TT-CIS-XB-006

UNIVERSIDAD NACIONAL DE LOJA

Área de la Energía, las Industrias y los Recursos Naturales No Renovables

CARRERA DE INGENIERÍA EN SISTEMAS

"Desarrollo de una aplicación móvil Android para la búsqueda de plazas disponibles en un parqueadero"

"*Tesis previa a la Obtención del título de Ingeniero en Sistemas*"

Autora:

- Chinchay Cuenca, Marjorie Juliana.

Tutor:

- Ing. Paz Arias, Henry Patricio., Mg. Sc

LOJA-ECUADOR
2015

CERTIFICACIÓN DEL DIRECTOR

Ing. Henry Patricio Paz Arias

DOCENTE DE LA CARRERA DE INGENIERÍA EN SISTEMAS

CERTIFICA:

Que la egresada Marjorie Juliana Chinchay Cuenca, autora del presente trabajo de titulación cuyo tema versa sobre "DESARROLLO DE UNA APLICACIÓN MÓVIL ANDROID PARA LA BÚSQUEDA DE PLAZAS DISPONIBLES EN UN PARQUEADERO", ha sido dirigido, orientado y discutido bajo mi asesoramiento y reúne a satisfacción los requisitos exigidos en una investigación de este nivel por lo cual autorizo su presentación y sustentación.

Loja, octubre de 2014

Ing. Henry Patricio Paz Arias

DIRECTOR DEL TRABAJO DE TITULACIÓN

AUTORÍA

Yo**, MARJORIE JULIANA CHINCHAY CUENCA,** declaro ser autor del presente trabajo de tesis y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido de la misma.

Adicionalmente acepto y autorizo a la Universidad Nacional de Loja, la publicación de mi tesis en el Repositorio Institucional – Biblioteca Virtual.

Autor: Marjorie Juliana Chinchay Cuenca

Firma: 

Cédula: 0705470656

Fecha: 22 de julio de 2015

CARTA DE AUTORIZACIÓN DE TESIS POR PARTE DEL AUTOR, PARA LA CONSULTA, REPRODUCCIÓN PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO.

Yo, **MARJORIE JULIANA CHINCHAY CUENCA,** declaro ser la autora de la tesis titulada: **"DESARROLLO DE UNA APLICACIÓN MÓVIL ANDROID PARA LA BÚSQUEDA DE PLAZAS DISPONIBLES EN UN PARQUEADERO"**, como requisito para optar al grado de: **INGENIERA EN SISTEMAS**; autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional:

Los usuarios pueden consultar el contenido de este trabajo en el RDI, en las redes de información del país y del exterior, con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia de la tesis que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los veinte y dos días del mes de julio del dos mil quince.

Firma: 

Autor: Marjorie Juliana Chinchay Cuenca

Cédula: 0705470656

Dirección: Loja (Las Palmas, Av. Santiago de las Montañas)

Correo Electrónico: : marjoriejuliana2911@gmail.com, mjchinchayc@unl.edu.ec

Teléfono: 072574730 **Celular:** 0988151573

DATOS COMPLEMENTARIOS

Director de Tesis: Ing. Henry Paz

Tribunal de Grado: Ing. Marco Augusto Ocampo Carpio, Mg. Sc
Ing. Aurilia Virginia Torres Ontaneda, Mg. Sc
Ing. Walter Rodrigo Tene Ríos, Mg. Sc.

DEDICATORIA

El desarrollo del presente trabajo de titulación lo dedico a Dios por ser mi guía y mi sostén, a mis amados y abnegados padres, Sergio y Rocio por su cariño y apoyo incondicional, y en especial a mi adorado hijo, Julián Mateo, quien con sus tiernas acciones es el motor de mi vida, inspiración de mi lucha y dedicación.

AGRADECIMIENTO

Mi total agradecimiento va dirigido en primer lugar hacia Dios por acompañarme a lo largo de mi vida y prestarme su mano amiga en cada dificultad. De la misma forma a la Universidad Nacional de Loja, el Área de la Energía, las Industrias y los Recursos Naturales No Renovables, a la Carrera de Ingeniera en Sistemas, y a cada uno de los docentes de mi vida estudiantil que me han impartido los conocimientos necesarios para llegar a la culminación de mi carrera profesional.

De igual forma, al ingeniero Henry Patricio Paz Arias, profesional de altos valores profesionales y humanos, quien con su perseverancia, responsabilidad y dedicación supo guiarme en el desarrollo del presente trabajo de titulación.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

a. Título

“DESARROLLO DE UNA APLICACIÓN MÓVIL ANDROID PARA LA BÚSQUEDA DE PLAZAS DISPONIBLES EN UN PARQUEADERO”

b. Resumen

El presente trabajo de titulación trata acerca de la realización de una aplicación móvil Android para la obtención de parqueaderos cercanos a la ubicación del dispositivo o a una ubicación alternativa ingresada por el usuario. Para su realización ha sido necesario abordar conceptos relacionados a aplicaciones móviles, sistemas operativos móviles, la conexión a datos desde un dispositivo móvil, servicios de google y metodologías de aplicaciones móviles.

La metodología utilizada para el desarrollo de la aplicación es mobile-d, metodología ágil que no exige un nivel alto de documentación permitiendo al desarrollador enfocarse más en el desarrollo de la aplicación y terminar en un menor tiempo gracias a la codificación y pruebas que van de la mano.

Dentro de los documentos obtenidos siguiendo la metodología mobile-d se encuentran el modelo del dominio, el modelo de base de datos, el esquema de navegación, story board y story cards que componen una base fundamental para realizar una adecuada codificación y ofrecen información oportuna y eficiente acerca del funcionamiento de la aplicación.

Cada una de las fases realizadas dentro del presente proyecto ha permitido obtener los resultados necesarios para cumplir con cada uno de los objetivos planteados y así obtener una aplicación móvil que permita buscar plazas disponibles en un parqueadero como posible solución ante la problemática de los conductores de encontrar una plaza disponible en un parqueadero cercano.

Summary

This research work is about making a mobile Android application to find parking place near the location of the device or to an alternate location entered by the user. For its realization it has been necessary to analyze concepts related to mobile applications, mobile operating systems, and data connection from a mobile device, google services and methodologies of mobile applications.

The methodology used to develop the application is known as Mobile-D, an agile methodology that does not require a high level of documentation allowing the developer to focus more on application development and finished in less time thanks to the coding and testing ranging from the hand.

Among the documents obtained following the mobile-D methodology are the domain model, the model database, the navigation scheme, story board and story cards that make up a fundamental basis for proper coding and provide timely and efficient information on the operation of the application.

Each of the phases carried out within this project has yielded the results needed to meet each of the objectives stated and so to obtain a mobile application that allows to search for available spaces in a parking lot as a possible solution to the problem of drivers in finding a place available to park their vehicle.

c. Introducción

Actualmente, los conductores tienen problemas para encontrar una plaza disponible en un parqueadero, ya sea en lugares que visitan con poca frecuencia o que se encuentran cerca de las principales instituciones públicas y privadas.

Por ello, se ha propuesto la realización del presente trabajo cuyo objetivo principal es desarrollar una aplicación móvil Android de Geolocalización para la búsqueda y rutas de plazas disponibles en un parqueadero cercano de la ciudad de Loja.

Para el cumplimiento del objetivo general se establecieron tres objetivos específicos, los cuales son el desarrollar una aplicación móvil Android para la obtención de los parqueaderos; aplicación cuyos datos son obtenidos a través de un webservice y de una base de datos interna de la aplicación.

De la misma forma se propuso como objetivo el implementar el módulo de Geolocalización para la visualización de los parqueaderos y el desarrollar el módulo de búsqueda y rutas de plazas disponibles, cuyo cumplimiento se dio gracias a la utilización de los servicios de google como el api de direcciones y place autocomplete.

La Universidad Nacional de loja y el área de la Energía, las Industrias y los Recursos Naturales no Renovables poseen lineamientos establecidos que rigen la estructura del proyecto de titulación, el cual tiene el siguiente orden descrito:

El Resumen presenta una sintesís del trabajo desarrollado resaltando los resultados obtenidos, los Índices correspondientes al contenido, figuras, tablas y diagramas; la Introducción que contiene una descripción del tema tratado junto al contendido del presente documento.

Asimismo, existe la Revisión de Literatura dónde se ha incluido toda la información bibliográfica utilizada en tres capítulos, los cuales son Aplicaciones móviles, Conectividad y Geolocalización para dispositivos móviles, y Metodologías para aplicaciones Móviles.

También, posee una sección denominada Materiales y Métodos que describe materiales y métodos usados para la obtención del presente trabajo; además contiene una breve

información acerca de la metodología mobile-d, metodología usada para realizar la aplicación Android.

En el apartado Resultados se incluyen todos los documentos obtenidos a lo largo del desarrollo del presente trabajo, esto de acuerdo a las fases planteadas en el alcance del anteproyecto (Ver anexo1) y de acuerdo a la metodología mobile-d.

En cambio, en la sección Discusión se hace un análisis acerca de los resultados obtenidos y su justificación como una propuesta alternativa a la problemática encontrada para la realización del presente trabajo.

El documento también contiene las Conclusiones y Recomendaciones obtenidas a través de la realización del proyecto; cuenta además con la bibliografía de la información recolectada en la revisión de literatura.

Por último, contiene los Anexos en dónde se encuentra el anteproyecto, la tabulación de una encuesta realizada para conocer los problemas de los conductores, el poster expuesto en el Congreso de Cediac celebrado en la Universidad de Cuenca y el resumen publicado en Maskana, revista de la Universidad de Cuenca.

d. Revisión de Literatura

La información bibliográfica recopilada para el desarrollo del presente proyecto de titulación se ha organizado en tres capítulos: Aplicaciones Móviles, Conectividad y Geolocalización de Dispositivos Móviles, y Metodologías para Aplicaciones Móviles

Capítulo I:

1. Aplicaciones Móviles

En la actualidad, la tecnología ha visto un gran crecimiento que se refleja en la aparición de dispositivos móviles cada vez más llamativos. Ante esto los informáticos han creado una serie de aplicaciones móviles para el uso de los usuarios; en este capítulo se abordan los conceptos acerca de dispositivo móvil, aplicación móvil, los tipos de aplicaciones móviles que pueden ser desarrolladas, los sistemas operativos para los cuales pueden ser desarrolladas y termina con una comparativa que refleja las ventajas y desventajas de cada uno de estos sistemas.

1.1. Teléfono Móvil

El teléfono móvil se puede definir como un aparato de pequeño tamaño, con algunas capacidades de procesamiento, con conexión permanente, intermitente o nula a una red, con memoria limitada, que ha sido diseñado específicamente para una función, pero que puede llevar a cabo otras funciones más generales. [1]

1.2. Smarthphone

Los smartphones o teléfonos inteligentes son teléfonos que soportan más funciones que un teléfono común como gestionar correo electrónico, ofrecer la funcionalidad completa de organizador personal, acceder de manera continua a Internet, la posibilidad de instalar programas adicionales, entre otros. [2]

1.3. Aplicación Móvil

Las aplicaciones móviles son programas software que pueden ser descargadas y a las que se puede acceder directamente desde un teléfono o desde algún otro dispositivo móvil, como por ejemplo una tablet o un reproductor de música. [3]

1.4. Tipos de aplicaciones Móviles

Las aplicaciones que se desarrollan en la actualidad para los dispositivos móviles pueden ser hechas de tres formas: desarrollo web, entornos de desarrollo nativos y entornos de desarrollo multiplataforma.

1.4.1. Desarrollo Web

Este tipo de aplicaciones se basan en lenguajes de marcas, añadiendo la facilidad de poder programar y probar sin necesidad de un emulador o un dispositivo real. A estas aplicaciones se accede directamente mediante la red.

Para el desarrollo de este tipo de aplicaciones se puede utilizar cualquier entorno de desarrollo conocido.

Para adaptar una aplicación móvil web a un dispositivo móvil se pueden utilizar las siguientes formas:

a) Tener el conocimiento (por ejemplo, el proyecto WURLF [4], que consiste en una base de datos de todos los dispositivos y sus capacidades).
b) Utilizar servidores que incorporen ese conocimiento y lo automaticen, de manera que lleguen a realizar transformaciones automatizadas. Esta opción tiene el inconveniente de que las últimas versiones no suelen estar soportadas.

Las pruebas se suelen empezar en navegadores de escritorio con soporte HTML5 pero, luego se deben realizar en dispositivos reales para comprobar la compatibilidad de los componentes desarrollados. [5]

1.4.2. Entornos de desarrollo nativos

Estas aplicaciones se realizan pensando específicamente en las características del dispositivo o de la plataforma en la cual van a ser ejecutadas. Esto permite sacarle el máximo partido a estas características o a su vez estar condicionado a normas específicas del fabricante.

Para su desarrollo es necesario utilizar el entorno de desarrollo propio de la plataforma. Las pruebas se realizan en los emuladores que suelen ser incluidos dentro de cada IDE o en dispositivos reales.

Para la implementación cada sistema utiliza su propio método y sus propios patrones, pero hay algunos puntos comunes:

a) Existe un emulador con el que se prueban las aplicaciones. Sin embargo, en ocasiones el emulador no permite emular todas las acciones de usuarios o la emulación no es lo suficientemente ágil, por lo que se necesita un dispositivo real.
b) Separación de presentación y lógica, de manera que se aprovecha al máximo los componentes.
c) Posibilidad de "debugar" la aplicación para poder tener mayor control.
d) Generalmente existen herramientas que facilitan la construcción de las interfaces gráficas o UI (user interface)

Para las pruebas, cada IDE (Integrated Developer Environmet) tiene sus herramientas, desde las típicas tecnologías de pruebas unitarias hasta sistemas más complejos, como el monkey runner de android. Sin duda, las posibles pruebas que se pueden realizar sobre las aplicaciones nativas son mucho más extensas y están más controladas que aquellas que se puedan realizar en otro tipo de aplicación, ya que se tienen las herramientas propias de la plataforma. [5]

1.4.3. Entornos de desarrollo multiplataforma

Las aplicaciones multiplataforma o aplicaciones híbridas son aquellas que desde una misma línea de código permite realizar aplicaciones nativas.

Para lograr este objetivo hay aplicaciones que realizan pre-procesamiento para acabar generando aplicaciones 100% nativas. Hay otras alternativas que proporcionan su propia arquitectura y sus propios lenguajes, y también mediante un sistema de compilación o ejecución vía máquina virtual consiguen tener aplicaciones nativas.

Hay aspectos que estas aproximaciones no van a poder evitar fácilmente (a no ser que tengan código condicional específico para cada plataforma). Son los siguientes:

- **Pérdida de controles específicos de una plataforma:** Si se tiene un control de la User Interface o una funcionalidad concreta que solo existe en una plataforma, no se la podrá generar de manera única, por el desarrollo multiplataforma.
- **Integración en el escritorio del dispositivo:** Según la plataforma, las posibilidades de añadir elementos en el escritorio de cada usuario varían.

- **Gestión de la multitarea:** Debido a que se trata de conceptos de bajo nivel de cada plataforma, cada una la trata de manera diferente, con restricciones diferentes, por lo que no será fácil hacer código común para todas sin perder mucha potencia.
- **Consumo de la batería:** Estas aproximaciones requieren de una capa de abstracción sobre el dispositivo, lo que provoca problemas como la multitarea. De la misma forma, el control sobre el consumo de batería se hace más difícil cuando no se tienen las capacidades concretas de la plataforma.
- **Servicios de mensajería asíncrona o push services:** Sirven para implementar elementos como la mensajería instantánea, pero debido a que cada plataforma los implementa de una manera, se hace complicado atacarlos conjuntamente.

Para el desarrollo, cada uno de los entornos proporciona su entorno de desarrollo completo y las aplicaciones se prueban en los emuladores de las plataformas nativas, de manera que hay que instalar el IDE propio del entorno de desarrollo y los emuladores o, en ocasiones, los SDK.

Durante la implementación, variará mucho en función de cada plataforma, pues hay algunas que podrán aprovechar todas las herramientas de desarrollo, otras que no lo necesitarán en exceso y algunas en las que el desarrollo será más difícil. En general, las herramientas facilitan el proceso de desarrollo, aunque sin llegar al nivel de las herramientas de desarrollo nativas. [5]

1.4.4. Cuadro Comparativo de aplicaciones Móviles

En base a los diferentes tipos de aplicaciones móviles, se ha realizado una comparativa de las ventajas y desventajas al momento de desarrollar cada una de ellas, descrita en la tabla I,

TABLA I.

CUADRO COMPARATIVO DE APLICACIONES MÓVILES

	Desarrollo Web	**Entornos de desarrollo nativos**	**Entornos de desarrollo multiplataforma**
Entorno de Desarrollo	Se puede utilizar cualquier entorno de desarrollo	Se debe utilizar el entorno de desarrollo propio de cada plataforma	No prestan las mismas características que los IDE'S de desarrollo nativo.
Adaptabilidad a los dispositivos	Se necesita conocimiento de las características de los dispositivos	Son compatibles para la plataforma para la que fueron creadas.	Son compatibles para la plataforma para la que fueron creadas.
Pruebas	Se realizan en navegadores web y luego en dispositivos reales.	Se realizan en los emuladores que vienen integrados en los IDE'S o en dispositivos reales.	Se realizan en los emuladores de los entornos de desarrollo nativo de cada plataforma.
Aprovechamiento de los recursos del dispositivo	Varía de acuerdo a las características del dispositivo. Existe dificultad para el acceso a ciertos recursos. Su acceso suele ser más lento.	Existe una manipulación directa de los recursos. Hay que tomar en cuenta la versión del Sistema Operativo que se ejecuta en el dispositivo.	Conlleva cierta dificultad, ya que el manejo de recursos es diferente en cada plataforma.
Dispositivos Clientes	Cualquier dispositivo móvil que cuente con un navegador web.	Dispositivos móviles de la plataforma para la cual fue realizada la aplicación.	Dispositivos móviles de las plataformas para las cuales fue realizada la aplicación.

Luego de analizar la anterior comparativa se determinó que la mejor opción para realizar la aplicación es la aplicación móvil con entorno de desarrollo nativo puesto que este tipo de aplicaciones cuentan con su propio IDE y emulador lo que facilita contrastar directamente el desarrollo de la aplicación con su funcionamiento, sin tener que probar primero en un navegador web y luego en el dispositivo, como sucede con las aplicaciones de desarrollo web.

A pesar de que abarca a un tipo específico de dispositivos, a diferencia de las aplicaciones de entorno de desarrollo multiplataforma, se puede explotar más profundamente las características de una plataforma específica al tener una manipulación directa de recursos y así obtener una aplicación mucho más robusta y eficiente.

1.5. Sistemas Operativos para Dispositivos Móviles

Un sistema operativo móvil es aquel que controla un dispositivo móvil, sirve de interface entre el hardware y el usuario; facilita al usuario o al programador las herramientas e interfaces adecuadas para el manejo del dispositivo. Este tipo de sistemas suelen ser más simples que los sistemas operativos de computadoras de escritorio o de las computadoras portátiles; suelen estar más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos. [6]

1.5.1. Componentes de los Sistemas Operativos para Dispositivos Móviles

Los sistemas operativos para dispositivos móviles poseen los siguientes componentes:

1.5.1.1. Capas

Al igual que los sistemas operativos presentes en los componentes informáticos más grandes, un sistema operativo móvil también se encuentra compuesto por varias capas. [6]

1.5.1.2. Kernel

Es la capa de software que permite el acceso a los diferentes elementos de hardware que conforman un dispositivo móvil. También es el encargado de brindar diferentes servicios a las capas superiores como los controladores de hardware, gestión de procesos, sistemas de archivos, además del acceso y administración de la memoria del sistema.

Los sistemas operativos para móviles pueden basarse en núcleos Linux, tal como lo hace Android, o hasta inclusive IOS, el SO del iPhone que utiliza un kernel heredado de Unix; sin embargo, hay otros que prefieren usar sus kernels propios como Windows Mobile y RIM. [6]

1.5.1.3. Middleware

Esta capa es el conjunto de módulos que permite que las aplicaciones diseñadas y escritas para tales plataformas puedan ser ejecutadas. Su funcionamiento es totalmente trasparente para el usuario, no debiendo realizar ninguna acción ni configurar alguna para su correcto desenvolvimiento.

Entre los servicios que presta están los motores de comunicaciones y mensajería, funciones de seguridad, servicios para la gestión de diferentes aspectos del móvil, ofrece servicios claves como el motor de mensajera y comunicaciones, codecs multimedia, intérpretes de páginas Web y servicios WAP, además de soporte para una gran variedad de servicios concernientes al apartado multimedia que es capaz de ejecutar el móvil. [6]

1.5.1.4. Entorno de ejecución de aplicaciones

Esta capa provee de todos los elementos necesarios para la creación y desarrollo de software a los programadores. Entre los servicios que los programadores pueden encontrar, se destacan un gestor de aplicaciones y una serie de interfaces programables (APIs) o "Application Programming Interface" abiertas. [6]

1.5.1.5. Interfaz de usuario

Es el elemento que permite la interacción del usuario con el dispositivo móvil, ésta presenta todos los elementos necesarios (botones, menús, pantallas y listas, entre otros) para facilitar cualquier tipo de tarea que se desee realizar en el terminal. [6]

1.5.2. Sistema Operativo IOS

Inicialmente el sistema operativo se denominó iPhone OS, ya que corría únicamente en los productos iPhone. El 7 de Junio de 2010, durante la presentación del iPhone4, Steve Jobs anunció que iPhone OS pasaría a ser llamado iOS. iOS, por tanto es el sistema operativo que ejecuta en dispositivo como iPhone, iPod, iPad, Apple TV. [7]

1.5.2.1. Arquitectura

En lo que compete a su kernel, iOS se deriva de Mac OS X, que a su vez está basado en Darwin BSD, y por lo tanto es un sistema operativo Unix. iOS es un intermediario entre el hardware y las aplicaciones , cuenta con cuatro capas de abstracción: Cocoa Touch, Media, Core Services y Core OS. (Ver figura 1)

La administración de procesos se realiza de acuerdo a una cola de prioridades pudiendo ser éstas normales, de prioridad alta, modo kernel o hilos de tiempo real. La administración de memoria la realiza de manera virtual. [8]

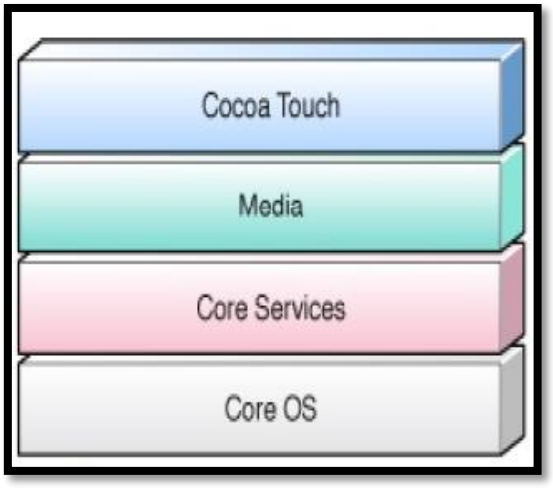


Figura 1. Arquitectura de iOS [9]

1.5.2.2. Características

- Está basado en tecnología multitáctil, empleándose gestos como deslizar, tocar o pellizcar para interactuar con las aplicaciones.
- Su última versión es iOS 6.
- Es soportado sólo por arquitecturas ARM.
- Sólo funciona con dispositivos Apple. [10]
- La multitarea se da a partir de iOS4, pero sólo a través de siete apis que proveen audio en segundo plano, voz ip, localización, notificaciones push, notificaciones locales, completado de tareas, cambio rápido de aplicaciones.[7]
- Los programadores que deseen realizar aplicaciones para esta plataforma deben de pagar.

1.5.3. Sistema Operativo Windows Phone

Anteriormente fue llamado Windows Mobile, llegó al mercado en 2003, es un sistema operativo de 32 bits en tiempo real, abierto y escalable; como Windows Phone salió en febrero de 2010, esta versión busca mantener calidad en una resolución alta de pantalla, con soluciones táctiles de tipo capacitivo y configuraciones de hardware muy eficaces.[11]

1.5.3.1. Arquitectura

Windows Phone 7 fue construido sobre el kernel de Windows Embedded CE en su núcleo, y el equipo de Windows Phone incorporó características y funcionalidades en la parte superior de la plataforma para satisfacer las necesidades de los fabricantes de teléfonos móviles.

En la arquitectura de la plataforma Windows Phone se pueden utilizar dos formas de código que son: Silverlight y XNA, ambos están incluidos en una plataforma de ejecución, (Framework Managed Code) donde es administrada y aislada, como se observa en la Figura 2.

La arquitectura de Windows Phone recurre al aislamiento de procesos (Sandbox) para ejecutar las aplicaciones con seguridad y de manera separada, cada aplicación se ejecuta en su propio entorno limitado, aislado, para evitar que los archivos de la aplicación sean dañados por otras. [11]

La Máquina Virtual esta codificada contra una capa de abstracción llamada PAL (Platform Abstraction Layer), que habilita el framework a ejecutar sobre cualquier dispositivo móvil que tenga una PAL para dicha plataforma. [12]

1.5.3.2. Características

Los desarrolladores pueden realizar dos tipos de aplicaciones para Windows Mobile: con código nativo o con código administrado (managed code). El código nativo se refiere a código C++ que utiliza directamente la API de Windows Mobile y puede ser escrito con Visual C++, y código administrado al que utiliza las clases del .NET Compact Framework con C# o VB.Net.

No brinda una versión gratuita de la plataforma de desarrollo o SDK. [13]

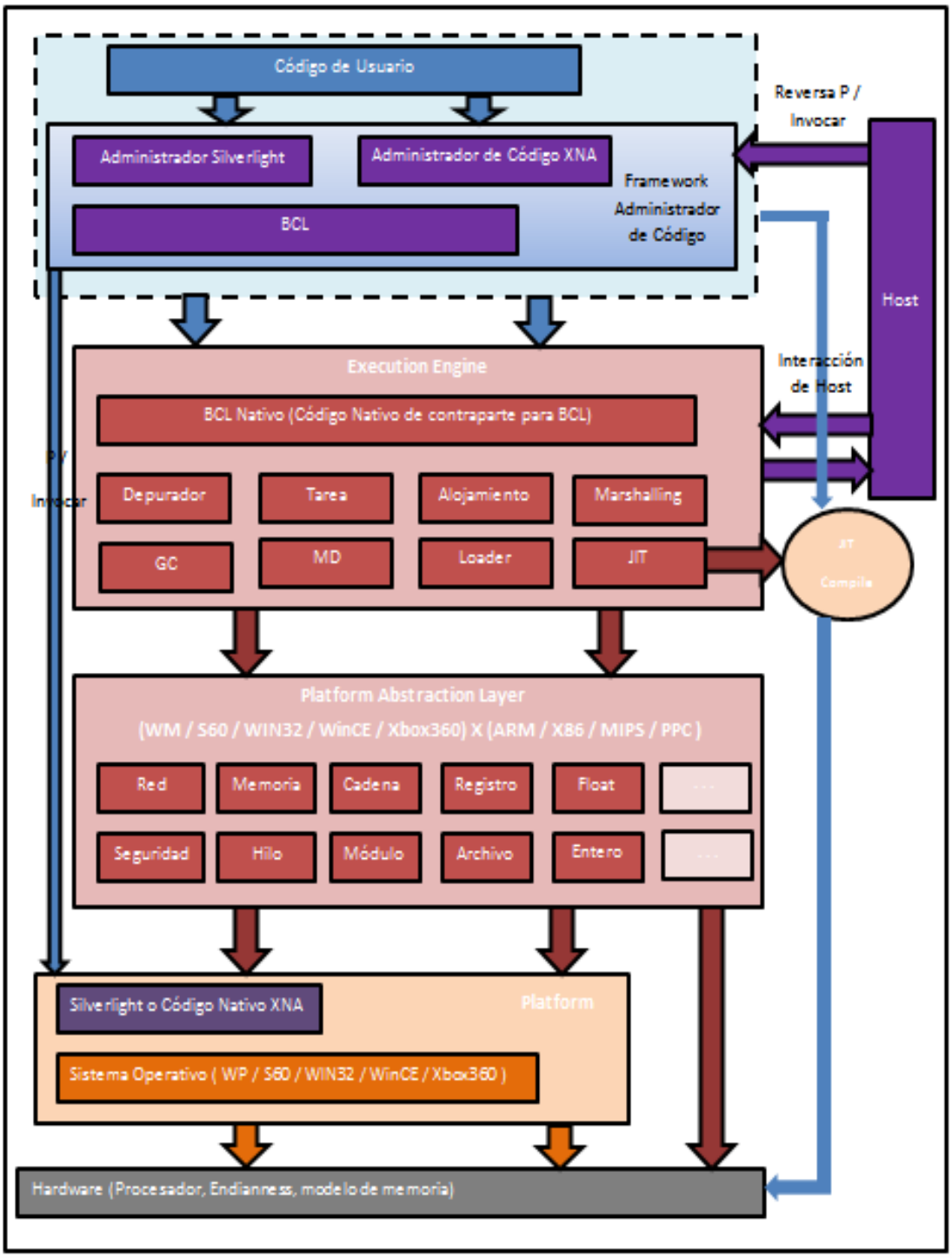


Figura 2. Arquitectura de la Plataforma de Windows Phone 7S [12]

1.5.4. Sistema Operativo Android

Android es un sistema operativo móvil basado en Linux enfocado para ser utilizado en dispositivos móviles como teléfonos inteligentes, tabletas, Google TV y otros dispositivos. Es desarrollado por la Open Handset Alliance, liderada por Google.

Android se desarrolla de forma abierta y se puede acceder tanto al código fuente como a la lista de incidencias donde se pueden ver problemas aún no resueltos y reportar problemas nuevos.

En la actualidad existen más de 700.000 aplicaciones para Android y se estima que 1.000.000 teléfonos móviles se activan diariamente. [14]

Según un estudio de International Data Corporation (IDC), empresa orientada a la investigación de mercado y asesoramiento en áreas de tecnologías de la información y la comunicación, en el estudio denominado la cuota de mercado en todo el mundo de sistemas operativos de smarthphones [15], Android es el líder indiscutible desde el año 2011 hasta el presente 2014, además de presentar un incremento año a año.

Estos son reflejados en la figura 3 realizada por la Empresa Ditrendia de España que refleja la diferencia de la cuota de mercado de Android en comparación con otros sistemas operativos.

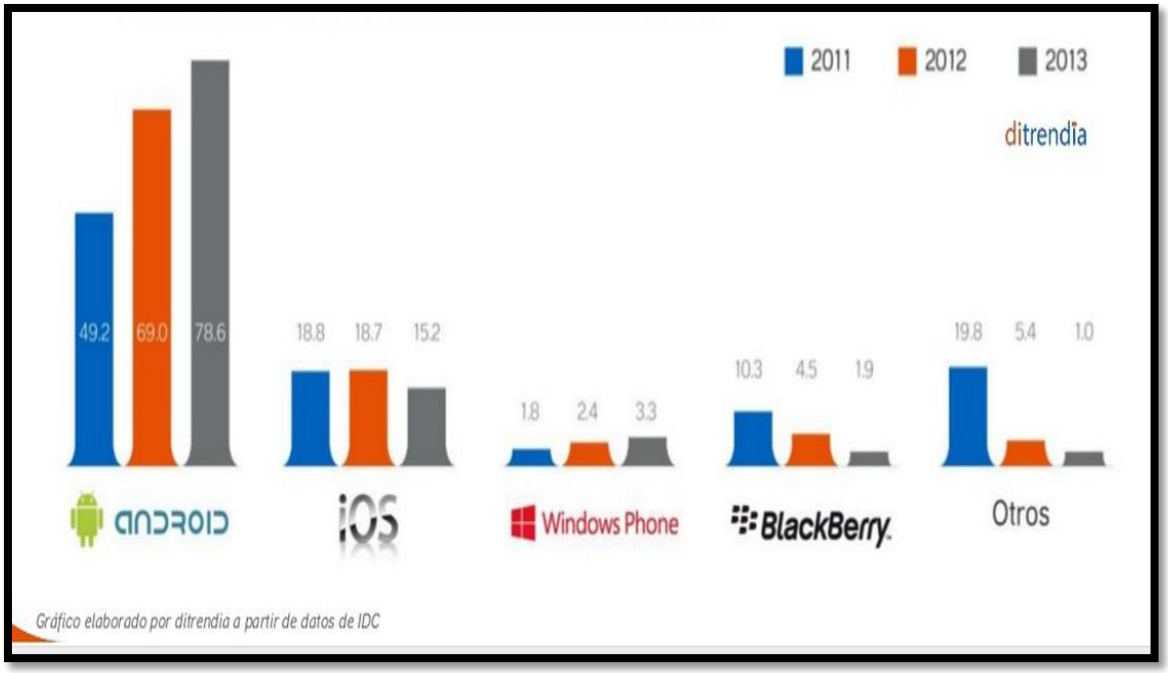


Figura 3. Sistemas operativos de los dispositivos más vendidos [16]

1.5.4.1. Arquitectura

La estructura del sistema operativo Android graficada en la figura 4 se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución. [14]

Para la gestión de procesos, Android utiliza el método CFS(Completely Fair Scheduler o Planificador Completamente Justo) [8] que busca mantener el balance en el tiempo que se le otorga a cada proceso; para lo cual guarda el tiempo de asignación llamado Virtual Runtime, el proceso con menor “Virtual Runtime” es el más próximo a ser ejecutado. [17]

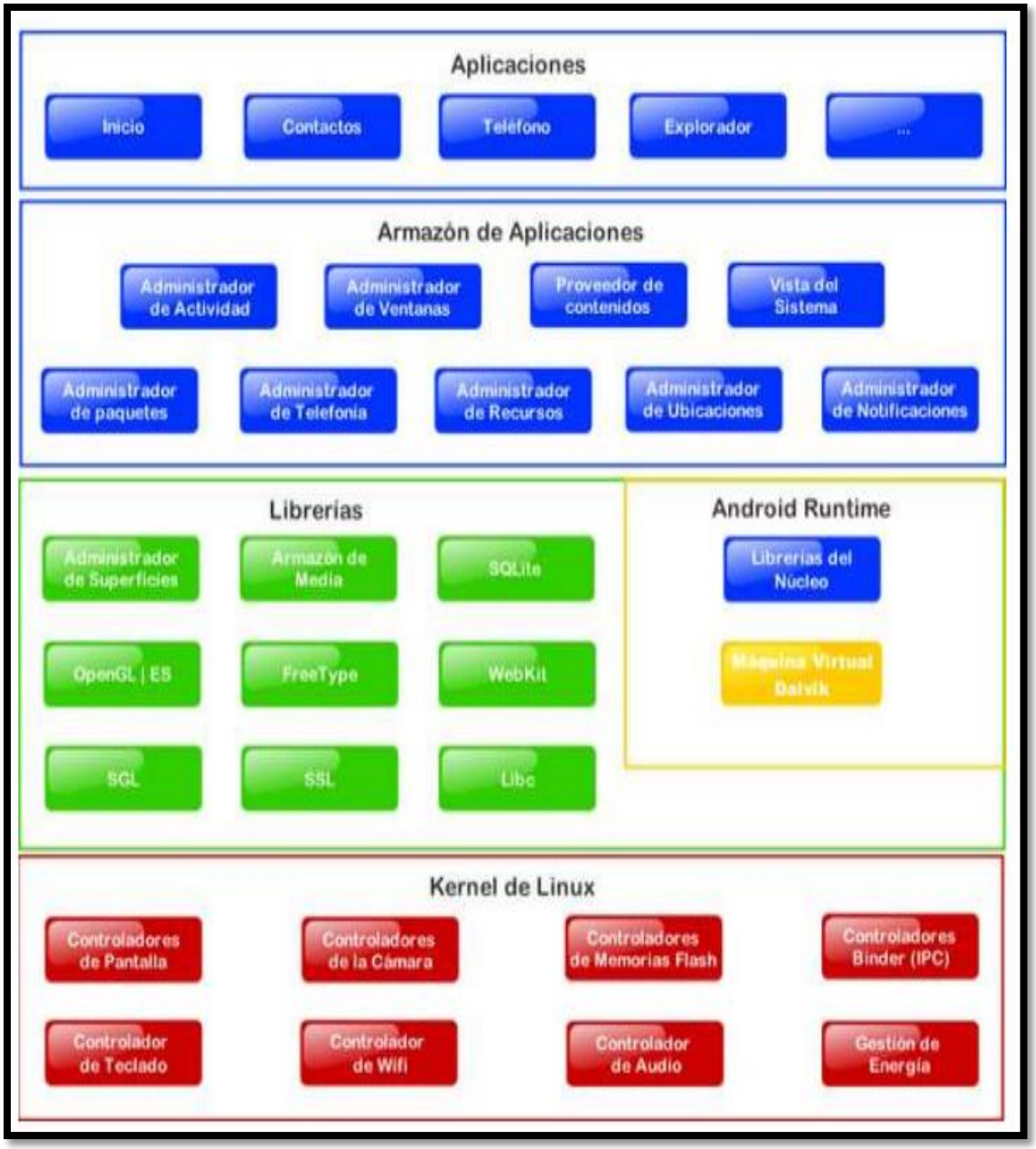


Figura 4. Arquitectura de Android [18]

El planificador de Linux mantiene un árbol rojo negro de acuerdo al tiempo de ejecución [17], este árbol rojo negro es un árbol binario de búsqueda que debe cumplir con ciertas condiciones al fin de balancear el número de nodos rojos y negros dentro del árbol. [19]

Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica, un framework OpenCore, una base de datos relacional SQLite, una Interfaz de

programación de API gráfica OpenGL ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic.

