BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

TESIS

Aplicación Móvil Para la Información de Rutas y Movilización de Pasajeros en la Ciudad

Presenta:
Ángel Hernández Joaquín

Asesor:
Dr. Ivo Humberto Pineda Torres

Tesis presentada para obtener el tfulo de: Licenciatura en Ingeniería en Ciencias de la Computación

en

Facultad de Ciencias de la Computación

Febrero 2016

Declaración de Autoría

Yo, Ángel Hernández Joaquín, declaro que esta tesis titulada, «Aplicación Móvil Para la Información de Rutas y Movilización de Pasajeros en la Ciudad» y el trabajo presentado en el son míos. Confirmo que:

- Este trabajo fue realizado en su totalidad o principalmente, mientras se estaba en la candidatura para un grado de esta universidad.
- Cuando cualquier parte de esta tesis haya sido presentado para un grado o cualquier otra calificación en esta Universidad o cualquier otra institución, esto se ha manifestado claramente.
- Dónde he consultado la obra publicada de otros, esto es siempre claramente atribuido.
- Dónde he citado del trabajo de los demás, la fuente siempre se da. Con la excepción de estas citas, esta tesis es enteramente mi propio trabajo.
- He reconocido todas las principales fuentes de ayuda.
- Cuando la tesis se basa en el trabajo realizado por mí mismo en conjunto con otros, he dejado claro exactamente lo que se hizo por los demás y lo que he contribuido yo mismo.

Firma:

Fecha:

«Podemos enumerar muchas figuras importantes que no tenían talento pero que conquistaron su mérito y se trasformaron en genios, lo consiguieron superando dificultades.»

Friedrich Wilhelm Nietzsche

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Resumen

Facultad de Ciencias de la Computación

Licenciatura en Ingeniería en Ciencias de la Computación

Aplicación Móvil Para la Información de Rutas y Movilización de Pasajeros en la Ciudad

por Ángel Hernández Joaquín

El transporte es una de las necesidades para todo aquel que requiera trasladarse, ya sea para fines personales o para llegar a su centro de labores, en diversas ocasiones no se conoce ningún transporte, las mejores opciones respecto a costos y tiempo, lo cual puede significar perdidas en tiempo y dinero. Los dispositivos móviles gracias a sus funcionalidades cada vez toman mayor importancia para muchas de las actividades que se hacen todos los días, en especial los dispositivos Android, la plataforma predominante del mercado. Se desarrolló una aplicación móvil para la plataforma android que dada una ubicación y un destino muestra las posibles alternativas que el usuario puede elegir para llegar a su destino. El plan para desarrollar esta aplicación, tendrá cuatro etapas: recolección de datos, análisis y diseño, implementación y pruebas. Cada una de estas pretende cumplir los objetivos planteados en cada etapa. Con el uso del transporte público se busca reducir el congestionamiento vehicular, reducción de gases resultado de quema de combustibles fósiles, costos monetarios, e incluso generar algunos empleos como en la sección del mantenimiento

Agradecimientos

A mi asesor: Dr. Ivo Humberto Pineda Torres, por todo el apoyo y los conocimientos transmitidos.

A todos los amigos y colegas que han, de alguna manera, influenciado el desarrollo de esta tesís. Entre ellos puedo mencionar a Ernesto Lira Huerta, Salvador Sánchez Juárez y Karina Tentle Morales.

A todos los profesores incluido a mi supervisor quienes fueron artífices en mi formación como profesional.

Índice general

Índice de figuras

Índice de cuadros

Lista de Abreviaturas

ADT	**A**ndroid **D**evelopment **T**ools
AJAX	**A**synchronous **J**ava**S**cript and **X**ML
API	**A**pplication **P**rogramming **I**nterface
IDE	**I**ntegrated **D**evelopment **E**nvironment
CSS	**C**ascading **S**tyle **S**heets
HTML	**H**yper**T**ext **M**arkup **L**anguage
PHP	**P**HP **H**ypertext **P**re-processor
SDK	**S**oftware **D**evelopment **k**it
UML	**u**nified **M**odeling **L**anguage
XML	E**X**tensible **M**arkup **L**anguage

Dedicado a mis padres: Rodrigo Hernández Fernández y Tomasa Joaquín Alberto; Ustedes que en todo momento siempre han estado alentándome a terminar mis metas, apoyado en cada uno de mis pasos y me han hecho ser lo que soy ahora, porque sin ustedes, definitivamente no lo hubiese logrado...

Capítulo 1

Introducción

El hombre siempre ha tenido la necesidad de disponer de un sistema que le permita manejar gran cantidad de información con relativa rapidez así como efectuar cálculos a gran velocidad y de un modo mecánico que le libre de las arduas tareas asociadas con estas actividades.

A lo largo de este trabajo se tratan los antecedentes, justificación y objetivos de nuestro proyecto. Esto es, explicar por qué es conveniente llevarlo a cabo: sus beneficios, utilidad y que oportunidades ofrece. Debemos conocer el ámbito en el que trabajamos, para ello se investigo y busco todos aquellos desarrollos de última tecnología realizados, que han sido probados, acogidos y aceptados. Ademas del uso que tiene en el determinado sector de la población en que se va incursionar.

El desarrollo de nuestro proyecto implica tener herramientas y una metodología. Se describirán cada una de las herramientas de software empleadas para la implementación, sin dejar de un lado la metodología, es decir, nuestra serie de pasos que debemos seguir para alcanzar los objetivos planteados.

Siguiendo nuestra metodología, se procede a implementar el sistema. Se describe como se estructuro, los detalles mas relevantes que dan funcionalidad a los sistemas desarrollados, como lo son: análisis, algoritmos, configuraciones y protocolos de comunicación.

Una vez terminada la implementación se procede a realizar las pruebas y evaluar cada uno de los resultados obtenidos, ademas de realizar comparaciones en diferentes ambientes de prueba para apreciar el comportamiento de nuestra aplicación.

Por último se discute si nuestra solución cumple con los objetivos planteados al principio, el alcance y las limitaciones que tiene, las mejoras y el trabajo a futuro. Se expone el esfuerzo, las dificultades que se tuvieron a lo largo de este proyecto.

1.1. Antecedentes del problema

La necesidad de transportarse a través de la ciudad es algo trascendental para llevar a cabo la mayoría de nuestras actividades, por lo general es requerido recorrer distancias considerables y es necesario tomar una o más rutas de transporte público. En cuantiosas ocasiones no se tiene conocimiento sobre las distintas rutas para llegar o aproximarnos a nuestro destino, a veces por inseguridad o desconfianza de la gente no se les solicita información a las personas que circulen en donde nos encontremos. Otro factor importante es la cuestión económica; si se requiere hacer numerosos traslados y se cuenta con un presupuesto limitado se necesita hacer un uso óptimo de este. Un tercer factor es el tiempo, si se necesita llegar en el menor tiempo posible, se puede requerir tomar más de una ruta y transbordar a otra para llegar al destino.

El rápido desarrollo de la tecnología de los dispositivos móviles y de los servicios que la telefonía móvil ofrece facilita el desarrollo de nuevas aplicaciones más sofisticadas,

con la consiguiente demanda por parte de los consumidores. Tal es la importancia que están teniendo estos dispositivos en la sociedad actual, que las organizaciones proveedoras de formación se han visto en la necesidad de producir contenidos específicamente dirigidos a los dispositivos móviles puesto que se trata de un mercado con millones de usuarios y en constante crecimiento. Podemos mencionar aplicaciones enfocadas a la salud y el bienestar como lo son planificadores de rutinas de ejercicio, bitácoras para pacientes médicos, dietas, monitores peso, entre otros.

1.2. Justificacion

El beneficio que obtendremos al tener un sistema con el que podemos administrar toda la información de las rutas de una ciudad y ademas que se pueda extraer lo mas relevante para nosotros en un momento determinado, es de gran importancia para realizar nuestra movilización en la ciudad utilizando el transporte público. Otro punto interesante es aprovechar los recursos de computo que actualmente se tienen en nuestros dispositivos móviles, ya que actualmente tienen gran capacidad de almacenamiento, memoria y procesamiento.

1.3. Objetivos Generales y Específicos del Trabajo.

Objetivo General: Desarrollar una plataforma para administrar la información de rutas y una aplicación móvil para la plataforma Android que proporcione de sugerencias de transporte a los habitantes y visitantes de la ciudad de Puebla.
La aplicación móvil dada una ubicación y un destino debe mostrar las posibles alternativas que el usuario podrá elegir para llegar a su destino además de que esta información sea proporcionada sea verídica y oportuna. El usuario podrá guardar los puntos de su interés para utilizarlos posteriormente en la búsqueda de alternativas.

Objetivos Específicos del proyecto:

- Describir y documentar el proceso de desarrollo de ambas aplicaciones
- Fomentarel uso de aplicaciones móviles para la resolucion de problemas comunes
- Crear una interfaz amigable y comprensible para los usuarios.

Capítulo 2

Estado del Arte

2.1. Publicaciónes relacionadas al problema abordado

Los problemas relacionados con el transporte y el uso de modelos para dar solución a búsqueda a rutas no son nuevos, el trabajo titulado Un modelo de asignación probabilístico de tráfico de trayectos múltiples que evita la enumeración de rutas. El modelo de asignación probabilística presentado es un intento de eludir el problema de enumeración camino. Asigna viajes a todos los caminos razonables al mismo tiempo, de tal manera que el efecto resultante es idéntica a lo que se habría obtenido si hubiera sido asignado cada ruta viajes por separado bajo ciertos supuestos elección de probabilidad. En comparación con otras técnicas de propagación por trayectos múltiples, el modelo es teóricamente atractivo y computacionalmente muy eficiente. En el lado teórico que muestra algunas características muy deseables rara vez presentes en otras técnicas. Evalúa como equiprobables todos los caminos razonables de igual longitud. Como se hacen más largos caminos se vuelven menos probable que se utilicen. De esta manera, asignan viajes simultáneamente a todo un conjunto de caminos razonables. El modelo no examina explícitamente un camino, y no es en ningún sentido un "modelo de optimización". (Dial 1971)

Otros trabajos que vale la pena mencionar están mas enfocadas a la planificación de rutas, trabajos como La Mejor Planificación ruta para Sistemas de transporte público (Liu 2002), Modelo matemático de Algoritmos de Planificacióón de la mejor para sistemas Transporte Público (Chao-Lin Liu y Hsieh 2001), Diseño y optimización de rutas y frecuencias en el transporte colectivo urbano, modelos y algoritmos (Antonio Mauttone 2003) ya que tener una buena infraestructura puede hacer mas eficiente la carga y traslado de personas a través de una red de transporte, mas adelante se tocará el impacto sobre la distribución de rutas en el sistema de transporte.

Existe una propuesta de solución en tiempo real que ayuda a los usuarios a encontrar la optima combinación de rutas en un sistema de transporte masivo de pasajeros basado en buses, TransMilenio (TM) en Bogotá Colombia. La solución considera algunos datos extra para determinar la ruta más corta, tales como niveles de ocupación, horarios, tiempos estocásticos . El problema es modelado con un grafo con arcos no negativos y para determinar la ruta mas corta de una estación a otra desarrollaron una solución basada en el trabajo de Aproximación al Problema de Ruta más Corta con Trasbordo (Suárez-Sánchez 2005), tal que pueda ser implementada en un dispositivo móvil. (Sonia Vivas 2006)

En el trabajo de Aproximación al Problema de Ruta más corta con Trasbordo se propuso un algoritmo extendido que permite incorporar comportamiento dinámico en el

modelo. El autor abordó el problema de TransMilenio, teniendo en cuenta que los objetivos de la minimización del tiempo y de los cambios de autobús. Su algoritmo requiere una gráfo ponderado sin ciclos negativos, los nodos de inicio y destino y las restricciones de tiempo para los bordes, por lo que el grafo dinámico (los arcos son o no válidos en función del tiempo). Además se pide al usuario final para un número máximo de autobuses a cambiar que él está dispuesto a hacer, vamos a llamarlo M. El algoritmo funciona como el original, pero que almacena todas las soluciones posibles para cada nodo, y antes de almacenar una solución, revisa la restricciones de tiempo, por lo que la solución sólo será almacenada si todos los bordes están activos en el momento de la consulta. El algoritmo devuelve un conjunto de soluciones para cada nodo, a fin de encontrar las rutas de acceso es necesario (como en el algoritmo original) dar marcha atrás cada ruta que comienza en el nodo de destino. Con estos caminos, el autor utiliza el índice M para elegir los que son válidos, de acuerdo con el índice, y él devuelve la ruta con menos tiempo, el que es válido de acuerdo con el índice, y que tiene el menor número de cambios de autobús. (Suárez-Sánchez 2005)

El articulo llamado Enrutamiento rápido en grandes redes de transporte público utilizando patrones de Transferencia muestra cómo enrutar en grandes redes de transporte público (hasta medio billón de arcos) con tiempos medios de consulta de unos pocos milisegundos. Se tienen en cuenta factores como días de tráfico, caminar entre las estaciones, las consultas entre ubicaciones geográficas en lugar de una fuente y una estaciń de destino, y las funciones de costos de varios criterios. El algoritmo se basa en dos observaciones clave; (1) Muchos caminos más cortos comparten el mismo dibujo de transferencia, es decir, la secuencia de estaciones donde se produce un cambio de vehículo; (2) conexiones directas sin cambio de vehículo se pueden buscar rápidamente. Ademas de pre-computar los datos respectivos. (Hannah Bast 2010)

2.2. Soluciones relacionadas para nuestro problema

En el centro de descarga de aplicaciones (Google Play Store) de Android hay un vasto catálogo, en particular existe una aplicación llamada “Moovit” (disponible también para iOS) desarrollada inicialmente por la empresa Israelí Tranzmate Ltd. Su funcionamiento se basa en un “crowdsourcing” (colaboración abierta y distribuida). Los datos obtenidos se comparan con los presentes en los servicios oficiales de información y el resultado final determinara la mejor ruta, según el tráfico en tiempo real y tomando en cuenta diferentes opciones. El cruce de estos datos (los que proveen los usuarios y los servicios) dan más precisión a la aplicación y así poder brindar un mejor servicio en términos de tiempo de llegada, retrasos o disturbios en la vía. La aplicación está disponible para más de 32 países, incluido México, pero su alcance se limita solo a la ciudad de México, Guadalajara, Monterrey y Querétaro.

Otro conjunto importante son los sitios que permiten obtener información sobre el transporte público, para lo que nos concierne para la ciudad de Puebla el sitio “Ruta Directa” provee a través de Mapa de Google como llegar a tu destino, dando un punto de origen a un punto destino, la aplicación se encargará de mostrar el o los distintos autobuses que se pueden tomar, una vez seleccionado en el mapa se resaltará las rutas que toman los distintos autobuses. Otras opciones interesantes son la de “Rutas por punto” y “Puntos de Interés”, la primera dado un punto mostrara las rutas más cercanas en al menos unos cien metros, la siguiente como su nombre lo indica mostrara

los puntos de interés más cercanos en el mapa dado un punto en el mapa. La página dispone este servicio para diez ciudades de México, una de Argentina, una de Estados Unidos y una de Perú. Este servicio cuenta con aplicación móvil para Android.

Existe otra parecida a la alternativa anterior conocida como "Busca tu Ruta", esta tiene la particularidad que nos sugiere donde tomar el punto de partida y donde hacer nuestro descenso del autobús, es más de una ruta, se nos indicará donde hacer nuestro trasborde de autobús.

2.3. Situación en México y los teléfonos inteligentes

El Smartphone lo podemos definir como un dispositivo móvil que cuenta con características básicas como llamadas y mensajes de texto, además de capacidades avanzadas como un sistema operativo, acceso a Internet y uso de aplicaciones para distintos fines. Los Smartphone se han convertido en un factor común en la población mexicana, tal es así que a finales de 2014 la telefonía móvil alcanzó un total de 103.9 millones de líneas móviles a final de 2014, de las cuales 52.6 millones son a aquellos usuarios que poseen un teléfono inteligente (Smartphone), lo que es equivalente a 50.7 % del total. (Rolando Alamilla 2015a) La mayor parte del mercado está conformada por dispositivos Android (85.2 %), le sigue iOS con (5.3 %), Windows (4.9 %), Blackberry (2.7 %) y otros (1.9 %) (Rolando Alamilla 2015b) (Kantar-Worldpanel-ComTech 2012) De estos Google Inc. nos indica que la mayoría de las versiones activas de Android son superiores a la IceCream (4.0) y las versiones inferiores representan el apenas el 4 % de los dispositivos Android. (Google-Inc 2015a)

Capítulo 3

Análisis del Sistema

3.1. Infraestructura

3.1.1. Herramientas de Modelado

Se describirán cada una de las herramientas (lenguajes de programación, entornos de desarrollo y plataformas) que se usaron para el desarrollo del proyecto.

Star UML

Para crear diagramas del Lenguaje de Modelado Unificado (UML) se utilizo la herramienta Dia para la creación de los distintos diagramas. Dia es un programa para dibujar diagramas de estructura. Se puede utilizar para dibujar diferentes tipos de diagramas.

Justinmind Prototyper

Justinmind Prototyper es una herramienta de creación de prototipos de software. Se capacidades que se encuentran típicamente en herramientas como lo son arrastrar y colocación, redimensionamiento, formateo, la exportación e importación de widgets. Además, tiene características para crear anotaciones en los widgets y la definición de las interacciones, como enlaces, animaciones, vinculación condicional, cálculos, simulando controles de pestañas, mostrar u ocultar elementos y simulación de base de datos. El programa crea prototipos de alta calidad un paso antes de la primera versión de una aplicación móvil o página web. El prototipo se puede utilizar para mostrar carretes y propósitos de prueba. Justinmind Prototyper se puede utilizar para crear prototipos y simulaciones de software sin ningún tipo de codificación, lo que permite a los no programadores estar involucrados en el proyecto.

Puede ser utilizado para la pruebas remotas utilizando Justinmind Usernote, que permite a los usuarios navegar por el prototipo de forma remota. Los usuarios también pueden crear comentarios sobre elementos específicos en el prototipo con el fin de dar su opinión. La aplicación también es compatible con la integración de las herramientas de análisis como Google Analytics, WebTrends, ClickTale, UserTesting y otros para llevar a cabo pruebas remotas en tiempo real.

3.1.2. Entorno de Desarrollo: Eclipse IDE

Android Development Tools

Android Development Tools (ADT) es un plugin para Eclipse que incrementa las capacidades de Eclipse para que podamos configurar rápidamente nuevos proyectos para Android, crear interfaces de usuario, agregar paquetes basados en el API de Android FrameWork, debugear nuestra aplicación usando las herramientas del SDK de Android e incluso exportar archivos "*.apk" firmados (o sin firmar) con el fin de distribuir la aplicación.

Las aplicaciones que se realizan en este entorno se programan en Java, aunque es posible hacerlo con C y/o C++ haciendolo con el Kit de Desarrollo Nativo, este se introduce en las aplicaciones a través de JNI (Java Native Interface). El NDK hace que la ejecución de la aplicación sea en cierto modo más rápida, ya que pasará a ejecutarse directamente en el procesador y no es interpretado por una máquina virtual.

El NDK puede ser útil y beneficioso para ciertos tipos de aplicaciones, pero no para otros. El uso de código nativo en Android no se traduce en una notable mejora de rendimiento.

La versión de JDK (Java SE Development Kit) utilizada para la compilación es la 1.7.79, se puede obtener en la pagina oficial de Oracle.

