Ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Además funciona en máquinas basadas en Win32 y bajo sistema operativos Windows.

1.6 Tecnología del lado del cliente

Básicamente un framework consiste en un conjunto de utilidades y bibliotecas que facilitan la realización de un desarrollo de software. Constituyen implementaciones basadas en patrones de diseño, soluciones comunes normalmente empaquetadas para hacer más rápido el desarrollo teniendo en cuenta las buenas prácticas. Son soluciones completas que contemplan herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución).

Los framework JavaScript se pueden dividir en dos grupos, los que no tienen dependencia de la plataforma del servidor y los que necesitan la programación de lado del servidor. A continuación se hace referencia a algunos de estos frameworks.

1.6.1 GWT

Google Web Toolkit (GWT) 1.7.1 es un framework desarrollado por Google y escrito en Java que permite crear aplicaciones con AJAX fácilmente. Traduce el código Java a JavaScript lo que permite crear aplicaciones Web dinámicas, compatibles con la mayoría de los navegadores (FireFox, Internet Explorer, Mozilla, Safari, y Opera).

Algunas de las características de GWT son [14]:

- **Componentes de Interfaz gráfica reusables:** permite crear widgets[13] personalizados y compartirlos con otros proyectos como archivos JAR.
- **RPC verdaderamente simple:** El RPC[14] de GWT automáticamente serializa y des-serializa las llamadas a códigos remotos de manera que el programador no debe preocuparse por crear los métodos que usualmente se necesitan para manejar la respuesta y las implicaciones que la depuración de errores y del manejo de las excepciones.
- **Administración de la historia del navegador:** Permite de una manera automática tener un control de la historia de la aplicación sin tener que recurrir a los botones del navegador.
- **Depurador real:** Presenta el error y las excepciones ocurridas con la salida completa a través de la consola, lo cual facilita el proceso de depuración.
- **Internacionalización:** Fácil internacionalización de aplicaciones y bibliotecas.
- **Interoperabilidad y Control:** Permite escribir código JavaScript para incluir en las aplicaciones.

GWT Ext o GXT 2.0.2 es un wrapper (envoltorio) que permite crear una aplicación Ext JS desde GWT**.** Este aporta una gran cantidad de widgets más elegantes y más variados.

1.6.2 Dojo

El framework Dojo facilita el desarrollo de aplicaciones Web que utilicen la tecnología AJAX. Resuelve asuntos de usabilidad comunes como la navegación y detección del navegador. Provee además un sistema de paquetes para facilitar el desarrollo modular. Proporciona una gama más amplia en una sola biblioteca JavaScript y es compatible con navegadores antiguos.

Con Dojo se pueden construir interfaces con degradados de color, widgets y transacciones animadas de forma rápida y sencilla.

[13] Es un componente gráfico, o control, con el cual el usuario interactúa, como por ejemplo, una ventana o una caja de texto.

[14] El mecanismo para interactuar con el servidor a través de la red es llamado Remote Procedure Call (RPC). El RPC en GWT permite fácilmente al cliente enviar y recibir objetos de Java sobre HTTP.

1.6.3 Prototype

Prototype es un framework basado en JavaScript que se orienta al desarrollo sencillo y dinámico de aplicaciones Web. Es una herramienta que implementa las técnicas AJAX lo que simplifica el trabajo cuando se pretende desarrollar páginas altamente interactivas.

Una de las características fundamentales de Prototype es que utiliza JSON (JavaScript Object Notation). JSON es una alternativa rápida y más ligera que XML para las solicitudes de AJAX.

1.7 Tecnología del lado del Servidor

1.7.1 Spring framework

Spring es un framework de código abierto que facilita el desarrollo de aplicaciones para la plataforma Java. Está basado en la técnica de Inversión de Control (IoC) y una implementación de desarrollo según el paradigma Orientación a Aspectos (AOP).

Una de las principales ventajas de Spring es que proporciona la posibilidad de integrar al framework con otras herramientas o incluso otros frameworks con el fin de obtener los beneficios que el desarrollador desea de cada uno de ellos.

Spring 2.5.6 está diseñado en módulos que puedes utilizar o no al desarrollar una aplicación. Estos módulos son [15]:

- **Spring ORM:** Presenta una Capa de Integración con APIs de Mapeo Objeto – Relacional.
- **Web:** Provee características de integración orientadas a la Web, como funcionalidad continua (para realizar la carga de archivos), inicialización de contextos mediante servlet listeners y un contexto de aplicación orientado a web.
- **Spring AOP:** La Programación Orientada a Aspectos (AOP) permite una adecuada modularización de las aplicaciones y posibilita una mejor separación de conceptos. Permite encapsular los diferentes conceptos que componen una aplicación en entidades bien definidas, eliminando las dependencias entre cada uno de los módulos.

- **Spring Core:** Es el núcleo de Spring, permite técnicas de Inversión del Control (IoC) como la inyección de dependencias.
- **DAO:** Proporciona una capa de abstracción JDBC y una forma de administrar transacciones.
- **Context:** Es un archivo de configuración que provee información del framework. Incluye los servicios empresariales como e-mail, JNDI, EJB, internacionalización y aplicaciones de eventos del ciclo de vida del software.
- **Spring MVC:** Provee las implementaciones que ayudan al uso de una arquitectura Modelo - Vista - Controlador para aplicaciones Web.

Spring facilita buenas prácticas de diseño y programación. Presenta una programación basada en POJOs[15]

El sub proyecto **Spring Web Services 1.5.5** [16] permite la configuración y consumo de un servicio web (web service[16]). Se basa en el concepto de contrato primero, con el cual se define primero el contrato del servicio y luego se implementa, evitando atar al contrato como sucede en los casos en los cuales se genera el mismo a partir de las clases Java.

Una de las mayores ventajas al utilizar Spring-WS es la posibilidad de inyectarle de forma transparente alguna librería que se encargue de convertir el mensaje que recibimos en un objeto java y también realizar el proceso inverso.

Spring Security 2.0.4 es una API que provee los servicios de autenticación y autorización a las aplicaciones basadas en Spring. Provee un conjunto de clases que se configuran en el contexto de la aplicación, explotando al máximo las características de programación orientada a aspectos y de inyección de dependencias que contiene Spring. Soporta la mayoría de los procesos de autenticación existentes.

[15] POJOs: Plain Old Java Object revalora la simplicidad de las clases Java aportando manejo de transacciones de forma no intrusiva.

[16] Un servicio web es un componente programable que proporciona un servicio y es accesible por Internet. Sistema de software identificado por una URI, cuya interfaz pública y enlaces son definidos y descritos utilizando XML.

1.8 IDE

Un Entorno de Desarrollo Integrado (IDE, Integrated Development Enviroments) es un programa informático compuesto por herramientas de programación. Consiste en un compilador, un editor de código, un depurador y un constructor de interfaz gráfica (GUI), empaquetados como un programa de aplicación. Provee un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, Visual Basic, Delphi, etc. [17]

1.8.1 Eclipse

Eclipse 3.4 es principalmente una plataforma de programación, diseñada para la integración de IDEs, presenta una arquitectura abierta y basada en plug – ins[17], lo que le permite integrar diversos lenguajes sobre un mismo IDE e introducir otras aplicaciones accesorias. Soporta a los proyecto durante todo el ciclo de vida de desarrollo e incluye soporte de modelado. Permite generar y mantener documentación de cada etapa del proyecto y conservar el registro de versiones.

Eclipse es soportado en los principales sistemas operativos: Linux, Windows, Solaris 8, entre otros.

Cypal Studio Para GWT [18] [19] (conocido antes como el Googlipse) fue diseñado para el Eclipse IDE, y proporciona una ayuda para crear nuevos módulos GWT, apoya la creación de procedimientos remotos y facilita la tarea para ver y desplegar las aplicaciones Web.

Sub Eclipse es un plug-in para eclipse. Permite el desarrollo colaborativo, facilita el envió y descarga de información desde el eclipse hacia un repositorio de datos. Soporta conectarse a varios repositorios de control de versiones al mismo tiempo, permitiendo hacer operaciones sobre el repositorio directamente.

Este plug-in es muy útil para el desarrollo colaborativo, en el que intervienen un conjunto de desarrolladores trabajando sobre el mismo proyecto, poniendo a disposición del equipo de desarrollo facilidades para el trabajo en equipo.

[17] Es la mínima unidad de la plataforma que puede ser desarrollado por separado y que aporta una nueva funcionalidad.

1.9 Metodologías de Desarrollo de Software

Las metodologías son un conjunto de procedimientos, técnicas, herramientas y ayudas a la documentación para el desarrollo de software. Guían el proceso de desarrollo de software, e imponen una disciplina sobre el desarrollo del mismo con el fin de hacerlo más predecible y eficiente. Las características de cada proyecto exigen que sea configurable, condiciones que llevan a decir que no existe una metodología de software universal. [20]

Existen en el mundo diferentes metodologías para llevar a cabo el desarrollo de software. Agrupadas en las conocidas metodologías pesadas o tradicionales y las metodologías ágiles.

1.9.1 Proceso Unificado de Desarrollo (Rational Unified Process, RUP)

RUP es una metodología de desarrollo, se encuentra en el grupo de las metodologías pesadas. Utiliza como lenguaje de modelado UML, es orientada a objetos y genera una gran cantidad de documentación, elemento importante para los equipos de desarrollo cuyo personal varía constantemente y en proyectos con una gran duración.

RUP es una forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cómo y cuándo). Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales, respetando cronograma y presupuesto.

El proceso de desarrollo de RUP tiene tres características esenciales: está dirigido por casos de uso, centrado en la arquitectura, y es iterativo e incremental.

- **Dirigido por Casos de Usos:**

Los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que sería bueno contemplar. Se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Estos representan los requisitos funcionales del sistema. [21]

En RUP los Casos de Uso no son solo una herramienta para especializar los requisitos del sistema sino que guían su diseño, implementación y prueba.

- **Centrado en la arquitectura:**

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo. [21] [22]

En el caso de RUP además de utilizar los Casos de Uso para guiar el proceso se presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento. Cada producto tiene tanto una función como una forma. La función corresponde a la funcionalidad reflejada en los Casos de Uso y la forma la proporciona la arquitectura.

- **Iterativo e Incremental:**

El equilibrio correcto entre los Casos de Uso y la arquitectura es algo muy parecido al equilibrio de la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. Para esto, la estrategia que propone RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) de la cual se obtiene un incremento que produce un crecimiento en el producto. [22]

El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajos relevantes y refinando la arquitectura.

1.9.2 Programación Extrema (Extreme Programming, XP)

La metodología de desarrollo XP nace en busca de simplificar el desarrollo del software. Combina las que se consideran las mejores prácticas de programación y las lleva al extremo.

Los objetivos de XP son muy simples: la satisfacción del cliente y potenciar al máximo el trabajo en equipo. Trata de dar al cliente el software que necesita y cuando lo necesita. Tanto los jefes de proyectos, clientes y desarrolladores son parte del equipo y están involucrados en el desarrollo del software.

Programación Extrema es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. La metodología se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. [23]

Programación Extrema es una disciplina de desarrollo que se basa en la simplicidad, comunicación y retroalimentación. Es un sistema de prácticas mínimas, supone utilizarlas todas en el principio de un proyecto y adaptarlas, y agregar adicionales a medida que se experimente la necesidad. Propone cuatro fases: Planificación, Diseño, Desarrollo y Pruebas.

1.10 Herramientas Case

La principal ventaja de la utilización de las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) es la mejora en la calidad de los desarrollos realizados y, en segundo término, el aumento de la productividad.

Estas herramientas permiten un desarrollo y un refinamiento visual de las aplicaciones mediante la utilización de controles gráficos, permiten además generar automáticamente el código base, proporcionan la reutilización de componentes y hace más portable la documentación a generar.

Entre ellas se pueden mencionar dos grandes herramientas de modelado visual que soportan UML, estas son: Visual Paradigm y Rational Rose. A continuación se expone una breve referencia del estudio realizado sobre estas herramientas.

1.10.1 Visual Paradigm

Visual Paradigm para UML es una herramienta CASE multiplataforma que da soporte al modelado visual con UML 2.0. Soporta el ciclo de vida completo del desarrollo de software: procesos del negocio, análisis y diseño orientados a objetos, implementación, pruebas y despliegue. Brinda la posibilidad de crear un conjunto bastante amplio de artefactos utilizados con mucha frecuencia durante la confección de un software.

Puede ser utilizado en varios idiomas y por una amplia gama de usuarios, incluyendo ingenieros de software, analistas de sistema, analistas de negocio, arquitectos de sistema, así como todos los que estén interesados en la construcción de sistemas confiables a gran escala usando un enfoque orientado a objetos. Se hace muy fácil la creación de cualquier tipo de diagrama ya que sus componentes se encuentran relacionados y un componente sugiere la utilización de nuevos posibles componentes a colocar en un diagrama que se encuentre en desarrollo. Se integra fácilmente con varios IDEs, entre ellos Visual Studio y Eclipse. Brinda facilidades para redactar especificaciones de Casos de Uso sin necesidad de una herramienta externa como editor de texto. Mediante la utilización de esta herramienta se puede generar código fuente a partir de los diagramas creados para plataformas como .Net, Java, y PHP; así como la obtención de diagramas a partir del código fuente, proceso conocido como ingeniería inversa.

Como elemento importante, Visual Paradigm brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose. Visual Paradigm ofrece además:

- Diseño centrado en Casos de Uso y enfocado al negocio para obtener un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Modelo y código que permanecen sincronizados en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.

1.10.2 Rational Rose

Rational Rose es la herramienta CASE que soporta de forma completa la especificación de UML. Propone la utilización de cuatro tipos de modelados para realizar un diseño del sistema utilizando una vista estática y otra dinámica de los modelos del sistema: uno lógico y otro físico. Cubre todo el ciclo de vida del proyecto y está disponible para la plataforma Windows. Esta herramienta permite la generación de código a partir de un diseño en UML para lenguajes como: C++, Java, Visual Basic, Ada, CORBA. Además, permite realizar ingeniería inversa por lo que se puede obtener un diseño a partir del código fuente de un programa.

1.11 Conclusiones del Capítulo

En este capítulo se realizó un estudio de los conceptos necesarios para el desarrollo de la aplicación. Teniendo en cuenta las características de las metodologías de desarrollo, herramientas y tecnologías descritas, y que se adecuan correctamente al tipo de desarrollo a realizar; se selecciona la metodología de desarrollo de software RUP para la planificación y elaboración del sistema. Se escoge como plataforma de desarrollo J2EE y los frameworks Google Web Toolkit y Spring. Se propone Visual Paradigm para el modelado; y Eclipse como Entorno de Desarrollo Integrado. Para la elaboración de interfaces más amigables se escogió GXT 2.0.2. Se elige Spring Web Service para consumir servicios web y Spring Security para implementar la seguridad de la aplicación.

Capítulo 2: Descripción y Diseño del Sistema

2.1 Introducción

En este capítulo se realiza una breve descripción del sistema. Se presentan los requisitos funcionales y no funcionales, se definen los actores y casos de uso, y se hace una descripción breve de estos. Se describe la arquitectura base del sistema a partir de las tecnologías seleccionadas para la construcción del mismo. Además se diseñan las clases teniendo presente el uso de patrones de diseño.

2.2 Descripción del sistema a desarrollar

El sistema a desarrollar tiene como objetivo fundamental proporcionar una interfaz amigable para la gestión de los servicios de la plataforma COMCEL. Debe ser capaz de gestionar información de los proveedores de contenidos, de los servicios que ofrece la plataforma, así como de los contenidos y su información complementaria asociada. El sistema debe permitir el acceso a las personas autorizadas desde cualquier sitio físico, garantizando que los proveedores puedan gestionar sus contenidos desde cualquier área o local donde se encuentren. Debe ser capaz de integrarse a la plataforma telefónica y acceder a sus funcionalidades. Además debe garantizar la seguridad de la información que maneja así como tiempos de respuestas aceptables para las peticiones de los usuarios.

2.2.1 Requerimientos del Sistema

La IEEE Standard Glossary of Software Engineering Terminology define un requerimiento como condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo. Todas las ideas que tengan clientes, usuarios y miembros del equipo de proyecto acerca de qué debe hacer el sistema, deben ser analizadas como candidatas a requisitos. Los requisitos se pueden clasificar en: funcionales y no funcionales. [22]

A continuación se presentan los requerimientos funcionales y no funcionales capturados para el desarrollo del sistema.

Requisitos Funcionales

R1. Autenticar usuario.

R2. Gestionar contenido.

- 2.1 Insertar contenido.
- 2.2 Modificar contenido.
- 2.3 Eliminar contenido.
- 2.4 Deshabilitar contenido.

R3. Gestionar proveedor.

- 3.1 Insertar proveedor.
- 3.2 Modificar proveedor.
- 3.3 Eliminar proveedor.

R4. Gestionar promoción.

- 4.1 Insertar promoción.
- 4.2 Modificar promoción.
- 4.3 Eliminar promoción.

R5. Gestionar noticia.

- 5.1 Insertar noticia (RSS[18]).
- 5.2 Modificar noticia (RSS).
- 5.3 Eliminar noticia (RSS).
- 5.4 Eliminar categoría de noticia

R6. Gestionar estado del tiempo.

- 6.1 Modificar estado del tiempo.

R7. Gestionar Directorio de Páginas Amarillas.

- 7.1 Insertar entrada del Directorio de Páginas Amarillas.
- 7.2 Modificar entrada del Directorio de Páginas Amarillas.
- 7.3 Eliminar entrada del Directorio de Páginas Amarillas.

R8. Gestionar usuario.

- 8.1 Insertar usuario.

[18] El RSS es un sistema ideado para extraer la información que se actualiza con frecuencia, como noticias, mensajes de un foro o artículos de un blog (o bitácora) y usarla en otra web o en un programa.

8.2 Modificar usuario.

8.3 Eliminar usuario.

8.4 Cambiar contraseña.

Requisitos No Funcionales

Los requisitos no funcionales se definen como las cualidades o propiedades que el software debe tener [22]. Debe pensarse en estas propiedades como las características que hacen un producto atractivo, usable, rápido y confiable. Normalmente están vinculados a los requerimientos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer, se puede determinar cómo ha de comportarse. Los clientes y usuarios pueden valorar las características no funcionales del producto, y así marcar la diferencia ente un producto bien aceptado y uno con poca aceptación.

1. Usabilidad.
   El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.
   1.1. Mensajes y Textos en la interfaz: Tanto la interfaz, como los menajes para interactuar con los usuarios, así como los mensajes de error, deberán ser en idioma castellano y tener una apariencia estándar.
   1.2. El sistema debe tener acceso al menú general desde cualquiera de sus páginas:
   1.3. El sistema funcionará de manera óptima en los navegadores Web, Mozilla Firefox 3.5.0 en adelante e Internet Explorer 6.
   1.4. El sistema funcionará correctamente con dimensiones de 1024 x 768 pixeles en las ventanas de los navegadores Web.
2. Fiabilidad.
   2.1 El sistema deberá estar disponible los 7 días de la semana durante 24 horas al día.
   2.2 Políticas de seguridad por usuarios y roles*:* el sistema debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado.
   2.3 Servicios Web restringidos: los servicios Web que consume el sistema deben estar restringidos a grupos de usuarios definidos y aprobados previamente.
3. Seguridad.

3.1 Para que se realice cualquier operación el usuario debe estar registrado. Todos los servicios Web y funcionalidades deben tener seguridad a través de cuentas de usuario con roles que definen el nivel de acceso.

1. Eficiencia.

1.1 Tiempo de respuesta por transiciones: los tiempos de respuesta deberán ser reducidos al máximo.

2. Restricciones del Diseño.

2.1 Metodología de Desarrollo: RUP.

2.2 Lenguaje de Modelado: UML.

2.3 Plataforma de desarrollo: J2EE.

2.4 Lenguaje de Programación: Java.

2.5 IDE: Eclipse.

2.6 Herramientas: Visual Paradigm, Sub Eclipse, Cypal Studio.

3. Apariencia o Interfaz Externa.

3.1 El sistema debe implementar una interfaz Web con un diseño sencillo y debe comunicarse a través del protocolo de comunicación http.

4. Requisitos Legales, de Derecho de Autor y otros.

4.1 El sistema debe ser sometido a una evaluación y certificación por parte del cliente.

2.2.2 Actores del Sistema

Tabla 1. Actores del Sistema.

Actores	Descripción
Proveedor	Es el encargado de gestionar el contenido.
Administrador	Es el encargado de gestionar todo lo referente a las noticias, directorio telefónico, estado del tiempo, usuarios y promociones.

2.2.3 Casos de Uso del Sistema

A continuación se exponen los casos de uso de alto nivel, para lograr entender los procesos globales de la aplicación.

Tabla 2. Caso de Uso Autenticar Usuario.

CU-1	Autenticar Usuario
Actor	Proveedor, Administrador
Descripción	El actor se autentica en el sistema mediante usuario y contraseña.
Referencia	RF1

Tabla 3.Caso de Uso Gestionar Contenido.

CU-2	Gestionar Contenido
Actor	Proveedor
Descripción	El Proveedor realiza sobre el contenido múltiples operaciones como pueden ser: insertar, modificar, eliminar y/o habilitar contenido.
Referencia	RF2

Tabla 4.Caso de Uso Gestionar Proveedor.

CU-3	Gestionar Proveedor
Actor	Administrador
Descripción	El Administrador puede insertar un nuevo proveedor, así como modificar y/o eliminar los ya existentes.
Referencia	RF3

Tabla 5.Caso de Uso Gestionar Promoción.

CU-4	Gestionar Promoción
Actor	Administrador
Descripción	El Administrador realiza sobre la promoción determinadas acciones como son insertar, eliminar una promoción, así como modificar las ya existentes.
Referencia	RF4

Tabla 6.Caso de Uso Gestionar Noticia.

CU-5	Gestionar Noticia
Actor	Administrador
Descripción	El Administrador puede insertar un nuevo RSS de noticias, así como modificar y/o eliminar los ya existentes.
Referencia	RF5

Tabla 7.Caso de Uso Gestionar Estado del Tiempo.

CU-6	Gestionar Estado del Tiempo
Actor	Administrador
Descripción	El Administrador puede modificar el RSS del Estado del Tiempo existente.
Referencia	RF6

Tabla 8.Caso de Uso Gestionar Directorio de Páginas Amarillas.

CU-7	Gestionar Directorio de Páginas Amarillas
Actor	Administrador
Descripción	El Administrador puede realizar sobre el directorio

	múltiples operaciones como pueden ser: insertar, modificar y/o eliminar una entrada del directorio.
Referencia	RF7

Tabla 9.Caso de Uso Gestionar Usuario.

CU-8	Gestionar Usuario
Actor	Administrador
Descripción	El Administrador puede insertar un nuevo usuario, así como modificar, y/o eliminar los ya existentes.
Referencia	RF8

2.2.4 Diagrama de Casos de Uso

En la figura 2 se observan los Casos de Uso definidos para representar el flujo de los eventos inicializados por los actores de la aplicación.

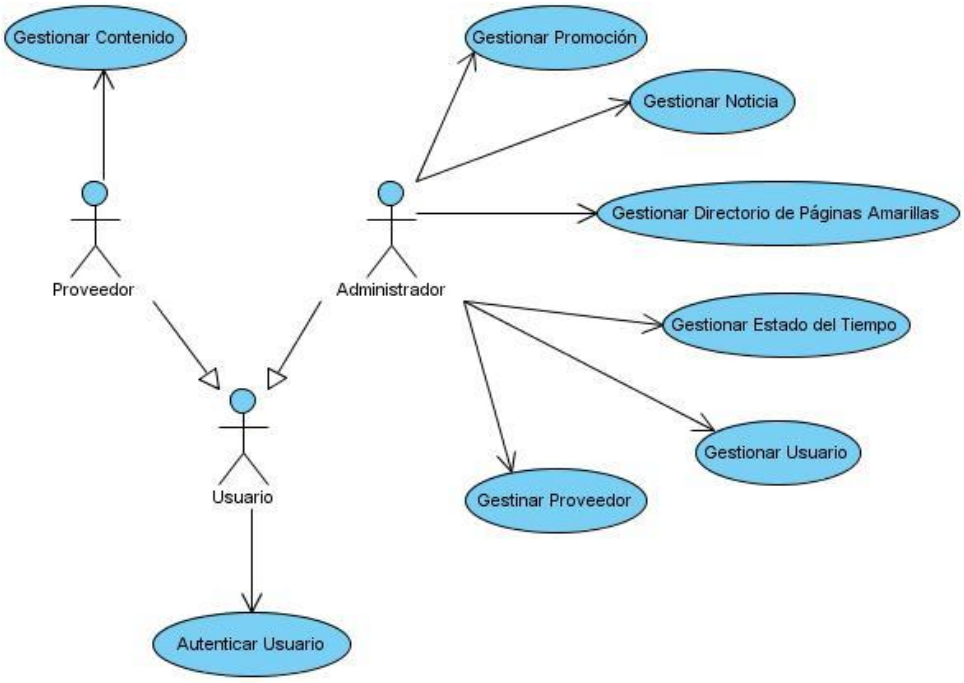


Figura 2. Diagrama de Casos de Uso del Sistema.

2.3 Descripción de la Arquitectura

En los inicios de la informática, la programación se consideraba un arte y se desarrollaba como tal, debido a la dificultad que entrañaba para la mayoría de las personas, pero con el tiempo se han ido descubriendo y desarrollando formas y guías generales, con base a las cuales se puedan resolver los problemas. A estas, se les ha denominado Arquitectura de Software, porque, a semejanza de los planos de un edificio o construcción, estas indican la estructura, funcionamiento e interacción entre las partes del software. La Arquitectura de Software tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad. [22] [24]

La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (IEEE[19]). [25]

Una de las definiciones más completas y por la cual se guía el diseño de la arquitectura del sistema es la formulada por la IEEE.

2.3.1 Estilos Arquitectónicos

En la Arquitectura de Software, se observa que en la práctica del diseño y la implementación aparecen ciertas regularidades como respuesta a problemas similares. Al estudiar dichos problemas se pudieron agrupar en tipos específicos de acuerdo a las características de las situaciones. A estos tipos se les llamó estilos, por analogía con el uso del término en arquitectura de edificios.

Un estilo arquitectónico define una familia de sistemas que siguen un mismo patrón estructural, estableciendo un vocabulario de tipos de componentes y conectores y un conjunto de restricciones sobre cómo combinar esos componentes y conectores. [26]

[19] Instituto de Ingenieros Electricistas y Electrónicos (Institute of Electrical and Electronics Engineers).

Entre los estilos arquitectónicos más difundidos y usados se encuentran [27]:

- Sistemas de flujo de datos
  - Secuencial en lote
  - Tubos y filtros
- Sistemas centrados en los datos
  - Arquitecturas basadas en pizarras o repositorios
- Sistemas de llamada y retorno
  - Modelo – Vista – Controlador (MVC[20])
  - Arquitecturas en Capas
  - Arquitecturas Orientadas a Objetos
  - Arquitecturas Basadas en Componentes
- Peer to Peer
  - Procesos comunicativos
  - Arquitecturas Basadas en Eventos
  - Arquitecturas Orientadas a Servicios
  - Arquitecturas Basadas en Recursos
- Máquinas virtuales
  - Intérpretes
  - Sistemas basados en reglas

2.3.2 Arquitectura Cliente-Servidor

Esta arquitectura se encuentra dentro de la clasificación de estilo de llamada y retorno. El cliente y el servidor generalmente están localizados en diferentes sistemas, pero es posible que se encuentren en el mismo sistema. El cliente es el que realiza la petición por un servicio y el servidor provee el servicio correspondiente a la petición. El servicio debe procurar el resultado, el cual es retornado al cliente. [28]

[20] Del Ingles (Model-View-Controller)

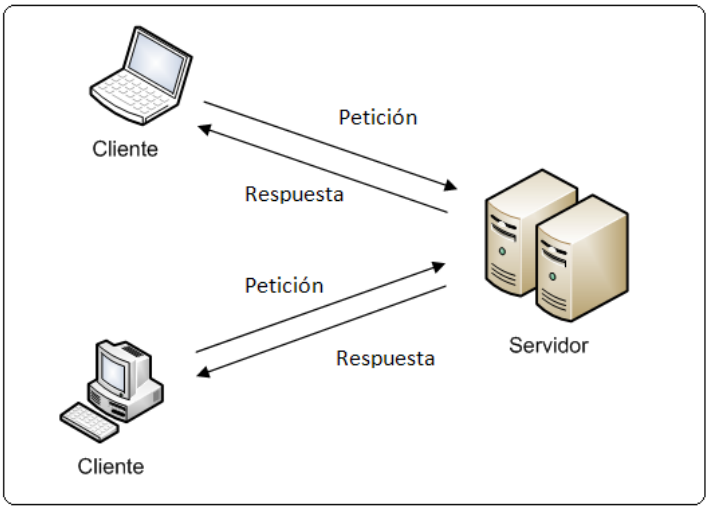


Figura 3. Arquitectura Cliente-Servidor.

Los clientes pueden ser clasificados como clientes ligeros o pesados. Lo clientes pesados típicamente contienen, además de la lógica de presentación, gran parte de la lógica de negocio de la aplicación. Lo clientes ligeros manejan usualmente sólo la lógica de presentación, permitiendo que futuros cambios de negocio en la aplicación no afecten al cliente. [28]

2.3.3 Arquitectura en Capas

La arquitectura en capas es una organización jerárquica, donde cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. El estilo soporta un diseño basado en niveles de abstracción lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos menos complejos o sub-problemas. Esta proporciona amplia reutilización dada la división bien definida de responsabilidades en capas. [28]

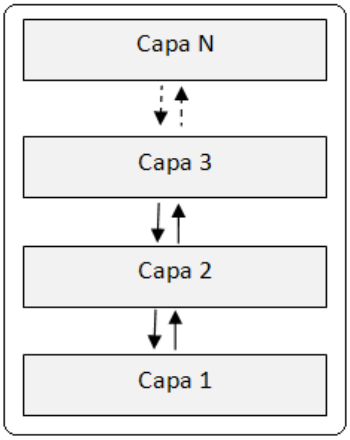


Figura 4. Arquitectura en Capas.

Una especialización de la arquitectura en capas es la arquitectura de tres capas. En la misma se encuentran bien delimitadas las partes funcionales de la aplicación: presentación, negocio y datos.

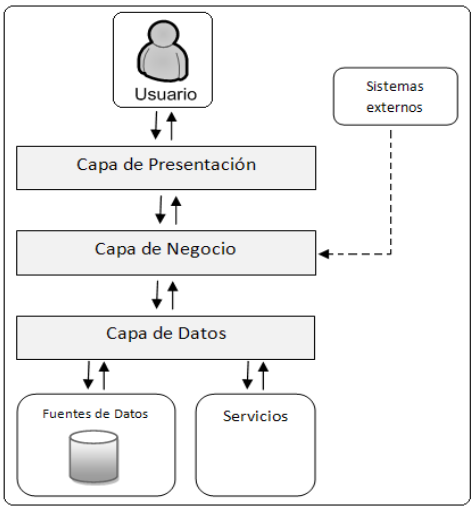


Figura 5. Arquitectura tres Capas.

Capa de presentación: en esta se encuentran los servicios orientados al usuario, lo que les permite la interacción con el sistema. Esta capa contiene los elementos que garantizan la interacción del usuario con el negocio del sistema. [29]

Capa de negocio: esta capa implementa las funcionalidades básicas del sistema, y encapsulan la lógica del negocio. Generalmente en esta se tratan los componentes situados en la capa de negocio, que se podrían exponer a las interfaces de servicio para ser usadas por otros usuarios[21]. [29]

Capa de datos: esta capa permite el acceso a los datos que se alojan dentro de los límites del sistema, y los datos expuestos por otros sistemas, a los cuales tal vez pueda accederse a través de servicios. La capa de datos expone los datos a la capa de negocio a través de interfaces genéricas diseñado para ser práctico para el uso de los servicios empresariales. [29]

2.3.4 Modelo – Vista – Controlador

Modelo – Vista – Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

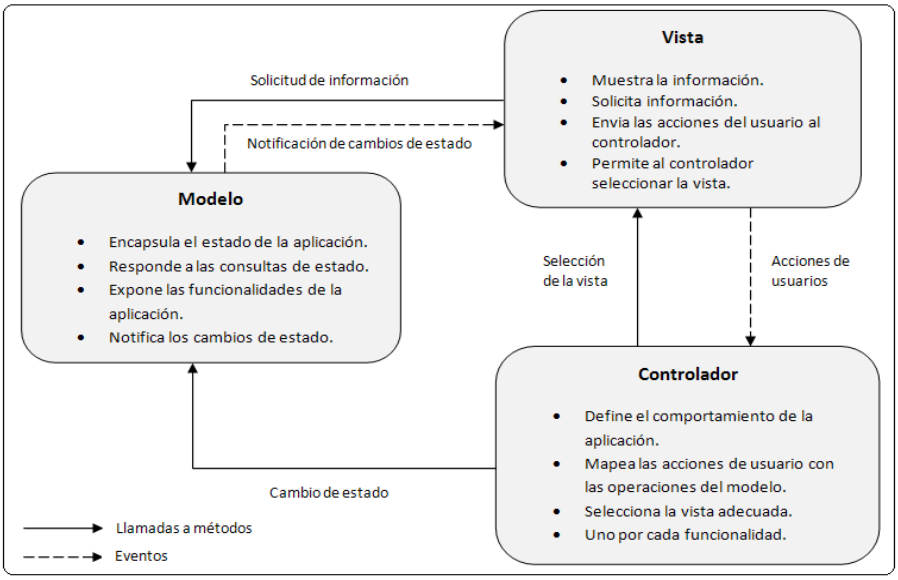


Figura 6. Modelo-Vista-Controlador.

El modelo representa los datos y las reglas de negocio que rigen su acceso y actualización; puede verse como una modelación de los procesos del mundo real [30].

[21] En este caso se refiere a sistemas externos.

Las vistas se encargan de presentar los datos obtenidos del modelo. Es responsabilidad de las vistas mantener la información actualizada, esto se puede logar a través de peticiones de actualización al modelo o a través de notificaciones de cambio que el modelo emite (eventos). [30]

El controlador actúa como un traductor de las acciones que se realizan en las vistas en las operaciones que ocurren en el modelo. Las acciones realizadas por el modelo desencadenan la activación de procesos de negocio o cambian el estado del modelo. Sobre la base de las acciones del usuario y los resultados del modelo, el controlador responde mediante la selección de la vista apropiada. [30]

2.3.5 Diseño de la Arquitectura del Sistema

El sistema para la administración de los contenidos de la plataforma COMCEL es una aplicación Web; en la misma todas las reglas fundamentales del negocio se encuentran en el lado servidor, dejando en el lado cliente las reglas de presentación y validación de la información. Con este diseño proponemos que los usuarios accedan a través de un navegador web a las funcionalidades de un sistema centralizado.

Se propone un diseño por capas, encapsulando y delimitando las responsabilidades de cada una de estas en el sistema. Delimitando el número de capas a tres: presentación, negocio y datos. En la siguiente figura se representa la arquitectura del sistema.

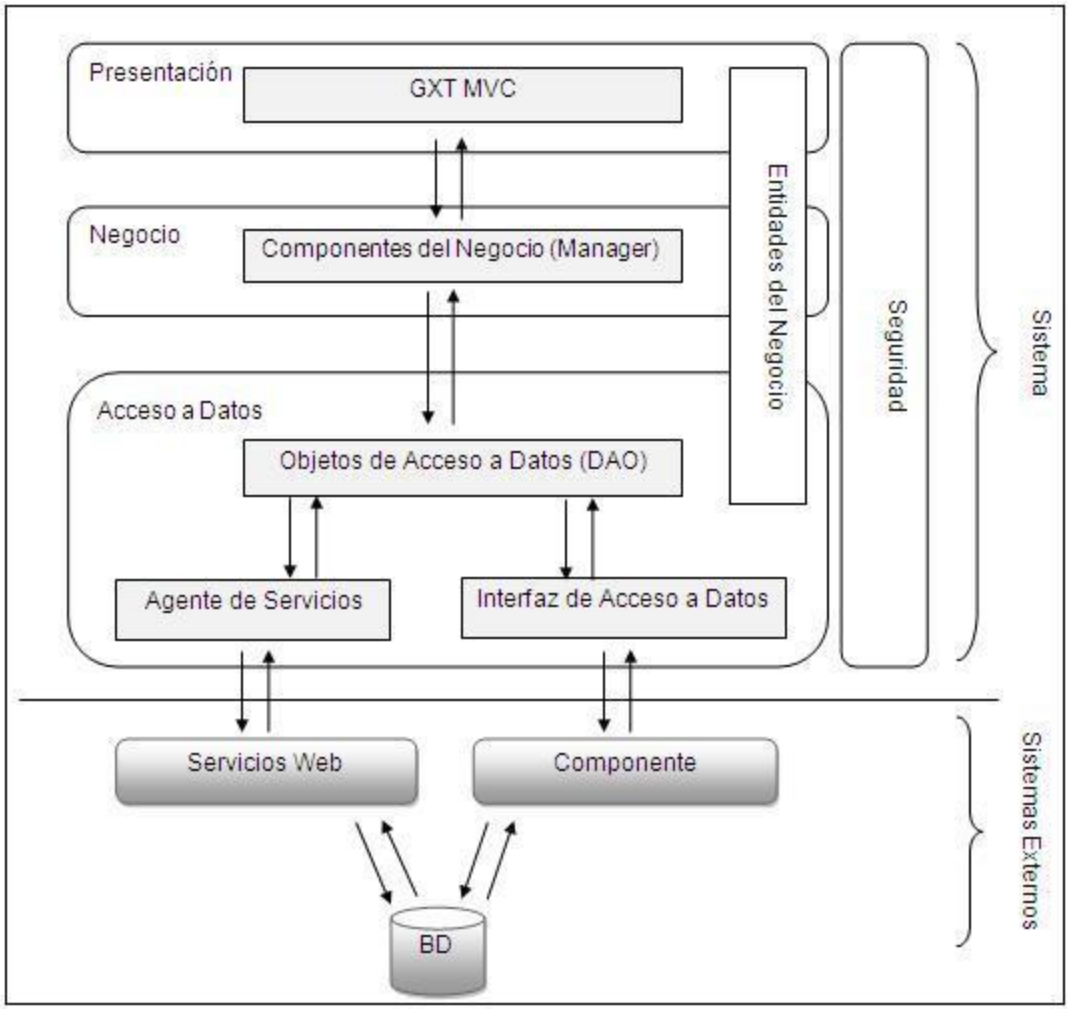


Figura 7. Arquitectura del Sistema.

Capa de Presentación: Es la encargada de interactuar con el usuario, en la misma se encuentran los componentes de presentación (widgets), vistas y controladores. La capa de presentación se implementa usando GXT[22], el cual constituye un marco de trabajo para la creación de componentes visuales y la manipulación de los eventos. La capa de presentación interactúa con la capa de negocio a través de la arquitectura que propone GXT.

[22] Extends Google Web Toolkit Frameworks (GXT).

El framework GXT implementa el patrón Modelo-Vista-Controlador, en el cual la Vista dispara eventos que son atendidos por los controladores que tengan registrado dicho evento. Cuando llega el evento al controlador este solicita la información necesaria al modelo, devolviendo la respuesta a la vista que disparó inicialmente el evento. La vista recibe la información del modelo y se actualiza.

Capa de Negocio: En la capa de negocio del sistema se encuentran los managers que contienen las reglas o lógica de la aplicación. Los managers interactúan con los objetos de acceso a datos para consultar, insertar, modificar o eliminar información. Esta capa responde a las solicitudes que realiza la capa de presentación.

Capa de Acceso a Datos: Se encuentran los objetos de acceso a datos, DAO, que se encargan de intercambiar información con la capa de negocio. Un DAO encapsula la lógica del acceso a datos separando la petición de los mismos, de la lógica de negocio. El Agente de Servicios garantiza la comunicación del sistema con los servicios web que brinda COMCEL, mientras que la Interfaz de Acceso a Datos garantiza la comunicación con componentes encargados de interactuar con la base de datos. Para consumir los servicios que brinda la plataforma se utiliza Spring Web Service[23].

Entidades del Negocio o Dominio: Son las entidades que modelan el problema, es la abstracción de la situación real. Se representan de forma vertical ya que todas las capas interactúan con ellas, las consultan, las crean, las modifican y las eliminan.

Seguridad: Están presentes las funcionalidades que garantizan el nivel de acceso de los usuarios al sistema, verificando en cada petición que el usuario tenga permiso para ejecutarla. Para implementar la seguridad se utiliza Spring Security.

[23] Ver el epígrafe: 1.6 Spring Framework.

2.4 Patrones de Diseño

Christopher Alexander en 1979 dijo “cada patrón describe un problema que ocurre infinidad de veces en nuestro entorno, así como la solución al mismo, de tal modo que podemos utilizar esta solución un millón de veces más adelante sin tener que volver a pensarla otra vez”. [31]

Un patrón de diseño aborda problemas recurrentes del diseño orientado a objetos. En él se describe el problema, la solución, el momento de aplicar la solución, y sus consecuencias. La solución es un conjunto de clases, objetos y relaciones que se establecen entre ellos para resolver el problema; personalizada y aplicada para resolver el problema en un contexto particular. [31]

2.4.1 Patrones GRASP[24]

Una de las actividades más complicadas en Orientación de Objetos consiste en elegir las clases adecuadas y decidir como estas clases deben interactuar. Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento.

Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada.

Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar la clase responsable de crear una nueva instancia de determinada clase.

Controlador: Un Controlador es un objeto que se encarga de manejar un evento del sistema. Este no realiza mucho trabajo por sí mismo, sino que delega en otros objetos, coordina y controla el trabajo que se necesita hacer.

[24] Acrónimo de General Responsability Asignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades).

Bajo Acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y recurre a ellas. El Bajo Acoplamiento soporta un diseño de clases más independientes, que reducen el impacto de cambios, y permite que sean más reutilizables.

Alta Cohesión: La cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos, y auto-identificable. Una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo.

2.4.2 Patrones GoF[25]

Singleton (Instancia Única): Está diseñado para restringir la creación de objetos pertenecientes a una clase a un único objeto. Asegura que una clase tiene una sola instancia y proporciona un punto de acceso global a ella.

Factory Method (Método de Fabricación): Define una interfaz para crear un objeto dejando a las subclases decidir al tipo específico al que pertenecen.

Algunas de las ventajas de este patrón son:

- Centralización de la creación de objetos.
- Facilita la escalabilidad del sistema.
- El usuario se abstrae de la instancia a crear.

Facade (Fachada): Simplifica los accesos a un conjunto de clases relacionadas proporcionando una única clase que todos utilizan para comunicarse con dicho conjunto de clases.

[25] Gang of Four (Banda de los cuatro). Patrones de diseño orientado a objetos.

2.5 Modelo de Diseño

El modelo de diseño es una realización del diseño del sistema. Es un modelo físico y concreto. Se centra en cómo los requisitos funcionales y no funcionales tienen impacto en el sistema a desarrollar. Dentro de sus propósitos están: crear una entrada apropiada y un punto de partida para la implementación, descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo. Adquirir comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnología de distribución y concurrencia. [22]

2.5.1 Diagrama de Clases del Diseño

Un diagrama de Clases del Diseño es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Una clase es una descripción de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica. [22] [32]

El diagrama general de las clases del diseño representa la interacción entre las clases de las diferentes capas que integran el sistema. Las clases View, Controller, Dispatcher y WebServiceGatewaySupport constituyen las clases que proporcionan los frameworks utilizados y se representan para mayor comprensión del diagrama.

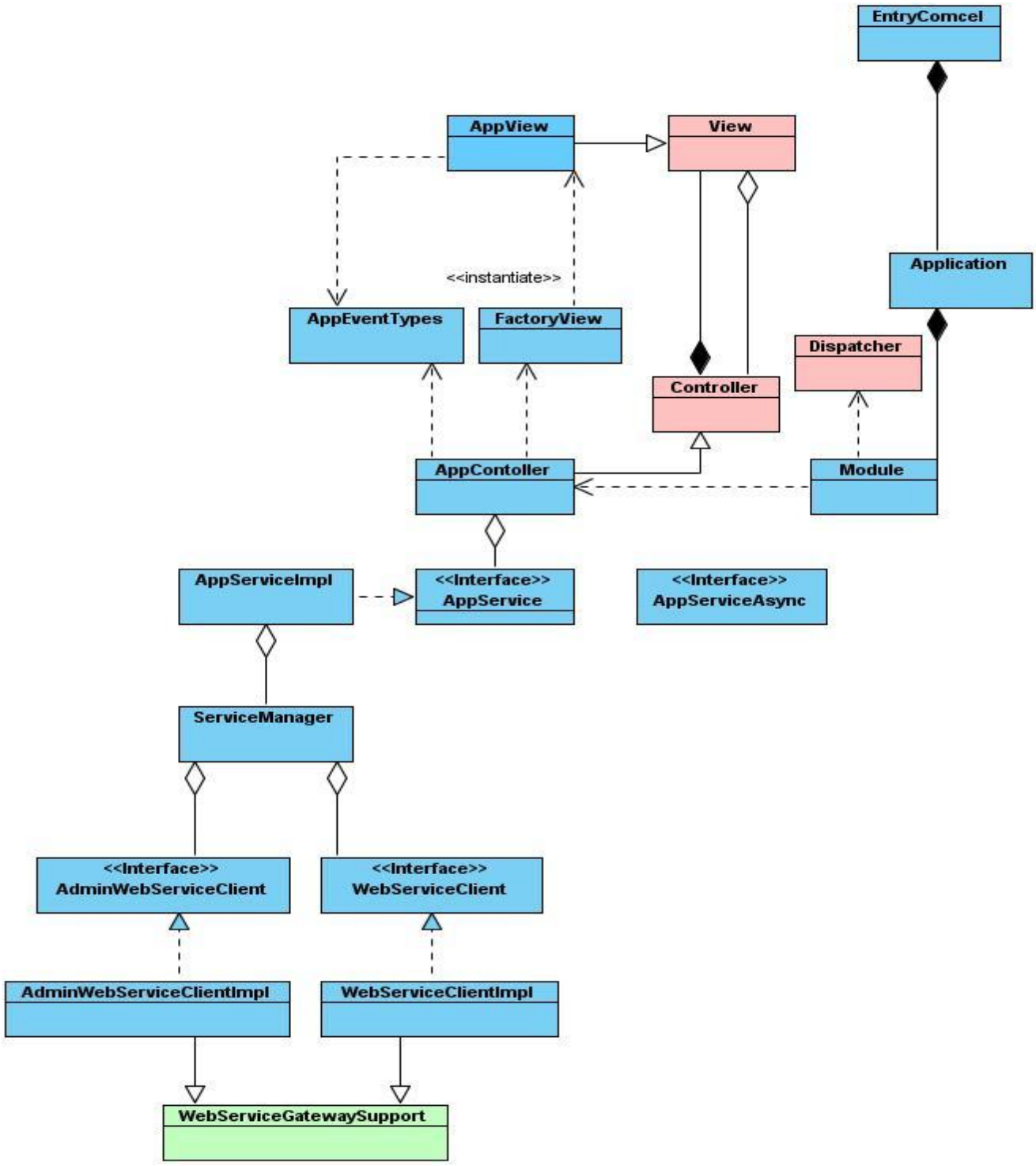


Figura 8. Diagrama General de Clases del Diseño.

El sistema está estructurado por módulos los que agrupan funcionalidades comunes. Cada módulo se puede traducir en términos de clases y sus relaciones. A continuación se muestran los diagramas de clases pertenecientes a cada uno de los módulos del sistema.

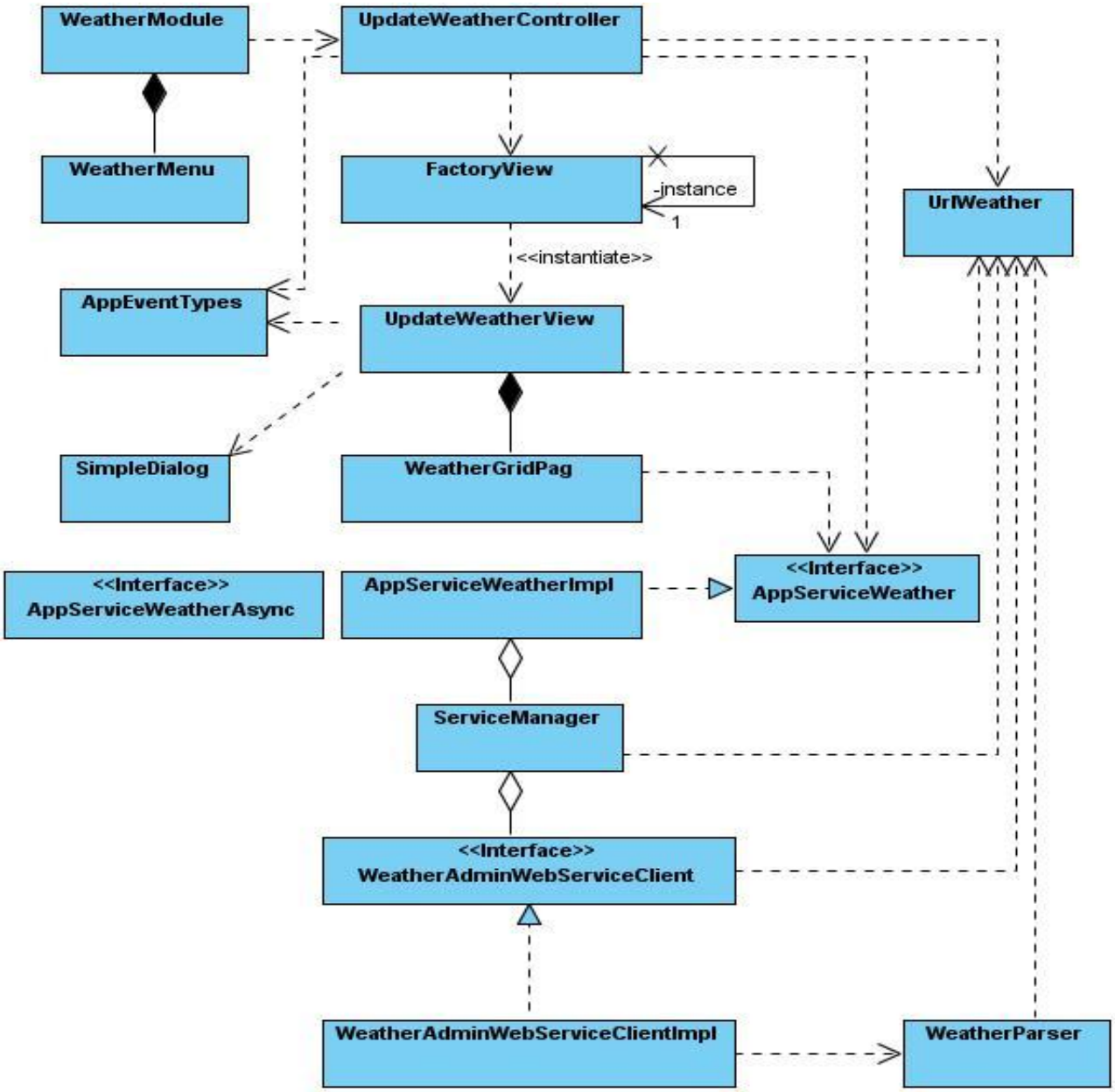


Figura 9. Diagrama de Clases Gestionar Estado del Tiempo.

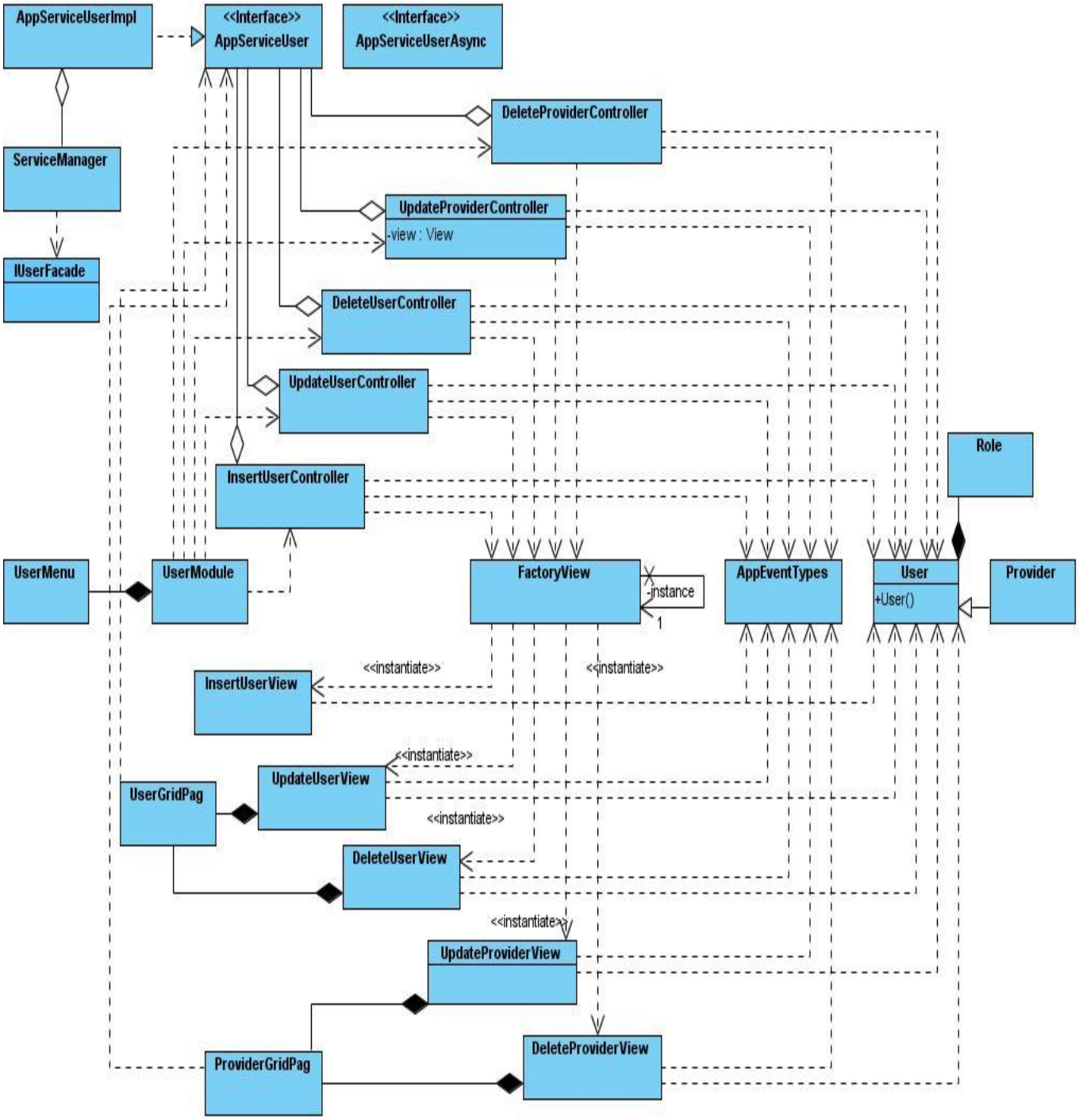


Figura 10. Diagrama de Clases Gestionar Usuario.

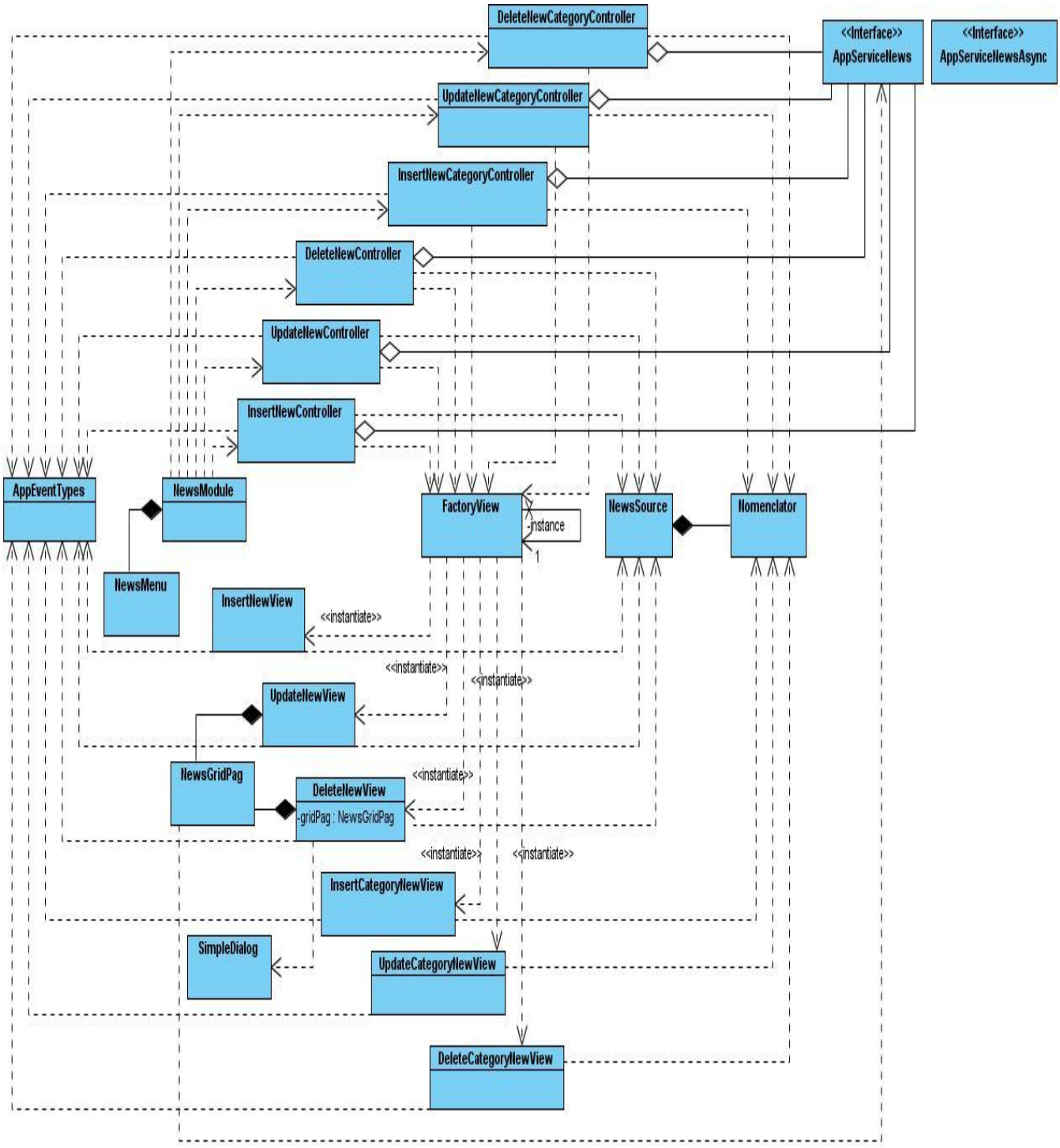


Figura 11. Diagrama de Clases Gestionar Noticia (Parte 1).

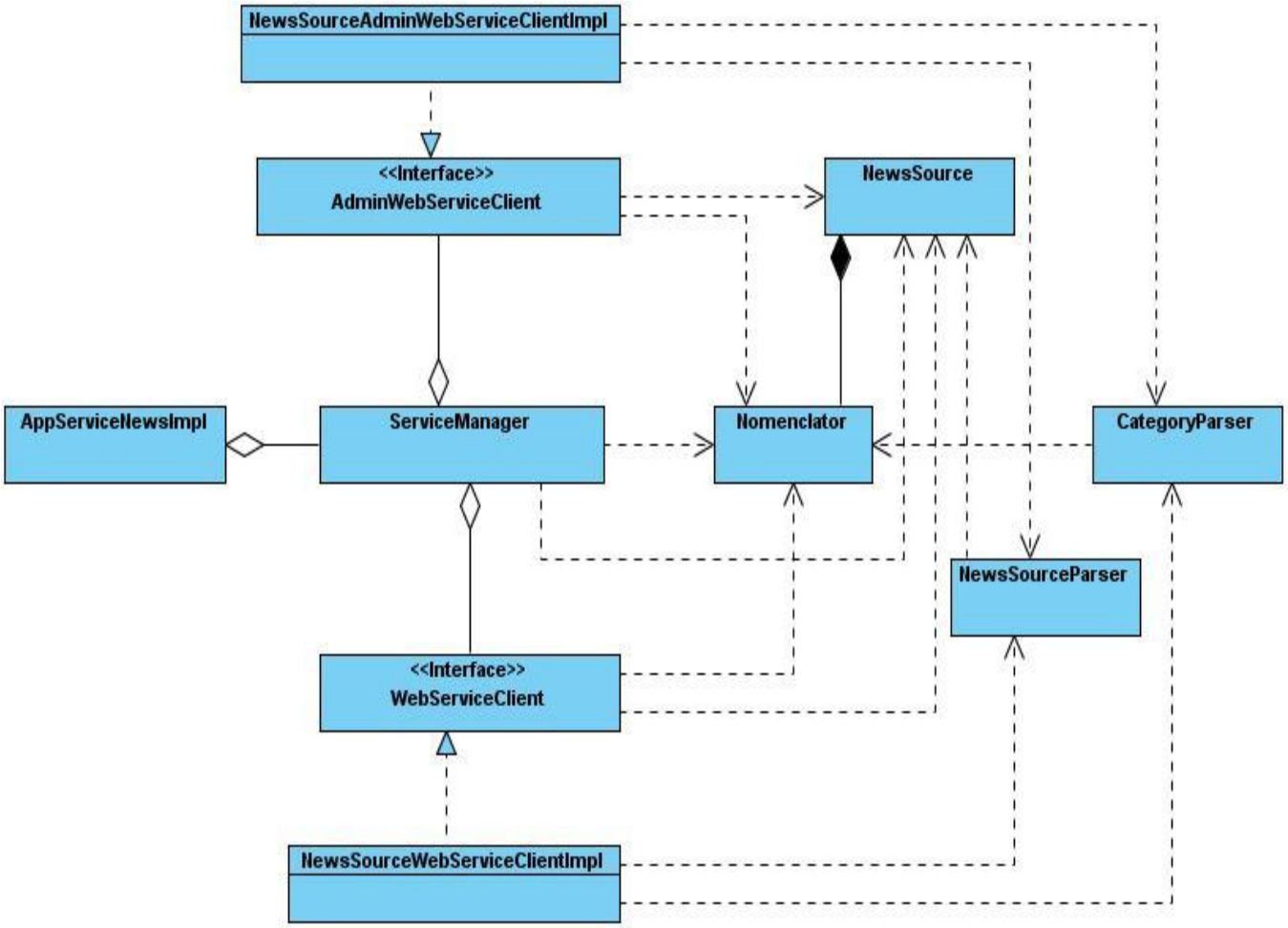


Figura 12. Diagrama de Clases Gestionar Noticia (Parte 2).

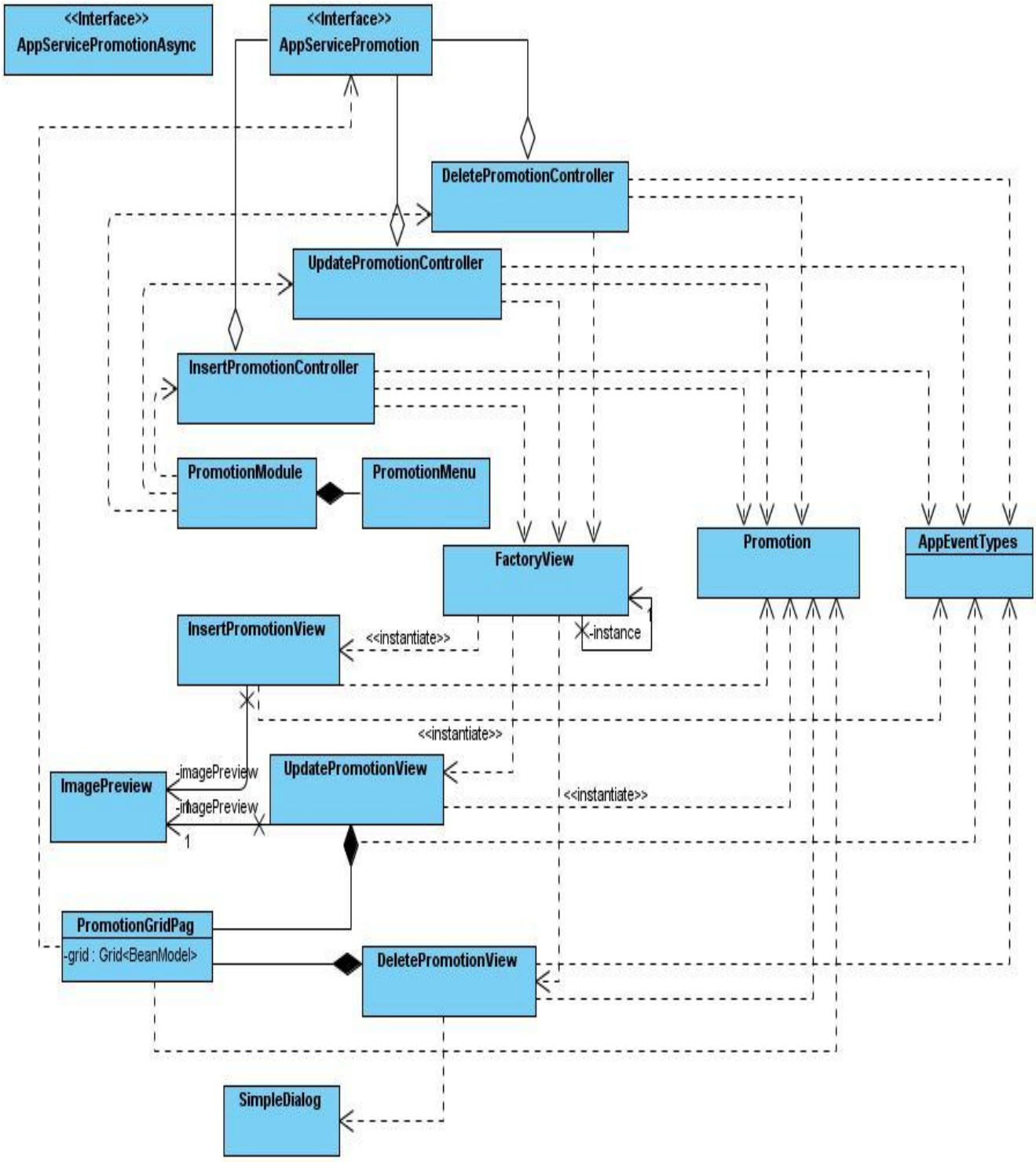


Figura 13. Diagrama de Clases Gestionar Promoción (Parte 1).

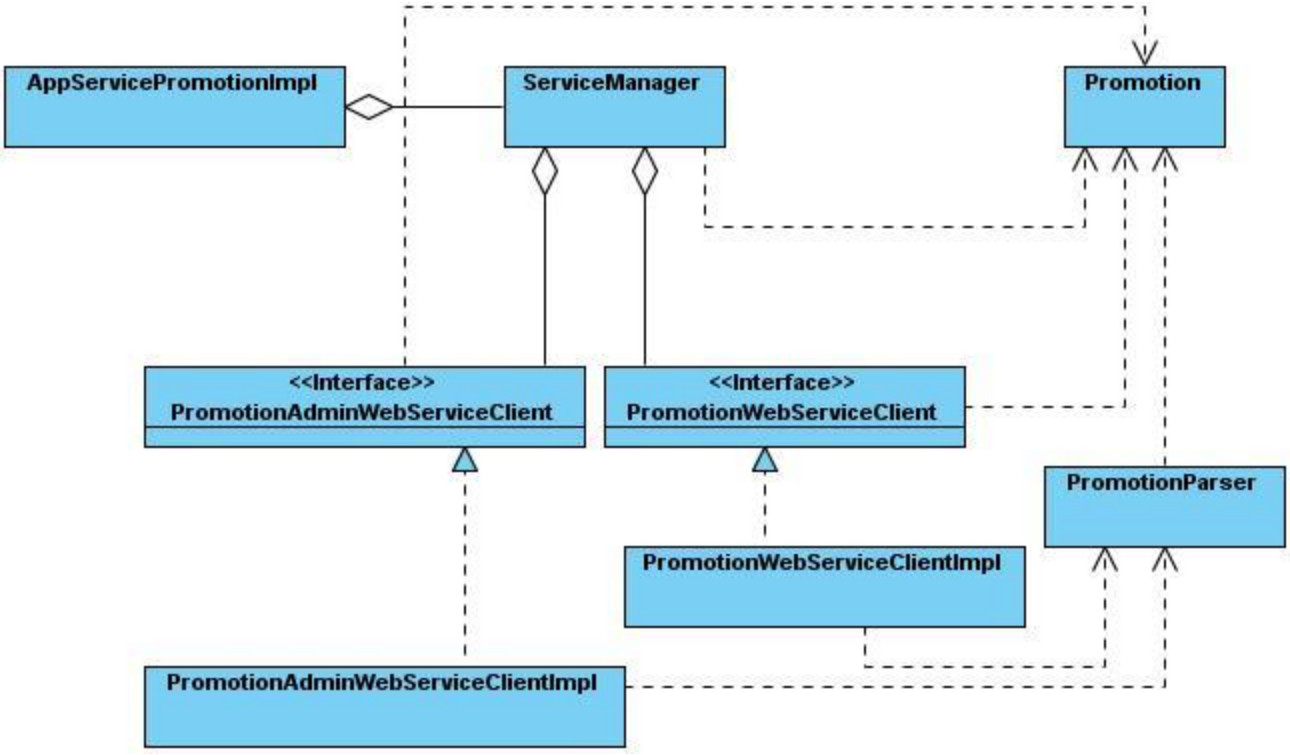


Figura 14. Diagrama de Clases Gestionar Promoción (Parte 2).

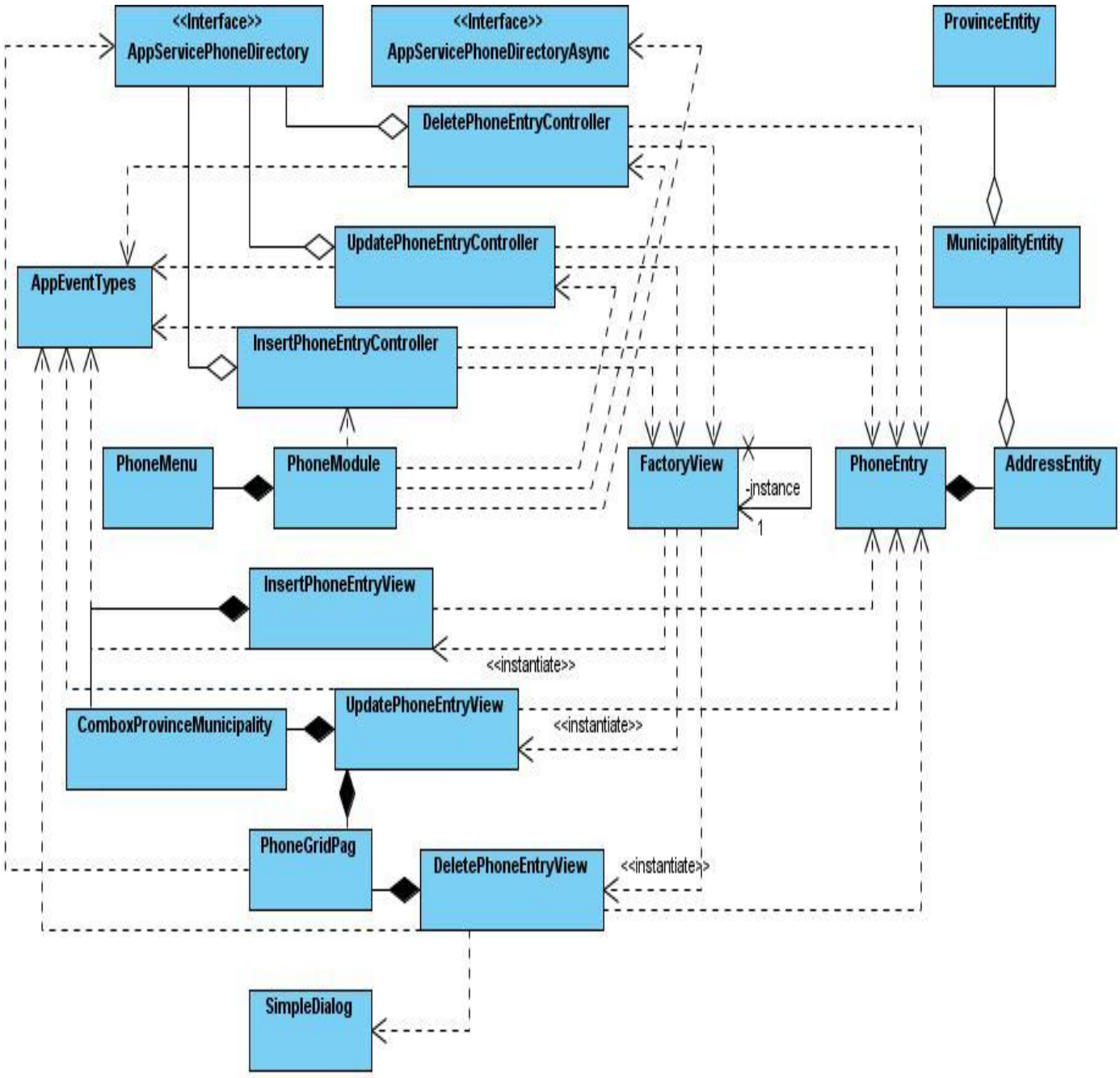


Figura 15. Diagrama de Clases Gestionar Directorio de Páginas Amarillas (Parte 1).

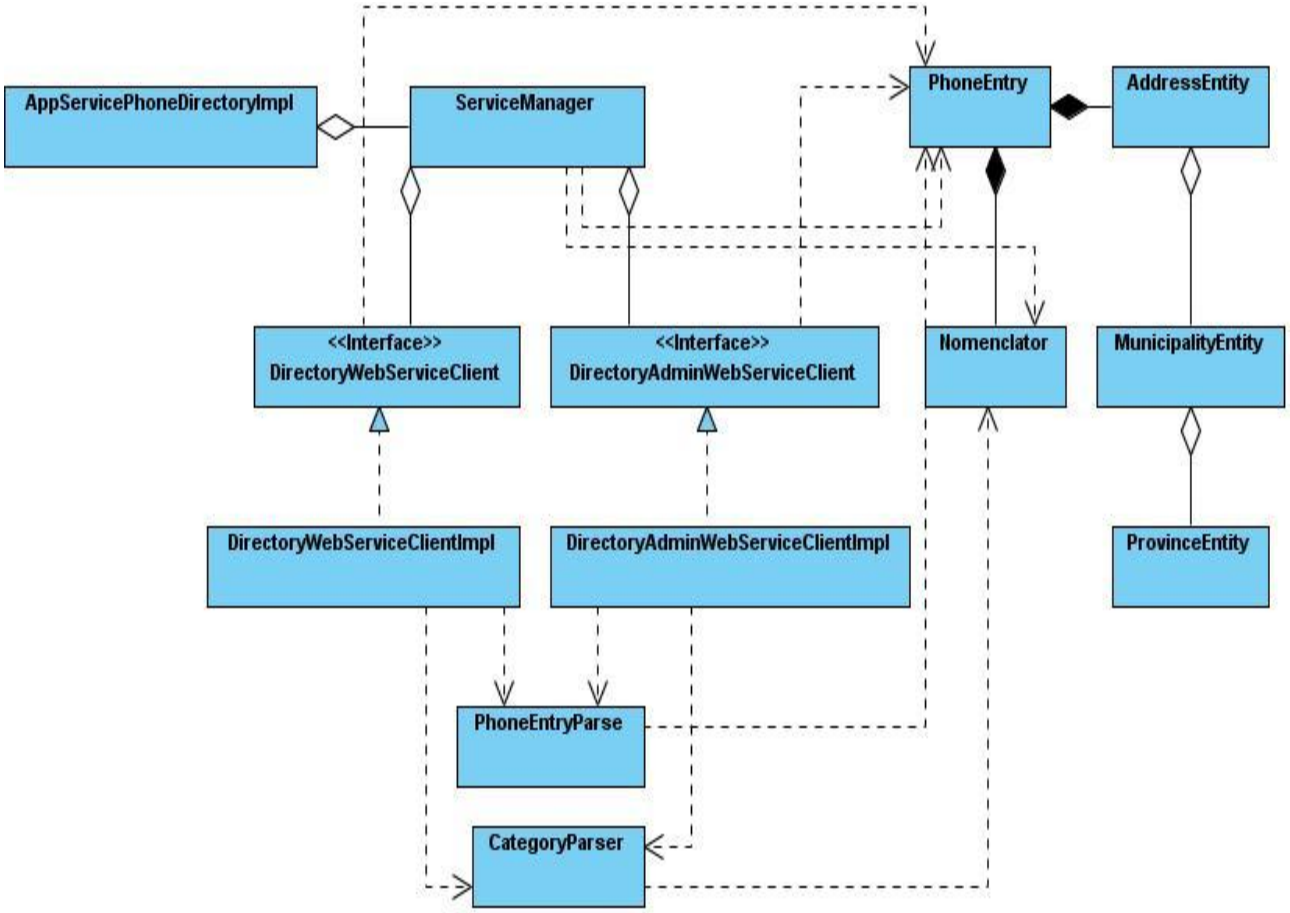


Figura 16. Diagrama de Clases Gestionar Directorio de Páginas Amarillas (Parte 2).

2.5.2 Diagramas de Interacción

Un diagrama de interacción consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos, se utilizan para modelar los aspectos dinámicos de un sistema. Los diagramas de secuencia destacan el orden temporal de los mensajes. Los diagramas de colaboración destacan la organización estructural de los objetivos que envían y reciben mensajes. Ambos diagramas (secuencia y colaboración) son semánticamente equivalentes. Se puede pasar de uno a otro sin pérdida de información. [22] [33]

Los diagramas de secuencia proporcionan una forma de ver el escenario en un orden temporal (¿qué sucede primero?, ¿qué sucede después?). Los clientes entienden fácilmente este tipo de diagramas, por lo que resultan útiles.

En el **Anexo** se muestra un ejemplo de los Diagramas de secuencias realizados para este trabajo.

2.5.3 Diagrama de Paquetes

Un paquete de Diseño es una colección de clases, relaciones, diagramas y otros paquetes que estén de alguna forma relacionados. Es utilizado para estructurar el modelo de diseño dividiéndolo en partes más pequeñas.

Los paquetes de diseño deben utilizarse fundamentalmente como herramienta organizacional del modelo para agrupar elementos relacionados. Pueden ser usados en cualquier nivel, desde el nivel más alto, donde son usados para dividir el sistema en dominios, hasta el nivel más bajo, donde son usados para agrupar casos de uso individuales, clases, o componentes. Para el diseño del sistema las clases se agruparon por paquetes según su funcionalidad y las entidades que gestionan.

A continuación se muestra el diagrama de paquetes del sistema.

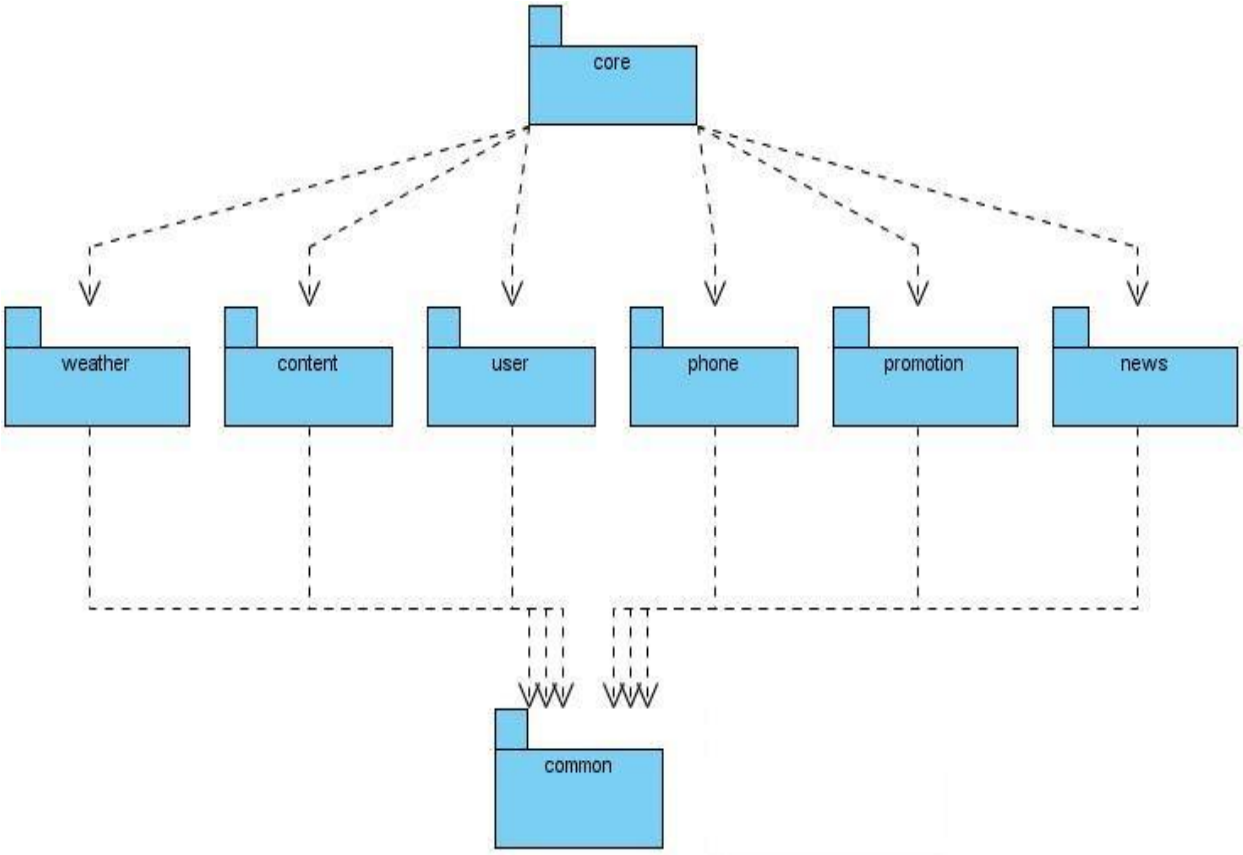


Figura 17. Diagrama de Paquetes del Sistema.

Esta división se hace para comprender con más exactitud, de manera macro, cada uno de los elementos que contiene el diseño del sistema. En cada paquete se agrupan funcionalidades comunes, ejemplo, en el paquete ***content*** se encuentran todas las clases necesarias para la gestión del contenido y solo él se encarga de realizar dicha función. El paquete ***common*** contiene clases que son utilizadas en más de un módulo y el paquete ***core*** se encarga de iniciar todas las funcionalidades e integrar los módulos del sistema.

2.6 Conclusiones del Capítulo

En este capítulo se resumieron los requisitos funcionales y las cualidades que el producto debe tener, recogidas en los requisitos no funcionales. Se definió una arquitectura base flexible y adaptada a las funcionalidades del sistema. Se estudiaron y aplicaron patrones de diseño para asignar responsabilidades a las clases, necesarios para la implementación. Además, se ilustraron los diagramas de clases del sistema, se describió la interacción entre objetos a través de los diagrama de secuencias y se expuso el diagrama de paquetes.

Capítulo 3: Implementación del Sistema

3.1 Introducción

A partir de los resultados obtenidos del Diseño, en el presente capítulo se realiza el modelo de implementación. El propósito es implementar las clases y objetos en forma de componentes y exponer la distribución del sistema en nodos y los protocolos de comunicación entre cada uno de ellos.

3.2 Modelo de Implementación

El modelo de implementación describe como los elementos del diseño, como las clases, se implementan en términos de componentes. Contiene los diagramas de componentes y el diagrama de despliegue que comenzó a desarrollarse en el flujo de trabajo de análisis y diseño y que se perfecciona en implementación. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros. [22]

3.2.1 Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente. El diagrama de componentes estructura el modelo de implementación en términos de subsistemas de implementación y muestra las relaciones entre los elementos del modelo.

En la figura que se muestra a continuación se representa la distribución de los componentes de la aplicación, a partir del WAR[26] generado, que es el componente principal del sistema.

[26] Archivo Web (Web Archive)

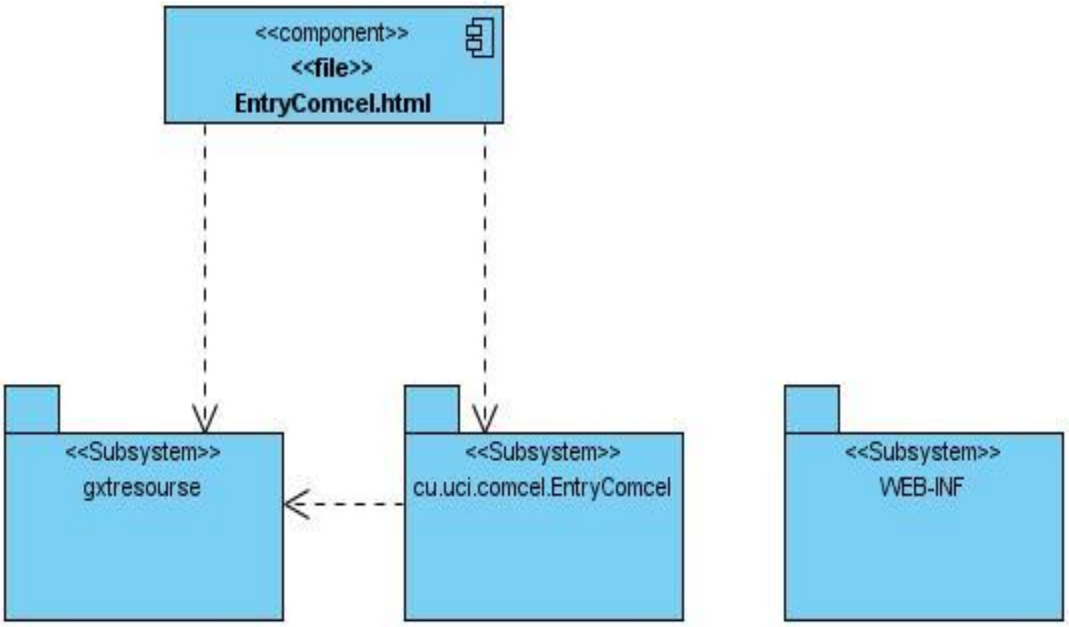


Figura 18. Diagrama de Componentes.

3.2.2 Descripción de los Componentes

a) Componente EntryComcel.html

Propósito

Componente mediante el cual se accede a la aplicación y en el que se renderizan los componentes solicitados en cada petición.

b) Subsistema gxtresourse

Propósito

Subsistema que agrupa a los elementos de estilos (CSS), imágenes y planillas predeterminados que usa GXT para la construcción de los componentes visuales.

c) Subsistema cu.uci.comcel.EntryComcel

Propósito

Subsistema que contiene los ficheros java scripts y otros que son usados para construir los componentes visuales y las funcionalidades de los mismos.

d) Subsistema WEB-INF

Propósito

Contenedor físico de los ficheros que conforman el núcleo de la aplicación.

Contenido

- Subsitema clases
- Sibsistema lib
- configuration.xml
- web.xml

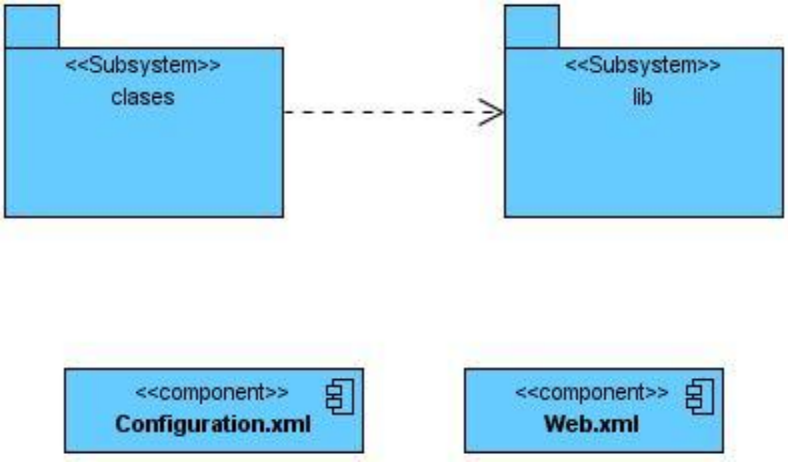


Figura 19. Diagrama de Componentes del Subsistema WEB-INF.

d1) Subsistema clases

Propósito

Contenedor físico del compilado de cada una de las clases de la aplicación, archivos con extensión .class, .xml.

d2) Subsistema lib

Propósito

Contenedor físico de librerías que se utilizan en el sistema.

d3) Componente configuration.xml

Propósito

Componente que guarda la configuración de los xml de ficheros contexto del sistema.

d4) Componente web.xml

Propósito

Descriptor de despliegue para la aplicación Web.

3.2.3 Diagrama de Despliegue

Un diagrama de despliegue modela la topología del hardware sobre el que se ejecuta un sistema. Muestra la configuración de los nodos que participan en la ejecución y de los componentes que residen en ellos. Un nodo es un elemento físico que existe en tiempo de ejecución. Representa un recurso computacional que, por lo general, tiene memoria y capacidad de almacenamiento. Además, representa el despliegue físico de un componente. [22]

A continuación se muestra el diagrama de despliegue del sistema:

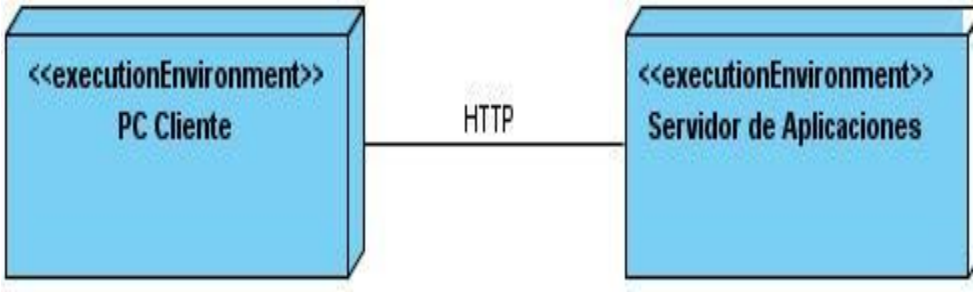


Figura 20. Diagrama de Despliegue.

La computadora cliente mediante un navegador Web se conecta a través del protocolo http al servidor de aplicaciones, donde se encuentra desplegado el WAR de la aplicación.

3.3 Conclusiones del Capítulo

Los diagramas de componentes descritos en este capítulo, representan en términos de componentes y subsistemas de implementación, los elementos del modelo de diseño obtenidos en el desarrollo del capítulo anterior, que fueron implementados para dar cumplimiento al objetivo general. Además, se describió la distribución del sistema en nodos, especificados en el diagrama de despliegue.

Capítulo 4: Pruebas

4.1 Introducción

En este capítulo se diseñan los casos de prueba que permitirán evaluar y valorar la calidad del producto.

4.2 Modelo de Prueba

La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación. [34] [35]

Las pruebas de software consisten en una serie de actividades en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, realizando una evaluación del sistema o del componente probado. Su principal objetivo es evaluar o valorar la calidad del producto. Las pruebas constituyen un elemento crítico para la garantía de la calidad del software. [22] [35]

4.2.1 Métodos de Pruebas

Las pruebas de caja negra también denominada prueba de comportamiento se centran el los requisitos funcionales de software. Permite al ingeniero de software obtener un conjunto de condiciones de entrada que ejerciten completamente todos lo requisitos funcionales de un programa. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores de estructuras de datos o en acceso a bases de datos externas, errores de rendimiento y errores de inicialización y terminación. [22]

4.2.2 Diseño de Casos de Prueba

Los casos de prueba que se utilizarán para las pruebas de la aplicación, están diseñados para verificar los requerimientos del usuario. A continuación se muestran dos de los casos de pruebas realizados a las principales funcionalidades del sistema.

Tabla 10. Pruebas de caja negra del caso de uso "Autenticar Usuario".

Condición de Entrada	Casos válidos	Casos no válidos
Usuario	Cadena de caracteres	Campo vacío
Contraseña	Cadena de caracteres	Campo vacío

Caso de Uso	Autenticar Usuario
Caso de Prueba	Permitir realizar las demás funcionalidades del sistema con el usuario que introduzca correctamente los datos de autenticación.
Entrada	El usuario introduce los datos correctamente de la forma: **Usuario:** proveedor **Pass:** proveedor
Resultado	Se proporciona el acceso a la aplicación al usuario autenticado.
Condiciones	No debe existir ningún campo vacío y los datos deben ser del tipo especificado.

Caso de Uso	Autenticar Usuario
Caso de Prueba	Permitir realizar las demás funcionalidades del sistema con el usuario que introduzca incorrectamente los datos de autenticación.
Entrada	Cualquiera de los campos está vacío o simplemente no son los correctos
Resultado	El sistema muestra un mensaje de error especificando que los datos introducidos son incorrectos.

Condiciones	Deben existir campos vacíos o con datos incorrectos

Tabla 11. Pruebas de caja negra del caso de uso "Gestionar Promoción" escenario "Insertar Promoción".

Condición de Entrada	Casos válidos	Casos no válidos
Título	Cadena de caracteres	Dejar vacío el campo título
Tiempo de Duración	Número entero	Dejar vacío el campo.
Fecha de Inicio	Seleccionar	Dejar vacío el campo.
Texto	Cadena de caracteres	Dejar vacío el campo.
Imagen	Byte	No cargar imagen.

Caso de Uso	Gestionar Promoción escenario Insertar Promoción
Caso de Prueba	Permitir insertar una nueva promoción en la base de datos introduciendo correctamente los datos.
Entrada	El usuario introduce los datos correctamente de la forma: **Título:** Rebaja de tarifas telefónicas. **Tiempo de Duración :** 30 **Fecha de Inicio:** 27/05/2010 **Texto:** Nueva promoción, económicas tarifas telefónicas. **Imagen:** Se seleccionó una imagen.
Resultado	El sistema introduce una nueva promoción en la base de datos.
Condiciones	Los datos introducidos son correctos y no existen campos vacios.

Caso de Uso	Gestionar Promoción escenario Insertar Promoción
Caso de Prueba	Permitir insertar una nueva promoción en la base de datos introduciendo incorrectamente los datos.
Entrada	El usuario deja campos en blanco, o los introduce incorrectamente.
Resultado	El sistema muestra un mensaje de error especificando que los datos introducidos son incorrectos o debe llenarlos todos.
Condiciones	Existen campos vacíos.

Tabla 12. Pruebas de caja negra del caso de uso "Gestionar Promoción" escenario "Modificar Promoción".

Condición de Entrada	Casos válidos	Casos no válidos
Título	Cadena de caracteres.	Dejar vacío el campo.
Tiempo de Duración	Número entero.	Dejar vacío el campo.
Fecha de Inicio	Seleccionar.	Dejar vacío el campo.
Texto	Cadena de caracteres.	Dejar vacío el campo.
Imagen	Byte.	No cargar imagen.

Caso de Uso	Gestionar Promoción escenario Modificar Promoción
Caso de Prueba	Modificar una promoción existente introduciendo correctamente sus datos.
Entrada	El usuario selecciona la promoción haciendo clic en la tabla que se muestra en la interfaz, luego modifica el campo que desee,

	para la promoción seleccionada.
Resultado	El sistema modifica la promoción en la base de datos.
Condiciones	Los datos introducidos son correctos y no deben existir campos en blanco.

Caso de Uso	Gestionar Promoción escenario Modificar Promoción
Caso de Prueba	Modificar una promoción introduciendo incorrectamente los datos o dejando campos vacíos.
Entrada	El usuario deja alguno de los campos requeridos en blanco.
Resultado	El sistema muestra un mensaje especificando que no se permiten campos en blanco.
Condiciones	Algún campo vacío.

Tabla 13. Pruebas de caja negra del caso de uso "Gestionar Promoción" escenario "Eliminar Promoción".

Caso de Uso	Gestionar Promoción escenario Eliminar Promoción
Caso de Prueba	Eliminar una promoción seleccionando alguna de las existentes.
Entrada	El usuario selecciona la promoción de la tabla que se muestra en la interfaz.
Resultado	El sistema elimina la promoción de la base de datos.
Condiciones	Seleccionar una promoción y confirmar que desea eliminarla.

Caso de Uso	Gestionar Promoción escenario Eliminar Promoción
Caso de Prueba	Eliminar una promoción existente sin seleccionar alguna.
Entrada	El usuario no selecciona la promoción de la tabla que se muestra en la interfaz.
Resultado	El sistema muestra un mensaje especificando que debe seleccionar la promoción que desea eliminar.
Condiciones	No seleccionar la promoción, o no confirmar que se desea eliminar.

4.3 Conclusiones del Capítulo

En este capítulo se puede apreciar el desarrollo de los casos de pruebas realizados a algunas funcionalidades de la aplicación, mostrando que dichas funcionalidades no presentan anomalías en su funcionamiento ya que realizan un correcto tratamiento de los datos que manejan. Por lo que se concluye que el resultado de las pruebas realizadas ha sido satisfactorio.

Conclusiones Generales

En este trabajo se realizó el estudio de las herramientas y tecnologías de desarrollo que se utilizaron en la confección de la solución. Finalmente se escogieron las que resultaron más adecuadas para el desarrollo del sistema. Se desarrolló la aplicación Web sobre la plataforma J2EE, específicamente con las tecnologías GWT y Spring frameworks, con los sub proyectos Spring Web Service y Spring Security.

Se utilizó la metodología RUP, se concibió la arquitectura del sistema y se generaron los demás artefactos pertenecientes al diseño. Durante el flujo de trabajo de implementación se realizaron los diagramas de componentes y de despliegue, lo que facilitó la comprensión de la comunicación entre los componentes y la distribución física de estos para la implementación del sistema. Además, se diseñaron casos de prueba para evaluar y valorar la calidad del sistema, logrando así una revisión final de las especificaciones del diseño y de la codificación.

Como resultado de este trabajo se desarrolló una interfaz de usuario para administrar los servicios disponibles en la plataforma COMCEL que cumple con los requisitos establecidos inicialmente. Por todo lo anterior expuesto se concluye que los objetivos trazados fueron cumplidos.

Recomendaciones

Una vez concluido el desarrollo del módulo de administración de la plataforma COMCEL, cumplidos los objetivos trazados y teniendo en cuenta que es una primera versión del sistema, se recomienda:

- Refinar las funcionalidades ya implementadas.
- Realizar mejoras a la apariencia visual del sistema.
- Someter el sistema a pruebas de calidad.

Referencias Bibliográficas

1. Las compañías móviles más grandes del mundo. 2010. [Consultada el: 20/03/2010]. Disponible en: http://www.celularis.com/mercado/las-companias-moviles-mas-grandes-del-mundo.php
2. Mobile Web & Content Delivery Platform. *BeeWeb technologies.* [Consultado el: 02/12/2010] http://www.beeweeb.com/mwt/index.php/products/mobile-web-content-delivery-platform/
3. Volantis Content Delivery Platform. [Consultado el: 13/01/2010]. Disponible en: http://www.volantis.com/content-delivery-platform
4. Guerrillero, prensa de Pinar del Río versión digital. [Consultado el: 13/04/2010]. Disponible en: http://www.guerrillero.cu/index.php?option=com_content&view=article&id=4265:rebajaran-tarifas-para-llamadas-de-telefonia-movil-en-cuba&catid=35:cuba&Itemid=55
5. En dos años se triplicaron las líneas de celular en Cuba. [Consultado el: 13/04/2010]. Disponible en: http://www.cubadebate.cu/noticias/2010/04/23/en-dos-anos-se-triplicaron-las-lineas-de-celular-en-cuba/
6. Sitio Oficial. [Consultado el: 13/04/2010]. Disponible en: http://www.entumovil.cu/
7. Interfaz de Usuario [Consultada el: 02/12/2009]. Disponible en: http://www.fismat.umich.mx/~crivera/tesis/node6.html
8. Luciano Moreno. Qué son las interfaces, las interfaces web y las interfaces gráficas.2005. [Consultado el: 02/12/2009]. Disponible en: http://www.desarrolloweb.com/articulos/2171.php
9. Pirámide del Diseño Web [Consultada el: 02/12/2009] Disponible en: http://www.rena.edu.ve/cuartaEtapa/Informatica/Tema14.html
10. Gerti Kappel et al. Web Engineering: The Discipline of Systematic Development of Web Applications, chapter An Introduction to Web Engineering, pages 1- 22. John Wiley & Sons Ltd., 2006.
11. Christian Van Der Henst S. (2005). ¿Qué es la Web 2.0? [Consultado el: 12/05/2008]. Disponible en: http://www.maestrosdelweb.com/editorial/web2/
12. J2EE Desarrollo de Aplicaciones Web, 2002, Benjmin Aumaille.
13. Investigación de la Plataforma J2EE y su Aplicación Práctica, 2003, JUAN MANUEL BARRIOS UNÑEZ.

14. Emilio Bravo García; El framework Google Web Toolkit. Ingeniería Informática Empresarial. Disponible en : http://www.i2e.com.es
15. Alternativa a Spring, 2009, Pedro Pablo Hernández (Universidad de San Carlos de Guatemala, Facultad de Ingeniería).
16. http://static.springsource.org/spring-ws/sites/1.5/. [Consultado el: 29/01/2010]
17. Luciano; Entornos de Desarrollo Integrados para Java. [Consultado el: 08/02/2010]. Disponible en: http://luauf.com/2008/05/13/entornos-de-desarrollo-integrado-para-java/
18. Cypal Studio [Consultada el: 29/01/2010]. Disponible en: http://www.ibm.com/developerworks/java/library/os-eclipse-ajaxcypal/index.html
19. Plu-ing Cypal. [Consultada el 08/02/2010]. Disponible en: http://www.cypal.in/studio.
20. Barzanallana, Rafael. Universidad de Murcia. Apuntes. Ingeniería del software. Sistemas Informáticos. [Consultado: 2/01/ 2010.]
21. Kruchten, P., The Rational Unified Process: An Introduction, 2000 Addison Wesley
22. Jacaboson, I., Booch, G., Rumbaugh J., El Proceso Unificado de Desarrollo de Software, 2000 Addison Wesley
23. http://www.extremeprogramming.org [Consultada 02/12/2009]
24. Arquitectura de software. Disponible en: http://es.wikipedia.org/wiki/Arquitectura_de_software
25. Arquitectura - IEEE 1471-2000.
26. Garlan, David y Shaw, Mary. An Introduction to Software Architecture. [Consultado el: 10/03/2010]. Disponible en: http://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf
27. Reynoso Carlos Billy. Introducción a la Arquitectura de Software. UNIVERSIDAD DE BUENOS AIRES. 2003.
28. Frank Ernesto Verdecia Rodríguez, Karel Osorio Ramírez. Sistema de Gestion Policial. Especificación de la Arquitectura. 2009.
29. Application Architecture Guide 2.0, J.D. Meier, Alex Homer, David Hill, Jason Taylor, Prashant Bansode, Lonnie Wall, Rob Boucher Jr, Akshay Bogawat, 2008 Microsoft Corporation.
30. Sun Microsystems, Inc. Java BluePrints: Model-View-Controller. [Consultado el: 03/03/2010] Disponible en: http://java.sun.com/blueprints/patterns/MVC-detailed.html
31. Gamma, Erich, y otros. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l.: Addison-Wesley, 1995.

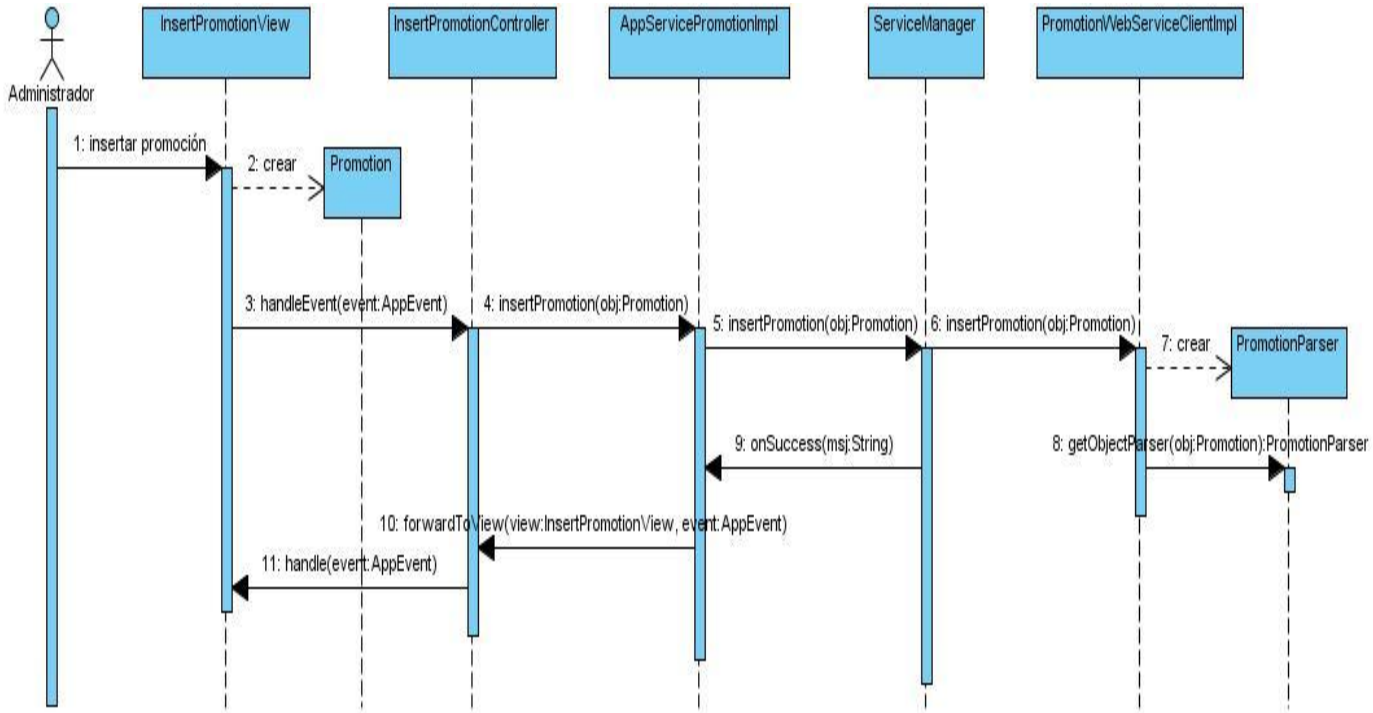
32. Daniel Riesco. UML Diagrama de Clases y de Objetos. [Consultad el: 12/04/2010]. Disponible en : http://sel.unsl.edu.ar/licenciatura/ingsoft2/UML-DiagramaClaseObjeto.pdf
33. Luisa A. Guerrero Universidad de Chile. Departamento de Ciencias de la Computación. UML Diagramas de Interacción.
34. Software, C. d. a. D. d. l. y. G. d. (2005). "Fase de Elaboración. FT Prueba (procedimientos genéricos y aplicación de algunos tipos de pruebas simples)."
35. Pressman Roger S. Ingeniería de Software Un enfoque práctico.p.281 – 294

Bibliografía

- ✓ Hernández León Rolando A., Coello González Sayda. El Paradigma Cuantitativo de la Investigación Científica. EDUNIV 2002.
- ✓ Cooper Robert, Collins Charles. GWT in Practice. Manning Publications Co.2008.
- ✓ Larman Craig. "*UML y Patrones"*. Prentice Hall Hispanoamericana 1999.
- ✓ Jacobson, I.; Booch, G. y Rumbaugh, J.; *"El Proceso Unificado de Desarrollo de software"*, Addison-Wesley, 2000.

Anexo

Anexo #1: Diagrama de Secuencia Insertar Promociones



Glosario de Términos

- ✓ IU: Interfaz de Usuario.
- ✓ XHTML: XHTML Extensible Hypertext Markup Language (Lenguaje Extensible de Marcado de Hipertexto) es una versión más estricta y limpia de HTML, que nace con el objetivo de remplazar a HTML ante su limitación de uso con las cada vez más abundantes herramientas basadas en XML. XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos. Está orientado al uso de un etiquetado correcto.
- ✓ JSP: Java Server Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.
- ✓ EJB: Enterprise JavaBeans, proporcionan un modelo de componentes distribuido estándar del lado del servidor.
- ✓ JDBC: La conectividad de la base de datos de Java (JDBC**,** Java Database Connectivity) es un marco de programación para los desarrolladores de Java, se utiliza comúnmente para conectar un programa del usuario con una base de datos.
- ✓ JTA: Java Transaction API (API para transacciones en Java) establece una serie de Interfaces java entre el manejador de transacciones y las partes involucradas en el sistema de transacciones distribuidas: el servidor de aplicaciones, el manejador de recursos y las aplicaciones transaccionales.
- ✓ JNDI: La Interfaz de Nombrado y Directorio Java (JNDI) es una Interfaz de Programación de Aplicaciones (API) para servicios de directorio.
- ✓ JMS: La API Java Message Service (servicio de mensajes Java), es la solución creada por Sun Microsystems para el uso de colas de mensajes. Es un estándar de mensajería que permite a los componentes de aplicaciones basados en la plataforma Java2 crear, enviar, recibir y leer mensajes.
- ✓ RMI/IIOP: RMI-IIOP denota la interfaz RMI de Java sobre el sistema CORBA. Este estándar fue creado intentando simplificar el desarrollo de las aplicaciones CORBA, mientras preservaba todos los beneficios principales.

- ✓ JavaMail: JavaMail es una expansión de Java que facilita el envío y recepción de e-mail desde código java.
- ✓ XML : Extensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.
- ✓ CSS: Las hojas de estilo en cascada (en inglés *Cascading Style Sheets*), CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).