APP ANDROID "REPARAWEB"

Aplicación android sobre la red social de talleres de coches, reparaweb

CÓDIGO P.F.C.: DSIC-151

UNIVERSIDAD POLITÉCNICA DE VALENCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

MEMORIA DEL PROYECTO FINAL DE CARRERA

AUTOR: Victor Torrijos Alonso

DIRECTOR: Ismael Torres Boigues

VALENCIA, 12 DE SEPTIEMBRE 2013

CONTENIDO

1.- Resumen

Este documento describe el trabajo realizado en el proyecto:

"APLICACIÓN ANDROID SOBRE LA RED SOCIAL DE TALLERES DE COCHES. REPARAWEB".

El objetivo principal de este proyecto es plasmar una idea, en este caso una aplicación para los dispositivos móviles Android, de la red social de talleres de coches, reparaweb. Haciendo uso de los conocimientos adquiridos durante la carrera, así como cómo aprender sobre el uso y manejo de las tecnologías/aplicaciones necesarias para llevar a cabo el proyecto.

Con la aparición de los teléfonos inteligentes en relativamente poco tiempo, las aplicaciones en los "smartphones" han pasado de ser un fenómeno apenas conocido, a convertirse en un elemento importante en la vida de muchas personas.

Las aplicaciones para móviles han revolucionado la forma en la que vivimos, en la que trabajamos, la forma de hacer negocios, de comunicarnos, de vivir el ocio y ha supuesto un cambio en la forma en la que los alumnos pueden aprender.

Cuando mencionamos el término "apps" nos referimos a una serie de aplicaciones para teléfonos inteligentes que proporcionan servicios interactivos en red dando al usuario la posibilidad de compartirlos con muchísimos otros usuarios.

El desarrollo de aplicaciones para dispositivos móviles requiere tener en cuenta las limitaciones de estos dispositivos. Los dispositivos móviles funcionan con batería y tienen procesadores menos poderosos que los ordenadores personales.

Los desarrollos de estas aplicaciones también tienen que considerar una gran variedad de tamaños de pantalla, datos específicos de software y configuraciones.

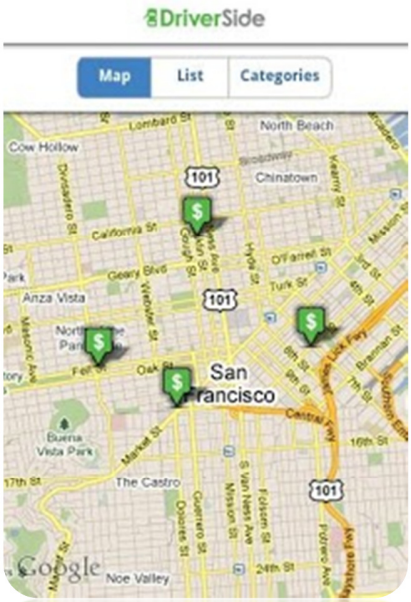
Las aplicaciones móviles suelen ser probadas primero usando emuladores y más tarde se ponen en el mercado en periodo de prueba. Actualmente un gran número de empresas se dedica a la creación profesional de aplicaciones. Aun así, han surgido páginas web como Mobincube donde un usuario común puede crear aplicaciones de manera gratuita y sin conocimiento de programación.

2.- Referencias

En mi búsqueda de aplicaciones de referencia que cumpliera con las funcionalidades que buscaba, si bien es cierto hay web/foros dedicados al automóvil, véase el conocidísimo forocoches.com, páginas para compartir/alquilar vehículos o páginas de rutas, pero apenas hay redes sociales dedicados a la reparación y valoración por parte de usuarios.

Ahora expondré brevemente 2 páginas que puedan haber servido de referencia en el diseño de la aplicación de este proyecto.

2.1.- Driverside



Driverside es un portal web dedicado a nuestro vehículo donde conseguir información de talleres y reparaciones, apartado para tasar el precio de nuestro coche, piezas y accesorios, análisis y reportajes de coches, todo esto según el año, marca y modelo que seleccionemos. La ventaja de este servicio es poder conseguir un presupuesto para determinadas reparaciones sin coste alguno o tasar nuestro modelo actual, haciendo un baremo de otras ventas de dicho modelo en fúnción de estado y km.

¿Cuál es la pega? Que la aplicación es íntegramente en inglés y solo disponible para residentes en USA, puesto que nos pide a parte del año, marca y modelo el código zip (equivalente al cp), en el cual se basa para buscar talleres próximos, siendo válido únicamente en dicho pais.

2.2.- Foodspotting

He tomado como referencia foodspotting.com que aunque su temática no tenga nada que ver con el ámbito que ocupa mi proyecto, trata sobre platos de cocina, tiene un diseño simple y sencillo de usar, justo lo que se busca para una aplicación móvil. Por lo que aprovecharemos la visión general de un mapa para localizar los talleres, y la sencillez para la distribución del sitio web.

3.- Especificación

3.1.- Introducción

En el siguiente apartado vamos a describir el funcionamiento y la implementación de una red social sobre talleres y mecánica en general sobre una aplicación android.

3.2.- Descripción general del sistema

El proyecto a desarrollar consistirá en una aplicación móvil en Android, que conecte con la redsocial "reparaweb" que es un sistema de información de reparaciones y talleres donde han sido efectuadas. Para ello, la aplicación deberá acceder a la información sobre talleres, reparaciones y las reparaciones/mantenimientos en dichos talleres. Este sistema permite la recomendación y la votación online de talleres en distintas ciudades, aprovechando información sobre preferencias, gustos y valoraciones de tus amistades y usuarios con características de vehículos parecidas.

Al sistema se podrán conectar tanto usuarios anónimos como registrados. Los anónimos sólo podrán visualizar información general sobre talleres y puntuaciones de los talleres, pero no tendrán acceso a definir relaciones de amistad ni a comentar reparaciones. Por su parte, los registrados podrán comentar y valorar las reparaciones y/o mantenimientos de los vehículos en los talleres, así como visualizar cualquier comentario y las valoraciones de las relaciones de amistad u otros usuarios, también podrán gestionar relaciones de amistad con otros usuarios registrados, así como dar de alta talleres nuevos, vehículos oreparaciones.

El sistema debe ofrecer la posibilidad de recomendar talleres a los usuarios registrados.

3.3.- Requisitos de gestión de información

El sistema deberá proporcionar el acceso a toda la información talleres y reparaciones, con el objetivo de realizar valoraciones o ranking de talleres desde la misma aplicación móvil sin pasar por la web. En concreto, se podrán valorar reparaciones de TALLERES, y en base a una media de estas reparaciones se asignará una puntuación al taller correspondiente.

Cada reparación tendrá un precio, descripción de la reparación o mantenimiento, información de la duración aproximada, etc. Además, las reparaciones estarán etiquetadas en TIPOS (ej. Mantenimiento, Cambio de aceite, Cambio de ruedas, etc.), de manera que podrá pertenecer a uno o varios tipos.

El sistema mantendrá información sobre los USUARIOS REGISTRADOS, que contendrá información identificativa, y las valoraciones/comentarios emitidos. De cada usuario registrado el sistema debe mantener el histórico de valoraciones/comentarios, el PERFIL DE PREFERENCIAS o gustos (vehículos, zonas, marcas), y un listado de usuarios AMIGOS, y un listado de PETICIONES DE AMISTAD pendientes de comprobar.

3.4.- Requisitos funcionales

A efectos del caso de estudio, se pedirá que se implemente al menos la siguiente funcionalidad:

- Mapa mundi con la localización de los talleres (presentes en la BD) desde donde acceder a la web/ficha del taller en cuestión.
- Registrar un usuario.
- Petición de amistad.
- Enviar mensaje privado a otro usuario/amigo.
- Añadir comentario a la reparación.
- Valorar (puntuar) reparación.
- Dar de alta un taller.
- Dar de alta una reparación.
- Dar de alta un/varios vehiculo/s del usuario.

3.5.- Requisitos de la interfaz de usuario

Se debe desarrollar una interfaz que permita que el sistema sea usado por:

Anónimos, Usuarios Registrados. Cada uno de estos usuarios tiene unos objetivos diferentes con el sistema, que se describen a continuación.

3.5.1.- Usuarios registrados

Estos usuarios podrán acceder a su perfil de preferencias y consultar en cualquier momento su histórico de comentarios que han realizado o han recibido en la pagina web.

Además, también podrán acceder al listado de sus amistades, pudiendo buscar nuevos amigos o eliminarlas de su listado de amigos.

También podrán acceder al listado de vehiculos, talleres dados de alta por el usuario.

3.5.2.- Usuarios anónimos

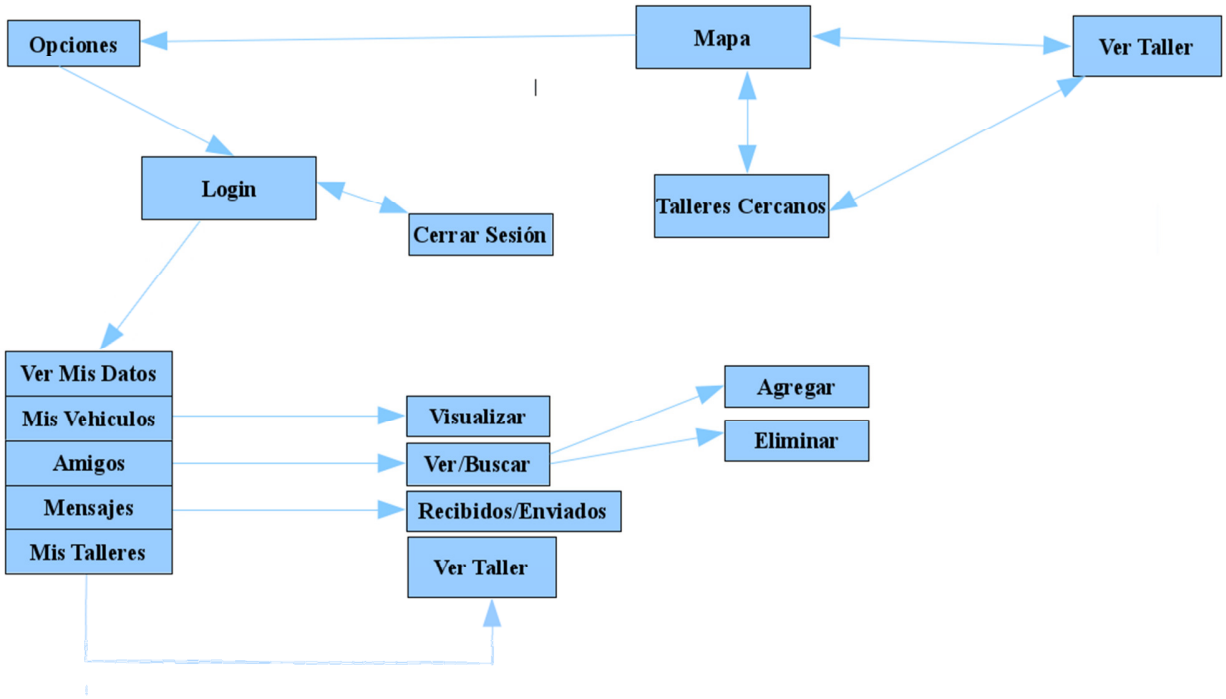
Estos usuarios podrán ver información sobre los talleres y la valoración general del mismo (incluyendo promociones disponibles). No podrán comentar, ni ver los comentarios, ni valorar, ni explorar las amistades existentes, ni acceder a las opciones personales del usuario.