PHP Development Tools

Para Eclipse existen diversos plugins o añadidos para proveer de nuevas utilidades al programa, enfocadas a diversos usos que los distintos tipos de programadores pueden necesitar. Unos de los añadidos de Eplipse para el desarrollo páginas web sería el módulo para programación en PHP, que en realidad está formado de varios componentes.

Entre las partes del programa están un editor de código PHP, con diversas ayudas a la vez que escribes, un editor de proyectos o archivos, que deben estar subidos en algún servidor web con soporte para PHP, y un sistema para hacer debug en PHP y depurar las aplicaciones web en PHP de una manera extremadamente sencilla.

La versión de PHP es la 5.5.31 y para MySQL, que soportará nuestra base de datos es la versión 5.0. Estas son las versiones mínimas requeridas para el sistema.

3.1.3. Google Maps

Para la definición de rutas necesitaremos una API que nos pueda proveer de mapas, El API de JavaScript de Google Maps permite insertar Google Maps en nuestras páginas web la cual viene con la mayoria de los elementos requeridos: utilidades para manipular mapas, añadir contenido al mapa mediante diversos servicios, entre ellos esta el servicio de direcciones, el cual nos proporcionará la información necesaria para trazar rutas.

La versión 3 del API de JavaScript de Google Maps (La más reciente y recomendada para su uso) es un servicio gratuito disponible para cualquier sitio web que sea gratuito para el consumidor.

Los sitios web lucrativos pueden generar hasta 25.000 cargas de mapas al día mediante la versión 3 del API de JavaScript de Google Maps. La carga de un mapa se

contabiliza cuando un mapa se inicializa en una página web. La interacción del usuario con el mapa una vez que este se haya cargado (por ejemplo, desplazarse por el mapa, ampliar el mapa o cambiar de mapa de carretera a mapa de satélite) no tiene ningún efecto sobre los límites de uso. Estos límites solo se aplican a los sitios que hayan excedido el límite durante más de 90 días consecutivos debido a los aumentos de trafico a corto plazo.

El uso del API de Google Maps requiere el acceso mediante una clave de API. Utilizar una clave de API nos permite controlar cómo utiliza tu aplicación el API de Google Maps y nos aseguramos de que Google puede ponerse en contacto contigo con respecto a tu aplicación si fuese necesario.

Para fines los fines de elaboración de este trabajo solo se requerirá la licencia gratuita, pero si en un momento dado llegase a aumentar considerablemente la demanda del sitio, será necesario obtener adquirir un limite adicional o una licencia para el API de Google Maps for Business. Ambas opciones permiten ampliar el limite de carga que podemos hacer.

3.1.4. Bootstrap

Bootstrap es el framework de HTML, CSS y JavaScript más popular para el desarrollo de sitios web responsivos. Esta hecho para hacer mas rápido y fácil el desarrollo web del front-end, para desarrolladores de todos los niveles, dispositivos de todas las formas y proyectos de todos los tamaños.

En diseño de software front-end es la parte del software dedicada a interactuar con los usuarios de la aplicación y el back-end es la encargada de procesar la entrada de datos del front-end. Esta separación ayuda a mantener las diferentes partes del sistema separadas.

Existen varias formas diferentes de empezar con Bootstrap, cada una orientada a un tipo de público en función de su nivel técnico. La forma más facil es descargar el código CSS y JavaScript compilado de su pagina oficial o de alguna plantilla personalizada.

El desarrollo del sitio web se utilizó la plantilla basada en Bootrap SPOT Theme, es una de las diferentes plantillas gratuitas que ofrece en `http://www.blacktie.co`

3.1.5. XAMPP

XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl. Este paquete de software nos ayudará a realizar el diseño del primer sistema del cual se estará conformado este trabajo.

3.2. Análisis de Requerimientos

Para cualquier desarrollo de un sistema de software es muy importante seguir alguna especificación que permita al desarrollador el tener una disciplina que haga todas las etapas del desarrollo del sistema, desde la información inicial de requerimientos, hasta las pruebas finales del sistema, estas deben ser lo mas coherentes y formales.

El desarrollo de este proyecto propone, al ser un instrumento que se pretende utilizar en el contexto de un problema real, debe seguir un proceso de análisis y diseño

que proporcione las bases bajo las cuales se va a desarrollar. En esta sección y en las siguientes se detallarán los procesos de ingeniería de software, análisis y diseño que implican el desarrollo de una aplicación de software. Se detallaran los procesos y principios de análisis y diseño del software que sustentan al proyecto.

La ingeniería de software es una disciplina o área de las ciencias de la computación que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelve problemas de todo tipo. Esta proporciona diversas métricas y metodologías que pueden usarse como especificaciones para todo lo referente a la administración del capital humano, ciclos de vida un proyecto de software, costos de un proyecto, y en si todo el aspecto administrativo que implica el desarrollar software. (Pressman 2005)

En general, todo proceso de ingeniería debe comenzar por contestar las siguientes preguntas: ¿Cuál es el problema a resolver?, ¿Cuáles son las características de la entidad que se utiliza para resolver el problema? ¿Cómo se realizará la entidad y la solución? ¿Cómo se creará la entidad? ¿Cómo va a probarse la entidad? y ¿Cómo se apoyará a la entidad cuando los usuarios finales soliciten correcciones y adaptaciones a la entidad? Para los fines que se desarrolla el software propuesto dentro de este proyecto, podemos contestar estas preguntas en una primera instancia desde un punto de vista global y sin considerar detalles específicos, de tal manera que se pueden establecer los siguientes puntos:

- Desarrollar una aplicación móvil que muestre las alternativas de rutas de transporte publico en la ciudad.
- Desarrollar un servicio que pueda proveer los datos de cada una de las rutas existentes.
- Desarrollo de una plataforma web que pueda ser utilizada para crear y modificar información para cada una de las rutas
- La aplicación principal debe tener características tales que se cumpla con el objetivo del proyecto, esto es, que la aplicación este orientada a los usuarios para que pueda ser aplicado al área del problema.
- El sistema se realizará con las herramientas descritas en la sección 3.1.1, tanto como para modelar y desarrollar.
- El sistema deberá probarse en un intervalo de tiempo adecuado y lo suficientemente amplió como para poder obtener retroalimentación por parte de los usuarios y hacer las correcciones pertinentes.
- Se deberá documentar adecuadamente para facilitar posibles procesos futuros tales como expansiones ó adaptaciones a las exigencias de los usuarios.

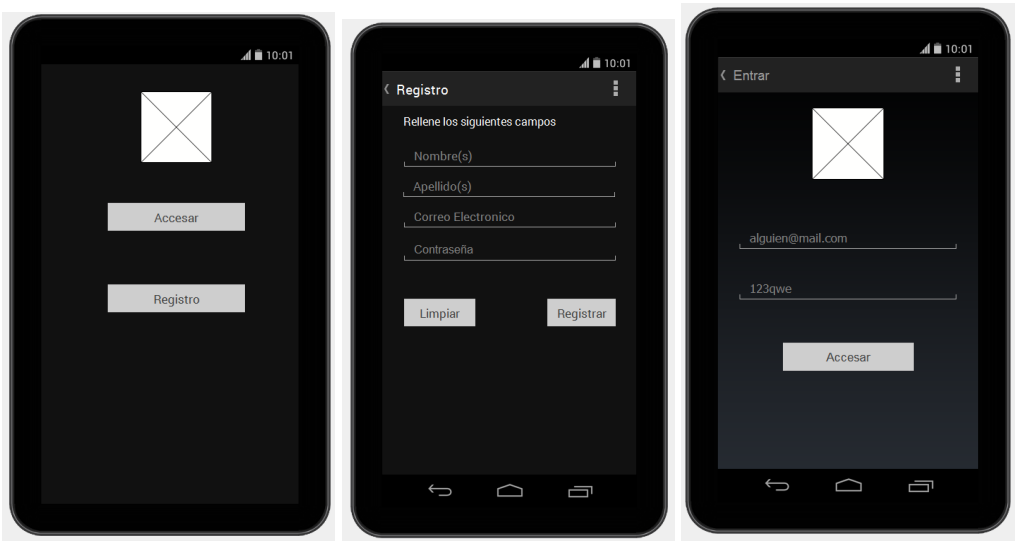
Para el desarrollo de esta solución, tanto web como móvil, se contemplaron cuatro etapas, en primer lugar tenemos el desarrollo de un prototipo para tener un panorama de la apariencia de nuestro sistema y a su vez obtener los datos de cada una de las rutas, los cuales serán base indispensable para resolver el problema. El modelado de la aplicación se modelara la interacción del usuario con la aplicación de un Lenguaje Unificado y con casos de uso. Se identificaran los casos necesarios, tales como: registro, autenticidad, captura, visualización de información. A partir de los casos de uso identificados se realizará un modelo de análisis en el cual se presentara en tres capas: capa de

interfaz con el usuario, capa lógica y de control, por último la capa de almacenamiento y comunicación; dentro de estas tres capas se modelaran nuestras clases, en las cuales se mencionaran los atributos y métodos que se tendrán. Posteriormente se realizara un diagrama de clases refinado, en este caso se refina lo que se haya hecho en el modelo de análisis. Una vez definido estos elementos se podrán implementar nuestro sistema, tomando en cuenta que ya se tendrá un modelo y será más fácil poder modificar o incrementar determinadas funciones.

3.3. Creación de un prototipo de la aplicación móvil

Con la herramienta para prototipos Justinmind se desarrollo un primer ejemplar para tener un panorama de los módulos y funcionalidades que tendrá la aplicación móvil. Esto con el fin de detectar posibles deficiencias en la interfaz. El modelo consta del modulo principal y opciones generales. La ventaja de la herramienta es que es posible (Registrando una cuenta en su sitio web) cargar nuestros prototipos al teléfono móvil. La herramienta puede exportar este proyecto para que podamos visualizarlo en cualquier navegador, la interfaz que se maneja es similar a la de un teléfono.

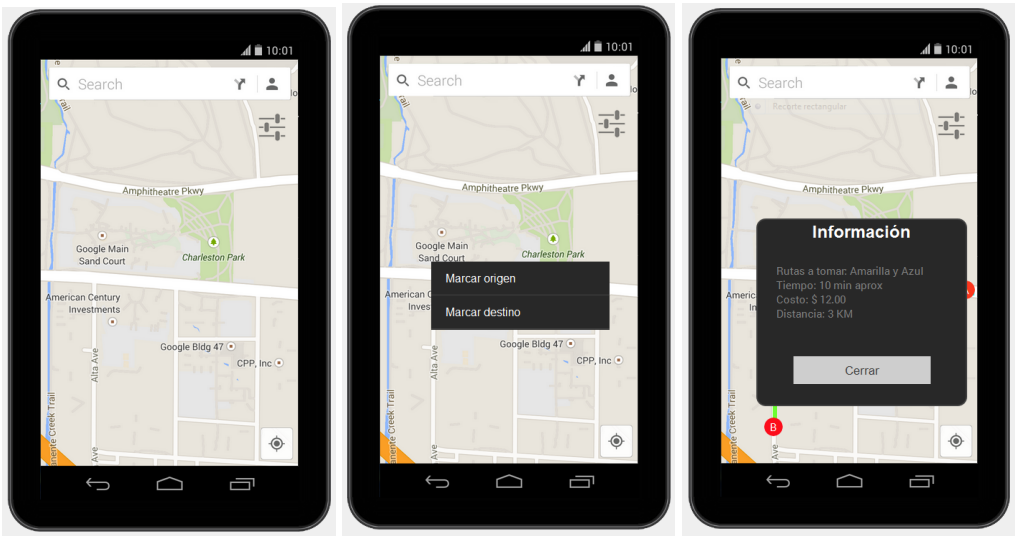
La realización del prototipo se efectuara utilizando colores neutros (Escala de grises) e imágenes ilustrativas.



(A) Inicio (B) Registro (C) Acceso

FIGURA 3.1: Módulos iniciales

El modulo de inicio del prototipo da un par de opciones: Registro y Acceso. El primero permite registrar una cuenta de usuario y en el segundo con tus datos podemos acceder a las funcionalidades del sistema.



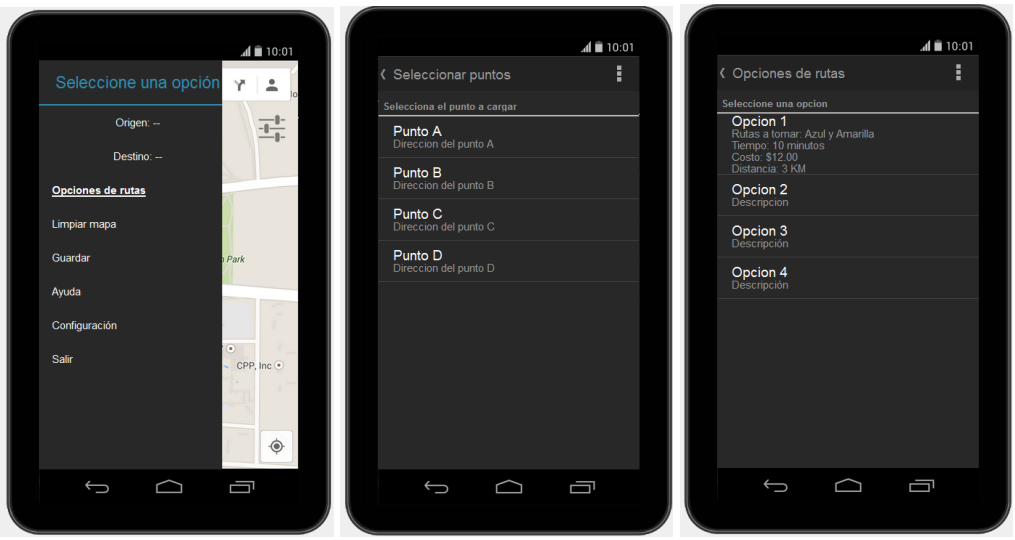
(A) Mapa Inicial

(B) Marcar Puntos

(C) Resultado

FIGURA 3.2: Funcionamiento de la búsqueda de rutas

El modulo principal esta compuesto por un mapa en el cual se marcarán dos puntos: el punto de origen, del cual partirá el usuario, el siguiente será el destino al cual usuario desea trasladarse. Una vez que estos dos pasos sean completados se determinará una solución aproximada a la optima. Como ayuda adicional se mostrara una ventana emergente que nos mostrará los detalles de la solución encontrada.



(A) Menú Lateral

(B) Seleccionar un punto frecuente

(C) Alternativas

FIGURA 3.3: Uso de las funcionalidades extra

Algunas de las funcionalidades que se consideraron son: la posibilidad de guardar puntos frecuentes y utilizarlos en diferentes momentos y la otra es ver las posibles alternativas encontradas para trasladarse. Guardar permitirá guardar el o los puntos que defina el usuario. Ayuda mostrará información sobre como utilizar la aplicación. En configuración podremos realizar diversos ajustes como lo son: el criterio de ordenamiento, actualización de la base de datos.

Algunas observaciones hechas a este prototipo fueron que se vio innecesario los módulos de registro y acceso dado que no se respaldan los datos en otro lugar que no sea el dispositivo con Android. En vez de esto se propuso colocar una guía informativa en la cual se explique el uso de la interfaz y de las funciones de la aplicación.

3.4. Arquitectura del Sistema

Como se mencionó anteriormente, se requieren tres módulos a desarrollar para que el sistema pueda operar. Para ello debemos partir de lo general a lo mas particular, por ello lo primero a hacer, teniendo en cuenta el prototipo realizado, se realizaran los casos de uso, posteriormente diagramas de componentes, secuencia y clases. En esta sección describiremos casos de uso, funcionalidades centrales del sistema final planteado, los casos de uso permiten descubrir y diseñar la arquitectura del sistema.

Para el diseño del sistema Web, se identifican como los casos de uso relevantes desde el punto de vista de la arquitectura, el sistema ha sido divido en tres partes (Una para cada tipo de usuario), esto con la finalidad de tener mayor seguridad y control de la informaciń que apreciara cada usuario. El sistema constara de los siguientes subsistemas:

- Registro

- Acceso
- Administración de mapas
- Administración de rutas
- Administración de usuarios

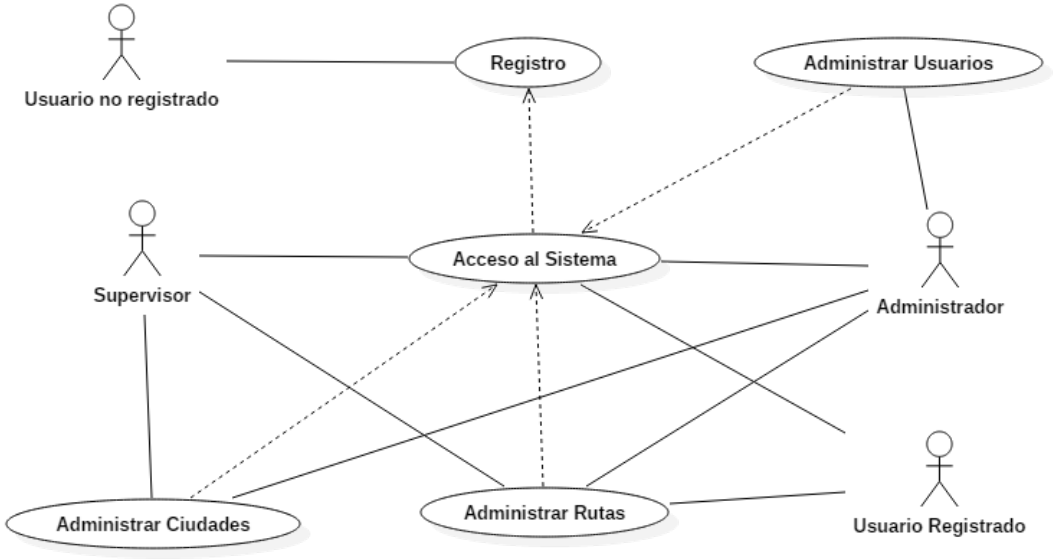


FIGURA 3.4: Casos de uso de la plataforma web

CUADRO 3.1: Descripción del caso de uso de registro

Nombre del Caso de Uso	Registro
Fecha	Septiembre 2014
Resumen	El usuario que quiera interactuar con el sistema deberá darse de alta
Curso Básico de Eventos	1. Entrar al sistema 2. Rellenar cada uno de los campos 3. Enviar datos para validar 4. Confirmar
Caminos Alternativos	Si existiese un problema con los datos introducidos, el sistema deberá notificar cuales son
Pre-condiciones	Acceder a la pagina del sistema que contiene al registro
Post-condiciones	Algún administrador deberá aprobar al usuario registrado para que pueda acceder al sistema

CUADRO 3.2: Descripción del Caso de Uso Acceso al Sistema

Nombre del Caso de Uso	Acceso al Sistema
Fecha	Septiembre 2014
Resumen	El usuario que este dado de alta en el sistema podra introducir sus datos para entrar e interactuar con el sistema
Curso Básico de Eventos	1. Entrar al sistema 2. Rellenar cada uno de los campos 3. Enviar datos para validar 4. Entrar al sistema
Caminos Alternativos	Si existiese un problema con los datos introducidos, el sistema deberá notificar cuales son
Pre-condiciones	1. Acceder al formulario de acceso 2. Estar habilitado para entrar al sistema
Post-condiciones	Se genera una bitácora de accesos al sistema

CUADRO 3.3: Descripción del Caso de Uso Administrar Usuarios

Nombre del Caso de Uso	Administrar Usuarios
Fecha	Septiembre 2014
Resumen	El usuario que esté autorizado podra modificar los datos de los usuarios, tales como nombre, apellidos, correo, contraseña y privilegios
Curso Básico de Eventos	1. Estar logueado como Administrador 2. Elegir el usuario a editar 3. Modificar la información requerida 4. Enviar cambios
Caminos Alternativos	Si existiese un problema con los datos introducidos, el sistema deberá notificar cuales son
Pre-condiciones	1. Tener permisos de Administrador 2. Estar habilitado para entrar al sistema
Post-condiciones	Se registra el ultimo usuario que hizo la modificación ademas de la fecha

CUADRO 3.4: Descripción Del Caso de Uso Administrar Ciudades

Nombre del Caso de Uso	Administrar Ciudades
Fecha	Septiembre 2014
Resumen	El usuario que esté autorizado podra ver, crear y modificar los centros de las ciudades en los cuales se podrán agregar rutas
Curso Básico de Eventos	1. Estar logueado como supervisor 2. Elegir una opción de la pagina 3. Agregar la información requerida 4. Enviar cambios
Caminos Alternativos	Si existiese un problema con los datos introducidos, el sistema deberá notificar cuales son
Pre-condiciones	1. Tener permisos de supervisor o admin 2. Estar habilitado para entrar al sistema
Post-condiciones	Se registra el ultimo usuario que hizo la modificación ademas de la fecha

CUADRO 3.5: Descripción del Caso de Uso Administrar Rutas

Nombre del Caso de Uso	Administrar Rutas
Fecha	Septiembre 2014
Resumen	El usuario que esté autorizado podrá ver, crear y modificar los trayectos de cada una de las rutas de cada ciudad
Curso Básico de Eventos	1. Estar logueado 2. Elegir una opción de la pagina 3. Agregar la información requerida 4 Describir el trayecto de la ruta en el mapa 5. Enviar cambios
Caminos Alternativos	Si existiese un problema con los datos introducidos, el sistema deberá notificar cuales son
Pre-condiciones	1. Estar habilitado para entrar al sistema
Post-condiciones	Se registra el ultimo usuario que hizo la modificación ademas de la fecha

El diseño de la Aplicación Móvil se enfoca como un cliente que consume información por parte de un servidor, pero con la particularidad que hace el procesamiento de manera local, esto con el fin de aprovechar los recursos de hardware que se cuentan en el teléfono.

