Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Ciencias y Sistemas

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LOCALIZACIÓN DE PARQUEOS EN LA CIUDAD DE GUATEMALA

Gersson Marco Vinicio Herrarte Barrios
Asesorado por el Ing. José Manuel Ruíz Juárez

Guatemala, octubre de 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LOCALIZACIÓN DE PARQUEOS EN LA CIUDAD DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

GERSSON MARCO VINICIO HERRARTE BARRIOS
ASESORADO POR EL ING. JOSÉ MANUEL RUÍZ JUARÉZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, OCTUBRE DE 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Oscar Humberto Galicia Nuñez
VOCAL V	Br. Carlos Enrique Gómez Donis
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. Edgar Estuardo Santos Sutuj
EXAMINADOR	Ing. Oscar Alejandro Paz Campos
EXAMINADOR	Ing. Everest Darwin Medinilla Rodríguez
SECRETARIO	Ing. Pablo Christian de León Rodríguez a.i.

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LOCALIZACIÓN DE PARQUEOS EN LA CIUDAD DE GUATEMALA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 14 de febrero de 2017.

Gersson Marco Vinicio Herrarte Barrios

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 29 de agosto de 2018

Señor
Ing. Carlos Azurdia
Facultad de Ingeniería
Universidad de San Carlos de Guatemala
Guatemala, Ciudad

Respetable Ing. Azurdia

El motivo de la presente es para informarle que como asesor del estudiante Gersson Marco Vinicio Herrarte Barrios he procedido a revisar el trabajo de graduación: "DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LOCALIZACIÓN DE PARQUEOS EN LA CIUDAD DE GUATEMALA" y que de acuerdo a mi criterio el mismo se encuentra concluido.

He tenido comunicación periódica con el estudiante y luego de haber revisar el trabajo, considero que cumple con los requisitos de calidad y profesionalismo que debe caracterizar a un futuro profesional de la informática.

Sin otro particular me suscribo a usted,

Atentamente,

José Manuel Ruiz Juárez
Ing. en Ciencias y Sistemas
Colegiado No. 7945

Ing. José Manuel Ruíz Juaréz

Colegiado No. 7945

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 12 de septiembre de 2018

Ingeniero
Marlon Antonio Pérez Türk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **GERSSON MARCO VINICIO HERRARTE BARRIOS** con carné **201222689 y CUI 2312 40457 0101** titulado **"DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LOCALIZACIÓN DE PARQUEOS EN LA CIUDAD DE GUATEMALA"** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo aprobado.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
INGENIERIA EN CIENCIAS Y SISTEMAS
COORDINACION TESIS

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS
TEL: 24188000 Ext. 1534

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación, ***"DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LOCALIZACIÓN DE PARQUEOS EN LA CIUDAD DE GUATEMALA"*** *realizado por el estudiante, GERSSON MARCO VINICIO HERRARTE BARRIOS, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Marlon Antonio Pérez Türk
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 25 de octubre de 2018

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

DTG. 428.2018

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **"DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LOCALIZACIÓN DE PARQUEOS EN LA CIUDAD DE GUATEMALA",** presentado por el estudiante universitario: **Gersson Marco Vinicio Herrarte Barrios,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Pedro Antonio Aguilar Polanco
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala octubre de 2018.

/echm

ACTO QUE DEDICO A:

Dios	Por darme la vida, la salud, guiarme en cada etapa de mi vida y permitirme culminar esta carrera.
Mis padres	Nery de Jesús Herrarte y Marta Zoé Barrios, por educarme y enseñarme valores con su ejemplo y por su amor incondicional.
Mis hermanos	Nery Abner Herrarte Barrios, Nataly Saraí Herrarte Barrios, Ethel Noemí Herrarte Barrios, por ser personas importantes en mi vida.

.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser la casa de estudios que me permitió desarrollarme como profesional.
Facultad de Ingeniería	Por haberme calificado para egresar como profesional de esta facultad.
Dios	Por haberme dado la vida, salud y proveerme los medios necesarios para alcanzar esta meta.
Mis padres	Nery de Jesús Herrarte y Marta Zoé Barrios, por su amor e incondicional apoyo a lo largo de mi vida.
Mis hermanos	Nery Abner Herrarte Barrios, Nataly Saraí Herrarte Barrios, Ethel Noemí Herrarte Barrios, por su amistad y apoyo moral**.**
Mi asesor de proyecto	Ing. José Manuel Ruiz, por aportar sus conocimientos para la corrección de este trabajo.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
GB	Gigabyte
MB	Megabytes
MVC	Modelo vista controlador

GLOSARIO

AMOLED — Por sus siglas en inglés, a*ctive-matrix organic light-emitting diode*; matriz activa de diodos orgánicos emisores de luz, una tecnología de fabricación de pantallas basada en OLED, utilizada en dispositivos móviles.

Android — Sistema operativo desarrollado con base en el núcleo de Linux. Fue diseñado especialmente para dispositivos móviles.

Angular JS — Es un *framework* de Javascript utilizado para el desarrollo de páginas web.

API — Por sus siglas en inglés, *application programming interface*. Son un conjunto de procedimientos y funciones utilizados para implementar programas sin escribir todo el código.

Aplicación móvil — Es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos; permite al usuario efectuar una tarea concreta de cualquier tipo: profesional, de ocio, educativas, de acceso a servicios, etc.

Arquitectura MVC	Modelo Vista Controlador; es un patrón de arquitectura para el diseño de software.
Arquitectura de *Software*	Conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.
Base de datos	Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su uso posterior.
Bootstrap	*E*s un *framework* web o conjunto de herramientas de código abierto utilizado para el diseño de sitios y aplicaciones web.
Framework	Es un entorno de trabajo. Contiene una estructura y herramientas de software utilizadas para el desarrollo de software.
Formato JSON	Acrónimo de *JavaScript object notation*, formato de texto ligero para el intercambio de datos.
GPRS	Servicio general de paquetes vía radio para tecnologías GSM.
GPS	Sistema de posicionamiento global. Permite a los dispositivos móviles obtener la ubicación exacta con solo tener activada la opción de GPS.

Hardware	Parte física de un sistema informático.
Hosting	Servicio que una empresa provee a los usuarios de internet, para almacenar cualquier contenido accesible vía internet.
IONIC	Es un *framework* gratuito para el desarrollo de aplicaciones móviles.
IOS	Sistema operativo desarrollado por la empresa Apple; diseñado para trabajar únicamente con dispositivos de Apple.
IPS	Un sistema de prevención de intrusos. Software que ejerce el control de acceso en una red informática para proteger a los sistemas informáticos de ataques.
Javascript	Es un lenguaje de programación orientado a la programación basada en objetos; utilizado mayormente por los navegadores web.
LCD	Por sus siglas en inglés, *liquid cristal display*, sistema que utilizan determinadas pantallas electrónicas para mostrar información visual.
Login	En español, ingresar; es el proceso mediante el cual se controla el acceso individual a un sistema.

Memoria RAM	Del inglés, random access memory. La memoria RAM es donde una computadora almacena información que se está utilizando en el momento presente.
MySQL	MySQL es un gestor de bases de datos multiusuario y multihilo el cual permite gestionar bases de datos.
OLED	Diodo orgánico de emisión de luz, en inglés o*rganic light-emitting diode;* es un tipo de diodo que se basa en una capa electroluminiscente.
PhoneGap	*Framework* para el desarrollo de aplicaciones móviles.
Campo	En informática, espacio de almacenamiento para un dato en particular.
PHP	Acrónimo recursivo en inglés de *hypertext preprocessor*. Es un lenguaje de programación adecuado para el desarrollo web y uso generalmente del lado de la aplicación.
Protocolo HTTP	Protocolo de transferencia de hipertexto, es el protocolo de comunicación que permite las transferencias de información en internet.
Redes GSM	Sistema global para las comunicaciones móviles.

Roaming	Itinerancia de datos.
Servidor Apache	Servidor web HTTP de código abierto, para plataformas Unix, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.
Servidor FTP	Es un programa especial que se ejecuta en un servidor conectado normalmente en internet o a otro tipo de redes.
Servicio web	Tecnología que utiliza un conjunto de protocolos para intercambiar información entre aplicaciones con distinta plataforma de software utilizando tecnología web.
Sistema operativo	Es el software principal o conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación de software.
Software	Parte lógica de un sistema informático-
WAP	Protocolo de aplicaciones inalámbricas.

RESUMEN

La tecnología ha evolucionado exponencialmente en la actualidad, así como su uso; ya que cada día facilita más la comunicación, el desarrollo, la eficacia en la educación, incrementa la producción y el desarrollo de la economía y la localización de personas, sitios y/o servicios. En la comunicación, responde a la necesidad de afiliación por naturaleza del hombre a ser social; para ello, es común utilizar distintos medios como el correo electrónico, las redes sociales o los dispositivos móviles, entre otros. Uno de los medios más comunes en la sociedad de Guatemala, lo representan los dispositivos móviles; ya que se encuentran al alcance de muchas personas en la sociedad de Guatemala.

Se implementará una aplicación móvil, que buscará parqueos dentro de la ciudad capital será llamada PFINDER. La letra P, representa la palabra parqueos y *find*, la palabra buscar en idioma inglés.

La aplicación móvil PFINDER es una solución que permite a las personas ubicar un parqueo de una manera fácil y eficiente. La aplicación utilizará tecnologías que proporcionan los dispositivos móviles. Se utilizará el GPS para obtener la ubicación actual de la persona. La aplicación implementará la API de Google Maps para mostrar las ubicaciones geográficas de los parqueos.

A continuación, se explican las partes importantes de un dispositivo móvil para su correcto funcionamiento; así como, los tipos de aplicaciones en el mercado y las tecnologías en las cuales están implementadas. Se investigaron los antecedentes de este tipo de aplicaciones en Guatemala.

Se plantea el análisis y diseño de la aplicación; también, un módulo de administración que se utilizará para la gestión de los parqueos. Además, se realiza una breve explicación del uso de la solución implementada.

OBJETIVOS

General

Diseñar e implementar una aplicación móvil que permita a las personas elegir un parqueo de acuerdo a las características de tarifa, horario y ubicación y que se encuentre disponible para *smartphones* con sistema operativo *Android* e *IOS*.

Específicos

1. Documentar y explicar las herramientas que se utilizaron para el desarrollo de la aplicación móvil.

2. Explicar los componentes más importantes que integran un dispositivo móvil para su correcto funcionamiento.

3. Presentar una solución de software que facilite la búsqueda de parqueo en la ciudad de Guatemala utilizando el GPS con dispositivos móviles.

INTRODUCCIÓN

En la actualidad, el uso de dispositivos móviles se ha convertido en algo muy común y útil. La mayor parte de la población tiene un dispositivo móvil el cual tiene incorporado un sistema operativo que le permite ejecutar un sinfín de aplicaciones que le sirven para resolver problemas.

Los dispositivos móviles son utilizados por las personas en sus actividades diarias: ocio, trabajo, salud, entretenimiento y comunicación. Se producen de forma masiva y su costo ha disminuido.

Según el artículo del diario de *La Hora, ¿Dónde parquearse*?, esta es una pregunta que atormenta a los guatemaltecos que transitan en la ciudad de Guatemala. Encontrar un parqueo cercano, de fácil acceso y con una tarifa adecuada es realmente difícil. Por tal motivo, se plantea la implementación de una aplicación móvil que permita a las personas localizar un parqueo de forma sencilla. En este trabajo de investigación se describe la solución de software y su utilización.

1. DISPOSITIVOS MÓVILES Y APLICACIONES

1.1. Dispositivo móvil

Los dispositivos móviles son aparatos electrónicos con distintas capacidades de procesamiento. Los dispositivos tienen acceso a redes electrónicas y se caracterizan por tener una capacidad de almacenamiento, en este caso, memoria interna. Se utilizan para llevar a cabo diversas funcionalidades; llamar, enviar mensajes de texto, correos electrónicos y utilización de varias aplicaciones. En la actualidad, se trasportan en el bolsillo debido a su pequeño tamaño el cual ha disminuido en los últimos 10 años.

Los dispositivos móviles tienen muchas características diferentes a las computadoras:

- No necesariamente extensible y actualizable
- Funcionalidad limitada
- En pocos años el usuario deberá cambiarlo
- Menos complicado en su manejo
- Fácil de aprender su operación
- No se requieren usuarios expertos

1.2. Partes de los dispositivos móviles

- Placa base: es una de las partes principales del dispositivo, contiene circuitos integrados y componentes electrónicos.

- Antena wifi: permite la comunicación inalámbrica tanto para envió como recepción al dispositivo.

- Pantalla: es un componente del dispositivo que ha evolucionado; en la actualidad se utilizan pantallas táctiles, estas le permiten al usuario interactuar directamente con el dispositivo sin utilizar un botón o un teclado como se realizaba con los dispositivos móviles más antiguos.

Las pantallas móviles transmiten información al dispositivo y muestran los resultados al usuario de manera instantánea. Existen distintos tipos de pantallas: LCD, Amoled, Retina, OLED e IPS.

- Antena NFC: este tipo de antena la utilizan los dispositivos para enviar y recibir señales a una distancia corta.

- Batería: almacenan energía para uso del dispositivo. Las baterías están formadas por litio y son recargables.

Los dispositivos se han diseñado para funcionar a nivel mundial a través de redes GSM, ya que el para que el usuario pueda conectarse a través de el y realizar distintas operaciones: navegar por internet, transmitir datos, enviar correos, entre otras.

1.3. Características de los dispositivos móviles

Un dispositivo móvil puede ser un *smartphone (*teléfono inteligente*),* un ipod o una tableta; todos estos comparten un conjunto de características a nivel interno y externo:

- Son aparatos electrónicos de tamaño pequeño y se pueden transportar fácilmente.

- Tienen una memoria limitada.

- Tienen una capacidad de procesamiento.

- Permiten la interacción con las personas a través de una pantalla táctil o un teclado.

- Permiten la conexión a redes inalámbricas.

Los *smartphones* incorporan un conjunto de funciones que hacen que su uso sea indispensable para muchas personas:

- Realización y recepción de llamadas de voz.
- Envió y recepción de mensajes cortos SMS y mensajes multimedia MMS.
- Roaming.
- Acceso a internet utilizando WAP.
- Acceso a internet utilizando GPRS.
- Acceso a internet utilizando HSPD.
- Acceso a internet utilizando LTE.
- Aplicaciones de software básicas como reloj, alarma, calendario, calculadora, juegos, entre otras.
- Conexiones en red con tecnologías como infrarrojo, bluetooth, wi-fi.
- sistema de posicionamiento global GPS.
- Sistemas de entretenimiento como reproducción de audio y video.
- Cámaras fotográficas y video frontales y posteriores.

- Visualización de televisión.
- Personalización de contenidos.
- Proyección de imágenes.
- Visualización de imágenes y videos 3D

Los dispositivos móviles han evolucionado hasta convertirse en *smartphones*; los cuales tienen un sistema operativo con funciones para la administración de varias aplicaciones.

1.4. Tipos de aplicaciones móviles

Existen diferentes tipos de aplicaciones móviles; se pueden clasificar y agrupar, de acuerdo a sus características.

Las aplicaciones se pueden clasificar de acuerdo al mercado para las que han sido desarrolladas: el lenguaje de programación utilizado para su implementación, si han sido desarrolladas de forma nativa, multiplataforma o de híbrida, si son aplicaciones empresariales o para el uso común de los usuarios.

1.4.1. Aplicaciones nativas

Las aplicaciones nativas son aquellas que se desarrollan para un sistema operativo en específico. Tienen un mejor funcionamiento y rendimiento en el sistema operativo en el cual fueron desarrolladas y permiten explotar al máximo las funcionalidades de los dispositivos.

Sin embargo, las aplicaciones nativas presentan desventajas: funcionan únicamente en el sistema operativo para el cual fue diseñado el código fuente

para este tipo de aplicaciones es exclusivo y personalizado no es posible reutilizar código para otra plataforma o sistema operativo.

1.4.2. Aplicaciones móviles web

Las aplicaciones móviles web están implementadas en los lenguajes Java Script, Html y CSS. Para el uso de estas aplicaciones es necesario contar con un navegador web; se caracterizan por que su programación no es muy compleja y no necesitan muchas herramientas para implementarse.

Las aplicaciones móviles también tienen desventajas: no se aprovecha al máximo el hardware y los recursos del dispositivo móvil. Para promocionar estas aplicaciones es necesario invertir en publicidad ya que no es posible publicarlas en la tienda online.

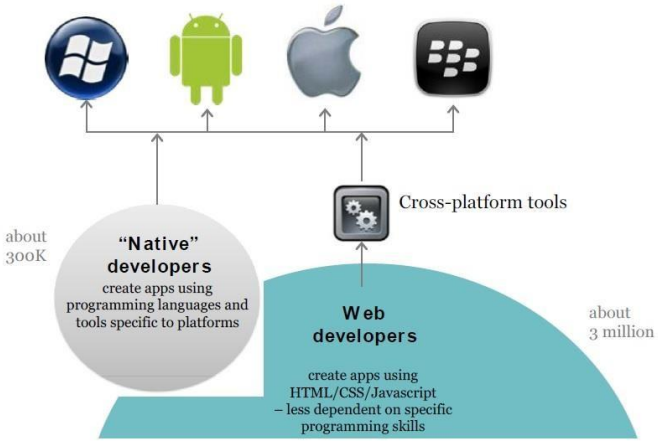
1.4.3. Aplicaciones híbridas

Las aplicaciones híbridas realizan una mezcla de características de las aplicaciones nativas y de las aplicaciones web. Para implementar este tipo de aplicaciones es necesario utilizar un *framework,* se crean utilizando lenguajes de desarrollo web HTML, CSS y JavaScript. Estas aplicaciones son multiplataforma; es decir, es posible generar un compilado sobre distintos sistemas operativos. Este tipo de aplicaciones permite el acceso al hardware del dispositivo por lo que es posible aprovecharlo. El costo de desarrollo es menor que el de una aplicación nativa y es posible distribuirlo en las tiendas online. Una de sus mayores ventajas es la reutilización de código.

Con este tipo de aplicación algunas veces no se aprovecha al máximo toda la capacidad de los dispositivos. Sus funciones pueden ser limitadas ya

que no se tiene acceso a todos los recursos del *smatphone*. Su rendimiento es menor al de una aplicación nativa.

Figura 1. **Aplicaciones hibridas**



Fuente: *Mobile Megatrends 2012*. http://www.visionmobile.com/blog/2012/05/report-mobile-megatrends-2012/. Consulta: 14 de febrero de 2017.

1.5. Uso de la tecnología

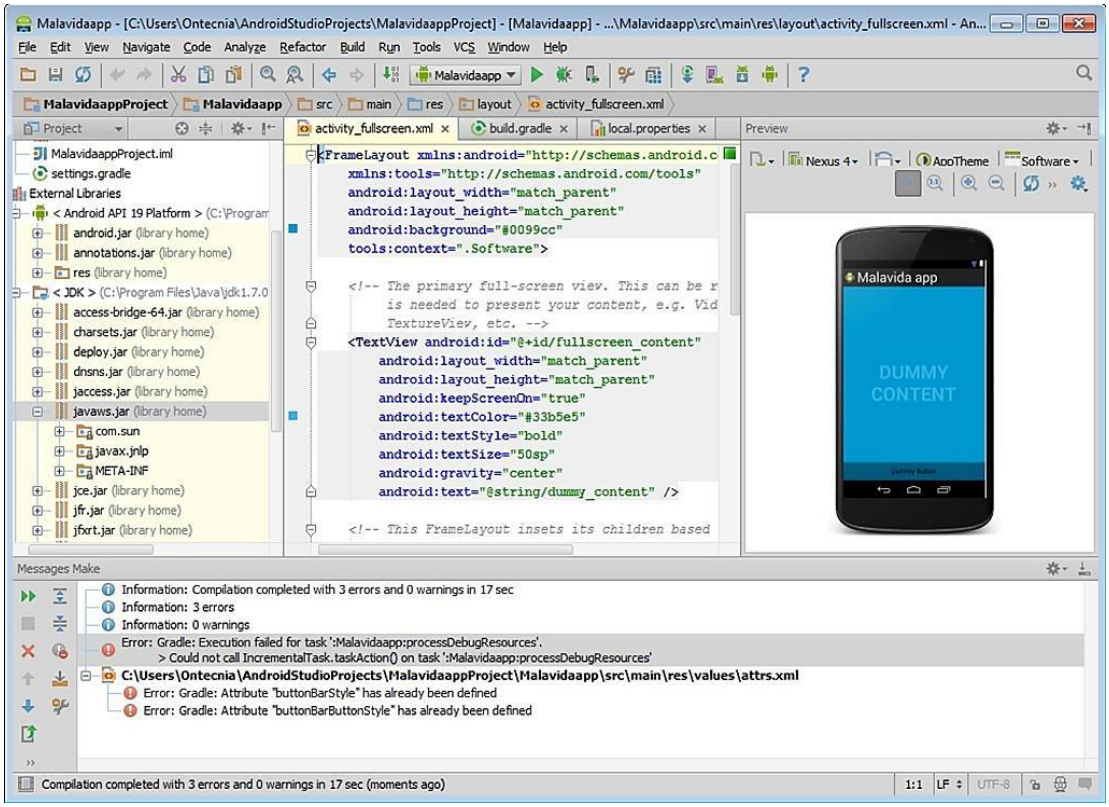
Como se ha mencionado, las aplicaciones permiten resolver problemas de forma directa. Para desarrollar este tipo de aplicación se analizaron las distintas tecnologías para el desarrollo de móviles: *Android Studio*, *Ionic* y *Phonegap*.