4.- Requisitos

A continuación detallaremos el mapa conceptual de navegación entre las diferentes partes de la aplicación, como también las capturas de las pantallas que se muestran en los diferentes apartados de la aplicación.

4.1.- Mapa de navegación

- Desde la pantalla principal que es el mapa, tenemos acceso desde las dos pestañas superiores a:
    - Mapa, talleres cercanos
- Si le das a la tecla de menú podrás ir a opciones y allí podrás iniciar sesión y acceder al menú privado de usuario. También se podrá acceder desde el icono de opciones que está arriba a la derecha



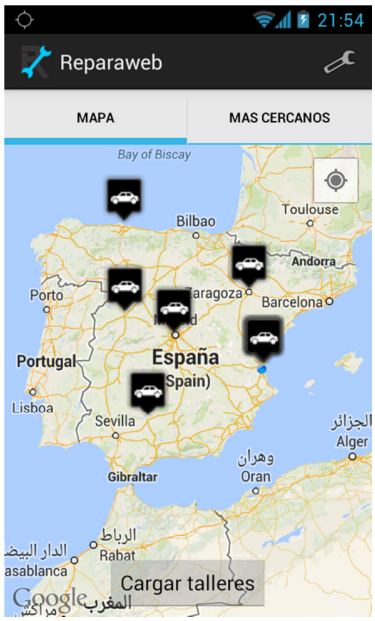
Una vez has iniciado sesión tenemos acceso al menú privado de usuario

4.2.- Mapa

Esta pantalla nos muestra un mapa donde mostramos los talleres introducidos en el sistema. De esta forma posibilitamos la búsqueda geográfica de los talleres que tenemos insertados en la base de datos.

Mockup

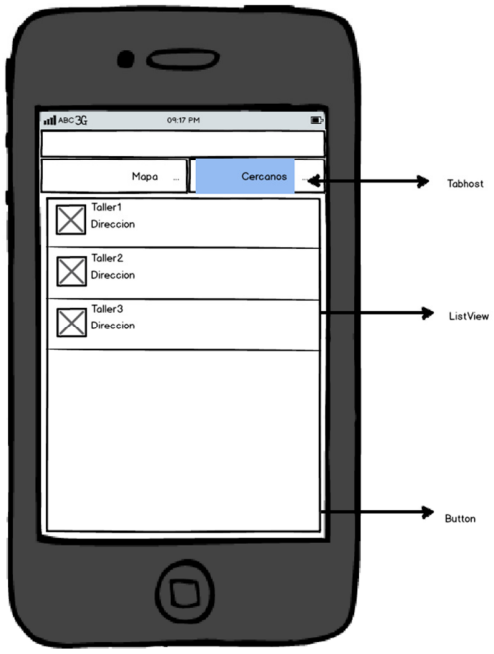
Captura



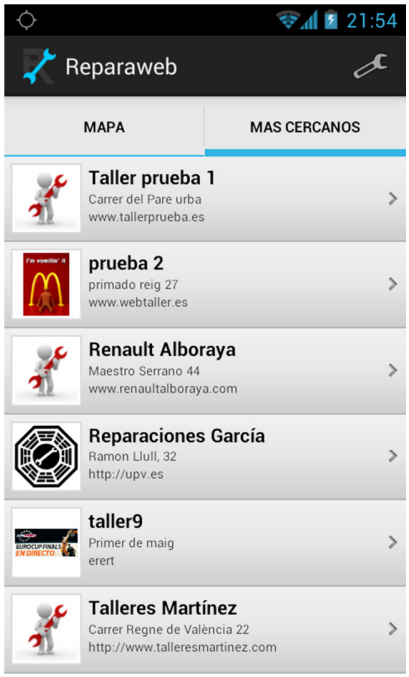
4.3.- Talleres cercanos

En esta pantalla mostramos los talleres más cercanos a nuestra posición gps y para ello el móvil nos calcula la posición que tenemos actualmente (latitud, longitud) y se la envía al servidor, para devolver mediante una consulta sql, un listado xml con los más cercanos.

Mockup



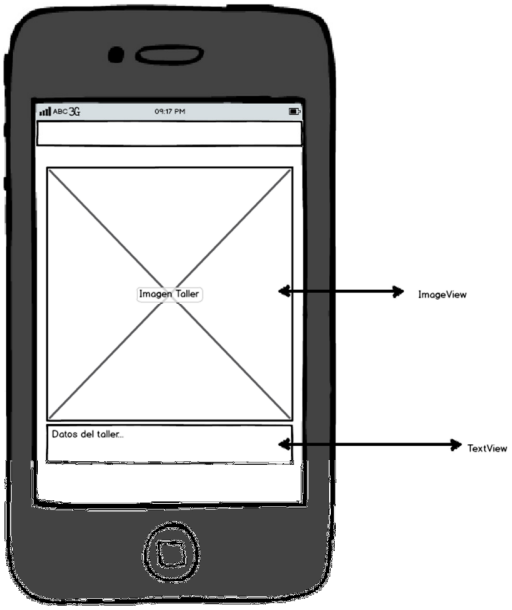
Captura



4.4.- Visualizar taller

Desde esta pantalla veremos los datos de un taller en concreto que hemos pulsado sobre el mapa o sobre alguno de los talleres cercanos.

Mockup

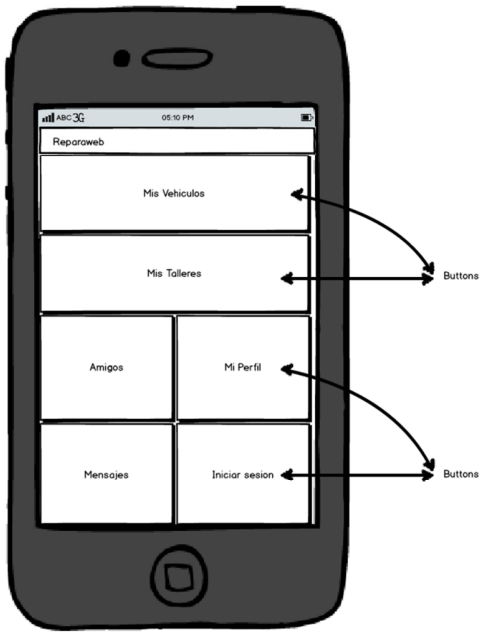


Captura

4.5.- Opciones privadas

Estas opciones sólo serán accesibles cuando un usuario haya iniciado sesión.

Mockup

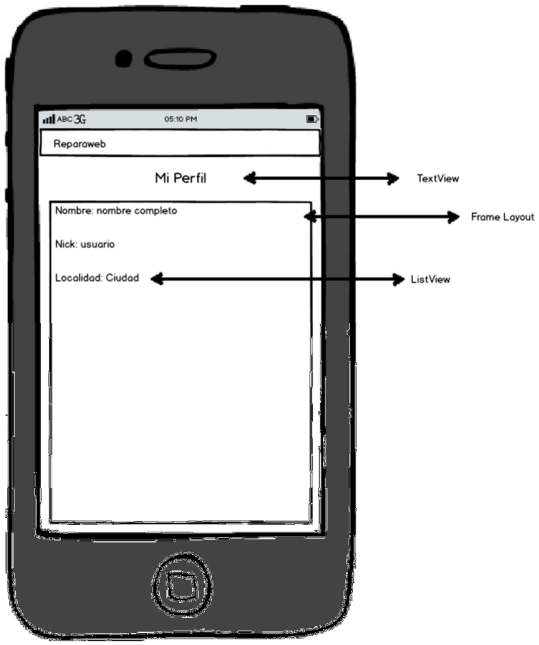


Captura

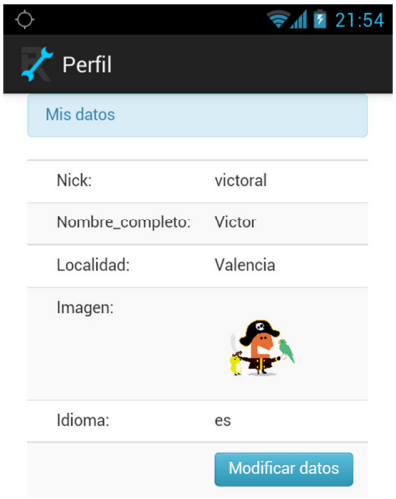
4.6.- Ver mis datos

Esta pantalla muestra a un usuario registrado la información que se está almacenando sobre él. Ofrece un acceso a la pantalla de edición de usuarios.

Mockup



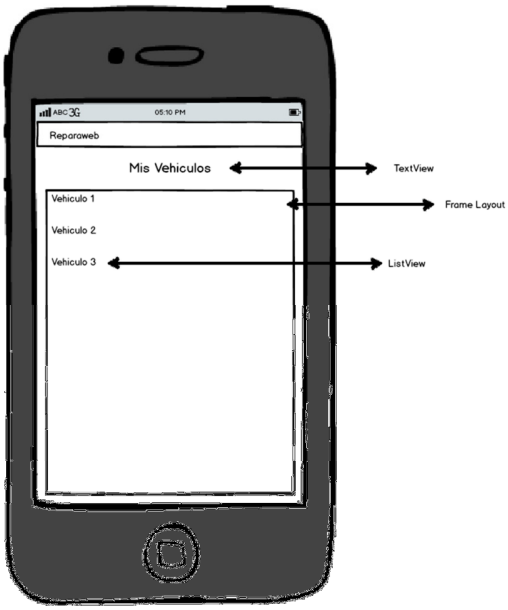
Captura



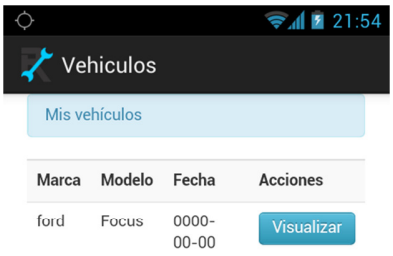
4.7.- Mis vehículos

Esta pantalla ofrece a los usuarios registrados un listado con todos los vehículos introducidos por el. También ofrece la posibilidad de acceder a la pantalla de visualización completa de la información introducida.

