Universidad de las Ciencias Informáticas

Facultad 2

Título: Servicio Basado en Localización para la plataforma Comcel.

Trabajo de Diploma para optar por el

Título de Ingeniero en Ciencias Informáticas

Autor(es): René Torres Velázquez.

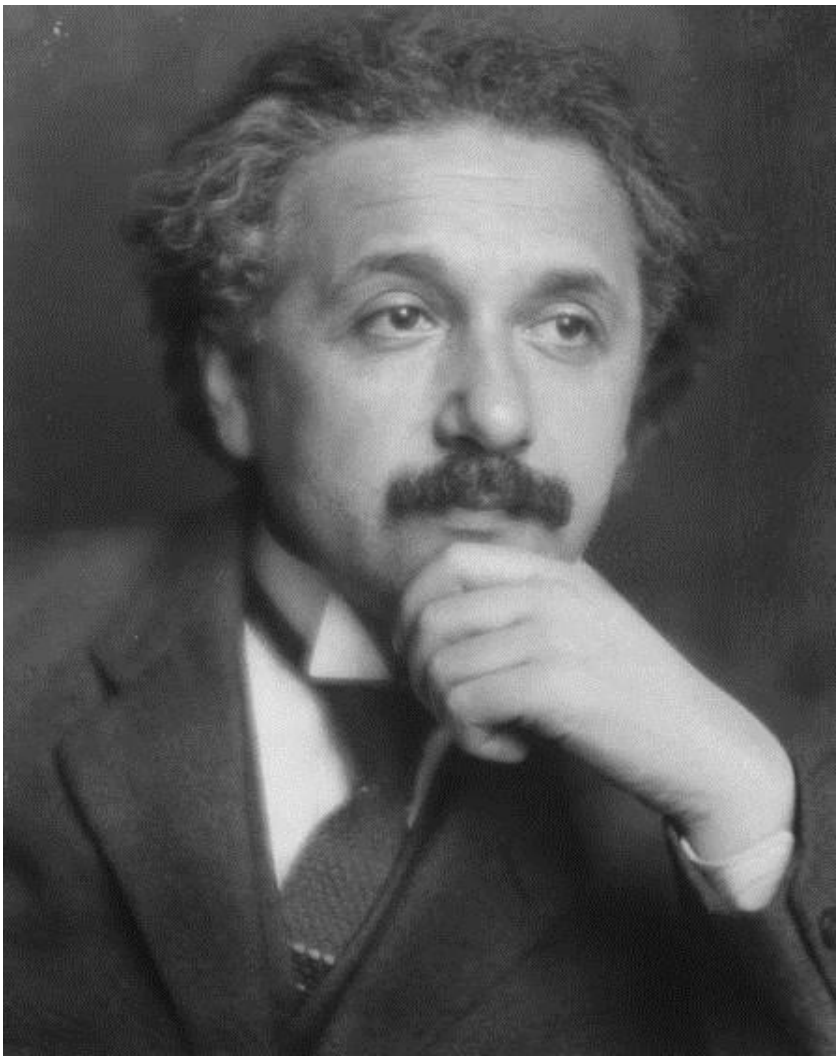
Luis Angel Santos Guevara.

Tutor: Ing. Darién Jesús Álvarez de la Cruz.

"Junio del 2011"

Pensamiento

Todos somos muy ignorantes. Lo que ocurre es que no todos ignoramos las mismas cosas.



Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Telemática de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

René Torres Velázquez

_____

Ing. Darién Jesús Álvarez de la Cruz

_____

Luis Angel Santos Guevara

_____

DATOS DE CONTACTO

Ing. en Ciencias Informáticas.

Graduado año 2007.

Categoría Docente Instructor.

Años de experiencia en el tema 2.

Años de graduado 3.

AGRADECIMIENTOS

DEDICATORIA

Esta tesis está dedicada a nuestros padres y hermanos.

RESUMEN

Los avances en las telecomunicaciones y la electrónica han propiciado el desarrollo de aplicaciones y servicios cada vez más versátiles y personalizados. Los servicios de Orientación y Navegación son aplicaciones LBS (Servicio Basado en Localización del inglés Location Based Service) que han despertado el interés de investigadores y clientes, por lo que han surgido nuevas necesidades de aplicaciones. Estos servicios ofrecen una ayuda personalizada que permite mantener orientado a los usuarios a partir de informaciones provistas por ellos. Esta orientación consiste en conocer, en sus dispositivos móviles, la ubicación de un determinado lugar, así como información relevante del mismo y la ruta entre dos lugares.

ETECSA (Empresa de Telecomunicaciones de Cuba S.A), encargada de la prestación de servicios de telefonía móvil, en convenio con la UCI (Universidad de Ciencias Informáticas), propone ofrecer servicios y contenidos para dispositivos móviles, para ello se desarrolla una Plataforma de Gestión de Contenidos para Dispositivos Móviles (Comcel).

El presente trabajo propone el desarrollo de un servicio de Orientación y Navegación para integrarlo a la plataforma Comcel con el fin de brindar servicios de orientación a los usuarios.

Palabras claves: GIS, LBS, WAP.

ÍNDICE

INTRODUCCIÓN

La telefonía móvil ha experimentado un progresivo desarrollo durante la última década, marcado por el notable crecimiento del número de usuarios y la evolución de la tecnología, lo cual ha llevado a un desarrollo de las diferentes plataformas de prestación de servicios móviles que han ido evolucionando del simple tráfico de voz, al envío de SMS (Sistema de Mensajería Corta del inglés Short Message System), MMS (Sistema de Mensajería Multimedia del inglés Multimedia Messaging System), acceso a internet, juegos y actualmente a los populares LBS. Esta tendencia enfrenta a los operadores y desarrolladores al reto de brindar contenidos y aplicaciones atractivos al usuario.

En Cuba muchos ciudadanos y principalmente turistas cuando se proponen dar un paseo, encontrar a un individuo o simplemente encontrar un lugar, deben haberse hecho la pregunta: ¿Dónde estoy?, ¿Dónde está la calle…?, ¿Dónde está el restaurante…?, ¿Cómo llego al hospital más cercano?, ¿Dónde está el museo…?, ¿Dónde está el hotel…?, sin encontrar una respuesta a sus interrogantes o simplemente derrochando tiempo para obtenerla, además de no cumplir con el propósito de llegar a los lugares planificados en tiempo o la ruta escogida no fue la más favorable.

Los LBS dan solución a las anteriores interrogantes ya que ofrecen una ayuda personalizada a los usuarios basándose en informaciones provistas por ellos que los mantienen orientados. A partir de estas informaciones permiten conocer dónde está situado determinado lugar en un mapa, obtener información relevante del mismo y qué sitios de interés se encuentran en ese entorno y qué ruta escoger para llegar a un determinado destino.

El proyecto Cubacel, perteneciente al centro de Telemática de la UCI, desarrolla una plataforma de gestión de contenidos para dispositivos móviles (Comcel) que permite la descarga online de contenidos de diferentes categorías y prestación de servicios de valor agregado. Esta plataforma no cuenta con un LBS que pueda beneficiar a los usuarios que la utilicen.

Con este objetivo ETECSA, encargada de la prestación de servicios de telefonía móvil, en convenio con la UCI, propone el desarrollo de un portal web para dispositivos móviles que brinde un servicio de Orientación y Navegación e integrarlo a la plataforma Comcel.

A partir de estas necesidades surge el ***problema a resolver:*** ¿Cómo brindar un Servicio Basado en Localización para dispositivos móviles?

Se define como ***objeto de estudio*** de la investigación:

- Servicios Basados en Localización.

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

- Servicios de Orientación y Navegación para la plataforma Comcel.

El ***objetivo general*** de la investigación es:

- Desarrollar un sistema que permita brindar un servicio de Orientación y Navegación e integrarlo a la plataforma Comcel.

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

- Estudiar y comprender el funcionamiento de los GIS (Sistema de Información Geográfica del inglés Geographic System Information).
- Estudiar y comprender el funcionamiento de los LBS.
- Estudiar y analizar las principales aplicaciones existentes en el mundo que utilizan los LBS.
- Estudiar y comprender el funcionamiento del servidor de transcodificación de media Alembik.
- Estudiar y analizar las características y el funcionamiento del framework Spring MVC (Modelo Vista Controlador del inglés Model View Controller).
- Estudiar y comprender el funcionamiento de la tecnología WAP (Protocolo de Aplicaciones Inalámbricas del inglés Wireless Application Protocol).
- Estudiar y analizar las características de WURFL (Archivo de Recursos Universales Móviles del inglés Wireless Universal Resource File).
- Investigar cómo integrar la solución a la plataforma de gestión de contenidos para dispositivos móviles Comcel.

Para llevar a cabo estas tareas de investigación se utilizaron los siguientes **métodos de investigación:**

Métodos teóricos:

- Analítico - Sintético: Permitió procesar la información para guiar la investigación de los LBS, simplificando y organizando el análisis de todos los datos.

- Análisis Histórico - lógico: Permitió realizar un estudio tanto de los antecedentes como de las tendencias actuales de los LBS.

Esta investigación está desglosada en 5 capítulos, las conclusiones generales, recomendaciones, referencias bibliográficas y bibliografía utilizada, un glosario de términos y por último los anexos. Este trabajo ha sido estructurado de la siguiente forma:

Capítulo 1 Fundamentación Teórica contiene un estudio de los principales conceptos tratados en el trabajo. Cuenta con una reseña de otros sistemas utilizados en el mundo para iguales propósitos y un estudio de las tecnologías y herramientas actuales, haciendo énfasis en las utilizadas en el desarrollo del Portal.

Capítulo 2 Características del Sistema contiene la descripción de los tipos de objetos más importantes que existen en el dominio. Se define como se interrelacionan estos objetos. Contiene además la captura de los requisitos funcionales y no funcionales, el diagrama de casos de usos del sistema, así como las descripciones de los casos de usos.

Capítulo 3 Diseño del Sistema se define el modelo de clases del diseño, los diagramas de interacción y los diagramas de paquete del diseño y la arquitectura utilizada. Este capítulo está enfocado hacia cómo construir técnicamente el software.

Capítulo 4 Implementación y Pruebas se define el diagrama de despliegue y componentes y se describen las pruebas realizadas al sistema.

Capítulo 5 Estudio de Factibilidad contiene un estudio sobre los beneficios que le puede aportar el sistema al país.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El presente capítulo contiene un estudio de los LBS y GIS, aborda los conceptos y características fundamentales de estos. Se define el lenguaje, las herramientas y tecnologías que se utilizarán en el desarrollo del sistema propuesto, así como la metodología a usar.

Las tecnologías de posicionamiento y las comunicaciones móviles son el marco adecuado para permitir el acercamiento de los LBS a las masas. Se han desarrollado muchos sistemas que brindan LBS destinados a beneficiar a las personas.

1.2 Servicios Basados en Localización

A principios de la década de los 90s se comenzó a dar impulso a los LBS por una determinación de la FCC (Comisión Federal de Comunicaciones del inglés Federal Communications Commission) en Estados Unidos para la implementación de E911 (Extenden 911) para redes móviles y en Asia NTT DoCoMo (Primer Operador Móvil Japonés) lanzó un servicio de localización de automotores que permitía hallar la ubicación del automóvil y cuanto tardaría en llegar a su destino.

Uno de los aspectos principales de los LBS es determinar la ubicación del dispositivo móvil dentro del área de cobertura. Los sistemas de localización se basan en una serie de tecnologías y métodos básicos:

- **COO (Celda de Origen):** En el método de localización por celda, también conocido como Método por Punto de Acceso para el caso de redes WLAN (Red Inalámbrica de Área Local del inglés Wireless Local Area Network) y WPAN (Red Inalámbrica de Área Personal del inglés Wireless Personal Area Network), la posición se obtiene directamente en función de la identidad de la celda o punto de acceso que da cobertura al área en el que se encuentra el terminal móvil [1].
- **ToA (Tiempo de Llegada):** El método Tiempo de Llegada utiliza la medida del tiempo de llegada de una señal transmitida por un terminal móvil a diferentes estaciones fijas o viceversa. En este método es medido el tiempo de propagación de la señal entre la estación base y el dispositivo móvil, para ello es necesario una correcta sincronización entre ambas estaciones con el propósito de determinar el tiempo de propagación [2].

- **AOA (Ángulo de Llegada):** El método de Ángulo de Llegada puede ser dividido en dos clases: la que utiliza la respuesta de amplitud de la antena receptora y la que se basa en la respuesta de fase de la antena receptora [3]. Para la primera de ellas, el patrón de la antena se hace girar mecánica o eléctricamente y la dirección donde la potencia recibida sea máxima, se escoge como la dirección de la antena transmisora. La otra clase, respuesta de fase de la antena receptora, emplea un arreglo de antenas para determinar la diferencia de fases de dos elementos adyacentes del arreglo y con ello estimar la distancia a la que se encuentra el transmisor del arreglo de antenas receptoras.
- **RSS (Potencia de la Señal Recibida):** Este método se basa en la pérdida de potencia que la señal sufre debido al medio de propagación, en el caso de espacio libre, la potencia de la señal decae con el cuadrado de la distancia al punto de emisión [1].
- **GPS (Sistema de Posicionamiento Global):** Es un sistema de posicionamiento terrestre, la posición la calculan los receptores GPS gracias a la información recibida desde satélites en órbita alrededor de la Tierra. Consiste en una red de 24 satélites, propiedad del Gobierno de los Estados Unidos de América y gestionada por el Departamento de Defensa, que proporciona un servicio de posicionamiento para todo el globo terrestre. Cada uno de estos 24 satélites, situados en una órbita geoestacionaria a unos 20.000 km. de la Tierra y equipados con relojes atómicos, transmiten ininterrumpidamente la hora exacta y su posición en el espacio [4].

Los LBS se pueden dividir en 3 capas, ver la Figura 1.4:

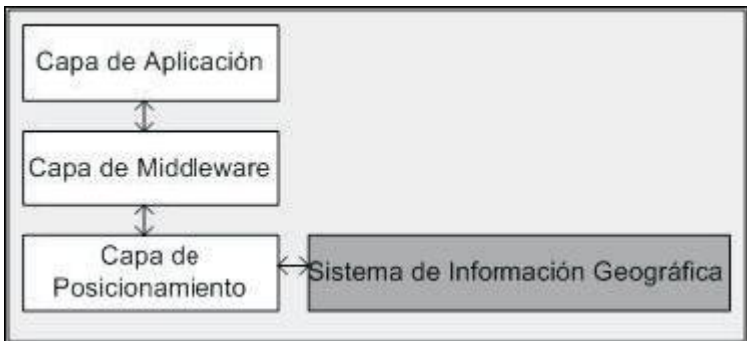


Figura 1.1: Capas de un sistema LBS.

- **Capa de Posicionamiento:** Se encarga de determinar la ubicación del dispositivo móvil mediante las técnicas o tecnologías descritas anteriormente y se apoya en un GIS.

- **Capa de Middleware:** Está entre la capa de posicionamiento y la de aplicación, de manera que esconde la complejidad de la información de posicionamiento. Además al tener una interfaz global permite la integración con aplicaciones de terceros y facilita el desarrollo.
- **Capa de Aplicación:** Comprende todos los servicios que hacen uso de la información de la capa de posicionamiento para su ejecución.

Los LBS son muy útiles para los usuarios y brindan muchos servicios que los benefician, algunos ejemplos son:

- Mostrar la ubicación de un determinado lugar.
- Determinar y mostrar la ruta a seguir a un conductor o peatón.
- Determinar la ruta óptima en función del tráfico.
- Determinar y mostrar la ubicación de un dispositivo móvil.
- Ubicar y seguir dispositivos móviles.
- Enviar alertas cuando un dispositivo móvil está en una determinada área o abandonó una determinada ubicación.
- Enviar automáticamente la posición ante emergencias.
- Enviar a los usuarios que posean dispositivos móviles información sensible respecto a la posición en la que se encuentren.
- Enviar a los usuarios que posean dispositivos móviles información relacionada con destinos o servicios cercanos.

