Universidad de las Ciencias Informáticas

Facultad 2

Sistema de Mensajería Empresarial basado en SMS

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN

CIENCIAS INFORMÁTICAS

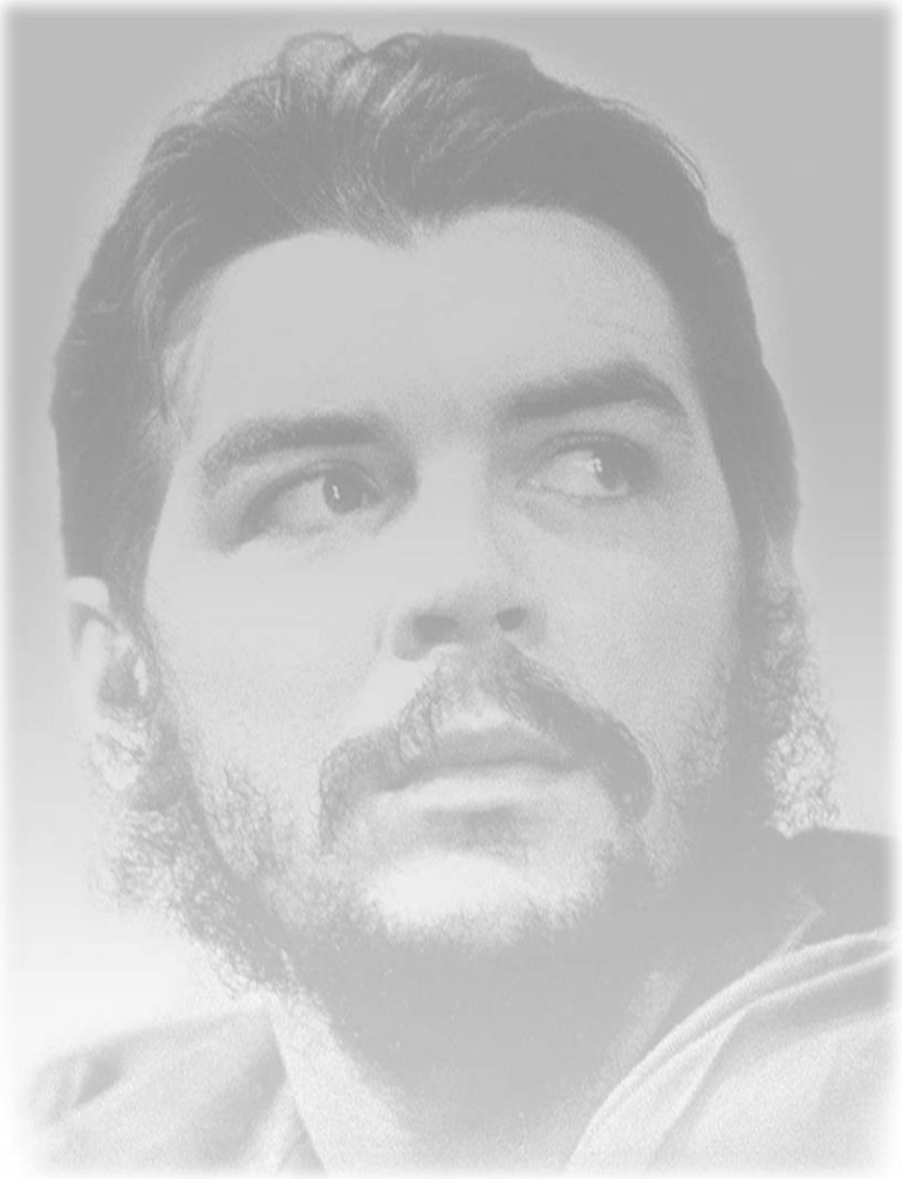
Autor(es):
René Manuel Sánchez Acevedo
Iliana Durand Silva

Tutor:

Ing. Damián Ilizastegui Arriba

Ciudad de La Habana, 2010

"Año 52 de la Revolución"



Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte.

Che

DECLARACIÓN DE AUTORÍA

Declaramos que somos los autores de este trabajo y autorizamos a la Facultad 2 de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2009.

AUTOR
René Manuel Sánchez Acevedo

AUTOR
Iliana Durand Silva

TUTOR
Damián Ilizastegui Arriba

AGRADECIMIENTOS

A mis padres porque sin ellos no hubiese llegado a este momento.

A mi hermano por toda la confianza que me ha dado y su preocupación por mi tesis.

A mi segunda familia, mami Nena, Ivian y Ada por quererme tanto desde que tenía tres meses y medio, y a papi Ideliso que en paz descanse que tanto le hubiera gustado ver este momento.

A mi tutor que ha contribuido con la preparación de este trabajo.

A mi novio, Enrique, por su constante apoyo, sin él hubiera sido mucho más difícil.

A todos aquellos que, de una forma u otra, han contribuido a mi formación tanto profesional como personal durante estos seis años.

Muchas Gracias

ILIANA

Agradecer a todos mis compañeros y amigos, al profesor Abel Velásquez por ser ante todo un educador, a mi compañera de tesis y a los que brindaron su ayuda incondicional durante estos cinco años.

RENÉ

DEDICATORIA

Dedicada a mis queridos padres, y a mi hermano, por todo su amor y apoyo. Para darles un motivo más por el cual sentirse orgullosos. Que en sus recuerdos quede este momento, en el cual se hacen realidad también sus sueños.

ILIANA

Dedicada a los cuatro principales artífices de poder llegar a ser un profesional, a ellos les dedico este trabajo, mis padres en especial a mi madre Nieves María y a dos de mis hermanos queridos: Ilena Mendez y Pedro Luis.

A toda mi familia en sentido general, especialmente a mis tios René y Leticia.

RENÉ

RESUMEN

El Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC) de la República Bolivariana de Venezuela constituye una institución única de su tipo, posee como objetivo detectar, procesar y esclarecer hechos delictivos cometidos a diario.

Un estudio realizado en la institución demostró que presentaba problemas en las comunicaciones operativas con los funcionarios que se encuentran operando en las calles.

Como una solución, en el presente trabajo se desarrolló una Plataforma de Mensajería de Texto como parte de una Aplicación Web centralizada para el envío de información basado en Servicios de Mensajes Cortos (SMS) hacia los dispositivos móviles de los funcionarios operativos que se encuentran operando en las calles.

Palabras Claves: Plataforma de Mensajería de Texto, dispositivos móviles, funcionarios operativos.

ÍNDICE DE CONTENIDOS

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La República Bolivariana de Venezuela se encuentra inmersa en un proceso de transformaciones sociales que abarca diferentes esferas, entre ellas: la salud, la educación y la seguridad ciudadana. A esta última se le ha dado una gran importancia debido al alto índice de hechos delictivos. Estudios realizados reflejan cifras muy preocupantes en las ciudades venezolanas.

Como argumento de lo planteado se muestran estadísticas de los delitos registrados en Venezuela en la encuesta nacional de victimización, realizada entre septiembre y octubre de 2006 con una muestra de 5496 hogares:

Delito	**Tasa CMH(por cada cien mil habitantes)**	**Incremento (%)**
Robo	3881,5	43,1
Hurto en todas sus modalidades	2057,9	22,9
Amenazas	889,2	9,9
Estafa	203,0	2,2
Extorsión	54	0,6
Homicidio	49,6	0,5
Secuestro	38,2	0,4

Tabla 1: Estadísticas de hechos delictivos.

Por esta razón se ha comenzado a realizar una serie de cambios en el Ministerio del Poder Popular para las Relaciones del Interior y Justicia, al que pertenece el Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC) tratando por todos los medios posibles de mejorar la situación actual.

El CICPC es la institución que se encarga de investigar y registrar toda la información referente a los delitos que ocurren en la República Bolivariana de Venezuela. Este organismo cuenta con un Sistema de Investigación e Información Policial (SIIPOL), surgido como producto de la modernización realizada al Sistema Integrado de Información Policial e incrementando las prestaciones de servicios del antiguo.

Según estudios realizados en la situación general que tiene actualmente la institución, el sistema de comunicación que presenta con los funcionarios operativos que se encuentran operando en las calles se realiza mediante teléfonos celulares particulares y radios; dicho sistema presenta deficiencias debido a que todos los funcionarios no están dispuestos a emplear el saldo de sus teléfonos en la comunicación y no se cuenta con suficientes radios para todos.

Con el surgimiento de los convenios de la Alternativa Bolivariana para América Latina y el Caribe (ALBA) se han formalizado contratos entre la República de Cuba y la República Bolivariana de Venezuela. Dichos acuerdos están enmarcados en diferentes esferas productivas, dentro de las cuales se incluye la producción de software. La Universidad de las Ciencias Informáticas (UCI) lleva la delantera en esta rama de la producción y a ella se le han asignado la materialización de varios productos de software. Entre ellos se encuentra el Sistema de Investigación e Información Policial WEB (SIIPOL WEB) surgido en los marcos de este convenio. Este nuevo sistema proveerá al CICPC de una aplicación Web la cual contará con una Plataforma de Mensajería basada en SMS para darles indicaciones y proveer de información necesaria a los funcionarios operativos que se encuentran operando fuera de la institución.

Por tanto surge la necesidad de darle solución a la problemática planteada, definiendo como **problema científico** de la investigación: ¿Cómo lograr que los funcionarios operativos del CICPC reciban orientaciones e indicaciones inmediatas en las condiciones del campo, desde un centro de control de operaciones?

Tomando como **objeto de estudio** de esta investigación el proceso de comunicaciones operativas del CICPC y el **campo de acción** se enmarca en la mensajería SMS para el CICPC.

Se trazó como **objetivo general** el desarrollo de una Plataforma de Mensajería de Texto centralizada que garantice el envío de mensajes a los funcionarios operativos.

Como **objetivos específicos** se presentan**:**

- Elaborar el diseño teórico de la investigación.
- Establecer la lógica del negocio.
- Realizar el análisis y diseño de la Plataforma de Mensajería de la Aplicación Web del SIIPOL Móvil.
- Implementar la Plataforma de Mensajería dando respuesta a una parte de las funcionalidades de la Aplicación Web del SIIPOL Móvil.
- Validar la propuesta solución.

Para dar cumplimiento a estos objetivos, se han definido las siguientes **tareas investigativas**:

- Estudio de herramientas, tecnologías y metodologías de desarrollo de software.
- Análisis de la lógica del negocio a establecer.
- Estudio de la arquitectura del SIIPOL adaptándola al contexto del SIIPOL Móvil.
- Estudio de las pautas de diseño y codificación del SIIPOL Móvil.
- Estudio de los niveles de prueba para la validación del sistema.

Los **métodos científicos** utilizados son:

Teóricos:

- Analítico-Sintético: Está dado por el análisis de los documentos generados por los analistas para el SIIPOL Móvil: especificación de casos de usos, glosario de términos, diagramas de entidades, extrayendo y analizando los principales elementos relacionados con el objeto de estudio.

Empíricos:

- Entrevista: Entrevista con el analista del SIIPOL Móvil con el objetivo de obtener a través de una conversación planificada, información cualitativa de los eventos y fenómenos que puedan ocurrir en el sistema.

Idea a Defender: Con el desarrollo de una Plataforma de Mensajería de Texto podrá dársele solución al problema presentado en el proceso de comunicaciones operativas por los funcionarios del CICPC.

El contenido de la investigación que se presenta está desglosado en 5 capítulos, las conclusiones generales, recomendaciones, bibliografía utilizada, un glosario de términos y los anexos que están presentes en el desarrollo de la investigación.

Capítulo 1. "Fundamentación Teórica": Se incluyen los resultados del estudio sobre el estado del arte de los conceptos fundamentales que se tratan en el trabajo, así como los principales aspectos teóricos y el análisis de las principales herramientas a utilizar en el desarrollo de la aplicación.

Capítulo 2. "Características del Sistema": Se hace una descripción general de la propuesta de sistema y de cómo debe funcionar el negocio, también se definen los requisitos fundamentales que debe cumplir la aplicación que se va a desarrollar.

Capítulo 3. "Análisis y Diseño del Sistema": Este capítulo está centrado hacia como construir el software, se describen los principales artefactos generados durante el proceso de análisis y diseño regido por la metodología seleccionada en el diseño teórico.

Capítulo 4. "Implementación y Prueba": Se describen todos los elementos relacionados con la implementación del sistema, así como las pruebas que se le realizan al software.

Capítulo 5. "Estudio de Factibilidad": Se describen los beneficios tangibles e intangibles obtenidos durante el desarrollo del trabajo.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

INTRODUCCIÓN

En este capítulo se realiza una breve descripción de los Sistemas de Mensajería Empresarial que ofrecen guía y sustento en el desarrollo del trabajo. El enfoque está dirigido de acuerdo a las necesidades planteadas por los funcionarios del CICPC, por lo que no se hizo énfasis en el estudio de sistemas similares. Se hace referencia a la metodología y herramientas utilizadas, además se emplean aquellas tecnologías utilizadas en el proyecto SIIPOL antecedente previo del SIIPOL Móvil y que serán reutilizadas para el desarrollo de la aplicación.

1.1. SISTEMAS DE MENSAJERÍA EMPRESARIAL

La Mensajería Empresarial es una solución que permite a la empresa utilizar SMS como canal de comunicación interno con acceso a cuatro diferentes modalidades:

- **Móvil a Correo Electrónico:** Consiste en el envío de SMS desde el celular hacia una cuenta de correo específica (la cual debe configurarse en el servicio) para que los empleados se comuniquen a través de este medio.

- **Correo Electrónico a Móvil:** Permite el envío de SMS desde el servidor de correo de la empresa (previamente configurado en el servicio) a los celulares de los empleados.

- **Aplicación a Móvil:** Permite enviar SMS desde una aplicación de la empresa, en forma automática para alarmas, recordatorios o eventos recurrentes. Este servicio es a la medida de cada empresa, ya que cada empresa realiza el desarrollo de una aplicación simple para conectarse al servicio. Por ejemplo: Alarmas, envío de SMS en caso de falla.
- **Web a Móvil:** Es una interfaz que permite enviar SMS desde una página Web a uno o varios usuarios contando con las siguientes funcionalidades:

    - ✓ Envío de SMS de forma programada o inmediata.
    - ✓ Creación de grupos de usuarios permitiendo así segmentar la información a enviar, por ejemplo, envío dirigido solo a un área de la empresa.

- ✓ Permite programar SMS para que sean enviados a una fecha y hora específica.
- ✓ Los accesos a la interfaz pueden realizarse bajo 2 distintos perfiles:

    1. Acceso como Administrador de la Empresa para administrar el servicio contratado así como enviar SMS.
    2. Acceso como Operador para envío de SMS y obtención de reportes.

