



Escuela
Politécnica
Superior

Aplicación web y móvil para buscar aparcamiento y compartir coche en la universidad de Alicante



Trabajo Fin de Grado

Autor:

Ouadi Chamit

Tutor/es:

Estela Saquete Boro

Septiembre 2017



Universitat d'Alacant
Universidad de Alicante

Tabla de contenido

INTRODUCCION	4
RESUMEN	4
MOTIVACION Y ESTUDIO DE MERCADO	5
➤ BlaBlaCar	5
➤ Uber	5
➤ Amovens	5
➤ Carpooling	6
➤ Wazypark	6
➤ Parkopedia	6
➤ Aparca&Go	6
METODOLOGÍA.....	7
HERRAMIENTAS, TECNOLOGIAS Y LENGUAJES DE PROGRAMACION EMPLEADOS.....	8
phpMyAdmin.....	8
Github.....	8
Balsamiq	9
Codigneiter	9
PhpStorm	10
Draw.io	10
Selenium IDE	10
Android Studio	11
Eclipse.....	11
REQUERITOS FUNCIONALES Y NO FUNCIONALES	12
Requisitos funcionales.....	12
Requisitos no funcionales Los requisitos no funcionales serán:.....	12
DISEÑO	13
Mockups.....	13
Web	13
Móvil.....	17
Wireframe	18
DISEÑO FINAL	19
Web	19
Móvil.....	38
ARQUETECTURA.....	65

Web	65
Móvil.....	67
IMPLEMENTACION	68
- Front end.....	68
Web	68
Móvil.....	77
- Back end.....	87
Web	87
Móvil.....	92
Base de datos	101
Esquema conceptual	101
Modelo Relacional.....	102
Diccionario de Datos	104
SEGURIDAD	111
Web.....	111
Balancear riesgo y usabilidad.....	111
Rastrear el paso de los datos	111
<u>XSS Prevention.....</u>	<u>111</u>
Ataques de inyección SQL	111
Contraseña del usuario	112
Móvil.....	114
APIS UTILIZADAS.....	115
Api Google maps	115
Web	115
Móvil.....	118
SIGUA	122
PRUEBAS.....	125
CONCLUSIONES, TRABAJOS FUTUROS Y POSIBLES MEJORAS.....	135
REFERENCIAS.....	136

INTRODUCCION

RESUMEN

El proyecto consiste en la creación de una aplicación web y móvil, que ayuda a los usuarios a publicar anuncios para compartir coche o buscar aparcamiento en la universidad de Alicante, con el fin de facilitar a encontrar un aparcamiento libre, o poder compartir coche para ir y venir de la universidad.

El proyecto nace con la idea de crear una aplicación dedicada a todos los usuarios de la universidad de Alicante, en concreto a aquellos usuarios que no tienen medios para venir a la universidad o los que no tienen suficiente los gastos de los viajes y ayudar a los propietarios de los coches poder ahorrar y gastar menos. También sirve para buscar aparcamiento de forma cómoda, ya que la universidad consta de varios parkings, cada uno se identifica por una referencia y está ubicado en una zona, por lo que resulta difícil, encontrar un hueco en una zona predeterminada, según el interés de los usuarios, y como resultado, perder tiempo, gastos de energía etc.

MOTIVACION Y ESTUDIO DE MERCADO

Siendo alumno de la universidad, siento esa necesidad, y la de los demás usuarios, por lo que me motiva desarrollar esa aplicación y poder resolver uno de los problemas que afrontan los usuarios de la universidad que se encuentran a día hoy, y poder proveer una herramienta, para poner fin a esa necesidad o por lo menos paliar el problema.

Al analizar la competencia nos hemos dado cuenta de que no existe una aplicación en concreto con el mismo objetivo que gestiona tanto el aparcamiento como compartir coche a la vez específicamente para el sector universitario. No obstante, hay algunas aplicaciones encontradas que coinciden en algún apartado/sección, y que nos ha servido para mejorar nuestra idea.

Las aplicaciones buscadas han sido, por ejemplo:

➤ BlaBlaCar

Es una red social de viajes de largas y cortas distancias en coche compartido. Permite a los usuarios ponerse en contacto cuando quieren realizar un trayecto común y coinciden para hacerlo el mismo día, así como permitir a los conductores ahorrar el coste del litro de gasolina por cada pasajero.

Los usuarios comparten los gastos del viaje sin que el conductor tenga beneficios, para eso la red social recomienda a los usuarios por cada viaje la aportación de 0.06 euros por cada kilómetro cada uno, y limita la aportación máxima que pueden solicitar los conductores de tal manera que no se superen estos gastos.

➤ Uber

Uber es una aplicación para compartir viajes rápidos y fiables en cuestión de minutos, de día o de noche. No hace falta aparcar ni esperar taxis o autobuses.

Mediante la cual, comunicas tu ubicación, el destino, el tipo de vehículo y el tiempo de espera estimado. Tras el viaje, lo puedes pagar con tarjeta, efectivo o con tu teléfono móvil y el 20% se lo queda la propia empresa a modo de comisión

➤ Amovens

Es similar a Uber en el concepto urbano de la aplicación, pero el objetivo de esta herramienta es encontrar a gente que realice el mismo trayecto que tú a diario y poder crear vínculos de transporte entre las personas, para ir al trabajo o a la universidad, sin que la empresa se lleve ningún tipo de comisión ni facilite modos de pago.

➤ **Carpooling**

Es una aplicación similar a BlaBlaCar. Sirve exactamente para lo mismo, consiste en compartir nuestros viajes en auto con otras personas de forma cotidiana. El carpooling es una práctica popular en Estados Unidos y Europa, donde se realiza de manera organizada para lograr aumentar el número de viajes compartidos y que estos sean concretados con otras personas además de nuestros vecinos y amigos.

Sin embargo ofrece elementos como la elección de viajes solo entre mujeres o fijar un radio de búsqueda a tu alrededor para empezar el viaje.

➤ **Wazypark**

Se trata de una aplicación de inteligencia colectiva en la que mediante geolocalización del móvil en un mapa, podemos saber dónde hay una plaza libre en las calles de nuestra ciudad para aparcar. Wazypark mejora a medida que aumenta el número de usuarios, que deben registrarse ellos mismos tanto como a su vehículo. Cada usuario avisa desde su móvil cuando deja un hueco libre en un aparcamiento, de modo que otro usuario cercano nos pueda detectar y aparcar en la plaza. Disponible para Android e iOS.

➤ **Parkopedia**

Tiene el mismo espíritu que Wazypark pero con elementos de Foursquare, en la que los usuarios dejan información de zonas de aparcamientos libres en plena calle o de aparcamientos públicos o privados, con los precios por hora o día de cada parking. Su interés reside en que hay información de numerosas ciudades. Tanto en Android como en iOS.

➤ **Aparca&Go**

Disponible solo para Android como aplicación, pero se puede usar desde la web. Se ha especializado en la gestión de aparcamientos mediante acuerdos por los que el usuario puede ganar un descuento en su ticket. La aplicación gestiona los pagos, hay que dejar los datos de una tarjeta, por lo que las molestias son mínimas; además se puede reservar la plaza por adelantado. Un valor fuerte de esta aplicación es la creación de aparcamientos propios y alternativos a los de aeropuertos y estaciones de tren, pero con precios reducidos. De modo que ellos gestionan el transporte hasta el aeropuerto desde sus aparcamientos y de vuelta.

METODOLOGÍA

Para llevar acabo el desarrollo de la aplicación he utilizado el modelo iterativo y incremental, En ese tipo de desarrollo el proyecto se planifica en diversos bloques, en cada iteración se repite un proceso similar para un resultado completo sobre el producto final, de manera que los beneficios se obtienen de forma incremental, para ello los requisitos determinados para una determinada iteración, se deben completar en esa única iteración.

El ciclo de vida incremental:

Desarrollar por partes el producto software, para después integrarlas a medida que se completan.

El ciclo de vida iterativo:

En cada ciclo, iteración, se revisa y mejora el producto. Un ejemplo de desarrollo iterativo es aquel basado en refactorizaciones, en el que cada ciclo mejora más la calidad del producto.

Características:

- Es un poco complicado de estimar el costo total
- Se requiere experimentación
- Permite la involucración del usuario

Ventajas

- Cada etapa consiste en análisis, diseño, codificación, pruebas.
- No es necesario de mucho personal al principio como para una implementación completa.
- Ofrece una gran ventaja al usuario o cliente, la cual es la entrega oportuna de partes operativas del software.
- Se disminuye el tiempo de desarrollo inicial, ya que se incorpora la funcionalidad parcial.

Desventajas

- El modelo incremental no es recomendable para casos de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido.
- Se necesita de mucha planeación, tanto a nivel administrativo como técnico.
- Requiere unas metas fijas para determinar el estado del proyecto.
- Las primeras versiones son incompletas pero proporcionan al usuario la funcionalidad que precisa y una plataforma para la evaluación.

HERRAMIENTAS, TECNOLOGIAS Y LENGUAJES DE PROGRAMACION EMPLEADOS

Para llevar a cabo el proyecto he utilizado diversas tecnologías y herramientas seleccionadas para cumplir con ciertos propósitos concretos. Para la realización de la aplicación parte web, he utilizado el lenguaje de programación PHP en el lado Backend y el framework bootstrap, HTML, CSS, JAVASCRIPT, JQUERY en lado de FrontEnd. Sin embargo en la parte Móvil he utilizado Android en Frontend y el framework jersy en Backend, tanto Android como jersy, el desarrollo se lleva con Java. Por último, para realizar la base de datos he utilizado SQL. La herramienta utilizada para manejar la base de datos es phpMyAdmin.

phpMyAdmin

phpMyAdmin es la herramienta que he utilizado para el manejo de la base de datos, MySQL está alojado en el servidor junto con la página web. La versión de MySQL es la 10.1.25-MariaDB - mariadb.org binary distribution y la versión de phpMyAdmin utilizada es la 4.7.0 y está alojada sobre un servidor Apache/2.4.26 (Win32) OpenSSL/1.0.2I PHP/7.1.7.



Github

GitHub es una plataforma de **desarrollo colaborativo de software** para alojar proyectos utilizando el sistema de control de versiones Git. Sirve para alojar un repositorio de código y brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto. Además de todo lo mencionado anteriormente, se ofrecen varias herramientas útiles para el trabajo en equipo



- Una wiki para el mantenimiento de las distintas versiones de las páginas.
- Un sistema de seguimiento de problemas que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.
- Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.
- Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio

Balsamiq

He utilizado Balsamiq Mockups (versión escritorio y web). Balsamiq es la herramienta que he utilizado para realizar los mockups de las pantallas que va a tener mi aplicación. Con el objetivo claro de conseguir un enfoque sobre el diseño básico en etapas más tempranas, es decir, antes de comenzar a diseñar las pantallas ya sé cuál es el diseño que van a tener.



Codigneiter

CodeIgniter es un entorno de desarrollo abierto que utiliza el **MVC** y permite crear webs dinámicas con PHP . Su principal objetivo es ayudar a que los desarrolladores, puedan realizar proyectos mucho más rápido que creando toda la estructura desde cero, proveyendo un rico juego de librerías para tareas comúnmente necesarias, así como una interface simple y estructura lógica para acceder a esas librerías.



Ventajas de utilizar un framework como CodeIgniter

- Las páginas se procesan más rápido, el núcleo de CodeIgniter es bastante ligero.
- Es sencillo de instalar, basta con subir los archivos al ftp y tocar un archivo de configuración para definir el acceso a la bd.
- Reutilización de código, desarrollo ágil.
- Existe abundante documentación en la red.
- Facilidad de edición del código ya creado.
- Facilidad para crear nuevos módulos, páginas o funcionalidades.
- Acceso a librerías públicas y clases. Entre otras, hay librerías para el login, paginador, calendarios, fechas,....
- Estandarización del código. Fundamental cuando hay que tocar código hecho por otra persona o cuando trabaja más de una persona en un mismo proyecto.
- URLs amigables con SEO. Hoy en día creo que nadie duda de la importancia del posicionamiento web.
- Separación de la lógica y arquitectura de la web, el MVC.

- CodeIgniter es bastante menos rígido que otros frameworks. Define una manera de trabajar, pero podemos seguirlo o no (esto puede convertirse en un inconveniente también)
- Cualquier servidor que soporte PHP+MySQL sirve para CodeIgniter.
- CodeIgniter se encuentra bajo una licencia open source, es código libre.
- CodeIgniter usa una versión modificada del Patrón de Base de Datos Active Record. Este patrón permite obtener, insertar y actualizar información en tu base de datos con mínima codificación. Permite queries más seguras, ya que los valores son escapadas automáticamente por el sistema.

PhpStorm

Es un IDE de programación desarrollado por JetBrains. Es uno de los entornos de programación más completos de la actualidad, permite editar código de varios lenguajes de programación aunque su nombre empieza por php.



algunas de las características principales.

- Proporciona un fácil autocompletado de código.
- Soporta el trabajo con PHP 5.5
- Sintaxis abreviada.
- Permite la gestión de proyectos fácilmente.

Draw.io

Es una aplicación web interesante que es de gran ayuda para la realización de diagramas tales como: UML, diagramas de flujo, diagramas BPMN, creación de mockups, sin necesidad de instalar nada en la PC.



Selenium IDE

SeleniumHQ es un conjunto de herramientas de pruebas open-source para automatizar pruebas funcionales sobre aplicaciones Web



Android Studio

Android Studio es el IDE oficial para desarrollar aplicaciones para Android. Es un IDE basado en IntelliJ IDEA, creado y mantenido por el equipo de desarrollo de las toolkits de Google.



Eclipse

Es una plataforma de desarrollo, diseñada para ser extendida de forma indefinida a través de plug-ins. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo. No tiene en mente un lenguaje específico, sino que es un IDE genérico, aunque goza de mucha popularidad entre la comunidad de desarrolladores del lenguaje Java usando el plug-in JDT que viene incluido en la distribución estándar del IDE.

Proporciona herramientas para la gestión de espacios de trabajo, escribir, desplegar, ejecutar y depurar aplicaciones.



REQUISITOS FUNCIONALES Y NO FUNCIONALES

Requisitos funcionales

Los principales requisitos funcionales de la aplicación son:

- Cada usuario registrado es dado de alta, también existe el caso de dar de alta al coche si ese usuario es dueño de un coche en la cual comparte los viajes a/y aparca en la universidad.
- Cada usuario puede dar de alta a una notificación de compartir su coche cuando tiene plazas libres.
- Cada usuario puede buscar aparcamiento, elegiendo la referencia del aparcamiento y ver las zonas libre.
- Cada usuario cuando sale del aparcamiento, debe actualizar la zona y ponerla como libre.
- Cuando un usuario aparca su coche, debe poner como ocupada la zona.
- Cada usuario puede comentar otro cuando comparte los viajes en su coche, así se lo permite a los demás usuarios conocer a las situaciones tanto del dueño como el coche.
- Cuando el usuario pretende compartir ruta, realiza una reserva a la espera de la aceptación del propietario del anuncio.

Requisitos no funcionales Los requisitos no funcionales serán:

- Usabilidad. La usabilidad en la aplicación será esencial ya que la mayoría de los usuarios que utilizarán la aplicación no estarán relacionados con el entorno informático. La tasa de errores cometidos por el usuario deberá ser menor. El sistema debe proporcionar mensajes de error que sean informativos y orientados a usuario final. El sistema debe poseer interfaces gráficas bien formadas.
- Eficiencia. El sistema debe ser capaz de procesar N transacciones por segundo. Toda funcionalidad del sistema y transacción de negocio debe responder al usuario en menor tiempo posible. El sistema debe ser capaz de operar adecuadamente con hasta N usuarios con sesiones concurrentes. Los datos modificados en la base de datos deben ser actualizados para todos los usuarios que acceden en menor tiempo.
- Dependibilidad. El sistema debe tener una disponibilidad del 99,99% de las veces en que un usuario intente accederlo. El tiempo para iniciar o reiniciar el sistema no podrá ser mayor. La tasa de tiempos de falla del sistema no podrá ser mayor. El promedio de duración de fallas no podrá ser mayor. La probabilidad de falla del Sistema no podrá ser mayor.

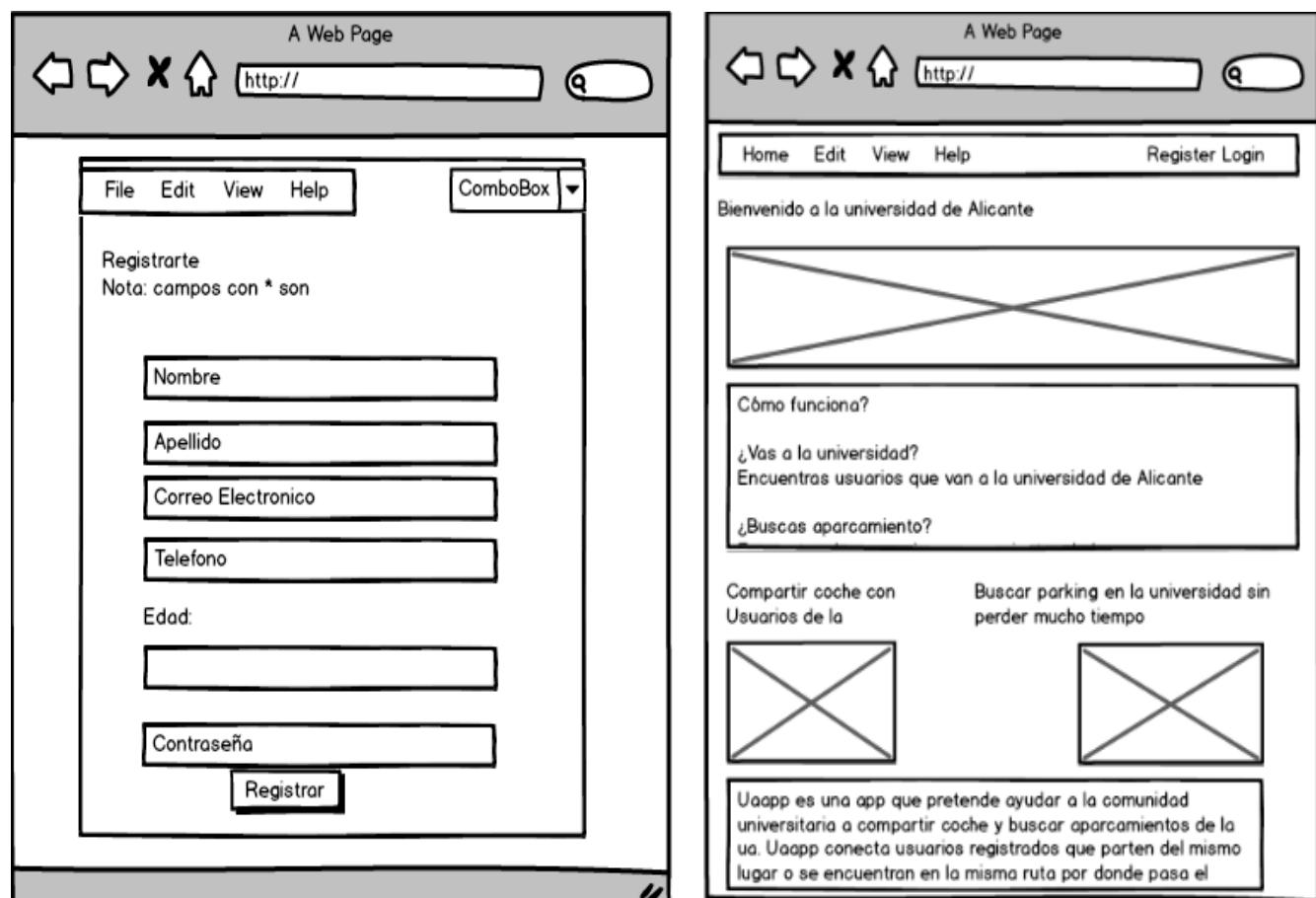
DISEÑO

Mockups

Antes de comenzar a implementar el proyecto, he creado los mockups de la aplicación. Consiguiendo así definir el diseño básico que tendrán las pantallas en fases tempranas del desarrollo .

➤ Web

Por ejemplo vamos a ver el mockup de la pantallas de la aplicación.



Pantalla Login/Actualizar datos usuario

A Web Page

http://

File Edit View Help ComboBox

Identificate

Correo Electronico

Contraseña

Has olvidado

Login

A Web Page

http://

Home Editar perfil Viajes publicados Buscar Publicar ComboBox

Información del perfil

Información del perfil

Foto

Coche

Opiniones

Opiniones recibidas

Opiniones dejadas

Nombre

Apellidos

Correo electrónico

Teléfono

Edad

Contraseña

Actualizar

Posted by: Hége Refsnes

Contact information: someone@example.com

Pantalla Coche/Foto usuario

A Web Page

http://

Home Editar perfil Viajes publicados Buscar Publicar ComboBox

Información del perfil
Información del perfil
Foto
Coche

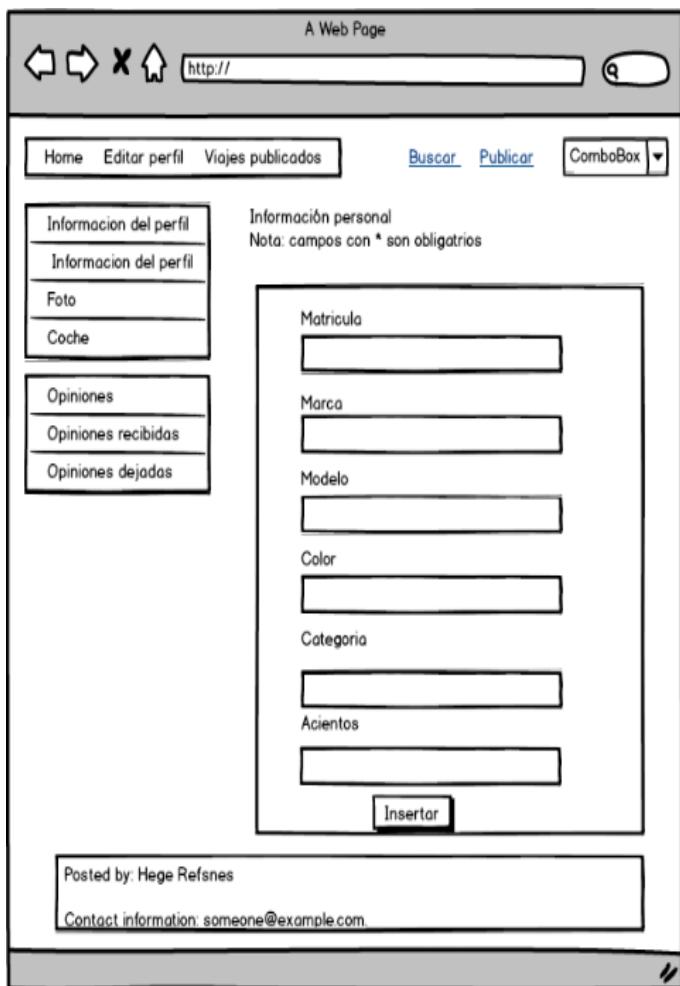
Opiniones
Opiniones recibidas
Opiniones dejadas

Información personal
Nota: campos con * son obligatorios

Matrícula
Marca
Modelo
Color
Categoría
Acentos

Insertar

Posted by: Hege Refsnes
Contact information: someone@example.com



A Web Page

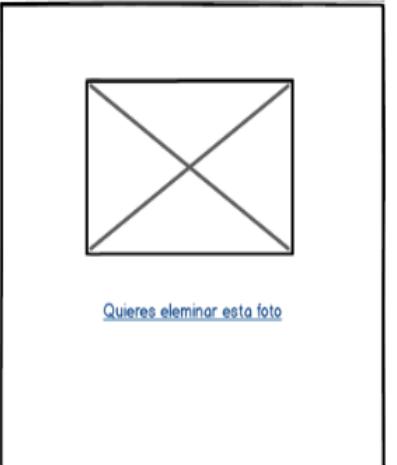
http://

Home Editar perfil Viajes publicados Buscar Publicar ComboBox

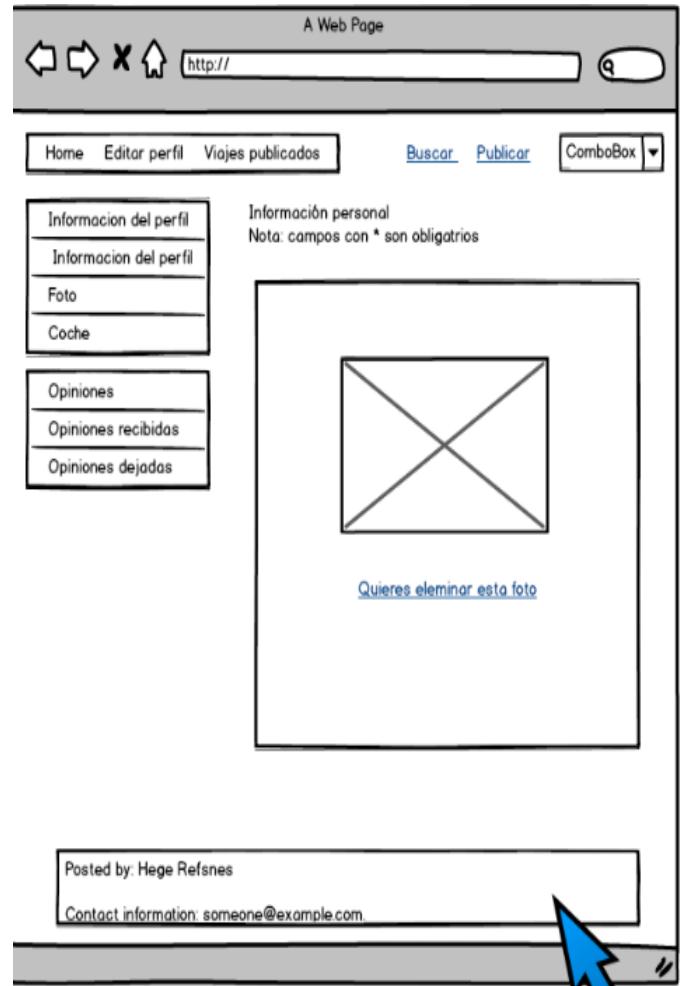
Información del perfil
Información del perfil
Foto
Coche

Opiniones
Opiniones recibidas
Opiniones dejadas

Información personal
Nota: campos con * son obligatorios

 Quieres eliminar esta foto

Posted by: Hege Refsnes
Contact information: someone@example.com



Pantalla Publicar anuncio/Buscar coche para compartir

A Web Page
http://

Home Editor perfil Viajes publicados Buscar Publicar ComboBox ▾

Publicar Ruta
Nota: campos con * son obligatorios

coche:

Salida:

Precio:

Plazas disponible

Detalles
Ejemplo: Voy a La universidad de Alicante todos los días, salgo de Torrevieja, tengo clases solo por las mañanas

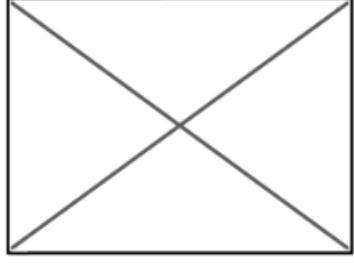
Posted by: Hege Refsnes
Contact information: someone@example.com.

A Web Page
http://

Home Editor perfil Viajes publicados Buscar Publicar ComboBox ▾

Buscar Ruta
Nota: campos con * son obligatorios

Salida:



Posted by: Hege Refsnes
Contact information: someone@example.com.

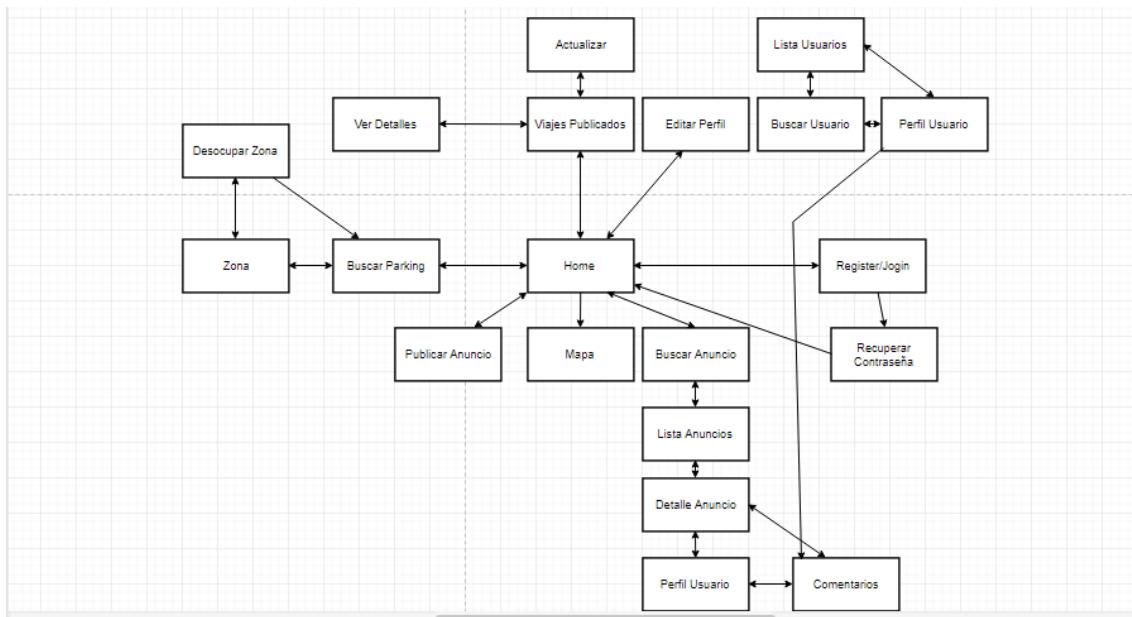


➤ Móvil

Wireframe

Desde home podemos acceder a las páginas funciones principales del menú de la cabecera. Además también podrás acceder a tu cuenta mediante login y registro. Una vez se registra o se inicia sesión, se podrá acceder a todas las funcionalidades de la app, por ejemplo editar perfil, donde el usuario puede actualizar sus datos, viajes publicados donde el usuario tiene todo el historial de sus publicaciones, con los cuales puede ver información al respecto o eliminarlos.

Tambien puede buscar anuncio, buscar usuario, ver perfil de usuario, ver detalle de anuncio, etc.



DISEÑO FINAL

Web

Pantalla principal de la aplicación

En la pantalla principal, se encuentra la formación que provee la aplicación, Como las funcionalidades, el registro y el login, Además cualquier usuario tiene la opción de ver los parkings de la ua, sin que este dado de alta.