Las aplicaciones se desarrollan habitualmente en el lenguaje Java con Android Software Development Kit (Android SDK), Existen otras herramientas de desarrollo, incluyendo un Kit de Desarrollo Nativo para aplicaciones o extensiones en C, C++ u otros lenguajes de programación. [14]

1.5.4.2. Características

Cada aplicación de Android corre su propio proceso de Linux, y a su vez, cada uno de estos procesos corre su propia Máquina virtual Java. (Se aísla la ejecución entre aplicaciones).

Para facilitar la reutilización de código y agilizar el proceso de desarrollo, las aplicaciones Android se basan en componentes, ya sean actividades, servicios, receptores de eventos y proveedores de contenido.

Además, todas las aplicaciones Android deben tener un fichero AndroidManifest.xml donde se definan todos los componentes de la aplicación así como los permisos que requiere, o los recursos y librearías que utiliza. [14]

1.5.5. Cuadro Comparativo de Sistemas Operativos Móviles

En base a la información recolectada sobre los sistemas operativos iOS, Windows Phone y Android se han determinado características principales de cada una de ellas. (Ver tabla II)

TABLA II .

CUADRO COMPARATIVO DE SISTEMAS OPERATIVOS MÓVILES

	iOs	Windows Phone	Android
Kernel	Su kernel se deriva de Mac OS X, que a su vez está basado en Darwin BSD (Sistema operativo de Unix)	Su kernel fue construido sobre Windows Embedded CE	Posee un kernel basado en Linux.
Gestión de procesos	Se realiza de acuerdo a una cola de prioridades pudiendo ser éstas normales, de prioridad alta, modo kernel o hilos de tiempo real.		Utiliza el planificador completamente justo a través de la implementación de un árbol rojo-negro.
Multitarea	Es permitida desde la versión iOS4 pero sólo a ciertos procesos	Está permitida, para un máximo de 5 procesos	Está permitida
Hardware	Puede ser implementado solamente en dispositivos de la plataforma Mac	Puede ser utilizado en varios dispositivos	Puede ser utilizado en varios dispositivos
Cuota de Mercado	iOS es el segundo sistema operativo con una mayor cuota de mercado a nivel mundial desde 2011 hasta ahora 2014.	Android es el sistema operativo con mayor cuota de mercado a nivel mundial desde 2011 hasta ahora 2014.	Windows Phone es el tercer sistema operativo con una mayor cuota de mercado a nivel mundial desde 2011 hasta ahora 2014.

En base a la comparativa realizada se ha seleccionado Android como el Sistema Operativo indicado para realizar la aplicación puesto que muestra algunas ventajas como el hecho de ser el más utilizado en la actualidad debido a que es compatible con un gran número de dispositivos y permitir la multitarea de una manera más óptima que los otros dos sistemas operativos; por ejemplo, Windows pone sólo permite multitarea para cierto número de procesos

Asimismo, la multitarea de iOS que es sólo para ciertos procesos, en dónde no se incluye los procesos de una aplicación tercera.

Finalmente, Android es el sistema operativo con una mayor cuota lo que significa que los usuarios al momento de comprar un smarthphone lo prefieren sobre otros como iOS o Windows Phone; lo que significa que una aplicación hecha en Android tiene muchas más oportunidades de ser usada.

Capítulo II:

2. Conectividad y Geolocalización para Dispositivos Móviles

En el presente capítulo se aborda la información relacionada a la conexión de los dispositivos móviles a través de un web Service para la obtención de datos de una base de datos externa. Además, se abordan los servicios de Google utilizados para realizar la aplicación.

2.1. Conectividad para dispositivos Móviles

Actualmente, la forma en que se comunica una aplicación móvil con una base de datos externa es a través de un Web Service.

2.1.1. Web Service

Es un servicio o funcionalidad que se encuentra disponible a través de Internet o una Intranet , usa una forma estandarizada de mensajería (generalmente XML); no se encuentra atado a ningún sistema operativo ni ningún lenguaje de programación; también puede ser descubierto a través de un mecanismo de búsqueda. [20]. El protocolo HTTP suele ser el preferido a la hora de transportar los mensajes del Web Service. [21]

Actualmente hay dos tipos de Web Service que estandarizan su información a través de mensajes XML, en formato SOAP, llamados servicios SOAP; y los que no estandarizan su información, denominados servicios REST. [20]

2.1.1.1. Servicios SOAP

Los servicios SOAP (Simple Object Acces Protocol) implican el intercambio de mensajes XML, codificados según el protocolo SOAP.

Estos mensajes en formato SOAP (Ver figura 5) son movidos de un sistema a otro, utilizando HTTP; el sistema recibe el mensaje, hace lo que tiene que hacer, y devuelve una respuesta también en formato SOAP. Es un sistema simple, que no tiene en cuenta aspectos importantes del desarrollo de soluciones empresariales, pero que son tenidas en cuenta a través de extensiones a los estándares. [20]

