Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación Android: Red social de búsqueda de aparcamiento

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Miguel Company Soler

Tutores: Pedro Valderas Aranda
Victoria Torres Bosch

2014 - 2015

Resumen

El presente Trabajo Fin de Grado consiste en la creación de una aplicación móvil desarrollada para sistemas operativos Android. Se trata de una red social de búsqueda de aparcamiento para coches, que utiliza un sistema de puntuaciones basado en el buen o mal comportamiento de los usuarios. Estos pueden promocionar sus puntuaciones mediante el buen uso de la aplicación, y así obtener una visualización completa de aparcamientos libres cerca de su destino. En cambio, los usuarios que no tengan suficiente puntuación por mal uso de la aplicación no van a tener una visualización completa.

El proceso de desarrollo de la aplicación se ha llevado a cabo en dos fases bien diferenciadas. Un primer proceso de diseño centrado en el usuario, atendiendo sobretodo al contexto de uso, y posteriormente, se ha realizado la implementación y desarrollo de la aplicación utilizando tecnologías como Java, XML, JSON y PHP, además de los Servicios Web de Google Maps necesarios tanto para la geolocalización de los usuarios como de los aparcamientos.

Por último, la aplicación se sirve de una base de datos MySQL para el almacenamiento de los datos de los usuarios y de las localizaciones de aparcamientos libres, entre otros.

Palabras clave: Aplicación móvil, geolocalización, red social, diseño centrado en el usuario.

Abstract

This Final Project consists on the creation of a mobile application developed for Android operating systems. This is a social network of parking search for cars that uses a scoring system based on the good or bad behavior of users. These can promote their scores through good use of the application, and get a complete view of free parkings close to their destination. However, users who do not have enough scoring because of misuse of the application will not have a complete view.

The development process of the application has been accomplished in two phases. A first process of user-centered design, especially focusing on the context of use, and subsequently it is been accomplished the implementation of the application using technologies such as Java, XML, JSON and PHP, furthermore the Web Services of Google Maps needed to the geolocation of users and car parkings.

Finally, the application uses a MySQL database for storing user data and locations of free parkings, among others.

Keywords: Mobile app, geolocation, social network, user-centered design.

Tabla de contenidos

1. Introducción

1.1. Antecedentes.

La realización del presente Trabajo Final de Grado (TFG) me permite finalizar los estudios del Grado en Ingeniería Informática y con ello poder alcanzar el objetivo de adquirir los conocimientos de base necesarios para desarrollar una carrera profesional en el mundo de las tecnologías de la información, que tanta ilusión y perspectivas de futuro despierta actualmente en mí.

Anteriormente, estudié la carrera de Ingeniero Técnico en Topografía también en esta Universidad, donde adquirí habilidades y conocimientos en el área de la cartografía, la topografía, el diseño asistido por ordenador y en las tecnologías de posicionamiento GPS, entre otras muchas. Debido a la confusa perspectiva de desarrollo profesional de dicha carrera durante estos últimos años, decidí continuar formándome en las nuevas tecnologías y cumplir además uno de mis sueños, cursar una carrera de Informática.

Por la experiencia académica y profesional que he ido adquiriendo a lo largo de este tiempo decidí tomar como idea base para la realización del TFG algún proyecto que uniera la informática con el geoposicionamiento, que tan de moda se encuentra actualmente. Es por ello que me puse en contacto con Pedro Valderas Aranda (tutor del TFG), a quién le trasladé la idea de realizar una aplicación móvil con componente de geolocalización. De esta manera, Pedro me puso en contacto con Victoria Torres Bosch (cotutora del TFG), quién tenía la idea de desarrollar una red social para la búsqueda de aparcamiento. Así que entre ambos me ofrecieron la posibilidad de desarrollar una aplicación de este tipo.

Al cursar la rama de Tecnologías de la Información en el Grado, no me llevo mucho tiempo asimilar que el proyecto propuesto se adecuaba de forma precisa a las exigencias tecnológicas y de conocimientos adquiridos que iban a ser necesarios para el desarrollo del mismo, por lo que no tuve ninguna duda en aceptar el reto.

Durante estos años estudiando el Grado he cursado asignaturas como: “Diseño Centrado en el Usuario” e “Interfaces Persona Computador”, que me han ayudado a conocer la usabilidad desde un punto de vista histórico y entenderla desde el punto de vista de las metodologías que se emplean en la actualidad para realizar el diseño de interfaces usables.

Además, también he cursado asignaturas que me han ayudado a realizar el desarrollo e implementación de la aplicación. Algunas de estas asignaturas son: “Programación”, “Estructuras de datos y algoritmos”, “Concurrencia y sistemas distribuidos”, “Ingeniería del Software”, “Integración de aplicaciones”, “Bases de datos y sistemas de información”, etc., que me han ofrecido los conocimientos de base necesarios para llevar a cabo un proyecto de forma autónoma, como el que se describe en la presenta memoria.

Si a estas asignaturas se suman los conocimientos adquiridos en Cartografía y posicionamiento GPS de los que previamente he hablado, tenemos el punto de partida idóneo para desarrollar el TFG.

1.2. Objetivo.

El principal objetivo es realizar un trabajo desde el inicio del proceso de desarrollo de una aplicación móvil, poniendo especial atención en el diseño, la usabilidad y la funcionalidad acorde con lo que se pide. Por lo que es importante destacar que el desarrollo de este proyecto, fundamentalmente, se divide en dos fases bien distintas: Por un lado, el diseño de la interfaz de usuario, y por otro lado, la implementación de la aplicación móvil.

La usabilidad tiene gran importancia en el manejo de dispositivos móviles, sobretodo en ciertos contextos de uso. Así que en este desarrollo tiene una importancia suprema la usabilidad debido a que el usuario va a utilizar la aplicación desde dentro de un coche, ya sea como conductor o como acompañante, por lo que se debe prestar especial atención en realizar un diseño usable, de fácil manejo y rapidez en las interacciones entre el usuario y la pantalla.

Es por ello que se va a llevar a cabo un diseño centrado en el usuario, y más concretamente, utilizando la técnica *Personas* junto con la definición de unos escenarios de uso concretos.

La técnica *Personas* consiste en crear usuarios ficticios que representen a una parte de la población y estudiar sus cualidades, características personales y necesidades. Para este proyecto, el tutor me indicó como requisito, enfocarlo a un público en particular. En concreto, se toma como modelo una persona que vive en el centro de Valencia, que trabaja en las afueras de la ciudad (para lo cual se desplaza en coche), que tiene conocimientos escasos de informática y que es de mediana edad (unos 35 años). Por lo que no ha sido necesario realizar un estudio previo cualitativo para definir a qué audiencia va a ir enfocada la aplicación.

Los escenarios de uso se especificarán más adelante cuando se expliquen la metodología *Personas* y el proceso de diseño de prototipos. Además se va a realizar una evaluación heurística de las interfaces diseñadas mediante guías de usabilidad.

Por otra parte, en cuanto a la implementación de la aplicación, de manera personal y en consenso con los tutores, se ha decidido realizar el desarrollo de la aplicación para sistemas operativos Android. En gran parte, esta decisión se debe a que en el último año he hecho cursos de programación en este tipo de sistemas y tengo un gran interés en seguir aprendiendo.

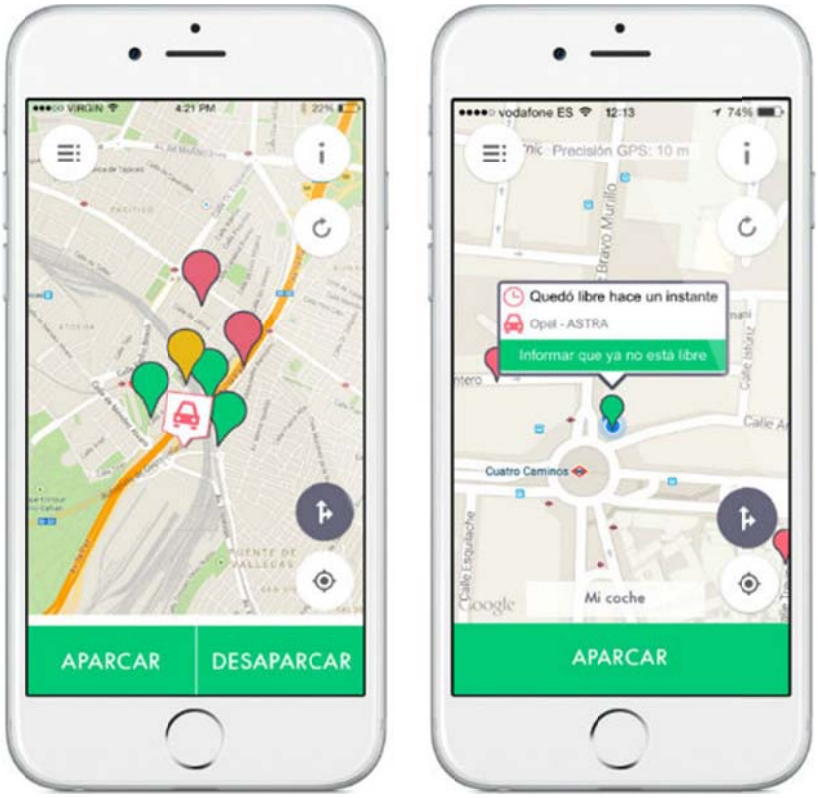
2. Estado del arte

Existen en el mercado diferentes aplicaciones móviles que ofrecen servicios de búsqueda de aparcamiento. No todas tienen el mismo objetivo, algunas se centran en parkings privados y otras, más similares al objetivo de este proyecto, basan su idea en ayudar a encontrar plazas de aparcamiento libre en la calle. Además existen algunas que combinan ambos objetivos, e incluso, pueden llegar a incluir la reserva de plazas de aparcamiento en servicios de pago como el estacionamiento regulado.

Entre todas estas aplicaciones existentes actualmente, se han elegido las dos que se asemejan más al objetivo que se persigue con el presente proyecto. Destacando que se va a realizar una comparativa poniendo el énfasis en diferenciar estas aplicaciones ya existentes con la aplicación desarrollada.

2.1. *Wazypark*.

Esta aplicación de origen español se puede descargar gratuitamente tanto para sistemas operativos Android como iOS, y cuenta en la actualidad con más de 110.000 usuarios registrados (fuente: *www.wazypark.es*). Su objetivo principal es que los usuarios estén relacionados mediante lo que se conoce como '*crowd parking*', es decir, una comunidad donde los usuarios dejan huecos libres o los ocupan, emitiendo avisos a través de la aplicación y recibiendo puntos por cada acción que realizan.



Si bien es una aplicación móvil reciente, está en continuo desarrollo y mejorando su modelo de negocio día a día. Entre los aspectos más interesantes de la aplicación destacan:

- Ofrecer cambio de puntos por dinero para gastarlo en gasolineras, restaurantes, taxis, talleres, etc.
- Como todos los usuarios registrados están obligados a introducir la matrícula del coche, cualquier usuario puede alertar al resto de situaciones como: un robo, un policía poniendo una multa, quitar un coche aparcado en doble fila porque está molestando para salir a otro, etc.
- Entre las últimas novedades destaca *Bluetooth Parking*. Con este sistema se puede vincular el dispositivo móvil con el bluetooth del coche, de esta manera la aplicación sabrá cuando arranca el coche del usuario, compartiendo automáticamente el sitio que deja libre y ahorrando al usuario los pasos de tener que encender el móvil, abrir la aplicación y pulsar en la acción correspondiente.

Como adelanto de las funcionalidades de la aplicación desarrollada en el presente proyecto, y realizando una comparativa con *Wazypark*, se describen una serie de diferencias que saltan a la vista. Cabe destacar por supuesto, la naturaleza académica de la aplicación desarrollada en comparación con *Wazypark*, que tiene un recorrido y un fin comercial muy claros.

- *Wazypark* te indica mediante colores el tiempo transcurrido desde que el aparcamiento se quedó libre o incluso cuándo quedará libre. La aplicación desarrollada muestra la probabilidad de encontrar un aparcamiento libre, teniendo en cuenta no solo el tiempo que ha transcurrido desde que se quedó libre, sino también el tamaño del vehículo que va a aparcar, dándole mayores probabilidades a coche más pequeños.

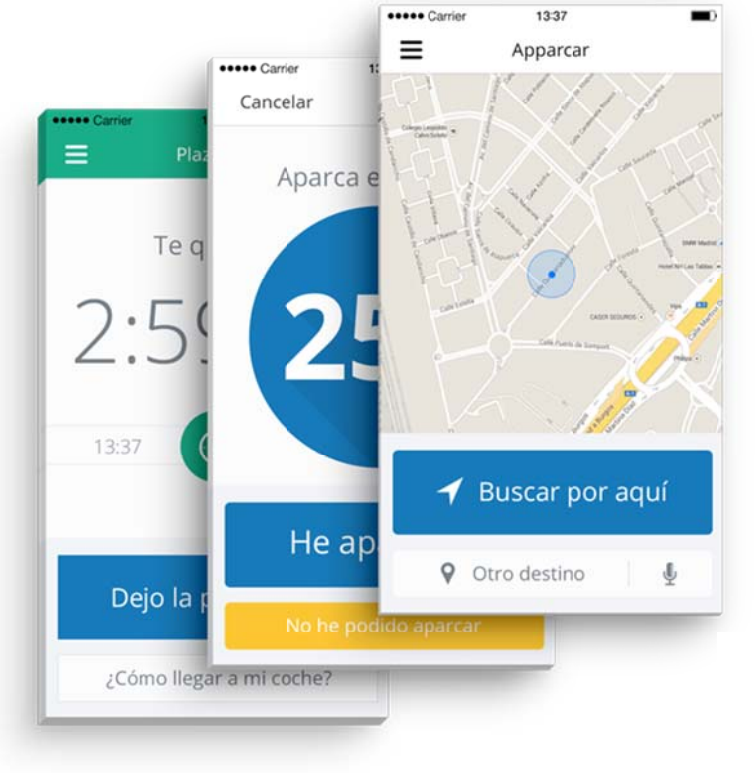
- En la aplicación desarrollada no es necesario publicar datos del vehículo como la matrícula, ya que el objetivo básico de la aplicación es funcionar únicamente como una red social donde los usuarios intercambian tan solo información sobre las plazas de aparcamiento que van dejando libres y las que van ocupando.

- Desde el punto de vista de la usabilidad, la aplicación desarrollada muestra mensajes informativos al usuario continuamente por cada acción que realiza, lo que favorece que la curva de aprendizaje sea rápida.

- *Wazypark* está orientado a ofrecer puntos para canjear por dinero, seguramente debido a que busca un fin comercial, mientras que la aplicación desarrollada tiene un sistema de puntuaciones orientado a ofrecer privilegios en la búsqueda de aparcamiento, y en caso de que el usuario no participe correctamente, quitarle esos privilegios para restringir la vista de los aparcamientos libres que estén mejor situados.

2.2. *Apparcar*.

Esta aplicación móvil también de origen español, se puede descargar libremente tanto para sistemas operativos Android como para iOS. Dispone de una funcionalidad similar al resto, pero además cuenta con una línea de negocio interesante, orientada a la venta a empresas y ayuntamientos de un servicio de gestión de plazas de aparcamiento regulado, pudiendo realizar el pago desde el propio dispositivo móvil sin tener que acudir en busca del parquímetro, con la comodidad que ello conlleva. Incluso se puede reservar una plaza de aparcamiento mientras el usuario se está dirigiendo hacia un destino (fuente: *http://www.apparcar.com*).



Entre las características más interesantes de la aplicación *Apparcar* destacan:

- No dispone de un sistema de puntuaciones. Su modelo de negocio se basa en explotar el pago del aparcamiento regulado a través de los dispositivos móviles, añadiendo una gestión más eficiente para empresas y ayuntamientos de manera remota, con el consiguiente ahorro de costes y control de las zonas de aparcamiento.
- Ofrece una funcionalidad añadida en el hecho de poder reservar una plaza de aparcamiento.
- El usuario puede consultar como llegar al lugar donde aparcó su coche, muy útil para cuando el usuario no conoce la ciudad o la zona donde se encuentra.
- Interesante también es que el usuario puede informar al sistema cuando no ha podido aparcar en una plaza, bien porque ya estaba ocupada cuando llegó, bien porque el usuario decide finalmente cambiar de destino, etc.

De la misma manera, se hace una comparativa entre las funcionalidades de *Apparcar* y la aplicación desarrollada. Destaca también en este caso, el mayor recorrido en cuanto al desarrollo y con fines comerciales que tiene *Apparcar* a diferencia de la aplicación que se ha desarrollado, que tiene un carácter meramente académico. Las principales diferencias son:

- El sistema de puntuaciones de la aplicación desarrollada busca una fidelización de los usuarios mediante la promoción de puntos a través de los cuales ofrezcan visualizaciones más ventajosas de aparcamientos libres sobre el mapa, mientras que Apparcar carece de un sistema de este tipo. Su manera de fidelizar a los clientes es simplemente por la necesidad que tienen estos de utilizar la aplicación para poder pagar el tiempo que permanezca su coche en zonas de aparcamiento regulado.
- *Apparcar* ofrece la posibilidad de reservar plazas, haciendo que el resto de usuarios no puedan visualizarlas sobre el mapa. Esta funcionalidad de reserva de plazas de aparcamiento no está presente en la aplicación desarrollada puesto que está basada en compartir la ubicación de las plazas de aparcamiento que se dejan libres y la visualización de las probabilidades de poder encontrar esas plazas de aparcamiento sin ocupar, de la otra forma, los usuarios reservarían las plazas de aparcamiento sin dejar oportunidad al resto de usuarios a ocuparlas libremente.

3. Contexto tecnológico

En este apartado se van a describir los distintos lenguajes, tecnologías y entornos de programación que se han utilizado para el desarrollo de la aplicación, sin entrar en detalles de implementación. Estos detalles se abordarán posteriormente en otro apartado.

3.1. Entornos de desarrollo.

Se mencionan tanto el sistema operativo como los entornos de programación utilizados para abordar el TFG.

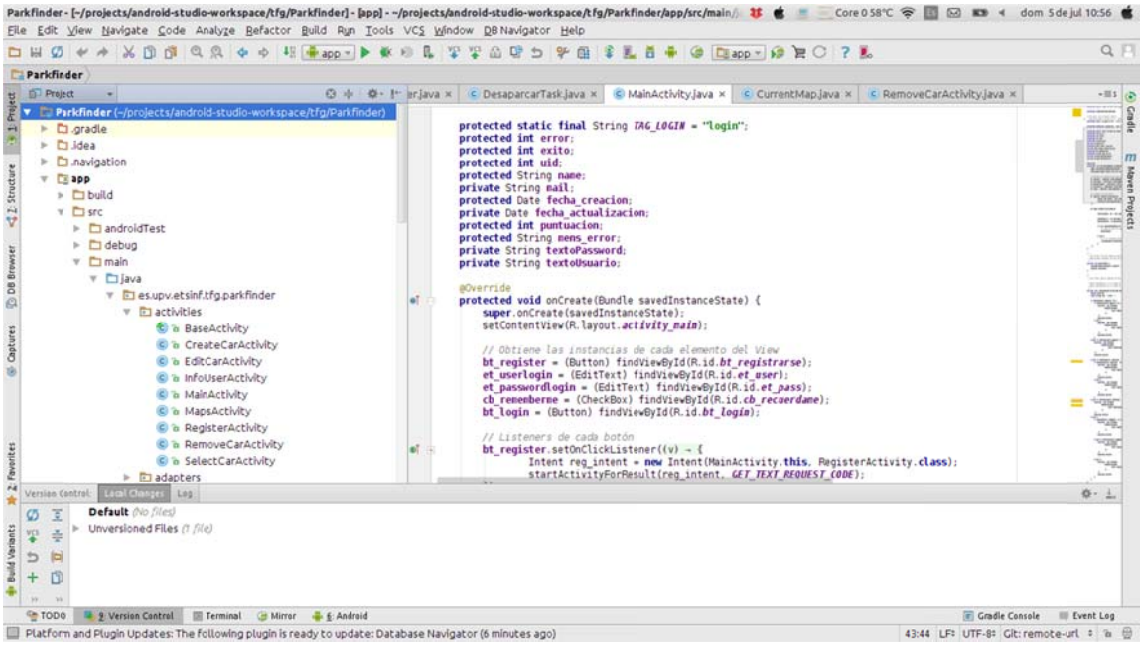
3.1.1. Ubuntu.

Sistema operativo basado en GNU/Linux distribuido bajo licencia de software libre y código abierto. Es ideal para desarrollo web y aplicaciones móviles como Android por su velocidad, seguridad, robustez, estabilidad y flexibilidad. Para este desarrollo se ha utilizado la versión 14.04 LTS.

3.1.2. Android Studio.

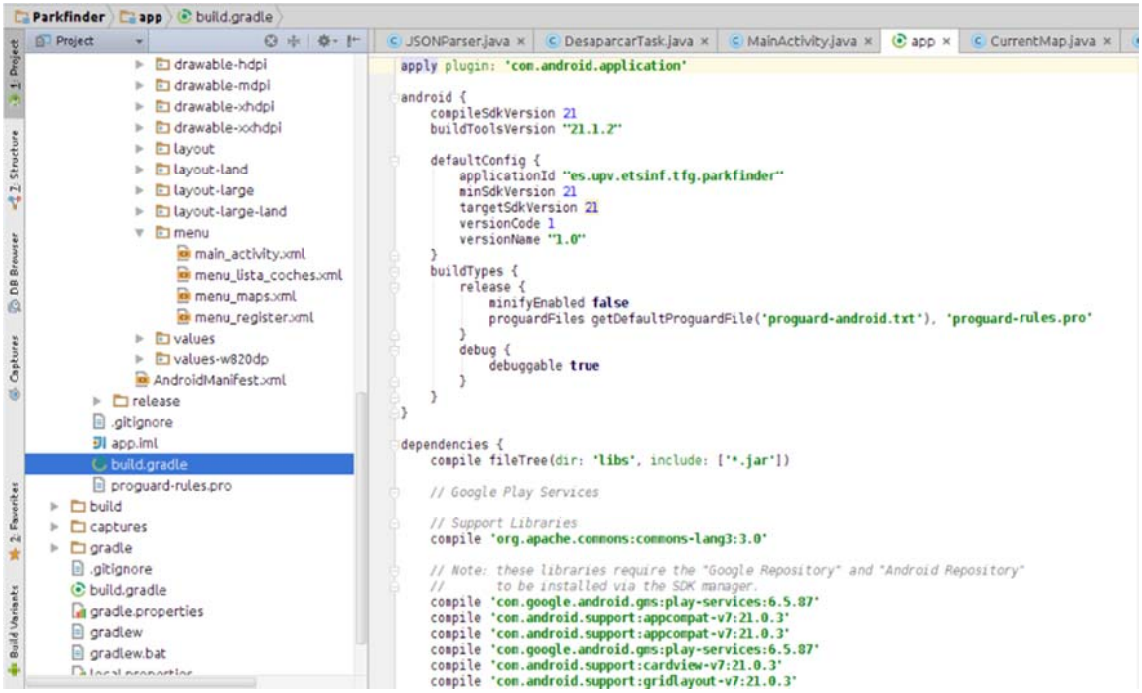
Entorno de desarrollo (IDE) para la plataforma Android. Está basado en *IntelliJ IDEA*, otro entorno de desarrollo que fue creado por *JetBrains*. Entre sus características más importantes se tienen:

- Renderizado de vistas en tiempo real.
- Consola de desarrollador con ayudas para traducción y consejos de optimización.
- Soporte para construcción basada en *Gradle*.
- Refactorización específica de Android.
- Plantillas específicas para crear diseños Android.



3.1.2.1. *Gradle*.

Herramienta de construcción multiplataforma y basada en *Apache Ant* y *Apache Maven*. Introduce un lenguaje basado en *Groovy* para declarar las dependencias del proyecto en vez del tradicional XML.



Ejemplo de un fichero build.gradle

3.2. Java.

Lenguaje de programación de propósito general, orientado a objetos y concurrente, entre otros. Las aplicaciones generadas con Java son compiladas previamente y pueden ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura subyacente del dispositivo. Actualmente, el lenguaje Java es propiedad de Oracle.

3.3. Android.

Sistema operativo basado en el núcleo Linux y diseñado especialmente para dispositivos móviles como *Smartphones* o *Tablets*. Actualmente se está potenciando su desarrollo para relojes inteligentes, televisores y automóviles. Está soportado por Google desde el año 2005.

3.3.1. Dalvik.

Máquina virtual distribuida como software libre y creada por Google. La incorporan los dispositivos móviles Android y se diferencia en ciertos aspectos a las máquinas virtuales de Java, como por ejemplo: Necesita poca memoria, está optimizada para ejecutar varias instancias simultáneas en la propia máquina virtual y está basada en registros.

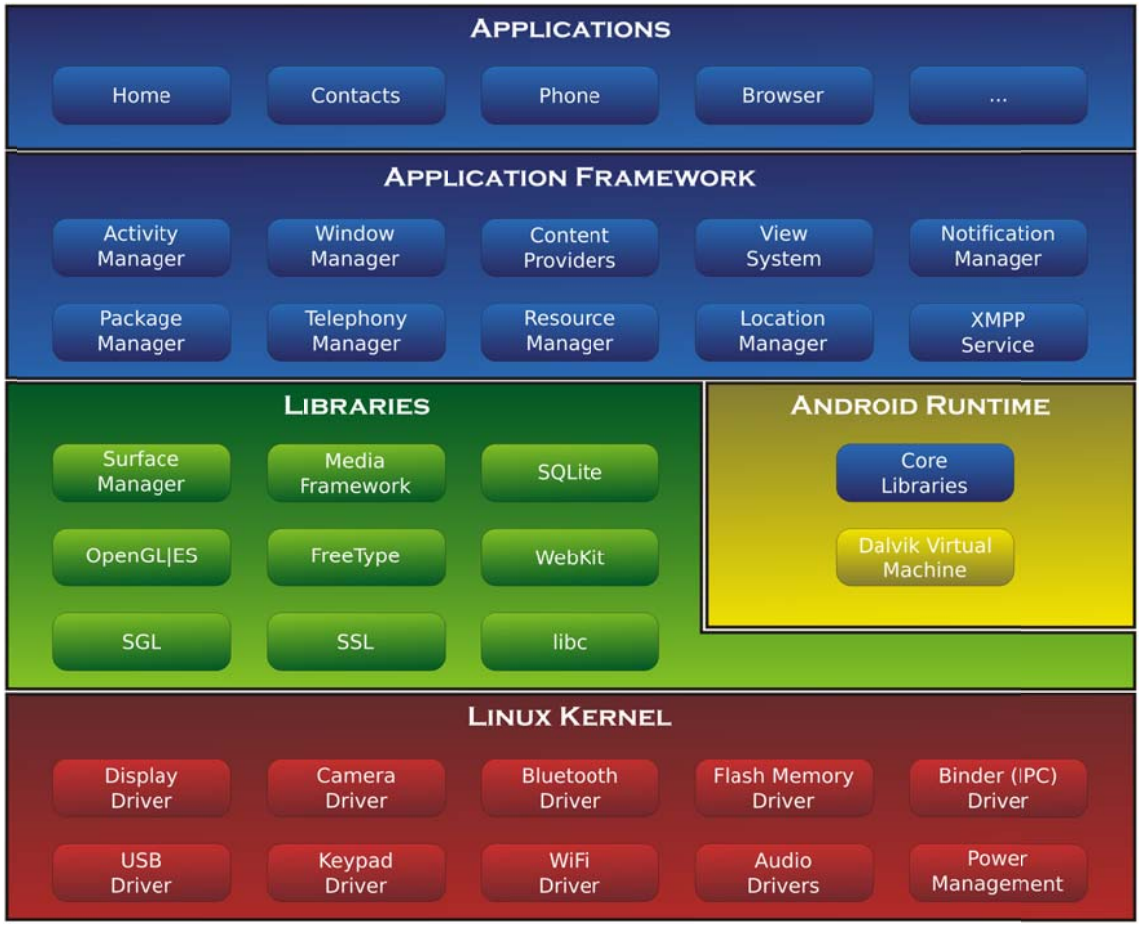


Ilustración de los componentes que forman un sistema Android.