The screenshot shows the main page of the application. At the top, there is a navigation bar with links for Home, Quien somos, Contact, Register, and Login. The main content area features a large image of a modern university building reflected in water. Below the image, there is a section titled "Cómo funciona?" (How it works) with six items arranged in a grid:

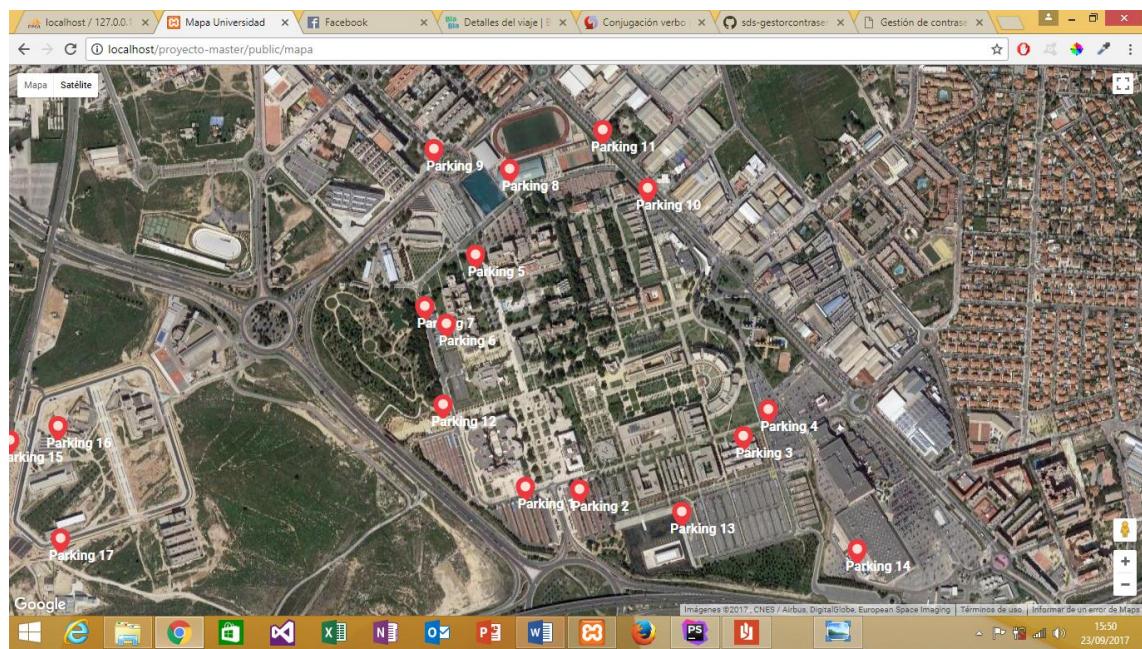
¿Vas a la universidad?	Confirmas la reserva	¡A viajar!
Encuentras usuarios que van a la universidad de Alicante.	Los usuarios reservan las plazas libres online.	Compartes coche y dejas una opinión a tus compañeros.
¿Buscas aparcamiento?	Confirmas el aparcamiento	¡Apurar!
Encuentras hueco en los aparcamientos de la ua.	Los usuarios reservan las plazas libres online.	una vez sales de tu aparcamiento, ha de indicar que el espacio esta libre.

At the bottom of the main content area, there are three buttons: "Compartir coche con Usuarios", "Ver parkings de la universidad", and "Buscar parking en la". The taskbar at the bottom of the screen shows various open applications like Facebook, Google, and Microsoft Office.

The screenshot shows the main page of the application with three main sections:

- Compartir coche con Usuarios de la universidad:** An image of four people in a car. Description: Los usuarios que pretenden buscar alguien con un coche que a su vez necesita gente para ir o venir, mediante la uaapp, se puede buscar quien se ofrece a llevar gente en su coche, el precio es muy justo, no se puede ganar mas de lo que se pueda.
- Ver parkings de la universidad:** An aerial map of the university campus showing various buildings and parking areas.
- Buscar parking en la universidad sin perder mucho tiempo:** An image of a full parking lot. Description: Ahora ya no se pierde mucho tiempo para buscar aparcamiento en la ua, solo por ser usuario de la uaapp, se sabe las zonas que estan libres, y asi segun el aparcamiento donde quiera el usuario aparcar, lo consulta, y le sale el resultado, de los huecos libres.

At the bottom of the main content area, there is a footer note: "Uaapp es una app que pretende ayudar a la comunidad universitaria a compartir coche y buscar aparcamientos de la ua. Uaapp conecta usuarios registrados que parten del mismo lugar o se encuentran en misma ruta por donde pasa el propietario del coche, así como encontrar un aparcamiento, sin perder mucho tiempo." The taskbar at the bottom of the screen shows various open applications like Facebook, Google, and Microsoft Office.



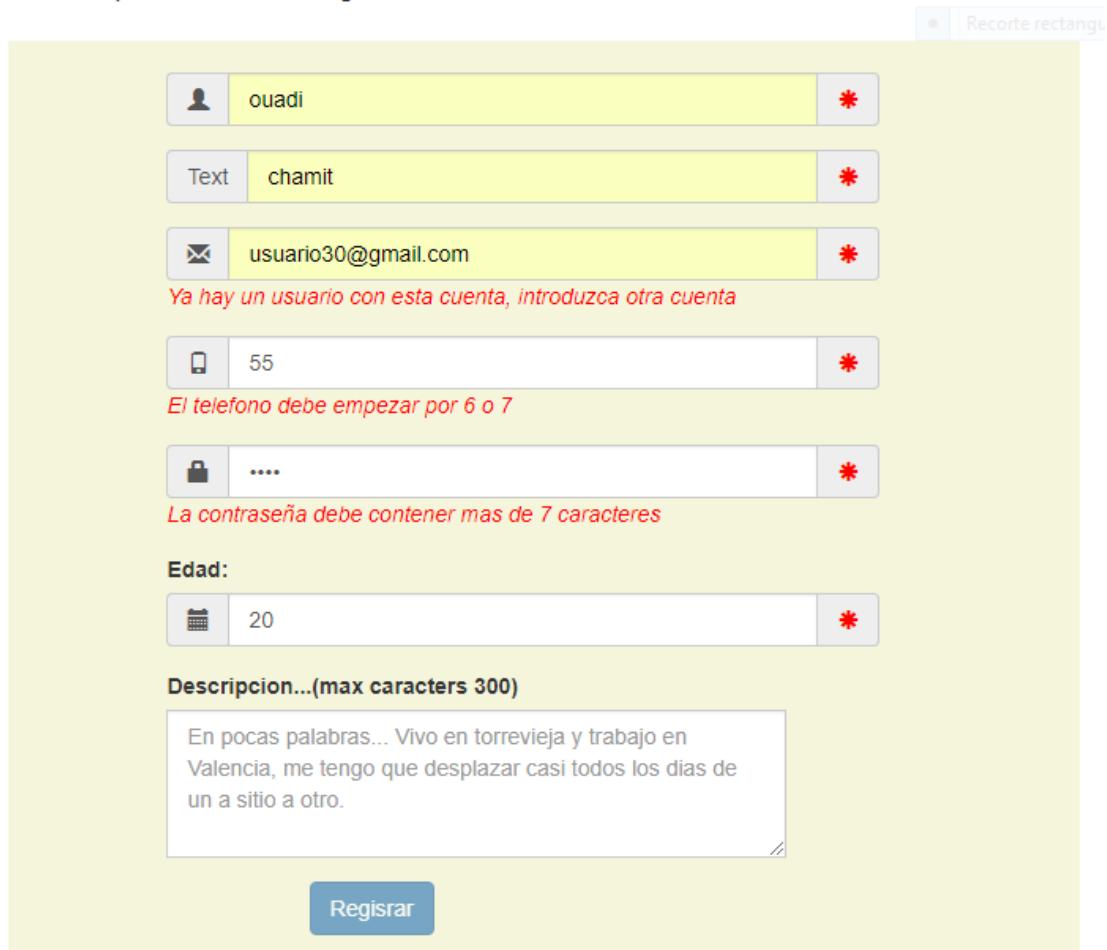
Pantalla de Registro de Usuario

Esta pantalla permite a los usuarios crear una nueva cuenta. Como se puede observar, Consta de varios campos todos son obligatorios, indicados con el asterisco rojo, controlados con la propiedad de html required, la cual impide hacer un submit si un campo esta vacío.

Además, se comprueba el campo correo con Ajax para que no sea duplicado en la base de datos, ya que es un dato único, También el campo teléfono y contraseña son controlados con Jquery para que sean datos validos.

Una vez se comprueba todo que es correcto, cuando el controller registro recibe los datos de parte del cliente, se mantiene todos los datos en plano, menos la contraseña que se almacena de forma cifrada, para realizar el cifrado, se Utiliza SHA-512 como hash sobre la contraseña, se crea una salt aleatoria. Esta salt servirá para “mezclarla” con el hash creado y realizar un encriptado con “crypt”, y por ultimo realizar el almacenamiento en la base de datos.

Nota: campos con * son obligatorios



The screenshot shows a user registration form with several fields and validation messages:

- Nombre:** ouadi (Required)
- Apellido:** chamit (Required)
- Correo:** usuario30@gmail.com (Required)
Ya hay un usuario con esta cuenta, introduzca otra cuenta
- Teléfono:** 55
El telefono debe empezar por 6 o 7
- Contraseña:**
La contraseña debe contener mas de 7 caracteres
- Edad:** 20 (Required)
- Descripción (max 300 caracteres):** En pocas palabras... Vivo en torrevieja y trabajo en Valencia, me tengo que desplazar casi todos los días de un a sitio a otro.

A blue "Registrar" button is at the bottom right of the form.

Cuando se realiza el registro de forma correcta, se le envía al usuario un correo electrónico, para confirmar el registro pulsando sobre el enlace enviado, en caso contrario, no puede acceder al sistema.

Email Test Recibidos x Imprimir Compartir

chamit 20 sept. (hace 3 días) istar Responde Detalles

para mí ▼

haga click en el enlace abajo para activar tu cuenta, y poder utilizar la aplicación <http://localhost/proyecto-master/confirmRegistro/everis@gmail.com>

▼ Haz clic aquí para [Responder](#) o para [Reenviar](#)

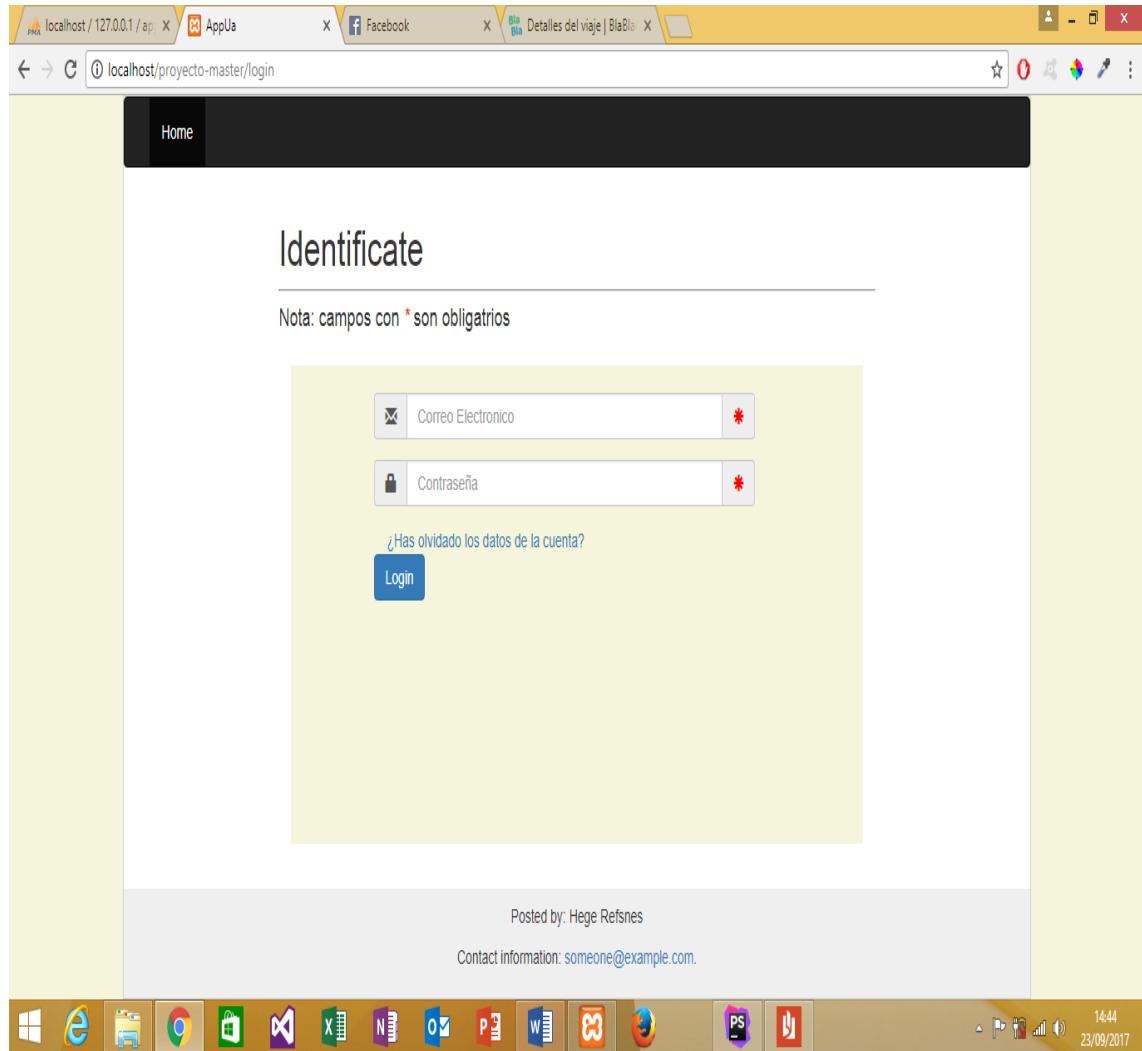
12,95 GB (86%) ocupados de 15 GB [Administrar](#) [Condiciones](#) - [Privacidad](#)

Última actividad de la cuenta: hace 4 días [Información detallada](#)

Pantalla de Inicio de Sesión

La pantalla de inicio de sesión sirve para que los usuarios puedan acceder con sus credenciales a la aplicación.

La pantalla tendrá varias campos, un botón para realizar la petición, un enlace en caso de perder la contraseña y dos campos para introducir texto en el cuál se deberán introducir las credenciales de acceso del usuario, esto es, nombre de usuario y contraseña.



Una vez esta diseñada la pantalla, se implementa una función para comprobar la coincidencia de los datos proveídos con los datos que están en la base de datos, esta función Utiliza SHA-512 como hash sobre la contraseña, se recupera la salt aleatoria de la base de datos a base del correo introducido, porque cada usuario tiene un campo salt, con la cual se hace la mezcla con el hash con la función crypt y se guarda la contraseña mezclada. Este salt servirá para “mezclarla” con el hash relalizado sobre la contraseña y realizar un encriptado con “crypt” y por tanto comprobar la coincidencia de las contraseñas si son correctas.

Identificate

Nota: campos con * son obligatorios

The screenshot shows a login form with two fields: 'Correo Electronico' and 'Contraseña'. Both fields have red asterisks indicating they are required. A pink error message box at the top says 'Info! La contraseña es incorrecta'. Below the fields is a link '¿Has olvidado los datos de la cuenta?' and a blue 'Login' button.

Como se observa en la captura, si la contraseña es incorrecta, se le indica al usuario con el mensaje con color morado. En caso que el correo no existe en la base de datos se le indica con el mensaje reflejado en la captura siguiente:

Identificate

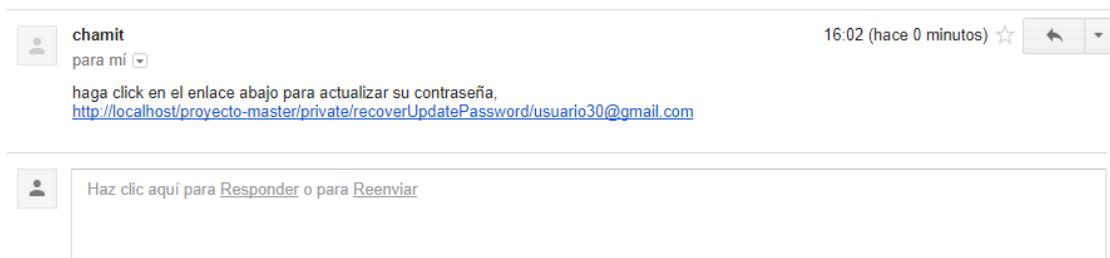
Nota: campos con * son obligatorios

The screenshot shows a login form with two fields: 'Correo Electronico' and 'Contraseña'. Both fields have red asterisks indicating they are required. A pink error message box at the top says 'Info! Los datos son incorrectos'. Below the fields is a link '¿Has olvidado los datos de la cuenta?' and a blue 'Login' button.

Pantalla recuperación de contraseña

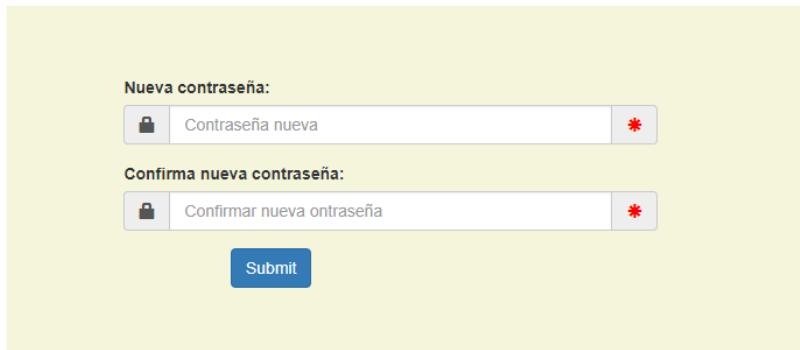
Si el usuario no se acuerda de su contraseña, bajo cualquier constancia, tiene la posibilidad de recuperar su cuenta, solo introducir su correo, pulsar botón send, recibirá un correo con un enlace mediante el cual se le lleva a otra vista donde introduzca su nueva contraseña.

The screenshot shows a web page titled "Introduzca tu correo" (Enter your email). A note below it states: "Nota: campos con * son obligatorios" (Note: fields with * are mandatory). There is a text input field containing "usuario30@gmail.com" with a small mail icon to its left. To the right of the input field is a red asterisk (*) symbol. Below the input field is a blue "Send" button. The background of the form area is light yellow.



Gestión de la clave

Nota: campos con * son obligatorios



Nueva contraseña:

Contraseña nueva *

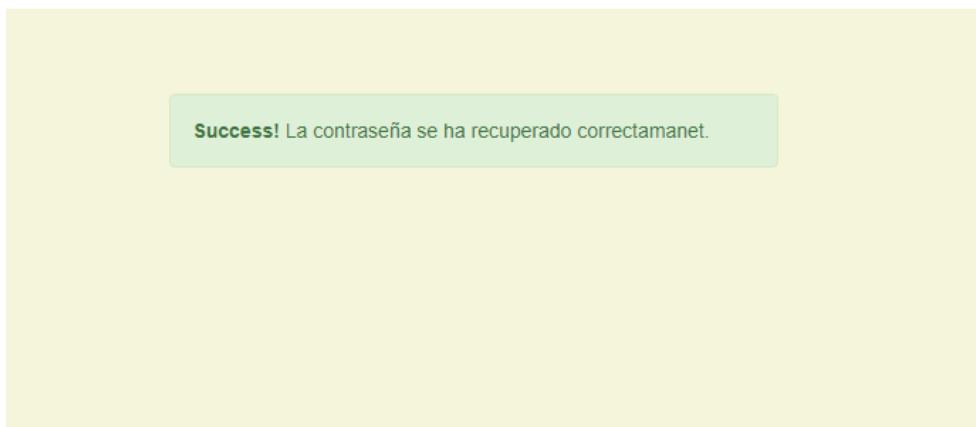
Confirmar nueva contraseña:

Confirmar nueva contraseña *

Como resultado de la recuperación de la cuenta, se le notifica al usuario con el mensaje “La contraseña se ha recuperado correctamente”

Gestión de la clave

Nota: campos con * son obligatorios



Success! La contraseña se ha recuperado correctamente.

Pantalla Editar Perfil

Como se observa en la captura, cuando el usuario inicia sesión, aparecen los menús, uno horizontal y dos verticales, los cuales llevan a todas las funcionalidades de la aplicación, por ejemplo en la captura siguiente, el usuario puede editar la información personal, su foto, coche, ver comentarios hechos y recibidos y actualizar contraseña etc.

La captura muestra la interfaz de usuario para editar un perfil. En la parte superior, hay un menú horizontal con enlaces a 'Home', 'Editar perfil', 'Viajes publicados', 'Buscar anuncio', 'Publicar anuncio', 'Buscar parking' y 'Bienvenido usuario30'. Un menú vertical desplegable aparece sobre el enlace 'Bienvenido usuario30', mostrando opciones como 'Editar perfil', 'CSS', 'Viajes publicados', 'Buscar usuario' y 'Cerrar sesion'. El contenido principal es un formulario titulado 'Información personal'. Se incluye una nota que dice 'Nota: campos con * son obligatorios'. Los campos obligatorios llenados son: 'Nombre' (usuario30), 'Apellido' (usuario30), 'Correo electrónico' (usuario30@gmail.com) y 'Teléfono' (654879321). El campo 'Edad' (22) no es obligatorio. A continuación del formulario hay un botón 'Submit'.

La captura muestra la misma interfaz de usuario que la anterior, pero con un mensaje de éxito en la parte superior: 'Success! Los datos se han actualizado correctamente.' El resto del formulario y su contenido es idéntico al de la captura anterior.

Pantalla editar foto

En esta pantalla el usuario, puede poner foto, borrarla y cambiarla por otra.

The screenshot shows the 'Foto de perfil' (Profile Photo) section of a user profile. At the top, there is a navigation bar with links: Home, Editar perfil, Viajes publicados, Buscar anuncio, Publicar anuncio, Buscar parking, and Bienvenido usuario30. On the left, there is a sidebar with sections: Información de perfil (with fields for informacion personal, foto, and coche), Opiniones (with fields for Opiniones recibidas and Opiniones hechas), and Cuenta (with fields for Contraseña and Darse de baja). The main area is titled 'Foto de perfil' and contains a placeholder text: 'Añade una foto para tu perfil. A los demás usuarios les tranquilizará saber con quién van a viajar, y a ti te será más fácil encontrar un viaje. Las fotos os ayudarán a reconoceros en el punto de encuentro.' Below this text is a placeholder image of a man wearing glasses and a dark jacket. A blue button at the bottom left of the placeholder image says 'Quieres eliminar esta foto' (Do you want to delete this photo?).

Cuando el usuario, no tiene foto o la borra, en esta vista puede incluir una.

The screenshot shows the 'Foto de perfil' (Profile Photo) section of a user profile. The layout is identical to the previous screenshot, with the same navigation bar and sidebar. The main area is titled 'Foto de perfil' and contains a placeholder text: 'Añade una foto para tu perfil. A los demás usuarios les tranquilizará saber con quién van a viajar, y a ti te será más fácil encontrar un viaje. Las fotos os ayudarán a reconoceros en el punto de encuentro.' Below this text is a placeholder image of a black Sony digital camera. To the right of the camera image, there is a file upload interface with the text 'Upload:' and a button labeled 'Seleccionar archivo'. Below this, another button says 'Ningún archivo seleccionado'. At the bottom of the placeholder area is a blue 'Submit' button.

Pantalla opiniones realizados

Home Editar perfil Viajes publicados Buscar anuncio Publicar anuncio Buscar parking Bienvenido usuario30 ▾

Informacion de perfil
informacion personal
foto
coche

Opiniones
Opiniones recibidas
Opiniones hechas

Cuenta
Contraseña
Darse de baja

Comentario realizados

Opiniones

usuario30usuario300 Todo perfecto, puntualidad y seriedad

usuario30usuario300 Muy buen conductor. Un viaje muy cómodo y agradable.

usuario30usuario300 Es un chico muy agradable, fuimos hablando todo el camino, es muy flexible en hora y lugar.

123Siguiente >

Pantalla comentarios recibidos

Home Editar perfil Viajes publicados Buscar anuncio Publicar anuncio Buscar parking Bienvenido usuario30 ▾

Informacion de perfil
informacion personal
foto
coche

Opiniones
Opiniones recibidas
Opiniones hechas

Cuenta
Contraseña
Darse de baja

Comentario recibidos

Opiniones

usuario11usuario11 Muy simpático. 100% recomendable.

usuario1000usuario100 Genial, un chico super simpático divertido y muy buen conductor, repetiría sin dudar. Hasta la próxima!

OuadiOuadiChamit Un chico muy correcto y servicial. Llevó a mi madre y hermana de Alcoy a Valencia y les hizo el favor de recogerlas en otro sitio porque hacía mal tiempo. Y el viaje fue genial. Para repetir

Pantalla publicar anuncio

En esta vista el usuario, publica un anuncio, para buscar usuarios con el fin de compartir el coche, el usuario, pone el punto de salida, el precio, las plazas disponibles y un detalle del anuncio.

Si el usuario no dispone de un coche, no puede publicar ningún anuncio, ya que cuando introduzca los datos y los envíe al servidor, se le notifica con un mensaje de que debe introducir datos del su coche

The screenshot shows a web application interface for publishing a route. At the top, there is a navigation bar with links: Home, Editar perfil, Viajes publicados, Buscar anuncio (with a magnifying glass icon), Publicar anuncio (with a car icon), Buscar parking, and Bienvenido usuario30 (with a dropdown arrow). Below the navigation bar, the main content area has a title "Pblicar Ruta". A note below the title states: "Nota: campos con * son obligatorios". The form itself is divided into several sections: "Salida" (with a "Text" input field containing "Ejemplo: Elche" and a red asterisk), "Precio" (with a "€" input field containing "Ejemplo: 3" and a red asterisk), "Plazas disponibles" (with an "N" input field containing "Ejemplo: 3" and a red asterisk), and "Detalles" (with a text area containing "Ejemplo: Voy a La universidad de Alicante todos los dias, salgo de Torrevieja, tengo clases solo por las mañanas"). At the bottom of the form is a blue "Publicar" button.

Pblicar Ruta

Nota: campos con * son obligatorios

Info! No tienes coche para publicar anuncio, ponga datos del coche, y luego, empiece a publicar anuncios.

Pantalla buscar parking

Cuando el usuario busca parking, la opción Buscar parking lo lleva a una vista con el mapa de la universidad, con los parkings indicados con los puntos rojos.

Home Editar perfil Viajes publicados Buscar anuncio Publicar anuncio Buscar parking Bienvenido usuario30 ▾

Buscar Parking

Elige parking

Elige parking:
parking 1

Buscar aparcamiento

Mapa Satélite

The map displays a satellite view of a university campus. Eleven parking zones are marked with red dots and labeled Z6 through Z11. Zone Z6 is located near a building complex. Zone Z7 is near a green area. Zone Z8 is near a large blue swimming pool. Zone Z9 is near a cluster of buildings. Zone Z10 is near a road. Zone Z11 is near a green area. There are also other buildings, roads, and green spaces visible on the map.

Pantalla Elegir zona

Mediante la opción del select, cuando el usuario elige un parking mediante su numero, se le lleva otra vista con las zonas vacías de este parking.

Elige Zona para aparcar y actualizar para indicar como zona ocupada:

Z 461

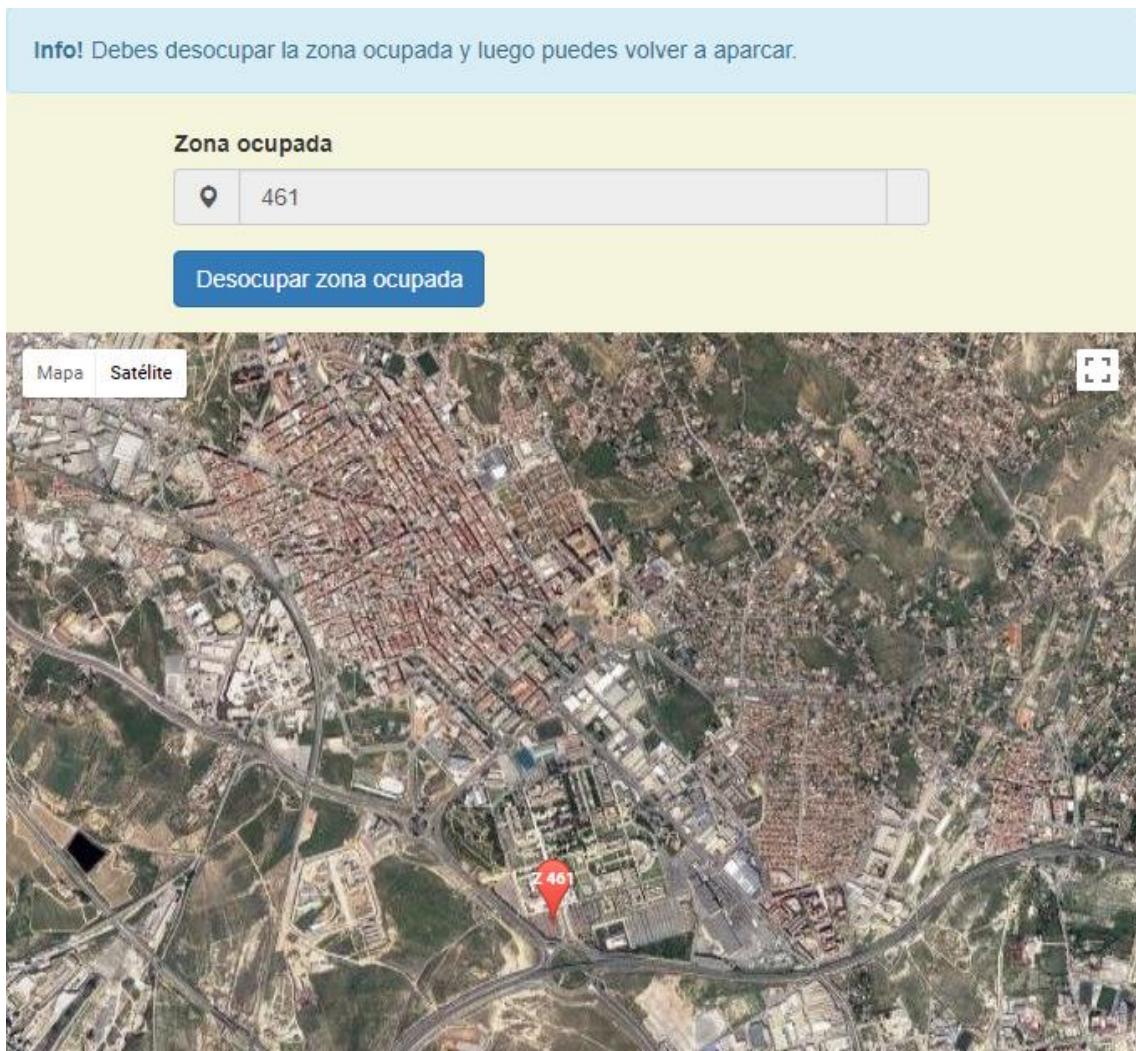
Actualizar Zona como ocupada

Mapa Satélite

A satellite map of a city area, likely Madrid, showing a dense urban grid. A red marker with the text "Z 461" is placed on a building complex near a highway interchange. The map includes a legend at the top left with "Mapa" and "Satélite" options, and a zoom control icon at the top right.