1.5.1. *Android Studio*

Es un entorno de desarrollo para aplicaciones móviles en *Android***.** Tiene una serie de ventajas como un óptimo rendimiento. Posee un conjunto de bibliotecas y herramientas para desarrollar aplicaciones muy completas de inicio a fin. El inconveniente de esta herramienta es que las aplicaciones funcionan

únicamente sobre la plataforma de *Android*; si se quisiera utilizar la aplicación en un móvil iPhone, no sería posible ya que este utiliza otro sistema operativo. Para el desarrollo de aplicaciones *Android* Studio se requiere de una computadora muy potente con un procesador de mínimo 2 núcleos físicos y suficiente memoria RAM. Es necesario, tener una buena capacidad de almacenamiento en este caso mínimo 4 GB de memoria.

Figura 2. ***Android Studio***



Fuente: Android Studio. http://imag.malavida.com/mvimgbig/download-fs/android-studio-12654-1.jpg. Consulta: 15 de enero de 2018.

1.5.2. *Framework Phonegap*

Es un *framework* para el desarrollo de aplicación móviles desarrollado por Nitobi. Nitobi es una empresa de software estadounidense que desarrolla aplicaciones para dispositivos móviles. *Phonegap* se caracteriza por proveer muchas facilidades y utilizar herramientas genéricas como Java Script y CSS3. Permite ejecutar aplicaciones sobre plataformas como *Android* e IOS. La curva de aprendizaje puede ser compleja comparada a otras herramientas como *Android Studio*. *Phonegap* es un software de código abierto es decir tiene licencia gratuita.

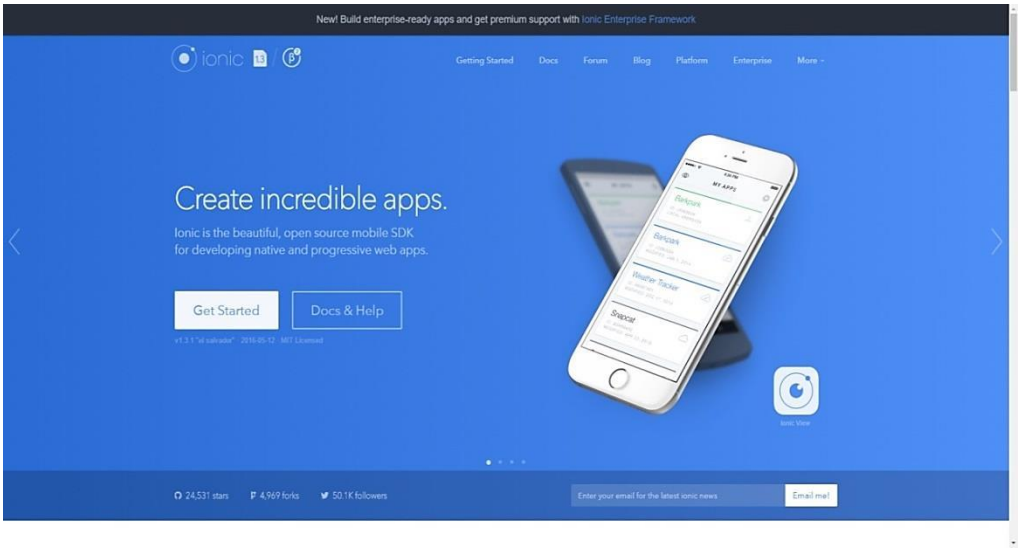
Figura 3. ***Phonegap***

Fuente: *Phonepag*. http://renaun.com/html5/introphonegap/assets/debug.phonegap.com.png
Consulta: 16 de enero de 2018.

1.5.3. *Framework IONIC*

Este *framework* es muy parecido a *Phonegap* para el desarrollo de aplicaciones móviles, ya que funciona en distintas plataformas. Está desarrollado con Angular Js y Apache Cordova. Se diferencia de *Phonegap* por utilizar Angular Js para el mejor aprovechamiento y rendimiento en la aceleración de hardware nativo, ya que *Phonegphap* utiliza Jquery.

Figura 4. ***IONIC***



Fuente: *IONIC*. https://i.imgur.com/mMcRKW5.png. Consulta: 17 de enero de 2018.

Las herramientas mencionadas son las que cuentan con mayor auge para el desarrollo de aplicaciones móviles. Se decidió utilizar *Ionic* debido a que es muy robusto y posee la ventaja de desarrollar aplicaciones híbridas; es decir, en un sistema operativo *Android e IOS*. *Ionic* se caracteriza por el bajo consumo de recursos, lo cual es de mucha utilidad. Otra ventaja de *Ionic* es que es una herramienta de código abierto; es decir, los desarrolladores pueden

personalizar sus interfaces de manera sencilla. Las aplicaciones desarrolladas en *Ionic* son muy limpias, de fácil mantenimiento y escalables.

Ionic utiliza componentes CSS, JSS y HTML lo que permite reutilizar código. Es posible desarrollar interfaces muy amigables e intuitivas en un periodo de tiempo relativamente corto, gracias a estos componentes.

1.6. Primeros pasos en *IONIC*

Para empezar a desarrollar una aplicación móvil en *Ionic framework,* es indispensable contar con una computadora. La computadora debe tener cualquiera de los siguientes sistemas operativos: Linux, Windows o Mac. En cuanto a capacidad de la computadora, es recomendable que tenga mínimo dos núcleos de procesamiento y 2 gigabytes de memoria RAM.

Es necesario tener un navegador web instalado en la computadora, preferiblemente el navegador web Google Chrome en su última versión ya que su depurador web es óptimo.

Es indispensable contar con un editor de texto en la computadora. Los editores recomendables son: Notepad, Atom, Visual Studio Code o Sublime Text; preferiblemente, en la versión más actualizada.

Para proceder a instalar Ionic, es necesario instalar manualmente algunas dependencias en la computadora las cuales se describen a continuación:

- Node Js

Es un marco de trabajo de ejecución de Java script de código abierto diseñado para la capa de servidor. Node Js implementa un administrador de paquetes llamado NPM el cual permite instalar paquetes según sea necesario y un intérprete de Java Script llamado V8, desarrollado por Google. El proceso más sencillo para instalar Node Js es descargar el instalador disponible en la página oficial, el cual se encuentra disponible para los sistemas operativos Windows, Mac Os y Linux.

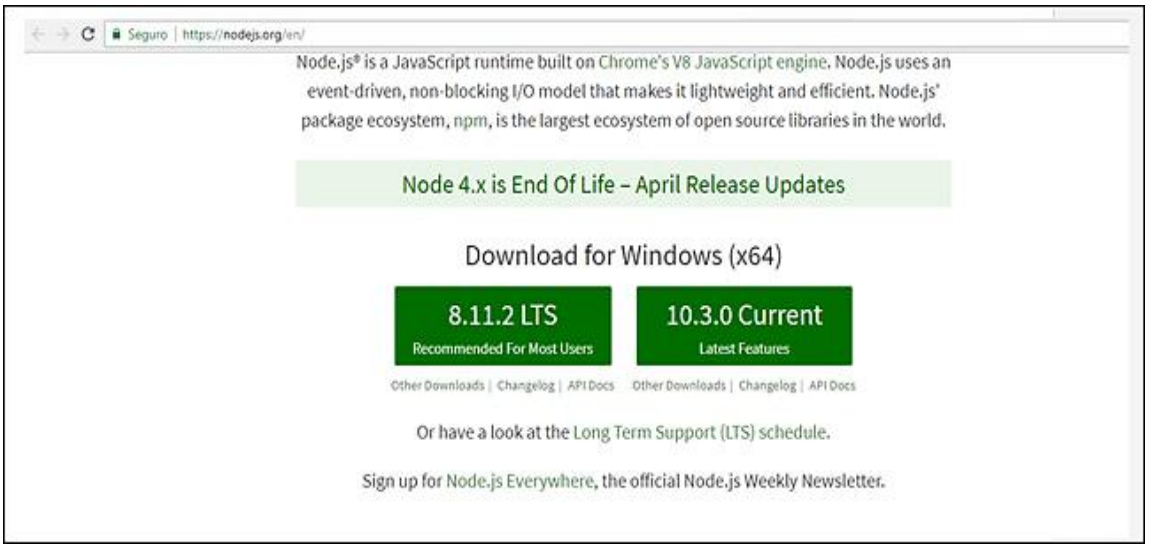
La página oficial ofrece dos versiones LTS y Current:

- LTS: es la versión recomendada por mayoría de usuarios. Esta versión tiene soporte a largo plazo y es más estable. Para instalarla, es necesario seguir los pasos del instalador.

- Current: es la última versión de Node Js, contiene todas las funcionalidades, aunque algunas no se consideren las más estables.

Para desarrollar en el *framework* Ionic se recomienda utilizar la versión LTS de Node Js.

Figura 5. ***Node Js***



Fuente: Node Js. https://nodejs.org/en/. Consulta: 23 de Junio de 2018.

1.6.1. Cordova y *IONIC*

Después de haber instalado correctamente Node Js, es necesario instalar Apache Cordova. Este es un entorno de desarrollo para aplicaciones móviles, con el cual es posible desarrollar las mismas utilizando HTML5 (*Hypertext markup language*) o CSS3 (*Cascading style sheets*). Gracias a estas herramientas, es posible personalizar de una manera más estética las aplicaciones móviles. Las aplicaciones resultantes con Cordova son híbridas.

Para instalar Cordova e *Ionic*, es necesario acceder a la consola de comando del sistema operativo donde está instalando. Si es en el sistema operativo Linux o Mac, es necesario escribir el siguiente comando:

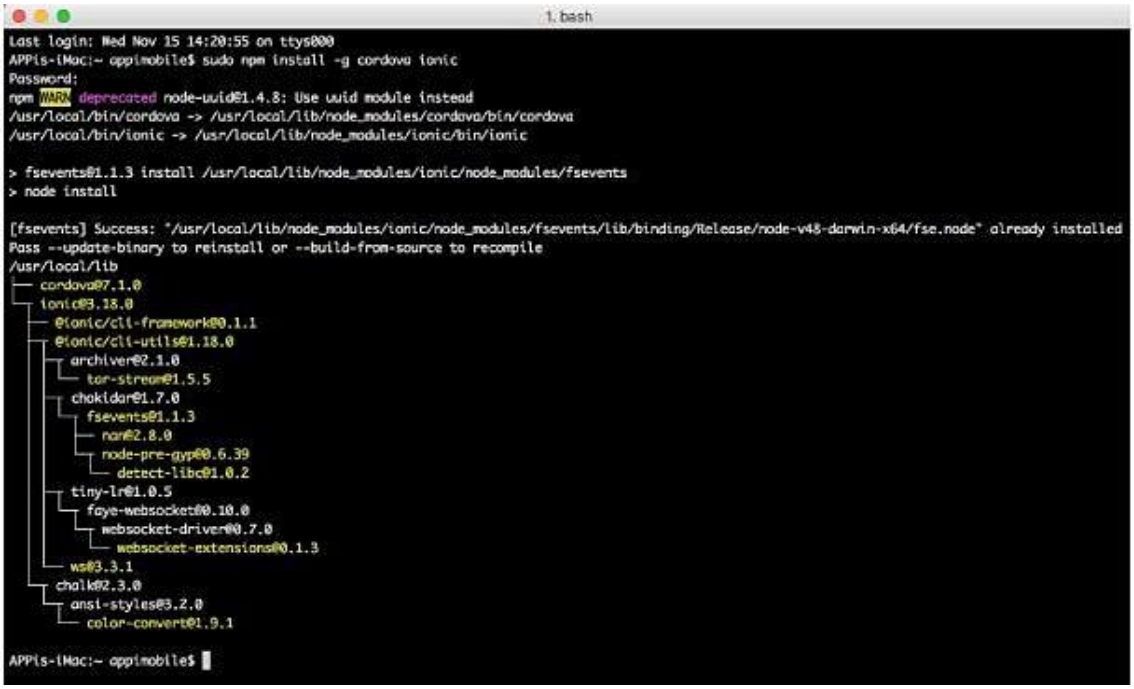
```
sudo npm install -g cordova ionic
```

Si se está trabajando en el sistema operativo de Windows, se debe ingresar a la consola de comandos e ingresar el siguiente comando:

npm install -g cordova ionic

Previo a realizar la instalación de Cordova e *Ionic*, es necesario instalar Node Js; ya que es necesario el administrador de paquetes NPM. Luego de haber ingresado el comando, deberán aparecer en la consola las siguientes instrucciones:

Figura 6. ***Framework Ionic***



Fuente: *Framework Ionic*. https://code.tutsplus.com/es/tutorials/ionic-from-scratch-getting-started-with-ionic--cms-29862. Consulta: 27 de junio de 2018.

Una vez instalado el *framework Ionic*, se puede proceder a crear la aplicación. Es necesario posicionarse sobre el directorio donde se quiere tener almacenada la aplicación. Se utilizará el comando 'start', el cual tiene la función de generar aplicaciones con distinta estructura como se detallan a continuación:

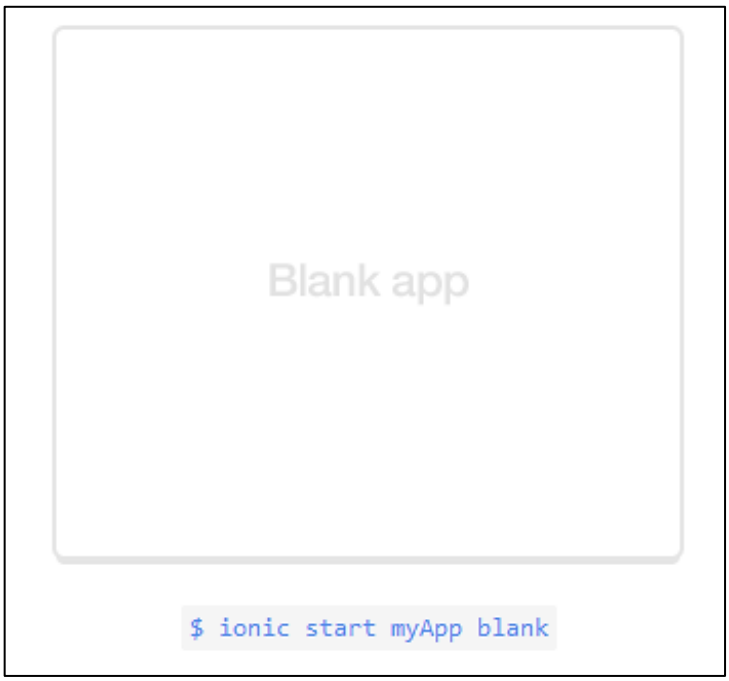
- *Tabs*: se crea una aplicación con una estructura con interfaz de pestañas.

Figura 7. ***Tabs***

Fuente: *Tabs*. https://i1.wp.com/expocodetech.com/wp-content/uploads/2014/09/ionic-tabsapp.png?w=335&ssl=1. Consulta: 29 de junio de 2018.

- *Blank*: se crea una aplicación con el proyecto en blanco.

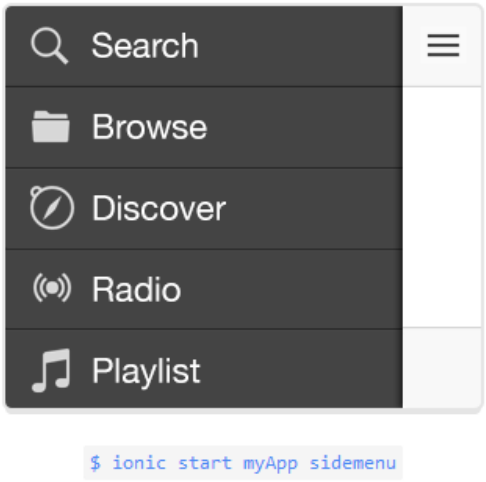
Figura 8. ***Blank***



Fuente: *Blank*. https://i2.wp.com/expocodetech.com/wp-content/uploads/2014/09/ionic-blankapp.png?w=351&ssl=1. Consulta: 30 de junio de 2018.

- Sidemenu: se crea una aplicación con una estructura con un menú lateral de navegación.

Figura 9. ***Sidemenu***

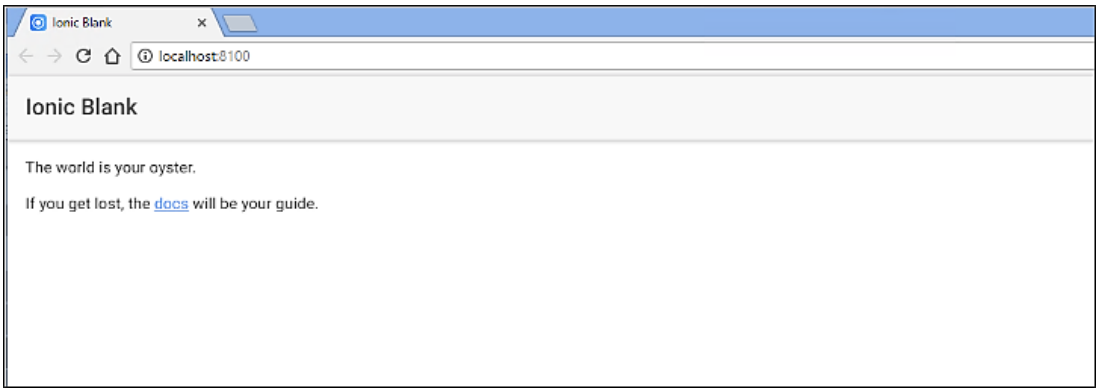


Fuente: *Sidemenu*. https://i1.wp.com/expocodetech.com/wpcontent/uploads/2014/09/ionic sidemenuapp.png?w=333&ssl=1. Consulta: 2 de junio de 2018.

En la etapa de desarrollo, se deben ir realizando pruebas en el navegador. Para ello, es necesario ejecutar el siguiente comando sobre el directorio donde actualmente se encuentra el código fuente de la aplicación.

- *Ionic server*: al ejecutar el comando, automáticamente se abrirá la ventana del navegador que esté configurado por defecto mostrando la aplicación.

Figura 10. ***Ionic server***

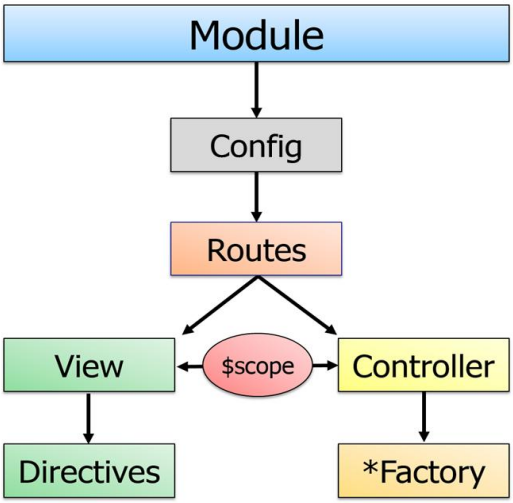


Fuente: *Ionic server*. https://cdn-images-1.medium.com/max/800/1*5gRFhvKYhhxoThXbfncd1Q.png. Consulta: 4 de julio de 2018.

1.6.2. Arquitectura de una aplicación en *Ionic*

Es importante saber que *Ionic* está basado en Angular, utiliza el patrón de diseño Modelo Vista Controlador; en este patrón la interfaz está divida en vistas. Los controladores se encuentran asociados a las vistas y proporcionan los datos necesarios para darle funcionalidad a los elementos.

Figura 11. **Arquitectura de una aplicación**



Fuente: *Arquitectura angular*. https://ajgallego.gitbooks.io/ionic/content/images/arquitectura_angular.png. Consulta: 5 de julio de 2018.

La arquitectura *Ionic* se encuentra formada por distintos elementos; vistas, controladores, rutas, directivas y servicios. Las principales funciones que tienen estos componentes dentro de una aplicación en *Ionic* son:

- Las vistas

Son páginas en formato HTML que contienen la descripción visual de la aplicación. Se almacenan en la carpeta 'templates'. Es posible mostrar información en ellas a través de una variable llamada $scope. Son una mezcla de HTML y CSS; es posible utilizar en ellas componentes de *Ionic* como barras, botones, modales, entre otros.

