

I.E.S. Las Salinas

VEHIMOVÍ

Desarrollo de Aplicaciones
Multiplataforma

Manuel Salvador García Mora

20



Índice

Tabla de contenido

1.	Abstract.....	2
2.	Justificación.....	3
3.	Objetivos	4
4.	Desarrollo.....	5
	4.1 Acerca de la organización	5
	4.2 Análisis de requerimientos	6
	4.3 Hardware	8
	4.4 Sistemas operativos y aplicaciones	8
	4.5 Bases de datos	9
	4.6 Diseño de la aplicación	11
	4.7 Programación	13
	4.8 Desarrollo de interfaces	25
5.	Bibliografía	26

El documento solo cuenta con 27 páginas ya que se trata de un proyecto de 25 horas y rellenar contenido sería innecesario para la explicación de la aplicación.

1. Abstract

En este proyecto, vamos a explicar las diferentes áreas que debe cubrir la creación de una aplicación. Desde el punto de vista empresarial, veremos cómo está organizada nuestra empresa.

Veremos que requisitos debe cumplir nuestra aplicación después de hablar con el cliente, el hardware y software necesario y su base de datos. Incluiremos la programación de la aplicación y los diagramas de casos de uso y de clases con los que nos hemos ayudado.

Finalmente, explicaremos la interfaz gráfica de la aplicación.

In this project, we are going to explain the different areas to be covered in the creation of an application. From the business point of view, we will see how our company is organized.

We will see what requirements must meet our app after talking with the client, the hardware and software needed and its database. We will include the programming of the app and the use case diagrams and class diagrams with whom we used to help.

Finally, we will explain the graphic interface of the app.

2. Justificación

El objetivo de esta aplicación es dar un ejemplo de cómo una agencia de alquiler de coches podría establecer un sistema sencillo para que sus clientes pudieran **consultar los vehículos** de los que disponen.

Esto busca aprovechar las grandes producciones cinematográficas que se llevan a cabo para que les resulte **más atractivos nuestros servicios** al ofrecerles una plataforma amigable con los detalles que necesitan saber sobre el vehículo que necesitan para su película, desde coches hasta barcos.

Además, ofrecemos un enlace a una web externa para aquellos que quieran conocer con más profundidad los detalles del vehículo.

The objective of this application is to give an example of how a car rental agency could establish a simple system for their clients to check **the vehicles** the agency has available.

This tries to take advantage of the big cinematographic productions that are done nowadays to make **our services more attractive** to them offering a friendly platform with the details that they need to know about the vehicles that they need to their film, from cars to boats.

Furthermore, we offer a link to an external website to those who want to know about the details of the vehicle.

3. Objetivos

Los **objetivos básicos** del proyecto son:

- Crear un **sistema de registro de usuarios** conectado a una base de datos.
- Tener una **base de datos que almacene una serie de vehículos**.
- Utilizar esta base de datos para que el **usuario pueda consultar los vehículos** que ofrece la agencia.
- Ofrecer al usuario una vista con la **información del vehículo**.

The **basics objectives** of the project are:

- Create a **users registration system** connected to a database.
- Have a **database that stores vehicles**.
- Use this database to let the **user check the vehicles** that the agency offers.
- Offer to the user a view with **the vehicle data**.

4. Desarrollo

4.1 Acerca de la organización.



Nuestra empresa se llama **Hook Me**, somos una empresa pequeña de tres trabajadores, los cuales nos dedicamos al desarrollo de aplicaciones móviles.

Somos tres programadores que estudiamos el grado superior de DAW, aunque cada uno de especializa en un área como el backend, frontend y diseño gráfico.

Nuestros costes son los siguientes:

Our company is called **Hook Me**, we are a small company of 3 workers, which we are dedicated to the development of mobile applications.

We are three programmers that studied the superior grade of Web Applications Development, each one of us is specialized in one area like backend, frontend and graphic design.

Our costs are the followings:

Costes iniciales:

CONCEPTO	PRECIO
Ordenador Tipo A (torre, ratón, teclado, monitor) x 2	1714€
Ordenador Para Diseño (torre, ratón, teclado, monitor, tarjeta gráfica)	2600€
Publicar app Google Play Store	20€
Publicar app Apple Store	99€
Total	4433€

Costes mensuales:

CONCEPTO	PRECIO
Alquiler de la oficina	180€/mes
Servidor Hostinet	12€/mes
Dominio página web	1€/mes
Sueldos	1200€ al mes por empleado (x3, 3600)
Total	3793€

4.2 Análisis de requerimientos.

Tras la entrevista con nuestro cliente, AgenciaCochesPelis, nos ha concretado los siguientes **requisitos**:

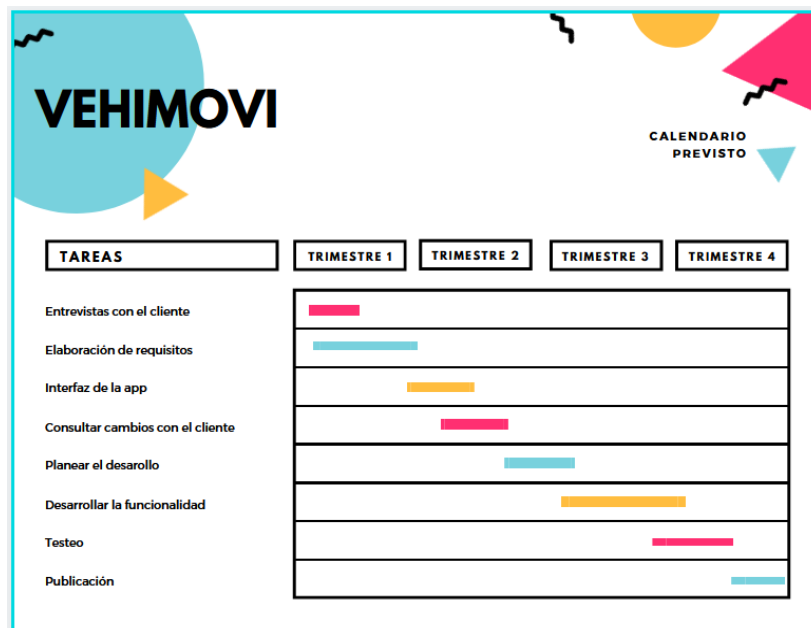
- El usuario debe poder **logearse con su cuenta tras registrarse**, dando datos de la productora para la que trabaja.
- El usuario podrá **consultar el catálogo de vehículos**.
- El usuario podrá **visualizar la información relativa a ese vehículo**.
- El usuario dispondrá de un **enlace** que le **redirigirá a una web con más información** del vehículo.
- El **administrador** podrá **añadir, modificar y borrar vehículos**.

After the interview with our client, AgenciaCochesPelis, they have specified the following **objectives**:

- The user must be able to **log in with his account after sign in**, giving data about the producer which he works.
- The user must be able **to check the vehicles catalog**.
- The user must be able **to view information relative to the vehicles**.
- The user will have a **link** that **will redirect him to a webpage with more information** about the vehicle.
- The **admin** will be able **to add, modify and delete vehicles**.

A continuación, mostraremos un **Diagrama de Gantt** con los tiempos estimados del desarrollo de la aplicación.

Now, we will show a **Gantt Diagram** with the stimated development times of the application.



Respecto al presupuesto de la aplicación, debemos sumar los costes iniciales y los costes mensuales (ambos detallados en el punto 5.1, Acerca de la organización), estos últimos multiplicados por 12 meses, que es el tiempo estimado para el desarrollo de la aplicación.

Regarding the application budget, we must add the initial costs and the monthly cost (both explained in the point 5.1), these last ones, multiplied by 12 months, which is the estimated time of the development of the application.

4433 (iniciales) * (3793 * 12) (mensuales durante un año) = 49949€

A esto le añadiremos unos 5000€ más para imprevistos, por lo que contaremos con un presupuesto aproximado de **55000€**.

We will add about 5000€ more for incidentals, so we will count with an approximate budget of **55000€**.

4.3 Hardware.

Para el **desarrollo de nuestra aplicación** dispondremos de tres ordenadores, uno para cada trabajador, los cuales han sido mencionados en el punto 5.1. Para realizar las distintas pruebas, necesitaremos distintos **dispositivos inteligentes** para comprobar que nuestra aplicación funciona en distintas plataformas, por lo que dispondremos de:

For the **development of our app**, we will have three computers, one for each worker, which have been mentioned in the point 5.1 To do the different tests, we will need different **smart devices** to check that our app works on different platforms, so we will have:

- Smartphone Android
- Smartphone iOS
- Tablet Android
- Tablet iOS

4.4 Sistemas operativos y aplicaciones.

En lo relativo al **software**, la aplicación será desarrollada en **Windows**, ya que es el sistema operativo al que más estamos acostumbrados.

En cuanto a programas, utilizaremos **Android Studio** para el desarrollo de la aplicación, el cual ya incluye un emulador para poder ir viendo los resultados de la programación. También, utilizaremos **Photoshop** para lo relativo a imágenes y diseño gráfico.

Para transformar la aplicación Android a iOS, utilizaremos **MechDome**, un programa que transforma las aplicaciones Android para que sean compatibles con sistemas iOS

Regarding the software, the application will be developed on **Windows**, which is the operating system that we are more used to.

Regarding programs, we will use **Android Studio** for the development of the application, which includes an emulator to check the results of the programming. Also, we will use **Photoshop** for the graphic desing.

To transform the application from Android to iOS, we will use **MechDome**, a program that transform the Android Applications to make them compatible with iOS systems.

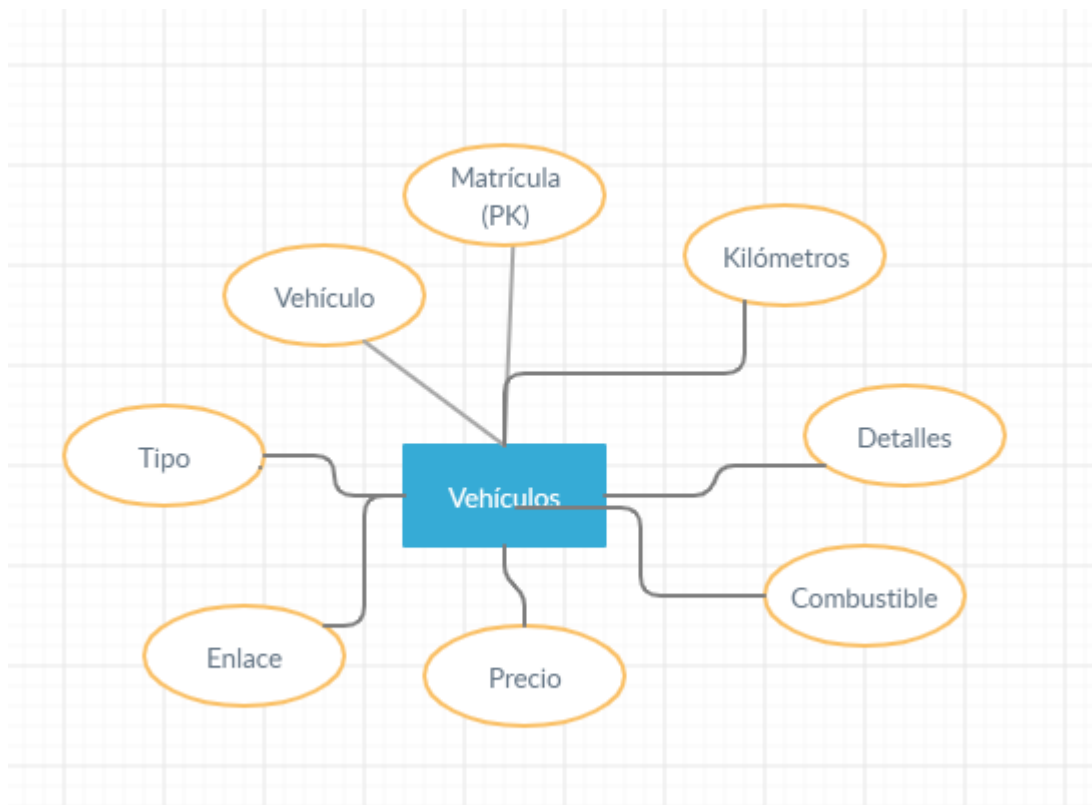
4.5 Base de datos y acceso a datos.

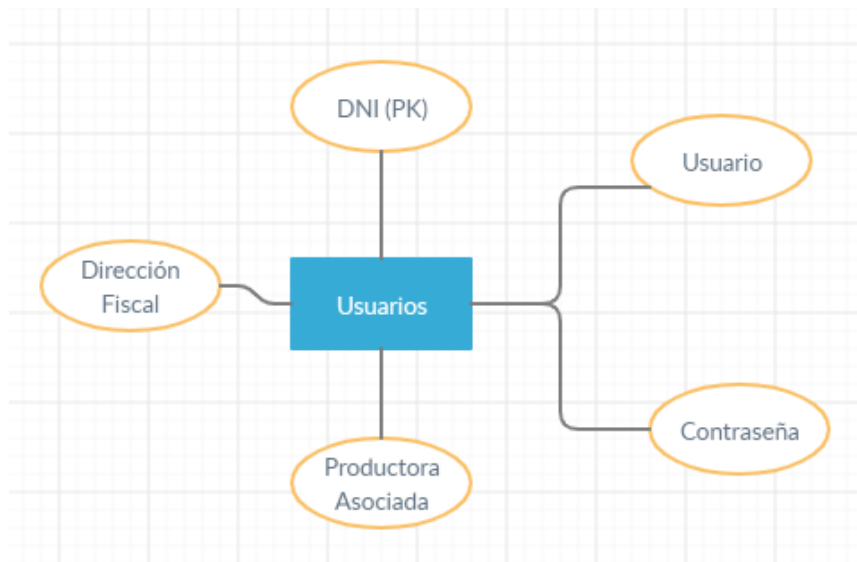
Para el desarrollo de la aplicación, he utilizado **SQLite**, que es un sistema de gestión de bases de datos relacional contenida en una relativamente pequeña biblioteca en C. He elegido este sistema ya que es un sistema sencillo, fácil de utilizar y que no necesita de conexiones externas para funcionar.

For the design of the application, I used **SQLite**, which is relational database management system stored in a small C library. I have chosen this system because it is a simple, easy to use and doesn't need external connections.

En cuanto a la estructura de la base de datos, la aplicación cuenta con dos tablas, **usuarios** y **vehículos**, cuya estructura es la siguiente:

Regarding the structure of the database, the application has two tables, **users** and **vehicles**, which structure is the following:





Para la creación de la base de datos, en primer lugar debemos ir al archivo **AndroidManifest.xml** de nuestra aplicación y añadir los permisos necesarios para poder escribir y leer datos de la base de datos:

To create the database, first of all, we must go the **AndroidManifest.xml** file of our application and add the needed permission to be able to write and read data from our database:

```
AndroidManifest.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    package="com.example.tfgcoches">
4
5    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
6    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
7  </manifest>
```

A continuación, tenemos que crear una clase Java que implemente la interfaz **SQLiteOpenHelper**, la cual al implementarla, crea un constructor para la clase, y los métodos **onUpdate** y **onCreate**.

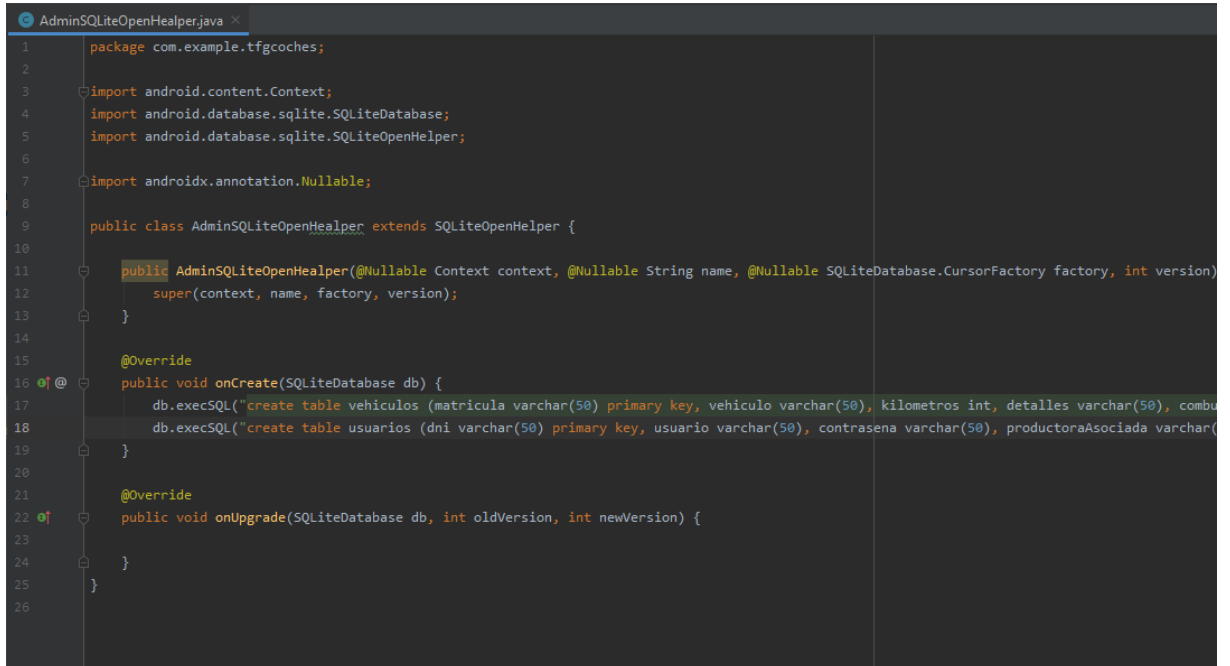
El método **onUpdate** se encarga de actualizar el esquema de la base de datos cuando hay cambios en su estructura.

El método **onCreate** se encarga de crear las tablas de nuestra base de datos, para ello debemos escribir la siguientes sentencias, una para cada una de las tablas que necesitemos, con su nombre y propiedades y el tipo de estas:

Next, we have to create a Java Class that implements the **SQLiteOpenHelper** interface, which when it's implemented, creates the class constructor, and the **onUpdate** and **onCreate** methods.

The **onUpdate** methods, updates the scheme of the database when there are changes on its structure.

The **onCreate** method creates the tables of our database, to do that, we must write the following sentences, one for each table that we need, with its name and properties and the types of these:



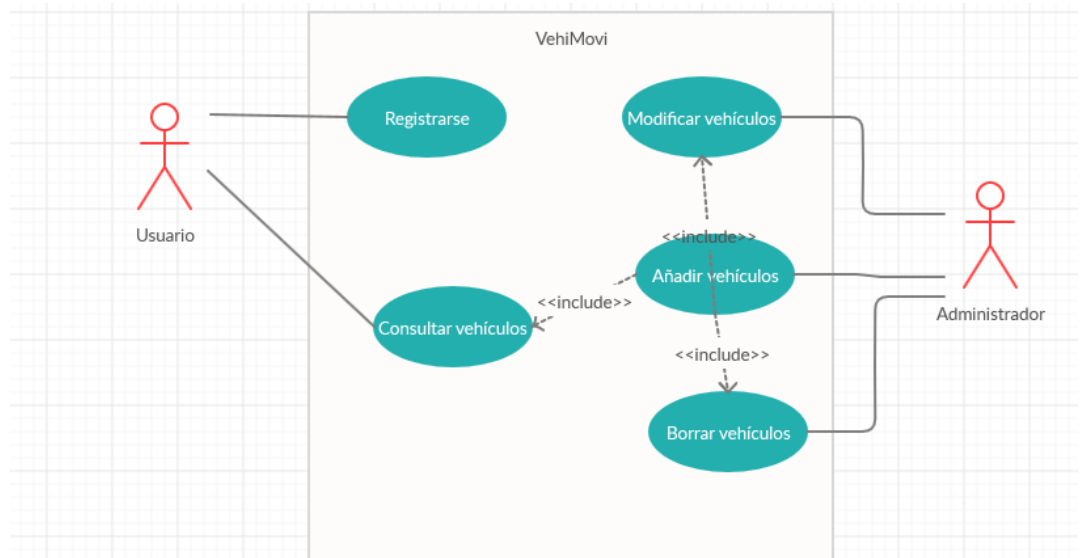
```
1 package com.example.tfgcoches;
2
3 import android.content.Context;
4 import android.database.sqlite.SQLiteDatabase;
5 import android.database.sqlite.SQLiteOpenHelper;
6
7 import androidx.annotation.Nullable;
8
9 public class AdminSQLiteOpenHelper extends SQLiteOpenHelper {
10
11     public AdminSQLiteOpenHelper(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {
12         super(context, name, factory, version);
13     }
14
15     @Override
16     public void onCreate(SQLiteDatabase db) {
17         db.execSQL("create table vehiculos (matricula varchar(50) primary key, vehiculo varchar(50), kilometros int, detalles varchar(50), combustible varchar(50))");
18         db.execSQL("create table usuarios (dni varchar(50) primary key, usuario varchar(50), contrasena varchar(50), productoraAsociada varchar(50))");
19     }
20
21     @Override
22     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
23
24     }
25 }
26
```

4.6 Diseño de la aplicación.

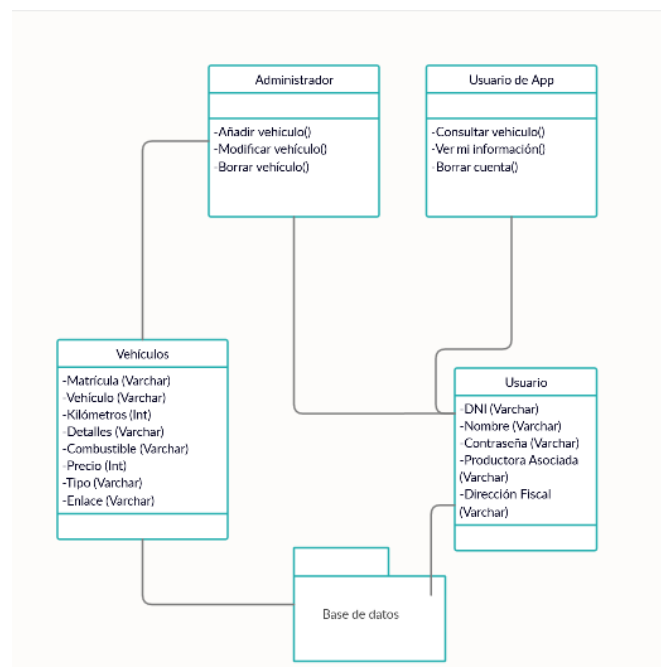
A continuación, para entender mejor la estructura y funcionamiento de la aplicación, se mostrarán los diagramas de casos de uso y diagramas de clase.

Next, to a better understanding of the structure and working of the application, we will show the use case and class diagram:

-Diagrama de casos de uso:



-Diagrama de clases:



4.7 Programación.

En este apartado, se explicará como funciona la aplicación.

In this part, we will explain how the application Works.

-Registro de usuarios.



En esta pantalla, el usuario rellenará el formulario con sus datos. Los campos de Contraseña y Repetir contraseña deben coincidir para que pueda llevarse a cabo el registro. En caso de que estos campos no coincidan, o el usuario deje algún campo en blanco, al presionar el botón de Registrar usuario, mostrará un mensaje por pantalla indicando estos errores; en caso de que los datos sean correctos, se le llevará a la pantalla de perfil de usuario que se explicará más adelante y sus datos serán insertados en la base de datos. El botón de Volver, le llevará de vuelta al menú de login.

In this screen, the user will fill the form with his data. The Password and Repeat password fields must be the same to be able to register. If they are not the same, or if the user leaves a blank field, when pressing the Register button, a message will be shown telling this errors; if the data is correct, he will be redirected to the profile screen, which will be explain later and his data will be inserted in the database. The Back button, will redirect him to the Log In menu.

Aquí se muestra el código que se ejecuta al pulsar el botón Registrar Usuario que realiza las comprobaciones que he mencionado antes:

Here is shown the code that is executed when pressing the Register User button that does the checking previously mentioned:

```
12 public void darAlta(View v){
13     if(dni.getText().toString().isEmpty() || usuario.getText().toString().isEmpty() || contra.getText().toString().isEmpty() || repeContra.getText().toString().isEmpty()){
14         Toast.makeText(this, "Rellena todos los campos", Toast.LENGTH_LONG).show();
15     }
16     else if(!contra.getText().toString().equals(repeContra.getText().toString())){
17         Toast.makeText(this, "Las contraseñas deben coincidir", Toast.LENGTH_LONG).show();
18     }
19     else{
20         AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper(this, "usuarios", null, 1);
21         SQLiteDatabase baseDatos = admin.getWritableDatabase();
22
23         ContentValues contenido = new ContentValues();
24         contenido.put("dni", dni.getText().toString());
25         contenido.put("usuario", usuario.getText().toString());
26         contenido.put("contrasena", contra.getText().toString());
27         contenido.put("productoraAsociada", productora.getText().toString());
28         contenido.put("direccionFiscal", direccion.getText().toString());
29
30         baseDatos.insert("usuarios", null, null, contenido);
31         baseDatos.close();
32
33         Intent i = new Intent(this, InfoUsuario.class);
34         i.putExtra("nombreUsuario", usuario.getText().toString());
35         startActivity(i);
36     }
37 }
```

En caso de pulsar el botón volver, se crea y ejecuta un Intent que le devuelve a la pantalla de Log In:

If the Back button is pressed, an intent is created and executed that redirect him back to the Log In screen:

```
public void volver(View v){  
    Intent i = new Intent( packageContext: this, MainActivity.class);  
    startActivity(i);  
}
```

-Log In



En la pantalla de Log In, se muestra el logo de la aplicación, junto con los campos de usuario y contraseña, además de los botones para ir al formulario de registro, explicado anteriormente, y el botón para Entrar. Dependiendo si el usuario es administrador o no, el botón le llevará a su menú correspondiente. Para iniciar sesión, el botón entrar realiza una consulta a la base de datos, donde saca la contraseña del usuario que se haya escrito en el campo usuario, en el caso de que esta coincida con lo que el usuario ha escrito en el campo contraseña, si estas coinciden podrá entrar, en caso contrario, saldrá un mensaje por pantalla notificando este error.

In the Log In screen, the application logo is shown, with the user and password fields, besides of the buttons to go to the Register form, previously explained, and the button to Log In. Depending of if the user is an admin or not, the button will redirect him to the correspondent menu. To log in, the button does a query to the database where it takes the password of the user that has been written on the user field, in case this match with the password that the user wrote on the password field, he will be able to log in, if not, a message will be shown telling this error.

A continuación se muestra el código que realiza las comprobaciones con la base de datos para iniciar sesión:

Next, the code that does the checking with the database to log in is shown:

```
MainActivity.java
38 }
39
40 public void login(View v){
41     Intent i;
42     if(usuario.getText().toString().isEmpty() || contra.getText().toString().isEmpty())
43         Toast.makeText(context, this, text: "Rellena todos los campos", Toast.LENGTH_LONG).show();
44
45     AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper(context, this, name: "usuarios", factory: null, version: 1);
46     SQLiteDatabase baseDatos = admin.getReadableDatabase();
47     Cursor fila = baseDatos.rawQuery( sql: "SELECT contrasena FROM usuarios WHERE usuario = " + usuario.getText().toString() + "'", selectionArgs:
48     String contraUsuario = "";
49
50     if(fila.moveToFirst()) {
51         contraUsuario = fila.getString( columnIndex: 0);
52     }
53
54     if(usuario.getText().toString().equals("admin") && contra.getText().toString().equals(contraUsuario)){
55         i = new Intent( packageContext: this, AdminMenu.class);
56         startActivity(i);
57     }
58
59     if(!usuario.getText().toString().equals("admin") && !usuario.getText().toString().equals("") && contra.getText().toString().equals(contraUs
60     i = new Intent( packageContext: this, UserMenu.class);
61     i.putExtra( name: "nombreUsuario", usuario.getText().toString());
62     startActivity(i);
63 }
64 baseDatos.close();
65 }
```

-Menú usuario.



En el menú del usuario se muestran varios botones con las opciones que tiene el usuario, dependiendo de cual seleccione, se le llevará a la pantalla correspondiente. El botón cerrar sesión, le llevará de vuelta al menú de Log In

In the user menú, various buttons are shown with the options that the user has, depending of which button is selected, he will be redirected to the corresponding screen. The Log Off button, he will be redirected to the Log In menu.

Funcionamiento de los botones:

Functions of the buttons:

```
public void logOut(View v){
    Intent i = new Intent( packageContext: this, MainActivity.class);
    i.putExtra( name: "nombreUsuario", datos.getString( key: "nombreUsuario"));
    startActivity(i);
}

public void irConsultar(View v){
    Intent i = new Intent( packageContext: this, ConsultarVehiculo.class);
    i.putExtra( name: "nombreUsuario", datos.getString( key: "nombreUsuario"));
    startActivity(i);
}

public void irInfo(View v){
    Intent i = new Intent( packageContext: this, InfoUsuari.class);
    i.putExtra( name: "nombreUsuario", datos.getString( key: "nombreUsuario"));
    startActivity(i);
}
```

-Acerca de



La pantalla de Acerca De simplemente muestra información sobre la app, indicando que tras encontrar el vehículo deseado, el cliente debe llamar a la empresa para reservarlo. El botón de volver le devuelve al menú de usuario.

The About screen simply shows information about the application, showing that after finding the needed vehicle, the client must call the company to book it. The button Back redirects the user to the User menu.

-Mi información



En esta pantalla, se saca la información del usuario que está logeado con una consulta a la base de datos y se coloca en el campo correspondiente. El botón volver le devuelve a su menú, mientras que el de Borrar Cuenta borra sus datos de la base de datos, en caso de seleccionar esta opción, se mostrará antes un cuadro de texto preguntando al usuario si está seguro de esta acción.

In this screen, the information about the user that is logged with a query to the database and it's placed in the corresponding field. The Back button redirects the user back to his menu, while the Delete Account button deletes his data from the database, in case of selecting this option, a message box is shown asking the user if he is sure of this action.

Consulta a la base de datos para sacar su información:

Query to the database to get its data:

```
public void sacarInformacion(){
    AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper( context, this, "usuarios", factory: null, version: 1);
    SQLiteDatabase baseDatos = admin.getReadableDatabase();
    Cursor fila = baseDatos.rawQuery( sql: "SELECT dni, usuario, productoraAsociada, direccionFiscal FROM usuarios WHERE usuario = '" + intent.ge

    if(fila.moveToFirst()){
        dni.setText(fila.getString( columnIndex: 0));
        usuario.setText(fila.getString( columnIndex: 1));
        productora.setText(fila.getString( columnIndex: 2));
        direccion.setText(fila.getString( columnIndex: 3));
    }

    baseDatos.close();
}
```

Proceso para borrar cuenta:

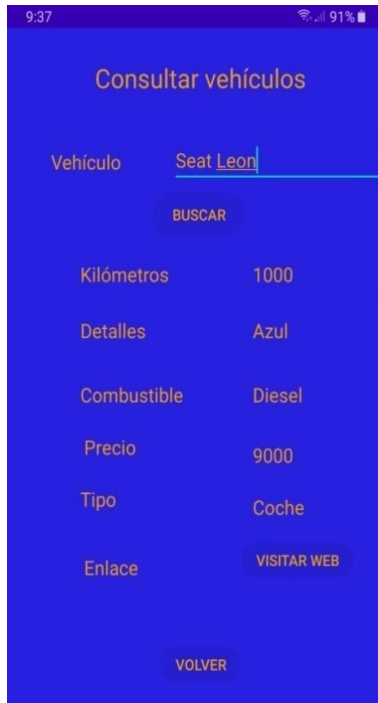
Process to delete its account:

```
public void borrarUsuario(View v){
    AlertDialog.Builder builder = new AlertDialog.Builder( context, this);
    builder.setMessage("¿Estás seguro de borrar tu cuenta?");
    builder.setPositiveButton("Si", (dialog, id) -> {
        accionBorrar();
        Intent volver = new Intent( packageContext: InfoUsuario.this, MainActivity.class);
        startActivity(volver);
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
        }
    });
    builder.create();
    builder.show();
}

public void accionBorrar(){
    AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper( context, this, "usuarios", factory: null, version: 1);
    SQLiteDatabase baseDatos = admin.getReadableDatabase();

    baseDatos.delete( table: "usuarios", whereClause: "usuario=" + intent.getString( key: "nombreUsuario") + "'", whereArgs: null);
    baseDatos.close();
}
```

-Consultar vehículo



En esta pantalla, el usuario introducirá el vehículo que quiere y se ejecutará una consulta a la base de datos para ver si la agencia lo tiene disponible, en caso afirmativo se mostrará su información, en caso de que no lo esté, se mostrará un mensaje por pantalla. Si el vehículo está disponible, se activará el botón de visitar web, el cual lleva al usuario a una web externa donde podrá ver más información sobre el vehículo, en caso de que el enlace proporcionado no sea válido, se mostrará un mensaje informando de este error. El botón volver devuelve al usuario a su menú.

In this screen, the user will introduce the vehicle that he wants and a query will be executed to the database to check if the agency has it available, if it is available, its information will be shown, if it isn't available, a message will be shown. If the vehicle is available, the visit web button will activate, which redirects the user to an external website where the user will can see more information about the vehicle, in case that the link it's not valid, a message will be shown telling this error. The Back button redirects the user to his menu.

Consultar y visualizar información del vehículo:

Query and visualize information of the vehicle:

```
48
49
50 public void buscar(View v){
51     AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper( context, "vehiculos", factory: null, version: 1);
52     SQLiteDatabase baseDatos = admin.getReadableDatabase();
53
54     String nomVehiculo = vehiculo.getText().toString();
55     Cursor fila = baseDatos.rawQuery( "sql: 'SELECT kilometros, detalles, combustible, precio, enlace, tipo FROM vehiculos WHERE vehiculo = '" + nomVehiculo + "'" );
56
57     if (fila.moveToFirst()) {
58         conKm.setText(fila.getString( columnIndex: 0));
59         conDetalles.setText(fila.getString( columnIndex: 1));
60         conCombust.setText(fila.getString( columnIndex: 2));
61         conPrecio.setText(fila.getString( columnIndex: 3));
62         direccion = fila.getString( columnIndex: 4);
63         conTipo.setText(fila.getString( columnIndex: 5));
64         visitar.setEnabled(true);
65     } else{
66         Toast.makeText( context, this, text: "No tenemos ese vehículo disponible", Toast.LENGTH_LONG).show();
67         visitar.setEnabled(false);
68         conKm.setText("");
69         conDetalles.setText("");
70         conCombust.setText("");
71         conPrecio.setText("");
72         direccion = "";
73         conTipo.setText("");
74     }
75     baseDatos.close();
76 }
```

Visitar enlace externo:

Visit external link:

```
ConsultarVehiculo.java
82     }
83
84     public void visitarWeb(View v){
85         Intent intent = new Intent(Intent.ACTION_VIEW);
86         intent.setData(Uri.parse(direccion));
87         if(intent.resolveActivity(getPackageManager())!= null){
88             startActivity(intent);
89         }
90     }
91     AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
92     builder.setMessage("Enlace no válido")
93     .setPositiveButton("Volver", new DialogInterface.OnClickListener() {
94         public void onClick(DialogInterface dialog, int id) {
95
96         }
97     });
98     builder.create();
99     builder.show();
100 }
101 }
102
103 }
104 }
```

-Menú administrador



El menú de administrador funciona de la misma forma que el del usuario, solo que tiene opciones diferentes que explicaremos a continuación.

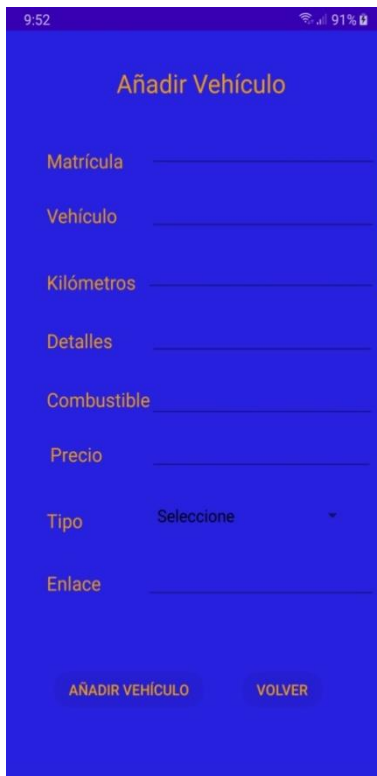
The admin menu Works the same way as the user menu, only that it has different option that we will explain next.

Funcionamiento de los botones:

Function of the buttons:

```
AdminMenu.java
14 setContentView(R.layout.activity_admin_menu);
15 }
16
17 public void logOut(View v){
18     Intent i = new Intent( packageContext: this, MainActivity.class);
19     startActivity(i);
20 }
21
22 public void irAnhadir(View v){
23     Intent i = new Intent( packageContext: this, AnhadirVehiculo.class);
24     startActivity(i);
25 }
26
27 public void irBorrar(View v){
28     Intent i = new Intent( packageContext: this, BorrarVehiculo.class);
29     startActivity(i);
30 }
31
32 public void irModi(View v){
33     Intent i = new Intent( packageContext: this, ModificarVehiculo.class);
34     startActivity(i);
35 }
36 }
37
```

-Añadir vehículo



En esta pantalla, el administrador deberá rellenar todos los campos y seleccionar un tipo de vehículo del spinner (coche, moto, camión, yate). En caso de que deje algún campo vacío, no se insertará el vehículo y se le notificará con un mensaje, en caso contrario, se hará un insert a la base de datos en la tabla de vehículos

In this screen, the admin must fill all the fields and select a vehicle type from the spinner (car, motorbike, truck and boat). In case that he leaves a blank field, the vehicle won't be added and he will be notified with a message, in other case, an insert will be done to the database to the vehicles table.

Funcionamiento del insert:

Insert functioning:

```
public void anhadir(View v){
    if(matricula.getText().toString().isEmpty() || vehiculo.getText().toString().isEmpty() || kilometros.getText().toString().isEmpty() || detalles.getText().toString().isEmpty() || combustible.getText().toString().isEmpty() || precio.getText().toString().isEmpty() || enlace.getText().toString().isEmpty() || tipo.getText().toString().isEmpty())
        Toast.makeText( context: this, text: "Rellena todos los campos", Toast.LENGTH_LONG).show();
    }
    else{
        AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper( context: this, name: "vehiculos", factory: null, version: 1);
        SQLiteDatabase baseDatos = admin.getWritableDatabase();

        String datMatri = matricula.getText().toString();
        String datVehi = vehiculo.getText().toString();
        String datKilo = kilometros.getText().toString();
        String datDeta = detalles.getText().toString();
        String datCombus = combustible.getText().toString();
        String datPrecio = precio.getText().toString();
        String datEnlace = enlace.getText().toString();
        String datTipo = tipoSeleccionado;

        ContentValues contenido = new ContentValues();
        contenido.put("matricula", datMatri);
        contenido.put("vehiculo", datVehi);
        contenido.put("kilometros", datKilo);
        contenido.put("detalles", datDeta);
        contenido.put("combustible", datCombus);
        contenido.put("precio", datPrecio);
        contenido.put("enlace", datEnlace);
        contenido.put("tipo", datTipo);
```

```
baseDatos.insert( table: "vehiculos", nullColumnHack: null, contenido);
Toast.makeText( context: this, text: "Vehiculo insertado", Toast.LENGTH_LONG).show();
matricula.setText("");
vehiculo.setText("");
kilometros.setText("");
detalles.setText("");
combustible.setText("");
precio.setText("");
enlace.setText("");
tipo.setSelection(0);

baseDatos.close();
    }
}
```

-Modificar vehículo



En esta pantalla el administrador podrá modificar la información de uno de los vehículos, excepto el tipo de vehículo por razones obvias, antes de hacer la modificación, se mostrará un cuadro de texto preguntado de si está seguro de estos cambios, en caso afirmativo se hará un update en la tabla vehículos a ese vehículo en concreto. No se podrán dejar campos vacíos.

In this screen, the admin will be able to modify one of the vehicles, except the type for obvious reasons, before modifying, a message box will be show asking if the admin is sure of this changes, if he is sure, an update will be done to the vehicles table to that vehicle. He won't be able to leave blank fields.

Proceso de modificar:

Modifying process:

```
public void modificar(View v){
    if(mVehiculo.getText().toString().isEmpty() || mKilometros.getText().toString().isEmpty() || mDetalles.getText().toString().isEmpty() ||
        Toast.makeText( context: this, text: "Rellena todos los campos", Toast.LENGTH_LONG).show();
    }
    else {
        AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
        builder.setMessage("¿Estás seguro de modificar el vehículo?")
            .setPositiveButton("Si", (dialog, id) -> {
                accionModificar();
            })
            .setNegativeButton("No", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                }
            });
        builder.create();
        builder.show();
    }
}
//fin del modificar
```

```
public void accionModificar(){
    AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper( context: this, name: "vehiculos", factory: null, version: 1);
    SQLiteDatabase baseDatos = admin.getReadableDatabase();

    ContentValues contenido=new ContentValues();
    contenido.put("kilometros", mKilometros.getText().toString());
    contenido.put("detalles", mDetalles.getText().toString());
    contenido.put("combustible", mCombustible.getText().toString());
    contenido.put("precio", mPrecio.getText().toString());
    contenido.put("enlace", mEnlace.getText().toString());

    baseDatos.update( table: "vehiculos", contenido, whereClause: "vehiculo=" + mVehiculo.getText().toString() + "", whereArgs: null);
    Toast.makeText( context: this, text: "Se ha modificado el vehículo " + mVehiculo.getText().toString(), Toast.LENGTH_LONG).show();
    baseDatos.close();
}
```

-Borrar vehículo

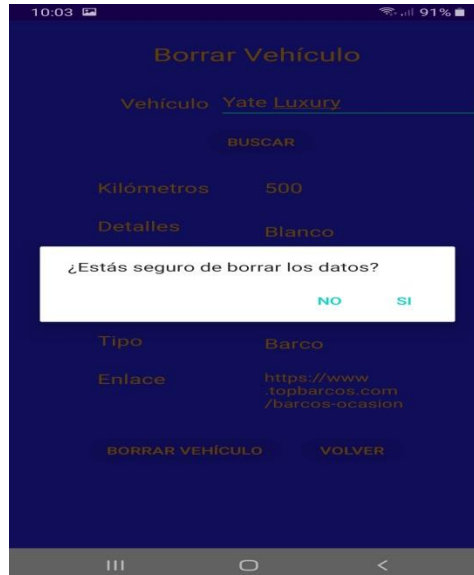


En esta pantalla el administrador podrá borrar uno de los vehículos de la base de datos, antes de realizar esta acción, se mostrará un mensaje preguntando si se está seguro de esto.

In this screen, the admin will be able to delete one of the vehicles of the database, before doing this action, a message box will be shown asking if he is sure of this action.

Mensaje de confirmación:

Confirmation message:



Proceso de borrar:

Deleting process:

```
BorrarVehiculo.java
67     }
68
69     public void accionBorrar(){
70         AdminSQLiteOpenHelper admin = new AdminSQLiteOpenHelper( context: this, name: "vehiculos", factory: null, version: 1);
71         SQLiteDatabase baseDatos = admin.getReadableDatabase();
72
73         baseDatos.delete( table: "vehiculos", whereClause: "vehiculo="+bVehiculo.getText().toString()+"", whereArgs: null);
74         Toast.makeText( context: this, text: "He borrado el vehiculo "+ bVehiculo.getText().toString(), Toast.LENGTH_LONG).show();
75     }
76
77     public void borrar(View v){
78         AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
79         builder.setMessage("¿Estás seguro de borrar los datos?")
80         .setPositiveButton("Si", (dialog, id) -> {
81             accionBorrar();
82             limpiarText();
83         })
84         .setNegativeButton("No", new DialogInterface.OnClickListener() {
85             public void onClick(DialogInterface dialog, int id) {
86
87             }
88         });
89     }
90
91     builder.create();
92     builder.show();
93 }//fin del metodo borrar
94
95
```

4.8 Desarrollo de interfaces.

Para la **apariciencia** de la aplicación, he optado por utilizar una técnica que se utiliza mucho a la hora de hacer posters para películas, que es el **contraste azul y naranja**, por lo que he elegido el azul como fondo de pantalla y el naranja como color de fuente.

He elegido utilizar **estructuras sencillas** para facilitar lo más posible al usuario el uso de la aplicación.

El **logo de la aplicación** también utiliza el naranja para hacer contraste con el fondo azul de la aplicación, utilizando la técnica que he mencionado antes. A continuación, mostraré unos posters en los que se utiliza esta técnica y un ejemplo de mi aplicación aplicándola.

For the **appearance** of the application, I decided to use a technique that it used a lot when doing movie posters, which is the **blue and orange contrast**, so I decided to use blue as the background and orange for the font color.

I decided to use **simple structures** to facilitate as much as possible the use of the application to the user.

The **application logo** also uses the orange to make contrast with the blue background of the application, using the technique that I mentioned previously. Next, I will show some posters that use this technique and an example of my application using this technique.



5 Bibliografía

Páginas consultadas en la realización del proyecto:

- <https://developer.android.com>
- <https://support.office.com>
- <https://es.venngage.com>
- <https://www.fayerwayer.com/>
- <https://es.wikipedia.org/>
- <https://riptutorial.com/>
- <https://www.paredro.com/>
- <https://app.creately.com>