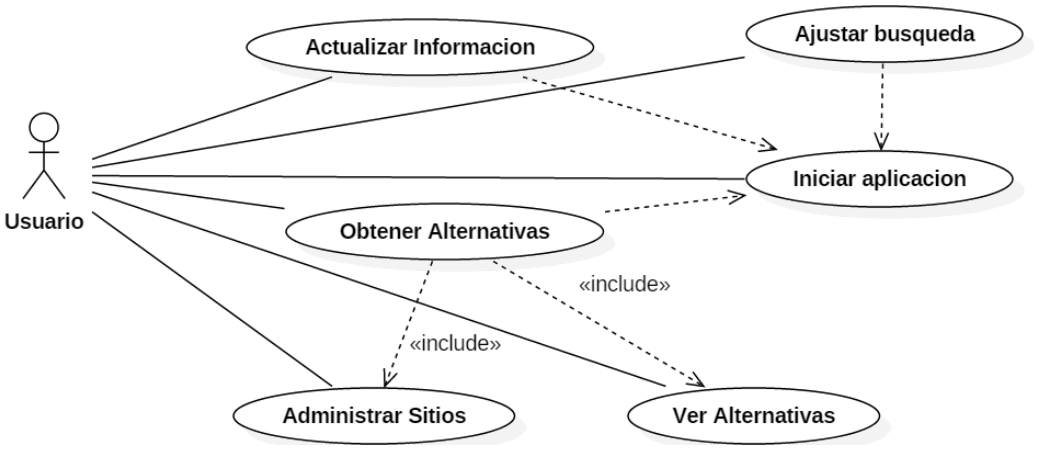


FIGURA 3.5: Casos de uso de la Aplicación Móvil

CUADRO 3.6: Descripción del Caso de Uso Iniciar Aplicación

Nombre del Caso de Uso	Iniciar Aplicacion
Fecha	Septiembre 2014
Resumen	El usuario iniciará la aplicación en su telefono y dependiendo si es la primera vez se mostrará una guia informativa, de lo contrario se redirigirá al inicio de la app
Curso Básico de Eventos	1. Instalar la aplicación 2. Deslizar los slides de la presentacion 3. Al proseguir se cargará la información 4. Esperar a que termine el proceso 5. Accesar al inicio de la app
Caminos Alternativos	Si ya ha sido iniciada la aplicación, automáticamente se nos redirigirá al inicio
Pre-condiciones	Instalar la app en un telefono con Android 2.3 o superior
Post-condiciones	La proxima ocasion que se inicie, se nos redirigirá al inicio de la app.

CUADRO 3.7: Descripción del Caso de Uso Actualizar Información

Nombre del Caso de Uso	Actualizar Aplicacion
Fecha	Septiembre 2014
Resumen	Una vez iniciada la aplicación y dentro de la seccion opciones se procede a actualizar la información de las rutas que hayan sido modificadas
Curso Básico de Eventos	1. Iniciar la aplicación 2. Ir al apartado de opciones 3. Hacer clic en Actualizar Información 4. Esperar a que termine el proceso 5. Cambios realizados
Caminos Alternativos	Si no existiese información nueva, no habrá cambio alguno en la base de datos
Pre-condiciones	Instalar y acceder a la aplicacion
Post-condiciones	La aplicación tendrá los datos mas recientes de las rutas de determinada ciudad

CUADRO 3.8: Descripción del Caso de Uso Ajustar Búsqueda

Nombre del Caso de Uso	Ajustar Búsqueda
Fecha	Septiembre 2014
Resumen	Una vez iniciada la aplicacion y dentro de la seccion opciones se procede a ajustar el tamaño del rango de busqueda, tanto del origen como del destino
Curso Básico de Eventos	1. Iniciar la aplicacion 2. Ir al apartado de opciones 3. Hacer clic en Ajuste del Punto 4. Ajustar el tamaño de busqueda 5. Guardar ambios realizados
Caminos Alternativos	Se puede cancelar la modificacion sin afectar el futuro funcionamiento
Pre-condiciones	Instalar y acceder a la aplicación
Post-condiciones	La aplicacion hara una busqueda mas extensa o reducida dependiendo de la amplitud que se le haya asignado

CUADRO 3.9: Descripción del Caso Obtener Alternativas

Nombre del Caso de Uso	Obtener Alternativas
Fecha	Septiembre 2014
Resumen	Una vez iniciada la aplicacion y con el mapa en la vista se procede a marcar un punto de origen y de destino, seguido del proceso de búsqueda de alternativas de rutas necesarias para llegar al destino
Curso Básico de Eventos	1. Iniciar la aplicacion 2. Mostrar el mapa 3. Marcar el par de puntos 4. Esperar a que el proceso finalice 5. Se notificará de la finalización
Caminos Alternativos	Si hacemos clic sobre un punto marcado podremos guardarlo y reutilizarlo en el futuro Los puntos que tengamos guardados podremos utilizarlos como origen y destino
Pre-condiciones	Instalar y acceder a la aplicación
Post-condiciones	La seccion del alternativas contendrá las rutas que nos llevarán de nuestro origen al destino indicado

CUADRO 3.10: Descripción del Caso Administrar Sitios

Nombre del Caso de Uso	Administrar Sitios
Fecha	Septiembre 2014
Resumen	Si se han registrado puntos, estos estarán en la sección Mis Sitios. Aquí se podrá realizar algún cambio, eliminar o usar para la búsqueda de alternativas
Curso Básico de Eventos	1. Iniciar la aplicacion 2. Ir a la sección Mis Sitios 3. Seleccionar un elemento 4. Elegir una opción
Caminos Alternativos	Si realizamos clic en usar como origen o destino se nos trasladará al mapa y se marcará el punto Si se desea editar saldrá un formulario en el cual podremos realizar las respectivas modificaciones Si se desea eliminar, se retirará el elemento de la lista de Mis Sitios
Pre-condiciones	Tener al menos un Sitio o punto registrado
Post-condiciones	Si se elige la opción usar, este podrá desencadenar la búsqueda de alternativas si faltase el origen o el destino por definir

CUADRO 3.11: Descripción del Caso de Uso Ver Alternativas

Nombre del Caso de Uso	Ver Alternativas
Fecha	Septiembre 2014
Resumen	Posterior al proceso de búsqueda de rutas, en la sección Alternativas podremos ver las alternativas encontradas, estas están ordenadas de manera ascendente, teniendo en cuenta el criterio de la distancia recorrida
Curso Básico de Eventos	1. Iniciar la aplicación 2. Ejecutar la búsqueda de alternativas 3. Ir a la sección Alternativas 4. Observar y seleccionar una opción
Caminos Alternativos	Sin variante
Pre-condiciones	Que se haya ejecutado el proceso de búsqueda de alternativas.
Post-condiciones	En el mapá se mostrará el trayecto de la ruta, el punto de abordaje, el punto de trasborde, si es requerido, y el punto de descenso

3.4.1. Vista Lógica

En este punto tan importante de nuestra arquitectura, se presenta las unidades lógicas correspondientes. Se comienza con el refinamiento conciso de la descomposición en subsistemas, los cuales representan cortes verticales al diseño del sistema. Cada subsistema consiste se refiere a agrupar diferentes funciones del sistema que se relacionan entre si y que es capaz de funcionar con un sistema en sí mismo. Por otra parte se explora la composición de cada uno de los subsistemas.

La descomposición en subsistemas organiza la arquitectura de forma que cada uno de estos sea cohesivo y capaz de interactuar entre ellos para lograr cumplir sus funciones respectivas, para ello realizamos el diagrama de tareas, en cual describe de forma concreta y específica cada proceso a realizar en nuestro sistema, el cual como se ha descrito anteriormente consta de tres partes esenciales. Ademas de que separaremos los módulos por capas como se aprecia en la figura 3.6, esto es para especializar cada segmento en una tarea en especifico, sera útil para un mantenimiento a futuro.

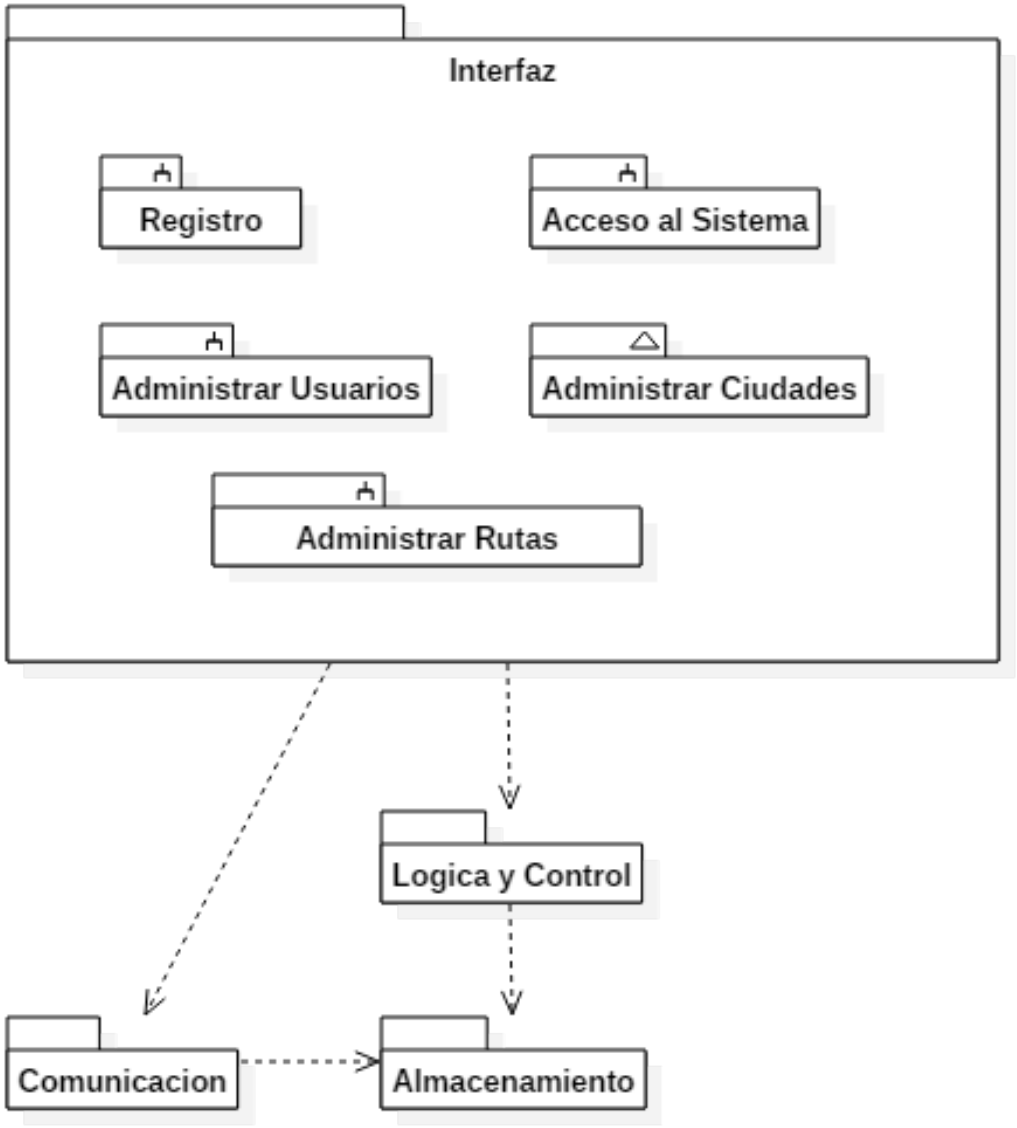


FIGURA 3.6: Diagrama de Subsistemas de la Aplicación Web

En el caso de la aplicación móvil lo visualizaremos de la misma manera, separando en subsistemas, pero en este caso es para un único usuario que interaccionará con la aplicación. En la figura 3.7 podemos apreciar la distribución de los subsistemas en cada una de las capas.

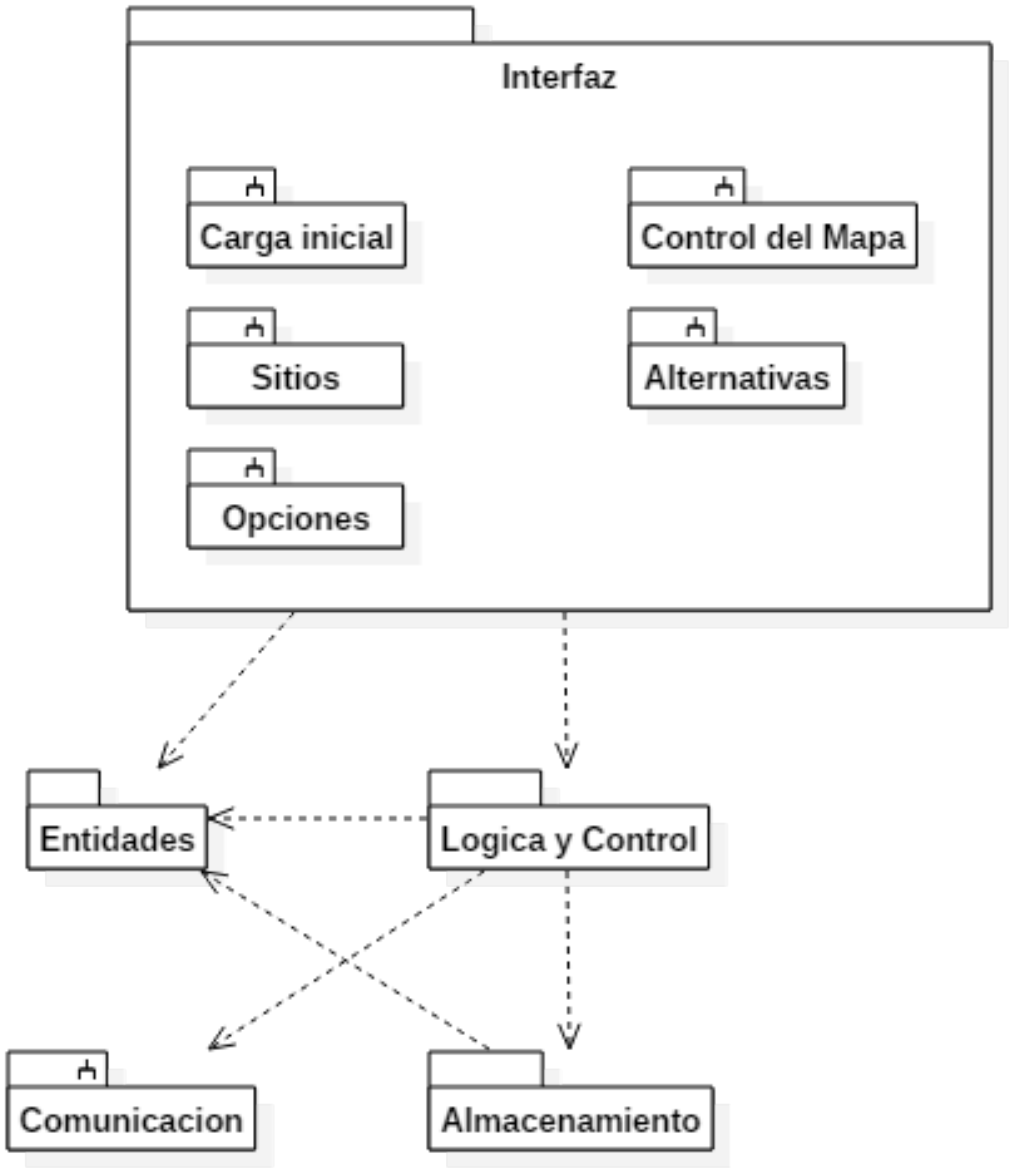


FIGURA 3.7: Diagrama de Subsistemas de la Aplicación Móvil

3.4.2. Modelo de Clases y Diagramas de Secuencia del Sistema

El propósito de este diagrama es el de representar los objetos fundamentales de nuestro sistema propuesto, de una manera mas clara, es decir queremos que nuestro usuario pueda percibir y entender con lo que va a interactuar y los elementos que va a contener el sistema. A continuación se mostrara cada clase que conforma nuestro diagrama por separado y para cada uno de los sistemas

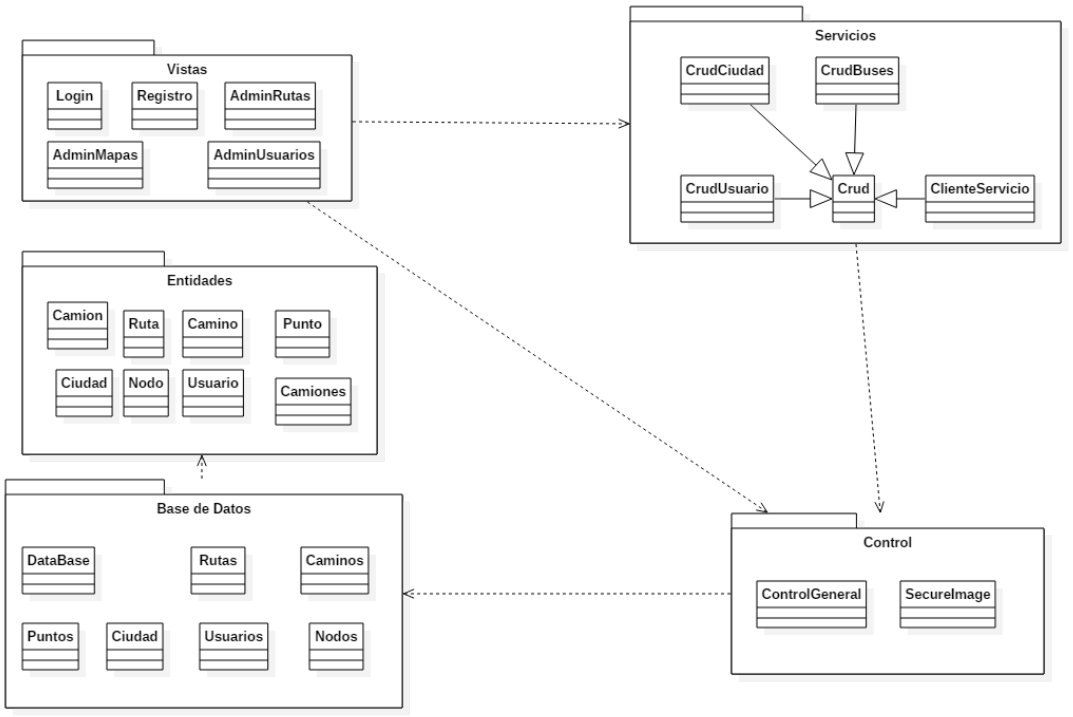


FIGURA 3.8: Diagrama de Clases de la Aplicación Web

Nuestro primer componente son las entidades las cuales representan los componentes de nuestro sistema, los cuales soportan las tablas de la base de datos. En la figura básicamente se representan los atributos y métodos para accesar a cada uno de los atributos del objeto. El control y el enlace a base de datos están fuertemente ligados, así que se incluyen en una sola capa. Tenemos una clase especial para auxiliarnos a la conexión al servidor donde esta alojada nuestra base de datos. Tendremos la clase ControlGeneral que sera auxiliar en cargar las configuraciones globales, cada instancia de esta sera necesaria para cargar el enlace a funciones de base de datos y las entidades del sistema.