Figura 12. **Vistas**

```
<ion-view title="About">
  <ion-content>
    Contenido de la vista
  </ion-content>
</ion-view>
```

Fuente: *Vistas*. https://ajgallego.gitbooks.io/ionic/content/arquitectura_vistas.html. Consulta: 6 de julio de 2018.

En el ejemplo anterior se define una vista con ion-view, a la cual se le define un título y dentro de la vista se agrega contenido con ion-cotent.

- Los controladores

Son los encargados de obtener la información a través de servicios o factorías; contienen dentro de ellos toda la lógica de la aplicación. Cuando se muestra una página en la aplicación se llama al controlador y este invoca un template o vista para generar la página. El controlador envía información a la vista a través de la variable $scope.

Figura 13. **Controladores**

```
<ion-view title="About">
  <ion-content>
    Mi nombre es {{user.name}}.
  </ion-content>
</ion-view>
```

Para definir el controlador asociado `miSuperPaginaCtrl` lo podríamos realizar de la forma:

```
.controller('miSuperPaginaCtrl', function($scope, $stateParams, superService) {
    $scope.user = superService.getUser();
})
```

Fuente: *Controladores*. https://ajgallego.gitbooks.io/ionic/content/arquitectura_controladores.html. Consulta: 7 de julio de 2018.

En el ejemplo anterior se define una vista llamada mipagina.html, al acceder a ella se invocará automáticamente al controlador miSuperPaginaCtrl. El controlador recibe una función la cual será utilizada por Angular; se utiliza la variable $scope para pasar información del controlador hacia la vista y se carga un servicio que mostrará el nombre del usuario.

- Las rutas

Permiten acceder a una vista o un controlador en específico a través de su configuración. Es necesario, especificar la ruta inicial o por defecto un template y una vista.

Figura 14. **Rutas**

```
.config(function($stateProvider, $urlRouterProvider) {

    $stateProvider
        .state('home', {
          url: '',
          templateUrl: 'templates/home.html',
          controller: 'homeCtrl'
        })
        .state('page', {
          url: '/page',
          templateUrl: 'templates/page.html',
          controller: 'pageCtrl'
        });

    // Página por defecto
    $urlRouterProvider.otherwise('');
})
```

Fuente: *Rutas*. https://ajgallego.gitbooks.io/ionic/content/arquitectura_config.html. Consulta: 8 de julio de 2018.

En el ejemplo anterior se muestra la configuración de rutas para modelo y controlador. La vista home.html se asociará al controlador homeCtrl. Cuando la ruta se encuentre vacía se debería utilizar la ruta por defecto 'home'.

- Las directivas

Son un conjunto de elementos en Angular que permiten crear módulos personalizados de una manera muy sencilla. Estas directivas son aplicadas sobre el DOM y generan un comportamiento sobre este.

1.6.3. Componentes *Ionic*

Ionic tiene un conjunto de componentes que incluye en su librería; los cuales se utilizan para crear interfaces intuitivas fácilmente. Se utilizan etiquetas HTML para crear los componentes con opciones de configuración.

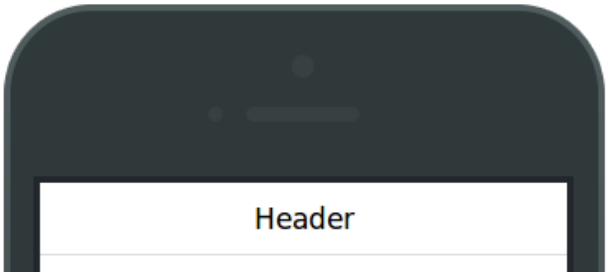
Los principales componentes que contiene la librería de Ionic son los siguientes:

- Cabeceras
- Áreas de contenidos
- Botones y enlaces
- Listados
- Pies de página

1.6.3.1. Cabeceras

Las cabeceras son elementos que se encuentran en la parte superior de la aplicación y pueden tener texto, botones e iconos; además, es posible personalizarlas con título y colores.

Figura 15. **Componentes cabeceras**




```
<ion-header-bar>
    <h1 class="title">Header</h1>
</ion-header-bar>
```

Fuente: *Componentes cabeceras*. https://ajgallego.gitbooks.io/ionic/content/componentes_cabeceras.html. Consulta: 10 de julio de 2018.

1.6.3.2. Área de contenidos

Todas las etiquetas que se encuentren dentro de la etiqueta de <body> se mostrarán en la interfaz de la aplicación. Dentro de esta sección se escribirán las sentencias HTML.

En el siguiente ejemplo, se utiliza la etiqueta <ionic-pane>, con esta etiqueta se define un panel en *Ionic*; y la etiqueta <ionic-content>; se utiliza para que el *scroll* funcione correctamente. Además, es posible agregar contenido dentro de esta etiqueta.

Figura 16. **Contenidos**

```
<body ng-app="starter">
    <ion-pane>
        <ion-content>
            <h1>I'm an H1!</h1>
            <h2>I'm an H2!</h2>
            <h3>I'm an H3!</h3>
            <h4>I'm an H4!</h4>
            <h5>I'm an H5!</h5>
            <h6>I'm an H6!</h6>
            <p>I'm a paragraph with a <a href="#">link</a>!</p>
        </ion-content>
    </ion-pane>
</body>
```

Fuente: *Componentes contenidos*. https://ajgallego.gitbooks.io/ionic/content/componentes_contenidos.html. Consulta: 11 de julio de 2018.

1.6.3.3. Botones

Los botones son componentes muy importantes en una aplicación móvil; le permiten interactuar al usuario con la aplicación y ejecutan una acción cada vez que el mismo los presiona. Además, es posible personalizarlos. En *Ionic* se utiliza la etiqueta ion- button.

```
<button ion-button>Button</button>
```

1.6.3.4. Listas

Las listas son componentes muy importantes en las aplicaciones móviles y permiten mostrar un conjunto de información de distintas formas. Los elementos

que pueden contener son: botones, iconos, imágenes y casi cualquier elemento HTML.

Figura 17. **Listado**

```
<ion-list>
  <ion-item>
    Elemento 1
  </ion-item>
  <ion-item>
    Elemento 2
  </ion-item>
  <ion-item>
    ...
  </ion-item>
</ion-list>
```

Fuente: *Listados*. https://ajgallego.gitbooks.io/ionic/content/componentes_listados.html. Consulta: 13 de julio de 2018.

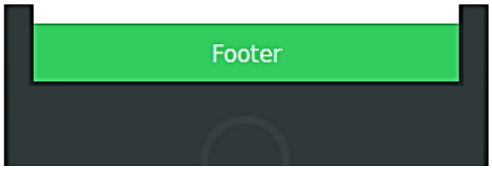
1.6.3.5. Pies de página

Los pies de página son elementos que se encuentran en la parte inferior de la aplicación y pueden contener texto, botones e íconos; es posible personalizarlos con título y colores.

Figura 18. ***Footers***

```
<ion-footer-bar class="bar-balanced">
  <h1 class="title">Footer</h1>
</ion-footer-bar>
```

Con lo que obtendríamos un resultado similar al siguiente:



Fuente: *Footers*. https://ajgallego.gitbooks.io/ionic/content/componentes_footers.html. Consulta: 17 de julio de 2018.

1.7. Antecedentes de las aplicaciones móviles

El propósito de las aplicaciones móviles es facilitar la vida de los usuarios. Encontrar un parqueo adecuado o localizar su vehículo; es un inconveniente que diariamente tienen que soportar miles de personas.

Se realizó una investigación en internet y en las tiendas en línea de *Android e IOS* y no existen aplicaciones que busquen parqueos en la ciudad de Guatemala con características como: tarifa, distancia y horario. Por el contrario, existen aplicaciones que pueden utilizarse en Guatemala para almacenar la ubicación del vehículo y así, evitar perderlo en centros comerciales o algún lugar.

A continuación, se presentan ejemplos de ambos tipos de aplicaciones. Entre las aplicaciones que almacenan la ubicación del automóvil, se pueden mencionar:

- Iparking

Es una aplicación para el control de estacionamiento de vehículos; se encuentra disponible para los sistemas operativos *Android e IOS*. Permite configurar rápidamente el parqueo y encontrar al instante el automóvil. La aplicación fue desarrollada por un italiano llamado Andrea Rosini y se encuentra disponible en las tiendas en línea de ambos sistemas operativos.

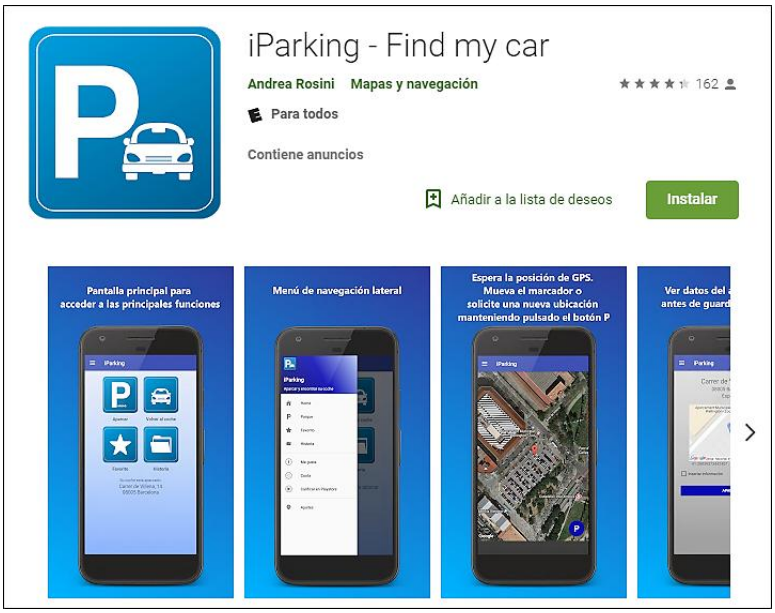
La aplicación obtiene la posición actual con ayuda del GPS; permite mover el marcador de posición de la ubicación detectada. Tiene distintas opciones como agregar una nota, una foto y un recordatorio. Iparking requiere de conexión a internet para mostrar el mapa y recuperar la información de ubicación.

La aplicación es compacta para los dispositivos IOS requiere de 8,9 MB para su almacenamiento. Además, requiere de la versión IOS 8.0 o posterior y es compatible con los siguientes dispositivos de Apple: *ip*hone, ipad, e ipod touch.

Para los dispositivos *Android* es requerido un espacio de almacenamiento de 4,6 MB. Es compatible, con versiones del sistema operativo *Android* 4.1 y posteriores. Se encuentra disponible en los idiomas español, inglés e italiano.

La licencia de la aplicación es gratuita para ambos sistemas operativos.

Figura 19. ***Iparking***



Fuente: *Iparking Android*. https://play.google.com/store/apps/details?id=andros.iparking&hl=es. Consulta: 21 de julio de 2018.

1.7.1. ***Sally Park***

Se encuentra disponible únicamente para los dispositivos con IOS. Utiliza el GPS para localizar el automóvil. Está disponible en español e inglés.

La aplicación sincroniza tiempos de parquímetros y envía notificaciones cuando el tiempo está por agotarse. La licencia de la aplicación es gratuita.

Los usuarios pueden agregar notas y capturar imágenes, como punto de referencia. *Sally Park* les permite a los usuarios saber dónde estacionaron su automóvil, tiene un sistema de mapas integrados y de navegación. Les brinda a sus usuarios un conjunto de instrucciones para encontrar el automóvil.

Figura 20. ***Sally park IOS***

Fuente: *Sally park IOS*. http://appcrawlr.com/ios/sally-park-free-the-best-car-fi#authors-description. Consulta: 23 de julio de 2018.

1.7.2. Find My Car

La aplicación está disponible para *Android e IOS*. Utiliza el GPS para fijar la posición en la cual se encuentra estacionado el vehículo y para ubicar nuevamente el vehículo utiliza Google Navigator. El uso de esta aplicación es gratuito.

Posee un conjunto de herramientas como brújula de navegación, permite almacenar notas y compartir la posición del automóvil con otras personas. Además, también es posible fotografiar el lugar donde el automóvil se encuentra parqueado.

Figura 21. ***Find my car***

Fuente: *Find my car*. https://www.motorafondo.net/las-mejores-aplicaciones-para-encontrar-tu-coche-en-el-aparcamiento/. Consulta: 25 de julio de 2018.

Tabla I. **Aplicaciones de búsqueda de vehículos**

Descripción	***Iparking***	***Find My Car***	***Sally Park***
Sistema operativo	*Android, IOS*	*Android IOS*	IOS
Tecnología	GPS	Google Navigator, GPS	GPS
Licencia	Gratuita	Gratuita	Gratuita
Funcionalidades	Guardar posición de vehículo, fotografías, notas	Guarda posición vehÍculo, compartir posición, notas, fotografías	Sincronización parquímetros, instrucciones, notas, fotografías

Fuente: elaboración propia.

Otros países en donde existen aplicaciones utilizadas para localizar vehículos son: Colombia, Estados Unidos y España.

Algunas aplicaciones existentes para ayudar a los usuarios a encontrar parqueos cercanos a su lugar de destino se encuentran:

1.7.2.1. Aplicación *City Parking*

La aplicación fue desarrollada por una empresa colombiana, la cual funciona en estacionamientos privados y públicos; permite a los usuarios encontrar fácilmente parqueos disponibles utilizando el GPS.

Permite encontrar parqueos cercanos a los usuarios, dependiendo su ubicación e incluso traza la ruta para que puedan llegar de una manera más fácil.

La aplicación ofrece varias herramientas de utilidad como consultar la disponibilidad de parqueos y tarifas de acuerdo al lugar; posee más de 91 parqueos registrados según soy502; en donde se encuentra la noticia al respecto:

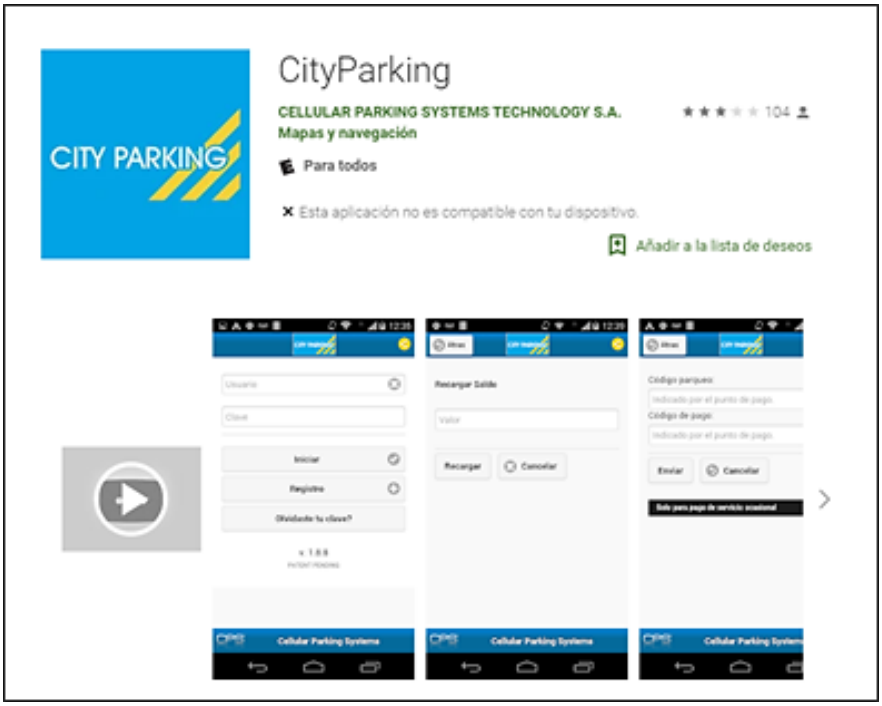
Sería una aplicación perfecta para una ciudad como Guatemala y cualquiera en el mundo, sin embargo, son cinco ciudades de Colombia las que cuentan con esta herramienta para los usuarios de teléfonos inteligentes.

City Parking tiene una billetera virtual donde los usuarios pueden recargar saldo para pagar su parqueo con anticipación y reservarlo.

Se encuentra disponible para las plataformas *Android e IOS*. En Android, es compatible para las versiones 4.0 y posteriores; en el sistema operativo IOS, es compatible con versiones IOS 8.0 y posteriores. Es compatible con todos los dispositivos de Apple.

La licencia de la aplicación es gratuita en IOS y Android.

Figura 22. ***City parking***



Fuente: *City parking*. https://www.soy502.com/sites/default/files/styles/escalar_image_inline/public/screen_shot_2014-08-18_at_5.49.01_pm.png. Consulta: 27 de Julio de 2018.

1.7.2.2. Aplicación *ParkMe*

La aplicación está desarrollada para dispositivos *Android e IOS* y ofrece parqueos en tiempo real.

Posee una base de datos muy completa con alrededor de 28 000 localidades en el mundo; permite acceder a un determinado parqueo y traza la ruta para que el usuario pueda dirigirse hacia el parqueo. Los usuarios pueden verificar la disponibilidad y las tarifas de los parqueos en tiempo real. Además, les permite reservar y pagar por anticipado un estacionamiento, ahorrandoles así tiempo.

La aplicación se encuentra disponible únicamente en idioma inglés y para grandes ciudades: New York, Tokio, Londres, Múnich, San Francisco, San Pablo.

Figura 23. ***Parkme android***

Fuente: *Parkme*. https://play.google.com/store/apps/details?id=com.parkme.consumer. Consulta: 27 de julio de 2018.

1.7.3. Aplicación *Waze*

Además de las aplicaciones mencionadas anteriormente no se puede obviar Waze. Esta aplicación comparte información vehicular del tráfico en tiempo real; ahorrándole así, tiempo a los usuarios para dirigirse a lugares específicos. Waze no está diseñada específicamente para la localización de parqueos, pero muestra a los usuarios parqueos cercanos a su destino.

Tampoco, posee la funcionalidad de buscar parqueos con base en características específicas como tarifa u horario.

Waze le brinda a los usuarios información acerca del tráfico, accidentes y peligros, entre otros. Además, permite a los usuarios enviar alertas de lo que ocurre en su camino para alertar a otros usuarios.

Se encuentra disponible para los sistemas operativos *Android e IOS*.

Figura 24. ***Waze***

Fuente: *Waze*. https://i2.wp.com/lopezdoriga.com/wp-content/uploads/2016/09/estacionamientos.jpg?resize=602%2C1024&ssl=1. Consulta: 29 de julio de 2018.

Tabla II. **Aplicaciones para búsquedas de parqueos**

Descripción	*City Parking*	*Park Me*	*Waze*
Sistema operativo	Android, IOS	Android, IOS	Android, IOS
Tecnología	GPS	GPS	GPS
Licencia	Gratuita	Gratuita	Gratuita
Funcionalidades	Búsqueda de parqueo, reservación, información de parqueos, billetera virtual	Reservas, Búsqueda de parqueos, información específica de parqueo, información en tiempo real	Información vehicular en tiempo real, recomendación de parqueos cercanos, envío de alertas

Fuente: elaboración propia.

1.8. Planteamiento del problema

La población en el país de Guatemala ha crecido exponencialmente y a consecuencia de ello, ha aumentado la cantidad de automóviles que circulan a diario en la ciudad de Guatemala. Se implementó una aplicación que facilitará al usuario la búsqueda de parqueos con distintas características como: tarifa adecuada, disponibilidad de horario, buena reputación y que se encuentre a una distancia cercana a su destino.

El gasto de gasolina en la búsqueda de parqueo es una consecuencia del problema. La mayoría de las personas que se transportan en automóviles cuenta con un *smartphone* con acceso a internet con aplicaciones que facilitan a sus dueños muchas tareas; en este caso, una es encontrar un parqueo adecuado.

Al contrario, de los países desarrollados como: Estados Unidos, España y Alemania que cuentan con aplicaciones especializadas para la búsqueda de parqueos. En Guatemala no existe alguna aplicación para el mismo uso.

La mayoría de personas utilizan aplicaciones como Google Maps o *Waze* para ubicar lugares. Por ello, se pensó en implementar una aplicación móvil utilizando la API de Google Maps, la cual les permite dirigirse a un determinado lugar de una manera muy fácil con un *smartphone*.

Esta serie de preguntas ayudará a determinar si es realmente factible implementar una aplicación que ayude a las personas a buscar parqueos.

- ¿El desarrollo de una aplicación permitirá a los usuarios ubicar un parqueo de una manera más fácil?

- ¿La aplicación reducirá los tiempos de búsqueda de un parqueo?

- ¿Se podría utilizar de manera gratuita con la ayuda de software libre?

2. DISEÑO DE APLICACIÓN

2.1. Módulo de administración

Página web de administración: se desarrollará en el lenguaje de programación PHP; tendrá distintas funcionalidades: registro de parqueos, modificación de parqueos y dar de baja a parqueos. Además, permitirá generar los inventarios de parqueos almacenados en la base de datos.