3.3.2. XML.

Es un lenguaje de marcas (*eXtensible Markup Language*) utilizado para almacenar datos de forma legible por las personas. Android lo utiliza básicamente para definir las interfaces gráficas de usuario.

3.3.3. Servicios Web de Google Maps.

Se pueden añadir mapas a una aplicación móvil haciendo uso de la API de Google Maps. Gracias a este servicio, automáticamente se tiene el control sobre el acceso a los servidores, descargar datos, mostrar mapas y capturar eventos que se producen sobre los mapas mediante gestos. Además se pueden añadir iconos (*markers*) y polígonos, que añaden interactividad del usuario con el mapa.

3.4. PHP.

Lenguaje de script de código abierto muy extendido para su uso como lenguaje *back-end* en el lado del servidor.

```php
/**
** Elimina el aparcamiento que ocupa el usuario
*/
public function eliminarAparcamiento($lat,$lon) {

    // Selecciona la fila más recientemente creada de un aparcamiento que se encuentre a menos de 5 metros
    $sql = "SELECT *,
    ( 6371 * ( acos( cos( radians( `latitud` ) ) * cos( radians($lat) ) * cos( radians( `longitud` ) - radians($lon) )
    + sin( radians( `latitud` ) ) * sin( radians($lat) ) ) ) )
    AS `distancia`
    FROM Aparcamiento
    HAVING `distancia` < 0.010
    ORDER BY `distancia` ASC LIMIT 0,1";

    $query = mysqli_query($this->conn,$sql);
    $creador = 0;
    $checkBorrado = false;

    if ($query) {

        $i = 0;
        while ($fila = mysqli_fetch_array($query)) {
            $aid = $fila['aid'];
            $creador = $fila['creador'];

            // Borra el aparcamiento actual
            $borrado = mysqli_query($this->conn,"DELETE FROM Aparcamiento WHERE aid = '$aid'");

            if ($borrado) {
                $checkBorrado = true;
            }

            if ($checkBorrado) {
                return $fila;
            }

            $i++;
        }

    }
```

Ejemplo de función en lenguaje PHP.

3.4.1. JSON.

Es un formato de datos ligero utilizado para el intercambio de información entre aplicaciones o sistemas. Su simplicidad de uso va ligada al empleo de un analizador sintáctico (*parser*) de JSON.

3.5. MySQL.

Sistema Gestor de Bases de Datos relacional, multihilo y multiusuario. Se trata de un software libre propiedad de Oracle. Su popularidad va muy ligada a PHP y frecuentemente aparecen ambos en combinación. Como base de datos destaca su rapidez de lectura, sobretodo en entornos de baja concurrencia en la modificación de datos.

3.5.1. SQL.

Lenguaje de consulta estructurado y declarativo, de acceso a bases de datos relacionales. Permite realizar gran variedad de consultas más o menos complejas y cálculos sobre campos de una o varias tablas, cuyos resultados pueden ser añadidos como un nuevo campo.

Dentro de SQL se caracteriza por ofrecer un el lenguaje de definición de datos (*DDL*) y un lenguaje de manipulación de datos (*DML*). Con el primero se definen los esquemas y sus modificaciones, y con el segundo, se pueden realizar consultas basadas tanto en el álgebra relacional como en el cálculo relacional de tuplas.

3.5.2. MySQL Workbench.

Herramienta visual de creación y diseño de base de datos para el sistema MySQL.

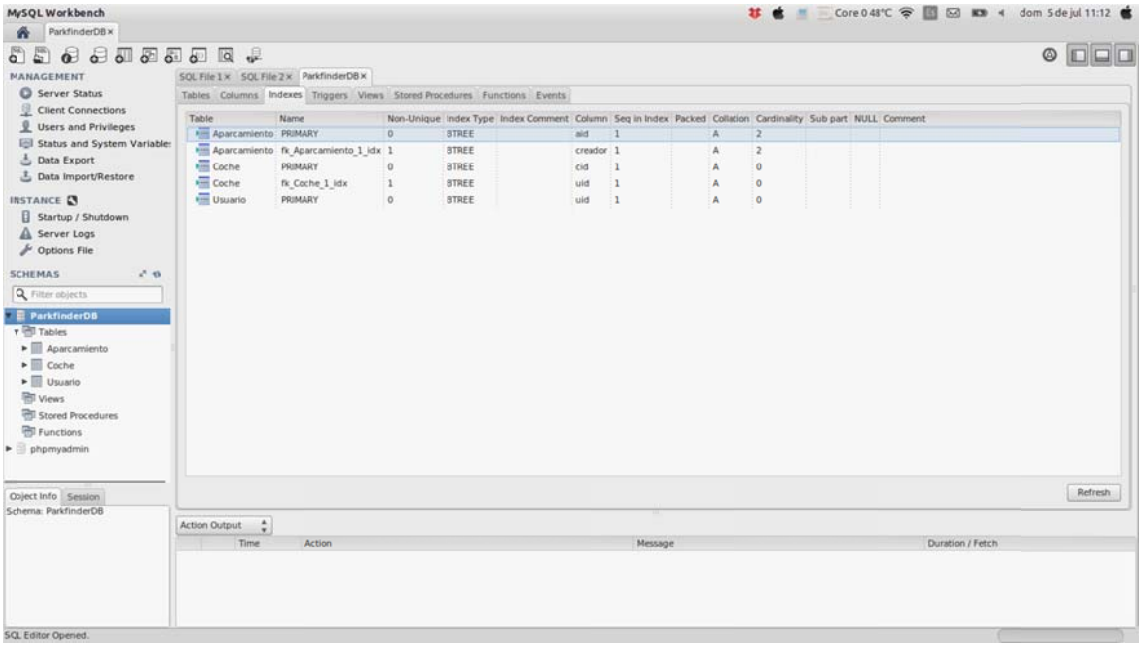


Imagen de la herramienta MySQL Workbench

3.5.3. phpMyAdmin.

Herramienta escrita en PHP con la que se pueden administrar bases de datos MySQL a través de una aplicación web.

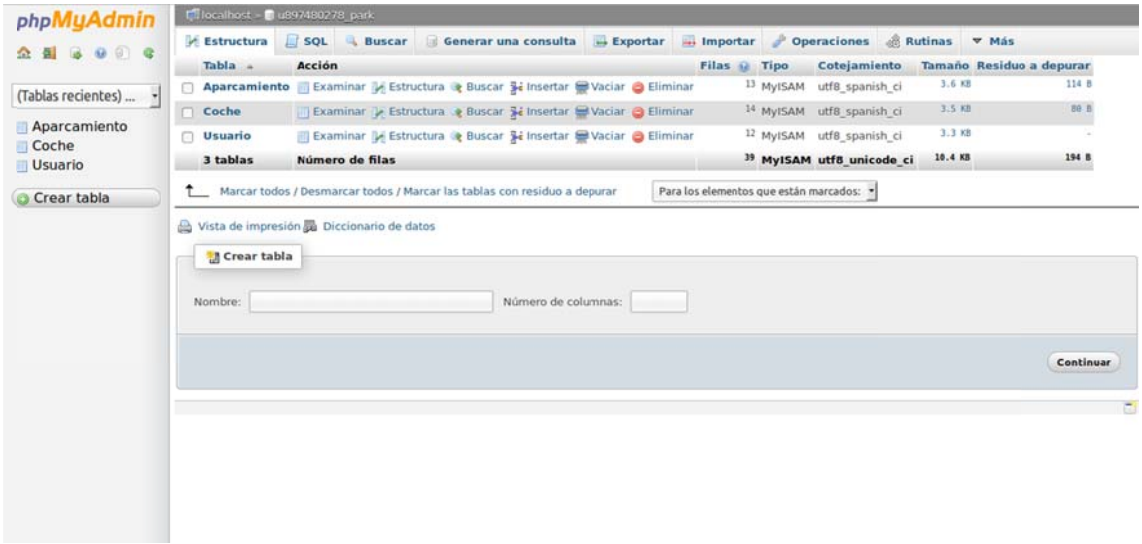
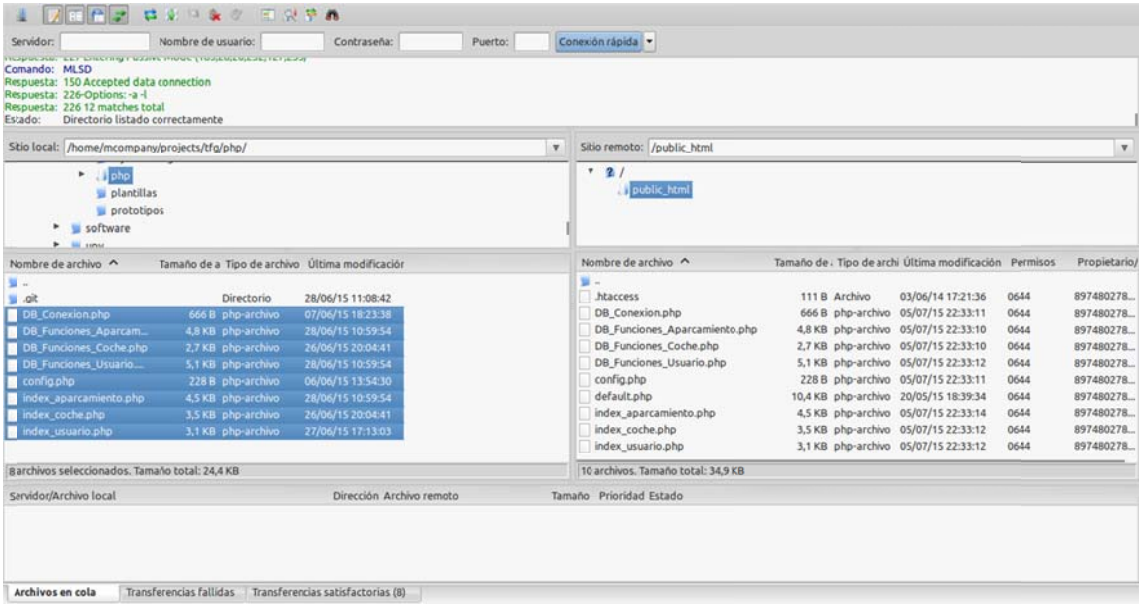


Imagen de la herramienta phpMyAdmin

3.5.4. FileZilla.

Cliente FTP de código abierto utilizado para subir y descargar ficheros desde el servidor remoto. Facilita mucho las tareas de intercambio de ficheros al ser rápido, en contraposición a la aplicación web del propio servidor remoto.



Pantalla principal de FileZilla

4. Arquitectura de la aplicación

La arquitectura de la aplicación es una arquitectura típica ***cliente – servidor***. Por un lado el cliente es la propia aplicación Android y por otro, el servidor se corresponde con una base de datos MySQL que ofrece soporte para la persistencia.

En este apartado se va a describir el software utilizado para acometer el desarrollo del proyecto. Cabe destacar que desde un primer momento se ha buscado utilizar software libre (*opensource*) por el hecho de no tener que pagar licencias, la versatilidad que ofrece al desarrollador y principalmente, por apoyarse en tecnologías que actualmente están muy extendidas y demandadas en el mundo laboral de hoy en día, lo cual es un aliciente para realizar este TFG.

Del párrafo anterior además se destaca el hecho de que por las mismas razones, software libre y demanda laboral, se ha desarrollado todo el TFG utilizando un sistema Linux. En concreto, la distribución Ubuntu 15.04.

Como ya se ha comentado en la introducción, la decisión de programar en Android nativo es entre otras razones, el uso de Java como lenguaje de programación. Este lenguaje tan extendido hoy en día es el que hemos estudiado en la carrera ampliamente y en el que me siento cómodo. Además durante estos últimos tiempos, he realizado dos cursos de programación para dispositivos móviles en Android y quería seguir aprendiendo a desarrollar para esta plataforma.

Es cierto que existían otras opciones, incluso mi tutor Pedro me propuso en un inicio realizar el desarrollo en la plataforma *Phonegap*, tecnología basada en HTML5, CSS3 y Javascript. Dicha plataforma tiene la gran ventaja de que se pueden desarrollar aplicaciones multiplataforma, lo cual es muy interesante cuando el desarrollador tiene experiencia en desarrollo web y va a desarrollar un proyecto tanto para Android como para iOS u otros. Sin embargo, tenía la decisión tomada por las razones expuestas: La oportunidad de aprender a desarrollar en Android una aplicación completa desde cero.

4.1. Entorno de desarrollo.

En un primer momento se debía elegir el entorno de desarrollo en el cual desarrollar la aplicación para Android nativo. Justo en el momento en el que comencé a trabajar en el TFG, a finales del año 2014, aparece la noticia de que Google deja de dar soporte al plugin ADT de Eclipse (*Android Development Toolkit*), convirtiendo el entorno **Android Studio** como el IDE (*Integrated Development Environment*) oficial para Android. Con lo cual hubo que tomar la decisión de cambiar a un entorno nuevo como Android Studio o continuar con el plugin ADT de Eclipse para realizar la parte de desarrollo, ya que hasta la fecha había trabajado únicamente con Eclipse. Al principio fue una decisión difícil de tomar por no tener experiencia previa con Android Studio, pero finalmente la decisión fue decantarme por este último y formarme en el uso de un nuevo IDE con futuro en el desarrollo de aplicaciones Android.

4.1.1. Técnicas de *debugging* adoptadas.

Para el desarrollo de una aplicación Android de este tipo la cantidad de veces que hay que realizar tareas de *debugging* (depuración) es inmensa. Es muy importante disponer de los medios suficientes para conseguir trabajar en un entorno que nos muestre la mayor cantidad de información posible acerca de los errores y los valores que posee cada variable en tiempo real.

Por defecto, Android Studio ofrece un entorno muy bien pensado para la depuración de código Java. Sin duda es una de las cosas que más me han ayudado a lo largo de estos meses de desarrollo.

Como apoyo a la potente herramienta de *debugging* de Android Studio, el desarrollador también puede visualizar la ejecución de la aplicación en un emulador, como por ejemplo **Genymotion**, el cual corre sobre una máquina virtual de forma muy rápida y estable. Android Studio también dispone del emulador nativo AVD (*Android Virtual Devices*) que a diferencia de Genymotion es más lento y complica las tareas de *debugging*.

Captura de pantalla de la aplicación Genymotion

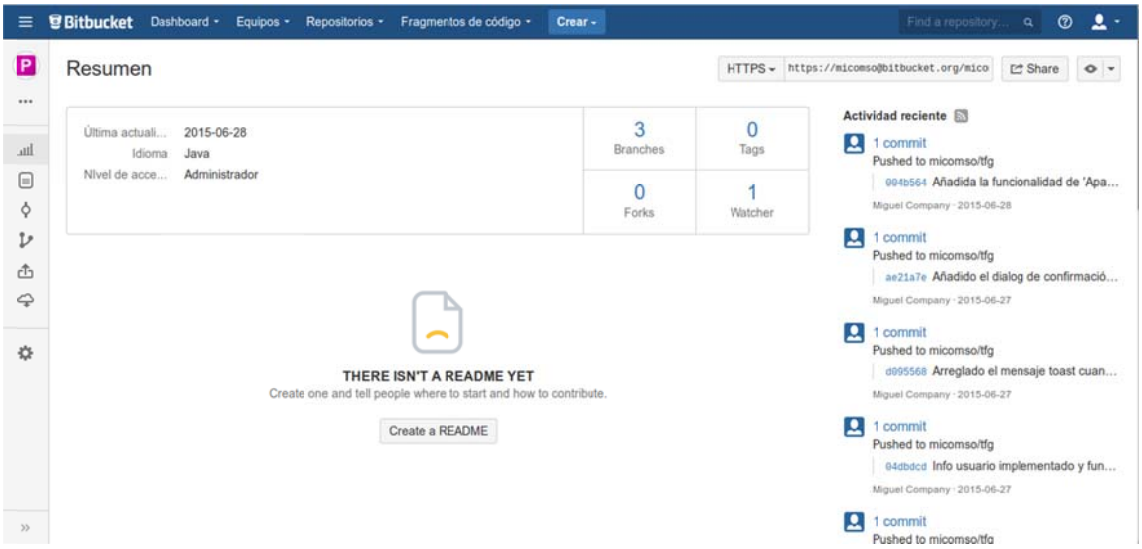
Sin embargo, como **Genymotion** no incluye por defecto la aplicación de mapas de Google Maps ni los servicios web, cambié el modo de visualizar la aplicación y utilicé un dispositivo Android real para continuar con las tareas de *debugging*. Obviamente, tarde o temprano tendría que hacer esto para realizar pruebas con el GPS del dispositivo móvil, pero al menos, las

interfaces gráficas de usuario pude visualizarlas con la ayuda de esta aplicación, lo cual me fue de gran ayuda para comprobar el funcionamiento de transiciones entre vistas y comprobar el funcionamiento de la consistencia en los formularios de registro y de autenticación.

Por otro lado, y en adelanto a la parte de desarrollo *back-end*, las tareas de *debugging* con el lenguaje de servidor PHP no han sido tan intuitivas y fáciles de tratar, con lo que ha sido necesario acceder a los ficheros de tipo *log* que el servidor me proporcionaba. Estas tareas, sin duda, han sido las que más quebraderos de cabeza me han dado por la nula experiencia que tenía con el lenguaje PHP.

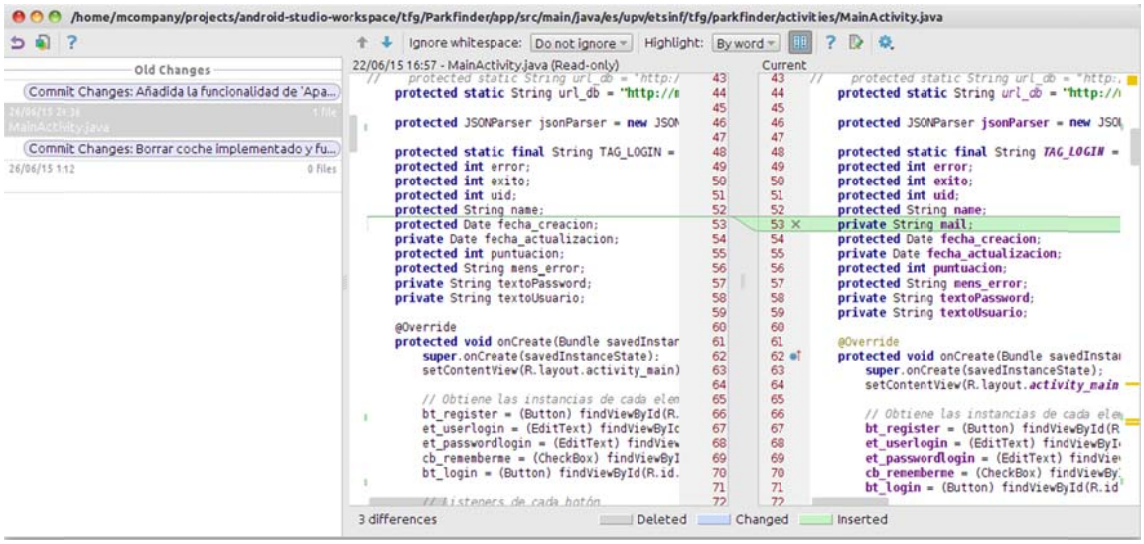
4.1.2. Control de versiones.

Aunque no era imprescindible utilizar un software de control de versiones puesto que he trabajado individualmente, me decidí desde un principio a utilizar **Git** por la seguridad de mantener el trabajo realizado en un repositorio remoto, evitando sufrir una catástrofe en caso de pérdida de datos, por ejemplo. Además he podido contar con la gran ventaja de poder recuperar, en cualquier momento, partes de código que había modificado posteriormente. Para ello he utilizado como repositorio remoto uno creado en la plataforma gratuita BitBucket.



Aplicación web de BitBucket

Además, Android Studio trabaja de manera excepcional con este tipo de software, ya que incluye un plugin por defecto que incorpora toda la funcionalidad de Git. Algunas de las acciones que se pueden realizar con este plugin son: Comparar versiones subidas al servidor con los cambios realizados en la versión local, almacenar cambios realizados y poder volver atrás cuando lo necesitaba, etc. Esta herramienta ha sido uno de los mayores aciertos para el desarrollo de este TFG, pues me ha facilitado mucho el trabajo.



Android Studio. Comparativa entre la versión local y la versión del repositorio BitBucket de un fichero java

4.2. API de Android.

Android se puede considerar como un *framework* del lenguaje de programación Java, es decir, un esquema de desarrollo específico creado para la implementación de una aplicación sobre dicho lenguaje. De manera general, la ventaja de utilizar un *framework* radica en que el desarrollador no tiene que definir una estructura global de la aplicación, sino que tiene que ir añadiendo estructuras y servicios que le ofrece el propio *framework* en el lenguaje base en el que se desarrolla. Es por ello que a continuación se van a detallar una serie de elementos y estructuras básicas de la programación en Android, entendido como un *framework* de desarrollo en lenguaje Java.

4.2.1. Activities.

Una *Activity* representa la lógica de negocio de una pantalla de la aplicación visualizada gracias a una interfaz gráfica de usuario (en Android estas interfaces gráficas se implementan en formato XML).

La manera de implementar una *Activity* nueva es muy sencilla, tan solo hay que heredar de la clase predefinida *Activity* de Android. Si bien es cierto, existen algunas subclases de esta misma clase a las que también se les puede aplicar la herencia y que añaden alguna funcionalidad extra. Por ejemplo: Para implementar listas (*ListActivity*), para añadir *fragments* (*FragmentActivity*), etc. Después de aplicar la herencia se pueden sobreescribir una gran cantidad de métodos propios del *framework* como: *onCreate*, *onPause*, *onResume*, etc.

El que más se ha utilizado para el desarrollo de la aplicación, sin duda, es el primero, *onCreate*. En este método se inicializan todas las referencias a botones y cuadros de texto, además de añadir la funcionalidad de cada uno de ellos, los eventos de escucha, etc.

4.2.2. *Layouts* y ficheros XML.

Un *layout* es una estructura que define como se reparten los elementos dentro de una interfaz gráfica. En Android, los *layouts* pueden ser declarados directamente sobre el código e instanciarlos en tiempo de ejecución, o de otra forma mucho más fácil, que es como se han utilizado en el desarrollo de esta aplicación, mediante la edición de ficheros XML.

De esta forma se separa la capa de presentación de la capa de lógica. Los ficheros XML se pueden incluso previsualizar en Android Studio mientras se van añadiendo y modificando elementos, tal y como se podía hacer también con el plugin ADT de Eclipse.

Para enlazar estos ficheros con la capa de negocio, se le pasa una referencia (identificador del *layout*) al método *onCreate* nombrado en el apartado anterior.

```
package es.upv.etsinf.tfg.parkfinder.activities;

import ...

public class InfoUserActivity extends Activity {

    private Button bt_aceptar;
    private String name;
    private String mail;
    private int puntuacion;
    private String cadenaFechaCreacion;
    private TextView tv_name;
    private TextView tv_mail;
    private TextView tv_fecha;
    private TextView tv_puntuacion;
    private TextView tv_prioridad;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_info_user);

        bt_aceptar = (Button) findViewById(R.id.aceptar_info_user);
        tv_name = (TextView) findViewById(R.id.info_user_text_name);
        tv_mail = (TextView) findViewById(R.id.info_user_text_mail);
        tv_fecha = (TextView) findViewById(R.id.info_user_text_fecha);
        tv_puntuacion = (TextView) findViewById(R.id.info_user_text_puntuacion);
        tv_prioridad = (TextView) findViewById(R.id.info_user_text_prioridad);
```

Método onCreate() de una Activity

Existen muchos tipos de *layouts*. Estos son algunos de los que se han utilizado en la aplicación, y cuyas implementaciones se pueden consultar en el apartado ANEXOS de la presente memoria:

- **LinearLayout**: Ordena los elementos vertical u horizontalmente, uno después de otro. Ha sido muy empleado en formularios, por ejemplo.

- **RelativeLayout**: Ordena los elementos en posiciones relativas, indicando la posición relativa entre elementos considerados como "hermanos" (*siblings)*. Muy empleado en formularios y para colocar botones.

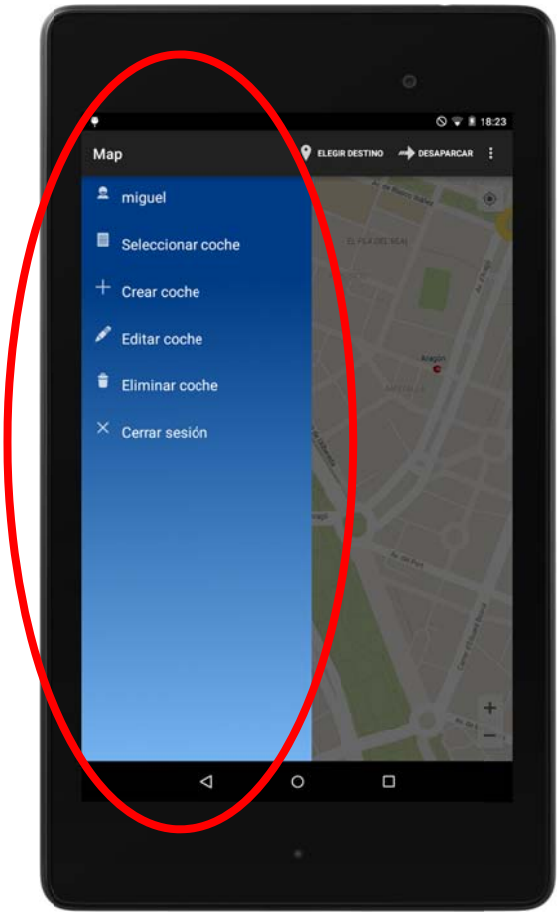
- **GridView**: Crea una matriz de dos dimensiones y coloca los elementos (imágenes o botones, típicamente) en celdas. En este desarrollo ha sido empleado para posicionar botones dividiéndolos en filas y columnas en una de las vistas, añadiendo cualidades de usabilidad a la interfaz como veremos más adelante.