- ✓ Reportes de SMS enviados (por usuario, estado de entrega, fechas, grupos, entre otros).

Esta última modalidad reúne las funcionalidades necesarias como solución a los problemas existentes en el CICPC, por estas razones el desarrollo del sistema está enfocado en el sistema de mensajería empresarial: Web a Móvil.

La puesta en marcha de un sistema de mensajería empresarial basado en SMS proporciona numerosos beneficios, tales como la optimización del tiempo por lo cual mejora la comunicación, permite comunicación en tiempo real con los empleados dentro y fuera de la oficina, movilidad y reducción de costos operativos a la empresa por medio del envío de SMS.

1.2. EL CUERPO DE INVESTIGACIONES CIENTÍFICAS, PENALES Y CRIMINALÍSTICAS

El Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC), es el organismo de investigación criminal donde se registran hechos delictivos que ocurren en la República Bolivariana de Venezuela, en ella laboran funcionarios operativos, incluyendo los que se encuentran operando en las calles, encargados de detectar cualquier delito que ocurra.

1.3. EL SISTEMA DE INVESTIGACIÓN E INFORMACIÓN POLICIAL MÓVIL

El proyecto SIIPOL Móvil es un proyecto productivo creado con el objetivo de resolver algunas dificultades que impiden el proceso organizacional del CICPC. Los funcionarios de esta estructura no cuentan con una aplicación de mensajería de texto, que permita el desarrollo de las investigaciones penales, criminalísticas y forenses y el cumplimiento de las misiones trazadas. A raíz de los problemas planteados se lleva a cabo este proyecto de modernización del CICPC con el propósito de que los funcionarios adscritos a la

institución puedan recibir información en sus dispositivos móviles , ayudando de esta forma a agilizar el trabajo, y mejorar el nivel de respuestas a las necesidades de seguridad del ciudadano venezolano.

1.4. METODOLOGÍA DE DESARROLLO

El Proceso Unificado de Rational (RUP) es un proceso de desarrollo de software que junto al Lenguaje Unificado de Modelado (UML), constituye una metodología estándar muy utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos y es además, un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software.

- Se utiliza en proyectos que se desarrollan a largo plazo.
- Permite una mejor comunicación entre los integrantes del proyecto.
- Genera un volumen considerable de documentación, posibilitando que los cambios realizados en los miembros del equipo no resulten un problema para el avance del proyecto.
- Propone el desarrollo en ciclos e iteraciones con los artefactos que se generan, siendo esto un elemento importante para alcanzar una categoría de certificación en el desarrollo del software.
- Asegura la producción de software de calidad dentro de plazos y presupuestos predecibles.

Un proyecto realizado siguiendo RUP se divide en cuatro fases: Inicio, Elaboración, Construcción y Transición.

En cada fase se ejecutarán una o varias iteraciones (de tamaño variable según el proyecto), y dentro de cada una de ellas seguirá un modelo de cascada para los flujos de trabajo. RUP define nueve disciplinas a realizar en cada ciclo del proyecto, donde 6 de ellas son flujos de trabajo básicos y las otras disciplinas de soporte. (Ver Anexo 1)
La metodología escogida es RUP ya que tiene como una de sus características principales el desarrollo incremental que posee las ventajas siguientes:

- Los clientes no esperan hasta el fin del desarrollo para utilizar el sistema. Pueden empezar a usarlo desde el primer incremento.

- Los clientes pueden aclarar los requisitos que no tengan claros conforme ven las entregas del sistema.
- Se disminuye el riesgo de fracaso de todo el proyecto, ya que se puede distribuir en cada incremento.
- Las partes más importantes del sistema son entregadas primero, por lo cual se realizan más pruebas en estos módulos y se disminuye el riesgo de fallos.

Otra de las facilidades por la cual se escogió esta metodología fue debido a la distancia geográfica que existe entre el cliente y el equipo de desarrollo, para lo cual RUP genera la documentación necesaria para validar los artefactos, además de poseer varios elementos de planificación que permiten llevar el control del desarrollo del proyecto. Además se tiene en cuenta que el producto a entregar al cliente necesita tener documentación y una detallada guía de uso.

Es importante señalar que la metodología no recomienda que se lleven a cabo estrictamente todas las actividades y artefactos que se describen, sino por el contrario, se recomienda que en dependencia de las características del proyecto y de la organización se seleccionen los artefactos, actividades y roles que van a ser utilizados.

1.5. LENGUAJES

Modelado

El Lenguaje Unificado de Modelado (UML) es un lenguaje gráfico para especificar, construir, visualizar y documentar todas las disciplinas de un proyecto informático: desde el análisis con los casos de uso, hasta la implementación y configuración mediante los diagramas de despliegue.

UML permite:

- Especificar cuáles son las características de un sistema antes de su construcción.
- Construir sistemas diseñados a partir de modelos especificados.
- Visualizar gráficamente un sistema de manera que otros puedan entenderlo.
- Documentar los elementos gráficos del sistema desarrollados para futuras revisiones.

- Verificar y validar el modelo realizado.
- Generar código a partir de los modelos y viceversa.

Programación

Java es un lenguaje de programación de alto nivel, de propósito general y orientado a objetos; desarrollado por Sun Microsystems a mediados de los años 90. Java es el resultado de las características esenciales de otros lenguajes como C y C++.

Las características principales que ofrece son:

- Lenguaje Simple: Elimina las características menos usadas y más confusas de otros lenguajes como C++. Posee un reciclador de memoria dinámica (colector de basuras) que se ocupa de liberar grandes bloques de memoria.
- Distribuido: Proporciona una colección de clases para su uso en aplicaciones de red, permitiendo a los programadores acceder a la información con mucha facilidad; lo que contribuye a la creación de aplicaciones distribuidas.
- Robusto: Realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria, reduciendo considerablemente el tiempo empleado en el desarrollo de las aplicaciones.
- Multihilo: Permite muchas actividades simultáneas en un programa, de esta forma se mejora el rendimiento interactivo y el comportamiento en tiempo real.
- Seguro: Limita cualquier aplicación del tipo Caballo de Troya a través del Cargador de Clases, separando el espacio de nombres de los ficheros locales del sistema y el de los recursos procedentes de la red. Además, elimina los punteros para prevenir el acceso ilegal a la memoria.

1.6. PLATAFORMA DE DESARROLLO

Se entiende por plataforma el entorno de software común en el cual se desenvuelve la programación de un grupo definido de aplicaciones, ya sea de hardware o software, sobre el cual un programa puede

ejecutarse. Ejemplos típicos incluyen: arquitectura de hardware, sistema operativo, lenguajes de programación, Interfaz de programación de aplicaciones y sus librerías de tiempo de ejecución.

Plataforma Java

La plataforma Java es el nombre de un entorno o plataforma de computación originaria de la compañía Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el Lenguaje de programación Java u otros lenguajes. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino más bien una máquina virtual encargada de la ejecución de aplicaciones, y un conjunto de librerías estándar que ofrecen funcionalidad común. Hoy en día esta plataforma ha evolucionado en concordancia con el avance tecnológico y se ha convertido en una de las plataformas de programación más usadas por los desarrolladores. Su principal ventaja es que su entorno de desarrollo es independiente de la plataforma sobre la que se trabaje, es decir, sus aplicaciones son funcionales independientemente del sistema operativo sobre el que estén operando. Es toda una tecnología orientada al desarrollo de software con el cual podemos realizar cualquier tipo de programa. Ha cobrado mucha importancia en el ámbito web con su plataforma JEE (Java Enterprise Edition).

1.7. HERRAMIENTAS

Ingeniería de Software Asistida por Ordenador (CASE)

Visual Paradigm para UML (VP-UML) es una herramienta CASE multiplataforma (Windows/Linux/Mac OS X) que soporta el ciclo de vida completo del desarrollo de software. Está diseñado para un amplio rango de usuarios, incluyendo ingenieros de software, analistas de sistema, analistas de negocio, arquitectos de sistema y quienes estén interesados en la construcción de sistemas de software confiables mediante el uso de la Orientación a Objetos. Facilita una rápida construcción de aplicaciones de calidad y a un menor coste. Visual Paradigm para UML soporta un conjunto de lenguajes (Java, C + +, PHP, Ada y Python), tanto en generación de código como ingeniería inversa. Entre sus principales características se pueden señalar:

- Facilita la comunicación de todo el equipo de desarrollo mediante el uso de un lenguaje estándar común.

- Posibilita el desarrollo de la ingeniería directa e inversa.
- Durante todo el ciclo de desarrollo el modelo y el código permanecen sincronizados, permitiendo la generación de código a partir de diagramas y viceversa.
- Permite la generación de bases de datos a partir de la transformación de diagramas de Entidad-Relación en tablas de base de datos y viceversa.

Entorno de Desarrollo Integrado (IDE): Eclipse 3.3.0

Eclipse es una plataforma universal para integrar herramientas de desarrollo, posee una arquitectura abierta y basada en plug-ins. La principal característica de Eclipse es la extensibilidad, porque permite integrar diversos lenguajes sobre un mismo IDE e introducir otras aplicaciones accesorias que resultan útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces y ayuda en línea para librerías.

De manera general, presenta las siguientes características:

- Es multiplataforma (GNU/Linux, Solaris, Mac OSX).
- Es soportado para distintas arquitecturas (x86, 64 bits, entre otros).
- Posee una estructura de plug-in que hace sencillo añadir nuevas características y funcionalidades.
- Dispone de un Control de versiones integrado a Subversión.
- Incluye muchas utilidades de edición que ayudan al programador a desarrollar el producto con rapidez, algunas son: resaltado de sintaxis, autocompletado de código, tabulador de un bloque de código seleccionado y formateado automático de código, tributando a la obtención de código de mayor calidad.
- Posee asistentes para la creación, exportación e importación de proyectos y para generar esqueletos de códigos.

Sistema Gestor de Bases de Datos: Oracle

Oracle es un Sistema Gestor de Bases de Datos relacional fabricado por Oracle Corporation. Se considera uno de los sistemas de bases de datos más completos, destacando su soporte de transacciones, estabilidad y escalabilidad. Ha sido diseñado para que las organizaciones puedan controlar y gestionar grandes volúmenes de contenidos no estructurados en un único repositorio con el objetivo de reducir los costes y los riesgos asociados a la pérdida de información.

Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales, por norma general. En el desarrollo de aplicaciones web pasa lo mismo: como es un sistema muy caro no está tan extendido como otros, por ejemplo, Access, MySQL y SQL Server. Aún así fue seleccionado para la de modernización del CICPC y pagada su licencia dentro del presupuesto del proyecto, basando esta decisión en la seguridad que aporta y las muchas ventajas que posee, entre ellas:

- Puede ejecutarse en todas las plataformas, desde una simple computadora personal hasta un supercomputador utilizado como servidor. El software del servidor puede ejecutarse en multitud de sistemas operativos.
- Oracle soporta todas las funciones que se esperan de un buen servidor de bases de datos: un lenguaje de diseño de bases de datos muy completo (PL/SQL, lenguaje de quinta generación, muy potente para tratar y gestionar la base de datos) que permite implementar diseños con disparadores y procedimientos almacenados, con una integridad referencial declarativa muy potente.
- Permite el uso de particiones para la mejora de la eficiencia, de replicación e incluso ciertas versiones admiten la administración de bases de datos distribuidas.
- Este sistema ha comenzado a evolucionar en cuanto al mundo objetual, añadiendo tipos de clases, referencias, tablas anidadas, matrices y otras estructuras de datos complejas.

Control de Versiones: TortoiseSVN 1.6.0

Es un cliente gratuito de código abierto para el sistema de control de versiones Subversion. Se encarga del manejo de ficheros y directorios a lo largo del tiempo, los cuales se almacenan en un repositorio central.

Entre las características de TortoiseSVN se destacan:

- Integración con el explorador de Windows.
- Íconos sobreimpresionados: el estado de cada carpeta y fichero versionado se indica por pequeños íconos sobreimpresionados. De esta forma, se puede ver fácilmente el estado en el que se encuentra la información.
- Fácil acceso a los comandos de Subversion.
- Proporciona una serie de herramientas internas:
    - TortoiseMerge: permite ver las diferencias entre ficheros de texto y fusionar los cambios.
    - TortoiseBlame: hace más fácil la lectura de los ficheros de autoría.
    - SubWCRev: es un programa de consola para Windows que puede utilizarse para leer el estado de una copia de trabajo local y opcionalmente realizar sustituciones de palabras claves en un fichero.

Servidor Web: Apache Tomcat 6.0.14

Es un software de código fuente que implementa las tecnologías Java Server Page (JSP) y Java Servlet**.** Está desarrollado en un entorno abierto y participativo y publicado bajo la licencia del software de Apache. Es importante destacar algunas de las características por la cual fue escogido este servidor WEB:

- Implementado de Servlet 2.5 y JSP 2.1
- Soporte para Lenguaje Unificado de Expresión 2.1
- Diseñado para funcionar en Java SE 5.0 y posteriores
- Soporte para Comet a través de la interfaz CometProcessor.

1.8. FRAMEWORKS

Un framework es una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En términos informales, un framework se puede considerar como una

aplicación genérica incompleta y configurable a la que se puede añadir las últimas piezas para construir una aplicación concreta.

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

Java soporta gran números de frameworks especializándose cada uno de ellos en una parte específica del sistema, ya sea la capa de presentación, la capa de la lógica de negocio o la capa de acceso a datos. A continuación se presentan los principales frameworks por cada una de las capas lógicas de la aplicación que se seleccionaron para integrarlos y dar soporte a la arquitectura del proyecto.

Presentación

Java Server Faces (JSF) es un framework que facilita la construcción de aplicaciones siguiendo el patrón Modelo Vista Controlador (MVC), modelo de componentes orientado a objetos, conversión de tipos, separación de responsabilidades, desarrolladores de componentes, desarrolladores de lógica de aplicación, montadores de páginas, poderoso sistema de navegación declarativa, uso de simples clases java como controladores, fácil incorporación de potencialidades Ajax, posee un conjunto prefabricado de componentes de interfaz de usuario y un modelo de programación orientado a eventos.