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <miCabecera>miValor</miCabecera>
  </soap:Header>
  <soap:Body>
    <MiMetodo>
      <MiParametro>miValor</MiParametro>
    </MiMetodo>
  </soap:Body>
</soap:Envelope>
```

Figura 5. Estructura de un mensaje SOAP [21]

Los Web Services SOAP necesitan que se cumplan ciertas características:

- Excepto para datos binarios anexos, los mensajes deben ser transportados s obre SOAP.
- La descripción de un servicio debe ser hecha en WSDL.
- Uso de UDDI (Universal Description, Discovery and Integration).

Este tipo de web Service cuyo funcionamiento consta en la figura 6 necesita de ciertos elementos que interactúan entre sí:

Lógica. Se trata del componente que procesa la petición para generar la información solicitada por el cliente. Para realizar su función puede comunicarse con otros Servicios Web, acceder a bases de datos o bien invocar API de otras aplicaciones solicitando la información (o parte de ella) que ha de generar para enviar en formato XML.

SOAP (Simple Object Access Protocol). Protocolo de comunicación, basado en XML, que sirve para la invocación de los Servicios Web a través de un protocolo de transporte, como HTTP, SMTP, entre otros. Consta de tres partes: una descripción del contenido del mensaje, unas reglas para la codificación de los tipos de datos en XML y una

representación de las llamadas RPC para la invocación y respuestas generadas por el Servicio Web.

UDDI (Universal Description, Discovery and Integration): Directorio donde es posible publicar los Servicios Web, permitiendo con ello que los posibles usuarios de ese servicio puedan obtener toda la información necesaria para la invocación y ejecución del Servicio Web.

WSDL (Web Services Description Language): Lenguaje basado en XML que permite la descripción de los Servicios Web definiendo la gramática que se debe usar para permitir su descripción y capacidades (datos, comandos que aceptan o producen), y su publicación en un directorio UDDI. [22]

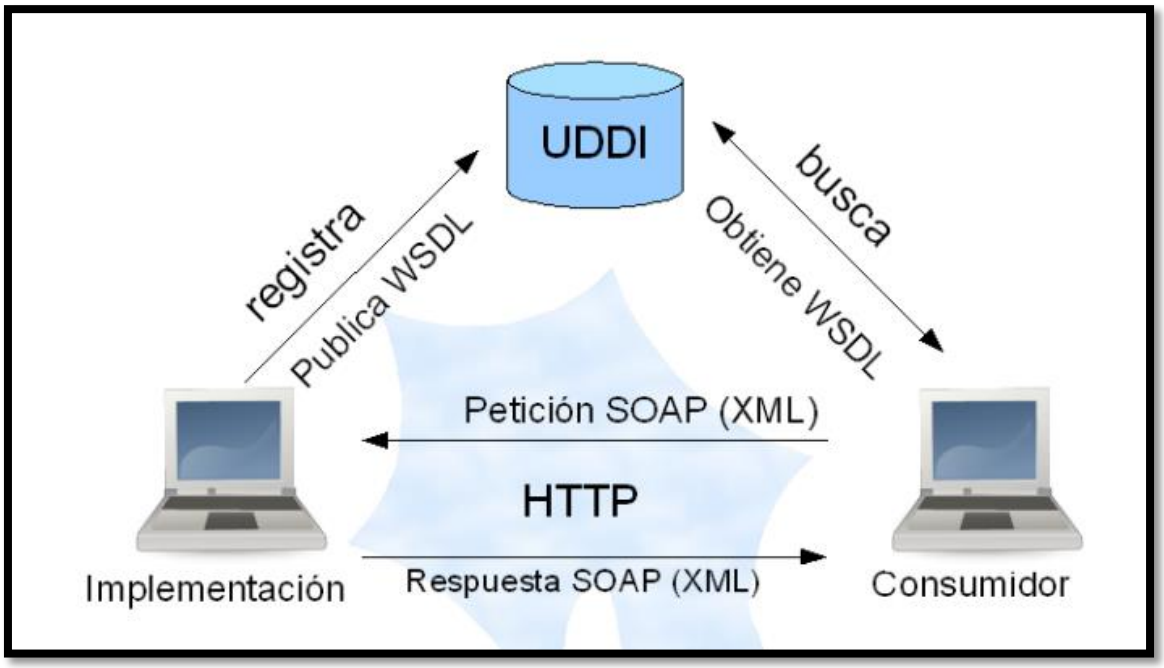


Figura 6 Funcionamiento de un web Service [22]

2.1.1.2. Servicios REST

Los servicios REST (Representational State Transfer) definen un estilo arquitectónico para la construcción de software, éste fue desarrollado junto con el protocolo HTTP/1.1, el cual se adhiere a REST.

La arquitectura está compuesta por clientes y servidores, es decir peticiones y respuestas que son construidos en base a la idea de transferir representaciones de recursos. Los clientes contienen representaciones y los servidores contienen los recursos en sí. [19]

Los servicios Rest se caracterizan por ofrecer sencillez al momento de realizar cualquiera de sus funciones, que es realizada mediante una url que define la dirección del recurso al que se desea acceder.

Un Servicio Web REST tiene las siguientes características:

a) Las interfaces deben construirse sobre HTTP. Las siguientes funciones son definidas:

- **HTTP GET:** Usado para obtener una representación de un recurso. Un consumidor lo utiliza para obtener una representación desde una URI. Los servicios ofrecidos a través de este interfaz no deben contraer ninguna obligación respecto a los consumidores. (Ver figura 7)
- **HTTP DELETE:** Se usa para eliminar representaciones de un recurso. (Ver figura 7)
- **HTTP POST:** Usado para actualizar o crear las representaciones de un recurso. (Ver figura 7)
- **HTTP PUT:** Se usa para crear representaciones de un recurso. (Ver figura 7)

b) La mayoría de los mensajes son XML, definidos por un esquema XML.
c) Mensajes simples se pueden codificar en las URL.
d) Los servicios y los proveedores de servicios deben ser recursos, mientras que los consumidores pueden ser un recurso. [22]

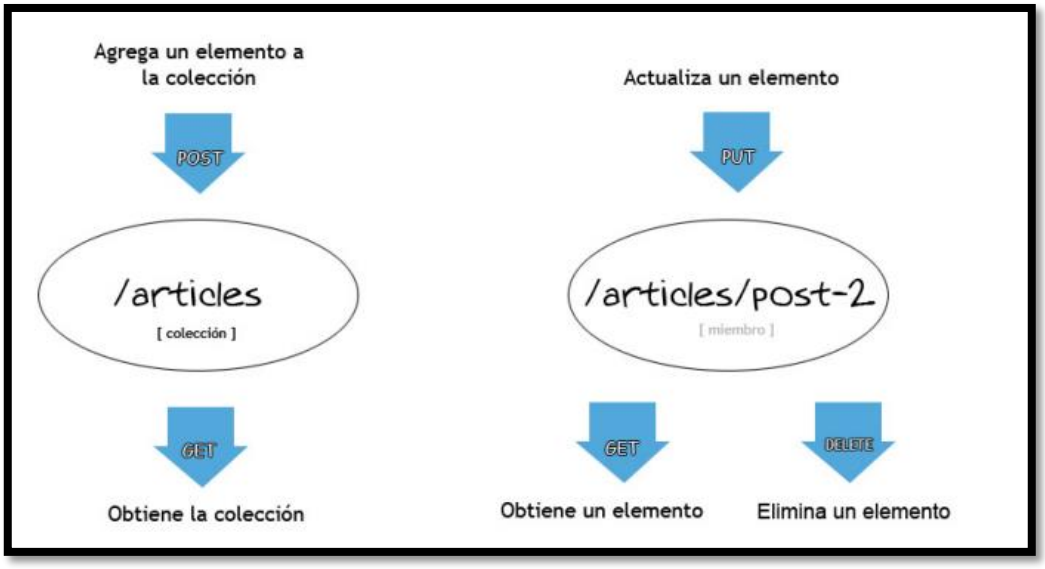


Figura 7 Funcionamiento de las interfaces de un Servicio Web Rest [22]

2.1.2. Diseño de un Web Service Rest

Para conseguir un web Service Rest se han determinado una serie de pasos que se debe seguir:

- Identificar todas las entidades conceptuales que se desean exponer como servicio.
- Crear una uri para cada recurso. Los recursos deberían ser nombres no verbos (acciones). Por ejemplo: http://www.service.com/entities/001
- Identificar la función a utilizar de acuerdo al tipo de acceso a los recursos que posea el cliente, es decir si obtiene únicamente una representación se usaría Http Get, y si puede modificar sería conveniente Http Post, Put y/o Delete.
- Asegurarse de que todos los recursos accesibles mediante GET no tengan efectos secundarios. Es decir, que no sean modificados luego de la invocación
- Especificar el formato de los datos de respuesta mediante un esquema (DTD, W3C Schema, entre otros). Para los servicios que requieran un POST o un PUT es aconsejable también proporcionar un esquema para especificar el formato de la respuesta.
- Describir como nuestro servicio ha de ser invocado, mediante un documento WSDL/WADL o simplemente HTML. [23]

2.2. Geolocalización para dispositivos móviles

Actualmente, la Geolocalización y otros servicios ofrecidos de google pueden ser utilizados en una aplicación móvil para obtener grandes resultados y dar una mejor ayuda al usuario.

2.2.1. Geolocalización

La Geolocalización se basa en el uso de la tecnología GPS que facilitan los satélites que orbitan alrededor de la Tierra, la geolocalización nos habla de situar a una persona, empresa u organización en un punto concreto del espacio. [24]

2.2.2. Google Maps

Google Maps, durante una época llamado Google Local, es un servidor de aplicaciones de mapas en web desarrollado con la tecnología de Google.

En sus inicios, ofrecía mapas de ciudades de diversos países (EEUU, Canadá, Reino Unido, Japón), en principio como complemento y ayuda al usuario que realiza búsquedas en Google Local.

Google Maps fue anunciado por primera vez en el Google Blog el 8 de Febrero de 2005. Originariamente era sólo soportado por los navegadores Internet Explorer y Mozilla Firefox. El soporte para Opera y Safari fue agregado el 25 de Febrero de ese mismo año. El software estuvo en su fase beta durante 6 meses antes de convertirse en parte de Google Local, el 6 de Octubre de 2005

En Abril de 2005 comenzó a ofrecer además, imágenes vía satélite provenientes de la firma Keyhole, adquirida por Google. De esta manera, podemos ver fotografías aéreas de todo el planeta de mayor o menor resolución dependiendo si se trata o no de importantes núcleos urbanos.

Todas las imágenes de satélite de Google Maps son las mismas que las que podemos encontrar en otra famosa herramienta de Google llamada Google Earth.

Del mismo modo que otras aplicaciones web desarrolladas por Google, para implementar Google Maps, se usan un gran número de ficheros JavaScript. Como el usuario puede mover el mapa, la visualización del mismo se descarga desde el servidor.

Cuando este busca un punto determinado, la ubicación está marcada por un indicador en forma de pin, el cual es una imagen PNG transparente sobre el mapa.

Para conseguir la conectividad sin sincronía con el servidor, y así proporcionar al usuario mayor interactividad con el mapa, mediante la realización de peticiones asíncronas a la red con JavaScript y XMLHttpRequest. [25]

2.2.3. Google Maps Api

Google Maps API se trata de una tecnología que permite la visualización de Google Maps en páginas web con JavaScript. El API proporciona unas determinadas herramientas para interaccionar con los mapas y añadir contenido a los mismos a través de una serie de servicios, permitiendo llegar a crear aplicaciones con mapas de gran complejidad y robustez.

Para implantar esta tecnología en una página web, google proporciona una clave “single Maps API key”, que es válida para un único directorio o dominio. Para obtener esta clave se debe tener una cuenta de google, y la clave que se proporciona estará conectada a dicha cuenta.

Google ofrece dos tipos de documentación: Una de ellas, está diseñada para permitir al nuevo usuario, empezar rápidamente a experimentar y desarrollar sencillas aplicaciones con Google Maps API. El otro tipo de documentación que se ofrece, se trata de una guía de referencia completa y exhaustiva: Google Maps API Reference. [25]

2.2.4. Place Autocomplete - Google Places API

Google Places es un servicio de Google gracias al cual hay como dar de alta y gestionar la información de la ubicación física de un negocio, además de otra información relacionada.

A través de las búsquedas locales en Google se puede acceder a la ficha de la empresa y sus datos de contacto, además de ubicarlo en el mapa para facilitar la localización. [26]

Gracias a la gran información de Google Places API puede ofrecer un servicio como el autocompletado de sitios que consiste en un servicio web que devuelve predicciones de sitios en respuesta a una solicitud HTTP.

La solicitud especifica una búsqueda textual y límites geográficos específicos. Este servicio se puede utilizar para proporcionar funciones de autocompletado para búsquedas geográficas basadas en texto, mostrando sitios como empresas, direcciones y lugares de interés a medida que el usuario escribe. [27]

2.2.5. Api de Rutas

El API de rutas de Google es un servicio que utiliza una solicitud HTTP para calcular rutas para llegar de una ubicación a otra. Se puedes buscar rutas de varios métodos de transporte, como en transporte público, en coche, a pie o en bicicleta. Las rutas pueden especificar los orígenes, los destinos y los hitos como cadenas de texto (por ejemplo, “Chicago, IL” o “Darwin, NT, Australia”) o como coordenadas de latitud/longitud. [28]

2.2.6. Navegation

Google Maps ofrece el servicio de navegación guiada por voz giro a giro, ya sea conduciendo o caminando desde la ubicación actual hacia un destino. [29]

Capítulo III

3. Metodologías para aplicaciones Móviles

En este capítulo se abordan las características principales de las metodologías para el desarrollo de aplicaciones móviles, las metodologías descritas son: Extreme Programing, Scrum para dispositivos móviles y Mobile-D

3.1. Metodología Extreme Programing (XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. [30] [31]

3.1.1. Fases de la metodología extreme programing

La metodología extreme programing (XP) posee las fases: planificación, diseño, desarrollo y pruebas.

3.1.1.1. Planificación

a. Se redactan las historias de usuarios.
b. Se crea un plan de entregas.
c. Se controla la velocidad del proyecto.
d. Se divide el proyecto en iteraciones.
e. Al comienzo de cada iteración se traza el plan de iteración
f. Se rota al personal
g. Cada día se convoca una reunión de seguimiento
h. Corregir la propia metodología XP cuando falla

3.1.1.2. Diseño

a. Realizar las cosas de la manera más sencilla
b. Coherencia de los nombres de todo lo que se va a implementar
c. Usar tarjetas CRC. En esta tarjeta se registra los nombres de las clases, sus responsabilidades y con qué otras clases colaboran
d. Soluciones puntuales para reducir riesgos
e. No se añadirá funcionalidad en las primeras etapas
f. Reaprovechar cuando sea posible

3.1.1.3. Desarrollo

a. El cliente está siempre disponible
b. Se debe escribir código de acuerdo a los estándares
c. Desarrollar la unidad de pruebas primero
d. Todo el código debe programarse por parejas
e. Sólo una pareja se encargará de integrar el código
f. Actualizar las versiones de los módulos lo más rápido posible
g. Todo el código es común a todos
h. Dejar las optimizaciones para el final

3.1.1.4. Pruebas

a. Todo el código debe ir acompañado de su unidad de pruebas
b. Todo el código debe pasar las unidades de pruebas antes de ser implantado
c. Crear una unidad de pruebas para protegerse del mismo
d. Se deben ejecutar pruebas de aceptación a menudo y publicar los resultados
e. Las pruebas unitarias [32]

3.2. Metodología Scrum para dispositivos móviles

Scrum es una metodología de desarrollo muy simple, que requiere trabajo duro, porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto, realizado por Hirotaka Takeuchi e Ikujijo Nonaka a mediados de los 80 y restructurado por Jeff Sutterland. [33]

Scrum es un marco de trabajo ágil que se basa en la iteración y entrega de incrementales de desarrollo de un producto o servicio. [34] Posee las siguientes características:

- Su prioridad es la satisfacción del cliente; que se da con la continua interacción entre éste y el equipo de desarrolladores.
- Se aceptan requisitos cambiantes.
- Enfocado a conseguir pequeños incrementos de software completamente funcionales
- Es un modo de desarrollo adaptable, antes que predictivo.
- Orientado a las personas, más que a los procesos.
- Emplea el modelo de construcción incremental basado en iteraciones y revisiones.
- Alta flexibilidad [35]

3.2.1. Fases de la Metodología Scrum para móviles

Scrum no posee las mismas fases que otras metodologías, se caracteriza por la realización de sprints con duración de dos o tres semanas. (Ver figura 8)

3.2.1.1. Planificación Inicial

Esta reunión sucede una única vez al inicio del proyecto. Se reúne el grupo de trabajo con el representante del cliente. Se determinan los requisitos iniciales y la visión del producto desde el punto de vista del negocio. Se consolida esta información en el Product Backlog [33]

3.2.1.2. Product Backlog

Su desarrollo parte de la visión del propietario del producto, la cual debe ser real, comprendida y compartida por parte de todo el equipo.

Es un inventario de funcionalidades, mejoras, tecnología y corrección de errores que deben incorporarse al producto a través de las sucesivas iteraciones del desarrollo. En él se documenta todo lo que implique un trabajo. Nunca está completo, se considera en permanente evolución. [34]

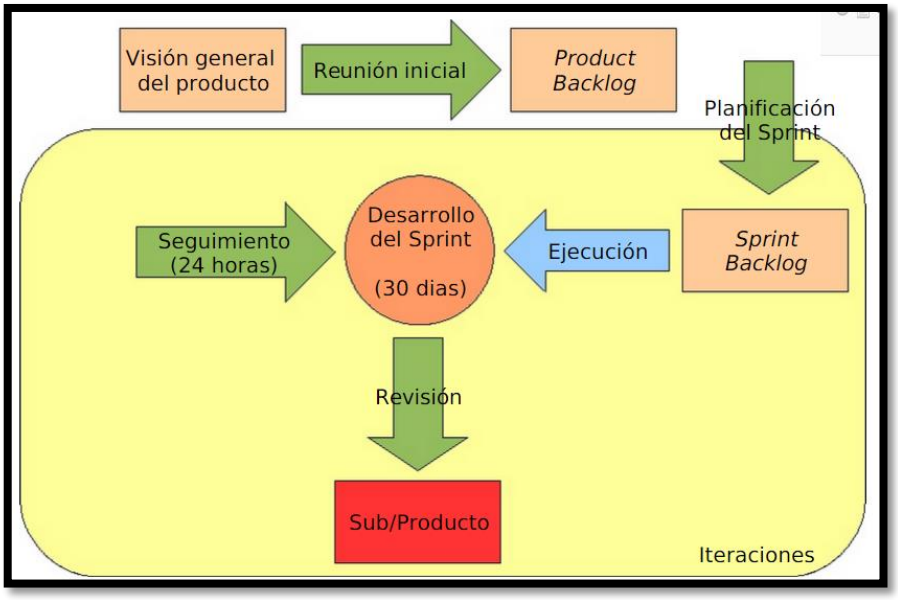


Figura 8 Ciclo de vida de Scrum para móviles [35]

3.2.1.3. Planificación del Sprint

Es gestionada por el Scrum manager. Se realiza en dos etapas. En total, su duración no deberá exceder un día de trabajo.

La primera etapa dura máximo 4 horas y en ella se definen cuales funcionalidades del Product Backlog se implementarán en la siguiente iteración. La segunda se descompone las funcionalidades elegidas en las tareas necesarias para implementar el incremento del producto.

Cada sprint, a excepción del primero, deberá generar un producto probado y funcional de interés para el cliente. Se auto-asignan las tareas a un integrante del equipo y se estiman los recursos necesarios para su desarrollo. El producto resultante de esta planificación es el Sprint Backlog. [35]

3.2.1.4. Sprint

Un Sprint en Scrum es el término que denomina a una iteración que está acotada en el tiempo, usualmente entre 2 y 4 semanas, durante la cual el Equipo trabaja. [35]

3.2.1.5. Sprint Backlog

Como se mencionó en la sección de la Planificación del sprint, el Sprint Backlog incluye un listado de las funcionalidades que se van a desarrollar durante la siguiente iteración, desglosado en tareas con recursos definidos y asignadas a un integrante del grupo de trabajo.

Incluye un objetivo del sprint al rededor del cual giran las funcionalidades a implementarse. Sólo los miembros del equipo de trabajo pueden modificarlo durante el sprint. Es visible para todos los miembros del equipo. [34]

3.2.1.6. Desarrollo del Sprint

Después de realizado el Sprint Backlog el sprint puede iniciar su desarrollo. Su duración no deberá exceder los 30 días. Es desarrollado por el grupo de trabajo de acuerdo a la planeación realizada anteriormente. Durante su desarrollo el Product Backlog se congela y no se aceptan modificaciones. Su ejecución es continuamente monitoreada. [34]

3.2.1.7. Revisión del Sprint

Al final de cada sprint se realiza una reunión en la que el equipo de trabajo presenta al propietario del producto y demás involucrados el incremento construido durante el sprint. Su duración es de máximo 4 horas. Su preparación es de máximo 1 hora.

Le permite al propietario del producto conocer de primera mano el estado actual del desarrollo del proyecto. El producto expuesto debe estar en su etapa final: terminado, probado y operando en el entorno del cliente (incremento).Puede incluir también documentación de usuario o técnica según se haya pactado.

Al final de la reunión se interroga individualmente a todos los asistentes para recabar impresiones, sugerencias de cambio y mejora, y su relevancia. El propietario del producto trata con los asistentes y con el equipo las posibles modificaciones en el Product Backlog. Se convoca para la reunión de planeación del próximo sprint. [35]

3.2.1.8. Retrospectiva del Sprint

En Scrum, la Retrospectiva del Sprint es una reunión facilitada por el Scrum Master en la cual los Miembros del equipo de Scrum discuten el Sprint que acaba de finalizar, y determinan qué podría cambiarse en el próximo Sprint para que sea más productivo y mejor. La Revision del Sprint se focaliza en "Qué" construye equipo, mientras que la Retrospectiva se centra en "Cómo" están construyendo el sistema.

Cualquier cosa que afecte cómo el equipo construye software está abierto para discutirse. Esto puede incluir: procesos, prácticas, comunicación, entorno, artefactos y herramientas. [33]

3.3. Metodología Mobile-D

El objetivo de este método es conseguir ciclos de desarrollo muy rápidos en equipos muy pequeños. Fue creado en un proyecto finlandés en 2005, pero sigue estando vigente. Basado en metodologías conocidas pero aplicadas de forma estricta como: extreme programming, Crystal Methodologies y Rational Unified Process.[36]

3.3.1. Fases de la Metodología Mobile-D

El ciclo del proyecto se divide en cinco fases: exploración, inicialización, productización, estabilización y prueba del sistema. (Ver figura 9.) En general, todas las fases (con la excepción de la primera fase exploratoria) contienen tres días de desarrollo distintos: planificación, trabajo y liberación. Se añadirán días para acciones adicionales en casos particulares (se necesitarán días para la preparación del proyecto en la fase de inicialización) [37]

3.3.1.1. Exploración

Se centra la atención en la planificación y a los conceptos básicos del proyecto. Aquí es donde hacemos una definición del alcance del proyecto y su establecimiento con las funcionalidades donde queremos llegar.

En la iniciación configuramos el proyecto identificando y preparando todos los recursos necesarios como hemos comentado anteriormente en esta fase la dedicaremos un día a la planificación y el resto al trabajo y publicación.

Subfases:

- Establecimiento de los stakeholders
- Definición del alcance
- Establecimiento del proyecto

Establecimiento de las partes interesadas: es una etapa en la que todos los grupos de interés relevantes - excluyendo el equipo del proyecto en sí - se necesita en el establecimiento, así como en las diferentes tareas del proyecto incipiente se definen con las funciones y los recursos pertinentes. Además del grupo de clientes (que se define en el patrón de tareas Establecimiento del cliente), los grupos de interés en Mobile-D pueden incluir, por ejemplo, del grupo de supervisión, gestión de proyectos, grupos de la arquitectura, y el proceso de los especialistas. Todos estos actores juegan un papel vital en las tareas posteriores de Explora fase y en la ejecución del proyecto.

Definición del Alcance: es una etapa en la que los objetivos y el alcance de la incipiente proyecto de desarrollo de software se definen y acordado por la partes interesadas grupos. Esto incluye temas como los requisitos (iniciales) para el producto y la línea de tiempo del proyecto.

Establecimiento del Proyecto: es una etapa de acordar las cuestiones ambientales del proyecto (físicas y técnicas), así como el personal necesario en el desarrollo de software (promotores y de apoyo). Además, las cuestiones de proceso se definen en esta etapa.

Salidas:

1) Los requisitos iniciales documentan en que se han definido los requisitos iniciales,
2) Plan del proyecto que incluye la línea de tiempo, el ritmo, las terminaciones, los recursos proyecciones / grupos de interés y sus responsabilidades,
3) Base descripción del proceso que incluye el proceso de la línea de base, actividades documentación, puntos de integración (con, por ejemplo, salidas proyecciones hardware), y
4) Plan de Medida,
5) Plan de capacitación y
6) Descripción de línea Architecture.

3.3.1.2. Inicialización

El propósito de esta fase es asegurar el éxito de las próximas fases del proyecto mediante la preparación y verificación de todas las cuestiones fundamentales del desarrollo a fin de que todo esté en plena disposición para la aplicación de los requisitos seleccionado por el cliente.

Entre los documentos obtenidos en esta etapa están:

- El plan de proyecto actualizado
- La primera versión de la Arquitectura del Software y la descripción del Diseño.
- Primera versión de Product Backlog.
- Documento de requisitos iniciales actualizado
- Notas de Desarrollo e Interfaz de usuario

3.3.1.3. Producción

Su objetivo es la implementación de las funcionalidades requeridas del producto mediante la aplicación de un ciclo de desarrollo iterativo e incremental.. Se repiten interativamente las subfases. Se usa el desarrollo dirigido por pruebas (TDD), antes de iniciar el desarrollo de una funcionalidad debe existir una pueba que verifique su funcionamiento. En esta fase podemos decir que se lleva a acabo toda la implementación.

Entre los documentos obtenidos se encuentran los siguientes:

- ✓ Funcionalidades implementadas
- ✓ Notas Notas de Desarrollo
- ✓ Ilustraciones acerca de la Interfaz de Usuario
- ✓ StoryBoards y StoryCards
- ✓ Documento de Requisitos actualizado

3.3.1.4. Estabilización

El propósito de esta fase es el aseguramiento de la calidad de la implementación del proyecto. Esta etapa es en la que se realizan las acciones de integración para enganchar los posibles módulos separados en una única aplicación.

Luego de culminada su duración se obtiene:

- La funcionalidad implementada de todo el software de proyecto,
- La documentación del producto finalizado.

3.3.1.5. Pruebas

Una vez parado totalmente el desarrollo se pasa una fase de testeo hasta llegar a una versión estable según lo establecido en las primeras fases por el cliente. Si es necesario se reparan los errores, pero no se desarrolla nada nuevo.

Una vez acabada todas las fases se debería tener una aplicación publicable y entregable al cliente. [36]

Figura 9. Ciclo de Desarrollo de Mobile-D [38]

3.4. Comparativa entre Metodologías para aplicaciones móviles

De acuerdo a cada una de las fases de las diferentes metodologías planteadas se realizó un cuadro comparativo descrito en la tabla III.

TABLA III

CUADRO COMPARATIVO DE METODOLOGÍAS PARA DESARROLLO DE APLICACIONES MÓVILES

	Metodología Extreme Programing	**Metodología Scrum para móviles**	**Metodología Mobile-D**
Fases	Fases: planificación, diseño, desarrollo y pruebas.	Se define por Sprints de dos semanas aproximadamente y de un producto entregable.	Fases: exploración, inicialización, productización, estabilización y prueba del sistema.
Comunicación con el cliente	Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo	El Product Owner maneja la comunicación con el cliente	La comunicación con el cliente es menor que en Scrum.
Programación	Programación en parejas, en jornadas largas; revisión de código mutuo.	El tiempo de programación se determina de acuerdo a la puntuación dada a cada tarea.	Es la parte más importante, y que mayor tiempo posee junto con las pruebas.
Documentación	Lo principal son las historias de usuario y tarjetas CRC.	Lo principal lo constituyen las historias de usuario.	Lo princiapal modelo del dominio, storycards y storyboards,
Pruebas	Se realizan unidades de pruebas, pruebas de aceptación y pruebas unitarias.	Se realizan pruebas unitarias pruebas de integración, pruebas de aceptación.	Se realizan pruebas unitarias, pruebas de interfaz, pruebas de datos ingresados y pruebas funcionales.

Luego de realizar la comparativa se determinó que las fases de las metodologías xp y mobile-d se parecen a las fases de análisis, diseño, codificación y pruebas; en cambio scrum se caracteriza por ser iterativo, se podría decir que realiza las fases de análisis, diseño, codificación y pruebas por cada iteración.

La documentación en la metodología xp y en scrum tiene un nivel bajo en comparación con la metodología mobile-d. Sin embargo mobile-d no exige una gran cantidad de documentación, lo necesario para guiarse en el desarrollo de la aplicación.

e. Materiales y Métodos

Para el presente trabajo de titulación se ha creído conveniente utilizar los materiales y métodos descritos a continuación.

1. Materiales

Los materiales utilizados para el desarrollo del presente trabajo han sido seleccionados bajo las comparativas realizadas en el apartado de la revisión bibliográfica, es así que se ha usado Eclipse ADT Bundle para el desarrollo de la aplicación móvil Android. El web Service del cual se obtienen los datos que utiliza una base de datos Mysql.

Otra herramientas utilizadas son Gantt Project para la realización del cronogramas y pencil Project para diseñar el Prototipado de pantallas, y para la realización de pruebas, jUnit para las unitarias y el recurso online apkudo para la comprobación de ejecución en diferentes dispositivos.

2. Métodos

Para el desarrollo del presente proyecto se utilizó diferentes métodos y técnicas que ayuden a lograr cumplir con los objetivos planteados.

2.1. Metodología

Se planteó la utilización de una metodología ágil que permita la elaboración de la aplicación móvil de una forma rápida y con un nivel medio de documentación. Es por ello, que se ha escogido la metodología Mobile-D.

2.2. Métodos

Método Deductivo: Fue utilizado para organizar la información de una forma racional e interactuante partiendo de lo general (efectos) hacia la particularidad o causa del problema que implica para el usuario encontrar un parqueadero en la ciudad de Loja; este proceso de búsqueda de la información partió de los siguientes interactuantes:

- Lluvia de ideas para la tesis

- Selección de la idea pertinente de la tesis: **“**Desarrollo de una aplicación móvil Android para la búsqueda de plazas disponibles en un parqueadero”,
- Desarrollo del Anteproyecto
- Desarrollo de la fase Análisis: Establecimiento de StakeHolders, Elaboración de Requerimientos, Definición del Alcance y Módulos.
- Desarrollo de la fase Diseño: Configuración del Ambiente de Desarrollo, Planificación Inicial, Planificación de fases, Modelo del dominio, StoryBoards y StoryCards.
- Desarrrollo de la fase de Codificación
- Desarrollo de Pruebas Unitarias, Datos Ingresados y Pruebas Funcionales.

Método Inductivo: Este método fue utilizado partiendo de la particularidad del proyecto o causa del mismo hacia la generalidad o efectos determinados en el método deductivo. Este proceso es pertinente ya que todo como **“**Desarrollo de una aplicación móvil Android para la búsqueda de plazas disponibles en un parqueadero”, se posiciona en un área social ambiental cuyas variables económicas- técnicas-culturales están interactuando e interrelacionándose entre sí y por esta razón se tuvo que partir de la solución del problema hacia la generalidad o efectos del proyecto para que se consolide el proyecto y brinde un servicio al usuario de la ciudad de Loja.

2.3. Técnicas

Encuesta: Con esta técnica se obtuvo información relevante y necesaria referente al tema de la investigación, la cual ayudara a sustentarlo y justificarlo.

Técnica de la investigación Bibliográfica: Con esta técnica se sustentara la base teórica de la investigación, mediante consultas a: fuentes bibliográficas confiables, libros, revistas indexadas, artículos científicos, base de datos científicas entre otras.

f. Resultados

Como una de las primeras instancias de la investigación se definió la hipótesis a comprobar gracias al desarrollo del presente trabajo. Además, se plantearon cuatro fases que ayuden en la realización de una investigación exitosa; a continuación, se describen los resultados obtenidos a lo largo del proceso.

1. HIPÓTESIS

Hipótesis: ¿La búsqueda de parqueaderos a través de una aplicación móvil permite el ahorro de tiempo al conductor?

En cada una de las fases se han realizado diferentes tareas con el fin de comprobar la interrogante planteada, las cuales se detallan a continuación.

2. PRIMERA FASE: Análisis

Con esta fase se busca realizar el análisis pertinente de la problemática y obtener los requerimientos del proyecto a desarrollar.

2.1. EXPLORACIÓN

En esta etapa se definen los requerimientos y alcance de la investigación, bases fundamentales para un adecuado desarrollo y exitoso producto.

2.1.1. Establecimiento de los Grupos de Interés o Stakeholders

Los grupos de personas interesadas en la realización de la presente investigación son los siguientes:

- **Desarrollador:** Es la persona encargada del análisis, desarrollo y pruebas de la aplicación; en este caso la tesista.
- **Conductores:** Son todas las personas que poseen un vehículo y un dispositivo móvil con Sistema Operativo Android.
- **Propietarios de Parqueaderos:** Son los dueños de los parqueaderos, quienes ofrecerán la información que alimente la base de datos de los parqueaderos.

2.1.2. Requerimientos Iniciales

Se pretende realizar una aplicación móvil para el sistema operativo Android que permita visualizar las plazas que se encuentren disponibles en un parqueadero. En base a ello se ha recolectado la información que permita conocer las necesidades de los usuarios de los parqueaderos.

2.1.2.1. Requerimientos Funcionales

Los requerimientos que el Sistema permitirá se encuentran en la tabla V.

TABLA IV.

REQUERIMIENTOS FUNCIONALES

Código	Descripción	Categoría
RF001	Al usuario visualizar su ubicación.	EVIDENTE
RF002	Marcar la ubicación de los parqueaderos	EVIDENTE
RF003	Al usuario buscar una dirección.	EVIDENTE
RF004	Al usuario marcar una ubicación alternativa	EVIDENTE
RF005	Al usuario visualizar la información de un parqueadero (razón social, dirección, plazas disponibles, plazas totales, distancia e imagen).	EVIDENTE
RF006	Al usuario visualizar los parqueaderos de la ciudad de Loja	EVIDENTE
RF007	Al usuario buscar un parqueadero por su razón social o dirección.	EVIDENTE
RF008	Al usuario buscar el parqueadero más cercano a su localización.	EVIDENTE
RF009	Al usuario buscar el parqueadero más cercano a una localización alternativa.	EVIDENTE
RF010	Calcular la distancia menor desde la ubicación del usuario o una ubicación alternativa hacia un parqueadero.	EVIDENTE
RF011	Trazar la ruta hacia el parqueadero más cercano.	EVIDENTE
RF012	Abrir la aplicación maps con la ruta hacia el parqueadero más cercano desde una ubicación alternativa.	EVIDENTE
RF013	Al usuario utilizar navigation para navegar desde su ubicación hacia el parqueadero más cercano.	EVIDENTE
RF014	Proporcionar información acerca de la aplicación	EVIDENTE

2.1.2.2. Requerimientos no Funcionales

Los requerimientos funcionales para la aplicación a realizar son descritos en la tabla VI.

TABLA V.

REQUERIMIENTOS NO FUNCIONALES

Código	Descripción
RNF01	El Sistema será desarrollado para la plataforma android.
RNF02	El Sistema posee una arquitectura cliente-servidor.
RNF03	El Sistema será desarrollado bajo la plataforma de programación Java.
RNF04	El sistema utilizará interfaz amigable.
RNF05	El sistema será multiusuario.
RNF06	El sistema permitirá el acceso a los usuarios sin necesidad de registrarse.

2.1.2.3. Análisis de los Requerimientos

En base a los Requerimientos establecidos se ha podido determinar los procesos a realizar en la tabla VII.

TABLA VI.

MÓDULOS Y PROCESOS DE LA APLICACIÓN

MÓDULO	CÓDIGO	PROCESO	REQUERI-MIENTOS
Módulo de Visualización de Parqueaderos	P001	Visualizar parqueaderos desde la ubicación del usuario.	RF001, RF002
	P002	Visualizar parqueaderos desde una ubicación alternativa	RF002, RF003, RF004
	P003	Visualizar información de un parqueadero	RF005
	P004	Visualizar los parqueaderos de la ciudad de Loja.	RF002, RF006
Módulo de Búsqueda de Parqueaderos	P005	Buscar un parqueadero por su razón social o dirección	RF007
	P006	Búsqueda de un parqueadero cercano	RF008,RF009, RF010,RF011, RF012, RF013
	P007	Visualizar información de la aplicación	RF014

2.1.3. Definición del Alcance

El alcance del proyecto de titulación se determina a través de su categoría, limitaciones y de los supuestos y dependencias.

2.1.3.1. Establecimiento de Categoría

La aplicación móvil es orientada por objetivos, de tipo empresarial de información [30], puesto que es necesario enviar y recibir información acerca de los parqueaderos y de la localización de manera individual.

2.1.3.2. Limitaciones

Las limitaciones de la aplicación son:

- La aplicación sólo puede ser ejecutada en dispositivos con el Sistema Operativo Android.
- Para la ejecución de la aplicación es necesario tener una conexión a internet y al gps.
- Para un correcto funcionamiento de la aplicación el dispositivo debe tener instalada la aplicación maps y navigation.

2.1.3.3. Establecimiento de Categoría

Los supuestos y dependencias de la aplicación son:

- La aplicación recoge datos de los parqueaderos a través de un Web Service
- Los datos obtenidos de la aplicación son únicamente acerca de la ciudad de Loja.
- Los usuarios manejan como idioma principal el español, siendo éste el lenguaje que se opera en la interfaz de la aplicación.

2.1.4. Establecimiento del Proyecto

En esta etapa se determinan los recursos físicos y técnicos necesarios para el desarrollo del proyecto. Las herramientas a utilizar son las siguientes:

Eclipse ADT Bundle: Eclipse ADT Bundle tiene el propósito de brindar un completo entorno de desarrollo para la codificación de aplicaciones Android al estar compuesto por el IDE Eclipse y el sdk de Android. Su licencia es libre.

BlueStacks: Esta herramienta permite emular un dispositivo android, de esta manera se comprueba que el desarrollo de la aplicación va de acuerdo a lo deseado. Su licencia es libre.

Pencil Project: Es una aplicación útil en el diseño de los prototipos de pantalla de sistemas web y de aplicaciones móviles. Su licencia es libre.

3. SEGUNDA FASE: Diseño

A través del diseño se desea obtener los diagramas que representen el funcionamiento de la aplicación, los procesos que va a ejecutar con el fin de cumplir los requerimientos definidos en la primera fase; además la descripción de cada uno de ellos y las configuraciones necesarias para desarrollar una aplicación móvil.

3.1. INICIALIZACIÓN

En esta etapa se realizan las actividades relacionadas a la configuración del ambiente de desarrollo y al diseño de la aplicación.

3.1.1. Configuración del Ambiente de Desarrollo

En esta actividad se debe realizar la configuración de los elementos físicos y técnicos, en este caso se realizó el entorno de desarrollo para aplicaciones android.

3.1.1.1. Configuración para aplicaciones móviles android

- **Tipo de Proyecto:** Android Application Project
- **Configuraciones:** inclusión de las librerías Sherlock Action Bar para el menú lateral, google maps para el uso de los servicios de google, universal-image-loader para la inclusión de imágenes externas a la aplicación.

3.1.2. Planificación Inicial

Como resultado de esta actividad se busca establecer los pre-requisitos de cada proceso

3.1.2.1. Análisis de Procesos y Pre-requisitos

Para la realización de los procesos es necesario antes cumplir con ciertos pre-requisitos con el fil de implementar la funcionalidad del proceso.

P001: Visualizar parqueaderos desde la ubicación del usuario:

- Registro, consulta, actualización y eliminación de parqueaderos.
- Marcar coordenadas de los parqueaderos en el mapa.
- Consulta y registro de la ubicación del usuario

P002: Visualizar parqueaderos desde una ubicación alternativa:

- Registro, consulta, actualización y eliminación de parqueaderos.
- Marcar coordenadas de los parqueaderos en el mapa.
- Consulta de coordenadas de la dirección alternativa
- Registro y marcado de la dirección alternativa en el mapa.

P003: Visualizar información de un parqueadero:

- Registro, consulta, actualización y eliminación de parqueaderos.
- Marcar coordenadas de los parqueaderos en el mapa.
- Establecer información de parqueaderos a ser mostrada.

P004: Visualizar los parqueaderos de la ciudad de Loja:

- Registro, consulta, actualización y eliminación de parqueaderos.
- Marcar coordenadas de los parqueaderos en el mapa.

P005: Buscar un parqueadero por su razón social o dirección:

- Registro, consulta, actualización y eliminación de parqueaderos.

- Registro, consulta, actualización y eliminación de parqueaderos en base de datos interna.
- Marcar coordenadas de los parqueaderos en el mapa.

P006: Búsqueda de un parqueadero cercano

- Registro, consulta, actualización y eliminación de parqueaderos.
- Marcar coordenadas de los parqueaderos en el mapa.
- Detección de tipo de ubicación: ubicación del usuario o una ubicación alternativa.
- Calculo de distancias entre los parqueaderos y la ubicación.

3.1.2.2. Planificación de fases

Las fases y las iteraciones que se darán en la investigación se reflejan en l tabla VIII, donde se explica las iteraciones que se realizan para el módulo de visualización de parqueaderos y para el módulo de búsqueda de parqueaderos en la etapa de producción y estabilización.

TABLA VII.

PLANIFICACIÓN DE FASES

Fase	Iteración	Descripción
Exploración	Iteración 0	Establecimiento de stakeholders, requerimientos iniciales, definición del alcance, establecimiento del proyecto.
Inicialización	Iteración 0	Configuración del ambiente de desarrollo, Planificación inicial, planificación de fases
Producción	Iteración Módulo de Visualización de Parqueaderos	Implementación del Módulo de Visualización de Parqueaderos, Refinamiento y actualización de storycards, Refinamiento de interfaces.
	Iteración Módulo de Búsqueda de Parqueaderos	Implementación del Módulo de Búsqueda de Parqueaderos, Refinamiento y actualización de storycards, Refinamiento de interfaces.
Estabilización	Iteración Módulo de Visualización de Parqueaderos	Refactorización del Módulo de Visualización de Parqueaderos, Refinamiento de interfaces, Generación y ejecución de pruebas de aceptación
	Iteración Módulo de Visualización de Parqueaderos	Refactorización del Módulo de Visualización de Parqueaderos, Refinamiento de interfaces, Generación y ejecución de pruebas de aceptación
Pruebas del Sistema	Iteración Pruebas del Sistema	Evaluación de Pruebas y Análisis de Resultados

3.1.3. Diseño del Sistema

El diseño general del sistema está compuesto por un servidor principal, un servidor de base de datos, el web Service, firewall y el usuario móvil.

En el servidor de base de datos se encuentra la base de datos administrada bajo el gestor de base de datos Mysql quien proporciona la información acerca de los parqueaderos al servidor principal que gestiona estos datos hacia el Web Service que utiliza GlassFish como servidor.

El web Service es el encargado de enviar las respuestas a las consultas hechas por la aplicación móvil acerca de la información de los parqueaderos. (Ver figura 10)

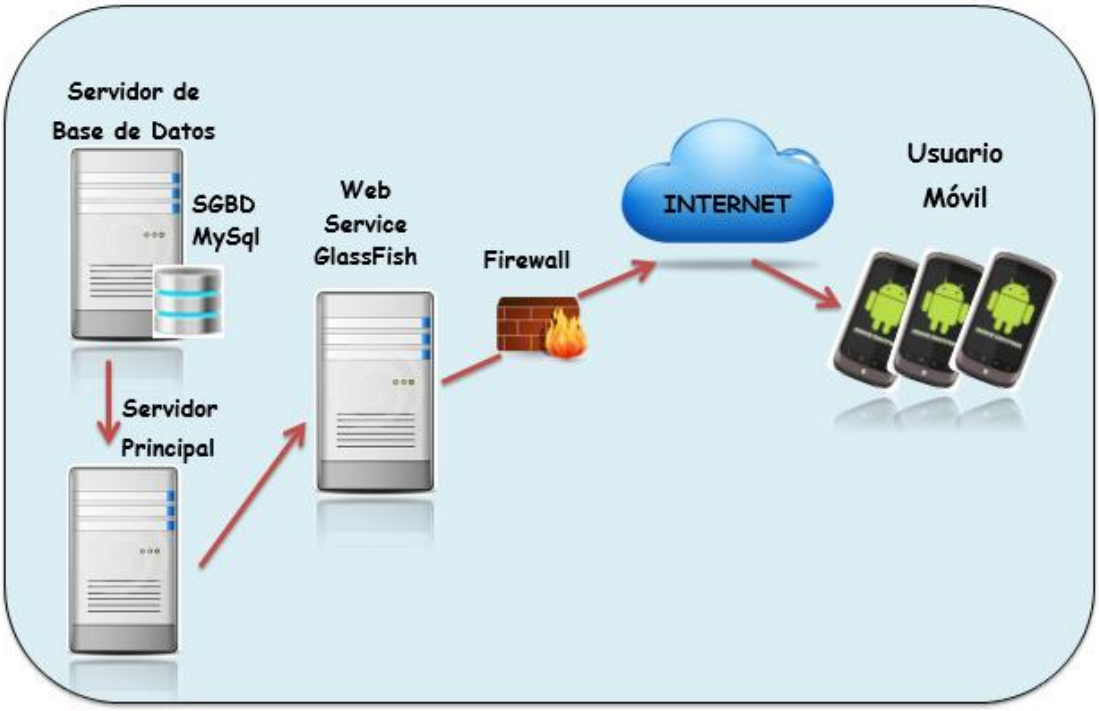


Figura 10. Diseño del Sistema

3.1.3.1. Modelo del Dominio

A continuación se detallan las clases que componen el Sistema de Parqueaderos que sirve de base para el desarrollo del presente proyecto de titulación.

Sin embargo, las clases fundamentalmente útiles para su funcionamiento son la clase Parqueadero y Plaza Alquilada, puesto que la clase Parqueadero contiene toda la información de los parqueaderos visualizados en el mapa y Plaza Alquilada posee los datos relativos a los alquileres de las plazas de cada parqueadero.

Las clases, atributos y sus relaciones son representados en el diagrama 1.

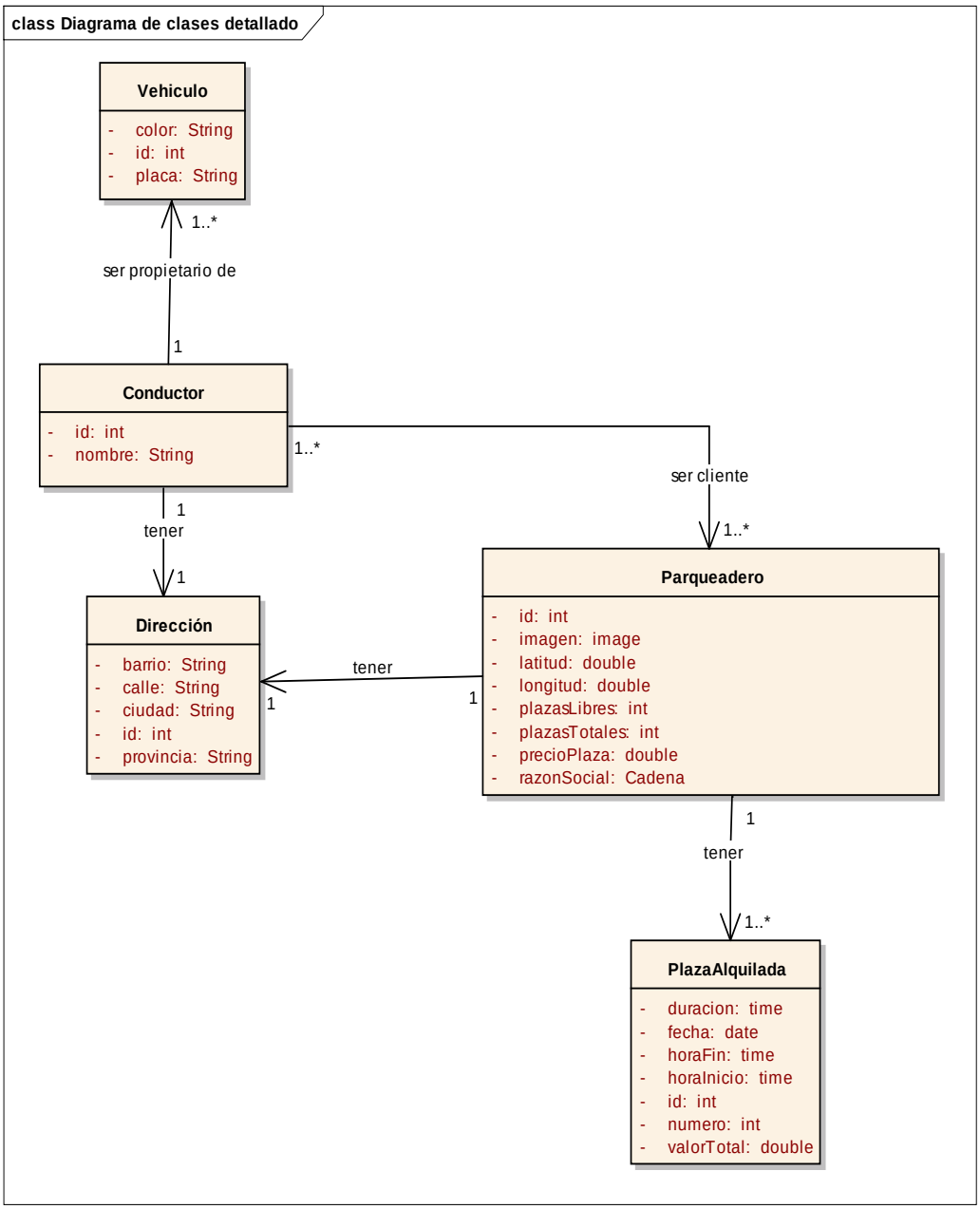


Diagrama 1. Modelo del Dominio

3.1.3.2. Modelo Entidad Relación

El Modelo Entidad Relación sirve para representar las entidades y relaciones existentes en un Sistema y cuyos datos necesitan ser almacenados para su funcionamiento. A continuación, el esquema diseñado para la base de datos del Sistema de Reservación de Parqueaderos que sirve como fuente de información de la aplicación desarrollada se observa en el diagrama 2.

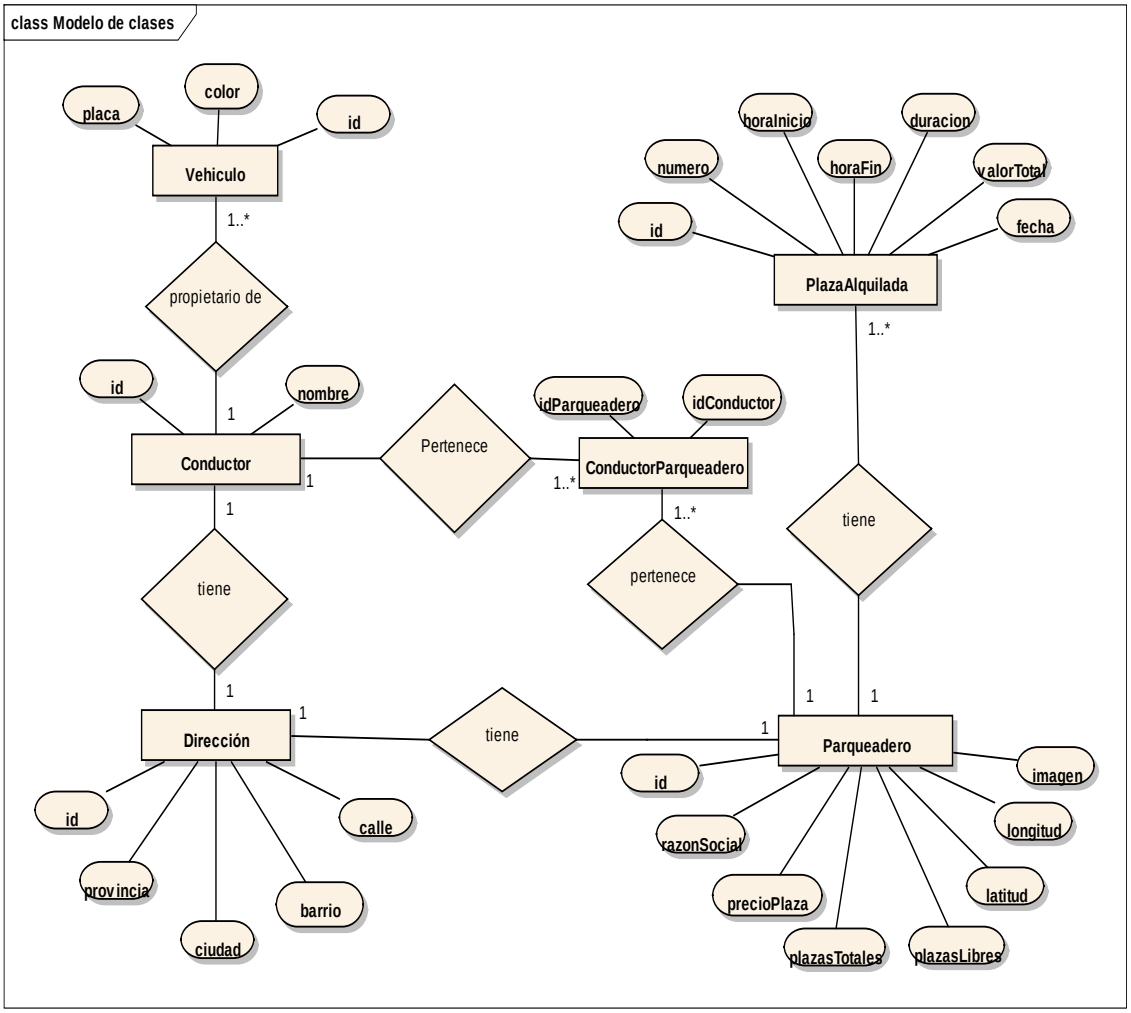


Diagrama 2. Modelo Entidad-Relación

3.1.3.3. Descripción de la Interfaz de Usuario

A continuación, se describe el esquema de navegabilidad del sistema o storyboard cuyo objetivo es describir la navegabilidad y conexiones entre las principales pantallas de la aplicación desarrollada.

La primera pantalla a visualizar es el splash con el logo de la aplicación, que dará paso a la pantalla principal dónde el usuario accederá directamente a la búsqueda de parqueaderos por su razón social, parqueadero más cercano y al menú de navegación lateral con las opciones de ubicación actual, ubicación alternativa, todos los parqueaderos y acerca de. (Ver diagrama 3)

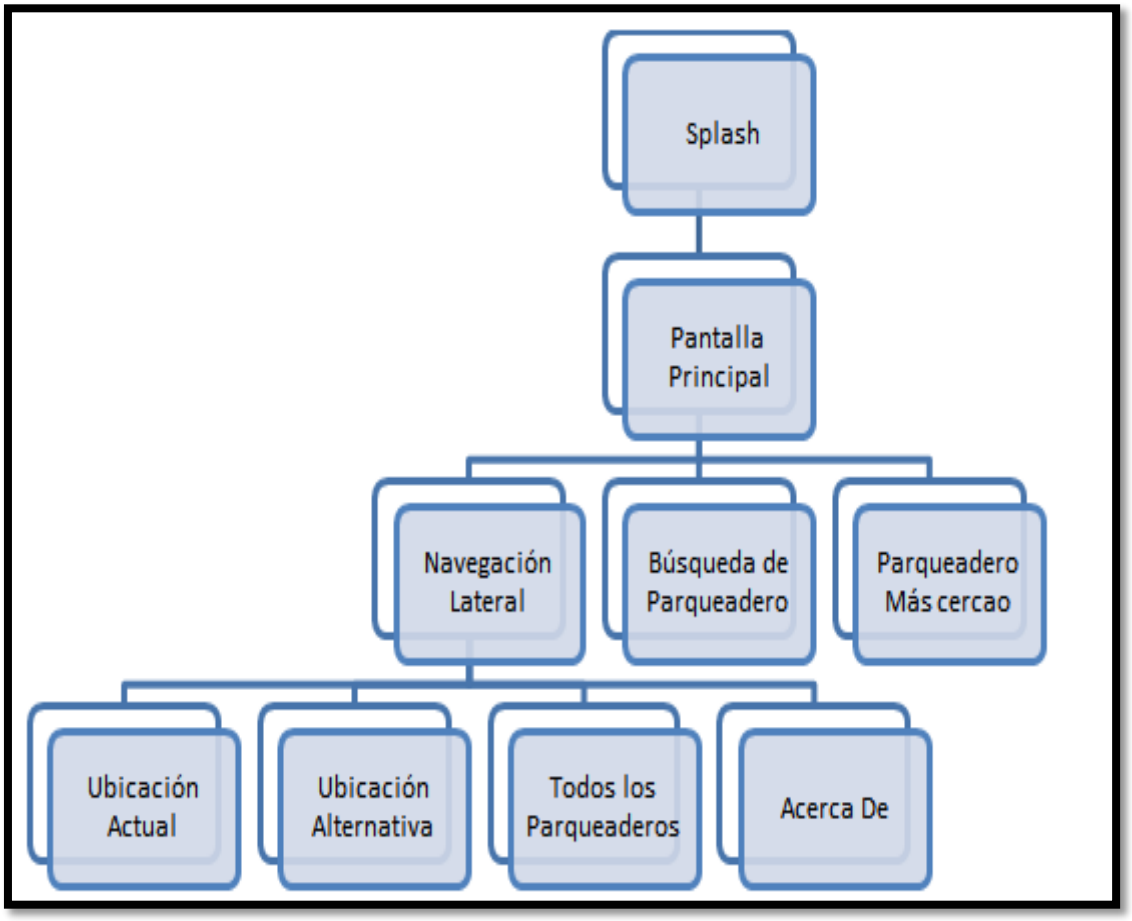


Diagrama 3. Storyboard de la Aplicación

- **Pantalla Principal**

La pantalla principal está formada por una barra de acción superior, un menú de navegación lateral y un mapa. Además, será presidida por un splash con el logo de la aplicación y un diálogo que informe acerca de la obtención de los parqueaderos. (Ver figura 11).

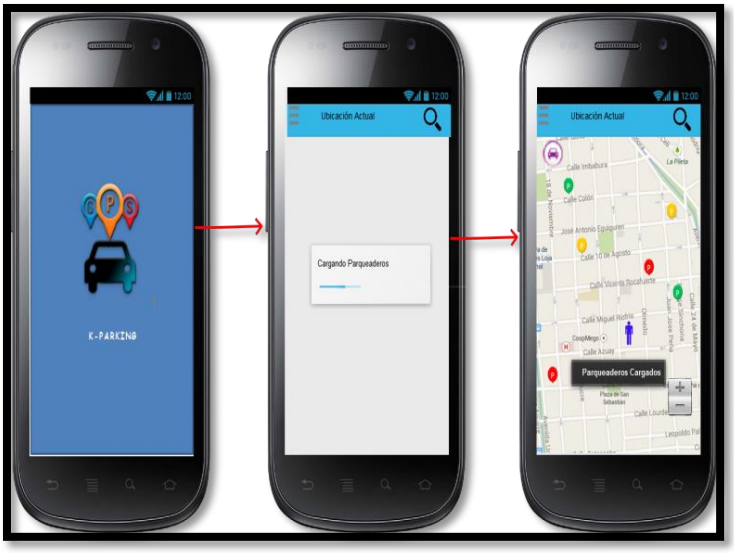


Figura 11. Prototipado de Pantalla: Pantalla Principal

A continuación, se describe el StoryCard del Prototipado de Pantalla: Pantalla Principal en la tabla 9. El storycard muestra una descripción del funcionamiento de la Pantalla Principal y de las fechas de su desarrollo.

TABLA VIII .

STORYCARD DE LA PANTALLA PRINCIPAL

Número /Id	Tipo	Dificultad		Esfuerzo		Prioridad	Nota
		Antes	Después	Estimado	Gastado		
01	**Nuevo** **Fijo** **Mejora**	**Fácil** **Moderado** **Duro**	**Fácil** **Moderado** **Duro**			**Baja** **Media** **Alta**	
Descripción							
Cuando el usuario ingresa a la aplicación se muestra el logo de la aplicación, se marcan los parqueadero en el mapa con su ubicación y con el color de icono de acuerdo al número de plazas libres.							
Excepciones							
Si el dispositivo no cuenta con conexión a internet se presenta un diálogo con un mensaje relativo al error y se obtienen los parqueaderos de la base de datos interna de la aplicación. Si el dispositivo no cuenta con google play services actualizado, la aplicación presenta un diálogo con un mensaje relativo al error. En caso de que exista un problema con el web Service se presenta un diálogo con un mensaje relativo al error y se obtienen los parqueaderos de la base de datos interna de la aplicación.							
Fecha	**Estado**	**Comentario**					
20/05/2014	**Definido**	Sin Comentarios					
24/06/2014	**Implementado**	Sin Comentarios					
21/08/2014	**Hecho**	Sin Comentarios					
10/10/2014	**Verificado**	Sin Comentarios					
	Pospuesto/ Cancelado/ Comparado	Sin Comentarios					

- **Menú de Navegación Lateral**

El Menú de Navegación Lateral, muestra una lista con las opciones: ubicación actual, ubicación alternativa, todos los parqueaderos y acerca de. (Ver figura 12).

Figura 12. Prototipado de Pantalla: Menú de Navegación Lateral

A continuación, se describe el StoryCard del Prototipado de Pantalla: Menú de Navegación Lateral en la tabla 10.

TABLA IX.

STORYCARD DEL MENÚ DE NAVEGACIÓN LATERAL

Número /Id	Tipo	Dificultad		Esfuerzo		Prioridad	Nota
		Antes	Después	Estimado	Gastado		
02	Nuevo Fijo Mejora	Fácil Moderado Duro	Fácil Moderado Duro			Baja Media Alta	
Descripción							
El usuario presiona el botón de navegación, y aparece un menú lateral con las opciones de ubicación actual, ubicación alternativa, todos los parqueaderos y acerca de. El título del action bar cambia de acuerdo a la opción escogida.							
Fecha	**Estado**	**Comentario**					
20/05/2013	**Definido**	Sin Comentarios					
24/06/2014	**Implementado**	Sin Comentarios					
21/08/2014	**Hecho**	Sin Comentarios					
10/10/2014	**Verificado**	Sin Comentarios					
	Pospuesto/ Cancelado/ Comparado	Sin Comentarios					

- **Ubicación Actual**

La pantalla ubicación actual está formada por un mapa y la ubicación del dispositivo marcada con el icono de google. (Ver figura 13).

En la pantalla también se visualizan los marcadores correspondientes a los parqueaderos

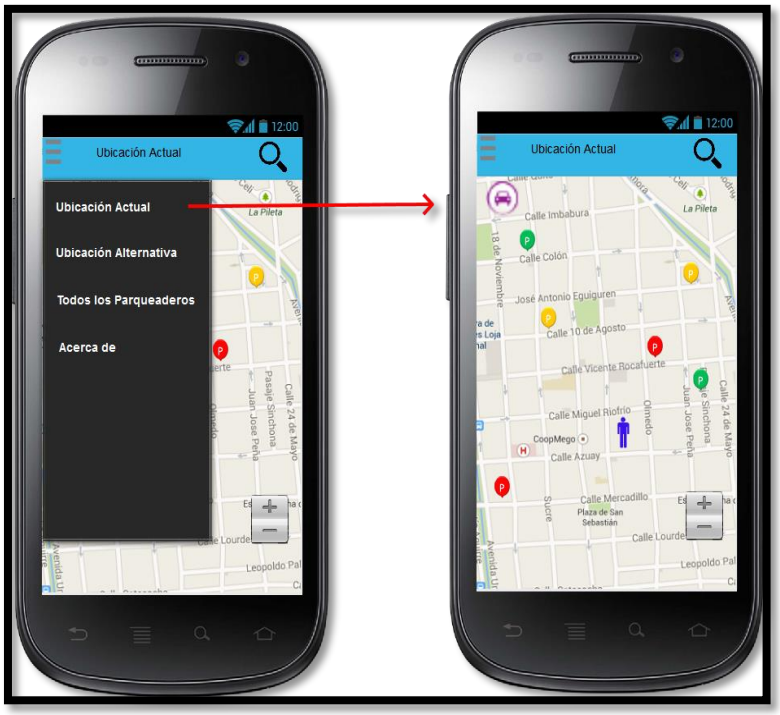


Figura 13. Prototipado de Pantalla: Ubicación Actual

A continuación, se describe el StoryCard del Prototipado de Pantalla: Ubicación Actual en la tabla 11.

TABLA X.

STORYCARD DE LA UBICACIÓN ACTUAL

Número /Id	Tipo	Dificultad		Esfuerzo		Prioridad	Nota
		Antes	Después	Estimado	Gastado		
03	**Nuevo** **Fijo** **Mejora**	**Fácil** **Moderado** **Duro**	**Fácil** **Moderado** **Duro**			**Baja** **Media** **Alta**	
Descripción							
Al presionar la opción de ubicación actual, el mapa ajusta su posición y se visualiza la ubicación del usuario en el centro. Se actualiza el tipo de ubicación a ubicación actual.							
Excepciones							
Si el dispositivo no cuenta con con conexión a internet se presenta un diálogo con un mensaje relativo al error y se presenta la pantalla principal							
Fecha	**Estado**	**Comentario**					
22/05/2014	**Definido**	Sin Comentarios					
21/07/2014	**Implementado**	Sin Comentarios					
21/08/2014	**Hecho**	Sin Comentarios					
10/10/2014	**Verificado**	Sin Comentarios					
	Pospuesto/ Cancelado/ Comparado	Sin Comentarios					

- **Ubicación Alternativa**

Esta pantalla está compuesta por un diálogo con la función de autocompletado; y un mapa con la dirección alternativa marcada con un icono. (Ver figura 14).

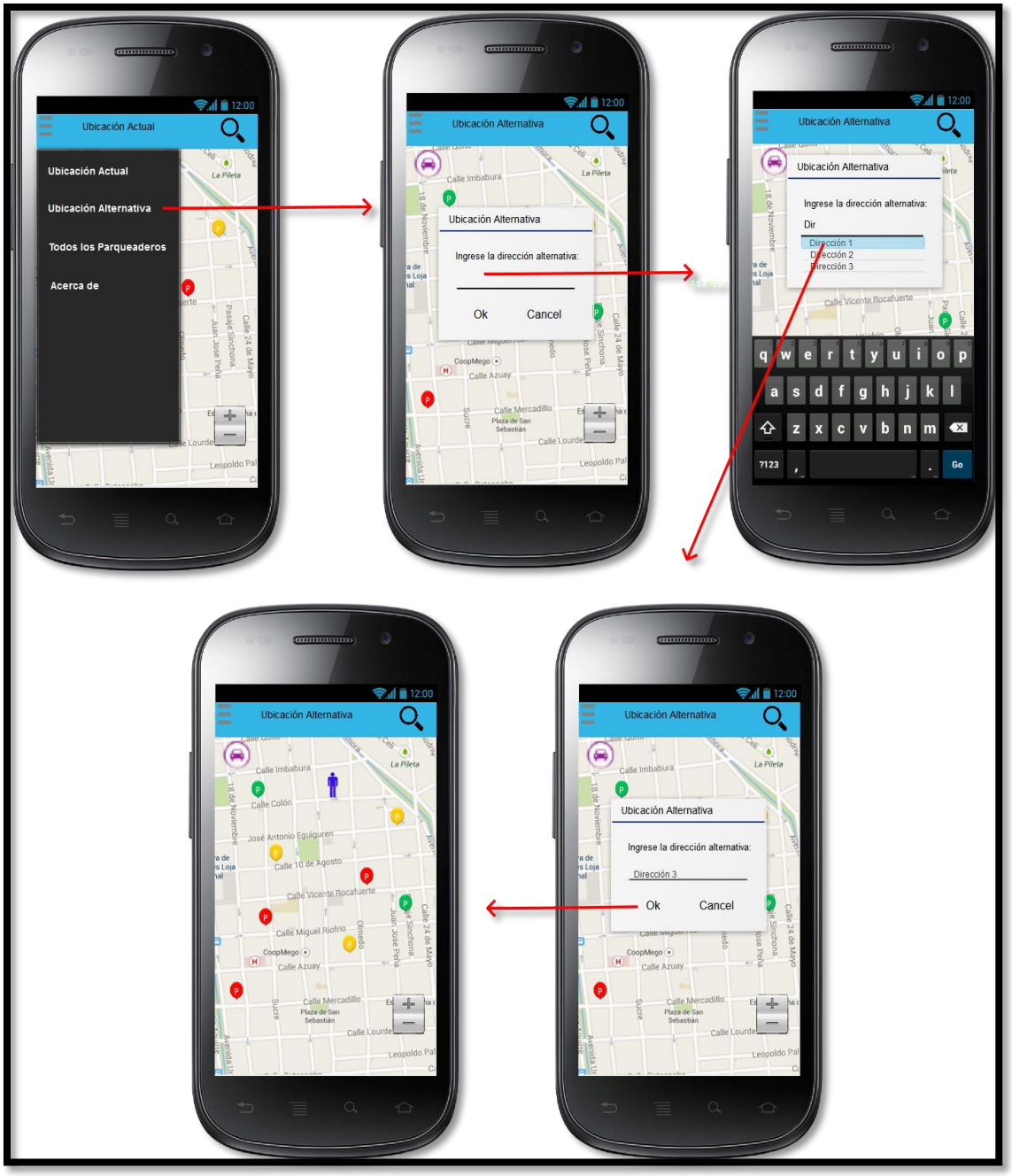


Figura 14. Prototipado de Pantalla: Ubicación Alternativa

A continuación, se describe el StoryCard del Prototipado de Pantalla: Ubicación Alternativa en la tabla 12.

TABLA XI.

STORYCARD DE LA UBICACIÓN ALTERNATIVA

Número /Id	Tipo	Dificultad		Esfuerzo		Prioridad	Nota
		Antes	Después	Estimado	Gastado		
04	**Nuevo** **Fijo** **Mejora**	**Fácil** **Moderado** **Duro**	**Fácil** **Moderado** **Duro**			**Baja** **Media** **Alta**	
Descripción							
Al presionar la opción de ubicación alternativa, el sistema presenta un diálogo con el servicio de autocompletado para que el usuario ingrese texto de la dirección a buscar y luego selecciona la opción requerida. La aplicación dibuja un marcador en la ubicación de la dirección seleccionada, en el mapa; se actualiza el tipo de ubicación a ubicación actual y se registra el valor de la ubicación seleccionada.							
Excepciones							
En caso de que el dispositivo no cuente con conexión a internet se presenta un diálogo con un mensaje relativo al error y se presenta la pantalla principal. En caso de que la dirección ingresada no sea encontrada o el texto esté vacío se presenta un mensaje relativo al error.							
Fecha	**Estado**	**Comentario**					
23/05/2014	**Definido**	Sin Comentarios					
21/07/2014	**Implementado**	Sin Comentarios					
21/08/2014	**Hecho**	Sin Comentarios					
10/10/2014	**Verificado**	Sin Comentarios					
	Pospuesto/ Cancelado/ Comparado	Sin Comentarios					

- **Todos los parqueaderos**

Esta pantalla está compuesta por un mapa y todos los parqueaderos de la ciudad de Loja marcados. (Ver figura 15).

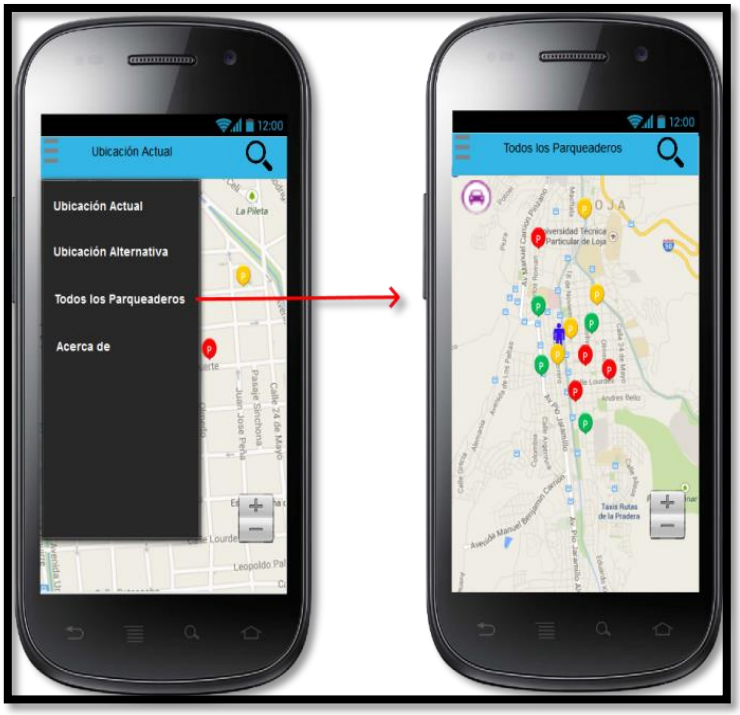


Figura 15. Prototipado de Pantalla: Todos los Parqueaderos

A continuación, se describe el StoryCard del Prototipado de Pantalla: Todos los parqueaderos en la tabla 13.

TABLA XII.

STORYCARD DE TODOS LOS PARQUEADEROS

Número /Id	Tipo	Dificultad		Esfuerzo		Prioridad	Nota
		Antes	Después	Estimado	Gastado		
05	**Nuevo** **Fijo** **Mejora**	**Fácil** **Moderado** **Duro**	**Fácil** **Moderado** **Duro**			**Baja** **Media** **Alta**	
Descripción							
Al presionar la opción de todos los parqueaderos el mapa se visualiza con un zoom menor para que el usuario tenga una mejor visión de los parqueaderos en general.							
Fecha	**Estado**	**Comentario**					
26/05/2014	**Definido**	Sin Comentarios					
24/06/2014	**Implementado**	Sin Comentarios					
21/08/2014	**Hecho**	Sin Comentarios					
10/10/2014	**Verificado**	Sin Comentarios					
	Pospuesto/ Cancelado/ Comparado	Sin Comentarios					

- **Acerca de**

La pantalla acerca de es un diálogo con información de la aplicación. (Ver figura 16).

Figura 16. Prototipado de Pantalla: Acerca De

A continuación, se describe el StoryCard del Prototipado de Pantalla: Acerca de en la tabla 14.

TABLA XIII.

STORYCARD DE ACERCA DE

Número /Id	Tipo	Dificultad		Esfuerzo		Prioridad	Nota
		Antes	Después	Estimado	Gastado		
06	**Nuevo** **Fijo** **Mejora**	**Fácil** **Moderado** **Duro**	**Fácil** **Moderado** **Duro**			**Baja** **Media** **Alta**	
Descripción							
Al presionar la opción de acerca de se visualiza un diálogo con información acerca de la Aplicación.							
Fecha	**Estado**	**Comentario**					
26/05/2014	**Definido**	Sin Comentarios					
24/06/2014	**Implementado**	Sin Comentarios					
21/08/2014	**Hecho**	Sin Comentarios					
10/10/2014	**Verificado**	Sin Comentarios					
	Pospuesto/ Cancelado/ Comparado	Sin Comentarios					

- **Información de un parqueadero**

Esta pantalla la compone una ventana de información con los principales datos del parqueadero. (Ver figura 17).

Figura 17. Prototipado de Pantalla: Información de un parqueadero

A continuación, se describe el StoryCard del Prototipado de Pantalla: Información de un parqueadero en la tabla 15.

TABLA XIV.

STORYCARD DE INFORMACIÓN DE UN PARQUEADERO

Número /Id	Tipo	Dificultad		Esfuerzo		Prioridad	Nota
		Antes	Después	Estimado	Gastado		
07	**Nuevo** **Fijo** **Mejora**	**Fácil** **Moderado** **Duro**	**Fácil** **Moderado** **Duro**			**Baja** **Media** **Alta**	
Descripción							
Al presionar sobre un marcador que represente a un parqueadero se visualiza información como su razón social, dirección, número de plazas totales, plazas libres y una imagen que represente su local.							
Excepciones							
En caso de que el parqueadero no cuente con una imagen se presenta el logo de la aplicación.							
Fecha	**Estado**	**Comentario**					
27/05/2014	**Definido**	Sin Comentarios					
21/07/2014	**Implementado**	Sin Comentarios					
21/08/2014	**Hecho**	Sin Comentarios					
10/10/2014	**Verificado**	Sin Comentarios					
	Pospuesto/ Cancelado/ Comparado	Sin Comentarios					

- **Búsqueda de parqueadero**

La búsqueda de un parqueadero se realiza a través de una opción de búsqueda en la barra superior de la pantalla principal, ésta ofrece el servicio de autocompletado. El parqueadero es marcado en el mapa con un círculo. (Ver figura 18).

Figura 18. Prototipado de Pantalla: Búsqueda de un Parqueadero

A continuación, se describe el StoryCard del Prototipado de Pantalla: Búsqueda de un parqueadero en la tabla 16.

TABLA XV.

STORYCARD DE BÚSQUEDA DE UN PARQUEADERO

Número /Id	Tipo	Dificultad		Esfuerzo		Prioridad	Nota
		Antes	Después	Estimado	Gastado		
08	**Nuevo** **Fijo** **Mejora**	**Fácil** **Moderado** **Duro**	**Fácil** **Moderado** **Duro**			**Baja** **Media** **Alta**	
Descripción							
Cuando el usuario presiona la lupa del action bar se habilita un campo de texto para ingresar la razón social o nombre de una calle de la dirección del parqueadero que se desea buscar. Al encontrar el parqueadero y enviar la consulta se marca una circunferencia alrededor del parqueadero buscado junto con un mensaje; además, el mapa se ajusta para una mejor visualización. La aplicación ofrece el servicio de autocompletado para la razón social de los parqueaderos y utiliza la información de su base de datos interna.							
Excepciones							
En caso de que la razón social del parqueadero ingresado no se encuentre o el texto esté vacío se presenta un mensaje relativo al error.							
Fecha	**Estado**	**Comentario**					
27/05/2014	**Definido**	Sin Comentarios					
15/08/2014	**Implementado**	Sin Comentarios					
21/08/2014	**Hecho**	Sin Comentarios					
10/10/2014	**Verificado**	Sin Comentarios					
	Pospuesto/ Cancelado/ Comparado	Sin Comentarios					

3.1.3.3.1. Búsqueda de un parqueadero cercano

Al hacer uso de esta función el usuario podrá encontrar el parqueadero más cercano en cuanto a su ubicación o una ubicación alternativa. (Ver figura 19).

Figura 19. Prototipado de Pantalla: Búsqueda de un parqueadero cercano

A continuación, se describe el StoryCard del Prototipado de Pantalla: Búsqueda de un parqueadero cercano en la tabla 17.

TABLA XVI.

STORYCARD DE BÚSQUEDA DE UN PARQUEADERO CERCANO

Número /Id	Tipo	Dificultad		Esfuerzo		Prioridad	Nota
		Antes	Después	Estimado	Gastado		
09	**Nuevo** **Fijo** **Mejora**	**Fácil** **Moderado** **Duro**	**Fácil** **Moderado** **Duro**			**Baja** **Media** **Alta**	
Descripción							
El usuario presionará el icono de búsqueda de parqueaderos cercanos y a continuación se mostrará un diálogo con la razón social y distancia en metros del parqueadero más cercano para confirmar el trazado de la ruta hacia a la ubicación del usuario o a una ubicación alternativa, esto de acuerdo a la opción seleccionada del menú lateral. Luego de encontrar el parqueadero más cercano y trazar la ruta se visualizará un nuevo diálogo para confirmar el acceso a la aplicación maps en caso de una ubicación alternativa o la aplicación Navigation en caso de que la ubicación sea la del usuario.							
Excepciones							
En caso de que el dispositivo no cuente con conexión a internet al momento de trazar la ruta se presentará un diálogo con un mensaje relativo al error y se presentará la pantalla principal. En caso de que la aplicación maps no se encuentre instalada en el dispositivo, se presentará un diálogo con un mensaje relativo al error y se presentará la pantalla principal con la ruta trazada hacia el parqueadero. En caso de que la aplicación navigation no se encuentre instalada en el dispositivo, se presentará un diálogo con un mensaje relativo al error y se presentará la pantalla principal con la ruta trazada hacia el parqueadero.							
Fecha	**Estado**	**Comentario**					
28/05/2014	**Definido**	Sin Comentarios					
15/08/2014	**Implementado**	Sin Comentarios					
21/08/2014	**Hecho**	Sin Comentarios					
10/10/2014	**Verificado**	Sin Comentarios					
	Pospuesto/ Cancelado/ Comparado	Sin Comentarios					

4. TERCERA FASE: Codificación

En esta fase se realiza el desarrollo de las funcionalidades planteadas en el diseño mediante un lenguaje de programación, para aplicaciones móviles Android se utiliza java; además se plantean los estándares de codificación y la estructuración de la aplicación.

4.1. PRODUCCIÓN Y ESTABILIZACIÓN

En la producción se realiza la codificación de los procesos diseñados y la posterior integración entre cada una de ellas para obtener el total funcionamiento de la aplicación.

4.1.1. Estándares de Codificación

- **Paquetes:** Los paquetes están escritos en minúsculas.
- **Clases:** Las clases están escritas con la primera letra en mayúscula, si se compone de dos o más palabras la primera letra de cada una de ellas está en mayúscula.
- **Variables:** Las variables tienen un nombre relacionado a su valor, están escritas con la primera letra en minúscula, si se compone de dos o más palabras la primera letra de ellas está en mayúscula, a excepción de la primera palabra.
- **Métodos:** Los métodos llevan un nombre relacionado al proceso que ejecutan, están escritos con la primera letra en minúscula, si se compone de dos o más palabras la primera letra de ellos está en mayúscula, a excepción de la primera palabra.
- **Layouts:** Los layouts tienen todo su nombre en minúsculas, si poseen más de dos palabras tendrán un sub-bajo para separar cada una de ellas.

4.1.2. Estructura de Directorios

A continuación, en el diagrama 4, se describe la estructura de la aplicación y una breve descripción de los directorios más importantes.

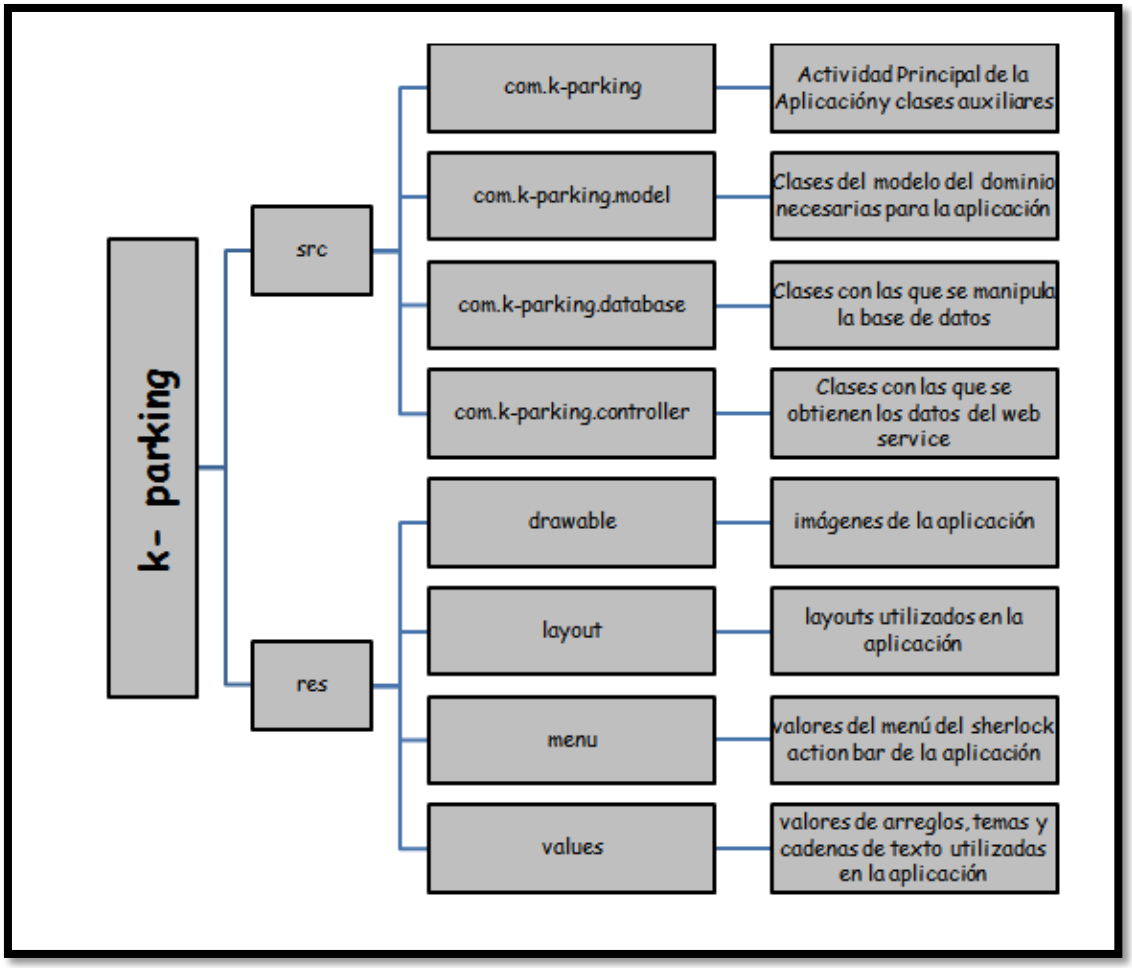


Diagrama 4. Estructura de Directorios de la Aplicación

4.1.3. Codificación

La codificación se realizó basándose en cada uno de los storycards realizados en la fase anterior.

4.1.3.1. Pantalla Principal y Menú de Navegación Lateral

Para la presentación de la pantalla principal se realizaron un conjunto de procesos como la construcción del splash, la configuración de la librería sherlock action bar, obtención de los datos de los parqueaderos a través del web service, almacenamiento de los parqueaderos en una base de datos interna, marcación de los parqueaderos y obtención de la ubicación del usuario.

- **Construcción del Splash**

Para la creación del Splash se creó una actividad llamada SplashScreenActivity y se utilizó algunas clases y métodos como las siguientes:

setRequestedOrientation(): Este método sirve para fijar una orientación al dispositivo, en este caso ActivityInfo.SCREEN_ORIENTATION_PORTRAIT se utiliza para el modo retrato; permite que el dispositivo no cambie de orientación mientras se carga el splash y así éste no se reinicie con algún movimiento del dispositivo. (Ver figura)

requestWindowFeature(): Este método es utilizado con el fin de que la imagen del splash ocupe toda la pantalla del dispositivo. (Ver figura)

TimerTask: Sirve para ejecutar una tarea durante un tiempo determinado, a través de esta clase permite que el splash se ejecute durante 3000 milisegundos. (Ver figura)