Pantalla Desocupar zona

Cuando el usuario indica que ha ocupado una zona dada, se le lleva a una vista para desocupar esta zona, además, si no la desocupa, cuando vuelve a buscar zonas vacías en los parkings, siempre se le lleva a esa vista para marcar como desocupada la zona, así puede volver a buscar huecos.



Pantalla buscar anuncio

En esta pantalla el usuario, pone un origen, en el campo origen del formulario, como resultado, le salen todos los anuncios desde ese origen con plazas disponibles.

Screenshot of the 'Buscar Ruta' (Search Route) page. The top navigation bar includes 'Home', 'Editar perfil', 'Viajes publicados', 'Buscar anuncio' (with a magnifying glass icon), 'Publicar anuncio' (with a camera icon), 'Buscar parking', and 'Bienvenido usuario30'. Below the navigation is a search form with a 'Salida' field containing 'Ejemplo: Elche' and a red asterisk indicating it's required. A 'Buscar' button is next to the field. To the right is a map showing a route from Elche to San Vicente del Raspeig. The map highlights two routes: one taking 52 min and 64.3 km, and another taking 50 min and 71.4 km. The map also shows other locations like Monóvar, Novelda, Aspe, and Alicante.

Como resultado, le sale una vista en modo paginación para ver todos los anuncios.

Screenshot of the '12 viajes disponibles de a la universidad de alicante' (12 trips available to the University of Alicante) page. The top navigation bar is identical to the previous page. The main content displays four trip listings:

- aux aux** (Profile picture of a baby, 22 years old)
Elche -> Universidad de Alicante **3 €**
por plaza
2 plazas disponibles
- usuario2 usuario2** (Profile picture of a man, 2 years old)
Elche -> Universidad de Alicante **4 €**
por plaza
2 plazas disponibles
- usuario3 usuario3** (Profile picture of a person at a computer, 2 years old)
Elche -> Universidad de Alicante **3 €**
por plaza
3 plazas disponibles
- usuario4 usuario4** (Profile picture of a person at a computer, 3 years old)
Elche -> Universidad de Alicante **3 €**
por plaza
3 plazas disponibles

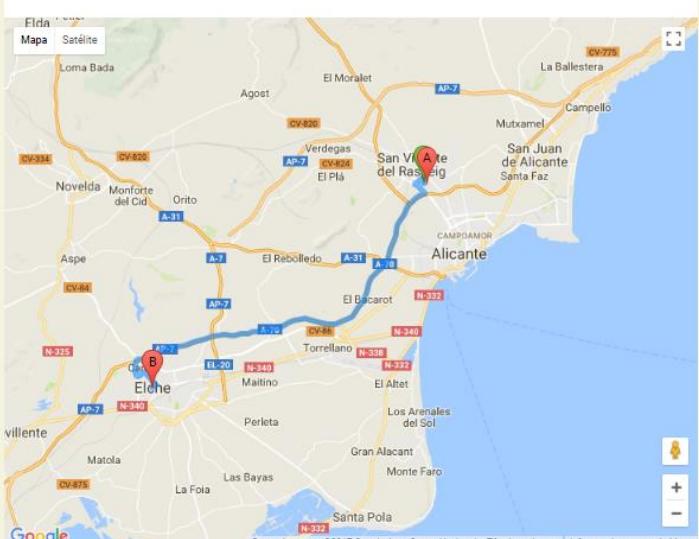
	usuario4 usuario4 3 años	Elche -> Universidad de Alicante	3 € por plaza 3 plazas disponibles
	usuario5 usuario5 2 años	Elche -> Universidad de Alicante	3 € por plaza 3 plazas disponibles

123Siguiente ->

Cuando el usuario selecciona un anuncio le sale todos los datos del anuncio en otra vista

Con una mapa indicando la ruta desde el punto de salida hasta la universidad

Elche --> Universidad de Alicante

Salida:	Elche	Precio	3 €
Fecha Publicacion	2017-07-09	Plazas	2 disponibles
Detalles: Los puntos de recogida y dejada son en Valencia plaza de toros, mestalla, rotonda de los anzuelos, en Alicante renfe, estación de autobuses, aeropuerto, telepizza en san vicente, en Elche estación de tren elche carrus y parque, estación de autobuses, en torrevieja estación de autobuses.			
		Conductor  aux aux 22 años	
Coche  A15 Categoría: TURISMO Color: BLANCO		Opiniones  usuario30usuario30 capullo usuario30usuario30 khjhvhvjhbh	
Ver todos los comentarios			
Actividades Ver Perfil			

Administración

En la parte de administración, donde el usuario que tiene privilegios de administrador, puede gestionar la aplicación, dar de alta, modificar, borrar, actualizar toda la información contenida en la aplicación, en esta parte he utilizado una librería que provee el framework, la librería permite crear un formulario que contiene toda la información de una tabla en la base de datos, así como permitir hacer la operaciones crud, y buscar información por filtro. Para llevar acabo el desarrollo, en el controlador se realiza los pasos siguientes:

```
$crud = new grocery_CRUD();
$crud->set_table('coche');
$crud->set_subject('coche');
$crud->set_theme("datatables");
```

En el ejemplo anterior la librería crea un objeto coche como un formulario que tiene toda la información de los coches dados de alta en el sistema, en este formulario, se pueden realizar todas las operaciones, incluso buscar por filtro.

Coches Gestión de coches								
<input type="button" value="Add coche"/> <input type="button" value="Print"/>								
Show <select>10</select> entries <input type="text" value="Search:"/>								
Matricula	Modelo	Color	Acientos	Usuario	Marca	Categoría	ImageFoto	
11111	10	BLANCO	4	occ10@alu.ua.es	A15	TURISMO	<input type="button" value="View"/> <input type="button" value="Edit"/>	
3214BKC	10	BLANCO	4	usuario10@gamail.com	A15	TURISMO	ferrari2.jpg	<input type="button" value="View"/> <input type="button" value="Edit"/>
3214BKK	10	BLANCO	2	occ10@alu.ua.es	A15	TURISMO	<input type="button" value="View"/> <input type="button" value="Edit"/>	
3214LLL	10	BLANCO	2	usuario30@gmail.com	A15	TURISMO	ferrari22.jpg	<input type="button" value="View"/> <input type="button" value="Edit"/>
auxx	10	BLANCO	2	aux@aux.com	A15	TURISMO	<input type="button" value="View"/> <input type="button" value="Edit"/>	
usuario1	10	BLANCO	2	usuario1@gamail.com	A15	TURISMO	<input type="button" value="View"/> <input type="button" value="Edit"/>	
usuario10	10	BLANCO	2	usuario10@gamail.com	A15	TURISMO	<input type="button" value="View"/> <input type="button" value="Edit"/>	
usuario11	10	BLANCO	2	usuario11@gamail.com	A15	TURISMO	<input type="button" value="View"/> <input type="button" value="Edit"/>	
usuario12	10	BLANCO	2	usuario12@gmail.com	A15	TURISMO	<input type="button" value="View"/> <input type="button" value="Edit"/>	
usuario2	10	BLANCO	2	usuario2@gamail.com	A15	TURISMO	<input type="button" value="View"/> <input type="button" value="Edit"/>	
<input type="text" value="Search Matricu"/>	<input type="text" value="Search Mod"/>	<input type="text" value="Search Ci"/>	<input type="text" value="Search Acient"/>	<input type="text" value="Search Usuario"/>	<input type="text" value="Search Ma"/>	<input type="text" value="Search Catego"/>	<input type="text" value="Search ImageFo"/>	<input type="button" value="Clear"/>

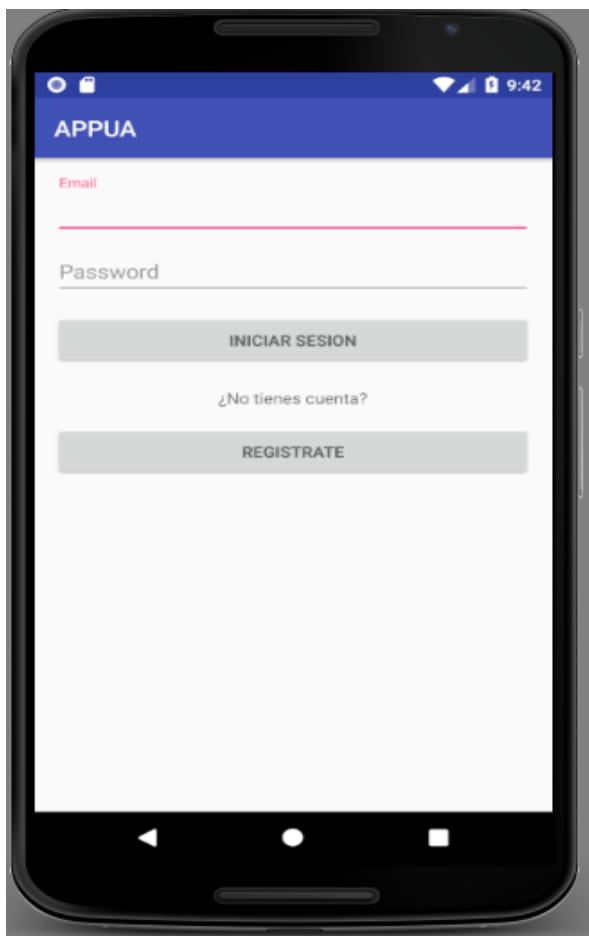
Screenshot of a web application interface titled "AppUA Admin". The main title is "Coches Gestión de coches". The left sidebar contains navigation links: Inicio, Usuarios, Coches, Ruta, Parking, Zona, Comentarios, and Historial Parking. The main content area shows a table of car data with columns: Matricula, Modelo, Color, Acientos, Usuario, Marca, Categoría, and ImageFoto. The table has 10 entries. The bottom of the screen shows a taskbar with various icons and a system tray indicating the date and time.

Matricula	Modelo	Color	Acientos	Usuario	Marca	Categoría	ImageFoto		
11111	10	BLANCO	4	occ10@alu.ua.es	A15	TURISMO			
3214BKC	10	BLANCO	4	usuario10@gmail.com	A15	TURISMO	ferrari2.jpg		
3214BKK	10	BLANCO	2	occ10@alu.ua.es	A15	TURISMO			
3214LLL	10	BLANCO	2	usuario30@gmail.com	A15	TURISMO	ferrari22.jpg		
auxx	10	BLANCO	2	aux@aux.com	A15	TURISMO			
usuario1	10	BLANCO	2	usuario1@gmail.com	A15	TURISMO			
usuario10	10	BLANCO	2	usuario10@gmail.com	A15	TURISMO			
usuario11	10	BLANCO	2	usuario11@gmail.com	A15	TURISMO			
usuario12	10	BLANCO	2	usuario12@gmail.com	A15	TURISMO			
usuario2	10	BLANCO	2	usuario2@gmail.com	A15	TURISMO			

Móvil

Pantalla de Inicio de Sesión

La pantalla de inicio de sesión sirve para que los usuarios puedan acceder con sus credenciales a la aplicación. La pantalla tendrá varias etiquetas de texto, dos botones uno para registrarse y otro para iniciar sesión, dos campos para introducir texto en el cuál se deberán introducir las credenciales de acceso del usuario.



En caso las credenciales no son correctas , se le notificara al usuario, que debe confirmar su registro mediante el correo enviado cuando se dio de alta o el login no es correcto si tiene la cuenta confirmada, y si son correctas, se creará la sesión del usuario y se abrirá la pantalla principal con el usuario conectado. al hacer login recibimos todos los datos del usuario en formato json.

```
{"correo":"kkk@kkk.com","nombre":"kkk","apellido":"kkkkk","edad":33,"telefono":"654123987","foto":"c:/uploadedFiles/71mv2ecGuBL.jpg","detalles":"lo que sea love question sea","opcion":2,"confirmado":"SI"}
```

Con esos datos creamos un objeto de la clase usuario y enviamos ese objeto a la pantalla principal de la siguiente forma:

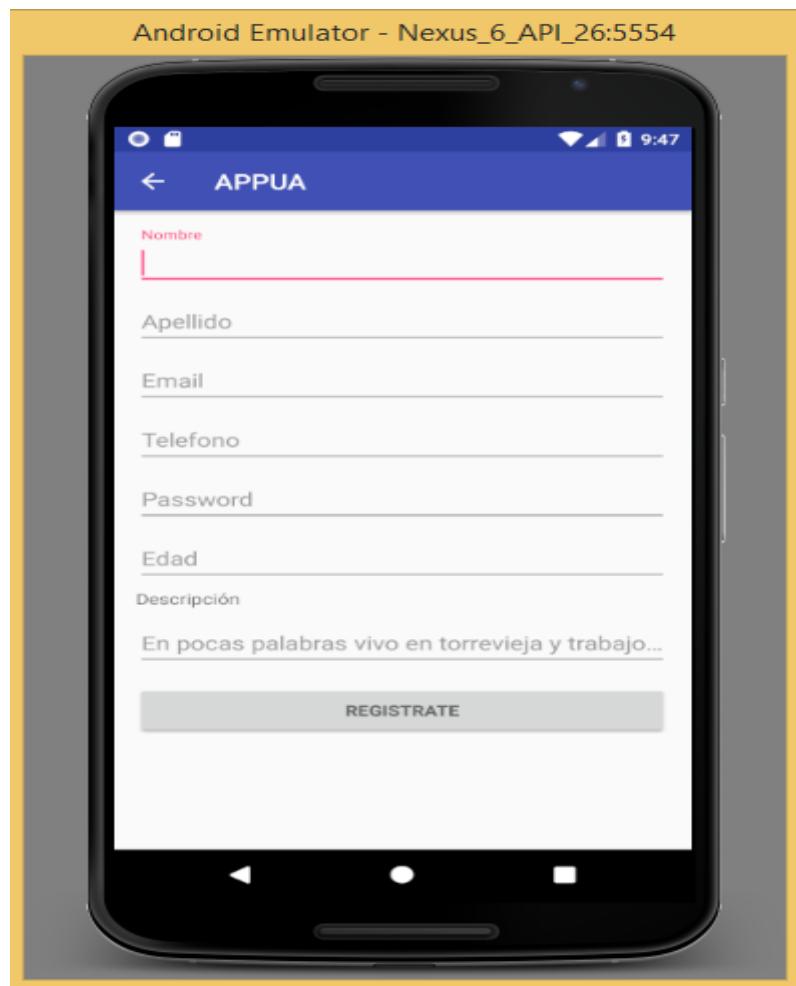
```
SharedPreferences.Editor editor = sharedpreferences.edit();
editor.putString("usuario", usuario.getEmail());
editor.commit();
intent.putExtra("usuario", usuario);
startActivity(intent);
```

Para poder pasar un objeto de una actividad a otra actividad, la clase usuario debe implementar la interface Parceable de cada modelo. Hacemos esto para poder convertir el objeto en bytes y poder pasarlo a la otra actividad. Para recuperar el objeto usuario desde la otra actividad (actividad principal) que contiene a su vez tres actividades que son fragmentos, lo hacemos de la siguiente forma:

```
Bundle extras = getActivity().getIntent().getExtras();
usuario = extras.getParcelable("usuario");
```

Pantalla de Registro de Usuario

Esta pantalla permite a los usuarios crear una nueva cuenta. Como se puede observar, contiene varios campos obligatorios, mas adelante en otra pantalla se puede editar el perfil para poner foto y modificar los datos en caso que sea necesario.



Al igual que la anterior pantalla, al pulsar en el botón de registrarse, si los campos no son válidos se mostrará un mensaje al usuario para que pueda solucionarlo y vea el motivo por el cual no son correctos los datos introducidos.

Por ejemplo si un campo no contiene datos, se le notifica al usuario con un mensaje de la forma siguiente.



Y si el dato no es correcto.



Si al intentar registrarnos y se produce algún fallo se notificará al usuario con un mensaje que ha habido un error en el servidor, como se observa en las líneas siguientes.

```
Toast.makeText(RegistryActivity.this,  
    "try mas later, an error has occurred in the server",  
    Toast.LENGTH_LONG).show();
```

Cuando Los datos son correctos, se da de alta al usuario en el sistema, y a la vez recibe un mensaje para confirmar su registro, si no confirma el registro, no se le permite acceder al sistema.

Pantalla principal de la aplicación

Como he mencionado antes, la pantalla principal se compone de tres fragmentos que están dentro de una actividad, cada fragmento tiene un contenido en concreto.

Para usar esta organización, debemos utilizar TabHost y TabWidget, TabHost debe ser el nodo raíz del diseño y contendrá al TabWidget. En el TabWidget tendremos las pestañas, el FragmentTabHost tendrá bajo el TabWidget un FrameLayout para mostrar el contenido. Tanto el TabWidget como el FrameLayout se agruparán dentro de un LinearLayout que los contenga.

```
<android.support.v4.app.FragmentTabHost
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tabhost"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
        <TabWidget
            android:id="@+id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom" />
        <FrameLayout
            android:id="@+id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:layout_weight="1" />
    </LinearLayout>
</android.support.v4.app.FragmentTabHost>
```

Y dentro de la actividad contendora, se siguen los pasos siguientes.

Se crea el objeto FragmentTabHost, para contener el layout.
private FragmentTabHost tabHost;
tabHost = (FragmentTabHost) findViewById(android.R.id.tabhost);

Se le asigna el FrameLayout para
tabHost.setup(this, getSupportFragmentManager(),
 android.R.id.tabcontent);
Resources res = getResources();

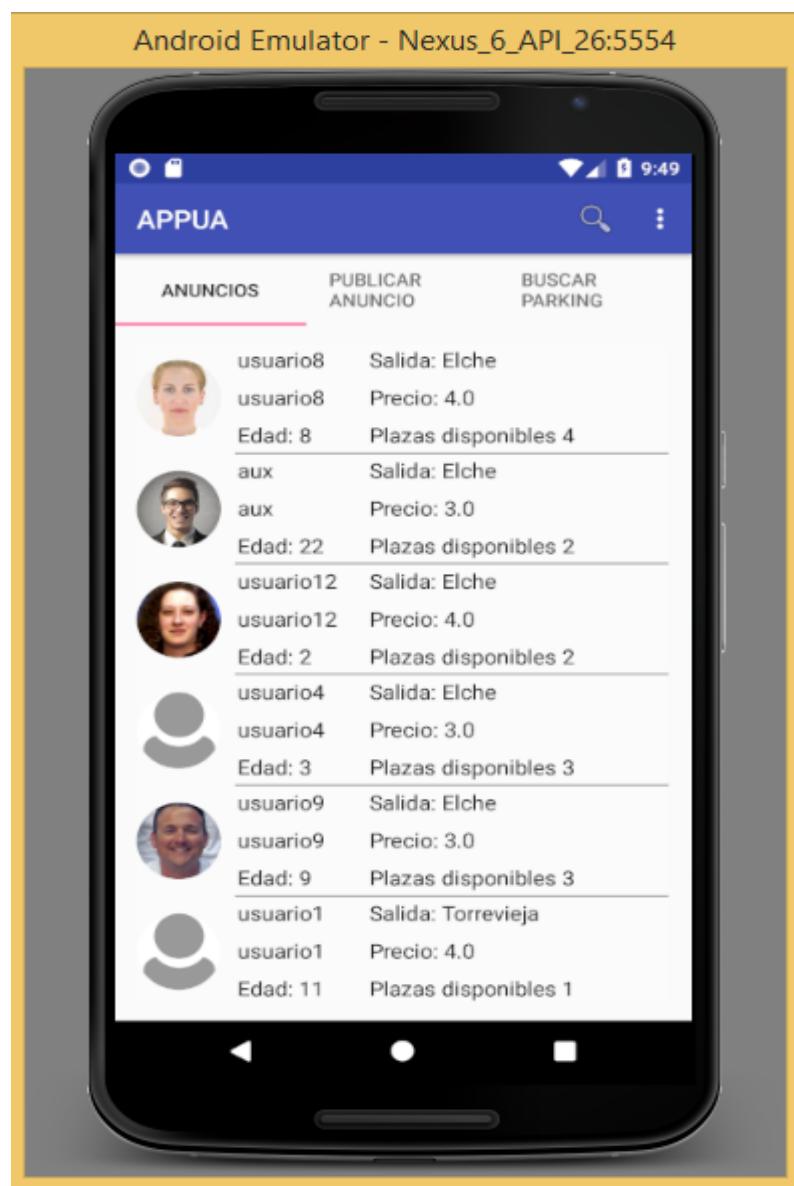
se crean las pestañas.
TabHost.TabSpec tab1 = **tabHost.newTabSpec("tab1");**
TabHost.TabSpec tab2 = **tabHost.newTabSpec("tab2");**
TabHost.TabSpec tab3 = **tabHost.newTabSpec("tab3");**

A cada pestaña, se le asigna una etiqueta
tab1.setIndicator("Anuncios", null);
tab2.setIndicator("Publicar Anuncio", null);
tab3.setIndicator("Buscar Parking", null);

Cada pestaña se le relaciona con un fragmento.
tabHost.addTab(tab1, FragmentTabOne.class, null);
tabHost.addTab(tab2, FragmentTabTwo.class, null);
tabHost.addTab(tab3, FragmentTabTree.class, null);

Como resultado de los pasos anteriores se crea la vista de la forma como se observa en la captura siguiente.

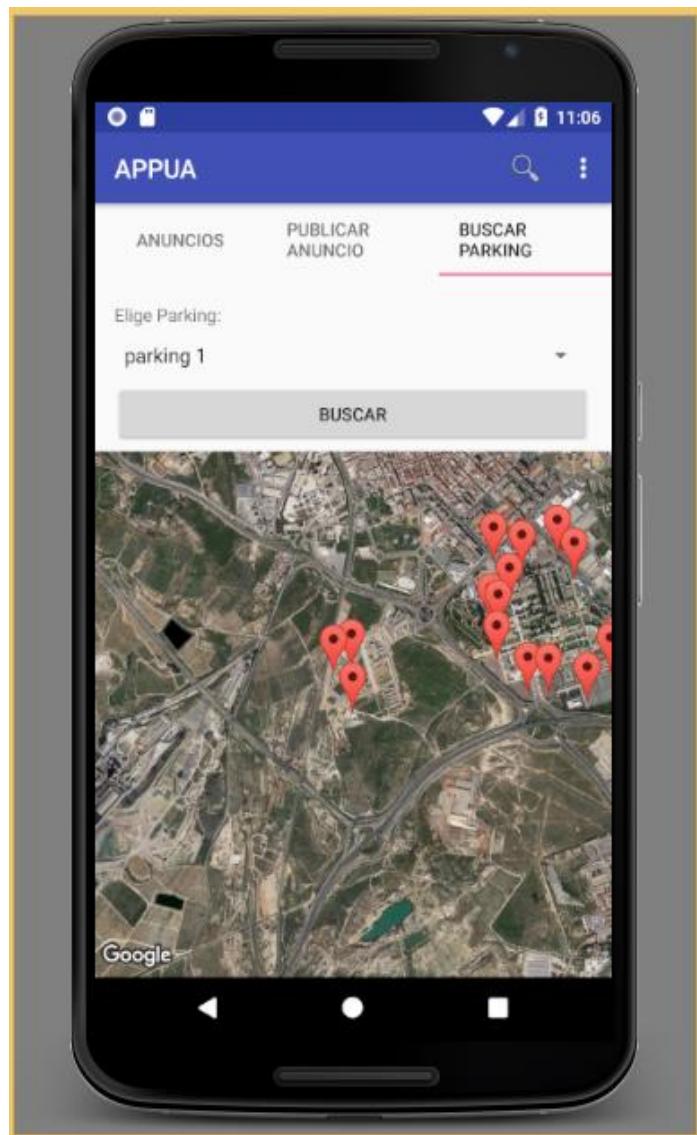
Por ejemplo en la primera pestaña, se demuestran los anuncios ordenados por fecha, para ver los últimos en publicar, si le resultan de interés al usuario.



En la pestaña PUBLICAR ANUNCIO, le permite al usuario poner un anuncio nuevo.



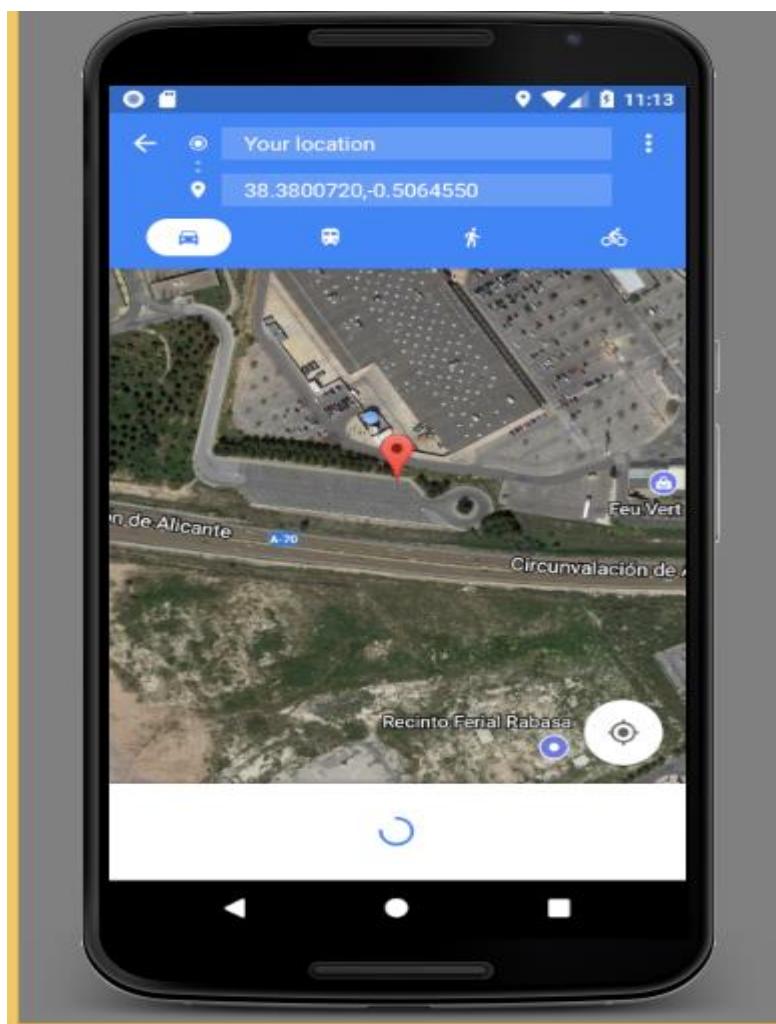
En la pestaña BUSCAR APARCAMIENTO, se le demuestra al usuario una etiqueta , un control select, un botón, un mapa con marcadores rojos sobre los parkings de la universidad.



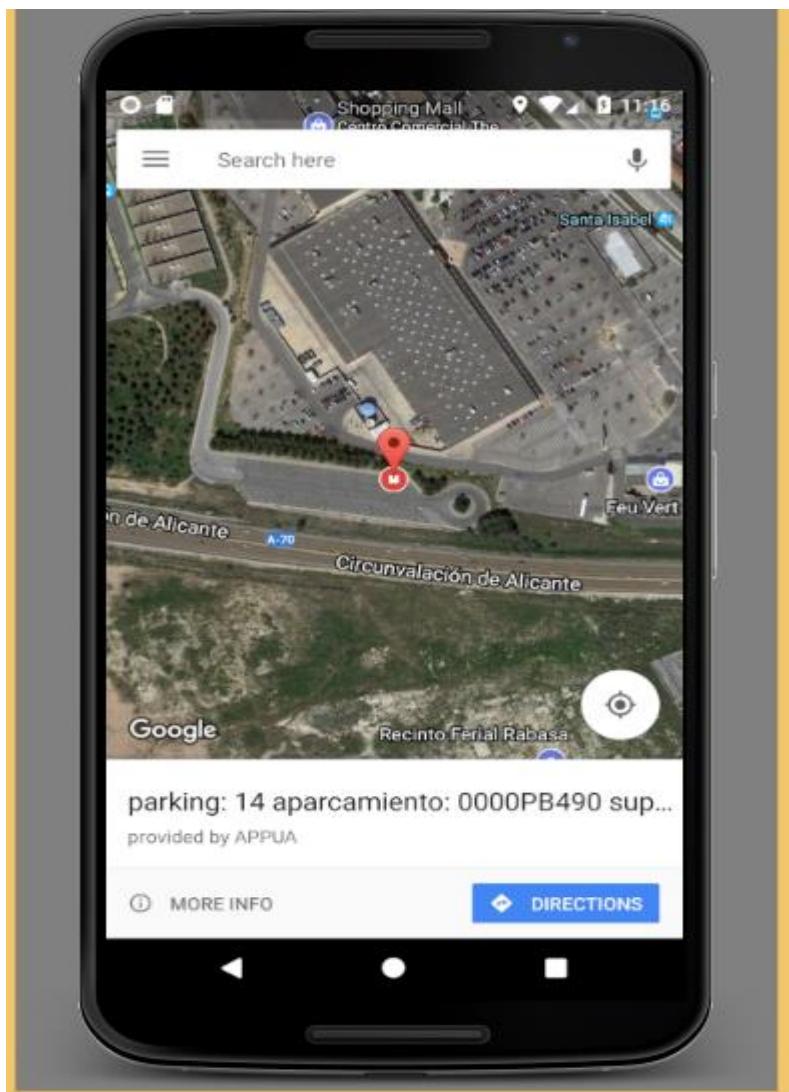
Para tener toda la información sobre un parking y como se puede llegar a tal, se pulsa sobre el icono rojo, como resultado sale un mensaje con la información del parking con puntos sucesivos.