Los servicios mencionados anteriormente demuestran que los LBS constituyen unos de los paradigmas de aplicaciones distribuidas que ayudan a mejorar la calidad de vida de la población. Es por ello que se han desarrollado disímiles aplicaciones que brindan estos servicios, tales como:

- **HERMES:** Es una aplicación que muestra en un mapa la trayectoria que haría un objeto que busca a otro en un lugar determinado, aplicándolo para encontrar el taxi más cercano.
- **Bluetooth Location – Based System:** Es un sistema bluetooth usado para mantener ubicado a los usuarios en un área específica enviándoles mensajes.

- **Google Latitude:** Permite saber la localización del usuario por medio de GPS y localización por WiFi (Fidelidad Inalámbrica del inglés Wireless Fidelity). La característica principal es poder encontrar amigos cercanos y contactarlos por SMS, IM (Mensaje Instantáneo del inglés Instant Messaging) o una llamada. Esta aplicación está disponible para teléfonos celulares y PCs (Computadoras Personales del inglés Personal Computer).

1.2.1 Servicios de Orientación y Navegación

Los servicios de Orientación y Navegación son una aplicación específica de los LBS. Estos servicios emplean la información suministrada por los usuarios para permitirle construir rutas sobre un mapa, llevarlos paso a paso por la misma, determinar la distancia a recorrer entre dos lugares y mostrar sitios de interés para el usuario. Esta información puede ser el nombre de un lugar o calle (se acepta tanto el nombre de la calle como el número).

Debido al gran impacto y auge que han alcanzado estos servicios en la sociedad se han desarrollado diversos sistemas por diferentes compañías de gran prestigio destinados a mostrar y editar cartografías a través de GIS, ejemplo de ellos son:

Google Maps: La herramienta cartográfica de Google ofrece mapas en 2D de calles y carreteras, mapas satelitales, vista dual y vista con el relieve del terreno. Google Maps está disponible vía Web en aplicaciones de escritorio como Google Earth y también para dispositivos móviles. Ofrece varios servicios como:

- Detecta la locación del usuario.
- Encontrar empresas junto con su información de contacto y como llegar a ellas dando una dirección o su nombre.
- Permite ver imágenes de las calles de una ciudad y poder navegar como si se estuviera ahí.
- Calcular rutas para caminar, trasladarse en auto, transporte público o bicicleta, determinando distancia y el tiempo del recorrido, además de mostrar sitios de interés en la ruta y tráfico. Esta opción solo está disponible en Estados Unidos.
- Incorporar otras búsquedas relacionadas con la ubicación actual del mapa como videos, fotografías y artículos de Wikipedia.

Ovi Maps Nokia: Es un servicio similar a Google Maps que también muestra mapas satelitales con los siguientes servicios:

- Cálculo y despliegue de rutas.
- Buscar direcciones y puntos de interés como restaurantes, hospitales o aeropuertos.
- Sincronizar sus lugares favoritos desde un dispositivo móvil compatible con los favoritos de una cuenta de Ovi.

Las aplicaciones Google Maps y Ovi Maps de Nokia ofrecen servicios que no le reportan ingresos a nuestro país. No son aplicaciones de código abierto, por tanto no pueden ser modificadas ni adaptadas a las exigencias de los usuarios cubanos, es decir, ante la necesidad de utilizar nuevos servicios que estas aplicaciones no brinden, se tendría que esperar por las nuevas versiones y estas incluso pueden no contar con los servicios deseados.

Los LBS proveen muchas informaciones que pueden ser accedidas desde los dispositivos móviles a través de redes de comunicación utilizando los siguientes protocolos:

HTTP (Protocolo de Transferencia de Hipertexto del inglés Hypertext Transfer Protocol): Protocolo utilizado para las comunicaciones con el proveedor de servicios.

WAP: Consiste en un conjunto de especificaciones, que se han desarrollado por medio del WAP Forum y que permite la utilización del WML (Lenguaje de Marcado Inalámbricas del inglés Wireless Markup Language), así como de WBMP (Wireless Bitmap) utilizando gráficos monocromáticos, permitiendo que los desarrolladores diseñen aplicaciones de interconexión para dispositivos portátiles. Gracias a esta tecnología dispositivos pequeños pueden conectarse a la Web. El diseño de WAP fue creado para trabajar bajo restricciones de memoria y procesadores, pequeñas pantallas monocromáticas capaces de desplegar muy pocas líneas de texto y conexiones irregulares debido al ancho de banda reducido. WAP permite una navegación cómoda, pensada para los usuarios con dispositivos móviles, que necesitan la información de forma rápida, oportuna y corta. La navegación se realiza mediante menús y submenús hasta llegar a lo que se quiere [5].

Ventajas de utilizar WAP [5]:

- Independencia sobre estándares para la creación de redes de telefonía.
- Completamente abierto y escalable.

- Independiente del sistema de transporte: GSM (Sistema Global para las Comunicaciones Móviles del inglés Global System for Mobile Communications), DECT (Telecomunicaciones Inalámbricas Mejoradas Digitalmente del inglés Digital Enhanced Cordless Telecommunications), TDMA (Acceso Múltiple por División de Tiempo del inglés Time Division Multiple Access).
- Independiente del tipo de dispositivo, ya sea teléfono celular o PDA (Asistente Personal Digital del inglés Personal Digital Assistant).
- Adaptable a nuevas tecnologías de transporte como: GPRS (Servicio General de Radio por Paquetes del inglés General Packet Radio Service) y UMTS (Sistema Universal de Telecomunicaciones Móviles del inglés Universal Mobile Telecommunications System).

1.3 Sistema de Información Geográfica

Aunque los GIS se empezaron a generalizar a partir de la década de los 80, su gestación y desarrollo se remonta dos décadas atrás. En la década de los 60 se desarrollaron varios proyectos donde se comienzan a manejar cartografías e información geográficas: se desarrolla el CGIS (Sistema de Información Geográfica de Canadá del inglés Canada Geographic Information System), Ian McHarg desarrolla su obra Design with nature, se desarrollan los sistemas SYMAP (Técnica Sinagráfica para elaboración de mapas del inglés Synagraphic Mapping Technique) y el MAP (Programa de Análisis de Mapas del inglés Maps Analysis Program) por la Universidad de Yale [6].

Los GIS han ido evolucionando y en la actualidad se asiste a la consolidación del GIS como industria, caracterizado por una progresiva integración de sistemas raster[1] y vectoriales[2], y por el aumento de la importancia de las comunicaciones entre sistemas y de la Interface de usuario, así como por el uso de herramientas de programación. Los nuevos campos de innovación de los GIS son la integración en

[1] **GIS raster:** Los elementos gráficos capturados y almacenado por un GIS raster desde un mapa analógico, lo son a través de la superposición sobre ellos de una rejilla de unidades regulares, de igual forma y tamaño, y donde cada unidad de la rejilla registra el valor que el mapa analógico adopta.

[2] **GIS vectoriales:** Un GIS vectorial está basado en la representación vectorial de la componente espacial de los datos geográficos, representando los objetos mediante las coordenadas de los puntos o vértices que los delimitan.

sistemas de soporte de decisiones, los llamados sistemas de sobremesa (divulgación de la cartografía[3] y de la Información Geográfica), los sistemas y servidores de información geográfica en red y distribuidos (Internet) y los llamados GIS móviles.

Los GIS son utilizados por universidades, gobiernos, empresas e instituciones, que lo han aplicado a sectores como los bienes raíces, la salud pública, la criminología, la defensa nacional, el desarrollo sostenible, los recursos naturales, la arqueología, la ordenación del territorio, el urbanismo, el transporte y la logística, entre otros.

Algunas definiciones de Sistema de Información Geográfica recogidas por Gutiérrez y Gould (1994) son [6]:

- Una base de datos computarizada que contiene información espacial.
- Un sistema que utiliza una base de datos espacial para generar respuestas ante preguntas de naturaleza geográfica.
- Un conjunto de procedimientos manuales o computarizados usados para almacenar y tratar datos referenciados geográficamente.
- Un potente conjunto de herramientas para recolectar, almacenar, recuperar a voluntad, transformar y presentar datos espaciales procedentes del mundo real.
- Sistema de hardware, software y procedimientos diseñado para realizar la captura, almacenamiento, manipulación, análisis, modelización y presentación de datos referenciados espacialmente para la resolución de problemas complejos de planificación y gestión.
- Sistema de Información diseñado para trabajar con datos georeferenciados[4] mediante coordenadas espaciales o geográficas.

Un GIS se puede definir como aquel método o técnica de tratamiento de la información geográfica que permite combinar eficazmente información básica para obtener información derivada. Para ello, cuentan

[3] **Cartografía:** Técnica que estudia los diferentes métodos o sistemas que permiten representar en un plano una parte o la totalidad de la superficie terrestre

[4] **Georeferencia:** Refiere al posicionamiento con el que se define la localización de un objeto espacial en un sistema de coordenadas.

tanto con las fuentes de información como con un conjunto de herramientas informáticas (hardware y software) que permiten esta tarea; todo ello enmarcado dentro de un proyecto que habrá sido definido por un conjunto de personas, y controlado, así mismo, por los técnicos responsables de su implantación y desarrollo. En definitiva, un GIS es una herramienta capaz de combinar información gráfica (mapas) y alfanumérica[5] (estadísticas) para obtener una información derivada sobre el espacio.

Dentro del software GIS se distinguen siete grandes tipos de programas informáticos:

- **GIS de escritorio**: Son aquellos que se utilizan para crear, editar, administrar, analizar y visualizar los datos geográficos. Se clasifican en tres subcategorías según su funcionalidad:
    1. **Visor GIS**: Suelen ser software sencillos que permiten desplegar información geográfica a través de una ventana que funciona como visor y donde se pueden agregar varias capas de información.
    2. **Editor GIS:** Es aquel software GIS orientado principalmente al tratamiento previo de la información geográfica para su posterior análisis.
    3. **GIS de análisis:** Disponen de funcionalidades de análisis espacial y modelización cartográfica de procesos.
- **SGBD (Sistemas de Gestión de Bases de Datos Espaciales)**: Se emplean para almacenar la información geográfica. También proporcionan la funcionalidad de análisis y manipulación de los datos. Una base de datos geográfica o espacial es una base de datos con extensiones que dan soporte de objetos geográficos permitiendo el almacenamiento, indexación, consulta y manipulación de información geográfica y datos espaciales.
- **Servidores cartográficos**: Se utilizan para distribuir mapas a través de Internet.
- **Servidores GIS**: Proporcionan básicamente la misma funcionalidad que los GIS de escritorio pero permiten acceder a estas utilidades de geoprocesamiento a través de una red informática.
- **Clientes web GIS**: Permiten la visualización de datos y acceder a funcionalidades de análisis y consulta de servidores GIS a través de Internet o intranet.

[5] **Alfanumérica:** Es un término colectivo para identificar letras del alfabeto latino y de números arábigos. Hay 36 o 62 caracteres alfanuméricos.

- **Bibliotecas y extensiones espaciales**: Proporcionan características adicionales que no forman parte fundamental del programa ya que pueden no ser requeridas por un usuario medio de este tipo de software. Estas nuevas funcionalidades pueden ser herramientas para el análisis espacial, herramientas para la lectura de formatos de datos específicos, herramientas para la correcta visualización cartográfica de los datos geográficos, o para la implementación de las especificaciones del OGC (Consorcio Geoespacial Abierto del inglés Open Geospatial Consortium).
- **GIS móviles**: Se usan para la recogida de datos en campos a través de dispositivos móviles (PDA, Smartphone).

Los GIS móviles son un segmento de rápido crecimiento en el mercado, puesto que muchas empresas importantes, como la Google, han incorporando esta tecnología para desarrollar diferentes sistemas, en el caso de la Google, Google Map. Todo esto ha sido posible gracias a los paulatinos avances en las aplicaciones GIS para dispositivos móviles y la mejora de los mismos que cuentan con gran capacidad de procesamiento y almacenamiento de datos.

Los GIS funcionan como una base de datos con información geográfica (datos alfanuméricos) que se encuentra asociada por un identificador común a los objetos gráficos de un mapa digital. De esta forma, señalando un objeto se conocen sus atributos e inversamente preguntando por un registro de la base de datos se puede saber su localización en la cartografía.

La razón fundamental para utilizar un GIS es la gestión de información espacial. El sistema permite separar la información en diferentes capas temáticas y las almacena independientemente, permitiendo trabajar con ellas de manera rápida y sencilla, facilitando la posibilidad de relacionar la información existente a través de la topología de los objetos, con el fin de generar otra nueva que no se obtendría de otra forma.

Las principales cuestiones que puede resolver un Sistema de Información Geográfica, ordenadas de menor a mayor complejidad, son:

- **Localización:** Preguntar por las características de un lugar concreto.
- **Rutas:** Cálculo de rutas óptimas entre dos o más puntos.

Por ser tan versátiles, el campo de aplicación de los GIS es muy amplio, se pueden utilizar en la mayoría de las actividades con un componente espacial. La profunda revolución que han provocado las nuevas tecnologías ha incidido de manera decisiva en su evolución.

1.4 Metodología de Desarrollo y Lenguaje de Modelado

1.4.1 Metodología de Desarrollo

Una metodología de desarrollo es una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia (cumplir los requisitos iniciales) y la eficiencia (minimizar las pérdidas de tiempo) en el proceso de generación de software. En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo [7]. En la actualidad el desarrollo del software se hace muy difícil pues los sistemas que se construyen crecen con gran facilidad.

Escoger la metodología apropiada para desarrollar un software es un tanto difícil, el tiempo de desarrollo, la cantidad de personas que desarrollarán y los requerimientos del cliente son factores que influyen en la selección de la metodología a utilizar que sea más adaptable al sistema.

Aplicando lo antes expuesto, la metodología de desarrollo seleccionada es RUP (Proceso Unificado de Rational del inglés Rational Unified Process) dada la alta complejidad y magnitud del sistema. También porque el objetivo principal de esta metodología es asegurar la construcción de sistemas de software de alta calidad que satisfagan las necesidades de los usuarios finales y clientes cumpliendo con los cronogramas y presupuestos previstos. Además genera una amplia documentación del desarrollo del software y contribuye a la organización del equipo de trabajo, factor imprescindible en el desarrollo del sistema debido a la gran cantidad de estudiantes que participan en el desarrollo.

Metodología RUP.

RUP es una metodología para proyectos de largo alcance, pero es estructurable y puede ser modificado en dependencia de las necesidades del proyecto y de la experiencia del desarrollador en el uso de esta metodología, logrando que se ajuste también a proyectos con un corto tiempo de desarrollo. RUP

constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

En RUP existen definidos 9 flujos de trabajo, los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo, y en 4 fases. A continuación se especificarán los flujos de trabajo y las fases.

Los flujos de trabajo son los siguientes:

- Modelamiento del negocio.
- Requerimientos.
- Análisis y diseño.
- Implementación.
- Prueba (Testeo).
- Instalación.
- Administración de configuración y cambios.
- Administración del proyecto.
- Ambiente.

En RUP existen 4 fases importantes:

- Inicio.
- Elaboración.
- Construcción.
- Transición.

Esta metodología de desarrollo de software se caracteriza por:

- Dirigido por casos de uso.
- Centrado en la arquitectura.
- Iterativo e Incremental.

1.4.2 Lenguaje de Modelado

UML (Lenguaje de Modelado Unificado del inglés Unified Modeling Language) fue originalmente concebido por la Corporación Rational Software. El lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, está respaldado por el OMG (Grupo de Gestión de Objetos del inglés Object Management Group). Es un lenguaje de modelado gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. La finalidad de estos es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementarlo. Ha sido adoptado por muchos proveedores de herramientas CASE (Ingeniería de Sistemas Asistida por Ordenador del inglés Computer Aided Software Engineering) y es el lenguaje de modelado soportado por RUP [8].

1.5 Tecnologías y herramientas utilizadas

1.5.1 Visual Paradigm

Es una herramienta CASE multiplataforma utilizada en un ambiente de software libre. Permite crear diferentes tipos de diagramas en un ambiente totalmente visual. Soporta además el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite graficar todos los tipos de diagramas de clases, generar documentación, generar código desde diagramas y código inverso. También proporciona tutoriales, demostraciones interactivas y proyectos UML.

Como principales características se encuentran [9]:

Beneficios:

- Navegación intuitiva entre el modelo visual y el código.
- Entorno visual de modelado superior.
- Soporte para toda la notación UML.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.

- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de integrarse en los principales IDEs (Entorno de Desarrollo Integrado del inglés Integrated Development Environment).
- Disponibilidad en múltiples plataformas.

1.5.2 Plataforma de Desarrollo Java

La tecnología Java está compuesta básicamente por 2 elementos: el lenguaje Java y su plataforma. Java es un lenguaje de alto nivel, orientado a objetos, multiplataforma, es decir, que permite la ejecución de un mismo programa en múltiples sistemas operativos, como dice el axioma de Java, "write once, run everywhere". Es distribuido ya que contiene un conjunto de clases que permiten la comunicación a través de la red facilitando así la creación de aplicaciones distribuidas. Tiene una plataforma, que a diferencia de las demás que son una combinación de hardware y software, está basada en software, que corre sobre cualquier hardware. Consta de 2 componentes: la máquina virtual de java (JVM) y la interfaz de programación de aplicaciones (API). Este lenguaje permite a los desarrolladores implementar software en una plataforma y ejecutarlo prácticamente en cualquier otra y combinar aplicaciones o servicios basados en la tecnología Java para crear servicios o aplicaciones totalmente personalizados, es libre y les proporciona a los desarrolladores una mayor flexibilidad, ahorro en los costes de desarrollo, independencia, compatibilidad con otros sistemas, entre otros beneficios [10] [11].

La plataforma Java fue desarrollada por Sun Microsystems. Esta plataforma se ha establecido en la industria como una de las principales herramientas de construcción de aplicaciones en las corporaciones, otorgándoles diversos beneficios, así como de un universo de aplicaciones, frameworks y estándares generados alrededor de la plataforma, que la complementan y extienden. La plataforma Java es una máquina virtual encargada de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java.

La plataforma Java es originaria de Sun Microsystems. Dentro de esta plataforma se pueden destacar tres subestándares [12]:

- JSE (Java Edición Estándar del inglés Java Standard Edition).

- JME (Java Edición Micro del inglés Java Micro Edition).
- JEE (Java Edición Empresa del inglés Java Enterprise Edition).

El lenguaje de programación Java es el único soportado por la inmensa mayoría de dispositivos móviles [13]. Los programas de Java son mucho más seguros que los de C o C++; es decir, existen menos posibilidades de que generen errores graves. Java no tiene punteros. La Máquina Virtual de Java se ocupa además de administrar la memoria. En su diseño, se incorporaron los últimos resultados en Ingeniería de Software orientado a objetos. Las bibliotecas estándar de Java también recurren con frecuencia a los patrones de diseño, que incrementan su flexibilidad. Java es de código abierto. Tiene licencia de software libre GPL (Licencia Pública General del inglés General Public License).

1.5.3 Lenguaje de Programación Java

Java es un lenguaje de desarrollo de propósito general orientado a objetos, para realizar todo tipo de aplicaciones, fue introducido por Sun Microsystems en 1995, es orientado a objetos y el heredero legítimo de C y C++. Uno de los principales elementos que hacen diferente a Java es que usa una máquina virtual para la ejecución de programas lo que permite que las aplicaciones escritas en java se puedan ejecutar en cualquier máquina, independientemente del sistema operativo y de la configuración de hardware.

El desarrollo de aplicaciones en Java nunca empieza desde cero. Java implementa consigo un gran número de clases, incluidas gratuitamente en su entorno de desarrollo, para realizar variadas tareas que permiten al desarrollador centrarse en el negocio y no en la implementación. Permite además, la incorporación de un gran número de componentes reutilizables.

Java ha sido mejorado, ampliado y probado por una comunidad especializada de más de 6,5 millones de desarrolladores, la mayor y más activa del mundo. Gracias a su versatilidad, eficiencia y portabilidad, Java se ha convertido en un recurso inestimable ya que permite a los desarrolladores [13]:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma.
- Crear programas para que funcionen en un navegador web y en servicios web.
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML (Lenguaje de Marcado de Hipertexto del inglés Hypertext Markup Language).
- Combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados.

- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier tipo de dispositivo digital.

1.5.4 J2EE (Java Edición Empresa 2)

JEE. Plataforma del lenguaje Java que provee una especificación de cómo debe construirse una aplicación empresarial, esta especificación describe como los servidores de aplicación deben proporcionar seguridad, escalabilidad, portabilidad, consistencia, manejo transaccional robusto e independencia de la plataforma tanto de hardware como de sistemas operativos, y al mismo tiempo para que el desarrollador final pueda desarrollar aplicaciones empresariales con menor esfuerzo.

1.5.5 Eclipse Galileo

Eclipse es un entorno de desarrollo integrado de código abierto y multiplataforma, es desarrollado por la Fundación Eclipse, una organización independiente que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. La definición que da el proyecto Eclipse acerca de su software es: "una especie de herramienta universal, un IDE abierto y extensible para todo y nada en particular" [14].

Tiene además una comunidad de desarrolladores realmente activa. Continuamente se están desarrollando nuevos plug-ins y revisando los anteriores. Al igual que en cualquier otro software, Eclipse permite la instalación de plug-ins destinados a mejorar las funcionalidades del propio IDE y a extenderse en cada vez más tecnologías. Eclipse Galileo es la versión 3.5 de este IDE presentada en el año 2009 [15].

1.5.6 Framework Spring

Spring es un framework de aplicación desarrollado por la compañía Interface 21, para aplicaciones escritas en el lenguaje de programación Java. Fue creado gracias a la colaboración de grandes programadores, entre ellos se encuentran como principales partícipes y líderes de este proyecto Rod Johnson y Jürgen Höller. Estos dos desarrolladores, además de otros colaboradores que juntando toda su experiencia en el desarrollo de aplicaciones J2EE, incluyendo EJB (Enterprise JavaBeans), Servlets y JSP (Página Servidora de Java del inglés Java Server Page), lograron combinar dichas herramientas y otras más en un sólo paquete, para brindar una estructura más sólida y un mejor soporte para este tipo de aplicaciones [16].

Además se considera a Spring un framework lightweight, es decir liviano o ligero, ya que no es una aplicación que requiera de muchos recursos para su ejecución, además el framework completo puede ser distribuido en un archivo .jar de alrededor de 1 MB, lo cual representa muy poco espacio, y para la cantidad de servicios que ofrece es relativamente insignificante su tamaño [16].

Spring es un framework de código abierto, lo cual implica que no tiene ningún costo, ni se necesita una licencia para utilizarlo, por lo tanto da la libertad a muchas empresas y desarrolladores a incursionar en la utilización de este framework. Además de que está disponible todo el código fuente de este framework en el paquete de instalación.

Spring no intenta "reinventar la rueda" sino integrar las diferentes tecnologías existentes, en un sólo framework para el desarrollo más sencillo y eficaz de aplicaciones J2EE portables entre servidores de aplicación.

Otro de los principales enfoques de Spring y por el cual está ganando dicha popularidad es que simplifica el desarrollo de aplicaciones J2EE, al intentar evitar el uso de EJB, ya que como menciona Craig Walls en su libro Spring in Action, "En su estado actual, EJB es complicado. Es complicado porque EJB fue creado para resolver cosas complicadas, como objetos distribuidos y transacciones remotas." Muchas veces aunque el proyecto no es lo suficientemente complejo, se utiliza EJB, contenedores de alto peso y otras herramientas que soportan un grado mayor de complejidad, como una solución a un proyecto. "Con Spring, la complejidad de tu aplicación es proporcional a la complejidad del problema que se está resolviendo." Esto sin embargo no le quita crédito a EJB, ya que también ofrece a los desarrolladores servicios valiosos y útiles para resolver ciertas tareas, la diferencia radica en que Spring intenta brindar los mismos servicios pero simplificando el modelo de programación [16].

Spring fue creado basado en los siguientes principios:

- El buen diseño es más importante que la tecnología subyacente.
- Los JavaBeans ligados de una manera más libre entre interfaces es un buen modelo.
- El código debe ser fácil de probar.

Spring es un framework modular que cuenta con una arquitectura divida en siete capas o módulos, como se muestra en la Figura 1.5, lo cual permite tomar y ocupar únicamente las partes que interesen para el proyecto y juntarlas con gran libertad.

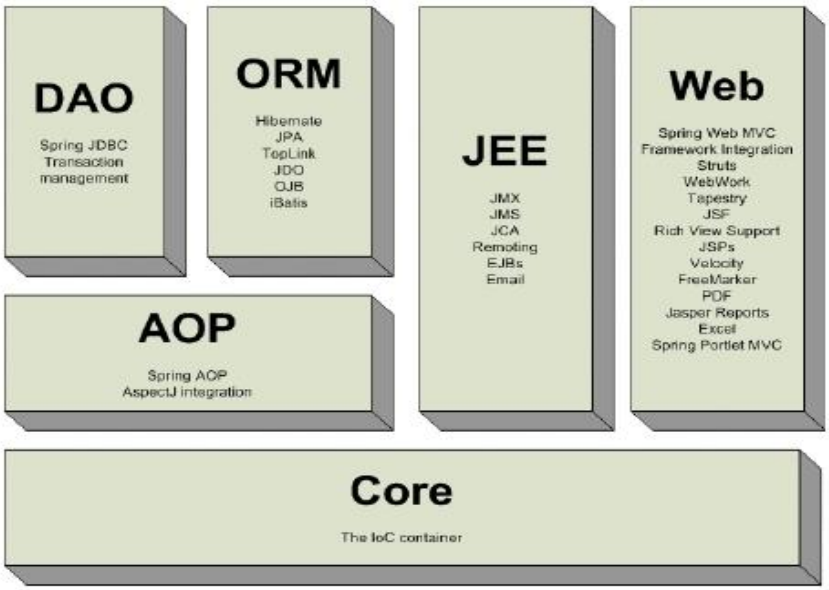


Figura 1.2: Arquitectura de Spring.

Spring Modelo Vista Controlador

Spring brinda un MVC para web flexible y altamente confiable, pero esta flexibilidad no le quita sencillez, ya que se pueden desarrollar aplicaciones sencillas sin tener que configurar muchas opciones. Para esto se puede utilizar muchas tecnologías ya que Spring brinda soporte para JSP, Struts[6], entre otros [16].

Spring MVC presenta algunas similitudes con otros frameworks para web existentes, pero son las siguientes características las que lo vuelven único [16]:

- Spring hace una clara división entre controladores, modelos de JavaBeans y vistas.
- El MVC de Spring está basado en interfaces y es bastante flexible.
- Provee interceptores al igual que controladores.
- Spring no obliga a utilizar JSP como única tecnología View también se puede utilizar otras.
- Los Controladores son configurados de la misma manera que los demás objetos en Spring, a través de IoC (Inversión de Control del inglés Inversion of Control).
- Los web tiers[7] son más sencillos de probar que en otros frameworks.

[6] **Struts:** Es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma Java EE.

- El web tiers se vuelve una pequeña capa delgada que se encuentra encima de la capa de bussiness objects.

La arquitectura básica de Spring MVC está ilustrada en la Figura 1.6:

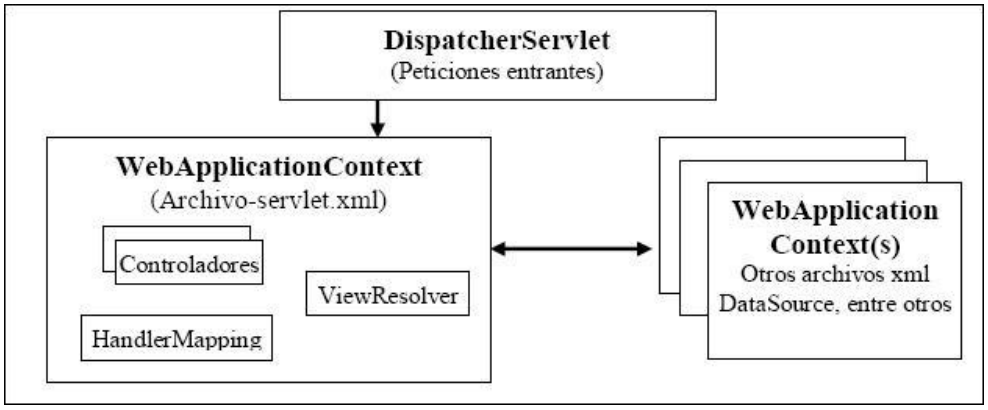


Figura 1.3: Arquitectura de Web MVC de Spring.

1.5.7 Servidor Apache Tomcat

Apache es un servidor de páginas web de código abierto multiplataforma y modular, se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. Tomcat es un contenedor de Servlets con un entorno JSP, puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad [17].

Características principales:

- Trabaja sobre múltiples plataformas (Unix, Linux, MacOSX, Vms, Win32, OS2).
- Incluye módulos que se cargan de forma dinámica.
- Soporta CGI (Interfaz de Entrada Común del inglés Common Gateway Interface), Perl, PHP (Procesador de Hipertexto del inglés Hypertext Preprocessor).
- Soporte para Bases de datos.

[7] **Tiers:** Es un sistema para clasificar la fiabilidad y hacer negocios certificando los centros de datos.

- Código Abierto.

1.5.8 PostgreSQL

PostgreSQL es un ORDBMS (sistema de gestión de bases de datos objeto-relacional del inglés Object Relational Database Management System) basado en el proyecto POSTGRES, de la universidad de Berkeley. PostgreSQL es una derivación libre de este proyecto, y utiliza el lenguaje SQL92/SQL99, así como otras características [**Error! Reference source not found.**]:

- **Atomicidad:** Es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- **Consistencia:** Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
- **Aislamiento:** Es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generará ningún tipo de error.
- **Durabilidad:** Es la propiedad que asegura que una vez realizada la operación, esta persistirá y no se podrá deshacer aunque falle el sistema.
- **Multiplataforma:** Corre en los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows.
- **Documentación:** Muy bien organizada, pública y libre, con comentarios de los propios usuarios.

1.5.9 JPA (Java Persistence API)

JPA proporciona un modelo de persistencia basado en POJO's (Objetos Antiguos de Java Planos del inglés Plain Old Java Object) para mapear bases de datos relacionales en Java. JPA fue desarrollado por el grupo de expertos de EJB 3.0 como parte de JSR (Solicitud de Especificaciones de Java del inglés Java Specification Request) 220, aunque su uso no se limita a los componentes software EJB. También puede utilizarse directamente en aplicaciones web y aplicaciones clientes. En su definición, se han combinado ideas y conceptos de los principales frameworks de persistencia como Hibernate, Toplink y JDO (Objetos de Datos Java Objects), y de las versiones anteriores de EJB, todos estos cuentan actualmente con una implementación JPA.

El mapeo objeto/relacional, es decir, la relación entre entidades Java y tablas de la base de datos, se realiza mediante anotaciones en las propias clases de entidad, por lo que no se requieren ficheros descriptores XML (Lenguaje de Marcas Extensibles del inglés eXtensibles Markup Language). También pueden definirse transacciones como anotaciones JPA [**Error! Reference source not found.**].

1.5.10 Herramienta de Control de Versiones

SVN (Subversion) es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS (Sistema de Versiones Concurrentes del inglés Concurrent Versions System). Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como SVN por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente, en cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

Subversion maneja ficheros y directorios a través del tiempo. Permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Subversion es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros [19].

1.6 Conclusiones del Capítulo

En el presente capítulo se abordaron los conceptos y características tanto de los LBS como de los GIS. Se analizó el estado actual de soluciones existentes en el mundo que emplean los LBS. La metodología de desarrollo, lenguaje de modelado, herramientas y tecnologías utilizadas para el desarrollo del sistema fueron definidas por el equipo de desarrollo del proyecto Cubacel.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En el presente capítulo se realiza un estudio del problema con el objetivo de diseñar una interfaz web para dispositivos móviles empleando LBS y eliminar los problemas de orientación y navegación de los usuarios cubanos. Además se identifican los requerimientos funcionales del sistema, con el propósito de encontrar actores y casos de uso que brinden solución al problema planteado. Se confeccionan el modelo de dominio y los diagramas de caso de uso y se define la solución propuesta, así como la descripción de los casos de uso, para una mejor comprensión de la misma.