```
public class SplashScreenActivity extends Activity {
    private static final long SPLASH_SCREEN_DELAY = 3000;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
       // Orientación retrato
      setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        // Esconder título del actionbar
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.splash_screen);
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
                // Empezar MainActivity
                Intent mainIntent = new
Intent().setClass(SplashScreenActivity.this, MainActivity.class);
                startActivity(mainIntent);
                finish();
            }
        };
        Timer timer = new Timer();
        timer.schedule(task, SPLASH_SCREEN_DELAY);
    }
}
```

Figura 20 . Construcción del Splash

- **Configuración de la Librería SherlockActionBar para la creación del menú de navegación lateral.**

La configuración de la librería se la realizó en la actividad principal (Main.Activity), dentro de la cual se utilizaron varios métodos (Ver figura 21).

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   //Comprobación de Conexión de Internet
    …
   setContentView(R.layout.drawer_layout);
   contexto = this;
   title = drawerTitle = getTitle();
   //asignación de las opciones del menú lateral que a su vez servirán
   //como título del sherlockactionbar al momento de ser seleccionados
   titulosSherlockActionBar =
   getResources().getStringArray(R.array.drawer_menu);
   drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
   drawerList = (ListView) findViewById(R.id.left_drawer);
   //asignación de un adaptador a la lista del menú lateral.
   drawerList.setAdapter(new ArrayAdapter<String>(this,
   R.layout.drawer_list_item, titulosSherlockActionBar));
   //Comportamiento al abrir y cerrar el menu de navegación lateral
   drawerList.setOnItemClickListener(new DrawerItemClickListener());
   getSupportActionBar().setDisplayHomeAsUpEnabled(true);
   getSupportActionBar().setHomeButtonEnabled(true);
   drawerToggle = new ActionBarDrawerToggle(this,drawerLayout,
   R.drawable.ic_drawer, R.string.drawer_open, R.string.drawer_close ) {
         public void onDrawerClosed(View view) {
               getSupportActionBar().setTitle(title);
               supportInvalidateOptionsMenu();
         }
         public void onDrawerOpened(View drawerView) {
               getSupportActionBar().setTitle(drawerTitle);
               supportInvalidateOptionsMenu();
         }
   };
   drawerLayout.setDrawerListener(drawerToggle);
   …
   if (savedInstanceState == null) {
         selectItem(0);
   }
}
```

Figura 21. Configuración de la Librería SherlockActionBar: Método OnCreate.

onCreate(): Dentro de este método se configuró la asignación del layout (Ver figura), la asignación de un adaptador de listas para las opciones del menú lateral y el comportamiento al abrir y cerrar el menú lateral. (Ver figura 22)

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <!-- El contenido principal -->
    <FrameLayout
        android:id="@+id/frame_principal"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
        <!-- Mapa -->
        <fragment
            android:id="@+id/map"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            class="com.google.android.gms.maps.SupportMapFragment" />
        <!-- Botón de búsqueda -->
        <LinearLayout
            android:id="@+id/ll_EntradaF1"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:orientation="horizontal" >
            <Button
                android:id="@+id/btnparqueaderocercano"
                android:layout_width="72dp"
                android:layout_height="72dp"
                android:drawableTop="@drawable/parqueadero_cercano"
                android:onClick="buscarParqueaderoCercano"
                android:src="@drawable/parqueadero_cercano"
                android:textColor="#100719"
                android:textSize="15sp"
                android:textStyle="bold" />
        </LinearLayout>
    </FrameLayout>
    <!-- El menú lateral -->
    <ListView
        android:id="@+id/left_drawer"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:background="#555"
        android:choiceMode="singleChoice"
        android:divider="@android:color/transparent"
        android:dividerHeight="0dp" />
</android.support.v4.widget.DrawerLayout>
```

Figura 22. Configuración de la Librería SherlockActionBar: Layout drawer_layout

onCreateOptions() y onPrepareOptions(): Se configura la barra superior del sherlockactionbar. (Ver figura 23)

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
	MenuInflater inflater = getSupportMenuInflater();
	inflater.inflate(R.menu.main, menu);
	…
	return super.onCreateOptionsMenu(menu);
}
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
	boolean drawerOpen = drawerLayout.isDrawerOpen(drawerList);
	menu.findItem(R.id.menu_search).setVisible(!drawerOpen);
	return super.onPrepareOptionsMenu(menu);
}
@Override
public boolean onOptionsItemSelected(final MenuItem item) {
	if (drawerToggle.onOptionsItemSelected(getMenuItem(item))) {
		return true;
	}
}
private android.view.MenuItem getMenuItem(final MenuItem item) {
	return new android.view.MenuItem() {
		…
		…
	};
}
private class DrawerItemClickListener implements
	ListView.OnItemClickListener {
	@Override
	public void onItemClick(AdapterView<?> parent, View view, int
	position, long id) {
		selectItem(position);
	}
}
private void selectItem(int position) {
	Fragment fragment;
	if (position == 0) {
		…
	} else {
		…
	}
	drawerList.setItemChecked(position, true);
	setTitle(titulosSherlockActionBar[position]);
	drawerLayout.closeDrawer(drawerList);
}
```

Figura 23 . Configuración de la Librería SherlockActionBar: Barra Superior y Menú de Navegación Lateral (a)

onOptionsItemSelected(): Comprueba si un item del menu lateral ha sido seleccionado. (Ver figura 23)

ListView.OnItemClickListener: Esta interface sirvió para configurar el comportamiento en caso de que algún ítem se seleccione y recoger cuál de ellos fue. (Ver figura 23)

selectItem(): método creado para describir el funcionamiento de la aplicación de acuerdo a cada ítem seleccionado. (Ver figura 23)

setTitle(): Fue utilizado para colocar el título a la barra superior del sherlockactionbar. (Ver figura 24).

onConfigurationChanged(): con este método se otorga la nueva configuración de la aplicación en caso de que esta cambie. (Ver figura 24).

```
@Override
public void setTitle(CharSequence titulo) {
	title = titulo;
	getSupportActionBar().setTitle(title);
}
@Override
protected void onPostCreate(Bundle savedInstanceState) {
	super.onPostCreate(savedInstanceState);
	//Sincronización del estado luego de que un onRestoreInstanceState
	ha ocurrido
	drawerToggle.syncState();
}

@Override
public void onConfigurationChanged(Configuration newConfig) {
	super.onConfigurationChanged(newConfig);
	drawerToggle.onConfigurationChanged(newConfig);
}
```

Figura 24 . Configuración de la Librería SherlockActionBar: Barra Superior y Menú de Navegación Lateral (b)

- **Obtención de los datos de los parqueaderos a través del web Service**

Para poder obtener los datos del web service fue necesario llamar a un método dentro de la primera línea del onCreate() de la actividad principal (Ver figura), este método llama a una tarea que se repite cada determinado tiempo (300000 milisegundos)y a su

vez es la encargada de actualizar los datos de los parqueaderos y demás actividades necesarias antes de que pueda funcionar adecuadamente la aplicación. (Ver figura 25).

```
//Método para actualizar cada 5 minutos los datos de los parqueaderos
 public void actualizarParqueaderosWebService(){
       actualizarParqTask = new TimerTask() {
            public void run() {
                  handler.post(new Runnable() {
                        public void run() {
                              pDialog = new
                              ProgressDialog(MainActivity.this);
                              pDialog.setProgressStyle(ProgressDialog.S
                              TYLE_HORIZONTAL);
                              pDialog.setMessage("Actualizando datos de
                              los parqueaderos");
                               pDialog.setCancelable(false);
                               pDialog.setMax(100);
                              ActualizarParqWebService
                              actualizarParqueaderos = new
                              ActualizarParqWebService();
                               actualizarParqueaderos.execute();
                        }
                         });
              }};
         timer.schedule(actualizarParqTask, 0, 300000);
}
// Obtener Parqueaderos del Web Service
class ActualizarParqWebService extends AsyncTask<String,Integer,Boolean>{
      @Override
      protected Boolean doInBackground(String... par) {
            // obtener parqueaderos del web service
            webService = true;
            parqueaderos = new ArrayList<Parqueadero>();
            ParkingController parkController = new ParkingController();
            if (parkController.getParqueadros(") == null) {
                  webService = false;
                  cargarParqueaderosBaseDatos();
            } else {
                  parqueaderos = parkController.getParqueadros( );
                  publishProgress(10);
                  parqueaderosWebService = new ArrayList<Parqueadero>();
                  parqueaderosWebService = generarPlazasLibres();
                  publishProgress(40);
                  …
            }
            …
      }
      …
}
```

Figura 25. Obtención de los parqueaderos: Actualización de los datos y consulta al Web Service

ParkingController: Esta clase sirvió para hacer la consulta al web service y devolver una lista de los parqueaderos almacenados en su base de datos. (Ver figura 26).

```
public class ParkingController {
	private static final String url_parqueadero = "/K-Parking-
Rest/webresources/com.kradac.kparking.rest.entities.parqueaderos";
	private static final String tag_idParqueadero =
"idParqueadero";
	private static final String tag_razonSocial = "parqueadero";
	private static final String tag_direccion = "direccion";
	private static final String tag_plazasTotales = "plazas";
	private static final String tag_latitud = "latitud";
	private static final String tag_longitud = "longitud";
	private static final String tag_imagen = "imagen";

	public ArrayList<Parqueadero> getParqueadros(String
	url_parqueaderos) {
		HttpClient httpClient = new DefaultHttpClient();
		HttpGet del = null;
		del = new HttpGet(url_parqueaderos + url_parqueadero);
		del.setHeader("content-type", "application/json");
		try {
			HttpResponse resp = httpClient.execute(del);
			String respStr =
			EntityUtils.toString(resp.getEntity());
			JSONArray respJSON = new JSONArray(respStr);
			ArrayList<Parqueadero> listaParqueaderos = new
			ArrayList<Parqueadero>();
			for (int i = 1; i < respJSON.length(); i++) {
				JSONObject parqueaderoJSON =
				respJSON.getJSONObject(i);
				Parqueadero parqueadero = new Parqueadero(
				parqueaderoJSON.getInt(tag_idParqueadero),
				parqueaderoJSON.getString(tag_razonSocial),
				parqueaderoJSON.getString(tag_direccion),
				parqueaderoJSON.getDouble(tag_latitud),
				parqueaderoJSON.getDouble(tag_longitud),
				parqueaderoJSON.getInt(tag_plazasTotales),
				parqueaderoJSON.getString(tag_imagen));
				listaParqueaderos.add(parqueadero);

			}
			return listaParqueaderos;
		} catch (Exception ex) {
			Log.e("ServicioRest", "Error!", ex);
			return null;
		}
	}
}
```

Figura 26 . Obtención de los parqueaderos: Consulta al Web Service

- **Almacenamiento de los parqueaderos en una base de datos interna:**

Para almacenar los parqueaderos obtenidos del web service en la base de datos internas se debió ocupar dos clases auxiliares denominadas KParkingSqliteOpenHelper y ParqueaderoDataSource.