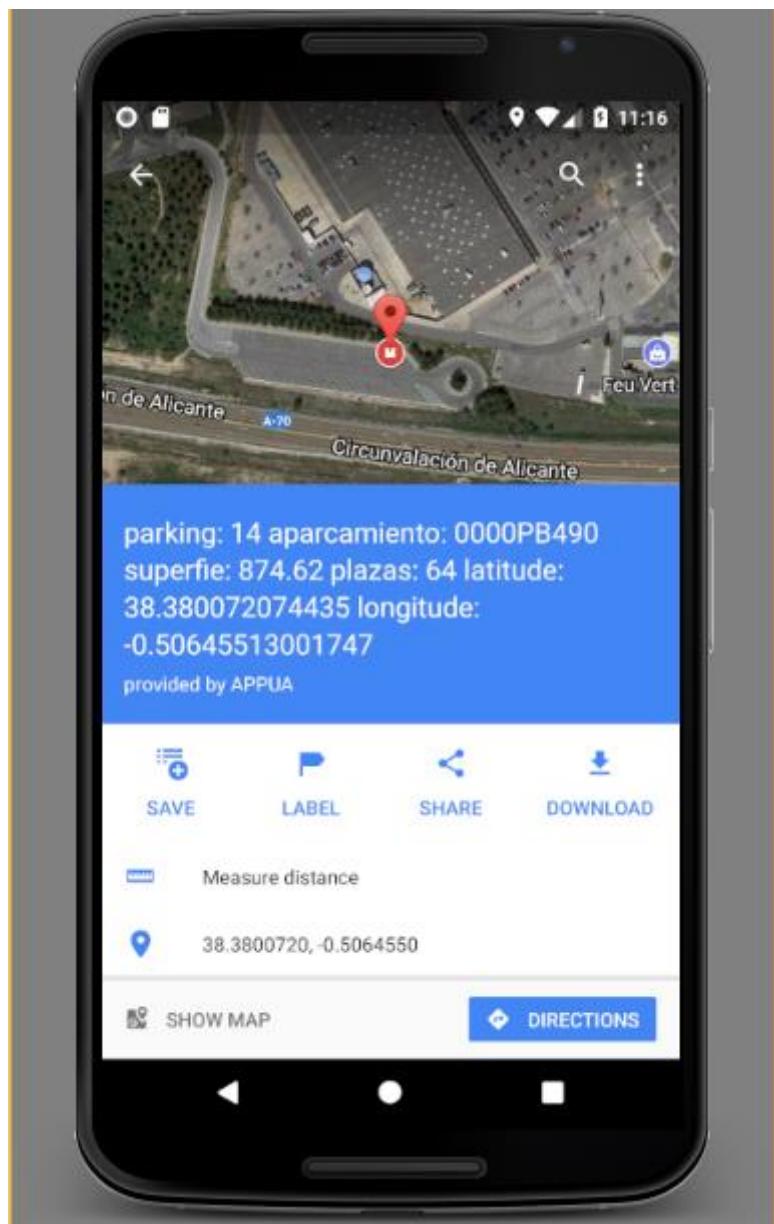


Abajo a la derecha hay dos funcionalidades que los provee la Api de Google maps que dan toda la información sobre el punto, por ejemplo para ir al sitio, al darle al botón flecha, se demuestra la pantalla siguiente.

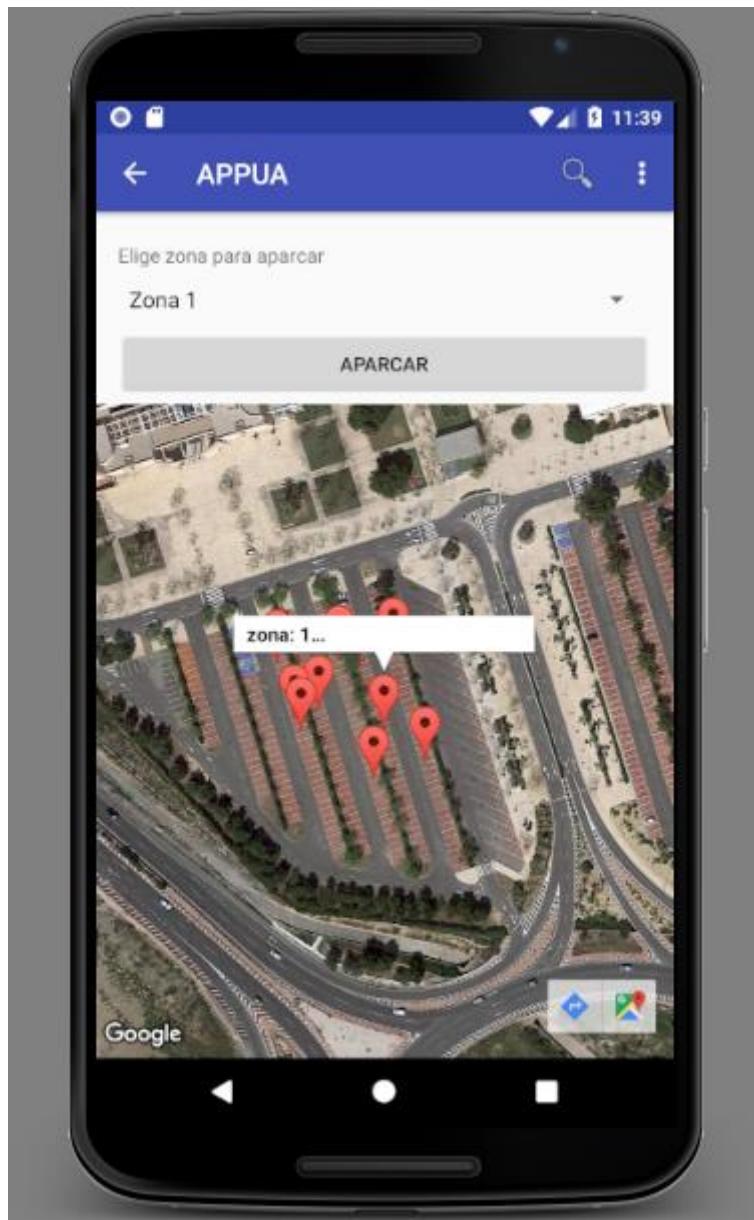


La otra opción, nos da toda la información sobre el punto.

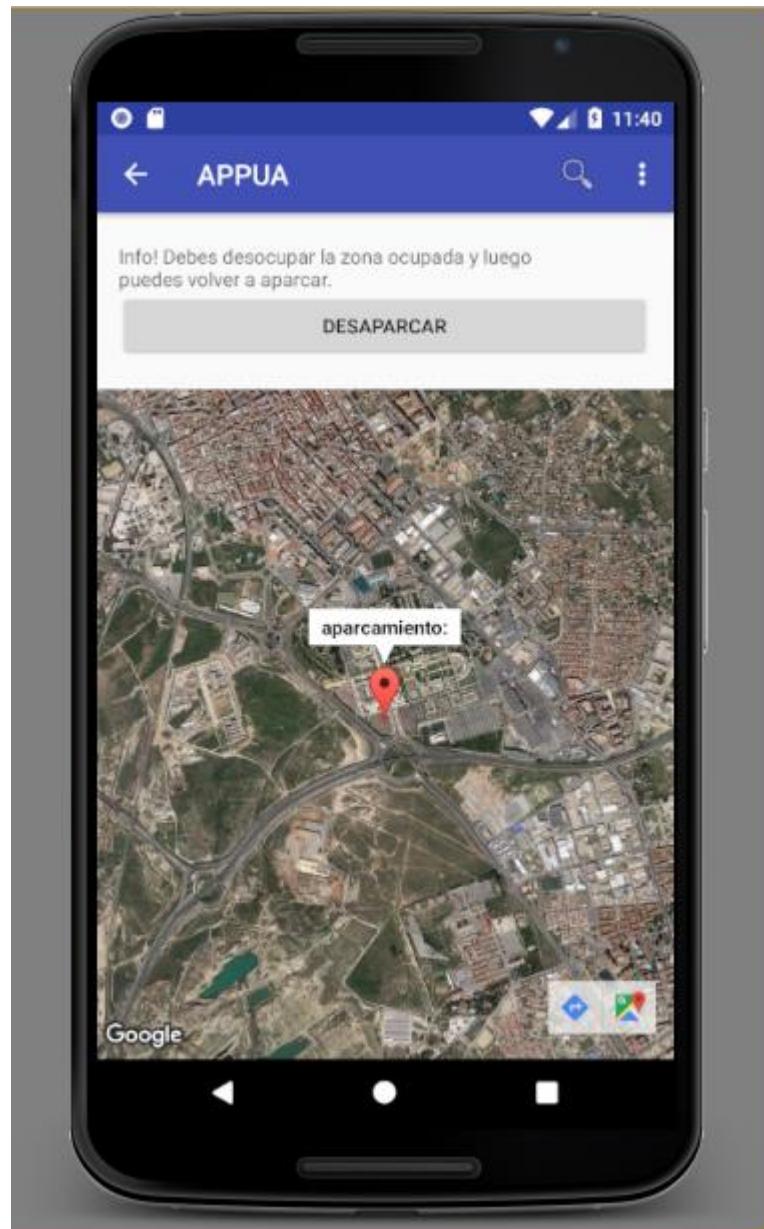




Mediante el control select, el usuario puede elegir que parking quiere consultar. Por ejemplo al elegir el parking 1 y pulsar el botón buscar, se le lleva a otra vista con todos los huecos vacíos del parking como se observa en la captura siguiente.

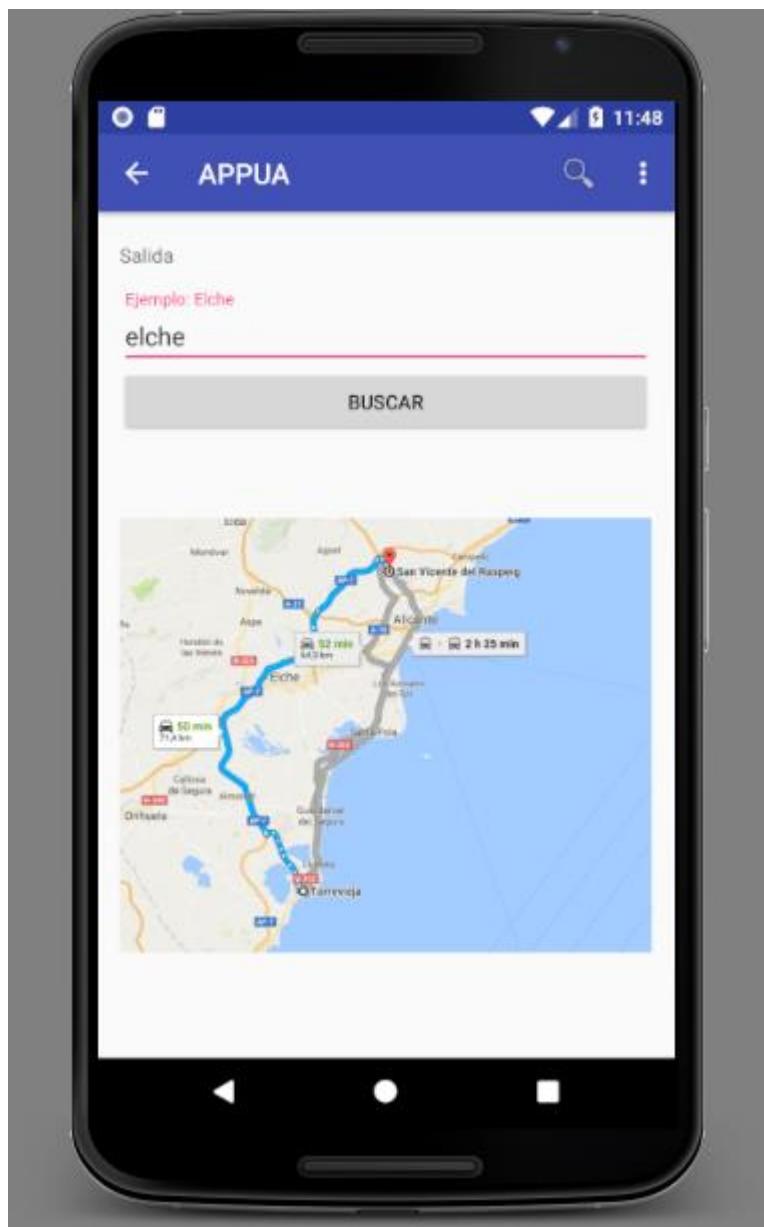


Al aparcar en la zona elegida, se le lleva a otra vista donde le indica que tiene que desaparcar una vez quiera salir, En caso no indique que no ha desaparcado, cuando quiere volver a buscar aparcamiento, se le lleva siempre a esta vista, hasta que desaparque, después puede elegir zona donde aparcar.

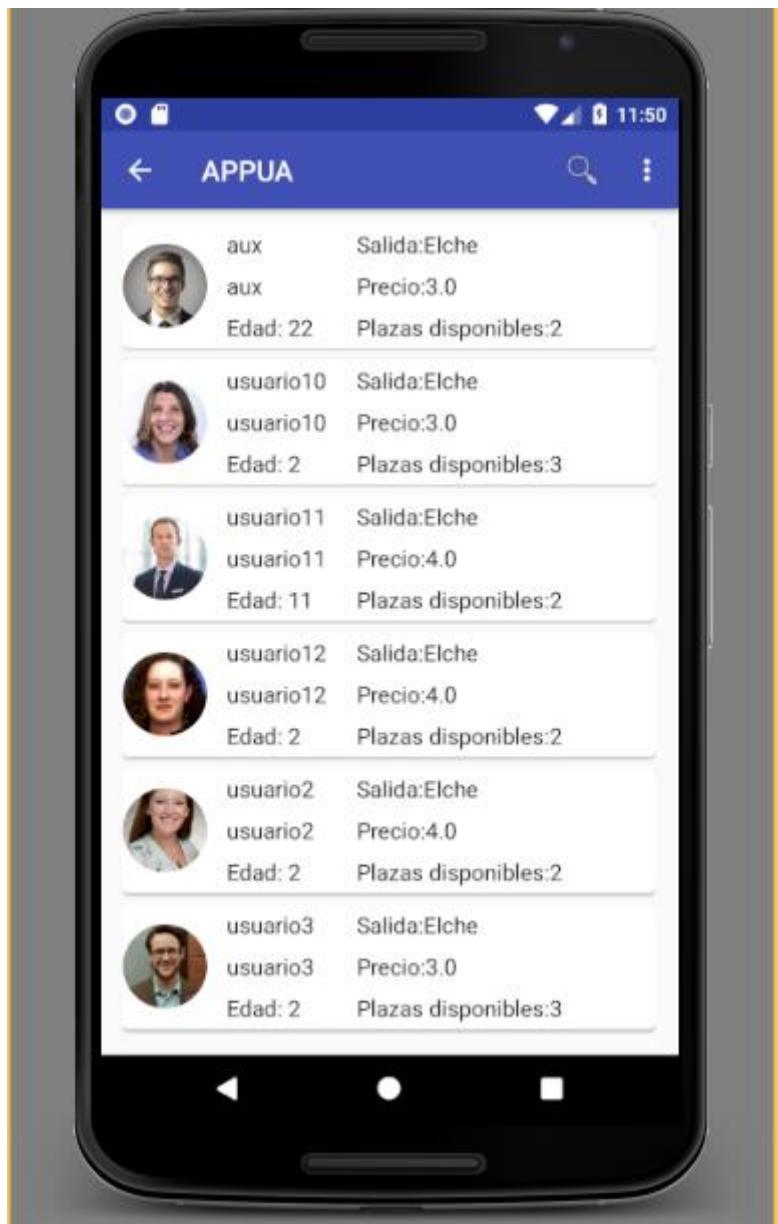


Vista buscar anuncios

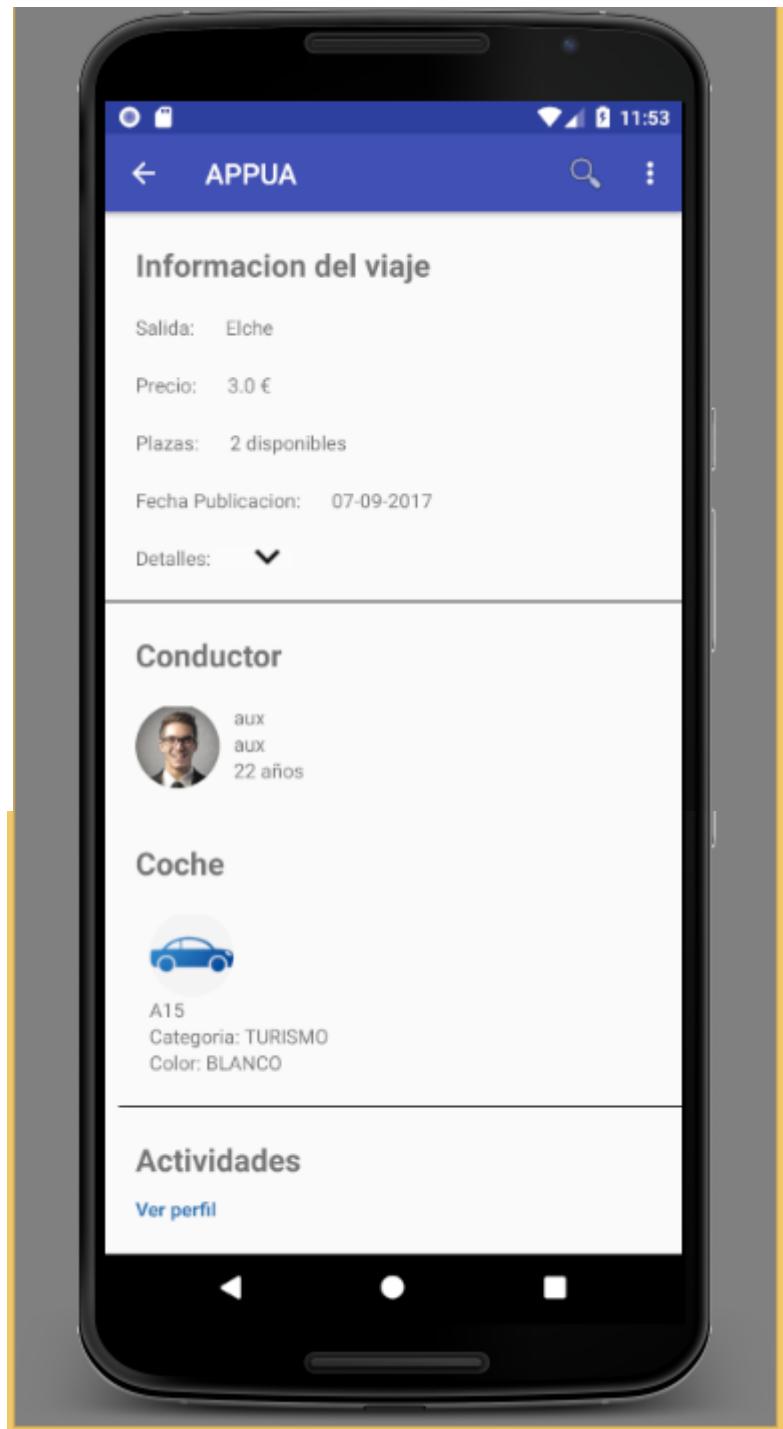
Para buscar anuncios desde un origen en concreto, mediante la loopa buscar en el actionBar, se le lleva a la vista donde puede introducir el origen, y ver todos los anuncios con plazas disponibles.



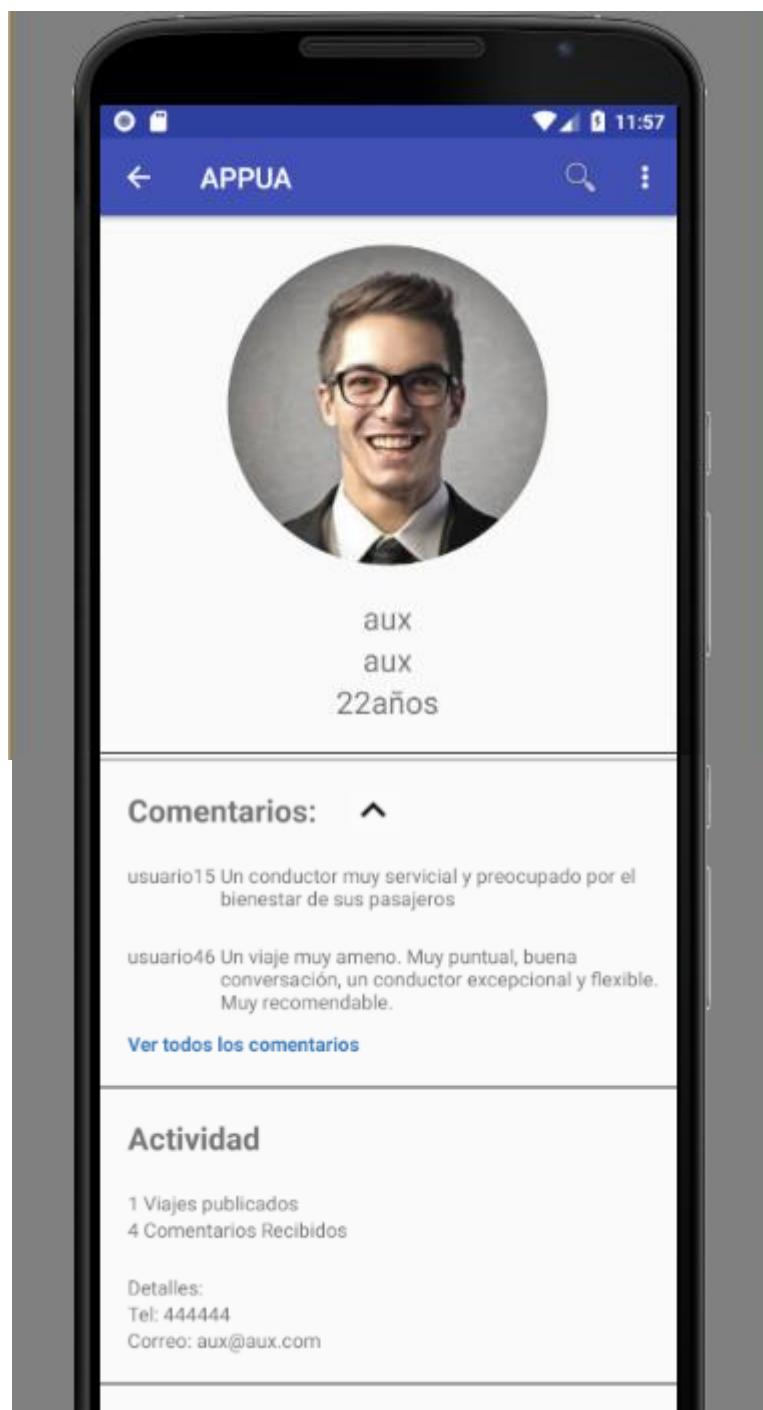
En el caso si el usuario quisiera ver los anuncios a partir de elche, se le lleva a una vista con una lista, cada fila con una foto e información del anuncio.



Una vez se da click sobre un anuncio, se demuestra una vista con toda la información del anuncio, el conductor, el coche, etc.



También puede ver el perfil del anunciante mediante el enlace ver perfil.

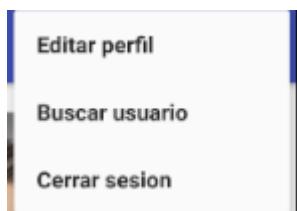


También mediante el enlace Ver todos los comentarios, se puede ver todos los comentarios del anunciante.



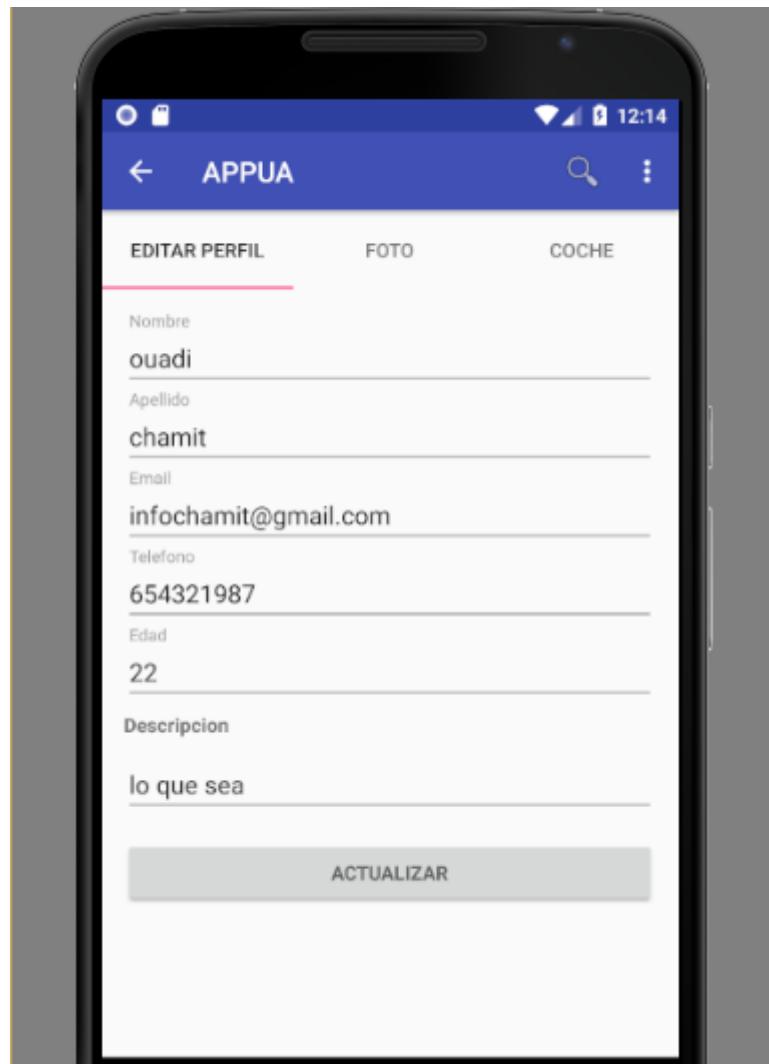
Vista Editar usuario

Mediante el menú , se ofrecen una serie de funcionalidades, por ejemplo para editar usuario.

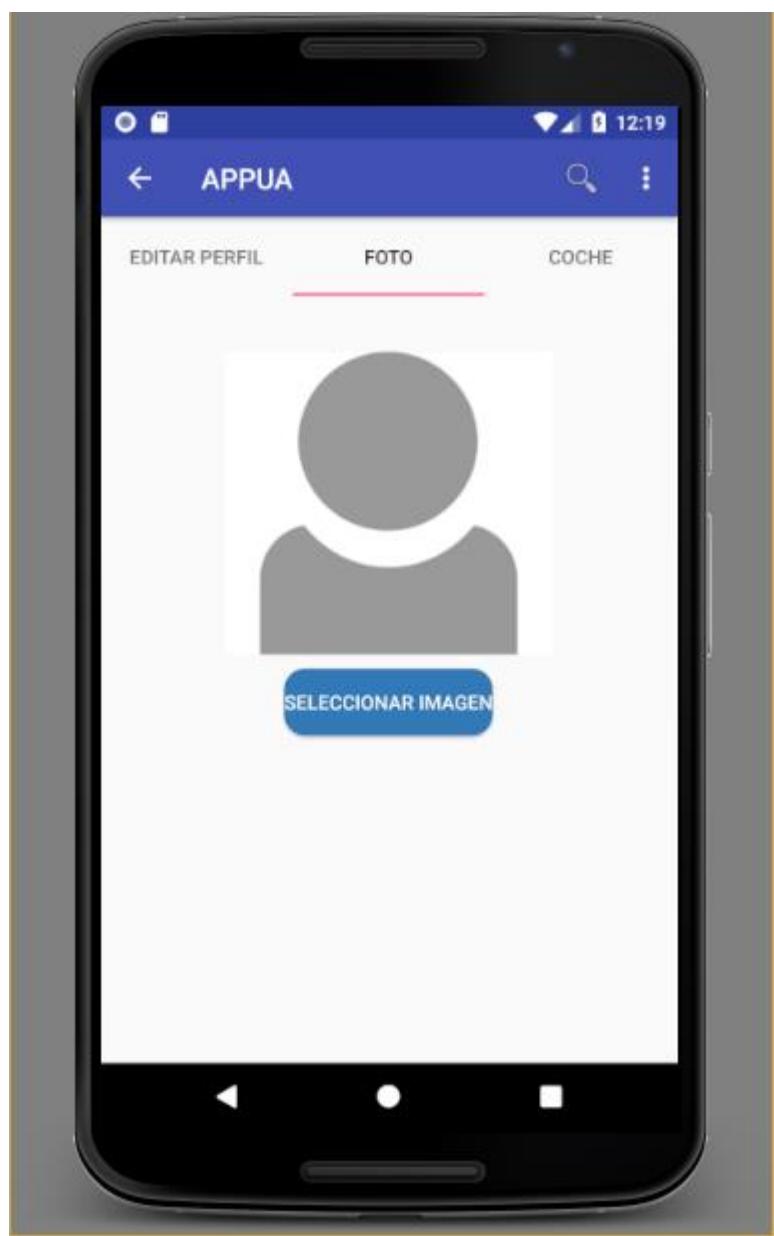


Al elegir la opción editar usuario, se le lleva a una vista que a su vez contiene fragmentos, por ejemplo, editar los datos, insertar o borrar foto del usuario, insertar o borrar coche, insertar o cambiar foto del coche.