- **ListView**: Permite visualizar listas e implementa una *barra de scroll* automática. Utilizado obviamente para aquellas vistas donde hay que listar elementos.

Cabe destacar que para rellenar los elementos *ListView* o *GridView* es necesario el uso de objetos de tipo *Adapter*.

4.2.3. *Adapters*.

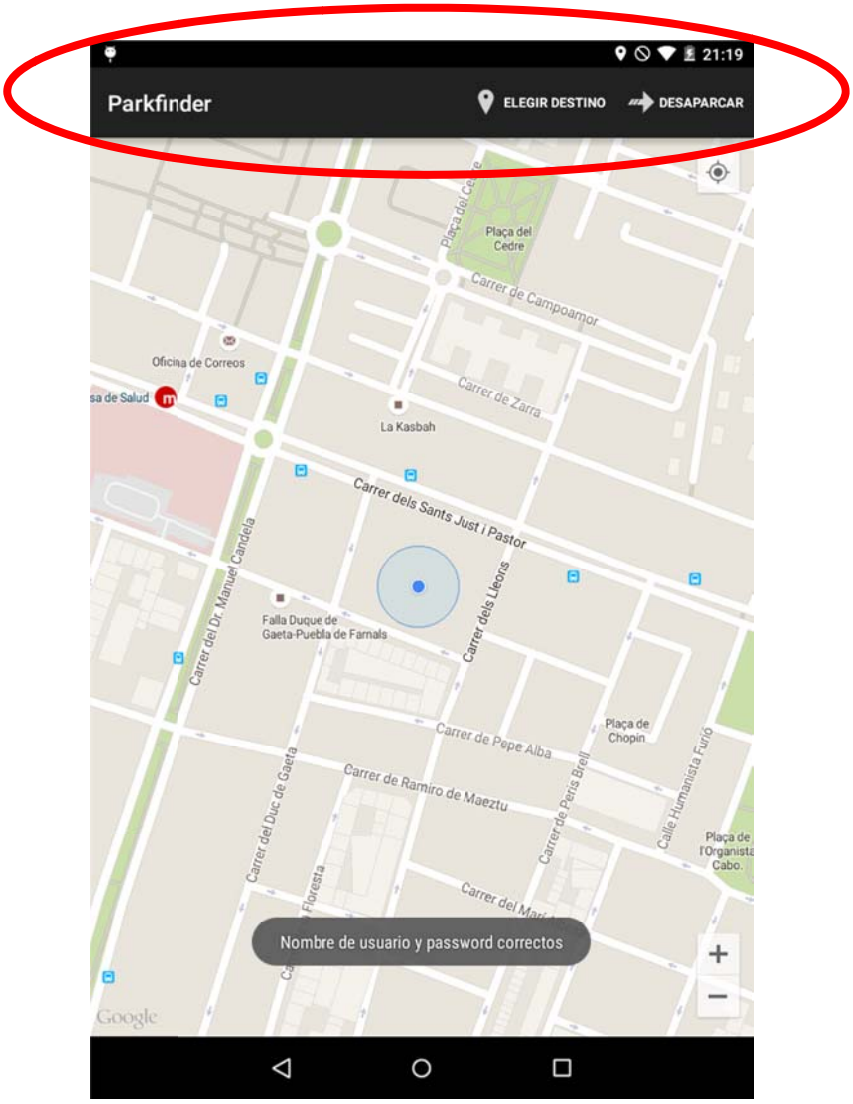
Este tipo de estructuras actúan como enlace entre las interfaces de usuario descritas en ficheros XML, tal y como se ha visto, y el código Java de la aplicación. Se han utilizado principalmente para rellenar listas y matrices de elementos. Existen algunos ya creados por defecto, pero también se pueden personalizar. En la aplicación desarrollada se han utilizado *ArrayAdapters* para rellenar listas, por ejemplo, y para el caso de *Adapters* personalizados, una de sus funciones ha sido rellenar un menú lateral deslizable (*NavigationDrawer*).



Navigation Drawer (Menú deslizable)

4.2.4. *ActionBar*.

Se corresponde con el menú horizontal que aparece arriba de la aplicación. Para implementarlo basta con extender de la clase *ActionBarActivity*. Se han incluido algunos botones de acciones principales de la aplicación como *Buscar destino* y *Desaparcar*. Gracias a este menú, el usuario a simple vista, dispone de las acciones principales de la aplicación. El resto de acciones secundarias se encuentran en el menú deslizable lateral, ocultas para que no interfieran en las acciones principales.



Menú ActionBar.

4.2.5. *Intents*.

Los objetos de tipo *Intent* se utilizan en Android básicamente para realizar transiciones entre *Activities* y enviar y recoger datos entre ellas. Existen dos tipos de *Intents*, por facilidad de uso se han utilizado para este desarrollo únicamente los de tipo explícito. Los *Intents* implícitos no tenían utilidad en este proyecto ya que se suelen utilizar para comunicar aplicaciones nativas de los dispositivos Android con las aplicaciones que se instala el usuario, como pueden ser: La aplicación de correo electrónico, la agenda telefónica, la aplicación que muestra las imágenes almacenadas en el dispositivo, etc.).

```
/**
 * Este método obtiene los valores devueltos por la activity RegisterActivity
 *
 * @param requestCode petición codificada que permite identificar de donde viene el resultado
 * @param resultCode  devuelto por la segunda Activity a través del método setResult().
 * @param data        en este parámetro la segunda Activity puede devolver datos como resultado de su ejecución.
 */
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    Bundle bundle = data.getExtras();

    if (requestCode == GET_TEXT_REQUEST_CODE && resultCode == RESULT_OK) {
        et_userlogin.setText(bundle.getString("usuario"));
        et_passwordlogin.setText(bundle.getString("password"));
        cb_rememberme.setChecked(false);

        Thread hiloToastOk = new Thread((Runnable) () -> {
                runOnUiThread(() -> {
                        Toast.makeText(getApplicationContext(), "Usuario creado correctamente", Toast.LENGTH_LONG).show();
                });
        });
        hiloToastOk.start();

    } else if (requestCode == GET_TEXT_REQUEST_CODE && resultCode == RESULT_CANCELED) {
        et_userlogin.setText("");
        et_passwordlogin.setText("");
        cb_rememberme.setChecked(false);
    }
}
```

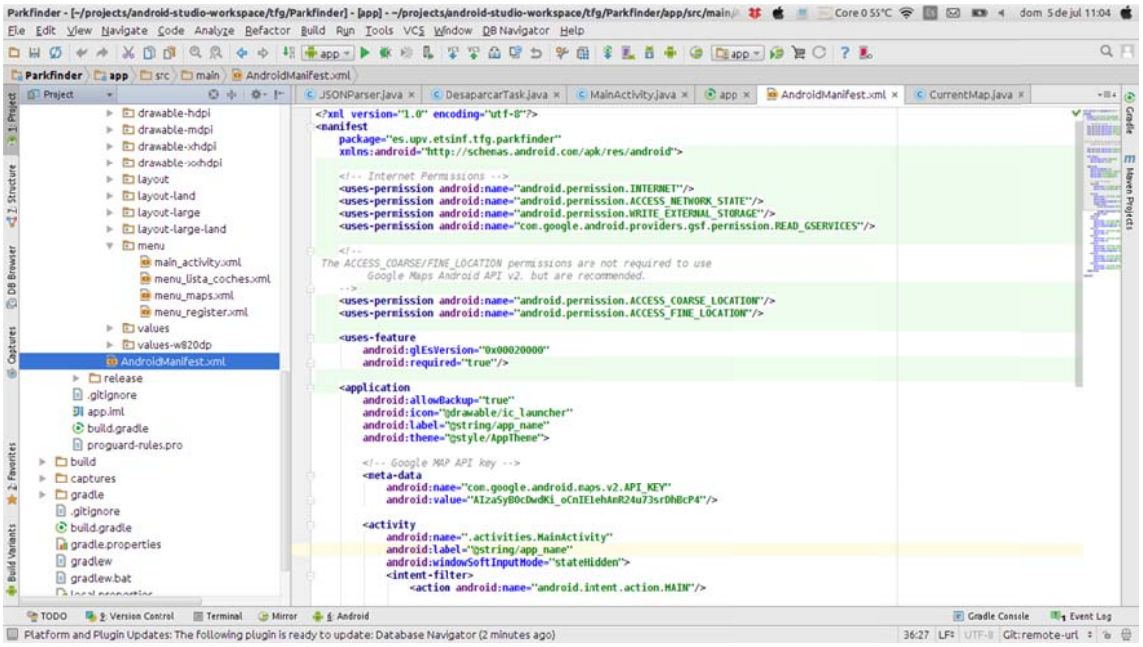
Ejemplo del método onActivityResult. Recoge los parámetros que le envía otra Activity.

4.2.6. El fichero *manifest.xml*

Todo proyecto de desarrollo en Android incluye este fichero. En él es obligatorio declarar todas las *Activities* que se crean en la aplicación, además de incluir los permisos de acceso a cierto hardware interno del dispositivo (cámara, GPS, etc.) y servicios software de esté a los que necesitará tener acceso la aplicación cuando se instale (carpeta de imágenes, contraseñas almacenadas, etc.).

Muy importante también es que incluye el nivel mínimo de la API de Android que requiere la aplicación para poder funcionar, es decir, que versión del sistema operativo deberemos tener instalada en nuestro dispositivo para que se pueda instalar la aplicación. En el caso de la aplicación desarrollada se ha decidido utilizar una de las últimas API publicadas, la API 22.

De manera particular en este desarrollo, fue necesario declarar en este fichero las librerías de Google Maps para poder utilizarlas (se detallará más adelante).



Fichero manifest.xml del proyecto Parkfinder

4.2.7. *AsyncTask*.

Esta clase predefinida en Android ha sido de gran utilidad para el desarrollo llevado a cabo. Facilita la descarga de trabajo al hilo principal de la aplicación o *UIThread*, creando y eliminando hilos de manera transparente al desarrollador. Con lo que en los dispositivos de hoy en día, que disponen de varios núcleos en el procesador, utilizar hilos de ejecución a parte del hilo principal se antoja necesario. En el caso de la aplicación desarrollada ha sido de gran utilidad para implementar las conexiones con la base de datos, de manera que el envío y recepción de datos no sobrecargaba la aplicación, lo cual hubiera hecho imposible su funcionamiento fluido.

La manera práctica de utilizar esta clase es creando una clase nueva que extienda de esta o utilizándola como clase interna. Para el desarrollo se ha utilizado tanto una manera como otra. Concretamente, cuando la *Activity* que necesitaba realizar una conexión con la base de datos incorpora muchas líneas de código, se optó por utilizar la implementación en forma de una nueva clase, descargando así la propia *Activity* de líneas de código que podrían llegar a generar ficheros muy extensos.

El funcionamiento de la clase AsyncTask se lleva a cabo realizando una herencia, como ya se ha comentado, y sobreescribiendo los métodos que se necesiten para el desarrollo. En particular para este desarrollo, se han utilizado siempre estos tres métodos:

- ***doInBackGround()***: Este método lo ejecuta el hilo creado de manera transparente al desarrollador.

- ***onPreExecute()***: Este método lo ejecuta el hilo principal mientras se está ejecutando el método anterior. Es aquí donde se ha implementado un objeto de tipo *ProgressDialog* que dibuja sobre la pantalla una barra de progreso circular que

retroalimenta visualmente al usuario mientras la aplicación se conecta con el servidor de la base de datos. De gran utilidad en el ámbito de la usabilidad de dispositivos móviles.

- ***onPostExecute()***: Este método lo ejecuta el hilo principal cuando se termina de ejecutar el primer método, es decir, cuando el hilo secundario termina de ejecutarse y finaliza. Es necesario utilizar variables globales para compartir datos entre el hilo principal y los hilos secundarios.

El uso de esta clase ha sido de gran utilidad, sin embargo, al principio fue difícil entender cómo utilizar los métodos heredados correctamente. El estudio en profundidad del manejo de esta clase me ha ayudado a entender la importancia de no sobrecargar el hilo principal de ejecución, puesto que además de ejecutar el código, es el que soporta la visualización en pantalla. De ahí el nombre *UIThread* (*User Inteface Thread*).

```
@Override
protected void onPreExecute() {
    super.onPreExecute();
    pDialog = new ProgressDialog(activity);
    pDialog.setMessage("Generando un nuevo aparcamiento...");
    pDialog.setIndeterminate(false);
    pDialog.setCancelable(true);
    pDialog.show();
}

@Override
protected String doInBackground(String... params) {

    StringBuilder sb = new StringBuilder();

    // Creando los parámetros
    this.params.add(new BasicNameValuePair("latitud", String.valueOf(cm.getLocation().getLatitude())));
    this.params.add(new BasicNameValuePair("longitud", String.valueOf(cm.getLocation().getLongitude())));
    this.params.add(new BasicNameValuePair("creador", String.valueOf(this.activity.getUid())));
    this.params.add(new BasicNameValuePair("dimension", String.valueOf(this.activity.getDimensionCoche())));
    this.params.add(new BasicNameValuePair("tipo_plaza", String.valueOf(this.tipo_parking)));
    this.params.add(new BasicNameValuePair("tag", TAG_CREAR_APARCAMIENTO));

    // Tiempo de espera para que se muestre el elemento Progress Dialog un tiempo suficiente
    try {
        sleep(2000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
```

Ejemplo de los métodos onPreExecute y doInBackground

```
        // Obteniendo el objeto JSON
        // Utilizando el método POST
        JSONObject json = jsonParser.makeHttpRequest(url_db, "POST", this.params);

        try {
            error = json.getInt("error");
            exito = json.getInt("exito");

            if (error == 1) {
                sb.append("ERROR: ");
                sb.append(json.getString("error_msg"));
                msg = sb.toString();

            } else if (exito == 1) {
                sb.append("INFO: ");
                sb.append((json.getString("exito_msg")));
                msg = sb.toString();

                latitud = Double.parseDouble(json.getString("latitud"));
                longitud = Double.parseDouble(json.getString("longitud"));
            }
        } catch (JSONException e) {
            e.printStackTrace();
            Log.d("Respuesta creada", json.toString());
        }
        return null;
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);

        creaMensaje();
        pDialog.dismiss();
    }
```

Ejemplo del método onPostExecute

4.2.8. *Location*. Sensor Wi-Fi y GPS.

Los servicios de localización de Android proveen una API muy potente para manejar los sensores que incorporan los dispositivos móviles, como el GPS o el Wi-Fi. Sin duda es una de las partes más complejas del desarrollo de la aplicación aunque Android facilita el acceso al hardware del dispositivo de una manera sencilla mediante la implementación de métodos como: *onLocationChanged*, *onStatusChanged*, *onProviderDisable*, *onProviderEnabled*, etc.

Obtener la posición en un dispositivo móvil es complicado ya que depende del sensor que esté recibiendo la señal en ese momento y de la calidad de la misma, por lo que la precisión de las coordenadas obtenidas puede variar considerablemente. En el desarrollo de esta aplicación la precisión de las coordenadas obtenidas es un detalle muy a tener en cuenta, puesto que como se verá más adelante y como adelanto, para comprobar cuándo un usuario ha aparcado en una plaza de aparcamiento, se comprueban las coordenadas del usuario en ese momento con las coordenadas almacenadas de la plaza de aparcamiento. Si coinciden dentro de un rango de tolerancia se considera que el usuario ha aparcado correctamente. Algo que puede ser complicado puesto que la precisión del navegador GPS instalado en los dispositivos móviles de hoy en día no es mejor de 3 metros en el mejor de los casos, es decir, con condiciones de lectura de la señal GPS idóneas (sin árboles o edificios alrededor, y con lecturas a varios satélites simultáneamente).

Como un coche ya tiene un tamaño entre 3 y 6 metros de longitud, la tolerancia debe ser un valor intermedio, con el riesgo de perder funcionalidad en la aplicación cuando las coordenadas no tienen buena precisión (ya sean las del usuario en ese momento y/o las almacenadas en la base de datos para cada plaza de aparcamiento). Quizás en un futuro con la llegada de las ciudades inteligentes se conseguirá que este tipo de inconvenientes se puedan mejorar por medio de las señales Wi-Fi, que también son capaces de dar posicionamiento a partir de triangulaciones entre las propias antenas emisoras y el receptor de la señal.

4.2.9. Mensajes *Toast.*

Los mensajes *Toast* son muy útiles para ofrecer información extra en forma de *popup* ante eventos o sucesos específicos. Tienen un tiempo determinado de duración y cuando este expira desaparecen automáticamente. Estos mensajes se han utilizado en el desarrollo de la aplicación para indicarle al usuario información sobre los siguientes eventos:

- Errores de conexión devueltos por la base de datos.
- Errores de consistencia en formularios.
- Información cuando ciertas acciones se han procesado correctamente.
- Avisos cuando se conecta o desconecta manualmente el GPS o Wi-Fi del dispositivo.

Estos mensajes ayudan a que el usuario no se sienta perdido cuando ocurre algún error y le dan seguridad cuando se introducen datos correctamente, lo cual es muy útil cuando no se tiene conocimiento del funcionamiento de la aplicación. En el caso de estudio planteado en la introducción y que se verá posteriormente, la aplicación va a ir dirigida a un grupo de usuarios con escasos conocimientos de informática. Este tipo de mensajes realimentan la comunicación entre la aplicación y los usuarios, además son sutiles, ocupan poco espacio y duran unos segundos tan solo, con lo que a los usuarios avanzados no les incomoda.

```
@Override
public void onProviderDisabled(String provider) {
    Log.i("PROVIDER disabled", provider);
    Thread hiloNoGPS = new Thread(new Runnable() {
        @Override
        public void run() {
            mapsActivity.runOnUiThread(() -> {
                    Toast.makeText(mapsActivity.getApplicationContext(),
                            "Servicios de geoposicionamiento DESHABILITADOS", Toast.LENGTH_LONG).show();
            });
        }
    });
    hiloNoGPS.start();
}
```

Implementación de un mensaje Toast de Android

Muestra del mensaje Toast implementado

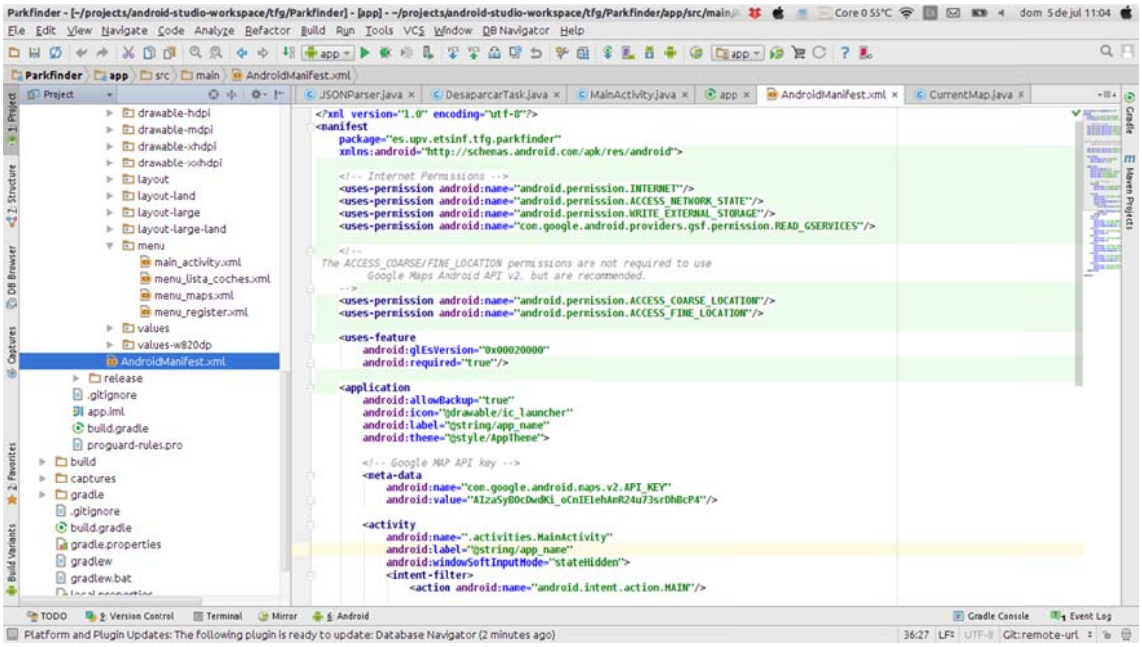
4.3. Servicios Web de Google Maps.

Para poder emplear los servicios web de Google es necesario crear una clave que nos servirá para acceder a los servidores donde se alojan dichos servicios.

Existen diversos tutoriales donde te explican cómo obtener dicha clave, paso a paso. La web oficial de desarrolladores de Google para Android dispone de un tutorial completo, que es el que se ha seguido para el desarrollo de la aplicación:

https://developers.google.com/maps/documentation/android/signup

Tras generar la clave es necesario que aparezca en el fichero *manifest.xml*, de manera que la aplicación obtenga los permisos necesarios para manejar los mapas de Google Maps. Además es necesario indicar en este fichero que la aplicación requiere permisos para acceder a los servicios de localización del dispositivo (WiFi y GPS).



Fichero manifest.xml del proyecto Parkfinder

4.4. Conexión con la base de datos.

En cuanto a la descripción de la arquitectura, falta por comentar la parte del servidor, donde el trabajo realizado ha sido casi tan extenso como la propia aplicación, incluso más complejo por las dificultades aparecidas a la hora de depurar el código PHP.

4.4.1. Servidor.

La base de datos MySQL está alojada en un servidor remoto gratuito utilizado para realizar pruebas de campo con comodidad. Además disponer de este servidor remoto ha servido para adquirir experiencia en el manejo de ficheros de *log* para poder depurar errores.

4.4.2. PHP.

Se ha realizado una programación modular, es decir, un fichero por cada parte diferencial de la aplicación. Esta modularidad se ve reflejada en los siguientes ficheros:

- Fichero con los datos de conexión a la base de datos MySQL (usuario, password, host, etc.).
- Fichero con el que se realiza la conexión a partir de los datos del fichero anterior.
- Ficheros index, donde se leen los *tags* y datos enviados por la aplicación cliente.
- Ficheros con las funciones donde se realizan las consultas a la base de datos en lenguaje SQL.

Para recargar los cambios realizados en los ficheros mientras se realizaban las pruebas se ha utilizado el software ***FileZilla***, que también se ha utilizado para descargar los ficheros de *log*.

Por motivos de seguridad, también se han subido estos ficheros al repositorio de BitBucket, lo que ha facilitado además agilizar las pruebas y volver atrás en ciertas modificaciones.

4.5. Envío de datos entre el servidor y la aplicación cliente.

El envío de datos entre la aplicación cliente y el servidor se ha realizado de manera similar en ambas direcciones mediante el formato ligero JSON. Para cada una de las direcciones hay algunas diferencias, debido sobretodo al lenguaje que recibe o envía los datos (Java o PHP):

- **Cliente → Servidor**: La aplicación cliente envía una etiqueta (*tag*) y variables mediante cadenas de texto (*String*), que más tarde se utilizarán en el servidor para realizar consultas a la base de datos.

- **Servidor → Cliente**: El servidor utiliza una función PHP (*json_encode*) para codificar el resultado de las consultas en formato JSON. El cliente recibe respuestas en dicho formato que son leídas mediante un analizador sintáctico (*parser*).

5. Metodología utilizada

En este apartado se va a detallar la metodología desarrollada para obtener el diseño de la interfaz gráfica de usuario. En posteriores apartados de la memoria se pueden visualizar los resultados y elementos que conforman el estudio de usabilidad, como los escenarios de uso, el modelo *Personas*, los prototipos y la evaluación de usabilidad.

5.1. Diseño de la interfaz gráfica de usuario.

Como se ha explicado en la introducción, se va a llevar a cabo un diseño centrado en el usuario utilizando para ello la técnica *Personas* y unos escenarios de uso en los que se basará más tarde el diseño y la evaluación del resultado obtenido.

Dicha técnica consiste básicamente en crear un usuario ficticio que represente a una agrupación de personas o colectivo, objeto del estudio. Se propone como requisito por parte de los tutores, que dicho usuario ficticio tomado como modelo tenga las siguientes características y cualidades personales:

- Vive en el centro de Valencia.
- Trabaja en las afueras de la ciudad.
- Se desplaza en su coche particular al trabajo.
- Tiene conocimientos informáticos escasos.
- Es de mediana edad (35 años).

El empleo de este usuario ficticio teniéndolo en mente ayuda a aumentar la empatía hacia las personas a las que va dirigido el estudio. Es algo fundamental que nos ofrece esta técnica, ya que si queremos enfocar el diseño a una agrupación de personas con sus necesidades, cualidades y objetivos particulares, es de ayuda enfocar el estudio como si de una única persona se tratara e imaginando que es una persona real. Además evita que el diseñador se tome como referencia a sí mismo, con los consiguientes errores que ello puede conllevar.

Para entenderlo mejor, si se intentara diseñar las interfaces atendiendo únicamente a valores estadísticos, estudios expertos, encuestas, etc., se perdería la visión real de lo que es una persona y por lo tanto, no se tendría esa empatía necesaria de la que se ha hablado antes.

En definitiva, con la técnica *Personas* se consiguen unos objetivos mucho más precisos y efectivos a la hora de establecer el diseño de la interfaz de usuario.

Entre las características que debe incorporar esta persona ficticia tomada como modelo se tienen las siguientes:

- Es una descripción de la persona, un estereotipo al que va dirigido el diseño de la interfaz de usuario.
- Debe ser específico en cuánto a sus cualidades y características, pero sin ser excesivo.
- Debe incluir datos reales. Ser creíble.
- Nos debe llevar a pensar que conocemos a esa persona, asociándola a alguien real.
- Debe tener características comunes de la agrupación a la que va dirigida el estudio.
- No debe incorporar valores medios de cada grupo de población.

- Debe incluir información personal acerca de la familia, su vida diaria, tipo de trabajo, lugar donde vive, estado civil, etc.
- Es importante incluir gustos o aficiones como si se tratara de una persona corriente.

Cabe destacar que existen varias tipologías de *Personas*. La más utilizada y que es la que se va a emplear para este proyecto es la *Persona primaria*. Esta es en la que se centra el diseño y debe ser única. Es decir, no pueden coexistir varios modelos ficticios de este tipo para el diseño de una misma interfaz.

Por otro lado, aunque no se va a emplear, también existe la *Persona secundaria*. Esta tipología añade necesidades y objetivos a los que ya incorpora la *Persona primaria* pero sin entrar en conflicto entre ellas.

Y por último, también existe otra tipología, la *Persona negativa*, que es aquella para la que no debe estar dirigido el diseño de la interfaz. Tampoco se va a utilizar en este trabajo.

A partir del modelo de *Persona* que se va a definir obtendremos posteriormente unos *escenarios de uso* con los que nuestro modelo de *Persona* alcanzará los objetivos buscados. Estos escenarios describen situaciones reales del usuario ante la aplicación a desarrollar, desde los momentos previos a la puesta en marcha de la aplicación, hasta cuando el usuario ha conseguido su objetivo. Esto le indica al diseñador las líneas a seguir a la hora de realizar el diseño de la interfaz.

Para describir estos escenarios existen diferentes tipos de descripciones, dependiendo del nivel de detalle. A continuación se ilustran los tipos de escenarios siguientes: *General*, *intermedio* y *específico*.

- El *Escenario General* es una simple descripción de lo que el usuario realiza de forma genérica, sin entrar en detalles de la aplicación ni en las acciones que realiza el usuario con ella.
- El *Escenario Intermedio* es una descripción de las acciones que realiza el usuario al utilizar la aplicación pero sin definir detalles del diseño de la misma.
- El *Escenario Específico* es una descripción más exhaustiva que las anteriores sobre las acciones que realiza el usuario al utilizar la aplicación, dando además detalles del diseño de la interfaz de usuario.

La *Persona primaria* y los escenarios se funden en un relato que ayuda al diseñador a describir cómo consigue el usuario ficticio alcanzar sus propios objetivos, dando pistas del diseño que es necesario acometer y de la funcionalidad básica que debe incorporar la aplicación.

6. *Persona* y escenarios

En primer lugar se va a mostrar el modelo de *Personas* confeccionado a partir de los requisitos anteriormente mencionados:

- Vive en el centro de Valencia.
- Trabaja en las afueras de la ciudad.
- Se desplaza en su coche particular al trabajo.
- Tiene conocimientos informáticos escasos.
- Es de mediana edad (35 años).

Posteriormente, se muestran los escenarios de uso general que se obtienen a partir de este modelo. Estos escenarios han sido detallados de manera general y sin entrar en detalles de la propia implementación de la aplicación. También se ha tenido en cuenta el contexto en el que se encuentra el propio usuario y cómo va realizando las acciones hasta llegar a los objetivos buscados.

6.1. Modelo *Personas*.

David Fernández

Biografía

- Tiene 35 años.
- Es extrovertido y muy ordenado.
- Nació en Valencia.
- Vive en un piso situado en el centro de la ciudad, junto con su mujer y su hijo de 4 años.
- Trabaja en una multinacional como operario de una cadena de fabricación.
- El lugar de trabajo se encuentra en las afueras de la ciudad.
- Tiene un coche de tipo monovolumen con el que se desplaza diariamente.
- Aparca el coche en la calle ya que no dispone de una plaza de parking.
- Todas las mañanas coge el coche para ir a trabajar y regresa por la tarde.
- De camino al trabajo, deja a su hijo en la guardería cada mañana.
- Sus conocimientos de informática son escasos.
- Dispone de un *Smartphone* que le regaló su mujer hace poco tiempo.
- Se comunica con sus amistades, únicamente por teléfono.
- No ha utilizado nunca las redes sociales.
- Hace deporte 3 o 4 veces por semana.

Objetivos

- No tienen pensado marcharse a vivir a otro lugar más alejado del centro puesto que su mujer trabaja muy cerca de donde viven.
- Está buscando una plaza de aparcamiento pero son muy caras por la zona, con lo que de momento se conforma con aparcar lo más cerca de casa posible.
- Siente curiosidad y le gustaría aprender a manejar aplicaciones en su Smartphone que sean divertidas y fáciles de utilizar.
- El año que viene tiene pensado correr una maratón aunque solo dispone de tiempo para entrenar cuando regresa de trabajar.
- A pesar de que le resulta incómodo aparcar un coche tan grande, no quiere venderlo porque les gusta viajar con él y visitar los Pirineos al menos una vez al año.

6.2. Escenarios de uso general.

Escenario General: Indicar cómo el usuario deja un aparcamiento libre

David se dirige andando hacia su coche de buena mañana. Lleva a su hijo en brazos y está deseando llegar al coche para colocarlo en su asiento. Como casi siempre, piensa en que necesitan una plaza de parking para no tener que aparcar tan lejos. Está pensando que hoy ha salido un buen día y esta tarde sería un buen momento para salir a entrenar, solo espera que cuando regresen, encuentre aparcamiento rápidamente y no tenga que dar muchas vueltas. Inmediatamente, deja constancia del hueco libre que va a dejar y arranca el coche para comenzar su rutina diaria.

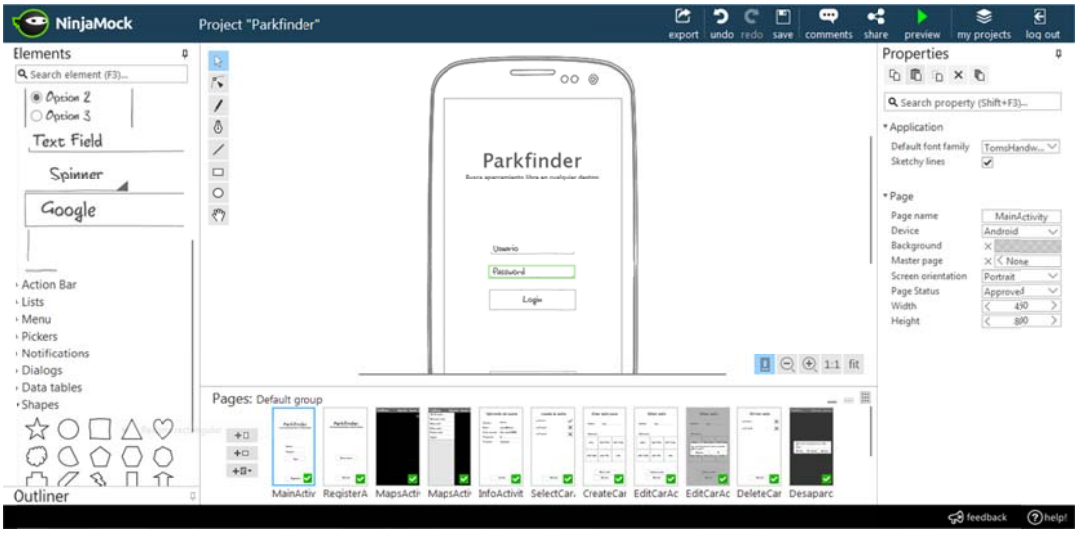
Escenario General: El usuario busca aparcamiento libre en un destino

David recoge a su hijo de la guardería como hace todas las tardes, antes de arrancar el coche consulta los aparcamientos libres que hay en la zona donde viven. Sabe que es hora punta y por eso necesita información previa para ver que trayecto sigue y por donde entrar al barrio donde viven. Cuando lo tiene claro, arranca el coche y se dirige hacia su destino con la mente puesta en el camino que ha de seguir. A pesar de que no es el camino que recorre a menudo y sabiendo que va a dar un cierto rodeo, tiene la seguridad de que aparcará más cerca y sin tener que dar varias vueltas como le ocurre de costumbre.

Tras el viaje de regreso, aparca el coche y deja constancia del hueco libre que ha ocupado con su coche. Ahora ya puede recoger a su hijo e irse andando a casa.

7. Diseño de la interfaz de usuario

Para el diseño de la interfaz gráfica de usuario se ha empleado la aplicación web ***ninjamock*** que permite realizar prototipos (*wireframes*) de manera rápida y sencilla. Además se pueden exportar los resultados a formato *PNG*.



Los prototipos realizados que se van a mostrar a continuación tienen la finalidad de representar los aspectos interactivos de la aplicación con un cierto nivel de precisión, y así poder evaluar su usabilidad y funcionalidad, pero sin entrar en detalles de implementación.

Por último, a modo de evaluación temprana se ha utilizado un método de inspección para realizar la evaluación de usabilidad. Concretamente, una evaluación heurística de los prototipos. Cabe destacar que al ser el presente trabajo un TFG individual, tan sólo interviene un evaluador en este estudio, cuando normalmente se recomienda utilizar entre 3 y 5 evaluadores.

7.1. Prototipos.

El diseño de los prototipos de la interfaz debe facilitar el uso de la aplicación a los usuarios por lo que se deben conocer qué acciones va a realizar el usuario objetivo, como si se estuviese en el interior de su mente. Para ello se va a hacer uso de los escenarios generales del apartado anterior.

7.1.1. Pantalla de *Login*.

La pantalla inicial se ha diseñado pensando en que el usuario introduzca sus credenciales y rápidamente pulse el botón de *Login*. También puede tener acceso a la pantalla de registro. Cualquier usuario nuevo en la aplicación entiende rápidamente que si no está registrado tiene una opción para realizarlo. El logo de la aplicación es orientativo. Se ha escogido este nombre realizando un sondeo de aplicaciones móviles existentes junto con la temática de la propia aplicación.

A. Pantalla inicial.

7.1.2. Pantalla de registro de usuario.

La pantalla de registro incluye las opciones necesarias para que el usuario pueda autenticarse posteriormente. Se ha incluido un campo *correo electrónico* con la idea de que en el futuro el usuario pueda recuperar su contraseña en caso de olvido o extravío (esta acción no será implementada como se verá más adelante, pero sí que aparece en la interfaz).

B. Pantalla de registro de usuario

7.1.3. Pantalla principal de la aplicación.

La pantalla principal está formada por un menú superior con las opciones básicas de la aplicación, de manera que un usuario intermedio pueda acceder rápidamente a las opciones básicas de búsqueda de aparcamiento y acción de desaparcar.

C. Pantalla principal

Como elemento añadido se ha diseñado un menú lateral al cuál se accederá mediante un movimiento de arrastre con el dedo (desde el lado izquierdo de la pantalla hacia la derecha). Este tipo de elementos ayudan a que la pantalla principal no tenga elementos secundarios que puedan interceder en la interacción rápida del usuario con la funcionalidad central de la aplicación.

En este menú deslizable se incluyen: Información del usuario, seleccionar coche, crear coche nuevo, editar coche, eliminar coche y cerrar la sesión actual.

D. Menú deslizable lateral

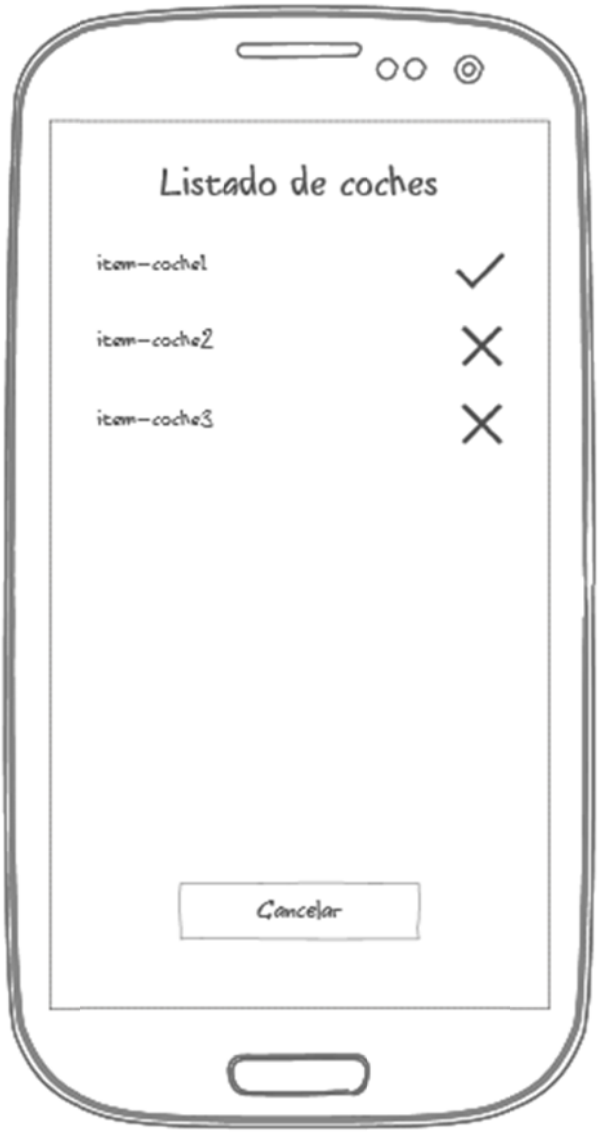
7.1.4. Información del usuario.

La pantalla de información muestra los datos personales del usuario además de la puntuación actual y la prioridad que supone tener dicha puntuación. El usuario puede consultar rápidamente este pantalla accediendo desde el menú deslizable.

E. Pantalla de información del usuario.

7.1.5. Seleccionar coche.

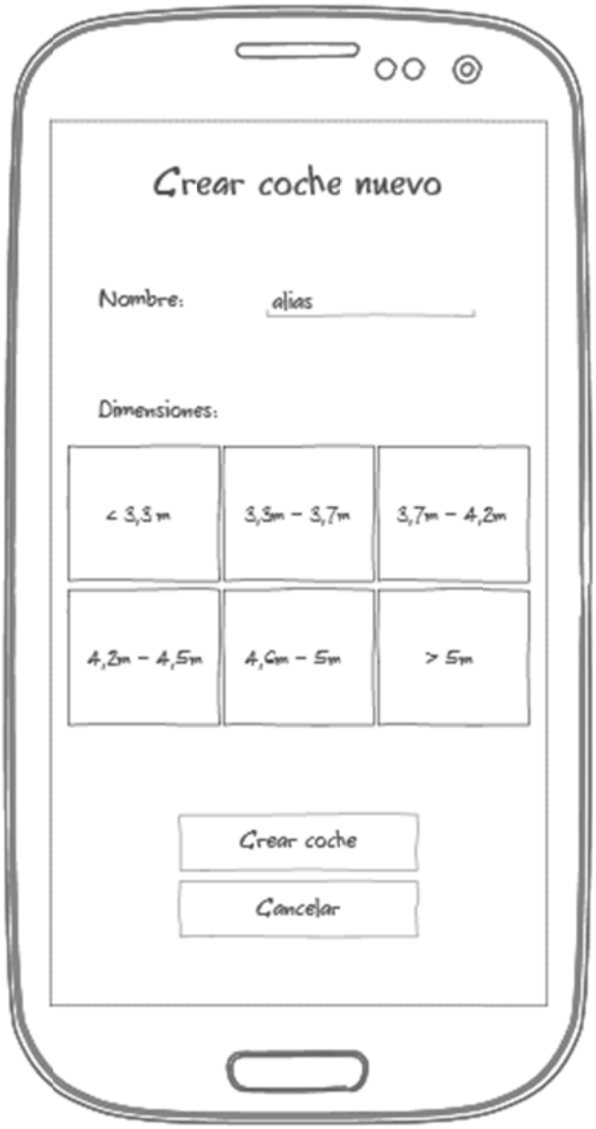
En esta pantalla el usuario puede seleccionar rápidamente el coche que quiera utilizar. Si no desea seleccionar ninguno podrá salir mediante el botón *Cancelar.*



F. Pantalla de selección de coche.

7.1.6. Crear coche nuevo.

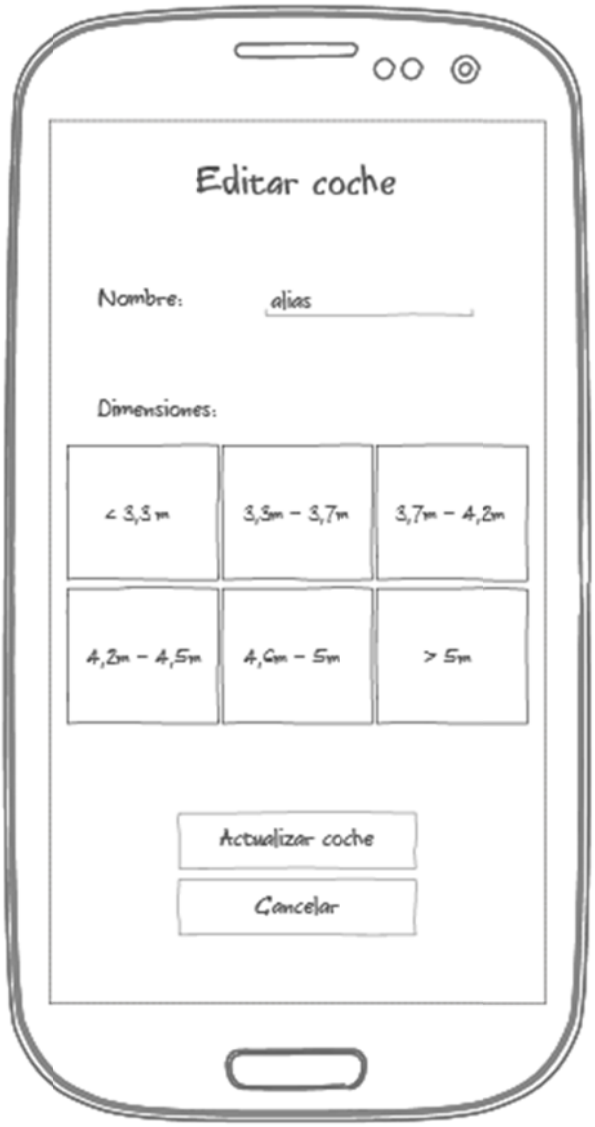
En esta pantalla el usuario podrá insertar rápidamente un nombre o alias para el coche que va a crear y el tamaño del mismo, seleccionándolo en uno de los botones mostrados en forma de matriz.



G. Pantalla de crear coche nuevo.

7.1.7. Editar coche.

En esta pantalla el usuario puede insertar los datos que quiera modificar o salir sin realizar ningún cambio, aunque haya escrito en algún campo. Podrá modificar el nombre y/o las dimensiones del coche.



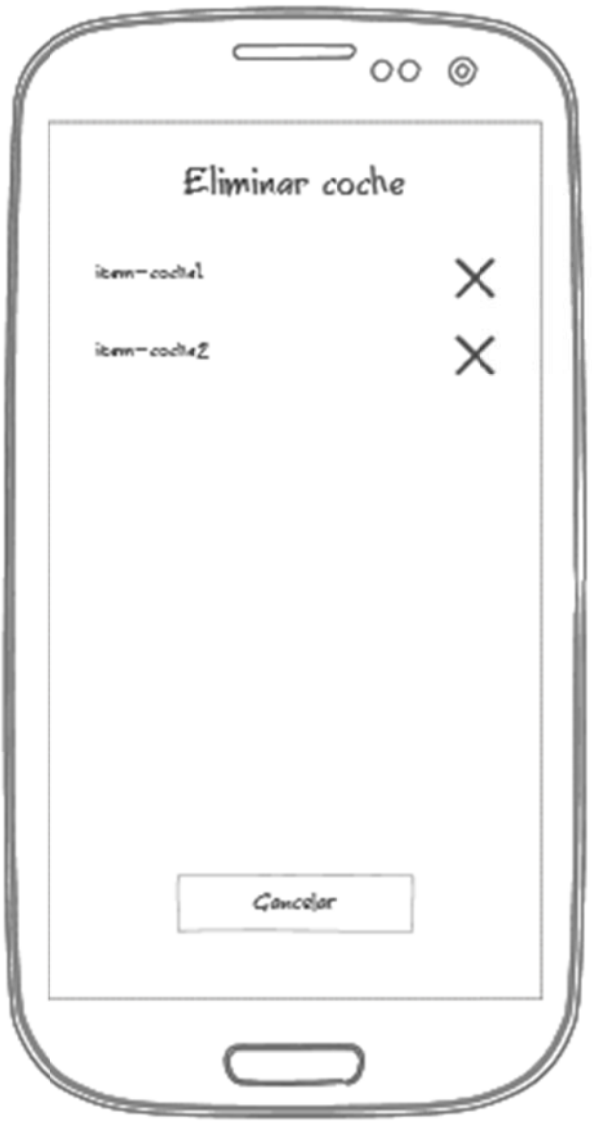
H. Pantalla de actualización del coche seleccionado.

Además, por seguridad se le pedirá al usuario confirmación de que desea actualizar el coche, evitando así errores o equivocaciones.

I. Ventana de confirmación de actualización del coche.

7.1.8. Eliminar coche.

En esta pantalla el usuario únicamente seleccionará el coche que quiera eliminar del listado o salir simplemente pulsando en el botón *Cancelar*.



J. Pantalla de eliminación de coche

Además de pulsar el coche que desea eliminar el usuario, obtendrá un mensaje de confirmación para evitar borrados por equivocación.

K. Ventana de confirmación de eliminar coche.

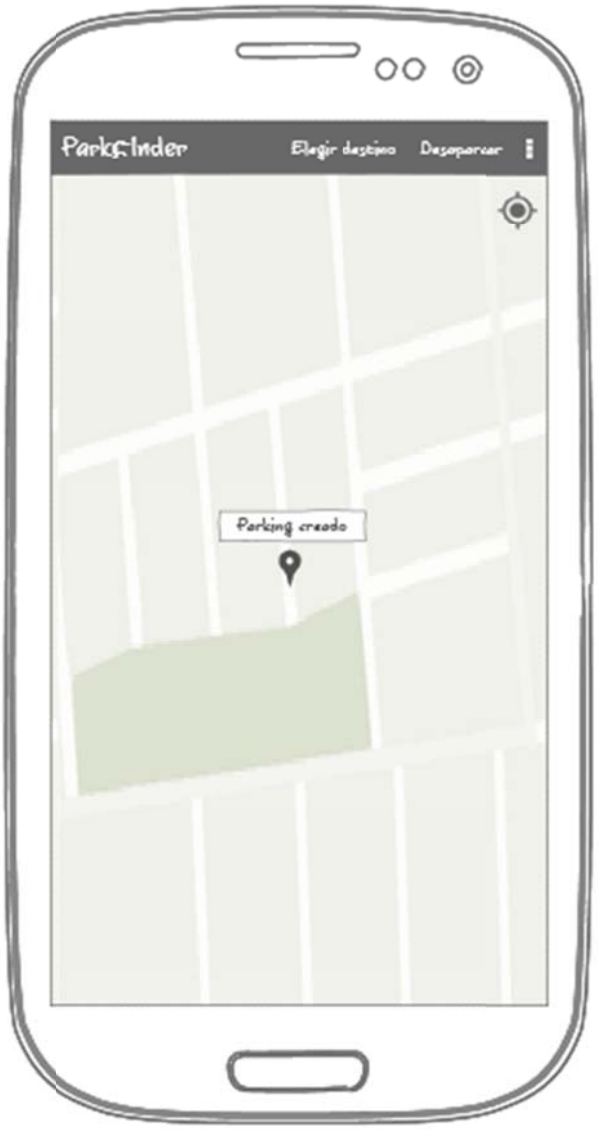
7.1.9. Opción principal: Desaparcar.

Cuando el usuario pulse en el botón *Desaparcar* aparecerá una ventana donde podrá seleccionar las opciones siguientes:

- Aparcamiento en línea.
- Aparcamiento en batería.
- Cancelar.

L. Ventana de selección de tipo de aparcamiento.

Después de seleccionar uno de los dos tipos de aparcamiento, la pantalla principal de la aplicación mostrará un icono en el lugar donde se ha creado el nuevo aparcamiento libre.



M. Pantalla principal con el nuevo aparcamiento creado.

7.1.10. Opción principal: Buscar destino.

Cuando el usuario pulse en *Elegir destino*, accederá a una ventana donde podrá insertar la dirección de destino. Después tendrá que pulsar el botón *Buscar destino* o el botón *Cancelar*.

N. Ventana de búsqueda de destino.

7.1.11. Pantalla con aparcamientos libres.

El usuario obtendrá la posición de destino en la pantalla principal junto con los aparcamientos libres. Al pulsar en cada uno de los iconos de los aparcamientos obtendrá un cuadro de texto informativo con la probabilidad, la distancia al destino y el tiempo que ha pasado desde que se creó el nuevo aparcamiento.

O. Pantalla principal con los aparcamientos libres.

En el momento en el que el usuario se encuentre físicamente en el lugar donde marca el mapa que hay una plaza de aparcamiento, pulsará sobre alguno de los cuadros de texto y aparecerá una ventana de confirmación preguntando si desea aparcar en ese aparcamiento.

P. Confirmación de ocupación de la plaza de aparcamiento.

***NOTA IMPORTANTE**:*

Para no extender demasiado el número de prototipos en esta memoria, no se han mostrado los mensajes de retroalimentación, información de errores, control de consistencia en formularios, etc. (hay una gran cantidad). Se mostrarán algunos de ellos más tarde en la aplicación implementada. En Android estos mensajes se implementan mediante mensajes Toast (Ver apartado "Arquitectura de la aplicación").

7.2. Escenarios de uso.

Siguiendo los escenarios generales del apartado anterior junto con los prototipos diseñados, se van a definir unos escenarios intermedios y específicos a partir de los prototipos realizados.

7.2.1. Escenarios de uso intermedio.

Escenario Intermedio: Indicar cómo el usuario deja un aparcamiento libre

David se dispone a arrancar el coche para llevar a su hijo a la guardería y después ir a trabajar. Justo antes de arrancar, coge su *Smartphone,* y accede a la aplicación. Se autentica en primer lugar y se propone indicar que va a dejar un hueco libre. Selecciona primero el coche que tiene creado y a continuación crea el nuevo aparcamiento libre. Confirma visualmente la acción realizada e inmediatamente apaga la pantalla del *Smartphone*. Ya está listo para arrancar el coche y dejar el aparcamiento libre.

Escenario Intermedio: El usuario busca aparcamiento libre en un destino

David recoge a su hijo de la guardería. Antes de arrancar, coge su *Smartphone* y accede a la aplicación. Primero se autentica y justo después selecciona el coche que ya tiene creado. Ahora puede buscar los aparcamientos libres que existan en la zona por donde vive, para ello escribe su dirección de destino y espera a que la aplicación le muestre el resultado. Observa sobre el mapa los distintos aparcamientos que han creado otros usuarios junto con la probabilidades que hay de encontrarlos libres. Se hace una idea de donde hay más posibilidades de encontrar aparcamiento, apaga la pantalla del móvil e inmediatamente arranca el coche con dirección al destino que tiene en mente.

Tras el viaje de regreso David llega con su coche al lugar donde vio más probabilidades de encontrar huecos para aparcar. Aparca en uno de ellos su coche y coge su *Smartphone.* Enciende la pantalla y abre la aplicación que se encontraba en segundo plano, visualiza el mapa y le indica a la aplicación que ha aparcado en un sitio libre. Espera la respuesta de la aplicación, que tras un breve instante le indica que su acción ha sido correcta. Ahora que ya ha terminado, guarda su *Smartphone* y se dispone a recoger a su hijo para irse a casa andando.

7.2.2. Escenarios de uso específico.

Escenario Específico: Indicar cómo el usuario deja un aparcamiento libre

David se dirige andando hacia su coche con su hijo en brazos. Cuando llega, abre la puerta trasera-derecha del coche y coloca a su hijo en su asiento. Cierra la puerta y se dirige inmediatamente hacia la puerta del conductor, y mientras la abre, va sacando el móvil del bolsillo. Se sienta en el asiento y se pone a buscar con el dedo el botón de encender la pantalla mientras que con la otra mano está colocándose el cinturón de seguridad. Una vez tiene encendida la pantalla, selecciona el icono que lanza la aplicación. Al aparecer la pantalla inicial, introduce en la caja de texto del usuario su nombre de usuario e inmediatamente después introduce la contraseña en la siguiente caja de texto. Pulsa sobre el botón de *Login* y espera a que la aplicación le muestre el mapa con su posición actual. Es ahora cuando selecciona en el menú la acción de seleccionar coche. En la vista nueva que aparece, selecciona el coche que ya tenía creado del listado y que se corresponde con el que va a empezar a conducir. La aplicación vuelve a mostrar el mapa donde se encuentra David en esos momentos. De manera sencilla pulsa en el botón de *Desaparcar* y sobre la ventana de diálogo que aparece, selecciona la opción de 'Aparcamiento en línea' entre las dos opciones que tiene (la otra es 'Aparcamiento en batería'). Ahora espera a que la aplicación le muestre un icono sobre el mapa indicándole que ha creado un aparcamiento nuevo. En ese momento, vuelve a pulsar el botón de encendido del *Smartphone* y la pantalla se apaga. Deja el móvil en la guantera del coche y se dispone a introducir la llave en el contacto para arrancarlo.

Escenario Específico: El usuario busca aparcamiento libre en un destino

David recoge a su hijo de la guardería tras regresar del trabajo. Antes de arrancar el coche con destino a la zona por donde vive, enciende la pantalla del *Smartphone* y selecciona el icono de la aplicación. En la primera pantalla que aparece, introduce en la caja de texto del usuario su nombre de usuario e inmediatamente después introduce la contraseña en la siguiente caja de texto. Pulsa sobre el botón de *Login* y espera a que la aplicación le muestre el mapa con su posición actual. Es ahora cuando selecciona en el menú la acción de seleccionar coche y la aplicación le muestra una nueva vista con un listado de coches. En este caso solo aparece el único que tiene creado, lo selecciona y después se dispone a insertar la dirección de destino. Para ello, selecciona el botón cuyo texto descriptivo es "Buscar destino". Pulsa en él e introduce la dirección donde vive en el campo de texto de la ventana de diálogo que le muestra la aplicación. Tras un breve instante la aplicación muestra los aparcamientos libres en la zona de destino. David selecciona con el dedo uno de ellos para visualizar la probabilidad que tiene de encontrarlo libre cuando llegue, la distancia que separa dicho aparcamiento de su destino y el tiempo que ha transcurrido desde que se quedó libre el aparcamiento. Ahora puede dejar el *Smartphone* en la guantera e insertar la llave para arrancar el coche. Tras unos minutos de viaje, se da cuenta de que todavía está sin ocupar el sitio libre que visualizó sobre el mapa instantes atrás. Se dirige hacia el lugar y aparca el coche. Apaga el motor e inmediatamente después coge el *Smartphone.* Es ahora cuando David enciende la pantalla de nuevo y abre la aplicación que estaba ejecutándose en segundo plano. Visualiza sobre el mapa el icono que le indica la posición actual y pulsa sobre el marcador próximo. Inmediatamente la aplicación le muestra a David un mensaje indicándole que si es cierto que va a ocupar una plaza de aparcamiento libre. Le indica que '*Si*' pulsando sobre el botón correspondiente y espera a que la aplicación le de alguna respuesta. Tras un breve lapsus de tiempo, la aplicación le indica por pantalla que ha aparcado correctamente. El indicador del hueco libre desaparece de la pantalla y David tras observar esto ya puede apagar la pantalla y salir del coche para recoger a su hijo.

7.3. Evaluación heurística.

Para realizar la evaluación heurística se han estudiado los prototipos mediante una guía de usabilidad de aplicaciones móviles (*https://pshyama.wordpress.com/2010/06/17/mobile-user-interface-and-usability-guide/*). Dicha guía marca las pautas a seguir para realizar un diseño usable a través de los siguientes principios básicos:

- **Facilidad de aprendizaje**
  - La interfaz debe ser fácil de usar desde el primer momento.
  - Debe limitarse la exposición de gran cantidad de datos y reducirlo a lo que el usuario requiere.

La interfaz es fácil de manejar, realizando las transiciones mediante la pulsación de botones sin necesidad de leer gran cantidad de datos.

- **Eficiencia**
  - El número de pasos que llevan al usuario a alcanzar un objetivo es importante.
  - Las tareas clave deben ser lo más eficientes posibles.

Se han minimizado al máximo el número de pulsaciones que debe realizar el usuario para conseguir sus objetivos, ahorrando pulsaciones innecesarias.

- **Retención en memoria**
  - La interfaz debe ser lo más fácil posible de utilizar cada vez que el usuario interactúe con ella.
  - La frecuencia de uso es la clave en la retención en memoria.

Al ser sencilla de entender, tras varios usos la interacción es muy fluida.

- **Recuperación ante errores**
  - La interfaz debe ser diseñada para que el usuario no cometa errores.
  - Si comete algún error, la interfaz debe proporcionar la oportunidad de rectificar el error.
  - El usuario debe ser informado cuando se produzcan situaciones de error sin ser incomodado y sin que se pierda información.

Se ha añadido control de consistencia en los formularios para que el usuario no cometa errores de creación. En caso de cometerlos la aplicación avisa al usuario para que los corrija.

- **Simplicidad**
  - La tarea habitual debe ser la más fácil, mientras que la menos habitual debe ser posible.
  - Evitar funcionalidad innecesaria y mantener el diseño visual descongestionado.
  - La información debe estar eficientemente utilizable en pantallas pequeñas.
  - El usuario debe estar habilitado para encontrar todas las funcionalidades fácilmente y realizar las acciones de manera efectiva, sin complicarse en cosas secundarias.
  - Los diseños deben ser ordenados. Los textos deben ser claros y cómodos de leer, sin decoración excesiva.
  - La transición entre vistas, la interacción y los gráficos deben ser armoniosos.

Todas las tareas de inserción de datos son sencillas, no requieren entradas de texto largas, ajustándose a lo estrictamente necesario para que el usuario consiga el objetivo perseguido.
Las funcionalidades están repartidas en dos menús: Una barra de menús superior, con opciones principales, y otra deslizable con opciones secundarias o menos importantes.
Los textos tienen el tamaño y contraste suficiente para leerse adecuadamente en cualquier contexto, incluso en el interior de un coche y a plena luz solar.
Las transiciones entre vistas son suaves y sin apenas retardos (*lag*).

- **Consistencia**
  - Debe ocurrir lo que el usuario espera cuando interactúa con la interfaz.

La aplicación realiza lo que el usuario le ordena mediante sus gestos.

- **Visibilidad**
  - La información importante debe ser la más visible y la menos importante la menos visible.
  - Entender los objetivos del usuario es crítico.

La información importante aparece resaltada mediante contraste de colores. Por ejemplo, botones de login, crear usuario, crear coche, etc.

- **Retroalimentación**
  - El usuario debe estar siempre en situación de control de la interfaz y actuar de la misma manera a lo largo de toda la aplicación.
  - El uso de colores debe ser consistente a lo largo de toda la aplicación.

En todo momento el usuario puede interactuar con la interfaz, salvo en aquellos momentos en los que se produce una comunicación con el servidor de la base de datos. En esos momentos se muestran diálogos de progreso para retroalimentar al usuario e indicarle que espere unos momentos.
Se ha empleado la misma tonalidad de colores a lo largo de toda la aplicación y coloreando elementos comunes con el mismo color de fondo y de texto.

- **Intuición**
  - Una interfaz debe ser lo más intuitiva y sencilla posible.
  - Debe estar basada en elementos que el usuario ya conozca sin la necesidad de comprender procedimientos complicados.

Se han utilizado componentes gráficos y textos con los que el usuario está familiarizado. Los iconos son intuitivos y acordes a la acción que ejecutan.

- **Entradas de texto del usuario**
  - Proveer valores por defecto en las cajas de texto.

En este caso no ha sido necesario puesto que no se repiten los valores introducidos por los usuarios.
Por otro lado, se han insertado elementos de tipo *placeholder*, que escriben en el interior de la caja de texto una descripción de lo que el usuario debe escribir. En el momento que el usuario tiene el foco de la caja de texto, desaparece el texto del *placeholder* y el usuario puede escribir desde cero.

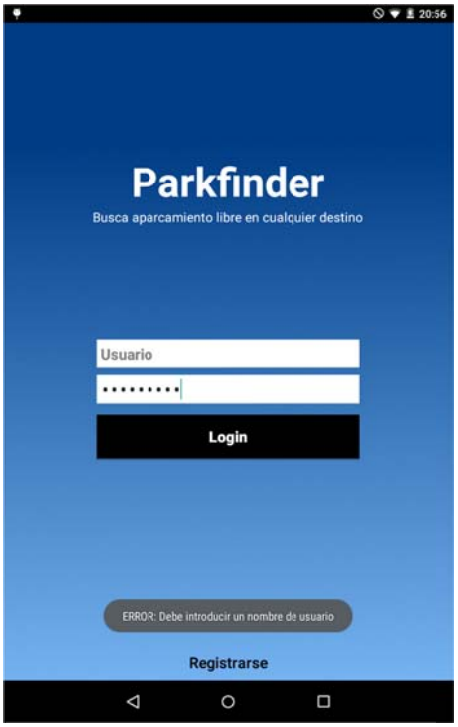
- **Satisfacción**
  - El usuario debe disfrutar usando la aplicación.

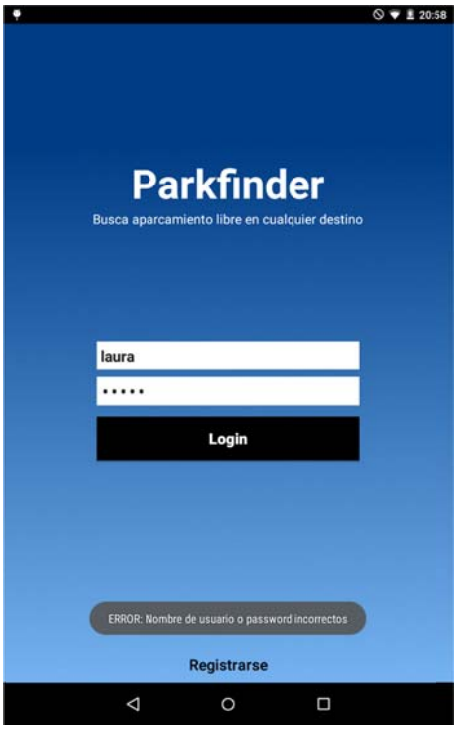
El sistema de puntuaciones de la aplicación junto con la visualización de mapas hace que sea divertido utilizarla. La parte de creación y edición de coches no lleva mucho tiempo. El usuario tiene libertad en todo momento para elegir las opciones de aparcamiento que más le guste (teniendo en cuenta las prioridades visualiza más o menos aparcamientos).

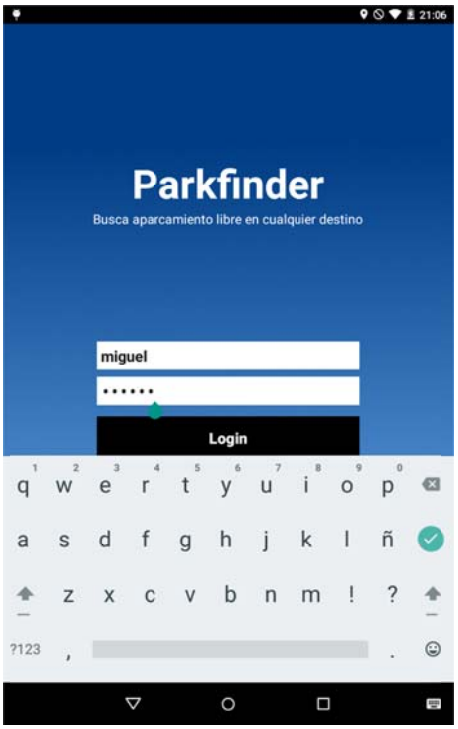
8. Aplicación final

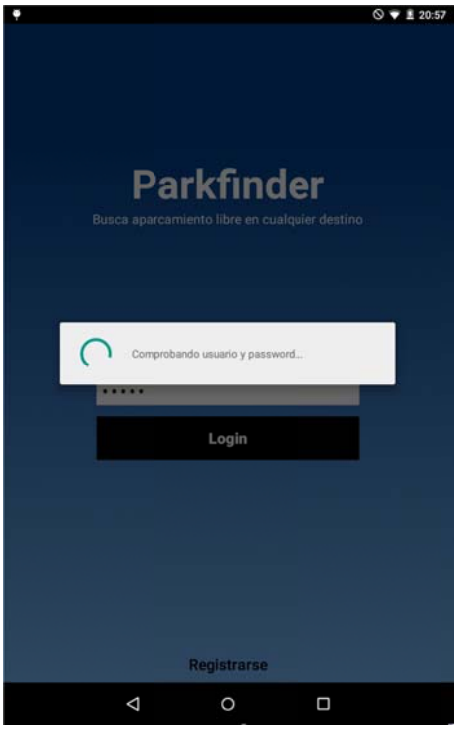
A continuación se van a mostrar las pantallas definitivas de la aplicación, creadas a partir del diseño de prototipos. Además se van a mostrar algunos de los mensajes de retroalimentación y de consistencia que le ofrece la aplicación al usuario.

8.2. Pantalla de *login*.

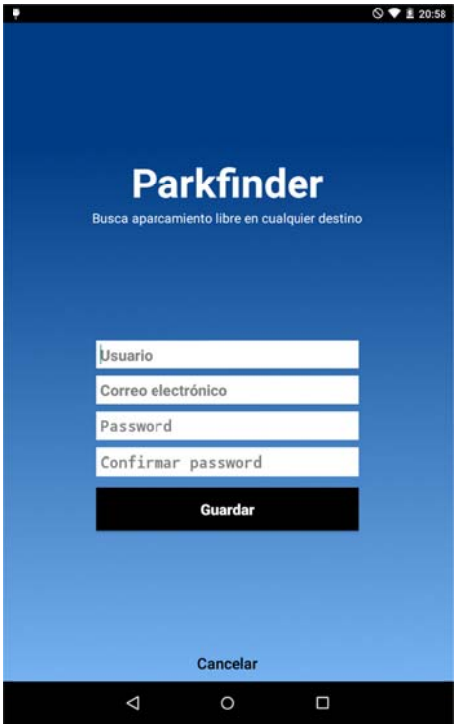


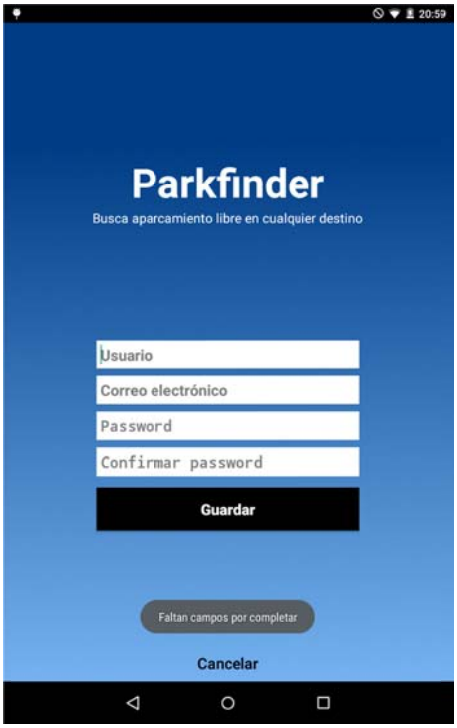


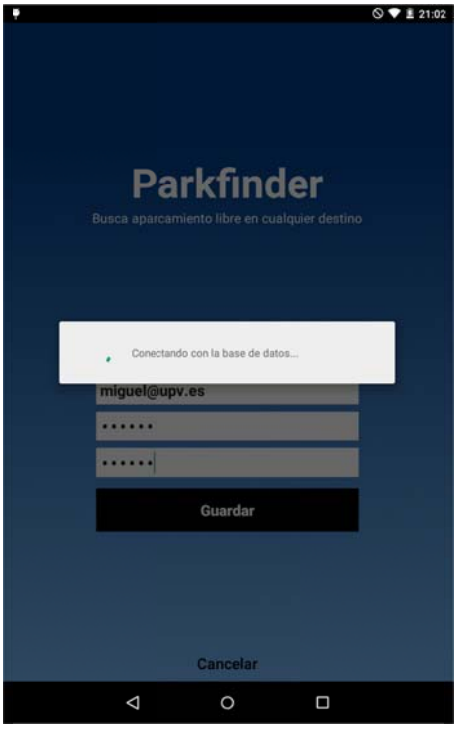


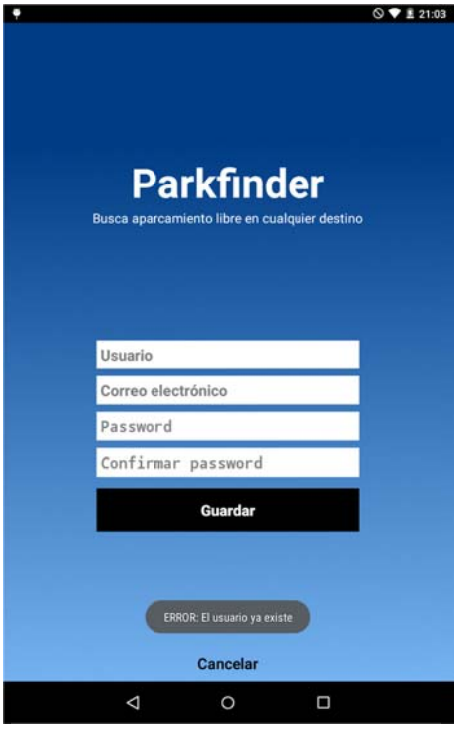


8.2. Pantalla de registro de usuario.

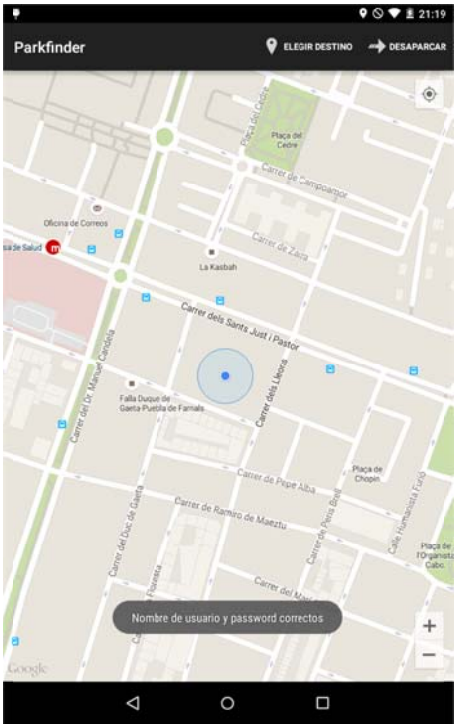




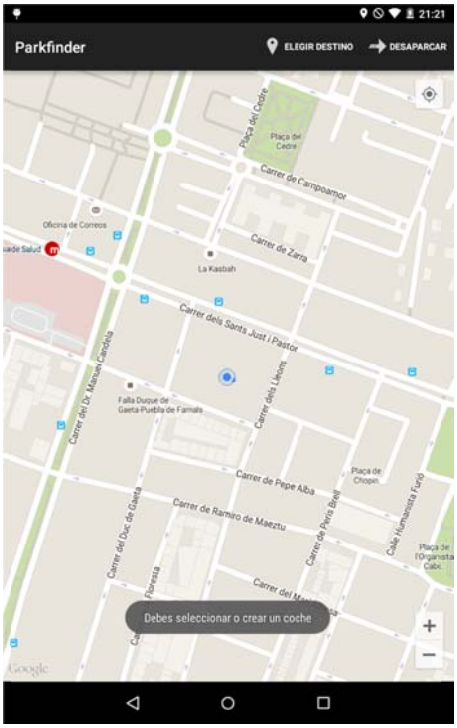




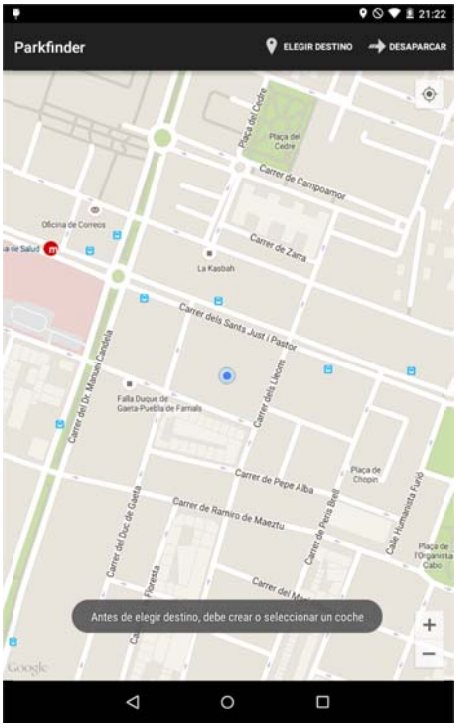
8.3. Pantalla principal de la aplicación.



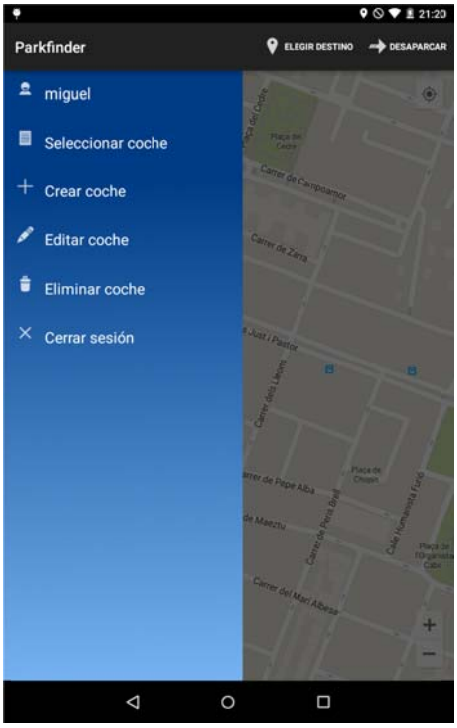
Cuando el usuario se autentica correctamente aparece el mensaje que se ve en la imagen.



En caso de no tener ningún coche seleccionado o creado, al pulsar el botón "Desaparcar" devuelve un error.

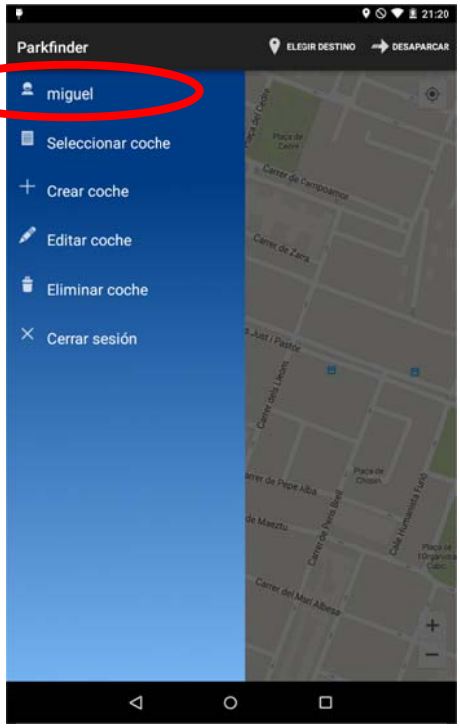


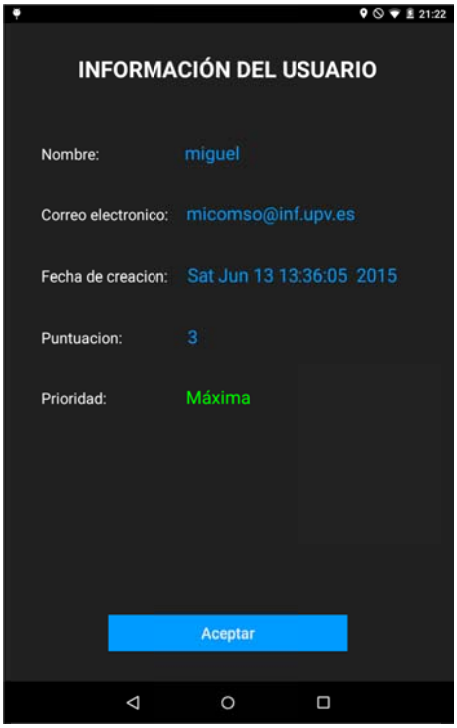
En caso de no tener ningún coche seleccionado o creado, al pulsar el botón "Buscar destino" devuelve un error.



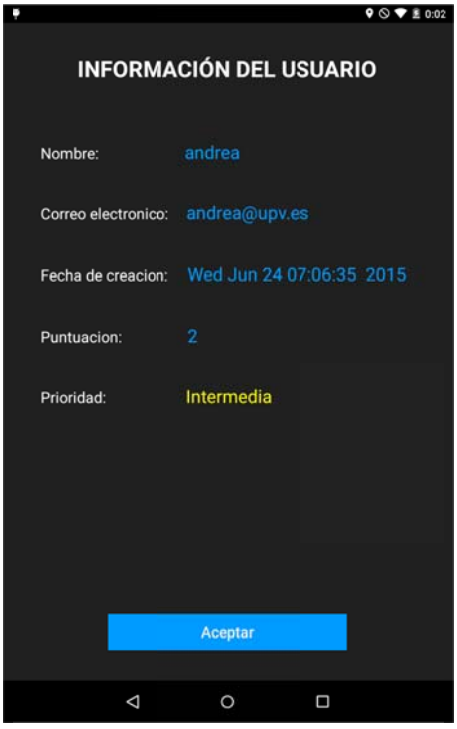
Al deslizar el dedo desde la parte izquierda hacia la derecha aparece el menú deslizable con las opciones secundarias del usuario.

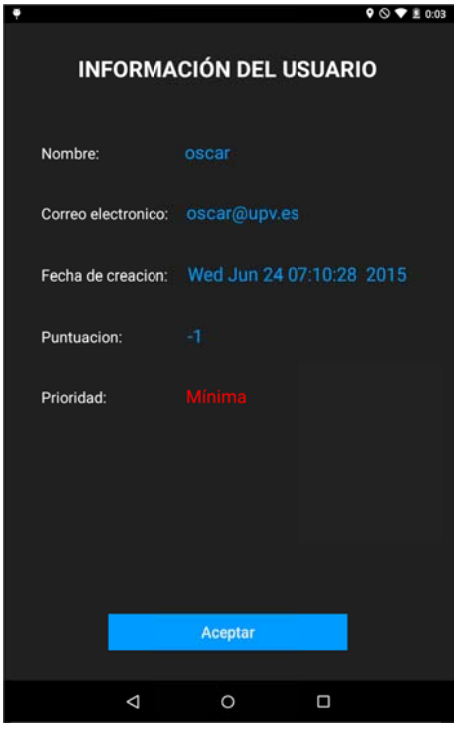
8.4. Información del usuario.





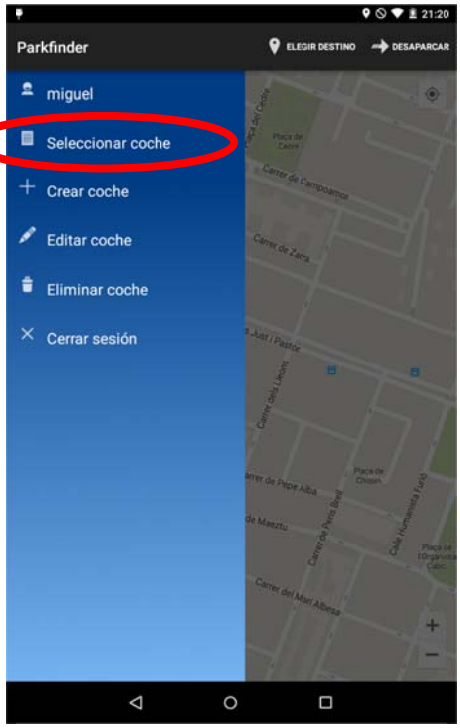
La primera opción del menú deslizable muestra una ventana informativa. Destaca la prioridad máxima en color verde (ver el Anexo de Manual de Usuario).

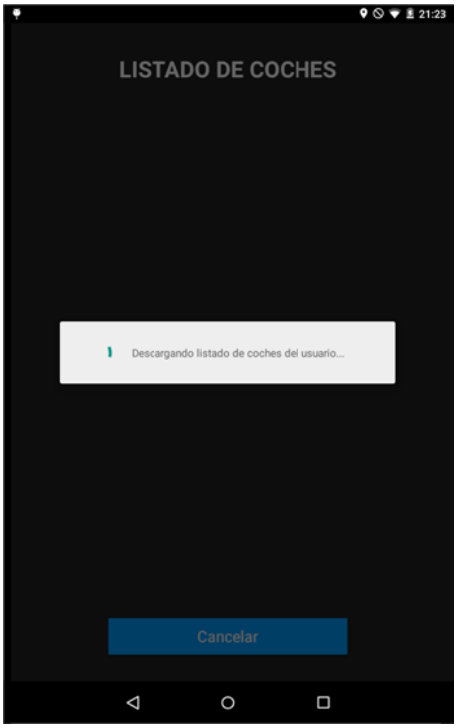


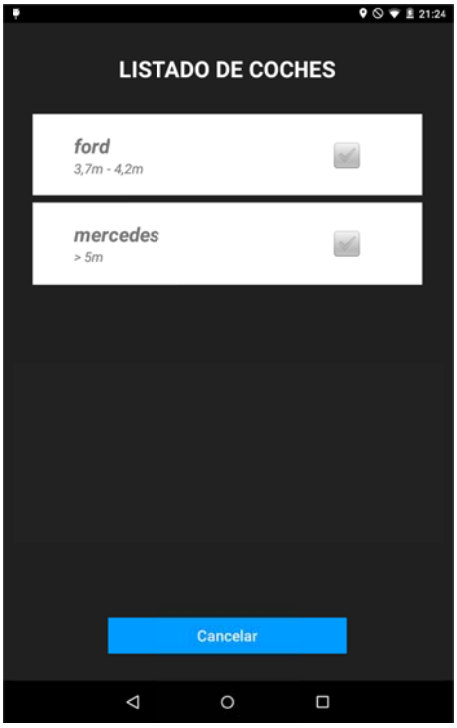


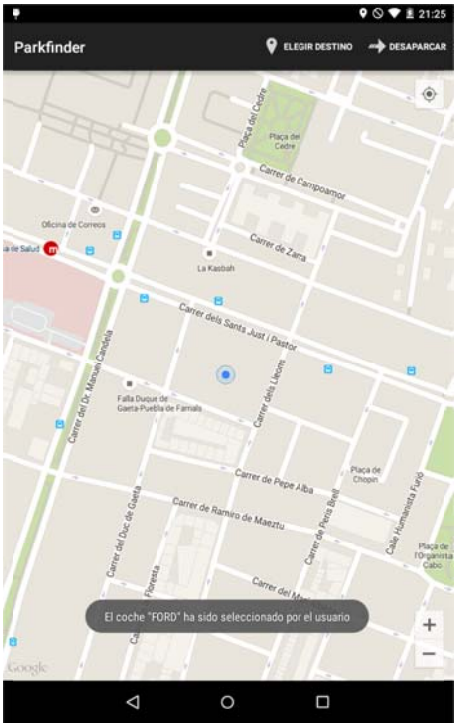
En estas dos últimas vistas, se han puesto de ejemplo dos usuarios más con prioridades distintas al primero, uno con prioridad Intermedia y otro con prioridad Mínima (Ver Anexo de Manual de Usuario).

8.5. Seleccionar coche.

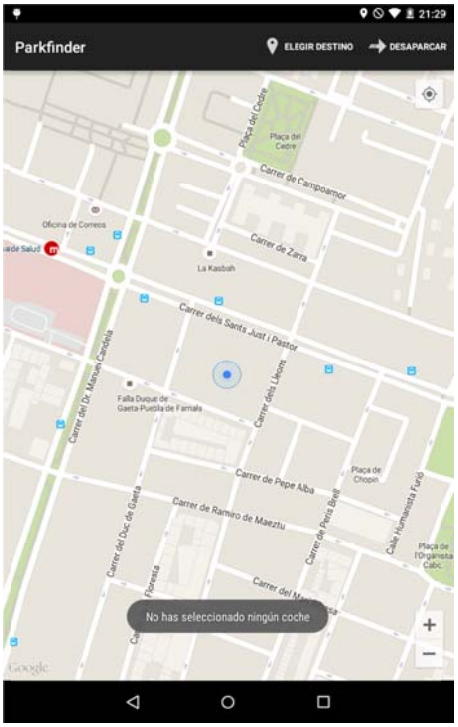






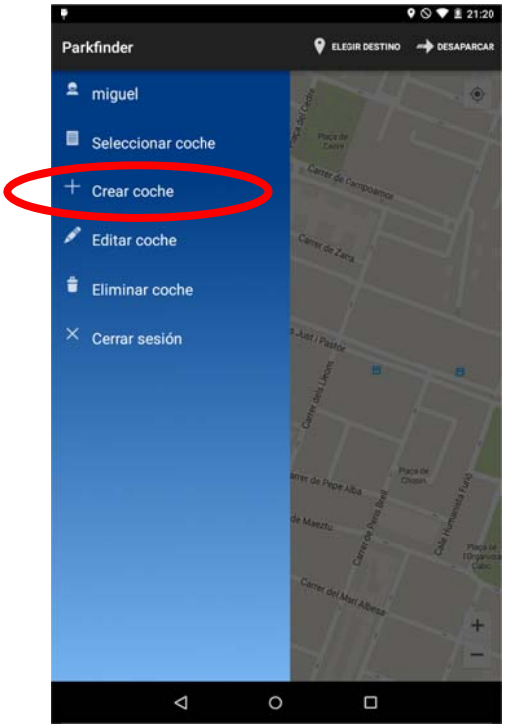


Mensaje de consistencia cuando el usuario elige un coche de la lista de selección.

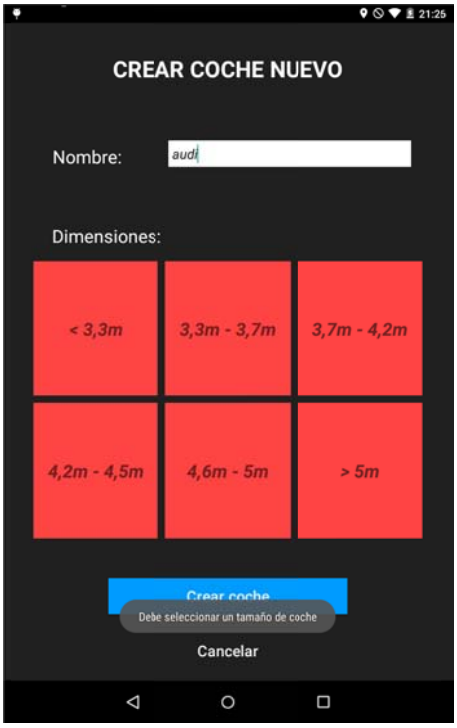


Mensaje de consistencia cuando el usuario no elige ningún coche de la lista de selección.

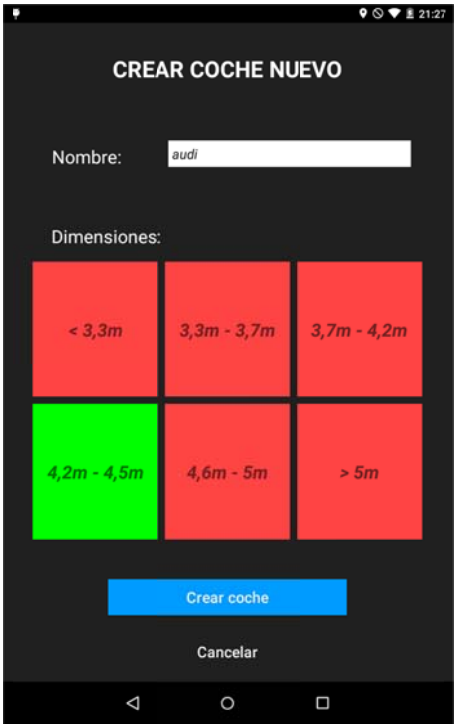
8.6. Crear coche.



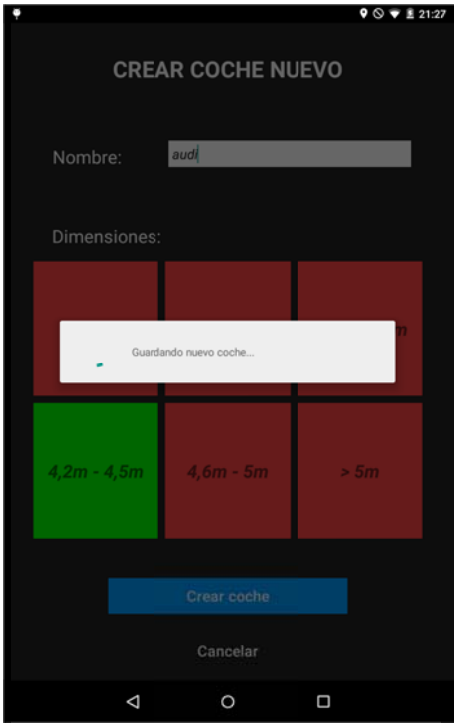
Mensaje de consistencia cuando el usuario pulsa el botón "Crear coche" y no ha escrito todavía un nombre.



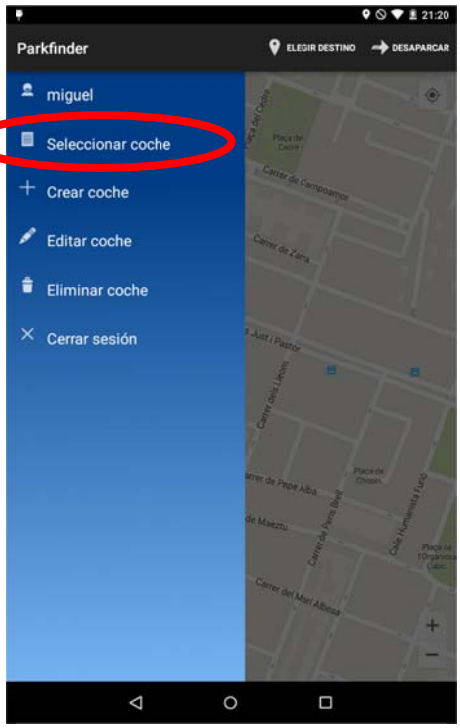
Mensaje de consistencia cuando el usuario pulsa el botón "Crear coche" y no ha seleccionado uno de los botones de dimensiones.



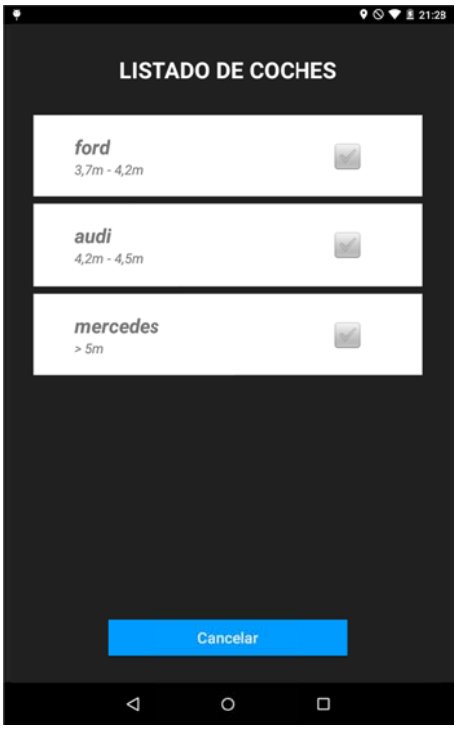
Cuando el usuario selecciona un botón de dimensiones se pinta de verde.



Diálogo de progreso indicando que se está accediendo a la base de datos para guardar el coche creado.



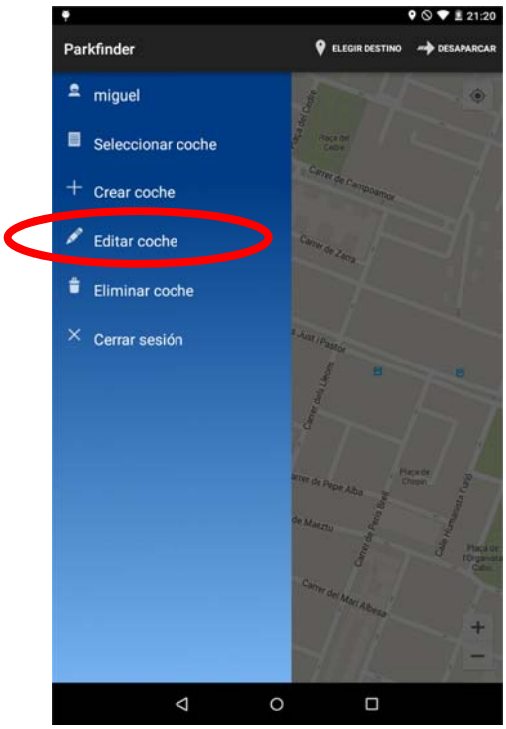
Comprobación de que el coche creado aparece como seleccionable.



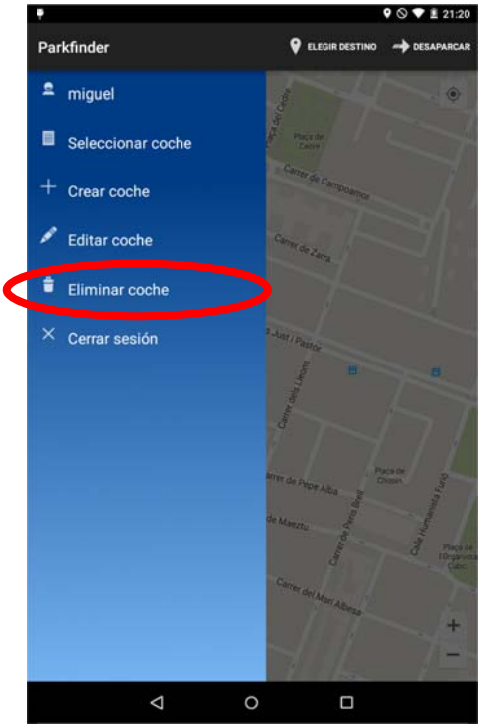
Comprobación de que el coche creado se ha insertado en la base de datos correctamente y se muestra en el listado cargado.

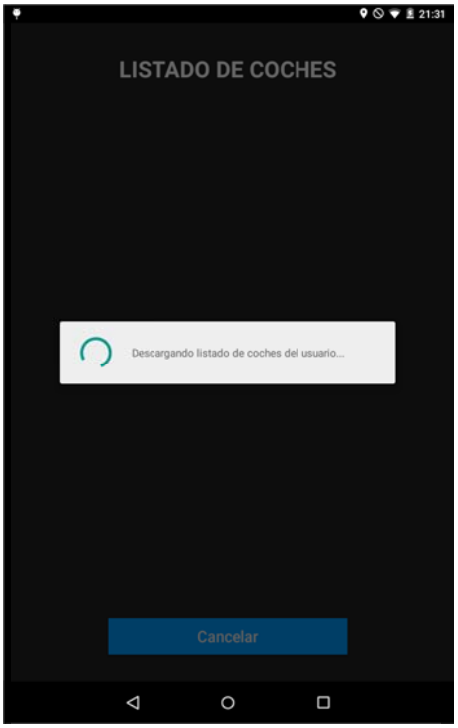
8.7. Editar coche.

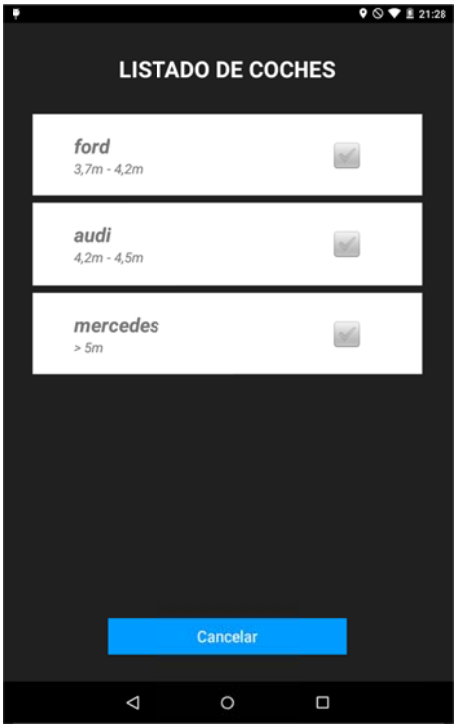
Por motivos de tiempo, se ha decidido no implementar esta opción. Simplemente es repetir lo realizado para otras operaciones como "*Crear coche*". Presentaría un formulario idéntico donde se cargarían los datos del coche seleccionado. El usuario podría cambiar algunos de sus parámetros y almacenar el coche modificado en la base de datos.



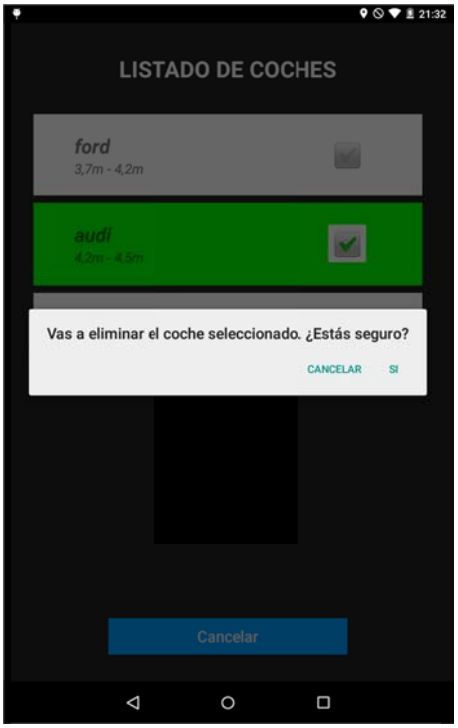
8.8. Eliminar coche.



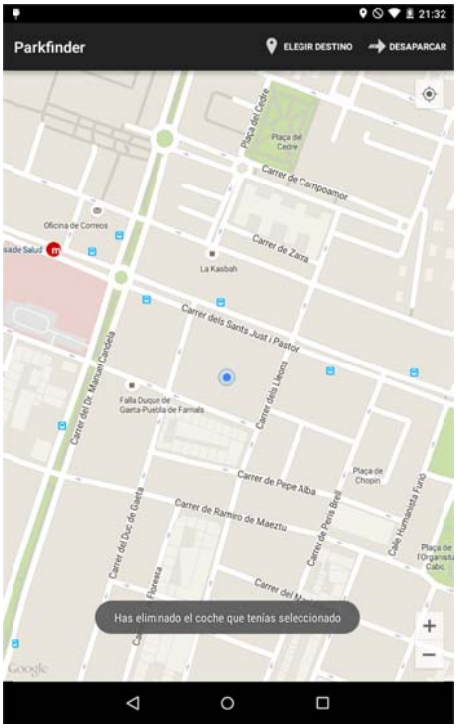




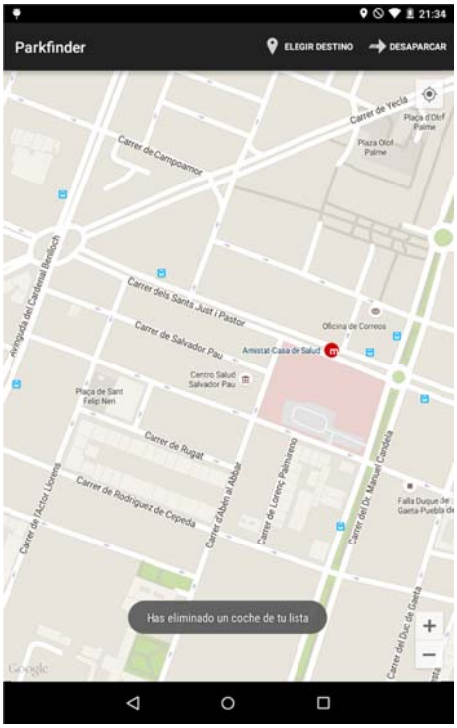
Se muestra una lista con los datos obtenidos de la base de datos.



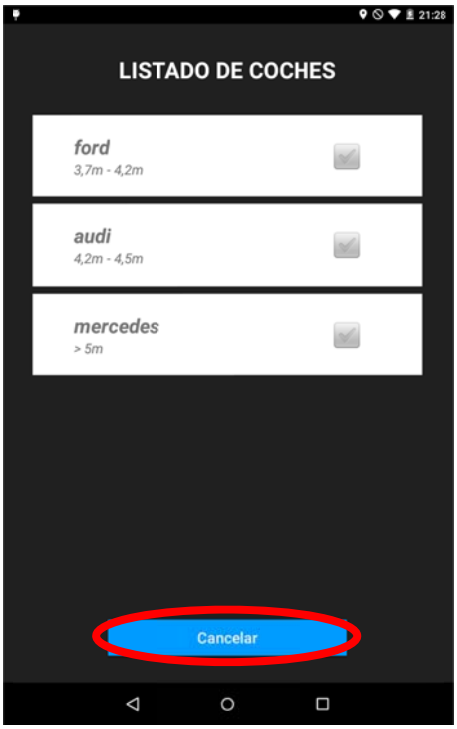
Ventana de confirmación para evitar errores al pulsar en la lista.



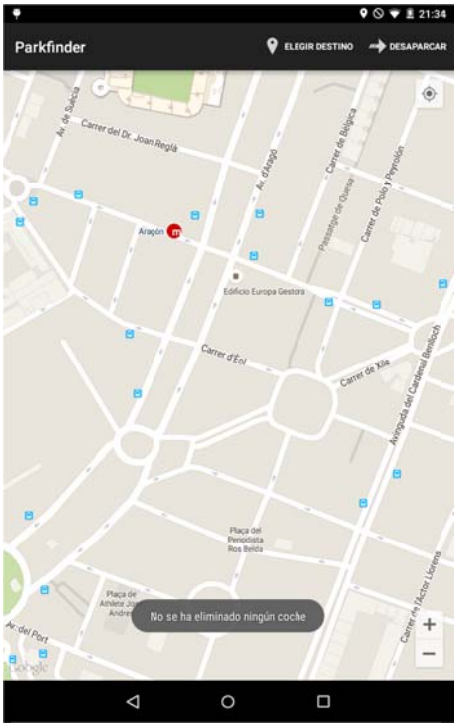
Cuando se elimina el coche que el usuario tenía seleccionado, aparece el mensaje.



Cuando se elimina un coche que el usuario no tenía seleccionado, aparece el mensaje.

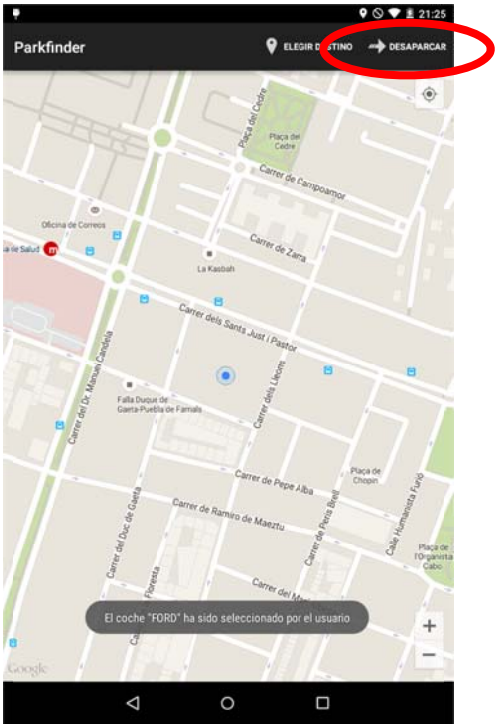


Si el usuario pulsa el botón "Cancelar"…

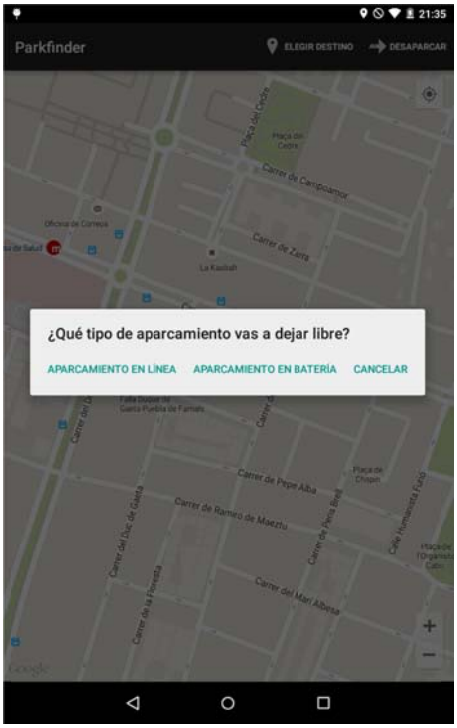


… aparece el mensaje informándolo

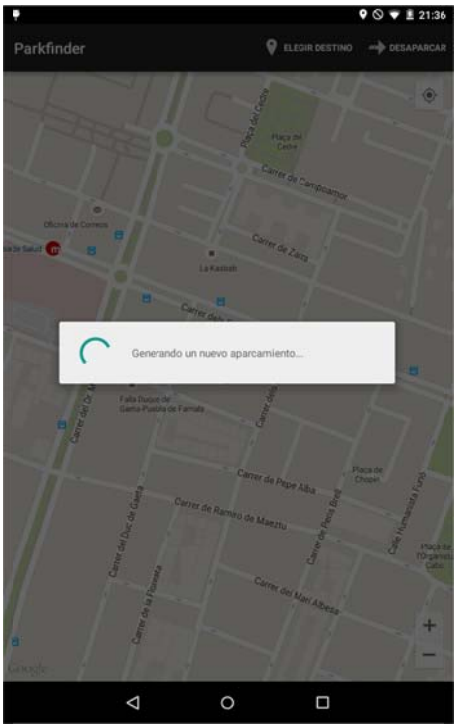
8.9. Desaparcar.



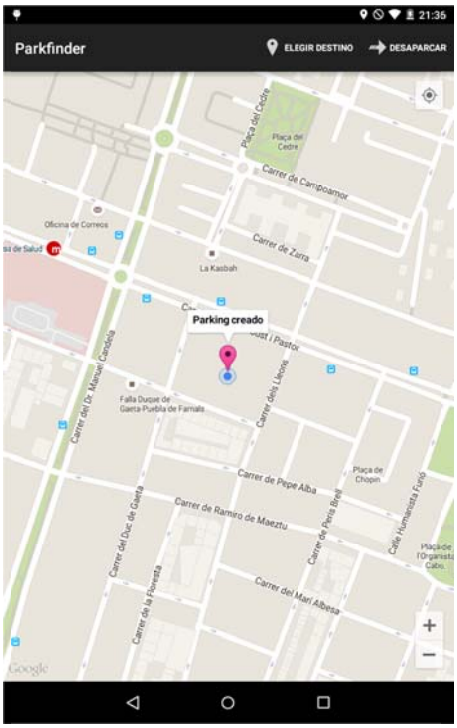
Cuando el usuario tiene un coche seleccionado puede crear un aparcamiento en "Desaparcar".



Aparece una ventana donde el usuario puede elegir el tipo de aparcamiento que va a dejar libre.

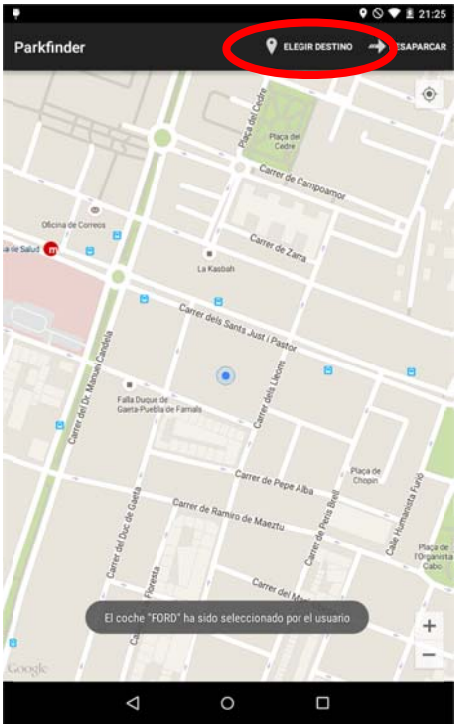


Ventana de progreso mientras la aplicación se conecta con la base de datos y almacena el nuevo aparcamiento.

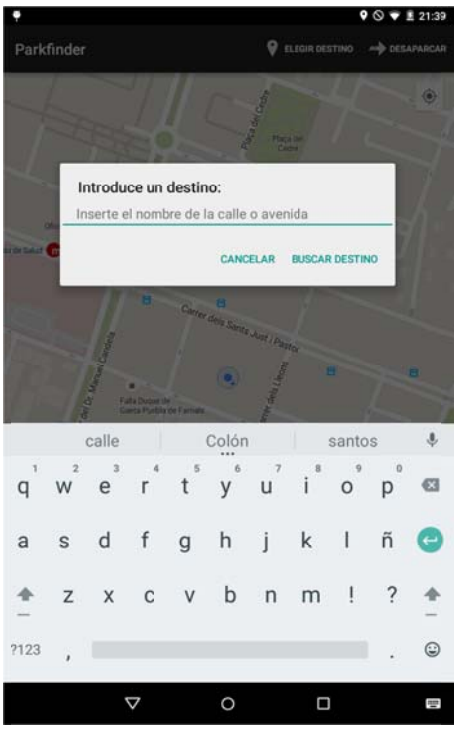


Si todo es correcto, aparece inmediatamente sobre el mapa el nuevo aparcamiento creado junto con un cuadro de texto indicándolo.

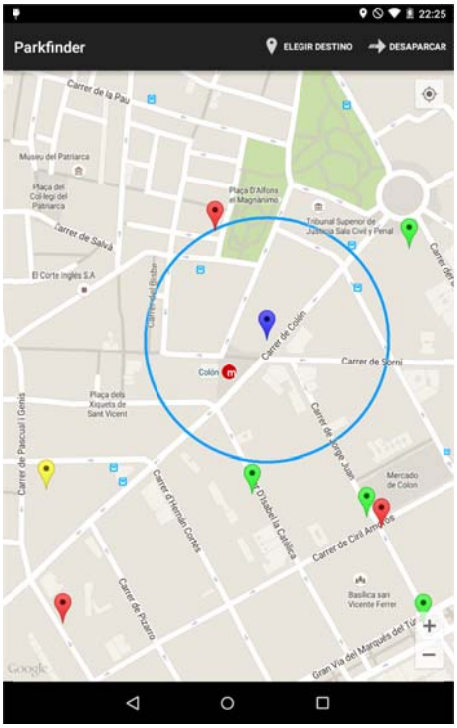
8.10. Buscar destino.



Cuando el usuario tiene un coche seleccionado puede buscar aparcamientos en un destino.

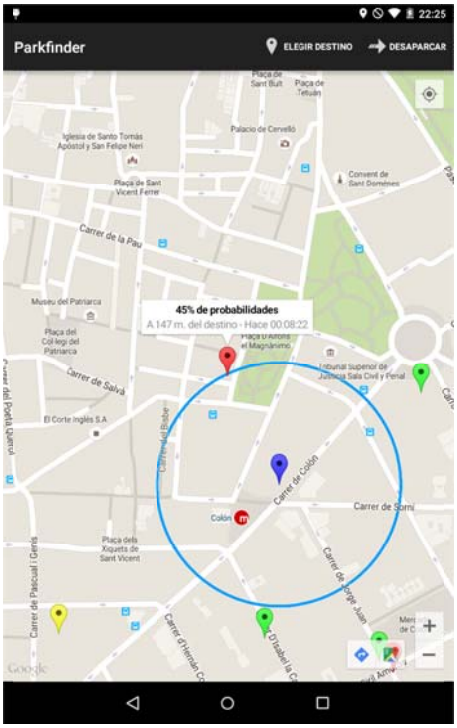


Aparece una ventana de diálogo donde el usuario puede introducir el nombre y número de calle, además del código postal.

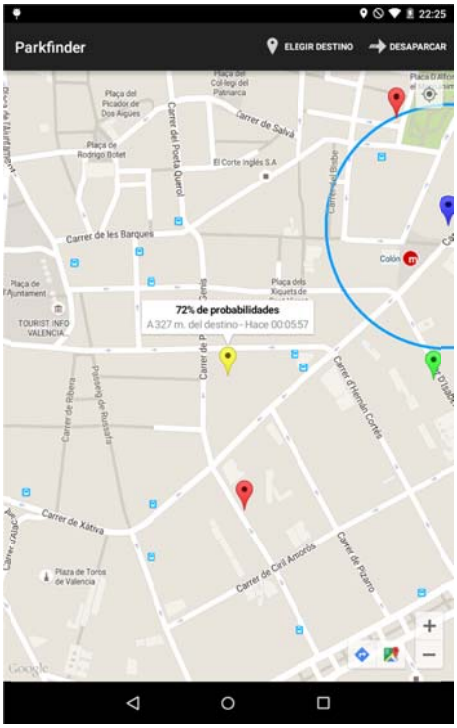


Aparecen los aparcamientos en diferentes colores junto con un círculo que marca la distancia al destino. El radio depende de la prioridad (Ver el Anexo de Manual de Usuario).

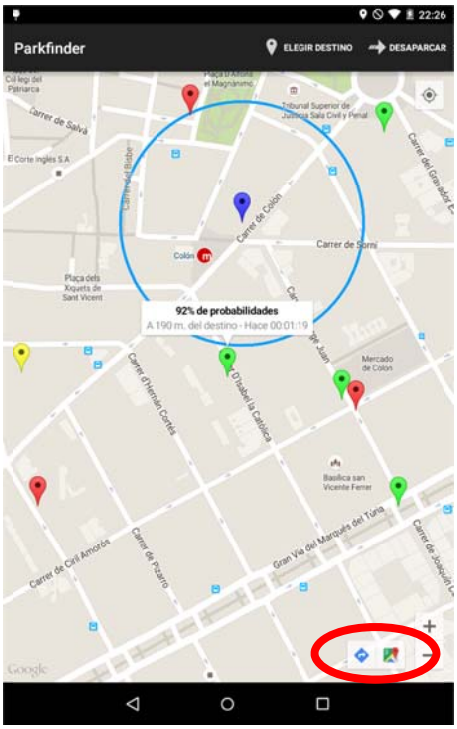
El usuario puede pulsar sobre los aparcamientos para obtener más información (prioridad, distancia al destino y tiempo de creación).



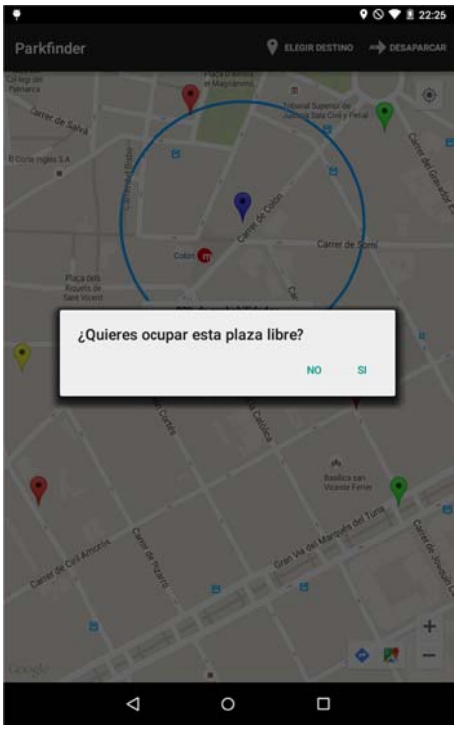
Los colores marcan la probabilidad de encontrar el aparcamiento libre.



El verde marca mayor probabilidad, el amarillo intermedia y el rojo la menor probabilidad.

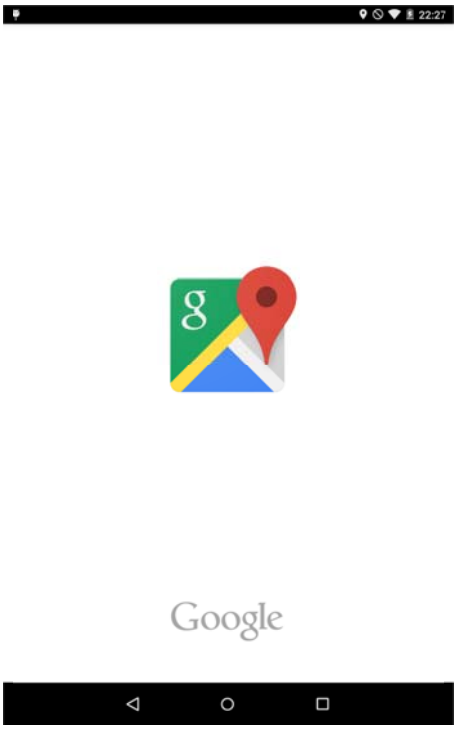


En la parte inferior derecha, cuando el usuario pulse sobre un aparcamiento aparecen las opciones nativas de la aplicación Google Maps.

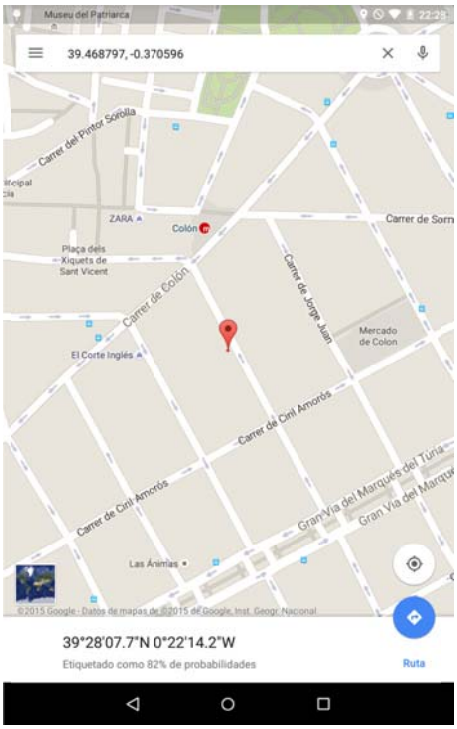


Si el usuario se encuentra en el lugar donde hay un aparcamiento libre, pulsando sobre el cuadro de texto obtiene una ventana de confirmación.

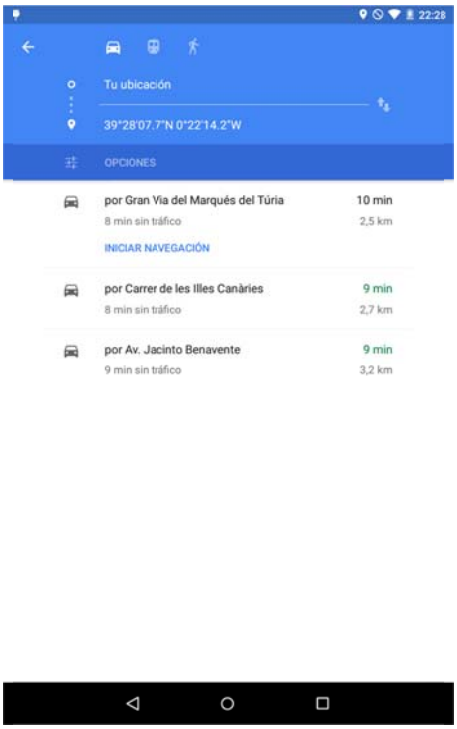
8.11. Aplicación nativa de Google Maps para Android.



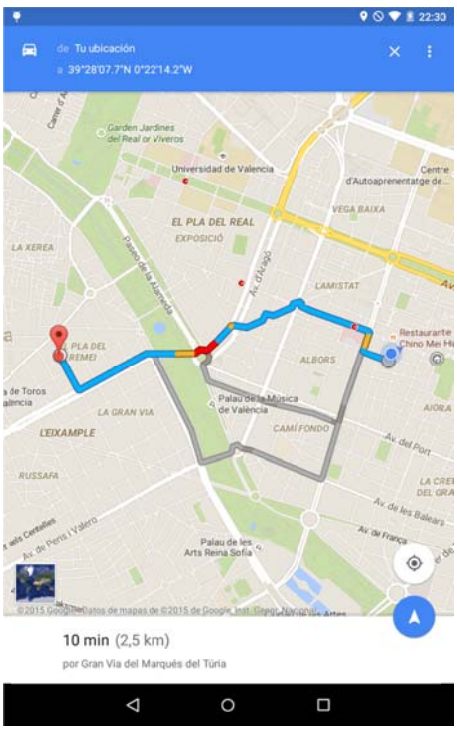
Al pulsar en los botones anteriormente nombrados, automáticamente se abre la aplicación Android de Google Maps.



Inmediatamente sitúa un marker en la posición donde se encuentra el aparcamiento que el usuario ha seleccionado.



Cabe la posibilidad de elegir entre varias rutas distintas a elección del usuario.



Al seleccionar una ruta se marca en el mapa dicha ruta indicando el tiempo y la distancia hasta alcanzar el objetivo.

ANEXOS

Anexo I. Manual de usuario.

En este Anexo se va a desarrollar un manual de usuario para aquellos usuarios que no han utilizado nunca la aplicación.

Los objetivos de la misma son:

- Encontrar aparcamientos libres en un destino junto con sus probabilidades.
- Dejar un aparcamiento libre para que otros usuarios puedan buscarlo con la aplicación.

REGISTRAR USUARIO

- Acceder a la aplicación y pulsar el botón *Registrarse* de la pantalla inicial.
- Insertar en los cuadros de texto el nombre de usuario, el mail y la contraseña (dos veces).
- Pulsar el botón *Crear usuario*.

AUTENTICAR USUARIO

- Acceder a la pantalla inicial de la aplicación.
- Insertar en los cuadros de texto el nombre de usuario y la contraseña.
- Pulsar en el botón *Login*.

CONSULTAR INFORMACIÓN DEL USUARIO

- Acceder a la pantalla principal de la aplicación autenticándose con un usuario válido.
- Arrastrar el dedo desde la parte izquierda de la pantalla principal hacia la derecha para que aparezca el menú deslizable.
- Seleccionar de este menú la primera opción, donde aparece escrito el nombre del usuario.
- El usuario puede consultar ahora la puntuación que tiene y la prioridad (Máxima, Intermedia o Mínima).

SELECCIONAR UN COCHE

- Acceder a la pantalla principal de la aplicación autenticándose con un usuario válido.
- Arrastrar el dedo desde la parte izquierda de la pantalla v hacia la derecha para que aparezca el menú deslizable.
- Seleccionar de este menú la segunda opción, donde aparece escrito *Seleccionar coche*.
- El usuario deberá tener creado un coche como mínimo para poder utilizar esta acción.
- En la nueva vista que aparece, seleccionar el coche de la lista que el usuario prefiera pulsando sobre el ítem de la lista.
- En caso de no querer seleccionar ningún coche, pulsar sobre el botón *Cancelar*.

CREAR UN COCHE

- Acceder a la pantalla principal de la aplicación autenticándose con un usuario válido.
- Arrastrar el dedo desde la parte izquierda de la pantalla principal hacia la derecha para que aparezca el menú deslizable.
- Seleccionar de este menú la tercera opción, donde aparece escrito *Crear coche*.
- En la nueva vista que aparece insertar el nombre o alias del nuevo coche y seleccionar uno de los cuadros donde aparece inscrito el tamaño del coche que se va a crear.
- Por último, pulsar sobre el botón *Crear coche*.
- En caso de que el usuario no quiera crear ningún coche puede pulsar el botón *Cancelar*.

ELIMINAR UN COCHE

- Acceder a la pantalla principal de la aplicación autenticándose con un usuario válido.
- Arrastrar el dedo desde la parte izquierda de la pantalla principal hacia la derecha para que aparezca el menú deslizable.
- Seleccionar de este menú la quinta opción, donde aparece escrito *Eliminar coche*.
- El usuario deberá tener creado un coche como mínimo para poder utilizar esta acción.
- En la nueva vista que aparece el usuario puede seleccionar uno de los ítems de la lista.
- Se le pedirá al usuario confirmación mediante una ventana de diálogo.
- En caso de no querer eliminar el coche seleccionado, el usuario puede cancelar la operación pulsando en el botón *No* de la ventana de diálogo.

CERRAR SESIÓN DE USUARIO

- Acceder a la pantalla principal de la aplicación autenticándose con un usuario válido.
- Arrastrar el dedo desde la parte izquierda de la pantalla principal hacia la derecha para que aparezca el menú deslizable.
- Seleccionar de este menú la última opción, donde aparece escrito *Cerrar sesión*.
- El usuario volverá a la pantalla inicial y deberá volver a introducir unas credenciales si desea utilizar de nuevo la aplicación.

DESAPARCAR UN COCHE

- El usuario debe estar situado con su dispositivo en el interior del coche.
- Acceder a la pantalla principal de la aplicación autenticándose con un usuario válido.
- Arrastrar el dedo desde la parte izquierda de la pantalla principal hacia la derecha para que aparezca el menú deslizable.
- Pulsar en la opción de *Seleccionar un coche* (deberá tener creado al menos uno).
- En la nueva vista el usuario selecciona el coche en cuyo interior se encuentra en esos momentos.
- Pulsa sobre el botón *Desaparcar* que aparece en el menú superior de la pantalla principal.
- Aparece una nueva ventana de diálogo donde el usuario elige el tipo de aparcamiento que se dispone a dejar libre (en línea o en batería).
- Si desea echarse atrás puede cancelar la acción pulsando en *Cancelar*.
- Sobre el mapa, aparece un icono con el aparcamiento que ha creado el usuario.
- Ahora puede continuar su marcha y dejar el aparcamiento libre.

BUSCAR APARCAMIENTO EN UN DESTINO

- El usuario pude o no estar situado con su dispositivo en el interior del coche.
- Acceder a la pantalla principal de la aplicación autenticándose con un usuario válido.
- Arrastrar el dedo desde la parte izquierda de la pantalla principal hacia la derecha para que aparezca el menú deslizable.
- Pulsar en la opción de *Seleccionar un coche* (deberá tener creado al menos uno).
- En la nueva vista el usuario selecciona el coche en cuyo interior se encuentra en esos momentos.

- Pulsa sobre el botón *Buscar destino* que aparece en el menú superior de la pantalla principal.
- Aparece una nueva ventana de diálogo donde el usuario elige el tipo de aparcamiento que se dispone a dejar libre (en línea o en batería).
- Si desea echarse atrás puede cancelar la acción pulsando en *Cancelar*.
- Sobre el mapa, aparece un icono con el aparcamiento que ha creado el usuario.
- Ahora puede continuar su marcha y dejar el aparcamiento libre.

BUSCAR APARCAMIENTO EN UN DESTINO

- El usuario pude o no estar situado con su dispositivo en el interior del coche.
- Acceder a la pantalla principal de la aplicación autenticándose con un usuario válido.
- Arrastrar el dedo desde la parte izquierda de la pantalla principal hacia la derecha para que aparezca el menú deslizable.
- Pulsar en la opción de *Seleccionar un coche* (deberá tener creado al menos uno).
- En la nueva vista el usuario selecciona el coche en cuyo interior se encuentra en esos momentos.
- Pulsa sobre el botón *Buscar destino* que aparece en el menú superior de la pantalla principal.
- En la ventana de diálogo que aparece el usuario puede insertar la dirección de destino (calle, número, código postal…).
- Aparecen sobre el mapa los aparcamientos libres (en caso de que los haya) con colores, dependiendo de la probabilidad que tenga el usuario de encontrar esos aparcamientos libres (verde, amarillo y rojo).
- También puede ver un círculo cuyo centro es la posición de destino. El radio de dicho círculo vendrá marcado por la prioridad que tenga el usuario en cuestión:
    - Prioridad máxima → Radio = 150m
    - Prioridad intermedia → Radio = 300m
- Estos círculos ayudan al usuario a obtener distancias reales sobre el mapa que le ayuden a decidir hacia qué plaza de aparcamiento dirigirse.
    - Los usuarios con prioridad máxima podrán visualizar todos los aparcamientos libres en un radio de 500m desde la posición de destino.
    - Los usuarios con prioridad intermedia podrán visualizar los aparcamientos que se encuentren entre 150m y 500 m de la posición de destino.
    - Los usuario con prioridad mínima podrán visualizar solo los aparcamientos que se encuentren a más de 300m de la posición de destino.

- El usuario puede seleccionar cualquiera de los aparcamientos y obtener un cuadro con información como:
    - Prioridad (en porcentaje).
    - Distancia al punto de destino.
    - Tiempo que lleva libre el aparcamiento (presumiblemente).

- En caso de que el usuario quiera obtener información de cómo llegar, al pulsar sobre un aparcamiento, aparecen dos botones sobre la parte inferior de la pantalla principal. Estos botones lanzan la aplicación nativa de Google Maps para Android, gracias a la cuál el usuario podrá obtener un itinerario hacia el aparcamiento seleccionado.
- Cuando el usuario se acabe de aparcar el coche sobre uno de los aparcamientos que obtuvo en el mapa, debe seleccionar el icono que marca donde se encuentra el aparcamiento y en el cuadro de texto informativo volver a pulsar. Aparece una ventana de diálogo preguntando si desea ocupar la plaza libre en la que se encuentra.
- El sistema comprueba que las coordenadas del aparcamiento que tiene almacenadas coinciden con las que el usuario tiene en esos momentos, con una tolerancia de 4m. En caso de tratarse del mismo aparcamiento, la aplicación devuelve un mensaje de validación. Por el contrario, si las coordenadas del usuario no están próximas (a menos de 4 metros) al aparcamiento dibujado sobre el mapa, la aplicación informa al usuario indicando que no ha realizado un aparcamiento correcto.
- Esto último puede deberse a un mal uso por parte del usuario, queriendo hacer trampas, o que la precisión de las coordenadas no es buena, tanto del aparcamiento como del usuario en esos momentos.

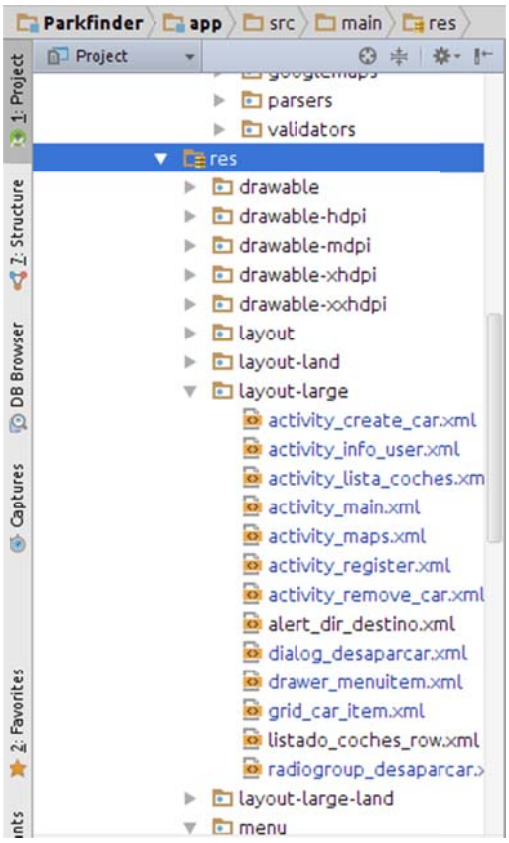
Anexo II. Interfaces gráficas de usuario en formato XML.