La capa de servicios, nos sirve como auxiliar para realizar la comunicación con la capa de control, pues para el control de nuestras vistas se debe hacer bajo peticiones asíncronas por medio de AJAX. El FrontEnd, como ya se menciono anteriormente se encarga del control de la interfaz de usuario, esto es mediante HTML5, Bootstrap, JQuery y JavaScript. Por otro lado el BackEnd se controla con PHP.

Para nuestra aplicación móvil contamos con cuatro capas principales: vistas, logica y control, base de datos y servicios. La primera se encarga de mostrar al usuario la interfaz, la segunda se encarga de controlar y validar cada uno de los elementos que entran al sistema, la tercera se encarga de acceder a la información existente en base de datos y la ultima se encarga de comunicarse con el servidor para realizar actualizaciones en la información.

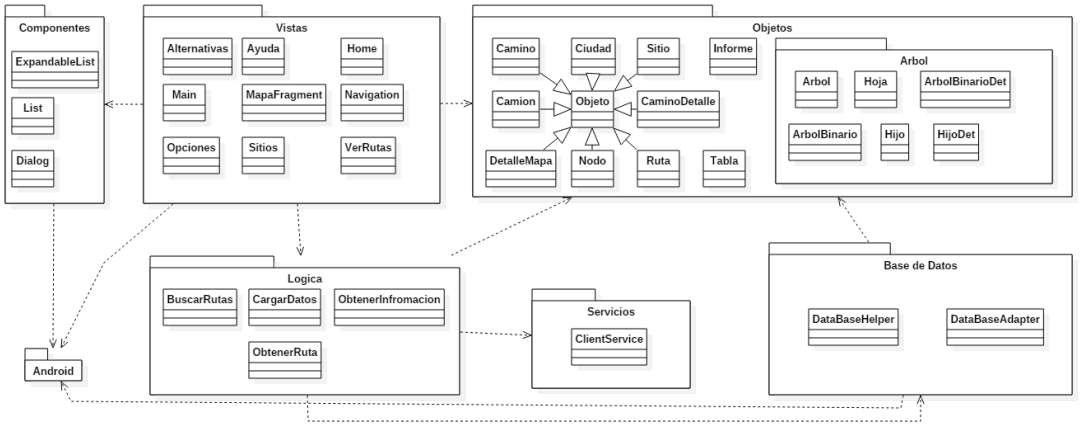


FIGURA 3.9: Diagrama de Clases de la Aplicación Móvil

Existen otros paquetes que debemos mencionar, tales como las objetos, componentes y comunes. El primero contiene todos los elementos que representan a algún elemento de nuestro sistema, componentes contiene elementos gráficos que son utilizados frecuentemente por la capa de vistas. Nuestro ultimo paquete contiene clases y métodos estáticos que son utilizados con frecuencia por todas las otras capas.

Todas las clases de la capa de vistas están relacionadas con la interfaz de usuario, podemos destacar que tendremos dos clases principales: La pantalla que mostrará las características de nuestra aplicación y la de inicio, la cual contiene todas las vistas de nuestra interfaz y que se pueden seleccionar accediendo desde este menú, inicialmente muestra el mapa de la ciudad. De manera general, cada capa se describe a continuación

- La capa de logica tiene clases que manejan la búsqueda de rutas de un punto a otro, la actualización de información y obtiene información de una ruta en particular.
- La capa de base de datos se especializa en hacer la conexión a nuestro soporte de datos, ademas de realizar funciones de obtener, actualizar, agregar y eliminar información.
- La capa de Servicios se encarga de hacer la comunicación con nuestro servidor para la obtención de actualizaciones de nuestro sistema.

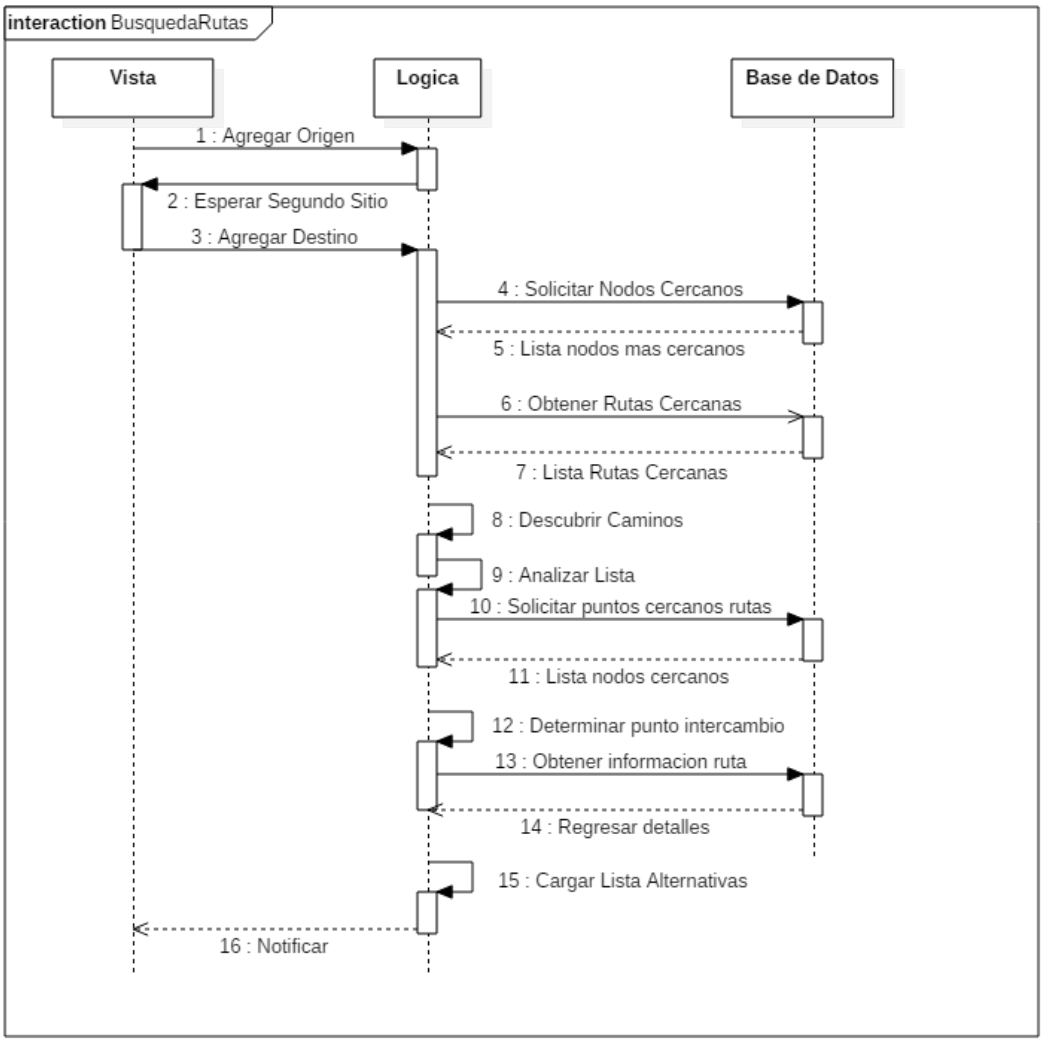


FIGURA 3.10: Diagrama de Secuencia de Búsqueda de Rutas

La parte mas importante en el desarrollo de este proyecto la describiremos en la figura 3.10 con un diagrama de secuencia. La cual requerirá de un par de datos de entrada, una vez teniendo estos se realiza una localización en la ciudad, posteriormente dada la ubicación se procede a hacer una búsqueda de a lo mas dos rutas por alternativa, posteriormente se determina el punto de abordaje, trasborde y descenso, por ultimo se genera muestra la lista en la interfaz y notificar al usuario de la finalización del proceso.

3.4.3. Diseño de la Base de Datos

El primer paso en el diseño de la base de datos fue analizar los datos que se recolectarían y determinar el uso que se pensaba hacer de los mismos. Una vez identificados cada grupo de datos se define como un grupo de tablas de datos relacionadas, estos ya siendo identificados se tienen los siguientes:

- Registro de Usuarios
- Registro de Ciudades

- Registro de Camiones
- Registro de Rutas
- Registro de Puntos
- Registro de Caminos

Con el fin de facilitar el manejo de los datos y las tablas, cada tabla tendrá por lo menos un campo que contiene un identificador único para ese registro, un campo para identificar quién ingresó los datos y otro campo para registrar cuándo se los ingresó. En la mayoría de los casos, estos campos están ocultos al usuario y el sistema los actualiza de manera automática. El administrador tendrá acceso a esta información con el fin de tomar medidas hacia los problemas que puedan surgir.

Después de identificar los grupos, se definió un identificador único para cada tabla, posteriormente los atributos, posteriormente, las relaciones entre clases, posteriormente que se tenía todo esto, se aplicó hasta la cuarta forma normal para tener un diseño óptimo de las tablas.

En la tabla 3.12 se maneja a los usuarios que interactúan con el sistema

CUADRO 3.12: Descripción de la tabla Login

Nombre	Descripción	Tipo	Tamaño
IdLogin	Llave primaria de la tabla	INT	11
Nombre	Nombre del usuario	VARCHAR	45
Apellidos	Apellidos del usuario	VARCHAR	64
Correo	Correo electrónico del usuario	VARCHAR	64
Contraseña	Contraseña del usuario. Se almacena cifrada	VARCHAR	45
Rol	Rol del Usuario El rango es de 1 a 3	INT	1
FechaCreación	Fecha y hora en que se registro el usuario	DATETIME	
FechaModificación	Fecha y hora de la ultima modificación	DATETIME	

La tabla 3.13 se llevan los registros de los accesos al sistema que realizan los usuarios registrados.

CUADRO 3.13: Descripción de la tabla AccesosLogin

Nombre	Descripción	Tipo	Tamaño
IdAccesosLogin	Llave primaria de la tabla	INT	11
FechaAcceso	Fecha y hora en la que el usuario entro al sistema	DATETIME	
Información	Informacíon adicional	VARCHAR	256
IdLogin	Id del Usuario que registro un acceso al sistema	INT	11

La tabla 3.14 lleva el registro de las ciudades dadas de alta en el sistema.

CUADRO 3.14: Descripción de la tabla Ciudad

Nombre	Descripción	Tipo	Tamaño
IdCiudad	Llave primaria de la tabla	INT	11
Nombre	Nombre de la ciudad	VARCHAR	45
Latitud	Latitud de la ubicación del centro de la ciudad	DOUBLE	11
Longitud	Longitud de la ubicación del centro de la ciudad	DOUBLE	11
Zoom	Nivel de zoom en el mapa	INT	2
FechaCreacion	Fecha y hora en que se creo el registro	DATETIME	
FechaModificacion	Fecha y hora en que se hizo la ultima modificación	DATETIME	
IdLogin	Id del usuario que hizo la ultima modificación	DATETIME	11

La tabla 3.15 lleva los camiones registrados.

CUADRO 3.15: Descripción de la tabla Camión

Nombre	Descripción	Tipo	Tamaño
IdCamion	Llave primaria de la tabla	INT	11
Nombre	Nombre del conjunto de camiones	VARCHAR	45
Imagen	Enlace de una imagen	VARCHAR	512
FechaCreacion	Fecha y hora en que se creo el registro	DATETIME	
FechaModificacion	Fecha y hora en que se hizo la ultima modifiación	DATETIME	
IdLogin	Id del usuario que hizo la ultima modificación	INT	11
IdCiudad	Id de la ciudad a la cual pertenece el camión	INT	11

La tabla 3.16 lleva las rutas registradas.

CUADRO 3.16: Descripción de la tabla ruta

Nombre	Descripción	Tipo	Tamaño
IdRuta	Llave primaria de la tabla	INT	11
Descripción	Descripción de la ruta	VARCHAR	28
Precio	Precio por un traslado en la ruta indicada	DOUBLE	5
FechaCreacion	Fecha y hora en que se creo el registro	DATETIME	
FechaModificacion	Fecha y hora en que se hizo la ultima modifiacion	DATETIME	
Estatus	Estatus de la ruta actual	INT	1
IdLogin	Id del usuario que hizo la ultima modificación	INT	11
IdCiudad	Id de la ciudad a la cual pertenece el camion	INT	11

La tabla 3.17 tiene cada uno de los puntos de control para una ruta, estos son importantes para describir el camino de una ruta

CUADRO 3.17: Descripción de la tabla PuntosCamino

Nombre	Descripción	Tipo	Tamaño
Numero	Identificador secuencial que indica el numero de marcador para cada camino	INT	4
Latitud	Ubicacion del punto respecto a la latitud	DOUBLE	
Longitud	Ubicacion del punto respecto a la longitud	DOUBLE	
IdRuta	Id de la ruta a la cual pertenece el segmento de ruta	INT	11

La tabla 3.18 tiene cada uno de los puntos que describen el trayecto de una ruta.

CUADRO 3.18: Descripción de la tabla Camino

Nombre	Descripción	Tipo	Tamaño
IdCamino	Identificador secuencial que indica el numero para cada segmento del camino	INT	11
IdRuta	Identificador que determina la ruta a la que pertenece el segmento del camino	INT	11
IdNodo	Identificador que determina el elemento del mapa al cual pernece el segmento	INT	11
Distancia-Recorrida	Id de la ruta a la cual pertenece el segmento de ruta	DOUBLE	

La tabla 3.19 contiene la información especifica de los segmentos o nodos de cada ciudad.

CUADRO 3.19: Descripción de la tabla Nodo

Nombre	Descripción	Tipo	Tamaño
IdNodo	Llave primaria que identifica el numero de nodo	INT	11
Latitud	Ubicacion del segmento respecto a la latitud	DOUBLE	
Longitud	Ubicacion del segmento respecto a la longitud	DOUBLE	
IdCiudad	Id de la ciudad a la cual pertenece el segmento	INT	11

En la figura 3.11 se muestra el esquema entidad relación de la base de datos.

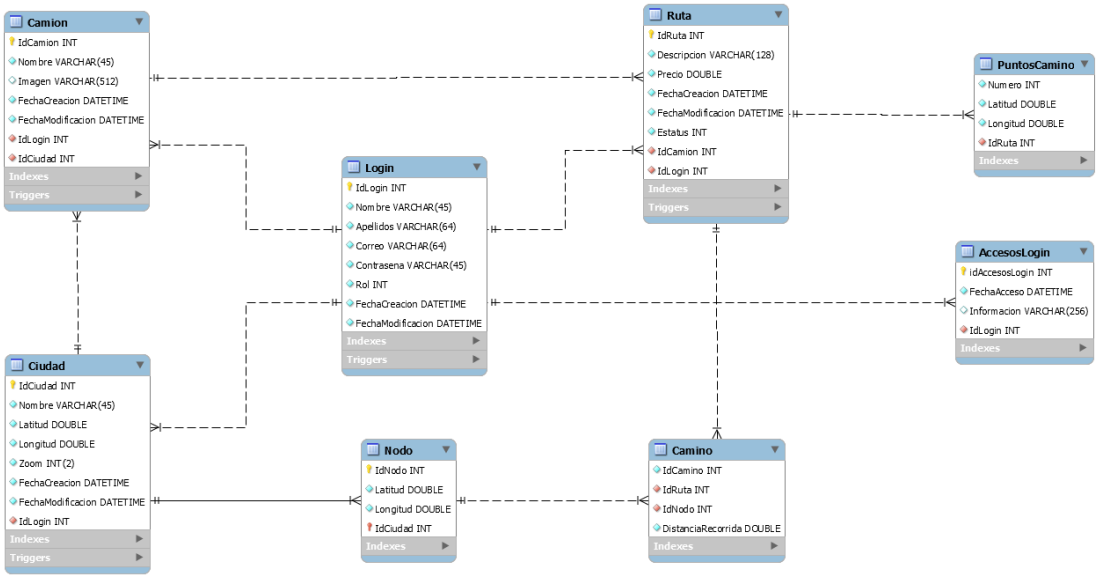


FIGURA 3.11: Diagrama entidad relación de la base de datos del sistema Web

Para la aplicación móvil es el mismo patrón de diseño, la descripción es la misma que para la aplicación web, salvo que no incluye usuarios, ademas tiene cambios en algunos tipos de datos, tales como VARCHAR Y DATETIME es TEXT, DOUBLE es REAL. Esto es debido a que el motor SQLite es limitado respecto a un motor mas robusto que MySQL. (SQLite-org 2015) La figura 3.12 muestra el diagrama de la base de datos en el dispositivo móvil.