KParkingSqliteHelper: Esta clase sirve para crear la base de datos interna sqlite, crear la tabla de parqueaderos y actualizar en caso de una nueva versión. (Ver figura 27).

```
public class KParkingSqliteHelper extends SQLiteOpenHelper {
	private SQLiteDatabase db;
	private static final String DATABASE_CREATE_PARQUEADERO =
	"create table " + TablaParqueadero.tabla_ + "(" +
	TablaParqueadero.campo_id	+ " integer primary key not
	null, "	+ TablaParqueadero.campo_razonSocial + " text not
	null, "	+ TablaParqueadero.campo_direccion + " text not
	null, " + TablaParqueadero.campo_latitud + " real not null, " +
	TablaParqueadero.campo_longitud + " real not null, "
	+ TablaParqueadero.campo_plazaTotal + " integer not null, "
	+ TablaParqueadero.campo_plazaLibre + " integer not null, "
	+ TablaParqueadero.campo_imagenParqueadero + " text );";
	private static final String DATABASE_NAME = "kParking";
	private static final int DATABASE_VERSION = 1;
	private static KParkingSqliteHelper mOpenHelper = null;
	public KParkingSqliteHelper(Context paramContext) {
		super(paramContext, DATABASE_NAME, null,
DATABASE_VERSION);
	}
	public static KParkingSqliteHelper getInstance(Context
paramContext) {
		if (mOpenHelper == null) {
			mOpenHelper = new KParkingSqliteHelper(
					paramContext.getApplicationContext());
		}
		return mOpenHelper;
	}
	// Base de datos creada por primer vez
	public void onCreate(SQLiteDatabase paramSQLiteDatabase) {
		paramSQLiteDatabase.execSQL(DATABASE_CREATE_PARQUEADERO);
	}
	// Base de datos actualizada
	public void onUpgrade(SQLiteDatabase paramSQLiteDatabase, int
	paramInt1,int paramInt2) {
		paramSQLiteDatabase.execSQL("DROP TABLE IF EXISTS "
		+ TablaParqueadero.tabla_);
		onCreate(paramSQLiteDatabase);
	}
}
```

Figura 27. Base de Datos interna dela Aplicación: Creación de la base de datos

ParqueaderoDataSource: Esta clase permite abrir y cerrar la conexión con la base de datos, insertar o borrar registros (Ver figura 28) , buscar un registro u obtener todos los registros. (Ver figura 29).

```
public class ParqueaderoDataSource {
	public ParqueaderoDataSource(Context context) {
		dbHelper = KParkingSqliteHelper.getInstance(context);
	}
	//abrir base de datos
	public void open() {
		db = dbHelper.getWritableDatabase();
	}
	//cerrar base de datos
	public void close() {
		dbHelper.close();
	}
	//borrar base de datos
	public void delete() {
		db.execSQL("DELETE FROM " + TablaParqueadero.tabla_);
	}
	//crear un parqueadero
	public boolean crearParqueadero(String razonSocial, String
direccion, double latitud, double longitud, int plazasTotales,
int plazasLibres, String imagenParqueadero) {
		if (razonSocial.length() > 0) {
			ContentValues values = new ContentValues();
			values.put(
KParkingSqliteHelper.TablaParqueadero.campo_razonSocial, razonSocial);
			values.put(
KParkingSqliteHelper.TablaParqueadero.campo_direccion, direccion);
			values.put(
KParkingSqliteHelper.TablaParqueadero.campo_latitud, latitud);
			values.put(
KParkingSqliteHelper.TablaParqueadero.campo_longitud,longitud);
			values.put(
KParkingSqliteHelper.TablaParqueadero.campo_plazaTotal,plazasTotales);
		values.put(
KParkingSqliteHelper.TablaParqueadero.campo_plazaLibre, plazasLibres);
			values.put(
KParkingSqliteHelper.TablaParqueadero.campo_imagenParqueadero,
                                imagenParqueadero);
			return (db.insert(TablaParqueadero.tabla_, null,
values) != -1) ? true
                        : false;
		} else
			return false;
	}
	…
}
```

Figura 28. Base de Datos interna dela Aplicación: Operaciones de la Base de Datos (a)

```
public class ParqueaderoDataSource {
	…
	//bOrrar un registro
	public boolean borrarRegistro(int id) {
		return (db.delete(TablaParqueadero.tabla_,
		TablaParqueadero.campo_id + " = " + id, null) > 0) ? true
		: false;
	}
	//Obtener  todos los parqueaderos
	public List<Parqueadero> getAllParqueaderos() {
		List<Parqueadero> listaParqueaderos = new
ArrayList<Parqueadero>();
		Cursor cursor = db.query(TablaParqueadero.tabla_,
columnas, null, null, null, null, null);
		cursor.moveToFirst();
		while (!cursor.isAfterLast()) {
			Parqueadero parqueadero =
			cursorToParqueadero(cursor);
			listaParqueaderos.add(parqueadero);
			cursor.moveToNext();
		}
		cursor.close();
		return listaParqueaderos;
	}
	//Obtener un registro de acuerdo a su nombre
	public Parqueadero getParqueadero(String nombre) {
		String where = TablaParqueadero.campo_razonSocial + "=?";
		String[] whereArgs = new String[] {
String.valueOf(nombre) };
		Cursor cursor = db.query(TablaParqueadero.tabla_,
columnas, where,
				whereArgs, null, null, null);
		if (cursor.getCount() > 0) {
			cursor.moveToFirst();
			Parqueadero p = cursorToParqueadero(cursor);
			return p;
		} else {
			return null;
		}
	}
}
```

Figura 29. Base de Datos interna dela Aplicación: Operaciones de la Base de Datos (b)

Dentro de la actividad principal las clases KParkingSqliteOpenHelper y ParqueaderoDataSource son utilizadas después de obtener los datos del web service en la AsynkTask ActualizarParqWebService, es decir que estos datos cambiarán a medida que cambien los datos del web service y mantendrán sus mismos datos durante 5 minutos (Ver figura 30).

El proceso que se realiza es abrir la base de datos, comprobar si posee datos si los posee se borra los registros, luego se insertan los parqueaderos obtenidos en la consulta al web service y por último se obtienen estos parqueaderos en un ArrayList denominado parqueaderosAuxiliar.

```
// Obtener Parqueaderos del Web Service
class ActualizarParqWebService extends AsyncTask<String, Integer,
Boolean> {
	//Ejecución de la tarea
	@Override
	protected Boolean doInBackground(String... par) {
		webService = true;
		parqueaderos = new ArrayList<Parqueadero>();
		ParkingController parkController = new ParkingController();
		if (parkController.getParqueadros("http://200.0.29.117") ==
null) {
			webService = false;
			cargarParqueaderosBaseDatos();
		} else {
			…
			publishProgress(40);
			pds = new ParqueaderoDataSource(contexto);
			pds.open();
			if (pds.getAllParqueaderos().size() > 0) {
				borrarParqueaderos(parqueaderosAuxiliar);
			}
			pds.close();
			guardarParqueaderos(parqueaderosWebService);
			obtenerParqueaderos();
			publishProgress(50);
		}
		return true;
	}
	//Mientras se ejecuta la tarea
	@Override
	protected void onProgressUpdate(Integer... prog) {
		int progreso = prog[0].intValue();
		pDialog.setProgress(progreso);
	}
	//Antes que se ejecute la tarea
	@Override
	protected void onPreExecute() {
		pDialog.setProgress(0);
		pDialog.show();
	}

}
```

Figura 30. Base de Datos interna dela Aplicación: Obtención de los parqueaderos desde la Base de Datos interna de la Aplicación

- **Marcación de los Parqueaderos:**

Una vez que se obtuvieron los parqueaderos se procedió a marcar cada uno de ellos en el Mapa. El icono de los parqueaderos marcados corresponde al número de plazas disponibles, para lo cual se utilizaron tres colores: verde para más de 10 plazas libres, amarillo para más de 6 y menos de 10 plazas libres y rojo para menos de 5 plazas libres. (Ver figura 31)

```
/* Agregar los marcadores correspondientes a cada parqueadero */
public void marcarParqueaderos() {
      MarkerOptions marcador = new MarkerOptions();
      for (Parqueadero p : parqueaderosAuxiliar) {
            marcador.position(new LatLng(p.getLatitud(),
p.getLongitud()));
            marcador.title(p.getRazonSocial());
            marcador.snippet(p.toString());
            if (p.getPlazaLibre() <= 5) {
                  marcador.icon(BitmapDescriptorFactory
.fromResource(R.drawable.icono_park_rojo));
            } else {
                  if (p.getPlazaLibre() <= 10) {
                        marcador.icon(BitmapDescriptorFactory
.fromResource(R.drawable.icono_park_amarillo));
                  } else {
                        marcador.icon(BitmapDescriptorFactory
.fromResource(R.drawable.icono_park_verde));
                  }
            }
            // Agregar marcador al mapa
            Marker markerP = mapa.addMarker(marcador);
            // Ventana de Información Personalizada
            …
      }
}
```

Figura 31 . Marcación de los parqueaderos: Colocación de marcadores

4.1.3.2. Ubicación Actual

Para la obtención de la ubicación actual se hace uso del método getMyLocation() invocado sobre un objeto google maps y siempre que exista conexión a internet. (Ver figura 32).

getMyLocation(): Obtiene la localización del dispositivo, devuelve un objeto tipo Location.

```
/* Opciones del menu de Navegación Lateral */

private void selectItem(int position) {
	//ubicación actual
	if (position == 0) {
		if (circuloParqueaderoEncontrado != null) {
			circuloParqueaderoEncontrado.remove();
		}
		if (lineOptionsRuta != null) {
			mapa.clear();
			marcarParqueaderos();
		}
		if (parqueaderosCargados) {
			conexionInternet = comprobarConexionInternet();
			if (!conexionInternet) {
				presentarError(errorConexionInternet);
			} else {
				ubicacionActual = true;
				ajustarMapa(16,new LatLng(mapa.getMyLocation().
getLatitude(), mapa.getMyLocation().getLongitude()));
			}
		}

	} else {
		//ubicación alternativa
		if (position == 1) {
			…
		}else{
			…
		}
	}

	drawerList.setItemChecked(position, true);
	setTitle(titulosSherlockActionBar[position]);
	drawerLayout.closeDrawer(drawerList);
}
```

Figura 32. Ubicación Actual

4.1.3.3. Ubicación Alternativa

Para que el usuario pueda ingresar una ubicación diferente a la de su dispositivo se utilizan los servicios de place autocomplete de google. (Ver figura 33.)

Para esto se crea un dialogo con un AutoCompleteTextView para que el usuario pueda ingresar la dirección con mayor facilidad y que escoja una dirección que conste en el place autocomplete.

```
/* Opciones del menu de Navegación Lateral */
private void selectItem(int position) {
	//ubicación actual
	if (position == 0) {
		…
	} else {
		//ubicación alternativa
		if (position == 1) {
			conexionInternet = comprobarConexionInternet();
			if (conexionInternet == false) {
				presentarError(errorConexionInternet);
			} else {
				ubicacionActual = false;
				if (circuloParqueaderoEncontrado != null) {
					circuloParqueaderoEncontrado.remove();
				}
				if (lineOptionsRuta != null) {
					mapa.clear();
					marcarParqueaderos();
				}
				final View addView =
getLayoutInflater().inflate(R.layout.ubicacion_alterna, null);
				input = null;
				input = (AutoCompleteTextView) addView
.findViewById(R.id.direccion_autocompletar);
				input.setAdapter(new
PlacesAutoCompleteAdapter(this,R.layout.list_item));
				input.setOnItemClickListener(this);
				AlertDialog.Builder alert = new
AlertDialog.Builder(this);
				alert.setTitle("Ubicación Alternativa");
				alert.setMessage("Ingrese la dirección
alterna");
				alert.setView(addView);
				…
				…
```

Figura 33. Ubicación Alterna: Ingreso de Dirección (a)

A medida que el usuario ingresa texto en el AutoCompleTextView se realiza la consulta de las posibles direcciones que coincidan. (Ver figura 34) y al presionar ok se buscará las coordenadas de dicha dirección.

```
                    …
                    alert.setPositiveButton("Ok", new
            DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int
                        whichButton) {
                              if (marcadorPosicion != null) {
                                    marcadorPosicion.remove();
                              }
                              if (input.getEditableText().toString() !=
null){
                                    String s1 =
                                    input.getEditableText().toString();
                                    String s = s1.substring(0,
                                    s1.lastIndexOf(","));
                                    if (obtenerCoordenadas(s) != null){
                                          LatLng loc =
                                          obtenerCoordenadas(s);
                                          localizacion = loc;
                                          marcarPosicion(localizacion);
                                          ajustarMapa(16,
localizacion);
                                    } else {
                                          presentarError(
                                          errorUbicacionAlternativa);
                                    }
                              } else {
                                    presentarError(
errorUbicacionAlternativa);
                              }
                        }
                        });
                    alert.setNegativeButton("Cancel",
new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int
whichButton) {
                              // Canceled.
                        }
                        });
                    alert.show();
              }
        } else {
              …
        }
        drawerList.setItemChecked(position, true);
        setTitle(titulosSherlockActionBar[position]);
        drawerLayout.closeDrawer(drawerList);
}
```

Figura 34. Ubicación Alterna: Ingreso de Dirección (b)

PlacesAutoCompleteAdapter: Es un adaptador utilizado del Place AutoComplete cuyo fin es obtener las direcciones que tengan semejanza con el texto ingresado en el AutoCompleteTextView del diálogo. (Ver figura 35)

```
/* Adaptador para el AutoCompleteTextView del Place AutoComplete */
public class PlacesAutoCompleteAdapter extends ArrayAdapter<String>
                  implements Filterable {
      private ArrayList<String> resultList;
      public PlacesAutoCompleteAdapter(Context context, int
textViewResourceId) {
            super(context, textViewResourceId);
      }
      @Override
      public int getCount() {
            return resultList.size();
      }
      @Override
      public String getItem(int index) {
            return resultList.get(index);
      }
      @Override
      public Filter getFilter() {
            Filter filter = new Filter() {
            @Override
            protected FilterResults performFiltering(CharSequence
constraint) {
                  FilterResults filterResults = new FilterResults();
                  if (constraint != null) {
                        resultList =
autocompletarDirecciones(constraint.toString());
                        // Asignar el filtro
                        filterResults.values = resultList;
                        filterResults.count = resultList.size();
                  }
                  return filterResults;
            }
            @Override
            protected void publishResults(CharSequence constraint,
                  FilterResults results) {
                  if (results != null && results.count > 0) {
                        notifyDataSetChanged();
                  } else {
                        notifyDataSetInvalidated();
                  }
            }
            };
            return filter;
      }
}
```

Figura 35. Ubicación Alterna: Adaptador para el texto de AutoCompletado.

autocompletarDirecciones: Este método se encarga de consultar al api place, obtener el objeto JSON y devolver una lista de direcciones semejantes a la ingresada. (Ver figura 36)

```
/* Conexión al servidor de Google para la obtencion de direcciones*/
private ArrayList<String> autocompletarDirecciones(String input) {
	String PLACES_API_BASE="https://maps.googleapis.com/maps/api/place";
	String TYPE_AUTOCOMPLETE = "/autocomplete";
	String OUT_JSON = "/json";
	String API_KEY = "AIzaSyCNVXBAUmiZeI8ZM4MrGvEiEr5haeqeT94";
	ArrayList<String> resultList = null;
	HttpURLConnection conn = null;
	StringBuilder jsonResults = new StringBuilder();
	try {
		StringBuilder sb = new StringBuilder(PLACES_API_BASE
		+ TYPE_AUTOCOMPLETE + OUT_JSON);
		sb.append("?sensor=false&key=" + API_KEY);
		sb.append("&components=country:ec");
		sb.append("&input=" + URLEncoder.encode(input, "utf8"));
		URL url = new URL(sb.toString());
		conn = (HttpURLConnection) url.openConnection();
		InputStreamReader in = new
InputStreamReader(conn.getInputStream());
		// Load the results into a StringBuilder
		int read;
		char[] buff = new char[1024];
		while ((read = in.read(buff)) != -1) {
			jsonResults.append(buff, 0, read);
		}
	} catch (MalformedURLException e) {
		Log.e(LOG_TAG, " Error al procesar URL de Api Place", e);
		return resultList;
	} catch (IOException e) {
		Log.e(LOG_TAG, "Error al conectarse al Api Place", e);
		return resultList;
	} finally {
		if (conn != null) {conn.disconnect();}
	}
	try {
		JSONObject jsonObj = new JSONObject(jsonResults.toString());
		JSONArray predsJsonArray=jsonObj.getJSONArray("predictions");
		resultList = new ArrayList<String>(predsJsonArray.length());
		for (int i = 0; i < predsJsonArray.length(); i++) {
			resultList.add(predsJsonArray.getJSONObject(i).
getString("description"));
		}
	} catch (JSONException e) {
		Log.e(LOG_TAG, "Cannot process JSON results", e);
	}
	return resultList;
}
```

Figura 36. Ubicación Alterna: Consulta de direcciones de api place de google

4.1.3.4. Todos los parqueaderos

Para observar todos los parqueaderos de la ciudad de Loja se cambió el nivel de zoom del mapa. (Ver figura 37)

```
/* Opciones del menu de Navegación Lateral */
private void selectItem(int position) {
	//ubicación actual
	if (position == 0) {
		…
	} else {
		…
		//Todos los parqueaderos
		if (position == 2) {
			if (circuloParqueaderoEncontrado != null) {
				circuloParqueaderoEncontrado.remove();
			}
			ubicacionActual = true;
			int zoom = 14;
			localizacion = localizacionLoja;
			ajustarMapa(zoom, localizacion);
		} else {
			//Acerca de
			…
		}
	}
}
```

Figura 37. Todos los parqueaderos

4.1.3.5. Acerca de

Se utilizó un diálogo para presentar la información de la aplicación.(Ver figura 38)

```
/* Opciones del menu de Navegación Lateral */
private void selectItem(int position) {
	…
	//Acerca de
	if (position == 3) {
		AlertDialog.Builder alert = new AlertDialog.Builder(this);
		alert.setTitle("Acerca De");
		alert.setMessage(informacionAplicacion);
		…
		alert.show();
	}
	…
}
```

Figura 38. Acerca de

4.1.3.6. Información de un parqueadero

Para la visualización de la información de un parqueadero fue necesaria la utilización de un objeto de la clase InfoWindowAdapter y de la librería universal-image-loader.

- **Construcción de una Ventana de Información Personalizada**

Se utilizó InfoWindowAdapter para personalizar la ventana de información de un marcador. Este objeto fue añadido al momento de ir marcando cada uno de los parqueaderos en el mapa. (Ver figura 39)

```
// Ventana de Información Personalizada
infoWindowAdapterPersonalizado = new InfoWindowAdapter() {
      @Override
      public View getInfoWindow(final Marker marker) {
            LayoutInflater inflater = (LayoutInflater)
contexto.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            View infoWindow = inflater.inflate(
R.layout.info_window_parqueadero, null);
            imagenParqueadero = (ImageView) infoWindow
.findViewById(R.id.imagePark);
            String descripcionParqueadero = marker.getSnippet()
.substring(0, marker.getSnippet().indexOf(":"));
            urlImagen = marker.getSnippet().substring(
marker.getSnippet().indexOf(":") + 1,marker.getSnippet().length());
            //Utilización de la librería universal-image-loader
            imageLoader.displayImage(urlImagen,imagenParqueadero,
options, new SimpleImageLoadingListener() {
                  @Override
                  public void onLoadingComplete(String imageUri, View
view, Bitmap loadedImage) {
                     super.onLoadingComplete(imageUri, view,loadedImage);
                     getInfoContents(marker);
                  }});
            TextView tvTitle=((TextView) infoWindow.findViewById(
R.id.nombrePark));
            tvTitle.setText(marker.getTitle());
            TextView tvSnippet = ((TextView) infoWindow
.findViewById(R.id.descripcionPark));
            tvSnippet.setText(descripcionParqueadero);
            return infoWindow;
      }
      @Override
      public View getInfoContents(Marker arg0) {
            return null;
      }
};
mapa.setInfoWindowAdapter(infoWindowAdapterPersonalizado);
```

Figura 39. Información de un Parqueadero: Construcción de Ventana de Información Personalizada

Librería universal-image-loader: La librería universal-image-loader es utilizada para la inclusión de imágenes externas a una aplicación, esta es utilizada sobre un objeto de la clase ImageLoader para realizar la carga de la imagen desde un link (Ver imagen n). Esta librería además necesita

- **Configuración de la Librería universal_image_loader:**

Para utilizar la librería universal-image-loader para cargar una imagen desde un link (Ver imagen n) es necesario realizar ciertas configuraciones en el método onCreate() y utilizar otros métodos descritos a continuación (Ver figura 40).

onCreate: En este método invoca a un método encargado de y de determinar la configuración de las imágenes, como por ejemplo que el link nos ofrezca una imagen vacía.(Ver figura 40)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	conexionInternet = comprobarConexionInternet();
	conexionInternet = comprobarConexionInternet();
	if (conexionInternet == false) {
		cargarParqueaderosBaseDatos();
	} else {
		actualizarParqueaderosWebService(0);
	}
	setContentView(R.layout.drawer_layout);
		contexto = this;
	…
	drawerLayout.setDrawerListener(drawerToggle);
	initImageLoader();
	imageLoader = ImageLoader.getInstance();
	options = new DisplayImageOptions.Builder()
.showStubImage(R.drawable.ic_launcher)
.showImageForEmptyUri(R.drawable.ic_launcher)
.cacheInMemory().cacheOnDisc()
.bitmapConfig(Bitmap.Config.RGB_565).build();
	…
}
```

Figura 40. Información de un Parqueadero: Configuración de la librería universal_image_loader (a)

initImageLoader: En este método se realiza la configuración del tamaño de las imágenes cargadas para no saturar la memoria caché del dispositivo. (Ver figura 41)

```
/* configuración de imágenes con la librería universal-image-loader */
private void initImageLoader() {
	//Configuración del tamaño de las imágenes
	int memoryCacheSize;
	if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.ECLAIR) {
		int memClass = ((ActivityManager) contexto
.getSystemService(Context.ACTIVITY_SERVICE))
		.getMemoryClass();
		memoryCacheSize = (memClass / 8) * 1024 * 1024;
	} else {
		memoryCacheSize = 2 * 1024 * 1024;
	}
	final ImageLoaderConfiguration config = new
	ImageLoaderConfiguration.Builder(contexto)
	.threadPoolSize(5)
	.threadPriority(Thread.NORM_PRIORITY - 2)
	.memoryCacheSize(memoryCacheSize)
	.memoryCache(new FIFOLimitedMemoryCache(memoryCacheSize -
1000000))
	.denyCacheImageMultipleSizesInMemory()
	.discCacheFileNameGenerator(new Md5FileNameGenerator())

	.tasksProcessingOrder(QueueProcessingType.LIFO).enableLogging()
			.build();
	ImageLoader.getInstance().init(config);
}
```

Figura 41. Información de un Parqueadero: Configuración de la librería universal_image_loader (b)

4.1.3.7. Búsqueda de Parqueadero

Para la búsqueda de un parqueadero por una razón social se utilizó un searchView implementado en el action bar de la aplicación. Para esto fue necesario utilizar algunos métodos.

onCreateOptionsMenu(): Dentro de este método se configura la opción de búsqueda en el action bar, para lo cual se define el conjunto de datos en dónde se va a realizar la búsqueda, el tipo de objeto a buscar, el adaptador que sirve para filtrar y los métodos al momento de ingresar texto en el search view, de seleccionar una opción y de enviar una búsqueda a través del teclado virtual del dispositivo. (Ver figura 42.)

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
	MenuInflater inflater = getSupportMenuInflater();
	inflater.inflate(R.menu.main, menu);
	if (mSuggestionsAdapter == null) {
		MatrixCursor cursor = new MatrixCursor(COLUMNS);
		for (Parqueadero p : parqueaderosAuxiliar) {
			cursor.addRow(new Object[] { p.getId(),
p.getRazonSocial(),p.getDireccion(), p.getLatitud(), p.getLongitud(),
p.getPlazaTotal(), p.getPlazaLibre(),p.getImagenParqueadero() });
		}
		mSuggestionsAdapter = new ParkingSearchAdapter(
getSupportActionBar().getThemedContext(), cursor);
	}
	searchView = (SearchView) menu.findItem(R.id.menu_search)
.getActionView();
	searchView.setOnQueryTextListener(this);
	searchView.setOnSuggestionListener(this);
	searchView.setSuggestionsAdapter(mSuggestionsAdapter);
	return super.onCreateOptionsMenu(menu);
}
```

Figura 42. Búsqueda de un parqueadero: Configuración del texto de búsqueda de la barra superior

onQueryTextSubmit(): Se busca en la base de datos interna el parqueadero cuya razón social coincida con el texto ingresado. Este método es invocado cuando el usuario envía la consulta mediante el teclado virtual del dispositivo. (Ver figura 43).

```
/* Busqueda de parqueadero por su razón social */
public boolean onQueryTextSubmit(String query) {
	pds = new ParqueaderoDataSource(this);
	pds.open();
	Parqueadero pEncontrado = pds.getParqueadero(query.toLowerCase());
	pds.close();
	if (pEncontrado != null) {
		localizacion = new LatLng(pEncontrado.getLatitud(),
pEncontrado.getLongitud());
		ajustarMapa(17, localizacion);
		// ocultar teclado
		InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
		imm.hideSoftInputFromWindow(searchView.getWindowToken(), 0);
		marcarParqueaderoBuscado(pEncontrado.getLatitud(),
pEncontrado.getLongitud());
	} else {
		Toast.makeText(this, "Parqueadero " + query + " no
encontrado", Toast.LENGTH_SHORT).show();
	}
	return true;
}
```

Figura 43. Búsqueda de un parqueadero: Consulta enviada desde el teclado virtual del dispositivo

onSuggestionClick(): En este método se realiza la búsqueda en la base de datos interna del parqueadero cuya razón social coincida con el texto ingresado. Este método es invocado cuando el usuario selecciona un parqueadero de la lista sugerida por la aplicación. (Ver figura 44)

```
/* Parqueadero buscado es seleccionado */
@Override
public boolean onSuggestionClick(int position) {
	Cursor c = (Cursor) mSuggestionsAdapter.getItem(position);
	String query = c.getString(c
.getColumnIndex(SearchManager.SUGGEST_COLUMN_TEXT_1));
	Toast.makeText(this, "Parqueadero seleccionado: " + query,
Toast.LENGTH_LONG).show();
	// Marcar parqueadero
	// false para que no se envie directamente
	searchView.setQuery(query, false);
	searchView.clearFocus();
	pds = new ParqueaderoDataSource(this);
	pds.open();
	Parqueadero pEncontrado = pds.getParqueadero(query);
	pds.close();
	localizacion = new LatLng(pEncontrado.getLatitud(),
pEncontrado.getLongitud());
	ajustarMapa(17, localizacion);
	marcarParqueaderoBuscado(pEncontrado.getLatitud(),
pEncontrado.getLongitud());
	return true;
}
```

Figura 44. Búsqueda de un parqueadero: Selección del parqueadero buscado

onQueryTextChange(): Este método es invocado a medida que el usuario ingresa texto en el searchView. El proceso realizado es la búsqueda del texto en la razón social de los parqueaderos y luego determinar el adaptador para la lista de parqueaderos que se muestran como sugerencia. (Ver figura 45)

ParkingSearchAdapter: Es una clase que sirve como adaptador de listas para la visualización de los parqueaderos que se muestran como sugerencia al texto ingresado al usuario. Para este adaptador es necesario utilizar un layout. (Ver figura 46)

```
/* Cambio de texto de consulta de parqueadero */
@Override
public boolean onQueryTextChange(String newText) {
	// Filtro
	MatrixCursor cursor = new MatrixCursor(COLUMNS);
	newText.toLowerCase();
	for (Parqueadero p : parqueaderosAuxiliar) {
		if (p.getRazonSocial().contains(newText)) {
			cursor.addRow(new Object[] { p.getId(),
p.getRazonSocial(),p.getDireccion(), p.getLatitud(), p.getLongitud(),
p.getPlazaTotal(), p.getPlazaLibre(), p.getImagenParqueadero() });
		}
	}
	mSuggestionsAdapter = null;
	mSuggestionsAdapter = new
ParkingSearchAdapter(getSupportActionBar().getThemedContext(), cursor);
	searchView.setSuggestionsAdapter(mSuggestionsAdapter);
	return false;
}
```

Figura 45. Búsqueda de un parqueadero: Cambio en el texto de búsqueda del parqueadero

```
/* Adaptador para la búsqueda de parqueaderos por su razón social */
private class ParkingSearchAdapter extends CursorAdapter {
	public ParkingSearchAdapter(Context context, Cursor c) {
		super(context, c, 0);
	}
	@Override
	public View newView(Context context, Cursor cursor, ViewGroup
parent) {
		LayoutInflater inflater = LayoutInflater.from(context);
		View v = inflater.inflate(R.layout.parqueadero_adapter,
parent, false);
		return v;
	}
	@Override
	public void bindView(View view, Context context, Cursor cursor) {
		TextView nombre = (TextView)
view.findViewById(R.id.nombrePark);
		final int textIndex =
cursor.getColumnIndex(SearchManager.SUGGEST_COLUMN_TEXT_1);
		nombre.setText(cursor.getString(textIndex));
		final int dirIndex = cursor.getColumnIndex("Direccion");
		TextView direccion = (TextView) view
.findViewById(R.id.descripcionPark);
		direccion.setText(cursor.getString(dirIndex));
	}
}
```

Figura 46. Búsqueda de un parqueadero: Adaptador para el texto de búsqueda

4.1.3.8. Buscar Parqueadero más cercano

Para la determinación del parqueadero más cercano se empleó un método con el fin de encontrar el parqueadero con menor distancia hacia la localización del dispositivo o de una dirección alternativa ingresada por el usuario; además se utilizó el api de direcciones de google para trazar la ruta hacia el parqueadero y por último se permite al usuario abrir la aplicación navigation o maps con la ruta trazada.

- **Obtener el parqueadero más cercano**

Para conseguir este parqueadero se utiliza la localización de cada uno de los parqueaderos y se utiliza el método distanceTo() para obtener la distancia entre dos localizaciones. (Ver figura 47)

```
/*calcular el parqueadero más cercano */
public Parqueadero calcularParqueaderoCercano() {
	Parqueadero parqueaderoMinDistancia = parqueaderosAuxiliar.get(0);
	for (int i = 1; i < parqueaderosAuxiliar.size(); i++) {
		if (calcularDistancia(parqueaderosAuxiliar.get(i)) <
calcularDistancia(parqueaderoMinDistancia)) {
			parqueaderoMinDistancia =parqueaderosAuxiliar.get(i);
		}
	}
	return parqueaderoMinDistancia;
}
/* calcular distancia desde una ubicación hasta un parqueadero */
public double calcularDistancia(Parqueadero p) {
	double distancia = 0;
	Location localizacionA = new Location("localizacion Actual");
	if (ubicacionActual) {
		localizacionA.setLatitude(
mapa.getMyLocation().getLatitude());
		localizacionA.setLongitude(
mapa.getMyLocation().getLongitude());
	} else {
		localizacionA.setLatitude(localizacion.latitude);
		localizacionA.setLongitude(localizacion.longitude);
	}
	Location localizacionParqueadero = new
Location("localizacionParqueadero");
	localizacionParqueadero.setLatitude(p.getLatitud());
	localizacionParqueadero.setLongitude(p.getLongitud());
	distancia = localizacionA.distanceTo(localizacionParqueadero);
	distanciaAux = String.valueOf(distancia);
	return distancia;
}
```

Figura 47. Búsqueda del parqueadero más cercano: Obtención del parqueadero con menor distancia en metros.

- **Presentar la ruta hacia el parqueadero más cercano**

Para obtener la ruta se determinó como origen la ubicación seleccionada por el usuario y como destino el parqueadero más cercano, con estos datos se construyó la url a consultar, se obtuvo la ruta y se procede a trazarla en el mapa. (Ver figura 48)

```
/* Presentar la información y ruta del parqueadero más cercano */
public void presentarParqueaderoCercano() {
	AlertDialog.Builder alert = new AlertDialog.Builder(this);
	alert.setTitle("Parqueadero Más cercano");
	alert.setMessage("El parqueadero más cercano posee "
+ parqueaderoCercano.getPlazaLibre() + "plazas libres y está a " +
distanciaAux + " metros. ¿Desea trazar la ruta hacia el parqueadero? ");
	alert.setPositiveButton("Ok",new DialogInterface.OnClickListener(){
		public void onClick(DialogInterface dialog,int whichButton) {
			// Trazar ruta
			if (circuloParqueaderoEncontrado != null) {
				circuloParqueaderoEncontrado.remove();
			}
			marcarParqueaderoBuscado(
parqueaderoCercano.getLatitud(),parqueaderoCercano.getLongitud());
			LatLng origin = new LatLng(-3.997154, -79.201347);
			conexionInternet = comprobarConexionInternet();
			if (!conexionInternet) {
				presentarError(errorConexionInternet);
			} else {
				if (ubicacionActual) {
					origin = new LatLng(mapa.getMyLocation().
getLatitude(), mapa.getMyLocation().getLongitude());
				} else { origin = localizacion;  }
				LatLng dest = new
LatLng(parqueaderoCercano.getLatitud(),parqueaderoCercano.getLongitud());
// Obtener la ruta del api de direcciones de google
				String url = construirUrlConsultaRuta(origin,
dest);
				ObtencionRutaTask obtencionTask = new
ObtencionRutaTask();
				obtencionTask.execute(url);
			}
		}
	});
	alert.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
		public void onClick(DialogInterface dialog, int whichButton)
{
		}
	});
	alert.show();
}
```

Figura 48. Búsqueda del parqueadero más cercano: Presentación de información y ruta del parqueadero más cercano

construirRutaUrlConsultaRuta(): Este método sirve para la construcción de la consulta a realizar al api de direcciones de google. (Ver figura 49)

ObtencionRutaTask: Esta tarea se utiliza para la obtención de la ruta como un objeto JSON a partir de la url enviada como consulta al api de direcciones de google y luego llama a la tarea TrazadoRutaTask para trazar la ruta en el mapa. (Ver figura 50)

```
/* Construcción de la url de consulta para el api de direcciones */
private String construirUrlConsultaRuta(LatLng origin, LatLng dest) {
	// Origen de la ruta
	String str_origin = "origin=" + origin.latitude + "," +
origin.longitude;
	// Destino de la ruta
	String str_dest = "destination=" + dest.latitude + "," +
dest.longitude;
	String sensor = "sensor=false";
	String parameters = str_origin + "&" + str_dest + "&" + sensor;
	String output = "json";
	// url final
	String url = "https://maps.googleapis.com/maps/api/directions/"
+ output + "?" + parameters;
	return url;
}
/* Tarea para la obtención de la ruta como objeto JSON */
private class ObtencionRutaTask extends AsyncTask<String, Void, String> {
	@Override
	protected String doInBackground(String... url) {
	String rutaJSON = "";
	try {
		rutaJSON = obtenerRuta(url[0]);
	} catch (Exception e) {
		Log.d("Background Task", e.toString());
	}
		return rutaJSON;
	}
	@Override
	protected void onPostExecute(String result) {
		super.onPostExecute(result);
		TrazadoRutaTask parserTask = new TrazadoRutaTask();
		parserTask.execute(result);
	}
}
```

Figura 49. Búsqueda del parqueadero más cercano: Construcción de url para consulta al api de direcciones

TrazadoRutaTask: Esta tarea se encarga de convertir el objeto JSON en una lista con las diferentes longitudes y latitudes de los puntos que conforman la ruta. Luego agrega un objeto PolylineOptions con la ruta a trazar sobre el mapa. (Ver figura 50.)

```
/* Tarea para la transformación del objeto JSON y trazado de la ruta */
private class TrazadoRutaTask extends AsyncTask<String, Integer,
List<List<HashMap<String, String>>>> {
	@Override
	protected List<List<HashMap<String, String>>> doInBackground(
String... jsonData) {
		JSONObject jObject;
		List<List<HashMap<String, String>>> routes = null;
		try {
			jObject = new JSONObject(jsonData[0]);
			//Transformación del objeto JSON en una lista de
listas de latitudes y longitudes que componen la ruta
			DirectionsJSONParser parser = new
DirectionsJSONParser();
			routes = parser.parse(jObject);
		} catch (Exception e) {
			e.printStackTrace();
		}
		return routes;
	}
	@Override
	protected void onPostExecute(List<List<HashMap<String, String>>>
result) {
		ArrayList<LatLng> points = null;
		lineOptionsRuta = null;
		for (int i = 0; i < result.size(); i++) {
			points = new ArrayList<LatLng>();
			lineOptionsRuta = new PolylineOptions();
			List<HashMap<String, String>> path = result.get(i);
			for (int j = 0; j < path.size(); j++) {
				HashMap<String, String> point = path.get(j);
				double lat =
Double.parseDouble(point.get("lat"));
				double lng =
Double.parseDouble(point.get("lng"));
				LatLng position = new LatLng(lat, lng);
points.add(position);
			}
			lineOptionsRuta.addAll(points);
			lineOptionsRuta.width(7);
			lineOptionsRuta.color(Color.RED);
		}
		// Dibujar la ruta
		mapa.addPolyline(lineOptionsRuta);
		if (ubicacionActual) {
			abrirNavigation();
		} else {
			abrirGoogleMaps();
		}
	}
}
```

Figura 50. Búsqueda del parqueadero más cercano: Obtención de los puntos de la ruta y su trazado.