La aplicación web utilizará la tecnología *bootstrap*. Esta tecnología hará el sitio *responsive;* es decir, se podrá visualizar desde el dispositivo móvil.

2.2. Aplicación Móvil *Pfinder*

La aplicación móvil se llamará *Pfinder*; será desarrollada con el *framework* de *Ionic*, consumirá información de parqueos y usuarios en *web services* desarrollados en PHP, los cuales se encontrarán en la nube. Al acceder a la página, el usuario podrá visualizar un conjunto de parqueos cercanos a su posición actual.

El usuario podrá seleccionar un parqueo y visualizar información importante como dirección, distancia, capacidad, precio e incluso una breve descripción. El usuario seleccionará el parqueo que considere conveniente y el sistema le trazará la ruta correcta para dirigirse al mismo; utiliza geolocalización con ayuda del GPS.

Figura 25. **Buscador de parqueos**

Fuente: elaboración propia.

2.3. Arquitectura de la aplicación

La arquitectura de la aplicación será 3 capas. Una arquitectura 3 capas está construida por: capa de presentación, capa de negocio y capa de datos.

2.3.1. Capa de presentación

Es conocida como interfaz gráfica y se comunica directamente con la capa de negocio para transmitirle las acciones que el usuario realiza.

La capa de presentación es lo que el usuario puede visualizar. En esta capa es donde el sistema captura la información que le muestra al usuario.

El módulo de administración y la aplicación *Pfinder* serán la capa de presentación. El módulo de administración será una página web, en donde el usuario administrador podrá visualizar el reporte de parqueos y administrar los

mismos. En *Pfinder*, el usuario visualizará los parqueos disponibles y elegirá el que considere más conveniente.

2.3.2. Capa de negocio

Es la capa encargada de gestionar la lógica de la aplicación, recibir las peticiones de la capa de presentación y establecer las reglas del negocio. Esta capa se comunica con la capa de presentación para recibir las solicitudes y presentarlas a la capa de datos. La información retornada por la capa de datos es enviada hacia la capa de presentación.

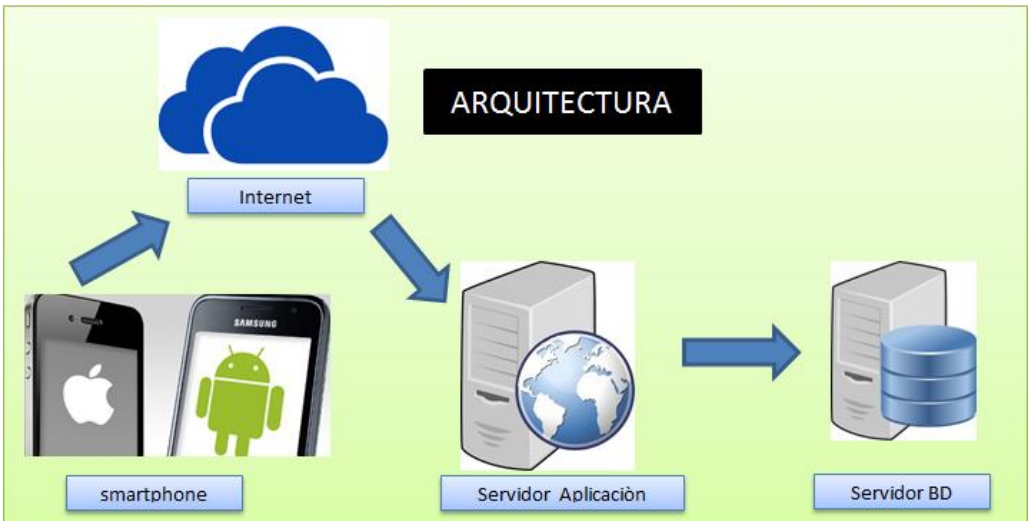
La aplicación PFINDER utiliza un servidor de *hosting* gratuito, el cual provee un servicio web en donde se encuentra toda la lógica de la aplicación y se comunica con el gestor de base de datos.

El *hosting* provee un conjunto de herramientas para su administración y cuenta con un servidor FTP, utilizado para subir archivos al servidor. Además, utiliza un servidor Apache para interpretar el lenguaje fuente de la aplicación web.

2.3.3. Capa de datos

Es la capa en donde se almacena la información de la aplicación; está formada por un gestor de base de datos que se encarga de la recuperación y administración de la base de datos. Se utilizó el gestor de base de datos *MySQL* el cual ofrece el servicio de *hosting* gratuito. Se cuenta con una capacidad de almacenamiento de 10 GB, el cual es posible ampliar en caso se necesite un escalamiento de información.

Figura 26. **Arquitectura de la app**



Fuente: elaboración propia.

2.4. Arquitectura aplicación móvil

Para el desarrollo de la aplicación móvil se utilizó un patrón de arquitectura MVC; el cual permitirá darle mantenimiento a la aplicación de una manera muy fácil en caso sea necesaria. Este modelo lo utiliza el *framework* Ionic.

Le brinda a los sistemas de software ventajas tales como autonomía y seguridad de la información.

Está dividido en tres componentes principales.

2.4.1. Componente vista

Es comúnmente la interfaz de usuario; en la vista se representa toda la lógica del negocio.

2.4.2. Componente controlador

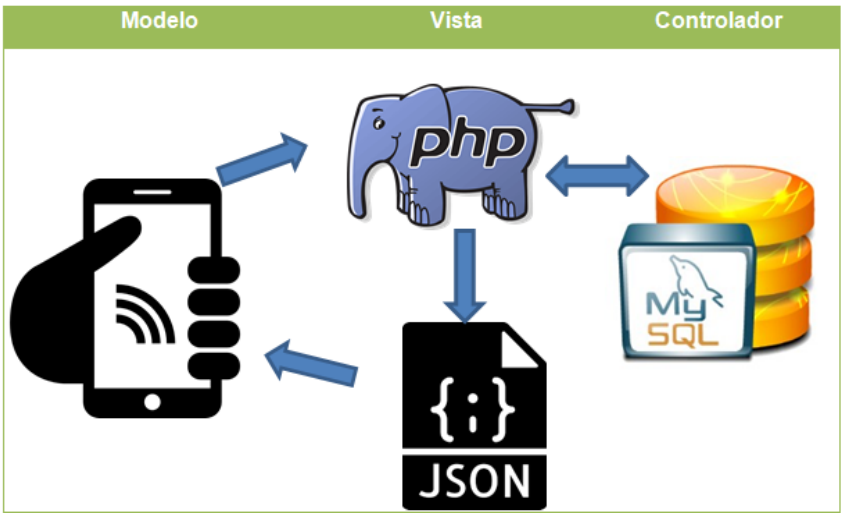
Es el encargado de interactuar directamente con la vista y el modelo; este responde a un evento en caso sea necesario.

2.4.3. Componente modelo

Es como se representa la información, en este caso es una base de datos. Se encarga de consultar, actualizar y buscar información en la base de datos.

La interfaz de usuario es la aplicación móvil, esta enviará peticiones al servidor web. El servidor web contiene la lógica del negocio. La lógica del negocio estará desarrollada en el lenguaje de programación PHP. El controlador y el modelo se comunican a través de archivos con formato *Json*; los archivos *Json* contienen en su estructura la información de los parqueos.

Figura 27. **Modelo**



Fuente: elaboración propia.

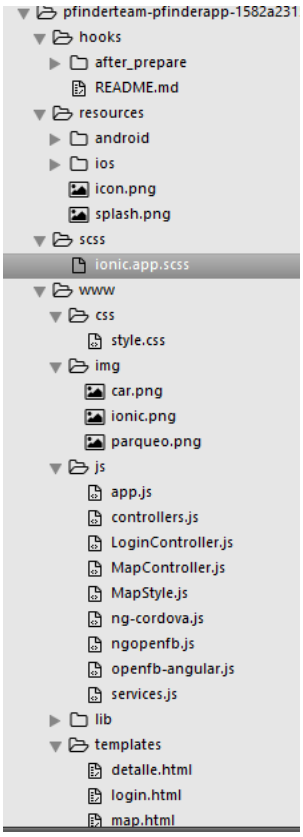
2.5. Código fuente y estructura

A continuación, se presenta la estructura de las carpetas.

2.5.1. Estructura de carpetas

La aplicación móvil tiene la estructura de un proyecto de *Ionic*. Está definida en carpetas las cuales tienen un funcionamiento en específico que se describe a continuación:

Figura 28. **Estructura de carpetas**



Fuente: elaboración propia

- hooks: son scripts que se ejecutan durante el proceso de construcción. Generalmente, se usan por comandos de Cordova para customización y la construcción de procesos automáticos.

- platforms: aquí es donde se crean los proyectos de *Android e IOS.* Podrían causar problemas específicos de la plataforma durante el desarrollo que requieren estos archivos; pero no se modificaran durante la mayor parte del tiempo.

- plugins: almacena los plugins de Cordova. Cuando se crea una aplicación de *Ionic*, ya hay algunos de estos plugins instalados.

- resources: estas carpetas se utilizan para añadir recursos como el icono y la pantalla de bienvenida.

- scss: esta es la carpeta donde se encuentra el archivo; desde que el núcleo de *Ionic* está construido con SASS.

- www: es la carpeta en donde se trabajará de forma habitual y mantiene una estructura de carpetas en su interior por defecto; pero se pueden modificar según las necesidades del proyecto.

    - css: aquí estarán los estilos CSS.

    - img: para las imágenes.

    - js: contiene el fichero principal de configuración app.js, los componentes de AngularJS (controladores, servicios, directivas) y todos los archivos js.

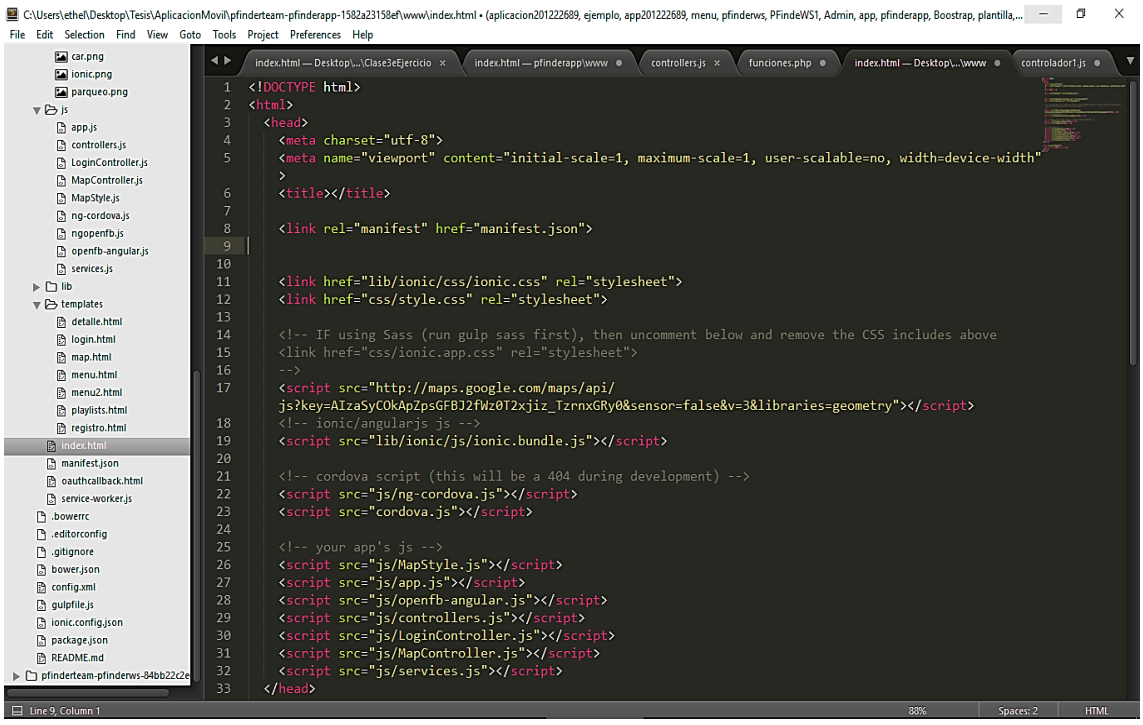
- libs: aquí deberán estar las librerías.
- templates: para los archivos html.
- index.html: el punto de inicio de la aplicación.

- Otros archivos
  - .bowerrc: el archivo de configuración de bower.
  - .editorconfig: el archivo de configuracion del editor.
  - .gitignore: sirve para elegir que parte de la aplicación es ignorada cuando se sube en un repositorio de Git.
  - bower.json: contiene los componentes y dependencias si se utiliza bower package manager.
  - gulpfile.js: Se utiliza para la creación de tareas automatizadas usando el administrador de tareas gulp.
  - config.xml: se refiere al archivo de configuración de Cordova.
  - package.json: contiene información sobre la app, dependencias y plugins ques son instalados usando np Miguel Angel junio 09.

2.5.2. Vista de estructura *html*

La aplicación móvil tiene una arquitectura formada por los componentes que se detallaron anteriormente. La capa de diseño o vista es la estructura html que el usuario puede visualizar.

Figura 29. **Estructura *html***



Fuente: elaboración propia.

En la página *html*, se importan los scripts que son necesarios para el buen funcionamiento de la aplicación móvil. En el *html* principal se importa la API de Google Maps que muestra los parqueos de la aplicación.

Figura 30. ***Google maps api***


```
    <script src="http://maps.google.com/maps/api/
    js?key=AIzaSyCOkApZpsGFBJ2fWz0T2xjiz_TzrnxGRy0&sensor=false&v=3&libraries=geometry"></script>
    <!-- ionic/angularjs js -->
    <script src="lib/ionic/js/ionic.bundle.js"></script>
```

Fuente: elaboración propia.

2.5.3. Controladores aplicación móvil

Los controladores contienen la lógica de la aplicación y se comunican directamente con la vista de la aplicación.

Figura 31. **Controladores de aplicación móvil**

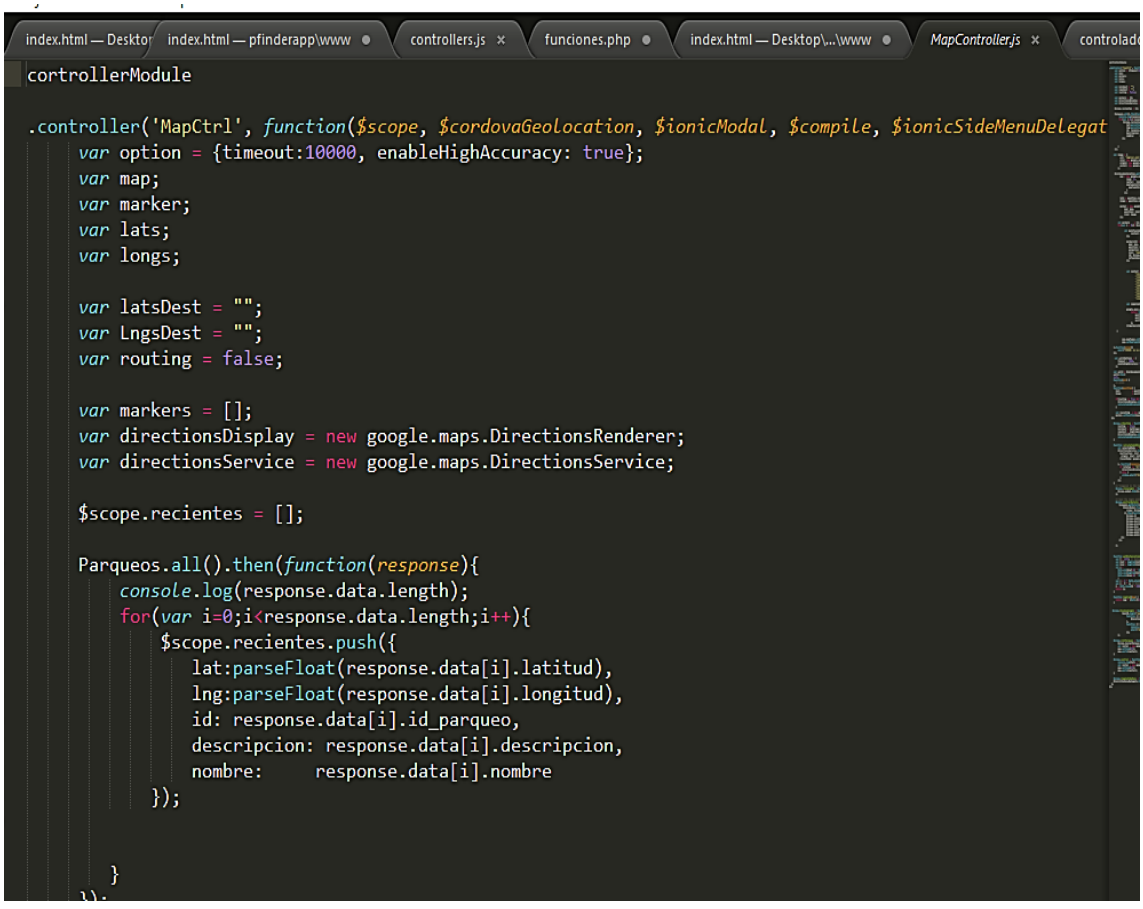
```
    <!-- your app's js -->
    <script src="js/MapStyle.js"></script>
    <script src="js/app.js"></script>
    <script src="js/openfb-angular.js"></script>
    <script src="js/controllers.js"></script>
    <script src="js/LoginController.js"></script>
    <script src="js/MapController.js"></script>
    <script src="js/services.js"></script>
</head>
```

Fuente: elaboración propia.

2.5.3.1. Controlador *MapController*

Este es el controlador más importante de la aplicación, contiene todos los métodos del mapa de la API de Google. Desde acá se dibujan todos los parqueos cercanos a la ubicación del usuario, se cargan los detalles de un parqueo en específico, se realizan los cálculos de distancia de parqueos y de ruta óptima.

Figura 32. ***Mapcontroller***



Fuente: elaboración propia.

2.5.3.2. Controlador *Service*

El archivo *Service js* contiene una 'Factory' que utiliza el protocolo HTTP para acceder al servicio web desarrollado en PHP. El servicio proporciona la información de los parqueos almacenados en la base de datos en una estructura *Jason*. El método 'All' obtiene todos los parqueos que se encuentran en la base de datos. El método 'getDetail' obtiene la información de un parqueo en específico solicitando como parámetro de entrada el identificador del parqueo.

Figura 33. ***Service js***

```
22
23      getDetail: function(id) {
24      var getParqueoWS = {
25              method: 'POST',
26              url: ApiEndpoint.url+'InfoParqueo.php',
27              data: {
28                  id: id
29              }
30             }
31
32      return $http(getParqueoWS)
33      .success(function(response){
34        return response;
35      }).error(function(data,status,headers,config){
36        console.log(data)
37        console.log(status)
38        console.log(headers)
39        console.log(config)
40        return null;
41      });
42    }
43  };
44 });
```

Fuente: elaboracion propia

Figura 34. ***Controllermodule***

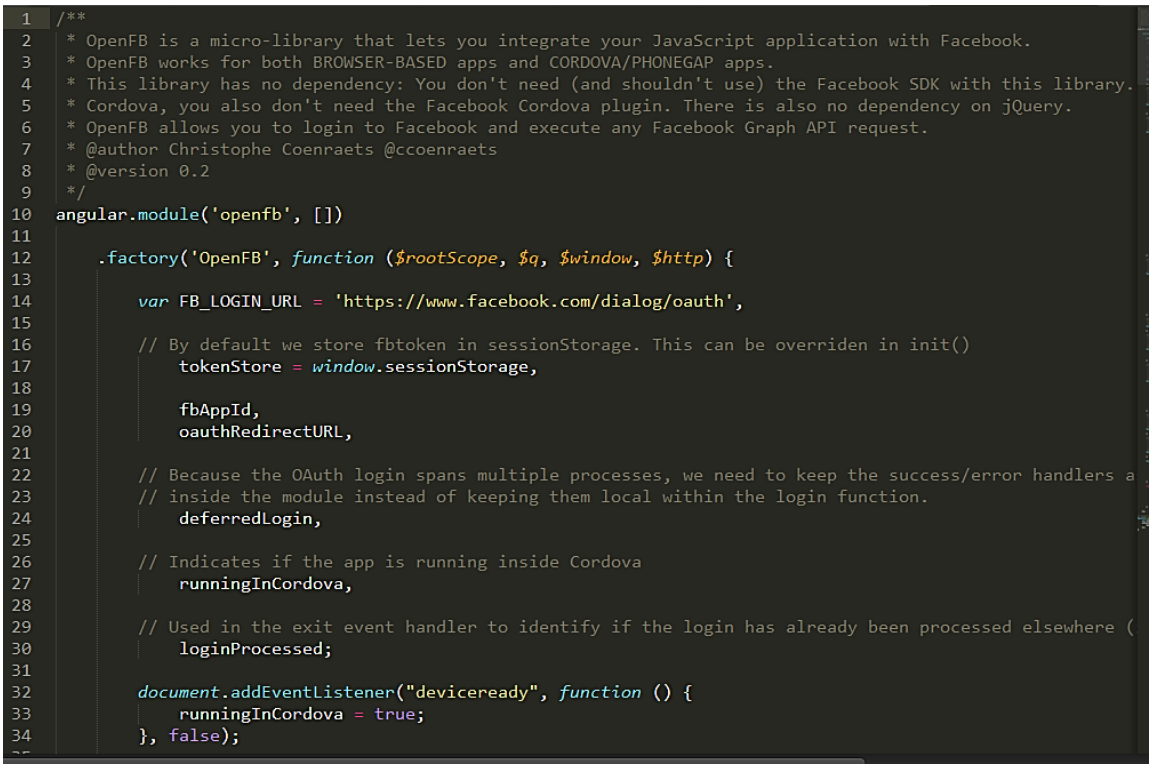
Fuente: elaboración propia.