2.1.1 Objeto de Automatización

Se debe desarrollar una aplicación web para dispositivos móviles que permita buscar rutas en dependencia del medio de transporte y realizar búsquedas ya sea simples o filtradas por categorías de lugares y que el resultado de dicha búsqueda sea mostrado en un mapa al cual se le puedan realizar modificaciones como: acercamientos, alejamientos, movimientos a la izquierda, derecha, arriba y abajo. Una vez realizadas estas modificaciones el sistema debe posibilitar mostrar el mapa en sus dimensiones originales. Además se debe mostrar información relevante del lugar que se esté buscando y permitir mostrar un lugar a partir de una búsqueda realizada anteriormente (se deben almacenar las últimas 10 búsquedas realizadas por el usuario).

2.2 Propuesta del Sistema

Se desea desarrollar una aplicación web para dispositivos móviles que implemente un servicio de Orientación y Navegación para proporcionarles servicios personalizados a los usuarios que la empleen. La aplicación debe brindar la posibilidad al usuario de buscar rutas en dependencia del medio de transporte y realizar búsquedas simples o filtradas por categoría de lugares de interés y mostrarle en el mapa el resultado de la búsqueda, así como de otros sitios que se encuentren a su alrededor. Además se debe permitir al usuario interactuar con el mapa. El sistema debe posibilitar a los usuarios visualizar el portal desde cualquier dispositivo móvil, ya sea desde uno estándar (menor velocidad de procesamiento y capacidad de almacenamiento) hasta uno más sofisticado (mayor velocidad de procesamiento sin restricciones de memoria).

2.3 Modelo de Dominio

Se propone un modelo de dominio para definir bien en el sistema los procesos del negocio. Además permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio de la aplicación en desarrollo y ayuda a los usuarios, clientes, desarrolladores e interesados, a utilizar un vocabulario común para poder entender el contexto en que se sitúa. Este modelo va a contribuir posteriormente a identificar algunas clases que se utilizarán en el sistema. Ver figura 2.1.

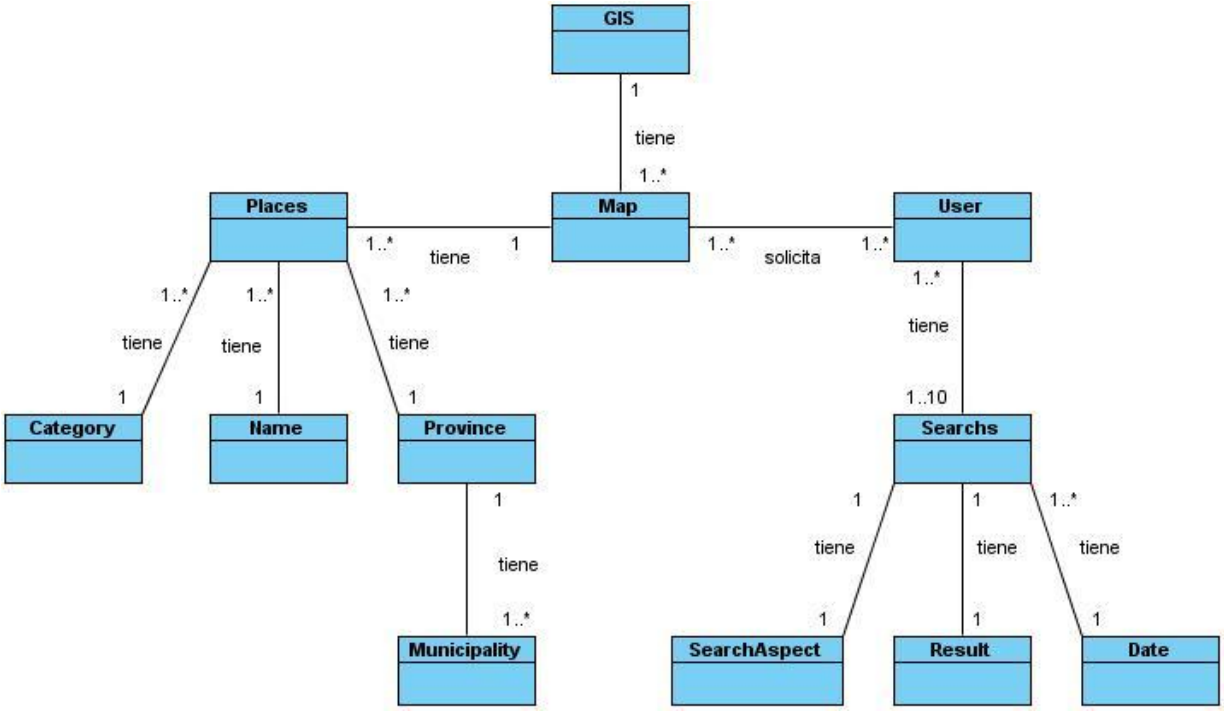


Figura 2.1: Modelo de Dominio.

User: Representa al usuario final del portal web para dispositivos móviles.

GIS: Representa el GIS.

Maps: Representa los datos de una localización.

Places: Representa los lugares de un mapa.

Category: Representa la categoría del lugar.

Name: Representa el nombre del lugar.

Province: Representa la provincia a la que pertenece el lugar.

Municipality: Representa los municipios pertenecientes a la provincia.

Searchs: Representa las búsquedas realizadas por el usuario.

SearchAspect: Representa el criterio de búsqueda.

Result: Representa el resultado de la búsqueda.

Date: Representa la fecha en la que fue realizada la búsqueda.

2.4 Especificación de Funcionalidades

2.4.1 Requerimientos Funcionales

RF1. Buscar Lugar por nombre

El sistema debe permitir realizar búsquedas a partir del lugar introducido por el usuario.

RF2. Buscar Lugar por categoría

El sistema debe permitir realizar búsquedas a partir de la categoría seleccionada por el usuario.

RF3. Mostrar lugar

El sistema debe permitir mostrar el mapa con el lugar señalado, así como otros sitios de interés que se encuentren a su alrededor.

RF4. Mostrar información relevante del lugar

El sistema debe mostrar información del lugar: nombre completo, dirección, código de la provincia y número de teléfono.

RF5. Guardar búsquedas

El sistema debe permitir almacenar los datos de las 10 últimas búsquedas realizadas por cada usuario.

RF6. Mostrar búsquedas realizadas

El sistema muestra las 10 últimas búsquedas realizadas por el usuario ordenadas por la fecha.

RF7. Mostrar lugar por búsquedas realizadas

El sistema debe permitir mostrar el mapa con el lugar señalado, así como otros sitios de interés que se encuentren a su alrededor, a partir de una búsqueda realizada.

RF8. Aumentar mapa

El sistema debe permitir incrementar el tamaño del mapa.

RF9. Disminuir mapa

El sistema debe permitir disminuir el tamaño del mapa.

RF10. Mover mapa a la derecha

El sistema debe permitir mover el mapa hacia la derecha.

RF11. Mover mapa a la izquierda

El sistema debe permitir mover el mapa hacia la izquierda.

RF12. Mover mapa arriba

El sistema debe permitir mover el mapa hacia arriba.

RF13. Mover mapa abajo

El sistema debe permitir mover el mapa hacia abajo.

RF14. Mostrar estado inicial del mapa

El sistema debe permitir mostrar el mapa en sus dimensiones originales.

RF15. Mostrar ruta teniendo en cuenta medio de transporte.

El sistema debe permitir mostrar la ruta entre dos lugares seleccionados por el usuario en dependencia de cómo se traslade: a pie o en ómnibus.

RF16. Mostrar ayuda

El sistema debe permitir mostrar información sobre el uso correcto de la aplicación.

2.4.2 Requerimientos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el sistema necesita. Estas propiedades se ven como las características que hacen al producto atractivo, usable, rápido o confiable. Teniendo en cuenta los requerimientos funcionales identificados, se plantean como requerimientos no funcionales los siguientes:

Requerimientos de Apariencia o Interfaz Externa

El diseño de la aplicación contará con interfaces sencillas donde se muestren las imágenes, vínculos, botones y textos necesarios, con el objetivo de mejorar la velocidad de respuesta del sistema y que le resulte fácil y agradable la navegación al usuario desde su dispositivo móvil.

Requerimientos de Interfaz Interna

La implementación de las funcionalidades debe seguir los lineamientos de código establecidos por la dirección técnica de producción de la UCI para los proyectos desarrollados en JAVA, para garantizar la comprensión del código por desarrolladores de futuras versiones.

Requerimientos de Software

Servidor:

- JVM (Máquina Virtual de Java del inglés Java Virtual Machine) 1.6 o superior.
- Servidor web Apache Tomcat versión 6.x o superior.
- Servidor de Base de Datos PostgreSQL 8.3 o superior.
- Sistema Operativo GNU Linux distribución Ubuntu Server 10.04 o superior.

Cliente:

- Navegador en el dispositivo móvil.
- Servicio de navegación GPRS habilitado.

Requerimientos de Hardware

Servidor:

- Procesador Intel Pentium IV Core™ 2 Duo 3.00 GHz o superior.

- Memoria RAM mínima de 2 GB.
- Al menos 160 GB de capacidad de disco duro.
- Tarjeta Ethernet de 1GBit.

Cliente:

- Tarjeta SIM (Módulo de Identificación del Suscriptor del inglés Subscriber Identity Module) de Cubacel.

Requerimientos de Usabilidad

La aplicación podrá ser manejada por cualquier usuario con conocimientos básicos en la utilización de un dispositivo móvil como:

- Acceder al navegador.
- Introducir textos.

Requerimientos de Portabilidad

La aplicación deberá ser compatible con cualquier dispositivo móvil que soporte XHTML (Lenguaje de Marcado de Hipertexto Extensible del inglés eXtensible Hypertext Markup Language), HTML, WML y que posea un navegador, además tenga activado el servicio de navegación GPRS en Cuba.

Requerimientos de Fiabilidad

Disponibilidad:

- El sistema debe estar disponible las 24 horas salvo en los momentos que sea necesario realizar tareas de mantenimiento.
- Las tareas de mantenimiento se realizaran después de las 2 am y antes de las 4am, los días que se considere necesario y tendrán una duración máxima de 30 min.

Confiabilidad:

- El sistema debe ser capaz de recuperarse ante la ocurrencia de un fallo, de no ser posible, emitir alertas al personal encargado de la administración del mismo, así como proteger la información y contenidos.

Requerimientos de Ayuda y Documentación

El sistema brindará a los usuarios una ayuda para mayor facilidad de trabajo con la aplicación.

2.5 Modelo de Caso de Uso del Sistema

2.5.1 Definición de Actores del Sistema.

Actores	Descripción
Usuario	El usuario puede realizar búsquedas, conocer la ruta entre dos lugares, consultar ayuda, realizar modificaciones al mapa como: acercamientos, alejamientos, movimientos a la izquierda, derecha, arriba y abajo. Ver el mapa en sus dimensiones originales.

Tabla 2.1: Descripción de Actores del Sistema.

2.5.2 Diagrama de Casos de Uso del Sistema a Automatizar

Un Diagrama de Casos de Uso del Sistema documenta el comportamiento de un sistema desde el punto de vista del usuario. Muestra la relación entre los actores y los casos de uso del sistema. Los elementos básicos que componen un Diagrama de Casos de Uso son: actores, casos de uso y relaciones entre casos de uso.

Después de haber confeccionado el modelo de dominio y definidos los requisitos funcionales, se obtuvo el siguiente Diagrama de Caso de Uso del Sistema como se muestra en la Figura 2.2.

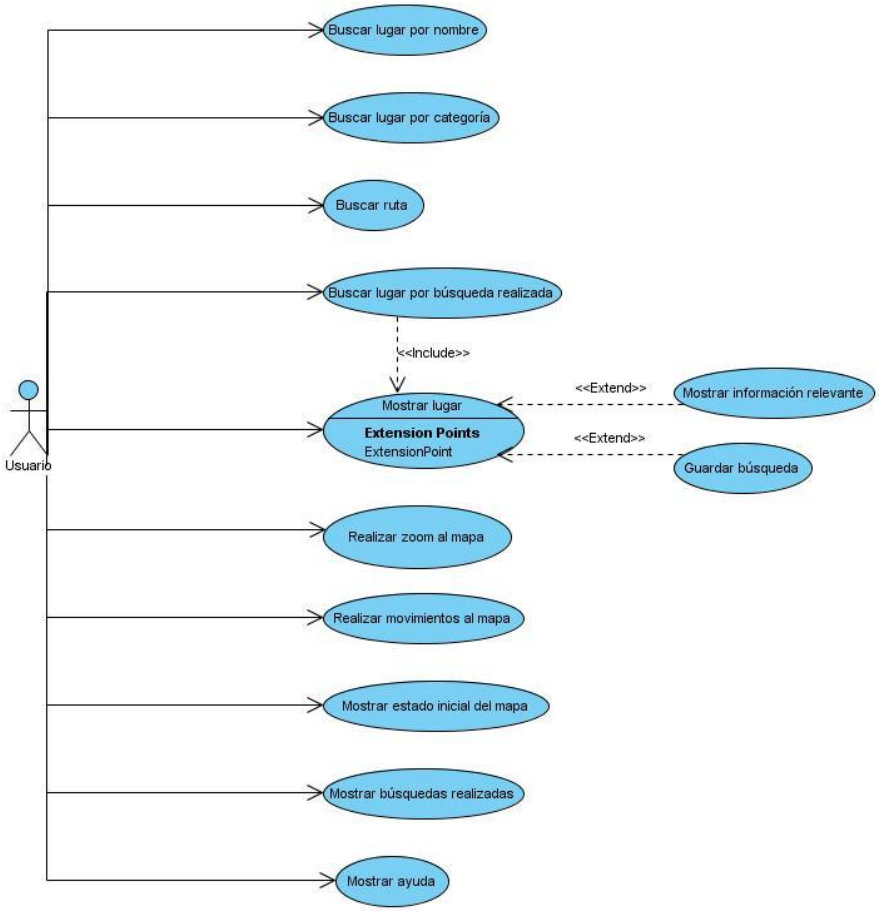


Figura 2.2: Diagrama de Caso de Uso del Sistema.

2.5.3 Descripción de Casos de Uso

En el ***Anexo2*** aparecen descritos los Casos de Uso del Sistema definidos en el diagrama.

2.6 Conclusiones del Capítulo

En el presente capítulo se realizó una descripción de la solución propuesta, se representó el modelo de dominio con los principales objetos del sistema. Se definieron los requisitos funcionales y no funcionales. Se realizó el diagrama de casos de uso del sistema, así como la descripción detallada de estos. Los casos de usos identificados dirigirán el trabajo a lo largo del resto de los flujos de trabajos propuestos por la metodología de desarrollo de software utilizada (RUP).

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1 Introducción

En este capítulo se describen los procesos que se llevan a cabo en el flujo de trabajo de Diseño. Está enfocado hacia como construir técnicamente el software. Se organizan los diagramas que describen el análisis y diseño de las clases.

3.2 Patrones de Diseño

Los patrones de diseño pueden usarse durante el diseño del software. Una vez que se ha realizado el modelo de análisis, el diseñador puede examinar una representación detallada del problema que debe resolver y las restricciones que impone el mismo. La descripción del problema se examina en varios grados de abstracción para determinar si es flexible para uno o más de los siguientes tipos de patrones de diseño:

Patrones arquitectónicos: Estos patrones definen la estructura general del software, indican las relaciones entre los subsistemas y los componentes del software y definen las reglas para especificar las relaciones entre los elementos (clases, paquetes, componentes, subsistemas) de la arquitectura.

Patrones de diseño: Estos patrones se aplican a un elemento específico del diseño como un agregado de componentes para resolver algún problema de diseño, relaciones entres los componentes o los mecanismos para efectuar la comunicación de componente a componente [21].

3.2.1 Patrón Arquitectónico

Patrón Modelo Vista Controlador

En la mayoría de las aplicaciones web la lógica de una interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio. Con el patrón MVC se realiza un diseño que desacopla la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos. Elementos del patrón:

- Modelo: Datos y reglas de negocio.
- Vista: Muestra la información del modelo al usuario.

- Controlador: Gestiona las entradas del usuario.

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Definir las reglas de negocio (la funcionalidad del sistema).
- Llevar un registro de las vistas y controladores del sistema.