- **Abrir la aplicación Maps o Navigation con la ruta trazada.**

abrirGoogleMaps(): Este método se encarga de abrir la aplicación maps con la ruta trazada hacia el parqueadero más cercano siempre y cuando la ubicación seleccionada por el usuario sea la alternativa. En caso de no existir la aplicación se presentará un diálogo de error. (Ver figura 51)

```
/* abrir google maps */
public void abrirGoogleMaps() {
	AlertDialog.Builder alert = new AlertDialog.Builder(this);
	alert.setTitle("Abrir Maps");
	alert.setMessage("Desea abrir la aplicación Maps con la ruta hacia
el parqueadero más cercano");
	alert.setPositiveButton("Ok", new DialogInterface.OnClickListener()
{
		public void onClick(DialogInterface dialog, int whichButton)
{
			String uri = "";
			uri = String.format(Locale.ENGLISH,
"http://maps.google.com/maps?saddr=%f,%f(%s)&daddr=%f,%f (%s)",
localizacion.latitude, localizacion.longitude, "Ubicación",
parqueaderoCercano.getLatitud(),parqueaderoCercano.getLongitud(),
"Parqueadero más cercano");
			Intent intent = new Intent(Intent.ACTION_VIEW,
uri.parse(uri));
			intent.setClassName("com.google.android.apps.maps",
"com.google.android.maps.MapsActivity");
			boolean maps = true;
			try {
				startActivity(intent);
			} catch (ActivityNotFoundException ex) {
				try {
					Intent unrestrictedIntent = new Intent(
					Intent.ACTION_VIEW, Uri.parse(uri));
					startActivity(unrestrictedIntent);
				} catch (ActivityNotFoundException innerEx) {
					maps = false;
					presentarError(errorMaps);
				}
			}
		}
	});
	alert.setNegativeButton("Cancel", new
DialogInterface.OnClickListener(){
		public void onClick(DialogInterface dialog,int whichButton){
		}
	});
	alert.show();
}
```

Figura 51. Búsqueda del parqueadero más cercano: Visualización de la ruta con la aplicación maps

abrirNavigation(): Este método se encarga de abrir la aplicación navigation con la ruta trazada hacia el parqueadero más cercano desde la ubicación del dispositivo como origen. En caso de no existir la aplicación se presentará un diálogo de error. (Ver figura 52)

```
/* abri Navigation */
public void abrirNavigation() {
	AlertDialog.Builder alert = new AlertDialog.Builder(this);
	alert.setTitle("Abrir Navigation");
	alert.setMessage("Desea abrir la aplicación Navigation con la ruta
hacia el parqueadero más cercano");
	alert.setPositiveButton("Ok", new DialogInterface.OnClickListener()
{
		public void onClick(DialogInterface dialog, int whichButton)
{
			Intent intent = new Intent(Intent.ACTION_VIEW, Uri
.parse("google.navigation:q="+ parqueaderoCercano.getLatitud() + ","
+ parqueaderoCercano.getLongitud()));
			try {
				startActivity(intent);
			} catch (ActivityNotFoundException ex) {
				presentarError(errorNavigation);
			}
		}
	});
	alert.setNegativeButton("Cancel",
new DialogInterface.OnClickListener() {
		public void onClick(DialogInterface dialog, int whichButton)
{
		}
	});
	alert.show();
}
```

Figura 52. Búsqueda del parqueadero más cercano: Visualización de la ruta con la aplicación navigation

5. CUARTA FASE: Pruebas

Luego de haber terminado con la codificación se deben realizar las pruebas que comprueben que es una aplicación de calidad y que cumple con los requerimientos por las que fue creada.

5.1. PRUEBAS DEL SISTEMA Y ARREGLOS

Las pruebas que la metodología Mobile-D establece para una aplicación son la comprobación de requerimientos, las pruebas unitarias, los resultados en tempos de respuestas y tiempos de accesos, además de una lista de dispositivos en dónde se ha comprobado el adecuado funcionamiento de la aplicación.

5.1.1. Pruebas Unitarias

Las pruebas unitarias se realizaron sobre el conjunto de clases y métodos que tienen funciones específicas como la obtención de datos del web Service.

5.1.1.1. Caso de Prueba 1: Obtención de parqueaderos del Web Service:

La realización de esta prueba se hizo para comprobar que la listas de parqueaderos necesarios para el funcionamiento de la aplicación reciben algún valor luego de invocar al método actualizarParqueaderosWebService. (Ver Tabla XVIII)

TABLA XVII.

PRUEBA UNITARIA: OBTENCIÓN DE PARQUEADEROS DEL WEB SERVICE

Código	Nombre
PU001	Obtención de parqueaderos del Web Service
Objetivo	Obtener la lista con los parqueaderos cargados de la aplicación
Pasos	1. Inicializar la actividad. 2. Invocar el método actualizarParqueaderosWebService(tiempo entre cada ejecución) con un tiempo entre ejecución de 30000 milisegundos. 3. Comprobar que la lista parqueaderos que recibe los parqueaderos cargados del WebService o de la base de datos interna no esté vacío. 4. Comprobar que la lista parqueaderosAuxiliar que recibe los parqueaderos guardados de la base de datos interna no esté vació.
Resultados Esperados	La lista parqueaderos, no debe ser nula La lista parqueaderosAuxiliar, no debe ser nula
Resultados Obtenidos	La lista parqueaderos no es nula La lista parqueaderosAuxiliar no es nula

El código utilizado para la elaboración de la prueba está descrita a continuación en la Figura .53

```
@Test
public void testActualizarParqueaderosWebService(){
	MainActivity ma = this.getActivity();
	ma.actualizarParqueaderosWebService(30000);
	assertNotNull(ma.parqueaderos);
	assertNotNull(ma.parqueaderosAuxiliar);
}
```

Figura 53. Prueba Unitaria: Obtención de parqueaderos del Web Service

La correcta ejecución del método se presenta en la figura n.54

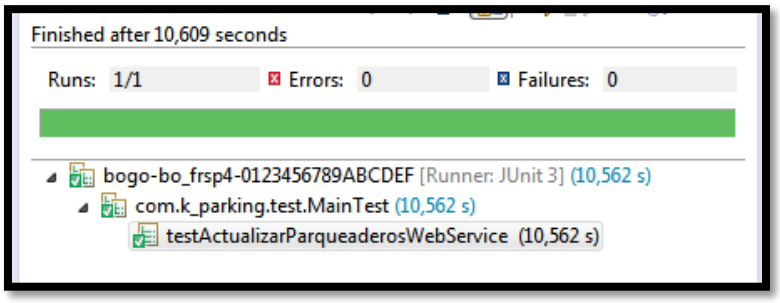


Figura 54. Ejecución de Prueba Unitaria: Obtención de parqueaderos del Web Service

5.1.1.2. Caso de Prueba 2: Almacenar parqueaderos en una base de datos interna

Esta prueba sirve para comprobar que los datos se almacenan correctamente en una base de datos, para esto se probó el método guardarParqueaderos. (Ver Tabla XIX)

TABLA XVIII.

PRUEBA UNITARIA: ALMACENAMIENTO DE PARQUEADEROS EN UNA BASE DE DATOS

Código	Nombre
PU002	Almacenamiento Parqueaderos en una Base de Datos
Objetivo	Almacenar Parqueaderos en una Base de Datos
Pasos	1. Inicializar la actividad. 2. Invocar el método actualizarParqueaderosWebService(tiempo entre cada ejecución) con un tiempo entre ejecución de 30000 milisegundos. 3. Obtener el número de parqueaderos de la Base de Datos 4. Crear un objeto Parqueadero y agregarlo a una lista 5. Invocar el método guardarParqueaderos (lista de parqueaderos a agregar) con la lista de parqueaderos creada. 6. Obtener el número de parqueaderos de la Base de Datos 7. Comprobar que el número de parqueaderos de la Base de Datos ha incrementado.
Resultados Esperados	El número de parqueaderos de la base de datos incremente al guardar un nuevo parqueadero.
Resultados Obtenidos	El número de parqueaderos aumenta al almacenar un nuevo parqueadero.

El código utilizado para la elaboración de la prueba está descrito a continuación en la figura 55, mediante el cual se comprobó que el método utilizado en la aplicación que permite guardar parqueaderos en la base de datos interna de la aplicación está funcionando adecuadamente.

```
@Test
public void testGuardarParqueaderos(){
	MainActivity ma = this.getActivity();
	ma.actualizarParqueaderosWebService(300000);
	ParqueaderoDataSource pds = new ParqueaderoDataSource(ma);
	pds.open();
	int parqueaderos = pds.getAllParqueaderos().size();
	pds.close();
	ArrayList<Parqueadero> parqueaderosTest = new
ArrayList<Parqueadero>();
	Parqueadero parkTest = new Parqueadero(25, "Parqueadero Nuevo
Horizonte","BernardoValdiviezo e/ Miguel Riofrío y Rocafuerte", -3.998391,
-79.201126, 12, 8, "http://www.elnuevosiglo.com.co/sites/default/
files/imagecache/400xY/fotoparqueadero_4.png");
	parqueaderosTest.add(parkTest);
	ma.guardarParqueaderos(parqueaderosTest);
	pds.open();
	int parqueaderosAuxiliar = pds.getAllParqueaderos().size();
	pds.close();
	parqueaderos++;
	 assertEquals(parqueaderos, parqueaderosAuxiliar);
}
```

Figura 55. Prueba Unitaria: Almacenar Parqueaderos en una base de Datos Interna de la aplicación

La correcta ejecución del método se presenta en la figura 56

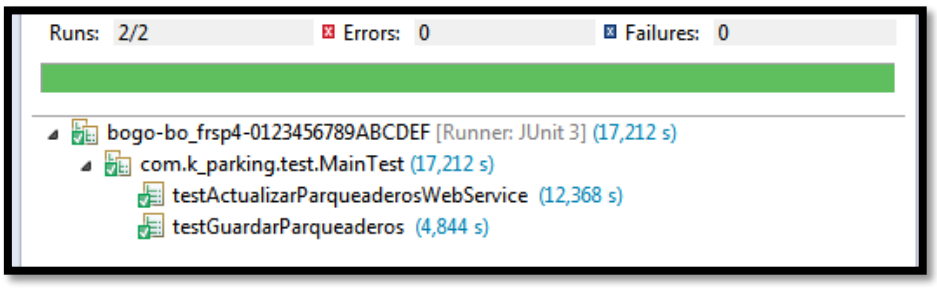


Figura 56. Ejecución de Prueba Unitaria: Almacenar Parqueaderos en una base de Datos Interna de la aplicación

5.1.1.3. Caso de Prueba 3: Obtener parqueaderos de la base de datos interna

Se realizó la prueba unitaria con el fin de verificar la obtención de los parqueaderos desde la base de datos interna. (Ver Tabla XX)

TABLA XIX.

PRUEBA UNITARIA: OBTENCIÓN DE PARQUEADEROS DE LA BASE DE DATOS INTERNA DE LA APLICACIÓN

Código	Nombre
PU003	Obtención de parqueaderos de la Base de Datos
Objetivo	Obtener una lista con los parqueaderos cargados desde la Base de Datos.
Pasos	1. Inicializar la actividad. 2. Invocar el método actualizarParqueaderosWebService(tiempo entre cada ejecución) con un tiempo entre ejecución de 30000 milisegundos. 3. Obtener los parqueaderos desde la Base de Datos. 4. Comprobar que la lista sí obtiene objetos.
Resultados Esperados	La lista parqueaderosAuxiliar con objetos, no debe ser nula
Resultados Obtenidos	La lista parqueaderosAuxiliar no es nula

El código utilizado para la elaboración de la prueba está descrito a continuación en la figura 57.

```
@Test
public void testObtenerParqueaderos(){
      MainActivity ma = this.getActivity();
      ma.actualizarParqueaderosWebService(300000);
      ma.obtenerParqueaderos();
      assertNotNull(ma.parqueaderos);
     assertNotNull(ma.parqueaderosAuxiliar);
}
```

Figura 57. Prueba Unitaria: Obtener Parqueaderos de la Base de Datos interna de la aplicación

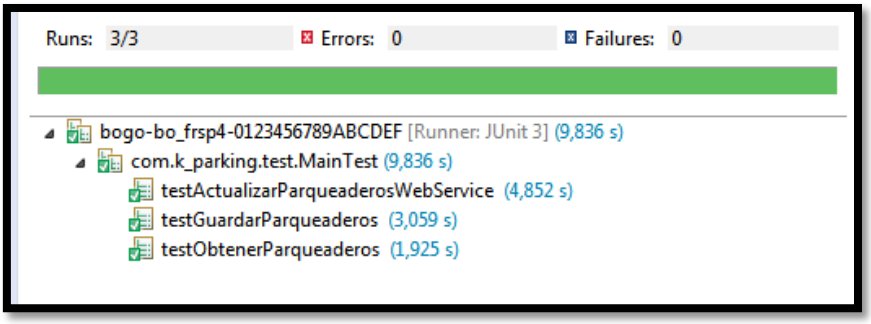


Figura 58. Ejecución de Prueba Unitaria: Obtener Parqueaderos de la Base de Datos interna de la aplicación

5.1.1.4. Caso de Prueba 4: Obtener las coordenadas a través de una dirección ingresada.

Esta prueba sirve para verificar que se están obteniendo las direccionas a través de la consulta al api places. (Ver Tabla XXI)

TABLA XX.

PRUEBA UNITARIA: OBTENCIÓN DE COORDENADAS A TRAVÉS DE UNA DIRECCIÓN

Código	Nombre
PU004	Obtención de coordenadas a través de una dirección
Objetivo	Obtener la lista con los parqueaderos cargados de la aplicación
Pasos	1. Inicializar la actividad. 2. Invocar el método actualizarParqueaderosWebService(tiempo entre cada ejecución) con un tiempo entre ejecución de 300000 milisegundos. 3. Invocar el método obtenerCoordenadas(dirección) con una dirección x y asignar a una cadena. 4. Comprobar que el método devuelve un valor diferente de nulo.
Resultados Esperados	La cadena que recibe el valor del método obtenerCoordenadas no debe ser nulo
Resultados Obtenidos	La cadena que recibe el valor del método obtenerCoordenadas no es nulo

El código utilizado para la elaboración de la prueba está descrito a continuación en la figura 59.

```
@Test
public void testActualizarParqueaderosWebService(){
      MainActivity ma = this.getActivity();
      ma.actualizarParqueaderosWebService(300000);
      String direccion = obtenerCoordenadas("10 de Agosto");
      assertNotNull(ma.parqueaderos);
     assertNotNull(direccion);
```

Figura 59. Prueba Unitaria: Obtener las coordenadas a través de una dirección ingresada

La correcta ejecución del método se presenta en la figura 60.

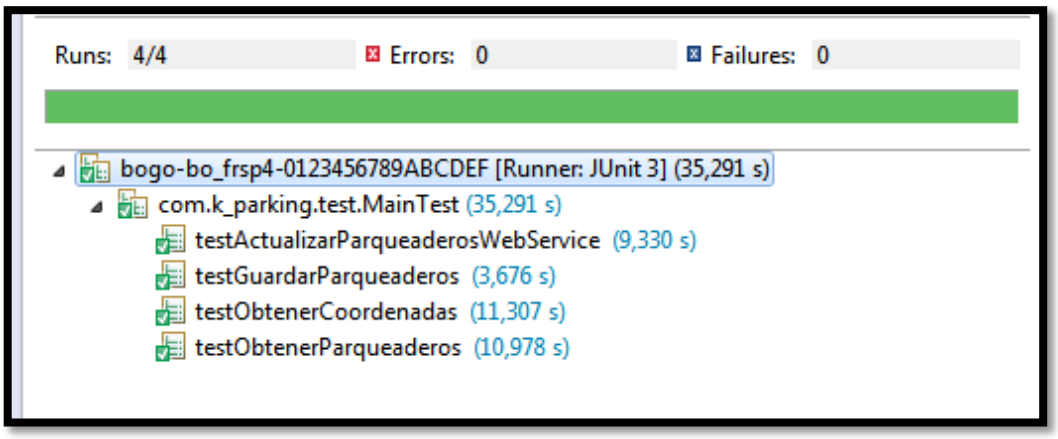


Figura 60. Ejecución de Prueba Unitaria: Obtener las coordenadas a través de una dirección ingresada

5.1.1.5. Caso de Prueba 5: Obtener el parqueadero más cercano

Gracias a este método se puede verificar que la aplicación encuentra el parqueadero más cercano dentro de la aplicación.

TABLA XXI.

PRUEBA UNITARIA: OBTENER EL PARQUEADERO MÁS CERCANO

Código	Nombre
PU001	Obtención del parqueadero más cercano
Objetivo	Obtener el parqueadero más cercano de la aplicación.
Pasos	1. Inicializar la actividad. 2. Invocar el método actualizarParqueaderosWebService(tiempo entre cada ejecución) con un tiempo entre ejecución de 300000 milisegundos. 3. Inicializar la lista de parqueaderos parqueaderosAuxiliar 4. Invocar el método calcularParqueaderoCercano y asignar el valor a un parqueadero. 5. Comprobar que el valor devuelto no sea null.
Resultados Esperados	El objeto parqueaderoCercano no debe ser null
Resultados Obtenidos	El objeto parqueaderoCercano no es null

El código utilizado para la elaboración de la prueba está descrito en la figura 61.

```
@Test
public void testObtenerParqueaderoCercano(){
            MainActivity ma = this.getActivity();
            ma.actualizarParqueaderosWebService(30000);
            ma.obtenerParqueaderos();
            ma.ubicacionActual=false;
            ma.localizacion = new LatLng(43.318703, -1.927353);
            Parqueadero parqueaderoCercano=
ma.calcularParqueaderoCercano(ma.parqueaderosAuxiliar);
            assertNotNull(parqueaderoCercano);
}
```

Figura 61. Prueba Unitaria: Obtener el parqueadero más cercano

La correcta ejecución del método se presenta en la figura 62.

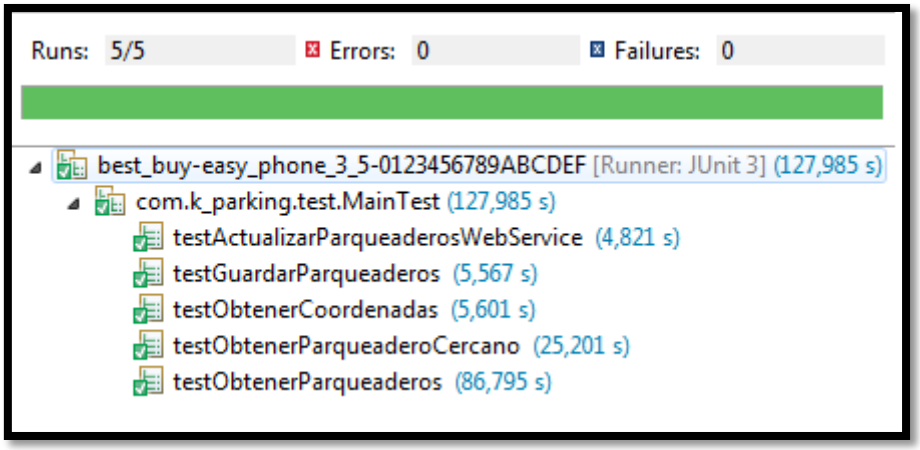


Figura 62. Ejecución de Prueba Unitaria: Obtener el parqueadero más cercano

5.1.2. Pruebas de Interfaz de Usuario

Este tipo de pruebas tienen por objetivo comprobar que el diseño y conexión de los prototipos de pantalla es igual al obtenido en la aplicación desarrollada. Además permite validar el control sobre el ingreso de datos.

5.1.2.1. Verificación de Pantallas

A continuación se describe una tabla en la que se puede verificar que la aplicación posee las mismas pantallas que el Prototipado de pantallas.

TABLA XXII.

VERIFICACIÓN DE PANTALLAS

Prototipos de Pantalla	Pantallas de Aplicación	Cumplimiento
Splash	Splash	✓
Pantalla Principal	Pantalla Principal	✓
Navegación Lateral	Navegación Lateral	✓
Ubicación Actual	Ubicación Actual	✓
Ubicación Alternativa	Ubicación Alternativa	✓
Todos los parqueaderos	Todos los parqueaderos	✓
Acerca de	Acerca de	✓
Búsqueda de Parqueadero	Búsqueda de Parqueadero	✓
Parqueadero más cercano	Parqueadero más cercano	✓

5.1.2.2. Comprobación de Datos Ingresados

Para comprobar que la aplicación realiza un adecuado control sobre la información ingresada hacia esta se desarrollaron pruebas específicas sobre las pantallas que poseen entrada de datos.

En la aplicación las pantallas que poseen un ingreso de datos son: búsqueda alternativa y búsqueda por razón social.

PIU01: Búsqueda Alternativa

Esta pantalla tiene como propósito el obtener una dirección ingresada por el usuario; la dirección puede tener números y letras.

TABLA XXIII.

COMPROBACIÓN DATOS INGRESADOS: BÚSQUEDA ALTERNATIVA

Código	Nombre
PIU01	Control de datos ingresados a la pantalla búsqueda alternativa
Objetivo	Verificar que los datos ingresados a la pantalla búsqueda alternativa cumplan con el formato requerido.
Casos	1. Ingresar un texto con cadenas y letras de acuerdo al texto del servicio de autocompletado. 2. Ingresar un texto con números y letras sin el servicio de autocompletado. 3. Ingresar un texto vacío.
Resultados Esperados	La aplicación debe funcionar correctamente cuando el usuario ingrese una cadena de texto con letras y/o números. En caso de que el texto esté vacío o no se encuentre la dirección, la aplicación debe presentar un mensaje de información.
Resultados Obtenidos	La aplicación funciona correctamente cuando se ingresan letras y números de acuerdo al servicio de autocompletado. En caso de no utilizar el servicio de autocompletado y la consulta no se encuentre o el texto esté vacío se presenta un mensaje.

La ejecución de la prueba realizada se muestra en la figura 63 y figura 64.

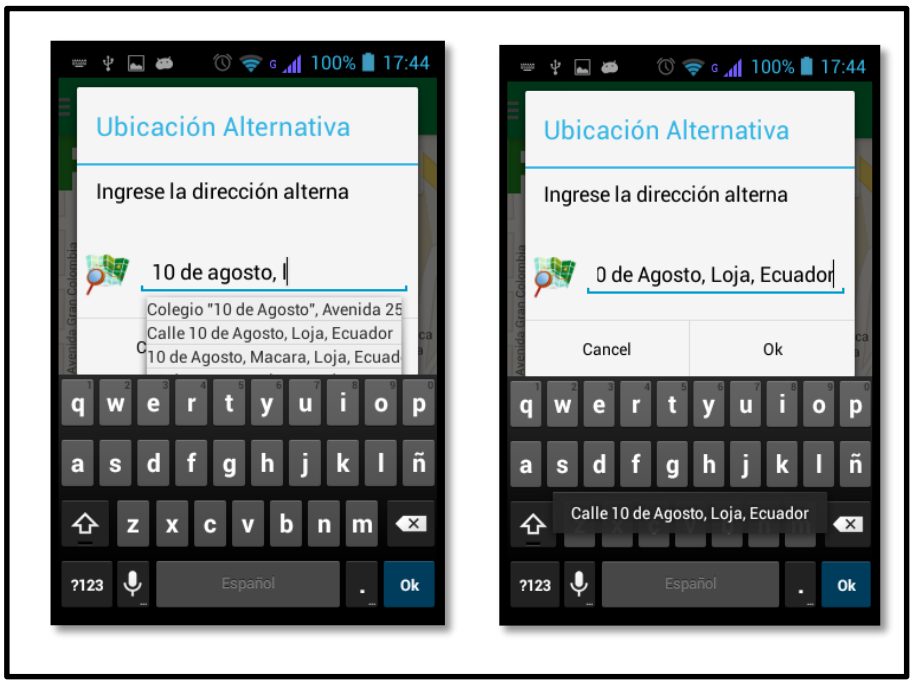


Figura 63. Comprobación de Datos Ingresados:Ubicación Alternativa (a)

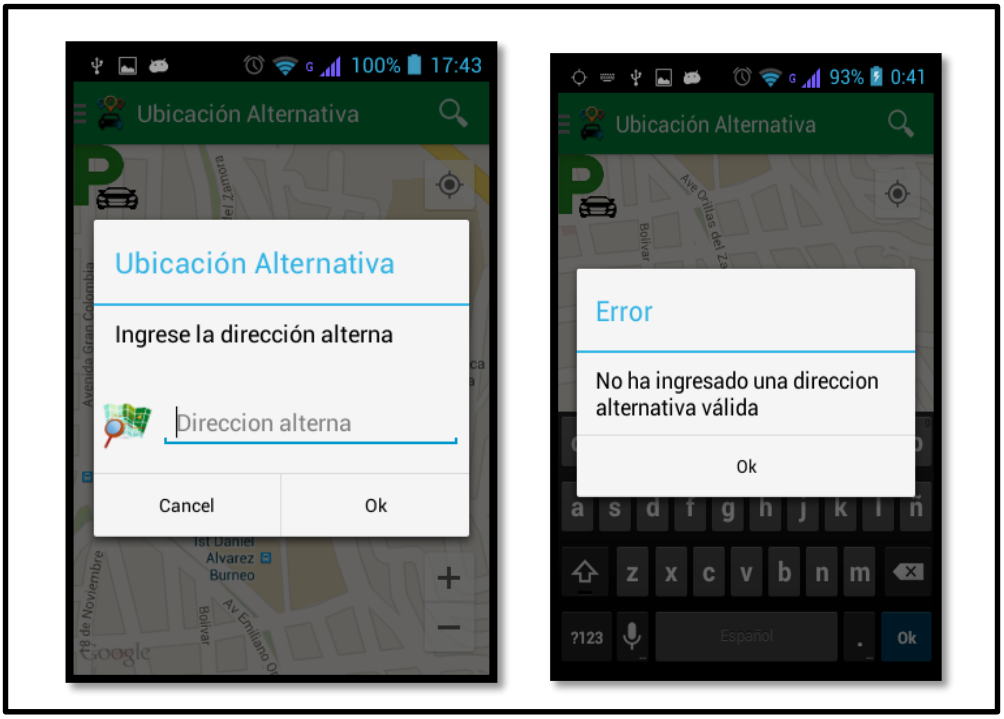


Figura 64 . Comprobación de Datos Ingresados: Busqueda Alternativa (b)

PIU02: Búsqueda de un Parqueadero por su Razón Social

El objetivo de esta pantalla es conseguir la razón social del parqueadero que el usuario desea encontrar.

TABLA XXIV.

COMPROBACIÓN DATOS INGRESADOS: BÚSQUEDA DE UN PARQUEADERO POR SU RAZÓN SOCIAL

Código	Nombre
PIU02	Control de datos ingresados a la pantalla búsqueda de un parqueadero por su razón social.
Objetivo	Verificar que los datos ingresados a la pantalla búsqueda de un parqueadero por su razón social.cumplan con el formato requerido.
Casos	1. Ingresar un texto con cadenas y letras de acuerdo al texto del servicio de autocompletado. 2. Ingresar un texto con números y letras sin el servicio de autocompletado. 3. Ingresar un texto vacío.
Resultados Esperados	La aplicación debe funcionar correctamente cuando el usuario ingrese una cadena de texto con letras y/o números. En caso de que el texto esté vacío o no se encuentre el parqueadero, la aplicación debe presentar un mensaje de información.
Resultados Obtenidos	La aplicación funciona correctamente cuando se ingresan letras y números de acuerdo al servicio de autocompletado. En caso de no utilizar el servicio de autocompletado y la consulta no se encuentre o el texto esté vacío se presenta un mensaje.

La ejecución de la prueba realizada se muestra en la figura 65 y figura 66.

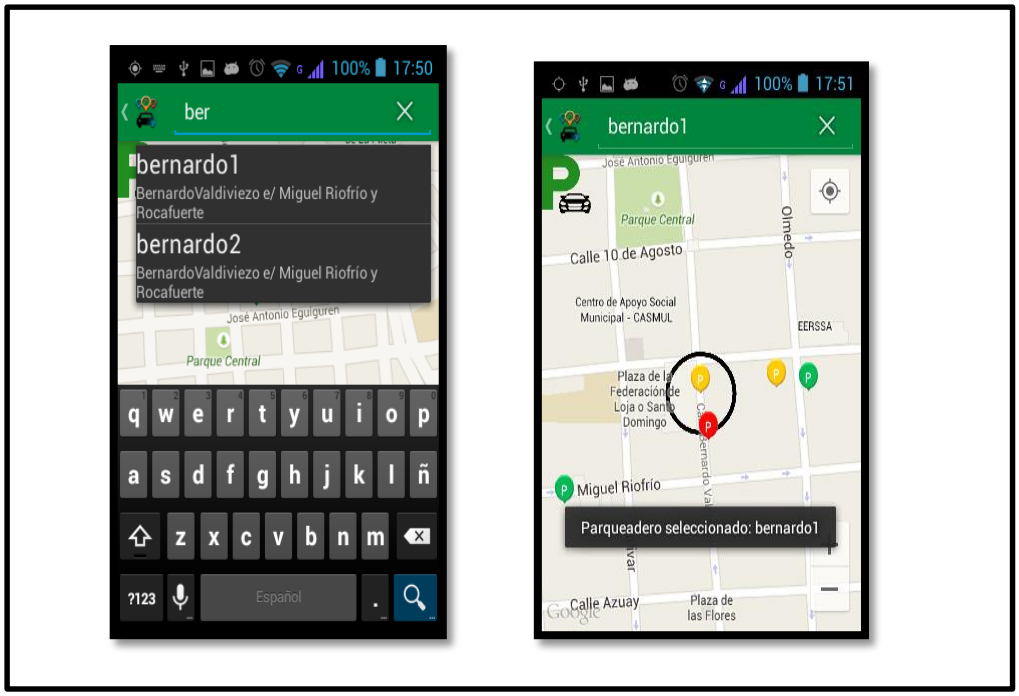


Figura 65 . Comprobación de Datos Ingresados: Búsqueda de un Parqueadero por su Razón Social (a)

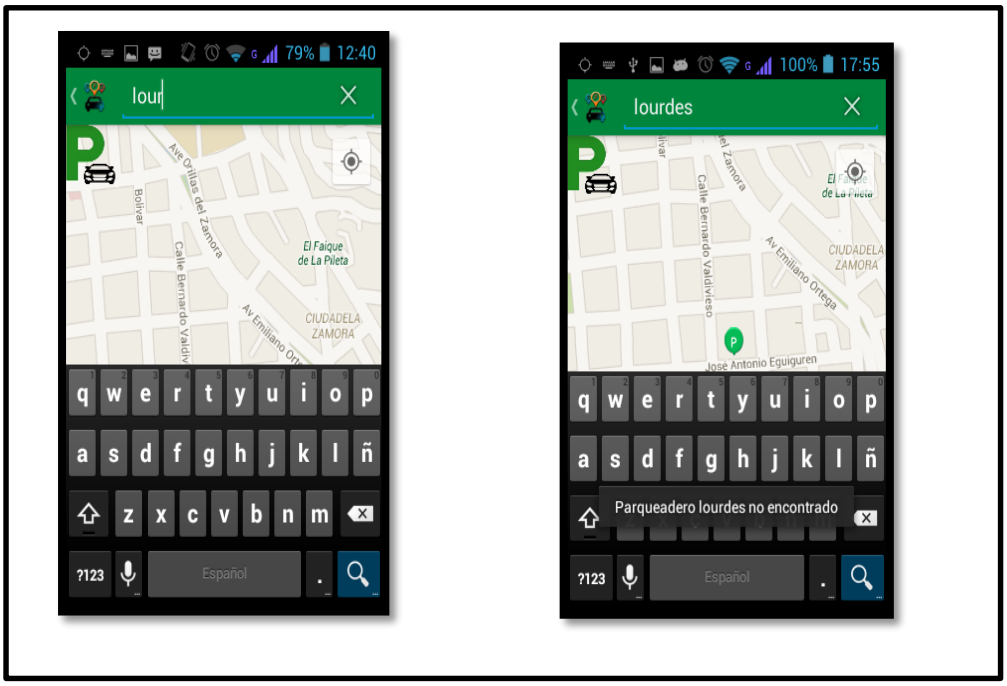


Figura 66. Comprobación de Datos Ingresados: Búsqueda de un Parqueadero por su Razón Social (b)

5.1.3. Pruebas Funcionales

En base a los requerimientos planteados en la etapa de Análisis (Ver tabla n) se procedió a verificar el cumplimiento de cada uno de ellos, de acuerdo al proceso al que pertenecen.

La comprobación de estos requerimientos se hizo de acuerdo a los procesos especificados en la tabla XXVIII.

TABLA XXV.

VERIFICACIÓN DE REQUERIMIENTOS

Código	Descripción	Categoría	Cumplimiento
RF001	Al usuario visualizar su ubicación.	EVIDENTE	✓
RF002	Marcar la ubicación de los parqueaderos	EVIDENTE	✓
RF003	Al usuario buscar una dirección.	EVIDENTE	✓
RF004	Al usuario marcar una ubicación alternativa	EVIDENTE	✓
RF005	Al usuario visualizar la información de un parqueadero (razón social, dirección, plazas disponibles, plazas totales, distancia e imagen).	EVIDENTE	✓
RF006	Al usuario visualizar los parqueaderos de la ciudad de Loja	EVIDENTE	✓
RF007	Al usuario buscar un parqueadero por su razón social	EVIDENTE	✓
RF008	Al usuario buscar el parqueadero más cercano a su localización.	EVIDENTE	✓
RF009	Al usuario buscar el parqueadero más cercano a una localización alternativa.	EVIDENTE	✓
RF010	Calcular la distancia menor desde la ubicación del usuario o una ubicación alternativa hacia un parqueadero.	EVIDENTE	✓
RF011	Trazar la ruta hacia el parqueadero más cercano.	EVIDENTE	✓
RF012	Abrir la aplicación maps con la ruta hacia el parqueadero más cercano desde una ubicación alternativa.	EVIDENTE	✓
RF013	Al usuario utilizar navigation para navegar desde su ubicación hacia el parqueadero más cercano.	EVIDENTE	✓
RF014	Proporcionar información acerca de la aplicación	EVIDENTE	✓

5.1.3.1. PFO1: Visualización de la ubicación actual

TABLA XXVI.

PRUEBA FUNCIONAL: VISUALIZACIÓN DE LA UBICACIÓN ACTUAL

Código	Nombre
PF001	Prueba funcional del Proceso
Objetivo	Comprobar que el usuario puede visualizar la ubicación de su dispositivo.
Pasos	1. El usuario presiona el botón propio de google maps y comprobar que coincide con la ubicación actual 2. El usuario selecciona la opción Ubicación Actual desde el menú lateral y comprobar la ubicación actual.
Resultados Esperados	El usuario podrá visualizar su ubicación desde el botón propio de google maps en la parte derecha superior del mapa o desde la opción ubicación actual del menú lateral de la aplicación.
Resultados Obtenidos	El usuario puede visualizar su ubicación ya sea desde el botón de google maps o desde la opción del menú lateral de la aplicación

La ejecución realizada muestra sus resultados en la figura 67.

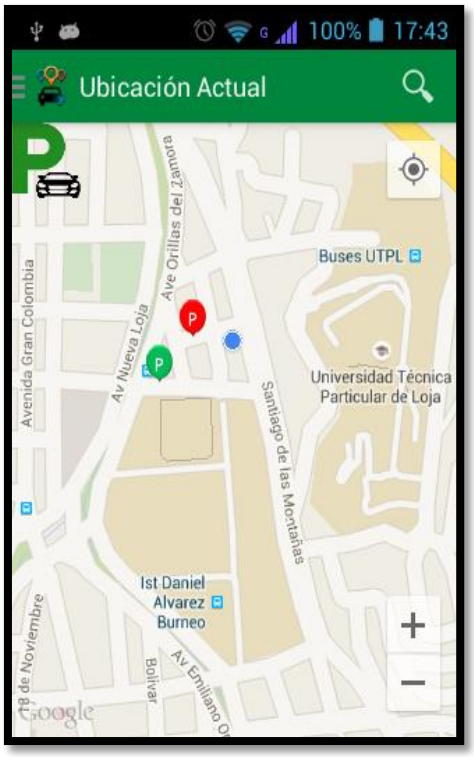


Figura 67. Prueba Funcional: Visualización de la ubicación actual

5.1.3.2. PFO2: Visualización de la ubicación alternativa

TABLA XXVII.

PRUEBA FUNCIONAL: VISUALIZACIÓN DE LA UBICACIÓN ALTERNATIVA

Código	Nombre
PF002	Prueba funcional del Proceso
Objetivo	Comprobar que el usuario puede visualizar una ubicación alternativa ingresada por él en la aplicación.
Pasos	1. El usuario selecciona la opción Ubicación Alternativa desde el menú lateral 2. El usuario ingresa una dirección en el diálogo presentado por la aplicación 3. El usuario da clic en Aceptar 4. El usuario visualiza la ubicación alternativa ingresada.
Resultados Esperados	El usuario podrá visualizar en el mapa, la ubicación correspondiente a la dirección ingresada por él a través de un marcador.
Resultados Obtenidos	El usuario puede observar un marcador en el mapa que representa la dirección ingresada.

La ejecución realizada muestra sus resultados en la figura 70.

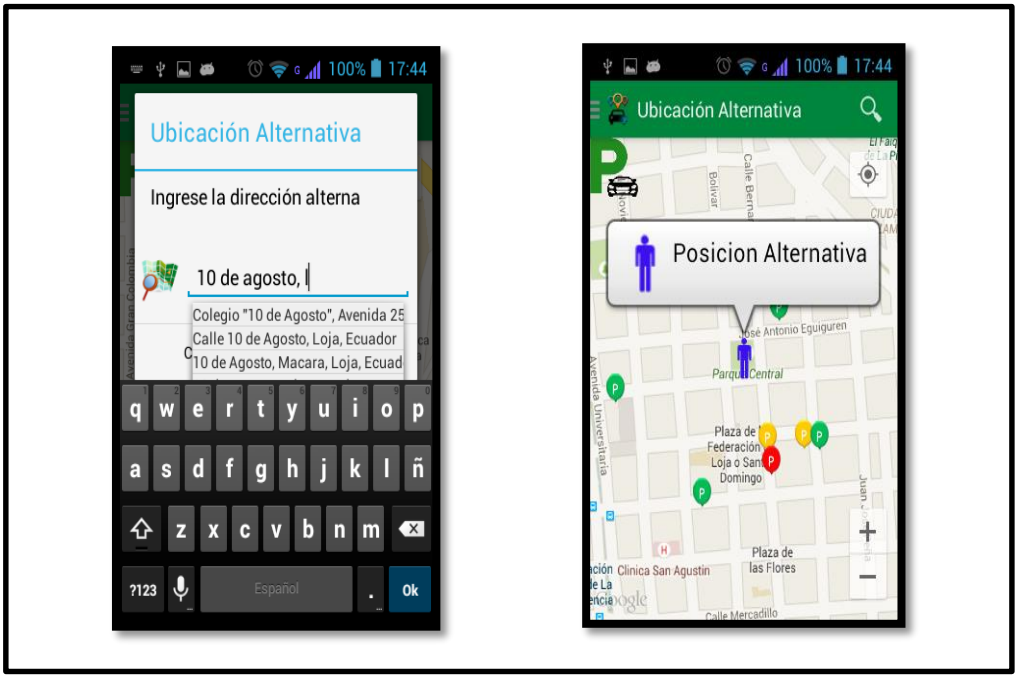


Figura 68. Prueba Funcional: Visualización de la ubicación alternativa

5.1.3.3. PFO3: Visualización de todos los parqueaderos

TABLA XXVIII.

PRUEBA FUNCIONAL: VISUALIZACIÓN DE TODOS LOS PARQUEADEROS

Código	Nombre
PF003	Prueba funcional del Proceso
Objetivo	Comprobar que el usuario puede visualizar todos los parqueaderos registrados en la aplicación.
Pasos	1. El usuario seleciona la opción Todos los parqueaderos desde el menú lateral de la aplicación. 2. El usuario visualiza los parqueaderos marcados en el mapa
Resultados Esperados	El usuario podrá visualizar los parqueaderos registrados en el Sistema, gracias a marcadores que corresponden a su ubicación y con el color de acuerdo a su número de plazas disponibles (verde=más de 10 plazas disponibles, amarillo más de 5 plazas disponibles y menos de 10 y rojo= menos de 5 plazas disponibles.).
Resultados Obtenidos	El usuario puede visualizar los parqueaderos registrados en el Sistema, gracias a marcadores que corresponden a su ubicación y número de plazas disponibles.

La ejecución realizada muestra sus resultados en la figura 69.

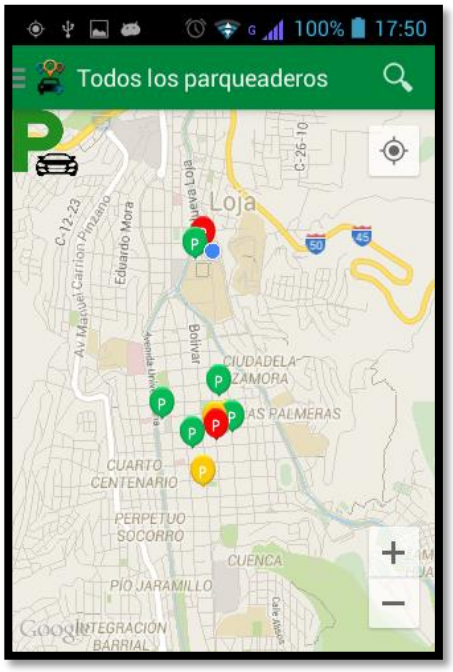


Figura 69. Prueba Funcional: Visualización de todos los Parqueaderos

5.1.3.4. PFO4: Visualización de Información acerca de un parqueadero

TABLA XXIX.

PRUEBA FUNCIONAL: VISUALIZACIÓN DE INFORMACIÓN DE UN PARQUEADERO

Código	Nombre
PF001	Prueba funcional del Proceso
Objetivo	Comprobar que el usuario puede visualizar la información de los parqueaderos (nombre, dirección, plazas totales y número de plazas libres).
Pasos	1. El usuario da click sobre un parqueadero, que está representado por un marcador de un color de acuerdo al número de plazas libres. 2. El usuario visualiza la información del parqueadero.
Resultados Esperados	El usuario podrá visualizar la información del parqueadero al cual señaló en el mapa.
Resultados Obtenidos	El usuario puede visualizar la información del parqueadero al cual señaló en el mapa.

La ejecución realizada muestra sus resultados en la figura 70.

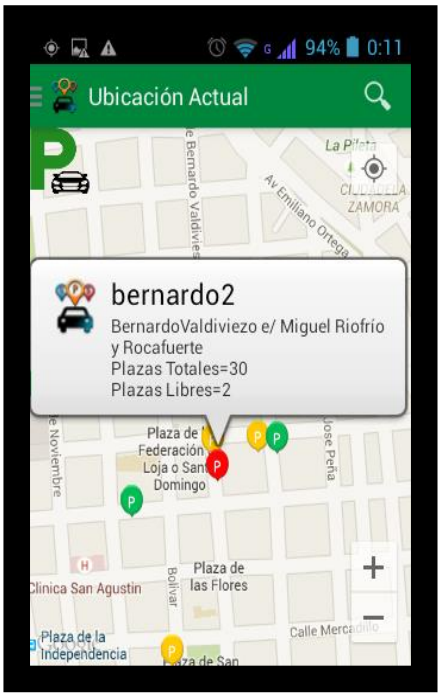


Figura 70. Prueba Funcional: Visualización de la información acerca de un parqueadero

5.1.3.5. PFO5: Búsqueda de un parqueadero por su razón social

TABLA XXX.

PRUEBA FUNCIONAL: BÚSQUEDA DE UN PARQUEADERO POR SU RAZÓN SOCIAL

Código	Nombre
PF005	Prueba funcional del Proceso
Objetivo	Comprobar que el usuario puede buscar un parqueadero por su nombre.
Pasos	1. El usuario da clic en el icono de buscar de la barra superior de la aplicación. 2. El usuario ingresa el nombre del parqueadero y selecciona el parqueadero buscado. 3. El usuario visualiza el parqueadero buscado enmarcado en una circunferencia.
Resultados Esperados	El usuario podrá buscar y seleccionar un parqueadero de acuerdo a su razón social.
Resultados Obtenidos	El usuario puede buscar y seleccionar un parqueadero de acuerdo a su razón social.

La ejecución realizada muestra sus resultados en la figura 71.

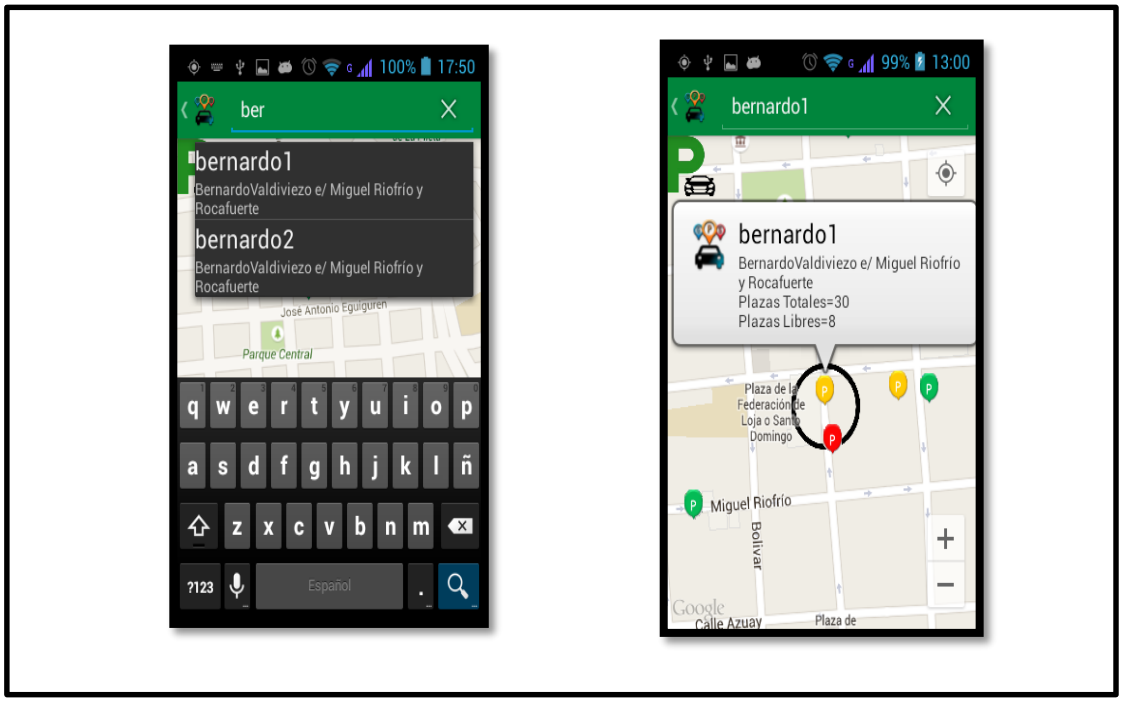


Figura 71. Prueba Funcional: Búsqueda de un parqueadero por su razón social

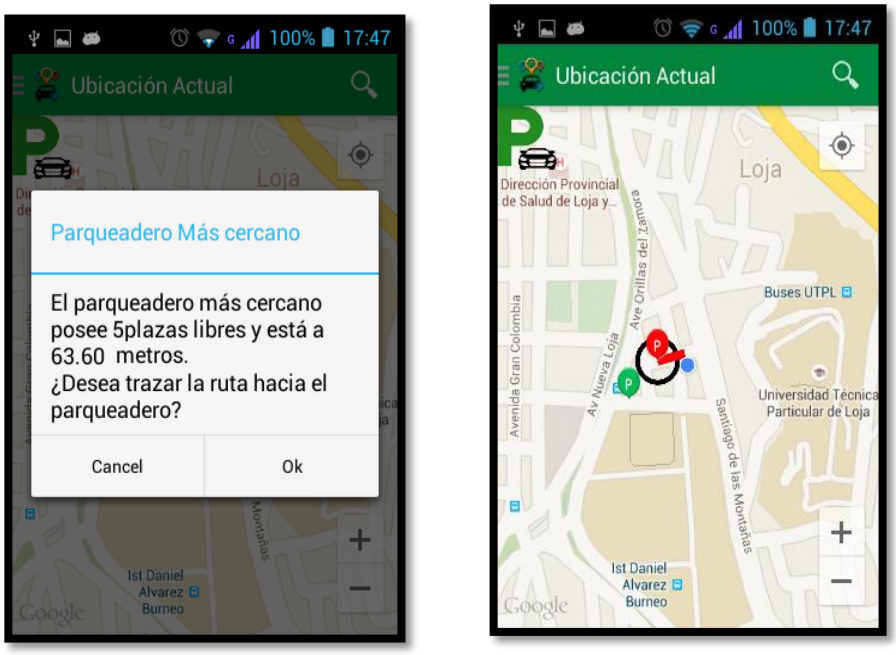
5.1.3.6. PFO6: Búsqueda de un parqueadero Cercano desde Ubicación Actual

TABLA XXXI.

PRUEBA FUNCIONAL: BÚSQUEDA DE UN PARQUEADERO CERCANO DESDE UBICACIÓN ACTUAL

Código	Nombre
PF006	Prueba funcional del Proceso
Objetivo	Comprobar que el usuario puede buscar un parqueadero cercano, desde la ubicación de dispositivo.
Pasos	1. El usuario elige ubicación actual desde el botón de google maps situado en la parte superior derecha del mapa, o desde la opción Ubicación Actual del Menú Lateral (Ver Prueba Funcional P001). 2. El usuario da click en el botón superior izquierdo del mapa para encontrar el parqueadero más cercano. 3. El usuario da clic en Aceptar para trazar la ruta hacia el parqueadero o da clic en Cancelar si desea visualizar el mapa sólo enmarcado en una circunferencia. 4. El usuario visualiza el parqueadero enmarcado en una circunferencia o con la ruta trazada hasta su ubicación; de acuerdo a su elección anterior. 5. El usuario visualiza una ventana de diálogo con la opción de visualizar la ruta en Google Navigation. El usuario da clic en Aceptar para hacer uso de Navigation o da clic en Cancelar.
Resultados Esperados	El usuario podrá buscar el parqueadero más cercano a la ubicación del dispositivo y visualizarlo enmarcado en una circunferencia, y/o con la ruta trazada entre los dos puntos; además de poder hacer uso de Google Navigation.
Resultados Obtenidos	El usuario puede buscar el parqueadero más cercano a la ubicación del dispositivo y visualizarlo enmarcado en una circunferencia, y/o con la ruta trazada entre los dos puntos; además puede hacer uso de Google Navigation.

La ejecución realizada muestra sus resultados en la figura 72.



5.1.3.7. PFO7: Búsqueda de un parqueadero Cercano desde Ubicación Alternativa.

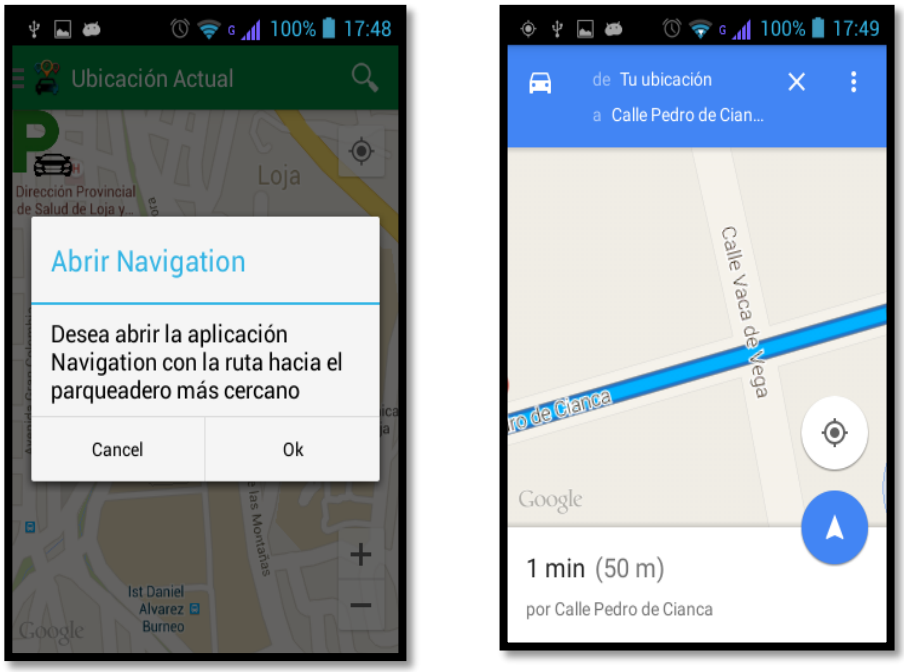


Figura 72. Prueba Funcional: Búsqueda de un parqueadero cercano desde Ubicación Actual

TABLA XXXII.

PRUEBA FUNCIONAL: BÚSQUEDA DE UN PARQUEADERO CERCANO DESDE UBICACIÓN ALTERNATIVA

Código	Nombre
PF007	Prueba funcional del Proceso
Objetivo	Comprobar que el usuario puede buscar un parqueadero cercano, desde una ubicación ingresada por el usuario.
Pasos	1. El usuario elige ubicación alternativa desde la opción Ubicación Actual del Menú Lateral (Ver Prueba Funcional P002). 2. El usuario da click en el botón superior izquierdo del mapa para encontrar el parqueadero más cercano. 3. El usuario da clic en Aceptar para trazar la ruta hacia el parqueadero o da clic en Cancelar si desea visualizar el mapa sólo enmarcado en una circunferencia. 4. El usuario visualiza el parqueadero enmarcado en una circunferencia o con la ruta trazada hasta su ubicación; de acuerdo a su elección anterior. 5. El usuario visualiza una ventana de diálogo con la opción de visualizar la ruta en Google Maps. El usuario da clic en Aceptar para hacer uso de Maps o da clic en Cancelar.
Resultados Esperados	El usuario podrá buscar el parqueadero más cercano a la ubicación del dispositivo y visualizarlo enmarcado en una circunferencia, y/o con la ruta trazada entre los dos puntos; además de poder hacer uso de Google Maps.
Resultados Obtenidos	El usuario puede buscar el parqueadero más cercano a la ubicación del dispositivo y visualizarlo enmarcado en una circunferencia, y/o con la ruta trazada entre los dos puntos; además puede hacer uso de Google Maps.

La ejecución realizada muestra sus resultados en la figura 73.

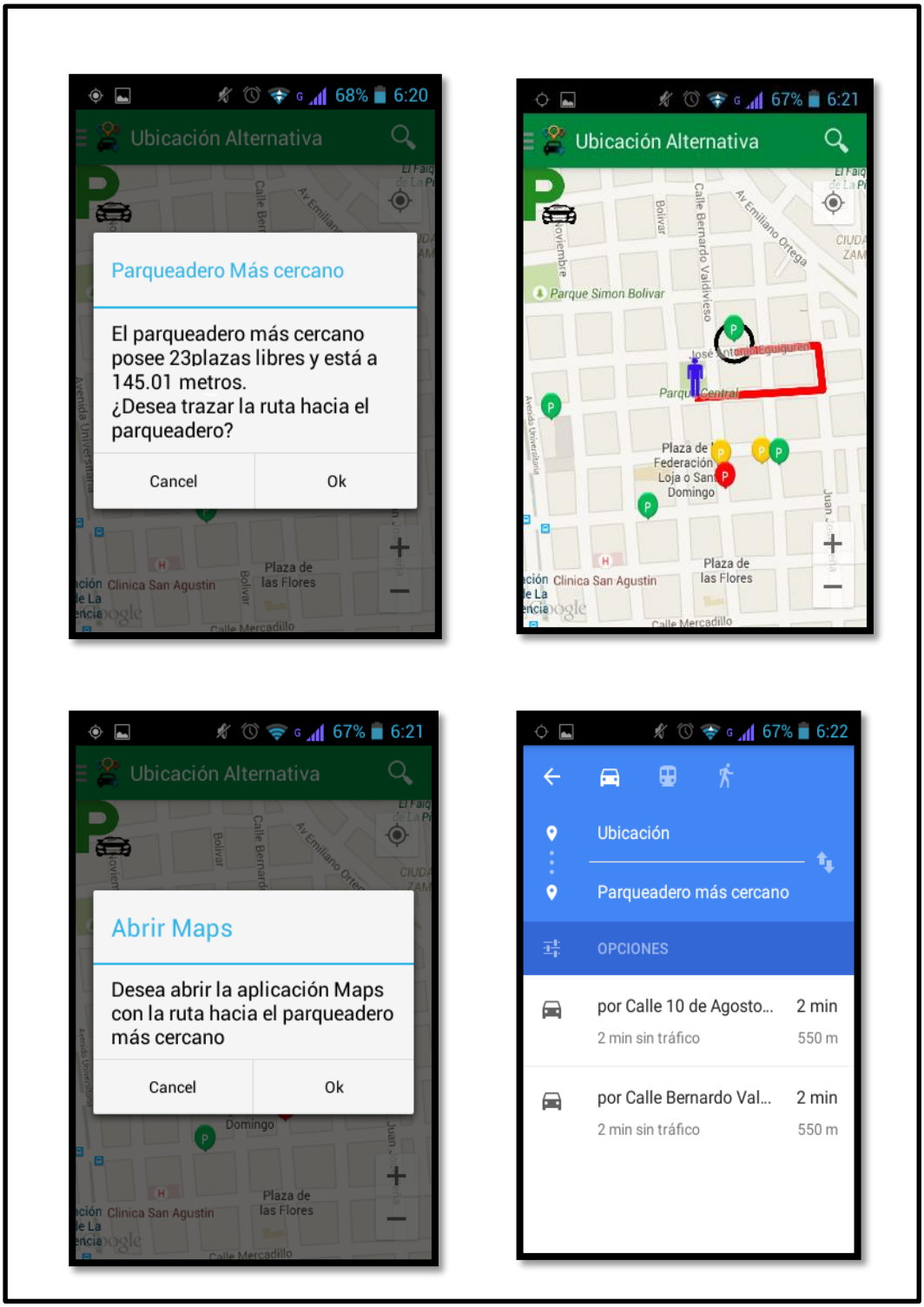


Figura 73. Búsqueda de un parqueadero cercano desde Ubicación Alternativa

5.1.4. Dispositivos Permitidos

Para comprobar que la aplicación se ejecuta correctamente en varios dispositivos, se realizó una prueba con la ayuda de la herramienta online apkudo[] la cual permite obtener una lista de los dispositivos móviles en los cuales una aplicación funciona sin ningún contratiempo gracias a la ayuda de Mono Running, una herramienta utilizada para determinar la compatibilidad de la aplicación con los diferentes tipos de versiones de Android y de modelos de celulares. En la tabla n se presentan los dispositivos en los que la aplicación funcionó adecuadamente.

TABLA XXXIII.

DISPOSITIVOS PERMITIDOS

Marca	Nombre	Modelo	Versión de Sistema Operativo Android
HTC	G2	PC10100	2.3.4
	Nexus One	PB99100	2.3.6
	One V	T-320e	4.0.3
	One X LTE	PJ83100	4.0.3
	One S	HTC One S	4.0.4
	One X	S720E	4.0.4
LG	Optimus	LGMS695	2.3.6
	Optimus L3	E400	2.3.6
	Viper 4G LTE	LS-840	2.3.7
	Optimus Elite	LG-LS696	2.3.7
	L5	E610	4.0.3
	Intuition	VS950 4G	4.1.2
	G2	VS980	4.4.2
	Nexus 4	E960	4.4.4
Motorola	Electrify M	XT901	4.0.4
	Droid Razr Maxx HD	XT926	4.4.2
	Moto X (2013)	XT1053	4.4.4
Samsung	Galaxy Pocket	GT-S5300	2.3.6
	Rugby Smart	SGH-I847	2.3.6
	Nexus S	SPH-D720	2.3.6
	Galaxy Express	SGH-I437	4.0.4
	Galaxy SIII	SGH-I747	4.1.1
	Galaxy SII	GT-I9100	4.1.2
	Epic 4G Touch	SPH-D710	4.1.2
	Nexus S	GT-I9020T	4.1.2
	Galaxy Note II	SGH-I317	4.3

g. Discusión

En el siguiente apartado se pretende mostrar un análisis acerca de los resultados obtenidos con el desarrollo del presente trabajo de titulación

1. Desarrollo de la propuesta alternativa

Actualmente, existe un gran congestionamiento a la hora de alquilar un espacio disponible debido a la gran cantidad de vehículos que utilizan este servicio, sobretodo en los parqueaderos que suelen ser más conocidos provocando una gran aglomeración [1]

En los lugares donde se encuentran laborando las principales instituciones públicas y privadas de una ciudad o provincia existe una gran ausencia de personas y por ende de vehículos que causan que la mayoría de plazas de los parqueaderos se encuentren ocupadas y que los conductores inviertan una gran cantidad de tiempo circulando por estos estacionamientos en búsqueda de algún espacio disponible. [2]

Los conductores suelen dirigirse a ciertos sitios específicos donde conocen algunos parqueaderos en los cuales pueden encontrar estacionamiento, pero estos no son todos los que existen, además, al dirigirse con su vehíículo a sitios nuevos existe un desconocimiento de los parqueaderos cercanos a dicho lugar. [2]

En consecuencia, las personas se encuentran con una serie de problemas para hallar una plaza disponible en un parqueadero, entre los cuales se evidencia el congestionamiento en los parqueaderos más conocidos, el desconocimiento de los parqueaderos cercanos y la pérdida de tiempo buscando una plaza disponible.[1]

Ante ello, se ha propuesto la realización del presente trabajo de titulación como una posible solución ante esta problemática.

1.1. Objetivo Específico 1: Desarrollar una aplicación móvil Android para la obtención de los parqueaderos.

Para llevar a cabo este objetivo se realizó la fase de análisis en dónde se determinaron los requerimientos que debía cumplir la aplicación, los procesos que podría realizar el

usuario con la aplicación, el tipo de aplicación, sus limitaciones, dependencias y la configuración de los recursos físicos y técnicos necesarios para desarrollar la aplicación.

Posteriormente, en la etapa de diseño se realizó la configuración del ambiente de desarrollo, indispensable para la posterior codificación; asimismo se utilizó la información obtenida en la etapa anterior y en el funcionamiento del Sistema de Parqueaderos y así diseñar la futura aplicación; es decir construir el modelo del dominio, el modelo entidad-relación, el diseño de su funcionamiento, analizar cada función que el usuario puede realizar, para obtener sus pre-requisitos, sus pantallas y el storycard de cada una de ellas.

Luego, en la etapa de codificación se desarrollaron todas las pantallas de la aplicación, junto con la conexión entre cada una de ellas con la ayuda de la librería SherlockActionBar con el fin de que el action bar sea visible y funcional en versiones anteriores de Android. También se realizó la conexión hacia el web Service para obtener la información de los parqueaderos y la configuración de una base de datos interna que sea utilizada mientras ocurre la próxima actualización de datos de los parqueaderos o en caso de problemas con el web Service.

1.2. Objetivo Específico 2: Implementar el módulo de Geolocalización para la visualización de los parqueaderos.

Éste objetivo fue cumplido con ayuda de la recopilación de la información y de la fase de codificación.

Para conseguirlo se hizo uso de los datos obtenidos desde el web Service, específicamente de la latitud y longitud con el objetivo de agregar cada parqueadero como un marcador dentro del mapa; estos marcadores son de un color correspondiente al número de plazas libres, es decir, rojo cuando hay 5 o menos plazas libres, 10 o menos plazas libres tendrán un color amarillo y el color verde para más de 10 plazas libres.

Igualmente, como parte de este objetivo se implementaron las funciones de visualización de la ubicación del usuario, de una ubicación alternativa, de todos los parqueaderos.

Para desarrollar la función de la visualización de una ubicación alternativa se hizo uso de place autocomplete de google con el fin de ofrecer al usuario el servicio de autocompletado automático de una dirección; aunque también se creó la opción de que pudiese hacerlo mediante una pulsación larga dentro del mapa.

1.3. Objetivo Específico 3: Desarrollar el módulo de búsqueda y rutas de plazas disponibles

El cumplimiento de este objetivo fue posible gracias al api de direcciones de google. Con el fin de cumplir este objetivo se desarrolló la función de búsqueda de un parqueadero por su razón social para lo cual fue necesario utilizar el action bar de la interfaz y la base de datos interna de la aplicación.

Otra de las funciones realizadas fue la búsqueda de un parqueadero cercano y el trazado de la ruta desde la ubicación del dispositivo o desde una ubicación alternativa hacia el parqueadero encontrado, esto mediante la utilización del api de direcciones de google y el trazado de polilíneas.

Además, se permite al usuario la opción de visualizar la ruta trazada hacia el parqueadero desde una ubicación alternativa a través de la aplicación maps y desde la ubicación del dispositivo mediante la aplicación navigation de Android.

Finalmente, la fase de pruebas permitió comprobar el funcionamiento de los procesos desarrollados como parte del cumplimiento de los objetivos.

2. Valoración técnica económica ambiental

Los resultados del presente trabajo de titulación contribuyen positivamente en los aspectos técnico, económico y ambiental.

En el aspecto técnico, el presente proyecto puede ser utilizado para consulta de los desarrolladores que deseen utilizar los mismos recursos técnicos como por ejemplo, los servicios de google, consultas hacia un webservice, construcción y consulta de una base de datos interna de una aplicación.

Además, sirve como base de futuros trabajos relacionados con la utilización de los servicios de google como por ejemplo visualización de diferentes locales comerciales de interés público, reservación de servicios de transporte, hotelería; entre otros.

En cambio, en el aspecto económico brinda una gran oportunidad a los propietarios de los parqueaderos ya que la aplicación desarrollada podría aumentar la posibilidad de que las plazas de su parqueadero sean alquiladas por los conductores de vehículos cercanos a su ubicación.

De la misma forma, los conductores ahorrarían tiempo al encontrar un parqueadero con plazas libres de forma más rápida; destinando el tiempo ahorrado a otras actividades de su beneficio.

Por último, la aplicación desarrollada como posible solución a la problemática encontrada ayuda en el aspecto ambiental puesto que no utiliza papel para su ejecución ni realiza mecanismos o procesos industriales que afecten al medio ambiente.

Asimismo, el hecho de que los conductores encuentren en un menor tiempo plazas disponibles en un parqueadero minimiza el tiempo que éstos estarán recorriendo la ciudad mientras emiten dióxido de carbono.

h. Conclusiones

Luego de haber realizado el presente trabajo de titulación se ha llegado a las siguientes conclusiones:

- La implementación de librerías en el desarrollo de aplicaciones móviles Android optimizan el tiempo del programador y pueden mejorar el desempeño de las mismas.

- Los servicios de Google relacionados con la localización y direcciones son de gran utilidad en el desarrollo de aplicaciones móviles puesto que al ser implementados de forma adecuada permiten optimizar el tiempo de los usuarios.

- La búsqueda de parqueaderos por la razón social o dirección permitirá a los usuarios acceder más fácilmente a los parqueaderos de su preferencia y la visualización de plazas libres optimizar tiempo en la búsqueda de un espacio disponible.

- La realización de las pruebas son fundamentales para la obtención de un producto de calidad; incluyendo la utilización de herramientas online como por ejemplo Mono Running.

- El proceso desarrollado para la culminación del trabajo de titulación fue plasmado de forma detallada en diferentes archivos, siendo estos la memoria final, el manual técnico, y el artículo científico, además se procedió a enviar el artículo a la revista MASKANA (Universidad de Cuenca), indexada en la base de datos de Latindex para su respectiva publicación.

i. Recomendaciones

Una vez finalizada la investigación se cree conveniente plantear las siguientes recomendaciones:

- Al usuario comprobar la correcta conexión a internet y habilitación del gps para un correcto funcionamiento de la aplicación desarrollada.

- Al usuario utilizar la aplicación como una herramienta informativa que le permita optimizar su tiempo.

- La utilización de herramientas como apkudo permiten mejorar la calidad de las aplicaciones, puesto que ayudan a detectar errores en la ejecución de la aplicación en poco tiempo.

- Incorporar nuevas funcionalidades a la aplicación como por ejemplo la reserva de plazas disponibles de un parqueadero con el fin de optimizar el tiempo del usuario y ampliar su alcance, es decir agregar datos de otras ciudades.

j. Bibliografía

[1] MARTÍNEZ GONZÁLEZ Felipe Luis; “Aplicaciones para dispositivos móviles”. 2010. [Online]. Disponible en: https://riunet.upv.es/bitstream/handle/10251/11538/Memoria.pdf?sequence=1. [Último acceso: 28 abril 2014].

[2] MORILLO POZO Julián David, “Introducción a los dispositivos móviles”, 2012. [Online]. Disponible en: http://www.exabyteinformatica.com/uoc/Informatica/Tecnologia_y_desarrollo_en_dispositivos_moviles/Tecnologia_y_desarrollo_en_dispositivos_moviles_(Modulo_2).pdf. [Último acceso: 28 abril 2014].

[3] COMISIÓN FEDERAL DE COMERCIO; “Aplicaciones móviles: Qué son y cómo funcionan”; 2013; [Online]; https://www.alertaenlinea.gov/articulos/pdf-s0004_0.pdf. [Último acceso: 28 abril 2014].

[4] WURFL; “Página official de WURFL (Wireless Universal Resource FiLe)”; 2014; [Online]. Disponible en: http://wurfl.sourceforge.net/. [Último acceso: 28 abril 2014].

[5] RAMÍREZ VIQUE Robert; “Métodos para el desarrollo de aplicaciones móviles”; 2012. [Online]. Disponible en: http://www.exabyteinformatica.com/uoc/Informatica/Tecnologia_y_desarrollo_en_dispositivos_moviles/Tecnologia_y_desarrollo_en_dispositivos_moviles_(Modulo_4).pdf. [Último acceso: 28 abril 2014].

[6] Xojo; “Página official de Xojo”; 2014; [Online]. Disponible en: https://xojo.com/. [Último acceso: 28 abril 2014].

[7] CAMPO Celeste, GARCIA RUBIO Carlos; “Sistemas Operativos de Dispositivos Móviles”; 2012; Universidad Carlos III de Madrid; 2012. [Online]. Disponible en: http://ocw.uc3m.es/ingenieria-telematica/aplicaciones-moviles/material-de-clase-2/sistemas-operativos. [Último acceso: 28 abril 2014].

[8 DOMÍNGUEZ Héctor, PINTO Krysler; “Sistemas Operativos de Dispositivos Móviles”; Universidad Simón Bolívar; 2013; [Online]. Disponible en:

http://ldc.usb.ve/~yudith/docencia/ci-4821/Temas/Exposicion_OS_MovilesKryslerHernan.pdf. [Último acceso: 29 abril 2014].

[9] APPLE; "Arquitectura de iOS"; [Online]. Disponible en: http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSOverview/iPhoneOSoverview.html#//apple_ref/doc/uid/TP40007898-CH4-SW1. [Último acceso: 29 abril 2014].

[10] CALVO Roberto; "Dispositivos Móviles"; Escuela de Organización Industrial, 2012; [Online]. Disponible en: http://api.eoi.es/api_v1_dev.php/fedora/asset/eoi:75585/componente75583.pdf. [Último acceso: 29 abril 2014].

[11] APONTE GOMEZ Sanly, DAVILA RAMIREZ Carlos; "Sistemas Operativos Móviles: Funcionalidades, efectividad y aplicaciones útiles en Colombia"; Universidad EAN; Bogotá; 2011; [Online]. Disponible en: http://repository.ean.edu.co/bitstream/10882/761/3/AponteSanly2011.pdf. [Último acceso: 29 abril 2014].

[12] Microsoft Develover Network; "Mi primera aplicación para Windows Phone 7 Series". [Online]. Disponible en: http://msdn.microsoft.com/es-es/library/jj130729.aspx. [Último acceso: 29 abril 2014].

[13] BARROS BUSTAMANTE Julia Marisol; "Investigación Análisis y Pruebas de los Sistemas Operativos para Equipos Móviles"; Universidad Tecnológica Israel; Cuenca; 2010; [Online]. Disponible en: http://186.42.96.211:8080/jspui/bitstream/123456789/670/1/Tesis%20Marisol%20Barros.pdf. [Último acceso: 29 abril 2014].

[14] GARRIDO COBO Juan; "TFC Desarrollo de Aplicaciones Móviles"; 2013. [Online]. Disponible en: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/18528/6/jugarridocoTFC0113memoria.pdf. [Último acceso: 29 abril 2014].

[15] "International Data Corporation; Cuota de mercado en todo el mundo de sistemas operativos de smarthphones"; 2014. [Online]. Disponible en:

http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Último acceso: 29 abril 2014].

[16] Ditrendia Digital Marketing Trends; “Informe Ditrendia:Mobile en España y en el Mundo”; 2014, [Online]. Disponible en: http://es.slideshare.net/ditrendia/informe-ditrendia-mobile-en-espaa-y-en-el-mundo. [Último acceso: 29 abril 2014].

[17] DÍAZ Gilberto; “Planificador de Linux (Scheduler)”; Universidad de los Andes; Mérida, 2013. [Online]. Disponible en: http://webdelprofesor.ula.ve/ingenieria/gilberto/so/02_Planificador.pdf. [Último acceso: 29 abril 2014].

[18] CABRERA Jorge, VILLA Pablo; “Análisis de uso de dispositivos celulares para el apoyo a la gestión empresarial, creación de un prototipo”; Universidad Politécnica Salesiana; Cuenca; 2012. [Online]. Disponible en: http://dspace.ups.edu.ec/bitstream/123456789/3267/6/UPS-CT002535.pdf. [Último acceso: 29 abril 2014].

[19] MENESES Esteban; “Estructuras de Datos 2 – Árboles Rojo-Negro”; Universidad Cenfotec; San José; 2003. [Online]. Disponible en: http://www.oocities.org/emenesesr/recursos/arbolesRojoNegro.pdf. [Último acceso: 29 abril 2014].

[20] GARBUSI Pablo; “Web Services”; Laboratorio de Integración de Sistemas – Universidad de la República Uruguay; Montevideo; 2012. [Online]. Disponible en: http://www.fing.edu.uy/inco/cursos/tsi/TSI2/2012/teorico/tsi2-07-web-services.pdf. [Último acceso: 30 abril 2014].

[21] “Web Services con c# - Principios de Web Services”. [Online]. Disponible en: http://www.bdp.com.bo/sistema/usuarios/archivos/capitulogratis.pdf. [Último acceso: 30 abril 2014].

[22] LOS SANTOS ARANSAY Alberto; “Revisión de los Servicios Web SOAP/REST: Características y Rendimiento”; Universidad de Vigo; Vigo; 2009; [Online]. Disponible en: http://www.albertolsa.com/wp-content/uploads/2009/07/mdsw-revision-de-los-servicios-web-soap_rest-alberto-los-santos.pdf. [Último acceso: 30 abril 2014].

[23] NAVARRO MARSET Rafael; Rest Vs Web Services; 2006. [Online]. Disponible en: http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf. [Último acceso: 30 abril 2014].

[24] BELTRÁN LÓPEZ Gersón; La Geolocalización social como herramienta de innovación empresarial en el desarrollo de los destinos turísticos; Universidad de Valencia; Valencia; 2011. [Online]. Disponible en: http://rua.ua.es/dspace/bitstream/10045/20821/1/Seminario_Renovestur_25.pdf. [Último acceso: 30 abril 2014].

[25] ALCÓN AYUSO Julio; ARAUZ MÉNDEZ Francisco Javier, CARMONO BERRIGUETE Iván; Aplicación web para la geolocalización y monitorización en tiempo real de los recursos integrantes de una red Grid; Universidad Complutense de Madrid; Madrid. 2009. [Online]. Disponible en: http://eprints.ucm.es/9085/1/Memoria.pdf. [Último acceso: 30 abril 2014].

[26] EnGoogle.Net, Que es Google Places. [Online]. Disponible en: link:http://www.googleplaces.es/que-es-google-places. [Último acceso: 30 abril 2014].

[27] Google Developers, Autocompletado de sitios. [Online]. Disponible en: link:https://developers.google.com/places/documentation/autocomplete?hl=es. [Último acceso: 31 abril 2014].

[28] Google Developers, El API de rutas de Google, [Online]. Disponible en: link:https://developers.google.com/maps/documentation/directions/?hl=eses. [Último acceso: 31 abril 2014].

[29] Google Developers, Google Maps for Android, [Online]. Disponible en: link:http://www.google.com/mobile/maps/. [Último acceso: 31 abril 2014].

[32] Beck, K.. “Extreme Programming Explained. Embrace Change”, Pearson Education, 1999. Traducido al español como: “Una explicación de la programación extrema. Aceptar el cambio”, Addison Wesley, 2000.

[33] LETELIER Patricio, PENADÉS María Carmen; Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP); Universidad Politécnica de

Valencia; Valencia. [Online]. Disponible en: http://www.willydev.net/descargas/masyxp.pdf. [Último acceso: 31 abril 2014].

[34] GUZMÁN ÁVILA Andrés Alejandro; Desarrollo de un sistema de puntos de ventas para micromercados, utilizando la metodología extreme programming; Escuela Politécnica del Ejército; Sangolquí; 2008. [Online]. Disponible en: http://repositorio.espe.edu.ec/bitstream/21000/848/1/T-ESPE-021852.pdf. [Último acceso: 31 abril 2014].

[35] PALACIO JUAN, Flexibilidad con Scrum, [Octubre – 2008], [20/03/2012]. [Online]. Disponible en: http://www.navegapolis.net/files/Flexibilidad_con_Scrum.pdf. [Último acceso: 30 abril 2014].

[36] RODRIGUEZ Tshema; Métodos aplicables para el desarrollo de aplicaciones móviles; 2011. [Online]. Disponible en: http://www.genbetadev.com/desarrollo-aplicaciones-moviles/metodos-aplicables-para-el-desarrollo-de-aplicaciones-moviles. [Último acceso: 31 abril 2014].

[37] BLANCO Paco, CAMARERO Julio, FUMERO Antonio, WERTERSKI Adam, RODRIGUEZ Pedro; Metodología Mobile D; Universidad Politécnica de Madrid; Madrid; 2009. [Online]. Disponible en: http://www.adamwesterski.com/wp-content/files/docsCursos/Agile_doc_TemasAnv.pdf. [Último acceso: 31 abril 2014].

[38] AGILE;Mobile-D; Metodologocía Mobile D. [Online]. Disponible en: http://agile.vtt.fi/mobiled.html. [Último acceso: 31 abril 2014].

k. Anexos

1. Anexo 1. Artículo Científico

Búsqueda de plazas disponibles en un parqueadero mediante una aplicación móvil

M. Chinchay, and H. Paz.

***Abstract*—The article presents the development of an application that can display the parking spaces available in the city of Loja and locate the closest site.**

***Index Terms*—Android, parqueadero, parking, aplicación, localizacion, Loja .**

I. INTRODUCCIÓN

Actualmente, existe un gran congestionamiento a la hora de alquilar un espacio disponible debido a la gran cantidad de vehículos que utilizan este servicio, sobretodo en los parqueaderos que suelen ser más conocidos provocando una gran aglomeración. [1]

En los lugares dónde se encuentran laborando las principales instituciones públicas y privadas de una ciudad o provincia existe una gran auencia de personas y por ende de vehículos que causan que la mayoría de plazas de los parqueaderos se encuentren ocupadas y que los conductores inviertan una gran cantidad de tiempo circulando por estos estacionamientos en búsqueda de algún espacio disponible.[2]

Los conductores suelen dirigirse a ciertos sitios especícos donde conocen algunos parqueaderos en los cuales pueden encontrar estacionamiento, pero éstos no son todos los que existen, además, al dirigirse con su vehículo a sitios nuevos existe un desconocimiento de los parqueaderos cercanos a dicho lugar. [2]

En consecuencia, las personas se encuentran con una serie de problemas para hallar una plaza disponible en un parqueadero, entre los cuales se evidencia el congestionamiento en los parqueaderos más conocidos, el desconocimiento de los parqueaderos cercanos y la pérdida de tiempo buscando una plaza disponible.[1]

Estas dicultades causan que los usuarios pierdan su tiempo, tengan atrasos en sus trabajos y sufran de stress al pasar por estas situaciones.[2]

Es por ello, que se plantea el desarrollo de una aplicación que permita encontrar un parqueadero cercano con plazas disponibles, para optimizar el tiempo de los conductores.

Además, el usuario podrá visualizar la ruta y ser dirigido hacia el parqueadero, ayudando a las personas que no conocen la ciudad y no saben por dónde dirigirse.

Asimismo, el usuario no sólo podrá visualizar los parqueaderos cercanos a su ubicación, también podrá ingresar una ubicación alterna.

La aplicación basa su funcionamiento en los servicios ofrecidos por Google como son Google Maps, Place AutoComplete - Google Places, Api de Rutas, Turn-by-Turn Navigation.

Para el desarrollo de la aplicación móvil android se han establecido las siguientes fases: análisis, diseño, codificación y pruebas [3].

II. ESTADO DEL ARTE

A. Aplicación móvil

Las aplicaciones móviles son programas software que pueden ser descargadas y a las que se puede acceder directamente desde un teléfono o desde algún otro dispositivo móvil, como por ejemplo una tablet. [4].

B. Sistema Operativo para móviles: Android

Android es un sistema operativo basado en Linux para dispositivos móviles, tales comoteléfonos inteligentes o tablets. Fue desarrollado inicialmente por Android Inc., una firmacomprada por Google en 2005.Es el principal producto de la Open Handset Alliance, unconglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio.

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de lasbibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución. Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica (surface manager), un framework OpenCore, una base de datos relacional SQLite,una API gráfica OpenGL ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico SGL,SSL y una biblioteca estándar de C Bionic.

M. Chinchay, Universidad Nacional de Loja, Loja, Ecuador, E-mail: mjchinchayc@unl.edu.ec
H. Paz, Universidad Nacional de Loja, Loja, Ecuador, E-mail: hpaz@unl.edu.ec

En la actualidad existen más de 700.000 aplicaciones para Android y se estima que 1.000.000 teléfonos móviles se activan diariamente. [5]:

C. *Google Maps v2*

Google Maps para Android v2 API, permite incrustar mapas en una actividad de una aplicación como un fragmento con un simple fragmento de código XML. Estos mapas ofrecen características interesantes como mapas vía satélite, mapas híbridos, agregar marcadores, polílineas, círcunferencias, entre otros; además de un uso más fácil que su primera versión. [6]

D. *Place AutoComplete - Google Places API*

Google Places es un servicio de Google gracias al cual hay como dar de alta y gestionar la información de la ubicación física de un negocio, además de otra información relacionada.

A través de las búsquedas locales en Google se puede acceder a la ficha de la empresa y sus datos de contacto, además de ubicarlo en el mapa para facilitar la localización. [7]

Gracias a la gran información de Google Places API puede ofrecer un servicio como el autocompletado de sitios que consiste en un n servicio web que devuelve predicciones de sitios en respuesta a una solicitud HTTP.
La solicitud especifica una búsqueda textual y límites geográficos específicos. Este servicio se puede utilizar para proporcionar funciones de autocompletado para búsquedas geográficas basadas en texto, mostrando sitios como empresas, direcciones y lugares de interés a medida que el usuario escribe.[8]

E. *Api de Rutas*

El API de rutas de Google es un servicio que utiliza una solicitud HTTP para calcular rutas para llegar de una ubicación a otra.

Se puedes buscar rutas de varios métodos de transporte, como en transporte público, en coche, a pie o en bicicleta. Las rutas pueden especificar los orígenes, los destinos y los hitos como cadenas de texto (por ejemplo, "Chicago, IL" o "Darwin, NT, Australia") o como coordenadas de latitud/longitud.
[9]

F. *Turn-by-Turn Navigation.*

Google Maps ofrece el servicio de navegación guiada por voz giro a giro, ya sea conduciendo o caminando desde la ubicación actual hacia un destino. [10]

III. CASO DE ESTUDIO

La aplicación android para la búsqueda de plazas disponibles en un parqueadero tiene como funciones el visualizar los parqueaderos de la ciudad de Loja, buscar parqueaderos por su nombre, marcar la ubicación actual del usuario o una ubicación alterna que sea ingresada por él, buscar el parqueadero más cercano y trazar la ruta hacia su ubicación.(ver figura 1)

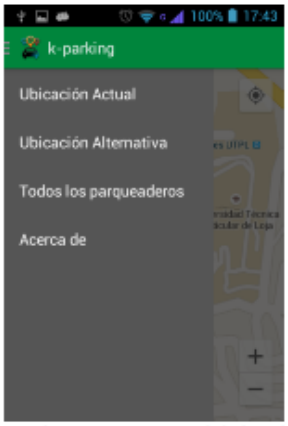


Fig.1.Menu de Navegación Lateral de la aplicación.

IV. RESULTADOS

A. *Análisis:*

Para la localización de los parqueaderos se utilizó la longitud y latitud correspondiente a la dirección de cada uno de ellos.

La obtención del parqueadero más cercano se realizó en base a un cálculo de distancias entre los parqueaderos y la ubicación real del usuario o una ubicación alternativa ofrecida por él.

B. *Diseño*

El diseño de las pantallas de la aplicación móvil android fueron realizadas con la herramienta Pencil Project [11], debido a su gran cantidad de elementos a la hora de realizar los prototipados de pantallas ya sea para un aplicación web o móvil Android o IOS, además permite la búsqueda de imágenes prediseñadas y vincular los elementos de la interfaz para simular el flujo de la aplicación. [12]

C. *Codificación*

La codificación se ha realizado con el Eclipse ADT Bundle el cual posee un completo entorno de desarrollo al estar compuesto por el IDE Eclipse y el sdk de Android.

C. *Codificación*

La codificación se ha realizado con el Eclipse ADT Bundle el cual posee un completo entorno de desarrollo al estar compuesto por el IDE Eclipse y el sdk de Android. La aplicación accederá a un web service que conectará la aplicación a una base de datos con la información de los parqueaderos de la ciudad de Loja. Se ha utilizado la librería sherlockactionbar para el menú.

1) **Visualización de los parqueaderos**

La visualización de los parqueaderos se hace a traves del servicio de mapas de Google en donde cada parqueadero se añade como un marcador de un color específico de acuerdo a la disponibilidad de plazas libres. (ver Figura 1).

Además se ha hecho uso de la clase InfoWindowAdapter para mostrar al usuario información acerca del parquedero como una imagen del establecimiento, razón social, dirección y número de plazas disponibles.

```
MarkerOptions marcador = new MarkerOptions();
for (Parqueadero p : parqueaderosAuxiliar) {
        marcador.position(new LatLng(p.
            getLatitud(), p.getLongitud()));
        marcador.title(p.getNombre());
        marcador.snippet(p.toString());
        if (p.getPlazaLibre() <= 5) {
        marcador.icon(BitmapDescriptorFactory.
            fromResource(R.drawable.
            icono_park_rojo));
        } else {
            if (p.getPlazaLibre() <= 10) {
            marcador.icon(
                BitmapDescriptorFactory.
                fromResource(R.drawable.
                icono_park_amarillo));
            } else {
                marcador.icon(
                    BitmapDescriptorFactory.
                    fromResource(R.drawable.
                    icono_park_verde));
            }
        }
        markerA = mapa.addMarker(marcador);
        distanciaAux = String.valueOf(
            calcularDistancia(p));
        infoWindowAdapterPersonalizado = new
            InfoWindowAdapter() {
        @Override
        public View getInfoWindow(final Marker
            marker) {
            LayoutInflater inflater = (
                LayoutInflater) contexto.
                getSystemService(Context.
                LAYOUT_INFLATER_SERVICE);
            View infoWindow = inflater.inflate(
                R.layout.
                info_window_parqueadero, null);
            imagenParqueadero = (ImageView)
                infoWindow.findViewById(R.id.
                imagePark);
        String descripcionParqueadero = marker.
            getSnippet().substring(0, marker.
            getSnippet().indexOf(":"));
        urlImagen = marker.getSnippet().
            substring(marker.getSnippet().
            indexOf(":") + 1, marker.getSnippet
            ().length());
        imageLoader.displayImage(urlImagen,
            imagenParqueadero, options, new
            SimpleImageLoadingListener() {
            @Override
            public void onLoadingComplete(
                String imageUri, View view,
                Bitmap loadedImage) {
            super.onLoadingComplete(imageUri,
                view, loadedImage);
            getInfoContents(marker);
            }
            });
            Toast.makeText(contexto, "Distancia
                hacia su localizaci n:"+
                distanciaAux, Toast.LENGTH_LONG)
                .show();
            TextView tvTitle = ((TextView)
                infoWindow.findViewById(R.id.
                nombrePark));
            tvTitle.setText(marker.getTitle());
            TextView tvSnippet = ((TextView)
                        infoWindow.
                findViewById(R.id.
                descripcionPark));
        tvSnippet.setText(
            descripcionParqueadero);
            return infoWindow;
         }
        @Override
        public View getInfoContents(Marker arg0
            ) {
            return null;
        }
        };
        mapa.setInfoWindowAdapter(
            infoWindowAdapterPersonalizado);
}
```

2) **Búsqueda de parqueaderos**

Para la búsqueda de parqueaderos de acuerdo a su nombre se utilizó el componente SearchView y algunos de sus métodos

En el método onQueryTextChange se controló que a medida que el usuario digite algún texto, las posibles opciones de selección se filtren de acuerdo al parecido del texto ingresado en comparación con el nombre de los parqueaderos.

```
@Override
public boolean onQueryTextChange(String newText
    ) {
        MatrixCursor cursor = new MatrixCursor(
            COLUMNS);
        newText.toLowerCase();
        for (Parqueadero p :
            parqueaderosAuxiliar) {
                if (p.getNombre().contains(
                    newText)) {
                        cursor.addRow(new
                            Object[] { p.getId
                            (), p.getNombre(),
                        p.getDireccion(), p.
                            getLatitud(), p.
                            getLongitud(),
                        p.getPlazaLibre(), p.
                            getPlazaOcupada(),
                        p.getImagenParqueadero
                            () });
                }
        }
        mSuggestionsAdapter = null;
```

```
        return false;
}
```

En el método onSuggestionClick se realiza una búsqueda en la base de datos interna de la aplicación del parqueadero seleccionado, para luego marcar una circunferencia alrededor de su ubicación en el mapa .

```
@Override
public boolean onSuggestionClick(int position)
    {
        Cursor c = (Cursor) mSuggestionsAdapter
            .getItem(position);
        String query = c.getString(c
                .getColumnIndex(
            SearchManager.SUGGEST_COLUMN_TEXT_1
            ));
        Toast.makeText(this, "Parqueadero_
            seleccionado:_" + query, Toast.
            LENGTH_LONG).show();
        searchView.setQuery(query, false);
        searchView.clearFocus();
        pds = new ParqueaderoDataSource(this);
        pds.open();
        Parqueadero pEncontrado = pds.
            getParqueadero(query);
        pds.close();
        localizacion = new LatLng(pEncontrado.
            getLatitud(),pEncontrado.
            getLongitud());
        ajustarMapa(17);
        marcarParqueaderoBuscado(pEncontrado.
            getLatitud(),pEncontrado.
            getLongitud());
        return true;
}

private void marcarParqueaderoBuscado(double
    latitud, double longitud){

   CircleOptions circulo = new CircleOptions().
       center(new LatLng(latitud,longitud)).
       radius(40);
    circuloParqueaderoEncontrado = mapa.
        addCircle(circulo);
    circuloParqueaderoEncontrado.setFillColor(
        Color.TRANSPARENT);
    circuloParqueaderoEncontrado.setStrokeColor
        (Color.RED);
    circuloParqueaderoEncontrado.setStrokeWidth
        (2f);
}
```

3) **Búsqueda de parqueadero más cercano**

Para la localización del parqueadero cercano se usó el método distanceTo() que calcula la distancia entre dos ubicaciones.

```
public double calcularDistancia(Parqueadero p){
        double distancia=0;
        Location localizacionActual = new
            Location("localizacionActual");
        localizacionActual.setLatitude(
            localizacion.latitude);
        localizacionActual.setLongitude(
            localizacion.longitude);
        Location localizacionParqueadero = new
            Location("localizacionParqueadero")
            ;
        localizacionParqueadero.setLatitude(p.
            getLatitud());
        localizacionParqueadero.setLongitude(p.
            getLongitud());
    distancia = localizacionActual.distanceTo(
        localizacionParqueadero);
        return distancia;
}
```

Se recorre el conjunto de parqueaderos, y se invoca sobre cada uno de ellos este método, para poder obtener el parqueadero con menor distancia.

4) **Trazado de Rutas y Navegación**
Se hará uso del servicio del api de rutas de google para trazar la ruta desde la ubicación dado por el gps o una ingresada por el usuario. Para la navegación por voz se utilizará Turn by Turn para que guie al usuario en su recorrido hacia un parqueadero.

D. Pruebas

Las pruebas realizadas a la aplicación comprobaron el funcionamiento de la aplicación, a continuación la ejecución de sus opciones:

- **Visualización de Parqueaderos** La aplicación permite visualizar todos los parqueaderos de la ciudad de Loja a traves de marcadores de diferentes colores de acuerdo a la disponibilidad de plazas disponible (verde para un número mayor a 10, amarillo para menor a 10 y mayor a 5, rojo para menor a 5) acompañado de un marcador referente a la ubicación actual del usuario. (ver figura 2)

Fig.2.Visualización de todos los parqueaderos de la ciudad de Loja.

- **Búsqueda de Parqueaderos** La aplicación permite la búsqueda de parqueaderos de acuerdo a su nombre, a través de la lupa del actionbar e ingresa el texto de búsqueda, la aplicación ofrece el servicio de autocompletado. (ver figura 3)

Fig.3.Búsqueda de parqueaderos de acuerdo al nombre.

Al encontrar el parqueadero la aplicación resalta el parqueadero encontrado con una circunferencia alrededor de su ubicación. (ver figura 4)

Fig.4.Parqueadero buscado.

- **Ubicación Alterna** El usuario puede ingresar el nombre de una calle con el fin marcar una ubicación diferente a la detectada por el gps. (ver figura 5)

Fig.5.Ubicación alterna.

- **Búsqueda de Parqueadero más cercano** El usuario puede encontrar el parqueadero más cercano de acuerdo a la ubicación señalada por el marcador, sea ésta la actual o una dada por el usuario. Al encontrar el parqueadero más cercano se presenta un mensaje de confirmación (ver figura 6)

Fig.6.Mensaje de confirmación.

Si el usuario decida aceptar, el parqueadero será marcado con una circunferencia a su alrededor.(ver figura 7)

Fig.7.Parqueadero más cercano.

V. CONCLUSIONES

- Los servicios de Google relacionados a google maps son de gran utilidad en el desarrollo de aplicaciones móviles puesto que al ser implementados de forma adecuada permiten optimizar el tiempo de los usuarios.

Fig.7.Parqueadero más cercano.

V. CONCLUSIONES

- Los servicios de Google relacionados a google maps son de gran utilidad en el desarrollo de aplicaciones móviles puesto que al ser implementados de forma adecuada permiten optimizar el tiempo de los usuarios.
- La implementación de sherlockactionbar permite que la aplicación pueda ser visualizada en diferentes versiones de Android.
- Para un adecuado desarrollo de aplicaciones android es necesario configurar de manera correcta el entorno y el emulador o dispositivo en el cual se pruebe la aplicación.

VI. TRABAJOS FUTUROS

A la presente aplicaión se le puede añadir algunas características que ayuden a optimizar su eficiencia. A continuación se describen los posibles temas a desarrollarse:

- ***Registro y modificación de parqueaderos:*** Los propietarios de los parqueaderos tendrían la posibilidad de registrar directamente los parqueaderos en la aplicación y modificar el número de plazas disponibles; lo que daría mayores posibilidades de tener una base de datos siempre actualizada.
- ***Reservación de plazas disponibles en un parqueadero:*** El usuario de la aplicación reservaría una plaza libre para su vehículo y así evitaría que las plazas del parqueadero se llenen antes de su llegada.
- ***Búsqueda de farmacias, supermercados, restaurantes, entre otros:*** Una aplicación que permita ubicar diferentes tipos de negocios dentro de una ciudad sería muy conveniente cuando no se conoce un sitio.

REFERENCIAS

[1] Centro de Computación de Universidad de Chile, *Gestión de Estacionamientos*, Universidad de Chile, 2014.[En linea] link:http:www.cec.uchile.cl.ci53g/clase28_gestion_estacionamiento.PDF

[2] Chinchay Marjorie, *Tabulación de la Encuesta realizada a usuarios de parqueaderos privados de Loja.*, Loja, 2014.

[3] *Ciclo de vida del software.* [En linea] link:http://img.redusers.com/imagenes/libros/lpcu097/capitulogratis.pdf.

[4] Comisión Federal de Comercio de los Estados Unidos, *Cómo funcionan las aplicaciones móviles: Preguntas y respuestas*, Estados Unidos 2009.[En linea] link:http://www.alertaenlinea.gov/articles/pdf-s0004_0.pdf

[5] Oropeza Rodríguez José Luis, *Sistema Operativo Android*, 2011.[En linea] link:http://148.204.64.201/paginas%20anexas/android/EL%20SISTEMA%20OPERATIV

[6] Google Developers, *Google Maps Android API v2*, [En linea] link:http://developer.android.com/google/play-services/maps.html

[7] EnGoogle.Net, *Que es Google Places*, [En linea] link:http://www.google-places.es/que-es-google-places

[8] Google Developers, *Autocompletado de sitios*, [En linea] link:https://developers.google.com/places/documentation/autocomplete?hl=es

[9] Google Developers, *El API de rutas de Google*, [En linea] link:https://developers.google.com/maps/documentation/directions/?hl=es-es

[10] Google Developers, *Google Maps for Android*, [En linea] link:http://www.google.com/mobile/maps/

[11] Evolus, *Pencil Project*,2012. [En linea] link:http://pencil.evolus.vn

[12] Evolus, *Pencil Project-Características*, 2012. [En linea] link:http://pencil.evolus.vn/Features.html

Marjorie Chinchay

Egresada de la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja, Conocedora de Desarrollo de Software, Provincia de Loja, Ciudad Loja, Ecuador, 2014.

Henry Paz

Ingeniero en Sistemas de la Universidad Nacional de Loja - Ecuador en el 2010, Maestro en Ciencias Computacionales de la Universidad Autónoma del Estado de Hidalgo – México en la especialidad de Computación Inteligente en el 2012, su interés por la investigación actual es el desarrollo de aplicaciones móviles para mejorar la calidad de vida.

2. Anexo 2. Poster expuesto en el Segundo Congreso de Tecnologías de la Información organizado en la Universidad de Cuenca.

UNIVERSIDAD NACIONAL DE LOJA

BÚSQUEDA DE PLAZAS DISPONIBLES EN UN PARQUEADERO MEDIANTE UNA APLICACIÓN MÓVIL.

MARJORIE J. CHINCHAY C, HENRY P. PAZ A.

INTRODUCCIÓN

Actualmente, existe un gran congestionamiento a la hora de alquilar un espacio disponible debido a la gran cantidad de vehículos que utilizan este servicio, sobretodo en los parqueaderos que suelen ser más conocidos provocando una gran aglomeración.
En los lugares dónde se encuentran laborando las principales instituciones públicas y privadas de una ciudad o provincia existe una gran ausencia de personas y por ende de vehículos que causan que la mayoría de plazas de los parqueaderos se encuentren ocupadas y que los conductores inviertan una gran cantidad de tiempo circulando por estos estacionamientos en búsqueda de algún espacio disponible.
Los conductores suelen dirigirse a ciertos sitios específicos donde conocen algunos parqueaderos en los cuales pueden encontrar estacionamiento, pero éstos no son todos los que existen, además, al dirigirse con su vehículo a sitios nuevos existe un desconocimiento de los parqueaderos cercanos a dicho lugar.
En consecuencia, las personas se encuentran con una serie de problemas para hallar una plaza disponible en un parqueadero, entre los cuales se evidencia el congestionamiento en los parqueaderos más conocidos, el desconocimiento de los parqueaderos cercanos y la pérdida de tiempo buscando una plaza disponible.

MATERIALES Y MÉTODOS

Metodología
Se planteó la utilización de una metodología ágil que permita la elaboración de la aplicación móvil de una forma rápida y con un nivel aceptable de documentación. Es por ello, que se ha escogido la metodología Mobile-D.
La metodología Mobile-d tiene como objetivo conseguir ciclos de desarrollo muy rápidos en equipos muy pequeños. Fue creado en un proyecto finlandés en 2005, pero sigue estando vigente. El ciclo del proyecto se divide en cinco fases: exploración, inicialización, productización, estabilización y prueba del sistema.
Exploración: En esta etapa se realiza el Establecimiento de los stakeholders, definición del alcance y el establecimiento del proyecto.
Inicialización: Se realiza la configuración del ambiente de desarrollo (hardware y software), el establecimiento de las iteraciones y diagramas (modelo del dominio, modelo entidad relación, storyboard o esquema de navegación) y los storycards con cada uno de los prototipos de la aplicación.
Producción: Se realiza la codificación de cada una de las partes de la aplicación.
Estabilización: Se integra cada parte codificada de la aplicación.
Pruebas del Sistema: Se realizan las pruebas unitarias y funcionales.

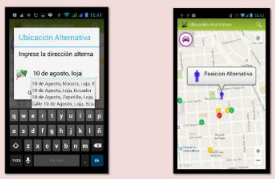
Métodos
Método Deductivo: Se aplicó este método para obtener la información necesaria sobre el alquiler de los parqueaderos privados, para detectar los problemas existentes al momento de encontrar un espacio disponible y determinar una posible solución para esto.
Método Inductivo: A través de este método se plantean las principales dificultades al alquilar un espacio disponible para determinar si la solución planteada permite resolverlos.
Técnicas
Encuesta: Con esta técnica se obtuvo información relevante y necesaria referente al tema de la investigación, la cual ayudara a sustentarlo y justificarlo.
Técnica de la investigación Bibliográfica: Con esta técnica se sustentara la base teórica de la investigación, mediante consultas a: fuentes bibliográficas confiables, libros, revistas indexadas, artículos científicos, base de datos científicas entre otras.

RESULTADOS

Como principal resultado del proyecto se ha desarrollado una aplicación móvil Android para la búsqueda de plazas disponibles en un parqueadero; sus principales funcionalidades se detallan a continuación:

Consulta de los datos del parqueadero

Visualización de los parqueaderos desde la ubicación del dispositivo

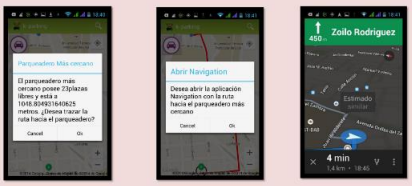


Visualización de los parqueaderos desde una ubicación alternativa

Visualización de todos los parqueaderos de la ciudad de Loja

Información del parqueadero

Búsqueda de un parqueadero por su razón social



Búsqueda de un parqueadero cercano

CONCLUSIONES:

- Los servicios de Google relacionados a google maps son de gran utilidad en el desarrollo de aplicaciones móviles puesto que al ser implementados de forma adecuada permiten optimizar el tiempo de los usuarios.
- La implementaciónn de sherlockactionbar permite que la aplicación pueda ser visualizada en diferentes versiones de Android.
- Para un adecuado desarrollo de aplicaciones android es necesario configurar de manera correcta el entorno y el emulador o dispositivo en el cual se pruebe la aplicación

3. Anexo 3. Licencia Creative Commons

4. Anexo 4. Certificado de Resumen

A quien corresponda:

Por el presente, Yo, Dr. Geovany Castillo Herrera, catedrático del área de idioma Inglés, con experiencia de aproximadamente 29 años en la docencia de nivel superior y 20 años de práctica en traducciones oficiales, C E R T I F I C O haber traducido al idioma inglés el ***Abstract*** que me presentó la Srta. Marjorie Chinchay.

Autorizo a la portadora hacer uso del presente certificado en la forma que mejor convenga a sus intereses.

Loja, 28 de mayo de 2015

Dr. Geovany Castillo Herrera

EFL CONSULTANT & FREELANCE TRANSLATOR

CI: 1102429592

5. Anexo 5. Certificado de que la Aplicación fue elaborada dentro de KRADAC Cía. Ltda, por la Tesista.

Yo, **Ing. Bruno Valarezo Correa,** con cédula de identidad N° 1104185630, **GERENTE GENERAL DE KRADAC,** con RUC 1191734609001, a petición verbal de la parte interesada.

CERTIFICO:

Que, la **Srta. MARJORIE JULIANA CHINCHAY CUENCA,** Identificado con cédula de identidad N° 0705470656, ha laborado en nuestra empresa como **DESARROLLADOR DE APLICACIONES ANDROID,** realizando una aplicación que permita la búsqueda de plazas disponibles en un parqueadero de la ciudad de Loja, durante el periodo comprendido desde abril de 2014 hasta octubre de 2014, así mismo que tengo conocimiento que la aplicación será utilizada para obtener el título de fin de carrera.

Atentamente:

Ing. Bruno Valarezo Correa
GERENTE GENERAL DE KRADAC.

KRADAC Cía. Ltda.
RUC: 1191734609001
Oficina: UTPL / 1er. Piso UGTI