Mockup



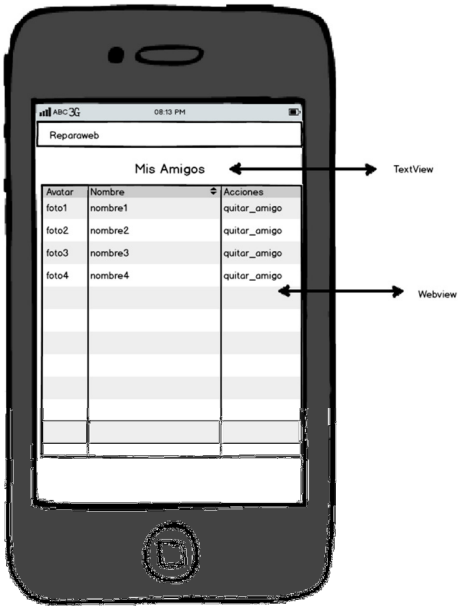
Captura



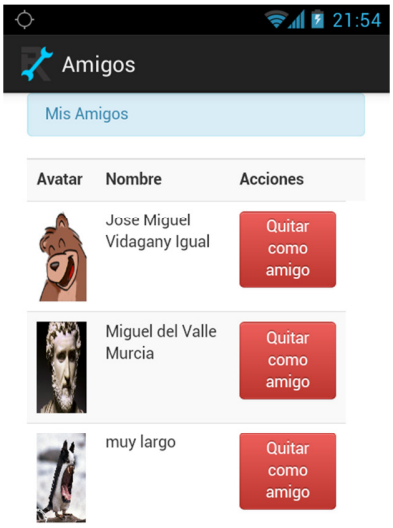
4.8.- Amigos

Desde esta pantalla podremos acceder al listado completo de amigos del usuario. Desde ella daremos acceso a la búsqueda de nuevos amigos, ofreceremos la posibilidad de eliminarlos.

Mockup



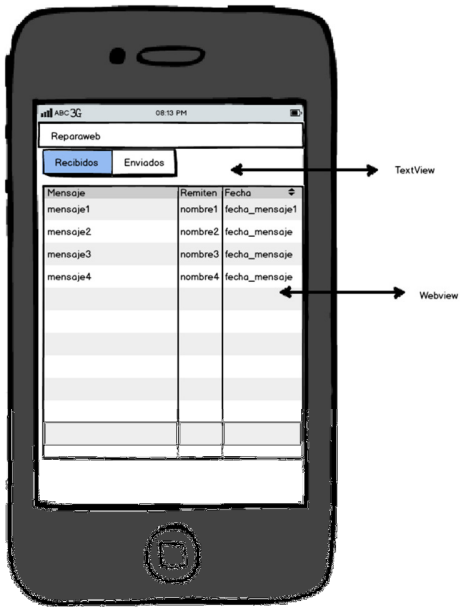
Captura



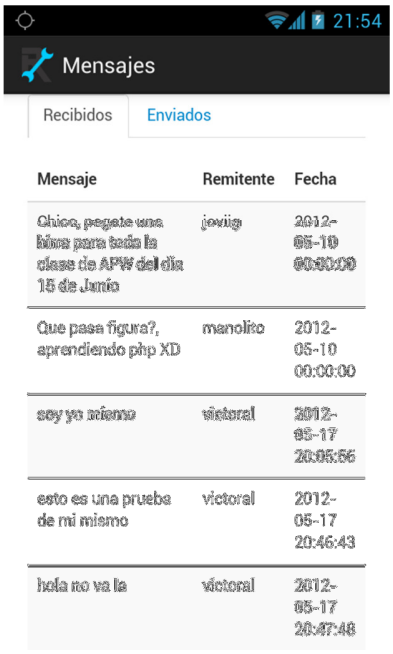
4.9.- Mensajes

Desde esta pantalla podremos acceder al listado de mensajes recibidos. Desde ella podremos visualizar los mensajes recibidos y los que ya hemos enviado a través de la web.

Mockup



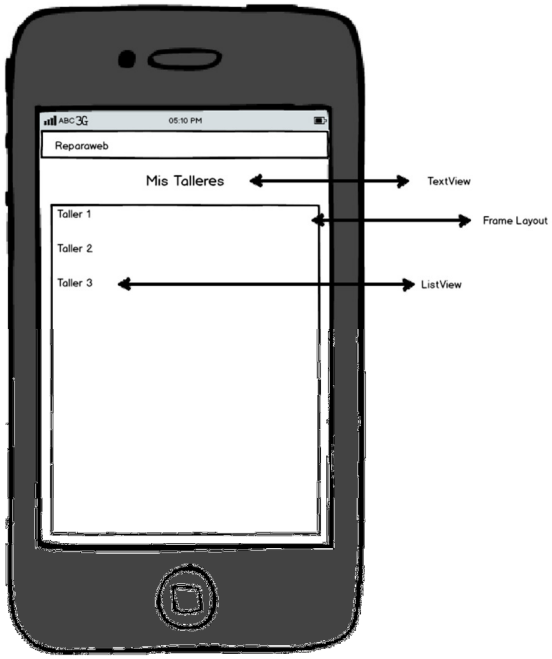
Captura



4.10.- Mis talleres

Desde esta pantalla permitiremos visualizar el listado de talleres introducidos por el usuario, y de aquí acceder a la visualización de alguno de estos talleres.

Mockup



Captura

5.- Implementación

5.1.- Lenguaje de programación

JAVA: Como lenguaje de programación en la aplicación móvil.

Todo el desarrollo de la aplicación se basa en el lenguaje JAVA para Android con su maquina virtual Dalvik.

XML: Como lenguaje para la representación de la parte visual.

El desarrollo de las pantallas que se visualizan en la aplicación, se escribe mediante un lenguaje XML.

PHP: Como lenguaje de programación en el servidor.

Es un lenguaje de programación interpretado o framework para HTML, diseñado originalmente para la creación de aplicaciones para servidores, o creación páginas web dinámicas. PHP es un acrónimo recursivo que significa "PHP Hypertext Preprocessor" (inicialmente PHP Tools, o, Personal Home Page Tools).

5.2.- Estructura de la aplicación

Paso a describir la estructura de los archivos que componen la aplicación para el usuario dando una breve descripción de su funcionamiento.

Archivos de actividades y clases.

- Amigos.java
  - Esta clase es la encargada mostrar la lista de amigos que tiene el usuario, para ello se carga una url php del servidor adaptada al movil, en un componente webview.

- FileCache.java
  - Este archivo se encarga de que la aplicación tenga la funcionalidad de traer desde internet, directamente las imágenes que se necesiten para su visualización en modo temporal, una vez ya no se necesitan visualizar las imágenes o se sale de la aplicación, los archivos temporales de las imágenes desaparecen.

- ImageLoader.java
  - Esta clase es la encargada de cargar las imágenes en el listado de los talleres. Se crea una cola con las imágenes a cargar y se van cargando poco a poco.

- LazyAdapater.java
  - Es el adaptadar encargado de en la vista de talleres cercanos en modo lista, se va encargando de añadir un taller debajo del otro.

- Login.java
  - Esta clase es la encargada proporcionar el inicio de sesión al usuario, para ello se carga una url php del servidor adaptada al movil, en un componente webview.

- MainActivity.java
  - Clase principal de la aplicación. Desde aquí se controlan y gestionan ciertos eventos como el intercambio entre las dos pestañas (Mapa y talleres cercanos) y el ir a las opciones .

- ManejadorXML.java
  - Esta clase nos ayuda cuando el mapa trae los datos desde la pagina web (los trae en formato XML), a tratar el contenido del XML para mostrarlo correctamente en la pantalla.

- Marker.java
  - Esta clase es la utilizada para crear los marcadores en el mapa, que serán los puntos donde hay talleres que estan dados de alta en el sistema.

- MemoryCache.java
  - Es utilizada en conjunción con ImageLoader.java y FileCache.java para obtener los bitmaps en memoria de los archivos temporales de imágenes que se han cargado en el listado.

- Mensajes.java
  - Esta clase es la encargada mostrar la lista de mensajes recibidos y enviados que tiene el usuario, para ello se carga una url php del servidor adaptada al movil, en un componente webview.

- Opciones.java
  - Esta clase es la encargada mostrar la lista de opciones disponibles, que solo se habilitaran una vez el usuario haya podido realizar el inicio de sesión.

- Pefil.java
  - Esta clase es la encargada mostrar los datos del usuario, para ello se carga una url php del servidor adaptada al movil, en un componente webview.

- SplashScreen.java
  - Esta clase es la encargada mostrar la pantalla de carga de la aplicación cuando se está iniciando.

- Talleres.java
  - Esta clase es la encargada mostrar la lista de talleres registrados por el usuario, para ello se carga una url php del servidor adaptada al movil, en un componente webview.

- Utils.java
  - Es utilizada en conjunción con ImageLoader.java, MemoryCache.java y FileCache.java para obtener los bitmaps en memoria de los archivos temporales de imágenes que se han cargado en el listado.

- Vehiculos.java
  - Esta clase es la encargada mostrar la lista de vehiculos del usuario, para ello se carga una url php del servidor adaptada al movil, en un componente webview.

- VistaTaller.java
  - Esta clase es la encargada mostrar la información detallada del taller seleccionado, para ello desde donde ha sido invocada la actividad, se habrá guardado en extras (forma de pasar contenido entre actividades), la información

que vamos a mostrar del taller. Luego la actividad lee los extras y los representa en la actividad, cambiando cada componente del xml con los contenidos que nos han llegado en los "extras".

- XMLParser.java
  - Esta clase nos ayuda a mostrar el contenido del XML que resulta de la petición de visualización de los talleres mas cercanos a tu posición gps, para luego mostrarlos correctamente en la pantalla.

Archivos de vistas y layouts.

- activity_amigos.xml
  - Es el archivo encargado de la visualización de la actividad de amigos una vez se ha pulsado desde opciones, lleva un webview que carga el listado de nuestros amigos.

- activity_login.xml

  Es el archivo encargado de la visualización de nuestros datos del perfil, lleva un webview que carga el formulario de inicio de sesion.

- activity_main.xml
  - Es el archivo encargado de la visualización de la pantalla principal, aquí lleva un tabhost donde se ven las dos pestañas princpiales : Mapa y talleres cercanos.

- activity_mensajes.xml
  - Es el archivo encargado de la visualización de los mensajes recibidos y enviados del usuario, lleva un webview que carga la tabla con los mensajes enviados y recibidos, seleccionado unos u otros a traves de una pestaña.

- activity_opciones.xml
  - Es el archivo encargado de la visualización de la pantalla de opciones, en esta vista se tienen unos botones para la realización de las diferentes opciones personales de cada usuario.

- activity_perfil.xml
  - Es el archivo encargado de la visualización de los datos de perfil del usuario, lleva un webview que carga la tabla con los datos del usuario.

- activity_talleres.xml
  - Es el archivo encargado de la visualización de los talleres dados de alta del usuario, lleva un webview que carga la tabla con los datos de los talleres del usuario.