El controlador es responsable de:

- Recibir los eventos de entrada (un clic, un cambio en un campo de texto).
- Contener reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada a actualizar.

Las vistas son responsables de:

- Recibir datos del modelo y mostrarlos al usuario.
- Tener un registro de su controlador asociado.
- Poder dar el servicio de actualizar, para que sea invocado por el controlador o por el modelo [22].

Se hace uso de este patrón en el sistema con la utilización del framework Spring MVC que permite desacoplar la lógica de la vista con respecto a la lógica del modelo y minimizar el impacto de los cambios entre la interfaz y el negocio; donde las clases vistas son las jsp, las controladoras son los controller y las clases principales del modelo son las manager y dao.

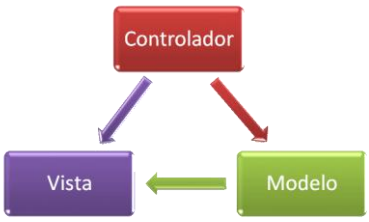


Figura 3.1: Diagrama de paquetes. Vista arquitectónica.

3.2.2 Patrones de Diseño

Patrón IoC (Inversión de Control) / DI (Inyección de Dependencia del inglés Dependency Injection): Inversión de Control es un principio que generalmente siguen los framework. Spring proporciona un contenedor que maneja el ciclo de vida de los objetos y resuelve las dependencias entre dichos objetos. Se suelen confundir los términos de Inversión de Control e Inyección de Dependencia, pero no son lo mismo. Mientras que la Inversión de Control es un principio de diseño general, la Inyección de Dependencia es un patrón de diseño concreto que encarna este principio. La Inyección de Dependencia es la más típica realización de la Inversión de Control. El patrón Inyección de Dependencia, es una técnica que busca facilitar la resolución de dependencias entre objetos. Consiste en inyectar objetos a una clase en lugar de ser la propia clase quien cree el objeto. La forma habitual de implementar este patrón es mediante un contenedor. El contenedor inyecta a cada objeto los objetos necesarios según las relaciones plasmadas en un fichero de configuración. Típicamente este contenedor es implementado por un framework externo a la aplicación (como Spring). El uso de este patrón permite construir aplicaciones mucho más flexibles y robustas [23]. Se hace uso de este patrón utilizando el framework Spring a la hora de inyectar las clases que implementan las interfaces incluidas en las Impl y Manager, por ejemplo:

<bean id = “iSearchDao” class = “wap.dao.repository.impl.SearchDaoImpl”/>

<bean id = “searchManager” class = “wap.standard.localization.manager.SearchLocalizationManager” >

<property name = “iSearch” ref = “iSearchDao”/>

</bean>

Patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades del inglés General Responsibility Asignment Software Patterns)

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

- **Controlador**

Asignar la responsabilidad del manejo de los eventos de un sistema a una clase [**Error! Reference source not found.**].

En la aplicación este patrón se ve reflejado en las clases controladoras (Controller), las cuales son las encargadas de manejar las peticiones del usuario y los eventos lanzados en la aplicación, obtener los datos del modelo y entregarlos a las vistas (jsp) que los mostrarán al usuario final, por ejemplo: la clase LocalizationMapController se encarga de recibir la petición de mostrar un lugar, obtener las coordenadas del lugar de la SearchLocalizationManager y mostrar en la localizationMap el lugar señalado en el mapa.

- **Experto**

Se utiliza para guiar la asignación de responsabilidades en cuanto a la creación de objetos, es decir que un objeto contenga, agregue, utiliza y/o registra otro con su respectiva información [**Error! Reference source not found.**].

Este patrón se ve reflejado en las clases del negocio como los Manager donde cada una realiza funciones específicas, por ejemplo: la clase LocalizationManager se encarga de calcular todas las modificaciones realizadas al mapa (resetear, ampliar, disminuir, mover hacia la izquierda, derecha, arriba y abajo).

- **Alta Cohesión**

Una clase de diseño cohesiva tiene un conjunto de responsabilidades pequeño y enfocado, y aplica atributos y métodos de manera sencilla para implementar dichas responsabilidades [24].

En el sistema se tiene bien definido las responsabilidades de cada clase, se definen clases del negocio por módulo como las Manager, para evitar la sobrecarga de funcionalidades en estas clases, por ejemplo: la clase ValidationManager se encarga solamente de validar los datos provenientes de las vistas, la clase LocalizationManager de calcular todas las modificaciones realizadas al mapa, la clase SearchLocalizationManager de guardar y obtener búsquedas y la clase GeograficLocalizationManager de realizar búsquedas de lugares y coordenadas.

- **Bajo Acoplamiento**

Dentro del modelo de diseño es necesario que las clases de diseño colaboren con alguna otra. Sin embargo, la colaboración se debe mantener en un mínimo aceptable. Si un modelo de diseño tiene un acoplamiento alto (todas las clases de diseño colaboran con todas las otras clases de diseño), el sistema es difícil de implementar, probar y mantener a través del tiempo. En general, las clases de diseño dentro de un subsistema deben tener solo un conocimiento limitado de las clases en otros subsistemas. Esta

restricción, llamada la Ley de Deméter [26], sugiere que un método sólo debe enviar mensajes a métodos de clases vecinas [24].

En el sistema se usa este patrón en las clases dao que contienen interfaces que son implementadas por los DaoImpl haciendo uso de las clases entidades, por ejemplo: la clase PlacesDaoImpl es la encargada de implementar los métodos contenidos en la clase interface IPlacesDao haciendo uso de la clase entidad PlacesEntity.

Patrones GOF

- **Patrón Instancia Única**

Este patrón tiene como propósito garantizar que una clase sólo tiene una única instancia, proporcionando un punto de acceso global a la misma [27].

En el sistema se utiliza este patrón para inyectar una única instancia de la clase LocalizationManager a las controladoras LocalizationMapController y ModificateLocalizationController y así mantener acceso global a la misma, por ejemplo: la clase LocalizationMapController obtiene las coordenadas del mapa a través de la instancia de la clase LocalizationManager. Estas coordenadas se necesitan pues se continuarán modificando en la clase ModificateLocalizationController de acuerdo a las modificaciones realizadas al mapa por el usuario (resetear, ampliar, disminuir, mover hacia la izquierda, derecha, arriba y abajo).

3.3 Diagramas de Paquetes del Diseño

Un diagrama de paquetes permite dividir al sistema orientado a objetos, organizándolo en subsistemas y detallando sus relaciones. Contiene dos tipos de elementos:

Paquetes: Es una colección de clases, relaciones, realizaciones de casos de uso, diagramas y otros paquetes que estén de alguna forma relacionados. Es usado para estructurar el modelo de diseño dividiéndolo en partes más pequeñas. Se representa mediante un símbolo en forma de carpeta, el nombre se coloca en la pestaña y el contenido dentro de la carpeta.

Dependencias: Indican que un elemento de un paquete requiere a otro de un paquete distinto. Se representan mediante una flecha discontinua con inicio en el paquete que depende de otro.

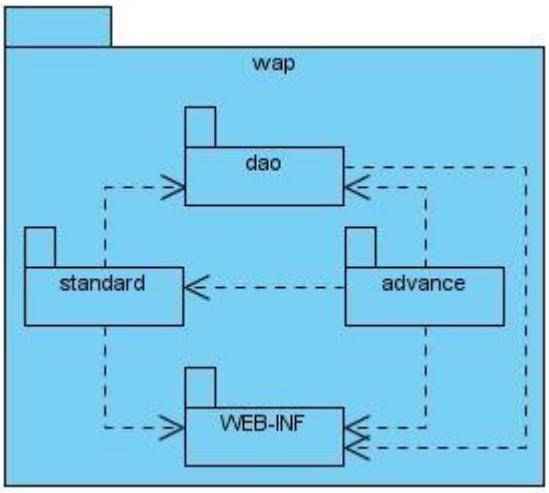


Figura 3.2: Diagrama de Paquetes de Diseño.

Paquete dao: Contiene los paquetes domain y repository, ver Figura 3.3. El paquete domain agrupa a las clases entidades y el paquete repository a las clases interfaces y al paquete impl. Este último es el encargado de implementar las clases interfaces que soportan los procesos de acceso a datos.

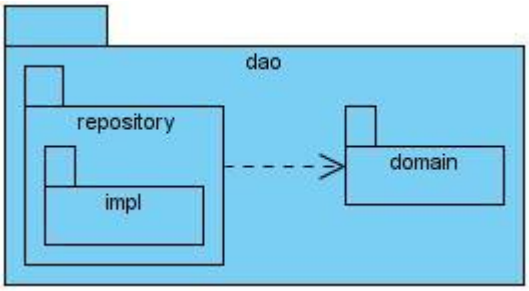


Figura 3.3: Diagrama de Paquetes del paquete dao.

Paquete standard: Contiene el paquete localization, el que contiene a su vez los paquetes controller, manager y conf, ver Figura 3.4. El paquete controller agrupa a todas las clases que soportan el proceso de atender las peticiones realizadas por el usuario y haciendo uso de las clases manager que se encuentran en el paquete manager, realizar las operaciones necesarias para dar respuesta a los mismos. El paquete conf contiene un archivo XML de configuración (localizationStandard-context.xml) que se encarga de inyectar objetos a una o varias clases, así como de asignarle una clase a determinada petición proveniente de las vistas.

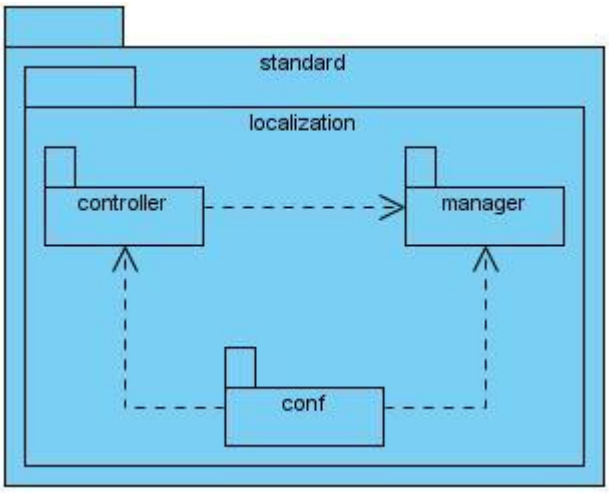


Figura 3.4: Diagrama de Paquetes del paquete standard.

Paquete advance: Contiene el paquete localization, el que tiene a su vez los paquetes controller y conf, ver Figura 3.5. El paquete controller agrupa a todas las clases que soportan el proceso de atender las peticiones realizadas por el usuario y haciendo uso de las clases manager que se encuentran en el paquete manager del paquete standard, realizar las operaciones necesarias para dar respuesta a los mismos. El paquete conf contiene un archivo XML de configuración (localizationAdvance-context.xml) que se encarga de inyectar objetos a una o varias clases, así como de asignarle una clase a determinada petición proveniente de las vistas.

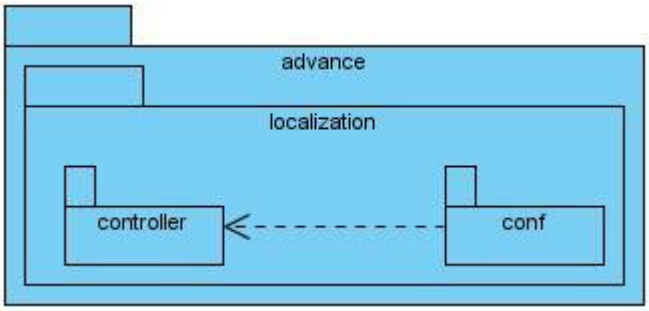


Figura 3.5: Diagrama de Paquetes del paquete advance.

Paquete WEB-INF: Contiene el paquete jsp y lib, ver Figura 3.6. El paquete jsp contiene los paquetes standard y advance, los cuales agrupan a todas las vistas que se encargan de visualizar la información proveniente del modelo. El paquete lib contiene todas las librerías java que son utilizadas por todos los paquetes contenidos en el paquete wap.

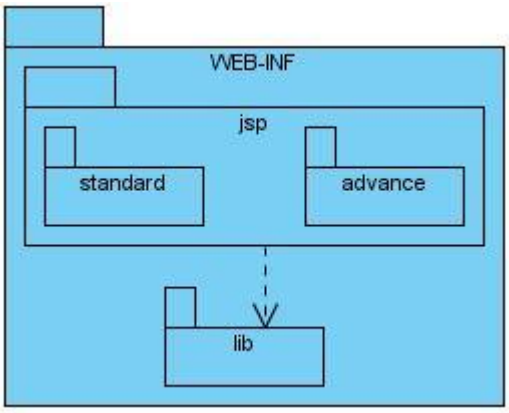


Figura 3.6: Diagrama de Paquetes del paquete WEB-INF.

3.4 Diagrama de Clases del Diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación.

Los diagramas de clases del diseño contienen la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información sobre los tipos de los atributos.
- Navegabilidad.
- Dependencias.

En el ***Anexo3*** aparecen los diagramas de Clases del Diseño.

3.4.1 Descripción de Clases

A continuación se realiza una descripción de las principales clases que implementan las funcionalidades de los casos de uso arquitectónicamente significativos.

- **Clase "PlacesEntity"**

Figura 3.7: Clase PlacesEntity.

Objetivo: Definir la información que se gestiona de cada lugar.

- **Clase "PlacesDaoImpl"**

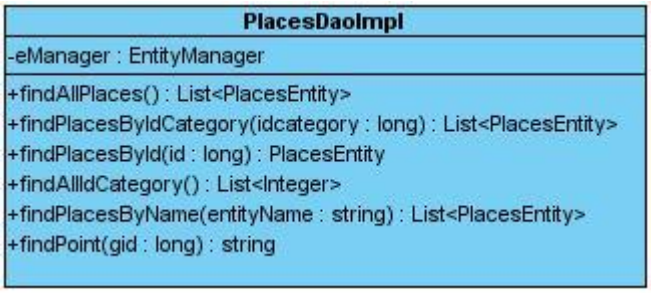


Figura 3.8: Clase PlacesDaoImpl.

Objetivo: Acceder y realizar operaciones con los datos almacenados en la tabla "americas_caribbean_cuba_poi".

- **Clase "PhoneCategoryEntity"**

Figura 3.9: Clase PhoneCategory.

Objetivo: Definir la información que se gestiona de cada categoría.

- **Clase "PhoneCategoryDaoImpl"**

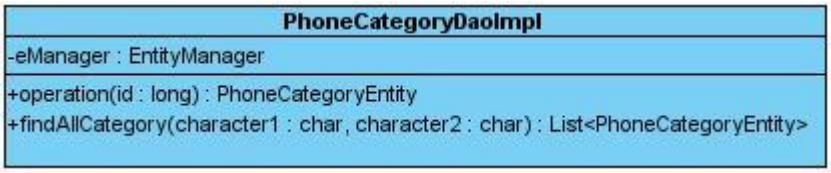


Figura 3.10: Clase PhoneCategoryDaoImpl.

Objetivo: Acceder y realizar operaciones con los datos almacenados en la tabla "phonecategory".

- **Clase "ClientEntity"**

Figura 3.11: Clase ClientEntity.

Objetivo: Definir la información que se gestiona de cada usuario.

- **Clase "SearchEntity"**

Figura 3.12: Clase SearchEntity.

Objetivo: Definir la información que se gestiona de cada búsqueda.

- **Clase "SearchDaoImpl"**

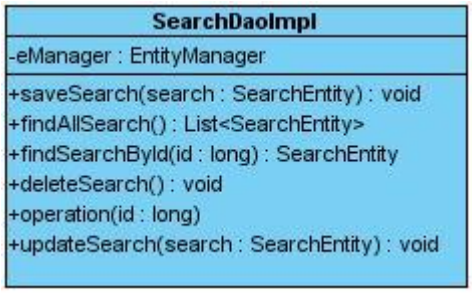


Figura 3.13: Clase SearchDaoImpl.

Objetivo: Acceder y realizar operaciones con los datos almacenados en la tabla "search".