2.5.3.3. Controlador *Open FB*

Para la comunicación de Facebook y la aplicación móvil se utilizó un Oauth. Un Oauth es un protocolo de autorización en el que un usuario autoriza a una aplicación a acceder a sus datos a través de sus credenciales. Facebook le permite a la aplicación un conjunto de permisos restringidos los cuales deben solicitarse.

Para la implementación del Oauth de Facebook se utilizó una Factory la cual contiene todos los métodos necesarios para implementar la funcionalidad.

Figura 35. ***Open FB***



```
/**
 * OpenFB is a micro-library that lets you integrate your JavaScript application with Facebook.
 * OpenFB works for both BROWSER-BASED apps and CORDOVA/PHONEGAP apps.
 * This library has no dependency: You don't need (and shouldn't use) the Facebook SDK with this library.
 * Cordova, you also don't need the Facebook Cordova plugin. There is also no dependency on jQuery.
 * OpenFB allows you to login to Facebook and execute any Facebook Graph API request.
 * @author Christophe Coenraets @ccoenraets
 * @version 0.2
 */
angular.module('openfb', [])

    .factory('OpenFB', function ($rootScope, $q, $window, $http) {

        var FB_LOGIN_URL = 'https://www.facebook.com/dialog/oauth',

        // By default we store fbtoken in sessionStorage. This can be overriden in init()
            tokenStore = window.sessionStorage,

            fbAppId,
            oauthRedirectURL,

        // Because the OAuth login spans multiple processes, we need to keep the success/error handlers a
        // inside the module instead of keeping them local within the login function.
            deferredLogin,

        // Indicates if the app is running inside Cordova
            runningInCordova,

        // Used in the exit event handler to identify if the login has already been processed elsewhere (
            loginProcessed;

        document.addEventListener("deviceready", function () {
            runningInCordova = true;
        }, false);
```

Fuente: elaboración propia.

2.5.4. Estructura *Json*

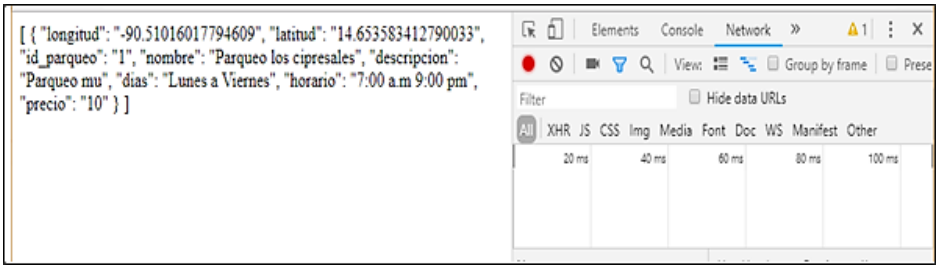
La información que necesita 'factory' es necesario visualizarla en una esctructura del archivo *Json*. Esta información es retornada por el método del servicio web BuscarParqueo.php.

Figura 36. ***Json***

Fuente: elaboración propia.

Cuando un usuario desea dirigirse a un parqueo en específico, es necesario obtener la información y detalle para dicho parqueo; esto se realiza a través del servicio web InfoParqueo.php. En donde es necesario enviar un parámetro de entrada, el cual es el identificador del parqueo. A continuación, se visualiza la estructura *Json* retornada.

Figura 37. **Estructura *Json***



Fuente: elaboración propia.

2.5.5. Servicios Web

La aplicación utiliza servicios web implementados en lenguaje de programación Php. El servicio de BuscarParqueo retorna la información de los parqueos realizando una conexión a la base de datos. La información obtenida la transforma en formato *'json'* a través de la función *json_encode* de php.

Figura 38. ***Json_encode***

```
function BuscarParqueo(){

$cadena=mysqli_connect("mysql.hostinger.es","u499246498_vinic","vinicio1993");


$line = array();

if($cadena)
{
mysqli_select_db($cadena,"u499246498_pfinb");
$res=mysqli_query($cadena,"select longitud, latitud,id_parqueo from parqueo");

$columnas=mysqli_affected_rows($cadena);
if($columnas>=1)
    {$i=0;
        while ($row=mysqli_fetch_array($res,MYSQLI_ASSOC)) {

            $line[$i]['longitud'] = $row['longitud'];
            $line[$i]['latitud'] = $row['latitud'];
            $line[$i]['id_parqueo'] = $row['id_parqueo'];
        $i++;
        }
    }
else
    {$resp="{respuesta:falso}";}

mysqli_close($cadena);


}
print json_encode($line, JSON_PRETTY_PRINT);
}
```

Fuente: elaboración propia.

Se utilizó un servicio web para obtener la información de un parqueo en específico, el servicio web necesita como parámetro de entrada el identificador del parqueo y retorna la información en un estructura *'json'.*

Figura 39. **Estructura *Json***

```
function ObtenerParqueo($id){
 $line = array();
 $cadena=mysqli_connect("mysql.hostinger.es","u499246498_vinic","vinicio1993");

 $resp="";

 if($cadena)
{

 mysqli_select_db($cadena,"u499246498_pfinb");
 $res=mysqli_query($cadena,"select nombre, direccion,id_parqueo,longitud, latitud, descripcion from parqueo where id_parqueo='$id'");

 $columnas=mysqli_affected_rows($cadena);


 if($columnas>=1)
    {$i=0;
        while ($row=mysqli_fetch_array($res,MYSQLI_ASSOC)) {

            $line[$i]['longitud'] = $row['longitud'];
            $line[$i]['latitud'] = $row['latitud'];
            $line[$i]['id_parqueo'] = $row['id_parqueo'];
            $line[$i]['nombre'] = $row['nombre'];
            $line[$i]['descripcion'] = $row['descripcion'];
        $i++;
        }
    }
 else
    {$resp="{respuesta:falso}";}

 mysqli_close($cadena);
}
print json_encode($line, JSON_PRETTY_PRINT);
}
```

Fuente: elaboración propia.

2.6. Base de datos de la aplicación

Una base de datos es un conjunto de información de forma ordena y organizada y tiene un contexto en común. La información de la base de datos puede expandirse y relacionarse de manera que se puede acceder a ella de manera sencilla.

La representación gráfica de una base de datos relacional se hace a través de un modelo entidad relación. El modelo entidad-relación permite visualizar la estructura lógica de la base de datos que almacenará la información. Está compuesta por los siguientes elementos:

2.6.1. Entidad

Es un objeto del mundo real del cual se desea almacenar información en la base de datos. Se representa en el diagrama entidad-relación por un rectángulo y en la base de datos se representa como una tabla con información.

2.6.2. Atributos

Los atributos son características o propiedades importantes que definen una entidad; pueden ser de distintos tipos: numéricos, texto o fecha.

Son muy importantes para las entidades ya que las entidades se diferencian por medio del valor de sus atributos.

2.6.3. Relaciones

Son vínculos que permiten asociar las entidades en un diagrama entidad-relación. Hacen posible que las entidades compartan ciertos atributos entre sí. Las relaciones se clasifican de acuerdo a su cardinalidad; es decir, indica el número de entidades con las que puede estar relacionada una entidad con otra.

2.6.3.1. Relación uno a uno

Una entidad A se relaciona únicamente con una entidad B y la entidad B se relaciona únicamente con la entidad A. Este tipo de relación no es muy común ya que la información generalmente se representa en una sola tabla.

2.6.3.2. Relación uno a muchos

Un registro de una entidad A puede estar relacionado uno o muchas veces con la entidad B.

2.6.3.3. Relación muchos a muchos

Un registro de la entidad A puede estar relacionada con muchos registros de la entidad B y viceversa. Este tipo de relaciones no es posible representarlo en un modelo entidad-relación; para representarlo, se utiliza una tabla intermedia que elimina la relación muchos a muchos; convirtiéndo dos relaciones de uno a muchos.

Las entidades del modelo entidad relación se convertirán en tablas de las cuales se identificaron:

- Parqueo: en tabla se almacenan las características más importantes de los parqueos.

- Estado: almacena los posibles estados que puede tener un parqueo. Un parqueo puede darse de bajo o de alta.

- Tipo vehículo: los parqueos pueden ser utilizados para distintos tipos de vehículos motocicletas, bicicletas o automóviles.

- Tarifa: los parqueos cobran de distintas maneras a sus clientes. En esta tabla se almacenarán estos tipos de tarifas, los cuales podrían ser por hora, fracción o nocturna.

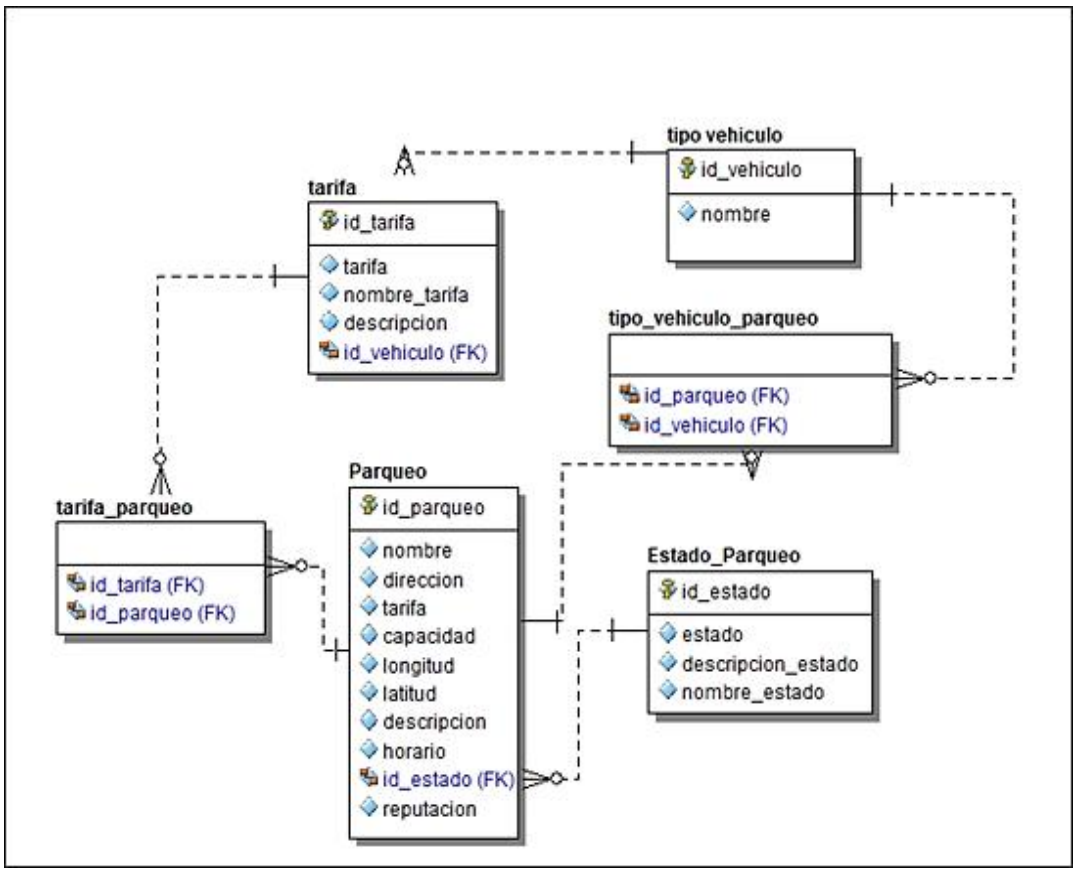
El diagrama entidad-relación planteado está compuesto por varias tablas con posibilidad de expandirse en un futuro. Entre las tablas más importantes en donde se almacenará la información está la de parqueos; la cual almacena metadatos del parqueo como nombre, dirección, descripción, capacidad, horario y lo más importante la longitud y la latitud, lo cual permite ubicar en el mapa los parqueos.

El administrador puede dar de baja o de alta un parqueo, para llevar ese control se utilizará la tabla de “Estado parqueo”. Un parqueo únicamente puede tener asignado un estado de parqueo.

Los parqueos pueden proveer servicio a varios medios de transporte: vehículos, motocicletas, bicicletas. Asimismo, un tipo de vehículo se encuentra en un parqueo o más. En este caso se identificó una relación de muchos a muchos; para eliminarla, se utilizó una entidad asociativa llamada 'tipo vehículo parqueo' la cual permite almacenar la información de una manera bien definida.

Un parqueo cobra su tarifa de distintas formas; esta información se almacenó en la tabla 'tarifa'. Las tarifas de los parqueos son variables de acuerdo al tipo de vehículo y al parqueo. Para almacenar la información de forma adecuada se utilizó la entidad asociativa 'tarifa parqueo'.

Figura 40. **Diagrama entidad-relación**

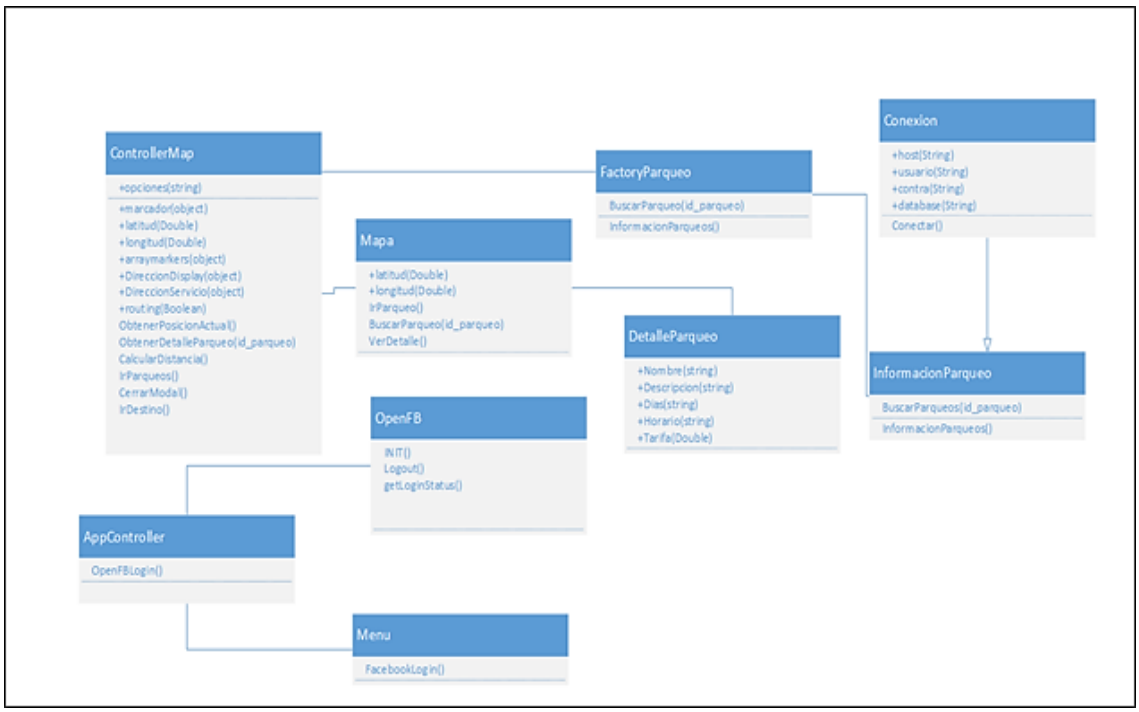


Fuente: elaboración propia.

2.7. Diagrama de clases

Un diagrama de clases es una representación gráfica de la estructura de un sistema de software; permite visualizar las clases que componen el sistema y las relaciones entre ellas. Además, describe los atributos y métodos que forman las clases.

Figura 41. **Diagrama de clases**



Fuente. elaboración propia.

2.8. Requerimientos del sistema

A continuación, se detallan los requerimientos funcionales y no funcionales del sistema en sus dos módulos.

Los requerimientos funcionales serán divididos en los que son necesarios para la aplicación móvil y los requerimientos para la plataforma web.

2.8.1. Requerimientos funcionales para la aplicación móvil

Los requisitos funcionales identificados para la aplicación móvil son los siguientes:

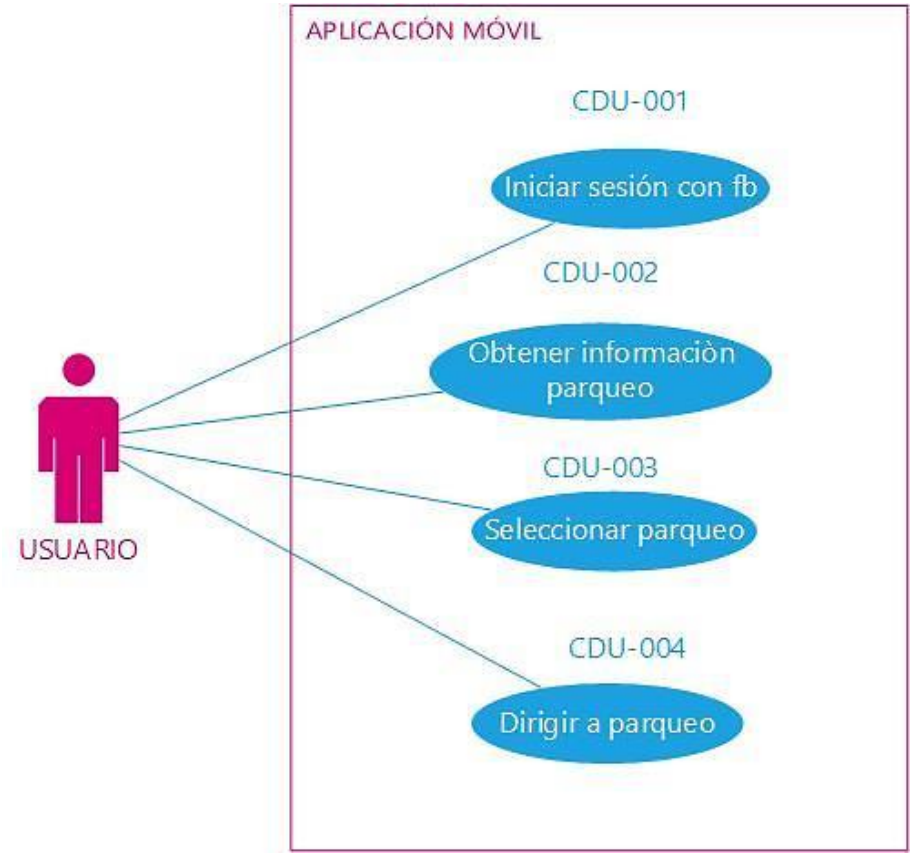
- Requerimiento funcional 01 (RF01): la aplicación solicitará al usuario registrarse por medio de la red social Facebook.

- Requerimiento funcional 02 (RF02): el usuario debe seleccionar un parqueo y puede obtener más información del mismo como distancia, tarifa, capacidad del parqueo y seleccionar las más adecuada para satisfacer sus necesidades.

- Requerimiento funcional 03 (RF03): el usuario debe seleccionar el icono de 'Ir' y la aplicación le trazará la ruta para llegar al parqueo desde su ubicación actual.

- Requerimiento funcional 04 (RF04): la aplicación le mostrará al usuario la ruta más óptima desde su ubicación actual hasta el parqueo seleccionado.

Figura 42. **Diagrama casos de uso plataforma**



Fuente: elaboración propia.

2.8.2. Requerimientos funcionales para la plataforma web

Los requerimientos funcionales identificados para la plataforma web del administrador son los siguientes:

- Requerimiento funcional 06 (RF06): el sistema tendrá un inventario de parqueos con la información más importante; en esta, el administrador puede registrar nuevos parqueos que él considere necesarios.