Las siguientes características diferencian a JSF de otros marcos de trabajo:

- Control de Beans de Respaldo: Los beans de respaldo son componentes JavaBeans asociados con componentes de interfaz de usuario (UI) utilizados en la página.
- Modelo de componentes UI extensible: Los componentes UI de JSF son elementos configurables, reutilizables que componen las interfaces de usuario de aplicaciones JSF.
- Modelo de Renderizado Flexible: Un renderizador (sistema que se encarga de convertir los componentes propios de JSF a componentes HTML clásicos) separa la vista y la funcionalidad de los componentes UI. Se pueden crear y utilizar varios renderizadores para definir diferentes apariencias del mismo componente para el mismo o diferentes clientes.
- Modelo de Conversión y Validación Extensible: Basados en los convertidores y validadores estándar, se pueden desarrollar convertidores y validadores personalizados, que proporcionan un mejor modelo de protección.

Ajax4JSF es una extensión de código abierto del estándar JSF que se integra a este con gran facilidad, evita escribir código Java Scripts al brindar una amplia gama de componentes, permite agregar capacidades AJAX (del término en inglés Asyncronous Java Script and XML) incluso a aplicaciones JSF ya creadas. En fin, las prestaciones que brinda la poderosa combinación de JSF con AJAX, hacen que sea más atractiva por la comodidad que supone para el desarrollador y además por integrarse eficientemente con otros frameworks.

RichFaces es una librería de componentes para JSF y un avanzado framework para la integración de AJAX con facilidad en la capacidad de desarrollo de aplicaciones empresariales. RichFaces facilita el desarrollo y permite la reusabilidad de componentes de una manera estándar y bastante bien documentada, pues los desarrolladores podrían crear sus propios componentes JSF artesanales con funcionalidad Ajax implementada con scripts propios, sin embargo RichFaces posibilita la programación con un extenso número de componentes listos para utilizarse. RichFaces también aprovecha al máximo los beneficios de JSF framework incluyendo, la validación y conversión de instalaciones, junto con la gestión estática y dinámica de los recursos. Ajax4JSF y RichFaces se integran totalmente en la arquitectura de JSF y hereda las funcionalidades de sus etiquetas dotándolas con tecnología Ajax de forma limpia y sin añadir código Java script. Mediante los mismos podemos variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones al servidor automáticas y control de cualquier evento de usuario. (Ver Anexo 2)

Lógica de Negocio

Spring es un framework de código abierto para desarrollo de aplicaciones en plataforma J2EE. Spring interviene en todas las capas arquitectónicas de una aplicación J2EE, brinda soporte a varios frameworks de presentación, entre ellos Java Server Faces (JSF), y se integra con varios frameworks de acceso a datos. (1)

Spring promueve el bajo acoplamiento de las clases conformantes de la solución por medio de la técnica conocida como inversión del control (IoC). Soporta la Programación Orientada a Aspectos (AOP) que complementa la Programación Orientada a Objetos (POO), proponiendo otra manera de pensar sobre la estructura de un programa. Mientras que la POO descompone las aplicaciones en una jerarquía de objetos, AOP descompone los programas en aspectos o preocupaciones. Dichas preocupaciones se

convierten en servicios del sistema, separados de la lógica de negocio y que se ejecutan de manera trasversal a la funcionalidad base, lo que proporciona una definición de responsabilidades superior. Spring básicamente es un contenedor, que se encarga de gestionar y administrar el ciclo de vida y configuración de las clases de la aplicación. (Ver Anexo 3)

Acceso a Datos

Hibernate es una solución ORM (Mapeo Objeto-Relacional) para Java, como todas las herramientas de su tipo. Hibernate busca solucionar el problema de la diferencia entre el modelo orientado a objetos y el usado en las bases de datos modelo relacional mediante archivos declarativos. Es una capa de persistencia objeto-relacional y un generador de sentencias Lenguaje de Consulta Estructurado (SQL). Permite diseñar objetos persistentes que podrán incluir polimorfismos, relaciones, colecciones, y un gran número de tipos de datos. Hibernate convierte los datos entre los tipos utilizados por Java y los definidos por SQL.

Hibernate libera al desarrollador del manejo manual de los datos que resultan de la ejecución de sentencias SQL, manteniendo la portabilidad entre todas las bases de datos con un ligero incremento en el tiempo de ejecución. Hibernate permite expresar consultas en su propio Lenguaje de Consulta de Hibernate (HQL), así como en SQL nativo, o con uno orientado a objetos. (2)

Para comprender mejor la arquitectura de Hibernate se recomienda ver la figura que se muestra en el anexo 4.

JUnit

Es un framework simple y de código abierto 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 seleccionado para la realización de dos tipos de pruebas fundamentalmente:

- Pruebas unitarias: consisten en probar la correcta funcionalidad del módulo en cuestión como si actuara independiente del resto.
- Pruebas de integración: se prueba la correcta integración de cada módulo.

El framework provee al desarrollador de herramientas, clases y métodos que le facilitan la tarea de realizar pruebas en el sistema y así asegurar su consistencia y funcionalidad, determinando si el código cumple con las funcionalidades específicas para el que fue realizado, evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera y comprobar la correcta funcionalidad de las aplicaciones.

Es además un medio para controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

Razones por las que utilizar JUnit:

- **Los test JUnit permiten escribir código más rápidamente e incrementa su calidad.**

  Al escribir tests utilizando JUnit, se pierde menos tiempo depurando, y se goza de la confianza de que los cambios del código realmente funcionan. Esta confianza permite ser más agresivo con la refactorización del código y la adición de nuevas características.

- **Los tests JUnit pueden componerse como un árbol de suites de tests.**

  Los tests JUnit se pueden organizar en suites de tests que contienen ejemplares de tests e incluso otras suites. El comportamiento compuesto de los tests JUnit permite ensamblar colecciones de tests y automáticamente hacer un test regresivo de toda la suite de tests con un sólo paso. También se puede ejecutar los tests de cualquier capa dentro del árbol de suites.

- **Los tests JUnit se escriben en Java.**

  Testear software Java utilizando tests Java forma una similitud entre el test y el código testeado. El test se convierte en una extensión del software general y el código se puede reconstruir partiendo de los tests. El compilador Java ayuda al proceso de testeo realizando el chequeo de la sintaxis estática de las unidades de testeo y asegurándose de que el contrato de interfaz del software se está obedeciendo.

- **Los tests JUnit son tests del desarrollador.**

Los tests JUnit son tests altamente localizados escritos para mejorar la productividad del desarrollador y la calidad del código. Al contrario que los tests funcionales, que tratan al sistema como una caja negra y aseguran que el software funciona como una totalidad, las unidades de tests están escritas para probar los bloques de construcción básicos del sistema desde dentro. Los desarrolladores escriben y poseen los tests JUnit. Cuando se completa una iteración de desarrollo, los tests se convierten en parte y parcela del producto entregado como una forma de comunicación.

1.9. APIs

Una Interfaz de Programación de Aplicaciones (API) representa una interfaz de comunicación entre componentes de software. Se trata del conjunto de llamadas a bibliotecas que ofrecen acceso a determinados servicios y representa un método para conseguir abstracción en la programación. (3)

JSR 212 SAMS

JSR 212 es un API del lado del servidor para servicios móviles. Por servicios móviles entendemos mensajería, servicios de ubicación geográfica, servicios de presencia y administración centralizada de dispositivos. Estos servicios se pueden acceder desde extremos móviles Java ME, pero también desde fijos Java SE/EE.

Responde a la necesidad de un acceso unificado desde el lado del cliente a dichos servicios, a la vez que aumenta el ingreso promedio por usuario con servicios del lado del servidor.

El objetivo principal de JSR 212 es entonces definir un acceso unificado a los servicios, con dos metas en orden de prioridad: Especificar un núcleo de funcionalidad común de API, que será utilizado por todos los servicios SAMS y especificar la API de mensajería SMS y MMS. (4)

RXTX comm

Rxtx es una librería nativa de java que provee comunicación para el puerto comm o serie, todas las versiones se encuentran bajo la Licencia Pública General Reducida (LGPL) de GNU.

1.10. ARQUITECTURA TÉCNICA

Con el objetivo de dar cumplimiento a los requisitos funcionales y no funcionales del SIIPOL Móvil, se tiene como propuesta de solución para el sistema una aplicación web con la estructura que se muestra en la figura.

Figura 1: Propuesta de Solución.

Se propone una Plataforma de Mensajería contenida en la Aplicación Web con arquitectura en tres capas (Presentación, Negocio y Datos) y basada en tecnología Java, a la que se conectarán los clientes mediante el Protocolo Seguro de Transferencia de Hipertexto (HTTPS). (Ver Anexo 5)

El estilo arquitectónico en capas da ventajas sustanciales como lo son: la centralización de los aspectos de seguridad, transaccionalidad y no replicación de lógica de negocio en los clientes. Esto posibilita, que las modificaciones y mejoras sean automáticamente aprovechadas por el conjunto de los usuarios reduciendo los costes de mantenimiento y garantizando a su vez una mayor sencillez de los clientes.

CONCLUSIONES

En este capítulo se realizó una descripción de los Sistemas de Mensajería Empresarial enfocándose el trabajo en la modalidad Web a Móvil, el cual reúne características muy novedosas que cubre las

necesidades actuales del CICPC de la República Bolivariana de Venezuela. También ha quedado reflejado el estudio realizado a las herramientas, tecnologías y metodología que sirven como punto de partida para la construcción de un software que cumpla con los objetivos propuestos.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

INTRODUCCIÓN

En este capítulo se hace referencia a las características generales del sistema, se realiza el proceso de captura de requisitos funcionales y no funcionales a los que se debe dar cumplimiento, se define el objeto de automatización y la propuesta de sistema que será desarrollada.

2.1. OBJETO DE AUTOMATIZACIÓN

Se pretende automatizar los procesos que intervienen en el envío de información del funcionario operador al funcionario operativo referente a las comunicaciones operativas.

2.2. PROPUESTA DE SISTEMA

A partir de los objetivos del trabajo, se pretende desarrollar un sistema de mensajería empresarial basada en la modalidad Web a Móvil. Como ya se había explicado en el capítulo anterior, es una interfaz que permite enviar SMS a uno o varios usuarios móviles. Los funcionarios operadores del centro de control del CICPC, a través de una Plataforma de Mensajería de Texto enviarán mensajes a los funcionarios operativos para ofrecer las indicaciones necesarias. Esta nueva propuesta de sistema permitirá que el proceso de comunicaciones operativas se realice de forma rápida.

2.3. MODELO DE NEGOCIO

El negocio propuesto aborda los temas relacionados con la comunicación entre los funcionarios operadores que realizan sus labores dentro de su despacho, y los funcionarios operativos que se encuentran operando fuera de ellos. El negocio actual es manejado por un proceso fundamental, el proceso de indicaciones y orientaciones desde el despacho. Este proceso es iniciado por un evento o circunstancia que indique la necesidad del envío de un mensaje operativo, hacia los funcionarios que operan fuera de su despacho, para darles indicaciones y proveerles de información necesaria por parte del jefe de guardia.

La estructura del modelo de casos de uso de negocio de SIIPOL Móvil está concebida teniendo en cuenta el proceso de indicaciones y orientaciones desde el despacho hacia los funcionarios que operan fuera de él, este proceso es el encargado de almacenar los subprocesos relacionados con el envío de orientaciones e indicaciones hacia los funcionarios.

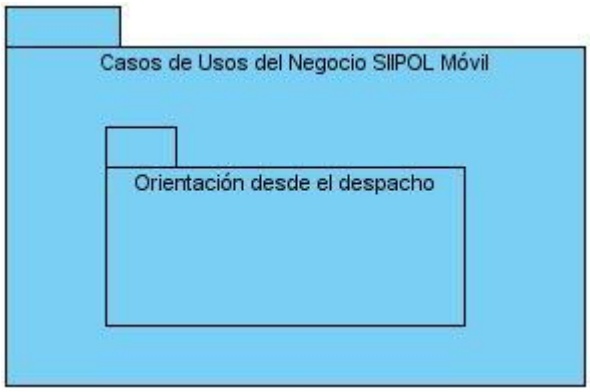


Figura 2: Diagrama de paquetes de Casos de Uso de Negocio.

2.3.1 ACTORES Y TRABAJADORES

Actores del negocio

- Evento: Enviar mensaje operativo (Inicia)

Circunstancia o evento que indica la necesidad del envío de un mensaje desde el despacho hacia los funcionarios que se encuentran operando fuera.

- Funcionario Operativo:

Es beneficiado por el Caso de Uso, ya que es el que recibe la información enviada.

Trabajadores del negocio

- Jefe de Guardia

Procesa el envío del mensaje operativo dado por el evento o circunstancia que indique el envío.

2.3.2 DIAGRAMA DE CASOS DE USO DEL NEGOCIO

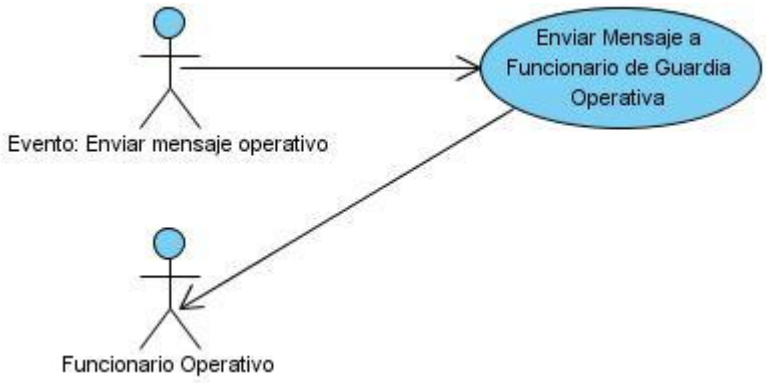


Figura 3: Diagrama de Casos de Uso del Negocio.

2.3.3 DESCRIPCIÓN TEXTUAL DEL CASO DE USO DEL NEGOCIO

Caso de Uso del Negocio	Enviar Mensaje a Funcionario de Guardia Operativa
Objetivo	Que el Jefe de Guardia dada una circunstancia o necesidad pueda hacer llegar información necesaria a un funcionario operativo que se encuentre de guardia operativa fuera del despacho.
Actores	Evento: Enviar mensaje operativo Funcionario Operativo
Trabajadores	Jefe de Guardia
Complejidad	Media
Descripción	El Jefe de Guardia dado una circunstancia o necesidad realiza una llamada telefónica al

	funcionario en caso de saberse su número telefónico, o realiza una llamada a través de radio trasmisor si el funcionario porta algún dispositivo de este tipo. Una vez establecida la comunicación pide confirmación de estar hablando con el funcionario deseado, comprobado que sea así se trasmite la información necesaria.

Tabla 2: Descripción del caso de uso del negocio.