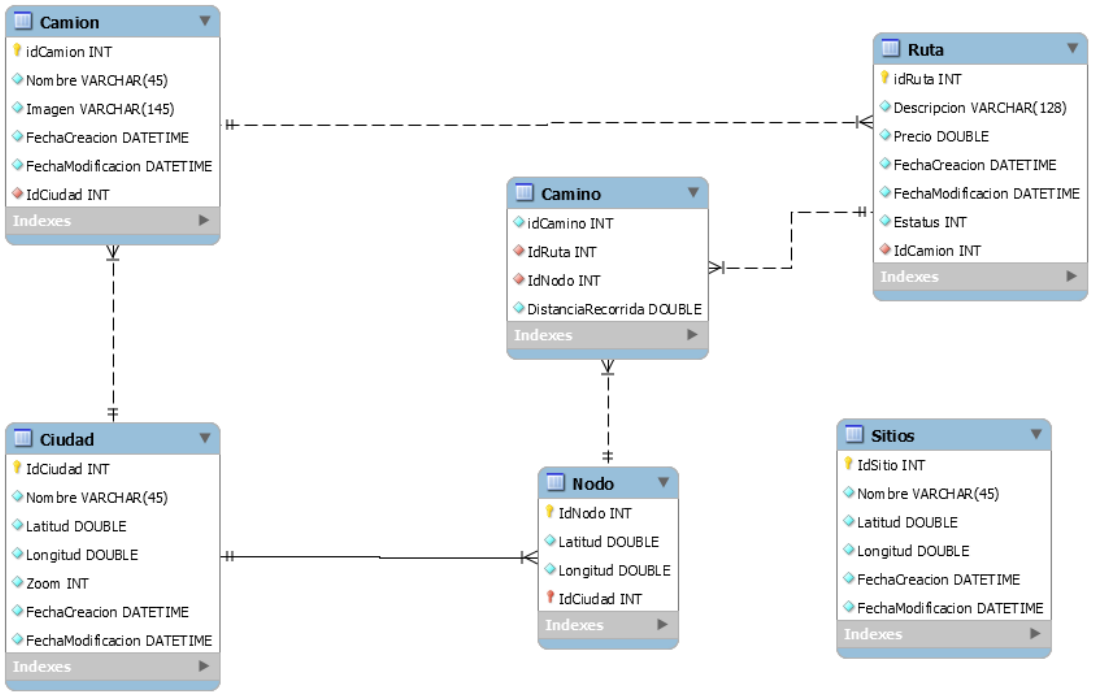


FIGURA 3.12: Diagrama entidad relación de la base de datos dela aplicación móvil

3.4.4. Diseño de la interacción

En esta sección se expondrá el hardware que se utilizará en la implementación del sistema y las relaciones entre sus componentes. Para ello utilizaremos un diagrama de despliegue muestra las relaciones físicas entre los componentes de hardware y software en lo que es nuestro sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes de software (procesos y objetos que se ejecutan en ellos)

La figura 3.13 presenta el escenario de distribución esperado para la instalación de la aplicación. El mismo se ubica sobre red con acceso a Internet para la conexión de servicios, a la base de datos o al mismo servidor.

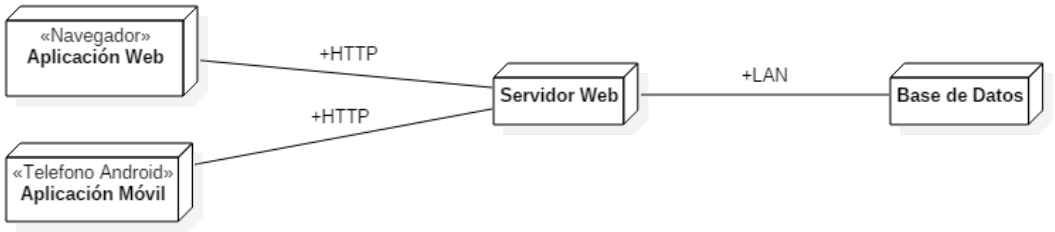


FIGURA 3.13: Diagrama de Distribución y Despliegue

En la figura 3.13 se tienen varios nodos los cuales se describen a continuación:

Cliente Tipo 1: Representa el equipo con el que los usuarios interactuarán en el sistema web

Cliente Tipo 2: Aquí está contenido el Front End del sitio.

Servidor: Representa el equipo donde se ejecuta el sistema de ejecución de procesos. Esto significa que aquí se ejecutan las funcionalidades de despacho a peticiones realizadas por cada uno de los múltiples clientes. Podría existir más de una instancia de este nodo a la cual acceder. Tales servicios son expuestos mediante Servicios Web, siempre está en continua ejecución

Servidor de bases de Datos: En continua ejecución, se encarga de resolver las consultas, albergar las tablas de usuarios, rutas, nodos y ciudades.

Las interfaces de comunicación previstas para los nodos, basadas en Servicios Web, estos posibilitan plantear múltiples escenarios de distribución y permiten el planteo de una infraestructura multiplataforma.

Capítulo 4

Implementación

4.1. Interfaz Web

En el capítulo anterior se establecieron las bases para el diseño del sistema. Una vez que han quedado claros los objetivos del desarrollo, las operaciones requeridas y las limitaciones, procedemos a definir la estructura del programa de manera clara y detallada. El primer modulo a desarrollar es la aplicación web, esta tendrá la función de administrar la información de cada una de las rutas y mapas para proveerla a la aplicación móvil.

4.1.1. Estructuración del proyecto

La aplicación contiene los siguientes módulos: Registro y acceso, administración de mapas, administración de rutas y administración de usuarios. Tomando en cuenta los modelos que se explicaron en el capitulo 3 se estructuro el proyecto de la siguiente manera:

Raíz: Aquí se alojan las vistas del sitio web.

Assets: Aquí está contenido el Front End del sitio.

Core: Aquí están todas las funciones de Validación, Entidades y conexión a la base de datos.

- **classes:** Contiene los objetos o entidades que manejaremos en la y sus métodos para el acceso a base de datos.
- **connect:** Configuración de acceso a la base de datos.
- **include:** Colección de métodos y funciones para realizar operaciones de validación creación, consulta, edición y borrado de datos
- **js:** Contiene todo el back end del sitio y control de Google Maps
- **securimage:** API para el manejo del captcha.

Sections: Elementos en común que comparten todas las vistas.

4.1.2. Módulos del sitio

Se describirá la división del sitio, se hará hincapié en los detalles de los módulos de Administración de Mapas y Rutas dado que son los más importantes para el funcionamiento de las otras dos entidades

4.1.3. Registro y Acceso

Para interaccionar con el sitio debe hacerse con una cuenta de usuario, la cual debe ser dada de alta y autorizada por un usuario para pueda accederse al sistema.
El registro es un modulo sencillo, el usuario requiere introducir su nombre, apellidos, correo electrónico, una contraseña e introducir el texto que aparece en la imagen, este fue desarrollado por Drew Phillips y se puede obtener en su sitio web (Phillips 2014). Los datos se validarán y se notificará con un mensaje que campos no son correctos. Una vez que la validación sea correcta. Al momento de hacer un registro, este se guardará en base de datos y la contraseña que se haya introducido sera cifrada.
Para acceder al sistema se necesita introducir el correo electrónico y la contraseña, ademas esta cuenta debe haber sido autorizada por algún usuario del sistema. El sistema indicará si no se puede acceder al sistema con el mensaje correspondiente a la causa.

4.1.4. Creación de una clave API

Nuestros archivos que necesiten usar mapas deben describir el enlace al API de Google Maps con la llave correspondiente, ClaveAPI debe ser reemplazada por la clave API. A continuación se muestra como se debe ser incrustada la llave :

La API señala que hay que se debe especificar un tamaño para nuestro elemento del mapa mediante CSS. Para los estilos que tengamos que definir, se hara en la hoja "style.css".

Para obtener una clave es necesario tener una cuenta de Google. Con esta se debe acceder Consola de Desarrollador y crear un nuevo proyecto. Se selecciona el proyecto creado y accedemos a la sección de "APIs y Autenticación", luego el apartado "Credenciales".

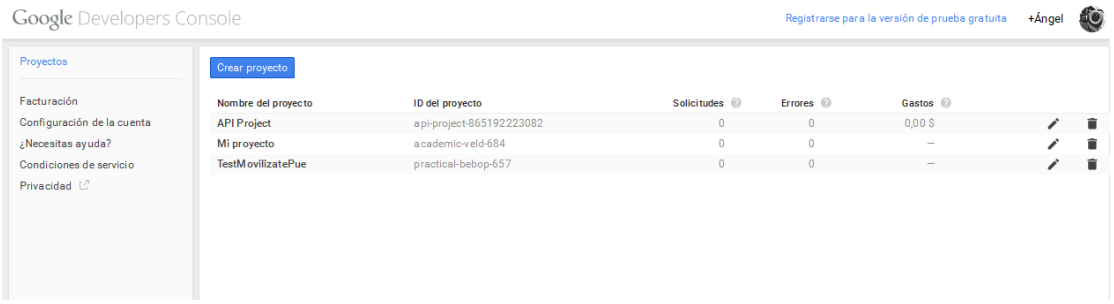


FIGURA 4.1: Consola de Google para desarrolladores

En el apartado de credenciales tenemos la posibilidad de ver, crear, modificar y eliminar claves para servidor, navegador, dispositivos Android e iOS. Nos interesa obtener una clave de navegador y otra para Android, en este capitulo mostraremos como se obtiene la clave para navegador. Debemos hacer clic en "Crear clave nueva" y elegiremos la opción clave de navegador, aquí introduciremos el nombre de nuestro dominio en el cual se podrá mostrar. Para fines de pruebas se utilizo "localhost". Después de que agreguemos el o los sitios nos mostrará la Clave API, esta cadena es la que utilizaremos en todo el sitio web.

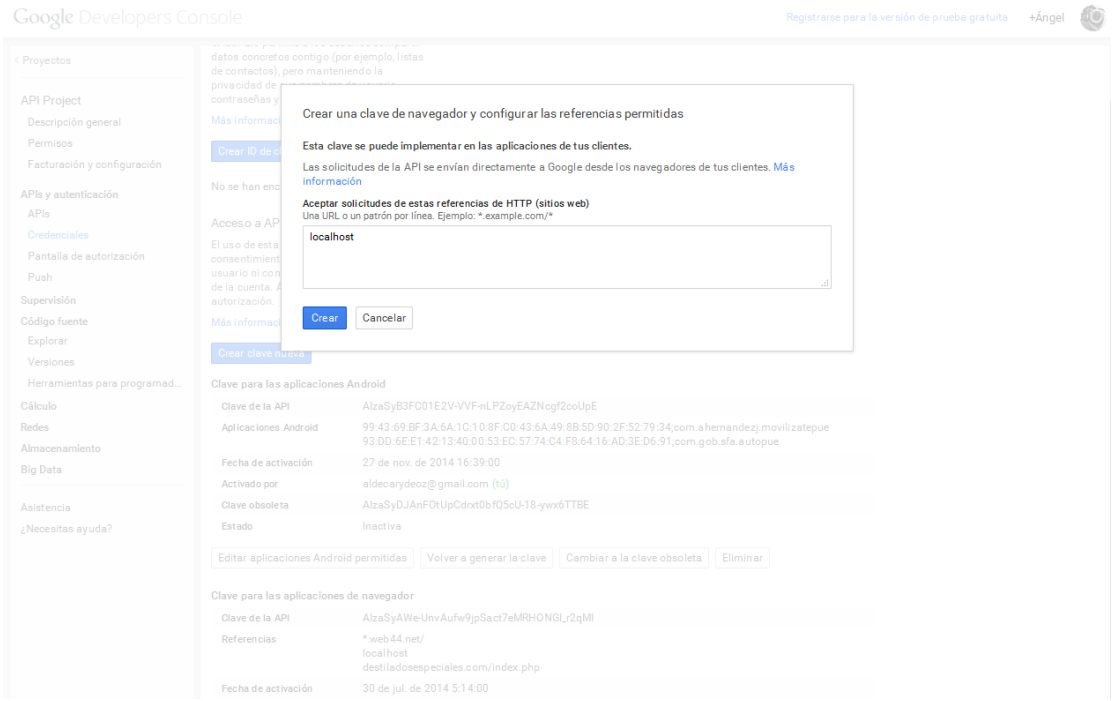


FIGURA 4.2: Alta de una clave para nuestra aplicación web

Ahora que tenemos todo lo necesario para empezar a definir la funcionalidad de nuestro mapa, para ello utilizaremos el archivo "maps.control.js" en el cual definiremos métodos y funciones que harán uso del API de Google Maps

4.1.5. Administración de Usuarios

Este modulo es exclusivo para aquellos usuarios que tengan un rol de Administrador, en este podemos hacer modificaciones a los demás usuarios, esto incluye permitirles o denegarles el acceso al sistema. La implementación es sencilla; del lado del FrontEnd se especifican los campos necesarios en la vista web, el archivo de JavaScript se especifica si se necesitan cargar o modificar los datos, mediante AJAX se hacen estas peticiones, del lado del BackEnd se recibe la petición y se hace una consulta a base de datos dependiendo el tipo de operación. Tanto las peticiones como las respuestas se hacen en formato JSON.

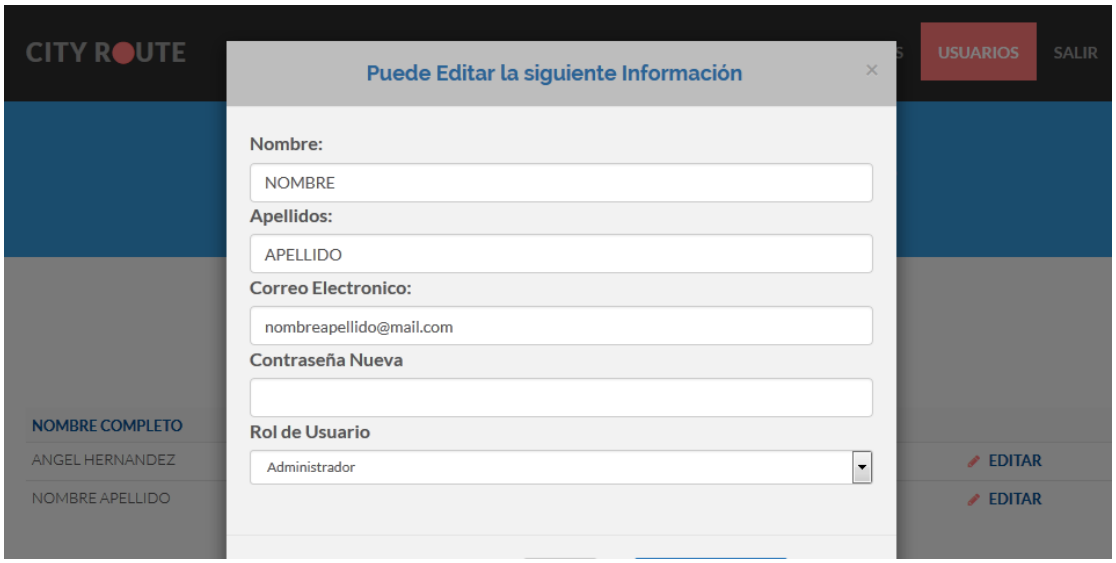


FIGURA 4.3: Modulo de Administración de Usuarios

4.1.6. Administación de Ciudades

Para la creación de este modulo tenemos dos archivos de JavaScript para el control: *maps.control.js* se encarga de todo lo relacionado el FrontEnd general de la pagina de este modulo, ademas del envío y solicitud de elementos de base de datos en formato JSON. *maps.mapas.js* esta destinado al control de los eventos de Google Maps.
La idea principal de este modulo es definir el centro de una ciudad con un marcador haciendo clic sobre el mapa y ajustar la posición del mismo y posteriormente agregar las rutas que están asociadas a tal, la ventaja de esto es que podremos redefinir el centro de nuestra ciudad y su nivel de acercamiento para los dispositivos clientes.

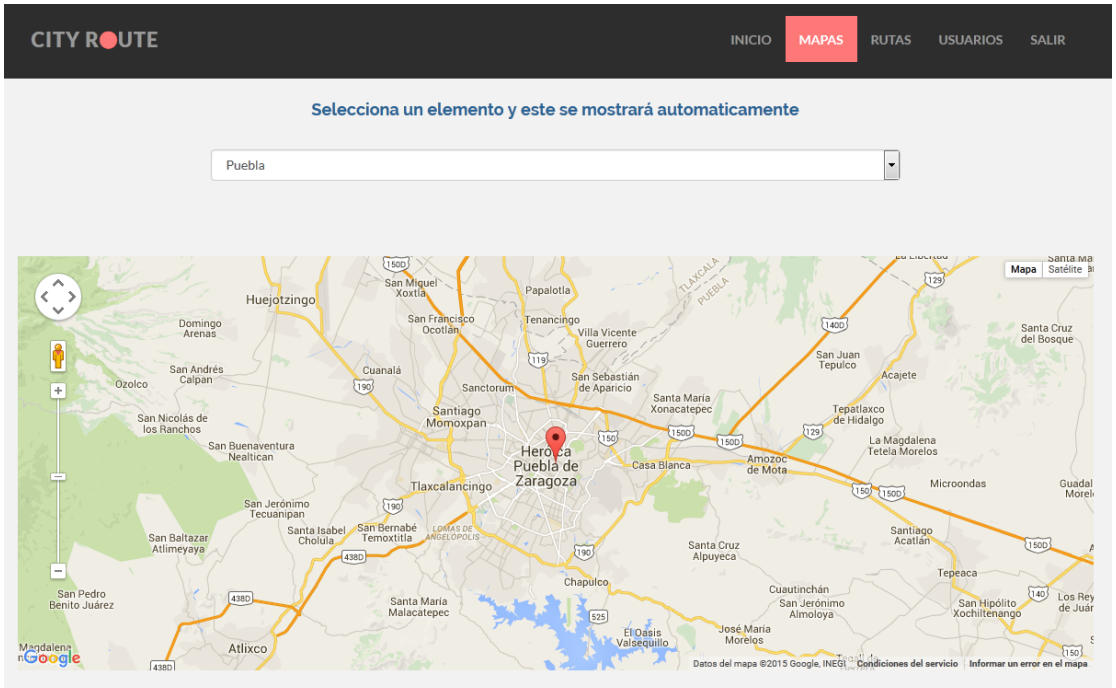


FIGURA 4.4: Modulo de Administración de Ciudades

4.1.7. Administación de Rutas

Los clientes en nuestra aplicacion movil deben ser suministrados con la información necesaria para arrojar las alternativas que el usuario necesita para llegar de un punto de origen a un destino. El fin de este modulo es poder definir las rutas en cada una de las distintas ciudades. Al igual que el modulo anterior el control del FrontEnd de la pagina se lleva con el archivo $rutas.front.js$ y el archivo $rutas.mapa.js$ es el encargado de hacer las peticiones al servicio y manipulación de elementos en nuestro mapa.
Para empezar debemos tener una ciudad sobre la cual podremos definir cada una de las rutas. En primer lugar debemos completar información básica acerca de la ruta: a que agrupación pertenece, descripción y precio. Para describir el camino la secuencia es sencilla para poder definir nuestra ruta: Se hace clic derecho y se selecciona marcar puntos. Estos se deben hacer por donde circule la ruta a agregar (esto incluye los sentidos de la calle), cada vez que agreguemos un marcador cada segundo se trazará la ruta en el mapa.