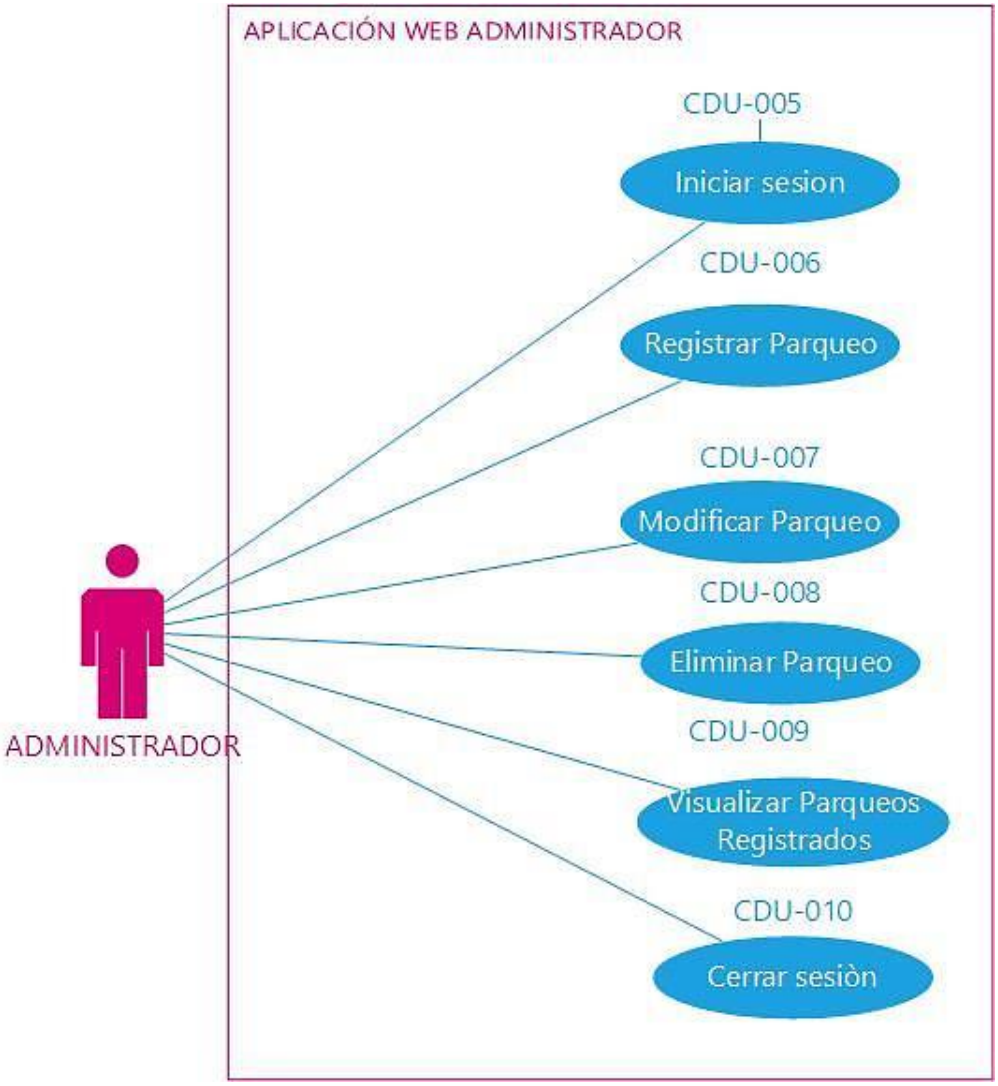
- Requerimiento funcional 07 (RF07): la aplicación tendrá un *login*, donde el administrador deberá ingresar sus credenciales para tener acceso al sistema y poder realizar diversas funciones.

- Requerimiento funcional 08 (RF08): el administrador deberá completar los campos con la información de cada parqueo, la cual será almacenada en la base de datos para registrar nuevos parqueos.

- Requerimiento funcional 9 (RF9): el sistema tiene la capacidad de modificar y eliminar parqueos, esto en caso se ingrese información errónea o se desee modificar la información de un parqueo en específico.

- Requerimiento funcional 10 (RF10): la aplicación web expirará la sesión del usuario administrador en caso este no haya cerrado sesión.

Figura 43. **Diagrama casos de uso plataforma web**



Fuente: elaboración propia.

2.8.3. Requerimientos no funcionales

Los requerimientos no funcionales establecidos para el sistema serán divididos en los que son necesarios para la aplicación móvil y para la plataforma web.

2.8.3.1. Requerimientos no funcionales para aplicación móvil

Los requerimientos no funcionales identificados para la aplicación móvil son los siguientes:

- Requerimiento no funcional 01 (RNF01): disponibilidad, el servicio de *hosting* debe estar disponible el 99 % del tiempo para que el usuario logré visualizar la información de parqueos.

- Requerimiento no funcional 02 (RNF02): conexión a internet, activar el GPS del dispositivo móvil; deberá tener conexión a internet para el envío de mensajes al servidor centralizador.

- Requerimiento no funcional 03 (RNF03): la aplicación móvil estará disponible para las versiones de Android Ice cream versión 4.1- 4.05, Jelly Bean 4.1 - 4.3.1, Kit Kat 4.4 -4.4.4, Lollipop 5.1- 5.1.1 Marshmallow 6.0-6.0.1, Nougat 7.0-7.1.2, Oreo 8.0-8.1. El dispositivo debe tener como mínimo 1 GB de memoria RAM. En el sistema operativo IOS, deberá contar como mínimo con 1 GB de memoria RAM y será compatible con las siguientes versiones: IOS 4, IOS5, IOS6, IOS7, IOS 8.

2.8.3.2. Requerimientos no funcionales plataforma web

Los requerimientos no funcionales identificados para la plataforma web son los siguientes:

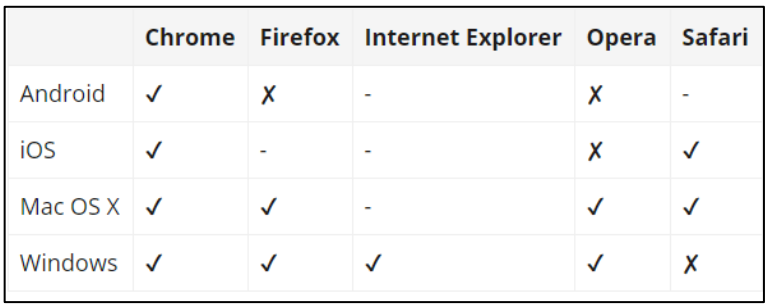
- Requisito no funcional 04 (RNF04): disponibilidad, la plataforma web deberá estar disponible el 99 % del tiempo, lo cual representa un margen de tiempo de 14,4 minutos en los que el sistema puede no estar disponible.

- Requisito no funcional 05 (RNF05): seguridad, se verificará la identidad de todos los usuarios mediante la autenticación a través de su usuario y contraseña. También, la plataforma deberá otorgar permisos de usuarios dependiendo de los roles que existan dentro del sistema.

- Requisito no funcional 06 (RNF06): la plataforma deberá proporcionar mensajes de error al usuario final que informen y orienten acerca de los fallos en el uso de una funcionalidad.

- Requisito no funcional 07(RF7): la aplicación se desarrolló utilizando la tecnología de Bootstrap 3.0 lo que permite que sea *responsive*; es decir, pueda visualizarse en un dispositivo móvil. Boostrap es compatible con las siguientes versiones de sistemas operativos.

Figura 44. **Compatibilidad con navegadores**



	Chrome	Firefox	Internet Explorer	Opera	Safari
Android	✓	✗	-	✗	-
iOS	✓	-	-	✗	✓
Mac OS X	✓	✓	-	✓	✓
Windows	✓	✓	✓	✓	✗

Fuente: *Compatibilidad con navegadores*. :http://librosweb.es/libro/bootstrap_3/capitulo_1/compatibilidad_con_los_navegadores.html. Consulta: 15 de abril de 2018.

- Requisito no funcional 07 (RNF07): compatibilidad, la plataforma deberá funcionar en los navegadores Internet Explorer: versión 5, versión 6, versión 7, versión 8, versión 9, versión 10, versión 11; Microsoft Edge, Google Chrome en versión 16 a versión 22; Mozilla Firefox versión 50.0 al versión 62.0; Safari versiones 3, 4,n5.

2.9. Casos de uso de la aplicación móvil

A continuación se presenta el caso de uso de la aplicación.

Tabla III. **Iniciar sesión**

Caso de uso	CDU-001 iniciar sesión
Actores	Usuario
Tipo	primario
Descripción	El usuario podrá iniciar sesión en la aplicación móvil por medio de un *oauth* de Facebook donde ingresará sus credenciales de correo y contraseña.

Fuente: elaboración propia.

Tabla IV. **Obtener información de parqueo**

Caso de uso	CDU-002 obtener información de parqueo
Actores	Usuario
Tipo	primario
Descripción	Al momento de ingresar a la aplicación móvil se le mostrarán al usuario, los parqueos que se encuentran cercanos a su ubicación actual a 10 km de su ubicación El usuario podrá seleccionar un parqueo y al presionar obtener información; se le mostrará la información más importante: tarifa, descripción, dirección exacta y distancia desde su punto actual.

Fuente: elaboración propia.

Tabla V. **Seleccionar parqueo**

Caso de uso	CDU-003 seleccionar parqueo
Actores	Usuario
Tipo	primario
Descripción	El usuario podrá seleccionar un parqueo en específico de los parqueos disponibles. El usuario visualizará la información del parqueo seleccionado.

Fuente: elaboración propia.

Tabla VI. **Dirigirse al parqueo**

Caso de uso	CDU-004 dirigirse al parqueo
Actores	Usuario
Tipo	primario
Descripción	Una vez el usuario haya visualizado la información del parqueo seleccionado, elegirá dirigirse al mismo. La aplicación le mostrará la ruta que deberá seguir para llegar a la ubicación del parqueo utilizando Google Maps.

Fuente: elaboración propia.

2.10. Casos de uso de *Pfinder*

Tabla VII. ***Pfinder*: iniciar sesión**

Caso de uso	CDU-005 iniciar sesión
Actores	Administrador de parqueos
Tipo	primario
Descripción	La persona encargada de administrar los parqueos deberá ingresar a la página web con sus credenciales (usuario y contraseña); desde allí podrá realizar las funciones necesarias para administrar la aplicación web de parqueos.

Fuente: elaboración propia.

Tabla VIII. ***Pfinder*: registrar parqueos**

Caso de uso	CDU-006 registrar parqueos
Actores	Administrador de parqueos
Tipo	primario
Descripción	La aplicación web le permitirá al administrador registrar nuevos parqueos en caso sea necesario; para ello el administrador ingresará los datos de longitud, latitud, descripción entre otros para registrar el parqueo.

Fuente: elaboración propia.

Tabla IX. ***Pfinder*: modificar parqueo**

Caso de uso	CDU-007 modificar parqueo
Actores	Administrador de parqueos
Tipo	primario
Descripción	El sistema permite modificar un parqueo previamente registrado. Para ello el usuario deberá buscar el parqueo, seleccionarlo y luego; ingresar la información para proceder a modificar el parqueo.

Fuente: elaboración propia.

Tabla X. ***Pfinder*: eliminar parqueo**

Caso de uso	CDU-008 eliminar parqueo
Actores	Administrador de parqueos
Tipo	primario
Descripción	La persona encargada de administrar los parqueos, puede eliminar parqueos de la base de datos en caso sea necesario dar de baja un parqueo. Para eliminar el parqueo, deberá buscarlo y luego proceder a eliminarlo.

Fuente: elaboración propia.

Tabla XI. ***Pfinder*: visualizar parqueos registrados**

Caso de uso	CDU-009 visualizar parqueos registrados
Actores	Administrador de parqueos
Tipo	primario
Descripción	La persona encargada de administrar los parqueos puede visualizar el catálogo de parqueos con la descripción de cada parqueo. Para realizar dicha acción deberá acceder a la opción reporte de parqueos.

Fuente: elaboración propia.

Tabla XII. ***Pfinder*: cerrar sesión**

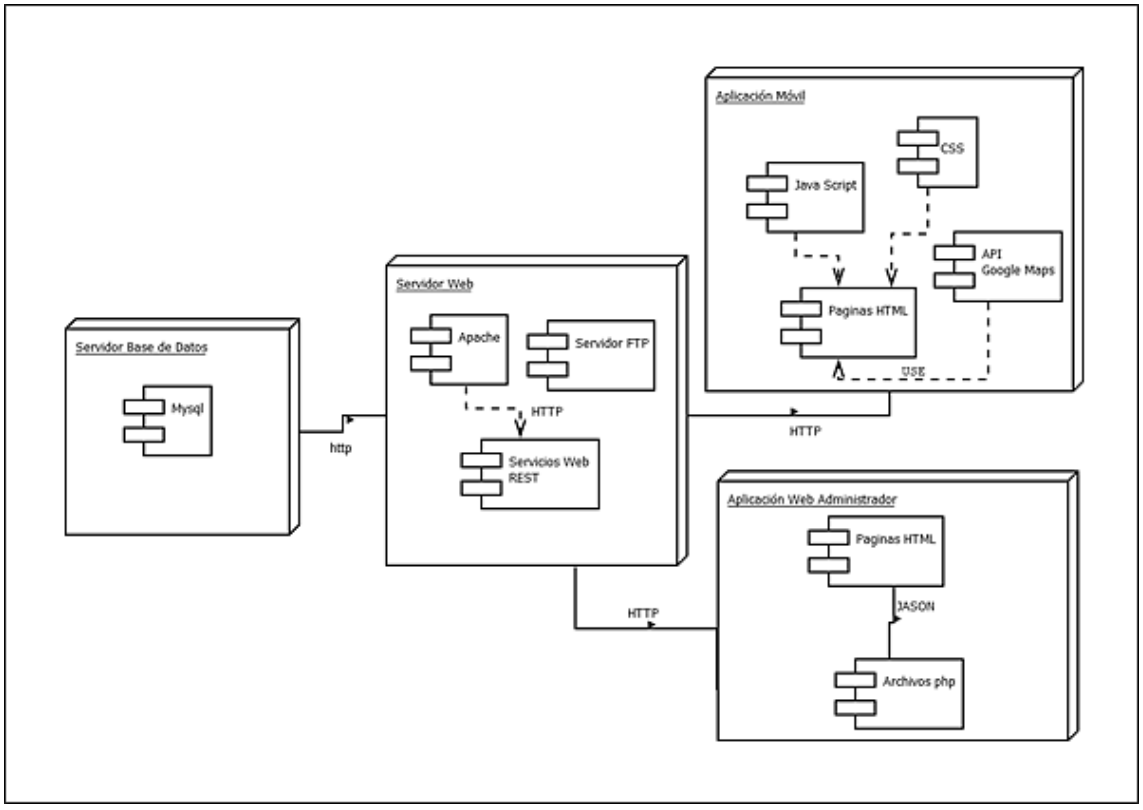
Caso de uso	CDU-010 cerrar sesión
Actores	Administrador de parqueos
Tipo	primario
Descripción	Una vez realizada las acciones necesarias el administrador puede cerrar sesión en el sistema para eso deberá seleccionar la opción 'cerrar sesión'.

Fuente: elaboración propia.

2.11. Diagrama de despliegue

El diagrama de despliegue o de implementación representa gráficamente los artefactos de software y componentes por los que está formado un sistema. Está compuesto de nodos; un nodo es un elemento de software o hardware dentro del sistema y se representa por una caja. Un nodo puede contener componentes, los cuales son denominados artefactos y se representan por un rectángulo en el diagrama.

Figura 45. **Diagrama de despliegue**

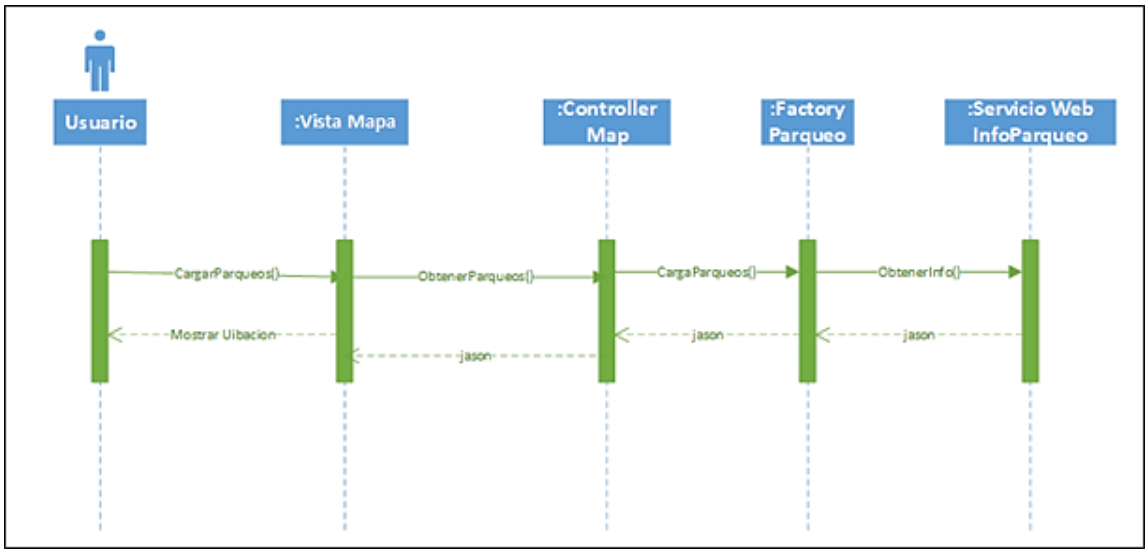


Fuente: elaboración propia.

2.12. Diagrama de secuencia

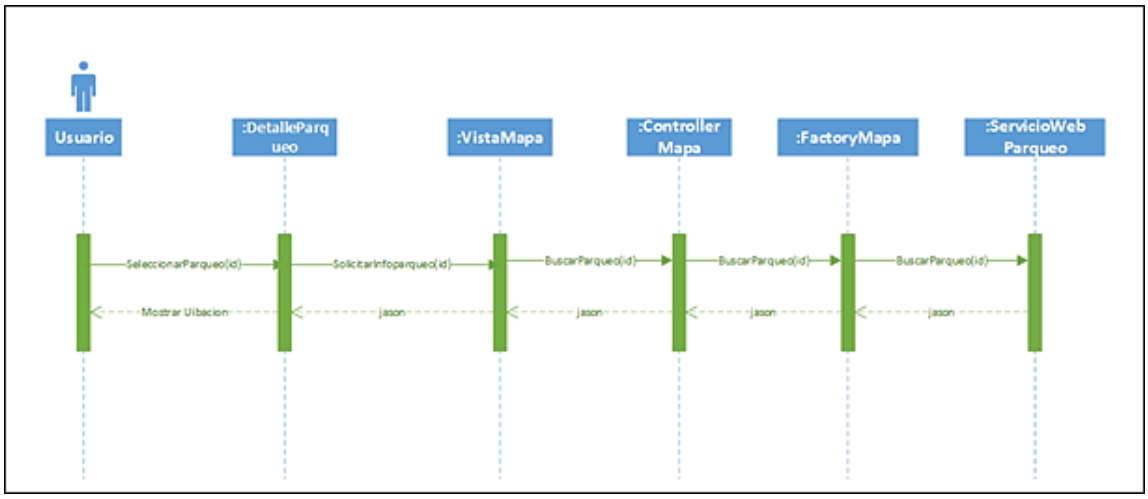
Un diagrama de secuencia permite visualizar la interacción entre los objetos de un sistema de software a través del tiempo. El diagrama de secuencia está formado por objetos, los cuales se comunican a través de mensajes. El objeto tiene una vida de línea representado por una línea vertical.

Figura 46. **Diagrama de secuencia: obtener información de parqueos**



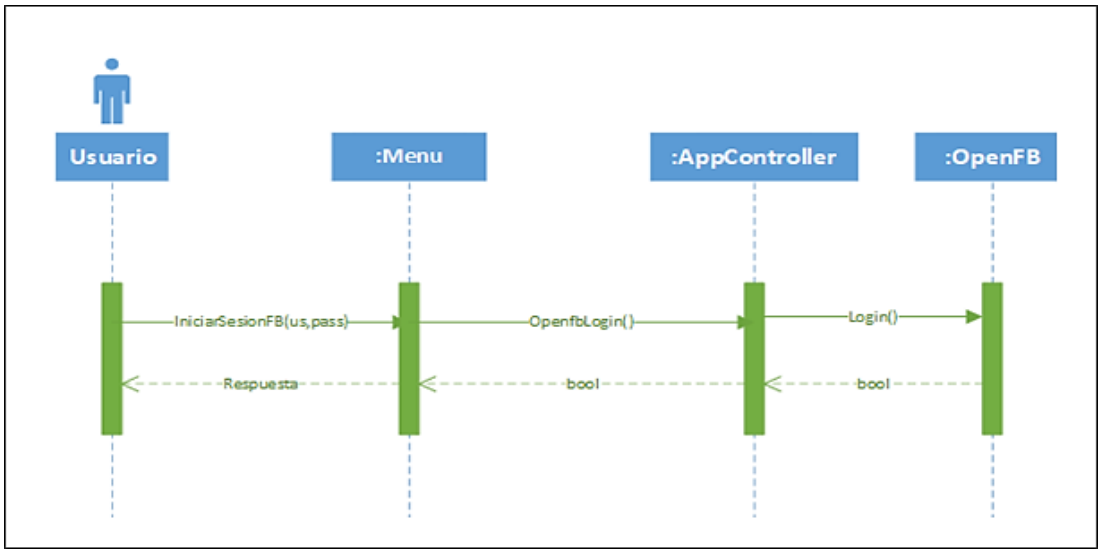
Fuente: elaboración propia.