- **Clase "LocalizationMapController"**

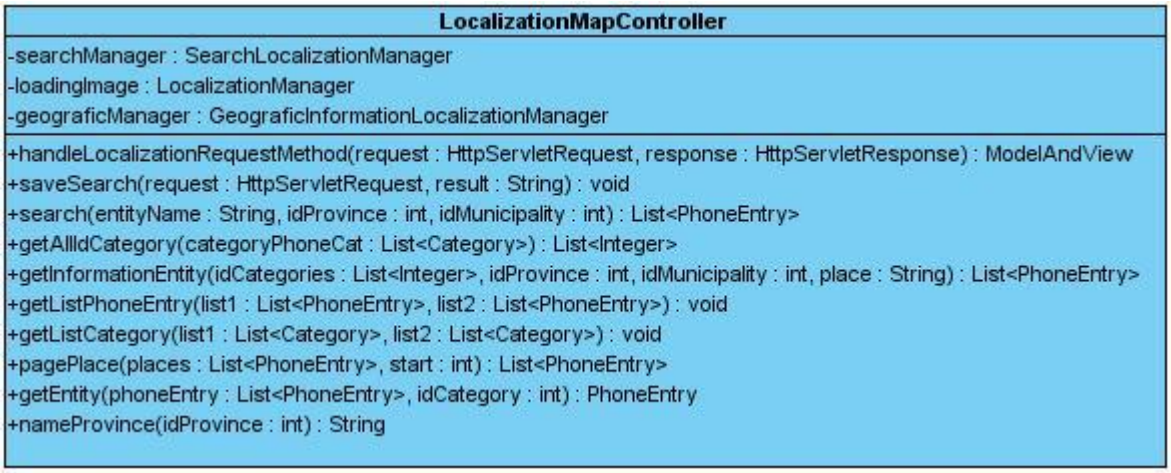


Figura 3.14: Clase LocalizationMapController.

Objetivo: Mostrar el mapa centrado en el lugar con información relevante del mismo.

- **Clase "ModificateLocalizationController"**

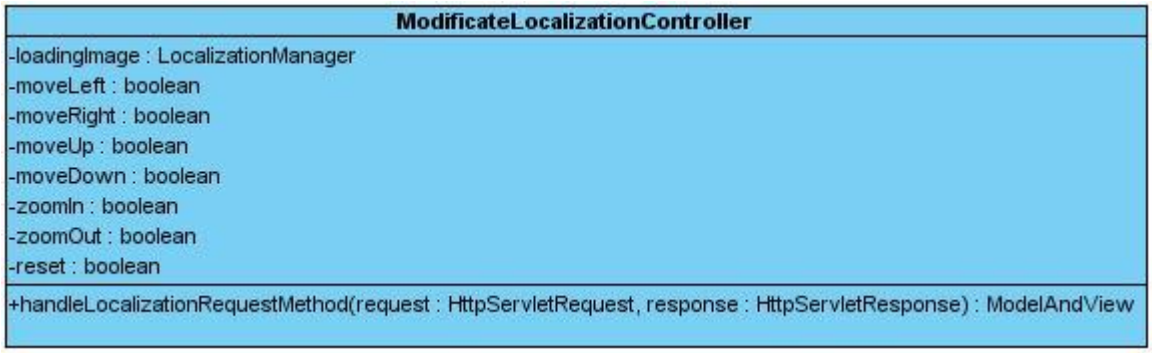


Figura 3.15: Clase ModificateLocalizationController.

Objetivo: Mostrar el mapa luego de modificaciones realizadas (resetear, ampliar, disminuir, mover hacia la izquierda, derecha, arriba y abajo).

- **Clase "SearchLocalizationController"**

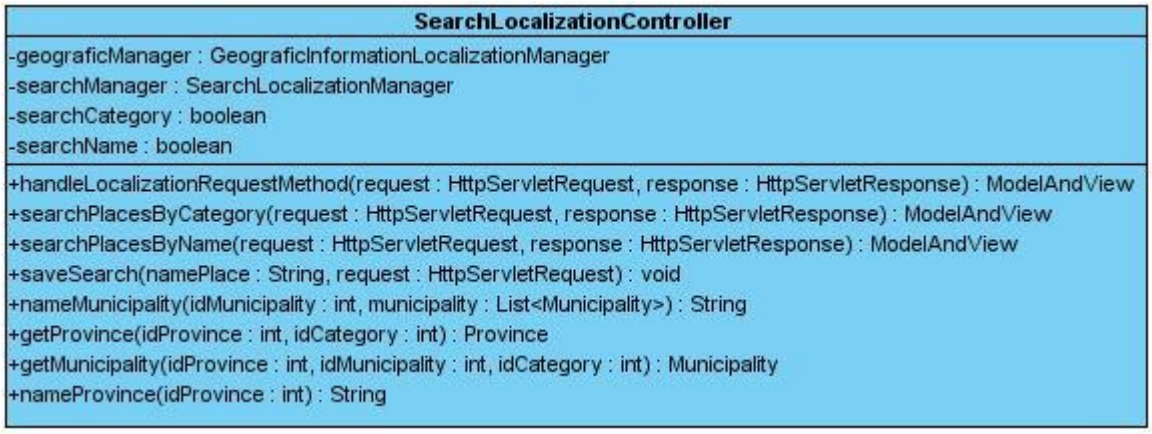


Figura 3.16: Clase SearchLocalizationController.

Objetivo: Mostrar los resultados de la búsqueda de un lugar.

3.5 Diagramas de Interacción

Un diagrama de interacción muestra una interacción, que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Estos diagramas se utilizan para modelar los aspectos dinámicos de un sistema. La mayoría de las veces, esto implica modelar instancias concretas o prototípicas de clases, interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento.

Diagrama de Secuencia

Un diagrama de secuencia destaca la ordenación temporal de los mensajes, se forma colocando en primer lugar los objetos que participan en la interacción en la parte superior del diagrama, a lo largo del eje X. Normalmente se coloca a la izquierda el objeto que inicia la interacción y los objetos subordinados a la derecha. A continuación se colocan los mensajes que estos objetos envían y reciben a lo largo del eje Y, en orden de sucesión en el tiempo, desde arriba hasta abajo. Esto ofrece al lector una señal visual clara del flujo de control a lo largo del tiempo.

En el ***Anexo4*** aparece el diagrama de Secuencia del Diseño.

3.6 Modelo de Datos

El modelo de datos está dado por las clases que intervienen en el almacenamiento de datos en la base de datos y las relaciones entre ellas.

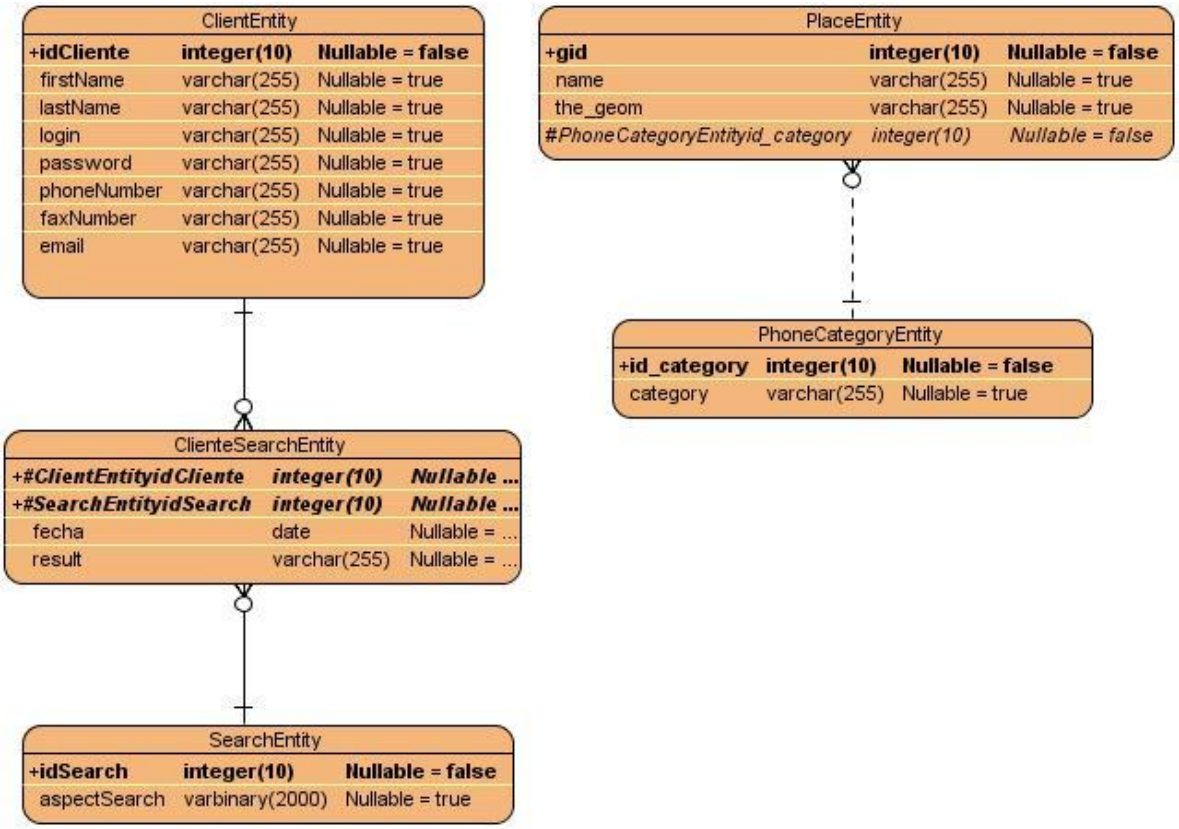


Figura 3.17: Modelo de Datos.

3.7 Conclusiones

En el presente capítulo se expusieron los principales artefactos del diseño del sistema propuesto como solución, en específico los diagramas de paquetes, clases del diseño y los diagramas de interacción. Además se describieron las clases y los patrones de diseño y el patrón arquitectónico a aplicar.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

En el presente capítulo se lleva a cabo la descripción de las principales tareas del flujo de implementación y de prueba, además de los artefactos que se generan en estos flujos. También se describirá cómo se relacionan y organizan los componentes de acuerdo con los mecanismos de estructuración y modularización en el entorno de implementación y en el lenguaje de programación utilizado. Se indicará la situación física de los componentes lógicos desarrollados en el Diagrama de Despliegue.

4.2 Implementación

El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. El modelo de implementación, principal artefacto que se genera en este flujo, establece la estructura de los elementos de implementación basándose en las responsabilidades asignadas de los subsistemas de implementación y su contenido.

4.2.1 Diagrama de Componentes

Los diagramas de componentes modelan la vista estática de un sistema. Se representa como un grafo de componentes software unidos por medio de relaciones de dependencia, pudiendo mostrarse las interfaces que estos soporten. Cada diagrama describe un apartado del sistema. En un diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

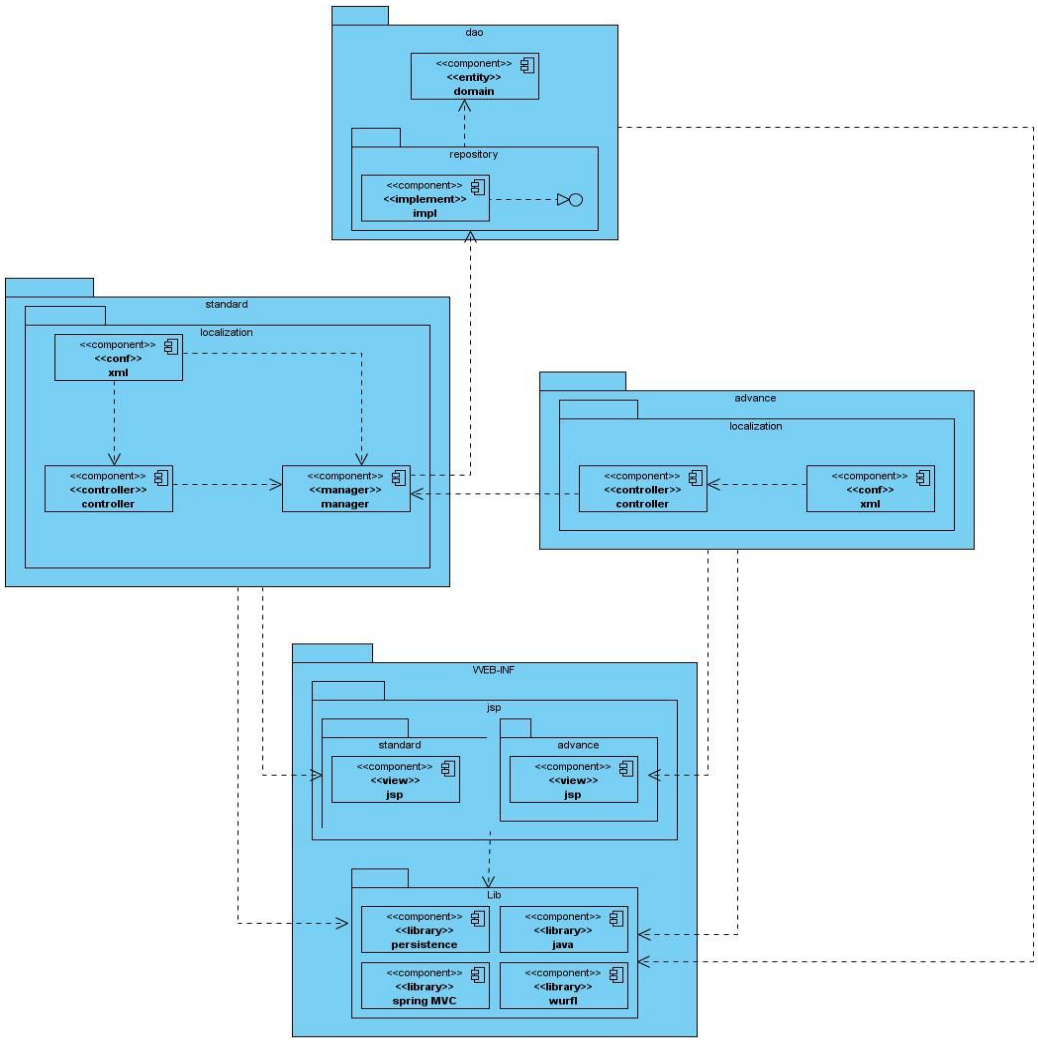


Figura 4.1: Diagrama de Componentes.

4.2.2 Descripción de Componentes

Paquete de componentes "dao": Está compuesto por los componentes domain e impl. El componente domain contiene las clases entidad del módulo. El componente impl contiene las clases implement que son las encargadas de implementar las clases interfaces haciendo uso de las clases entidad.

Paquetes de componentes "standard": Está compuesto por los componentes controller, manager y conf. El componente controller contiene una representación lógica de las clases controladoras que son las encargadas de gestionar los eventos de entrada, ya sea peticiones al modelo o a las vistas y haciendo uso de las clases manager contenidas en el componente manager, darle respuestas a estos eventos. El

componente conf contiene un archivo XML de configuración (localizationStandard-context.xml) que se encarga de inyectar objetos a una o varias clases, así como de asignarle una clase a determinada petición proveniente de las vistas.

Paquete de componentes "advance": Está compuesto por los componentes controller y conf. El componente controller contiene una representación lógica de las clases controladoras que son las encargadas de gestionar los eventos de entrada, ya sea peticiones al modelo o a las vistas y haciendo uso de las clases manager contenidas en el componente manager, que se encuentra en el componente standard, darle respuestas a estos eventos. El componente conf contiene un archivo XML de configuración (localizationAdvance-context.xml) que se encarga de inyectar objetos a una o varias clases, así como de asignarle una clase a determinada petición proveniente de las vistas.

Paquete de componentes "WEB-INF": Esta compuesto por los paquetes de componentes jsp y lib. El paquete de componentes jsp contiene los paquetes de componente standard y advance, los cuales agrupan a todas las vistas que se encargan de visualizar la información proveniente del modelo. El paquete de componentes lib contiene todas las librerías java que son utilizadas por todos los paquetes de componentes contenidos en el paquete de componentes wap.

4.2.3 Diagrama de Despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos.

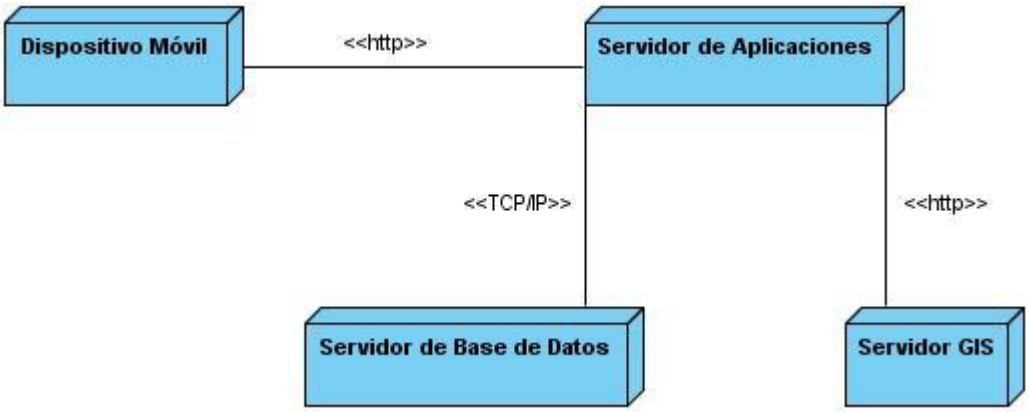


Figura 4.2: Diagrama de Despliegue.

4.2.4 Convenciones de archivos y paquetes

El proyecto que se crea en Eclipse Galileo se denomina “Localización”. Todas las clases, ficheros XML y otros ficheros del sistema están agrupados en una estructura de carpetas y paquetes.

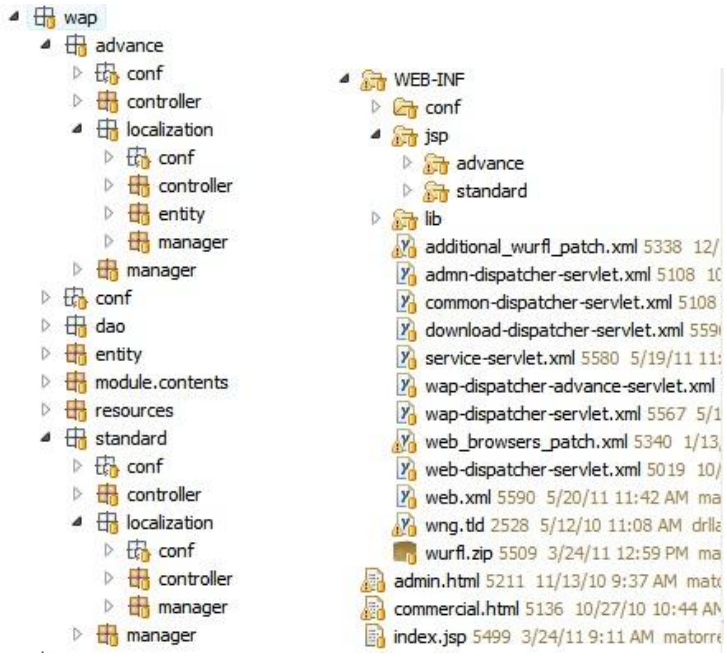


Figura 4.3: Convención de archivos y paquetes

La estructura es la siguiente:

- **manager:** Clases que realizan funciones específicas.
- **controller:** Controladoras de cada una de las vistas.
- **jsp:** Vistas del sistema.
- **dao:** Clases que realizan el acceso a datos.
- **conf:** Contiene un archivo XML de configuración.
- **advance:** Contiene los paquetes y clases del módulo avanzado.
- **standard:** Contiene los paquetes y clases del módulo estándar.
- **localization:** Contiene los paquetes y clases del módulo localización.

4.2.5 Estándar de Codificación

Las convenciones de código son importantes para los programadores por un gran número de razones:

- El 80% del coste del código de un programa va a su mantenimiento.
- Las convenciones de código mejoran la lectura del software, permitiendo entender el código nuevo más rápido y a fondo.
- Si se distribuye el código fuente como un producto, necesita estar bien hecho y presentable.

Para la implementación del sistema se utilizó el estándar de codificación definido por la UCI para el lenguaje de programación Java. Este estándar puede encontrarse en el ***Anexo5***.

4.3 Prueba

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos específicos, los resultados son observados, registrados y se realiza una evaluación de algún aspecto del sistema o componente. La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

4.3.1 Pruebas de Caja Negra

Las pruebas de caja negra se llevaron a cabo sobre la interfaz del software. Los casos de prueba demostraron que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Estas pruebas examinaron algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. Se centraron principalmente en los requisitos funcionales del software.

Los casos de prueba realizados a la aplicación y los resultados obtenidos se encuentran en el ***Anexo6***.

4.3.2 Pruebas de Caja Blanca

La prueba de caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba [28].

Las pruebas de caja blanca intentan garantizar que:

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Se utilizan las decisiones en su parte verdadera y en su parte falsa.
- Se ejecuten todos los bucles en sus límites.
- Se utilizan todas las estructuras de datos internas.

Las pruebas de caja blanca se realizaron utilizando la página web W3C mobileOK Checker que permite comprobar la compatibilidad del sistema con los estándares de programación definidos para aplicaciones WAP.

4.3.3 Resultados de las pruebas

Una vez aplicados los casos de prueba de caja negra se obtuvieron las siguientes no conformidades:

Total de Casos de Prueba	No Conformidades	Significativas	NC Resueltas
6	1	1	1

De los casos de pruebas aplicados al sistema se obtuvo una sola No Conformidad de tipo Significativa, a la cual se le dio solución, comprobando el correcto funcionamiento de la funcionalidad en una 2da iteración.

Las pruebas de caja blanca aplicadas al código de la aplicación mostraron en una primera iteración una compatibilidad con los estándares de programación definidos para aplicaciones WAP, de 49% para la vista avanzada y 53% para la vista estándar, ver Figura 4.4 y Figura 4.5.

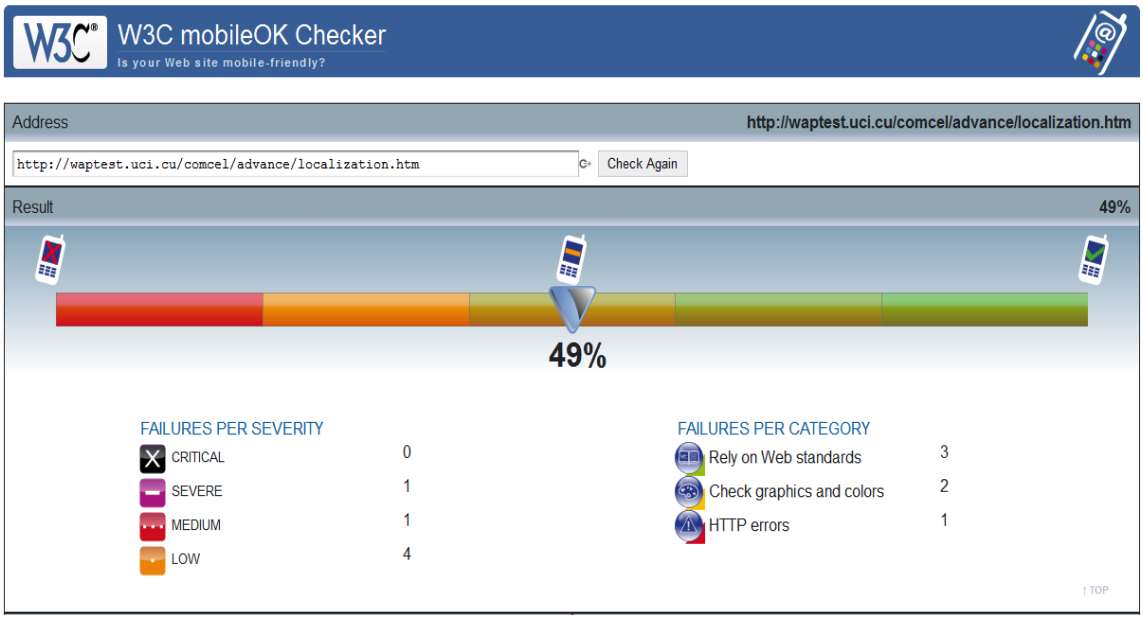


Figura 4.4: Prueba de caja blanca, primera iteración, vista advance.

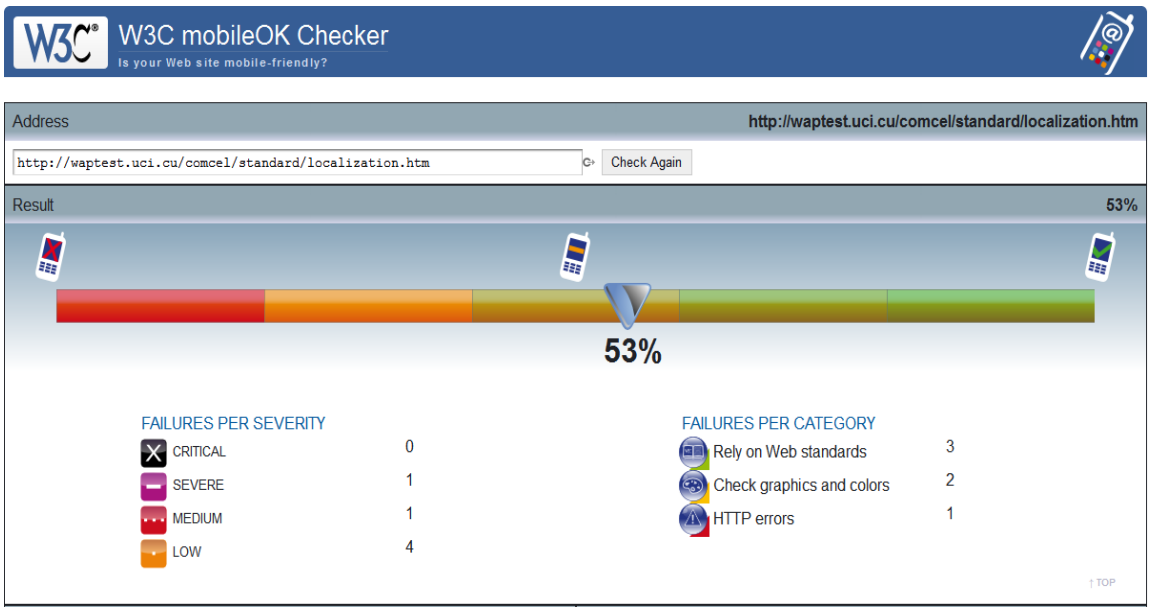


Figura 4.5: Prueba de caja blanca, primera iteración, vista standard.

Luego de una 2da iteración de pruebas de caja blanca se obtuvo un 95% para la vista estándar y un 90% para la vista avanzada. Ver Figura 4.6 y 4.7.

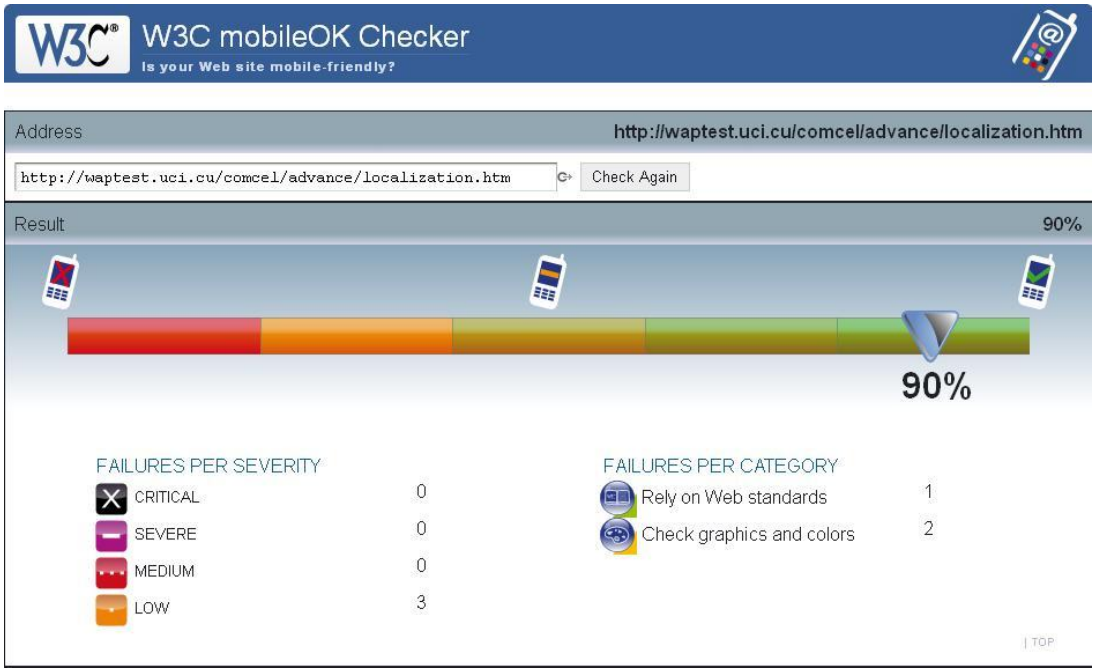


Figura 4.6: Prueba de caja blanca, segunda iteración, vista advance.

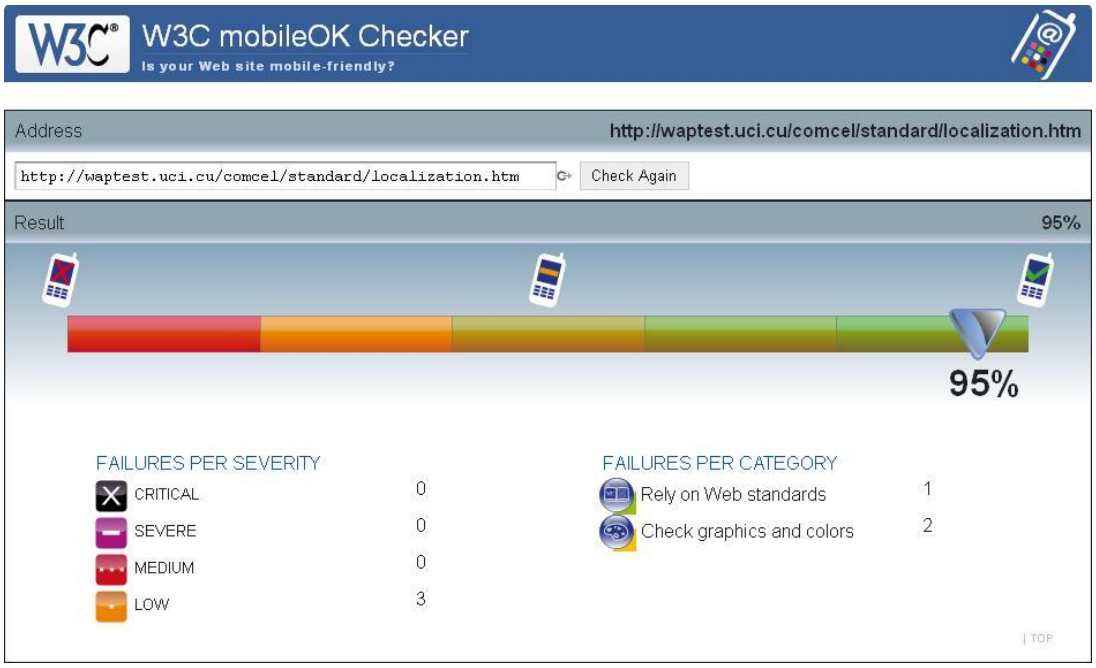


Figura 4.7: Prueba de caja blanca, segunda iteración, vista standard.

4.4 Conclusiones

En el presente capítulo se confeccionaron los diagramas de componentes y de despliegue de la aplicación, basados en la implementación de la misma. Se realizaron además las pruebas necesarias para comprobar el correcto funcionamiento del servicio de Orientación y Navegación, probando cada una de las funcionalidades implementadas.

CAPÍTULO 5: ESTUDIO DE FACTIBILIDAD

5.1 Introducción

En el presente capítulo abordará un análisis de la factibilidad de la herramienta, a través del método de estimación Puntos por Caso de Uso, el cual permitirá saber el costo y tiempo de desarrollo aproximado de la aplicación.

5.2 Método de Estimación

El método de Puntos en Casos de Uso es un enfoque bien documentado para estimar las actividades de desarrollo de software 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. El método utiliza los actores y casos de uso identificados para calcular el esfuerzo que costará desarrollarlos [28].

Cálculo de Puntos de Caso de Uso sin Ajustar

El primer paso del Método de Estimación Puntos por Caso de Uso es el cálculo de los puntos de caso de uso sin ajustar, y se calcula de la siguiente forma:

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.

- **Cálculo de UAW**

El **UAW** consiste en la evaluación de la complejidad de los actores con los que tendrá que interactuar el sistema. Este puntaje se calcula determinando si cada actor es una persona u otro sistema, además evalúa la forma en la que este interactúa con el caso de uso, y la cantidad de actores de cada tipo [28].

La fórmula sería: **UAW** = Sum (cantidadDeUnTipoDeActor*Factor)

Tipo de Actor	Descripción	Factor	Actores	Total
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación	1		
Medio	Otro sistema interactuando a través de un protocolo o una persona interactuando a través de una interfaz en modo texto.	2	1	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3	1	3
Total				**5**

Tabla 5.1: Factor de Peso de Actores

Entonces el **UAW** seria 5.

- **Cálculo de UUCW**

Este valor se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo, donde una transacción se entiende como una secuencia de actividades atómica, es decir, se efectúa la secuencia de actividades completa, o no se efectúa ninguna de las actividades de la secuencia [28].

Tipo de CU	Descripción	Peso	Cantidad de CU	Total
Simple	El caso de uso tiene de 1 a 3 transacciones.	5	6	30
Medio	El caso de uso tiene de 4 a 7 transacciones.	10	1	10
Complejo	El caso de uso tiene más de 8 transacciones.	15	0	0
Total				**40**

Tabla 5.2: Factor de Peso de Casos de Uso

Entonces el **UUCW** sería 40

Sustituyendo:

UUCP = UAW + UUCW

UUCP = 5 + 40

UUCP = 45

Cálculo de Puntos de Caso de Uso Ajustado

Una vez que se tienen los Puntos de Casos de Uso sin ajustar, se debe ajustar éste valor mediante la siguiente ecuación:

*UCP = UUCP * TCF * EF*

Donde

UCP: Puntos de Casos de Uso Ajustados

UUCP: Puntos de Casos de Uso sin Ajustar

TCF: Factor de Complejidad Técnica

EF: Factor de Ambiente

- **Cálculo de TCF**

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante [28]. Esto se calcula mediante la ecuación:

*TCF = 0.6 + 0.01 * Σ (pesoi * valor asignadoi)*

Factor	Descripción	Peso	Valor Asignado	Total
T1	Sistema distribuido.	2	3	6
T2	Objetivos del performance o tiempo de respuesta.	1	4	4
T3	Eficiencia del usuario.	1	2	2
T4	Procesamiento interno complejo.	1	4	4

T5	El código debe ser reutilizable.	1	0	0
T6	Facilidad de instalación.	0.5	5	2.5
T7	Facilidad de uso.	0.5	4	2
T8	Portabilidad.	2	5	10
T9	Facilidad de Cambio.	1	3	3
T10	Concurrencia.	1	2	2
T11	Incluye objetivos especiales de seguridad.	1	2	2
T12	Provee acceso directo a terceras partes.	1	1	1
T13	Se requieren facilidades especiales de entrenamiento a usuarios.	1	1	1
Total				**39.5**

Tabla 5.3: Factor de Complejidad Técnica

TCF = 0.6 + 0.01 * Σ (pesoi * valor asignadoi)

TCF = 0.6 + 0.01 * 39.5

TCF = 0.99

- **Cálculo de EF**

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos factores son los que se contemplan en el cálculo del Factor de ambiente. El cálculo del mismo es similar al cálculo del Factor de complejidad técnica, es decir, se trata de un conjunto de factores que se cuantifican con valores de 0 a 5 [28]. Se calcula mediante la siguiente ecuación:

EF =1.4 - 0.03 x Σ (pesoi x valor asignadoi)

Factor	Descripción	Peso	Valor	Total
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	5	7.5

E2	Experiencia en la aplicación.	0.5	3	1.5
E3	Experiencia en la orientación a objeto.	1	4	4
E4	Capacidad del analista líder.	0.5	3	1.5
E5	Motivación.	1	5	5
E6	Estabilidad de requerimientos.	2	3	6
E7	Personal Part–Time.	-1	1	-1
E8	Dificultad del lenguaje de programación.	-1	4	-4
Total				**20.5**

Tabla 5.4: Factor Ambiente

EF =1.4 - 0.03 * Σ (pesoi x valor asignadoi)

EF = 1.4 – 0.03 * 20.5

EF = 0.78

Sustituyendo

UCP = UUCP * TCF * EF

UCP = 45 * 0.99 * 0.78

UCP = 34.74

Estimación de Esfuerzo

Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por debajo del valor medio (3), para los factores E1 a E6. Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por encima del valor medio (3), para los factores E7 y E8 [28].

Si el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 20 horas-hombre. Si el total es 3 o 4, se utiliza el factor de conversión 28 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 28 horas-hombre. Si el total es mayor o igual que 5, se recomienda efectuar cambios en el proyecto, ya que se considera que el riesgo de fracaso del mismo es demasiado alto [28].

El esfuerzo en horas/ hombre viene dado por: *E = UCP * CF*

Donde:

E: Esfuerzo estimado en horas-hombre.

UCP: Puntos de Casos de Uso ajustados.

CF: Factor de conversión.

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

Total EF = 0 + 1

Total EF = 1

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

Sustituyendo

E = UCP * CF

E = 34.74 * 20

E = 694.8

Finalmente, para una estimación más completa de la duración total del proyecto, hay que agregar a la estimación del esfuerzo obtenida por los Puntos de Casos de Uso, las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo de software. Para ello se puede tener en cuenta el siguiente criterio, que estadísticamente se considera aceptable. El criterio plantea la distribución del esfuerzo entre las diferentes actividades de un proyecto, según la siguiente aproximación:

Actividad	Porcentaje	Horas/hombre
Análisis	10.00%	173.7 horas/hombre
Diseño	20.00%	347.4 horas/hombre
Implementación	40.00%	694.8 horas/hombre
Prueba	15.00%	260.55 horas/hombre

Sobrecarga	15.00%	260.55 horas/hombre
Total	100 %	1737 horas/hombre

Tabla 5.5: Distribución de Esfuerzo

Tomando como entrada la estimación de tiempo calculada a partir de los Puntos de Casos de Uso, se pueden calcular las demás estimaciones para obtener la duración total del proyecto [28].

El Esfuerzo Total (**ET**) sería 1737 horas-hombre, si estimamos teniendo en cuenta que un mes tiene 192 horas laborables (8 horas diarias * 24 días al mes) entonces el Esfuerzo Total en mes-hombre sería 9.04 mes-hombre.

Estimación de Costo

*Costo = CHM * ET / CH*

Donde:

CHM: Costo Hombre – Mes.

ET: Esfuerzo Total.

CH: Cantidad de hombres.

La cantidad de hombres que trabajan en el proyecto es 2, y tienen un salario de $100.00 mensuales. Por tanto:

CHM = CH * Salario Promedio

CHM = 2 * 100

CHM = 200

Costo = CHM * ET / CH

Costo = 200 * 9.04 / 2

Costo = $ 904

Tiempo de Desarrollo del Proyecto

En el desarrollo de la aplicación Servicio Basado en Localización para la plataforma Comcel participan 2 personas, y el tiempo que emplea una de ellas en realizar el proyecto completo es de 9.04 meses. Por tanto, el tiempo que demorarían 2 personas en realizar la aplicación seria:

Tiempo = ET / CH

Tiempo = 9.04 / 2

Tiempo = 4.7 ~ 5

5.3 Conclusiones

En el desarrollo del capítulo se realizó un análisis de los casos de uso del proyecto, para desarrollar un estudio de estimación y factibilidad de la herramienta. De acuerdo al método aplicado, se determino que el costo aproximado de la aplicación será de $904 y el tiempo de duración abarcara casi 5 meses.

CONCLUSIONES GENERALES

Se realizó un estudio del arte para conocer el estado actual de sistemas existentes en el mundo como Google Maps y Ovi Maps de Nokia que ofrecen LBS, arrojando como conclusión la necesidad de desarrollar un portal web para dispositivos móviles que brinde un servicio de Orientación y Navegación para la plataforma Comcel.

Se desarrolló una aplicación capaz de ofrecer orientación a los usuarios mediante búsquedas simples o filtradas por categorías de lugares, brindando información relevante de los mismos y conociendo la ruta existente entre dos lugares en dependencia del medio de transporte.

La aplicación de la metodología de desarrollo (RUP) permitió la construcción del sistema con calidad en el tiempo establecido cumpliendo con las necesidades del cliente.

Las pruebas de Caja Blanca realizadas utilizando la página web W3C mobileOK Checker al código de la aplicación y las pruebas de Caja Negra realizadas a la interfaz del sistema permitieron comprobar su correcto funcionamiento, dando cumplimiento al objetivo general de este trabajo.

RECOMENDACIONES

Cuando se realiza una búsqueda se muestra, además del lugar representado en el mapa, información relevante del lugar. Esta información puede o no mostrarse, esto se debe a que no se encuentra almacenada información para el nombre de ese lugar en la base de datos (páginas amarillas), dicho lugar es obtenido de otra base de datos (GIS). Se recomienda crear una sola fuente de información.

Los mapas que se muestran en el sistema son obtenidos del servidor de mapas GeoServer. Se recomienda crear una solución compatible con otros servidores como MapSever.

Para realizar una búsqueda ya sea simple o filtrada por categorías se debe seleccionar la provincia y el municipio perteneciente a la misma. Se recomienda integrarle al sistema un método de localización.

REFERENCIAS BILIOGRÁFICAS

1. **Bernardos, A. M.; Besada, J. A.; Casar, J. R.** "Tecnologías de localización: fundamentos y aplicaciones". Tecnologías y Servicios para la Sociedad de la Información, vol. I, no. 2 (enero 2005): 14-44.
2. **Chegung, K.W.: So, H. C.** "A Multidimensional Scaling Framework for Mobile Location Using Time-of-Arrival Measurements". IEEE Transactions on Signal Processing, vol. 53, no. 2 (february/2005): 460-470.
3. **Mao, G.; Fidan, B.; Anderson, B.** "Wireless Sensor Network Localization Techniques". The International Journal of Computer and Telecomunications Networking, vol. I, no. 2 (2007): 2529-2553.
4. **SoloSki,** El sistema de posicionamiento vía Satélite y los receptores GPS, [En línea] 2010 http://www.soloski.net/ficha_bricolaje.asp?Id=34.
5. **Aldrette Malacara, Rafael,** Protocolo WAP y su aplicación a la tecnología celular, Capítulo 2 Análisis de Herramientas y tecnologías, Universidad de las Américas Pueblas, [En línea] 2005 http://catarina.udlap.mx/u_dl_a/tales/documentos/msp/aldrette_m_r/capitulo2.pdf
6. **Domínguez Bravo, Javier,** Breve Introducción a la Cartografía y a los Sistemas de Información Geográfica (SIG), [En línea] 2000 http://www.preval.info/programa/wp-content/uploads/2008/09/itc_cartografia_sig.pdf
7. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** El Proceso Unificado de Desarrollo de Software. Addison Wesley 2000. Capítulo 1.
8. **Larman, Craig.** UML y Patrones. La Habana: Editorial Félix Varela, 2004.
9. **Colectivo de Autores.** Visual Paradigm for UML User´s Guide. 2007.
10. **Java.** Breve historia de la tecnología Java. [En línea] http://www.java.com/es/about/
11. **Cruz Martínez-Lacaci, Fernando.** El código abierto en el ámbito empresarial. [En línea] 2007. http://www.mkm-pi.com/mkmpi.php?article56.
12. **Qualitrain.** ¿Por qué seleccionar java como tecnología de desarrollo? Qualitrain México. [En línea] Empresa especializada en capacitación., 2009. http://www.qualitrain.com.mx/Blog.html.

13. **Java.** [En línea] 2010. http://www.java.com/es/about/.

14. **Foundation, E.** Eclipse. [En línea] 2009. http://www.eclipse.org/.

15. **Colectivo de Autores.** Help - Eclipse SDK. eclipse. [En línea] 2008. http://help.eclipse.org/galileo/index.jsp.

16. **Sánchez Rico, Mario Alfredo,** Sistema de administración y control de renta de películas y libros vía web utilizando Spring, Capítulo 3 Spring, un framework de aplicación, [En línea] 2006 http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo3.pdf

17. **Apache.** Tomcat – Introducción. [En línea]. http://www.programacion.com/tutorial/tomcatintro/1/#1_intro.

18. **Quiñones A, Ernesto,** Introducción a Postgresql. [En línea] 2009 http://www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf

19. **Oracle and/or its affiliates,** Introduction to the Java Persistence API. [En línea] 2010 http://download.oracle.com/javaee/5/tutorial/doc/bnbpz.html.

20. **Collins-Sussman, Ben, W. Fitzpatrick, Brian y Pilato, C. Michael**. Control de versiones con Subversion. [En línea]. http://svnbook.red-bean.com/nightly/es/index.html.

21. **Pressman,** Ingeniería del Diseño, Capítulo 9, pág. 270

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

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

24. **Visconti, Marcello y Astudillo, Hernán,** Fundamentos de Ingeniería de Software, Departamento de Informática Universidad Técnica Federico Santa María, pág. 6-23.

25. **Pressman,** Ingeniería del Diseño, Capítulo 9, pág. 260

26. **Lieberherr, K,** “Demeter: Aspect-Oriented Programming”, [En línea] 2003 http://www.ccs.neu.edu/home/lieber/LoD.html

27. **Facultad de informática - Universidad Politécnica de Madrid,** Patrones del Gang of Four, Unidad Docente de Ingeniería de Software, pág. 51

28. **Técnicas de Prueba,** Prueba de la Caja Blanca, [En línea] http://indalog.ual.es/mtorres/LP/Prueba.pdf

29. **Mario Peralta. 2008.** Estimación del esfuerzo basada en casos de Uso. [En línea] http://pisuerga.inf.ubu.es/rcobos/anis/estimacion-del-esfuerzo-basada-en-casos-de-usos.pdf

BIBLIOGRAFÍA

GLOSARIO

AOP: Presenta una estructura simplificada para el desarrollo y utilización de aspectos (módulos multiple object crosscutting).

API: Es el conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

CGI: Es una importante tecnología de la World Wide Web que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa.

CVS: El CVS es un modo muy elegante de administrar versiones de archivos que permite trabajar fácilmente a más de un desarrollador en el mismo proyecto.

GPL: Es una licencia creada por la Free Software Foundation en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

IDE: Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

IoC: Esta técnica promueve el bajo acoplamiento a partir de la Inyección de Dependencias (DI) entre los objetos (relaciones).

JDBC: Es un API para trabajar con bases de datos desde Java, independientemente de la base de datos a la que accedemos.

JDO: De la API estándar de Java es un modelo de abstracción basada en la interfaz de persistencia, desarrollado bajo los auspicios de la Comunidad Proceso de Java.

JEE: Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java con arquitectura de N niveles. La plataforma Java EE está definida por una especificación. Java EE es también considerada informalmente como un estándar debido a que los

suministradores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE.

JPA: Es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos, como sí pasaba con EJB2, y permitir usar objetos regulares.

JVM: Es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.

MMS: Es un estándar de mensajería que le permite a los teléfonos móviles enviar y recibir contenidos multimedia, incorporando sonido, video, fotos o cualquier otro contenido disponible en el futuro.

PHP: es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o' GTK

RUP: Es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Servlets: Aplicación sin interfaz gráfica que se ejecuta en un servidor de Internet, procesando información HTML previamente recogida por un navegador.

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

SVN: El Sistema de Control de Versiones es un sistema centralizado para compartir información. En su núcleo está un repositorio, que es un almacén central de datos.

UML: Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

WAP: Es un estándar abierto internacional para aplicaciones que utilizan las comunicaciones inalámbricas, p.ej. acceso a servicios de Internet desde un teléfono móvil.

WURFL: Es un fichero de configuración XML el cual contiene información acerca de características y capacidades de los dispositivos para una variedad de dispositivos móviles.

XML: Lenguaje de marcado utilizado para definir y utilizar las etiquetas necesarias para describir datos jerárquicos.