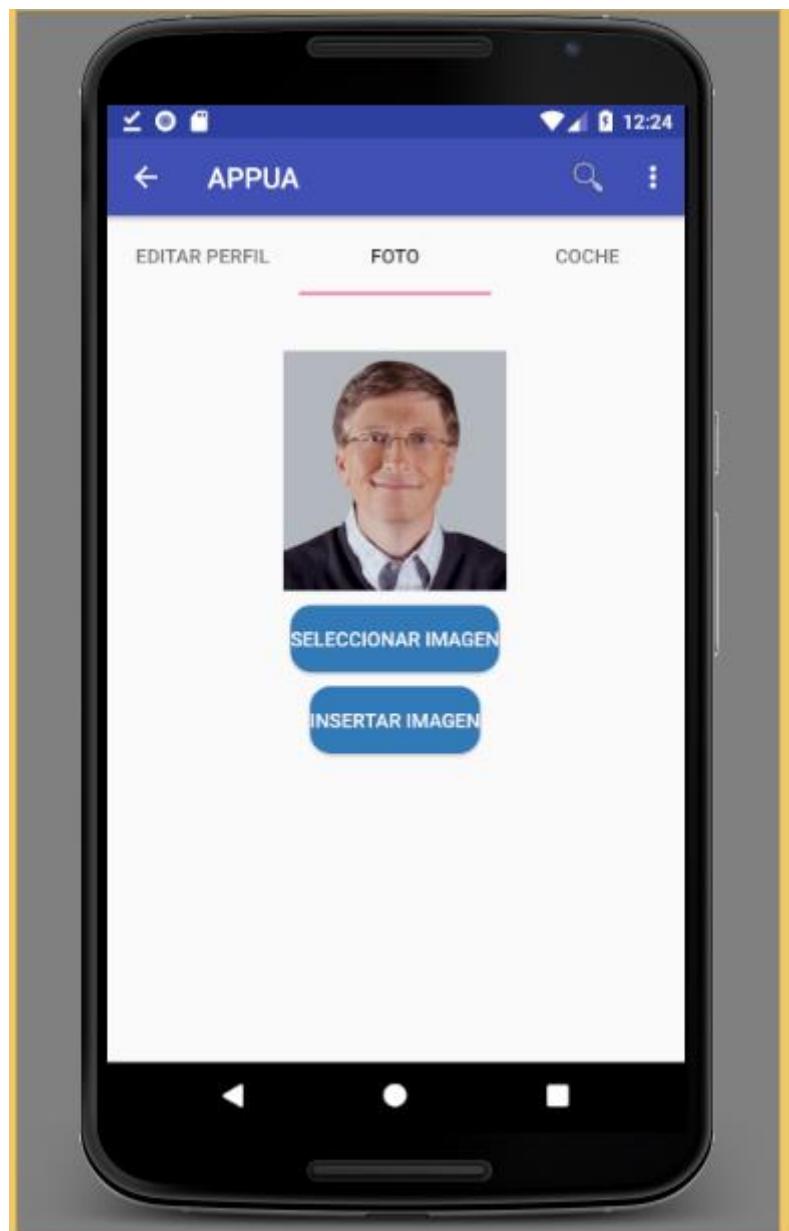
En el primer fragmento precargará los datos que tenga el usuario almacenados en base de datos y permitirá modificar dichos campos. Al igual que la pantalla de registro de usuarios, cuando intentamos ejecutar la acción pulsando el botón se comprobará que los campos son correctos. El resultado final de la pantalla es el siguiente:

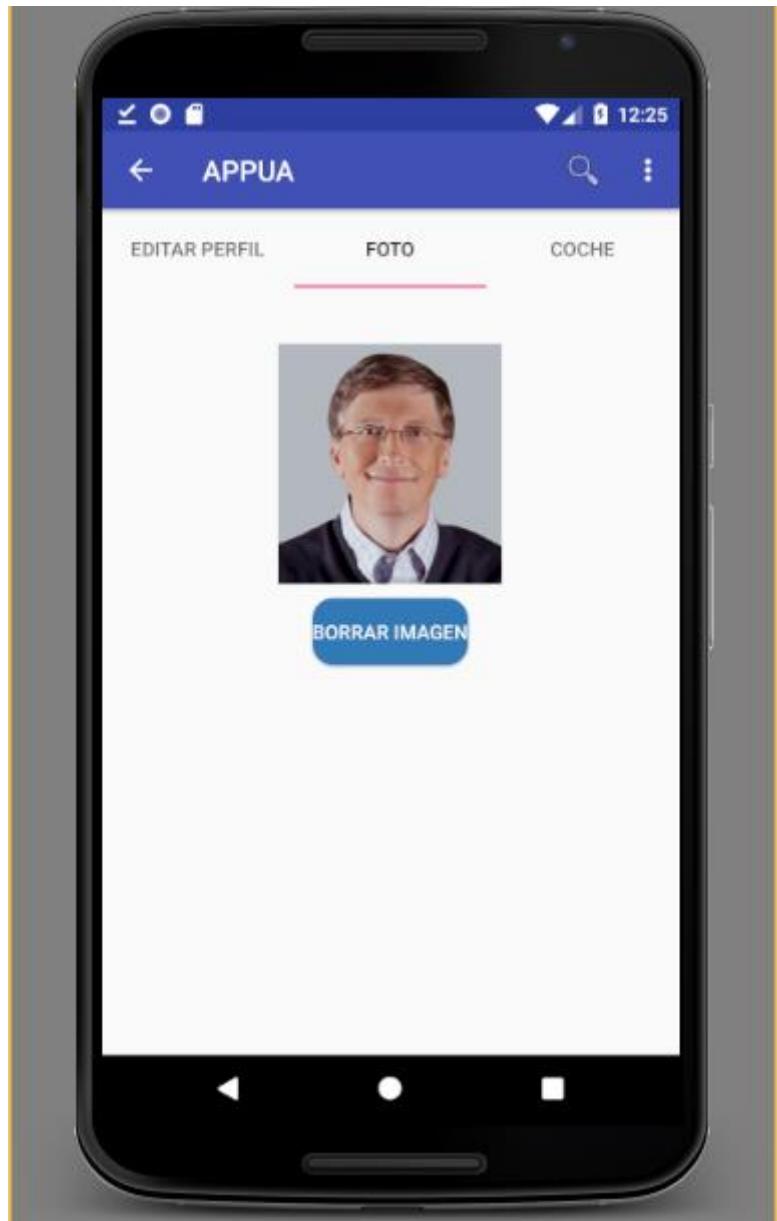


En el fragmento FOTO, se puede insertar en caso que el usuario no tenga foto, o cambiar en caso concreto. Por ejemplo en la pantalla siguiente, el usuario no tiene foto.

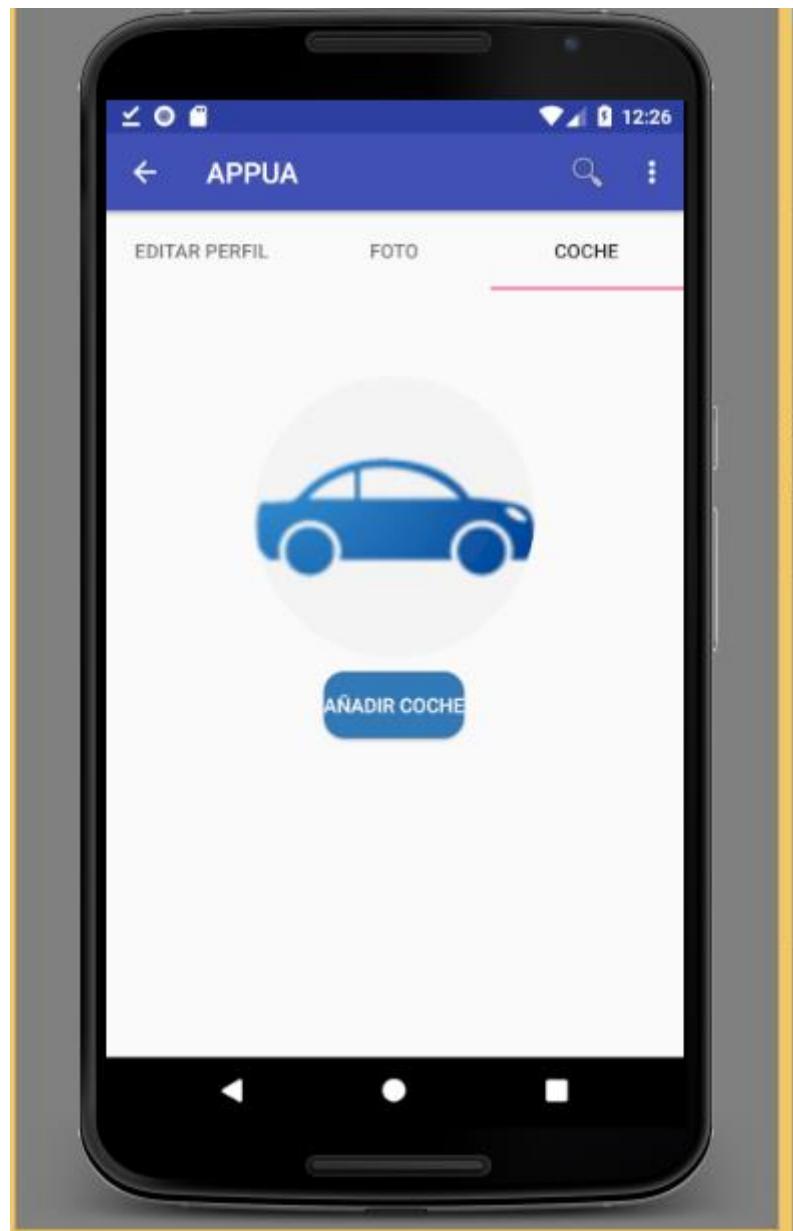


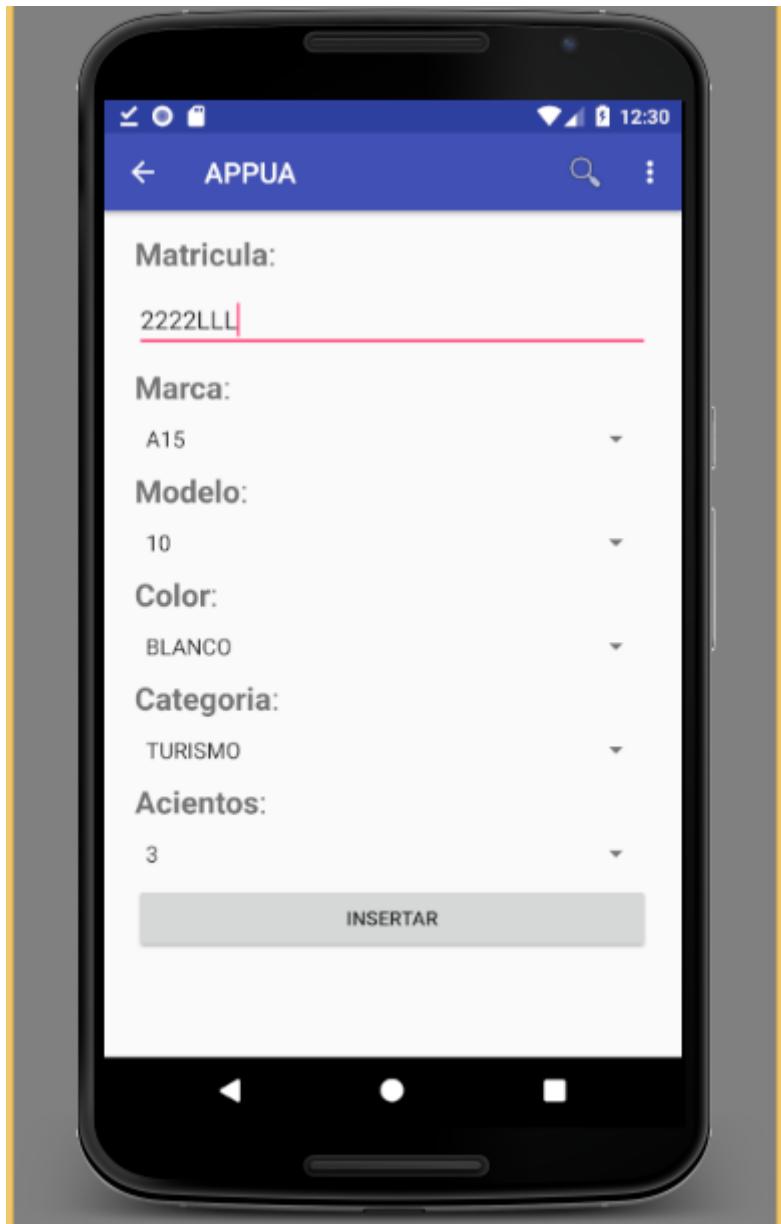
Pantalla insertar imagen del usuario



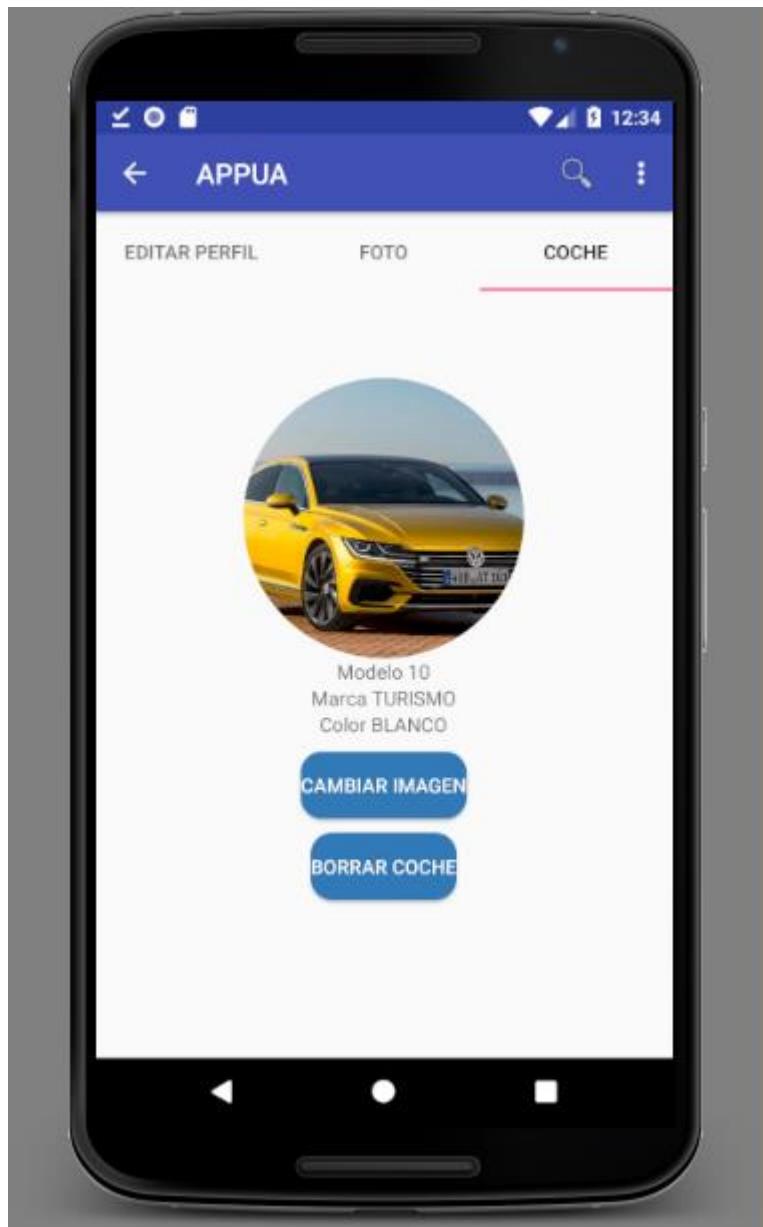


Fragmento insertar, cambiar, eliminar coche. Como el usuario no tiene ningún coche dado de alta, le sale la opción añadir coche, cuando intente añadir un coche, se le lleva a una vista, con un formulario, para introducir todos los datos del coche.





Una vez el usuario introduzca el coche, se le lleva al fragmento donde le demuestra tanto la foto como información del coche, además dos botones, una para insertar imagen en caso que no la tenga, y otro botón para dar de baja al coche.



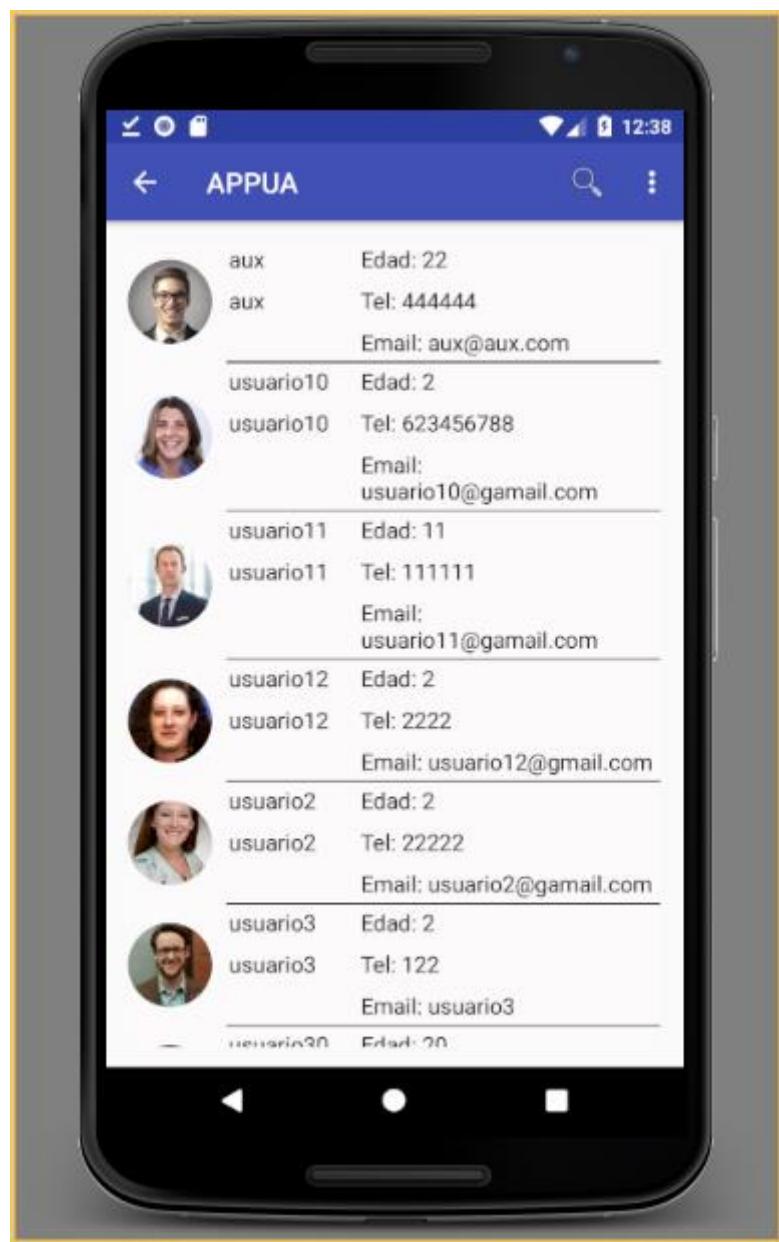
Vista buscar usuario

Se puede buscar un usuario en el sistema con el nombre, origen o teléfono. Como se observa, la vista contiene un texto descriptivo un campo y un botón por cada filtro.



Cuando se busca un usuario por origen, se le lleva a una vista con una lista de los usuarios que salen del origen con una foto y la información del usuario, por ejemplo el nombre. Apellido, etc.

Cuando se hace un click sobre un elemento de la lista, se demuestra el perfil del usuario, buscado, Como se hace en la búsqueda de los anuncios.

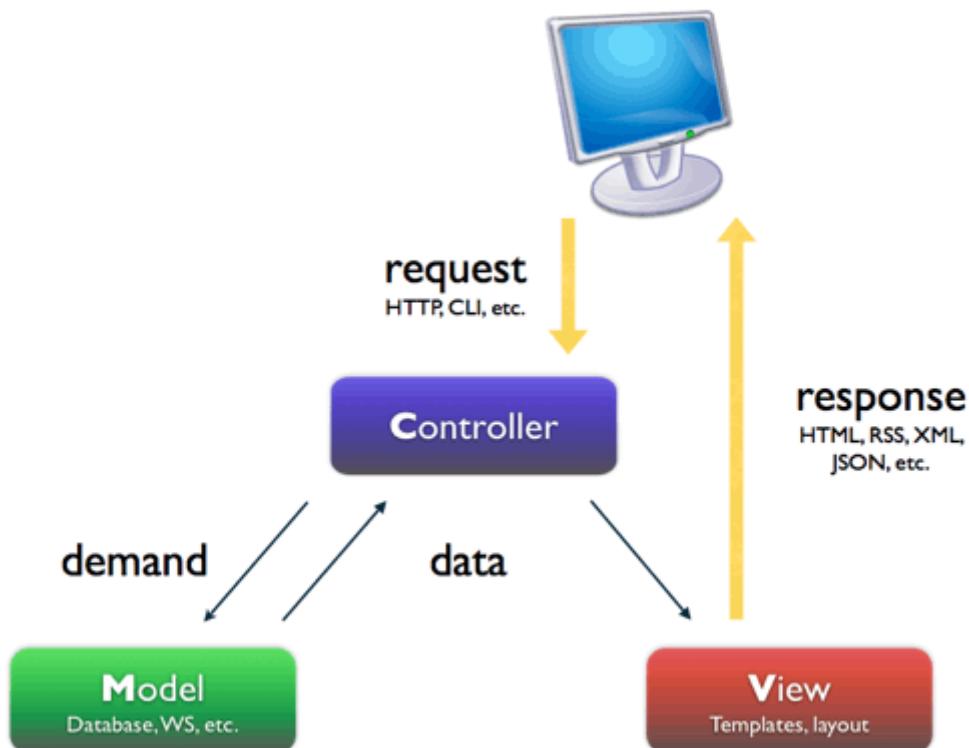


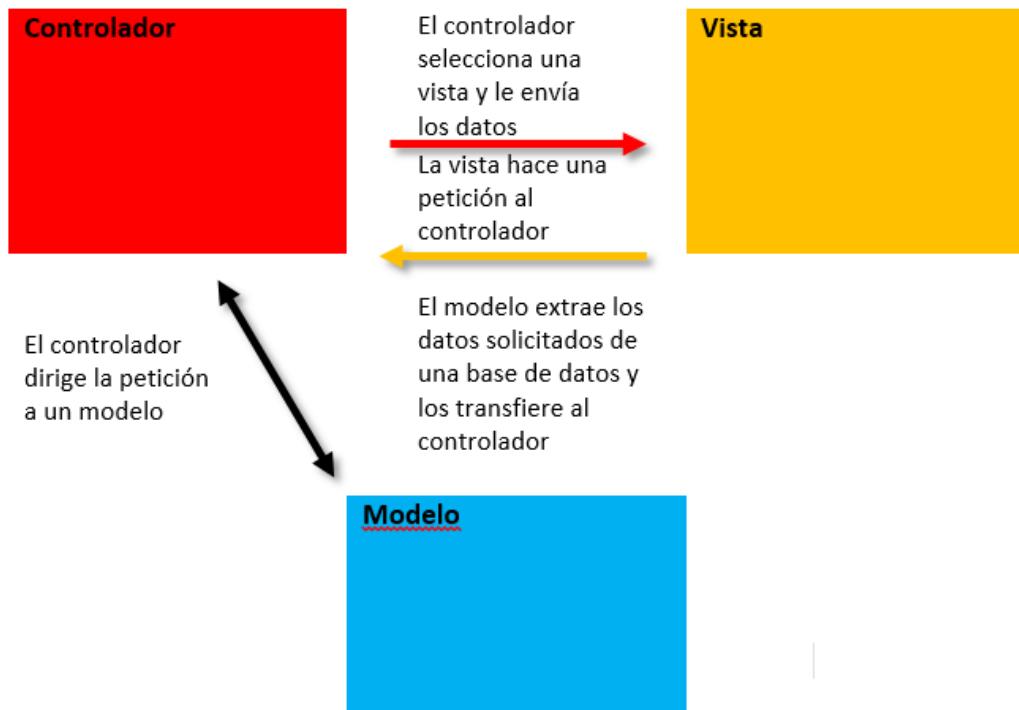
ARQUETECTURA

El modelo utilizado en el desarrollo tanto en la parte web como Móvil, es el Modelo vista controlador, un patrón de desarrollo que separa una aplicación en 3 capas:

Web

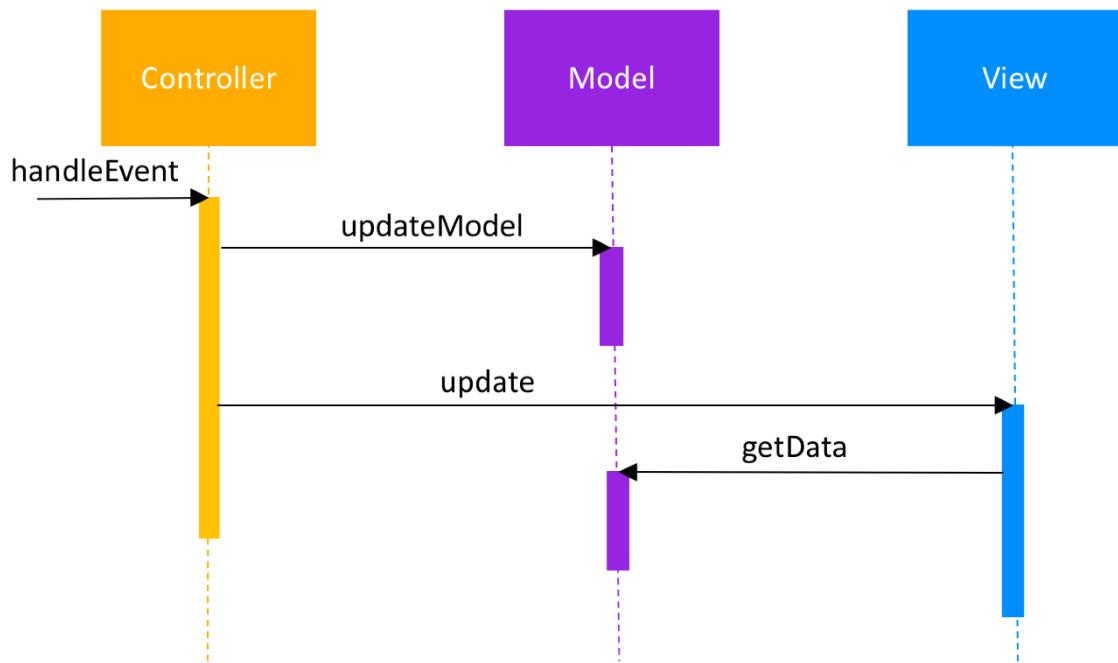
- **Modelo:** es la capa de acceso a datos, en ella se realizan las conexiones y consultas a la base de datos, y los datos que esta obtiene son recibidos por la capa Controlador.
- **Controlador:** aquí encontramos la lógica del programa, esta capa manipula la información que recupera de la base de datos, y la envía a la Vista. También recibe las acciones del usuario capturadas por la Vista y responde a ellas.
- **Vista:** en esta parte se define la interfaz de usuario, especifica cómo se mostrará la información obtenida del Controlador y capture los eventos del usuario.





Móvil

En Android las vistas son de tipo XML y cada una tiene una clase java asociada que la gestione, por ejemplo capturar los eventos disparados cuando el usuario interactue con la pantalla. Sin embargo Todas la actividades comparten un controlador, para poder realizar operaciones complejas o acceso a la red.



IMPLEMENTACION

Front end

Web

Html/css

En cuanto a la vista, en el Frontend, he utilizado el lenguaje de marcado HTML, es el elemento de construcción más básico de una página web y se usa para crear y representar visualmente una página web. Determina el contenido de la página web, pero no su funcionalidad. Otras tecnologías distintas de HTML son usadas generalmente para describir la apariencia/presentación de una página web (CSS) o su funcionalidad (JavaScript/jquery).

HTML usa "markup" o marcado para anotar textos, imágenes, y otros contenidos que se muestran en el Navegador Web. El lenguaje de marcado HTML incluye "elementos" especiales tales como <head>, <title>, <body>, <header>, <article>, <section>, <p>, <div>, , , y muchos otros más.

Para dar estilo a las páginas webs, he utilizado CSS Hojas de Estilo en Cascada (Cascading Style Sheets) y Bootstrap.

CSS es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne.

`h1 {color: red;}`

h1 es el selector

{color: red;} es la declaración

El selector funciona como enlace entre el documento y el estilo, especificando los elementos que se van a ver afectados por esa declaración. La declaración es la parte de la regla que establece cuál será el efecto. En el ejemplo anterior, el selector **h1** indica que todos los elementos h1 se verán afectados por la declaración donde se establece que la propiedad **color** va a tener el valor **red** (rojo) para todos los elementos **h1** del documento o documentos que estén vinculados a esa hoja de estilos.

Las tres formas que he utilizado para dar estilo a un documento son las siguientes:

Utilizando una hoja de estilo externa que estará vinculada a un documento a través del elemento `<link>`, el cual debe ir situado en la sección `<head>`.

```
<!DOCTYPE html>
<html lang="en">
<html>
  <head>
    <title>Título</title>
    <link href="php echo base_url();?&gt;assets/css/perfil.css"
rel="stylesheet"&gt;
  &lt;/head&gt;
  &lt;body&gt;
    .
    .
    .
    .
  &lt;/body&gt;
&lt;/html&gt;</pre
```

Utilizando el elemento `<style>`, en el interior del documento al que se le quiere dar estilo, y que generalmente se situaría en la sección `<head>`. De esta forma los estilos serán reconocidos antes de que la página se cargue por completo.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>hoja de estilo interna</title>
  <style>

    .navbar {
      margin-bottom: 0;
      border-radius: 0;
    }

    .sidenav {
      padding-top: 20px;
      background-color: #f1f1f1;
      height: 100%;
    }

    /* On small screens, set height to 'auto' for sidenav and grid */
    @media screen and (max-width: 767px) {
      .sidenav {
        height: auto;
        padding: 15px;
      }
    }

    ul {
      list-style-type: none;
    }

    .paginacion{
      background-color: #F5F5F5;
      padding: 20px 0 0 0;
      margin: auto;
      width: 500px;
      height:80px;
```

```
    text-align: center;
    margin-top: 40px;

}

.paginacion a{
    text-decoration: none;
    padding: 0;
    background-color: #4CAF50;
    color: white;
    border-radius: 5px;
    margin: 0;
    width: 80%;

}
.paginacion a:hover{
    background-color: #333333;
    color: #C0C0C0;

}
.pagination a:hover:not(.active) {
    background-color: #ddd;
    border-radius: 5px;
}
.pagination a.active {
    background-color: #4CAF50;
    color: white;
    border-radius: 5px;
}
.actual{
    color: #FFFFFF;
    padding: 4px;
}
.flex-container {
    display: -webkit-flex;
    display: flex;
    width: 100%;
    height: 100%;
    /* background-color: lightgrey; */
}

</style>
</head>
<body>
    <h1>Aquí se aplicará el estilo de letra para el Título</h1>
</body>
</html>
```

Y por ultimo Incluir CSS en los elementos HTML.