Figura 47. **Diagrama de secuencia: seleccionar parqueo**



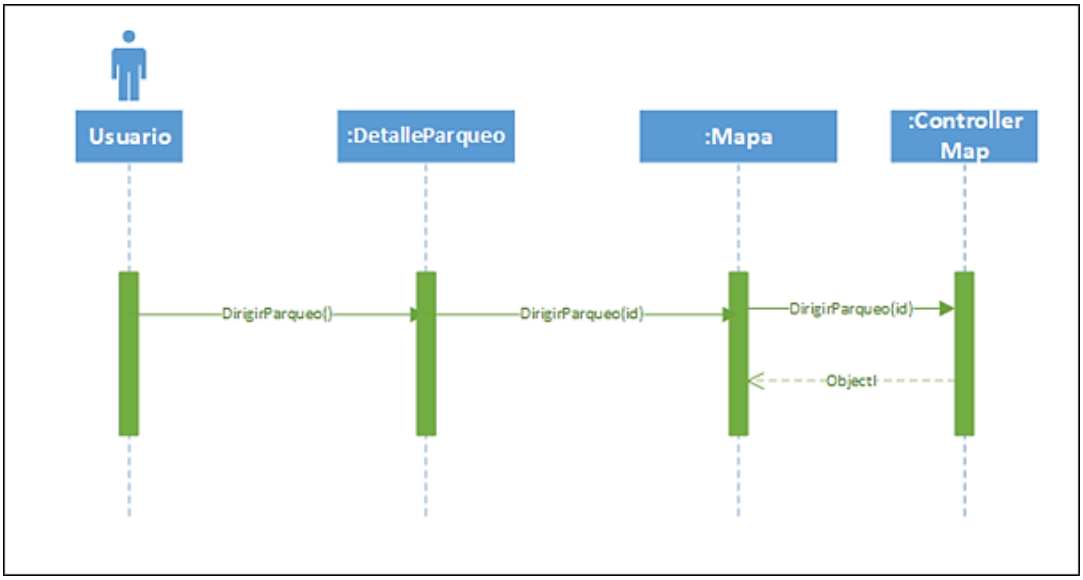
Fuente: elaboración propia.

Figura 48. **Diagrama de secuencia: iniciar sesión Facebook**



Fuente: elaboración propia.

Figura 49. **Diagrama de secuencia: dirigir parqueo**



Fuente: elaboración propia.

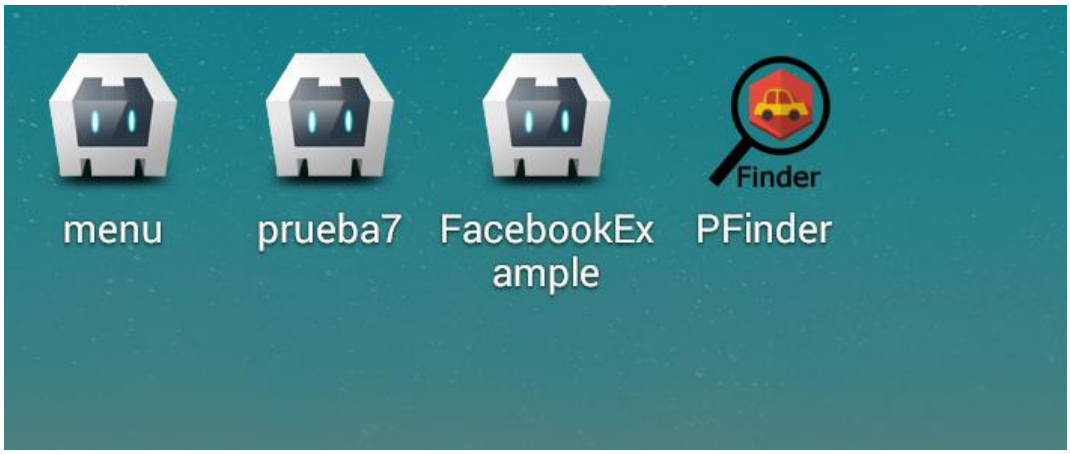
3. IMPLEMENTACIÓN DE SOLUCIÓN

3.1. Aplicación móvil

A continuación, se muestra como se visualiza la aplicación con las funcionalidades de búsqueda de parqueos y su manejo.

Se puede visualizar el icono al cual el usuario tiene que acceder para ingresar a la aplicación.

Figura 50. **Aplicación móvil**



Fuente: elaboración propia.

Se muestra el mapa de la aplicación móvil. El icono rojo representa la ubicación actual donde se encuentra posicionado el usuario. Para visualizar el mapa y la ubicación, es necesario activar el GPS del móvil y tener acceso a internet.

Figura 51. **Interfaz *Pfinder***

Fuente: elaboración propia.

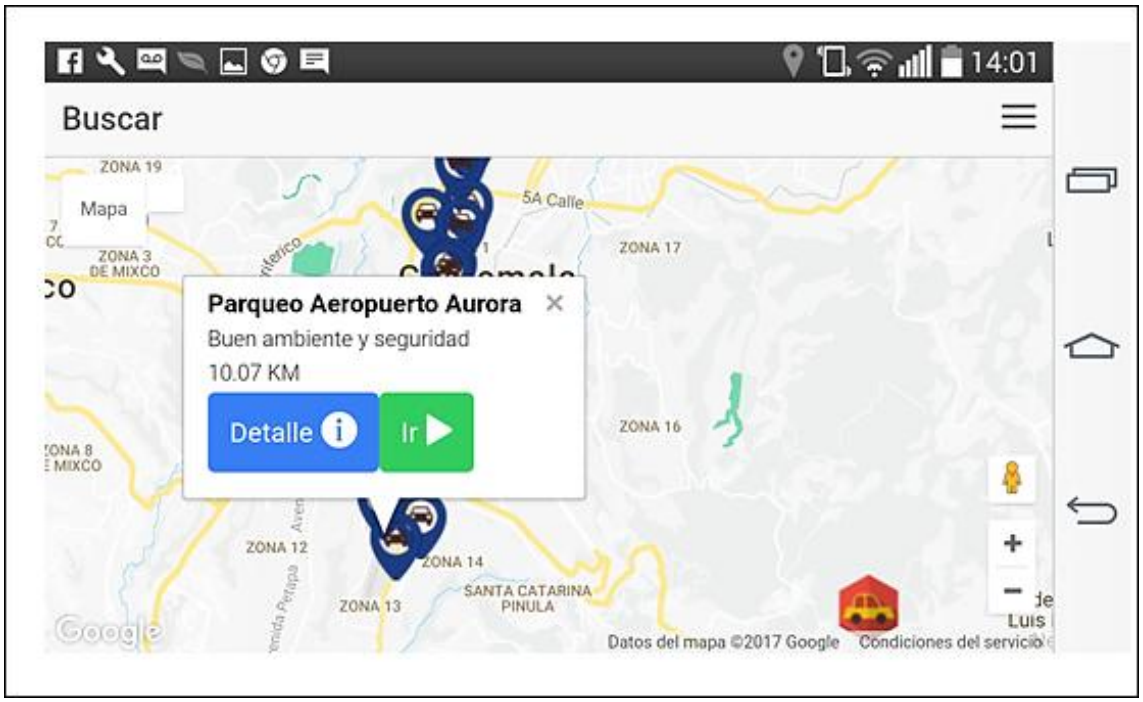
Se pueden visualizar los parqueos que están guardados en la aplicación. Se muestran los parqueos con ubicación más cercana al usuario.

Figura 52. **Localización de parqueos**

Fuente: elaboración propia.

El usuario seleccionará el parqueo que considere más conveniente y automáticamente se le mostrará la información del parqueo, así como la distancia a la que se encuentra respecto a su posición.

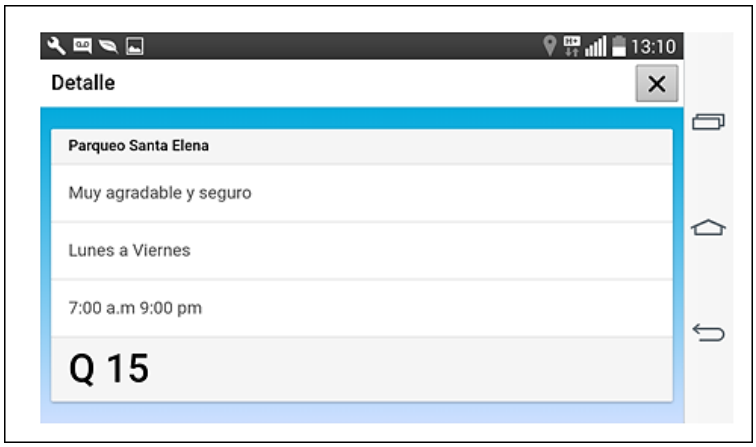
Figura 53. **Detalles de parqueo**



Fuente: elaboración propia.

Si el usuario desea visualizar información más importante como la tarifa, la capacidad y el horario debe seleccionar la opción 'detalles'; estos se mostrarán como se visualiza a continuación.

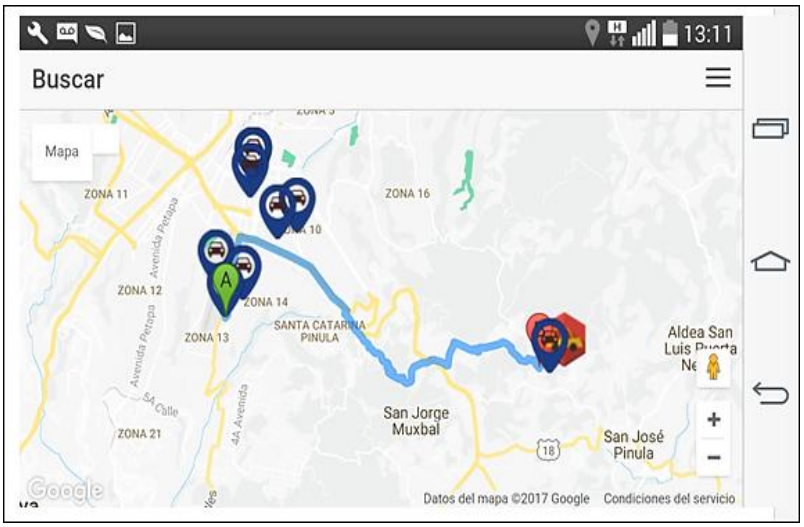
Figura 54. **Tarifa del parqueo**



Fuente: elaboración propia.

El usuario seleccionará el parqueo al cual se desea dirigir; en ese momento la aplicación le mostrará la ruta.

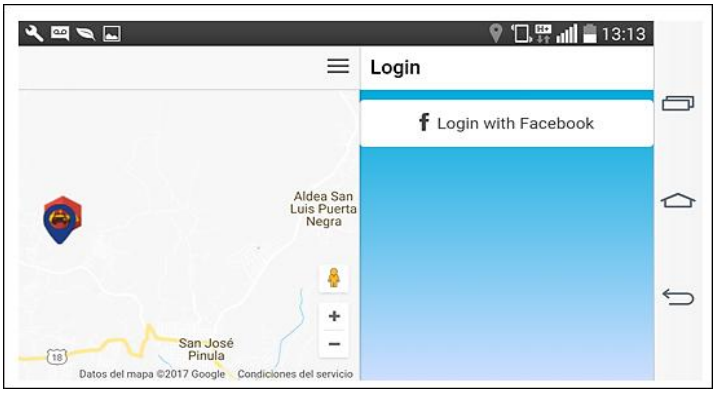
Figura 55. **Distancia del parqueo**



Fuente. elaboración propia.

El sistema ofrece al usuario otras funciones, como ingresar al mismo a través de la red social Facebook.

Figura 56. **Iniciar sesión con Facebook**



Fuente: elaboración propia.

El usuario deberá ingresar sus credenciales como se muestra a continuación:

Figura 57. **Interfaz de Facebook**

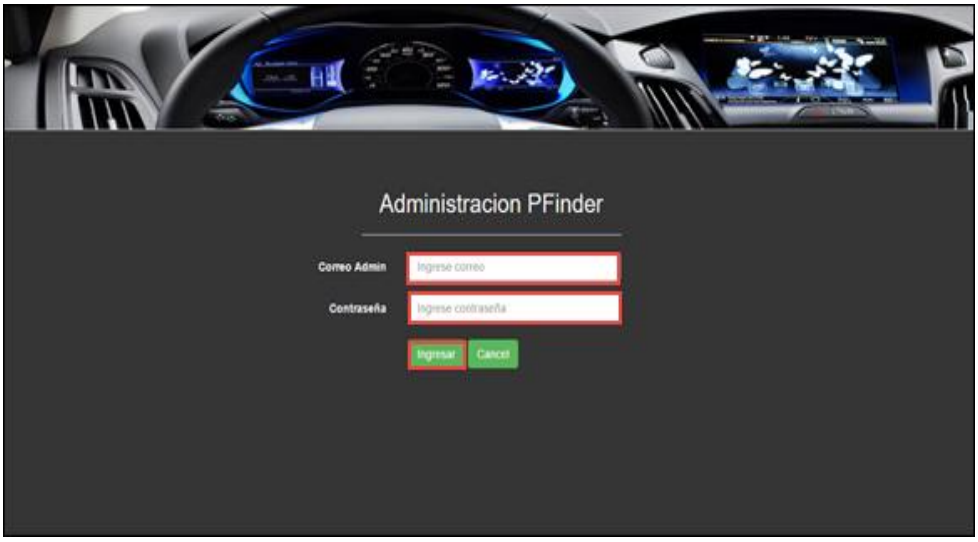
Fuente: elaboración propia.

3.2. Aplicación web administrador

El usuario administrador será el encargado de la administración de la aplicación móvil que se realiza a través de la plataforma web; es el responsable de administrar la información de parqueos en caso sea necesario.

Se puede visualizar un *login*, el usuario administrador deberá ingresar sus credenciales y presiona el botón 'Ingresar' para acceder a todas las funciones de administración.

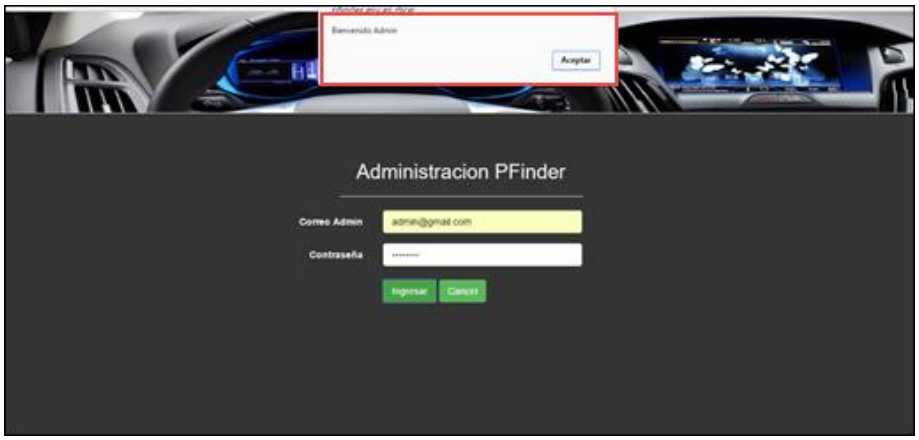
Figura 58. **Interfaz de administrador**



Fuente elaboración propia.

Si sus credenciales son correctas, se le mostrará el siguiente mensaje: 'Bienvenido Admin'.

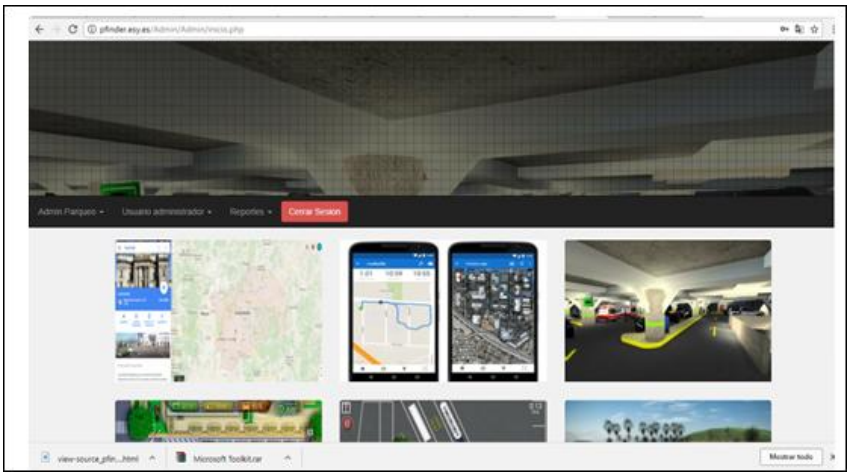
Figura 59. **Inicio de sesión exitoso**



Fuente: elaboración propia.

A continuación, se visualiza el menú principal de la página; en este se puede acceder a los menús de administración de parqueos y a los reportes de la información que se encuentra en el sistema.

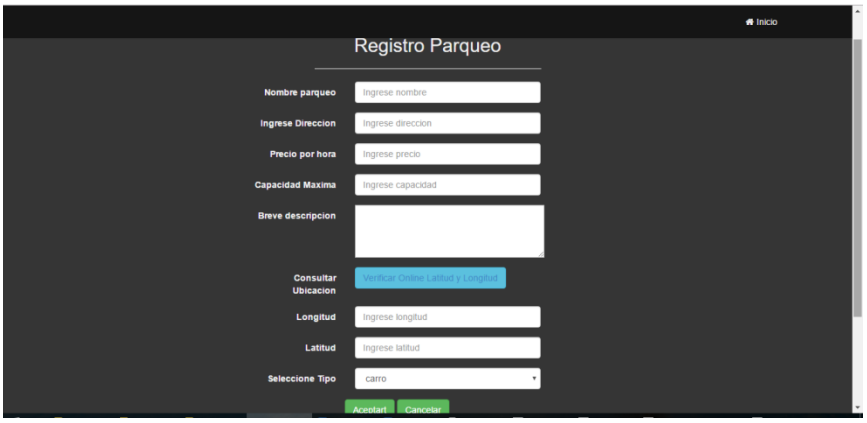
Figura 60. **Menú principal**



Fuente: elaboración propia.

El administrador puede agregar parqueos a la base de datos; para ello, debera ingresar toda la información para agregar uno nuevo.

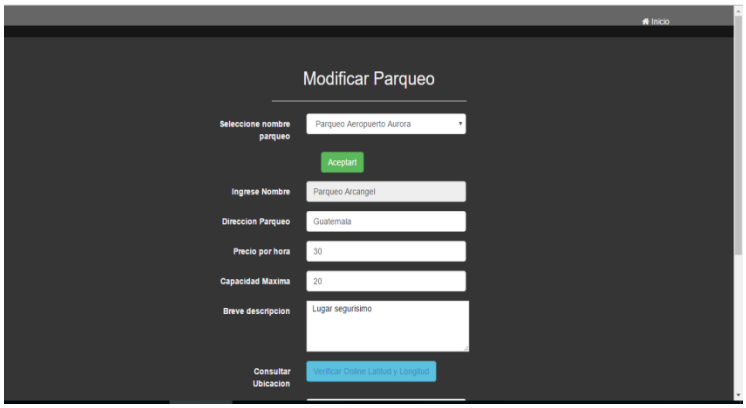
Figura 61. **Administrador: registro de parqueo**



Fuente: elaboración propia.

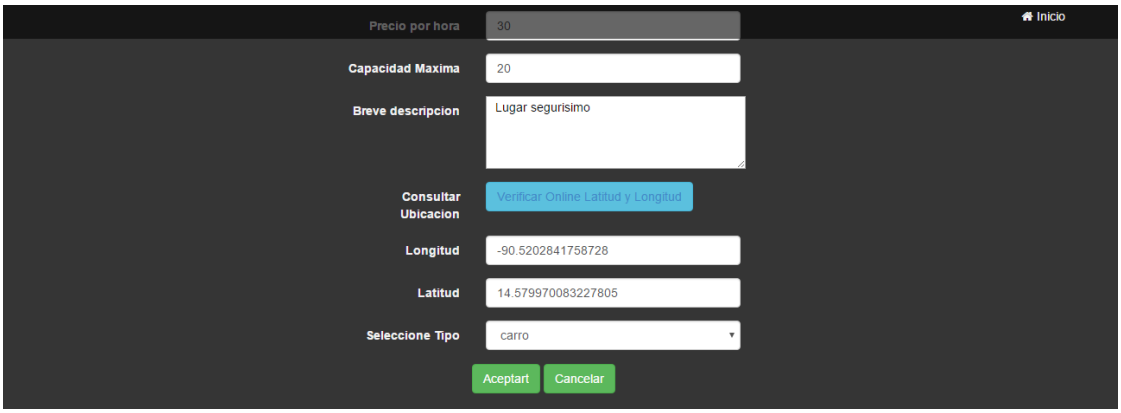
El administrador puede modificar los parqueos ya registrados seleccionando el nombre del parqueo.