- activity_vehiculos.xml
  - Es el archivo encargado de la visualización de los vehiculos que tiene el usuario, lleva un webview que carga la tabla con los datos de los vehiculos del usuario.

- list_row.xml
  - Archivo que define la vista en formato listview (listado) que se muestra en la pestaña de los talleres cercanos.

- splash_screen.xml
  - Es el archivo encargado de la visualización de la pantalla de carga o inicio de la aplicación.

- vista_taller.xml
  - Se define la vista de información del taller, que se ha seleccionado en el mapa o en el listado. Lleva un ImageView y varios TextView para la representación de los textos descriptivos del taller.

5.2.- Arquitectura

En este apartado veremos el tipo de arquitectura utilizada en el diseño además de las capas o niveles que aparecen (interfaz de usuario o presentación, lógica de negocio, persistencia o de datos).

La arquitectura es la de cliente-servidor ya que la aplicación movil consulta en todo momento lo que hay en el servidor de la web reparaweb.

Esta arquitectura es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios (servidores) , y los (clientes). Un (cliente) realiza peticiones a otro programa, (servidor), que le da respuesta.

La capacidad de proceso está repartida entre los clientes y los servidores, son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que hace más fácil y claro el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc.

Mientras que sus funcionalidades cambien de unos servicios a otros, la arquitectura básica es la misma. La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

Características Cliente:

1. Es quien solicita las peticiones, tienen un papel activo en la comunicación (dispositivo maestro o amo).
2. Espera y recibe las respuestas del servidor.
3. Normalmente, puede conectarse a varios servidores a la vez.
4. Casi siempre interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

Caracteristicas Servidor:

1. Esperan a que lleguen las solicitudes de los clientes, tienen un papel pasivo en la comunicación (dispositivo esclavo).
2. Tras recibir una solicitud, la procesan y luego envían la respuesta al cliente.
3. Normalmente, aceptan conexiones desde un gran número de clientes (a veces el número máximo de peticiones puede estar limitado).
4. No es frecuente que interactúen directamente con los usuarios finales.

Ventajas

1. Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos.
2. Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).
3. Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, y sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente).
4. Existen tecnologías diseñadas para asegurar la seguridad en las transacciones, la amigabilidad de la interfaz, y la facilidad de uso.

Desventajas

1. La congestión del tráfico ha sido siempre un problema en esta arquitectura. Cuando muchos clientes envían peticiones simultáneas almismo servidor, puede ser que cause muchos problemas para éste (a mayor número de cientes, más problemas para el servidor). Al contrario, en las redes P2P como cada nodo en la red hace también de servidor, cuantos más nodos hay, mejor es el ancho de banda que se tiene.
2. La arquitectura Cliente-Servidor no tiene la robustez de una red P2P. Cuando un servidor está caído, las peticiones de los clientes no pueden ser satisfechas. En la mayor parte de redes P2P, los recursos están generalmente distribuidos en varios nodos de la red. Aunque algunos salgan o abandonen la descarga; otros pueden todavía acabar de descargar consiguiendo datos del resto de los nodos en la red.
3. El software y el hardware de un servidor son generalmente muy determinantes. Un hardware regular de un ordenador personal puede no poder servir a cierta cantidad de clientes. Normalmente se necesita software y hardware específico, sobre todo en el lado del servidor, para satisfacer el trabajo. Por supuesto, esto aumentará el coste.

Algunas redes disponen de tres tipos de nodos:

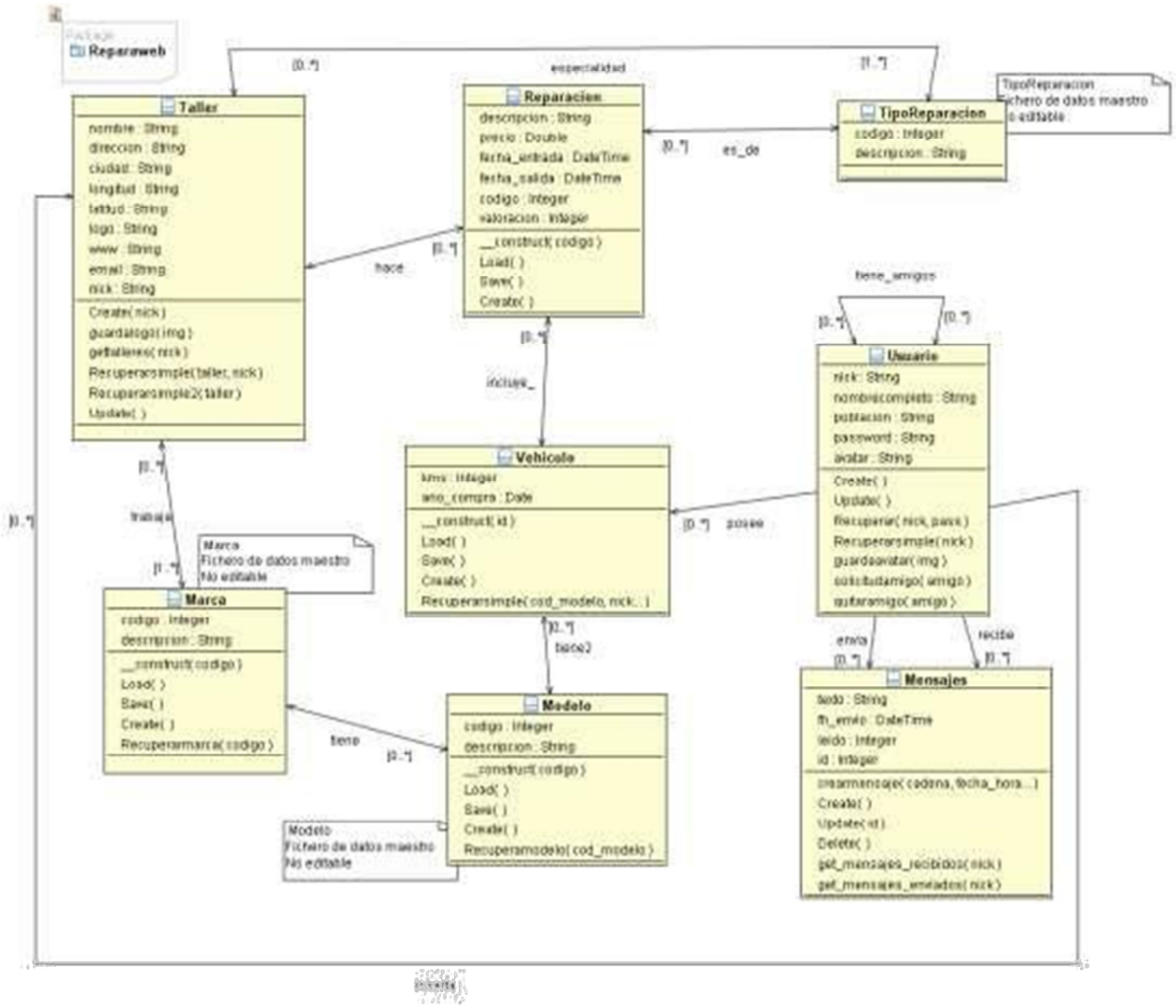
1. Clientes que interactúan con los usuarios finales.
2. Servidores de aplicación que procesan los datos para los clientes.
3. Servidores de la base de datos que almacenan los datos para los servidores de aplicación. Esta configuración se llama una arquitectura de tres-capas(como ya habíamos dicho usamos la separación en capas [presentación,lógica, datos]).

5.3.- Diagrama de clases

Los diagramas de clases son un tipo de diagramas estáticos que describen la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.

Utilizados durante el proceso de diseño de los sistemas informáticos, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

Para la realización de este diagrama de clases se ha utilizado la herramienta MOSKitt.



5.4.- Descripcion de las clases

- Usuario

La clase usuario es la más importante de todas pues todo gira en torno al usuario, que será el que podrá usar toda la funcionalidad de la web en caso de estar registrado.

- Mensajes

La clase mensaje representa a los mensajes que podrá enviar/recibir un usuario registrado de otro usuario.

- Vehículo

La clase vehículo representa a un vehículo del usuario con referencias a la marca y modelo.

- Marca

La clase marca representa la marcas disponibles para los distintos vehículos.

- Modelo

La clase modelo representa los distintos modelos que tiene cada marca en concreto.

- Taller

La clase taller representa a los distintos talleres que se puedan dar de alta en el sistema por un usuario o administrador, estos talleres serán usados para consulta de sus datos o seleccionado cuando se dé el alta de una nueva reparación.

- Reparación

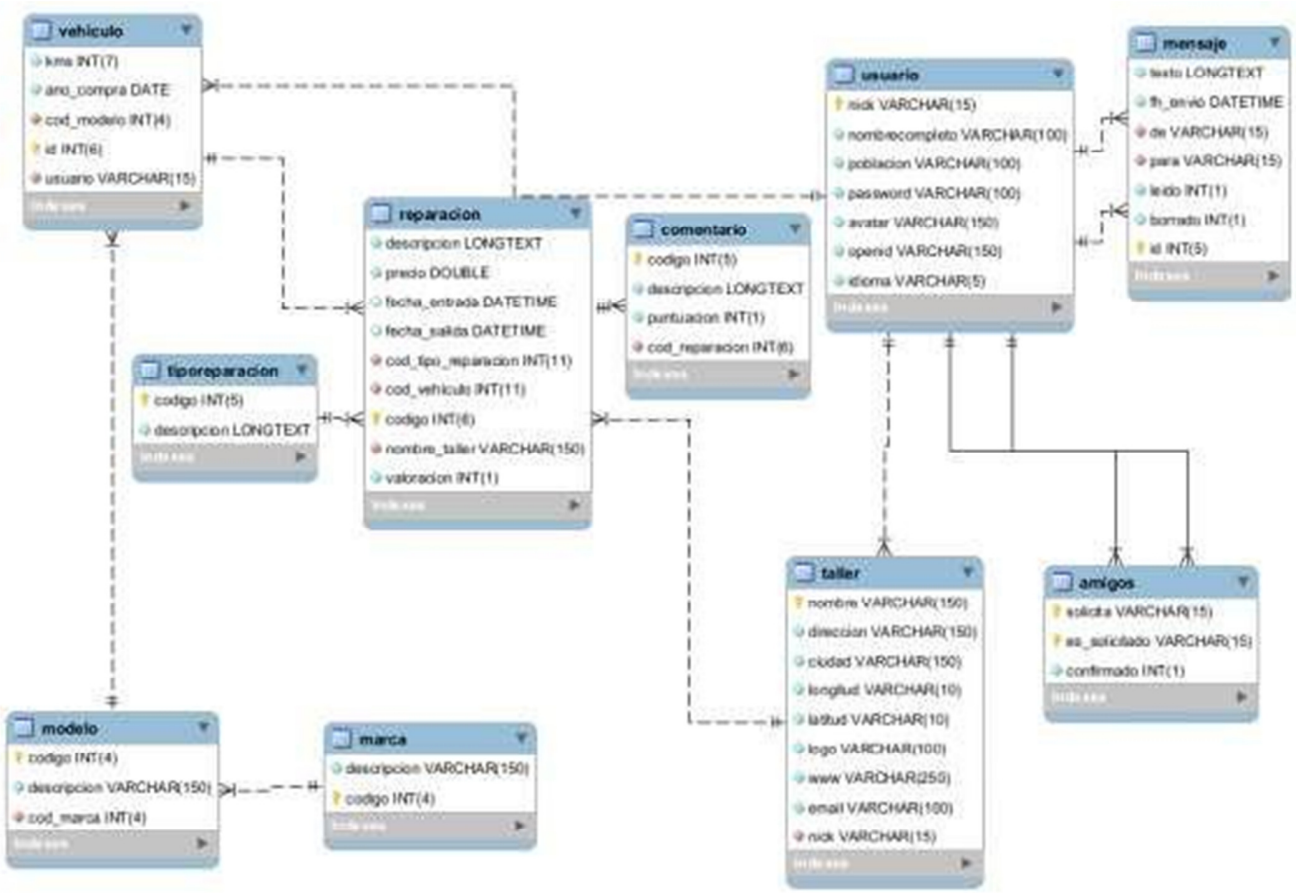
La clase reparación representa a las distintas reparaciones que pueda efectuar un usuario sobre alguno de sus vehículos.