```
<div class="col-md-4" style="border-right-style:groove;height:80%">
```

En cuanto al Bootstrap, para utilizarlo, se puede descargar de dos maneras, compilado o mediante el código fuente original. En mi caso he utilizado la versión compilada, la cual tiene la siguiente estructura de archivos y directorios:

```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.min.css
│   ├── bootstrap-theme.css
│   └── bootstrap-theme.min.css
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings-regular.eot
    ├── glyphicons-halflings-regular.svg
    ├── glyphicons-halflings-regular.ttf
    └── glyphicons-halflings-regular.woff
```

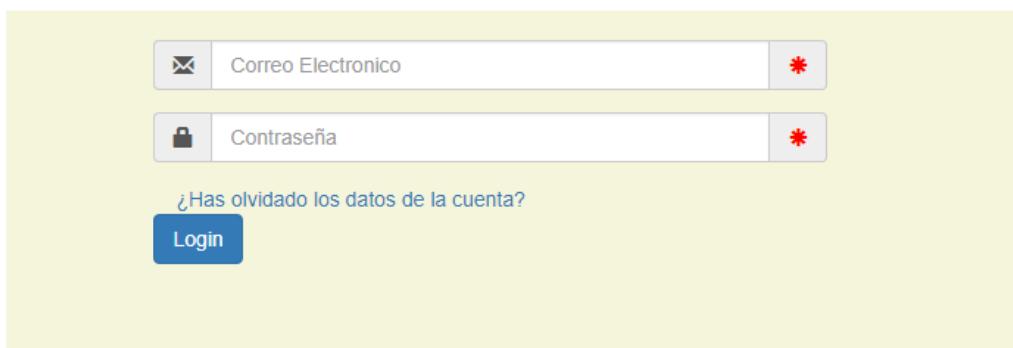
Después de descargarlo y ponerlo en una ruta especificada, para poder tener el acceso, y utilizar esas librerías, se debe incluir la ruta en el elemento <head> en el doc html como el ejemplo siguiente:

```
<link href="=base_url()?assets/include/css/bootstrap.min.css" rel="stylesheet">
```

Con este mecanismo, se puede acceder y utilizar la librería, como en este ejemplo:

```
<form action="php echo site_url('loginUser') ?" name="myForm"
      class="form-horizontal" method="post">
    <div class="form-group">
      <div class="input-group">
        <span class="input-group-addon"><i class="glyphicon glyphicon-envelope"></i></span>
        <input type="text" name="correo" class="form-control"
               id="correoElectronico" placeholder="Correo Electronico" required><span
               class="input-group-addon"><i style="color:red;" class="glyphicon glyphicon-asterisk"></i></span>
      </div>
    </div>
    <div class="form-group">
      <div class="input-group">
        <span class="input-group-addon"><i class="glyphicon glyphicon-lock"></i></span>
        <input class="form-control" name="contraseña"
               type="password" id="password1" placeholder="Contraseña" required><span
               class="input-group-addon"><i style="color:red;" class="glyphicon glyphicon-asterisk"></i></span>
      </div>
    </div>
    <div> <a href="php echo site_url("public/recoverPass") ?&gt;"¿Has
olvidado los datos de la cuenta?</a></div>
    <div class="form-group">
      <input type="submit" value="Login" class="btn btn-primary">
    </div>
</form>
```

Como se observa todas las clases son de la librería Bootstrap, con la cual se le da un estilo a los elementos de html, en este caso a los elementos input tienen dos elementos laterales, uno que demuestra el tipo del elemento de forma grafica y el otro como es un campo obligatorio, también el botón de login, mediante la clase btn btn-primary se le da buen estilo, con color azul y esquinas redondas.



Javascript/Jquery

Para dar dinamismo a las páginas web y incorporar efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario, he utilizado la librería JQuery, es una biblioteca multiplataforma de javascript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Se puede incluir de dos formas, una interna y otra externa:

La forma interna, se trata de incluir la librería descargada de la pagina

<http://getbootstrap.com/> oficial, o cualquier fuente, que la contiene, una vez esta descargada la librería, se incluye un link que contiene la ruta hasta su ubicación, este ejemplo refleja su la manera como se incluye en la aplicación web:

```
<script src="<?php echo base_url(); ?>assets/include/js/jquery-  
3.2.1.min.js"></script>
```

La forma externa, se puede utilizar desde Google o Microsoft. En mi caso he utilizado el enlace de Google.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery  
.min.js"></script>  
  
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-  
3.2.1.min.js"></script>
```

En este ejemplo, se controla el campo nombre, mediante el cual se busca un usuario con su nombre, solo esta permitido nombres con letras, si no es asi, se le muestra un mensaje al usuario, y se desactiva el botón submit, hasta que el usuario corrija el error, una vez desaparece el error, el usuario puede realizar la consulta.

```
$("#boBuscarUsuarioN").click(function(event){  
    var valor= $("#usuarioNombre").val();  
    var characters = /^[a-z," "]+$/i;  
    if (!characters.test(valor)) {  
        $("#messageTipoN").html("El nombre solo debe contener letras");  
        return false  
    }else{  
        $("#messageTipo").html("");  
    }  
});
```

Peticiones Ajax/json

En cuanto a la realización de peticiones de consulta de datos, para evitar duplicidad, a la hora de almacenar información en la base de datos y además sin recarga de la página, he utilizado la tecnología Ajax, la cual permite intercambiar información entre el servidor y el cliente (navegadores) sin la necesidad de recargar la página. De esta forma, ganamos en usabilidad, experiencia y productividad del usuario final.

En este ejemplo cuando el usuario pone el email en el campo de registro, una vez sale del campo, se realiza una petición al servidor, para comprobar si existe el dato:

```
$ (function () {
    $("#correoElectronico").blur(function (event)
    {

        event.preventDefault();
        var correo= $("#correoElectronico").val();
        var url= "<?php echo base_url(); ?>private/correo/" +correo;

        $.ajax(
        {
            type:"GET",
            url: url,
            success:function (response)
            {

                if(response=='false') {
                    $('#messageCorreo').html("Ya hay un usuario
con esta cuenta, introduzca otra cuenta");
                    $('form
input[id='boUsuario']").prop("disabled", true);
                }else{
                    $('#messageCorreo').html("");
                    $('form
input[id='boUsuario"]').prop("disabled", false);
                }

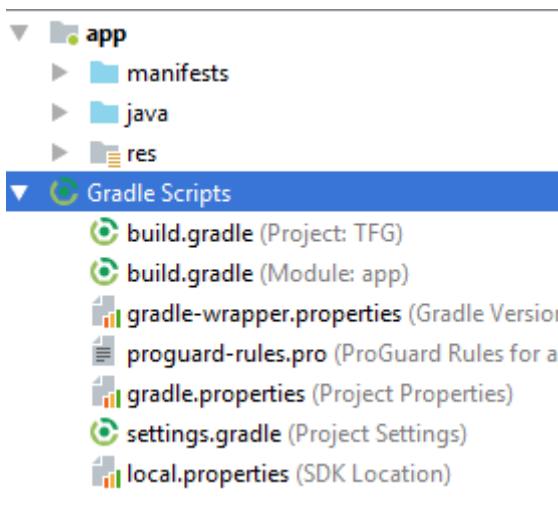
            },
            error: function(error)
            {
                $('#message').html(error);
            }
        }
    );
});
```

El servicio que responde la consulta anterior en el lado del servidor, devuelve true en caso de éxito o false en caso de que el correo existe en la base de datos:

```
public function correo($correo) {
    $resultado=$this->RegistroModel->consultarCorreo($correo);
    if (!$resultado)
    {
        echo "false";
    }
    else
    {
        echo "true";
    }
}
```

Móvil

Al crear un proyecto Android, se crean todos los elementos necesarios por defecto, como se observa en la captura.



Directorio raíz del proyecto :

Su elemento más importante es la carpeta app/ , además de app/ contiene ficheros importantes tales como :

- build.gradle con instrucciones de construcción del proyecto
- Varios ficheros relacionados con Gradle (settings.gradle, gradle.properties, etc.)
- local.properties indica en qué directorio se encuentra el SDK de Android
- Fichero app.iml con metadatos de proyecto que usará Android Studio En la carpeta app, se encuentra:
 - La carpeta manifests, contiene el descriptor de la APP, es un fichero xml, proporciona información esencial sobre la aplicación al sistema Android, información que el sistema debe tener para poder ejecutar el código de la app.
 - Código fuente en Java, por defecto al crear un proyecto Android, se crea una actividad, cuya función es crear y manejar la interfaz de usuario así como la interacción con el usuario, Las activities están conformadas por dos partes: la parte lógica y la parte gráfica.
 - La parte lógica es un archivo java que es la clase que se crea para poder manipular, interactuar y colocar el código de esa actividad.
 - La parte gráfica es un XML que tiene todos los elementos que estamos viendo de una pantalla declarados con etiquetas

Se pueden crear todas las actividades que sean necesarias para el proyecto, así como las clases que tienen los datos, la lógica de negocio etc.

- Ficheros con recursos, archivos estáticos que serán empaquetados junto con la propia aplicación, por defecto se crean dentro la carpeta res.
 - res/drawable/ para imágenes (PNG, JPEG, ...)
 - res/layout/ para definición de interfaces de usuario en archivos XML
 - res/menu/ para la definición de menús de aplicación en XML
 - res/raw/ para recursos en general, de cualquier tipo

- res/values/ para definiciones de cadenas, traducciones, dimensiones, matrices de constantes, etc.
- res/xml/ para recursos de carácter general siempre que estén definidos en XML

Estructura de una actividad:

Como he mencionado anteriormente, respecto a una actividad, voy a describir un caso por ejemplo, al crear la actividad login, se crea la clase java y el fichero xml que tiene la interfaz de usuario.

La parte Grafica:

Se pueden incluir todos los controles que sean necesarios, botones, campos de textos, controles de select, control scrollBar para poder ver todo el contenido de la vista en caso que haya mucho contenido.

```
<LinearLayout
    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:gravity="center_horizontal"
    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="com.example.walid.tfg.LoginActivity">

    <!-- Login progress -->
    <ProgressBar
        android:id="@+id/login_progress"
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:visibility="gone" />

    <ScrollView
        android:id="@+id/login_form"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

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

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <AutoCompleteTextView
                    android:id="@+id/email"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="@string/prompt_email"
                    android:inputType="textEmailAddress"
                    android:maxLines="1"
                    android:singleLine="true" />

            </android.support.design.widget.TextInputLayout>
        
```

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/prompt_password"
        android:imeActionId="6"

        android:imeActionLabel="@string/action_sign_in_short"
        android:imeOptions="actionUnspecified"
        android:inputType="textPassword"
        android:maxLines="1"
        android:singleLine="true" />

</android.support.design.widget.TextInputLayout>

<Button
    android:id="@+id/email_sign_in_button"
    style="?android:textAppearanceSmall"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="@string/action_sign_in"
    android:textStyle="bold" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="20dp"
    android:text="¿No tienes cuenta?"
    android:id="@+id/textView9"
    android:layout_gravity="center_horizontal" />

<Button
    style="?android:textAppearanceSmall"
    android:id="@+id/button2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="@string/action_register_in"
    android:textStyle="bold"
    android:layout_gravity="center_horizontal" />

</LinearLayout>
</ScrollView>
</LinearLayout>
```

La parte Logica:

```
public class LoginActivity extends AppCompatActivity implements
LoaderCallbacks<Cursor> {

    AsyncTask<Void, Void, Boolean> loadingTask;
    private SharedPreferences sharedpreferences;
    // UI references.
    private AutoCompleteTextView mEmailView;
    private EditText mPasswordView;
    private View mProgressView;
    private View mLoginFormView;
    SessionManager session;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        // Session Manager
        session = new SessionManager(getApplicationContext());
        // Set up the login form.
        mEmailView = (AutoCompleteTextView) findViewById(R.id.email);
        populateAutoComplete();

        mPasswordView = (EditText) findViewById(R.id.password);
        mPasswordView.setOnEditorActionListener(new
        TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView textView, int id,
KeyEvent keyEvent) {
                if (id == EditorInfo.IME_ACTION_DONE || id ==
EditorInfo.IME_NULL) {
                    // attemptLogin();
                    return true;
                }
                return false;
            }
        });
        Button mEmailSignInButton = (Button)
findViewById(R.id.email_sign_in_button);
        mEmailSignInButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                attemptLogin();
            }
        });
        sharedpreferences = getSharedPreferences(MyPREFERENCES,
Context.MODE_PRIVATE);
        Button registrarse = (Button) findViewById(R.id.button2);
        registrarse.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent myIntent = new Intent(LoginActivity.this,
RegistryActivity.class);
                //myIntent.putExtra("key", value); //Optional parameters
                LoginActivity.this.startActivity(myIntent);
            }
        });
        mLoginFormView = findViewById(R.id.login_form);
        mProgressView = findViewById(R.id.login_progress);
```

```
}

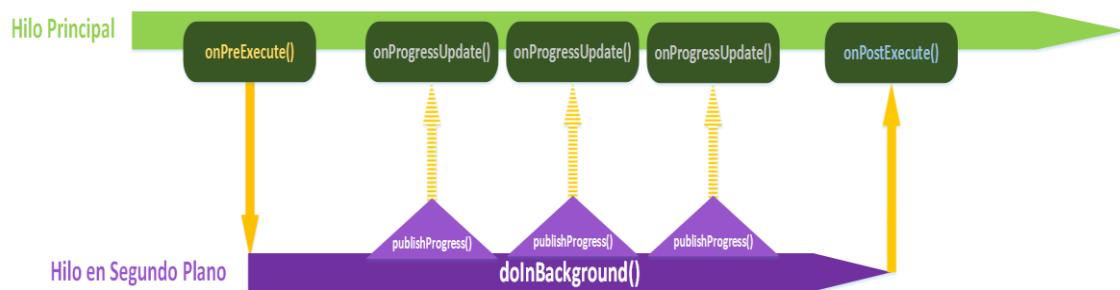
private void populateAutoComplete() {
    if (!mayRequestContacts())
        return;
}

}
```

En esta sección de código lo que estamos haciendo es crear una clase que se llama "LoginActivity" y la estamos extendiendo de AppCompatActivity, estamos diciendo que "LoginActivity" es una clase que hereda todo de la clase AppCompatActivity que ya tiene Android definida. Todas las activities deben llevar por lo menos un método, el método "oncreate", que es en donde se crea la actividad. Del método "onCreate" lo más importante es la línea de código: SetContentView(R.Layout.acivity_main), Que es la que hace el trabajo de enlazar la parte lógica con la parte gráfica. El archivo XML que va a mostrarse cuando se mande a llamar la clase ">LoginActivity" es el archivo XML llamado "activity_login".

Para realizar una petición al api desde la aplicación Android, cuando intentamos realizar un cálculo complejo o acceder a la red, en Android el hilo principal (el hilo del interfaz de usuario) se bloquea y la aplicación se cierra.

Para solventar este problema, debemos crear una clase dentro del activity que extienda de AsyncTask para poder realizar la petición al api. AsyncTask lo que nos permite es crear un hilo de ejecución nuevo, este hilo secundario comparte las variables con el hilo principal y nos ofrece la posibilidad de ejecutar esos cálculos complejos o los accesos a la red sin que la aplicación se cierre. Es decir, para la ejecución de tareas asíncronas utilizaremos AsyncTask.



Dicha clase tendrá 5 métodos que se sobrescriben para realizar las peticiones a nuestro api que son:

- **onPreExecute:** en este método realizamos los trabajos previos a la ejecución, por ejemplo demostrar un load, como forma de cargar los datos desde el server.
- **doInBackground:** conectar con el server, y obtener el objeto json que nos devuelve la petición. Este es el único método que se ejecuta en el hilo secundario, por tanto es donde deberemos hacer el acceso a la red para que la aplicación no falle.
- **onProgressUpdate:** el cual va recibiendo un valor desde el método doInBackground para reflejar el progreso de la carga desde el server median un progressBar, asi el usuario pueda ver como va el progreso de la tarea.
- **onPostExecute:** en este método lo usaremos para mostrar en la interfaz de usuario el resultado de respuesta de la petición. Por ejemplo un mensaje indicando que ha ido bien o mal.
- **onCancelled:** este método, se ejecuta cuando ha habido un error en el servidor sin pasar por el método onPostExecute, por ejemplo, devuelve un estado de error 500, como se ha habido un error en el servidor, también se usa para notificar al usuario, del error ocurrido.

En esta clase como se observa, se puede crear una instancia, con los atributos email y password, que son los valores que queramos comprobar si son correctos o no. Si las credenciales proveidas por parte del usuario son correctas, se hace otra consulta para devolver el objeto usuario y utilizarlo en otras actividades si es necesario.

```
public class LoadingTask extends AsyncTask<Void, Void, Boolean> {
    private Apua apua;
    public String email;
    public String password;
    Usuario usuario = null;
    public boolean exito = false;

    public LoadingTask(Apua apua, String email, String password) {
        this.apua=apua;
        this.email = email;
        this.password = password;
    }

    @Override
    protected Boolean doInBackground(Void... voids) {
        boolean success = false;

        try {

            success = apua.serverAgent.loginUsuario(email, password);

            if (success) {
                usuario = apua.serverAgent.getUser(email);
            }
        } catch (Exception e) {
            cancel(true);
            Log.d("APPUA", "Error trying to log. ", e);
        }
        return success;
    }

    @Override
    protected void onPostExecute(Boolean success) {
        super.onPostExecute(success);

        if (success) {

            if (usuario.getConfirmado().equalsIgnoreCase("si")) {
                Intent intent = new Intent().setClass(
                    LoginActivity.this, MainActivity.class);
                Sharedpreferences.Editor editor =
sharedpreferences.edit();
                editor.putString("usuario", usuario.getEmail());
                editor.commit();
                intent.putExtra("usuario", usuario);
                Toast.makeText(LoginActivity.this,
                    "Login correcto ",
                    Toast.LENGTH_SHORT).show();
                startActivity(intent);
                finish();
            } else {
                showProgress(false);
                Toast.makeText(LoginActivity.this,
                    "Debes confirmar tu registro mediante el
correo enviado a tu cuenta ",

```

```

        Toast.LENGTH_LONG).show();
    }
} else {
    loadingTask = null;
    showProgress(false);
    Toast.makeText(LoginActivity.this,
        "Login or password are not correct.",
        Toast.LENGTH_LONG).show();
}
}

@Override
protected void onCancelled() {
    showProgress(false);
    loadingTask = null;
    Toast.makeText(LoginActivity.this,
        "try mas later, an error has occurred in the server",
        Toast.LENGTH_LONG).show();
}

}
}

```

Para hacer la comunicación con el servidor, se realizan una serie de pasos, la arquitectura de la aplicación se compone de tres capas, la vista, el controlador y el modelo de datos, en la clase java de la actividad, se hace una instancia de la clase Asynctasks, se realiza una llamada o varias a la capa controlador, la cual recibe un objeto o unos parámetros, en esa capa se crea un objeto json si es el caso de hacer un post o put o delete con un body:

```

JSONObject json = new JSONObject();
try {
    json.put("correo", email);
    json.put("contraseña", password);
} catch (JSONException e) {

}

```

Se crean las cabeceras, por ejemplo el tipo mime de los datos, en nuestro caso aceptamos y mandamos json.

```

Map<String, String> headers= new HashMap<String, String>();
headers.put("Accept", "application/json");
headers.put("Content-type", "application/json");

```

Por ultimo se hace la comunicación, y se recibe la respuesta con el código de estado y el contenido en formato json si está en la respuesta, si fuese el caso get u otro método que devuelve además del código de estado un cuerpo. El modelo de datos dispone de la lógica para parsear el contenido de respuesta de json a objeto, así para utilizar los datos de forma sencilla en las actividades.

Por ejemplo cuando se hace el login se llama al método loginUsuario con dos parámetros, se crea el body en json, además de los headers, y se llama al método get user de la clase RestHelper, para recibir la respuesta. Si el código de respuesta es 401 devuelve false, porque los datos no son correctos, si es 200 los datos son correctos, si no es ninguno de los

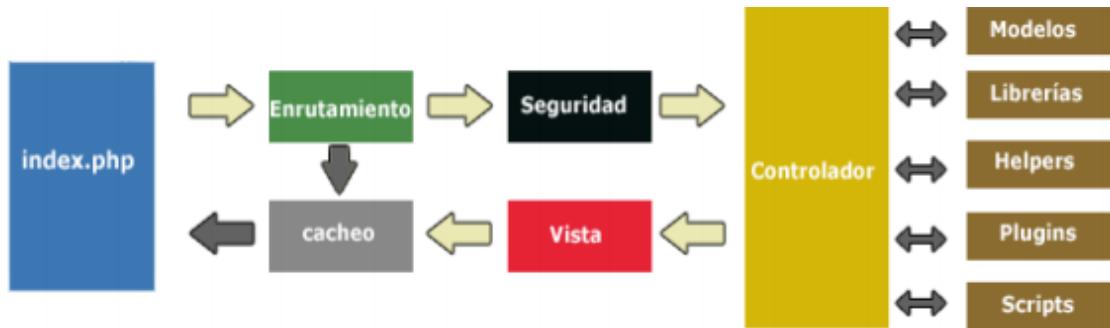
anteriores, porque ha habido un error en el servidor, se lanza una excepción, la cual se captura y se le notifica al usuario con un mensaje de que ha habido un error en el servidor.

```
public boolean loginUsuario(String email, String password) throws  
IOException, NetworkException {  
  
    JSONObject json = new JSONObject();  
    try {  
        json.put("correo", email);  
        json.put("contraseña", password);  
    } catch (JSONException e) {  
  
    }  
    Map<String, String> headers= new HashMap<String, String>();  
    headers.put("Accept", "application/json");  
    headers.put("Content-type", "application/json");  
  
    RestResponse response = restHelper.getUser(context, LOGIN_PATH,  
headers, json.toString());  
    //Log.d("Apua jsong", json.toString());  
    if (response.getHttpStatusCode() == 401) {  
        Log.d("Codigo Error :","401");  
        return false;  
    } else if (response.getHttpStatusCode() != 200) {  
        throw new NetworkException(new StringBuilder()  
            .append("HttpCode: ")  
            .append(response.getHttpStatusCode())  
            .append(" - ")  
            .append(response.getHttpContent())  
            .toString());  
    }  
    return true;  
}  
  
public RestResponse getUser(Context context, String url, Map<String,  
String> headers, String body)  
throws IOException, NetworkException {  
  
    if (!checkConnectivity(context)) {  
        throw new NetworkException(ERR_CONN_MSG);  
    }  
    HttpURLConnection urlConnection = null;  
    try {  
        urlConnection = getConnection(url, "POST", headers);  
        writepostData(urlConnection, body);  
        RestResponse response = connect(urlConnection);  
        //Log.d("url conection to string",response.getHttpContent());  
        return response;  
    } finally {  
        if (urlConnection != null) {  
            urlConnection.disconnect();  
        }  
    }  
}
```

Back end

Web

existe un procedimiento para atender una solicitud de página del cliente. Este proceso se realiza internamente por el propio CodeIgniter y de manera transparente. Durante el proceso participan varios módulos como el enrutamiento de la solicitud, la caché interna, etc.



el flujo de aplicación que implementa CodeIgniter, puedes seguir los siguientes puntos:

1. Toda solicitud de una página a partir de CodeIgniter comienza en un index.php que hay en la raíz del framework.
2. Luego se realiza un filtrado de la URL para saber cuál es elemento que tiene que procesar esta página.
3. Si la página se había generado antes y está en la caché de CodeIgniter, se devuelve el archivo de la caché ya generado, con lo que se ahorra procesamientos repetidos. La caché se puede configurar y si lo deseamos, incluso deshabilitar.
4. Antes de continuar con el proceso se realiza un tratamiento de seguridad sobre la entrada que tengamos, tanto de la información que haya en la URL como de la información que haya en un posible POST, si se ha configurado así.
5. El controlador adecuado realiza el procesamiento de la solicitud. CodeIgniter decide el controlador que debe procesar la solicitud en función de la URL solicitada.
6. El controlador comunica con una serie de módulos, los que necesite, para producir la página.
7. A través de las vistas adecuadas, el controlador genera la página, tal cual se tiene que enviar al navegador.
8. Si la página no estaba en la caché, se introduce, para que las futuras solicitudes de esta página sean más rápidas.

el funcionamiento empieza con con un URL: para dirigirse al controlador, como unidad de control central entre la vista y el modelo, es necesario introducir un URL en la barra de

búsqueda del navegador web. Para ello, los desarrolladores crean las denominadas clases de controlador, archivos PHP con diversas funciones que permiten cargar librerías, extensiones o helpers, establecer conexiones a bases de datos, integrar un modelo o seleccionar una vista determinada.

El flujo de aplicación de CodeIgniter se basa en el siguiente esquema de URL básico:

example.com/class/function/parameter

el caso de class/function existe la posibilidad que el usuario, pueda darle palabras intuitivas, por ejemplo si el usuario necesita registrarse, la clase puede ser Registro, y el método darseDeAlta, en el fichero de enrutamiento se puede relacionar cualquier ruta con la clase y el método, por ejemplo, si quisiera darse de alta, puede relacionar esa ruta example.com/public/registro/parameter con el controlador y el método con la forma siguiente Registro/darseDeAlta, con eso el sistema llama al controlador Registro y al método darseDeAlta, prepara la vista y devuelve el resultado a la petición.

CodeIgniter permite crear controladores individuales como clases definidas por el usuario. Para ello crean un archivo PHP separado para cada controlador en el directorio application/controllers/. Los controladores se crean como subclases de la clase CI_Controller. En el código fuente se realiza con ayuda de la palabra clave extends.

```
class Registro extends CI_Controller {  
}
```

Registro hereda todas las funciones de la visibilidad public y protected de la clase CI_Controller.

Cada clase también tiene un constructor, con lo cual se pueden integrar librerías, un modelo de datos, bases de datos o clases helper. Desde PHP5, se utiliza __construct() como estándar.

```
public function __construct() {  
    parent::__construct();  
  
    $this->load->database(); // podría hacerlo desde el autoload  
    $this->load->helper('url');  
    $this->load->library('session');  
    $this->load->model('CocheModel');  
  
}
```

Una vez creada la clase y definido el constructor, se crean los métodos, los cuales realizan una serie de operaciones, por ejemplo consultar datos del modelo, preparar la vista y pasar los datos consultados.

```
//funcion borrar ruta
public function borrarRutaCoches($id) {
    if($this->session->userdata('user')) {
        $data = array(
            'id' => $id,
        );
        $Resultado['ruta'] = $this->RutaModel-
>borrarRutaUsuario($data);

        $data2 = array(
            'usuario' => $this->session->userdata('user')->correo,
        );
        $output['rutas'] = $this->RutaModel-
>obtenerRutasUsuario($data2);
        if ($Resultado == true) {
            $output['Exito'] = 'exito';
        } else {
            $output['Error'] = 'error';
        }

        $this->load->view('private/listadoRutas', $output);
    } else{
        $this->load->view('public/home');
    }
}
```

En el ejemplo anterior, después que un usuario intente borrar una ruta, el cual se llama el método borrarRutaCoches, se comprueba si el usuario ha iniciado sesión, se coge el parámetro, se llama al modelo y se borra la ruta, prepara la respuesta, levanta la vista pasándole los datos del proceso, para reflejarlos al usuario, como resultado de la acción borrar ruta.

Para conectarse con la base de datos, y recuperar información solicitada, se utilizan los Modelos, Las clases modelo permiten a los desarrolladores definir funciones de forma individual para las operaciones de bases de datos. En este ejemplo se crea el modelo coche con el cual se realizan todas las operaciones posibles con la entidad coche, por ejemplo insertar, consultar, borrar y actualizar.

```
class CocheModel extends CI_Model
{
    function insertaCoche($data)
    {
        //Inserta coche en la base de datos
        $this->db->insert('coche', $data);
        return ($this->db->affected_rows() > 0);

    }
}
```

En el ejemplo anterior, en el modelo insertacoche, recibe los datos de un coche y los almacena en la base de datos.

En este ejemplo se hace una consulta para recuperar todos los datos del coche.

```
function get_contents($data)
{
    $this->db->select('*');
    $this->db->from('coche');
    $this->db->where('usuario', $data['usuario']);
    // $this->db->where('contraseña', $data['contraseña']);

    $query = $this->db->get();
    return $result = $query->row();
}
```

Enrutamiento con CodeIgniter

La dirección URL solicitada por el usuario, indica a CodeIgniter qué Controlador y qué método son invocados. Para ello, el framework utiliza el esquema clase/función/parámetro, un esquema básico que se puede modificar según las necesidades. CodeIgniter dispone del archivo routes.php en el directorio application/config/ que contiene un array denominado \$route, que es el que permite a los desarrolladores definir sus propios criterios de enrutamiento.

```
$route['registro'] = 'registro/index';
$route['registroUser'] = 'registro/indexRegister';
$route['confirmRegistro/(:any)'] = 'registro/confirmRegistro/$1';

$route['login'] = 'login/index';
$route['loginUser'] = 'login/indexLogin';
$route['cerrarSesion'] = 'login/unset_session_data'; // cerrar sesión

$route['private/modifyPerfil'] = 'modifyPerfil/index';
$route['private/coche'] = 'coche/index';
// foto del perfil
$route['private/fotoPerfil'] = 'foto/index';
// borrar foto
$route['private/borrarFoto'] = 'foto/borrarFoto';
// actualizar usuario
$route['updateUser'] = 'modifyPerfil/updateUser';
```

Cuando se requiere crear una regla de enrutamiento nueva para una dirección dinámica, los desarrolladores web disponen con CodeIgniter de dos opciones: definir entradas de enrutamiento para URL dinámicos con Wildcards (comodines) o con expresiones regulares.

El documento routes.php soporta dos tipos de comodín:

Comodines de routes.php	Descripción
:num	Actúa de comodín para números enteros
:any	Actúa de comodín para una cadena de caracteres (string)

Ejemplos:

```
$route['private/buscarAnuncios/(:num)'] = 'ruta/buscarAnuncios/$1';
//cambiar foto coche
$route['private/cambiarFotoCoche/(:any)'] = 'coche/fotoCoche/$1';
```

Móvil

El Backend en la parte móvil he creado un servicio REST con el framework Jersey que implementa y simplifica el framework JAX-RS y da más funcionalidades a los desarrolladores. Para llevar acabo el desarrollo he utilizado el IDE eclipse con la herramienta MVN para automatizar la construcción del proyecto, el cual genera inicialmente un fichero de configuración pom.xml para construir el proyecto. Por defecto el contenido del fichero de configuración es el siguiente.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.university</groupId>
  <artifactId>appua</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>appua</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

A través de este fichero se puede manejar las dependencias que se necesitan para el desarrollo, además se puede configurar la construcción del proyecto mediante la etiqueta <build> como se observa a continuación.

```
<build>
  <finalName>appua</finalName>
  <plugins>
    <plugin>
      <groupId>org.apache.tomcat.maven</groupId>
      <artifactId>tomcat7-maven-plugin</artifactId>
      <version>2.2</version>
      <configuration>
        <url>http://localhost:8080/manager/text</url>
        <server>miservidor</server>
        <path>/tfg</path>
      </configuration>
    </plugin>
  </plugins>
</build>
```

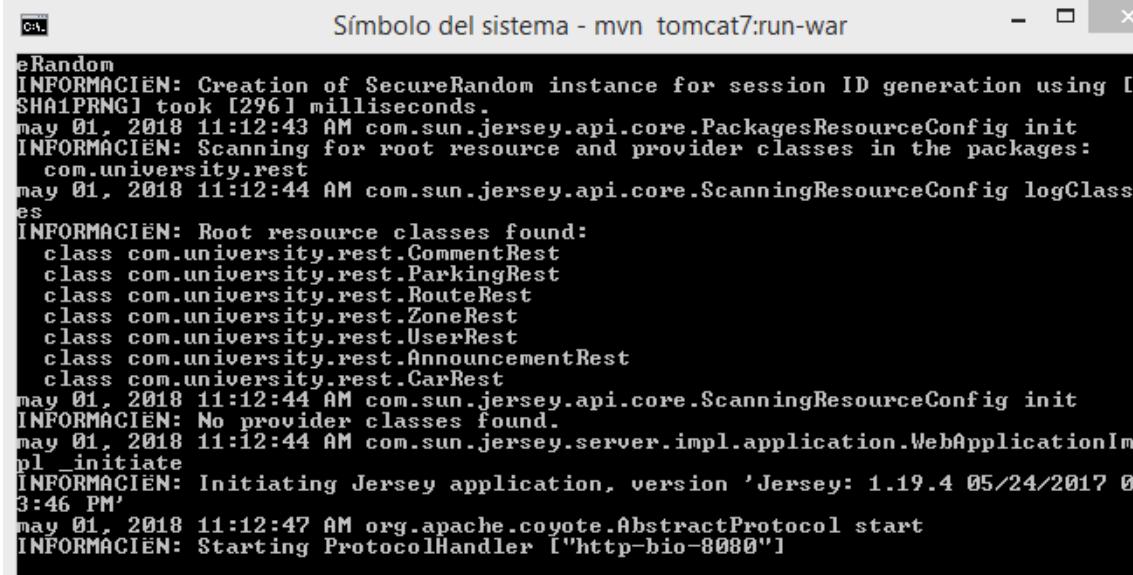
En el ejemplo anterior, le indico a mvn que añada el plugin tomcat a la aplicación. Por defecto, el plugin publica la aplicación con el nombre como contexto de aplicación. Se puede cambiar este contexto en la configuración del plugin añadiendo la entrada `<path>/nombre-contexto</path>`, en mi caso lo he configurado como `<path>/tfg</path>`.

Después de todo lo anterior, para crear el crear el fichero var y desplegar el proyecto en el servidor apache, que esta incluido por defecto en eclipse, ejecuto los comandos siguientes.