Figura 62. **Administrador: modificar parqueo**



Fuente:elaboración propia.

Figura 63. **Administrador: configuración del parqueo**



Fuente:elaboración propia.

Si el administrador desea, puede eliminar parqueos que ya se encuentran registrados en el sistema.

Figura 64. **Administrador: eliminar parqueo**

Fuente: elaboración propia.

El sistema contará con una opción de reportes; a continuación, se muestra el reporte de parqueos registrados en la base de datos.

Figura 65. **Parqueos registrados**

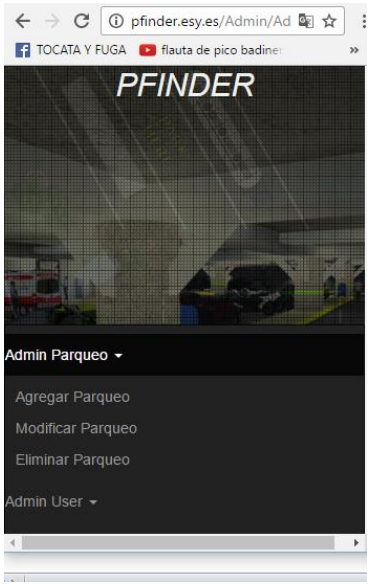
Parqueos Registrados

Nombre	Direccion	Longitud	Latitud	Descripcion	Precio	Capacidad
Parqueo los cipresales	17 Av 17-20 zona 1	-90.51016017794609	14.653583412790033	Parqueo mu	10	40
Parqueo Solares	7 av 20 calle 2-24 zona 1 Guatemala	-90.51409900188446	14.629285410734264	Frente a 100 Puertas	10	20
Parqueo Lux	14. calle C, Guatemala	-90.51186472177505	14.654681080163684	Bonito	20	40
Parqueote	Guatemala	-90.51230728626251	14.649340604221196	Bonito	20	40
Parqueo Geminis	11 calle zona 1 Guatema	-90.51888942718506	14.638981059318334	Muy seguro y ambiente agradable	10	40
Parqueo Santa Elena	6 av 16 calle 2-30 zona 10 Guatemala	-90.51060676574707	14.594713929348579	Muy agradable y seguro	15	100
Parqueo la Bendicion	15 Avenida A 18-52, Guatemala	-90.52536964416504	14.575775174801906	Muy Bonito y seguro	20	50
Parqueo Aeropuerto Aurora	zona 13 Guatemala	-90.52781581878662	14.583749585773747	Buen ambiente y seguridad	20	50

Fuente: elaboración propia.

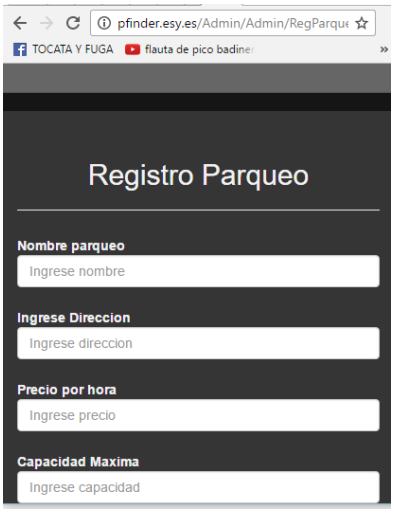
La aplicación es *responsive;* es decir, es posible utilizarla en el *smartphone;* solo es necesario el link de acceso para realizar todas estas funcionalidades.

Figura 66. ***Pfinder* web**



Fuente: elaboración propia.

Figura 67. ***Pfinder*: registro de parqueo web**



Fuente: elaboración propia.

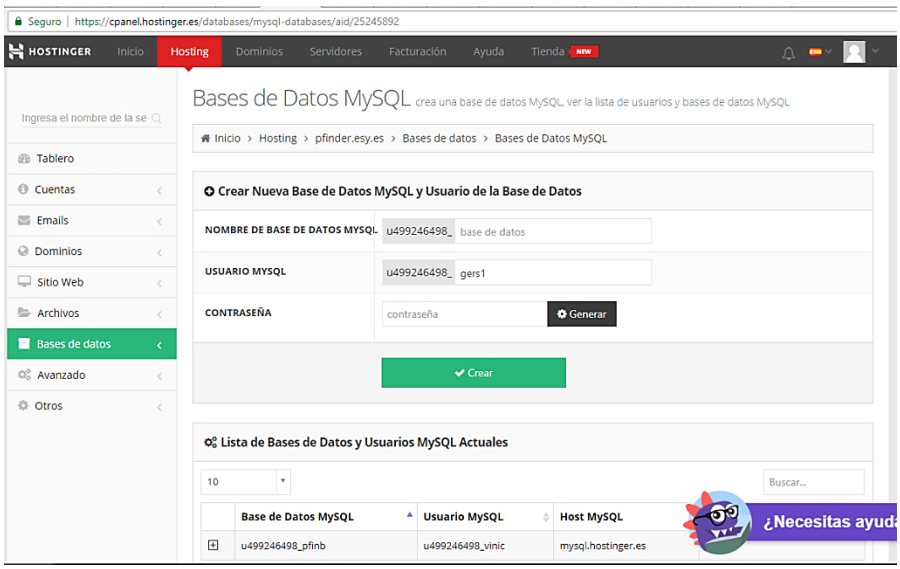
4. MANTEMIENTO Y RECURSOS

4.1. Mantenimiento de la aplicación

El mantenimiento de la aplicación e ingreso de información y administración de los parqueos se hará a través del módulo de administración del cual se describe su usabilidad en el punto 3.

La aplicación se encuentra implementada y almacenada en un *hosting* llamado *hostinge*r. El servicio gratuito proporciona un sitio web, 10 GB de espacio en disco, 100 GB de ancho de banda y un panel de control con un conjunto de herramientas las cuales son muy fáciles de utilizar.

Figura 68. **Base de datos MySQL**



Fuente: *MySQL*. https://www.hostinger.es/. Consulta: 27 de julio de 2018.

4.1.1. Cuentas de correo

La herramienta permite integrar varios administradores al sitio. Esto hace que sea más fácil su administración

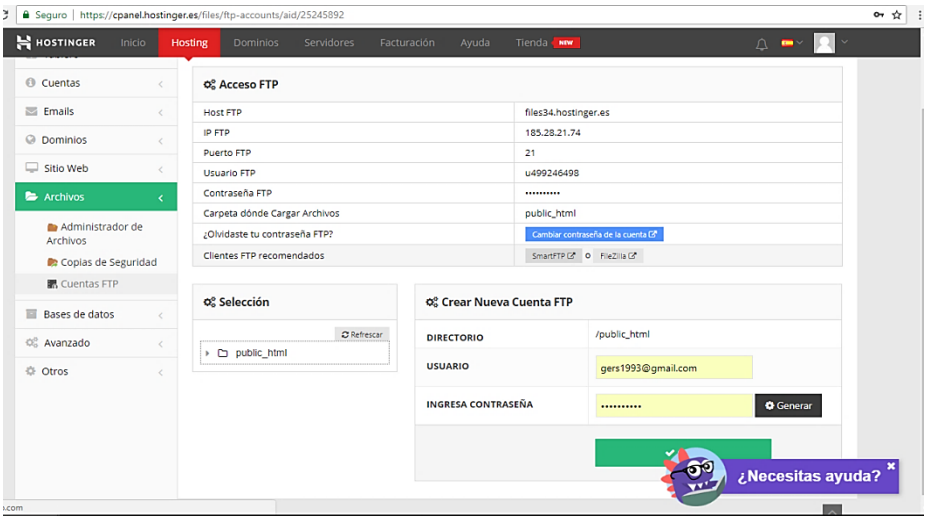
4.1.2. Administración de dominios

*Hostinge*r brinda a sus usuarios un subdominio gratuito; en este caso se implementó y se llama pfinder.esy.es. Si se desea obtener un dominio completo, es decir, una dirección web, es necesario realizar un pago de 0,99 $ mensuales. Un dominio personalizado brinda un mejor posicionamiento en las búsquedas y la web.

4.1.3. Administración archivos

La administración de los archivos es muy importante ya que el sitio web se encuentra publicado en un servidor y en caso sea necesario realizar cambios es necesario actualizar los archivos fuentes. Los servicios webs también son archivos. PHP y es muy importante su mantenimiento. El servicio de *hosting* brinda un servidor FTP gratuito con el cual se pueden realizar cambios de una manera muy sencilla. Permite realizar copias de seguridad por cualquier inconveniente.

Figura 69. ***Hostinger***

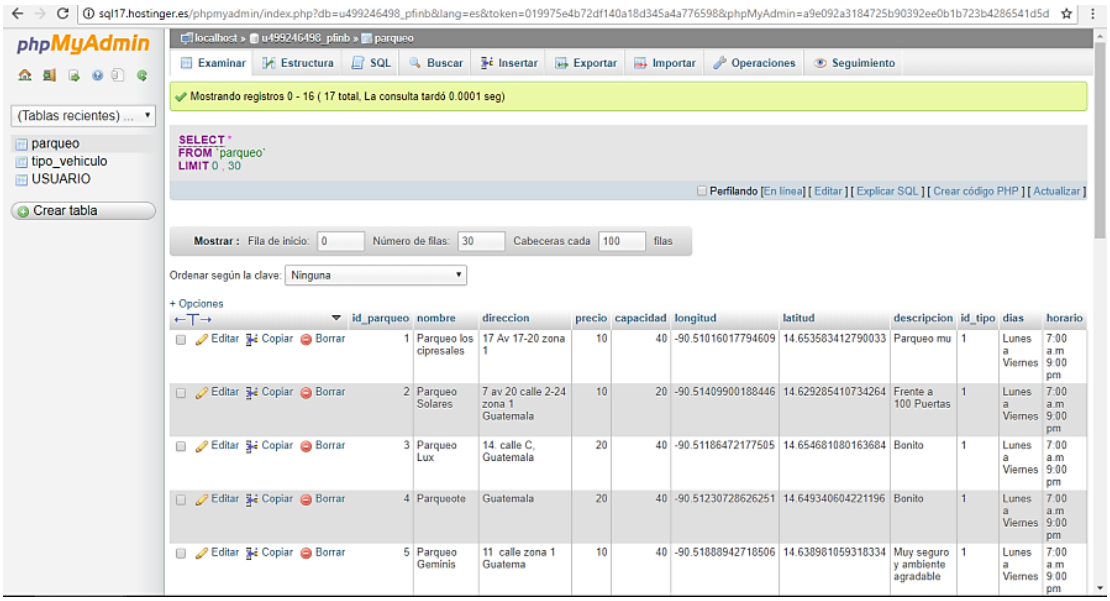


Fuente: *Hostinger*. Fuente: https://www.hostinger.es/. Consulta: 3 de junio de 2018.

4.1.4. Administración de base de datos

El mantenimiento de la base de datos es muy importante para el buen funcionamiento de la aplicación. *Hostinger* proporciona un PhpMyAdmin que es un administrador con una consola en donde se pueden escribir sentencias en SQL y es posible importar y exportar información de las tablas de información. Es posible editar directamente la información, así como realizar eliminaciones e inserciones a la base de datos.

Figura 70. ***Php Myadmin***



Fuente: *Php Myadmin*. https://www.hostinger.es/. Consulta: 17 de julio de 2018.

4.2. Recursos económicos

A continuación se describen los recursos económicos.

4.2.1. Factibilidad financiera

Es importante analizar los costos y el recurso humano necesarios para que el sistema funcione de una manera adecuada y sea posible que la solución no sea solo un prototipo sino una aplicación funcional.

Tabla XIII. **Costos económicos del proyecto**

Rubro	**Valor esperado**	**Valor real**
Costo de desarrollo e implementación (analista desarrollador)	3 meses (9 500) = Q 28 500,00	0
Costo de infraestructura	0	0
Costo de asesoría	(6 meses x 1500)=Q 9 000,00	0
Costo de servicios (luz, internet)	Servicio de luz (8 meses x 150) = Q 1 200,00 Servicio de internet (8 mesesx 250) = Q 2 000,00	Q 3 200,00
Total	Q 40 200	Q 3 200,00

Fuente: elaboración propia.

El proyecto es factiblemente rentable; las herramientas de software que se utilizaron para el desarrollo de la aplicación son de licencia gratuita. La infraestructura necesaria es accesible. El recurso humano que se utilizó para el desarrollo e implementación es el estudiante quien realizó todo el ciclo de desarrollo del software.

4.2.2. Modelo de negocio

Un plan de negocios es una herramienta que permite definir qué se ofrece al mercado y cómo se generan ingresos y beneficios a través de una solución que resuelve necesidades puntuales de las personas.

Es importante plantear un modelo de negocio para que la aplicación sea autosustentable. Se plantea utilizar un modelo de negocio por afiliación en este modelo se busca tener socios afiliados a cambio de un beneficio. En la aplicación se busca utilizar como afiliados a los dueños de parqueos. Los dueños de parqueo tendrán el beneficio de obtener publicidad gratuita y obtener clientes en sus respectivos parqueos a cambio de valor monetario literalmente

bajo. Los usuarios tendrán acceso gratuito a la aplicación y podrán dirigirse a todos los parqueos afiliados.

4.2.3. Costo de mantenimiento

A continuación, se presenta el costo de mantenimiento correctivo.

4.2.3.1. Costo de mantenimiento correctivo

El costo de mantenimiento correctivo se refiere al plan enfocado en solución de bugs y errores en la aplicación móvil. Corregir errores en la aplicación web y servicios. Mejora de funcionalidades y actualización de versiones. Actualización de librerías y cualquier mejora a nivel técnico. El mantenimiento de *hosting* con todas sus funcionalidades podría entrar en el mantenimiento, Para el tipo de mantenimiento correctivo es necesario contar con un analista desarrollador lo que implica un costo de aproximadamente Q 1 500,00 quetzales mensuales asumiendo 1,5 hora diarias de trabajo de lunes a viernes. El mantenimiento se podría obviar hasta que la aplicación sea autosustentable.

4.2.3.2. Costo de mantenimiento operativo

La publicación de la aplicación en *Android* se realiza a través de Google Play y tiene un costo de 25 $ anual. Para publicar la aplicación en la plataforma de IOS es necesario pagar una membresía anual de 99 $ la cual es renovable.

La aplicación actualmente funciona con un *hosting* gratuito, pero con el tiempo el crecimiento de la información y de los usuarios hace necesario

realizar un pago mensual de *hosting* el cual tiene un costo de 0,99 euros mensuales.

Figura 71. **Cuenta premium**

Fuente: *Cuenta premium*. https://www.hostinger.es/hosting-web. Consulta: 19 de julio de 2018.

Es necesario contar con un presupuesto mensual de gastos, se incluyen costos correctivos y de operación los cuales son necesarios para el correcto funcionamiento de la aplicación.

Tabla XIV. **Costos mensuales de mantenimiento**

Rubro	Valor esperado	Valor Real
Costo de mantenimiento correctivo	Analista Desarrollador Q 1 500,00 (1,5 horas diarias) = Q 1 500,00	0
Costo de infraestructura	*Hosting* Q 9,90	0
Costo tiendas online	Google Play (Q 190,00 / 12) = Q16,00 IOS (Q 750,00 / 12) = Q 62,50	78,50
Total	Q 1 588,40	Q 78,50

Fuente: elaboración propia.

CONCLUSIONES

1. Al implementar la aplicación móvil *Pfinder* se facilita la búsqueda de parqueos para los guatemaltecos en la ciudad de Guatemala. Se puede afirmar que la cantidad de automóviles aumentará en la ciudad de Guatemala. En el futuro será muy necesario utilizar aplicaciones para la búsqueda de parqueos.

2. Es importante usar la tecnología en todos los ámbitos posibles. El uso de aplicaciones móviles se ha convertido indispensable para la población. Las aplicaciones permiten explotar todas las funciones que proporcionan los *smartphones* y solucionar problemas de una manera eficiente.

3. *Pfinder* es una aplicación que sirve como herramienta de búsqueda de parqueo dentro de la ciudad de Guatemala permite conocer con anticipación ubicaciones, precios y horarios de los parqueos cercanos a la ubicación de los personas sin tener que esperar a llegar.

4. La aplicación *Pfinder* se desarrolló para dispositivos con sistema operativo *Android* e IOS. Es importante elegir una plataforma adecuada para el desarrollo de aplicaciones móviles. Se decidió utilizar el *framework* de *Ionic* debido a sus características como alto rendimiento, reconocimiento táctil, facilidad de crear aplicaciones móviles en múltiples plataformas realizando un único proceso de desarrollo e implementación, proporciona un fácil mantenimiento al utilizar tecnologías web. Además, tiene un diseño funcional y limpio.

RECOMENDACIONES

1. Se recomienda desarrollar e implementar una aplicación híbrida en casos en los que se tengan pocos recursos y un presupuesto limitado. El desarrollo de las aplicaciones híbridas se puede realizar con software gratuito y equipos con hardware con un precio accesible. Es necesario tomar en cuenta aspectos como la necesidad de actualización y complejidad de la aplicación. Las aplicaciones híbridas se caracterizan por proporcionar facilidad de actualización y brindan herramientas para acceder a características del hardware de los dispositivos lo que permite desarrollar aplicaciones sencillas y complejas

2. *Pfinder* es un tipo de aplicación híbrida; se decidió construir una aplicación de este tipo debido a las características propias de la aplicación como transaccionalidad, diseño visual, acceso a hardware, presupuesto. *Pfinder* no necesita características tan específicas a nivel de hardware únicamente el GPS lo que hace que su construcción con *Ionic* sea alcanzable. El diseño visual es sencillo y no tiene complejidad en sus pantallas. Utilizar un *framework* como *Ionic* con aplicaciones con estas características es una ventaja y su coste de utilización es gratuito.

3. Se recomienda mantener actualizada la aplicación en distintos aspectos: corrección de bugs o errores, versionamiento e información. La aplicación debe tener información de parqueos actualizada para brindar información confiable a los usuarios.

BIBLIOGRAFÍA

1. *Aplicación Parking App*. [En línea]. <http://www.soy502.com/articulo/encontrar-parqueo-tu-celular-colombia-ya-posible>. [Consulta: 25 de febrero de 2017].

2. *Compatibilidad navegadores bootstralp*. [En línea]. <http://librosweb.es/libro/bootstrap_3/capitulo_1/compatibilidad_con_los_navegadores.htm>. [Consulta: 8 de abril de 2018].

3. *Cordoba Apache*. [En línea]. <https://cordova.apache.org>. [Consulta: 22 de enero de 2018].

4. *Estructura de carpetas Ionic.* [En línea]. <http://mialtoweb.es/estructura-de-un-proyecto-con-ionic>. [Consulta: 31 de julio de 2018].

5. *Hardware. Dispositivos móviles*. [En línea]. <http://datateca.unad.edu.co/contenidos/233016/EXE_SAM/leccin_4_ha rdware_de_dispositivos_mviles.html>. [Consulta: 16 de junio de 2017].

6. *Metodología de investigación aplicada.* [En línea]. <https://metinvestigacion.wordpress.com>. [Consulta: 24 de julio de 2017].

7. NIELSE, Jacob. *Usabilidad en dispositivos móviles (diseño y creatividad).* México: McGraw-Hill, 2013. 75 p.

8. *Problema parqueo ciudad Guatemala.* [En línea]. <http://lahora.gt/hemeroteca-lh/idonde-parquearse-una-pregunta-que-atormenta-al-conductor-capitalino>. [Consulta: 9 de mayo de 2017].

9. *Tipos de dispositivos móviles*. [En línea]. <http://leo.ugr.es/J2ME/INTRO/intro_4.htm>. [Consulta: 16 de febrero de 2018].

10. *Tus apps saben dónde estás y envían tu localización a servidores.* [En línea]. <http://www.xataka.com/aplicaciones/tus-apps-saben-donde-estas-y-envian-tu-localizacion-a-sus-servidores-cada-pocos-minutos>. [Consulta: 7 de abril de 2017].

11. *Tutorial de desarrollo IONIC*. [En línea]. <https://ccoenraets.github.io/ionic-tutorial/>. [Consulta: 15 de julio de 2017].