- Tipo Reparación

Esta clase representa el listado de los posibles tipos de reparaciones disponibles cuando un usuario da de alta una reparación para alguno de sus vehículos, esta lista es fija y solo es modificable por un administrador de la web.

5.4.- Diagrama entidad-relación

Para la realización de este diagrama Entidad-relación se ha hecho uso de la herramienta MySQL Workbench.



5.5.- Capa de presentación

Para esta capa la de presentación o interfaz de usuario, se ha usado el el fragment de google maps, para la pestaña del mapa y la pestaña de los talleres cercanos con un listview.

Para la parte de las opciones y datos referentes al usuario, como mensajes, talleres y vehiculos, etc. Se ha realizado con un componente webview que carga el fichero php del servidor y lo representa en el componente.

Todos estos ficheros son implementados en una aplicación android mediante archivos xml para plasmar la representación de las distintas vistas de la aplicación.

Como podría ser de ejemplo la vista de listado de talleres cercanos y la vista de el taller seleccionado.

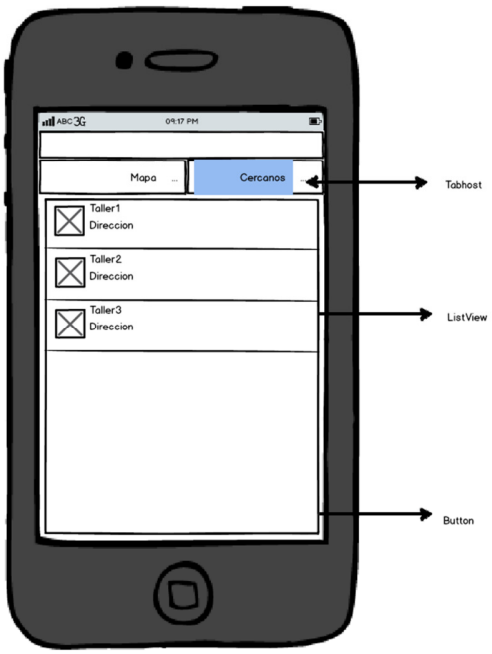
list_row.xml (listado de talleres cercanos)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/list_selector"
    android:orientation="horizontal"
    android:padding="5dip" >
    <LinearLayout
        android:id="@+id/thumbnail"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_marginRight="5dip"
        android:background="@drawable/image_bg"
        android:padding="3dip" >
        <ImageView
            android:id="@+id/list_image"
            android:layout_width="50dip"
            android:layout_height="50dip" />
    </LinearLayout>
    <TextView
        android:id="@+id/nombre"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/thumbnail"
        android:layout_toRightOf="@+id/thumbnail"
        android:text="Nombre taller"
        android:textColor="#040404"
        android:textSize="15dip"
        android:textStyle="bold"
        android:typeface="sans" />
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:src="@drawable/arrow" />
    <TextView
        android:id="@+id/ciudad"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/thumbnail"
        android:layout_toRightOf="@+id/thumbnail"
        android:gravity="right"
        android:text="Ciudad"
        android:textColor="#10bcc9"
        android:textSize="10dip"
        android:textStyle="bold" />
    <TextView
        android:id="@+id/direccion"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/nombre"
        android:layout_toRightOf="@+id/thumbnail"
        android:text="Direccion de taller"
        android:textColor="#343434"
        android:textSize="10dip" />
</RelativeLayout>
```

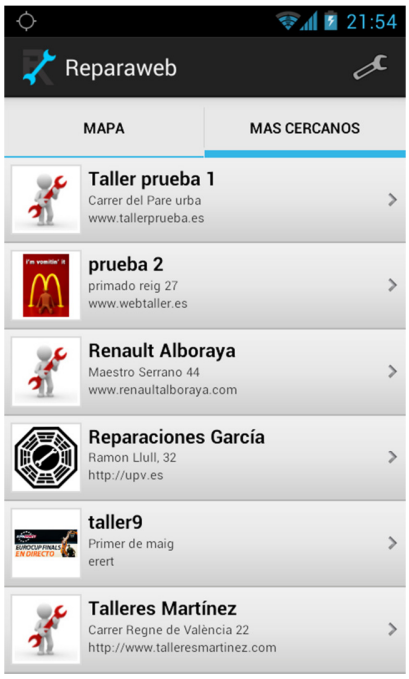
vista_taller.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/LinearLayout1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/gradient_bg"
    android:orientation="vertical"
    android:padding="5dip" >
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/image_bg"
        android:orientation="vertical" >
        <ImageView
            android:id="@+id/list_image"
            android:layout_width="309dp"
            android:layout_height="309dp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/list_selector"
        android:orientation="vertical" >
        <TextView
            android:id="@+id/nombre"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Nombre taller"
            android:textColor="#040404"
            android:textSize="15dip"
            android:textStyle="bold"
            android:typeface="sans" />

        <TextView
            android:id="@+id/direccion"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Direccion de taller"
            android:textColor="#343434"
            android:textSize="10dip" />

        <TextView
            android:id="@+id/ciudad"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="right"
            android:text="Ciudad"
            android:textColor="#10bcc9"
            android:textSize="10dip"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/web"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Web"
            android:textColor="#343434"
            android:textSize="10dip" />

        <TextView
            android:id="@+id/email"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="E-Mail"
            android:textColor="#343434"
            android:textSize="10dip" />
    </LinearLayout>
</LinearLayout>
```

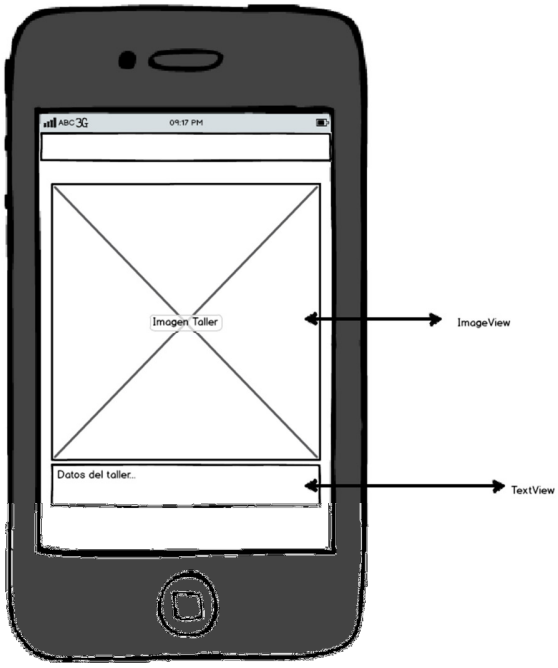
Mockup (list_row.xml)



Pantalla (vista_taller.xml)



Mockup (vista_taller.xml)



Pantalla (vista_taller.xml)

5.5.- Capa logica

El nivel lógico está formado por múltiples archivos en su mayoría *.java que implementan la funcionalidad necesaria para el correcto funcionamiento de la aplicación. Aunque en la parte de opciones tambien sus ficheros son java, llevan incrustado la carga de una url que lleva a un fichero .php adaptado para la correcta visualización en el dispositivo.

Gracias a la independencia entre el nivel de interfaz de usuario y el nivel de persistencia de datos es posible realizar cambios significativos en dicho nivel por ejemplo, cambiar el sistema de gestión de base de datos o incluso sustituir la propia base de datos por otro sistema de almacenamiento sin que esto afectase al nivel de interfaz, sencillamente cambiando la parte de acceso a la base de datos en nuestros ficheros de acceso.

Este nivel contiene distintos tipos de archivos que se diferencian según su propósito:

- Por un lado están los archivos java , que son la implementación de las clases de datos a utilizar, como por ejemplo *XMLParser.java*: Este archivo nos ayuda a mostrar el contenido del XML que resulta de la petición de visualización de los talleres mas cercanos a tu posición gps, para luego mostrarlos correctamente en la pantalla.

- Por otro lador están los archivos java que son la implementación de la funcionalidad de las pantallas al hacer alguna acción, es decir las actividades que se plasman con ficheros xml se les crea la funcionalidad mediante archivos java, como puede ser en este caso la clase principal *MainActivity.java*: Clase principal de la aplicación. Desde aquí se controlan y gestionan ciertos eventos como el intercambio entre las dos pestañas (Mapa y talleres cercanos) y el ir a las opciones.

XMLParser.java

```
package org.example.reparaweb;
import java.io.IOException;
import java.io.StringReader;
import java.io.UnsupportedEncodingException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
import org.w3c.dom.*;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import android.util.Log;

public class XMLParser
{
    public XMLParser()
    {
    }

    public String getXmlFromUrl(String url)
    {
        String xml = null;

        try
        {
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpGet httpGet = new HttpGet(url);
            HttpResponse httpResponse = httpClient.execute(httpGet);
            HttpEntity httpEntity = httpResponse.getEntity();
            xml = EntityUtils.toString(httpEntity);
        }
```

```
        catch (UnsupportedEncodingException e)
        {
            e.printStackTrace();
        }
        catch (ClientProtocolException e)
        {
            e.printStackTrace();
        }

        catch (IOException e)
        {
            e.printStackTrace();
        }
        return xml;
    }

    public Document getDomElement(String xml)
    {
        Document doc = null;
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        try
        {
            DocumentBuilder db = dbf.newDocumentBuilder();
            InputSource is = new InputSource();
            is.setCharacterStream(new StringReader(xml));
            doc = db.parse(is);
        }
        catch (ParserConfigurationException e)
        {
            Log.e("Error: ", e.getMessage());
            return null;
        }
        catch (SAXException e)
        {
            Log.e("Error: ", e.getMessage());
            return null;
        }
        catch (IOException e)
        {
            Log.e("Error: ", e.getMessage());
            return null;
        }
        return doc;
    }

    public final String getElementValue(Node elem)
    {
        Node child;
        if (elem != null)
        {
            if (elem.hasChildNodes())
            {
                for (child = elem.getFirstChild(); child != null; child = child.getNextSibling())
                {
                    if (child.getNodeType() == Node.TEXT_NODE)
                    {
                        return child.getNodeValue();
                    }
                }
            }
        }
        return "";
    }

    public String getValue(Element item, String str)
    {
        NodeList n = item.getElementsByTagName(str);
        return this.getElementValue(n.item(0));
    }
}
```

MainActivity.java

```
package org.example.reparaweb;

public class MainActivity extends FragmentActivity implements OnMapClickListener
{
    private GoogleMap mapa;
    private LatLng lastpos = new LatLng(39.482763, -0.346751);
    static final String URL = "http://apw.vidadel.es/junto/generaxml.php";
    static final String KEY_TALLER = "taller"; // parent node
    static final String KEY_ID = "id";
    static final String KEY_NOMBRE = "nombre";
    static final String KEY_DIRECCION = "direccion";
    static final String KEY_CIUDAD = "ciudad";
    static final String KEY_THUMB_URL = "thumb_url";
    ListView list;
    LazyAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final TabHost tabs = (TabHost) findViewById(android.R.id.tabhost);
        tabs.setup();

        TabHost.TabSpec spec = tabs.newTabSpec("Mapa");
        spec.setContent(R.id.tab1);
        Resources res = getResources();

        spec.setIndicator("Mapa", res.getDrawable(android.R.drawable.ic_menu_mapmode));
        tabs.addTab(spec);

        spec = tabs.newTabSpec("Mas cercanos");
        spec.setContent(R.id.tab2);
        spec.setIndicator("Mas cercanos", res.getDrawable(android.R.drawable.ic_menu_manage));
        tabs.addTab(spec);

        tabs.setCurrentTab(0);

        carga_contenido_tab1();

        tabs.setOnTabChangedListener(new OnTabChangeListener()
        {
            public void onTabChanged(String str)
            {
                if (tabs.getCurrentTab() == 0)
                {
                    carga_contenido_tab1();
                }
                else
                {
                    carga_contenido_tab2();
                }
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }


.
.
.
```

Contenido de la función carga_contenido_tab2

```
    public void carga_contenido_tab2()
    {
        // cosas para la pestaña 2
        ArrayList<HashMap<String, String>> tallerLista = new ArrayList<HashMap<String, String>>();

        String urlconpos;

        if (mapa.getMyLocation() != null)
        {
            urlconpos = URL + "?lat=" + mapa.getMyLocation().getLatitude() + "&long=" +
mapa.getMyLocation().getLongitude();
            XMLParser parser = new XMLParser();
            String xml = parser.getXmlFromUrl(urlconpos); // getting XML from
                                                          // URL
            Document doc = parser.getDomElement(xml); // getting DOM element

            NodeList nl = doc.getElementsByTagName(KEY_TALLER);

            for (int i = 0; i < nl.getLength(); i++)
            {
                // creating new HashMap
                HashMap<String, String> map = new HashMap<String, String>();
                Element e = (Element) nl.item(i);
                // adding each child node to HashMap key => value
                map.put(KEY_ID, parser.getValue(e, KEY_ID));
                map.put(KEY_NOMBRE, parser.getValue(e, KEY_NOMBRE));
                map.put(KEY_DIRECCION, parser.getValue(e, KEY_DIRECCION));
                map.put(KEY_CIUDAD, parser.getValue(e, KEY_CIUDAD));
                map.put(KEY_THUMB_URL, parser.getValue(e, KEY_THUMB_URL));

                // adding HashList to ArrayList
                tallerLista.add(map);
            }

            list = (ListView) findViewById(R.id.list);

            // Getting adapter by passing xml data ArrayList
            adapter = new LazyAdapter(this, tallerLista);
            list.setAdapter(adapter);

            // Click event for single list row
            list.setOnItemClickListener(new OnItemClickListener()
            {

                @Override
                public void onItemClick(AdapterView<?> parent, View view, int position, long id)
                {

                }
            });
        }

    }
```

5.6.- Capa de persistencia de datos

Ficheros de definición de la base de datos y ejemplo de archivo .java con php de acceso a datos del usuario

En esta capa se ha hecho uso de un sistema de gestión de bases de datos relacional (SGDBR) MySQL, administrado vía Web desde phpMyAdmin.

TABLA usuario (nick: VARCHAR(15), nombrecompleto: VARCHAR(100),
poblacion: VARCHAR(100), password: VARCHAR(100), avatar:
VARCHAR(150), openid: VARCHAR(150), idioma: VARCHAR(5))
CP: {nick}
VNN:{nick}

nick: Alias e identificador del usuario.
nombrecompleto: nombre completo del usuario.
poblacion: Poblacion del usuario.
password: Contraseña del usuario.
avatar: Foto del usuario.
openid: Identificador openid del usuario.
idioma: Idioma que el usuario tiene definido en la web.

TABLA amigos (solicita: VARCHAR(15), es_solicitado: VARCHAR(15),
confirmado: INT(1))
CP: {solicita}
CAJ: {solicita, es_solicitado} -> USUARIO
VNN:{solicita, es_solicitado, confirmado}

solicita: nick identificador del usuario que solicita la amistad.
es_solicitado: nick identificador del usuario que es solicitado la amistad.
confirmado: valor o ó 1 para si está confirmada la amistad o no.

TABLA tiporeparacion (codigo: INT(5), descripcion: LONGTEXT)
CP: {codigo}
VNN:{codigo, descripcion}

codigo: Identificador del tipo de reparacion.
descripcion: descripcion del tipo de reparacion.

TABLA marca (codigo: INT(4), descripcion: VARCHAR(150))
CP: {codigo}
VNN:{codigo, descripcion}

codigo: Identificador de la marca.
descripcion: descripcion de la marca.

TABLA modelo (codigo: INT(4), descripcion: VARCHAR(150), cod_marca:
INT(4))
CP: {codigo}
CAJ: {cod_marca} -> MARCA
VNN:{codigo, descripcion, cod_marca}

codigo: Identificador del modelo.
descripcion: descripcion del modelo.
cod_marca: Codigo identificador de la marca del modelo.

TABLA vehiculo (id: INT(6), kms: INT(7), ano_compra: DATE, cod_modelo: INT(4), usuario: VARCHAR(15))
CP: {id}
CAJ: {usuario} -> USUARIO
CAJ: {cod_modelo} -> MODELO
VNN:{id, kms, ano_compra, cod_modelo, usuario}

id: Identificador del vehiculo.
kms: Kilometros del vehiculo.
ano_compra: Año de compra del vehículo.
cod_modelo: Código identificador del modelo del vehiculo.
usuario: nick propietario del vehiculo.

TABLA taller (nombre: VARCHAR(150), direccion: VARCHAR(150), ciudad: VARCHAR(150), longitud: VARCHAR(10), latitud: VARCHAR(10), logo: VARCHAR(100), www: VARCHAR(250), email: VARCHAR(100), nick: VARCHAR(15))
CP: {nombre}
CAJ: {nick} -> USUARIO
VNN:{nombre, direccion, ciudad, longitud, latitud, logo, www, email, nick}

nombre: Identificador del nombre del taller.
direccion: Dirección del taller.
ciudad: Ciudad del taller.
longitud: Coordenada cartográfica del taller.
latitud: Coordenada cartográfica del taller.
logo: Imagen de marca del taller.
www: Dirección web del taller.
email: Correo electrónico del taller.
nick: Identificador de usuario que registro el taller.

TABLA reparacion (descripcion: LONGTEXT, precio: DOUBLE, fecha_entrada: DATETIME, fecha_salida: DATETIME, cod_tipo_reparacion: INT(11), cod_vehiculo: INT (11), codigo: INT(6), nombre_taller: VARCHAR(150), valoracion: INT(1))
CP: {codigo}
CAJ: {cod_tipo_reparacion} -> TIPOREPARACION
CAJ: {cod_vehiculo} -> VEHICULO
CAJ: {nombre_taller} -> TALLER
VNN:{codigo, descripcion, precio, fecha_entrada, fecha_salida, cod_tipo_reparacion, cod_vehiculo, nombre_taller, valoracion}

codigo: Identificador de la reparacion.
descripcion: Descripcion de la reparacion.
precio: Precio de la reparacion.
fecha_entrada: Fecha de entrada del vehiculo al taller.
fecha_salida: Fecha de salida del vehiculo al taller.
cod_tipo_reparacion: Codigo del tipo de reparación.
cod_vehiculo: Codigo del vehiculo reparado.
nombre_taller: Nombre del taller donde se ha hecho esta reparacion.
valoracion: Numero entre 0 y 9 valorando la reparacion.

TABLA comentario (codigo: INT(5), descripcion: LONGTEXT, puntuacion: INT(1), cod_reparacion: INT(6))
CP: {codigo}
CAJ: {cod_reparacion} -> REPARACION
VNN: {codigo, descripcion, puntuacion, cod_reparacion}

codigo: Identificador del comentario.
descripcion: Texto del comentario en cuestion.
puntuacion: Valor entre 0 y 9 del comentario.
cod_reparacion: Identificador de la reparacion comentada.

TABLA mensaje (texto: LONGTEXT, fh_envio: DATETIME, de: VARCHAR(15), para: VARCHAR(15), leido: INT(1), borrado: INT(1), id: INT(5))
CP: {id}
CAJ: {de} -> USUARIO
CAJ: {para} -> USUARIO
VNN: {texto, fh_envio, de, para, leido, borrado, id}

id: Identificador del mensaje.
texto: Texto del mensaje.
fh_envio: Fecha en la que se ha enviado el mensaje.
de: nick del usuario que ha enviado el mensaje
para: nick del usuario que recibe el mensaje.
leido: Valor 0 o 1, por si el usuario destinatario del mensaje, lo ha leido o no.
borrado: Valor 0 o 1, por si el usuario destinatario del mensaje, lo ha borrado o no.