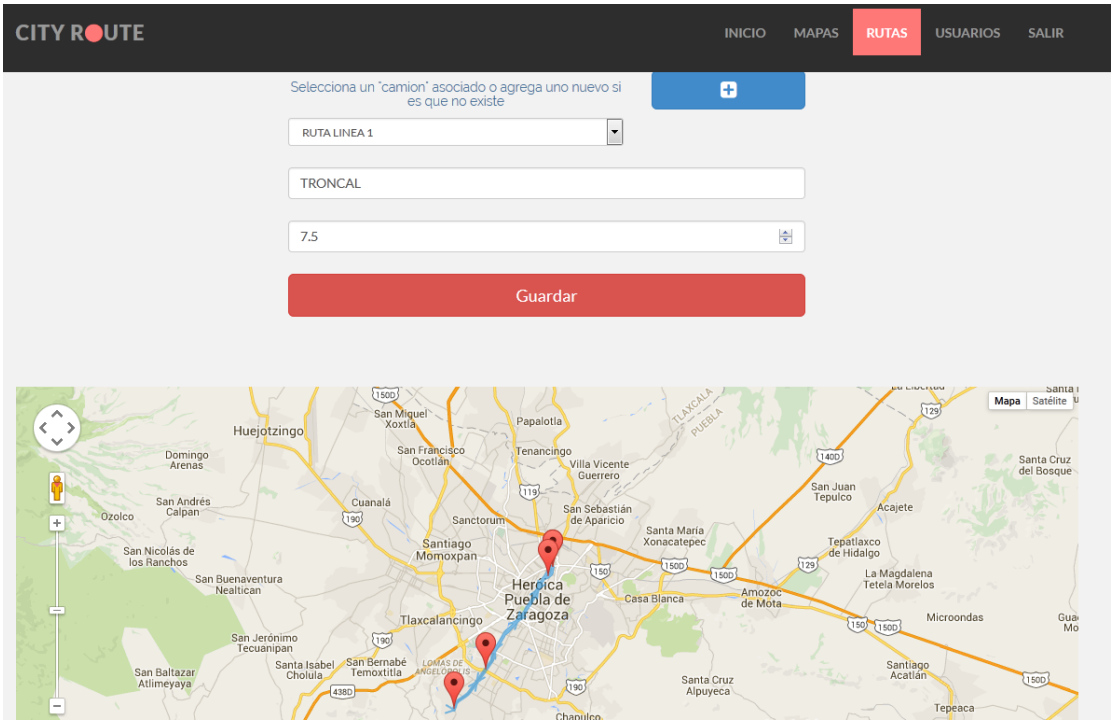


FIGURA 4.5: Modulo de Administración de Rutas

El API de Google Maps ofrece un servicio de direcciones con el cual podemos obtener trayectos en bicicleta, a pie, transporte publico y automovil. En México hasta 2015, en particular en particular a la ciudad de Puebla, no existe información relacionada a rutas para el servicio de direcciones de Google. Así que se decidió explotar la información proporcionada por la dada por automóvil. En primera no podemos hacerlo con el servicio de puntos de camino ya que nos limita a 8 puntos por petición y a 10 peticiones por segundo (Google-Inc 2015b). Lo que se hizo es hacer solicitudes por cada segmento, es decir solicitudes entre par de puntos del camino.

Algorithm 4.1 Algoritmo para obtener el camino de una ruta a travéz de una secuencia de puntos

```
function CALCRUTA(posicion,limite)
    posicion ← posicion > 0ANDBandCalcular?posicion − 1 : posicion
    BandCalcular = false
    if posicion < limite then
        delay ← 1000 + (ListaPuntos.length ∗ 250)
        request.origen(ListaPuntos[posicion])
        request.destino(ListaPuntos[posicion + 1])
        request.ModoViaje(DRIVING)
        Respuesta ← directionsService.route(request, delay)
        RenderizarSegmento(respuesta)
        calcRuta(posicion + 1, limite)
    else
        BandCalcular ← false
    end if
end function
```

En el algoritmo 4.1 podemos observar como se solicitan los segmentos de ruta por intervalos de tiempo, este intervalo irá creciendo en $0{,}25$ segundos conforme el numero de puntos que vayamos definiendo crezca. Posteriormente enviamos la solicitud en la variable $request$ y la respuesta que obtendremos es el segmento de trafico, el cual mostraremos en el mapa y guardaremos el segmento.

Cada marcador tiene la particularidad que si se modifica su posición actualiza tanto al segmento que lleva previamente y al que le precede. El algoritmo de calcular ruta se adapta para hacerlo por intervalos. Solo para el primer y ultimo caso solo actualiza el predecesor y el antecesor respectivamente.

Cuando recibimos la respuesta de un determinado segmento debemos hacer ciertas operaciones para preparar nuestro arreglo de datos, para el segmento actual borraremos los elementos que se hayan mostrado en el mapa, agregaremos y/o actualizamos el marcador bien posicionado de acuerdo a la respuesta del servicio de Google Maps y las coordenadas que describen a nuestra ruta parcial a una lista independiente para cada segmento. Posteriormente identificamos que segmento de ruta se esta procesando y se actualiza únicamente el fragmento correspondiente. Cuando hemos terminado de definir nuestra ruta debemos recoger todos los puntos de control, el pre-procesamiento incluye: los puntos de control, los puntos del camino y la longitud del trayecto de la ruta. Existe un servicio del API de Google Maps que nos permite realizar ese calculo, pero estamos limitados a determinadas consultas y ejecución de peticiones.

El calculo entre dos puntos nos es mismo que el de un espacio euclidiano, ya que uno de los postulados de Euclides afirma lo siguiente; Dados dos puntos, se puede trazar una recta (1 sÍa dimensión y sóla dirección) que los une. (Munkres 1999) Por lo tanto la linea debe ser recta, no curva. Existe otro tipo de geometría que se adapta a la que necesitamos, esta se denomina Geometría Elíptica. Esta no acepta el quinto postulado de Euclides, cambiándolo por otro el cual nos dice que por un punto exterior a una recta no pasa ninguna recta paralela a ella. Para la geometría no euclidiana sus rectas son rectas cerradas conocidas como geodésicas. Para este tipo de geometría la

suma de los angulos interiores es mayor que $180°$ (Lee 1997). El ejemplo de ella es globo terráqueo.

Teniendo esto en cuenta una manera de calcular distancias en la superficie de una esfera es usando fórmula del haversine o semiverseno. Se implemento el método que esta basado en la La fórmula del semiverseno, existen dos versiones: la versión para distancias cortas (se describe en la ecuación 4.1) y otra para distancias más grandes (MobileReference 2009). Como las distancias entre puntos de la ruta son pequeñas se opto por usar la ecuación para distancias cortas, ademas de representar el radio R de la tierra en metros.

$$\Delta\sigma = 2\arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos\phi_1 \cos\phi_2 \sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) R \tag{4.1}$$

El envió de información se hace mediante AJAX, con formato JSON. Posteriormente en el BackEnd se hace la transformación al objeto correspondiente. La estructura queda de la siguiente manera

```
e = {
        IdNodo: 0,
        IdCamino: 1,
        Latitud: 99.1455,
        Longitud: 24.9194,
        IdCiudad: 1,
        DistanciaRecorrida: 120
}
```

Una vez que hayamos cargado todos los elementos que conforman a nuestra ruta debemos agregar la información que hemos definido en el formulario y los puntos de control de la ruta para posibles modificaciones a que se requieran a la ruta. Deberemos agregar toda la informacion en un objeto JSON con la siguiente estructura

```
Object = {
        tipo: 4,
        Ruta : JSON.stringify({
                Descripcion: Descripcion,
                Precio: Precio,
                Estatus: 1,
                IdCamion: Camion,
                IdLogin: GetIdUsuarioActual()
        }),
        Puntos: ListaPuntosControl,
        Camino: DesCamino
}
```

Si requerimos editar una ruta, cargar la información es sencillo; solo necesitamos obtener los puntos de control y los datos de la ruta. Solo basta invocar al algoritmo 4.1 se empezará a dibujar nuestra ruta

4.2. Servicio Web

El servicio web prooverá a los clientes Android de la información necesaria para representar y buscar sobre las rutas definidas en el sistema web. El servicio estará contenido en un archivo $*.php$ dentro de la ruta $core/include/services$ de nuestro proyecto web

4.2.1. Esquema de comunicación

Para comunicarse con la aplicación es necesario acceder desde la siguiente dirección:

http://www.cityroute.96.lt/core/include/services/Service.php?json

La variable json debe contener ciertas atributos para que pueda hacerse correctamente la comunicación y dependerá de los datos que queramos solicitar. Esto quiere decir no todos los campos son obligatorios para determinada consulta, salvo por la variable key que es una clave hexadecimal para poder realizar solicitudes, $date$ que debe tener el formato especificado de fecha, y $tipo$ que nos indica que tipo de dato se obtendrá de la consulta representado por un numero entero, todos estos deben ir siempre en cada una de nuestras solicitudes.
Todas las respuestas que de el servicio serán en formato JSON del tipo arreglo.

```
1 json = {
2         "key": "key",
3         "date": "yyyy-MM-dd HH:mm:ss",
4         "tipo": "0",
5         "city": "0",
6         "start": "0",
7         "end": "0",
8         "route": "0",
9 }
```

A continuación se listan cada uno de los diferentes casos de consulta que es posible especificar para nuestro servicio.

- Si deseamos obtener la colección de las distintas ciudades basta con enviar los atributos obligatorios con $tipo = 0$.
- Para obtener la lista de camiones deberemos establecer $tipo = 1$ y $city$ con Id de alguna de nuestras ciudades dadas de alta.
- Para obtener la lista de rutas deberemos establecer $tipo = 3$ y $city$ con Id de alguna de nuestras ciudades dadas de alta.
- Para obtener los objetos tipo nodo debemos establecer $tipo = 2$, establecer el Id de alguna ciudad en $city$ y para $start$ y end debemos establecer el intervalo que queremos recuperar.

- Para obtener el camino de una ruta debemos establecer $tipo = 4$ y especificar en $route$ el Id de la ruta de la cual queremos su camino
- Si establecemos $tipo = 5$ se obtendrá el numero de nodos, esto es util para cuando necesitamos traer por segmentos toda la colección de nodos

Se mostrará un ejemplo respectivamente de cada uno los elementos listados previamente:

Consulta para obtener las ciudades registradas en el sistema

```
[
        {"Id":"1",
        "Nombre":"Puebla",
        "Latitud":"19.043108387892",
        "Longitud":"-98.199081122875",
        "Zoom":"11",
        "FechaCreacion":"2015-07-02 03:00:01",
        "FechaModificacion":"2015-07-02 03:00:01"}
]
```

Consulta para obtener la lista de camiones de una ciudad

```
[
        {"Id":"1",
        "Nombre":"RUTA LINEA 1",
        "Imagen":"",
        "FechaCreacion":"2015-07-02 03:06:25",
        "FechaModificacion":"2015-07-02 03:06:25",
        "IdCiudad":"1"},

        {"Id":"2",
        "Nombre":"RUTA LINEA 2",
        "Imagen":"",
        "FechaCreacion":"2015-07-02 03:06:51",
        "FechaModificacion":"2015-07-02 03:06:51",
        "IdCiudad":"1"}
]
```

Consulta para obtener rutas de una ciudad

```
[
        {"Id":"1",
        "Descripcion":"POPULAR ESTACION NUEVA 11 SUR",
        "Precio":"6",
        "FechaCreacion":"2015-07-25 00:37:45",
        "FechaModificacion":"2015-07-25 01:14:14",
        "Estatus":"1",
        "IdCamion":"3"}
]
```

Consulta para obtener un intervalo de nodos de una ciudad

```
[
        {"Id":"1",
        "Latitud":"19.08238",
        "Longitud":"-98.18061",
        "IdCiudad":"1"},
]
```

Consulta para obtener el recorrido de una ruta

```
[
        {"Id":"0",
        "IdRuta":"1",
        "IdNodo":"1",
        "DistanciaRecorrida":"0"},
]
```

Consulta para obtener el numero de nodos

```
[
        {"NumeroNodos":"21325"}
]
```

4.2.2. Implementación del servicio

El servicio recibe el parámetro JSON, verifica los atributos *key* y *tipo* que sean validos, de lo contrario manda un JSON con un atributo y un mensaje de error. Para realizar las consultas se hace el llamado a un método de base de datos en PHP con el esquema de sentencias preparadas (con el fin de agilizar y brindar seguridad al momento de hacer consultas a base de datos) de acuerdo al tipo de entidad que se especifique en la variable *tipo*, una vez terminada creada la lista esta se transformará a un arreglo en formato JSON y se enviará al cliente que la solicitó.

4.3. Aplicación Móvil

Una vez teniendo desarrollados los dos subsistemas anteriores, y teniendo un numero considerable de información almacenada en base de datos podemos proceder a empezar el desarrollo de la aplicación móvil en donde se ve reflejado el todo el trabajo desarrollado.

4.3.1. Estructura del proyecto

En nuestro entorno de trabajo Eclipse ADT definiremos los packages que contendrán a cada una de nuestras clases, cada paquete contendrá a determinadas clases que cumplen distintas funciones. En la tabla 4.1 se muestra la estructura con los detalles.

CUADRO 4.1: Estructura del proyecto en Eclipse

Paquete	Descripción
cityroute	Contiene a las vistas y a la logica de presentación de las mismas
cityroute.components	Contiene componentes de las vistas que son comunes y se reutilizan en diferentes instancias
cityroute.database	Contiene elementos que auxilian en el acceso a base de datos
cityroute.logic	Contiene las clases que controlan la lógica mas extensa, como lo es la actualización y búsqueda de rutas
cityroute.objetos	Contiene a nuestras entidades utilizadas para representar los objetos de nuestro sistema
cityroute.util	Contiene funciones diversas que son comunes en otros paquetes
cityroute.webservices	Contiene una clase que se encarga de la obtencion de datos desde el servicio web

4.3.2. Comunicación con el Servicio Web

La clase ClienteServicio es la encargada de realizar la solicitud al servicio, la aplicación consume datos de manera que llega a tener una replica simlar a la que tiene la base de datos de la aplicación web. El protocolo de transferencia de hipertexto (HTTP por sus siglas en ingles) esta diseñado para establecer comunicaciones entre clientes y servidores. HTTP trabaja como un protocolo de petición-respuesta entre cliente y servidor. Los dos métodos mas comunes usados para realizar una petición-respuesta entre el cliente y el servidor son: GET y POST. GET Solicita datos al servidor y POST los envía hacia el servidor (Fielding 2014). En su nivel mas bajo la implementación de la comunicación es sencilla, se estructura una solicitud similar descrita en la sección anterior en forma de una cadena, se agrega la URL del servicio, se envía, se transforma a una lista de objetos especificada y posteriormente es guardada en base de datos para ser usado en un momento dado.

4.3.3. Geolocalización y ubicación

Inicialmente requerimos indicar el origen y el destino en el mapa, esto se hace manteniendo pulsada la ubicación por un momento hasta que aparezca un marcador. Para determinar la ubicación se puede optar por el uso del GPS. Nos servirá para saber una posición (Puede ser el origen o el destino). Para ubicar las rutas que necesitamos debemos delimitar un área de búsqueda, para ello, cuando queremos buscar un lugar donde pase una o mas rutas y del conjunto que encontremos seleccionar la mas adecuada. Esto implica saber la distancia de nuestro punto de origen a los lugares donde haya rutas.

Para poder calcular la distancia entre dos puntos hay que tener en cuenta que la Tierra no es plana, básicamente se complica por que en el cálculo de la distancia entre ambas posiciones debemos contemplar la curvatura terrestre, esto ya se detallo en la sección anterior de este capitulo cuando tratamos de medir la distancia de un punto a otro en una ruta. Las distancias a un punto a otro dentro de la ciudad son "muy"pequeñas comparadas con el radio de la Tierra, por lo que podemos utilizar una aproximación de la tierra plana, esto se refleja en que triángulos pequeños sobre la superficie de la

esfera suman casi 180, triángulos de mayor tamaño claramente suman más de 180 esto lo podemos apreciar en la figura 4.6. Si nuestros puntos están a una distancia razonable del otro, es decir, no atraviesan la mitad del mundo ni la linea de cambio de fecha, podemos simplemente calcular la distancia como si la tierra fuese plana. Como lo que deseamos es ordenar valores , ni siquiera tenemos que sacar la raíz cuadrada, podemos simplemente añadir los cuadrados de las diferencias a nuestra consulta.

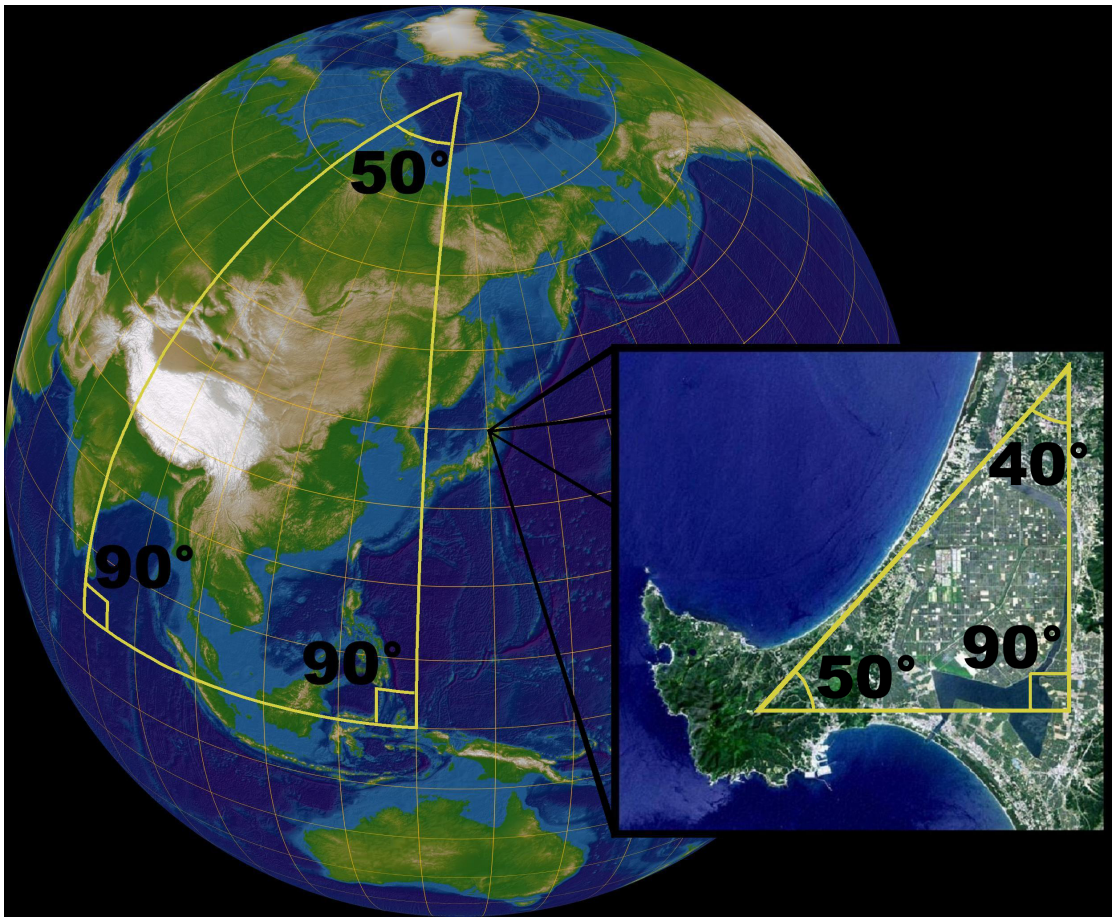


FIGURA 4.6: Representación de triángulos en la esfera terrestre

4.3.4. Algoritmo de Búsqueda de Rutas

Aquí se encuentra la parte angular del presente trabajo de tesis. Para plantear nuestra problemática debemos pensar de manera sistemática lo que hacemos cada vez que necesitamos transportarnos; En primer lugar buscamos las rutas que pasan cerca de determinada ubicación, ya sea nuestra localización actual o un punto de referencia. Del conjunto de rutas que estén cerca de la ubicación de partida elegimos una, en el mejor de los casos esta pasa cerca de nuestro destino, de lo contrario buscamos acercarnos a nuestro destino y realizar nuevamente la búsqueda buscando una ruta que nos lleve a nuestro destino.

Si analizamos un poco estos sencillos pasos podemos deducir lo siguiente: Para ambas ubicaciones necesitamos saber las rutas que están próximas respectivamente a cada ubicación. Podemos imaginarnos nuestro problema como un árbol, donde la raíz es nuestro punto de origen o destino y cada una de nuestras alternativas es un nodo.

Para la representación nos apoyaremos en un concepto interesante conocido como Arboles de Decisión; Un árbol de decisión es una representación gráfica de un problema, que tiene tres tipos de elementos o nodos que representan el árbol básico de componentes de un problema de desición: desiciones, eventos inciertos y resultados.(Sucar, Morales y Hoey 2011).

Si tomamos el concepto de árbol de desición y lo adaptamos a nuestro problema podemos hacer una representación como esta: las decisiones como las rutas que se encuentran durante la búsqueda, los eventos inciertos como los puntos de trasborde y los resultados como cada uno de los posibles caminos hacia el destino. Cabe destacar que el fin no es usar el método, mas bien usar sus elementos para ayudar a representar el problema.

Para hacer tratable nuestro problema se propuso dividir el problema en dos y buscar en extremos opuestos, esto es, hacer una búsqueda tanto en el origen como en el destino de las rutas más cercanas. ¿Por qué usar esta manera de resolver el problema? Para realizar búsqueda en Anchura o Profundidad, requeriríamos hacer una búsqueda mucho más exhaustiva, ademas de que en anchura su espacio de búsqueda es muy grande y en profundidad el algoritmo no es optimo ni completo, en cambio si hacemos una búsqueda bidireccional las debilidades de los algoritmos antes mencionados se reducen significativamente. Si limitamos la profundidad, en este caso estará acotada hasta un máximo de 2 la complejidad se convierte en lineal, el resultado que obtendremos es que generaremos un nuevo árbol acotado a una profundidad máxima de dos y el cual tiene las posibles rutas a tomar. En la tabla 4.2 se hace una comparativa entre los distintos algoritmos de búsqueda.

CUADRO 4.2: Comparación de Estrategias de Búsqueda

Citerio	**Primero en Anchura**	**Costo Uniforme**	**Primero en Profundidad**	**Profundidad Limitada**	**Profundidad Iterativa**	**Bidireccional (si aplica)**
Tiempo	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Espacio	b^d	b^d	bm	bl	bd	$b^{d/2}$
¿Optimo?	Sí	Sí	No	No	Sí	Sí
¿Completo?	Sí	Sí	No	Sí, si $l > d$	Sí	Sí

b es el factor de ramificación; d es la profundidad de la solución más superficial; m es la máxima profundidad del arbol de búsqueda; l es el límite de profundidad.

Para la búsqueda bidireccional la idea es explorar al mismo tiempo del estado inicial hacia la meta que de la meta hacia el estado inicial hasta que se encuentren en un estado. También cuando tenemos un factor de arborescencia igual en ambos sentidos se pueden conseguir buenos ahorros (Morales-Manzanares 2014) (Stuart J. Russell 1995). Esto aplicado a nuestra solución es buscar las rutas mas cercanas dada la ubicación (origen y destino) e ir construyendo nuestro árbol de alternativas, en seguida se compará si hay elementos en común en este nivel, se agregará el elemento a una lista de elementos en común, si por lo menos existen 2 alternativas que cumplan la condición de pertenencia se terminará la búsqueda, hasta aquí es el primer nivel de búsqueda. De lo contrario procederemos a hacer una búsqueda de las rutas que se interconecten

con cada uno de las alternativas del árbol generado a partir del origen, sin olvidar de seguir agregando elementos a nuestro árbol de alternativas, volvemos a buscar los elementos en común entre los dos arboles y se agregan al nodo padre correspondiente (Esto corresponderá al trasborde de una ruta a otra). Una vez que hemos terminado de analizar cada una de las opciones de búsqueda, terminamos el proceso ya que hemos alcanzado el tope de profundidad y terminamos devolviendo nuestro árbol de alternativas y nuestrá lista de candidatos.

Algorithm 4.2 Algoritmo de Búsqueda Bidireccional entre Rutas

```
function ENCONTRARRUTAS(ListaOrigen,ListaDestino,Padre,Profundidad)
    for i ← 0; i < ListaOrigen.dim(); i++ do
        arbolRutas.AgregarHijo(Padre, ListaOrigen.get(i));
        if ∼ ListaCandidatos.contiene(ListaOrigen.get(i)) then
            ListaCandidatos.agregar(ListaOrigen.get(i));
        end if
    end for
    if ListaCandidatos.dim() > 1ORProfundidad ≥ LimiteProfundidad then
        return ;
    else
        for i ← 0; i < ListaOrigen.dim(); i++ do
            elemento ← ListaOrigen.get(i)
            NuevasRutas ← BaseDatos.ObtenerRutasCamino(elemento);
            EncontrarRutas(NuevasRutas, elemento, Profundidad + 1);
        end for
    end if
end function
```

Posteriormente con nuestra Lista de Candidatos realizamos una poda en el árbol de alternativas de los elementos que coincidan, esto nos dará una aproximación de las alternativas a mostrar al usuario. En primera instancia se verificará si la ruta actual en el árbol de alternativas está contenida en la lista de candidatos, si es así se procederá a remover sus sucesores (esto con el fin de evitar alargar el trayecto y el costo de traslado), de lo contrario se verificarán sus sucesores. Si el nodo actual no tiene ningún sucesor y no está en la lista se eliminará del árbol de alternativas, en caso de que los tuviese se volverá a aplicar de manera recursiva todos los pasos descritos anteriormente.

Otro de los objetivos de la aplicación es sugerir los puntos mas adecuados para abordar, trasbordar (si aplica) y llegada. La idea es que a partir del árbol ya podado se extienda cada una de las alternativas, es decir si inicialmente tenemos dos rutas, para cada una de estas se extienda la secuencia con sus alternativas. En este punto podemos clasificar las alternativas en 2 tipos: aquellas que solo es una ruta y las que se debe hacer un trasborde de ruta.

Para el caso en el que tenemos una sola ruta el procedimiento es sencillo: del par de ubicaciones y dada la ruta obtendremos los segmentos de ruta mas cercanos a cada posición y de estos obtener el mas cercano a nuestra posición. Recordemos que el atributo Id del objeto $CaminoDetalle$ nos proporciona el orden de la ruta y el sentido en el cual circula, entonces verificamos que la distancia sea positiva, esto nos indica que hay una secuencia que nos lleva directamente a nuestro destino, ademas verificamos que la

combiandción tenga una distancia menor para recorrer. Una vez determinada la combinación adecuada se procede a calcular la longitud y costo monetario del recorrido, este se agrega a la lista de alternativas.

Si tenemos el caso en que la alternativa contiene dos rutas, el esquema es bastante parecido, solo que ahora nos apoyaremos en una estructura de datos en la cual nos apoyaremos para tener la zona en que la ruta esta más cercana al usuario (primera ruta a tomar) y al punto de llegada (segunda a tomar), cuando no exista en la estructura haremos un llamado a base de datos para traer ese conjunto de puntos y guárdalo para no tener que volver a cargarlos. Esto se hace porque puede haber rutas que usen las mismas alternativas. La estructura usada es un árbol binario, ya que su complejidad en tiempo en el peor de los casos es $\mathcal{O}(logn)$ y en memoria es similar para el caso promedio y en el peor de los casos es $\mathcal{O}(n)$, esto lo hace apropiado para nuestro problema dado que contamos con una memoria mas limitada. (Cormen y col. 2009) otro de los cambios es que debemos buscar un punto de trasborde. Primero debemos obtener los elementos donde se intersectan los caminos de ambas rutas, y este seleccionar el que esté mas cerca a nuestro punto de origen y que pueda llegarse del punto de cambio al punto de llegada.

Finalmente se va almacenando la solución en otro árbol binario para ordenarlos por la distancia recorrida de manera ascendente. Posteriormente se debe seleccionar la alternativa en la sección correspondiente. La solución se describe en los algoritmos 4.3 y 4.4.

La complejidad del algoritmo en tiempo esta dada por lo siguiente; El numero de rutas del punto de origen (R_o) por el numero de rutas que se le intersectan a cada una R_i. La búsqueda mediante $contains()$ en Java para una lista de objetos es hasta 10 veces menor que una búsqueda binaria y teniendo en cuenta que se manejan listas menores a 100 elementos $log(R_d)$, podríamos despreciar este tiempo, esto nos deja $R_o * R_i * logR_d$. La poda del árbol de Alternativas está dado por el numero de nodos en el árbol, esto en el peor caso, en el que no se tenga que podar ninguna hoja es N_a. Determinar las alternativas esta dado por el numero de posibles alternativas A por el numero de segmentos en los puntos de inicio, intercambio y destino, como estos los tenemos agrupados y el numero máximo de elementos por cada grupo es 4, dado que solo hacemos iteraciones entre solo dos grupos y la otra iteración solo se suma, tenemos que esta dado por la constante $K = (4 * 4) + 4 = 20$. Tenemos que contemplar también la estructura de árbol binario que su complejidad en el peor caso es $\mathcal{O}(logn)$. Un punto importante aquí es que no se esta contemplando el acceso a base de datos, ya que varia de acuerdo a cada dispositivo. Ademas de que realizar una operación se incrementa considerablemente a medida que se agregan mas registros a la base, en especifico a la tabla Nodo y Camino. Finalmente la complejidad esta descrita por la ecuación 4.2.

$$\mathcal{O}(AKlogn + N_a + R_oR_ilogR_d) \tag{4.2}$$

Algorithm 4.3 Algoritmo para determinar el trayecto entre alternativas

```
function DETERMINARALTERNATIVAS(PuntosCercanos,ListaCandidatos,Origen,Destino)
    dm ← newDetalleMapa[];
    listaOrdenada ← newArbolBinario()
    for e : ListaCandidatos do
        if e.length = 2 then
            p1 ← GetElementosCaminoCercanos(e[1], PuntosCercanos[0], Origen)
            p2 ← GetElementosCaminoCercanos(e[1], PuntosCercanos[1], Destino)
            origenCercano ← CaminoDetalle();
            destinoCercano ← CaminoDetalle();
            for c1 : p1 do
                for c2 : p2 do
                    distancia ← c2.Id − c1.Id
                    if distancia>1 then
                        if origenCercano.Id = -1 AND destinoCercano.Id = -1 then
                            origenCercano ← c1; destinoCercano ← c2;
                        else if destinoCercano.Id-origenCercano.Id >distancia then
                            origenCercano ← c1; destinoCercano ← c2;
                        end if
                    end if
                end for
            end for
            if origenCercano.Id>0 AND destinoCercano.Id>0 then
                distancia ← GetDistanciaRecorrida(e.get[1], origenCercano.Id, destinoCercano.Id);
                elem ← newDetalleMapa();
                elem.ListaRutas ← e[1];
                elem.Pasos ← origenCercano, destinoCercano;
                elem.Longitud ← distancia;
                elem.Descipcion ← GetNombreRuta(e[1]);
                dm.add(elem);
            end if
        else if e.size() = 3 then
            ...
        end if
    end forreturn dm;
end function
```

Algorithm 4.4 Sub parte del algoritmo 4.3 para alternativas con dos rutas

```
if e.size() = 3 then
    p1 ← listaOrdenada.obtener(e[1]);
    p1 ← listaOrdenada.obtener(e[2]);
    if origenCercano.Id>0 AND destinoCercano.Id>0 then
        p1 ← GetElementosCaminoCercanos(e[1], PuntosCercanos[0], Origen)
        listaOrdenada.Insertar(e[1], p1);
    end if
    if origenCercano.Id>0 AND destinoCercano.Id>0 then
        p2 ← GetElementosCaminoCercanos(e[2], PuntosCercanos[1], Destino)
        listaOrdenada.Insertar(e[1], p1);
    end if
    cambio ← GetCaminoEntreRutas(e[2], e[1];
    origenCercano ← p1.get(0);
    CaminoDetalledestinoCercano ← p2.get(0);
    CaminoDetallecambioCercano ← cambio.get(0);
    CaminoDetallecambioCercano2 ← cambio.get(0);
    for c2 : p2 do
        for c3 : cambio do
            distancia ← c2.Id − c3.Id
            if distancia >1 then
                if c2.Id-cambioCercano.Id >distancia OR c2.Id - cambioCercano.Id <0
then
                    cambioCercano ← c1; destinoCercano ← c2;
                end if
            end if
        end for
    end for
    cambioCercano2 ← cambioCercano;
    cambioCercano ← GetElementoCaminoRuta(e[1], cambioCercano.Id;
    for ( doCaminoDetalle c1: p1)
        distCambio ← cambioCercano.GetId() − c1.GetId();
        if ( thendistCambio>1)
            if ( thencambioCercano.GetId()-origenCercano.GetId()>distCambio)
                origenCercano ← c1;
            end if
        end if
    end for
    if destinoCercano.GetId()-cambioCercano2.GetId()>0 then
        distancia1 ← GetDistanciaRecorrida(e[1], origenCercano.Id, cambioCercano.Id;
        cambioCercano ← cambioCercano2;
        distancia2 ← GetDistanciaRecorrida(e[2], cambioCercano.Id, destinoCercano.Id;
        elem ← newDetalleMapa();
        elem.ListaRutas ← e[1], e[2];
        elem.Pasos ← origenCercano, destinoCercano;
        elem.Longitud ← distancia;
        elem.Descipcion ← GetNombreRuta(e[1]), GetNombreRuta(e[1]);
        dm.add(elem);
    end if
end if
```

Capítulo 5

Resultados

El objetivo de este capitulo es demostrar la utilidad del sistema Web para administrar rutas y de la aplicación móvil. Para el sistema web se dará de alta una ruta y se modificara en un momento dado. La aplicación movil por su lado se pobrará con distintos puntos de origen y destino, actualización de información, rango de búsqueda, validación en distintos dispositivos, entre otros.

5.1. Pruebas Aplicación Web

5.1.1. Alta de una Ruta en el Sistema Web

Definir cada una de las rutas es un proceso muy largo, aproximadamente 5 minutos. El proceso es sencillo, debemos acceder al sistema con nuestras credenciales, seleccionar la apartado de mapas la subsección de registrar nueva ruta. Rellenar la información correspondiente y empezar a definir la ruta entre las calles de la ciudad. El servicio de direcciones de Google regularmente da preferencia a avenidas principales, asi que es recomendable detallar nuestra ruta con puntos de control. En la figura 5.1 podemos apreciar como se da de alta una ruta. Posteriormente debemos dar clic en guardar y comenzará a determinar la longitud del trayecto, enviar la información y guardarla.

Como se menciono en el capitulo 3, registrar una ruta requiere de un camión asociado. Este se debe dar de alta haciendo clic en el botón azul con el símbolo de más (+). Cuando se cierre la ventana del formulario automáticamente se verá reflejado en la lista de camiones disponibles para el registro de una ruta.

En algunas ocasiones el servicio envía una alerta con el mensaje $OVERQUERYLIMIT$, eso es que porque se ha excedido el numero máximo de solicitudes al servicio por intervalo de tiempo. Es aconsejable dejar de marcar puntos durante algunos minutos ya que si no se hace es probable que el servicio se nos deniegue parcial o definitivamente.

Cada vez que agreguemos una nueva ruta a nuestra base de datos podrá ser vista y/o modificada mas adelante. La aplicación móvil deberá hacer uso de la funcionalidad de descargar información de rutas, contenida en la sección de opciones.

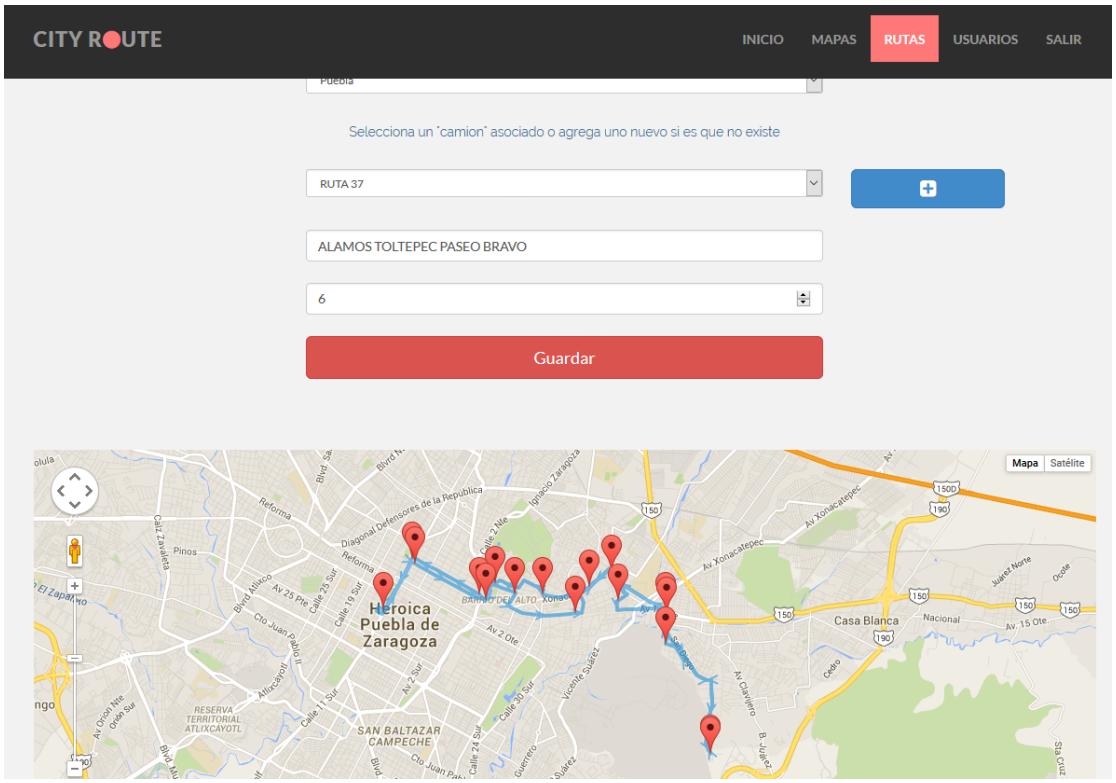


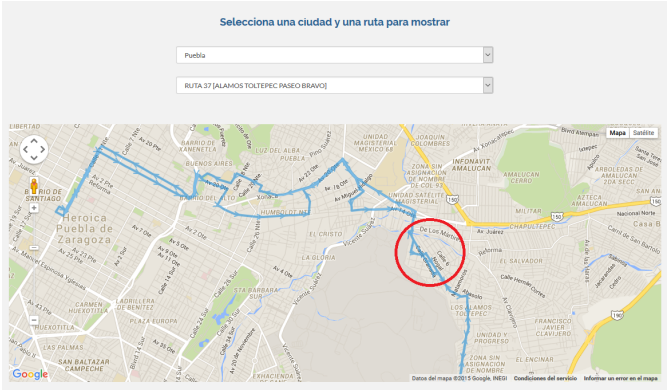
FIGURA 5.1: Representación de triángulos en la esfera terrestre

5.1.2. Modificación de una Ruta en el Sistema Web

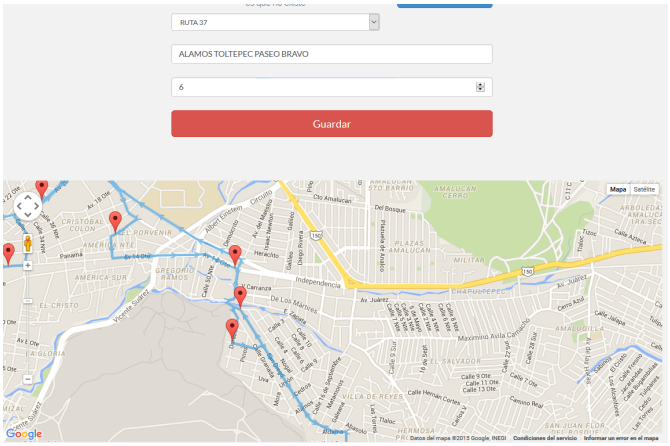
Existe la posibilidad que las rutas cambien su trayecto, en la prueba anterior se va a modificar el área señalada en la figura 5.2a, únicamente debemos seleccionar aquella ruta que nos interesa modificar, y alterar la posición del marcador que mas nos interese. La figura 5.2b muestra la modificación hecha tan solo que se reubicara un marcador.

El proceso vuelve a calcular la distancia recorrida. Ademas en base de datos se borra toda la información previa del camino que exista, reemplazando esta con la información actualizada de la ruta, La tabla nodo no sufrirá ningún cambio a menos que se agregue uno nuevo, ya que si decidimos eliminar alguno podremos afectar a varias rutas que utilicen a este mismo.

Se pueden hacer las modificaciones que se necesiten a cada ruta, incluso inhabilitar, esto es útil si pensamos en que exista una modificación en la estructura de transporte de la ciudad. Cada vez que realicemos una modificación debemos actualizar nuestra aplicación en el apartado de opciones, actualizar información de rutas



(A) Mapa Inicial



(B) Resultado

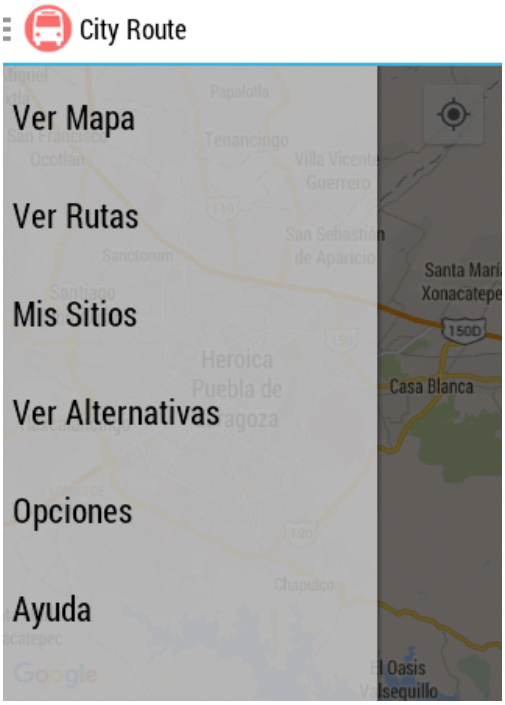
FIGURA 5.2: Modificación del trayecto una Ruta

5.2. Pruebas la Aplicación Móvil

5.2.1. Instalación y Actualización de la Aplicación Móvil

Tras instalar la aplicación e inicializarla tendremos una pantalla informativa, al final deberemos presionar en inicializar y se hará una carga de los datos tal cual existen en la base de datos del sistema web, esta copia es un respaldo adaptado al modelo utilizado en la aplicación móvil. Una vez cargado el mapa podemos notar en el margen superior izquierdo un menú con el cual apartados en los cuales podremos ver las rutas existentes en el sistema, nuestros sitios de interés, las alternativas generadas por una búsqueda y opciones para nuestro sistema. Podemos ver en el listado de rutas cada una de ellas, incluso sus recorridos. Basta con que seleccionemos una ruta y se desplegaran las variaciones de esta, deberemos hacer clic en una de ellas, nos cambiara automáticamente la pantalla al mapa y se mostrara el recorrido. Es posible actualizar la información de las rutas en nuestra aplicación móvil, solo basta con ir a la sección de opciones y seleccionar la opción Descargar Información de Rutas. Este proceso durará dependiendo del número de rutas nuevas y/o actualizadas, además de nuestra conexión a Internet. Podemos ajustar el rango de búsqueda de los puntos de origen y destino.

(A) Pantalla Inicial

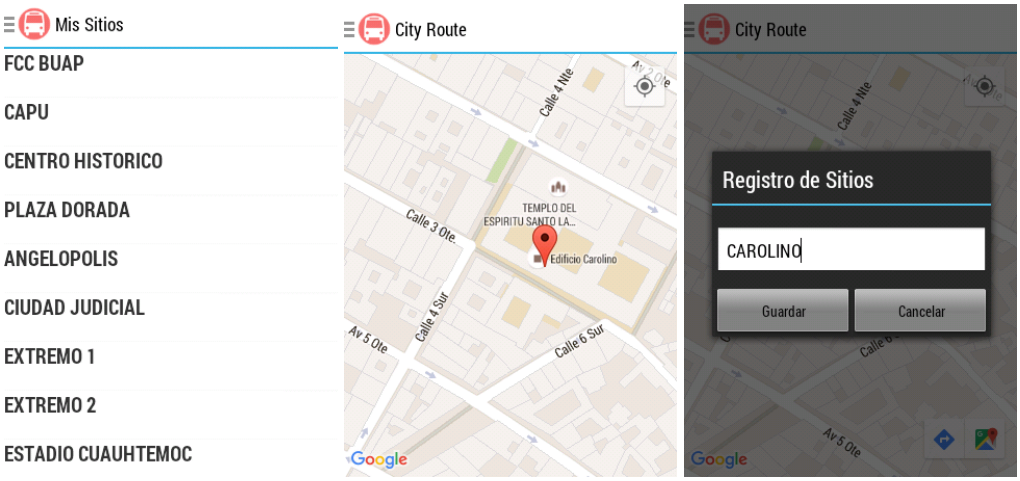


(B) Menú de Navegación

FIGURA 5.3: Modificación del trayecto una Ruta

5.2.2. Agregar Sitios Personalizados

La sección Mis Sitios permite guardar ubicaciones en el mapa que son frecuentes o son de nuestro interés para nosotros. Para agregar uno simplemente basta que vayamos al apartado del mapa, naveguemos y realicemos un acercamiento hacia el sitio de intereses que queramos guardar, mantendremos pulsado durante 2 segundos y se agregará un marcador. Para guardarlo solo hacemos clic sobre el marcador y nos saldrá una ventana de registro. Basta con que le demos un nombre al sitio, damos clic en guardar y nos mostrará un mensaje de confirmación. Para consultar este sitio debemos regresar a la sección y lo encontraremos en el listado. Podremos hacer uso de este punto como de origen o llegada, también existe la posibilidad de editar el nombre de este o de eliminarlo del listado.

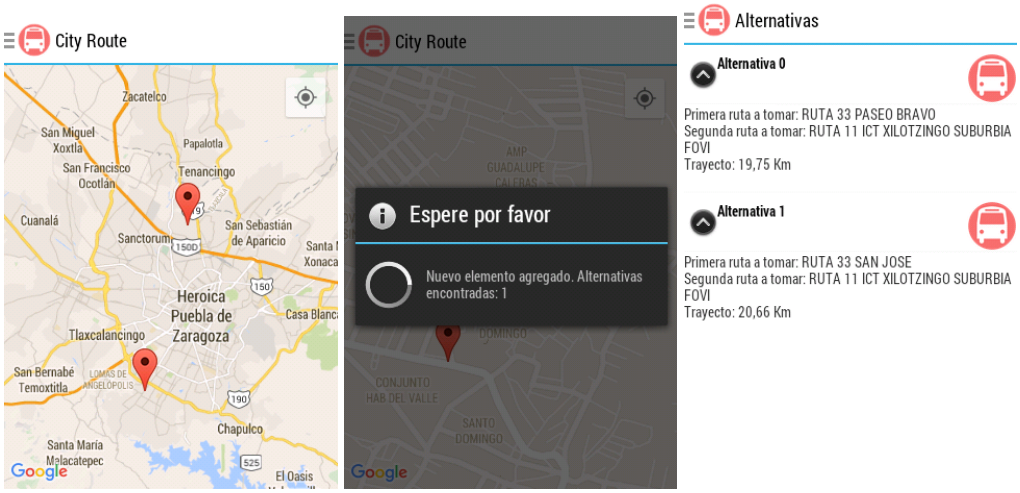


(A) Mis Sitios (B) Marcar Sitio (C) Registro

FIGURA 5.4: Sitios Personalizados

5.2.3. Pruebas para Búsqueda de Rutas

Para realizar una búsqueda de alternativas de transporte debemos marcar un origen y un destino. Pueden ser dos puntos marcados manualmente, dos de nuestros sitios o una combinación de ambos. Una vez ya marcados automáticamente se mostrará un cuadro de progreso, aquí se indica la etapa de búsqueda. Primero determinará la ubicación más cercana de acuerdo con los puntos que tenemos registrado, teniendo estos puntos el siguiente paso es obtener las rutas más cercanas, se detecta cuales rutas coinciden tanto en el origen y el destino. Posteriormente se busca la ubicación idónea para realizar el ascenso, posible trasbordo y descenso. Cuando haya finalizado el proceso se notificará y debemos ir a la sección de Alternativas para ver cuáles son aquellas que nos resulten convenientes, para mostrar una de las alternativas debemos seleccionarla y automáticamente nos cambiara la vista al mapa y se procederá



(A) Definir Origen y Destino

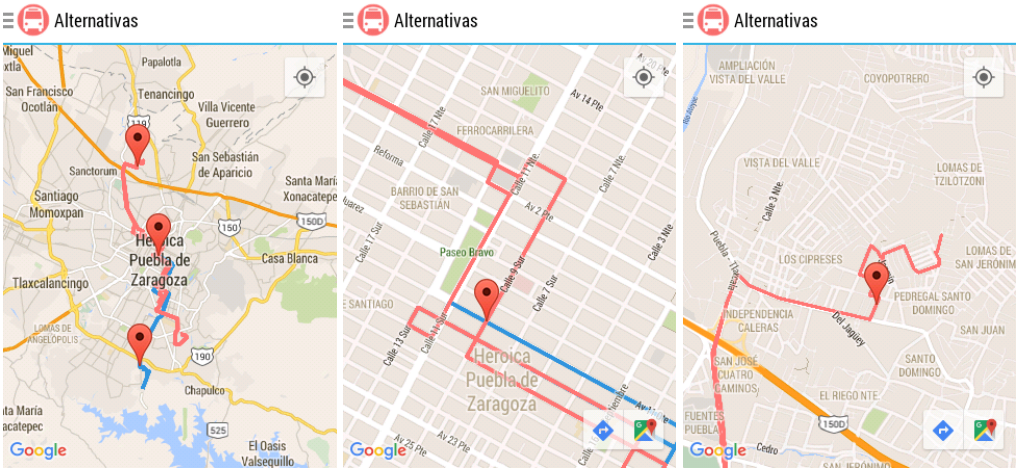
(B) Proceso Trabajando

(C) Alternativas

FIGURA 5.5: Búsqueda de Rutas

5.2.4. Muestra de alternativas

En la sección Alternativas se mostrarán las alternativas que existen entre los dos puntos, por cada una de ellas se mostrarán la o las rutas a tomar, el costo y la distancia a recorrer. Para poder mostrar esta basta con puntear la opción deseada y se nos mostrará el mapa con el trazo de la ruta o rutas que debemos tomar, además de los puntos en los cuales se sugiere tomar, trasbordar y bajar de una ruta. Si queremos repetir el proceso de búsqueda, basta con mantener presionado el mapa unos segundos y se limpiará el mapa.



(A) Desplegar alternativas

(B) Detalle de intercambio

(C) Detalle de Descenso

FIGURA 5.6: Detalle de Alternativas

5.3. Comparación

Se repitieron las pruebas en diversos dispositivos y se midió el tiempo que le tomo hacer la tarea. La carga inicial es la que se efectúa al inicio. La prueba 1 es aquella en la que los dos puntos están relativamente cerca. La prueba 2 consiste en realizar un traslado de un punto no popular a uno popular, con popular se da entender que es un sitio de interés. La prueba 3 es un traslado entre puntos que son populares. La prueba 4 consiste en hacer la búsqueda en extremos de la ciudad. Los resultados se muestran en la tabla 5.1, los tiempos se expresan en segundos. Cabe resaltar que estas cifras pueden variar dependiendo de la carga en segundo plano del dispositivo

CUADRO 5.1: Comparativa de la aplicación móvil en distintos dispositivos

	Sony Xperia X8	Alcatel 6030a	Sony Xperia Z3
Carga Inicial	35	20	13
Prueba 1	15	6	1
Prueba 2	23	9	4
Prueba 3	50	10	7
Prueba 4	120	35	9

Capítulo 6

Conclusiones

Durante el desarrollo de la solución se encontraron diferentes posibles alternativas de solución para el sistema web, tal como definir manualmente todo el trayecto de una ruta, en vez de hacerlo de manera asistida con los servicios de Google Maps. Se tenia en mente también optar por marcar nodos por toda la ciudad en cada esquina, pero dado el tiempo en que se tardaría en realizar la tarea se opto por descartar esta idea.

Las principales dificultades para el sistema almacenamiento en Android se tuvieron con el manejo de grandes cantidades de información: tanto para recibirla, almacenar de manera rápida y búsqueda en la misma. Por lo que se optó por el uso de transacciones y reducir el numero de columnas a utilizar para la manipulación de datos.

Dentro de las limitaciones del sistema esta en que el rendimiento esta relacionado por el tamaño de la información que se maneje. La búsqueda que solo implique una ruta en la alternativa no esta tan impactada por esto como lo es la búsqueda con 2 rutas en una alternativas ya que hay que hacer mas comparaciones y consultas a base de datos. La búsqueda entre rutas esta limitada a aquellas que comparten el mismo nodo para el mismo segmento de ruta, esto significa que no se reconoce como ruta cercana aquella ruta que no coincida en un mismo nodo.

Los resultados obtenidos por la aplicación móvil se puede apreciar que nos da la adecuada dado que podemos apreciar el recorrido (siempre y cuando este actualizado) de la o las rutas que se indican en cada una de las alternativas. Ademas la interfaz concentra solo lo esencial que necesita el usuario para su manipulación. Nuestra principal aportación es llevar a una plataforma flexible el manejo de la información de las rutas y un sistema que explote estos datos de tal manera que nos de la información adecuada en un momento dado. Ademas del aprovechamiento de los recursos de computo que tenemos en los smartphones.

Consideramos que los objetivos planteados desde el principio fueron cumplidos satisfactoriamente. La investigación que dio origen a este documento nos permitió conocer y comprender tecnologías alternativas e innovadoras a las empleadas tradicionalmente. El tiempo invertido en investigar, programar y verificar fue bastante productivo y nos condujo a una mejor concepción del desarrollo de aplicaciones móviles y web bajo una metodología, la programación orientada a objetos, algoritmos de búsqueda y la realización de pruebas.

El trabajo a futuro podemos enunciar que podemos mejorar la localización transformando y almacenando las coordenadas geográficas a coordenadas equivalentes en el plano cartesiano en base de datos, ambas estarán disponibles para el uso que se necesite. Mejorar la seguridad de la aplicación durante la transmisión de datos utilizando un protocolo mas seguro que HTTP, como lo es HTTPS. Se puede implementar el sistema de la aplicación en web, de esta forma eliminamos la limitación de plataforma si se tiene un navegador que soporte HTML5 y Javascript.

Bibliografía

Dial, Robert B. (1971). «A Probabilistic Multipath Traffic Assigment Which Obviates Path Enumeration». En: *Transportation Research* 5.1, págs. 83-111.

Liu, Chao-Lin (2002). «Best-path planning for public transportation systems». En: *Intelligent Transportation Systems*, págs. 834 -839.

Chao-Lin Liu Tun-Wen Pai, Chun-Tien Chang y Chang-Ming Hsieh (2001). «Path-Planning Algorithms for Public Transportation Systems». En: *The fourth international ieee conference on intelligent transportation systems*.

Antonio Mauttone Héctor Cancela, María Urquhart (2003). «Diseño y optimización de rutas y frecuencias en el transporte colectivo urbano, modelos y algoritmos.» Tesis de mtría. Facultad de Ingeniería, Universidad de la República, Uruguay.

Suárez-Sánchez, J. P. (2005). «Aproximación al problema de ruta más corta con trasbordos». Tesis de mtría. Universidad de los Andes.

Sonia Vivas, Germán Riaño (2006). «Mobile Device Programming for Finding Optimal Path on a Fast-Bus System». En: *Systems and Information Engineering Design Symposium*. IEEE.

Hannah Bast Erik Carlsson, Arno Eigenwillig Robert Geisberger Chris Harrelson Veselin Raychev Fabien Viger (2010). «Algorithms – ESA 2010». En: 1.ª ed. Springer Berlin Heidelberg. Cap. Fast Routing in Very Large Public Transportation Networks Using Transfer Patterns, págs. 209-301.

Rolando Alamilla, Fernando Esquivel (2015a). «Ecosistema Competitivo del Mercado de Smartphones 4T14». Configuración del Mercado de Smartphones. URL: http://the-ciu.net/nwsltr/350_1Distro.html.

— (2015b). «Mercado de Smartphones en México en 2014». Evolución de la Participación de Mercado de Sistemas Operativos en México al 4Q14. URL: http://the-ciu.net/nwsltr/237_1Distro.html.

Kantar-Worldpanel-ComTech (2012). *Smartphone OS sales market share*. English. URL: http://www.kantarworldpanel.com/global/smartphone-os-market-share/ (visitado 01-08-2015).

Google-Inc (2015a). *Dashboards. Platform Versions*. English. URL: http://developer.android.com/about/dashboards/index.html (visitado 01-08-2015).

Pressman, Ian (2005). *Ingenieria del Software*. Septima Edición. Addison-Wesley Publishers Limted.

SQLite-org (2015). *Datatypes In SQLite Version 3*. URL: https://www.sqlite.org/datatype3.html.

Phillips, Drew (2014). *Secure Image PHP*. URL: https://www.phpcaptcha.org/.

Google-Inc (2015b). *Google Maps Directions API Usage Limits. Standard Usage Limits*. URL: https://developers.google.com/maps/documentation/directions/usage-limits.

Munkres, James (1999). *Topology*. Prentice-Hall. ISBN: 0-13-181629-2.

Lee, John-M (1997). *Riemannian manifolds: an introduction to curvature.* Springer. ISBN: 0-387-98271-X.

MobileReference (2009). *Math Formulas and Tables for Smartphones and Mobile Devices*. USA: MobileReference.

Fielding, Roy T. (2014). *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. Inf. téc. RFC 7231. W3 Org.

Sucar, L.Enrique, Eduardo.F. Morales y J. Hoey, eds. (2011). *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*. USA: IGI Global. URL: http://www.igi-global.com/book/decision-theory-models-applications-artificial/45946.

Morales-Manzanares, Eduado (2014). *Búsqueda Bidireccional*. URL: https://ccc.inaoep.mx/~emorales/Cursos/Busqueda/node19.html (visitado 01-11-2014).

Stuart J. Russell, Peter Norvig (1995). *Artificial intelligence: A modern approach*. Prentice-Hall.

Cormen, Thomas H. y col. (2009). *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press. ISBN: 0262033844, 9780262033848.