2.3.4 DIAGRAMA DE ACTIVIDADES

Un diagrama de Actividad demuestra la serie de actividades que deben ser realizadas en un uso-caso, así como las distintas rutas que pueden irse desencadenando en el uso-caso. Es utilizado en conjunción de un diagrama uso-caso para auxiliar a los miembros del equipo de desarrollo a entender como es utilizado el sistema y cómo reacciona en determinados eventos. (5)

En este caso el actor que inicia es el evento Enviar mensaje operativo, el actor beneficiado es el Funcionario Operativo y por el último el trabajador es el Jefe de guardia. (Ver Anexo 6)

2.3.5 MODELO DE OBJETOS

Como se puede apreciar en el diagrama de actividades del caso de uso, el trabajador no se relaciona con al menos una entidad del negocio, por tanto el trabajo no presenta un Modelo de objetos.

2.4. REQUERIMIENTOS DEL SISTEMA

Una vez que se realiza el modelado del negocio se puede comenzar a ejecutar el proceso de captura de requisitos del sistema. Los requisitos son la condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente software para satisfacer un contrato, estándar, u otro documento impuesto formalmente. A continuación se exponen los siguientes requisitos que debe cumplir la aplicación a

desarrollar. La identificación de estos se ha realizado a partir de las necesidades reales planteadas por los funcionarios del CICPC.

2.4.1 REQUERIMIENTOS FUNCIONALES

Los requisitos funcionales son condiciones o capacidades que el software debe poseer para cumplir con las exigencias del cliente de acuerdo a sus necesidades. A continuación se enumeran los requerimientos que el sistema debe cumplir:

RF 1: Redactar Mensaje.
RF 2: Gestionar Mensaje en la Bandeja de Salida.
RF 3: Gestionar Mensaje en la Bandeja de Borradores.
RF 4: Gestionar Mensaje en la Bandeja de Enviados.
RF 5: Gestionar Mensaje en la Papelera de Reciclaje.
RF 6: Ver Mensaje.
RF 7: Crear Contacto de Usuario.
RF 8: Buscar Funcionario en SIIPOL.
RF 9: Ver Detalles de Contacto de Usuario.
RF 10: Modificar Contacto de Usuario.
RF 11: Listar Contactos de Usuario.
RF 12: Eliminar Contacto de Usuario.
RF 13: Crear Grupo de Contactos de Usuario.
RF 14: Ver Detalles de Grupo de Contactos de Usuario.
RF 15: Modificar Grupos de Contactos de Usuario.
RF 16: Listar Grupo de Contactos de Usuario.
RF 17: Eliminar Grupo de Contactos de Usuario.

2.4.2 REQUERIMIENTOS NO FUNCIONALES

Los requisitos no funcionales son propiedades o cualidades que debe presentar el sistema a desarrollar, cumpliendo de esta forma con las exigencias del cliente.

Disponibilidad

RNF 1: El sistema estará disponible durante 24 horas, los 7 días de la semana, todos los días del año.

Diseño de Interfaz

RNF 2: Se colocará la menor cantidad posible de campos en los formularios del sistema, solo aquellos que sean necesarios y suficientes.
RNF 3: Los campos de texto tendrán un tamaño estándar de acuerdo con el espacio con que se cuente en el área de la página y en la medida que se llene esa área primaria agregar la barra de desplazamiento vertical.
RNF 4: Los campos obligatorios serán señalados de alguna forma, preferiblemente colocándoles un asterisco (*) de color negro junto al componente.

Fiabilidad

RNF 5: El sistema necesitará un espacio de almacenamiento en base de datos de 94 GB para los datos de la aplicación y 30 GB para los ficheros de Archive_Log, Redolog y posibles ampliaciones. En total la base de datos tendrá un aproximado de 124 GB.
RNF 6: El sistema tendrá un respaldo de la información del centro de datos, permitiendo la recuperación ante la pérdida parcial o total de la información.
RNF 7: El sistema será capaz de hacer copias de respaldo de la base de datos a intervalos de tiempo fijos o a voluntad del usuario administrador.
RNF 8: El sistema brindará una manera óptima de recuperación ante fallos en el centro de datos en tiempo de ejecución de manera transparente al usuario sin necesidad de detener el servicio de la aplicación. Para ello se potenciará el uso de la configuración de los discos en espejo (RAID 1) para la protección de los datos, la cual es una de las mejores soluciones disponibles actualmente en el mercado.
RNF 9: El sistema hará un balanceo de carga entre los diferentes módems GSM conectados a él, de forma tal que los mensajes SMS tengan que esperar el menor tiempo posible para ser enviados.

Seguridad

RNF 10: El sistema concederá acceso al mismo a partir de un nombre de usuario y una contraseña.
RNF 11: El sistema solicitará el reacceso del usuario después de determinado tiempo de inactividad por parte de este.
RNF 12: El sistema implementará el uso de campos obligatorios y validaciones para garantizar la integridad de la información que se introduce por el usuario.
RNF 13: El sistema manejará mecanismos de encriptación de determinados datos que sean transmitidos por la red y que así lo requieran.

Rendimiento

RNF 14: El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria. Para ello se potenciará, como regla, guardar los datos y recursos de alta demanda en la memoria caché.

Restricciones de diseño

RNF 15: El sistema se implementará usando la plataforma JAVA.
RNF 16: El sistema estará basado en un estilo arquitectónico en capas. Las capas estarán distribuidas de la siguiente manera:

RNF 16.1: Capa de Presentación: incluye los componentes de interfaz de usuario (páginas web y componentes visuales) para interactuar y mostrar el resultado de las peticiones de servicio que ofrece la Capa de Aplicación, formateados para los distintos tipos de interfaces de usuario.
RNF 16.2: Capa de Aplicación: implementa la lógica de negocio. Esta capa a su vez se dividirá en dos subniveles:

- Fachada de Negocio: proporciona interfaces simples y débilmente acopladas a los clientes.
- Lógica de Negocio: implementa las clases presentes en el esquema conceptual.

RNF 16.3: Capa de Persistencia: implementa la persistencia y el acceso a los datos ocultando los detalles de los repositorios de datos a los niveles superiores. Este nivel estará constituido a su vez por dos subniveles:

- Capa de Acceso a Datos, donde se implementan componentes que se encargan de acceder a la base de datos para leer y escribir el estado de los objetos de negocio, independizando la aplicación del acceso a la base de datos.
- Capa de Datos, que se encarga de almacenar los datos de la aplicación en una base de datos específica.

RNF 17: El sistema usará el Framework de Presentación JSF para manejar la Capa de Presentación.

RNF 18: El sistema usará el Framework Spring para manejar la Capa de Lógica de Negocio, así como para el manejo de transacciones, concurrencia y seguridad.

RNF 19: El sistema usará el Framework de Persistencia Hibernate para manejar la capa de Acceso a Datos.

Interfaz de software

RNF 20: El sistema será diseñado siguiendo los principios de diseño orientado a objeto.

RNF 20.1: El Principio de Única Responsabilidad: Una clase solo puede tener una razón para cambiar.

RNF 20.2: El Principio de Sustitución de Liskov: Los subtipos deben ser sustituibles por sus supertipos.

RNF 20.3: El Principio de Inversión de la Dependencia: Los módulos de alto nivel no deben depender de los módulos de bajo nivel, ambos deben depender de las abstracciones. Las abstracciones no deben depender de los detalles, los detalles deben depender de las abstracciones.

RNF 20.4: El Principio de Segregación de la Interfaz: Los clientes no deben ser forzados a depender de interfaces que no utilizan.

RNF 20.5: El Principio abierto/cerrado: Las clases o componentes deben estar abiertas para su extensión pero cerradas para su modificación.

RNF 21: El sistema tendrá un alto desempeño con el objetivo de que el tiempo de respuesta sea lo más bajo posible.

Hardware

RNF 22: El sistema será desplegado en un clúster de 2 servidores HP Integrity RX6600, con sistema operativo HP-UX 11i v2 Mission Critical Operating Enviroment.

RNF 23: La Base de Datos estará en un clúster de 2 servidores HP Integrity RX7640.

2.5. MODELO DE CASOS DE USO DEL SISTEMA

El modelo de casos de uso describe la funcionalidad propuesta del nuevo sistema, donde cada caso de uso representa una unidad discreta de interacción entre un usuario y el sistema. Cada caso de uso tiene una descripción que describe la funcionalidad que se construirá en el sistema propuesto. Un caso de uso puede incluir la funcionalidad de otro caso de uso o extender a otro caso de uso con su propio comportamiento; un ejemplo de este último se presenta en uno de los diagramas de casos de uso desarrollados en el trabajo.

Para el desarrollo de la plataforma se hizo énfasis en el estudio de los Casos de Usos que fueron considerados como los más significativos, los restantes se podrán consultar en la sección Anexos del presente documento. A continuación se especifican los Casos de Uso más significativos:

- Redactar Mensaje.
- Gestionar Mensaje en la Bandeja de Borradores.
- Crear Contacto de Usuario.
- Crear Grupo de Contactos de Usuario.
- Buscar Funcionario en SIIPOL.

2.5.1 DESCRIPCIÓN GENERAL DEL SISTEMA

La aplicación Web contará con una Plataforma de Mensajería de Texto dividida en dos subsistemas: el subsistema Mensajes incluye funcionalidades afines a la gestión de los mensajes y el subsistema Contactos engloba la gestión de los contactos y grupo de contactos.

2.5.2 ACTOR DEL SISTEMA A AUTOMATIZAR

Gestor de Mensaje (Inicia): Es el encargado de realizar las funcionalidades pertenecientes a la gestión de mensajes, contactos y grupos de contactos.

2.5.3 DIAGRAMAS DE CASOS DE USO A AUTOMATIZAR

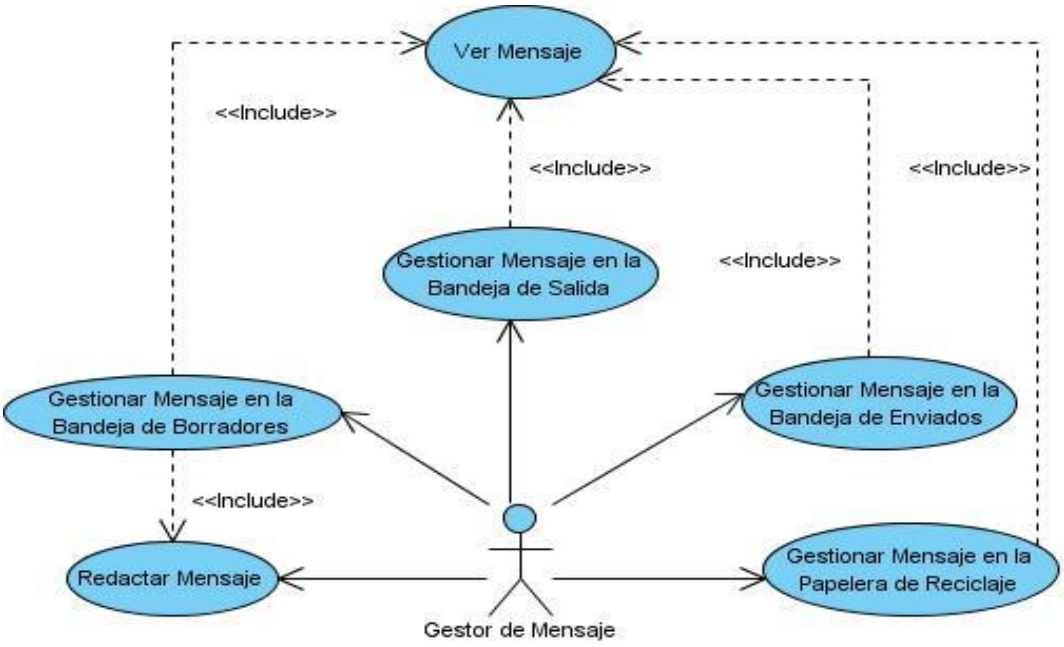


Figura 4: Diagrama de Casos de Uso. Subsistema Mensajes.

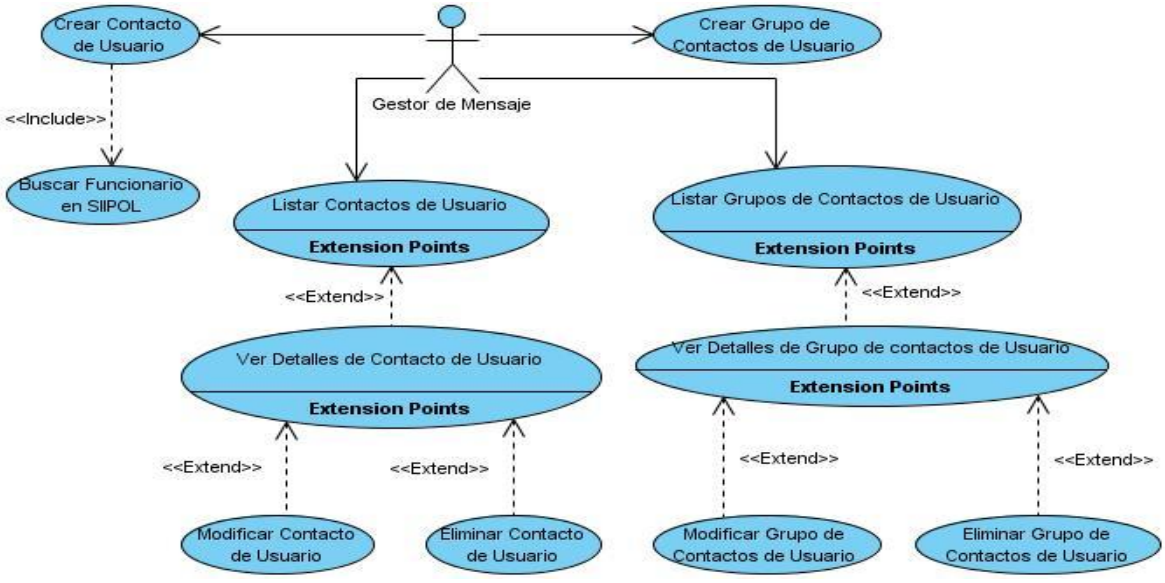


Figura 5: Diagrama de Casos de Uso. Subsistema Contactos.