Contenido del fichero que accede a los datos del usuario:

Perfil.java

```
package org.example.reparaweb;

import android.os.Bundle;
import android.app.Activity;
import android.app.ProgressDialog;
import android.view.Menu;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class Perfil extends Activity
{

    WebView navegador;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        navegador = (WebView) findViewById(R.id.webkit);
        navegador.loadUrl("http://apw.vidadel.es/junto/indexm.php?page=ver_misdatos");
        navegador.getSettings().setJavaScriptEnabled(true);
        navegador.getSettings().setBuiltInZoomControls(false);
        final ProgressDialog pd = ProgressDialog.show(this, "","Obteniendo datos...", true);
        navegador.setWebViewClient(new WebViewClient()
        {
            @Override
            public boolean shouldOverrideUrlLoading(WebView view, String url)
            {
                pd.show();
                view.loadUrl(url);
                return true;
            }

            @Override
            public void onPageFinished(WebView view, final String url) {
                pd.dismiss();
            }
        });

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        return true;
    }

    @Override
   public void onBackPressed()
    {
        finish();
        overridePendingTransition(R.anim.entrada_derecha, R.anim.salida_izquierda);
    }

}
```

ver_misdatos.php

```
<?  if(isset($_SESSION["usuario"])){?>

<div class="alert alert-info">
    Mis datos
</div>

<?  if ($error) {?>
        <div class="alert alert-warning"><? echo $tipoerror[$error]; ?></div>
        <? } ?>
                <?
                $usuario=new usuario(1);
                $usuario->Recuperarsimple($_SESSION['usuario']);
                echo "<table class='table table-striped' style='width:100%;'>
                    <tr></tr>";

                    echo "<tr><td>";
                    echo "</td><td>Nick :     </td><td>" . $_SESSION['usuario'] . "</td>";

                    echo "</tr><td>";
                    echo "</td><td>Nombre completo :      </td><td>" . $_SESSION['nombrecompleto'] .
"</td>";

                    echo "<tr><td>";
                    echo "</td><td>Localidad :  </td><td>"  .$_SESSION['poblacion']. "</td>";

                    echo "<tr><td>";
                    echo "</td><td>Imagen :  </td><td> <img class='float:left' src='".
URL_AVATARES.$_SESSION['avatar'] . "' width='75' height='75' alt='usuario' /></td>";

                    echo "<tr><td>";
                    echo "</td><td>Idioma :  </td><td>"  .$usuario->getidioma(). "</td>";

                    echo "<tr><td>";

                echo "</table>";
                ?>

<?
}else{ ?>
<div class="alert alert-warning"> <? echo $tipoerror[2]; ?> </div>
<? } ?>
```

De la misma forma el resto de clases tiene su correspondiente archivo.java con: constructor, update, funciones para recuperar el objeto, GETS y SETS, y las funciones de consulta y modificación de los datos de su clase pertinentes.

6.- Modelo de negocio

Aquí se detallarán de forma breve distintos modelos de negocio o de rentabilizar nuestra web.

- Una opción a tener en cuenta es la publicidad de Google (admob). Lo cual nos generara ingresos cuando los usuarios visiten o simplemente se muestre dicha publicidad en sus móviles.
- Se puede ofrecer dos modos de aplicación una con publicidad y gratuita y otra aplicación de pago sin publicidad y tener opciones más detalladas y opciones adicionales como promoción de ofertas de dicho taller.
- Siempre está la opción de habilitar un servicio de donaciones, para que los usuarios que lo deseen hagan donaciones como muestra de agradecimiento por nuestro servicio ofrecido.

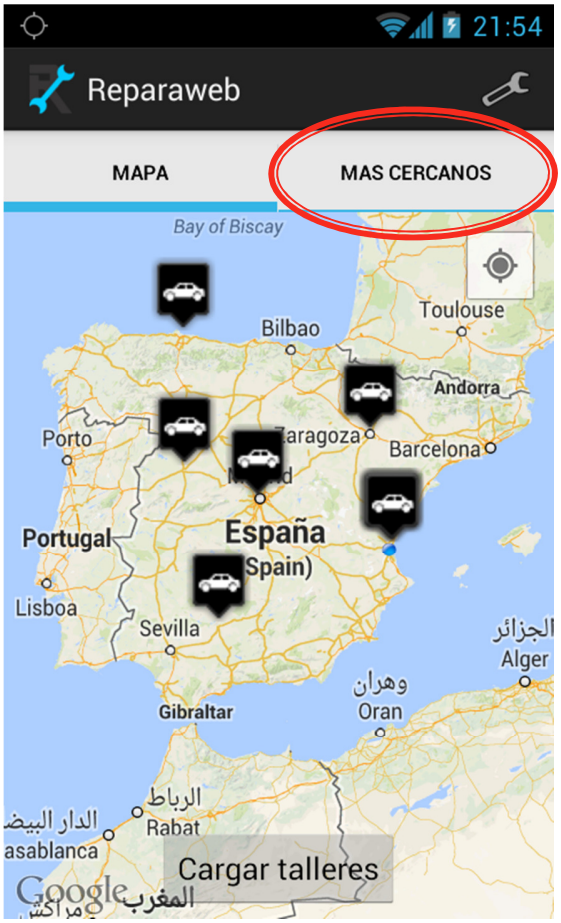
7.- Manual de usuario

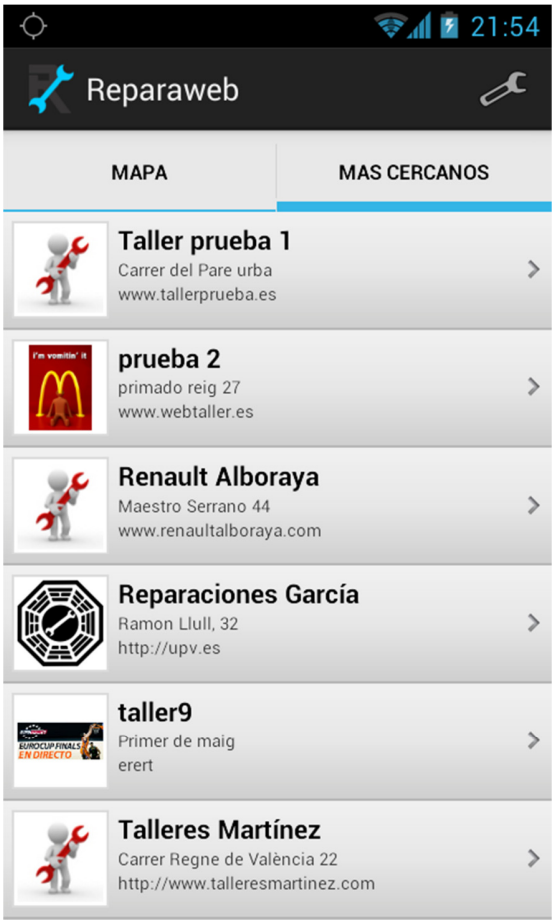
El manual de usuario es un documento que tiene como objetivo dar apoyo al usuario final sobre las características principales que poseen cada pantalla de la aplicación móvil de la "Red social de reparaciones de coches - Reparaweb" entre las pantallas tenemos.

Cuando la aplicación arranca, aparece un mapa y se activa la localización gps y el programa te situará en el mapa a lo largo del mundo mediante el punto azul. Si se le da al botón, el mapa centrará tu, es decir, tu posición será el centro de la vista actual en el mapa.

Luego tenemos la opcion de pulsar el botón, Cargar talleres desde este botón haremos que se muestre en pantalla los talleres que estan a nuestro alrededor, no los que realmente existen, sino de los que existen solamente los que están dados de alta en la red social.

Tambien podemos obtener un listado de los talleres mas cercanos a nuestra posicion, para ello haremos uso de la pestaña **MAS CERCANOS**



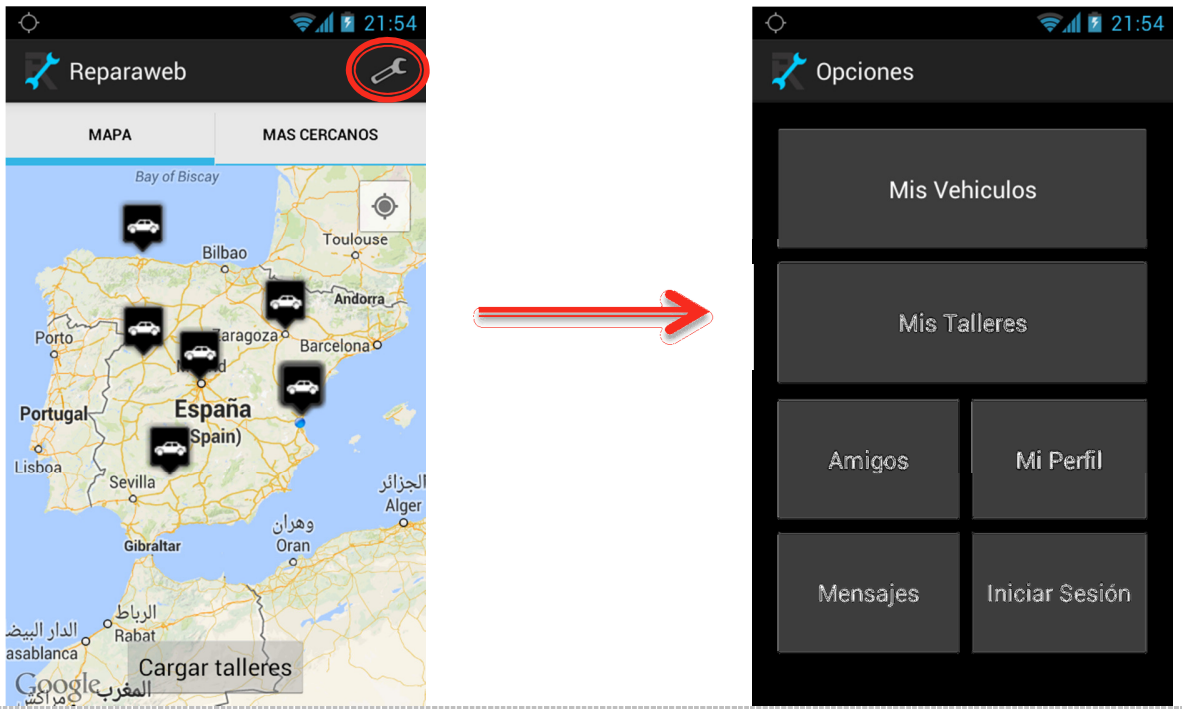


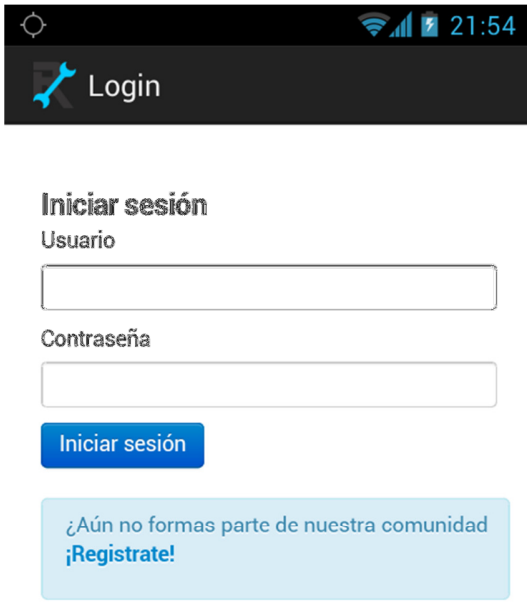
En la pestaña mas cercanos podremos ver una lista con: imagen del taller, si es que la hubiera, sino es la imagen por defecto, nombre, dirección y pagina web del taller. Con los diez talleres mas cercanos a tu posición.

Si en cualquiera de los talleres seleccionamos un taller pulsando sobre toda la fila del taller que nos interesa, veríamos en una pantalla individual, toda la información extendida del taller y la restante que no se muestra en la lista. Como vemos en la imagen de a continuación.

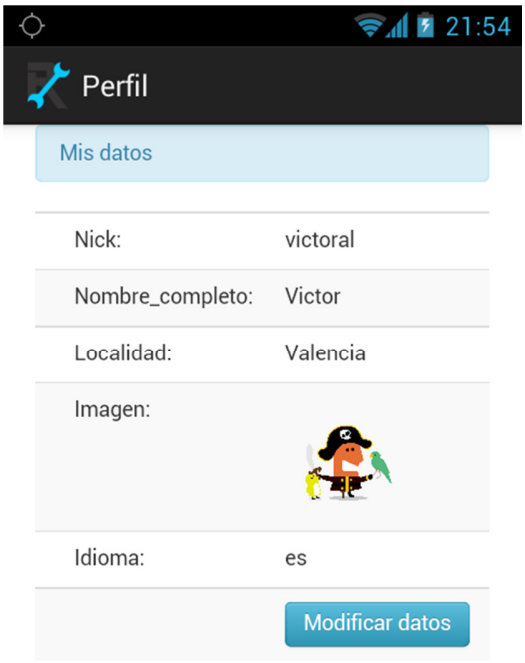
Si en cualquiera de las dos pestañas, mapa o mas cercanos, pulsamos sobre el boton de opciones podremos ir a la pantalla de opciones personales

.

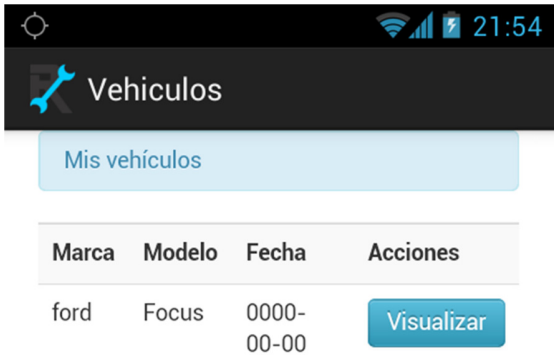




Desde la pantalla de opciones podremos ir a las diferentes opciones, pero estas opciones no estaran disponibles a no ser que se pase primero por la opción de iniciar sesión y así utilizar el usuario y contraseña que se tiene en la red social



Una vez ya con la sesión iniciada, podremos ir a las diferentes opciones. En este caso vamos a la opcion de perfil y podremos ver nuestro perfil con nuestros datos.
Incluso si pulsaramos sobre el botón de Modificar datos, podriamos modificar nuestros datos.

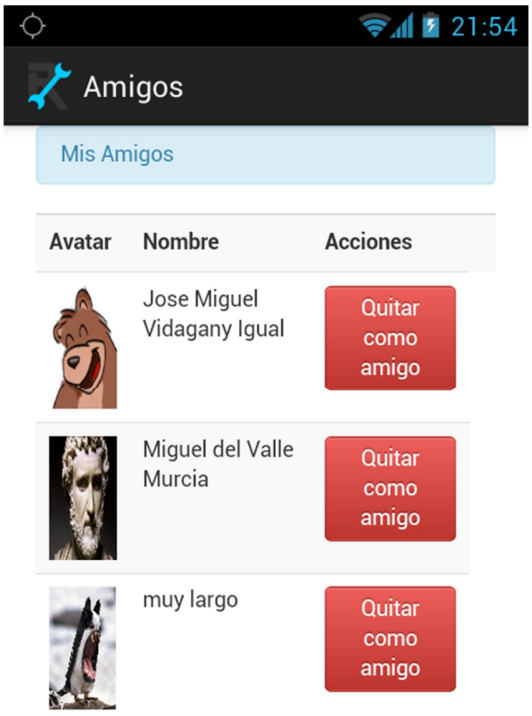


Una vez ya con la sesión iniciada, podremos ir en este caso a la opcion de mis vehiculos y podremos ver nuestros vehiculos, que tenemos registrados o dados de alta en nuestro perfil de la red social.

La pantalla muestra un resumen de cada vehiculo y si le damos a visualizar veremos todos los detalles del vehículo.

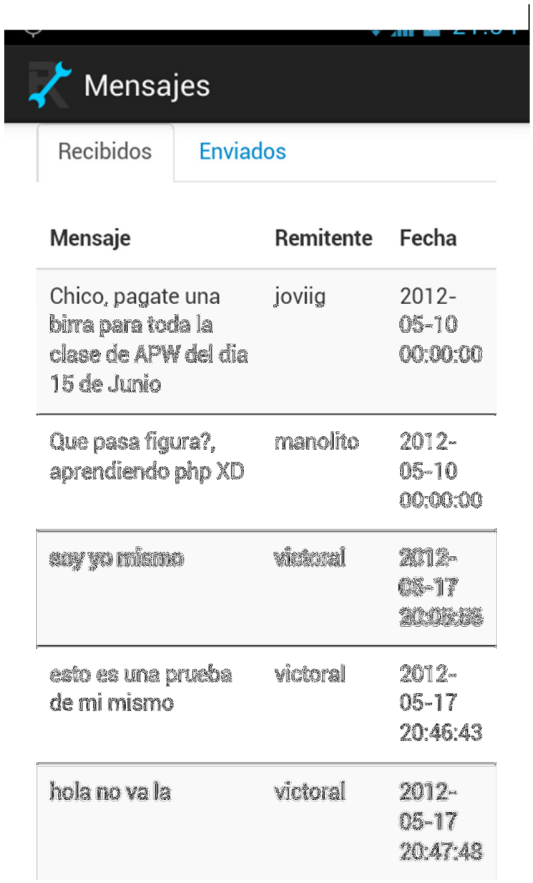
También podremos ir en este caso a la opcion de mis talleres, que tenemos registrados o dados de alta en nuestro perfil de la red social.

La pantalla muestra la foto y el nombre del taller, pero podemos darle a visualizar y así ver todos los detalles del taller.



En este caso vamos a la opcion de amigos, aquí pode ver nuestros amigos.

Tambien podemos quitar amigos, si pulsamos sobre el botón de Quitar como amigo, como tambien podriamos añadir mas amigos a nuestra cuenta, pulsando sobre el botón buscar y o buscar por un nombre de usuario o bien sin nombre para que te muestre todos los usuarios. En esa lista podremos añadirlo como amigo.



En la última opción de Mensajes, podremos ver todos los mensajes que hemos recibido y enviado en la red social reparaweb.

En la pestaña de Recibidos, vemos el mensaje, quien lo envia y la fecha/hora que se ha recibido.

En la pestaña de Enviados, vemos el mensaje, el destinatario y la fecha/hora que se ha enviado

8.- Conclusiones

8.1.- Fases del proyecto

El proyecto se ha realizado en las siguientes fases:

1. Búsqueda de Webs y aplicaciones similares.
2. Determinar la funcionalidad básica de nuestra aplicación.
3. Especificación de requisitos.
4. Diseño de diagramas
5. Implantación del diseño
6. Codificación
7. Pruebas
8. Corrección de errores
9. Pruebas finales
10. Corrección final de errores.

8.2.- Dificultades

La dificultad para llevar a cabo el proyecto ha sido grande, ya que era mi primera aplicación para móviles y en concreto para la plataforma Android, era la primera vez que usaba las tecnologías como la programación en java con php y sus diseños de vistas en xml, y también las herramientas empleadas como el Balsamiq, ADT de Google, he tenido que realizar un gran esfuerzo debido a la cantidad de programación y su complejidad, y sobre todo para la realización de la parte gráfica de la aplicación ya que se hace mediante archivos xml y no es para nada sencillo la forma en diseñar las interfaces. Nada que ver con lo que puede ser un Visual Studio o los RAD de Borland. Lo que ha requerido gran interés y ambición por mi parte para adquirir los conocimientos necesarios y ponerlos en práctica.

8.3.- Experiencia

He aprendido bastante, en gran parte, por las dificultades afrontadas. El desarrollo de esta aplicación me ha hecho conocer en profundidad un tipo de herramientas y lenguajes que antes no había manejado tan ampliamente y que son de un uso extendido, lo que me puede ayudar en un futuro a desarrollar proyectos y aplicaciones en distintos ámbitos.

De tener que destacar algo que me haya cautivado durante el desarrollo del proyecto sin duda han sido el manejo de la api de google, y ver como algo tan simple como un mapa puede ofrecer tantas posibilidades con la adecuada implementación y librerías usadas.

8.4.- Trabajo futuro

A pesar de la gran cantidad de tiempo dedicado y las funcionalidades implementadas, he pensado una serie de funcionalidades más que podría tener la aplicación, que ya serian parte de otro proyecto a desarrollar.

Ideas para posibles ampliaciones futuras de la funcionalidad:

- Se podría mejorar la base de datos, por ejemplo, agregando una nueva tabla de ofertas sobre algún tipo de reparación de un determinado taller.
- Implementar algún método de localización ip (para ofrecer al usuario ofertas personalizadas de su zona de residencia).
- Para la version gratuita y con publicidad el limite de resultados de talleres cercanos es de 10. En la versión de pago se podría aumentar o decrementar ese valor, según una caracteristica que se podría guardar en opciones.
- Acceso desde la aplicación móvil a un mini foro de la web, para cada modelo de coche o tipo de reparación, para que los usuarios comentaran acerca de ellos o buscaran información concreta.
- Agregar la opción a los usuarios de comentar una reparación de otro usuario, y no solo una valoración de la misma.
- Agregar la opción a los usuarios de valorar una reparación de otro usuario.
- Agregar redes sociales y la integración para poder postear en ella directamente comentarios o valoraciones de nuestra web.
- Implementar un buscador de talleres/reparaciones donde muestre las fichas de los talleres/reparaciones solicitados.

9.- Bibliografía

Android Developers http://developer.android.com

Manual php. http://php.net/manual/es/index.php

Stackoverflow http://stackoverflow.com

Curso Android Avanzado http://www.cfp.upv.es/formacion-permanente/cursos/android-online--programacion-avanzada_idiomaes-menuupvtrue-cid31990.html

Web 2.0 http://es.wikipedia.org/wiki/Web_2.0

PFC Aproximación Etica y legal a las redes sociales. http://riunet.upv.es/bitstream/handle/10251/9123/PFC%20-%20Aproximaci%C3%B3n%20%C3%A9tica%20y%20Legal%20a%20las%20Redes%20Sociales.pdf

Driverside http://www.driverside.com

Foodspotting http://www.foodspotting.com

Google maps, api v3 https://developers.google.com/maps/articles/phpsqlajax?hl=es

Infobubble http://google-maps-utility-library-v3.googlecode.com/svn/trunk/infobubble/examples/example.html

Cliente servidor. http://es.wikipedia.org/wiki/Cliente-servidor