```
mvn package clean
```

```
mvn tomcat7:run-war
```

Como resultado de las acciones anteriores, el servicio web se despliega en el servidor apache, y se pone a recibir peticiones en el puerto 8080 como se observa a continuación.



A screenshot of a terminal window titled "Símbolo del sistema - mvn tomcat7:run-war". The window contains the output of a Maven command. The output shows the creation of a SecureRandom instance, scanning for root resources in the 'com.university.rest' package, and starting a Jersey application. It also shows the start of the Apache Coyote HTTP protocol on port 8080.

```
eRandom
INFORMACIÓN: Creation of SecureRandom instance for session ID generation using [SHA1PRNG] took [2961] milliseconds.
may 01, 2018 11:12:43 AM com.sun.jersey.api.core.PackagesResourceConfig init
INFORMACIÓN: Scanning for root resource and provider classes in the packages:
  com.university.rest
may 01, 2018 11:12:44 AM com.sun.jersey.api.core.ScanningResourceConfig logClasses
INFORMACIÓN: Root resource classes found:
  class com.university.rest.CommentRest
  class com.university.rest.ParkingRest
  class com.university.rest.RouteRest
  class com.university.rest.ZoneRest
  class com.university.rest.UserRest
  class com.university.rest.AnnouncementRest
  class com.university.rest.CarRest
may 01, 2018 11:12:44 AM com.sun.jersey.api.core.ScanningResourceConfig init
INFORMACIÓN: No provider classes found.
may 01, 2018 11:12:44 AM com.sun.jersey.server.impl.application.WebApplicationImpl $1 initiate
INFORMACIÓN: Initiating Jersey application, version 'Jersey: 1.19.4 05/24/2017 03:46 PM'
may 01, 2018 11:12:47 AM org.apache.coyote.AbstractProtocol start
INFORMACIÓN: Starting ProtocolHandler ["http-bio-8080"]
```

Una vez que tengo la estructura del proyecto, se implementan la logica de la aplicación, que serán capas con clases Java, en la parte Rest las clases utilizarán anotaciones JAX-RS para enlazar y mapear peticiones HTTP específicas a métodos java, los cuales servirán dichas peticiones. En el proyecto, para separar la lógica, he creado cuatro capas, capa Rest que contiene **recursos JAX-RS**, capa model que contiene **entidades de nuestro dominio**, capa **servicio** para separar la lógica de negocio de la capa Rest y por último la **capa Dao** para acceder a la base de datos. En este caso, voy a ilustrar con el ejemplo Usuario.

Clase de nuestro dominio (entidad): Usuario.java

```
package com.university.model;

import java.io.Serializable;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.security.spec.InvalidKeySpecException;

import java.util.UUID;

import java.security.*;

import javax.crypto.Cipher;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.PBEKeySpec;

import javax.crypto.spec.SecretKeySpec;

import javax.xml.bind.DatatypeConverter;

public class User implements Serializable {

    private static final long serialVersionUID = 1L;

    private String correo;

    private String nombre;

    private String apellido;

    private int edad;

    private String contraseña;

    private String telefono;

    private String salt;
```

```
private String foto;

private String detalles;

private String Confirmado;

private int opcion;

private static final int HASH_BYTE_SIZE = 64; // 512 bits

private static final int PBKDF2_ITERATIONS = 1000;

public User() {

}

public User(String correo, String nombre, String apellido, int edad, String contraseña,
String telefono, String salt, String foto, String detalles, String confirmado, int opcion) {

    super();

    this.correo = correo;

    this.nombre = nombre;

    this.apellido = apellido;

    this.edad = edad;

    this.contraseña = contraseña;

    this.telefono = telefono;

    this.salt = salt;

    this.foto = foto;

    this.detalles = detalles;

    Confirmado = confirmado;

    this.opcion = opcion;

}
```

```
public String getCorreo() {return correo; }

public void setCorreo(String correo) {this.correo = correo; }

public String getNombre() {return nombre; }

public void setNombre(String nombre) {this.nombre = nombre; }

public String getApellido() {return apellido; }

public void setApellido(String apellido) {this.apellido = apellido; }

public int getEdad() {return edad; }

public void setEdad(int edad) {this.edad = edad; }

public String getContraseña() {return contraseña; }

public void setContraseña(String contraseña) {this.contraseña = contraseña; }

public String getTelefono() {return telefono; }

public void setTelefono(String telefono) {this.telefono = telefono; }

public String getSalt() {return salt; }

public void setSalt(String salt) {this.salt = salt; }

public String getFoto() {return foto; }

public void setFoto(String foto) {this.foto = foto; }

public String getDetalles() {return detalles; }

public void setDetalles(String detalles) {this.detalles = detalles; }

public String getConfirmado() {return Confirmado; }

public void setConfirmado(String confirmado) {Confirmado = confirmado; }

public int getOpcion() {return opcion; }

public void setOpcion(int opcion) {this.opcion = opcion; }

public static long getSerialversionuid() {return serialVersionUID; }

}
```

Clases de nuestro servicio RESTful: UsuarioRest.java

Una vez definido el objeto de nuestro dominio que representará un objeto *Usuario*, vamos a ver cómo implementar el servicio para permitir a los usuarios de la parte cliente poder interactuar y realizar operaciones respecto al objeto usuario, como login, modificar datos, darse de baja del sistema, etc.

```
package com.university.rest;

import {...};
@Path("/UserService")
public class UserRest {
    @Context
    UriInfo uriInfo;
    UserService userService = new UserService();
```

La implementación del servicio, se denomina un recurso clase, es una clase java que utiliza anotaciones JAX-RS.

La anotación `@Path` indica que la clase `UserRest` es un servicio JAX-RS.

La anotación tiene el valor `/ UserService`. Este valor representa la raíz relativa de la URI de nuestro servicio RESTful. Si la URI absoluta de nuestro servidor <http://localhost:8080/tfg/rest>, los métodos expuestos por nuestra clase `UserService` estarían disponibles bajo la URI <http://localhost:8080/tfg/rest/UserService>.

Creacion de usuario:

```
@POST
@Path("/insert")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public Response insertUser(User user) {
    user.setFoto("foto");
    try {
        UserDTO usuario = userService.insertUser(user);
        userService.sendMail(usuario.getCorreo());
        return
    }
    Response.status(Response.Status.CREATED).entity(usuario).build();
} catch (ServiceException e) {
    return
}
Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();
}
```

Para enlazar peticiones HTTP POST con el método `insertUser()`, se le pone la anotación `@POST`. La anotación `@Path`, combinada con la anotación `@POST`, enlaza todas las peticiones POST dirigidas a la URI relativa `/insert` al método Java `insertUser()`.

La anotación @Consumes especifica que tipo de datos se incluye en el cuerpo del mensaje HTTP de entrada. Si el cliente incluye un tipo diferente de JSON, se envía un código de error al cliente.

En JAX-RS, cualquier parámetro no anotado con anotaciones JAX-RS se considera una representación del cuerpo del mensaje de la petición de entrada HTTP. Después se mapea el json contenido en el cuerpo de la petición http de entrada en una instancia de nuestra clase Usuario.

El método insertUser () devuelve una respuesta de tipo .Response. El método estático Response.created() crea un objeto *Response* que contiene un código de estado 201 Created. También añade el objeto Usuario creado.

Consulta de clientes:

```
@GET  
@Path("/userByMail/{correo}")  
@Produces(MediaType.APPLICATION_JSON)  
public Response getUserByMail(@PathParam("correo") String  
correo) {  
    UserDTO user;  
    try {  
        user = userService.obtenerDatosUserByMail(correo);  
        if (user != null) {  
            return Response.ok(user).build();  
        } else  
            return  
        Response.status(Response.Status.NOT_FOUND).build();  
    } catch (ServiceException e) {  
        return  
        Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();  
    }  
}
```

En este caso, anotamos el método getUserByMail () con la anotación @ GET para enlazar las operaciones HTTP GET con este método Java.

También anotamos getUserByMail () con la anotación @ PRODUCES. Esta anotación indica a JAX-RS que tipo de datos esta en la respuesta proporcionada por la operación GET. En este caso, estamos indicando que será de tipo MediaType.APPLICATION_JSON.

Modificación de usuario.

```
@PUT
@Path("/update")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public Response updateUser(User user) {

    try {
        userService.updateUser(user);
        return
Response.status(Response.Status.NO_CONTENT).build();
    } catch (ServiceException e) {
        return
Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();
    }
}
```

Para realizar una operación sobre la modificación de un usuario y enlazarla las peticiones HTTP PUT, anotamos el método `updateUser ()` con la anotación `@PUT`, el método `updateUser ()` está anotado adicionalmente con `@Path`, de forma que podamos atender peticiones a través de las URLs **/update**.

Construcción y despliegue del servicio.

Una vez implementado nuestro servicio RESTful, necesitamos poner en marcha el proceso de construcción. El proceso de construcción compilará, empaquetará, y finalmente nos permitirá desplegar nuestro servicio en el servidor de aplicaciones.

Para poder, empaquetar nuestro servicio RESTful como un *war*, que se desplegará en el servidor de aplicaciones, vamos a incluir un "proveedor" de servicios JAX-RS, en el descriptor de despliegue de nuestra aplicación (fichero *web.xml*).

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
    <display-name>Archetype Created Web Application</display-name>
    <servlet>
        <servlet-name>Jersey REST Service</servlet-name>
        <servlet-
class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-
class>
        <init-param>
            <param-name>com.sun.jersey.config.property.packages</param-
name>
            <param-value>com.university.rest</param-value>
        </init-param>
        <init-param>
            <param-name>com.sun.jersey.api.json.POJOMappingFeature</param-
name>
            <param-value>true</param-value>
        </init-param>

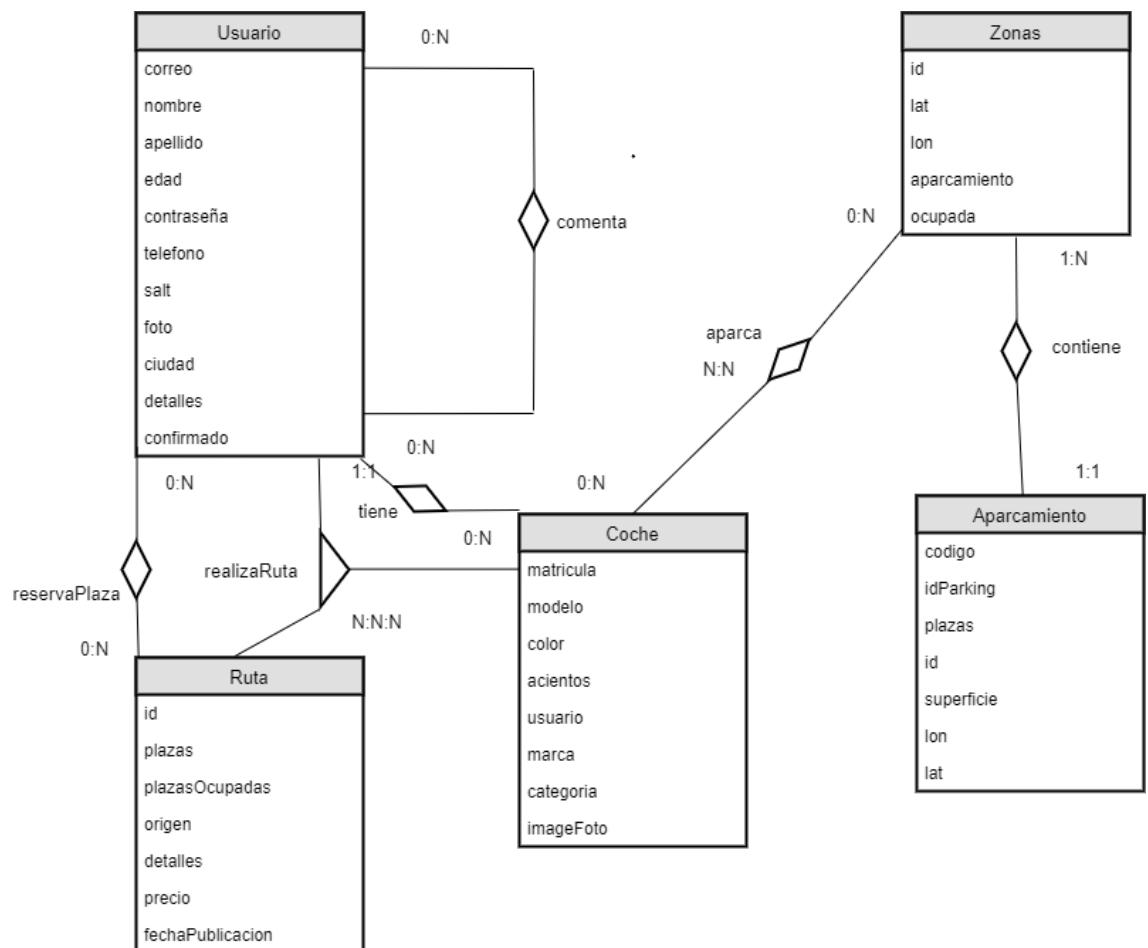
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Jersey REST Service</servlet-name>
        <url-pattern>/rest/*</url-pattern>
    </servlet-mapping>
</web-app>
```

En la configuración anterior estamos indicando de forma explícita el *servlet* JAX-RS que recibe las peticiones REST, que a su vez, utilizará la clase *Application* para detectar qué servicios y proveedores REST serán desplegados en el servidor.

Base de datos

Esquema conceptual

Para visualizar la base de datos de una forma más clara he tomado una captura al esquema Entidad-Relación que representa las entidades de la base de datos así como las relaciones entre las distintas entidades



Modelo Relacional

El modelo relacional nos aporta de una forma sencilla e intuitiva la representación de la base de datos.

Usuario(correo, nombre, apellidos, edad, contraseña, teléfono, salt, foto, ciudad, detalles, confirmado,opcion)

C.P.: correo

V.N.N.: correo, nombre, apellidos, edad, contraseña, teléfono, salt, ciudad, detalles, confirmado,opcion.

Coche(matricula, modelo, color, acientos, usuario, marca, categoría, iamgeFoto)

C.P.: matricula

C.aj.: usuario -> usuario

VNN.: modelo, color, asientos, usuario, marcia, categoría.

Ruta(id, plazas, plazasOcupadas, origen, detalles , precio, fechaPublicacion)

C.P.: Id

V.N.N.: plazas, plazasOcupadas, origen, destino, precio, fechaPublicacion.

Aparcamiento(Id, código, idParking, plazas, superficie, lon, lat)

C.P.: Id

V.N.N.: código, idParking, plazas, superficie, lon, lat.

Realiza_Ruta(usuario, coche, ruta)

C.P.: (usuario, coche, ruta)

C.aj.: usuario -> usuario

C.aj.: coche -> coche

C.aj.: ruta -> ruta

ReservaPlaza(usuario,ruta)

C.P.: (usuario, ruta)

C.aj.: usuario -> usuario

C.aj.: ruta -> ruta

Comenta(UsuarioA, UsuarioB, Comentario)

C.P.: (UsuarioA, UsuarioB)

C. Ajena: UsuarioA -> Usuario

C. Ajena: UsuarioB -> UsuarioB

V.N.N.: Comentario

Zonas(id, lat, lon, aparcamiento, ocupada)

C.P.: (id)

C. Ajena: aparcamiento -> aparcamiento

V.N.N.: lat, lon, aparcamiento, ocupada.

UsuarioAparcaCoche(coche, zona, fecha)

C.P.: (coche, zona)

C. Ajena: coche -> coche

C. Ajena: zona -> zonas

V.N.N.: fecha.

Diccionario de Datos

Otra forma de representar la información que contienen las tablas es a través del diccionario de datos, por ejemplo el diccionario de datos de la tabla usuario es la siguiente:

Donde podemos observar el nombre de la columnas, el tipo de datos, si es posible que el valor de los campos sea nulo, el valor por defecto que tendrán los campos si no se establecen y el comentario que sirve como nota aclaratoria.

Usuario

Columna	Tipo	Nulo	Predeterminado	Comentarios
Correo	Varchar	No		Correo del usuario
Nombre	Varchar	No		Nombre del usuario
Apellido	Varchar	No		Apellido del usuario
Edad	Int	No		Edad del usuario
Contraseña	Varchar	No		Contraseña del usuario para login
Telefono	Varchar	No		Teléfono del usuario
Salt	Varchar	No		Salt generada para crear la contraseña junto con el hash
Foto	Varchar	Sí	Null	Foto del usuario
detalles	Varchar	No		Detalles del usuario
Confirmado	Enum	No	0	1 indica que el usuario ha confirmado su registro mediante el

				correo recibido, 0 todavía el usuario no ha confirmado su registro
--	--	--	--	--

Coché

Columna	Tipo	Nulo	Predeterminado	Comentarios
matricula	Varchar	No		Correo del coche
modelo	Varchar	No		Modelo del coche
Color	Varchar	No		Color del coche
Asientos	Int	No		Acientos del coche
Usuario	Varchar	No		Usuario dueño del coche
Marca	Varchar	No		Marca del coche
Categoría	Varchar	No		Categoría del coche
ImagenFoto	Varchar	Sí	Null	Foto del coche

Ruta

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	Int	No		Identificador de la ruta
Plazas	Int	No		Plazas disponibles
Plazas ocupadas	Int	No		Plazas ocupadas, es decir, las que han sido solicitadas por los usuarios
Origen	Varchar	No		Punto de partida, es el origen donde sale el usuario
Detalles	Varchar	No		Detalles del viaje
Precio	Decimal	No		Precio por cada persona
FechaPublicacion	Date	No		Fecha cuando el usuario se ha dado de alta a la publicación de la ruta

Aparcamiento

Columna	Tipo	Nulo	Predeterminado	Comentarios
Id	Int	No		Numero incremental, se asigna a cada parking
Codigo	Varchar	No		Identificador del parking
IdParking	Int	No		Identificador de actividad según la tipología del aparcamiento
Plazas	Int	No		Numero de plazas que contiene
Superficie	Double	No		La superficie del parking
Lon	Double	No		Cordinada y
Lat	Double	No		Cordinada X

Zonas

Columna	Tipo	Nulo	Predeterminado	Comentarios
Id	Int	No		Identificador de la zona
Lat	Double	No		Cordinada x de la zona respecto del gps
Lon	Double	No		Cordinada y de la zona respecto a gps
Aparcamiento	Varchar	No		El aparcamiento al que pertenece la zona
Ocupada	Enum	No		0 indica que la zona no esta ocupada, 1 indica el contrario

Realiza ruta

Columna	Tipo	Nulo	Predeterminado	Comentarios
Usuario	Varchar	No		Usuario que publica la ruta
Coche	Varchar	No		Coche del usuario que publica la ruta
Ruta	Int	No		Identificador de la ruta

Reserva Ruta

Columna	Tipo	Nulo	Predeterminado	Comentarios
Usuario	Varchar	No		Usuario que publica la ruta
Ruta	Int	No		Identificador de la ruta

Usuario aparca coche

Columna	Tipo	Nulo	Predeterminado	Comentarios
Coche	Varchar	No		Coche la cual aparcra en una zona de un aparcamiento dado
Zona	Int	No		Zona de un aparcamiento
Fecha	Date	No		Fecha cuando el coche se aparaco

Comenta

Columna	Tipo	Nulo	Predeterminado	Comentarios
Id	Int	No		Identificador del comentario
UsuarioComentado	Varchar	No		Usuario que recibe el comentario
UsuarioComenta	Varchar	No		Usuario que realiza el comentario
Comentario	Varchar	No		El comentario introducido por el usuario

SEGURIDAD

Web

Balancear riesgo y usabilidad

Si bien la usabilidad y la seguridad en una aplicación web no son excluyentes la una de la otra, alguna medida tomada para incrementar la seguridad con frecuencia afecta a la usabilidad. Es conveniente emplear medidas de seguridad que sean transparentes a los usuarios y que no resulten engorrosas en su empleo. Por ejemplo, el uso de un login que solicita el nombre de usuario y contraseña, permite controlar el acceso de los usuarios hacia secciones restringidas de la aplicación.

Rastrear el paso de los datos

Existen funciones globales en la aplicación que sirven para identificar de forma clara las entradas proporcionadas por el usuario. En el caso de nuestra web se ha utilizado Sesión, mediante la cual se almacena la información del usuario y con esto se puede seguir el rastro durante su recorrido en la aplicación.

XSS Prevention

XSS significa cross-site scripting. CodeIgniter viene con XSS filtro de seguridad. Ese filtro evita cualquier javascript malicioso o cualquier código que intenta crear cookies y hacer actividades maliciosas, para filtrar los datos a través de XSS, se usa el método clean como se muestra abajo.

```
$data = $this->security->xss_clean($data);
```

Ademas de filtrar código, también existe otra manera para filtrar la fotos que se suben a la base de datos, poniendo otro campo opcional como true.

```
$data = $this->security->xss_clean($data, true);
```

Ataques de inyección SQL

Para evitar los ataques mediante inyección de SQL se filtrarán las entradas de datos para evitar los caracteres “especiales” que permiten realizar dichos ataques. Es muy importante realizar el filtrado en todas las posibles entradas de textos porque un ataque de inyección SQL puede suponer una gran pérdida de datos.

Contraseña del usuario

La seguridad de contraseñas es una de las partes mas importantes de la aplicación, así para evitar daños, o robo de información y como consecuencia manipular y poner en peligro información de los usuarios, he utilizado un cifrado muy complejo, utiliza SHA-512 como HASH y un salt aleatoria que se mezcla con el hash y construir la contraseña mediante el método crypt.

Cuando se lleva acabo el desarrollo, se guarda el resultado después de aplicar el método crypt, como contraseña cifrada, además se guarda el salt, porque cuando se realiza el login, se aplica el mismo método, el usuario introduce su contraseña, se le aplica el hash, se consulta el salt creado mediante el correo cuando se hizo el registro, se hace la mezcla y se compara con la contraseña cifrada en la base de datos.

Aquí esta el ejemplo del registro

```
public function indexRegister()
{
    $pass = hash('sha512', $this->input->post('password'));
    $salt = uniqid(mt_rand(), true);
    $pass = crypt($pass, $salt);
    $correo = $this->input->post('correo');
    $data = array(
        'correo' => $this->input->post('correo'),
        'nombre' => $this->input->post('nombre'),
        'apellido' => $this->input->post('apellido'),
        'edad' => $this->input->post('bday'),
        'contraseña' => $pass,
        'telefono' => $this->input->post('telefonoReg'),
        'detalles' => $this->input->post('detalles'),
        'salt' => $salt
    );

    if ($this->send_mail($correo)) {
        $this->RegistroModel->insertaUsuario($data);
        $Resultado = $this->RegistroModel->get_contents($data);
        $data['message'] = $Resultado;
    } else {
        $data['message'] = "el correo introducido no es
incorrecto";
    }
    $this->load->view('public/confirmRegister', $data);
}
```

Ejemplo del login:

```
public function indexLogin()
{
    //generar el hash a traves de la contraseña propuesta del usuario
    $pass =hash('sha512', $this->input->post('contraseña'));
    $user = $this->input->post('correo');
    $data = array(
        'correo' => $user,
    );
    $this->load->model('LoginModel');
    $salt=$this->LoginModel->get_salt($data);

    if(!empty($salt)){
        $pass=crypt($pass,$salt);
        $data2 = array(
            'correo' => $this->input->post('correo'),
            'contraseña' => $pass
        );
        $Resultado = $this->LoginModel->get_contents($data2);

        if (!empty($Resultado)) {
            if ($Resultado->confirmado == 'SI') {

                //Se guarda el objeto
                $this->session->set_userdata('user', $Resultado);

                redirect('/');
            }else{

                $output['Error'] = 'Debes confirmar tu registro
mediante el correo mandado con la direccion registrada';
            }
        }else{

            $output['Error'] = 'La contraseña es incorrecta';
        }
    }else{
        $output['Error'] = 'Los datos son incorrectos';
    }
    $this->load->view('public/log', $output);
}
```

Móvil

En cuanto a la seguridad en la parte móvil, utilizo el mismo método del cifrado de la contraseña como en la versión web, aplicar el hash sobre la contraseña, generar un salt, y por ultimo generar el cifrado de la contraseña.

```
byte[] hash = null;
SecureRandom sr = new SecureRandom();
byte[] salt = new byte[16];
sr.nextBytes(salt);

PBEKeySpec spec = new PBEKeySpec(password.toCharArray(),
salt, PBKDF2_ITERATIONS, HASH_BYTE_SIZE);
SecretKeyFactory skf =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512");

try {
    hash = skf.generateSecret(spec).getEncoded();

} catch (InvalidKeySpecException e) {
    e.printStackTrace();
}
String saltHex = DatatypeConverter.printHexBinary(salt);
String hashHex = DatatypeConverter.printHexBinary(hash);
```

APIS UTILIZADAS

Api Google maps

Web

Para reflejar los parkings de la ua, y facilitar a los usuarios ver los parkings y las zonas que se dispone la universidad, he utilizado la Api google maps que permite agregar un mapa de Google con un marcador al sitio web y así demostrar las zonas con puntos rojos.

Los pasos a seguir para crear un mapa de Google con un marcador en la página web:

Paso 1: En la pagina HTML, creo un elemento div con un identificador:

```
div id="map" style="width: 100%; height: 600px;"></div>
```

Paso 2: Agrega un mapa con un marcador

```
<script>
    //var jsn = [];
    var jsn = <?php if (isset ($Resultado)) {
        echo json_encode($Resultado);?>

        jsn = JSON.parse(jsn);

    <?php } else{ echo json_encode($corrdenes); }
    ?>

    function initMap() {

        var posArray = [];
        var count = 0;
        for(var i=0; i<jsn.length; i++) {
            //if (jsn[i].ocupada == '1') {
            <?php if (isset ($Resultado)){ ?>
                posArray[count] = {lat: parseFloat(jsn[i].lat), lng:
parseFloat(jsn[i].lon), id: jsn[i].id}
                count++;
            <?php } else{?>
                posArray[i] = {lat: parseFloat(jsn[i].lat), lng:
parseFloat(jsn[i].lon), id: jsn[i].id}
            <?php } ?>

        //}
    }

    var center = {lat: 38.385189, lng: -0.514053};

    var map = new google.maps.Map(document.getElementById('map'),
{
    zoom: 15,
    center: center,
    mapTypeId: 'satellite'
});

    var markerLabel = 'Z ';
    for (i = 0; i < posArray.length; i++) {
        console.log(posArray[i]);
        var marker = new google.maps.Marker({
            position: posArray[i],
            map: map,
            title: ""+i,
            label: {
                text: markerLabel+(posArray[i].id),
                color: "#FFFFFF",
                fontSize: "10px",
                fontWeight: "bold",
                borderColor: "#000000",
            }
        });
    }
}

</script>
```

Paso 3: Obtener una clave de API

Para conseguir la clave, ha de seguir unos pasos y conseguir la clave, como se refleja aquí.

Credenciales

Añadir credenciales al proyecto

- Averigua qué tipo de credenciales necesitas
Llamando a Google Maps JavaScript API

- 2 Ya tienes unas credenciales adecuadas para este fin

¿No quieres usar esta clave de API ya disponible? [Crear una nueva clave de API](#)

API key

Clave	AlzaSyAbvgfahBKagC52oJxu7necSaHbVBKIXzA
Tipo	Ninguna
Fecha de creación	5 may. 2017 18:54:39

[Listo](#) [Cancelar](#)

Y por ultimo agregar la clave al código.

```
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAG9NywQiRlbr3FJu
UhYBFCSFXZI79IF-tE&callback=initMap">
</script >
```



Móvil

Para incluir un mapa de la universidad en la aplicación móvil y reflejar los parkings con marcadores, se realizan los pasos siguiente:

Se crea un proyecto ne la pagina de Google.

The screenshot shows the 'Google APIs' section of the Google Cloud Platform. At the top, there's a banner for a free trial offer. Below it, the 'Nuevo proyecto' (New Project) button is highlighted. The 'Nombre del proyecto' (Project name) field contains 'APPUA'. The 'Ubicación' (Location) dropdown is set to 'Ninguna organización' (No organization). There are 'EXPLORAR' (Explore) and 'CREAR' (Create) buttons at the bottom. A status message below the location says 'ID del proyecto: appua-203210. No se puede cambiar más adelante.' (Project ID: appua-203210. Cannot be changed later.)

Como resultado de la creación del proyecto nos Crea una clave de api

The screenshot shows a modal dialog box titled 'Clave de API creada' (API key created). It contains instructions to transfer the key as a parameter and provides the generated key value: 'AIzaSyAQuqHG90mIubqXaH74u1URROsKGeSNqps'. A warning message encourages users to restrict the key. At the bottom, there are 'CERRAR' (Close) and 'RESTRINGIR CLAVE' (Restrict key) buttons.

Despues se introducir huella digital generada durante la creación de la aplicación Android, y se pulsa el botón guardar.

Las restricciones de aplicación especifican qué sitios web, direcciones IP o aplicaciones pueden usar esta clave.

Restricciones de aplicación

- Ninguna
- URLs de referencia HTTP (sitios web)
- Direcciones IP (servidores web, tareas cron, etc.)
- Aplicaciones para Android
- Aplicaciones para iOS

Restringir el uso a tus aplicaciones Android (Opcional)

Añade el nombre del paquete y la huella digital del certificado de firma SHA-1 para restringir el uso de tus aplicaciones de Android.

Puedes encontrar el nombre del paquete en el archivo `AndroidManifest.xml`. A continuación, usa el comando siguiente para obtener la huella digital:

```
$ keytool -list -v -keystore mystore.keystore
```

Nombre de paquete	Huella digital de certificado SHA-1
com.example.walid.appua	D2:04:57:B5:25:2D:C5:02:50:27:20:F7:27:27:1C:83:71:70:92:E7
+ Añadir nombre de paquete y huella digital	

Nota: Pueden pasar hasta 5 minutos antes de que se aplique la configuración.