En este anexo se pueden consultar los ficheros XML que describen las interfaces gráficas de usuario implementadas con Android Studio. Todo proyecto Android cuenta con una carpeta **res**, donde se almacenan los iconos, imágenes y fondos (*drawables*), los contenedores de componentes de la interfaz gráfica (*layouts*), elementos del menú, etc.

Se pueden almacenar iconos de diferentes tamaños, para poder ser visualizados correctamente en cualquier pantalla de hoy en día. El propio sistema Android detecta los elementos que hay en estas carpetas y selecciona para su visualización aquellos que se corresponden con el dispositivo donde se encuentra instalado ese sistema.

Para conseguir diseños ***responsive design***, cualquier proyecto Android incorpora unas carpetas donde se alojan los *layouts* modificados según el tipo de visualización. Para ello se dispone de las carpetas:

- **layout**: Pantallas de tamaño normal (menos de 5") y en modo *portrait* (dispositivo en vertical).
- **layout-land**: Pantallas de tamaño normal (menos de 5,5") y en modo *landscape* (dispositivo en horizontal).
- **layout-large**: Pantallas de tamaño grande (más de 7") y en modo *portrait* (dispositivo en vertical).
- **layout-large-land**: Pantallas de tamaño grande (más de 7") y en modo *landscape* (dispositivo en horizontal).



Pantalla de *login:* activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@drawable/cyan_bg"
            android:orientation="vertical">

   <TextView
      android:id="@+id/title"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="175dp"
      android:text="@string/app_name"
      android:textColor="#FFFFFFFF"
      android:textSize="55sp"
      android:textStyle="bold"/>

   <TextView
      android:id="@+id/textView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@+id/title"
      android:layout_centerHorizontal="true"
      android:text="@string/description"
      android:textAppearance="?android:attr/textAppearanceSmall"
      android:textColor="#FFFFFFFF"
      android:textSize="17sp"/>

   <EditText
      android:id="@+id/et_user"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/title"
      android:layout_gravity="center_horizontal"
      android:layout_marginLeft="125dp"
      android:layout_marginRight="125dp"
      android:layout_marginTop="175dp"
      android:background="#FFFFFF"
      android:hint="@string/usuario"
      android:inputType="textPersonName"
      android:padding="5dp"
      android:textSize="20sp"
      android:textStyle="bold"
      />

   <EditText
      android:id="@+id/et_pass"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/et_user"
      android:layout_gravity="center_horizontal"
      android:layout_marginLeft="125dp"
      android:layout_marginRight="125dp"
      android:layout_marginTop="10dp"
      android:width="200dp"
      android:background="#FFFFFF"
      android:hint="@string/password"
      android:inputType="textPassword"
      android:padding="5dp"
      android:textSize="20sp"
      android:textStyle="bold"
      />
```

```
    <Button
        android:id="@+id/bt_login"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/et_pass"
        android:layout_gravity="center_horizontal"
        android:layout_marginLeft="125dp"
        android:layout_marginRight="125dp"
        android:layout_marginTop="15dp"
        android:width="200dp"
        android:background="#ff000000"
        android:clickable="true"
        android:inputType="text"
        android:padding="15dp"
        android:text="@string/login"
        android:textColor="#FFFFFFFF"
        android:textSize="20sp"
        android:textStyle="bold"/>

    <Button
        android:id="@+id/bt_registrarse"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_gravity="center_horizontal"
        android:width="200dp"
        android:background="@android:color/transparent"
        android:clickable="true"
        android:inputType="text"
        android:padding="5dp"
        android:text="@string/registrarse"
        android:textColor="#ff000000"
        android:textSize="20sp"
        android:textStyle="normal"/>

</RelativeLayout>
```

Pantalla de registro: activity_register.xml

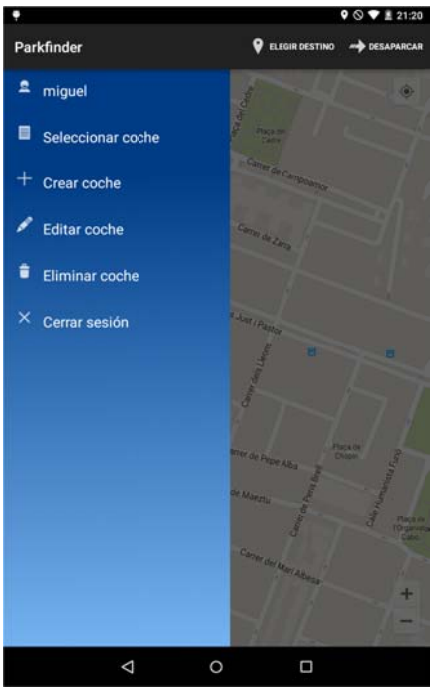
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@drawable/cyan_bg"
            android:orientation="vertical">

   <TextView
      android:id="@+id/titulo"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="175dp"
      android:text="@string/Parkfinder"
      android:textColor="#FFFFFFFF"
      android:textSize="55sp"
      android:textStyle="bold"/>

   <TextView
      android:id="@+id/textView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@+id/titulo"
      android:layout_centerHorizontal="true"
      android:text="@string/description"
      android:textAppearance="?android:attr/textAppearanceSmall"
      android:textColor="#FFFFFFFF"
      android:textSize="17sp"/>

   <EditText
      android:id="@id/et_userR"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/titulo"
      android:layout_gravity="center_horizontal"
      android:layout_marginLeft="125dp"
      android:layout_marginRight="125dp"
      android:layout_marginTop="175dp"
      android:background="#FFFFFF"
      android:hint="@string/usuario"
      android:inputType="textPersonName"
      android:padding="5dp"
      android:textSize="20sp"
      android:textStyle="bold"
      />

   <EditText
      android:id="@id/et_mailR"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/et_userR"
      android:layout_gravity="center_horizontal"
      android:layout_marginLeft="125dp"
      android:layout_marginRight="125dp"
      android:layout_marginTop="10dp"
      android:background="#FFFFFF"
      android:hint="@string/mail"
      android:inputType="textPersonName"
      android:padding="5dp"
      android:textSize="20sp"
      android:textStyle="bold"
      />
```

```
<EditText
    android:id="@id/et_passR"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/et_mailR"
    android:layout_gravity="center_horizontal"
    android:layout_marginLeft="125dp"
    android:layout_marginRight="125dp"
    android:layout_marginTop="10dp"
    android:width="200dp"
    android:background="#FFFFFF"
    android:hint="@string/password"
    android:inputType="textPassword"
    android:padding="5dp"
    android:textSize="20sp"
    android:textStyle="bold"
    />

<EditText
    android:id="@id/et_passconfR"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/et_passR"
    android:layout_gravity="center_horizontal"
    android:layout_marginLeft="125dp"
    android:layout_marginRight="125dp"
    android:layout_marginTop="10dp"
    android:width="200dp"
    android:background="#FFFFFF"
    android:hint="@string/confpassword"
    android:inputType="textPassword"
    android:padding="5dp"
    android:textSize="20sp"
    android:textStyle="bold"
    />

<Button
    android:id="@+id/bt_save"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/et_passconfR"
    android:layout_gravity="center_horizontal"
    android:layout_marginLeft="125dp"
    android:layout_marginRight="125dp"
    android:layout_marginTop="15dp"
    android:width="200dp"
    android:background="#ff000000"
    android:clickable="true"
    android:inputType="text"
    android:padding="15dp"
    android:text="@string/guardar"
    android:textColor="#FFFFFFFF"
    android:textSize="20sp"
    android:textStyle="bold"/>

<Button
    android:id="@+id/bt_cancel"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_gravity="center_horizontal"
    android:layout_marginLeft="125dp"
    android:layout_marginRight="125dp"
    android:width="200dp"
    android:background="@android:color/transparent"
```

```
        android:clickable="true"
        android:inputType="text"
        android:padding="5dp"
        android:text="@string/cancel"
        android:textColor="#ff000000"
        android:textSize="20sp"
        android:textStyle="normal"/>

</RelativeLayout>
```

Pantalla principal de la aplicación: activity_maps.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
   android:id="@+id/drawer_layout"
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:theme="@android:style/Theme.WithActionBar">

   <!-- Contenido principal -->
   <FrameLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent">

      <!--The map fragment-->
      <fragment
         android:id="@+id/map"
         android:name="com.google.android.gms.maps.MapFragment"
         android:layout_width="match_parent"
         android:layout_height="match_parent"/>

   </FrameLayout>

   <!-- Menú lateral -->

   <ListView
      android:id="@+id/left_drawer"
      android:layout_width="320dp"
      android:layout_height="match_parent"
      android:layout_gravity="start"
      android:background="@drawable/cyan_bg"
      android:choiceMode="singleChoice"
      android:clickable="true"
      android:divider="@android:color/transparent"
      android:dividerHeight="5dp"/>

</android.support.v4.widget.DrawerLayout>
```



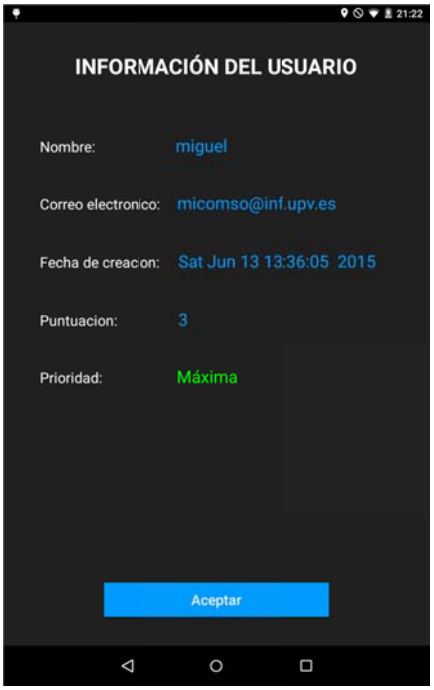
Pantalla de información del usuario: activity_info_user.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
            xmlns:tools="http://schemas.android.com/tools"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_gravity="center_horizontal"
            android:background="@color/cardview_dark_background"
            android:orientation="vertical"
            android:paddingBottom="@dimen/activity_vertical_margin"
            android:paddingLeft="@dimen/activity_horizontal_margin"
            android:paddingRight="@dimen/activity_horizontal_margin"
            android:paddingTop="@dimen/activity_vertical_margin"
            tools:context="es.upv.etsinf.tfg.parkfinder.activities.activity_info_user">

   <!-- Título -->
   <TextView
      android:id="@+id/lista_coches_usuario_titulo"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="25dp"
      android:text="@string/title_info_user"
      android:textAllCaps="true"
      android:textColor="#FFFFFFFF"
      android:textSize="30sp"
      android:textStyle="bold"/>

   <LinearLayout
      android:id="@+id/title_info1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@id/lista_coches_usuario_titulo"
      android:layout_marginBottom="25dp"
      android:layout_marginLeft="35dp"
      android:layout_marginTop="75dp"
      android:orientation="horizontal">

      <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_marginRight="115dp"
         android:text="@string/nombre"
         android:textColor="#ffffff"
         android:textSize="20sp"/>

      <TextView
         android:id="@+id/info_user_text_name"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="-"
         android:textColor="#ff009bff"
         android:textSize="24sp"/>

   </LinearLayout>
```

```
<LinearLayout
    android:id="@+id/title_info2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/title_info1"
    android:layout_marginBottom="25dp"
    android:layout_marginLeft="35dp"
    android:layout_marginTop="25dp"
    android:orientation="horizontal">

    <TextView

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="24dp"
        android:text="@string/email"
        android:textColor="#ffffff"
        android:textSize="20sp"/>

    <TextView
        android:id="@+id/info_user_text_mail"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="-"
        android:textColor="#ff009bff"
        android:textSize="24sp"/>

</LinearLayout>

<LinearLayout
    android:id="@+id/title_info3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/title_info2"
    android:layout_marginBottom="25dp"
    android:layout_marginLeft="35dp"
    android:layout_marginTop="25dp"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="26dp"
        android:text="@string/fecha_creacion"
        android:textColor="#ffffff"
        android:textSize="20sp"/>

    <TextView
        android:id="@+id/info_user_text_fecha"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="-"
        android:textColor="#ff009bff"
        android:textSize="24sp"/>

</LinearLayout>

<LinearLayout
    android:id="@+id/title_info4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/title_info3"
    android:layout_marginBottom="25dp"
    android:layout_marginLeft="35dp"
    android:layout_marginTop="25dp"
    android:orientation="horizontal">
```

```
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="88dp"
            android:text="@string/puntuacion"
            android:textColor="#ffffff"
            android:textSize="20sp"/>

        <TextView
            android:id="@+id/info_user_text_puntuacion"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="-"
            android:textColor="#ff009bff"
            android:textSize="24sp"/>

    </LinearLayout>

    <LinearLayout
        android:id="@+id/title_info5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/title_info4"
        android:layout_marginBottom="25dp"
        android:layout_marginLeft="35dp"
        android:layout_marginTop="25dp"
        android:orientation="horizontal">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="107dp"
            android:text="@string/prioridad"
            android:textColor="#ffffff"
            android:textSize="20sp"/>

        <TextView
            android:id="@+id/info_user_text_prioridad"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="-"
            android:textColor="#ff009bff"
            android:textSize="24sp"/>

    </LinearLayout>

    <!--Botones Crear coche y Cancelar-->
    <LinearLayout
        android:id="@+id/layout_crear_coche_botones"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:orientation="vertical">

        <!-- Botón "Crear" -->
        <Button
            android:id="@+id/aceptar_info_user"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginBottom="25dp"
            android:layout_marginLeft="125dp"
            android:layout_marginRight="125dp"
            android:width="200dp"
            android:background="#ff009bff"
            android:clickable="false"
            android:inputType="text"
            android:padding="5dp"
```

```
        android:text="@string/aceptar"
        android:textColor="@color/lb_basic_card_title_text_color"
        android:textSize="20sp"
        android:textStyle="normal"/>
    </LinearLayout>

</RelativeLayout>
```



Pantalla *Seleccionar coche*: activity_lista_coches.xml y listado_coches_row.xml

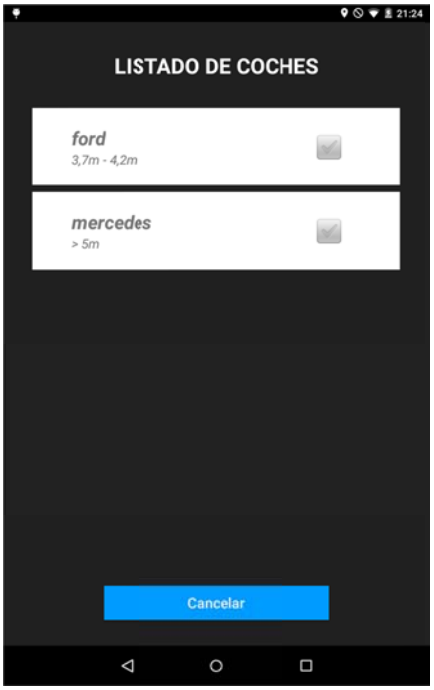
Pantalla *Eliminar coche*: activity_remove_car.xml y listado_coches_row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
            xmlns:tools="http://schemas.android.com/tools"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@color/cardview_dark_background"
            android:paddingBottom="@dimen/activity_vertical_margin"
            android:paddingLeft="@dimen/activity_horizontal_margin"
            android:paddingRight="@dimen/activity_horizontal_margin"
            android:paddingTop="@dimen/activity_vertical_margin"
            tools:context="es.upv.etsinf.tfg.parkfinder.activities.SelectCarActivity">

   <!-- Título -->
   <TextView
      android:id="@+id/lista_coches_usuario_titulo"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="25dp"
      android:text="@string/lista_coches_usuario_titulo"
      android:textAllCaps="true"
      android:textColor="#FFFFFFFF"
      android:textSize="30sp"
      android:textStyle="bold"/>


   <!-- Lista de coches del usuario -->
   <FrameLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_above="@+id/layout_boton_cancelar"
      android:layout_below="@+id/lista_coches_usuario_titulo"
      >

      <ListView
         android:id="@+id/lista_coches_usuario"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:layout_marginBottom="40dp"
         android:layout_marginLeft="25dp"
         android:layout_marginRight="25dp"
         android:layout_marginTop="40dp"
         android:choiceMode="singleChoice"
         android:divider="@android:color/transparent"
         android:dividerHeight="10.0sp"
         android:scrollbars="vertical"
         />
   </FrameLayout>

   <LinearLayout
      android:id="@+id/layout_boton_cancelar"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true"
      android:orientation="vertical">

      <Button
```

```
        android:id="@+id/bt_cancel"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:layout_marginLeft="125dp"
        android:layout_marginRight="125dp"
        android:width="200dp"
        android:background="#ff009bff"
        android:clickable="true"
        android:inputType="text"
        android:padding="5dp"
        android:text="@string/cancel"
        android:textColor="@color/lb_basic_card_title_text_color"
        android:textSize="20sp"
        android:textStyle="normal"/>
  </LinearLayout>

</RelativeLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="25dp">

  <TextView
     android:id="@+id/nombre_coche"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_alignParentLeft="true"
     android:layout_marginLeft="30dp"
     android:layout_toLeftOf="@+id/chkbox_selectCoche"
     android:focusable="false"
     android:text="Alias"
     android:textSize="26sp"
     android:textStyle="bold|italic"
     />

  <TextView
     android:id="@+id/segmento_coche"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_alignParentLeft="true"
     android:layout_below="@+id/nombre_coche"
     android:layout_marginLeft="30dp"
     android:focusable="false"
     android:text="Segmento"
     android:textSize="18dp"
     android:textStyle="italic"
     />

  <ImageView
     android:id="@+id/icon_tick"
     android:layout_width="50dp"
     android:layout_height="50dp"
     android:layout_alignParentRight="true"
     android:layout_centerVertical="true"
     android:layout_marginRight="50dp"
     android:background="#FFF"
     android:contentDescription="@string/icon_checkbox"
     android:focusable="false"
     />
```

```
    <RadioButton
        android:id="@+id/radioButtonItem"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:layout_marginRight="50dp"
        android:clickable="false"
        android:focusable="false"
        android:focusableInTouchMode="false"
        android:visibility="invisible"
        />

</RelativeLayout>
```



Pantalla *Crear coche*: activity_create_car.xml y grid_car_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
            xmlns:tools="http://schemas.android.com/tools"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_gravity="center_horizontal"
            android:background="@color/cardview_dark_background"
            android:orientation="vertical"
            android:paddingBottom="@dimen/activity_vertical_margin"
            android:paddingLeft="@dimen/activity_horizontal_margin"
            android:paddingRight="@dimen/activity_horizontal_margin"
            android:paddingTop="@dimen/activity_vertical_margin"
            tools:context="es.upv.etsinf.tfg.parkfinder.activities.activity_crear_coche">
   <!-- Título -->
   <TextView
      android:id="@+id/lista_coches_usuario_titulo"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="25dp"
      android:text="@string/title_activity_crear_coche"
      android:textAllCaps="true"
      android:textColor="#FFFFFFFF"
      android:textSize="30sp"
      android:textStyle="bold"/>

   <!-- Formulario para crear un coche nuevo -->
   <LinearLayout
      android:id="@+id/layout_nombre_coche"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/lista_coches_usuario_titulo"
      android:layout_marginBottom="25dp"
      android:layout_marginTop="75dp">

      <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_marginBottom="50dp"
         android:layout_marginLeft="50dp"
         android:layout_marginTop="5dp"
         android:width="150dp"
         android:text="@string/nombre"
         android:textColor="#ffffffff"
         android:textSize="24sp"/>

      <EditText
         android:id="@+id/et_nombre_coche_nuevo"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:layout_gravity="center_horizontal"
         android:layout_marginLeft="5dp"
         android:layout_marginRight="40dp"
         android:background="#FFFFFFFF"
         android:hint="@string/nombreCocheNuevo"
         android:inputType="text"
         android:padding="5dp"
         android:textSize="18sp"
         android:textStyle="italic"/>
   </LinearLayout>
```

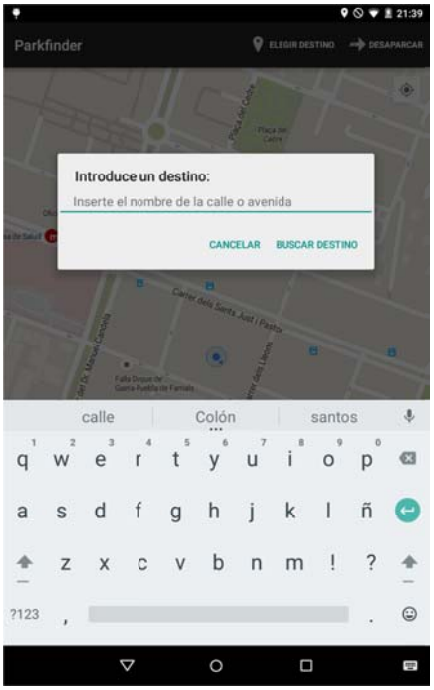
```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_above="@+id/layout_crear_coche_botones"
    android:layout_below="@+id/layout_nombre_coche"
    android:layout_marginBottom="30dp"
    android:layout_marginLeft="25dp"
    android:layout_marginRight="25dp">

    <TextView
        android:id="@+id/dimensiones_titulo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="25dp"
        android:text="@string/dimensiones"
        android:textColor="#ffffffff"
        android:textSize="24sp"/>

    <GridView
        android:id="@+id/grid_view_cars"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:choiceMode="singleChoice"
        android:columnWidth="150dp"
        android:horizontalSpacing="10dp"
        android:numColumns="3"
        android:verticalSpacing="10dp">

    </GridView>
</RelativeLayout>

<!--Botones Crear coche y Cancelar-->
<LinearLayout
    android:id="@+id/layout_crear_coche_botones"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:orientation="vertical">

    <!-- Botón "Crear" -->
    <Button
        android:id="@+id/bt_create"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginBottom="25dp"
        android:layout_marginLeft="125dp"
        android:layout_marginRight="125dp"
        android:width="200dp"
        android:background="#ff009bff"
        android:clickable="true"
        android:inputType="text"
        android:padding="5dp"
        android:text="@string/createCar"
        android:textColor="@color/lb_basic_card_title_text_color"
        android:textSize="20sp"
        android:textStyle="normal"/>

    <!-- Botón "Cancelar" -->
    <Button
        android:id="@+id/bt_cancel"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="125dp"
        android:layout_marginRight="125dp"
```

```
        android:width="200dp"
        android:background="@android:color/transparent"
        android:clickable="true"
        android:inputType="text"
        android:padding="5dp"
        android:text="@string/cancel"
        android:textColor="@color/lb_basic_card_title_text_color"
        android:textSize="20sp"
        android:textStyle="normal"/>
  </LinearLayout>
</RelativeLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:clickable="false"
          android:focusable="false"
          android:focusableInTouchMode="false"
          android:orientation="vertical"
          android:padding="5dp">
  <TextView
      android:id="@+id/grid_item_textView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_margin="10dp"
      android:width="150dp"
      android:height="150dp"
      android:background="@android:color/holo_red_light"
      android:clickable="false"
      android:focusable="false"
      android:focusableInTouchMode="false"
      android:gravity="center"
      android:text="item"
      android:textSize="24dp"
      android:textStyle="bold|italic"/>
</LinearLayout>
```

Ventana *Buscar destino*: alert_dir_destino.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@color/cardview_dark_background">

   <TextView
      android:id="@+id/elija_destino_titulo"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginLeft="25dp"
      android:layout_marginTop="30dp"
      android:text="Elija un destino:"
      android:textColor="#FFF"
      android:textSize="18dp"/>

   <EditText
      android:id="@+id/text"
      android:layout_width="550dp"
      android:layout_height="30dp"
      android:layout_below="@+id/elija_destino_titulo"
      android:layout_centerHorizontal="true"
      android:layout_margin="20dp"
      android:background="#ffffffff"
      android:hint="Calle, número"
      android:textColor="#FFF"
      />

   <LinearLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentRight="true"
      android:layout_below="@id/text"
      android:layout_marginBottom="30dp"
      android:layout_marginTop="1dp"
      android:orientation="horizontal">

      <Button
         android:id="@+id/dialogButtonOK"
         android:layout_width="100dp"
         android:layout_height="wrap_content"
         android:layout_marginRight="10dp"
         android:layout_marginTop="10dp"
         android:text=" Ir a... "
         android:textStyle="bold|italic"
         />

      <Button
         android:id="@+id/dialogButtonCancel"
         android:layout_width="100dp"
         android:layout_height="wrap_content"
         android:layout_marginRight="10dp"
         android:layout_marginTop="10dp"
         android:text=" Cancelar "
         android:textStyle="bold|italic"
         />
   </LinearLayout>

</RelativeLayout>
```


21:39
Parkfinder
ELEGIR DESTINO
DESAPARCAR
Introduce un destino:
Inserte el nombre de la calle o avenida
CANCELAR
BUSCAR DESTINO
calle
Colón
santos
?123

REFERENCIAS

Formación previa

- ***Curso de Introducción a la Programación Android.***
  Plataforma UPV[X]
  Fecha: Marzo 2014 - Mayo 2014

- ***Programming Mobile Applications for Android Handheld Systems***
  Plataforma Coursera
  Fecha: Octubre 2014 – Diciembre 2014

Bibliografía

- *Murphy, Mark L.* ***The Busy Coder's Guide to Android Development****. 2nd Edition. CommonsWare.*

- *Jason Lengstorf, Thomas Blom Hansen.* ***PHP for Absolute Beginners*** *2nd Edition. Apress.*

- *Julie Meloni.* ***Sams Teach Yourself PHP, MySQL and Apache All in One*** *5th Edition. Sams.*

- *Guía de evaluación de usabilidad móvil:* ***https://pshyama.wordpress.com/2010/06/17/mobile-user-interface-and-usability-guide/***

- *Android developers:* ***https://developer.android.com/guide/index.html***

- *Curso de Android:* ***http://www.sgoliver.net/blog/curso-de-programacion-android/***

- *Android Hive:* ***http://www.androidhive.info/***

- *PHP y JSON:* ***http://php.net/manual/es/book.json.php***

  http://www.desarrolloweb.com/articulos/producir-json-desde-php.html