Universidad de las Ciencias Informáticas

Facultad 2

Título: Plataforma de Servicios de Valor Agregado: Servicio de Suscripciones Móviles.

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

Autores: Leydis Martínez Camilo.

Yanetsy Jiménez Hidalgo.

Tutor: Ing. Rosa María Renté Labrada.

Co-Tutor: Ing. Joan Martínez Herrera.

Ciudad de la Habana, Junio del 2009

Agradecimientos

Queremos agradecer a todas las personas que de una forma u otra han ayudado al desarrollo de este trabajo, especialmente a Daymi Sablón (nuestra arquitecta particular) por su paciencia a lo largo de este año tan difícil, a Néstor Reina (Nestico) por estar siempre incondicional ante cualquier dificultad que se nos presentó, a Frank Verdecia por su valiosa ayuda, por las horas que empleó en nosotras, a Yusmar Castro y José Alexis Hernández por la disposición que siempre mostraron al ayudarnos, a Wilfredo Rosales por enseñarnos tantas cosas necesarias para nuestra vida profesional, a Joan Martínez que nos alegraba cada día, alentándonos a ser optimistas y a confiar más en nosotras mismas, a nuestra tutora Rosa María Renté y a José Antonio Pla por apoyarnos en la realización de esta investigación.

Leydis y Yanetsy

A mami y papi, por ser mi fuerza, mi vida y las personas que más quiero en el mundo.
A Nany y Yuni, por ser mis hermanos queridos, que me miman y me quieren mucho, por ser ejemplos para mí, por inspirarme a estudiar cada día más, espero algún día participar con ustedes en los Escriba y Lea familiares (jejeje). Los quiero mucho
A Eric, mi amor de pre, de universidad y espero que el de toda la vida, por su apoyo incondicional, por quererme tal y como soy, por todo, te quiero mucho nene.
A Dalia, por ser como una hermana para mí.
A Bibi, por apoyarme tanto en estos años de estudio.
A Merly, por apoyarme y casi siempre darme la razón (jejeje)
A mis abuelos, por ser tan buenos y cariñosos.
A mi sobrino Gabriel, que esta tesis sea un ejemplo para él, que siga estudiando para que un día sienta esta alegría inmensa que siento hoy.

A Yanetsy Jiménez, mi compañera de tesis, por aguantarme (nunca mejor dicho) todos estos años, mis malcriadeces, mis berrinches, mis caras de amargura (jejeje) y por ser mi mejor amiga, mi confidente.

A todos mis amigos, Yordán, Uliser, Igor, Nestico, Yadira, Yoslaine, Celia, Daya, mi primo Yadier, Joan (mi súper co-tutor), a todos los que han estado ahí para mí. Los quiero y nunca los olvidaré.

Leydis Martínez Camilo (Leydita)

A mis padres, por su amor incondicional. Por confiar siempre en mí y apoyarme, aún cuando no tomé las mejores decisiones. Por ser los padres que todo hijo desearía, por ser los mejores del mundo.

A Liudne, la persona que más me ayudó en la etapa de adaptación a esta universidad. Una persona que con su entrega y perseverancia me enseñó que darse por vencido no es una opción, que hay que luchar para tener éxito en la vida.

A Glemnis, por su apoyo en todos estos años, por estar siempre conmigo a pesar de la distancia.

A Yasel, por ser mi compañero más fiel, por apoyarme tanto todo este tiempo, por ser mi amigo.

A Leydis, mi compañera de tantos momentos, por compartir mis tristezas, alegrías, tensiones, por soportarme en mis días malos (principalmente acabada de levantar). Por ayudarme siempre, porque a pesar de ponerme de vez en cuando esa carita que solo ella sabe poner, es una gran persona y la quiero mucho. Por ser mi mejor amiga.

Yanetsy Jiménez Hidalgo (Yane)

Dedicatoria

A nuestros padres

Resumen

La comunicación es un proceso vital para el desarrollo individual y de la sociedad. Actualmente se han multiplicado en gran número las posibilidades de comunicación a distancia, y uno de los medios más populares y con más difusión en el mundo es el teléfono celular.

Con el transcurso de los años la utilidad de este dispositivo ha aumentado producto a la gran variedad de servicios que puede prestar. Entre ellos se encuentran los Servicios de Valor Agregado, por ejemplo, envío de titulares de noticias, partes climatológicos, información financiera, resultados deportivos, horóscopos, entre muchos otros, siendo además de común interés a la mayoría de los clientes, que reciben con agrado la eliminación de los procedimientos de búsqueda, acceso y consulta tradicional de determinadas informaciones en medios de prensa, Internet u otras fuentes, sustituidos por sencillos procedimientos de suscripción, que permiten a los clientes seleccionar los servicios que desean recibir así como su frecuencia de recibo u otras características.

En este trabajo se presenta una solución para la Unidad de Negocios de ETECSA, CUBACEL, encargada de prestar el servicio de telefonía móvil a nivel nacional, la cual ha solicitado un servicio de suscripciones para sus clientes. De esta forma se crea un módulo de suscripciones que formará parte de la Plataforma de Servicios de Valor Agregado de dicha entidad, compuesto por dos servicios web que cuentan con funcionalidades necesarias para realizar el proceso de suscripciones y una aplicación web a través de la cual se gestionarán los servicios disponibles.

Palabras Claves

Suscripciones, Servicios Web, Aplicación Web, Telefonía Celular.

ÍNDICE

Introducción

Las tecnologías inalámbricas han tenido un gran auge y desarrollo en estos últimos años, siendo una de ellas la telefonía celular, la cual ha revolucionado el área de las telecomunicaciones, redefiniendo cómo percibimos las comunicaciones de voz. La telefonía celular es un sistema de comunicación telefónica totalmente inalámbrica. Está básicamente formada por dos grandes partes: una red de comunicaciones (o red de telefonía móvil) y los terminales (o teléfonos móviles) que permiten el acceso a dicha red. Su principal característica es su portabilidad, que permite comunicarse desde casi cualquier lugar. Aunque su principal función es la comunicación de voz, como el teléfono convencional, su rápido desarrollo ha incorporado otras funciones como inclusión de cámaras fotográficas, agenda, acceso a Internet, reproducción de video, reproducción de archivos mp3, incluyendo también algunos servicios a los clientes como los de suscripciones, mediante las mismas las empresas brindan una serie de servicios como titulares nacionales, titulares sobre economía, horóscopo, información financiera, entre otros.

La Vicepresidencia de Servicios Móviles (VPSM) de la Empresa de Telecomunicaciones de Cuba, ETECSA, encargada de expandir y desarrollar los servicios de comunicaciones móviles en Cuba, presta servicios públicos de radio telefonía celular y de valor agregado, haciendo uso de tecnologías de avanzada como GSM (del inglés Global System Mobile Communications). La VPSM tiene previsto implementar para CUBACEL (operador celular existente en Cuba) la Plataforma de Servicios de Valor Agregado, los cuales son servicios que no son adquiridos con el paquete básico (recepción de llamadas, envío de SMS, entre otros) que brinda CUBACEL a sus clientes. Dicha plataforma contará con un módulo de suscripciones, el cual permitirá mediante servicios web realizar diferentes acciones que contribuyan al proceso de suscripción, entre ellas, brindar información, gestionar suscripciones y realizar reportes de las suscripciones. Los servicios disponibles dentro del sistema se gestionarán por medio de una aplicación web.

A partir de estas necesidades surge el problema científico: ¿Cómo posibilitar la realización de suscripciones a determinados servicios presentes en la Plataforma de Servicios de Valor Agregado?

Se define como objeto de estudio de la investigación:

- Principales características de los sistemas de suscripciones a servicios.

Dicho objeto enmarca el campo de acción de esta investigación como:

- Sistema de Suscripciones para la Plataforma de Servicios de Valor Agregado de la VPSM.

El objetivo principal de esta investigación es:

- Crear dos servicios web que permitan que se puedan llevar a cabo algunas de las funcionalidades necesarias para el proceso de suscripción y una aplicación web encargada de gestionar los servicios disponibles para realizar las suscripciones.

Los objetivos específicos que se persiguen son:

- Lograr que el sistema de suscripciones utilice información de los servicios ofrecidos por los proveedores de servicio.
- Lograr que el módulo de suscripciones sea capaz de funcionar como un todo. Teniendo un bajo acoplamiento con los otros módulos que conforman la plataforma.

Para lograr el objetivo propuesto se han trazado una serie de tareas de investigación mencionadas a continuación:

- Investigar ciertas características de las empresas que brinden servicios de suscripciones.
- Analizar el estado del arte para la evaluación del contenido que se investiga.
- Definir las herramientas y tecnologías a utilizar para el desarrollo de los servicios web y la aplicación web.

Como resultado de lo anteriormente planteado se espera obtener:

- Un sistema que cumpla con los requisitos del cliente.
- Un Sistema de Suscripciones que pueda ser utilizado por otras aplicaciones y se integre con otros servicios.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se muestran los resultados de la investigación relacionada con las diferentes herramientas, lenguajes y metodologías a utilizar para darle solución al problema planteado.

1.1 Telefonía Celular

La telefonía celular ha pasado por una serie de cambios reflejados en diferentes generaciones:

Primera generación (1G): La 1G de la telefonía móvil hizo su aparición en 1979 y se caracterizó por ser analógica y estrictamente para voz. La calidad de los enlaces era muy baja, tenían baja velocidad. En cuanto a la transferencia entre celdas, era muy imprecisa ya que contaban con una baja capacidad y la seguridad no existía. La tecnología predominante de esta generación es AMPS (del inglés Advanced Mobile Phone System).

Segunda generación (2G): La 2G no arribó hasta 1990 y a diferencia de la primera se caracterizó por ser digital. Los sistemas de 2G utilizan protocolos de codificación más sofisticados que aún se emplean en los sistemas de telefonía celular actuales. Las tecnologías predominantes son: GSM (del inglés Global System Mobile Communications); CDMA (del inglés Code Division Multiple Access). Los protocolos empleados en los sistemas 2G soportan velocidades de información más altas por voz, pero limitados en comunicación de datos. Se pueden ofrecer servicios auxiliares, como datos, fax y SMS (del inglés Short Message Service). La mayoría de los protocolos de 2G ofrecen diferentes niveles de encriptación.

Generación 2.5 G: Esta generación no existe oficialmente, constituye un paso intermedio entre la 2G y 3G. La tecnología 2.5G es más rápida, y más económica para actualizar a 3G. La generación 2.5G ofrece características extendidas, ya que cuenta con más capacidades adicionales que los sistemas 2G, como: GPRS (del inglés General Packet Radio System), además de que integra WAP (del inglés Wireless Application Protocol), MMS (del inglés Multimedia Messaging Service) y juegos para móviles.

Tercera generación 3G: La 3G se caracteriza por contener a la convergencia de voz y datos con acceso inalámbrico a Internet; en otras palabras, es apta para aplicaciones multimedia y altas transmisiones de datos. Los protocolos empleados en los sistemas 3G soportan altas velocidades de información y están

enfocados para aplicaciones más allá de la voz como audio (mp3), video en movimiento, acceso rápido a Internet, entre otros. [1]

Como continuidad de la 3G mediante la telefonía celular se pueden visitar páginas web, enviar imágenes o videos propios a cualquier lugar del mundo, reproducir archivos mp3, contar con televisión digital, juegos, tener una agenda electrónica propia, enviar mensajes escritos o de voz, mantener video-conferencias y contar con servicios de suscripciones.

1.2 Introducción a los Servicios de Suscripciones

Una suscripción es una solicitud para que se realice la entrega de un informe a una hora concreta o en respuesta a un evento. El proceso de suscripción enmarcado en el plano de la telefonía celular consta de dos fases: que el usuario se suscriba y que se le envíe la información. Un ejemplo de ello sería, a través de un Portal WAP, donde se ofrezcan diferentes categorías de servicios a los cuales suscribirse, como noticias, finanzas, clima, deportes. El usuario del Portal WAP selecciona el servicio de noticias, podría ser titulares nacionales, se suscribe y se le envía un mensaje confirmando su suscripción, luego él recibe el servicio de acuerdo a una frecuencia determinada por él o ya prevista por los administradores del servicio.

Se puede resumir que una suscripción es el acto mediante el cual un usuario recibe un servicio determinado, generalmente de acuerdo a sus preferencias ya sea de horarios o de frecuencia. Se oferta habitualmente como un servicio extra. Cada compañía ofrece el servicio de suscripción que crea más conveniente y de mayor aceptación por sus usuarios.

1.3 Estado del arte de los Servicios de Suscripciones

1.3.1 Actualidad del trabajo

En Cuba a partir del proceso de integración de los dos operadores celulares existentes (CUBACEL y C-COM) en uno único dentro de ETECSA, quedó consumado CUBACEL. El cual es el encargado de expandir y desarrollar los servicios de comunicaciones móviles. Este ofrece diferentes servicios a sus clientes, por ejemplo prepago (tarjetas de recarga, recarga por internet), pospago, mensajes (mensajería móvil, a través de operadoras), transmisión de datos (validación de tarjetas de crédito y sistemas de avisos), reparaciones, servicios de valores agregados (MMS, correo de voz, CUBACEL online), servicios adicionales (aplicaciones SIM, consulta de saldo, factura detallada), pero no cuenta con un servicio de

suscripciones, el cual es uno de los servicios que ha permitido que los teléfonos móviles se convirtieran en uno de los artículos electrónicos de consumo con el mayor volumen de ventas en el mundo. [2]

Actualmente existen diferentes empresas líderes en el marco de la telefonía móvil, mediante las cuales se han podido extender los servicios móviles a diversas partes del mundo, todas ellas ampliamente reconocidas por sus servicios de suscripciones, entre algunas de las más prestigiosas se encuentran:

Movistar: Representa a la compañía líder de telecomunicaciones de habla hispana más grande del mundo con más de 100 millones de clientes y presente en 15 países. Presenta un servicio de envío de titulares, por ejemplo, titulares de las últimas noticias, del canal de deportes, del canal informativo de Economía, del canal informativo Cultura y espectáculos y de todos los canales informativos. [3]

Nokia: Es actualmente el primer fabricante de teléfonos móviles a nivel mundial, además de una de las principales empresas del sector de las telecomunicaciones. Con sede en Finlandia, Nokia es una de las marcas más conocidas dentro y fuera de la Unión Europea. Nokia tiene un gran número de servicios de suscripciones, un ejemplo de ello es la suscripción y descarga de podcasts, las cuales son emisiones periódicas gratuitas de audio o video digital a las que se pueden suscribir permitiendo la descarga y reproducción en móviles y ordenadores personales. Permite a los clientes suscribirse mensualmente para escuchar la música de su preferencia. [4]

Orange: Forma parte del Grupo France Telecom (el quinto mercado europeo de telecomunicaciones), ofrece servicios de comunicación para grandes clientes, multinacionales y Administraciones Públicas. Su objetivo es contribuir al desarrollo de la Sociedad de la Información desarrollando una infraestructura para la creación de una oferta de nuevos servicios y aplicaciones para particulares y empresas. Este al igual que Movistar ofrece servicios de suscripciones sobre noticias, economía y deportes, además de tiempo, horóscopos, revistas, alertas en Latinoamérica, cocina, ocio y sorteos. [5]

Motorola: Es una empresa estadounidense especializada en la electrónica y las telecomunicaciones, establecida en Schaumburg, Illinois, Chicago. Los productos más conocidos de la compañía incluyen microprocesadores, teléfonos móviles y sistemas de conexión a redes de telefonía móvil e inalámbrica como Wimax y Wifi. Motorola ofrece variados servicios de suscripciones como un servicio de noticias tanto nacionales como internacionales, así como de un servicio de música que incluye 435 canales de radio de géneros variados. [6]

En la actualidad Cuba se está abriendo paso en el mundo de la telefonía celular. Mediante ETECSA cuenta con una serie de servicios ya mencionados anteriormente, pero no cuenta con un servicio de

suscripciones que oferte servicios de información financiera, partes meteorológicos, envío de titulares tanto deportivos, económicos, culturales y de ciencias como los mencionados anteriormente, los cuales permiten obtener ganancias, explotar al máximo las funcionalidades de los móviles y brindar mayores comodidades a los clientes.

1.4 Metodologías de desarrollo

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Las técnicas indican cómo debe ser realizada una actividad técnica determinada identificada en la metodología. Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. Indica cómo hay que obtener los distintos productos parciales y finales.

Aquí se describen las principales características de tres de las más famosas y conocidas metodologías de desarrollo de software: Proceso Unificado de Rational (Rational Unified Process, RUP), Programación Extrema (Extreme Programming, XP) y Microsoft Solution Framework (MSF).

XP (Extreme Programming) pertenece a las Metodologías Ágiles, está basada en heurísticas provenientes de prácticas de producción de código. Está orientada a proyectos de corto plazo de entrega y pequeños equipos de desarrollo donde se esté especialmente preparado para cambios durante el proyecto. Posee procesos poco controlados, con pocos principios. No existe contrato tradicional o al menos es bastante flexible. El cliente es parte del equipo de desarrollo, se desarrollan pocos artefactos y roles y se hace menos énfasis en la arquitectura del software. [7]

MSF: Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

MSF es una metodología que consta de cinco fases:

Visión y Alcances.
Planificación.
Desarrollo.

Estabilización.

Implantación.

Aunque mediante estas fases se logra desarrollar un proyecto con éxito, tienen poca interacción con el cliente debido a que en la primera fase se hace énfasis en la unificación del equipo detrás de una visión común donde es el equipo el que debe tener una visión clara de lo que quisiera lograr para el cliente, también se definen los líderes y responsables del proyecto, se identifican las metas y objetivos a alcanzar y se realiza la evaluación inicial de riesgos del proyecto. [8]

RUP es una metodología para proyectos de largo alcance, pero es estructurable y puede ser modificada en dependencia de las necesidades del proyecto y de la experiencia del desarrollador en el uso de esta metodología, logrando que se ajuste también a proyectos con un corto tiempo de desarrollo. Es fácil de entender tanto por usuarios avanzados como por clientes con pocos conocimientos técnicos. Está basado en normas provenientes de estándares seguidos por el entorno de desarrollo. Tiene procesos mucho más controlados, con numerosas políticas/normas. Existe un contrato prefijado donde el cliente interactúa con el equipo de desarrollo mediante reuniones. La arquitectura del software es esencial y se expresa mediante modelos. [9][10]

Por las características mencionadas de cada una de las metodologías, XP no cuenta con los requisitos necesarios para aplicarlo al sistema ya que existe un contrato prefijado, es un proyecto de largo alcance y el cliente no forma parte del equipo de desarrollo. MSF no presenta la utilidad requerida ya que no se enfoca en la primera fase de captura de requisitos, una etapa crucial, ya que, si los requisitos no son los correctos o no son bien comprendidos, el sistema que se construya tampoco será el correcto. Además, al estar basado en tecnología Microsoft, trata de obligar a usar herramientas de esta corporación. Por todo lo expuesto anteriormente y porque el proyecto acordado con la VPSM tiene como prerrequisito la entrega de una vasta documentación se empleará RUP como metodología de desarrollo, además del uso de UML (del inglés Unified Modeling Language) para la modelación de las clases pertinentes al desarrollo del mismo.

1.4.1 RUP (Rational Unified Process)

El Proceso Unificado de desarrollo de Software (RUP) divide en 4 fases el desarrollo del software:

- Inicio: El Objetivo en esta etapa es determinar la visión del proyecto.
- Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.

- Construcción: En esta etapa el objetivo es obtener la capacidad operacional inicial.
- Transición: El objetivo es llegar a obtener la versión lista para su instalación en las condiciones reales.

Características de RUP:

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo debido a que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura.

Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

Los elementos del RUP son:

- Actividades: Son los procesos que se llegan a determinar en cada iteración.
- Trabajadores: Son las personas o entes involucrados en cada proceso.
- Artefactos: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. [9][10]

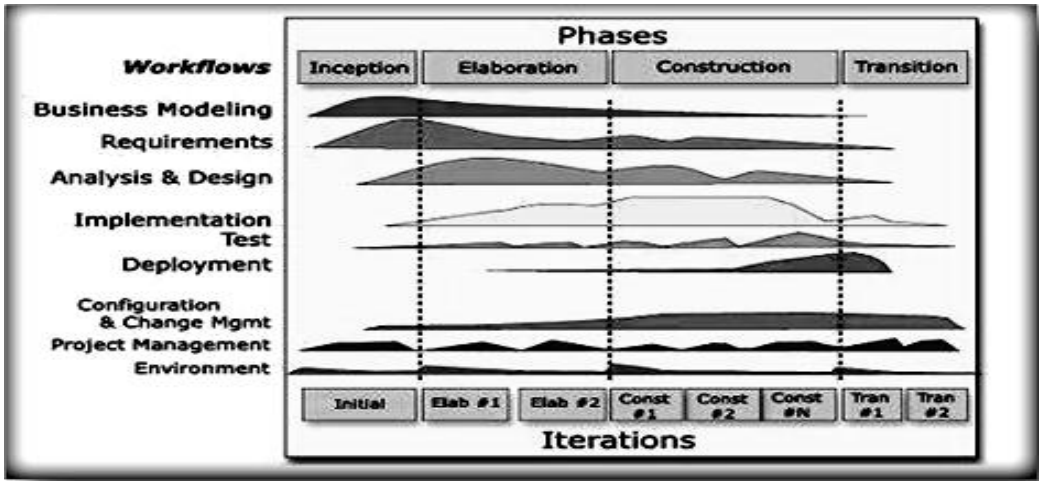

Figura 1: Fases e Iteraciones de la Metodología RUP

RUP es adaptable, según el tipo de proyecto, así serán las características del mismo, haciéndose énfasis en aquellos flujos de trabajo durante la vida del software que reporten más importancia y sean indispensables. RUP permite trazarse planes de riesgos y pruebas durante el ciclo de vida. Contiene artefactos que son diseñados durante las diferentes fases, los cuales describen detalladamente las características del software desde que se realiza el análisis del problema hasta la entrega final del producto.

1.4.2 UML: Unified Modeling Language

El Proceso Unificado RUP (Rational Unified Process) utiliza el Lenguaje de Modelado UML (Unified Modeling Language) para preparar todos los esquemas de un sistema software. UML es una parte esencial del Proceso Unificado, está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante UML es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código. [11]

UML recomienda 9 diagramas que son modelados durante las diferentes fases de desarrollo del software, es decir, representan las siguientes vistas del sistema:

- Diagrama de Casos de Uso: Modela la funcionalidad del sistema agrupándola en descripciones de acciones ejecutadas por un sistema para obtener un resultado.
- Diagrama de Clases: Muestra las clases que componen el sistema y cómo se relacionan entre sí.
- Diagrama de Objetos: Muestra una serie de objetos y sus relaciones.

- Diagrama de Secuencia: Enfatiza la interacción entre los objetos y los mensajes que intercambian entre sí junto con el orden temporal de los mismos.
- Diagrama de Colaboración: Muestra la interacción entre los objetos resaltando la organización estructural de los objetos en lugar del orden de los mensajes intercambiados.
- Diagrama de Estados: Modela el comportamiento de acuerdo con eventos.
- Diagrama de Actividades: Simplifica el Diagrama de Estados modelando el comportamiento mediante flujos de actividades.
- Diagrama de Componentes: Muestra la organización y las dependencias entre un conjunto de componentes.
- Diagrama de Despliegue: Muestra los dispositivos que se encuentran en un sistema y su distribución en el mismo. [12]

1.5 Lenguaje de Programación

1.5.1 Java:

La tecnología Java está compuesta básicamente por 2 elementos: el lenguaje Java y su plataforma. Java es un lenguaje de alto nivel, orientado a objetos, multiplataforma, es decir, que permite la ejecución de un mismo programa en múltiples sistemas operativos, como dice el axioma de Java, “write once, run everywhere”. Es distribuido ya que contiene un conjunto de clases que permiten la comunicación a través de la red facilitando así la creación de aplicaciones distribuidas. Es un lenguaje robusto, pues fue diseñado para crear software altamente fiable, en él se ha prescindido por completo de los punteros, y el recolector de basura (garbage collector) elimina la necesidad de liberación explícita de memoria.

Tiene una plataforma, que a diferencia de las demás que son una combinación de hardware y software, está basada en software, que corre sobre cualquier hardware. Consta de 2 componentes: la máquina virtual de java (JVM) y la interfaz de programación de aplicaciones (API). Este lenguaje permite a los desarrolladores implementar software en una plataforma y ejecutarlo prácticamente en cualquier otra y combinar aplicaciones o servicios basados en la tecnología Java para crear servicios o aplicaciones totalmente personalizados. Es libre y les proporciona a los desarrolladores una mayor flexibilidad, ahorro en los costes de desarrollo, independencia, compatibilidad con otros sistemas, entre otros beneficios. [13][14]

1.6 Tecnologías a utilizar

En el presente epígrafe se abordan las herramientas, tecnologías y frameworks seleccionados para el desarrollo del Módulo de Suscripciones y las principales características de estos, las cuales indican algunas razones por las que su uso es aplicable a este proyecto. Sin embargo, una de las razones de mayor peso en esta toma de decisiones es precisamente que todas ellas (a excepción de Visual Paradigm) son libres, lo que conlleva al no desembolso de capital extra por el uso de las mismas, ni el pago de ninguna licencia de uso.

1.6.1 JEE (Java Enterprise Edition)

Java Enterprise Edition es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java con arquitectura de n niveles, distribuida, basándose ampliamente en componentes de software modulares ejecutados sobre un servidor de aplicaciones. Incluye varias especificaciones de API, tales como EJB (del inglés Enterprise JavaBeans), Servlets (código en Java que se ejecuta en un servidor web), JSP (del inglés Java Server Pages) y varias tecnologías de servicios web. Es escalable, se puede añadir nuevos componentes JEE a una aplicación web para soportar al aumento de clientes, sin tener que reescribir todo el código de nuevo. [15]

1.6.2 Frameworks

Spring

Spring es un framework de aplicaciones Java/JEE desarrollado usando licencia de OpenSource. Es potente en cuanto a la gestión del ciclo de vida de los componentes y fácilmente ampliable.

Aspectos fundamentales de Spring:

- Está diseñado desde el principio para ayudar a escribir código fácil de testear.
- Se integra con Hibernate, JDO e IBatis SQL Maps en términos de soporte a implementaciones DAO.
- Presenta una potente gestión de configuración basada en JavaBeans, aplicando los principios de Inversión de Control (*IoC*). El uso de un contenedor de Inversión de Control reduce grandemente la complejidad del código a interfaces, más que a clases. El uso de los objetos a través de estas interfaces protege los requerimientos, los cuales pudieran cambiar en el desarrollo de la aplicación. [16][17]

El objetivo central de *Spring* es permitir que objetos de negocio y de acceso a datos sean reutilizables, no atados a servicios JEE específicos. Estos objetos pueden ser reutilizados tanto en entornos JEE (web o EJB), aplicaciones “standalone”, entornos de pruebas, etc.
La arquitectura en capas de *Spring* ofrece mucha flexibilidad. Toda la funcionalidad está construida sobre los niveles inferiores. Por ejemplo se puede utilizar la gestión de configuración basada en JavaBeans sin utilizar el framework MVC o el soporte AOP (Programación orientada a aspectos). [16][17]

Spring Security

Spring Security es la versión nueva de Acegi Security, la cual es una librería de seguridad para JEE. Spring Security es un framework de seguridad para las aplicaciones basadas en Spring. Proporciona una solución completa a la seguridad al implementar servicios de autenticación y autorización a recursos, tanto a nivel de peticiones Web como a nivel de invocación de métodos. Sobre la base de Spring, Spring Security aprovecha al máximo las técnicas de Inyección de Dependencia y orientación a aspectos. Posee seguridad basada en roles, en listas de control de acceso. Spring Security cuenta con una serie de ventajas:
Es capaz de gestionar seguridad en varios niveles: URLs que se solicitan al servidor, acceso a métodos y clases Java, y acceso a instancias concretas de las clases.
Permite separar la lógica de las aplicaciones del control de la seguridad, utilizando filtros para las peticiones al servidor de aplicaciones o aspectos para la seguridad en clases y métodos. La configuración de la seguridad es portable de un servidor a otro, ya que se encuentra dentro del WAR o el EAR de las aplicaciones.
Soporta muchos modelos de identificación de los usuarios (HTTP BASIC, HTTP Digest, basada en formulario, LDAP, OpenID, JAAS y muchos más). Además se pueden ampliar estos mecanismos implementando clases que extiendan el modelo de Spring Security. [18]

Spring Web Services

Spring Web Services (Spring-WS) es un producto de la comunidad de Spring centrada en la creación de documentos manejadores de servicios web. Spring Web Services tiene por objeto facilitar el contrato primero del servicio de despliegue SOAP, lo que permite la creación flexible de servicios web. La creación del contrato primero del servicio constituye una buena práctica a la hora de desarrollar servicios web ya que permite un bajo acoplamiento entre el contrato y la implementación del servicio.

El framework tiene soporte para varias tecnologías y APIs de manejo de contenidos de los mensajes XML. Brinda soporte al estándar Web Service Security permitiendo firmar mensajes SOAP, encriptar y desencriptar estos mensajes, así como la autenticación a través de los mismos. El producto se basa en Spring, lo que significa que pueden usarse los conceptos de Spring tales como la Inyección de Dependencia como parte integrante del servicio web. Spring-WS usa los contextos de aplicaciones de Spring para toda la configuración, lo que ayuda a los desarrolladores de Spring a desarrollar rápidamente. Además la arquitectura de Spring-WS se asemeja a la de Spring-MVC. [19]

JUnit

JUnit es un proyecto de código abierto que se convirtió rápidamente en el estándar para el desarrollo de pruebas de unidad y de integración para Java. JUnit es un conjunto de clases o *framework* que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente. [20]

IBatis

IBatis es un framework que facilita la implementación de la capa de persistencia utilizada en las aplicaciones. Es conocido como un mapeador de datos, lo que significa que mueve los datos entre los objetos y la base de datos manteniendo independientes el uno del otro y del propio mapeador.

Se divide en dos elementos [21]:

- IBatis Data Mapper (SQL Maps)**:** proporciona un modo simple y flexible de mover los datos entre los objetos Java y la base de datos relacional. Este framework mapea clases a sentencias SQL usando un descriptor XML muy simple.
- IBatis Data Access Object (DAO): Implementación realizado por el framework del patrón DAO. IBatis DAO es una capa de abstracción que oculta los detalles de la capa de persistencia y proporciona un API común para el resto de la aplicación. Con los DAOs se permite configurar una aplicación dinámicamente para usar distintos mecanismos de persistencia.

IBatis es ampliamente considerado como uno de los frameworks de persistencia más simples disponibles en la actualidad. Esta simplicidad está dada por la sólida base sobre la cual se construye IBatis: JDBC y SQL. Posibilita separar el trabajo del programador de SQL y el programador Java manteniendo la consistencia que se requiere. Es software libre, de código abierto, desarrollado por la Apache Software Foundation. [22]

1.6.3 Servicios Web

Servicios web, como indica su propio nombre, son servicios ofertados vía web. Son sistemas de software diseñados para lograr la interoperabilidad entre aplicaciones. Se puede definir de manera más sencilla como un conjunto de tecnologías estándares de software para el intercambio de datos entre aplicaciones tales como SOAP, WSDL y UDDI. Puede ser desarrollado en una gran variedad de lenguajes para muchos tipos de redes de computadores.

Funcionamiento:

Un servicio web, en vez de obtener peticiones desde un navegador y devolver páginas web como respuesta, recibe peticiones mediante un mensaje formateado con SOAP, desde otras aplicaciones realiza la labor que le han pedido y devuelve un mensaje de respuesta también con formato SOAP.

Estándares para servicios web: SOAP, WSLD, UDDI

Los estándares son definiciones o formatos que se aprueban o reconocen desde organizaciones de estandarización. Generalmente estos organismos están formados por el conjunto de empresas más representativas de un sector o de un campo de la producción. Los servicios web se construyen sobre estándares y a su vez pretenden ser un estándar con los que construir sistemas a partir de piezas dispares, desarrolladas por distintos fabricantes, funcionando en distintos sistemas, y construidas con distintas tecnologías. [23]

SOAP (del inglés *Simple Object Access Protocol*) es un protocolo de mensajería XML extensible que forma la base de los servicios web. SOAP proporciona un mecanismo simple y consistente que permite a una aplicación enviar mensajes XML a otra aplicación. SOAP es un protocolo de alto nivel que sólo define la estructura del mensaje y unas pocas reglas para su procesamiento. Es completamente independiente del protocolo de transporte subyacente, por eso los mensajes SOAP se pueden intercambiar sobre HTTP, SMTP, entre otros. SOAP define precisamente cómo se deben codificar las llamadas a los métodos de un servicio web, y cómo debe el servicio web codificar el resultado para que se pueda interpretar.

WSDL (del inglés *Web Services Description Language*) es un fichero XML usado para describir la interfaz de un servicio web como un conjunto de puntos finales de comunicación (métodos) capaces de intercambiar mensajes (es decir, recibir llamadas con sus parámetros correspondientes y generar respuesta con el resultado que le corresponda). Proporciona toda la información necesaria para acceder y utilizar un servicio web. Un documento WSDL describe qué hace el servicio web, cómo se comunica, y dónde reside.

UDDI (del inglés *Universal Description, Discovery and Integration*) es un directorio de servicios web distribuido y basado en web que permite que se listen, busquen y descubran este tipo de software. Su objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios web del catálogo de registros. El registro en el directorio se hace en XML. Se podría comparar con las típicas páginas amarillas. [23]

Algunas de las razones por las que se utilizan servicios web son:

- Son auto – descriptivos.
- Pueden buscar registros de otros servicios web.
- Tienen un bajo acoplamiento: El cliente no necesita conocer nada acerca de la implementación del servicio al que está accediendo, salvo la definición WSDL.
- Son independientes del lenguaje de programación: El servidor y el cliente no necesitan estar escritos en el mismo lenguaje
- Son independientes del modo de transporte: SOAP puede funcionar sobre múltiples protocolos de transporte, como por ejemplo HTTP, HTTPS, SMTP, etc.
- Son extensibles: Al estar basados en XML, que es un lenguaje abierto, son fáciles de adaptar, extender y personalizar.

1.7 Herramientas a utilizar

1.7.1 PostgreSQL 8.3

PostgreSQL es un Sistema Gestor de Bases de Datos Relacionales Orientadas a objetos, liberado bajo la licencia BSD o Berkeley Software Distribution (licencias de software libre). Cuenta con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, Perl y Python). Soporta casi toda la sintaxis SQL y tiene gran escalabilidad ya que es ajustable al número de procesadores y a la cantidad de

memoria que posee el sistema de forma eficiente, por este motivo es capaz de soportar una mayor cantidad de peticiones simultáneas. PostgreSQL está ampliamente considerado como uno de los sistemas de bases de datos de código abierto más avanzado del mundo. [24]

1.7.2 Apache Tomcat 6.0

Tomcat es un servidor web con soporte de servlets y JSPs. Puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java. [25]

1.7.3 Eclipse Ganymede

Eclipse es un entorno de desarrollo integrado de código abierto y multiplataforma. Es desarrollado por la Fundación Eclipse, una organización independiente que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. La definición que da el proyecto Eclipse acerca de su software es: "una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular" [26]. Constituye una plataforma universal para integrar herramientas de desarrollo, con una arquitectura abierta y basada en módulos (plugins) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están generalmente prefijadas, las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes de programación además de Java, así como introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo. [26]

1.7.4 Visual Paradigm

Herramienta Case para el modelado utilizando UML, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite realizar todos los tipos de diagramas de clases, generar código inverso, código desde diagramas y documentación. Es una herramienta multiplataforma a diferencia de otras herramientas de modelado como el Rational Rose. Además, la UCI (Universidad de Ciencias Informáticas) cuenta con la

licencia para el uso de Visual Paradigm, por eso se fomenta su utilización en la mayoría de los proyectos. [27]

1.7.5 SoapUI

SoapUI es una herramienta de Software Libre, gráfica, está basada en Java y sirve para el testeo de los servicios web, así como para la generación de clientes de estos. Permite a través de una interfaz gráfica obtener información de dichos servicios, realizar llamados generando las peticiones para cada método del servicio web y visualizando las respuestas enviadas por el servidor, permitiendo identificar las operaciones asociadas con el servicio web y permitiendo hacer pruebas de su funcionamiento sin necesidad de recurrir a escribir código para crear clientes que consuman dichos servicios. [28]

Conclusiones

En este capítulo se analizó el estado del arte, llegando a conclusiones previas de la situación de la telefonía celular en Cuba como en otros países. Se seleccionó el lenguaje de programación, así como las tecnologías y herramientas informáticas de mayor efectividad para llevar a cabo el presente trabajo.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

Introducción

En este capítulo se describe la propuesta de un Módulo de Suscripciones como parte de la Plataforma de Servicios de Valor Agregado para la Vice Presidencia de Servicios Móviles de CUBACEL, dicho módulo brindará algunas de las funcionalidades necesarias para llevar a cabo el proceso de suscripción. Se presenta un modelo de dominio como alternativa al modelo de negocio, modelando y relacionando los principales conceptos que se identificaron en el campo de acción. También se enumeran los requerimientos funcionales y no funcionales del sistema que se propone. Además se identificaron los actores y Casos de Uso (CU), que permitieron obtener el Modelo de CU del sistema utilizando la metodología RUP y UML como lenguaje de modelado.

2.1 Propuesta de sistema.

2.1.1 Descripción General

Se piensa desarrollar dos servicios web: Suscripciones_WAP y Suscripciones_Proveedor. El Portal WAP de CUBACEL será el que consuma el servicio Suscripciones_WAP, el cual permitirá que se gestionen las suscripciones y se obtengan diferentes informaciones necesarias para llevar a cabo la primera fase del proceso de suscripción, por ejemplo, todos los servicios disponibles, las categorías de los servicios, etc. En la Plataforma de Servicios de Valor Agregado se encontrarán los proveedores de servicio, los cuales utilizarán el servicio web Suscripciones_Proveedor. Dicho servicio contará con una serie de reportes, necesarios para garantizar el cumplimiento de la segunda fase del proceso de suscripción, por ejemplo, los teléfonos de todos los usuarios suscritos a un determinado servicio, la cantidad de usuarios de baja a un servicio determinado, etc. El sistema contará con una aplicación web que permita la gestión de los servicios disponibles para las suscripciones.

Esta investigación solo abarca el almacenamiento y devolución de la información requerida para el proceso de suscripción, no el proceso completo.

2.2 Modelo de dominio

Se propone un modelo del dominio, ya que en el sistema los procesos del negocio no están bien definidos, no son visibles y las fronteras no están bien establecidas. Además permite de manera visual mostrar al

usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo. Esto ayuda a los usuarios, clientes, desarrolladores e interesados, a utilizar un vocabulario común para poder entender el contexto en que se emplaza el sistema. Este modelo va a contribuir posteriormente a identificar algunas clases que se utilizarán en el sistema. Ver figura 2.

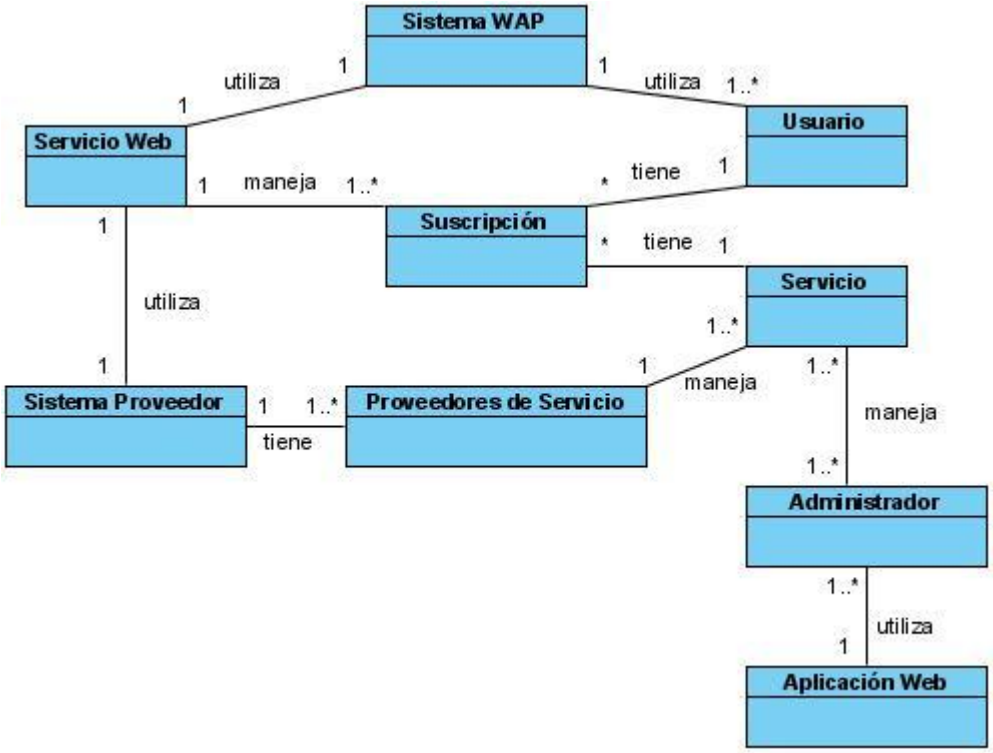

Figura 2.Diagrama de Objetos

Descripción de los objetos del dominio

Servicio Web: objeto que representa los servicios web.

Sistema WAP: objeto que representa al Portal Wap cuyo objetivo es consumir uno de los servicios web.

Sistema Proveedor: objeto que representa al sistema de los proveedores de servicios que va a consumir uno de los servicios web.

Proveedores de Servicio: objeto que representa a los proveedores de servicios.

Administrador: objeto que gestiona los servicios.

Servicio: objeto que representa los diferentes servicios, como, titulares internacionales, estado del tiempo, horóscopo, etc.
Suscripciones: objeto que representa a los suscriptores con sus servicios.
Usuarios: objeto que representa a las personas que se benefician con el servicio de suscripciones.
Aplicación Web: objeto que representa a la aplicación web que será utilizada por los administradores para gestionar servicios.

2.3 Relación de los Requerimientos.

Un requerimiento es una condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. [29]. Los requisitos se pueden clasificar en: funcionales y no funcionales.

2.3.1 Requerimientos Funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir.

R1. Autenticar.
- 1.1 Permitir a los administradores del sistema autenticarse.

R2. Gestionar Administrador
- 2.1 Permitir a los administradores adicionar administrador.
- 2.2 Permitir a los administradores modificar administrador.
- 2.3 Permitir a los administradores eliminar administrador.

R3. Gestionar Servicio
- 3.1 Permitir a los administradores adicionar servicio.
- 3.2 Permitir a los administradores modificar servicio.
- 3.3 Permitir a los administradores eliminar servicio.

R4. Gestionar Categoría
- 4.1 Permitir a los administradores adicionar categoría.
- 4.2 Permitir a los administradores eliminar categoría.

R5. Gestionar Cliente de Servicio
- 5.1 Permitir a los administradores adicionar cliente de servicio.
- 5.2 Permitir a los administradores modificar cliente de servicio.

5.3 Permitir a los administradores eliminar cliente de servicio.

R6. Gestionar Suscripción.

6.1 Permitir adicionar suscripción.

6.2 Permitir modificar suscripción.

6.3 Permitir eliminar suscripción.

R7. Mostrar Información.

7.1 Mostrar todas las categorías.

7.2 Mostrar todos los servicios.

7.3 Mostrar todos los servicios de una categoría.

7.4 Mostrar las suscripciones de un usuario.

R8. Mostrar al Proveedor de Servicio los reportes de las suscripciones.

8.1 Mostrar la cantidad de usuarios que se han dado de baja de un servicio determinado.

8.2 Mostrar la cantidad de usuarios que se han suscrito a un servicio determinado.

8.3 Mostrar todos los usuarios suscritos a un servicio.

8.4 Mostrar el valor del campo del servicio de un usuario determinado.

8.5 Mostrar los campos de un servicio.

2.3.2 Requerimientos no Funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener.

Software

- Se necesita tener instalada la máquina virtual de Java (JVM)
- Se necesita tener instalado el Sistema Gestor de Base de Datos PostgreSQL.
- Se necesita tener instalado el servidor web Apache Tomcat 6.0

Hardware

- Se necesita como requerimientos mínimos una PC con procesador Intel Pentium IV.
- Requiere como mínimo una memoria RAM de 1 GB.

Usabilidad

- Se debe tener un conocimiento previo sobre el manejo de servicios web. En cuanto a la aplicación web podrá ser usada por cualquier usuario con conocimientos básicos del tema.

Soporte

- Contará con la documentación generada por el proyecto.

Portabilidad

- El sistema deberá ser soportado por cualquier Sistema Operativo.

Seguridad

- El acceso debe ser controlado con nombres de usuario y contraseñas.

2.4 Modelo de Casos de Uso del Sistema.

A continuación se presenta el modelo de casos de uso del sistema que comprende los actores del sistema, casos de uso que estos inicializan, el diagrama de casos de uso del sistema y una descripción detallada de estos casos de uso.

2.4.1 Definición de los Actores.

Tabla 1.1 Definición de Actores

Actores	Justificación
Administrador	Representa a la persona que administra todos los servicios.
Sistema WAP	Representa al sistema que consume el servicio web Suscripciones_WAP.
Sistema Proveedor	Representa al sistema que consume el servicio web Suscripciones_Proveedor.

2.4.2 Listado de Casos de Uso del sistema

Los casos de uso son artefactos narrativos que describen el comportamiento del sistema desde el punto de vista del usuario. Establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema. A continuación se muestran los casos de uso del sistema identificados teniendo en cuenta los requisitos funcionales.

Autenticar.

Gestionar Administrador.

Gestionar Servicio.

Gestionar Categoría.

Gestionar Cliente de Servicio.

Gestionar Suscripción.

Mostrar Información.

Mostrar Reportes de Suscripciones.

2.4.3 Diagrama de Casos de Uso del sistema

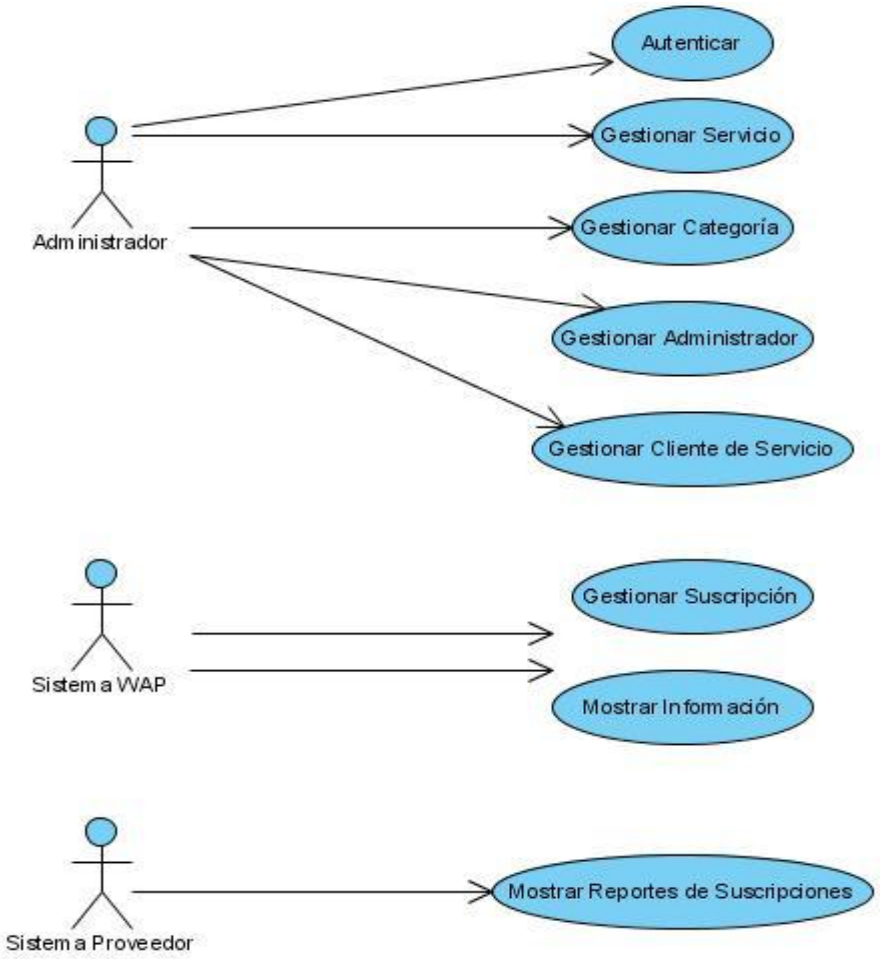

Figura 3. Diagrama de Casos de Uso del Sistema

2.4.4 Descripción de los Casos de Uso del sistema

A continuación se muestran la descripción de los casos de uso más significativos, las demás descripciones se pueden ver en el Anexo 1.

Tabla 1.4. Descripción del Caso de Uso Gestionar Servicio

Caso de Uso	**Gestionar Servicio**
Actor	Administrador.
Propósito	Permitir al administrador gestionar los servicios.
Resumen	El actor decide que acción va a realizar (adicionar, modificar, eliminar). En el caso de adicionar o modificar, llena o actualiza los datos. Si es eliminar selecciona el servicio y lo elimina.
Referencias	R3.
Precondiciones	El administrador debe estar autenticado en el sistema.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Gestionar Servicios del menú principal.	2. El sistema muestra una tabla con los diferentes servicios a gestionar.
3. El administrador selecciona la opción: Adicionar (Ver sección “Adicionar servicio”). Modificar (Ver sección “Modificar servicio”). Eliminar (Ver sección “Eliminar servicio”).	
Sección: “Adicionar Servicio”	
1. El administrador selecciona la opción Adicionar.	2. El sistema muestra la interfaz con los campos siguientes: Nombre Proveedor

	Categoría Descripción Código Campos
3. El administrador introduce los campos y la opción Aceptar.	4. El sistema verifica que los campos hayan sido llenados correctamente.
	5. Registra la nueva funcionalidad.
Flujos Alternos de los Eventos	
	4.1 Se comprueba que existen campos vacíos.
	4.2 Muestra un mensaje: “Datos incorrectos o no válidos”.
Sección:¨Modificar Servicios¨	
Acción del actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Modificar.	
3. El administrador selecciona el servicio a modificar.	4. El sistema muestra una interfaz con todos los campos: Nombre Proveedor Categoría Descripción Código

	Campos
5. El administrador realiza los cambios deseados y selecciona la opción Aceptar.	6. El sistema verifica que los datos sean válidos.
	7. Guarda los cambios realizados.
Flujos Alternos de los Eventos	
	6.1 Comprueba que los datos son incorrectos.
	6.2 Muestra un mensaje: “Datos incorrectos o no válidos”.
Sección: ¨Eliminar Servicio¨	
Acción del actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Eliminar.	2. El sistema muestra un mensaje: “Está seguro que desea eliminar el servicio.” Aceptar Cancelar
5. El administrador selecciona la opción Aceptar.	6. El sistema elimina el servicio.
Flujos Alternos de los Eventos	
5.1 El administrador selecciona la opción Cancelar.	6.1 El sistema cancela la operación.
Postcondiciones	Se actualiza toda la información referente a los servicios.
Prioridad	Crítica

Tabla 1.8 Descripción del Caso de Uso Mostrar Información

Caso de Uso	**Mostrar Información**
Actor	Sistema WAP
Propósito	Permitir que el sistema WAP obtenga la información necesaria para llevar a cabo parte del proceso de suscripción.
Resumen	El actor decide que acción va a solicitar (Ver todas las categorías, todos los servicios, todos los servicios de una categoría, las suscripciones de un usuario). En cualquier caso el sistema mostrará la información solicitada.
Referencias	R7.
Precondiciones	El sistema WAP debe estar identificado.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El sistema WAP realiza una acción: Ver todas las categorías (Ver sección “Todas las categorías”). Ver todos los servicios (Ver sección “Todos los servicios”). Ver todos los servicios de una categoría (Ver sección:”Servicios de una categoría ”) Las suscripciones del cliente(Ver sección :”Suscripciones de un usuario”)	
Sección: “Todas las categorías”	
Acción del actor	**Respuesta del sistema**
1. El sistema WAP selecciona la opción Ver todas las categorías.	2. El sistema busca todas las categorías existentes.

	3. Devuelve todas las categorías.
Sección: "Todos los servicios"	
1. El sistema WAP selecciona la opción Ver todos los servicios.	2. El sistema busca todos los servicios existentes.
	3. Devuelve todos los servicios.
Sección: "Servicios de una categoría"	
Actor	**Sistema**
1. El sistema WAP selecciona la opción "Servicios de una categoría".	
2. El sistema WAP le envía el identificador de la categoría.	3. El sistema busca todos los servicios relacionados con dicha categoría.
	4. Devuelve al sistema WAP todos los servicios de la categoría.
Sección: "Suscripciones de un usuario"	
Actor	**Sistema**
1. El sistema WAP selecciona la opción "Suscripciones del usuario".	
2. El sistema WAP le envía el número telefónico del usuario.	2. El sistema comprueba que dicho usuario esté suscrito.
	3. El sistema busca las suscripciones.
	4. El sistema le devuelve todas las suscripciones del usuario.
Flujos Alternos	
	3.1 El sistema comprueba que no tiene suscripciones

Postcondiciones	El sistema WAP obtiene las informaciones.
Prioridad	Crítico

Conclusiones

En este capítulo se realizó una descripción de la solución propuesta, se representó el modelo de dominio con los principales objetos del sistema, así como los requisitos funcionales y no funcionales. Se realizó el diagrama de casos de uso del sistema, así como la descripción detallada de estos.

CAPÍTULO 3. ANÁLISIS Y DISEÑO

Introducción

En este capítulo se presentan los diagramas de clases del análisis y del diseño, así como los diagramas de interacción de ambos flujos y el Modelo de Datos. También se describe la arquitectura utilizada así como los patrones empleados.

3.1 Análisis

3.1.1 Definición del modelo de análisis. Modelo de clases de análisis.

Un Diagrama de Clases del Análisis es un artefacto en el que se representan los conceptos en un dominio del problema. [30]

A continuación se muestran los Diagramas de Clases del Análisis de los casos de uso más significativos, los demás se pueden ver en el Anexo 2.

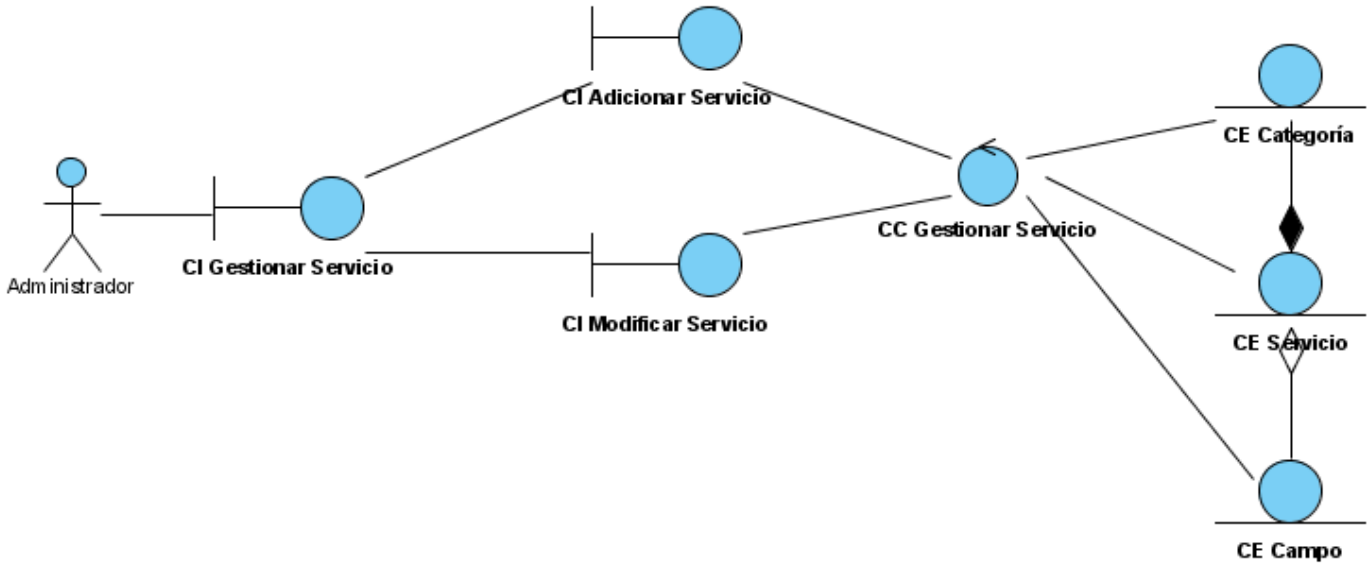

Figura 6. Diagrama de Clases del Análisis: Gestionar Servicio

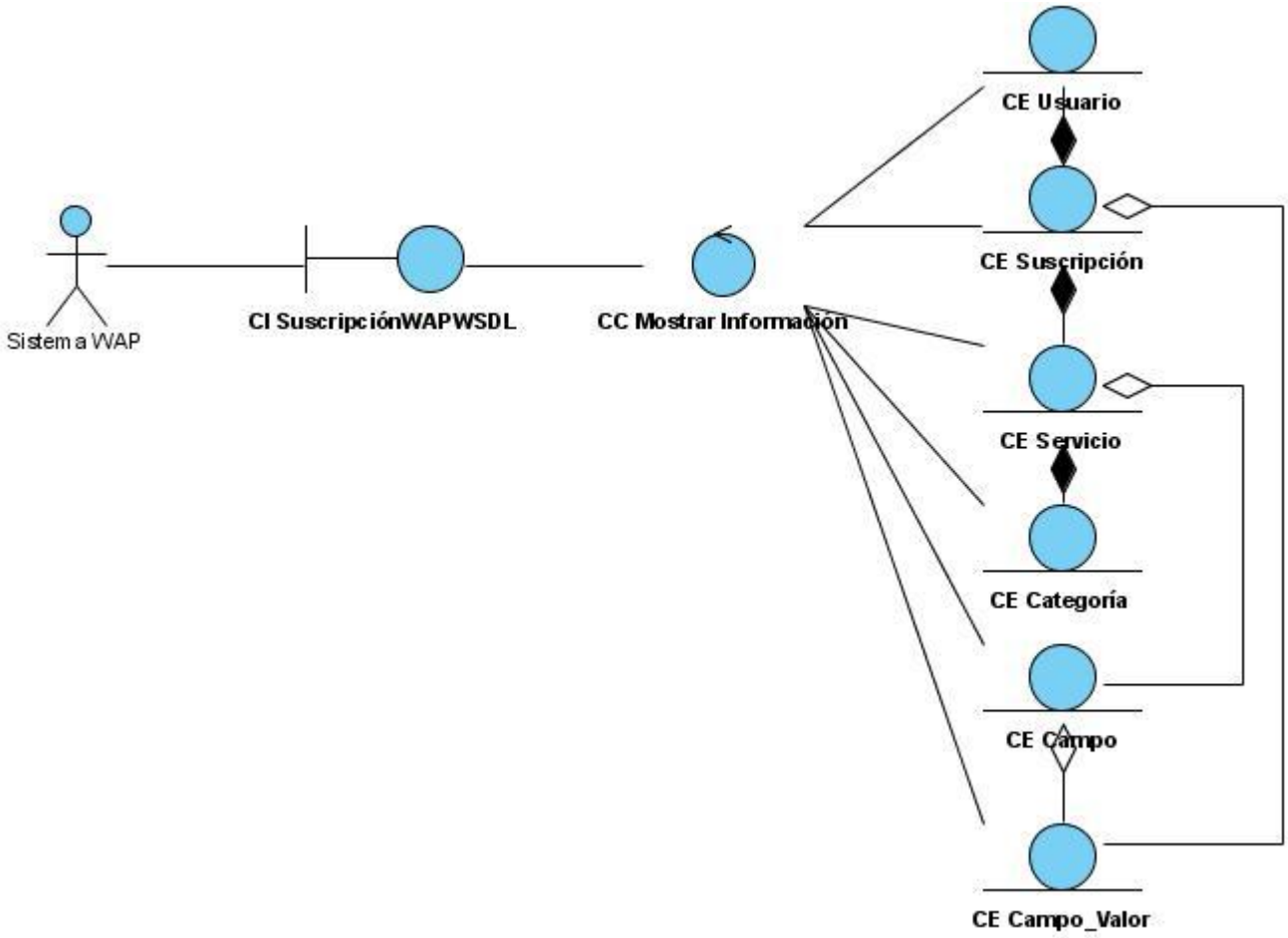

Figura 10. Diagrama de Clases del Análisis: Mostrar Información

3.2 Diseño

Un diagrama de Clases del Diseño muestra cómo puede ser construido el sistema.

A continuación se muestran los Diagramas de Clases del Diseño de los casos de uso más significativos, los demás se pueden ver en el Anexo 3.

3.2.1 Diagramas de Clases del Diseño

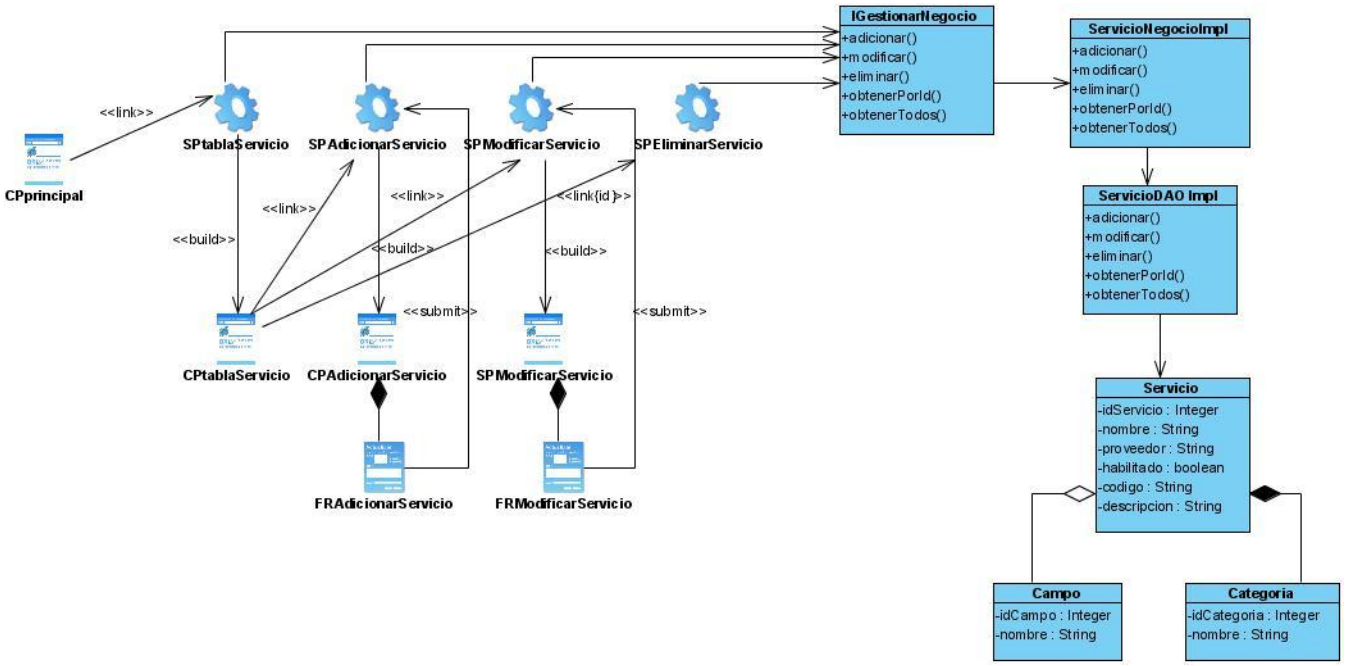

Figura 14. Diagrama de Clases del Diseño: Gestionar Servicio

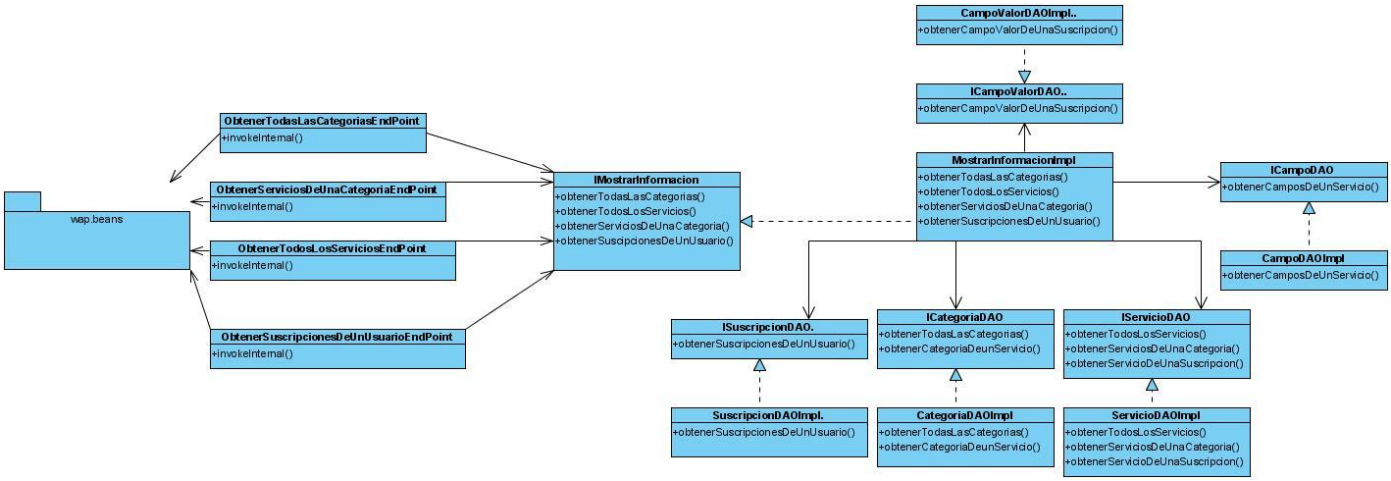

Figura 18. Diagrama de Clases del Diseño: Mostrar Información

3.3 Diagrama de Secuencia del Diseño

A continuación se muestran los Diagramas de Secuencia del Diseño de los casos de uso más significativos, los demás se pueden ver en el Anexo 4.

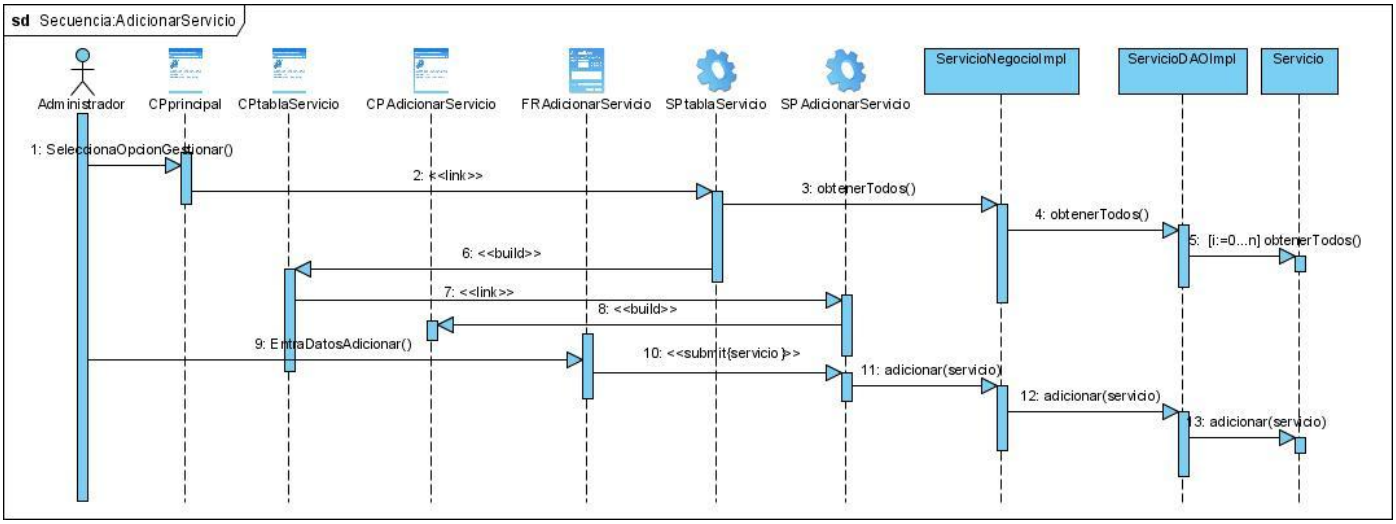

Figura 24. Diagrama de Secuencia del Diseño: Adicionar Servicio

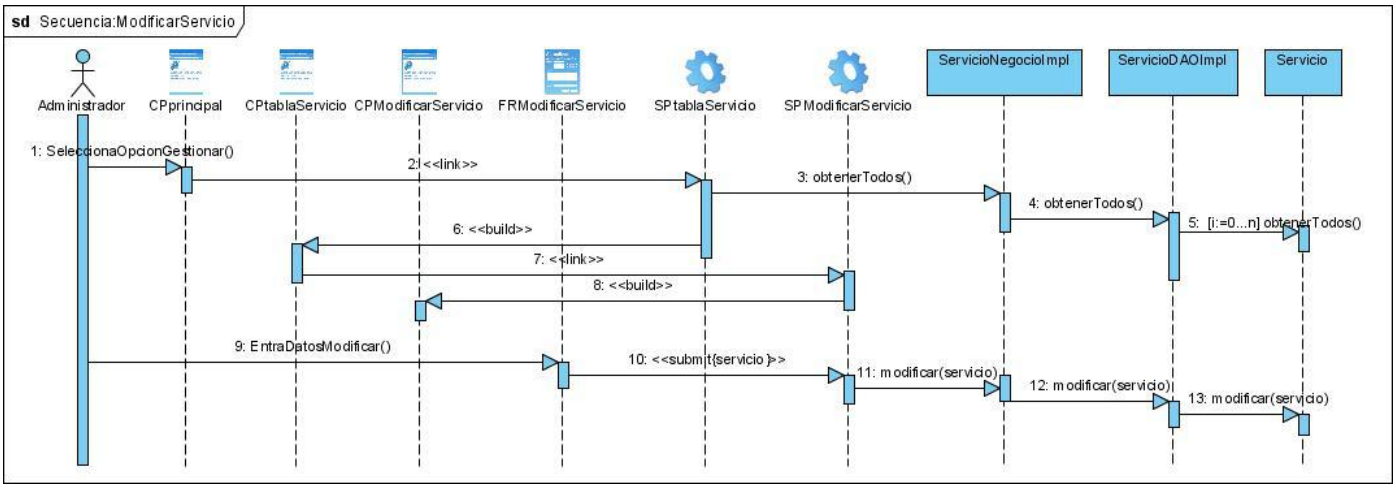

Figura 25. Diagrama de Secuencia del Diseño: Modificar Servicio

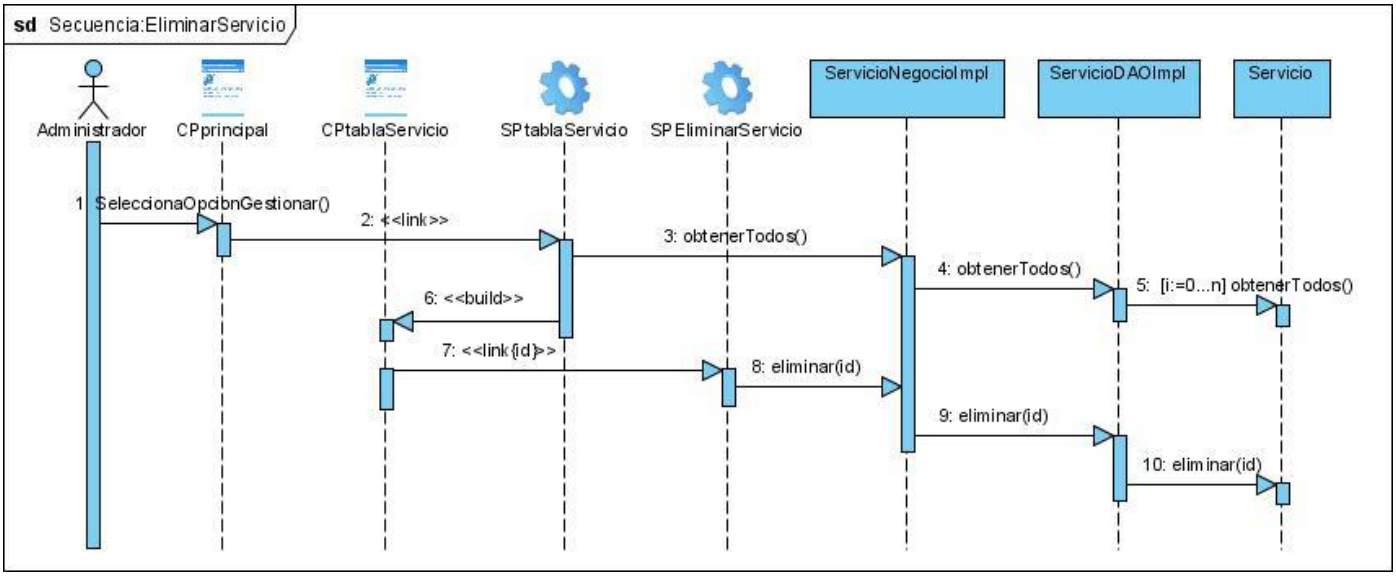

Figura 26. Diagrama de Secuencia del Diseño: Eliminar Servicio

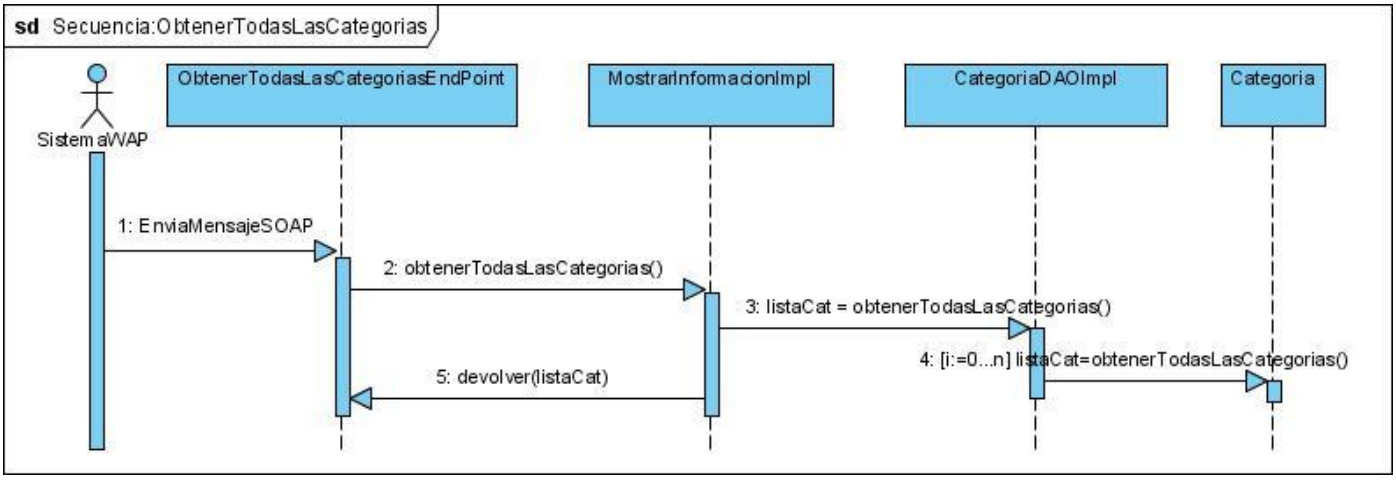

Figura 35. Diagrama de Secuencia del Diseño: Mostrar Información (Obtener todas las categorías)

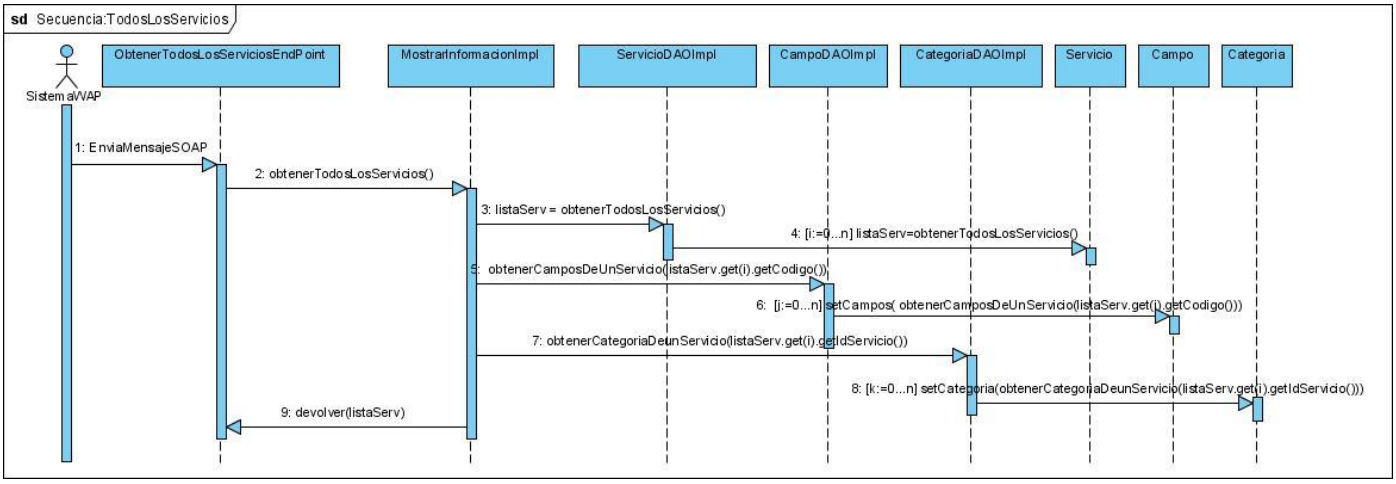

Figura 36. Diagrama de Secuencia del Diseño: Mostrar Información (Obtener todos los servicios)

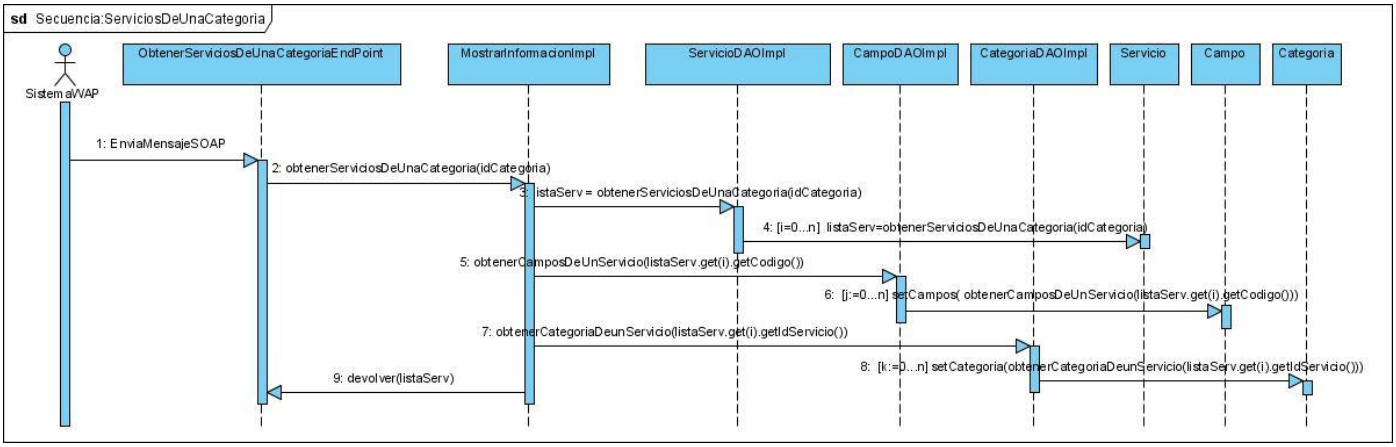

Figura 37. Diagrama de Secuencia del Diseño: Mostrar Información (Obtener servicios de una categoría)

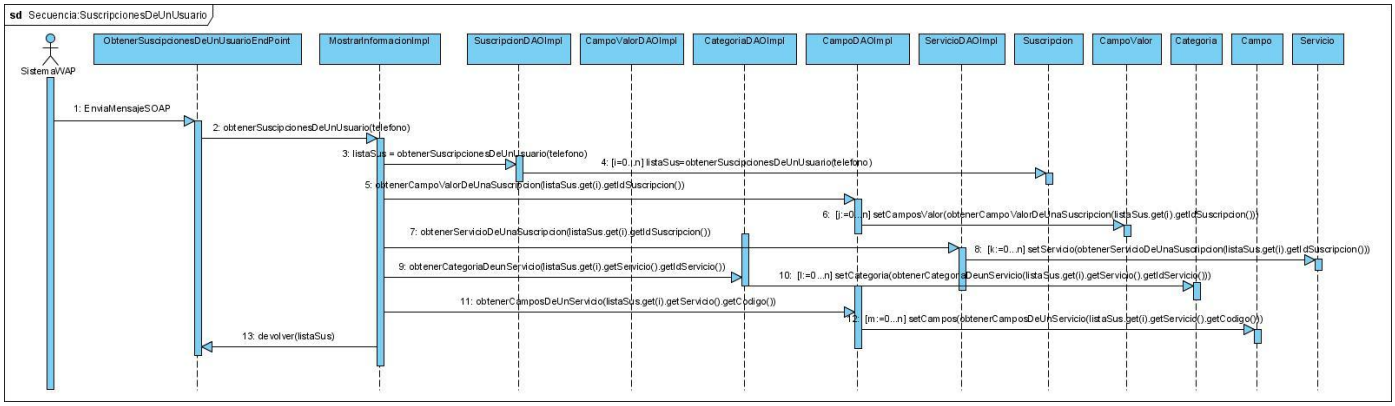

Figura 38. Diagrama de Secuencia del Diseño: Mostrar Información (Obtener suscripciones de un usuario)

3.4 Diagrama de clases persistentes

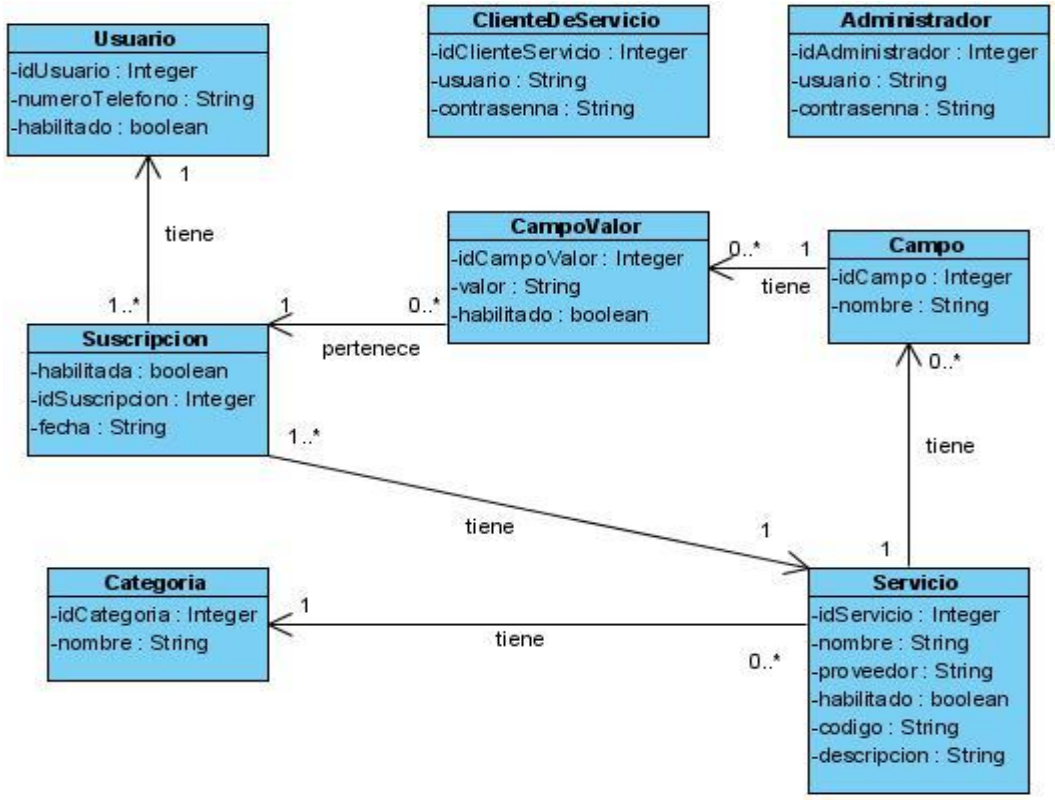

Figura 43. Diagrama de Clases Persistentes

3.5 Diseño de la Base de Datos

Una base de datos o banco de datos (del inglés Database) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. El diseño de la base de datos es una de las tareas más importantes en la construcción de un sistema, ya que una base de datos bien diseñada contendrá información correcta, almacenará los datos más eficientemente y será más fácil de gestionar y de mantener. [31]

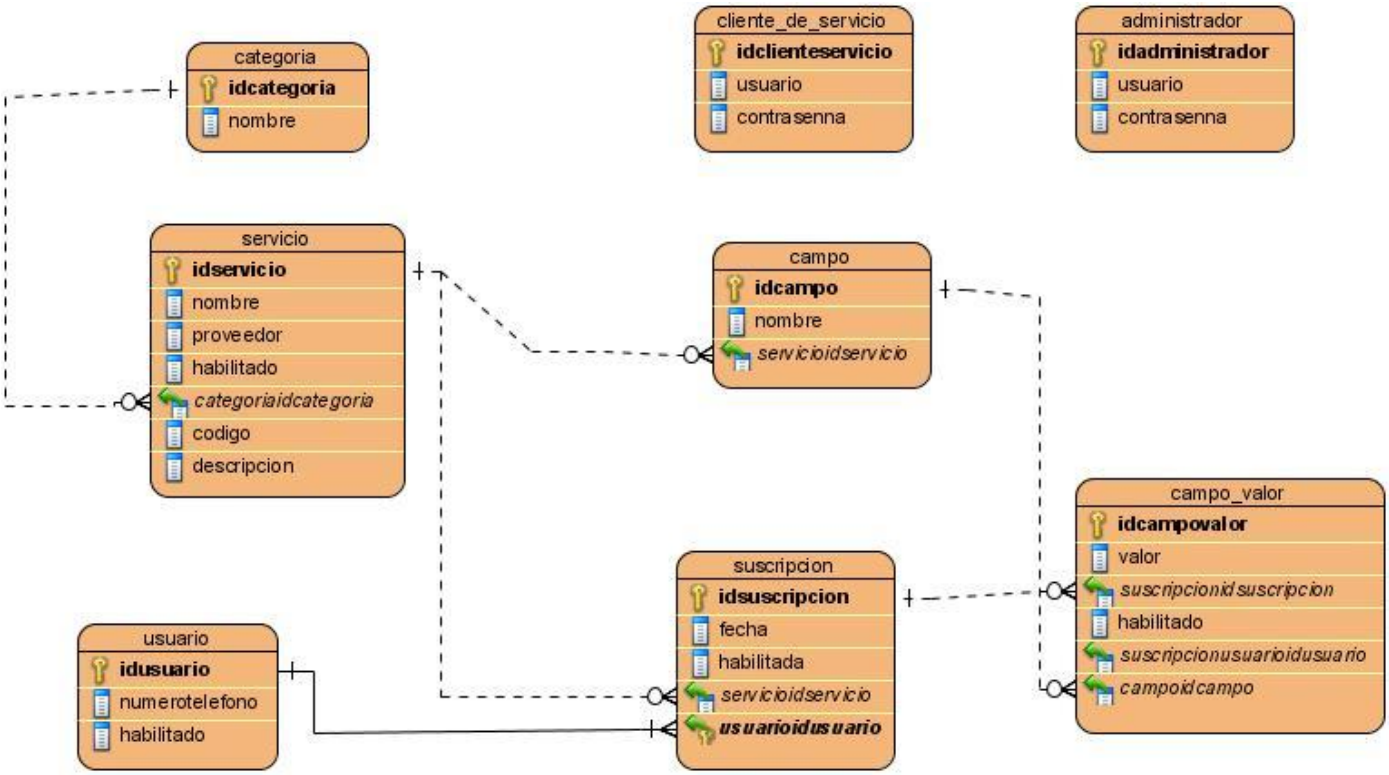

Figura 44. Diseño de la Base de Datos

3.5.1 Descripción de las tablas

Tabla 2.1 Descripción de la tabla: categoria

Nombre: categoria		
Descripción: Almacena todos los valores constantes relacionados con categorías.		
Atributo	**Tipo**	**Descripción**
idcategoria	int	Identificador de la tabla
nombre	string	Nombre de la categoría

Tabla 2.2 Descripción de la tabla: campo

Nombre: campo		
Descripción: Almacena todos los valores que puede tener un determinado campo de un servicio.		
Atributo	**Tipo**	**Descripción**
idcampo	int	Identificador de la tabla
nombre	string	Nombre del campo

Tabla 2.3 Descripción de la tabla: usuario

Nombre: usuario		
Descripción: Almacena todos los usuarios.		
Atributo	**Tipo**	**Descripción**
idusuario	int	Identificador de la tabla
numerotelefono	string	Número telefónico de cada usuario
habilitado	bool	Si el usuario no se ha dado de baja del servicio se mantiene habilitado.

Tabla 2.4 Descripción de la tabla: administrador

Nombre: administrador		
Descripción: Almacena todos los administradores.		
Atributo	**Tipo**	**Descripción**
idadministrador	int	Identificador de la tabla
usuario	string	Nombre del administrador
contrasenna	string	Contraseñas pertenecientes a cada uno de los

		administradores

Tabla 2.5 Descripción de la tabla: servicio

Nombre: servicio		
Descripción: Almacena todos los servicios.		
Atributo	**Tipo**	**Descripción**
idservicio	int	Identificador de la tabla
nombre	string	Nombre del servicio
proveedor	string	Los proveedores de cada servicio.
habilitado	bool	Si el servicio se mantiene activo(que esté disponible para realizar suscripciones) está habilitado
descripcion	string	Breve descripción del servicio
codigo	string	Código identificador de cada servicio

Tabla 2.6 Descripción de la tabla: suscripcion

Nombre: suscripción		
Descripción: Almacena todas las suscripciones.		
Atributo	**Tipo**	**Descripción**
idsuscripcion	int	Identificador de la tabla
fecha	string	Fecha en que fue realizada la suscripción
habilitada	string	Si la suscripción se mantiene activa (que el usuario no se haya dado de baja) está

		habilitada

Tabla 2.7 Descripción de la tabla: campo_valor

Nombre: campo_valor		
Descripción: Almacena todos los valores pertenecientes a los campos de los servicios		
Atributo	**Tipo**	**Descripción**
idcampovalor	int	Identificador de la tabla
valor	string	Valores de los campos
habilitado	bool	Si la suscripción a la que pertenece el campo_valor está habilitada, el campo_valor está habilitado.

Tabla 2.8 Descripción de la tabla: cliente_de_servicio

Nombre: cliente_de_servicio		
Descripción: Almacena los clientes que consumen los servicios web		
Atributo	**Tipo**	**Descripción**
idclienteservicio	int	Identificador de la tabla
usuario	string	Nombre del usuario
contrasenna	string	Contraseñas pertenecientes a cada uno de los clientes de Servicio.

3.6 Definiciones de diseño que se apliquen

3.6.1 Arquitectura

Arquitectura en Capas

Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas. [32]

La arquitectura utilizada en el Módulo de Suscripciones es la arquitectura en 3 capas. La calidad tan especial de la arquitectura de tres capas consiste en aislar la lógica de la aplicación y convertirla en una capa intermedia bien definida y lógica del software.

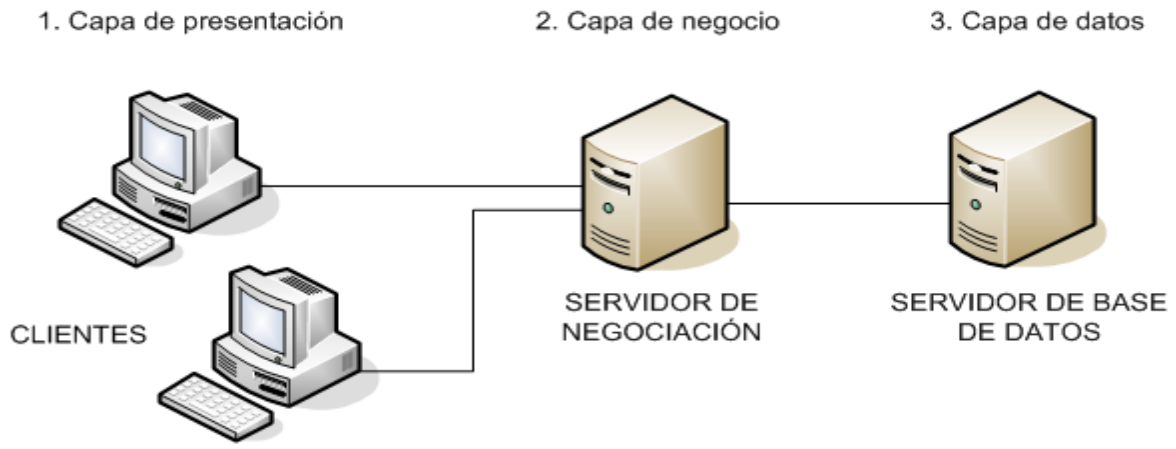

Figura 45. Arquitectura de 3 capas

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que ocurra algún cambio, sólo se afecta al nivel requerido sin tener que revisar entre código mezclado. [33]

Como se expuso anteriormente se utiliza la arquitectura en 3 capas. Para la aplicación web se tiene una capa de presentación, una de negocio y una de acceso a datos. Para los servicios web se tiene una capa de servicios, una de negocio y una de acceso a datos.

- Capa de Presentación de la aplicación web: es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario

en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Las clases fundamentales de esta capa de presentación son los controladores y validadores. En esta capa se utiliza el módulo de Spring Modelo-Vista-Controlador (MVC).

-Capa de Servicios de los servicios web: Esta capa es la encargada de exponer los servicios web y atender las peticiones hechas a estos. Las clases principales de esta capa son los Endpoint, las cuales son el concepto central de Spring Web Services del lado del servidor ya que se encargan de manejar las peticiones XML, invocan al método correspondiente en la capa de negocio y el resultado de esa invocación es la respuesta del mensaje.

- Capa de Negocio (La descripción de esta capa es común tanto para la aplicación web como para los servicios web): Es donde se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación o servicio, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

- Capa de Acceso a Datos (La descripción de esta capa es común tanto para la aplicación web como para los servicios web): Contiene las clases que se encargan de acceder a la base de datos cuando recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Patrón Modelo-Vista-Controlador

Modelo Vista Controlador (MVC)

En la mayoría de las aplicaciones web la lógica de un interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio. Con el modelo vista controlador se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

Elementos del patrón:

Modelo: Datos y reglas de negocio

Vista: Muestra la información del modelo al usuario

Controlador: Gestiona las entradas del usuario

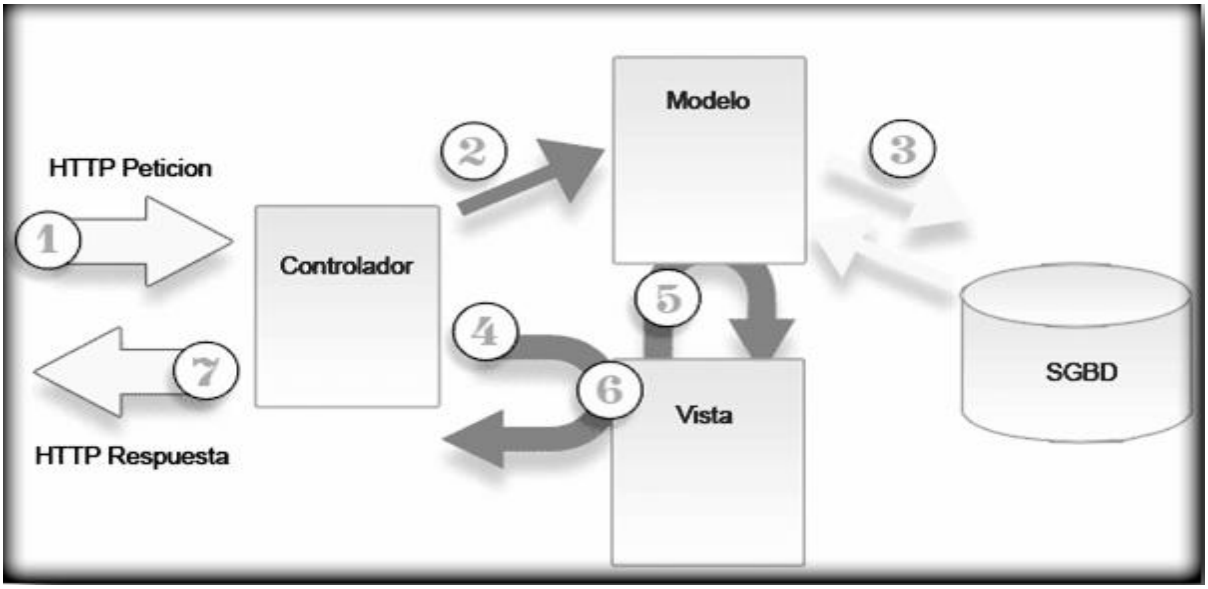

Figura 46. Modelo-Vista-Controlador

Un modelo puede tener diversas vistas, cada una con su correspondiente controlador.

1. El modelo es el responsable de:

Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.

Definir las reglas de negocio (la funcionalidad del sistema).

Llevar un registro de las vistas y controladores del sistema.

2. El controlador es responsable de:

Recibir los eventos de entrada (un clic, un cambio en un campo de texto, etc.).

Contener reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada a actualizar.

3. Las vistas son responsables de:

Recibir datos del modelo y mostrarlos al usuario.

Tener un registro de su controlador asociado.

Poder dar el servicio de actualizar, para que sea invocado por el controlador o por el modelo. [34]

3.6.2 Patrones de diseño

Son directrices y principios estructurados que describen un problema común y entregan una buena solución ya probada a la que le dan un nombre. Estos ayudan a diseñar correctamente en menos tiempo, ayudan a construir problemas reutilizables y facilitan la documentación. [35]

Patrones GRASP (General Responsibility Assignment Software Patterns)

Los patrones GRASP permiten describir los principios fundamentales de asignación de responsabilidades a objetos. Estos patrones son esenciales en el diseño eficaz de un software. Por las características de estos patrones, se consideró necesaria su utilización en este trabajo para obtener un producto de mayor calidad.

Controlador

Este patrón GRASP establece la asignación de la responsabilidad de manejar los eventos a una clase que será la encargada de controlarlos. Permite la reutilización de componentes al delegar las responsabilidades en una clase, lo que facilita que aunque existan cambios desde el punto de vista de la interfaz del sistema las funcionalidades se mantengan intactas y puedan ser reutilizadas. Aplicado en el sistema, en la aplicación web este patrón se ve reflejado en las clases controladoras, las cuales son las encargadas de manejar las peticiones del cliente, en el caso de los servicios web los controladores serían los Endpoint, encargados de manejar las peticiones XML.

Experto

Este patrón de diseño define asignar la responsabilidad a la clase que cuente con la información necesaria para manejar esta responsabilidad. Es necesario que la asignación de responsabilidades a las clases se lleve a cabo de una forma adecuada, lo que permitirá que el sistema sea más fácil de entender así como que se puedan reutilizar componentes. Con su aplicación se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. El comportamiento se distribuye entre las clases que cuentan con la información requerida permitiendo la creación de clases sencillas y más cohesivas. Tanto en la aplicación web como en los servicios web este patrón se ve reflejado en las clases del negocio y en las clases de acceso a datos, donde cada una realiza funciones específicas.

Alta Cohesión

Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un gran trabajo. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. En el sistema se tiene bien definido las responsabilidades de cada clase, por ejemplo, se definen clases del negocio por cada caso de uso, para evitar la sobrecarga de funcionalidades en una clase.

Bajo Acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Un alto acoplamiento tendría como consecuencia que las clases sean más difíciles de entender cuando estén aisladas, más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen. Por lo que un bajo acoplamiento plantea que debe haber pocas dependencias entre las clases. Spring plantea el uso de interfaces para evitar un alto acoplamiento, por lo que en el sistema este patrón se ve claramente reflejado, ya que tanto la capa de negocio como la capa de acceso a datos tienen interfaces que son implementadas por otras clases. [35]

Patrón Inversión de Control (IoC) / Inyección de Dependencia (DI)

Inversión de Control es un principio que generalmente siguen los framework, en lugar de tener las clases de la aplicación el control sobre la creación de objetos, le asigna esta responsabilidad a un agente externo como un contenedor. Spring proporciona un contenedor que maneja el ciclo de vida de los objetos y resuelve las dependencias entre dichos objetos. Se suelen confundir los términos de Inversión de Control e Inyección de Dependencia, pero no son lo mismo. Mientras que la Inversión de Control es un principio de diseño general, la Inyección de Dependencia es un patrón de diseño concreto que encarna este principio. La Inyección de Dependencia es la más típica realización de la Inversión de Control.

El patrón Inyección de Dependencia, es una técnica que busca facilitar la resolución de dependencias entre objetos. Consiste en inyectar objetos a una clase en lugar de ser la propia clase quien cree el objeto. La forma habitual de implementar este patrón es mediante un contenedor. El contenedor inyecta a cada objeto los objetos necesarios según las relaciones plasmadas en un fichero de configuración. Típicamente este contenedor es implementado por un framework externo a la aplicación (como Spring). El uso de este patrón permite construir aplicaciones mucho más flexibles y robustas. [36]

3.6.3 Patrones JEE

Los patrones JEE describen típicos problemas encontrados por desarrolladores de aplicaciones empresariales y proveen soluciones para estos problemas.

Patrón Front-Controller

Para aplicaciones web se recomienda utilizar este patrón ya que obliga a que todas las peticiones hechas a la aplicación pasen por un servlet Controlador. El controlador proporciona un punto de entrada único que controla y gestiona las peticiones web realizadas por los clientes, incluyendo la invocación de los servicios de seguridad como la autentificación y autorización, delegar el procesamiento de negocio, controlar la elección de una vista apropiada, el manejo de errores y el control de la selección de estrategias de creación de contenido. Teniendo este único punto de entrada se evita tener que repetir la misma lógica de control en todos los .jsp. Normalmente se utiliza junto con un Dispatcher que es el responsable de redirigir el flujo de ejecución hacia el jsp adecuado.

Centralizar el control en el controlador y reducir la lógica de negocios en la vista permite reutilizar el código entre peticiones. Es una aproximación preferible a la alternativa de embeber código en varias vistas porque esta aproximación trata con entornos más propensos a errores y de reutilización del tipo copiar-y-pegar. Aunque el patrón Front Controller sugiere la centralización del manejo de peticiones, no limita el número de manejadores en el sistema, como lo hace Singleton. Una aplicación podría utilizar varios controladores en un sistema, cada uno mapeado a un conjunto de servicios distintos. [37]

Patrón DAO (Data Access Object)

Es bastante normal hacer aplicaciones que almacenan y recogen datos de una base de datos. Suele ser habitual, también, querer hacer la aplicación lo más independiente posible de una base de datos concreta, de cómo se accede a los datos o incluso de si hay o no base de datos detrás. Hay una forma de hacer esto que ha resultado bastante eficiente en el mundo *JEE* y de aplicaciones web, pero que es aplicable a cualquier tipo de aplicación que deba recoger datos de algún sitio y almacenarlos. Es lo que se conoce como *patrón DAO* (**D**ata **A**ccess **O**bject).

El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos. Los componentes de negocio que tratan con el DAO utilizan una interfaz simple expuesta por este para sus clientes. El DAO oculta completamente los detalles de implementación del acceso a datos a sus clientes. Como la interfaz expuesta por el DAO no cambia cuando cambia la implementación de la fuente de datos subyacente, este

patrón permite al sistema adaptarse a diferentes esquemas de almacenamiento sin que esto afecte a los componentes de negocio. Esencialmente, el DAO actúa como un adaptador entre las clases del negocio y la fuente de datos, desacoplando la lógica de negocio de la lógica de acceso a datos. [37]

3.6.4 Seguridad

Un tema muy importante a tratar en la construcción de los servicios web es el referente a la seguridad, debido a que todas las funciones de estos servicios son públicas.

Para garantizar la seguridad se utilizó WS-Security (Seguridad en Servicios Web), el cuál es un estándar que suministra un medio para aplicar seguridad a los servicios web. Esta especificación define una serie de extensiones para el protocolo SOAP, que permiten el intercambio entre cliente y servidor de identificadores ("Tokens") de seguridad.

Cuando la aplicación cliente intente consumir los servicios, esta debe enviar sus credenciales (nombre de usuario y contraseña). Esta medida de seguridad implica el uso de SOAP y XML para enviar información de autenticación como parte de los comandos de servicio. Debido a que la información se envía directamente al servicio web, este puede implementar la autenticación que sea necesaria. Por ejemplo, se podría acceder a una base de datos para determinar la identificación adecuada de una aplicación cliente.

Para enviar la información de identificación cuando se intente consumir un servicio web, se usará un encabezado SOAP. Este es un mensaje XML adicional adjunto a la comunicación regular del servicio que ofrece información de acuerdo con el usuario. Aunque se transporta con los mensajes SOAP, es una entidad por separado.

El framework utilizado para exponer los servicios web, Spring Web Service, brinda soporte a dicho estándar.

3.6.5 Interfaz

La aplicación web que permite la gestión de los servicios y los administradores, contará con una interfaz sencilla que sea fácil de usar por los administradores.

Conclusiones

En este capítulo se expusieron los principales artefactos del análisis y diseño del sistema propuesto como solución, en específico los diagramas de clases de análisis y diseño y los diagramas de interacción. Además se describieron los patrones utilizados.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

Introducción

En este capítulo se lleva a cabo la descripción de las principales tareas del flujo de implementación y de prueba, además de los artefactos que se generan en estos flujos.

4.1 Modelo de Implementación

El modelo de despliegue se crea durante las últimas actividades del flujo de trabajo de diseño y provee la descripción de la distribución física del sistema. Se utiliza como entrada fundamental en las actividades de diseño e implementación. El diagrama de componentes, por su parte, estructura el modelo de implementación y muestra las relaciones entre los elementos de implementación. Ambos diagramas conforman el modelo de implementación, que describe como los elementos del diseño se implementan en términos de componentes, ficheros de código fuente, ejecutables, etc.

4.1.2 Diagrama de despliegue

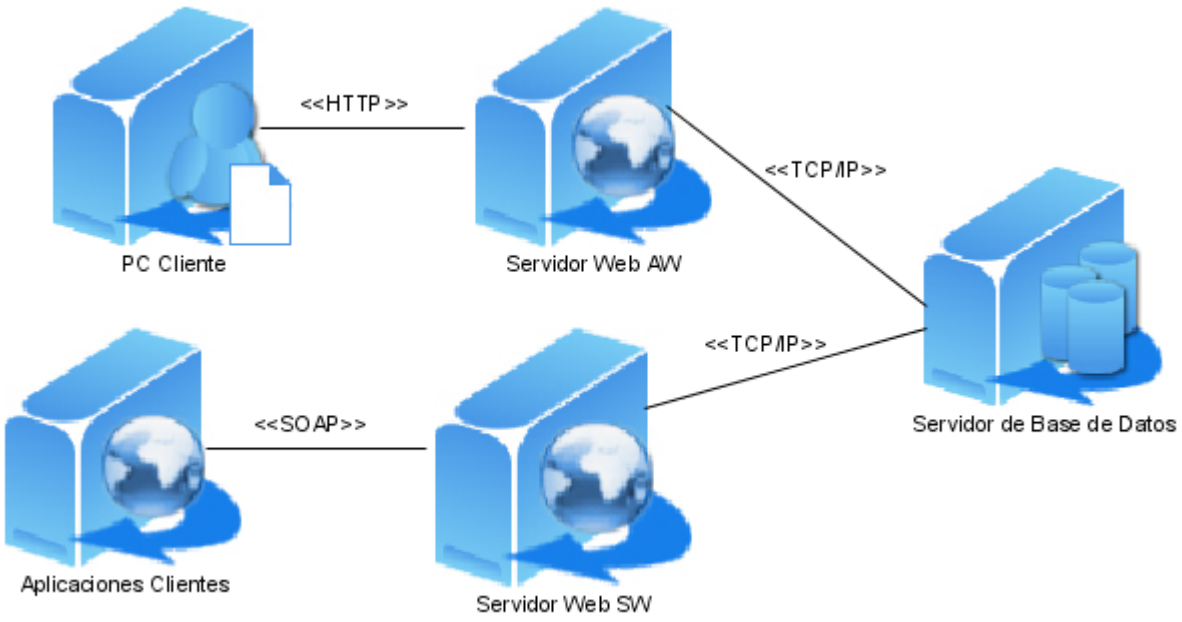

Figura 47. Diagrama de Despliegue: Aplicación Web y Servicios Web

En el nodo PC Cliente se puede acceder a la aplicación web mediante un navegador.

El nodo Aplicaciones Clientes representa las aplicaciones que van a consumir los servicios web.

En el nodo Servidor Web AW estará el servidor Web Apache Tomcat, donde se ejecutará la aplicación web.

En el nodo Servidor Web SW estará el servidor Web Apache Tomcat, donde se ejecutarán los dos servicios web.

En el nodo servidor de base de datos, se utilizará el Sistema gestor de Base de Datos Postgres SQL para el acceso a la base de datos, tanto la aplicación como los servicios web utilizan la misma base de datos.

4.1.3 Diagrama de componentes

Los diagramas de componentes se utilizan para modelar la vista de implementación estática de un sistema. Muestran tanto los componentes de software (código fuente, binario y ejecutable) como las relaciones lógicas entre ellos en un sistema. Y como todos los diagramas, también pueden contener paquetes utilizados para agrupar elementos del modelo.

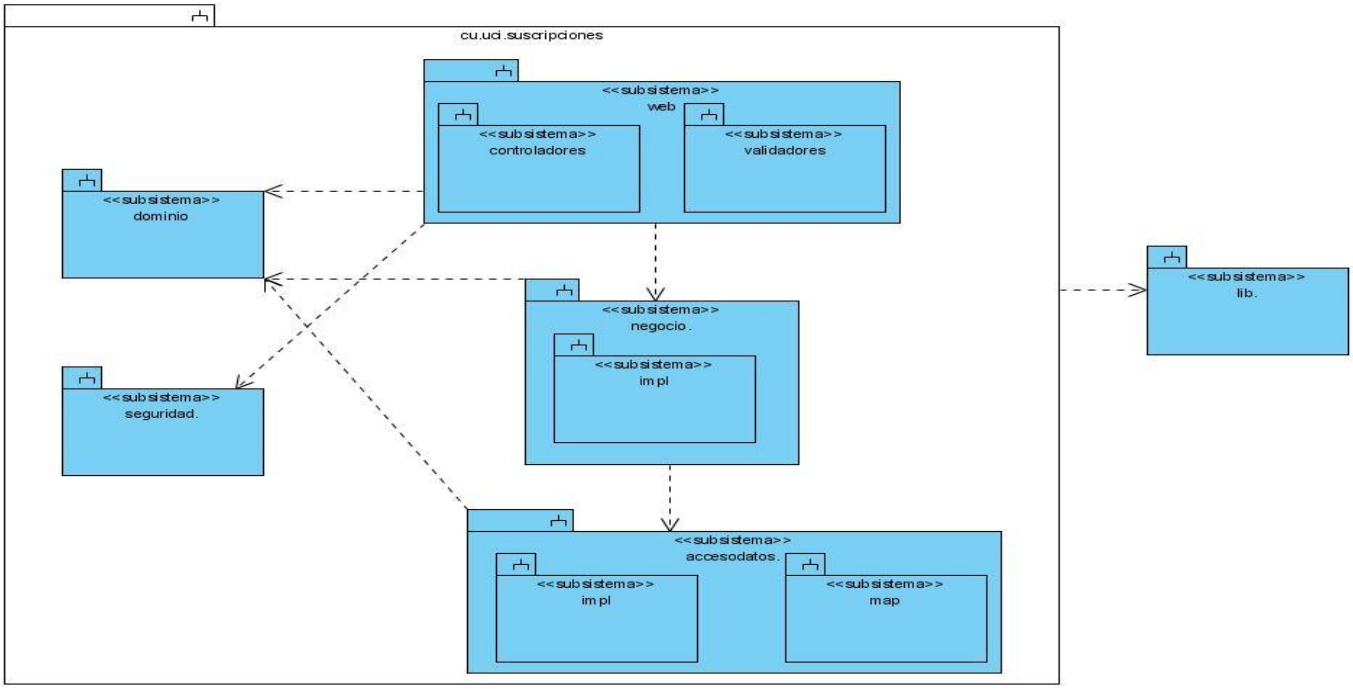

Figura 48. Diagrama de Componente General: Aplicación Web

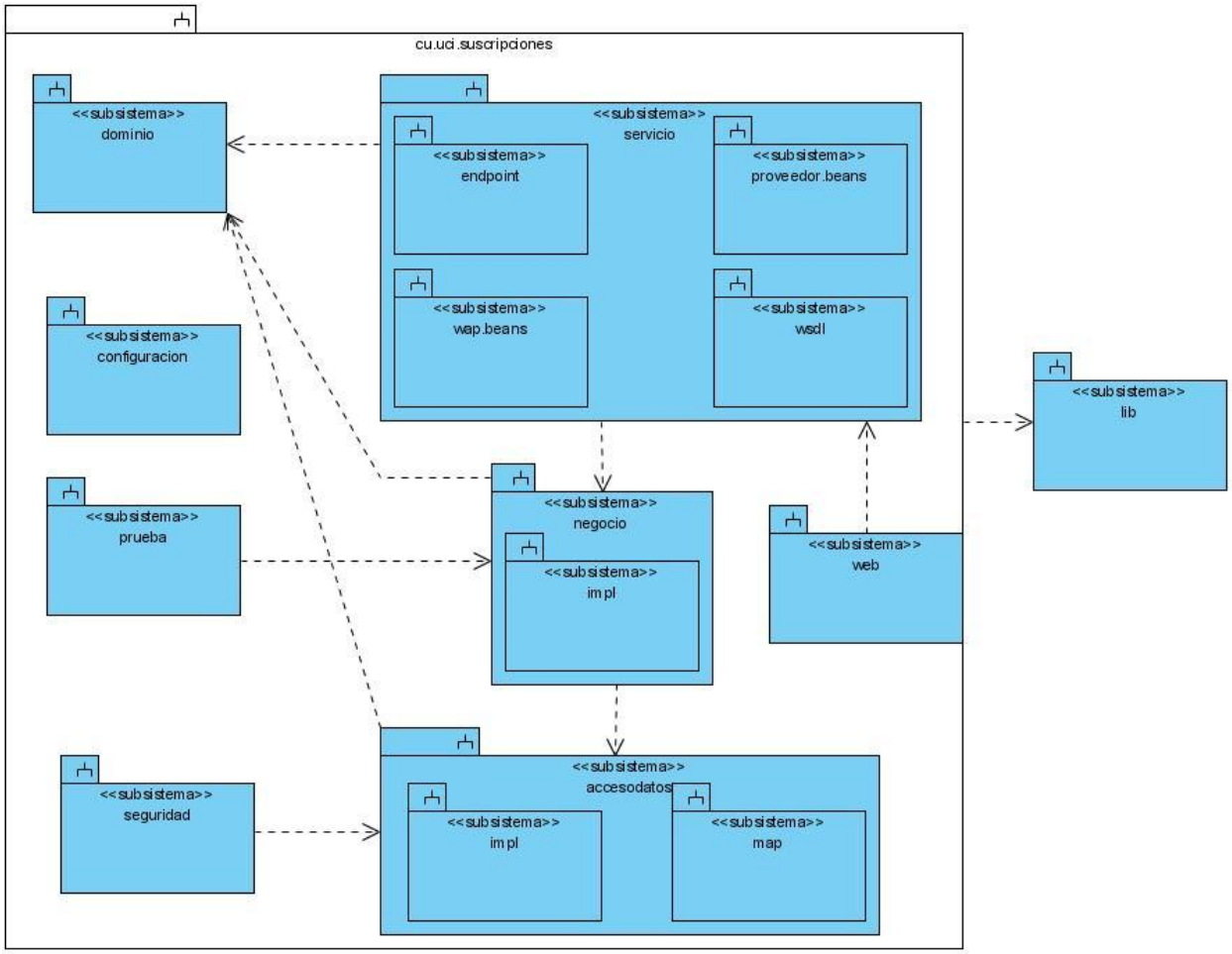

Figura 49. Diagrama de Componente General: Servicios Web

En la siguiente figura se representan los componentes que constituyen la capa de servicio de los servicios web. Esta capa contiene 4 subsistemas: endpoint, proveedor.beans, wap.beans y wsdl. El subsistema endpoint contiene varios subsistemas en los cuales están los componentes que representan a las clases Endpoint. El subsistema proveedor.beans y el subsistema wap.beans están integrados por los componentes que representan a las clases generadas a partir del wsdl. El subsistema wsdl está compuesto por los componentes que conforman los respectivos wsdl de cada servicio: Suscripciones_Proveedor y Suscripciones_WAP.

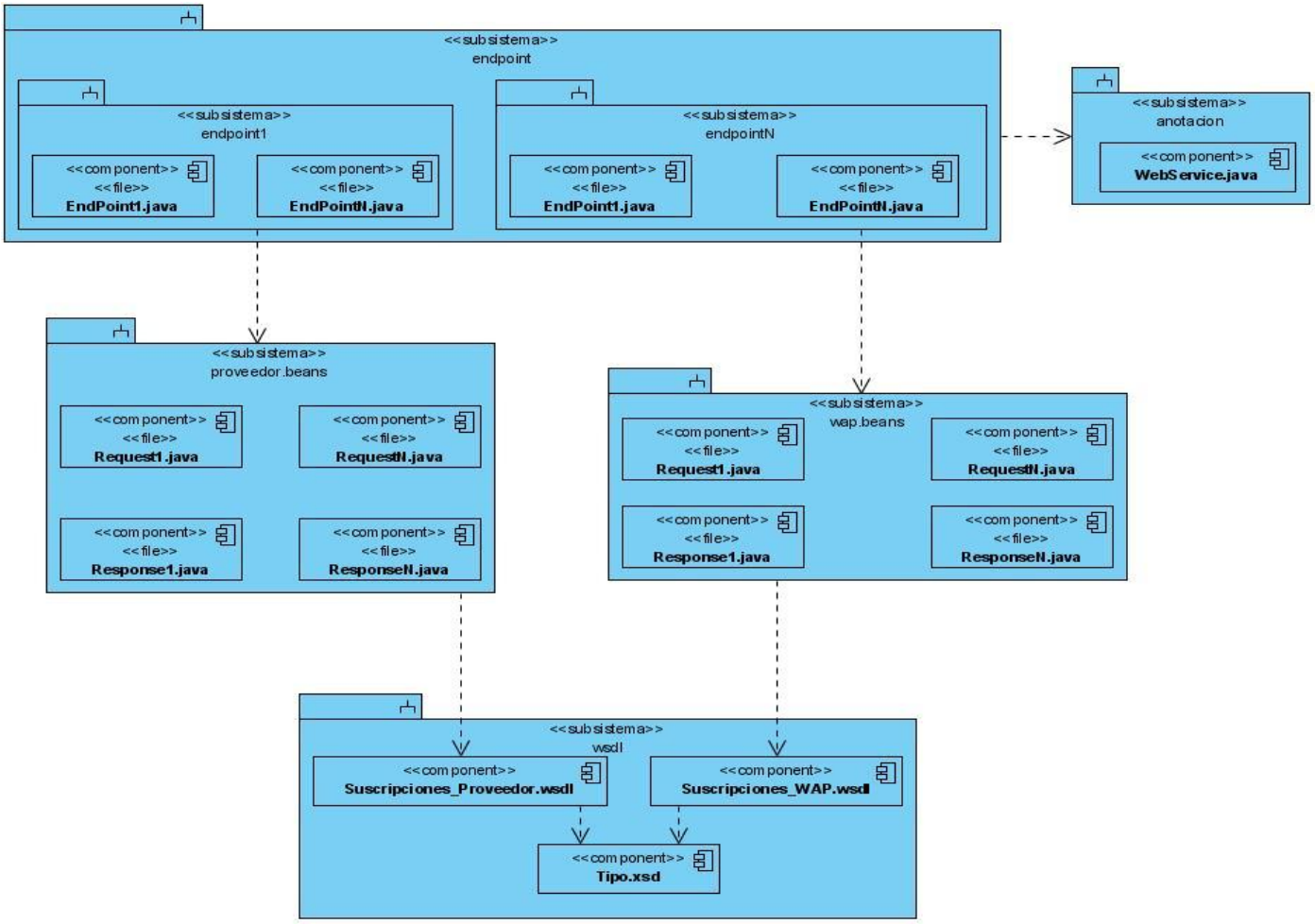

Figura 50. Diagrama de Componente (Capa de Servicio): Servicios Web

En la siguiente figura se representan los componentes que constituyen la capa de presentación de la aplicación web. Esta capa contiene 2 subsistemas: controladores y validadores. El subsistema controlador contiene controladores de Spring, que son las clases que manejan las peticiones del cliente. El subsistema validadores contiene las clases que se encargan de validar los datos provenientes del cliente.

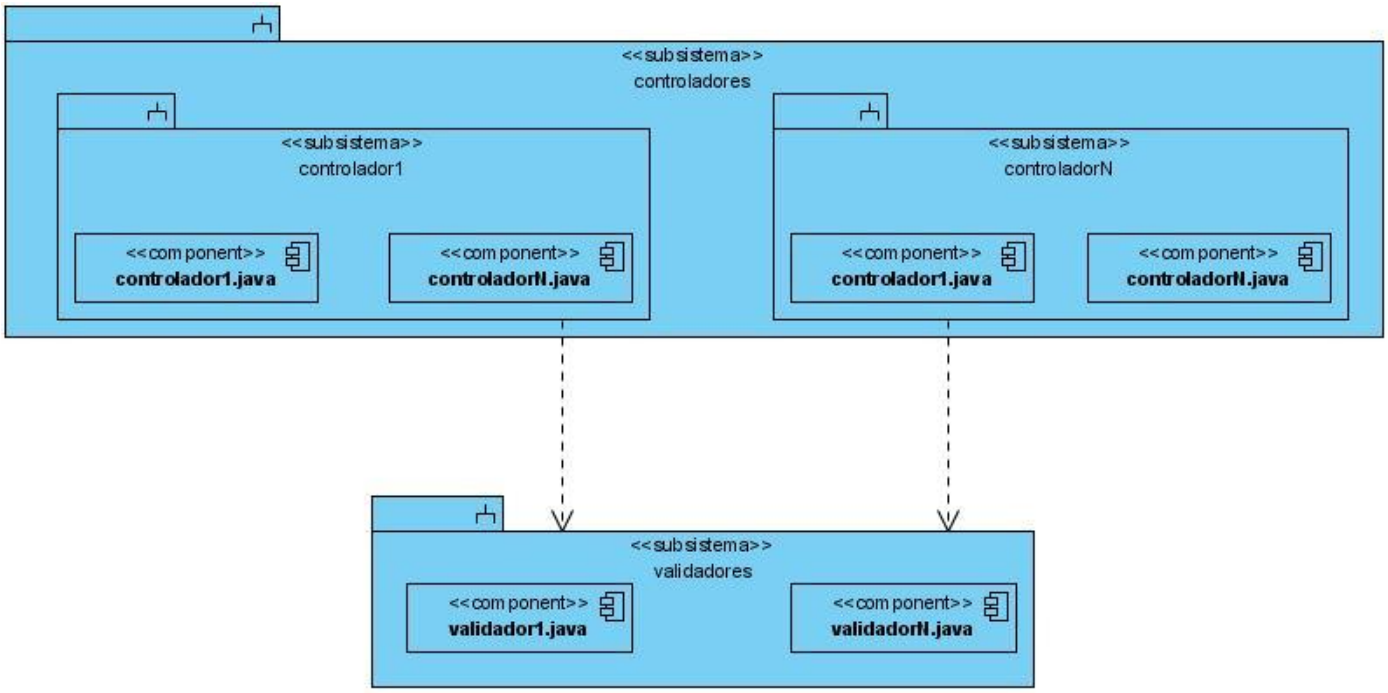

Figura 51. Diagrama de Componente (Capa de Presentación): Aplicación Web

El diagrama de componentes del subsistema de la capa del negocio está formado por las interfaces de las clases del negocio y por un subsistema que tiene un componente por cada clase del negocio con la implementación de cada una de ellas.

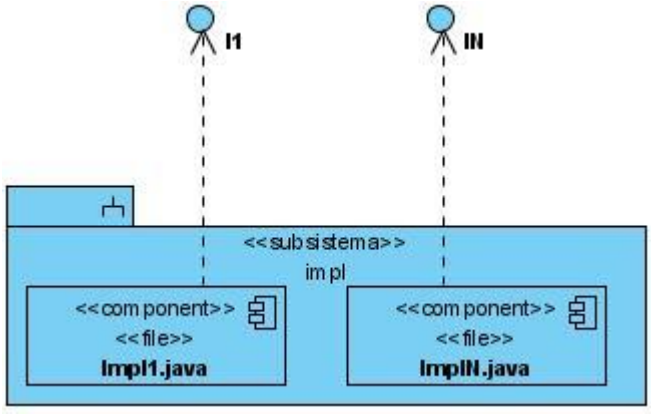

Figura 52. Diagrama de Componente (Capa de Negocio): Aplicación Web y Servicios Web

El diagrama de componentes del subsistema de la capa de acceso a datos está formado por las interfaces de las clases de acceso a datos y dos subsistemas: impl y map. El subsistema impl tiene un componente por cada clase de acceso a datos con la implementación de cada una de ellas. El subsistema map contiene un componente por cada clase del dominio, donde se realiza el mapeo de objetos a la base de datos y las consultas sql que serán utilizadas por las clases de acceso a datos.

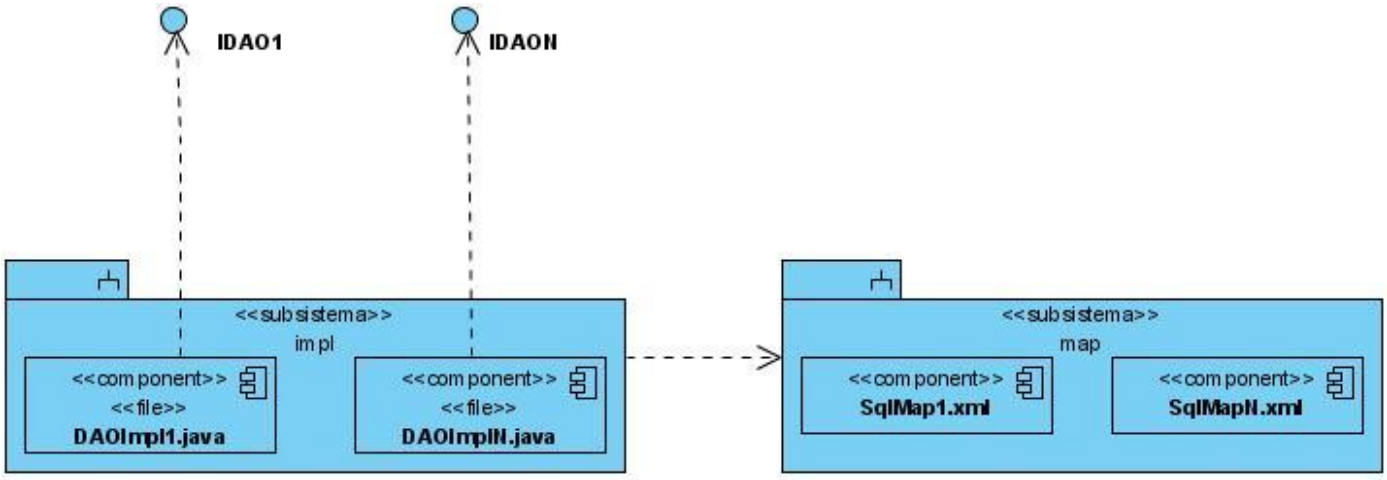

Figura 53. Diagrama de Componente (Capa de Acceso a Datos): Aplicación Web y Servicios Web

4.2 Modelo de Prueba

4.2.1 JUnit

JUnit provee al usuario de herramientas, clases y métodos que le facilitan la tarea de realizar pruebas en el sistema y así asegurar su consistencia y funcionalidad. Existen principalmente 2 tipos de pruebas que implementa JUnit, estas son:

Pruebas unitarias: Consisten en probar la correcta funcionalidad del módulo en cuestión como si actuara independiente de los demás.

Pruebas de integración: Como su nombre indica, se prueba la correcta integración de cada módulo (ya probado con pruebas unitarias) con los demás.

Para esta investigación se utilizarán las pruebas unitarias ya que su objetivo fundamental es aislar cada parte del programa y mostrar que las partes individuales son correctas. Particularmente se utilizó JUnit para comprobar el correcto funcionamiento de métodos del negocio y de acceso a datos. [20]

A continuación se muestra un ejemplo de prueba utilizando JUnit, llevándose a cabo satisfactoriamente.

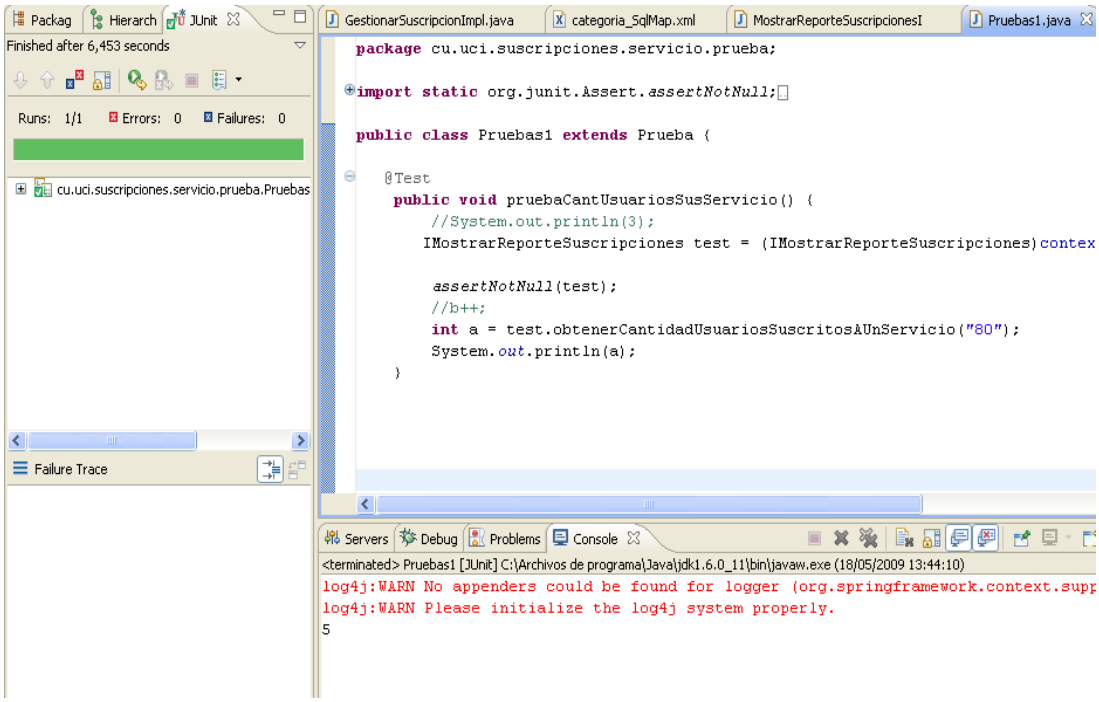

Figura 54. Caso de Prueba con JUnit: Cantidad de usuarios suscritos a un servicio

4.2.2 SoapUI

Para probar los servicios web a nivel de interfaz, se utilizó SoapUI, el cual funciona como un cliente de los servicios y permite visualizar los diferentes errores que puedan tener a nivel general.

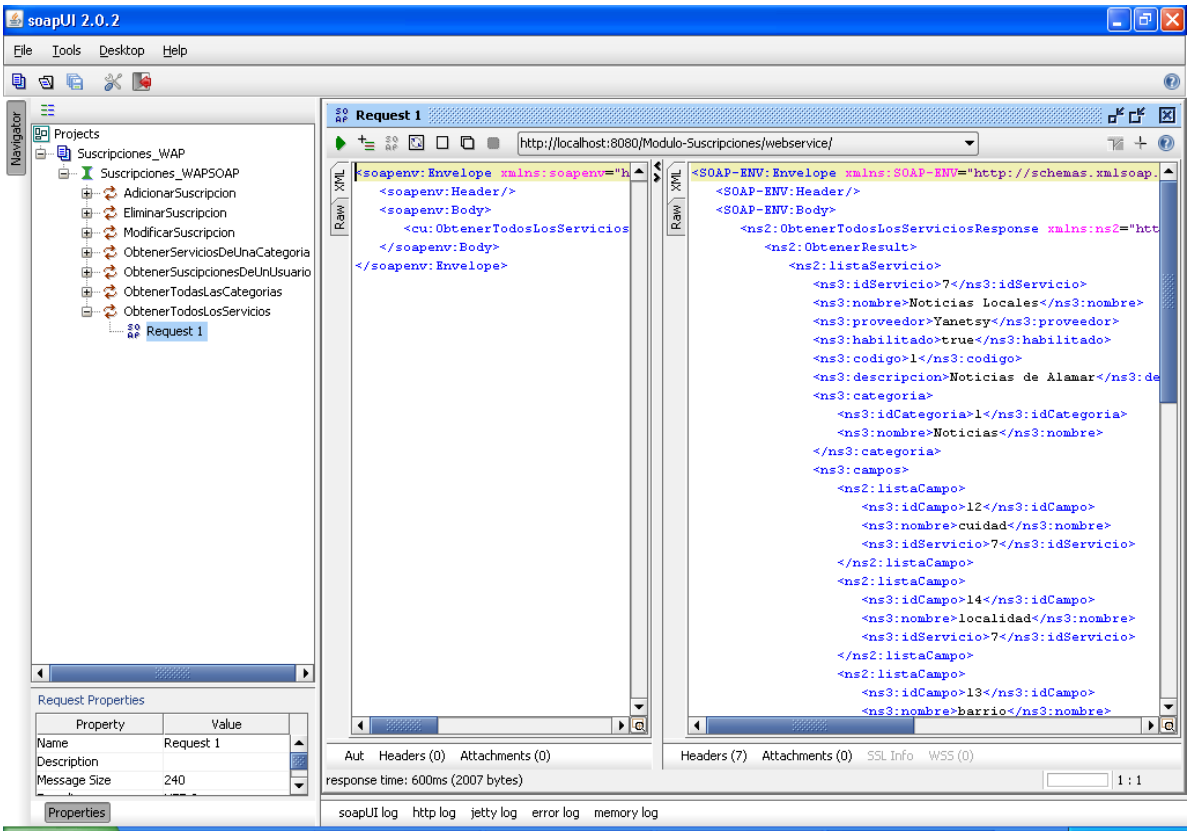

Figura 55. Caso de Prueba con SoapUI: Todos los servicios

Conclusiones

Se finaliza la etapa de Implementación del sistema, haciéndose los modelos de despliegue y los diagramas de componentes. También se concluye así con la etapa de prueba del sistema.

CAPÍTULO 5. ESTUDIO DE LA FACTIBILIDAD

Introducción

Para el estudio de la factibilidad existen varios métodos: COCOMO, COCOMO II, Análisis de Punto de función, Análisis de Puntos de Casos de Uso, entre otros.

La estimación a partir de Puntos de Función es difícil de realizar si no se cuenta con una base histórica de proyectos que provea los coeficientes de conversión. Además solo se aplica a proyectos extremadamente grandes, con varios casos de uso.

La estimación por COCOMO II (con Puntos de Función sin ajustar como entrada), resulta muy útil para estimar un proyecto en forma global, cuando se tiene un conjunto de Casos de Uso bastante amplio (del orden de 50) y con escaso nivel de detalle. Cabe aclarar que la estimación por COCOMO no está calibrada para proyectos menores a 2000 líneas de código, con lo cual no es aplicable a proyectos pequeños.

La estimación por Puntos de Caso de Uso resulta muy efectiva para estimar el esfuerzo requerido en el desarrollo de los primeros Casos de Uso de un sistema, si se sigue una aproximación iterativa como el Proceso Unificado de Rational. En éste tipo de aproximación, los primeros Casos de Uso a desarrollar son los que ejercitan la mayor parte de la arquitectura del software y los que a su vez ayudan a mitigar los riesgos más significativos. Además permite estimar el proyecto antes de empezar la etapa de implementación. [38]

En este capítulo se realizará mediante el método de Puntos de Caso de Uso una estimación del esfuerzo, costo y tiempo de desarrollo del proyecto. Además se hará un estudio de los costos y beneficios del mismo para llegar a la conclusión si el sistema es factible o no.

5.1 Planificación

5.1.1 Cálculo de Puntos de Casos de Usos sin ajustar

El cálculo de los Puntos de Caso de Uso sin ajustar constituye el primer paso y se calcula a partir de la siguiente ecuación:

UUCP = UAW + UUCW

Donde:

UUCP: Puntos de Casos de Uso sin ajustar.

UAW: Factor de Peso de los Actores sin ajustar.

UUCW: Factor de Peso de los Casos de Uso sin ajustar.

Para obtener el Factor de Peso de los Actores sin ajustar se debe tener en cuenta la cantidad de actores presentes en el sistema y la complejidad de los mismos. Los criterios se muestran en la siguiente tabla:

Tipo de Actor	**Descripción**	**Peso**	**Cantidad * Peso**
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación.	1	0 * 1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2	2* 2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3	1 * 3
Total			7

Para obtener el valor del Factor de Peso de los Casos de Uso sin ajustar se debe tener en cuenta la cantidad de casos de uso y la complejidad de cada uno de ellos. Los criterios se muestran en la siguiente tabla:

Tipo de Caso de Uso	Descripción	Peso	Cantidad * Peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones.	5	1 * 5
Medio	El Caso de Uso contiene de 4 a 7 transacciones.	10	7 * 10
Complejo	El Caso de Uso contiene más de 8 transacciones.	15	0 * 15
Total			75

Por lo que los Puntos de Caso de Uso sin ajustar resultan:

UUCP= UAW + UUCW = 7+75 = 82

5.1.2 Cálculo de Puntos de Casos de Uso ajustados

Una vez obtenidos los Puntos de Caso de Uso sin ajustar se procede al cálculo de los Puntos de Casos de Uso ajustados mediante la siguiente ecuación:

UCP = UUCP * TCF * EF

Donde:

UCP: Puntos de Casos de Uso ajustados.

UUCP: Puntos de Casos de Uso sin ajustar.

TCF: Factor de complejidad técnica.

EF: Factor de ambiente.

Factor de complejidad técnica: Este coeficiente se calcula mediante un conjunto de Factores que determinan la complejidad del sistema, cada factor se cuantifica con un valor de 0 a 5.

Significado de los valores:

0: No presente o sin influencia.

1: Influencia incidental o presencia incidental.

2: Influencia moderada o presencia moderada.

3: Influencia media o presencia media.

4: Influencia significativa o presencia significativa.

5: Fuerte influencia o fuerte presencia.

La ecuación para su cálculo es:

TCF = 0.6 + 0.01 * Σ (Peso i * Valor Asignado i)

Factor	**Descripción**	**Peso**	**Peso * Valor**
T1	Sistema distribuido.	2	2 * 5
T2	Objetivos de performance o tiempo de respuesta.	1	1 * 3
T3	Eficiencia del usuario final.	1	1 * 4
T4	Procesamiento interno complejo.	1	1 * 3

T5	El código debe ser reutilizable.	1	1 * 4
T6	Facilidad de instalación.	0.5	0.5 * 3
T7	Facilidad de uso.	0.5	0.5 * 5
T8	Portabilidad.	2	2 * 5
T9	Facilidad de cambio.	1	1 * 3
T10	Concurrencia.	1	1 * 4
T11	Incluye objetivos especiales de seguridad.	1	1 * 4
T12	Provee acceso directo a terceras partes.	1	1 * 3
T13	Se requieren facilidades especiales de entrenamiento a los usuarios.	1	1 * 2
Total			54

Entonces, **TCF** = 0.6 + 0.01 * 54 = **1.14**

Factor de ambiente: Contempla las habilidades y el entrenamiento del grupo de desarrollo por su importancia en las estimaciones de tiempo. Al igual que el factor de complejidad técnica se cuantifican con valores de 0 a 5. La ecuación para su cálculo es:

EF = 1.4 - 0.03 * Σ (Peso i * Valor Asignado i).

Factor	**Descripción**	**Peso**	**Peso * Valor**
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	1.5 * 2
E2	Experiencia en la aplicación.	0.5	0.5 * 2
E3	Experiencia en orientación a objetos.	1	1 * 4
E4	Capacidad del analista líder.	0.5	0.5 * 3
E5	Motivación.	1	1 * 5
E6	Estabilidad de los requerimientos.	2	2 * 3
E7	Personal part - time.	-1	-1 * 3

E8	Dificultad del lenguaje de programación.	-1	-1 * 3
Total			14.5

Entonces, EF = 1.4 - 0.03 * 14.5 = 0.97

Finalmente, UCP (Puntos de Caso de Uso ajustados) = UUCP * TCF * EF = 82 * 1.14* 0.97= 90.68

5.1.3 Estimación del esfuerzo

El esfuerzo en horas - hombre viene dado por:

E = UCP * CF

Donde:

E: Esfuerzo estimado en horas-hombre.

UCP: Puntos de Casos de Uso ajustados.

CF: Factor de conversión.

CF = 20 horas-hombre (si Total EF ≤ 2)

CF = 28 horas-hombre (si Total EF = 3 ó Total EF = 4)

CF = abandonar o cambiar proyecto (si Total EF ≥ 5)

Total EF = Cant EF < 3 (entre E1 –E6) + Cant EF > 3 (entre E7, E8)

Como Total EF = 2 + 0

Total EF = 2

CF = 20 horas-hombre (porque Total EF ≤ 2)

Luego E = 90.68 * 20 horas-hombre

E = 1813.5 horas-hombre

5.1.4 Distribución del Esfuerzo entre las diferentes actividades.

Para llevar a cabo una estimación más completa de la duración total del módulo, se agrega el esfuerzo de las demás actividades. Para ello se plantea la distribución del esfuerzo entre las diferentes actividades:

Actividad	**Porcentaje**	**Horas-Hombre**

Análisis	10%	453.38
Diseño	20%	906.75
Implementación	40%	1813.5
Pruebas	15%	680.06
Otras actividades	15%	680.06
Total	100%	4533.75

El Esfuerzo Total sería 4533.75 horas-hombre, si estimamos teniendo en cuenta que un mes tiene 176 horas laborables, pues se trabajan 8 horas diarias 22 días al mes, entonces el Esfuerzo Total en mes-hombre sería 25.76 mes-hombre.

5.1.5 Calcular el costo de todo el proyecto.

Costo = CHM * ET / CH

Donde:

CH: Cantidad de hombres.

CHM: Costo Hombre – Mes.

ET: Esfuerzo Total.

Si la Cantidad de hombres es 2 y se tiene un Salario Promedio mensual igual a $100.00.

Entonces CHM = CH * Salario Promedio

CHM = 2 * 100

CHM = 200.00

Luego Costo = CHM * ET / CH

Costo = 200.00 * 25.76 / 2

Costo = $ 2576

5.1.6 Calcular el tiempo de desarrollo de todo el proyecto.

A partir de los datos conocidos anteriormente se puede decir que una persona puede realizar el trabajo en un tiempo aproximado de 25.76 meses. Pero teniendo en cuenta que la cantidad de hombres para la realización del Módulo Suscripciones es 2 el tiempo va a ser:

Tiempo = ET / CH

Tiempo = 25.76 / 2

Tiempo = 12.88 ≈ 13 meses

De los resultados obtenidos se interpreta que con 2 hombres trabajando en el proyecto el mismo se desarrolla en 13 meses y su costo total se estima que sea $2576.

5.2 Beneficios Tangibles e Intangibles

La integración del módulo de Suscripciones con el Portal WAP y el sistema Proveedor de Servicio de la Plataforma de Servicios de Valor Agregado pretende ofrecer a los clientes del operador celular CUBACEL un nuevo servicio: suscripciones. Actualmente los servicios de suscripciones son un paso de avance en la telefonía celular ya que se ofrecen como servicios de valor agregado, obteniéndose ganancias y fomentando la popularidad que tienen estos dispositivos a nivel mundial, ya que brindan una serie de comodidades, por ejemplo, desde cualquier región del mundo se puede contar con los titulares del país de origen o las últimas informaciones de temas de interés.

Para la realización de dicho módulo se implementan dos servicios web y una aplicación web desarrollados sobre la plataforma Java, garantizando posibles migraciones de sistemas operativos o plataformas en un futuro, sin afectar a los beneficiarios.

5.3 Análisis de costos y beneficios

El análisis de los beneficios que aportará el desarrollo de un proyecto, constituye una ayuda importante en la toma de decisiones, ya que frecuentemente brinda la información necesaria para determinar si el producto a desarrollar es factible o no.

Con la realización de los Servicios y la aplicación web se reflejaron algunas ventajas que permitieron obtener un costo del proyecto no muy elevado, entre ellas: es un sistema fácil de usar, multiplataforma y que está soportado con tecnologías de software libre. Debido a estas características se considera que es factible la implementación del mismo.

Conclusiones

En este capítulo se llevó a cabo un estudio de la factibilidad del proyecto teniendo en cuenta el posible costo, el tiempo de duración y los posibles beneficios que podrá brindar.

CONCLUSIONES

Con el objetivo de darle cumplimiento a los objetivos generales y específicos de este trabajo se han llevado a cabo una serie de tareas que permitieron el buen desarrollo del sistema. Se llevó a cabo una previa investigación sobre la telefonía celular, reflejándose la situación actual de esta tecnología en el estado del arte. Se realizaron varios estudios para elegir las tecnologías, lenguaje de programación y metodología de desarrollo a utilizar donde se escogieron las que resultaron más adecuadas para el desarrollo del sistema. Se expusieron los conceptos fundamentales sobre servicios web. Se utilizó el lenguaje Java y como metodología RUP, para documentar el proceso de desarrollo del software y una arquitectura en capas utilizando el patrón modelo-vista-controlador. Se utilizaron diferentes patrones de diseño y patrones JEE que nos permitieron desarrollar un producto con calidad.

Como resultado de este trabajo se desarrollaron dos servicios web y una aplicación web cumpliendo con los requerimientos establecidos inicialmente. Por lo anteriormente dicho se concluye que se vencieron con éxitos los objetivos propuestos.

RECOMENDACIONES

Una vez terminada la implementación del Módulo de Suscripciones se recomienda:

- Tratar con mayor rigor el tema de la seguridad en los servicios web, mediante el uso de certificados digitales.
- Someter el sistema a pruebas de calidad, para validar el mismo para su posterior comercialización.
- Finalizar el desarrollo de los módulos que directamente intervienen en el funcionamiento del presente módulo, para la realización de pruebas sobre un entorno real, además de la comprobar la integración con estos.
- Incluir al módulo presente seguridad a nivel de transporte, para garantizar una mayor seguridad en el envío de datos.

REFERENCIAS BIBLIOGRÁFICAS

1. *Evolución de la tecnología celular.* **Martínez, Evelio.**

2. **ETECSA, Gerencia.** CUBACEL. [En línea] 2008. [Citado el: 22 de 10 de 2008.] http://www.cubacel.cu/.

3. **Alierta, César.** Movistar. [En línea] Telefónica S.A, 2007. [Citado el: 23 de 10 de 2008.] www.movistar.com.pa/informe2007.pdf..

4. **ARQHYS.com, Equipo Tecnología de Nokia.** Nokia. [En línea] 2008. [Citado el: 23 de 10 de 2008.] http://tecnologia.arqhys.com/moviles/nokia.html..

5. **Salazar-Simpson, Luis Alberto.** Fundación Orange. [En línea] 2008. [Citado el: 24 de 10 de 2008.] http://www.fundacionorange.es/areas/21_funda/funda_210.asp..

6. **Zander, Edward.** Motorola. [En línea] 2007. [Citado el: 24 de 10 de 2008.] http://www.motorola.com/..

7. *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* **Letelier Patricio, Penadés Mª Carmen.** 2008.

8. *Metodologías (Word) Northware.* 2007.

9. *Ayuda de la herramienta Case Rational Rose.* 2008.

10. **Booch Grady, Jacobson Ivar, Rumbaugh James.** *El Proceso Unificado de Desarrollo de Software.*

11. **Craig, Larman.** *UML y Patrones.* 1999.

12. Getting Started with UML. [En línea] 2007. [Citado el: 11 de 11 de 2008.] http://www.omg.org/UML/.

13. Java. *Breve historia de la tecnología Java.* [En línea] [Citado el: 15 de 11 de 2008.] http://www.java.com/es/about/.

14. **Cruz Martínez-Lacaci, Fernando.** El código abierto en el ámbito empresarial. [En línea] 2007. [Citado el: 16 de 11 de 2008.] http://www.mkm-pi.com/mkmpi.php?article56.

15. JEE. [En línea] [Citado el: 20 de 11 de 2008.] [http://www.epidataconsulting.com/tikiwiki/tiki-index.php?page=JEE].

16. **Jonhson, Rod.** Introduction to the Spring Framework. [En línea] 2005. [Citado el: 25 de 11 de 2008.] http://www.theserverside.com/tt/articles/article.tss?l=SpringFramework..

17. **Walls, Craig y Breidenbach, Ryan.** *Spring in Action.* 2005.

18. Spring Security. [En línea] [Citado el: 25 de 11 de 2008.] http://www.acegisecurity.org.

19. **Poutsma, A.E., Rick y Abed, Tareq.** *Spring Web Services – Reference.* 2005-2007.

20. **Massol, Vincent y Husted, Ted.** *JUnit in Action.* 2004.

21. **Clinton Begin, Brandon Goodin, Larry Meadors.** *IBATIS in Action.*

22. **Begin, Clinton.** *iBatis Data Mapper 2.0 Developer Guide.* 2006.

23. **Toledano, Moisés Daniel Díaz.** Web Services. Introducción y Escenarios para su Uso. [En línea] [Citado el: 3 de 12 de 2008.] http://www.moisesdaniel.com/es/wri/wsepsu.htm.

24. **Group, T.P.G.D.** Postgres. [En línea] [Citado el: 5 de 12 de 2008.] http://www.postgresql.org/docs/8.3/interactive/intro-whatis.html..

25. **Apache.** Tomcat – Introducción. [En línea] 2008. [Citado el: 8 de 12 de 2008.] http://www.programacion.com/tutorial/tomcatintro/1/#1_intro.

26. **Foundation, E.** Eclipse. [En línea] 2009. [Citado el: 16 de 1 de 2009.] http://www.eclipse.org/.

27. **Caballero, I.** Práctica de Ingeniería de Software 3. [En línea] [Citado el: 21 de 1 de 2009.] http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_VP.pdf..

28. **SoapUI.** SoapUI. [En línea] [Citado el: 25 de 1 de 2009.] http://www.soapui.org:80/.

29. **Jacobson, Ivar y Booch, Grady, Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* 2000.

30. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El lenguaje unificado de modelado. Manual de referencia.* 2000.

31. **Wesley**, **Addison.** *Introducción a los Sistemas de Bases de Datos*

32. **Garlan, D.S., Mary.** *An Introduction to Software Architecture.* 1994.

33. **Larman, C.** *UML y Patrones.* 1999.

34. **Burbeck, Steve.** *Application programming in Smalltalk-80: How to use Model-View-Controller (MVC).* 2007.

35. **Gamma, J.M.V.R.H.R.J.E.** *Design Patterns: Elements of Reusable Object-Oriented Software.* 1995.

36. **Breidenbach, C.W.w.R.** *Spring in Action.*

37. **Alur Deepak, Crupi John.** *Core J2EE patterns: best practices and design strategies.*

38. **Thomas Fihlman.** *Estimation in software development projects.*

GLOSARIO DE TÉRMINOS

AMPS (del inglés *Advanced Mobile Phone System*): Es un sistema de telefonía móvil de primera generación (1G, voz analógica)

API (del inglés *Application Programming Interface*): Conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Aplicaciones "standalone": Son aplicaciones que se pueden ejecutar como cualquier otro lenguaje de programación, sin necesidad de ningún elemento soporte.

Aplicaciones portables: Es una aplicación informática que puede ser utilizada en cualquier ordenador que posea el sistema operativo para el que fue programada sin instalación previa; esto significa que no es necesaria la instalación de bibliotecas adicionales en el sistema para su funcionamiento.

CDMA (del inglés *Code Division Multiple Access*): Sistema de telefonía digital inalámbrica basada en la tecnología de acceso múltiple por división por código o de espectro expandido. La conversación se marca con un código y se reparte entre varias frecuencias dentro de la banda. El receptor, a partir del código, reconstruye la señal.

*EJB (*del inglés *Enterprise JavaBeans):* Arquitectura que define la forma de construir componentes distribuidos del lado del servidor. Esta tecnología garantiza que los componentes programados sean escalables, eficaces y seguros a pesar de lo sencillo de su desarrollo.

Framework: Esquema, esqueleto o patrón para el desarrollo y/o la implementación de una aplicación. Conjunto de clases que cooperan y forman un diseño reutilizable para un tipo específico de software ofreciendo una guía arquitectónica partiendo el diseño en clases abstractas y definiendo sus responsabilidades y sus colaboraciones.

GSM (del inglés *Global System Mobile Communications*): Estándar internacional para comunicaciones digitales celulares, utilizado por más de la mitad de los teléfonos móviles del planeta. La familia de sistemas GSM incluye varias frecuencias, la original a 900 MHz, GSM 1800 y GSM 1900.

GPRS (del inglés *General Packet Radio System*): Tecnología que permite la transmisión de datos a alta velocidad a través de redes digitales inalámbricas de segunda generación, como GSM.

HTTP (del inglés *Hyper Text Transfer Protocol*): El protocolo de transferencia de hipertexto es el protocolo usado en cada transacción de la web (WWW).

HTTPS (del inglés *Hypertext Transfer Protocol Secure*): El protocolo seguro de transferencia de hipertexto, es un protocolo de red basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.

Inversión de Control (IoC): Es un concepto junto a unas técnicas de programación en las que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos o funciones.

Inyección de Dependencia: Propone que las dependencias entre clases no sean tan unidas, sino atadas a clases interfaces, es decir, crear objetos, sin importar quién provea la implementación. Solo se sabe que quién provea la construcción, va a implementar una clase interfaz determinada.

IDE (del inglés *Integrated Development Environment*): Programa compuesto por un conjunto de herramientas para un programador. Entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI.

*JSP (*del inglés *Java Server Pages):* Es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

JavaBeans: Son un modelo de componentes para la construcción de aplicaciones en Java. Se usan para encapsular varios objetos en un único objeto (la vaina), para hacer uso de un sólo objeto en lugar de varios más simples.

*JDBC (*del inglés *Java Database Connectivity):* Es un API para trabajar con bases de datos desde Java, independientemente de la base de datos a la que se acceda.

JDO: Resume en unas interfaces de programación los servicios necesarios para alcanzar la persistencia de los objetos Java sobre distintos sistemas de gestión de datos, de forma uniforme.

JDeveloper: Entorno de desarrollo integrado desarrollado por Oracle Corporation para lenguaje Java, HTML, XML, SQL, PL/SQL, Javascript, PHP, Oracle ADF, UML y otros.

*Licencia BSD (*del inglés *Berkeley Software Distribution):* Es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Pertenece al grupo de licencias de software Libre. Permite el uso del código fuente en software no libre.

*MMS (*del inglés *Multimedia Messaging Service):* El sistema de mensajería multimedia es un estándar de mensajería que le permite a los teléfonos móviles enviar y recibir contenidos multimedia, incorporando sonido, video, fotos o cualquier otro contenido disponible en el futuro.

*Netbeans: S*e refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) desarrollado usando la Plataforma Netbeans. La plataforma Netbeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos.

Plugin: Módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

*SMS (*del inglés *Short Message Service):* El servicio de mensajes cortos o SMS es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos (también conocidos como mensajes de texto, o más coloquialmente, textos o mensajitos) entre teléfonos móviles, teléfonos fijos y otros dispositivos de mano.

SMTP (del inglés *Simple Mail Transfer Protocol*): Protocolo Simple de Transferencia de Correo, es un protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA's, teléfonos móviles, etc.).

Servlet: Pequeño programa que corre en un servidor. Por lo general son aplicaciones Java que corren en un entorno de servidor web. Esto es análogo a una aplicación Java que corre en un navegador.

*SQL (*del inglés *Structured Query Language):* Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.

Singleton: El patrón de diseño Singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.
*URL (*del inglés *Uniform Resource Locator):* Es un localizador uniforme de recurso y se refiere a la dirección única que identifica a una página web en Internet.

*WAP (*del inglés *Wireless Application Protocol):* Es un estándar abierto internacional para aplicaciones que utilizan las comunicaciones inalámbricas, p.ej. acceso a servicios de Internet desde un teléfono móvil.

Wifi: Es un sistema de envío de datos sobre redes computacionales que utiliza ondas de radio en lugar de cables.

*Wimax (*del inglés *World Wide Interoperability for Microwave Access*): Es un protocolo de transmisión de datos inalámbrico que va un paso más allá de Wifi.

ANEXOS

Anexo 1: Descripción de Casos de Uso

Tabla 1.2 Descripción del Caso de Uso Autenticar.

Caso de Uso	**Autenticar**	
Actor	Administrador.	
Propósito	Autenticación del administrador en el sistema.	
Resumen	El caso de uso inicia cuando el administrador necesita autenticarse para poder acceder al sistema, entra su usuario, contraseña y accede al sistema.	
Referencias	R1.	
Precondiciones	El Administrador levanta el sistema y se encuentra en la página inicial de autenticación.	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
1. El administrador introduce su usuario y la contraseña.		2. Se comprueba que no existen campos vacíos.
		3. Se comprueba que el usuario sea correcto.
		4. Se comprueba que la contraseña sea correcta.
		5. El sistema le da acceso a las funciones de los administradores.
Flujos Alternos de los Eventos		
		2.1 Se comprueba que existen campos vacíos.
		2.2 Muestra un mensaje: “Datos incorrectos o no válidos”.

	3.1 Comprueba que el usuario no es válido.
	3.2 Muestra un mensaje: “Datos incorrectos o no válidos”.
	4.1 Comprueba que la contraseña no es válida.
	4.2 Muestra un mensaje: “Datos incorrectos o no válidos”.
Postcondiciones	El usuario accede al sistema.
Prioridad	Crítico

Tabla 1.3 Descripción del Caso de Uso Gestionar Administrador.

Caso de Uso	**Gestionar Administrador**
Actor	Administrador.
Propósito	Permitir al administrador gestionar los administradores.
Resumen	El actor decide que acción va a realizar (adicionar, modificar, eliminar). En el caso de adicionar o modificar, llena o actualiza los datos. Si es eliminar selecciona el administrador y lo elimina.
Referencias	R2.
Precondiciones	El administrador debe estar autenticado en el sistema.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Gestionar Administrador del menú principal.	2. El sistema muestra una tabla con los diferentes administradores a gestionar.
3. El administrador selecciona la opción:	

Adicionar (Ver sección “Adicionar administrador”). Modificar (Ver sección “Modificar administrador”). Eliminar (Ver sección “Eliminar administrador”).	
Sección: ¨Adicionar administrador¨	
1. El administrador selecciona la opción Adicionar.	2. El sistema muestra la interfaz con los campos siguientes: Usuario Contraseña
3. El administrador introduce los campos y la opción Aceptar.	4. El sistema verifica que los campos hayan sido llenados correctamente.
	7. El sistema adiciona al administrador.
Flujos Alternos de los Eventos	
	4.1 Se comprueba que existen campos vacíos.
	4.2 Muestra un mensaje: “Datos incorrectos o no válidos”.
Sección: ¨Modificar Administrador¨	
Acción del actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Modificar.	
2. El administrador selecciona el administrador a modificar.	3. El sistema muestra una interfaz con todos los campos:

	Usuario Contraseña
4. El administrador realiza los cambios deseados y selecciona la opción Aceptar.	5. El sistema verifica que los datos sean válidos.
	6. Guarda los cambios realizados.
Flujos Alternos de los Eventos	
	5.1 Comprueba que los datos son incorrectos.
	5.2 Muestra un mensaje: “Datos incorrectos o no válidos”.
Sección: ¨Eliminar Administrador¨	
Acción del actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Eliminar.	2. El sistema muestra un mensaje: “Está seguro que desea eliminar el administrador” Aceptar Cancelar
3. El administrador selecciona la opción Aceptar.	4. El sistema elimina el administrador.
Flujos Alternos de los Eventos	
3.1 El administrador selecciona la opción Cancelar.	3.2 El sistema cancela la operación.
Postcondiciones	Se actualiza toda la información referente a los administradores.
Prioridad	Crítico

Tabla 1.5 Descripción del Caso de Uso Gestionar Categoría

Caso de Uso	**Gestionar Categoría**
Actor	Administrador.
Propósito	Permitir al administrador gestionar las categorías.
Resumen	El actor decide que acción va a realizar (adicionar, eliminar). En el caso de adicionar, llena o actualiza los datos. Si es eliminar selecciona la categoría y la elimina.
Referencias	R4.
Precondiciones	El administrador debe estar autenticado en el sistema.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Gestionar Categoría del menú principal.	2. El sistema muestra una tabla con las diferentes categorías a gestionar.
3. El administrador selecciona la opción: Adicionar (Ver sección "Adicionar categoría"). Eliminar (Ver sección "Eliminar categoría").	
Sección: ¨Adicionar Categoría¨	
1. El administrador selecciona la opción Adicionar.	2. El sistema muestra la interfaz con el campo siguiente: Nombre
3. El administrador introduce el campo y la opción Aceptar.	4. El sistema verifica que el campo haya sido llenado correctamente.

	5. Registra la nueva funcionalidad.
Flujos Alternos de los Eventos	
	4.1 Se comprueba que el campo está vacío.
	4.2 Muestra un mensaje: “Datos incorrectos o no válidos”.
Sección: ¨Eliminar Categoría¨	
Acción del actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Eliminar.	2. El sistema muestra un mensaje: “Está seguro que desea eliminar la categoría.” Aceptar Cancelar
3. El administrador selecciona la opción Aceptar.	4. El sistema elimina la categoría.
Flujos Alternos de los Eventos	
3.1 El administrador selecciona la opción Cancelar.	3.2 El sistema cancela la operación.
Postcondiciones	Se actualiza toda la información referente a las categorías.
Prioridad	Crítica

Tabla 1.6 Descripción del Caso de Uso Gestionar Cliente de Servicio.

Caso de Uso	Gestionar Cliente de Servicio
Actor	Administrador.
Propósito	Permitir al administrador gestionar los Clientes de servicio

Resumen	El actor decide que acción va a realizar (adicionar, modificar, eliminar). En el caso de adicionar o modificar, llena o actualiza los datos. Si es eliminar selecciona el Cliente de servicio y lo elimina.
Referencias	R2.
Precondiciones	El administrador debe estar autenticado en el sistema.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Gestionar Cliente de Servicio del menú principal.	2. El sistema muestra una tabla con los diferentes clientes de servicio a gestionar.
3. El administrador selecciona la opción: Adicionar (Ver sección “Adicionar Cliente de Servicio”).) Modificar (Ver sección “Modificar Cliente de Servicio”). Eliminar (Ver sección “Eliminar Cliente de Servicio”).	
Sección: “Adicionar Cliente de Servicio”	
1. El administrador selecciona la opción Adicionar.	2. El sistema muestra la interfaz con los campos siguientes: Usuario Contraseña
3. El administrador introduce los campos y la opción Aceptar.	4. El sistema verifica que los campos hayan sido llenados correctamente.
	5. El sistema registra la nueva funcionalidad.

Flujos Alternos de los Eventos	
	4.1 Se comprueba que existen campos vacíos.
	4.2 Muestra un mensaje: “Datos incorrectos o no válidos”.
Sección: “Modificar Cliente de Servicio”	
Acción del actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Modificar.	2. El sistema muestra una interfaz con todos los campos: Usuario Contraseña
3. El administrador realiza los cambios deseados y selecciona la opción Aceptar.	4. El sistema verifica que los datos sean válidos.
	5. Guarda los cambios realizados.
Flujos Alternos de los Eventos	
	4.1 Comprueba que los datos son incorrectos.
	4.2 Muestra un mensaje “Datos incorrectos o no válidos”.
Sección: Eliminar “Cliente de Servicio”	
Acción del actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Eliminar.	2. El sistema muestra un mensaje: “Está seguro que desea eliminar el cliente de servicio.” Aceptar Cancelar
5. El administrador selecciona la opción Aceptar.	6. El sistema elimina el Cliente de Servicio.

Flujos Alternos de los Eventos	
5.1 El administrador selecciona la opción Cancelar.	5.2 El sistema cancela la operación.
Postcondiciones	Se actualiza toda la información referente a los Clientes de Servicio
Prioridad	Crítico

Tabla 1.7 Descripción del Caso de Uso Gestionar Suscripción

Caso de Uso	**Gestionar Suscripción**
Actor	Sistema WAP
Propósito	Permitir que el sistema WAP gestione las suscripciones.
Resumen	El actor decide que acción va a realizar (adicionar, modificar, eliminar). En el caso de adicionar o modificar, llena o actualiza los datos. Si es eliminar envía el identificador de la suscripción y la elimina.
Referencias	R6.
Precondiciones	El sistema tiene que estar identificado.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El sistema WAP realiza una acción: Adicionar Suscripción (Ver sección “Adicionar Suscripción”). Modificar Suscripción (Ver sección “Modificar Suscripción”). Eliminar Suscripción (Ver sección “Eliminar Suscripción”).	
Sección: ¨Adicionar Suscripción¨	
1. El sistema WAP solicita adicionar una suscripción.	
2. El sistema WAP envía el código del servicio, el número de teléfono, la fecha y una lista con los valores de los campos.	3. El sistema busca el servicio.
	4. El sistema comprueba que existe el servicio.
	5. El sistema busca la suscripción.
	6. El sistema comprueba que existe.
	7. El sistema comprueba que está deshabilitada la suscripción.
	8. El sistema busca el usuario por el número de teléfono.
	9. El sistema comprueba que está deshabilitado.
	10. El sistema habilita al usuario.
	11. El sistema modifica la fecha de la suscripción.
	12. El sistema busca los valores de los campos de esa suscripción.

	13. El sistema elimina los valores de los campos de esa suscripción.
	14. El sistema adiciona los nuevos valores de los campos.
	15. El sistema habilita la suscripción.
Flujos Alternos de los Eventos	
	6.1 El sistema comprueba que no existe la suscripción.
	6.2 El sistema busca el usuario por el número de teléfono.
	6.3 El sistema comprueba que existe el usuario.
	6.4 El sistema comprueba que está deshabilitado.
	6.5 El sistema habilita al usuario.
	6.6 El sistema adiciona la suscripción.
	6.7 El sistema adiciona los valores de los campos.
Flujos Alternos de los Eventos	
	6.3.1 El sistema comprueba que existe el usuario.
	6.3.2 El sistema adiciona el usuario. Retorna a la acción 6.6
Sección: ¨Modificar Suscripción¨	
Acción del actor	**Respuesta del Sistema**
1. El sistema WAP solicita modificar una suscripción.	
2. El sistema WAP envía la lista de valores de	3. El sistema modifica la suscripción.

los campos a modificar.	
Sección: ¨Eliminar Suscripción¨	
Acción del actor	**Respuesta del Sistema**
1. El sistema WAP solicita eliminar una suscripción.	
2. El sistema WAP envía el identificador de la suscripción a eliminar.	3. El sistema busca el usuario perteneciente a dicha suscripción.
	4. El sistema comprueba que existe el usuario.
	5. El sistema busca los valores de los campos de la suscripción.
	6. El sistema busca la cantidad de suscripciones del usuario.
	7. El sistema comprueba que tiene una sola suscripción.
	8. El sistema deshabilita el usuario.
	9. El sistema deshabilita los valores de los campos.
Flujos Alternos de los Eventos	
	4.1 El sistema comprueba que no existe el usuario.
Flujos Alternos de los Eventos	
	7.1 El sistema comprueba que tiene más de una suscripción.
	7.2 El sistema deshabilita los valores de los campos.

Postcondiciones	
Prioridad	Crítica

Tabla 1.9 Descripción del Caso de Uso Mostrar Reportes de Suscripciones

Caso de Uso	**Mostrar Reportes de Suscripciones**
Actor	Sistema Proveedor
Propósito	Permitir que el Sistema Proveedor solicite los reportes de suscripciones.
Resumen	El actor decide que acción va a solicitar (Mostrar la cantidad de usuarios que se han dado de baja de un servicio determinado, mostrar la cantidad de usuarios que se han suscrito a un servicio determinado, mostrar todos los usuarios suscritos a un servicio, mostrar valor del campo del servicio de un usuario determinado y mostrar los campos de un servicio) En cualquier caso el sistema mostrará la información solicitada.
Referencias	R8.
Precondiciones	El Sistema Proveedor tiene que estar identificado.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
El sistema Proveedor realiza una acción: Mostrar la cantidad de usuarios que se han dado de baja de un servicio determinado (Ver sección “Cantidad de Usuarios de Baja”). Mostrar la cantidad de usuarios que se han suscrito a un servicio determinado (Ver sección “Cantidad de Usuarios Suscritos”). Usuarios suscritos a un determinado servicio. (Ver sección: “Usuarios suscritos a un Servicio”) Mostrar valor del campo del servicio de un	

usuario determinado.(Ver sección: “Valor del campo”) Mostrar campos de un servicio.(Ver sección:”Campos de un Servicio”)	
Sección: “Cantidad de Usuarios de Baja”	
Acción del Actor	**Respuesta del Sistema**
1. El sistema Proveedor selecciona la opción “Cantidad de usuarios que se han dado de baja de un servicio determinado”.	
2. El sistema Proveedor envía el código del servicio.	3. El sistema obtiene la cantidad de usuarios dados de baja.
	4. Devuelve la cantidad.
Sección: “Cantidad de Usuarios Suscritos”	
Acción del Actor	**Respuesta del Sistema**
1. El sistema Proveedor selecciona la opción “Cantidad de Usuarios Suscritos”.	
2. El sistema Proveedor envía el código del servicio.	3. El sistema obtiene la cantidad de usuarios suscritos.
	4. Devuelve la cantidad de usuarios.
Sección: “Usuarios Suscritos a un Servicio”	
1. El sistema Proveedor selecciona la opción “Usuarios Suscritos a un Servicio”.	
2. El sistema Proveedor envía el código del servicio.	3. El sistema obtiene los usuarios suscritos.
	4. Devuelve los usuarios suscritos.

Sección: “Valor del campo”	
Acción del Actor	**Respuesta del Sistema**
1. El sistema Proveedor selecciona la opción “Mostrar valor del campo del servicio de un usuario determinado”	
2. El proveedor envía el número telefónico del usuario.	3. El sistema busca el valor de los campos que tengan los servicios de ese usuario.
	4. El sistema devuelve el valor.
Sección: “Campos de un Servicio”	
Acción del Actor	**Respuesta del Sistema**
1. El sistema Proveedor selecciona la opción “Mostrar campos de un servicio”	
2. El proveedor envía el código del servicio.	3. El sistema busca los campos que pertenezcan a ese servicio.
	4. El sistema devuelve los campos.
Postcondiciones	El actor obtiene los reportes.
Prioridad	Crítico.

Anexo 2: Diagramas de Clases del Análisis

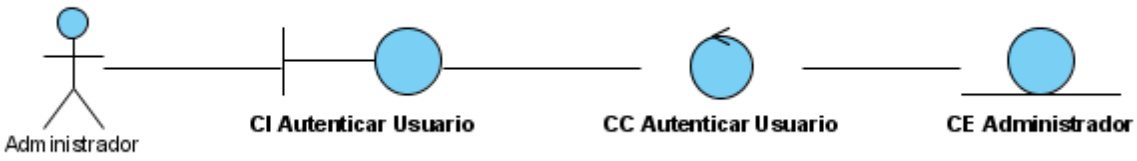

Figura 4. Diagrama de Clases del Análisis: Autenticar

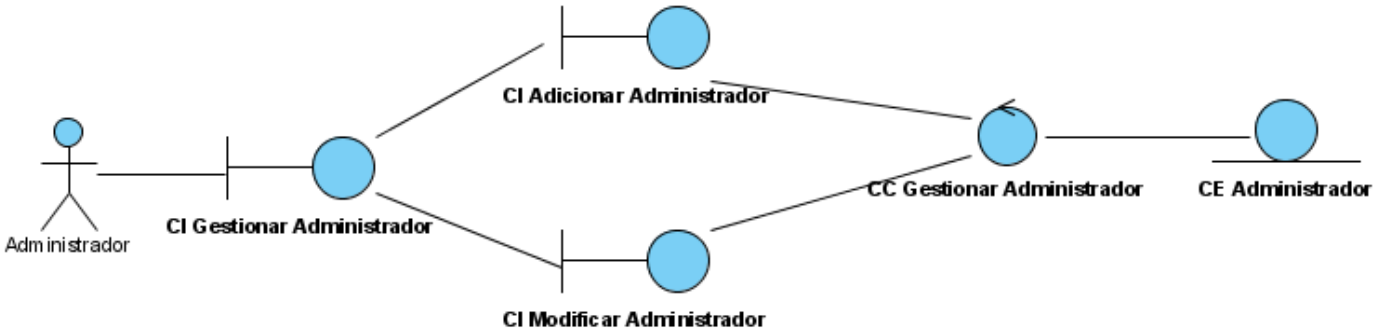

Figura 5. Diagrama de Clases del Análisis: Gestionar Administrador

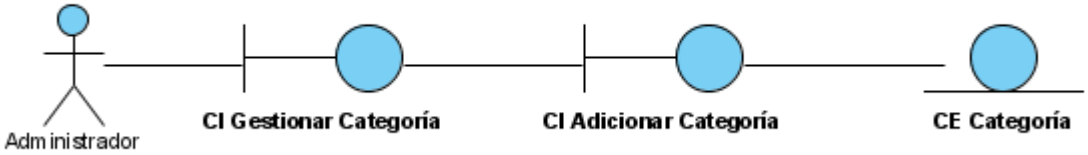

Figura 7. Diagrama de Clases del Análisis: Gestionar Categoría

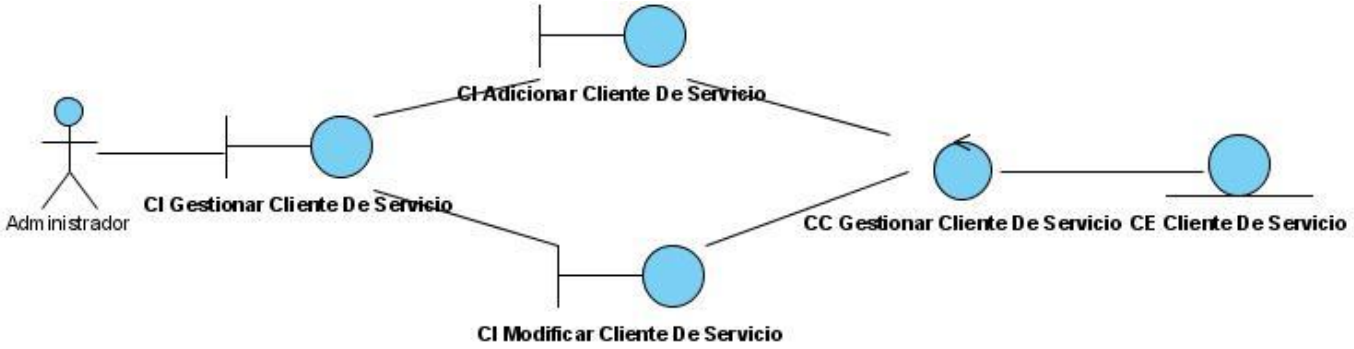

Figura 8. Diagrama de Clases del Análisis: Gestionar Cliente de Servicio

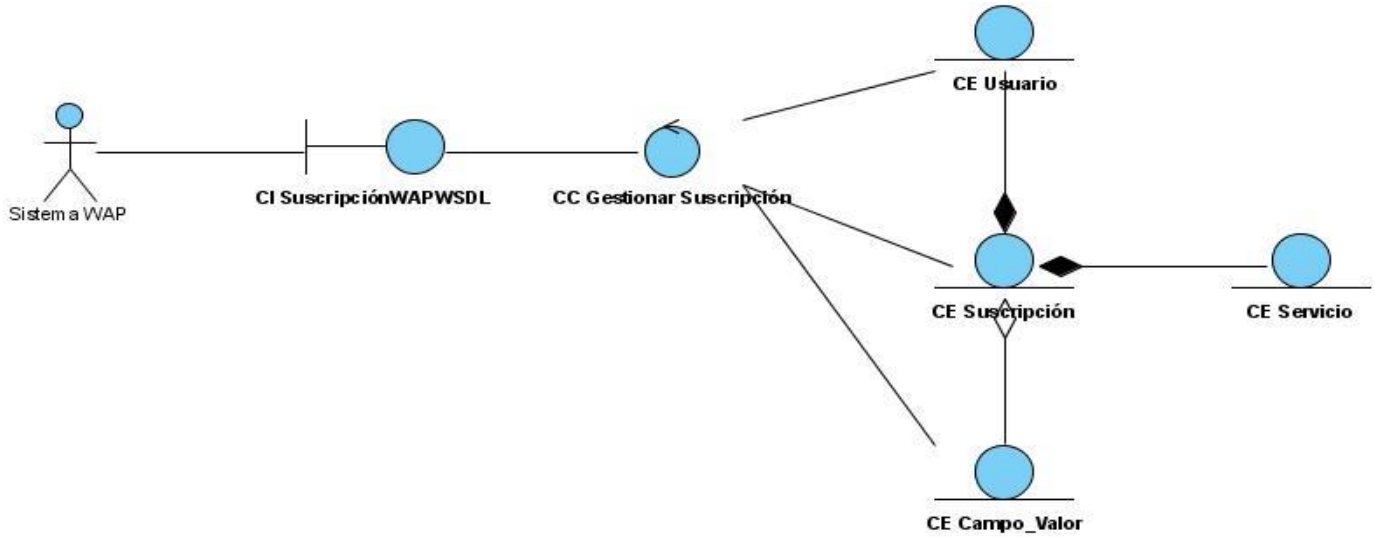

Figura 9. Diagrama de Clases del Análisis: Gestionar Suscripción

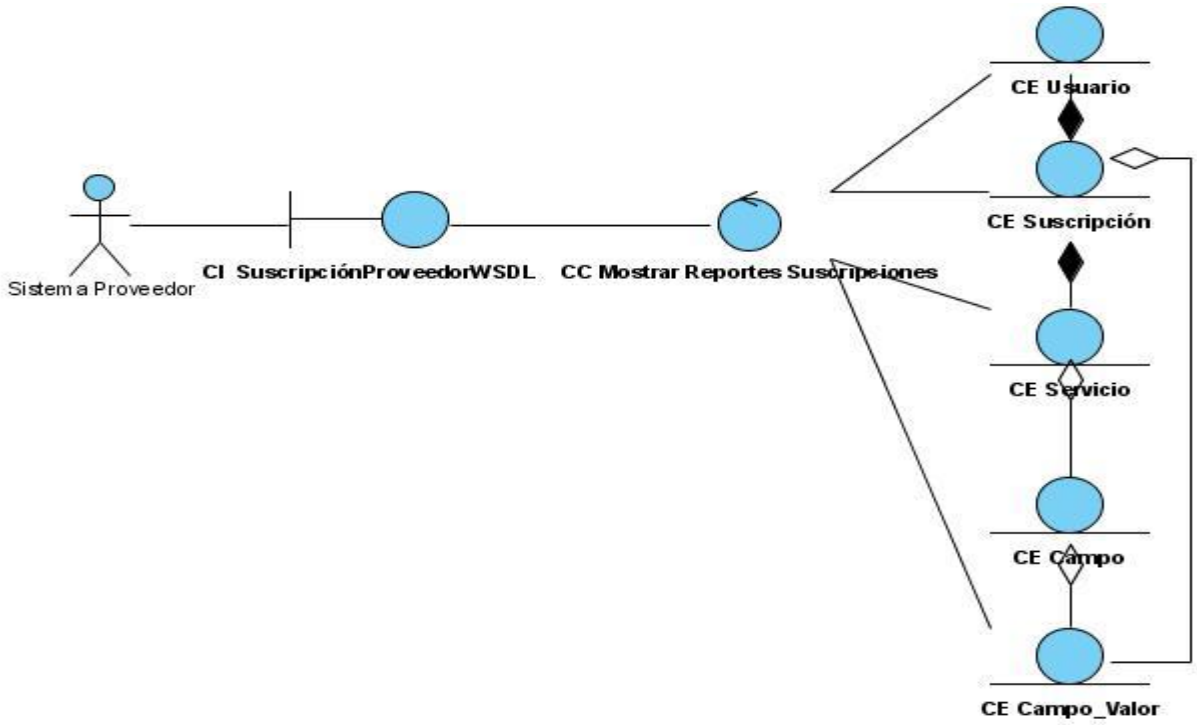

Figura 11. Diagrama de Clases del Análisis: Mostrar Reportes de Suscripciones

Anexo 3: Diagramas de Clases del Diseño

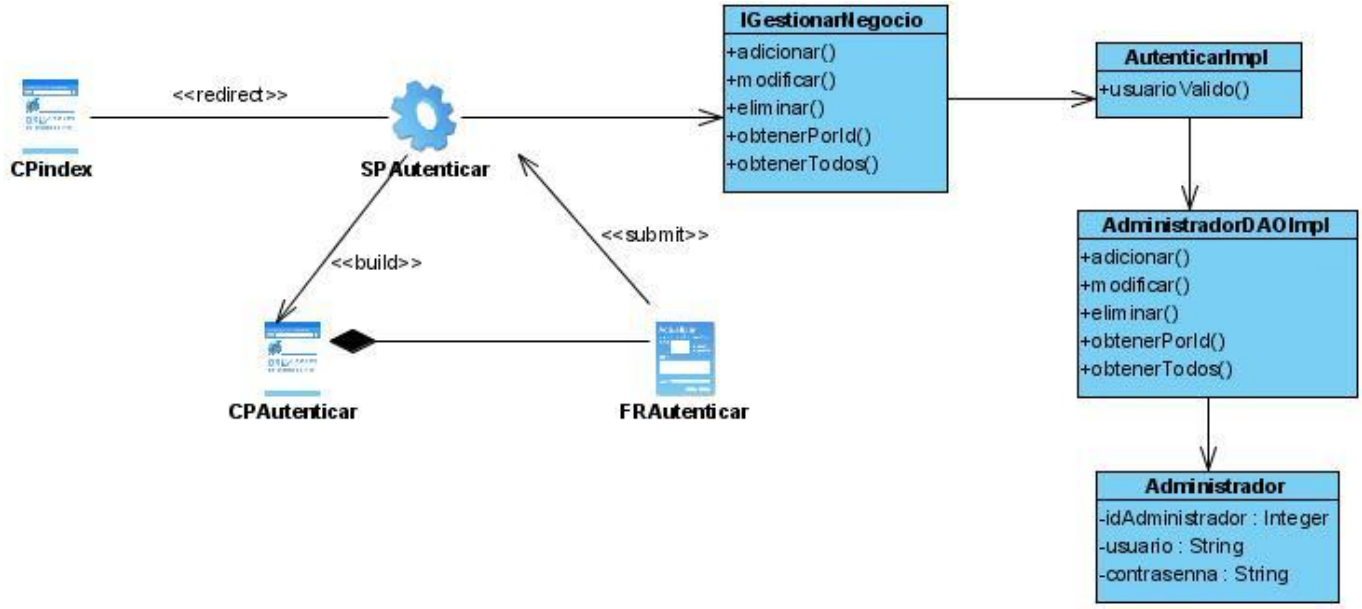

Figura 12. Diagrama de Clases del Diseño: Autenticar

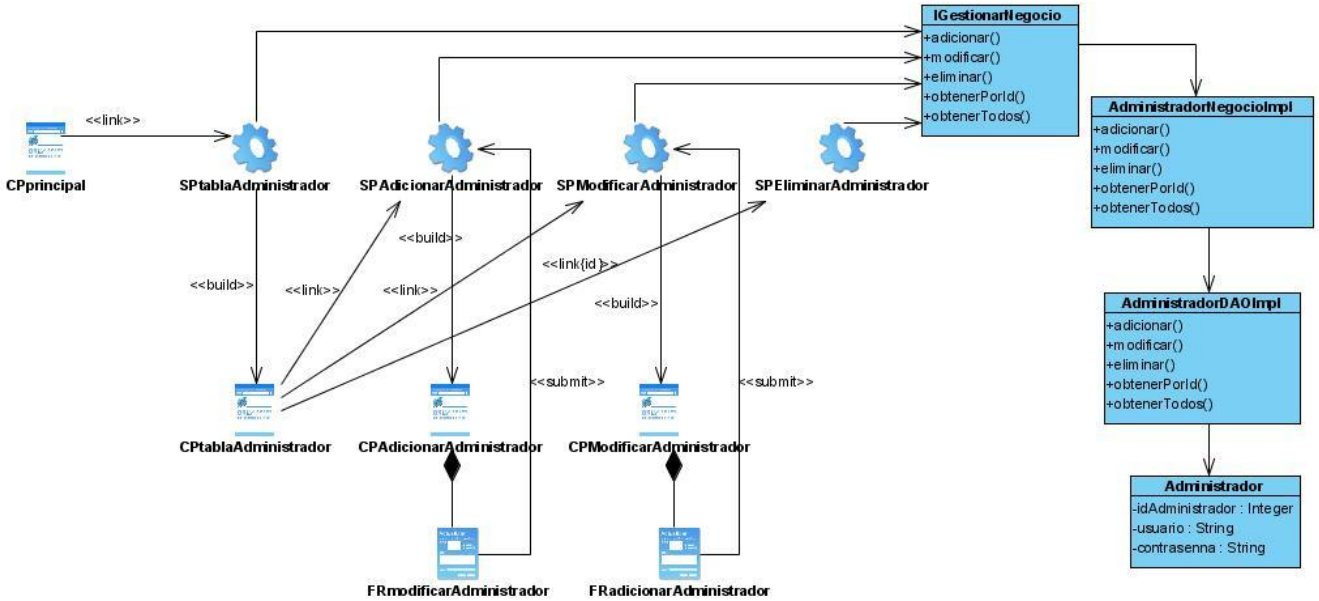

Figura 13. Diagrama de Clases del Diseño: Gestionar Administrador

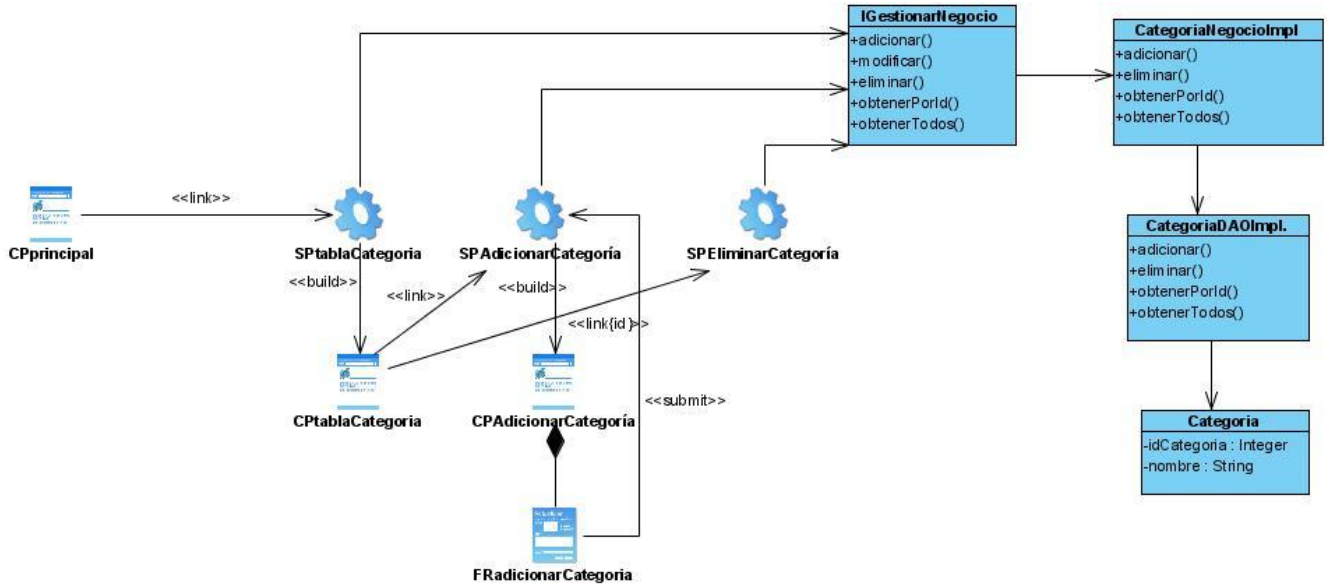

Figura 15. Diagrama de Clases del Diseño: Gestionar Categoría

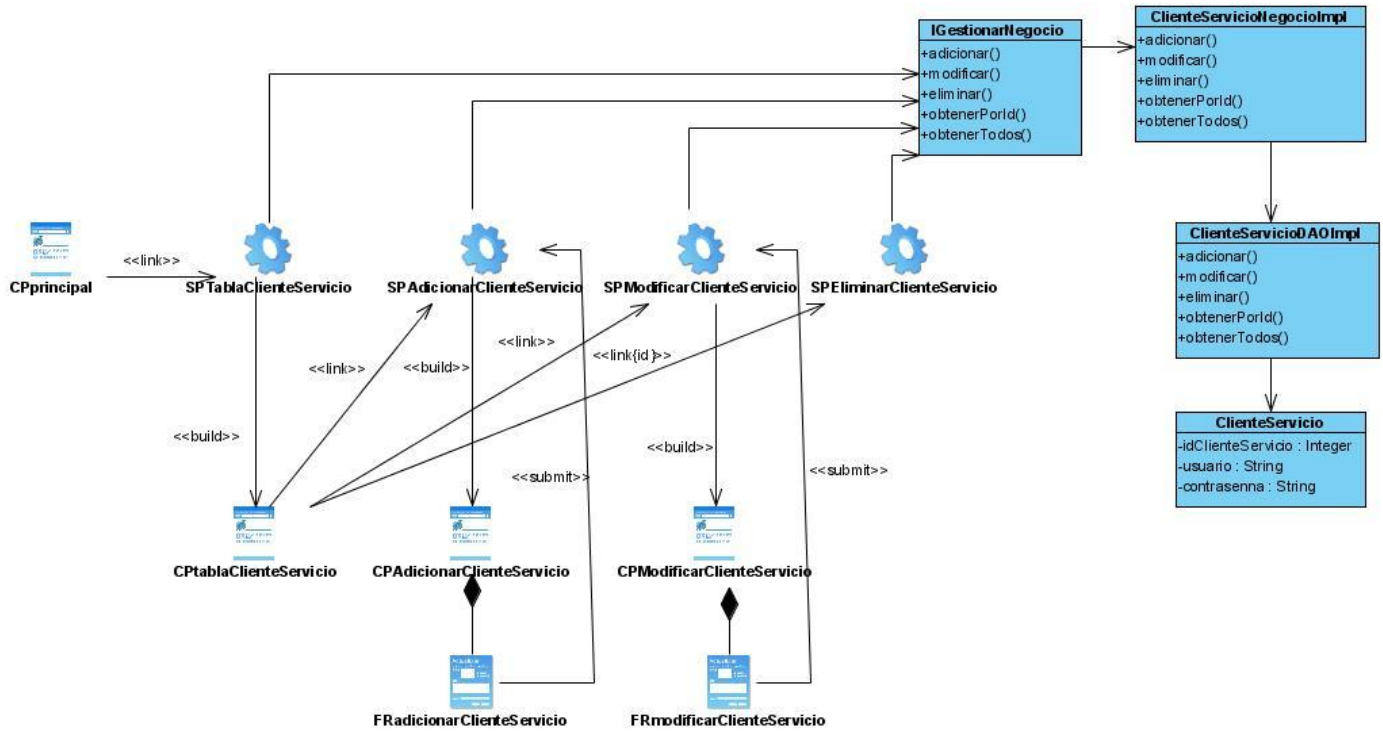

Figura 16. Diagrama de Clases del Diseño: Gestionar Cliente Servicio

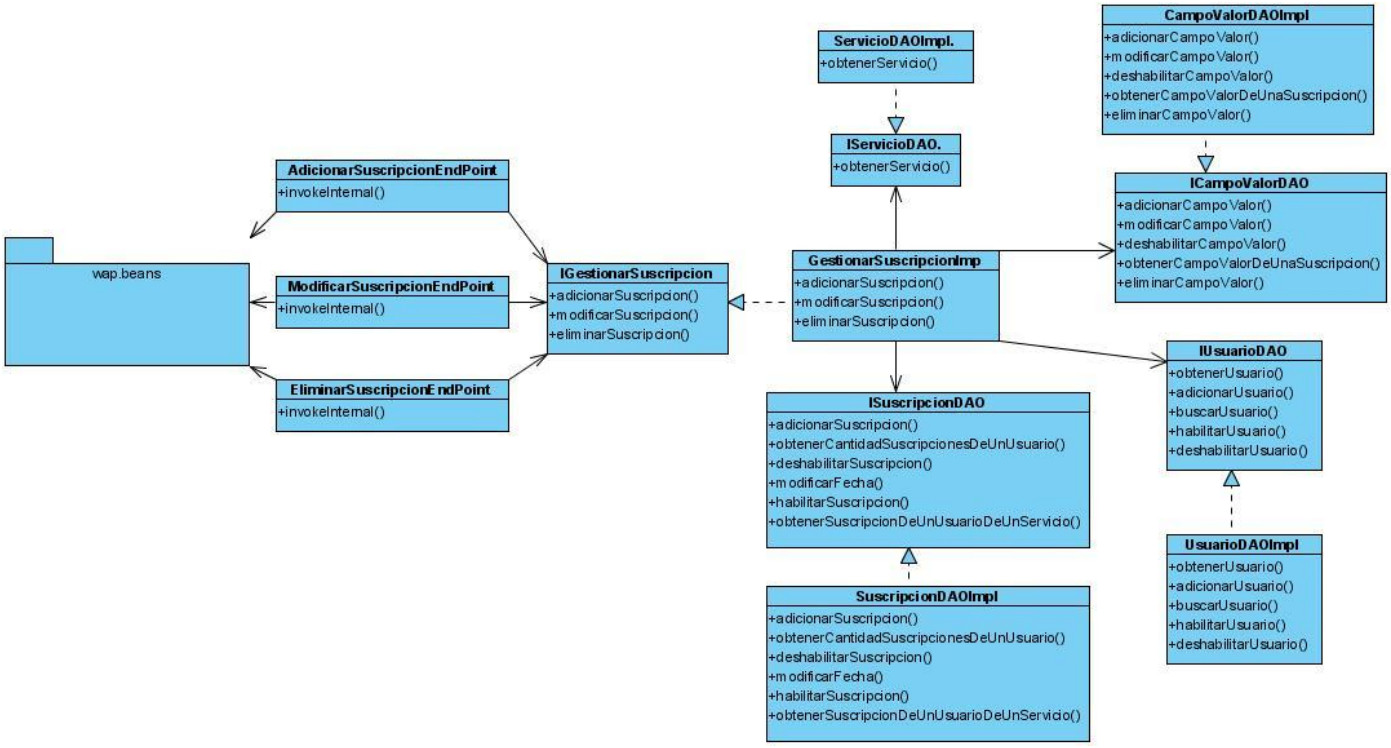

Figura 17. Diagrama de Clases del Diseño: Gestionar Suscripción

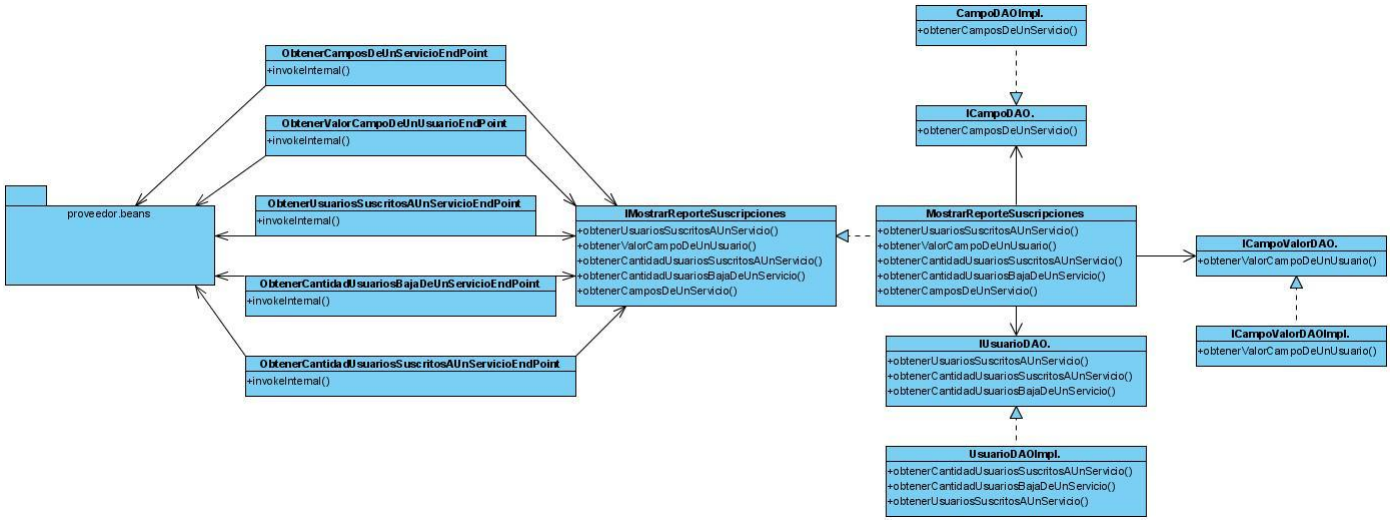

Figura 19. Diagrama de Clases del Diseño: Reportes de Suscripciones

Anexo 4: Diagramas de Secuencia

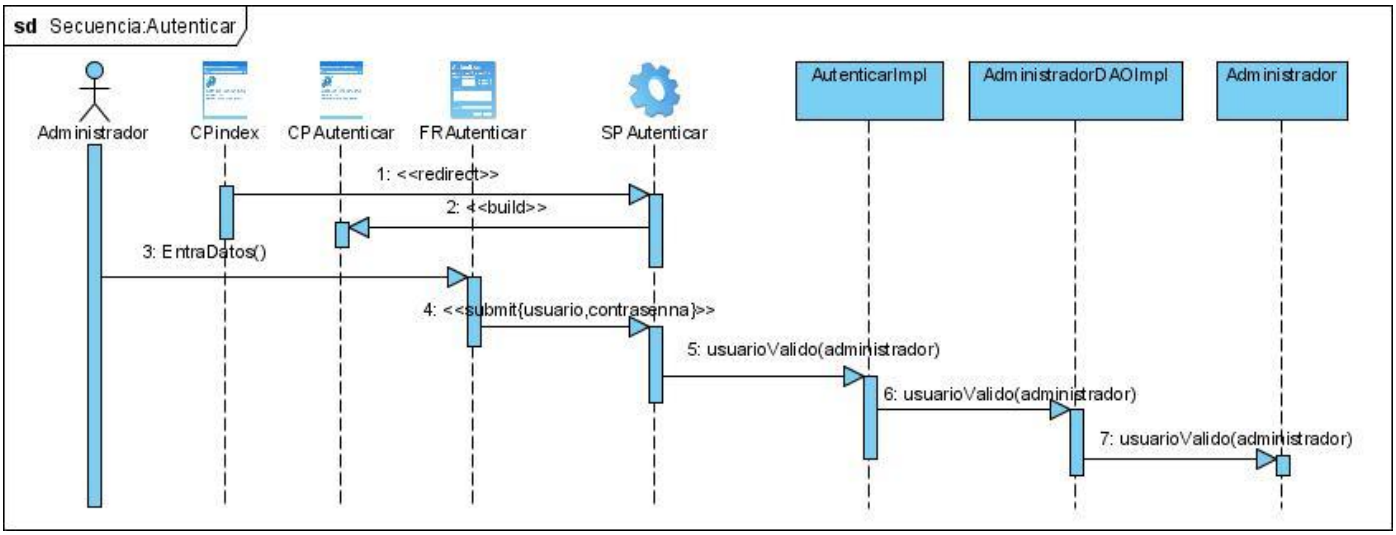

Figura 20. Diagrama de Secuencia del Diseño: Autenticar

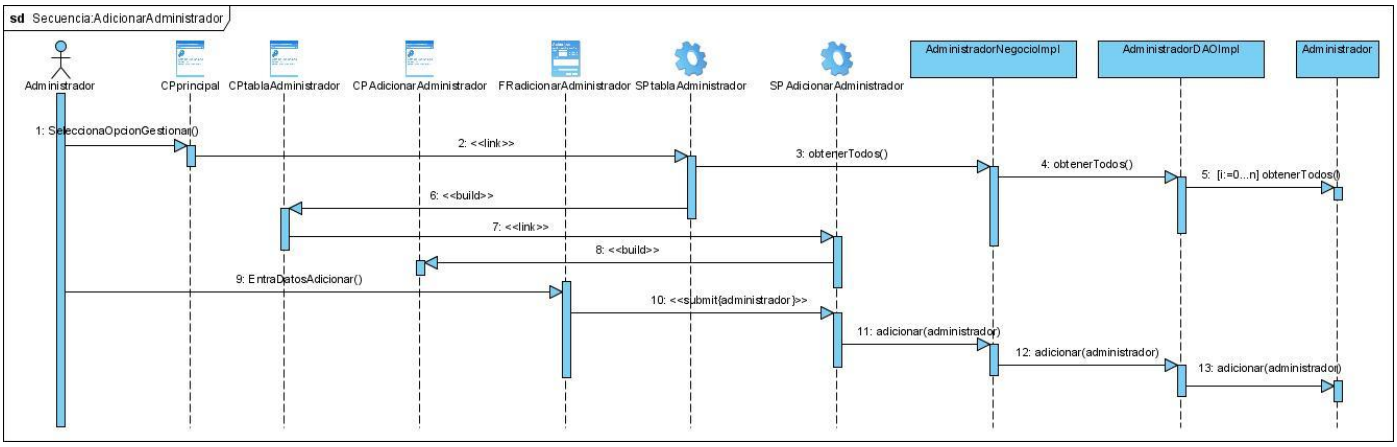

Figura 21. Diagrama de Secuencia del Diseño: Adicionar Administrador

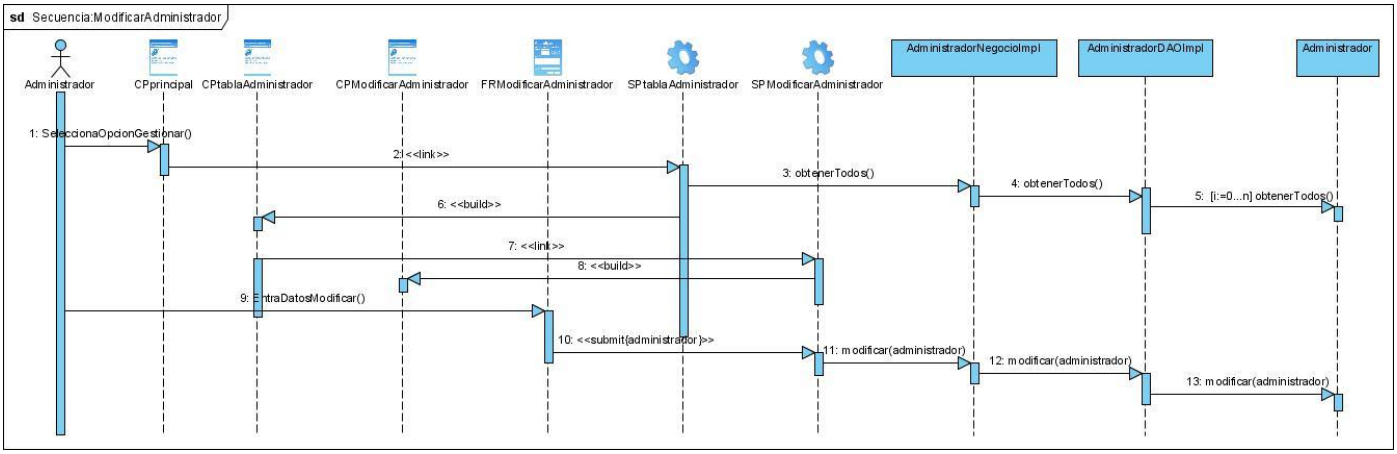

Figura 22. Diagrama de Secuencia del Diseño: Modificar Administrador

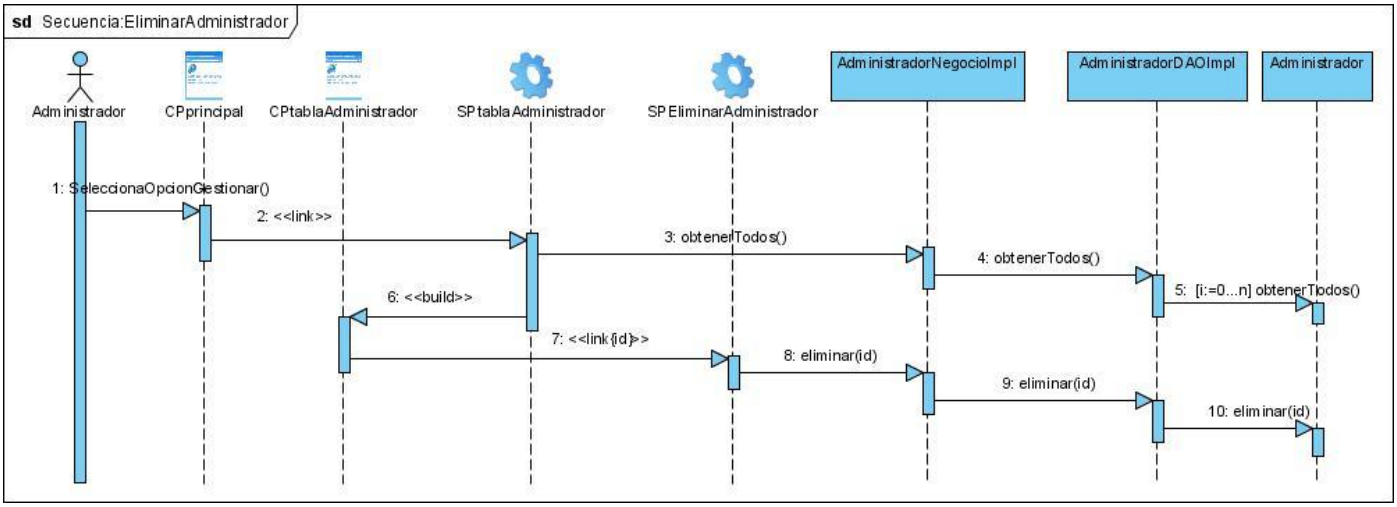

Figura 23. Diagrama de Secuencia del Diseño: Eliminar Administrador

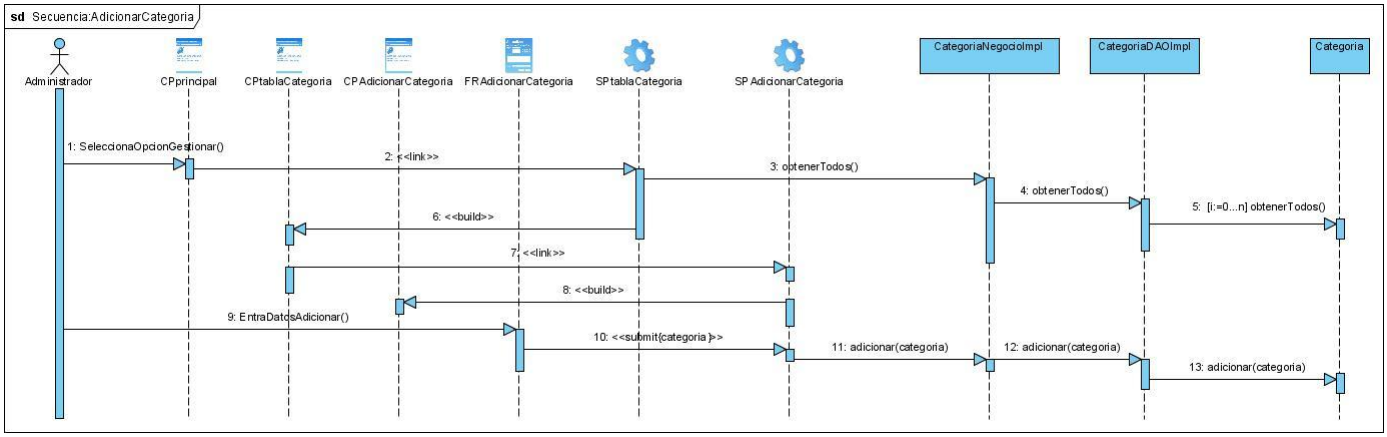

Figura 27. Diagrama de Secuencia del Diseño: Adicionar Categoría

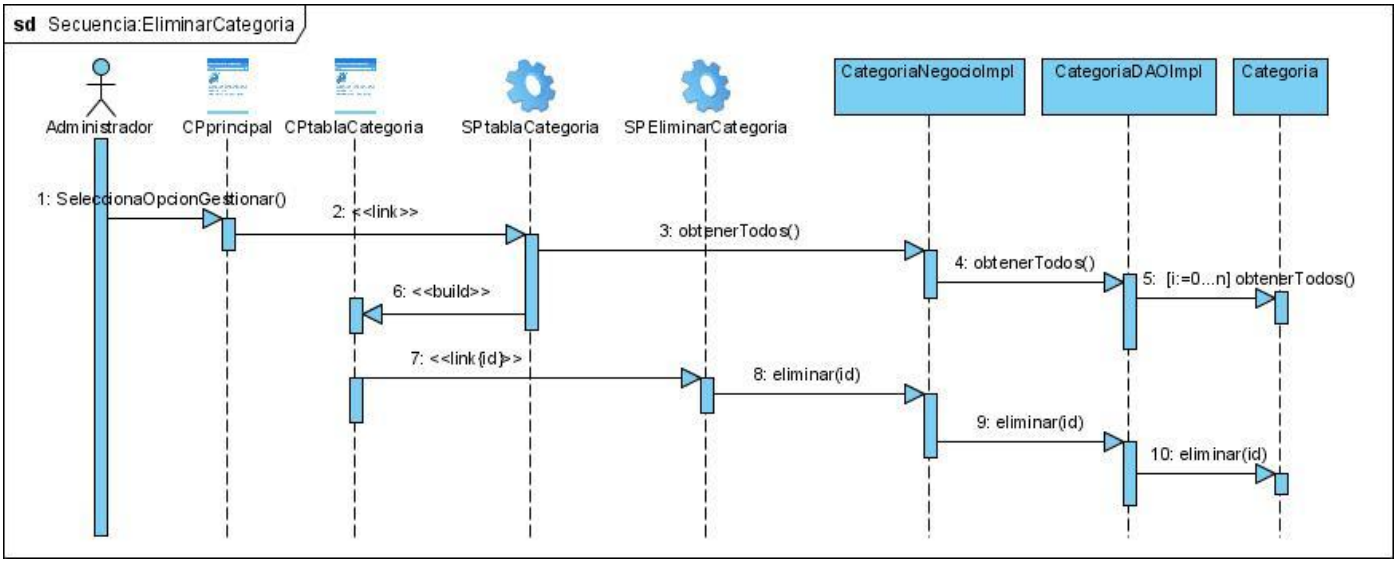

Figura 28. Diagrama de Secuencia del Diseño: Eliminar Categoría

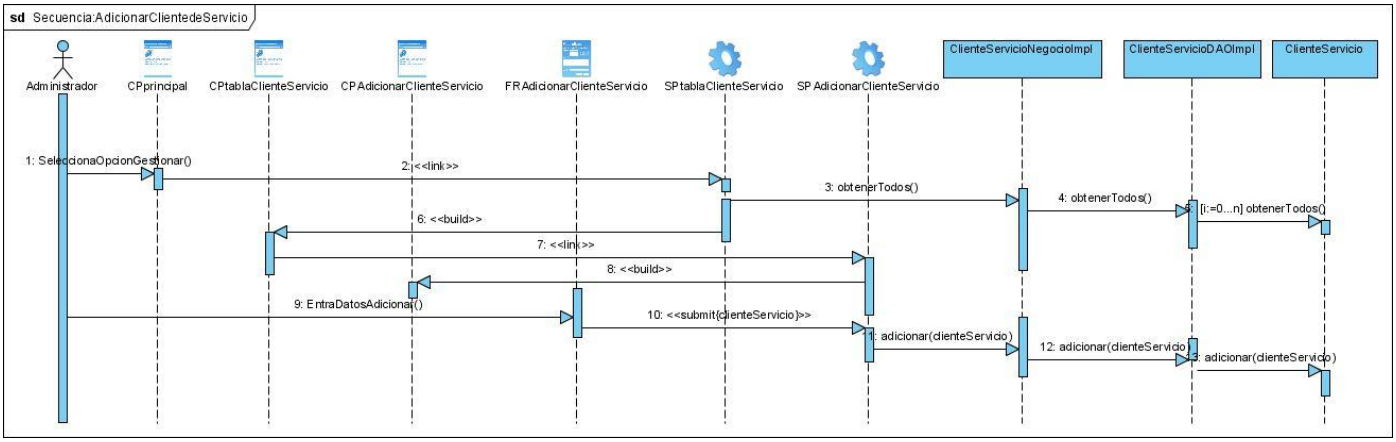

Figura 29. Diagrama de Secuencia del Diseño: Adicionar Cliente de Servicio

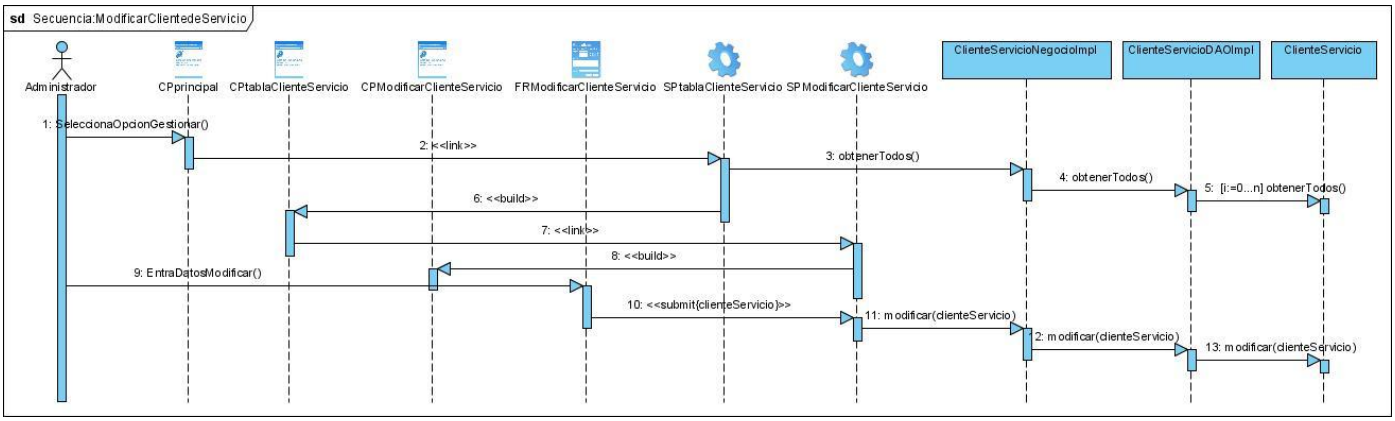

Figura 30. Diagrama de Secuencia del Diseño: Modificar Cliente de Servicio

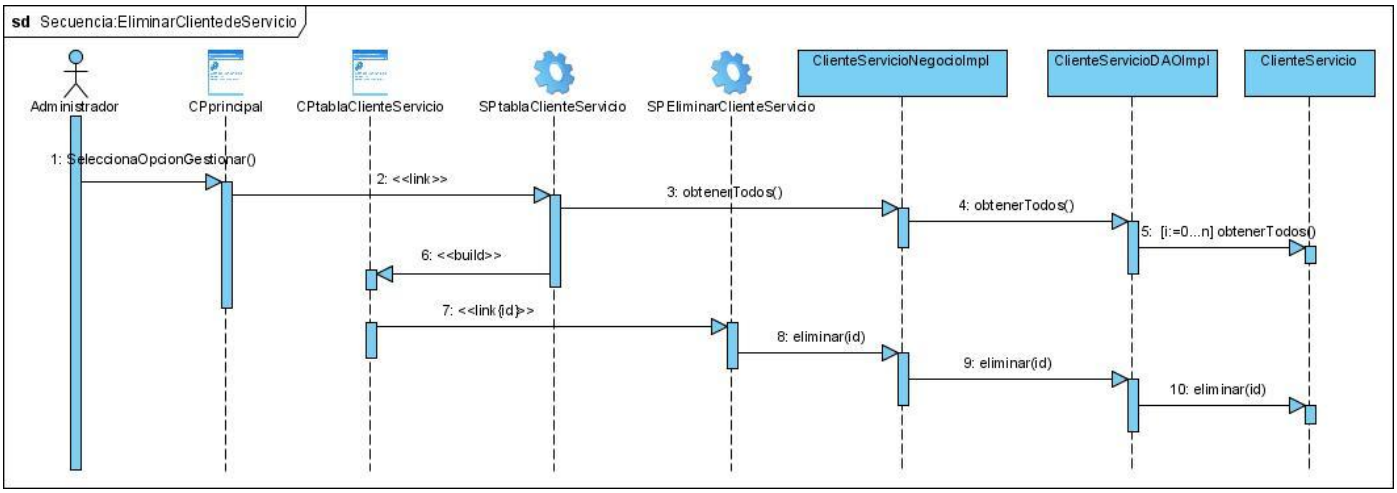

Figura 31. Diagrama de Secuencia del Diseño: Eliminar Cliente de Servicio

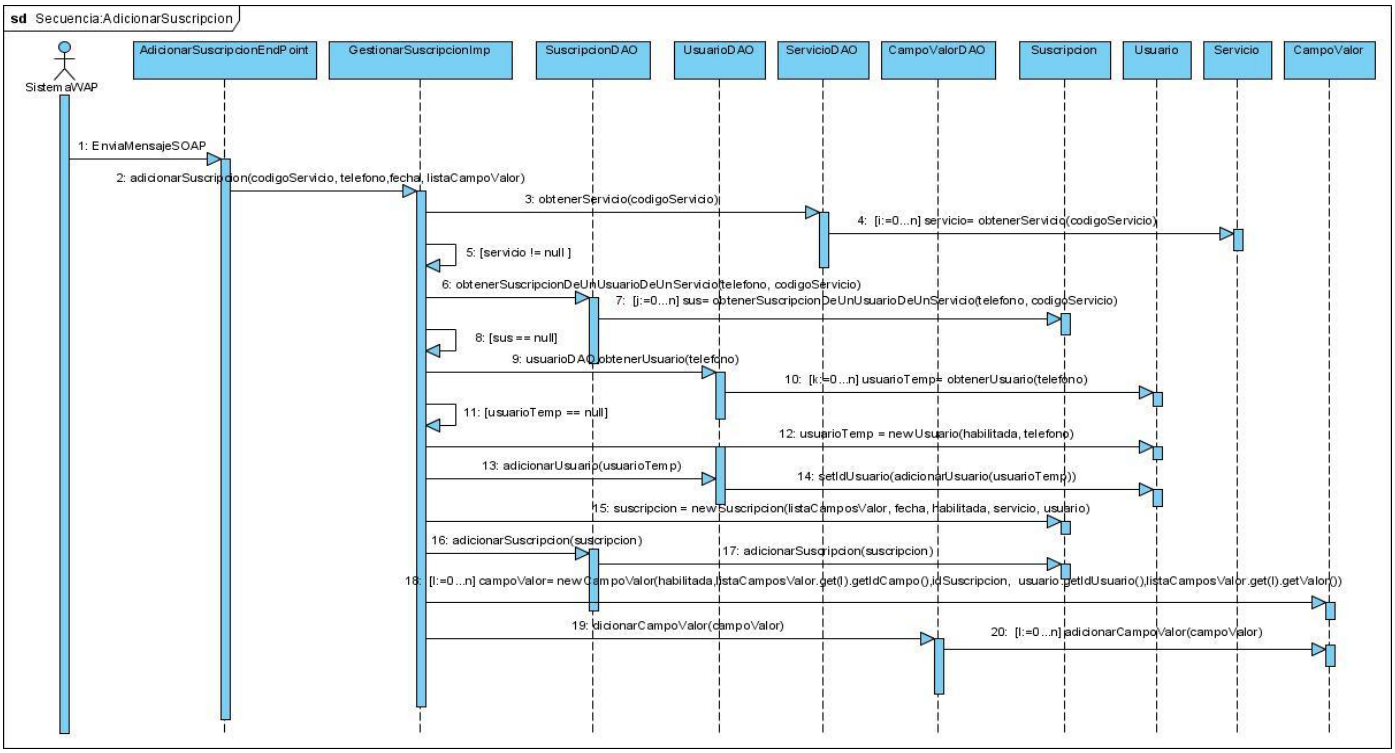

Figura 32. Diagrama de Secuencia del Diseño: Adicionar Suscripción.

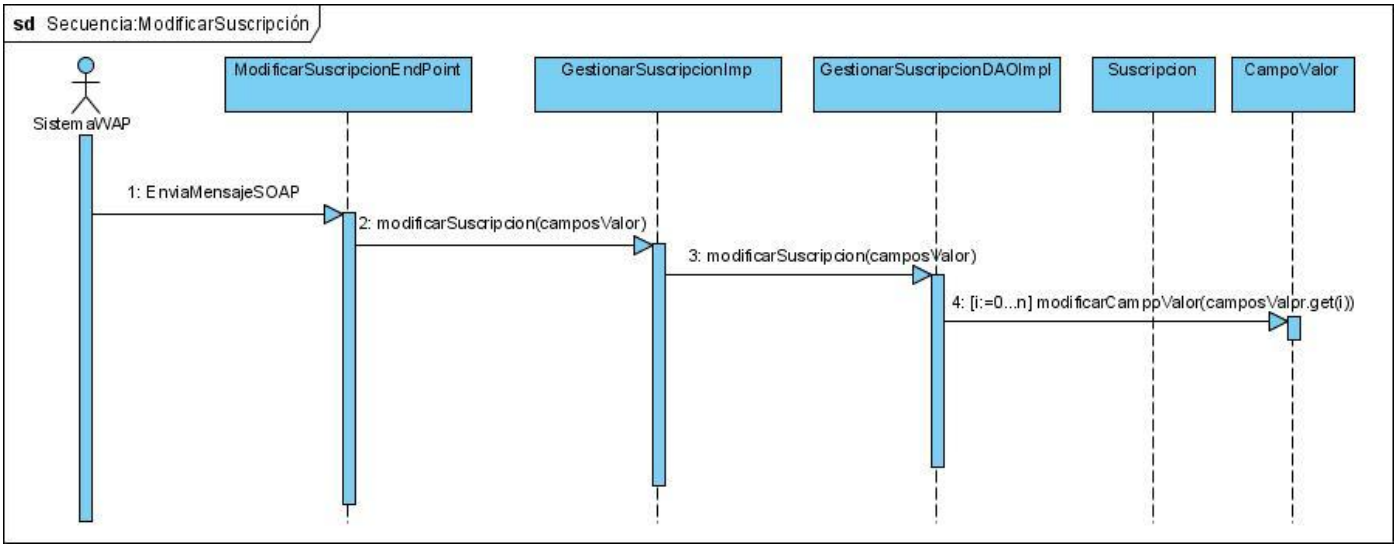

Figura 33. Diagrama de Secuencia del Diseño: Modificar Suscripción

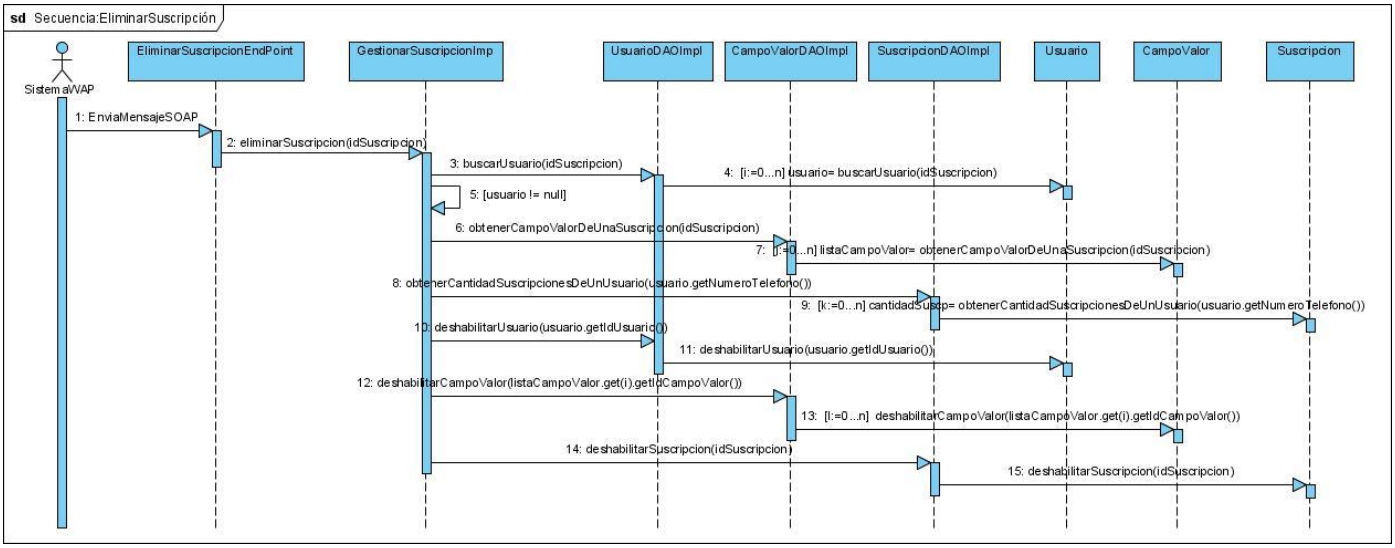

Figura 34. Diagrama de Secuencia del Diseño: Eliminar Suscripción

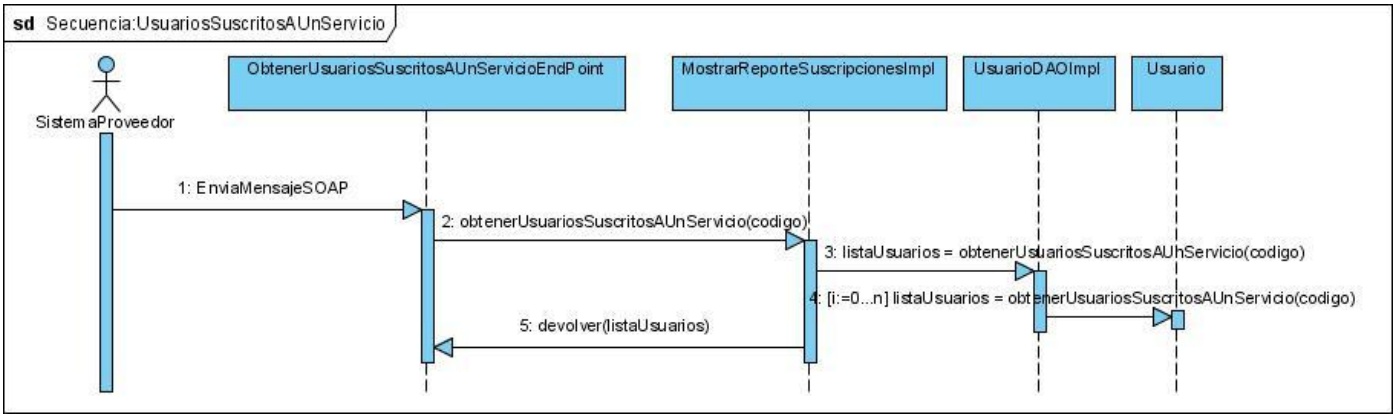

Figura 39. Diagrama de Secuencia del Diseño: Reportes de Suscripciones (Obtener usuarios suscritos a un servicio)

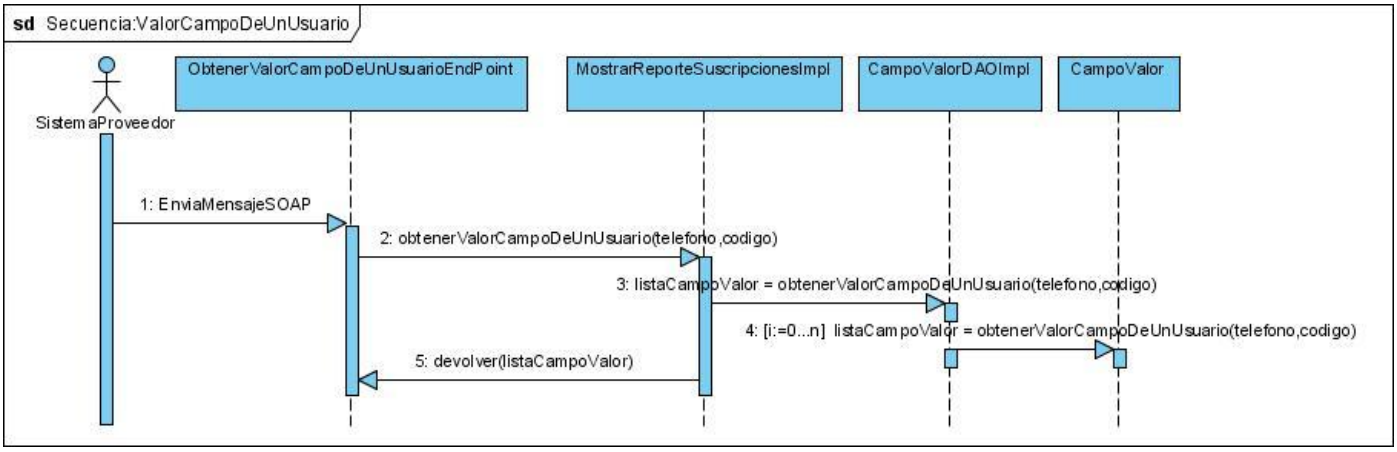

Figura 40. Diagrama de Secuencia del Diseño: Reportes de Suscripciones (Obtener valor campo de un usuario)

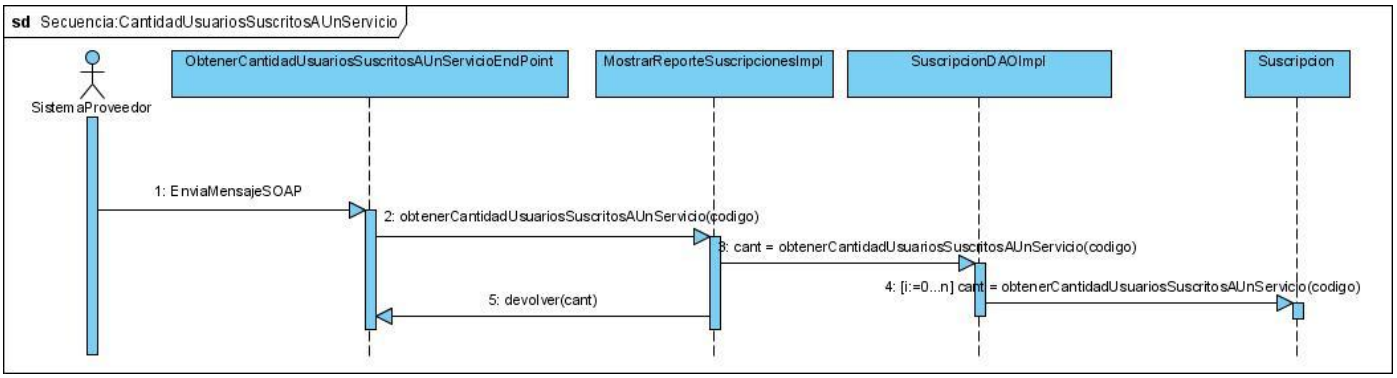

Figura 41. Diagrama de Secuencia del Diseño: Reportes de Suscripciones (Obtener cantidad de usuarios suscritos a un servicio)

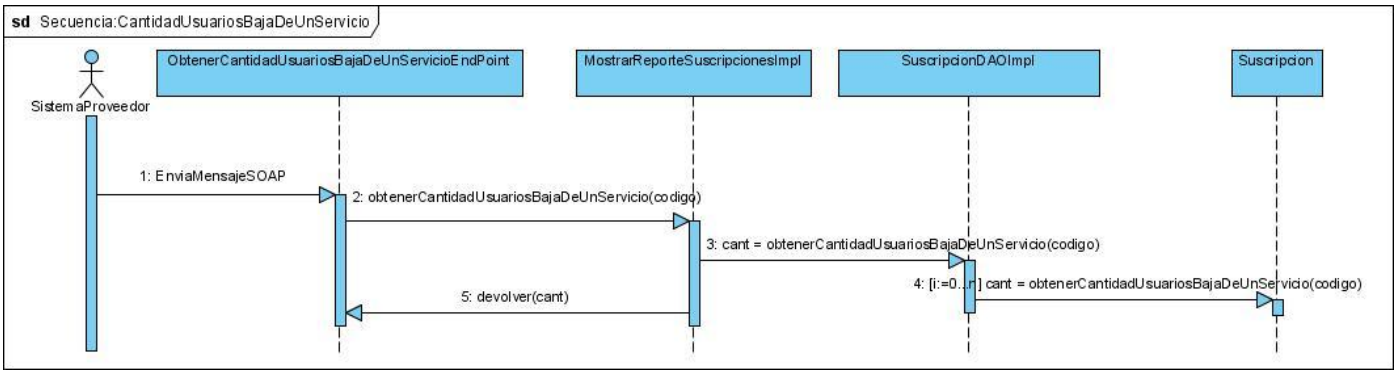

Figura 42. Diagrama de Secuencia del Diseño: Reportes de Suscripciones (Obtener cantidad de usuarios de baja de un servicio)