2.5.4 DESCRIPCIÓN DE LOS CASOS DE USO MÁS SIGNIFICATIVOS

Caso de Uso	Redactar Mensaje
Objetivo	Redactar un Mensaje hacia uno o más contactos con fines operativos.
Actor	Gestor de Mensaje
Resumen	El caso de uso inicia cuando el Gestor de Mensaje accede a la opción de enviar un Mensaje a uno o más Contactos, pudiendo para ello, crear un Mensaje completamente nuevo o modificar un Mensaje ya existente. El sistema coloca el Mensaje en la cola de salida según la prioridad seleccionada. Una vez que el Mensaje es enviado, se registra una traza y se mueve de la Bandeja de Salida a la Bandeja de Enviados. El sistema permite además guardar el Mensaje en la Bandeja de Borradores o cancelar el envío del Mensaje. El caso de uso termina.
Complejidad	Media
Nivel	Usuario
Precondiciones	Debe haberse autenticado en la aplicación Web y el Usuario debe tener privilegios para gestionar Mensajes.
Poscondiciones	Se redactó satisfactoriamente el Mensaje.
Referencias	Requisitos no funcionales
Requisitos no funcionales	RNF 1, RNF 2, RNF 3, RNF 4, RNF 5, RNF 6, RNF 7, RNF 8, RNF 9, RNF 10, RNF 11, RNF 12, RNF 13, RNF 15, RNF 16, RNF 17,RNF 18, RNF 19, RNF 20, RNF 21, RNF 22, RNF 23

Tabla 3: Descripción CU/Redactar Mensaje.

Caso de Uso	Gestionar Mensaje en la Bandeja de Borradores.
Objetivo	Listar todos los Mensajes que se hayan guardado antes de enviar.
Actor	Gestor de Mensaje
Resumen	El caso de uso inicia cuando el Gestor de Mensaje accede a la Bandeja de Borradores. El sistema muestra un listado con todos los Mensajes guardados por el usuario autenticado. El sistema brinda la posibilidad de seleccionar uno o más Mensajes para eliminarlo(s), o eliminarlos todos. El sistema brinda además la opción de seleccionar un Mensaje para continuar su redacción. Termina el caso de uso.
Complejidad	Media
Nivel	Usuario
Precondiciones	Debe haberse registrado en la aplicación y tener privilegios para gestionar Mensajes.
Poscondiciones	Se listó todos los mensajes guardados por el usuario registrado.
Referencias	Requisitos no funcionales CU Redactar Mensaje
Requisitos no funcionales	RNF 1, RNF 2, RNF 3, RNF 4, RNF 5, RNF 6, RNF 7, RNF 8, RNF 9, RNF 10, RNF 11, RNF 12, RNF 13, RNF 15, RNF 16, RNF 17,RNF 18, RNF 19, RNF 20, RNF 21, RNF 22, RNF 23

Tabla 4: Descripción CU/Gestionar Mensaje en la Bandeja de Borradores.

Caso de Uso	Crear Contacto de Usuario.

Objetivo	Crear un Contacto de Usuario para poder mandarle mensajes desde la plataforma de mensajería.
Actor	Gestor de Mensaje
Resumen	El caso de uso inicia cuando el Gestor de Mensaje accede a la opción de registrar un nuevo Contacto de usuario. El sistema brinda la opción de registrar los datos del Contacto, permitiendo buscar esta información dentro del conjunto de funcionarios de SIIPOL. El sistema brinda además la opción de cancelar la creación del Contacto. Termina el caso de uso.
Complejidad	Media
Nivel	Usuario
Precondiciones	Debe haberse autenticado en la aplicación Web y el Usuario debe tener privilegios para gestionar Mensajes.
Poscondiciones	Se creó satisfactoriamente un Contacto de Usuario.
Referencias	Requisitos no funcionales CU Buscar Funcionario en SIIPOL
Requisitos no funcionales	RNF 1, RNF 2, RNF 3, RNF 4, RNF 5, RNF 6, RNF 7, RNF 8, RNF 9, RNF 10, RNF 11, RNF 12, RNF 13, RNF 15, RNF 16, RNF 17,RNF 18, RNF 19, RNF 20, RNF 21, RNF 22, RNF 23

Tabla 5: Descripción CU/Crear Contacto de Usuario.

Caso de Uso	Crear Grupo de Contactos de Usuario
Objetivo	Crear un Grupo de Contactos de Usuario para poder mandarle

	mensajes a todos los Contactos de ese Grupo.
Actor	Gestor de Mensaje
Resumen	El caso de uso inicia cuando el Gestor de Mensaje accede a la opción de registrar un nuevo Grupo de Contactos de Usuario. El sistema brinda la opción de introducir los datos del grupo. El sistema brinda además la opción de cancelar la creación del Grupo de Contactos de Usuario. Termina el caso de uso.
Complejidad	Media
Nivel	Usuario
Precondiciones	Debe haberse autenticado en la aplicación Web y el Usuario debe tener privilegios para gestionar Mensajes.
Poscondiciones	Se creó satisfactoriamente un Grupo de Contactos de Usuario.
Referencias	Requisitos no funcionales
Requisitos no funcionales	RNF 1, RNF 2, RNF 3, RNF 4, RNF 5, RNF 6, RNF 7, RNF 8, RNF 9, RNF 10, RNF 11, RNF 12, RNF 13, RNF 15, RNF 16, RNF 17,RNF 18, RNF 19, RNF 20, RNF 21, RNF 22, RNF 23

Tabla 6: Descripción CU/Crear Grupo de Contactos de Usuario.

Caso de Uso	Buscar Funcionario en SIIPOL.
Objetivo	Consultar los datos de un funcionario en SIIPOL para asociarlo a alguna funcionalidad de SIIPOL Móvil.
Actor	Caso de Uso Base.

Resumen	El caso de uso inicia cuando el usuario accede a la opción de buscar un Funcionario de SIIPOL para usarlo en una funcionalidad del SIIPOL Móvil. Termina el caso de uso.
Complejidad	Media
Nivel	Subfunción
Precondiciones	Debe haberse autenticado en la aplicación Web y el Usuario debe estar haciendo en el sistema alguna acción que le haga falta los datos de un funcionario de SIIPOL.
Poscondiciones	Se mostraron las coincidencias de los funcionarios encontrados y se seleccionó uno.
Referencias	Requisitos no funcionales
Requisitos no funcionales	RNF 1, RNF 2, RNF 3, RNF 4, RNF 5, RNF 6, RNF 7, RNF 8, RNF 9, RNF 10, RNF 11, RNF 12, RNF 13, RNF 15, RNF 16, RNF 17,RNF 18, RNF 19, RNF 20, RNF 21, RNF 22, RNF 23

Tabla 7: Descripción CU/Buscar Funcionario en SIIPOL.

Las descripciones en formato expandido de los casos de uso restantes se pueden consultar en el anexo 7.

CONCLUSIONES

En el transcurso de este capítulo se definió el objeto de automatización y la propuesta de sistema para darle solución a la problemática planteada, además se registraron las funcionalidades que deben ser desarrolladas en el proceso de captura de requisitos, realizándose una descripción de cada uno de estos. Se identificaron y describieron los actores y casos de uso estableciéndose las relaciones correspondientes entre cada uno de ellos en el diagrama de casos de uso.

CAPÍTULO 3. ANÁLISIS Y DISEÑO

INTRODUCCIÓN

En el desarrollo de este capítulo se realiza la descripción de los procesos que se llevan a cabo en el flujo de trabajo análisis y diseño, obteniendo como resultado los artefactos más importantes para modelar el sistema, teniendo en cuenta los patrones de diseño que aportan soluciones concretas a problemas específicos para lograr un diseño eficaz en el software. También se muestran los modelos lógicos de datos y el modelo físico de datos con su respectivo diagrama de clases persistentes y el diagrama entidad relación.

3.1. MODELO DE ANÁLISIS

Durante el Análisis, se estudian los requerimientos descritos en la captura de requerimientos. Aquí se refinan y estructuran con el objetivo de conseguir una comprensión más precisa de los requerimientos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero. A partir de este flujo se comienza a analizar el sistema con el lenguaje del desarrollador. El lenguaje que se utilizará en el análisis se basa en un modelo de objetos conceptual, llamado Modelo de Análisis. (6)

3.1.1 DIAGRAMAS DE CLASES DE ANÁLISIS DE LOS CASOS DE USO MÁS SIGNIFICATIVOS

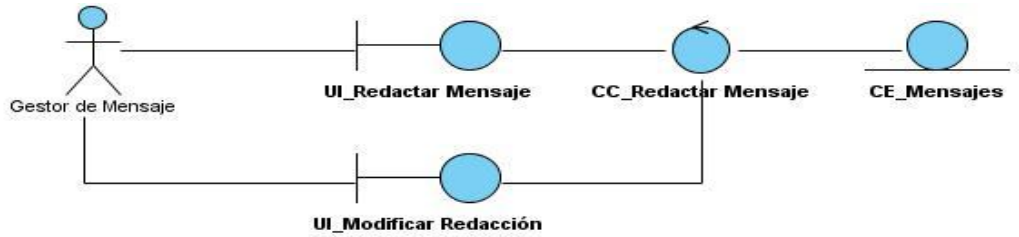


Figura 6: Diagrama de Clases de Análisis/CU Redactar Mensaje.

Los Diagramas de Clases de Análisis de los casos de uso más significativos restantes se pueden consultar en el anexo 8.

3.1.2 DIAGRAMAS DE COLABORACIÓN

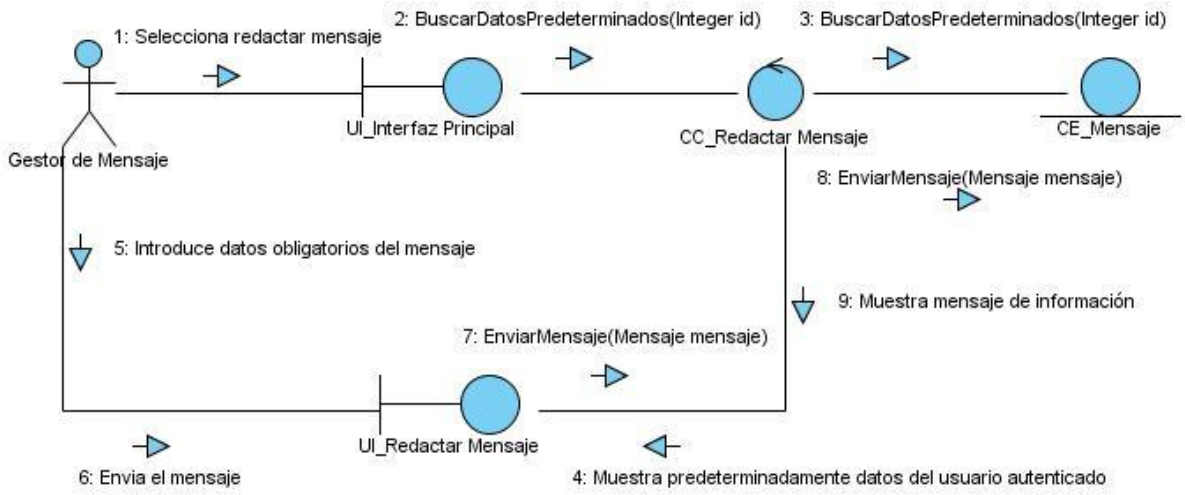


Figura 7: Diagrama de Colaboración CU/Redactar Mensaje.

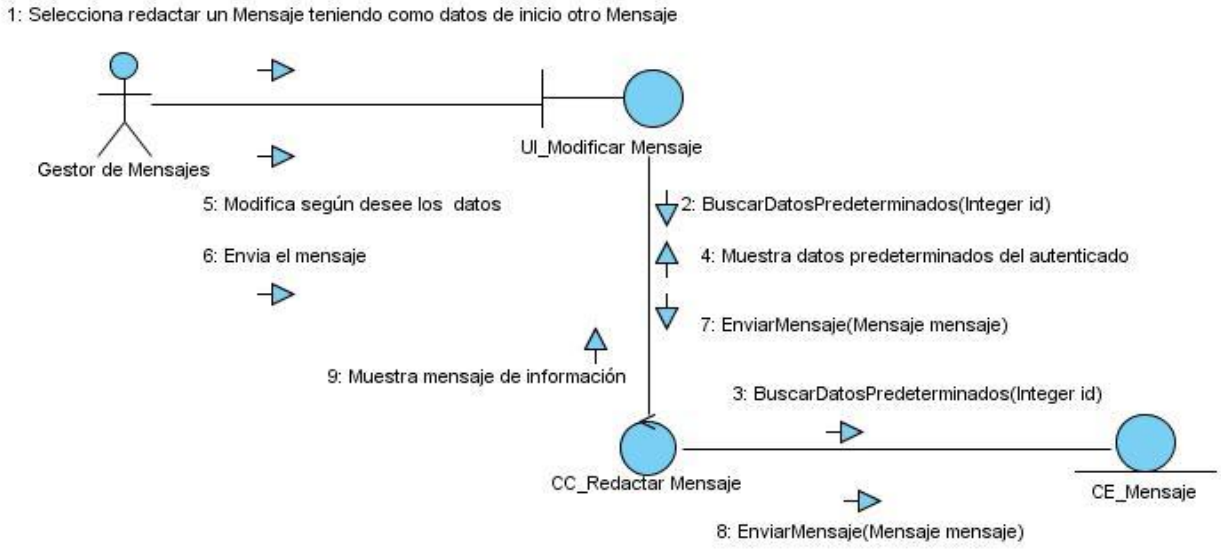


Figura 8: Diagrama de Colaboración CU/Modificar Redacción del CU/Redactar Mensaje.

Los Diagramas de Colaboración de los casos de uso más significativos restantes se pueden consultar en el anexo 9.

3.1.3 MODELO DE DISEÑO

En el Diseño se modela el sistema y se encuentra su forma (incluida la arquitectura) para que soporte todos los requerimientos (funcionales y no funcionales). En este flujo se adquiere una comprensión en profundidad de los aspectos relacionados con los requerimientos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables y sistemas operativos. Se crea una entrada apropiada y un punto de partida para actividades de implementación. Se descomponen los trabajos de implementación en partes más manejables que puedan ser manejadas por diferentes equipos de desarrollo. (6)

3.1.4 DIAGRAMAS DE CLASES DE DISEÑO

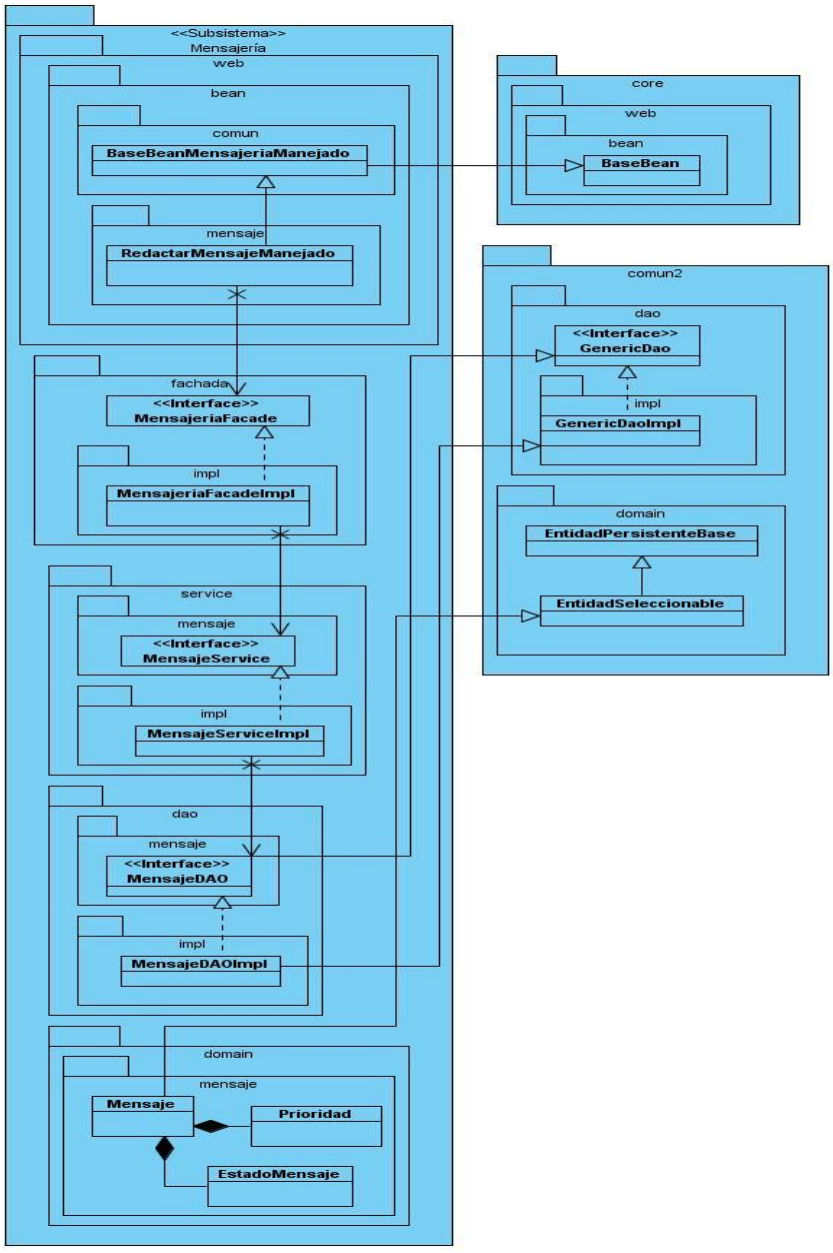


Figura 9: Diagrama de Clases de Diseño CU/Redactar Mensaje.

Los Diagramas de Clases de Diseño de los casos de uso más significativos restantes se pueden consultar en el anexo 10.

3.2 PATRONES DE DISEÑO

Con el fin de facilitar reusabilidad, extensibilidad y mantenimiento, se emplearon patrones de diseño en la solución propuesta. A continuación se exponen los patrones de diseño que se utilizaron y sus principales características:

- **Controlador Frontal**

Proporciona un punto de entrada único que controla y gestiona las peticiones web realizadas por los clientes, incluyendo la invocación de servicios de seguridad como la autenticación y la autorización, delegando el procesamiento de negocios a los objetos responsabilizados. En la solución propuesta el mismo está implementado por el framework JSF, en este caso el controlador es la clase de respaldo de la página que recibe la petición, la procesa invocando al modelo, y actualiza la página o redirecciona hacia otra vista; se utiliza la variante de utilizar un controlador por cada página.

- **Fachada**

Utilizado para proporcionar interfaces simples para subsistemas complejos. Cuando existen muchas dependencias entre clientes y clases que implementan una abstracción, este patrón proporciona independencia y portabilidad. Por otra parte, teniendo en cuenta que el diseño está separado por capas, cada capa posee su propia fachada o interfaz unificada de alto nivel que facilite su uso, lo cual trae como consecuencia que al separar al cliente de los componentes del subsistema, se reduzca el número de objetos con los que el cliente trata, facilitando así el uso del subsistema.

El patrón Fachada promueve además un débil acoplamiento entre el subsistema y sus clientes, eliminándose o reduciéndose las dependencias.

- **Objetos de Acceso a Datos (DAO)**

Consiste en utilizar un objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos; de manera que se pueda acceder a los datos que maneja la aplicación sin necesidad de conocer la fuente que suministra esa información.

- **Bajo Acoplamiento**

Este patrón le da respuesta a la problemática de soportar bajas dependencias, disminuyendo el impacto del cambio e incrementando la reutilización. El acoplamiento es una medida de la fuerza con que un elemento está conectado, tiene conocimiento y confía en otros elementos. Un elemento con bajo (o débil) acoplamiento no depende de muchos elementos. Estos pueden ser clases, subsistemas, entre otros, de ahí la importancia que se lleve a cabo el desempeño de este patrón, obteniendo de esta manera una aplicación lo más flexible posible.

- **Experto**

Consiste en asignar una responsabilidad al experto en información, o sea, a la clase que tiene la información necesaria para realizar la responsabilidad. Un modelo de diseño podría definir miles de clases y una aplicación podría requerir que se realicen gran cantidad de responsabilidades. Si se aplican de la forma correcta, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y existen más oportunidades para reutilizar componentes en futuras aplicaciones.

Con el uso de este patrón se mantiene el encapsulamiento de la información, puesto que los objetos utilizan su propia información para llevar a cabo las tareas. Normalmente, esto conlleva un bajo acoplamiento, que como se ha mencionado, da lugar a sistemas más robustos y más fáciles de mantener. Además, se distribuye el comportamiento entre las clases que contienen la información requerida, por tanto, se estimula la definición de clases más cohesivas y ligeras.

3.3. MODELO DE DATOS

Un modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general, un modelo de datos permite describir las estructuras de datos de la base (el tipo de los datos que incluye la base y la forma en que se relacionan), las restricciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos de la base). Los modelos de datos lógicos se centran en las operaciones y se implementan en algún manejador de base de datos. Por último,

podemos mencionar a los modelos de datos físicos, que son estructuras de datos a bajo nivel implementadas dentro del propio manejador. (7)

3.3.1 DIAGRAMA DE CLASES PERSISTENTES

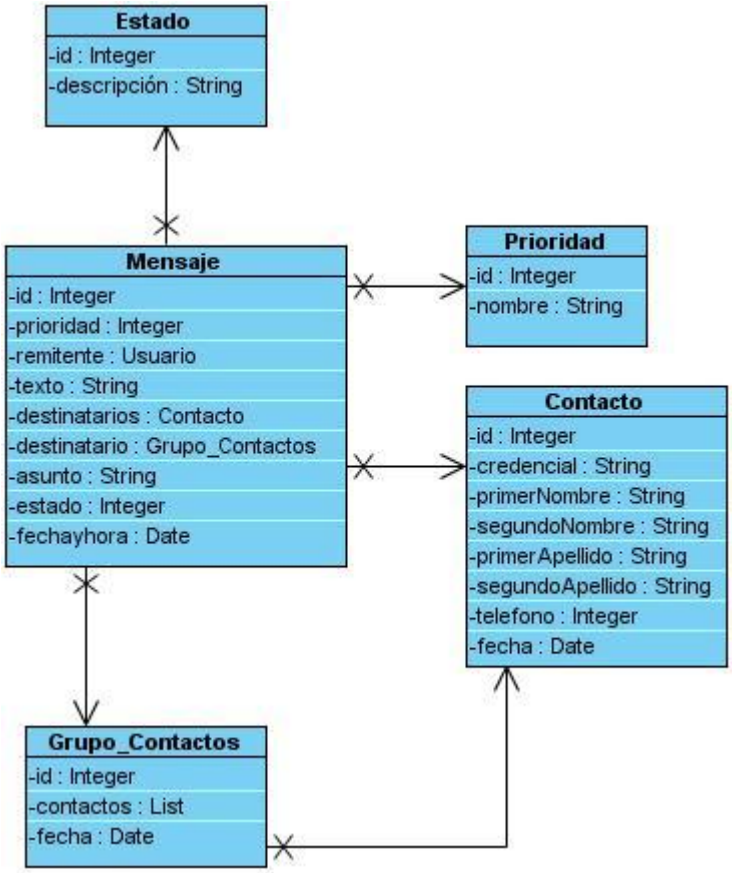


Figura 10: Diagrama de clases persistentes.

3.3.2 DIAGRAMA ENTIDAD RELACIÓN

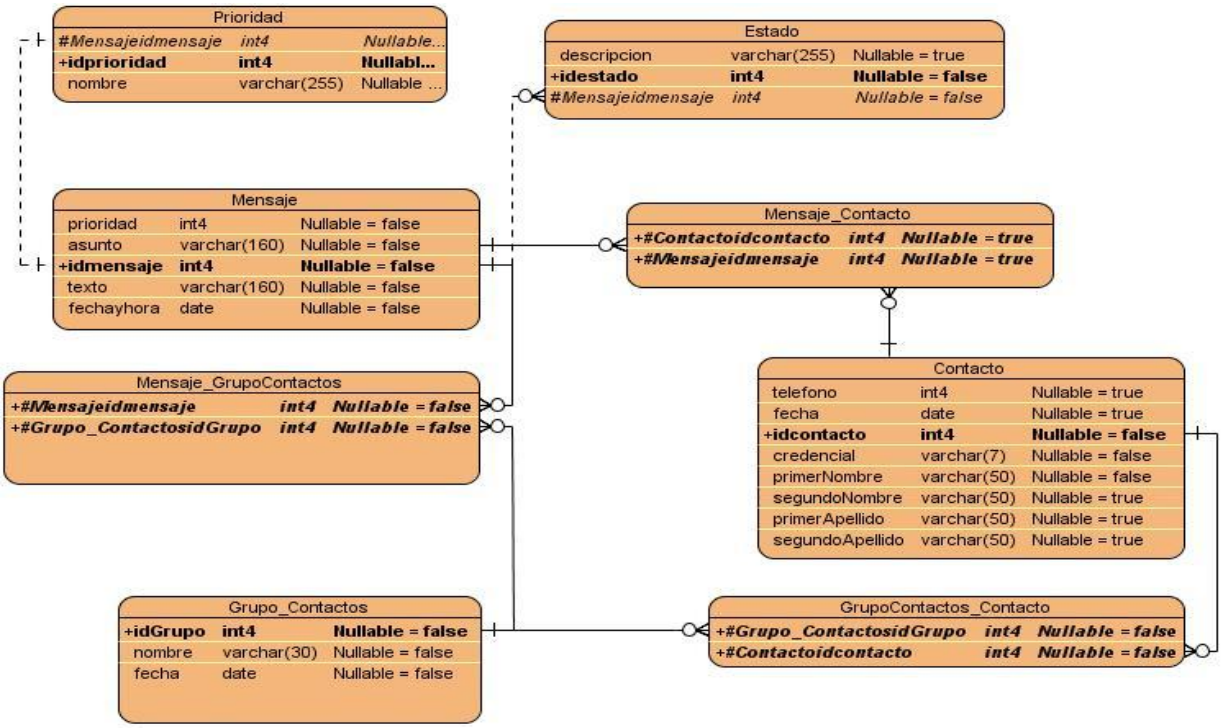


Figura 11: Diagrama entidad-relación.

3.4. INTERFAZ DE USUARIO

Figura 12: Interfaz Inicial.

Las interfaces de usuario restantes se pueden consultar en el anexo 11.

CONCLUSIONES

En este capítulo se obtuvieron los principales artefactos del análisis y diseño que sirven como entregables del sistema propuesto como solución a la problemática planteada. Se han descrito los patrones de diseño utilizados y los prototipos de interfaz de usuario y se tuvo en cuenta los modelos físicos y lógicos en el manejo de los datos del sistema a desarrollar.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

INTRODUCCIÓN

En el presente capítulo se describe el modelo de implementación utilizado y se muestran los diagramas de componentes y de despliegue. Además se realiza un estudio de los diferentes métodos y tipos de pruebas para llevar a cabo la ejecución de las mismas, y garantizar la calidad del sistema y el cumplimiento de los requerimientos establecidos con el cliente.

4.1. MODELO DE IMPLEMENTACIÓN

Consiste en una visión general de lo que tiene que ser implementado y un apartado para cada iteración con los componentes y subsistemas a implementar durante esa iteración. En el modelo de implementación se generan una serie de artefactos que constituyen la composición física de la implementación del sistema como son los diagramas de subsistemas y componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares, donde se detalla esencialmente la relación que existe desde las clases y componentes del modelo de diseño a subsistemas y componentes físicos. En este modelo se ajustan los subsistemas formados por los elementos de implementación y se definen las dependencias entre subsistemas.

4.1.1 DIAGRAMA DE SUBSISTEMAS DE IMPLEMENTACIÓN

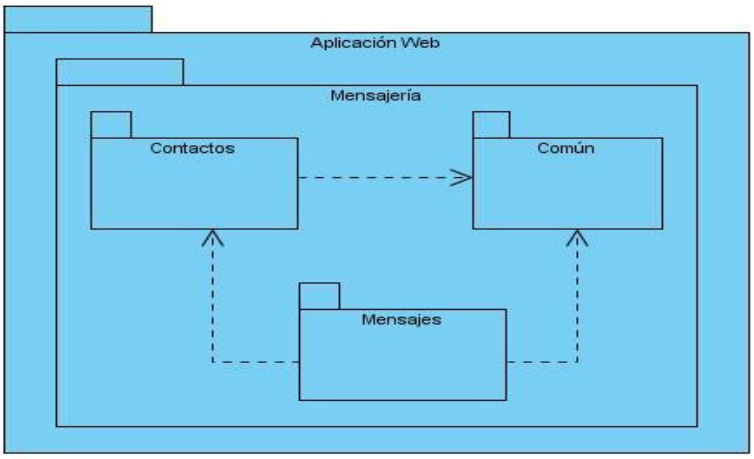


Figura 13: Diagrama de susbsistemas de implementación.

4.1.2 DIAGRAMA DE COMPONENTES

Un diagrama de componentes ilustra los fragmentos de software y los controladores embebidos que conforman un sistema, permiten editar de forma segura todo el contenido que llevarán los archivos de datos de la aplicación. Tienen un nivel de abstracción más elevado que un diagrama de clases. Usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos diagramas se utilizan para describir la vista de implementación estática de un sistema.

A continuación se representa el diagrama de componentes referente al subsistema Contactos.

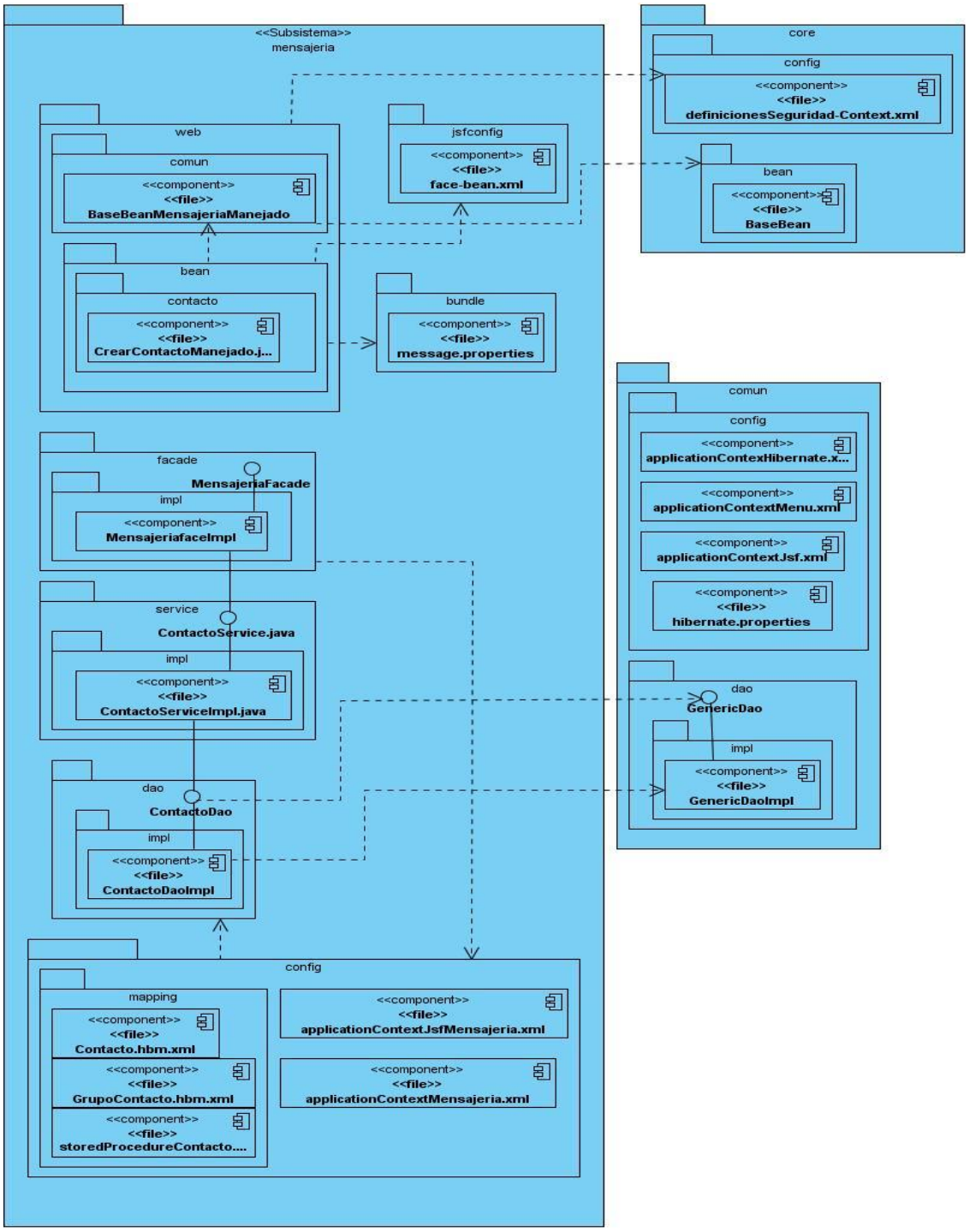


Figura 14: Diagrama de componentes.

4.2. DIAGRAMA DE DESPLIEGUE

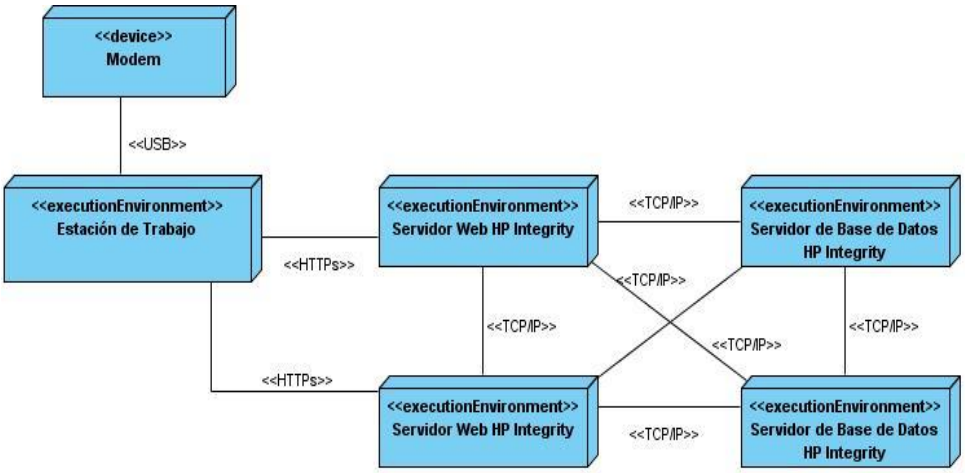


Figura 15: Diagrama de despliegue.

4.3. VALIDACIÓN DE LA PROPUESTA SOLUCIÓN

La prueba es un elemento crítico para la garantía de la calidad del software, es el proceso que permite verificar y revelar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa. Básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas. La etapa de pruebas implica:

- Verificar la interacción de componentes.
- Comprobar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.

Las pruebas se enfocan sobre la lógica interna del software y las funciones externas, para lo cual se definen los siguientes métodos:

- **Prueba de Caja Blanca**

Las pruebas de caja blanca realizan un seguimiento del código fuente según se van ejecutando los casos de prueba, de manera que se determinan de manera concreta las instrucciones en las que existen errores. Requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas del código.

- **Prueba de Caja Negra**

Pruebas que se llevan a cabo sobre la interfaz de usuario, se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Examina aspectos del modelo, principalmente del sistema, sin tener en cuenta la estructura interna del software.

Niveles de pruebas

A la hora de evaluar dinámicamente un sistema de software, la estrategia seleccionada debe permitir comenzar por los componentes más simples y pequeños e ir avanzando progresivamente hasta probar todo el sistema en su conjunto.

Para lograr una mayor efectividad, las pruebas de software se realizan a diferentes niveles:

Pruebas de Unidad

La mayoría de las arquitecturas de software modernas incorporan la modularidad como uno de sus elementos esenciales, lo cual posibilita el trabajo de varios programadores en distintos módulos sin que esto signifique un problema. De la misma forma es posible que los programadores ejecuten pruebas locales sobre sus módulos, a fin de comprobar el buen funcionamiento (autónomo) de lo que han realizado hasta el momento. A este tipo de prueba se le conoce como pruebas de unidad, las cuales centran su proceso de verificación en la menor unidad de software, el módulo.

El objetivo de estas pruebas es aislar cada parte del programa y mostrar que las partes individuales son correctas. Estas pruebas permiten llegar a la siguiente fase con un alto grado de seguridad de que el código está funcionando correctamente, facilitando de esta forma las pruebas de integración.

Pruebas de Integración

Son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a las pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez. Consisten en realizar pruebas para verificar que un gran conjunto de partes de software funcionan juntos.

Conjuntamente con el proceso de integración se aplican las pruebas de regresión, para asegurar que los módulos causantes de fallas fueron efectivamente corregidos y que no se introdujeron nuevos errores al tratar de solucionar alguno. Estas pruebas de regresión consisten en aplicar el mismo proceso de pruebas planificado originalmente, repitiéndose cada vez que se identifica un resultado erróneo y se le de solución.

Resultados de las pruebas

Para evaluar la solución desarrollada se planificaron dos iteraciones evaluando la aplicación con gran detalle. Se abarcaron los métodos y niveles de prueba expuestos anteriormente en cada una de las iteraciones, arrojando resultados visibles.

La automatización de las pruebas de unidad fue un aspecto fundamental para agilizar el proceso, para lo cual fue utilizado el framework JUnit. Estas pruebas no se planificaron ni se registraron sus resultados, se fueron haciendo a medida que la solución se desarrollaba.

Durante las pruebas se documentaron todas las No Conformidades surgidas, dándole la categoría de Alta, Media o Baja según su impacto en el negocio y el sistema. A continuación se muestra el resultado de las pruebas en cada una de las iteraciones.

Pruebas Internas

Pruebas realizadas al sistema por el equipo de calidad interna del proyecto. Se realizaron con el fin de entregar un producto lo más libre de errores posible al tercero encargado de validar la factibilidad del sistema. Se centraron en el cumplimiento de las funcionalidades descritas en el listado de requisitos y de casos de uso.

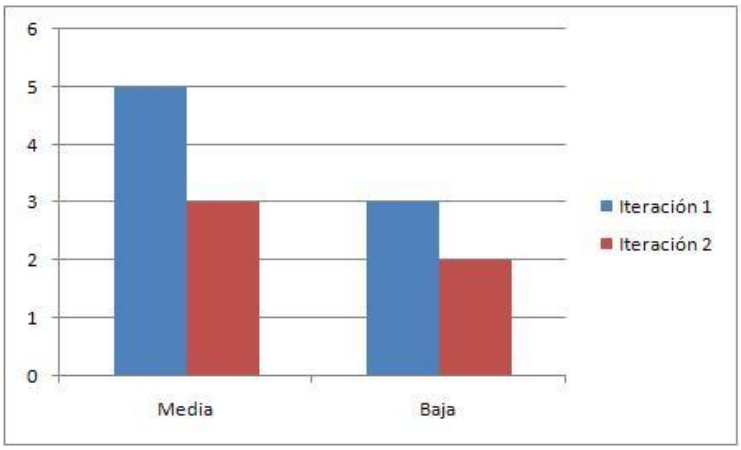


Figura 16: No Conformidades encontradas durante las pruebas internas.

Pruebas Cruzadas

Realizadas al sistema por el equipo de desarrollo del proyecto. Se realizaron con el fin de encontrar la mayor cantidad de errores posibles en término de validaciones, pautas definidas por arquitectura de información, formato de los campos, entre otras.

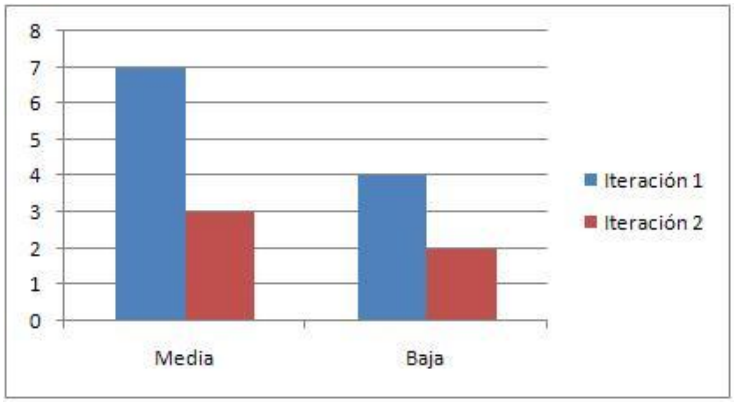


Figura 17: No Conformidades encontradas durante las pruebas cruzadas.

Luego de concluida cada iteración de prueba se analizaron, por parte del equipo de desarrollo, las no conformidades encontradas para determinar cuáles realmente constituyeron defectos del sistema o presentaban alguna variación según la descripción de los casos de uso y necesitaban ser solucionadas. Las pruebas se realizaron de forma iterativa e incremental, comprobando en cada iteración que hubiesen sido corregidos los errores detectados en la iteración anterior, lo que contribuyó a mejorar la calidad y eficiencia del software.

A continuación se muestra el gráfico con un resumen del total de las no conformidades encontradas en cada iteración de pruebas, evidenciándose la correcta evolución del software en cuanto a la reducción del número de defectos detectados.

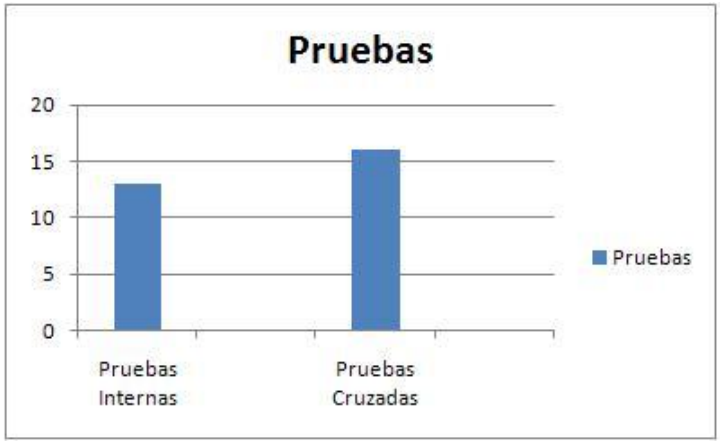


Figura 18: Resumen general de las pruebas.

CONCLUSIONES

En este capítulo se realizó el modelo de implementación, donde se generaron artefactos para el cumplimiento de los requisitos funcionales y no funcionales descritos por el sistema, los principales son el diagrama de subsistemas de implementación, los diagramas de componentes y el diagrama de despliegue. Mediante el estudio de los métodos y niveles de pruebas aplicados para la determinación de errores en el sistema, se realizó un análisis de los resultados obtenidos, garantizando de esta forma la solidez de la propuesta.

CAPITULO 5. ESTUDIO DE FACTIBILIDAD

INTRODUCCIÓN

En este capítulo se referencia la planificación del proyecto con el propósito de obtener una aproximación de los recursos necesarios así como del esfuerzo, costo y tiempo necesario. Mediante este estudio se podrá obtener la cantidad de personas que intervendrán y sobre todo, cuánto debe ser invertido por parte de las empresas interesadas en el producto.

5.1. ESTIMACIÓN POR PUNTOS DE CASOS DE USO

La estimación mediante el análisis de Puntos de Casos de Uso es un método propuesto originalmente por Gustav Karner de Objectory AB, y posteriormente refinado por muchos otros autores. Se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores. (8)

A continuación, se detallan los pasos a seguir para realizar la estimación:

5.1.1 CÁLCULO DE PUNTOS DE CASOS DE USO SIN AJUSTAR

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

- **Calculando el Factor de Peso de los Actores sin ajustar** (**UAW**)

Tipo	Descripción	Peso	Cant * peso

Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API)	1	0*1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2	0*2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3	1*3
		Total	**3**

Tabla 8: Factor de peso de los actores sin ajustar.

- **Calculando el Factor de Peso de los Casos de Uso sin ajustar** (**UUCW**)

Tipo	**Descripción**	**Peso**	**Cant * peso**
Simple	El Caso de Uso contiene de 1 a 3 transacciones	5	16*5
Medio	El Caso de Uso contiene de 4 a 7 transacciones	10	0*10
Complejo	El Caso de Uso contiene más de 8 transacciones	15	1*15
		Total	**95**

Tabla 9: Factor de peso de los casos de uso sin ajustar.

UUCP = 3 + 95 = 98

5.1.2 CÁLCULO DE PUNTOS DE CASOS DE USO AJUSTADOS

UCP = UUCP x TCF x 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

- **Calculando el Factor de Complejidad Técnica** (**TCF**)

Factor	Descripción	Peso	Valor	Σ **($Peso_i$ * $Valor_i$)**
T1	Sistema distribuido	2	0	0
T2	Objetivos de performance o tiempo de respuesta	1	4	4
T3	Eficiencia del usuario final	1	2	2
T4	Procesamiento interno complejo	1	2	2
T5	El código debe ser reutilizable	1	4	4
T6	Facilidad de instalación	0.5	3	1.5
T7	Facilidad de uso	0.5	4	2.0
T8	Portabilidad	2	3	6
T9	Facilidad de cambio	1	3	3
T10	Concurrencia	1	2	2

T11	Incluye objetivos especiales de seguridad	1	3	3
T12	Provee acceso directo a terceras partes	1	4	4
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1	1	1
	Total			**34.5**

Tabla 10: Factor de complejidad técnica.

Se utiliza la siguiente ecuación:

TCF= 0.6 + 0.01 x Σ (Pesoi x Valor asignadoi)

TCF = 0.6 + 0.01 x 34.5 = 0.95

- **Calculando el Factor de Ambiente** (**EF**)

Factor	Descripción	Peso	Valor	Σ **($Peso_i$ * $Valor_i$)**
E1	Familiaridad con el modelo de proyecto utilizado	1.5	3	4.5
E2	Experiencia en la aplicación	0.5	3	1.5
E3	Experiencia en orientación a objetos	1	4	4
E4	Capacidad del analista líder	0.5	4	2.0
E5	Motivación	1	4	4
E6	Estabilidad de los requerimientos	2	4	8
E7	Personal a tiempo parcial	-1	4	-4

E8	Dificultad del lenguaje de programación	-1	3	-3
	Total			**17**

Tabla 11: Factor de ambiente.

Se utiliza la siguiente ecuación:

EF = 1.4 - 0.03 x Σ (Pesoi x Valor asignadoi)

EF = 1.4 - 0.03 x 17 = 0.89

Entonces:

UCP = 98 x 0.95 x 0.89 = 82.9

5.2. CÁLCULO DEL ESFUERZO

E = UCP x CF

Donde:

E: Esfuerzo estimado en horas-hombre

UCP: Puntos de Casos de Uso ajustados

CF: Factor de conversión

Para calcular CF:

CF = 20 horas-hombre (si $\text{Total}_{EF} \leq 2$)

CF = 28 horas-hombre (si $\text{Total}_{EF} = 3$ ó $\text{Total}_{EF} = 4$)

CF = abandonar o cambiar proyecto (si $\text{Total}_{EF} \geq 5$)

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

Como **Total** $_{EF}$ = 0 + 1

Total $_{EF}$ = 1

CF = 20 horas-hombre (porque Total $_{EF}$ ≤ 2)

Luego

E = 82.9 * 20 horas-hombre

E = 1658 horas-hombre

5.3. DISTRIBUCIÓN DEL ESFUERZO ENTRE LAS DIFERENTES ACTIVIDADES DEL PROYECTO

Actividad	Porcentaje	Valor Esfuerzo (horas-hombre)
Análisis	10.00 %	165.8
Diseño	20.00 %	321.6
Programación	40.00 %	663.2
Pruebas	15.00 %	248.7
Sobrecarga (otras actividades)	15.00 %	248.7
Total	100.00%	1658

Tabla 12: Distribución del esfuerzo entre las actividades.

Para conocer la cantidad de meses que se invirtió para desarrollar la aplicación se realizó el siguiente cálculo:

1658 horas-hombre / 3 hombres =552.2 horas

Estimando que la cantidad de días laborables en el mes suman 22 y en cada día se labora 8 horas, se obtiene que la cantidad de horas/mes es igual a 176.

Entonces:

Meses = horas / horas-Mes

Meses = 552.2 horas / 176 horas-Mes

Meses =3.14

Por tanto el tiempo que tardó producir el sistema fue aproximadamente de 3 meses.

5.4. BENEFICIOS TANGIBLES E INTANGIBLES

Es importante especificar que la ejecución de este proyecto aportará beneficios tanto para el funcionamiento interno de la institución, como para la población venezolana que es la principal beneficiaria de los procesos que se llevan a cabo en la institución. Entre los beneficiados internos se encuentran el Ministerio para el Poder Popular para Relaciones Interiores y Justicia, particularmente el CICPC, y los beneficiados externos están la sociedad venezolana en su conjunto y otros organismos de seguridad del estado de la República Bolivariana de Venezuela.

Beneficios tangibles que provee la Plataforma de Mensajería de Texto

Con la realización exitosa del proyecto se incrementan notablemente las áreas desde donde un funcionario policial puede acceder información de interés policial actualizada y oportuna, especialmente hasta lugares donde se dificulta el acceso a las redes de computadoras.

Beneficios intangibles que provee la Plataforma de Mensajería de Texto

- Mejora la calidad de las decisiones operativas basadas en mayor disponibilidad de información policial.
- Se incrementa la capacidad operativa en el trabajo preventivo de campo.
- Se presta un servicio más profesional a la población.

Con un trabajo policial más efectivo se contribuye a la prevención del delito, la disminución de sus índices y con ello mejora la calidad de la seguridad ciudadana.

5.5. ANÁLISIS DEL COSTO

Algunos frameworks y tecnologías utilizadas para el desarrollo del sistema son distribuidos bajo licencias de Código Abierto y otros se reutilizaron del SIIPOL. En el caso del SGBD Oracle, fue elegida por el cliente asumiendo este el pago de la licencia.

El presupuesto asignado para la realización del proyecto SIIPOL Móvil fue aproximadamente $460 000, no se obtuvo un por ciento estimado del costo de la Plataforma de Mensajería de Texto.

CONCLUSIONES

En este capítulo se realizó un análisis de los distintos indicadores y se concluyó que era factible la realización de la Plataforma de Mensajería de Texto. Además se obtuvieron los estimados de horas por hombre, y a partir de este se calculó la cantidad de meses que se requieren para desarrollar la solución. Finalmente se hizo un pequeño análisis del costo, y se definieron los beneficios tangibles e intangibles que traerá consigo el uso de dicha solución para los funcionarios del CICPC.

CONCLUSIONES

Con el desarrollo del presente trabajo de diploma se cumplió el objetivo principal; desarrollar una Plataforma de Mensajería de Texto centralizada que garantice el envío de mensajes a los funcionarios operativos. Para obtener dicho resultado, se cumplieron los siguientes aspectos:

- Se elaboró el diseño teórico de la investigación donde se alcanzaron los conocimientos suficientes para el desarrollo de la solución.
- Se estableció la lógica del negocio, definiendo de esta forma el proceso de envío de mensajes de texto a los funcionarios operativos; así como también los casos de uso y actores involucrados.
- Se realizó el análisis y diseño de la Plataforma de Mensajería de Texto, obteniendo como resultado los artefactos más importantes para modelar el sistema, teniendo en cuenta los patrones de diseño.
- Se realizó la implementación de la Plataforma de Mensajería de Texto a partir de una serie de artefactos que dio lugar al cumplimiento de los requisitos funcionales y no funcionales.
- Se realizó la validación de la propuesta de solución para la determinación de errores, garantizando de esta forma la solidez del sistema.

RECOMENDACIONES

Luego de haberse cumplido con los objetivos propuestos mediante la realización del trabajo de diploma, se recomienda:

- Adicionar a la Plataforma de Mensajería la funcionalidad de recibir mensajes con el objetivo que los funcionarios operativos puedan realizar consultas mediante SMS.
- Estudiar la posibilidad de realizar un sistema similar para los órganos del Ministerio del Interior (MININT) en Cuba.
- Poner este trabajo a disposición de la comunidad universitaria como referencia para proyectos similares teniendo en cuenta las reglas de confidencialidad establecidas.

BIBLIOGRAFÍA

REFERENCIADAS

1. **Walls, Craig y Breidenbach, Ryan.** *Spring in Action. s.l. : Manning.* 2005.

2. **Albiol, Francesc Rosés.** *Introducción a Hibernate.* 2003.

3. Elwebmaster. [En línea] 7 de Noviembre de 2007. [Citado el: 26 de Enero de 2010.] http://www.elwebmaster.com/referencia/api-interface-de-programacion-de-aplicaciones.

4. Java Platform Micro Edition (Java ME). [En línea] [Citado el: 6 de Febrero de 2010.] http://www.sicuma.uma.es/sicuma/independientes/argentina08/Gaona-Perez/inicio.html.

5.Osmosis Latina. [Enlínea] 7 de Septiembre de 2005. http://www.osmosislatina.com/lenguajes/uml/actividad.htm.

6. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley : s.n., 2000.

7. Definición.de. [En línea] 2008. [Citado el: 25 de Abril de 2010.] http://definicion.de/modelo-de-datos/.

8. Clase Teórica-Práctica 2: Técnicas de estimación. [En línea] [Citado el: 20 de Marzo de 2010.] http://eva.uci.cu/mod/resource/view.php?id=22433.

CONSULTADAS

Walls, Craig y Breidenbach, Ryan. 2005.*Spring in Action. s.l. : Manning.*

Albiol, Francesc Rosés. 2003. *Introducción a Hibernate.*

Elwebmaster. 2007 [Disponible en: http://www.elwebmaster.com/referencia/api-interface-de-programacion-de-aplicaciones.]

Java Platform Micro Edition (Java ME). [Disponible en: http://www.sicuma.uma.es/sicuma/independientes/argentina08/Gaona-Perez/inicio.html.]

Osmosis Latina. 2005. [Disponible en: http://www.osmosislatina.com/lenguajes/uml/actividad.htm.]

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software. Madrid : Addison Wesley : s.n.*

Definición.de.2008. [Disponible en: http://definicion.de/modelo-de-datos/.]

Clase Teórica-Práctica 2: Técnicas de estimación. 2010. [Disponible en: http://eva.uci.cu/mod/resource/view.php?id=22433.]

Gabaldón, Luis Gerardo. 2007. SCielo. [Disponible en: http://www.scielo.org.ve/scielo.php?pid=S1315-64112007000300006&script=sci_arttext.]

Vegas, Jesús. 2002. El Servidor Web. [Disponible en: http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node20.html.]

Apache Tomcat. Apache Tomcat. [Disponible en: http://tomcat.apache.org/.]

Oracle. Oracle. [Disponible en: http://www.oracle.com/index.html.]

Telcel. Mensajería Empresarial. [Disponible en: http://www.telcel.com/portal/empresas/mensajeria_corporativa/mensajeria_grupos.html.]

Subversion. Subversion. [Disponible en: http://subversion.tigris.org./.]

Wikipedia. 2010. [Disponible en: http://es.wikipedia.org/wiki/Tomcat.]

Progarmación en castellano. [Disponible en: http://ww.programacion.com/.]

RXTX. 2006. [Disponible en: http://users.frii.com/jarvi/rxtx/intro.html.]

GLOSARIO DE TÉRMINOS

Access: es un programa sistema de gestión de base de datos relacional creado y modificado por Microsoft para uso personal en pequeñas organizaciones.

Ajax: Asyncronous JavaScript And XML, traducido al español como JavaScript asíncrono y XML, es una técnica de desarrollo web para crear aplicaciones interactivas.

Archive_Log: provee protección contra fallas del medio (disco por ejemplo) y también contra fallas de instancia (corte de energía, falla de CPU, entre otros).

Bean: se usan para encapsular varios objetos en un único objeto, para hacer uso de un sólo objeto en lugar de varios más simples.

Clúster: conjuntos o conglomerados de computadoras construidos mediante la utilización de componentes de hardware comunes y que se comportan como si fuesen una única computadora.

CometProcessor: es un neologismo para describir un modelo de aplicación web en el que una petición HTTP mantenida abierta permite a un servidor web enviar datos a un navegador, sin que el navegador los solicite explícitamente.

HTML: es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

HTTP: define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

JSP: Java Server Pages, traducido al español como Páginas Servidoras de Java, es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

MMS: Multimedia Messaging System o Sistema de Mensajería Multimedia es un estándar de mensajería que le permite a los teléfonos móviles enviar y recibir contenidos multimedia.

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

MySQL: es un sistema de gestión de base de datos relacional, multihilo y multiusuario.

Objective-C: es un lenguaje de programación orientado a objetos creado como un superconjunto de C que implementa un modelo de objetos parecido al de Smalltalk.

Plugin: es un pequeño programa que proporciona alguna funcionalidad específica a otra aplicación mayor o más compleja.

RAID: del inglés Redundant Array of Independent Disks, traducido al español como Conjunto Redundante de Discos Independientes, hace referencia a un sistema de almacenamiento que usa múltiples discos duros entre los que distribuye o replica los datos.

Redolog: monitorea una instancia de una base de datos y recoge el número promedio de peticiones de espacio de registro redo por minuto desde el inicio del servidor y el número promedio de intentos de asignación del buffer por minuto desde el inicio del servidor.

Renderizado: es un proceso para generar una imagen desde un modelo.

Script: es un guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades. Es un fragmento de código que puede recibir argumentos y devolver un valor.

Servlet: son objetos que corren dentro del contexto de un contenedor de servlets (como por ejemplo, Tomcat) y extienden su funcionalidad.

Smalltalk: es un lenguaje de programación que permite realizar tareas de computación mediante la interacción con un entorno de objetos virtuales.

Sun Microsystems: es una empresa informática recientemente comprada por Oracle Corporation antes era parte de Silicon Valley, fabricante de semiconductores y software.