[Guardar](#) [Cancelar](#)

Credenciales

[Credenciales](#) [Pantalla de autorización de OAuth](#) [Verificación de dominio](#)

[Crear credenciales](#) [Eliminar](#)

Crea credenciales para acceder a tus API habilitadas. Si quieres obtener más información, [consulta la documentación sobre las API](#).

Claves de API

Nombre	Fecha de creación	Restricciones	Clave
Clave APPUA	5 may. 2018	Aplicaciones para Android	AlzaSyAQuqHG90mlubqXaH74uiURROsKGeSNqps

Despues de los pasos anteriores, se habilitar una serie de librerias necesarias para la creación del mapa en Android.

 Google Maps Embed API Google Make places easily discoverable with interactive Google Maps.	 Google Maps JavaScript API Google Maps for your website	 Google Places API for iOS Google Make your iOS app stand out with detailed information about 100 million places	 Maps Elevation API Google Elevation data for any point in the world.
 Maps SDK for Android Google Maps for your native Android app.	 Maps SDK for iOS Google Maps for your native iOS app.	 Places API Google Get detailed information about 100 million places	 Places SDK for Android Google Make your Android app stand out with detailed information about 100 million places

Una vez hechos todos los pasos anteriores, se implementa la parte lógica en la clase java de la actividad, para consultar las cordenadas de la base de datos, e imprimir los parkings con marcadores.

```
View view = inflater.inflate(R.layout.fragment_layout_tree, container,
false);

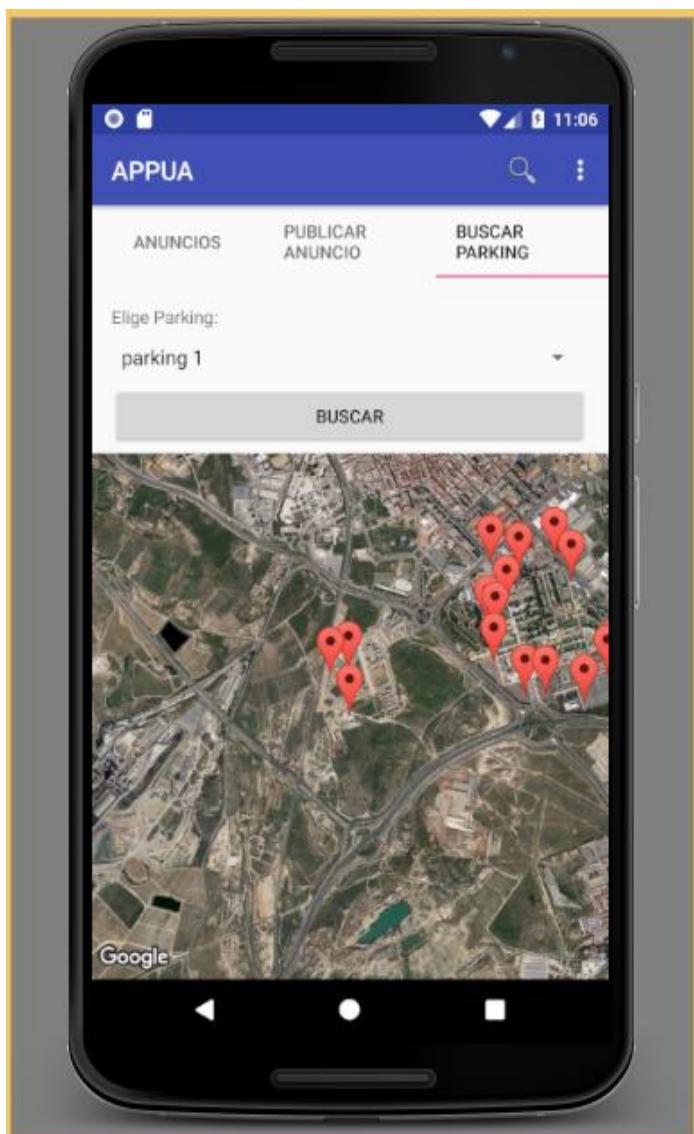
SupportMapFragment mapFragment = (SupportMapFragment)
this.getChildFragmentManager()
.findFragmentById(R.id.map);
mapFragment.getMapAsync(this);

@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    for (int i=0; i < parkingsList.size(); i++) {

        LatLng ua = new LatLng(parkingsList.get(i).getLat(),
parkingsList.get(i).getLon());
        mMap.addMarker(new
MarkerOptions().position(ua).title("parking: " + (i+1) +
"\naparcamiento: "+parkingsList.get(i).getCodigo() +
"\nsuperficie: " + parkingsList.get(i).getSuperficie() +
"\nplazas: " + parkingsList.get(i).getPlazas() + "\nlatitude: " +
parkingsList.get(i).getLat() +
"\nlongitude: " + parkingsList.get(i).getLon()));
    }
}
```

```
mMap.moveCamera(CameraUpdateFactory.newLatLng(ua));
mMap.setMapType(mMap.MAP_TYPE_SATELLITE);
}
}
```

Como resultado de los pasos anteriores, se consigue el mapa de la universidad con los parkings indicados con marcadores de color rojo.



SIGUA

SIGUA proporciona un interfaz de programación (API) para desarrolladores de aplicaciones que necesiten consumir datos de la base de datos geográfica. Se trata de un servicio web de tipo REST (*Representational State Transfer*), un estilo de arquitectura de software que emplea el conjunto básico de instrucciones HTTP para realizar operaciones de creación, lectura, modificación y borrado de datos en entornos web en contraposición a protocolos más complejos como SOAP o RPC. El API REST de SIGUA está basado en Slim para PHP 5, un *micro framework* de código abierto, y su principal consumidor es la propia aplicación WebGIS de SIGUA.

El API REST se estructura en varios niveles, entre los que destacan los de acceso público, que pueden ser consumidos tanto por las unidades y departamentos de la Universidad de Alicante que lo precisen como por desarrolladores externos. En este sentido, se ofrecen recursos para la obtención de datos referidos a entidades comunes de la base de datos geográfica (sedes, edificios, estancias, actividades, ubicación de puestos de trabajo, etc)

Para consumir los recursos públicos del API REST se debe consultar previamente la documentación y utilizar el punto de acceso cuyas URL se indican a continuación:

URL de la Documentación	URI Base
https://bitbucket.org/SIGUA/apirest-doc/src/master/specs.mkd	http://www.sigua.ua.es/api

Esquema:

```
"required":true,
"properties":{
    "codigo": {
        "type":"string",
        "required":true,
        "description":"Código SIGUA del aparcamiento"
    },
    "lon": {
        "type":"number",
        "required":true,
        "description":"Longitud del centroide en WGS84"
    }
    "lat": {
        "type":"number",
        "required":true,
        "description":"Latitud del centroide en WGS84"
    }
    "denominacion": {
        "type":["string", "null"],
        "required":true,
        "description":"Denominación conocida o código corporativo del
aparcamiento"
    },
    "id_actividad": {
        "type":"integer",
        "required":true,
        "description":"Identificador de actividad según la tipología del
aparcamiento"
    },
    "nombre_actividad": {
        "type":"string",
        "required":true,
        "description":"Denominación de la tipología del aparcamiento"
    },
    "superficie": {
        "type":"number",
        "required":true,
        "description":"Superficie en metros cuadrados"
    },
    "plazas": {
        "type":"number",
        "required":true,
        "description":"Número de plazas"
    }}}},
    "type": {
        "type":"string",
        "required":true
    }}}},
    "type": {
        "type":"string",
        "required":true
    }}}
```

GET Requests:

- Recupera todas las zonas de aparcamiento en batería
 - /pub/aparcamientos/bateria/items
- Recupera todas las zonas de aparcamiento en línea
 - /pub/aparcamientos/linea/items
- Recupera todas las zonas de aparcamiento de autobuses
 - /pub/aparcamientos/bus/items
- Recupera todas las plazas de aparcamiento adaptadas
 - /pub/aparcamientos/adaptado/items

PRUEBAS

Web

Para realizar las pruebas sobre la aplicación voy a realizar pruebas funcionales con la herramienta Selenium IDE, El cual es un conjunto de herramientas de pruebas open-source para automatizar pruebas funcionales sobre aplicaciones Web.

La forma más sencilla de escribir scripts de pruebas con Selenium, utilizar la herramienta Selenium IDE. Se instala como un plugin de Firefox, y nos permite crear scripts de pruebas utilizando la aplicación web tal y como un usuario haría normalmente: a través del navegador. Además, el mismo script de pruebas se puede ejecutar en diferentes navegadores

El objetivo de realizar pruebas es encontrar el mayor número de errores posibles realizando el menor número de pruebas. Las pruebas son importantes porque de ellas dependen el éxito del proyecto, las pruebas las realizamos para contribuir al éxito del proyecto en la satisfacción al cliente. Las pruebas más relevantes realizadas son las siguientes:

Casos de prueba de la pantalla de registro.

registro		
open	http://localhost/proyecto-master/	
clickAndWait	link=Register	
type	id=inputNombre	usuario300
type	id=Apellido	usuario300
type	id=correoElectronico	usuario.gmail.com
fireEvent	id=correoElectronico	blur
verifyText	id=messageCorreo	El formato del correo no es correcto
type	id=correoElectronico	usuario30@gmail.com
fireEvent	id=correoElectronico	blur
verifyText	id=messageCorreo	Ya hay un usuario con esta cuenta, introduzca otra cuenta
type	id=correoElectronico	occc10@hotmail.com
fireEvent	id=correoElectronico	blur
type	id=telefonoReg	56478
fireEvent	id=telefonoReg	blur
verifyText	id=messageReg	El telefono debe empezar por 6 o 7
type	id=telefonoReg	654789
fireEvent	id=telefonoReg	blur
verifyText	id=messageReg	El numero debe contener 9 cifras
type	id=telefonoReg	654789654
fireEvent	id=telefonoReg	blur
type	id=passwordReg	kjbjhff
fireEvent	id=passwordReg	blur
verifyText	id=passwordRegid	La contraseña debe contener mas de 7 caracteres
type	id=passwordReg	aaaaaaa1
fireEvent	id=passwordReg	blur
verifyText	id=passwordRegid	La contraseña debe contener al menos una letra mayuscula
type	id=passwordReg	aaaaaaAA

fireEvent	id=passwordReg	blur
verifyText	id=passwordRegid	La contraseña debe contener al menos un digito
type	id=passwordReg	aaaaaaA1
fireEvent	id=passwordReg	blur
type	id=bday	22
type	id=detalles	suelo salir de Alicante sobre las 8 de la
type	id=bday	22
clickAndWait	id=boUsuarioReg	
verifyText	css=h1	Informacion
verifyText	css=strong	Success!

Como se ve en la pantalla anterior, en la primera columna contiene los comandos, la segunda contiene la localización del elemento sobre el cual realizar el evento y por ultimo el valor, los comandos se pueden generar automáticamente cuando el usuario interactua con la aplicación a testear o crear los comandos de forma manual.

El caso anterior, se comprueba todos los datos y sus posibles mensajes cuando hay un error, se introduzca un dato incorrecto y cuando se dispara un evento mediante jquery se crea un elemento con un mensaje del tipo del error y luego se comprueba con selenium IDE si el resultado es el deseado, Como resultado después de ejecutar el tests

The screenshot shows the Selenium IDE interface with the following details:

- Title Bar:** registro (untitled suite) - Selenium IDE 2.9.1
- Menu Bar:** Archivo (F) Editar Actions Options Ayuda
- Toolbar:** Base URL: http://localhost/ (with dropdown), Fast Slow, Run, Stop, Refresh, Help.
- Test Case Panel:** Shows "registro" under "Test Case". It displays "Runs: 1" and "Failures: 0".
- Command Table:** A table showing the recorded test steps:

Command	Target	Value
verifyText	id=passwordRegid	La contraseña debe contener al menos un digito
type	id=passwordReg	aaaaaaA1
fireEvent	id=passwordReg	blur
type	id=bday	22
type	id=detalles	suelo salir de Alicante sobre las 8 de la
type	id=bday	22
clickAndWait	id=boUsuarioReg	
verifyText	css=h1	Informacion
verifyText	css=strong	Success!
- Log Panel:** Displays the execution log:

[info] Executing: type id=bday 22
[info] Executing: type id=detalles suelo salir de Alicante sobre las 8 de la
[info] Executing: type id=bday 22
[info] Executing: clickAndWait id=boUsuarioReg
[info] Executing: verifyText css=h1 Informacion
[info] Executing: verifyText css=strong Success!
[info] Test case passed

Casos de prueba de la pantalla de intento Inicio de Sesión con mensaje de confirmar rehistro

En este caso cuando el usuario se registra y todo es correcto, cuando intenta iniciar sesión sin confirmar el registro a través de un correo mandado, le sale el mensaje que debe confirmar el registro, con este script se comprueba que si un usuario no confirma el registro no inicia sesión y le sale el mensaje correspondiente.

login		
open	http://localhost/proyecto-master/	
clickAndWait	link=Login	
verifyText	css=h1	Identificate
type	id=correoElectronico	occc10@hotmail.com
type	id=password1	aaaa
clickAndWait	css=input.btn.btn-primary	
verifyText	css=div.alert.alert-danger	Info! La contraseña es incorrecta
type	id=correoElectronico	occc10@hotmail.com
type	id=password1	aaaaaaA1
clickAndWait	css=input.btn.btn-primary	
verifyText	css=div.alert.alert-danger	Info! Debes confirmar tu registro mediante el correo mandado con la dirección registrada

Resultado de ejecución de este script

The screenshot shows the Selenium IDE interface with the following details:

- Test Case:** login
- Runs:** 1
- Failures:** 0
- Log:**
 - [info] Executing: |verifyText | css=div.alert.alert-danger | Info! La contraseña es incorrecta |
 - [info] Executing: |type | id=correoElectronico | occc10@hotmail.com |
 - [info] Executing: |type | id=password1 | aaaaaaA1 |
 - [info] Executing: |clickAndWait | css=input.btn.btn-primary |
 - [info] Executing: |verifyText | css=div.alert.alert-danger | Info! Debes confirmar tu registro mediante el correo mandado con la dirección registrada |
 - [info] Test case passed

Casos de prueba de la pantalla de Inicio de Sesión después de confirmar registro

loginConfirm		
open	http://localhost/proyecto-master/	
clickAndWait	link=Login	
verifyText	css=h1	Identificate
type	id=correoElectronico	occc10@hotmail.com
type	id=password1	aaaaaaAa
clickAndWait	css=input.btn.btn-primary	
verifyText	css=div.alert.alert-danger	Info! La contraseña es incorrecta
type	id=correoElectronico	occc10@hotmail.com
type	id=password1	aaaaaaA1
type	id=correoElectronico	occ@hotmail.com
clickAndWait	css=input.btn.btn-primary	
verifyText	css=div.alert.alert-danger	Info! Los datos son incorrectos
type	id=password1	
type	id=correoElectronico	occc10@hotmail.com
type	id=password1	ddddddD1
clickAndWait	css=input.btn.btn-primary	
verifyText	css=h1	Bienvenido a la universidad de Alicante
click	//button[@type='button']	
clickAndWait	link=Editar perfil	
verifyText	css=h1	Información personal
click	css=body	
click	//button[@type='button']	
clickAndWait	link=Cerrar sesion	

Ejecución del test.

loginConfirm (untitled suite) - Selenium IDE 2.9.1

Archivo (F) Editar Actions Options Ayuda

Base URL: http://localhost/

Test Case: loginConfirm

Runs: 1 Failures: 0

Table Source

Command	Target	Value
type	id=password1	aaaaaaA1
clickAndWait	css=input.btn.btn-primary	
verifyText	css=h1	Bienvenido a la universidad d...
click	//button[@type='button']	
clickAndWait	link=Editar perfil	
verifyText	css=h1	Información personal
click	css=body	
click	//button[@type='button']	
clickAndWait	link=Cerrar sesion	

Command: type
Target: id=password1
Value: aaaaaaA1

Log Reference UI-Element Rollup

type(locator, value)

Arguments:

- locator - an element locator
- value - the value to type

Sets the value of an input field, as though you typed it in.

Can also be used to set the value of combo boxes, check boxes, etc. In these cases, value should be the value of the option selected, not the visible text.

Caso de prueba para modificar un dato

modificarInformacion		
open	http://localhost/proyecto-master/	
clickAndWait	css=span.glyphicon.glyphicon-log-in	
type	id=password1	aaaaaaA2
type	id=correoElectronico	occc10@hotmail.com
type	id=password1	ddddddD1
clickAndWait	css=input.btn.btn-primary	
click	//button[@type='button']	
clickAndWait	link=Editar perfil	
type	name=bdy	24
storeValue	name=bdy	edad
clickAndWait	id=boModifyPefil	
verifyText	css=strong	Success!
clickAndWait	link=informacion personal	
echo	\${edad}	
verifyValue	name=bdy	\${edad}
click	//button[@type='button']	
clickAndWait	link=Cerrar sesion	
verifyText	css=h1	Bienvenido a la universidad de Alicante

Resultado ejecución del test

modificarInformacion (untitled suite) - Selenium IDE 2.9.1

Archivo (F) Editar Actions Options Ayuda

Base URL <http://localhost/>

Test Case

modificarInformacion

Runs: 1 Failures: 0

Table Source

Command	Target	Value
storeValue	name=bday	edad
clickAndWait	id=boModifyPerfil	
verifyText	css=strong	Success!
clickAndWait	link=informacion personal	
echo	edad	

Command type
Target id=password1
Value aaaaaaA1

Log Reference UI-Element Rollup Info Clear

```
[info] Executing: |clickAndWait| link=Editar perfil ||  
[info] Executing: |type| name=bday | 24 |  
[info] Executing: |storeValue| name=bday | edad |  
[info] Executing: |clickAndWait| id=boModifyPerfil ||  
[info] Executing: |verifyText| css=strong | Success! |  
[info] Executing: |clickAndWait| link=informacion personal ||  
[info] Executing: |echo| ${edad} ||  
[info] echo: 24  
[info] Executing: |verifyValue| name=bday | ${edad} |  
[info] Executing: |click| //button[@type='button'] ||  
[info] Executing: |clickAndWait| link=Cerrar sesion ||  
[info] Executing: |verifyText| css=h1 | Bienvenido a la universidad de Alicante |  
[info] Test case passed
```

Caso de prueba insertar coche, publicar ruta, y actualizar información del anuncio.

plazasOcupadas		
open	http://localhost/proyecto-master/	
clickAndWait	link=Login	
type	id=correoElectronico	occc10@hotmail.com
type	id=password1	ddddddD1
clickAndWait	css=input.btn.btn-primary	
clickAndWait	link=Publicar anuncio	
verifyText	css=h3	Publicar Ruta
verifyText	css=div.alert.alert-info	Info! No tienes coche para publicar anuncio, ponga datos del coche, y luego, empieze a publicar anuncios.
click	//button[@type='button']	
clickAndWait	css=ul.dropdown-menu > li > a	
verifyText	css=h1	Información personal
clickAndWait	link=coche	
type	id=matricula	3652HHH
type	name=fotofile	C:\Users\walid\Desktop\Escritorio\ferrari.jpg
clickAndWait	id=boCoche	
verifyText	css=strong	Success!
clickAndWait	link=Publicar anuncio	
type	id=origen	Elche
type	id=precio	6
type	id=plaza	4
type	id=detalle	suelo salir una hora antes.
clickAndWait	css=input.btn.btn-primary	
verifyText	css=strong	Success!
clickAndWait	link=Viajes publicados	
verifyText	css=h2	Rutas publicadas
verifyText	css=td	Elche
verifyText	//td[2]	4

clickAndWait	link=Ver Detalles	
verifyText	css=td	Elche
clickAndWait	link=Viajes publicados	
clickAndWait	link=Actualizar	
type	id=plazaOcupadas	2
storeValue	id=plazaOcupadas	valor
clickAndWait	css=input.btn.btn-primary	
verifyText	css=h3	Actualizar Ruta
verifyText	css=strong	Success!
clickAndWait	link=Viajes publicados	
clickAndWait	link=Ver Detalles	
verifyText	//td[3]	\${valor}
selectWindow	null	
click	//button[@type='button']	
clickAndWait	css=ul.dropdown-menu > li > a	
clickAndWait	link=coche	
clickAndWait	link=Eliminar coche	
verifyText	css=strong	Success!
click	//button[@type='button']	
clickAndWait	link=Cerrar sesion	

Resultado ejecución del test

plazasOcupadas (untitled suite) - Selenium IDE 2.9.1

Archivo (F) Editar Actions Options Ayuda

Base URL <http://localhost/>

Test Case **plazasOcupadas**

Runs: 1 Failures: 0

Table Source

Command	Target	Value
open	http://localhost/proyecto-mast..	
clickAndWait	link=Login	
type	id=correoElectronico	occc10@hotmail.com
type	id=password1	aaaaaaA1
clickAndWait	css=input.htm.htm-primary	

Log Reference UI-Element Rollup Info Clear

```
[info] Executing: |verifyText | css=strong | Success! |
[info] Executing: |clickAndWait | link=Viajes publicados ||
[info] Executing: |clickAndWait | link=Ver Detalles ||
[info] Executing: |verifyText | //td[3] | ${valor} |
[info] Executing: |selectWindow | null ||
[info] Executing: |click | //button[@type='button'] ||
[info] Executing: |clickAndWait | css=ul.dropdown-menu > li > a ||
[info] Executing: |clickAndWait | link=coche ||
[info] Executing: |clickAndWait | link=Eliminar coche ||
[info] Executing: |verifyText | css=strong | Success! |
[info] Executing: |click | //button[@type='button'] ||
[info] Executing: |clickAndWait | link=Cerrar sesion ||
[info] Test case passed
```

Caso de prueba buscar aparcamiento.

buscarParking		
open	http://localhost/proyecto-master/	
clickAndWait	link=Login	
type	id=password1	aaaaaaA2
type	id=correoElectronico	occc10@hotmail.com
type	id=password1	ddddddD1
clickAndWait	css=input.btn.btn-primary	
clickAndWait	link=Buscar parking	
verifyText	css=h3	Buscar Parking
clickAndWait	css=input.btn.btn-primary	
verifyText	css=label	Elige Zona para aparcar y actualizar para indicar como zona ocupada:
clickAndWait	css=input.btn.btn-primary	
verifyText	css=h3	Zona ocupada
verifyText	css=div.alert.alert-info	Info! Debes desocupar la zona ocupada y luego puedes volver a aparcar.
clickAndWait	link=Buscar parking	
verifyText	css=h3	Buscar Parking
clickAndWait	css=input.btn.btn-primary	
verifyText	css=div.alert.alert-info	Info! Debes desocupar la zona ocupada y luego puedes volver a buscar aparcamiento.
clickAndWait	css=input.btn.btn-primary	
verifyText	css=h3	Buscar Parking
clickAndWait	css=input.btn.btn-primary	
verifyText	css=label	Elige Zona para aparcar y actualizar para indicar como zona ocupada:
clickAndWait	css=input.btn.btn-primary	
verifyText	css=div.alert.alert-info	Info! Debes desocupar la zona ocupada y luego puedes volver a aparcar.
clickAndWait	css=input.btn.btn-primary	
verifyText	css=h3	Buscar Parking
click	//button[@type='button']	
clickAndWait	link=Cerrar sesion	

Resultado ejecución del script:

The screenshot shows the Selenium IDE interface with the following details:

- Title Bar:** buscarrParking (untitled suite) - Selenium IDE 2.9.1
- Toolbar:** Archivo (F) Editar Actions Options Ayuda; Base URL: http://localhost/
- Test Case List:** Test Case, buscarrParking (highlighted in green)
- Run Summary:** Runs: 1, Failures: 0
- Script Editor:** Shows the recorded Selenium script steps in a table format.
- Log Panel:** Displays the execution log with info messages for each step, including UI interactions and alert messages.

```

[info] Executing: |clickAndWait| css=input.btn.btn-primary |
[info] Executing: |verifyText| css=div.alert.alert-info | Info! Debes desocupar la zona ocupada y luego puedes volver a buscar aparcamiento. |
[info] Executing: |clickAndWait| css=input.btn.btn-primary |
[info] Executing: |verifyText| css=h3 |Buscar Parking|
[info] Executing: |clickAndWait| css=input.btn.btn-primary |
[info] Executing: |verifyText| css=label |Elige Zona para aparcar y actualizar para indicar como zona ocupada:|
[info] Executing: |clickAndWait| css=input.btn.btn-primary |
[info] Executing: |verifyText| css=div.alert.alert-info | Info! Debes desocupar la zona ocupada y luego puedes volver a aparcar. |
[info] Executing: |clickAndWait| css=input.btn.btn-primary |
[info] Executing: |verifyText| css=h3 |Buscar Parking|
[info] Executing: |click| //button[@type='button'] |
[info] Executing: |clickAndWait| link=Cerrar sesion |
[info] Test case passed
  
```

CONCLUSIONES, TRABAJOS FUTUROS Y POSIBLES MEJORAS

Una vez terminado el desarrollo del trabajo fin de grado, puedo concluir que he cumplido con los objetivos propuestos inicialmente.

He implementado una base de datos sql que sirve como mecanismo de persistencia, para que tanto la versión web como móvil puedan interactuar con la base de datos a la hora de realizar las consultas y las operaciones sobre ella.

Respecto al proyecto, he creado dos versiones, una aplicación web y otra Android, la web se interactúa directamente con la base de datos, Sin embargo, para la versión Android, he creado una api Restful Java con el framework jersey, la cual tiene la implementación necesaria para recibir peticiones y procesar los resultados a la aplicación Android.

Mejoras y Ampliaciones

Al fin y al cabo, las aplicaciones están en permanente evolución por tanto una serie de mejoras y ampliaciones que tendrá la aplicación en el futuro son las siguientes:

General:

- Incluir un chat, así los usuarios puedan comunicarse de forma privada.
- Incluir la funcionalidad de seguir usuarios.
- Posibilidad de compartir los anuncios mediante redes sociales como por ejemplo whastapp, twitter y Facebook.
- Añadir más posibilidades de filtrado a la hora de realizar las búsquedas. Dar la posibilidad de filtrar anuncios que cumplan con un precio que este en un determinado intervalo o que sea mayor o menor a un determinado precio.
- Adaptar los anuncios destacados en base a las preferencias, gustos y necesidades de los usuarios. Para ello se monitorizará las búsquedas que realizan los usuarios en el buscador de la aplicación para descubrir sus preferencias, así como también ver cuál ha sido su historial en la aplicación.
- Dar la posibilidad de iniciar sesión/registrarse con la cuenta de Gmail o Facebook.
- Poder gestionar los pagos online.

Móvil:

- Incluir todas las funcionalidades tanto en la parte web como en la parte móvil, ya que la web tiene mas funcionalidades que la web.
- Añadir un segundo factor a la autenticación, para obtener mayor seguridad y aprovechar las ventajas de los móviles más modernos. Dicho de otra manera, permitir conectarse en la aplicación mediante huella dactilar (factor de autentificación basado en una característica física)
- Permitir al usuario tomar capturas, para la foto del perfil com la imagen del coche.

REFERENCIAS

- <https://www.formget.com/codeigniter-uri-segment/>
- <https://validator.w3.org/nu/#textarea>
- https://www.w3schools.com/bootstrap/bootstrap_templates.asp
- <https://www.w3schools.com/html/default.asp>
- <https://www.w3schools.com/css/default.asp>
- <https://www.w3schools.com/js/default.asp>
- <https://www.w3schools.com/sql/default.asp>
- <https://www.w3schools.com/php/default.asp>
- <https://www.w3schools.com/jquery/default.asp>
- <http://librosweb.es/>
- https://www.codeigniter.com/user_guide/
- <https://codepen.io/benske/pen/iAgpq>
- <https://web.ua.es/es/sigua/api-rest.html>
- <https://bitbucket.org/SIGUA/apirest-doc/src/master/specs.mkd?fileviewer=file-view-default>
- https://www.tutorialspoint.com/codeigniter/codeigniter_security.htm
- http://www.w3ii.com/es/codeigniter/codeigniter_security.html
- https://codeigniter.com/user_guide/database/queries.html
- <https://www.jssor.com/demos/simple-layer-animation.slider>
- <https://www.youtube.com/watch?v=ix4WqOaE4pk>
- <https://stackoverflow.com/questions/45972242/android-tab-fragment-map-not-showing>
- <https://stackoverflow.com/questions/15706438/how-to-wrap-googlemap-fragment-into-a-linearlayout>
- <https://developers.google.com/android/reference/com/google/android/gms/maps/SupportMapFragment>
- <https://developers.google.com/maps/documentation/android-api/map>
- <https://stackoverflow.com/questions/20155051/how-to-set-visibility-options-of-android-widget-textview>
- https://adambard.com/blog/3-wrong-ways-to-store-a-password/_cifrar contraseñas

<https://www.mkyong.com/java/javamail-api-sending-email-via-gmail-smtp-example/>

https://www.tutorialspoint.com/java/java_sending_email.htm

<https://mvnrepository.com/artifact/javax.mail/javax.mail-api>

<https://stackoverflow.com/questions/13397933/android-spinner-avoid-onitemselected-calls-during-initialization/25070707>

https://developer.android.com/reference/android/widget/TextView.html#attr_android:textStyle

https://material.io/icons/#ic_keyboard_arrow_up

https://stackoverflow.com/questions/38192977/how-do-i-create-drop-down-content-like-on-wikipedia?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

<https://gist.github.com/iperdomo/2867928>

<https://stackoverflow.com/questions/15185636/align-the-top-of-a-view-to-the-bottom-of-another-view-in-a-relativelayout>

<http://www.vogella.com/tutorials/REST/article.html>

<https://javatutorial.net/java-file-upload-rest-service>

<http://www.coderzheaven.com/2012/03/29/uploading-audio-video-or-image-files-from-android-to-server/>

<https://www.coderefer.com/android-upload-file-to-server/>

https://stackoverflow.com/questions/25398200/uploading-file-in-php-server-from-android-device?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

https://www.codicode.com/art/upload_files_from_android_to_a_w.aspx

<https://www.aeq-web.com/android-image-upload-http-post-php/?lang=en>

<https://acomputerengineer.wordpress.com/2016/03/29/upload-image-from-android-app-to-php-server-without-any-library/>

<https://www.androidpit.com/forum/626715/downloading-image-from-server-to-android-device>

https://github.com/codepath/android_guides/wiki/Displaying-Images-with-the-Glide-Library

https://github.com/codepath/android_guides/wiki/Displaying-Images-with-the-Picasso-Library

https://stackoverflow.com/questions/20823249/resize-image-to-full-width-and-fixed-height-with-picasso?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa