

Proyecto Final

PeluApp: la aplicación para pedir citas en tu peluquería amiga.



Alumno:

Darío Rodríguez Linares

Profesor:

Santiago Martín-Palomo García

Curso:

Desarrollo Aplicaciones Multiplataforma

ÍNDICE

1. Introducción.....	5
2. Objetivo y alcance.....	5
3. Metodología de trabajo.....	6
4. Presupuesto.....	7
6. Diseño de la interfaz.....	8
7. Diseño Base de Datos.....	16
8. Diseño de la estructura del software.....	17
9. Codificación.....	18
9.1. Desarrollo de los activity XML.....	18
9.2. Desarrollo de las clases Java.....	36
9.2. Desarrollo de los archivos PHP.....	58
10. Manual de Usuario.....	64
11. Puesta en marcha.....	74
12. Mejoras.....	77
13. Conclusiones.....	78
14. Anexos.....	79
15. Bibliografía.....	80

Índice de figuras

Figura 1: Pantalla Login.....	8
Figura 2: Pantalla Registro.....	9
Figura 3: Pantalla Hub.....	10
Figura 4: Pantalla Pedir Cita.....	11
Figura 5: Pantalla Ver Rerservas.....	12
Figura 6: Pantalla Productos.....	13
Figura 7: Pantalla Ver Carro.....	14
Figura 8: Pantalla Ubicación.....	15
Figura 9: Tablas Base de Datos.....	16
Figura 10: Tabla Citas.....	16
Figura 11: Tabla Usuarios.....	16
Figura 12: Estructura del proyecto.....	17
Figura 13: dbbarber.php.....	58
Figura 14: register.php.....	59
Figura 15: validateuser.php.....	60
Figura 16: cita.php.....	60
Figura 17: buscarcita.php.....	61
Figura 18: cancelar.php.....	62
Figura 19: Prueba Login.....	63
Figura 20: Usuario o contraseña incorrectos.....	64
Figura 21: Alguno de los campos está vacío.....	64
Figura 22: El email o usuario ya está en uso.....	64
Figura 23: Registrado con éxito.....	65
Figura 24: Se muestra el nombre de usuario.....	66
Figura 25: Selección de cita.....	67
Figura 26: La cita se creó correctamente.....	67
Figura 27: Esa cita no está disponible.....	67
Figura 28: Se muestra la cita.....	68
Figura 29: Se canceló correctamente.....	68
Figura 30: No se canceló.....	69
Figura 31: Selección de productos.....	70
Figura 32: Productos seleccionados.....	71

Figura 33: Ubicación.....	72
Figura 34: Generar APK.....	73
Figura 35: Generar Key store.....	74
Figura 36: Rellenar los datos para la Key store.....	74
Figura 37: Se han rellenado los campos.....	75
Figura 38: Generación de APK.....	75

1. Introducción

PeluApp es un proyecto que trata de una aplicación móvil para reservar citas en la peluquería. Permite elegir el día, la hora, el peluquero/a y el tipo de servicio que quiere el cliente. El problema a solucionar será poder pedir una cita sin tener que llamar al establecimiento y sin necesidad de reservar en horario laborable.

La idea es que tanto el cliente como el staff de la peluquería puedan manejar de manera sencilla la aplicación. Estará diseñada para hacer más sencilla la idea de ir a la peluquería en tiempos como los que vivimos hoy en día, donde debemos respetar las normas sanitarias. Ésta aplicación es una buena solución para ello, ya que le permitirá al cliente pedir su cita sin necesidad de ir presencialmente al establecimiento.

2. Objetivo y alcance

Esta aplicación tiene como objetivo permitir a los clientes pedir citas en la peluquería, pudiendo elegir tanto el día, la hora, el peluquero/a y el servicio. Por otro lado, la peluquería podrá obtener las citas y así saber que días, horas y peluqueros/as están disponibles.

El cliente va a obtener la aplicación móvil para su propio uso, instalada en un sistema Android. Además, se le dará el manual de usuario para que el cliente pueda aprender como se usa la aplicación correctamente. Por último, se le entregará la guía de instalación para que el cliente ponga en marcha la aplicación y conozca las normas de explotación de la app.

Los recursos que se van a utilizar serán:

Tabla 1: Recursos a usar

RECURSO	USO
Android Studios	Creación de código fuente
MYSQL	Creación base de datos SQL
GitHub	Alojamiento del proyecto y control de versiones

3. Metodología de trabajo

La metodología a usar en este proyecto será la metodología Scrum, que es una estructura de trabajo ágil que facilita la administración del proyecto.

Se realizarán las siguientes fases:

- Fase de Investigación y Análisis
- Fase de Diseño
- Fase de Implementación
- Fase de Pruebas

Los sprints serán los siguientes:

1º Idea y anteproyecto 02/04/2021

2º Versión básica documentada 23/04/2021

3º Versión mejorada documentada 07/05/2021

4º Entrega final 21/05/2021

4. Presupuesto

El precio medio para el desarrollo de aplicaciones en España es de 35 euros por hora aproximadamente.

Tabla 2: Presupuesto

FASE	TAREA	TIEMPO (HORAS)	PRECIO
Investigación y Análisis		5	175
	Toma de requisitos	2	
	Definición del sistema	2	
	Documentación	1	
Diseño		10	350
	Identificación de los elementos	2	
	Desarrollar el modelo	6	
	Documentación	2	
Implementación		20	700
	Desarrollo de la Aplicación	10	
	Desarrollo de la BBDD	8	
	Documentación	2	
Pruebas		5	175
	Unitarias	1,5	
	Funcionales	1	
	Integración	1	
	Corrección de errores	1	
	Documentación	0,5	
TOTAL		40 horas	1400 euros

6. Diseño de la interfaz

El diseño de la interfaz se basará en seis pantallas, Login, Registro, Hub, Pedir cita, Ver cita y Ubicación.

Pantalla Login:

Tenemos un botón para logearnos con nuestro usuario y contraseña y un TextView que al clicar iremos a la pantalla de Registro.

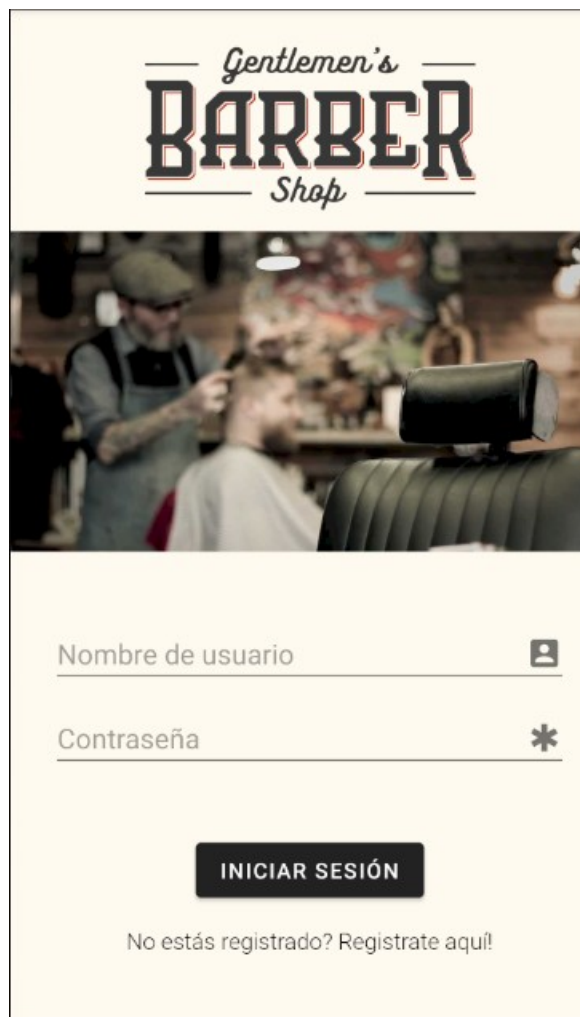


Figura 1: Pantalla Login

Una vez cliquemos en el botón de **Login** y el usuario y contraseña sean correctos, nos llevará la a pantalla Hub. En el caso de no esté registrado o el usuario y contraseña sean erróneos, se nos mostrará un mensaje de error, para que corrijamos nuestros datos o nos registremos.

Si no hemos rellenado alguno de los campos, al pulsar los botones no podremos seguir adelante y se nos avisará a través de un mensaje de que no hemos rellenado alguno de los campos.

Al pulsar el TextView nos llevará a la pantalla de registro con todos sus EditTexts para rellenarlos y registrarnos.

Pantalla Registro

Esta pantalla cuenta con un **botón Registro**, seis **EditText** y un **TextView**.

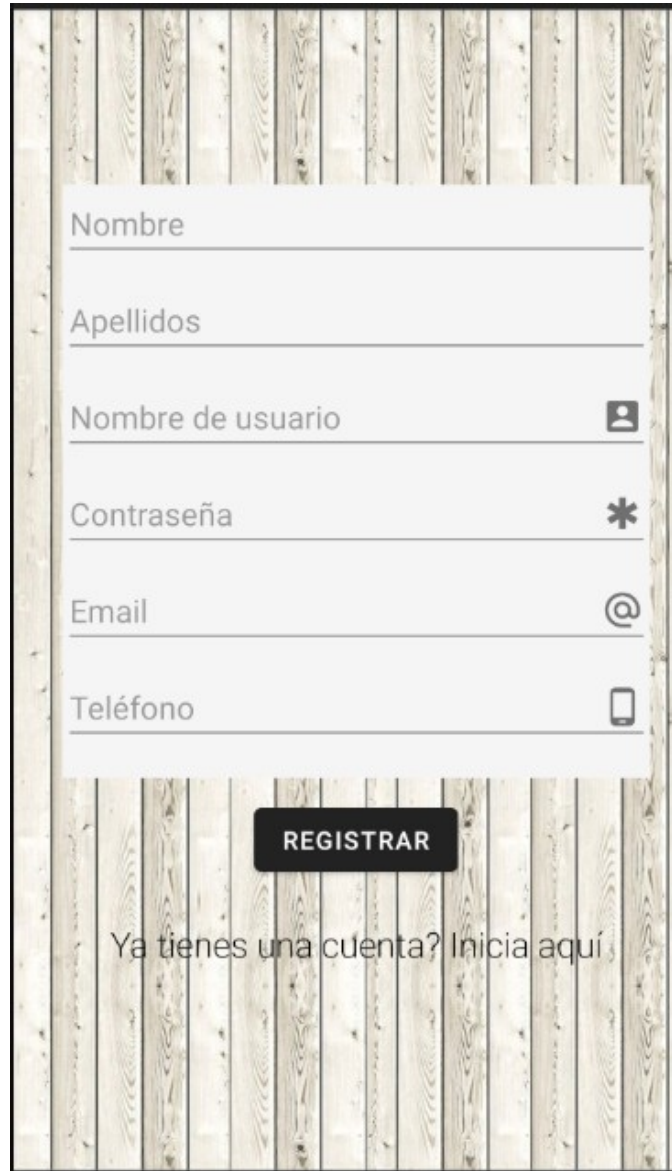


Figura 2: Pantalla Registro

Los **EditTexts** son para rellenar el Nombre, Apellidos, Nombre de usuario, Contraseña, Email y Teléfono.

Al pulsar el **botón Registro**, si todos los campos están llenos y el nombre de usuario y el email no están ya registrados, se nos reenviará a la pantalla de Login.

Si uno de los campos está vacío o el email o el usuario están ya registrados, se nos mostrará un mensaje de error.

Si pulsamos en el **TextView** se nos enviará a la pantalla de **Login**.

Pantalla Hub:

En esta pantalla podremos seleccionar la acción que queremos realizar, es decir, a que pantalla queremos ir.

Tenemos en total **cuatro botones, Pedir, Ver mis reservas, Comprar productos y Ubicación.**

También tenemos un **TextView** que se modificará según el nombre del Usuario.



Al pulsar en el **botón Pedir Cita** se nos enviará a la pantalla para poder seleccionar nuestra cita.

Si pulsáramos el **botón Ver mis reservas** se nos trasladará a la pantalla donde veremos las cistas que tenemos pendientes, además de poder cancelarla.

A través del **botón Comprar productos** iremos a una pantalla donde podremos seleccionar los productos o servicios que queremos comprar y podremos ver cuanto será el montante total que deberemos pagar.

Por último, el **botón Ubicación** nos llevará a una pantalla con un vídeo de la peluquería y la ubicación en Google Maps.

Pantalla Pedir cita:

Como vemos en la imagen, esta pantalla se compone de **tres spinners**, un **EditText** para seleccionar la fecha que queremos, un **botón** y un **TextView**.

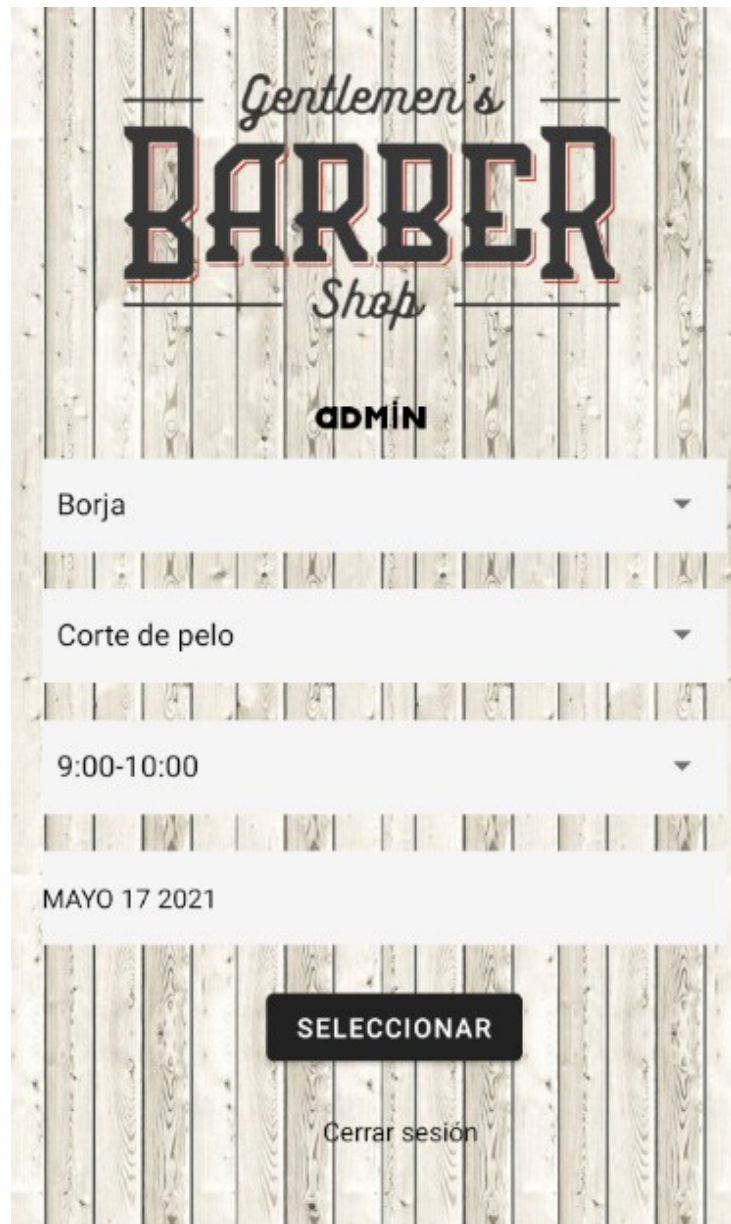


Figura 4: Pantalla Pedir Cita

El primer **EditText** nos permite seleccionar el peluquero que queremos que nos atienda.

El siguiente sería para seleccionar el servicio que queremos y el otro es para seleccionar el horario.

El **EditText** nos permite elegir la fecha que más nos convenga.

Con el **botón Seleccionar** finalizaríamos el registro de la cita.

Con el **TextView Cerrar Sesión**, nos llevará a la pantalla de Login.

Pantalla Ver reserva:

En esta pantalla tenemos dos **EditTexts** donde podemos ver nuestra cita. También hay un **botón Cancelar** para eliminar la cita que tenemos.



Figura 5: Pantalla Ver Reservas

Los **EditTexts** cambiarán en función del registro que hayamos realizado, y podremos cancelar cualquier cita con el **botón Cancelar**.

Pantalla Productos:

Esta pantalla contiene un **RecyclerView** y un **botón Ver carro de compras**.

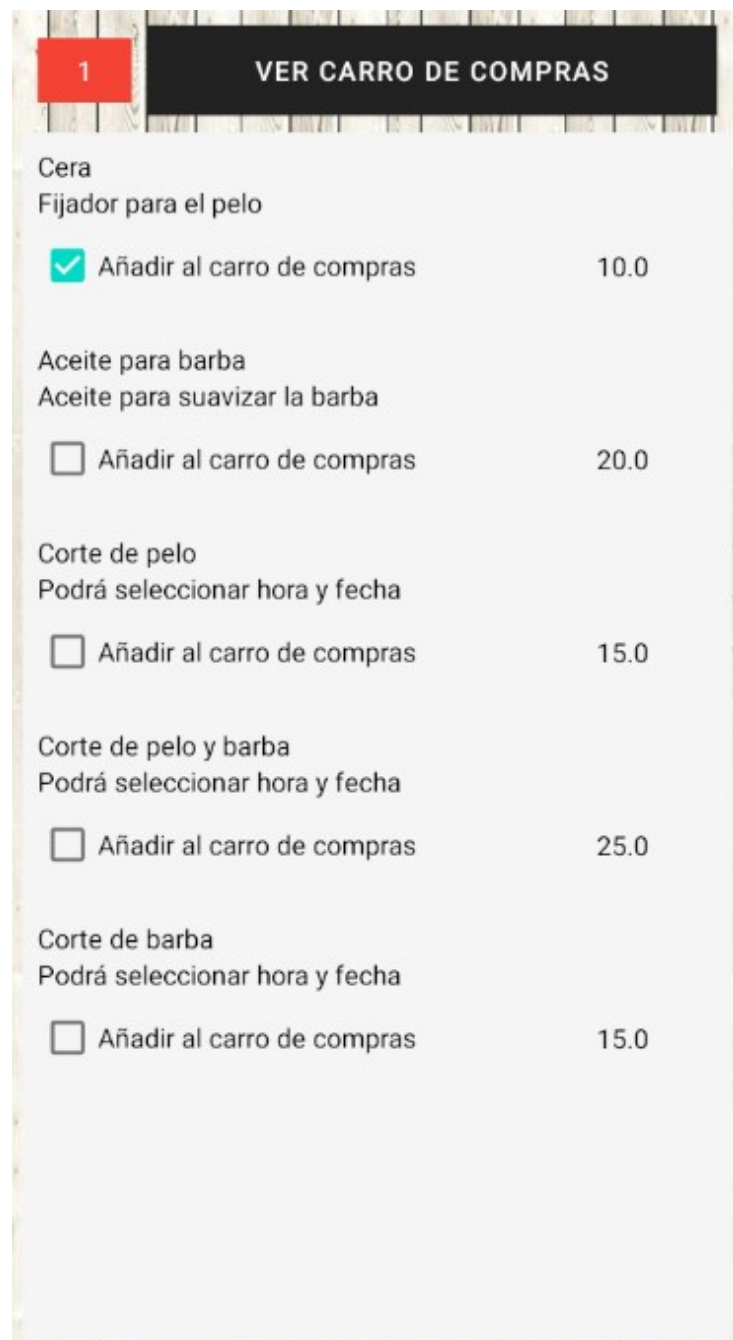


Figura 6: Pantalla Productos

En el **RecyclerView** podremos ver y seleccionar los productos disponibles y con el **botón Ver Carro** veremos el montante final.

Pantalla Ver Carro:

Veremos lo seleccionado previamente y el precio total.

[illegible]

Figura 7: Pantalla Ver Carro

Pantalla Ubicación:

Con el VideoView veremos el vídeo y con un MapView veremos la ubicación de la peluquería.

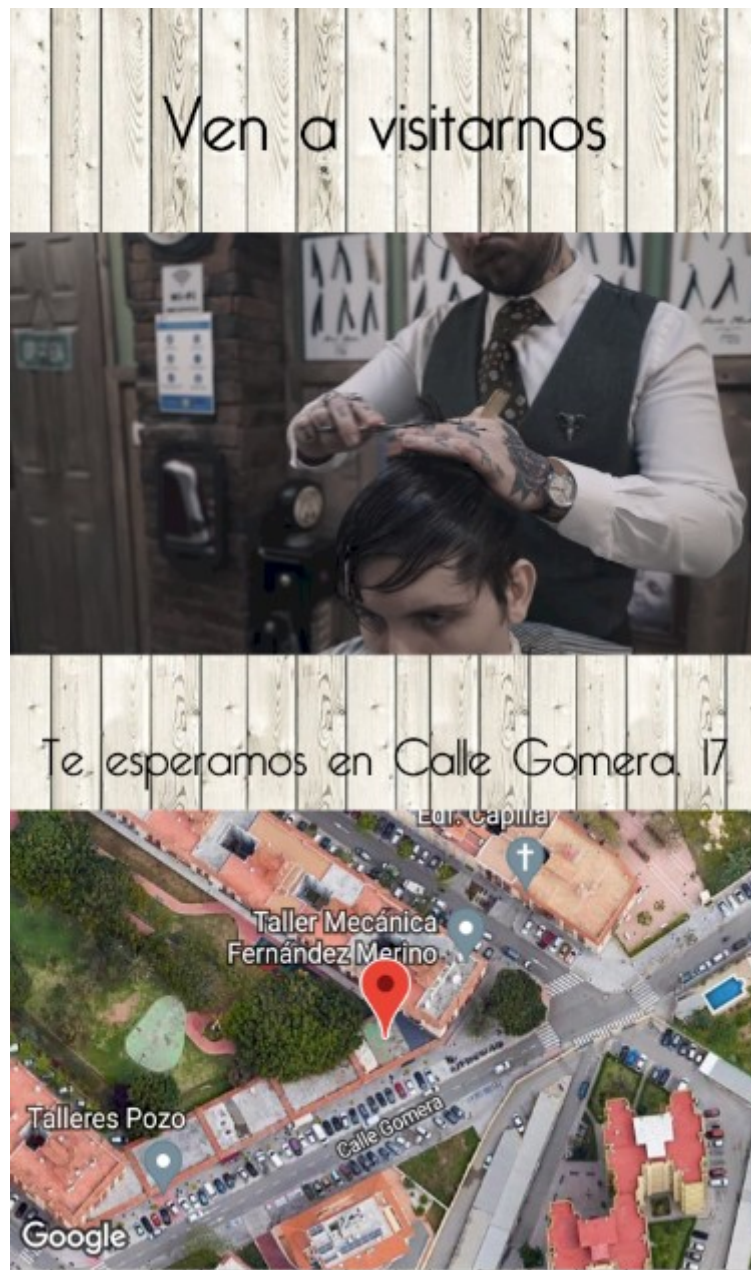


Figura 8: Pantalla Ubicación

7. Diseño Base de Datos

La base de datos tendrá dos tablas, una llamada **usuarios** para los datos del usuario, con su id, nombre, apellidos, usuario, contraseña, email y teléfono. La segunda tabla llamada **citas** que contiene la cita, el servicio y el nombre de usuario que será la clave foránea de la **tabla usuarios**.

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
<input type="checkbox"/> citas	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> usuarios	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_general_ci	32.0 KB	-
2 tablas Número de filas		4	InnoDB	utf8mb4_general_ci	48.0 KB	0 B

Figura 9: Tablas Base de Datos


	#	Nombre	Tipo
<input type="checkbox"/>	1	cita	varchar(300)
<input type="checkbox"/>	2	servicio	varchar(100)
<input type="checkbox"/>	3	username 	varchar(30)

Figura 10: Tabla Citas



Nombre	Tipo
id 	int(11)
nombre	varchar(30)
apellidos	varchar(60)
usuario 	varchar(30)
password	varchar(20)
email	varchar(60)
phone	int(12)

Figura 11: Tabla Usuarios

8. Diseño de la estructura del software

El proyecto se estructurará en una serie de carpetas. La carpeta **java/com.example.proyecto** donde estarán los archivos Java, la carpeta **drawable** donde estarán todos los archivos de imágenes, la carpeta **font** donde estarán las fuentes de letra, la carpeta **layout** donde estarán las pantallas en código xml y por último la carpeta **values** donde estarán los archivos con los valores necesarios para el proyecto como son colores y strings.

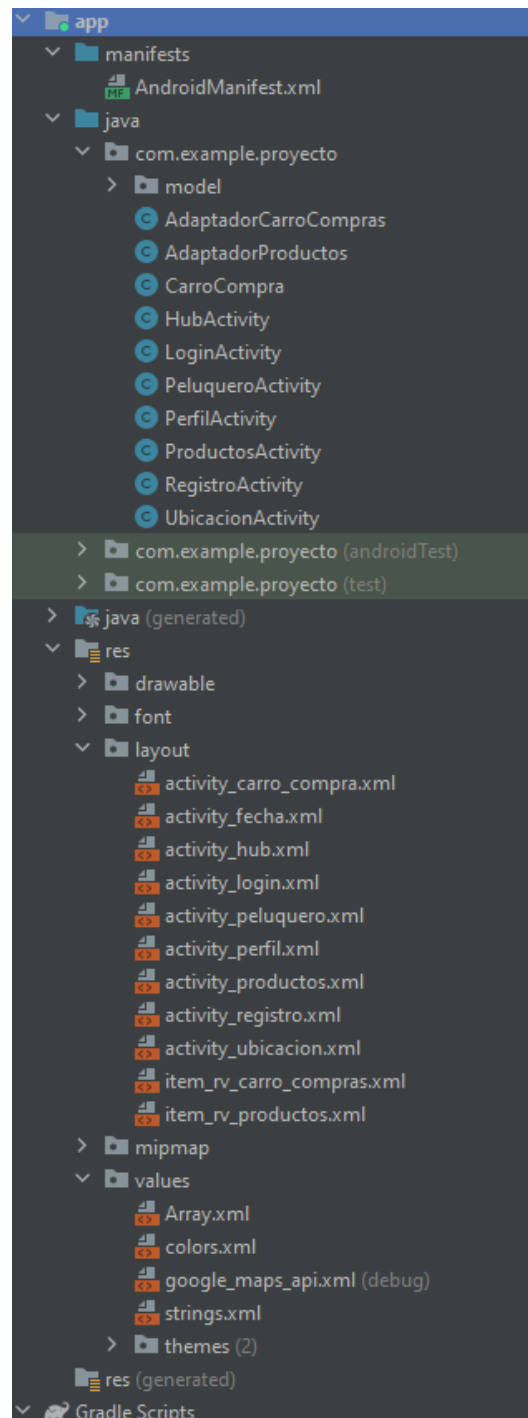


Figura 12: Estructura del proyecto

9. Codificación

Una vez realizada la interfaz y teniendo claro las funciones que vamos a implementar, empezaremos a escribir el código.

9.1. Desarrollo de los activity XML

activity_login:

En este caso he usado un Button y un LinearLayout para ordenar los dos EditTexts.

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

    <Button
        android:id="@+id/btnLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="130dp"
        android:layout_marginTop="540dp"
        android:layout_centerHorizontal="true"
        android:text="Iniciar sesión" />

    <LinearLayout
        android:layout_width="339dp"
        android:layout_height="135dp"
        android:layout_marginStart="35dp"
        android:layout_marginTop="400dp"
        android:layout_centerHorizontal="true"
        android:orientation="vertical">

        <EditText
            android:id="@+id/usuarioETLogin"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:hint="@string/usuario"
            android:drawableRight="@drawable/ic_baseline_account_box_24"
            android:inputType="textPersonName" />

        <EditText
            android:id="@+id/passwordLoginET"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
```

```

        android:ems="10"
        android:hint="@string/password"
        android:drawableRight="@drawable/ic_baseline_emergency_24"
        android:inputType="textPassword" />
</LinearLayout>

<TextView
    android:id="@+id/TextViewRegistro"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:fontFamily="@font/roboto_light"
    android:textColor="@color/black"
    android:textSize="15sp"
    android:layout_marginStart="85dp"
    android:layout_marginTop="600dp"
    android:text="No estás registrado? Regístrate aquí!" />
</RelativeLayout>

```

activity_registro:

Tenemos un LinearLayout con seis EditTexts, un Button y un TextView que usaremos para ir a la pantalla Login.

```

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

    <LinearLayout
        android:layout_width="336dp"
        android:layout_height="339dp"
        android:layout_marginStart="35dp"
        android:layout_marginTop="100dp"
        android:layout_centerHorizontal="true"
        android:background="@color/browser_actions_bg_grey"
        android:orientation="vertical">

        <EditText
            android:id="@+id/nombreETRegistro"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:hint="@string/nombre"
            android:inputType="textPersonName" />

        <EditText
            android:id="@+id/apellidosETRegistro"

```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
android:ems="10"
android:hint="@string/apellidos"
android:inputType="textPersonName" />
```

<EditText

```
android:id="@+id/usuarioETRegistro"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
android:ems="10"
android:hint="@string/usuario"
android:drawableRight="@drawable/ic_baseline_account_box_24"
android:inputType="textPersonName" />
```

<EditText

```
android:id="@+id/passwordETRegistrer"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
android:ems="10"
android:hint="@string/password"
android:drawableRight="@drawable/ic_baseline_emergency_24"
android:inputType="textPassword" />
```

<EditText

```
android:id="@+id/emailETRegistro"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
android:ems="10"
android:hint="@string/email"
android:drawableRight="@drawable/ic_baseline_alternate_email_24"
android:inputType="textEmailAddress" />
```

<EditText

```
android:id="@+id/phoneETRegistro"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:drawableRight="@drawable/ic_baseline_phone_android_24"
android:layout_marginTop="10dp"
android:ems="10"
android:hint="@string/telefono"
android:inputType="phone" />
```

</LinearLayout>

<Button

```
android:id="@+id/btnRegistrer"
android:layout_width="wrap_content"
```

```

        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginStart="150dp"
        android:layout_marginTop="450dp"
        android:text="Registrar" />

<TextView
    android:id="@+id/TextViewInicio"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="@color/black"
    android:onClick="irInicio"
    android:textSize="20sp"
    android:layout_centerHorizontal="true"
    android:fontFamily="@font/roboto_light"
    android:text="Ya tienes una cuenta? Inicia aquí"
    android:layout_marginStart="105dp"
    android:layout_marginTop="520dp"/>
</RelativeLayout>

```

activity_hub:

Colocamos un ImageView para mostrar el logo de la peluquería. Un TextView para mostrar el nombre del usuario que se ha logeado y un LinearLayout para ordenar de manera vertical los Buttons que nos llevarán a las distintas pantallas.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".HubActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:layout_width="266dp"
        android:layout_height="206dp"
        app:srcCompat="@drawable/logo" />

    <TextView
        android:id="@+id/nomUsuario"
        android:layout_centerHorizontal="true"
        android:text=""
        android:textColor="@color/black"
        android:layout_width="wrap_content"
        android:textSize="24sp"
        android:fontFamily="@font/classique_saigon"
        android:layout_height="wrap_content"

```

```
        android:layout_below="@+id/imageView"/>

<LinearLayout
    android:layout_width="263dp"
    android:layout_height="wrap_content"
    android:layout_below="@+id/nomUsuario"
    android:layout_centerHorizontal="true"
    android:orientation="vertical">

    <Button
        android:id="@+id/btnCita"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="Pedir cita" />

    <Button
        android:id="@+id/btnPerfil"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="Ver mis reservas" />

    <Button
        android:id="@+id/btnProductos"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="Comprar productos" />

    <Button
        android:id="@+id/btnUbicacion"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="Ubicación" />

</LinearLayout>

</RelativeLayout>
```

activity_peluquero:

En este XML he colocado varios LinearLayouts para meter tres spinners, uno en cada y por último un EditTexts que usaremos para seleccionar la fecha. Por último, un Button para realizar las selección y un TextView para cerrar la sesión y volver a la pantalla de Login.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:tools="http://schemas.android.com/tools"
    android:background="@drawable/background"
    tools:context=".PeluqueroActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
        android:layout_width="266dp"
        android:layout_height="206dp"
        app:srcCompat="@drawable/logo" />

    <TextView
        android:id="@+id/nomUsuario"
        android:layout_below="@+id/imageView"
        android:layout_centerHorizontal="true"
        android:text=""
        android:textColor="@color/black"
        android:layout_width="wrap_content"
        android:textSize="24sp"
        android:fontFamily="@font/classique_saigon"
        android:layout_height="wrap_content" />

    <LinearLayout
        android:background="@color/browser_actions_bg_grey"
        android:layout_width="366dp"
        android:layout_height="50dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="250dp"
        android:orientation="horizontal">

        <Spinner
            android:id="@+id/spinnerPelu"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

    </LinearLayout>

    <LinearLayout
```

```
android:background="@color/browser_actions_bg_grey"
android:layout_width="366dp"
android:layout_height="50dp"
android:layout_marginStart="20dp"
android:layout_marginTop="320dp"
android:orientation="vertical">

<Spinner
    android:id="@+id/spinnerServicio"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
</LinearLayout>

<LinearLayout
    android:background="@color/browser_actions_bg_grey"
    android:layout_width="366dp"
    android:layout_height="50dp"
    android:layout_marginStart="20dp"
    android:layout_marginTop="390dp"
    android:orientation="horizontal">

    <Spinner
        android:id="@+id/spinnerHoras"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1" />
    </LinearLayout>

<LinearLayout
    android:layout_width="366dp"
    android:layout_height="50dp"
    android:layout_marginStart="20dp"
    android:layout_marginTop="460dp"
    android:orientation="horizontal">

    <EditText
        android:background="@color/browser_actions_bg_grey"
        android:id="@+id/btnFecha"
        style="?android:spinnerStyle"
        android:text="Enero 01 2020"
        android:textColor="@color/black"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:onClick="openDatePicker"/>
    </LinearLayout>

<Button
    android:id="@+id/btnSelectPelu"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="140dp"
```



```

        android:layout_marginTop="530dp"
        android:text="@string/seleccionar" />

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="170dp"
    android:layout_marginTop="600dp"
    android:textColor="@color/black"
    android:text="Cerrar sesión" />

</RelativeLayout>

```

activity_perfil:

En este activity nos encontramos con cuatro TextViews donde nos aparecerá la fecha de la cita y el servicio que hemos seleccionado.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".PerfilActivity">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="334dp"
        android:layout_height="266dp"
        android:layout_centerHorizontal="true"
        app:srcCompat="@drawable/logo" />

    <TextView
        android:id="@+id/introduzcaView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="250dp"
        android:fontFamily="@font/classique_saigon"
        android:text=""
        android:textColor="@color/black"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/citaView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

```

```
android:layout_centerHorizontal="true"
android:layout_margin="60dp"
android:fontFamily="@font/classique_saigon"
android:textColor="@color/black"
android:textSize="24sp"
android:text="Su cita es:"
android:layout_below="@+id/introduzcaView"/>
```

<TextView

```
android:id="@+id/cita"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_below="@+id/citaView"
android:fontFamily="@font/classique_saigon"
android:textSize="20sp"
android:textColor="@color/black"
android:text="No tiene ninguna cita pendiente" />
```

<TextView

```
android:id="@+id/servicio"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:textColor="@color/black"
android:fontFamily="@font/classique_saigon"
android:textSize="20sp"
android:layout_below="@+id/cita"
android:layout_margin="15dp" />
```

<Button

```
android:id="@+id/btnCancelar"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/servicio"
android:layout_centerHorizontal="true"
android:layout_marginTop="35dp"
android:onClick="onClick"
android:text="Cancelar cita" />
```

```
</RelativeLayout>
```

activity_productos:

Este activity contiene un TextView donde aparecerán la cantidad de productos, un Button que nos enviará al activity_carro_compras y un RecyclerView donde aparecerán todos los productos disponibles para seleccionar.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".ProductosActivity">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/tvCantProductos"
                android:layout_width="50dp"
                android:layout_height="0dp"
                android:layout_marginStart="8dp"
                android:layout_marginLeft="8dp"
                android:layout_marginTop="8dp"
                android:layout_marginBottom="8dp"
                android:background="#F44336"
                android:gravity="center"
                android:text="0"
                android:textColor="#FFFFFF"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <Button
                android:id="@+id/btnVerCarro"
```

```
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="#4CAF50"
    android:text="VER CARRO DE COMPRAS"
    android:textColor="#FFFFFF"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/tvCantProductos"
    app:layout_constraintTop_toTopOf="parent" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
</LinearLayout>
```

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/rvListaProductos"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginBottom="8dp"
    android:background="@color/browser_actions_bg_grey"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_carro_compras:

Al igual que en activity_productos tenemos un RecyclerView donde vemos los productos que hemos seleccionado y precio total que se muestra a través de un TextView.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".CarroCompra">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvListaCarro"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:background="@color/browser_actions_bg_grey"
        app:layout_constraintBottom_toTopOf="@+id/tvTotal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tvTotal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:textColor="@color/black"
        android:layout_marginBottom="120dp"
        android:gravity="center"
        android:background="@color/white"
        android:text="0.0"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="236dp"
        android:layout_height="55dp"
        android:layout_marginTop="650dp"
        app:layout_constraintEnd_toEndOf="parent"
```

```

app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent">

<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    </androidx.constraintlayout.widget.ConstraintLayout>

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

```

activity_ubicacion:

Para mostrar el mapa de Google Maps usamos un **fragment** con un atributo name con el URL del soporte de Maps. En el caso de Youtube he escogido una API externa para mostrar vídeos de Youtube, donde en el atributo videoId pondremos parte del URL del vídeo de Youtube.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".UbicacionActivity">

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="240dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="1dp"
        tools:context=".MapsActivity" />

    <com.pierfrancescosoffritti.androidyoutubeplayer.core.player.views.YouTubePlayerView
        android:id="@+id/videoview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="120dp"
        app:autoPlay="true"
        app:showYouTubeButton="false"
        app:videoId="pflgK2eK18g" />

```

```

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Ven a visitarnos"
    android:layout_centerHorizontal="true"
    android:textColor="@color/black"
    android:textSize="40sp"
    android:fontFamily="@font/bellerose"
    android:layout_marginTop="40dp"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="10dp"
    android:layout_height="wrap_content"
    android:text="Te esperamos en Calle Gomera, 17"
    android:textColor="@color/black"
    android:textSize="27sp"
    android:layout_above="@id/map"
    android:fontFamily="@font/bellerose"/>

</RelativeLayout>

```

item_rv_productos:

Este XML nos sirve para añadirlo al XML activity_productos y así mostrar los productos con un Checkbox para seleccionarlos, con una descripción y el nombre del producto con TextViews.

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/tvNomProducto"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:textColor="@color/black"
        android:hint="Nombre"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView

```

```
android:id="@+id/tvDescripcion"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginStart="8dp"
android:layout_marginLeft="8dp"
android:textColor="@color/black"
android:layout_marginEnd="8dp"
android:layout_marginRight="8dp"
android:hint="Descripcion"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/tvNomProducto" />
```

<CheckBox

```
android:id="@+id/cbCarro"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginStart="8dp"
android:layout_marginLeft="8dp"
android:layout_marginBottom="8dp"
android:text="Añadir al carro de compras"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/tvDescripcion" />
```

<TextView

```
android:id="@+id/tvPrecio"
android:layout_width="100dp"
android:layout_height="wrap_content"
android:layout_marginEnd="8dp"
android:layout_marginRight="8dp"
android:textColor="@color/black"
android:layout_marginBottom="8dp"
android:gravity="center"
android:hint="0.0"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintTop_toBottomOf="@+id/tvDescripcion" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```


item_rv_carro_compras:

Como en el anterior XML en este tenemos tres TextViews donde se presentarán el nombre del producto, la descripción y el precio.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/tvNomProducto"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:textColor="@color/black"
        android:hint="Nombre"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tvDescripcion"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:textColor="@color/black"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:hint="Descripcion"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tvNomProducto" />

    <TextView
        android:id="@+id/tvPrecio"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:textColor="@color/black"
        android:gravity="center"
        android:hint="0.0"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
```

```
app:layout_constraintTop_toBottomOf="@+id/tvDescripcion" />  
</androidx.constraintlayout.widget.ConstraintLayout>
```

Values

Dentro de la carpeta values he creado el archivo Array.xml donde guardaremos los arrays de los peluqueros disponibles, los horarios y los servicios.

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
  <string-array name="peluqueros">  
    <item>Borja</item>  
    <item>Ángel</item>  
  </string-array>  
  
  <string-array name="hora">  
    <item>9:00-10:00</item>  
    <item>10:00-11:00</item>  
    <item>11:00-12:00</item>  
  </string-array>  
  
  <string-array name="servicios">  
    <item>Corte de pelo</item>  
    <item>Corte de pelo y barba</item>  
    <item>Corte de pelo y lavado</item>  
    <item>Corte de pelo, barba y lavado</item>  
  </string-array>  
</resources>
```

google_maps_api.xml:

Como ya comente anteriormente para usar Google Maps se necesita este XML para poder usar ésta función.

```
<resources>
  <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:

    https://console.developers.google.com/flows/enableapi?
apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=ED:55:65:DC:EF:
A3:18:D1:F9:FF:E7:5E:64:6A:E4:D1:09:93:39:C8%3Bcom.example.proyecto

    You can also add your credentials to an existing key, using these values:

    Package name:
    com.example.proyecto

    SHA-1 certificate fingerprint:
    ED:55:65:DC:EF:A3:18:D1:F9:FF:E7:5E:64:6A:E4:D1:09:93:39:C8

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
  -->
  <string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">YOUR_KEY_HERE</string>
</resources>
```

9.2. Desarrollo de las clases Java

Una vez tenemos los XML con el diseño de la aplicación ahora pasaremos a realizar el código de las clases Java.

LoginActivity:

```
public class LoginActivity extends AppCompatActivity {

    Button btnLogin;
    EditText usuarioETLogin, passwordLoginET;

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

        btnLogin = findViewById(R.id.btnLogin);
        usuarioETLogin = findViewById(R.id.usuarioETLogin);
        passwordLoginET = findViewById(R.id.passwordLoginET);

        btnLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                validarUsuario("http://192.168.0.10/barber/validateuser.php");
            }
        });

        //Método para ir a pantalla Registro
        TextView registerLink = findViewById(R.id.TextViewRegistro);
        registerLink.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(LoginActivity.this, RegistroActivity.class));
            }
        });
    }

    //Método para validar que el usuario esté registrado
    private void validarUsuario(String URL) {
        StringRequest stringRequest = new StringRequest(
            Request.Method.POST,
            URL,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    if(!response.isEmpty()) {
                        Intent intent = new Intent(getApplicationContext(), HubActivity.class);

                        Bundle bundle = new Bundle();
                        bundle.putString("usuario", usuarioETLogin.getText().toString());
                        intent.putExtras(bundle);
                        startActivity(intent);
                    } else {
                        Toast.makeText(LoginActivity.this, "Usuario o contraseña incorrectos",
                            Toast.LENGTH_SHORT).show();
                    }
                }
            }
        );
    }
}
```

```

    }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Toast.makeText(LoginActivity.this, "Error al iniciar sesión", Toast.LENGTH_SHORT).show();
        }
    }
) {
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        Map<String, String> parametros = new HashMap<String, String>();
        parametros.put("usuario", usuarioETLogin.getText().toString());
        parametros.put("password", passwordLoginET.getText().toString());
        return parametros;
    }

};

RequestQueue requestQueue = Volley.newRequestQueue(this);
requestQueue.add(stringRequest);
}
}

```

Esta clase nos permite logearnos con nuestro usuario y contraseña. A través del método **validarUsuario** se comprobará que el usuario esté registrado, que los campos sean correctos y que estén cumplimentados.

Con **Volley** podremos mandar lo que obtengamos en los EditTexts a través de la URL de PHP que nos conectará con la Base de Datos y comprobará que los parámetros son correctos.

RegistroActivity:

```

public class RegistroActivity extends AppCompatActivity implements View.OnClickListener
{
    EditText nombreETRegistro, apellidosETRegistro, usuarioETRegistro, emailETRegistro,
    passwordETRegistro, phoneETResgistro;
    Button btnRegistrar;
    TextView TextViewInicio;

    RequestQueue requestQueue;

    //URL del PHP para registrar datos
    private static final String URL1 = "http://192.168.0.10/barber/registrar.php";

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

        requestQueue = Volley.newRequestQueue(this);
    }
}

```

```

//EditTexts
nombreETRegistro = findViewById(R.id.nombreETRegistro);
apellidosETRegistro = findViewById(R.id.apellidosETRegistro);
passwordETRegistro = findViewById(R.id.passwordETRegistrer);
phoneETResgistro = findViewById(R.id.phoneETRegistro);
emailETRegistro = findViewById(R.id.emailETRegistro);
usuarioETRegistro = findViewById(R.id.usuarioETRegistro);
TextViewInicio = findViewById(R.id.TextViewInicio);

//Buttons
btnRegistrer = findViewById(R.id.btnRegistrer);

btnRegistrer.setOnClickListener(this);
}

@Override
public void onClick(View v) {
    int id = v.getId();

    if (id == R.id.btnRegistrer) {
        String nombre = nombreETRegistro.getText().toString().trim();
        String apellidos = apellidosETRegistro.getText().toString().trim();
        String phone = phoneETResgistro.getText().toString().trim();
        String usuario = usuarioETRegistro.getText().toString().trim();
        String email = emailETRegistro.getText().toString().trim();
        String password = passwordETRegistro.getText().toString().trim();

        createUser(nombre, apellidos, usuario, password, email, phone);
    }
}

//Método para crear un nuevo Usuario
private void createUser(final String nombre, final String apellidos, final String usuario,
final String password, final String email, final String phone) {
    StringRequest stringRequest = new StringRequest(
        Request.Method.POST,
        URL1,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String ServerResponse) {
                Toast.makeText(RegistroActivity.this, ServerResponse,
                Toast.LENGTH_LONG).show();
                startActivity(new Intent(RegistroActivity.this, LoginActivity.class));
            }
        },
        new Response.ErrorListener() {
            @Override

```

```

        public void onResponse(VolleyError error) {

            Toast.makeText(RegistroActivity.this, "Error al registrar usuario",
                Toast.LENGTH_SHORT).show();

        }

    }

    ){
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            Map<String, String> params = new HashMap<>();
            //Tienen que ser el mismo nombre del param que en Visual Studios
            params.put("nombre", nombre);
            params.put("apellidos", apellidos);
            params.put("usuario", usuario);
            params.put("password", password);
            params.put("email", email);
            params.put("phone", phone);

            return params;
        }
    };

    requestQueue.add(stringRequest);
}

public void irInicio(View view) {
    Intent intent = new Intent(this, LoginActivity.class);
    startActivity(intent);
}
}

```

A través del método **createUser** podemos crear un nuevo usuario, para ello usaremos **getParams** y el método **put** para enviar los datos que previamente hemos recogido de los EditText

HubActivity:

```

public class HubActivity extends AppCompatActivity {

    Button btnUbicacion, btnCita, btnPerfil, btnProductos;
    TextView nomUsuario;

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

        btnUbicacion = findViewById(R.id.btnUbicacion);
        btnCita = findViewById(R.id.btnCita);
        btnPerfil = findViewById(R.id.btnPerfil);
        btnProductos = findViewById(R.id.btnProductos);
        nomUsuario = findViewById(R.id.nomUsuario);

        Bundle datoEnviado = getIntent().getExtras();
        String usuario = null;

        if(datoEnviado!=null){
            usuario = datoEnviado.getString("usuario");
            nomUsuario.setText(usuario);
        }

        //Al pulsar nos lleva a la pantalla Ubicación
        btnUbicacion.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(HubActivity.this, UbicacionActivity.class));
            }
        });

        //Al pulsar nos lleva a la pantalla Productos
        btnProductos.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(HubActivity.this, ProductosActivity.class));
            }
        });

        //Al pulsar nos lleva a la pantalla Citas
        btnCita.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), PeluqueroActivity.class);
                Bundle bundle = new Bundle();
                bundle.putString("usuario", nomUsuario.getText().toString());
                intent.putExtras(bundle);
            }
        });
    }
}

```



```

        startActivity(intent);
    }
});

//Nos lleva a la pantalla Perfil
btnPerfil.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getApplicationContext(), PerfilActivity.class);
        Bundle bundle = new Bundle();
        bundle.putString("usuario", nomUsuario.getText().toString());
        intent.putExtras(bundle);
        startActivity(intent);
    }
});
}
}

```

Esta clase lo que nos permite es clicar en uno de los botones que nos llevará a la pantalla correspondiente.

Esto lo he realizado a través de un Intent con la clase de la pantalla a la que queremos ir y con el método startActivity se nos llevará a la pantalla correspondiente.

PeluqueroActivity:

```

public class PeluqueroActivity extends AppCompatActivity implements
View.OnClickListener {

    DatePickerDialog datePickerDialog;
    Button btnSelectPelu;
    EditText btnFecha;
    Spinner spinnerPelu, spinnerHoras, spinnerServicio;
    TextView textView, nomUsuario;

    RequestQueue requestQueue;

    //URL del PHP para registrar datos
    private static final String URL2 = "http://192.168.0.10/barber/cita.php";

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

        //Spinners
        spinnerPelu = findViewById(R.id.spinnerPelu);
        spinnerHoras = findViewById(R.id.spinnerHoras);
        spinnerServicio = findViewById(R.id.spinnerServicio);
    }
}

```

```
//Buttons
btnSelectPelu = findViewById(R.id.btnSelectPelu);
btnFecha = findViewById(R.id.btnFecha);

//TextViews
textView = findViewById(R.id.textView);
nomUsuario = findViewById(R.id.nomUsuario);

//Array para los peluqueros
ArrayAdapter<CharSequence> adapter=ArrayAdapter.createFromResource(this,
    R.array.peluqueros, android.R.layout.simple_spinner_dropdown_item);

spinnerPelu.setAdapter(adapter);

//Array para los horarios
ArrayAdapter<CharSequence> adapter2=ArrayAdapter.createFromResource(this,
    R.array.hora, android.R.layout.simple_spinner_dropdown_item);

spinnerHoras.setAdapter(adapter2);

//Array para los servicios
ArrayAdapter<CharSequence> adapter3=ArrayAdapter.createFromResource(this,
    R.array.servicios, android.R.layout.simple_spinner_dropdown_item);

spinnerServicio.setAdapter(adapter3);

Bundle datoEnviado = getIntent().getExtras();
String usuario = null;

if(datoEnviado!=null){
    usuario = datoEnviado.getString("usuario");
    nomUsuario.setText(usuario);
}

requestQueue = Volley.newRequestQueue(this);

btnSelectPelu.setOnClickListener(this);

//Mostrar el día de hoy
btnFecha.setText(getTodaysDate());

initDatePicker();

textView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(PeluqueroActivity.this, LoginActivity.class));
    }
});
```

```

    });

}

//Para que muestre el día actual
private String getTodaysDate() {
    Calendar calendar = Calendar.getInstance();
    int year = calendar.get(Calendar.YEAR);
    int month = calendar.get(Calendar.MONTH);
    month = month + 1;
    int day = calendar.get(Calendar.DAY_OF_MONTH);
    return makeDateString(day, month, year);
}

//Método para que nos aparezca el calendario a nuestro gusto
private void initDatePicker() {

    DatePickerDialog.OnDateSetListener dateSetListener = new
    DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int month, int day) {
            month = month + 1;
            String date = makeDateString(day, month, year);
            btnFecha.setText(date);
        }
    };

    Calendar calendar = Calendar.getInstance();
    int year = calendar.get(Calendar.YEAR);
    int month = calendar.get(Calendar.MONTH);
    int day = calendar.get(Calendar.DAY_OF_MONTH);

    int style = AlertDialog.THEME_HOLO_LIGHT;

    datePickerDialog = new DatePickerDialog(this, style, dateSetListener, year, month,
    day);

    //Para que empiece por la fecha actual y no permita escoger anteriores
    datePickerDialog.getDatePicker().setMinDate(calendar.getTimeInMillis());
}

//Convierte la fecha en String
private String makeDateString(int day, int month, int year) {
    return getMonthFormat(month) + " " + day + " " + year;
}

//Convierte los números de los meses en Escrito
private String getMonthFormat(int month)
{
    if(month == 1)
        return "ENERO";
    if(month == 2)
        return "FEBRERO";
}

```

```
if(month == 3)
    return "MARZO";
if(month == 4)
    return "ABRIL";
if(month == 5)
    return "MAYO";
if(month == 6)
    return "JUNIO";
if(month == 7)
    return "JULIO";
if(month == 8)
    return "AGOSTO";
if(month == 9)
    return "SEPTIEMBRE";
if(month == 10)
    return "OCTUBRE";
if(month == 11)
    return "NOVIEMBRE";
if(month == 12)
    return "DICIEMBRE";

//Por defecto saldrá Enero
return "ENERO";
}

//Abre el calendario para seleccionar fecha
public void openDatePicker(View view) {

    datePickerDialog.show();

}

//Obtenemos los parámetros
@Override
public void onClick(View v) {
    int id = v.getId();

    if (id == R.id.btnSelectPelu) {
        String peluquero = spinnerPelu.getSelectedItem().toString().trim();
        String fecha = btnFecha.getText().toString().trim();
        String hora = spinnerHoras.getSelectedItem().toString().trim();
        String servicio = spinnerServicio.getSelectedItem().toString().trim();

        //Obtenemos el nombre del usuario
        Bundle datoEnviado = getIntent().getExtras();
        String usuario = null;

        if(datoEnviado!=null){
            usuario = datoEnviado.getString("usuario");
        }
    }
}
```

```

        createCita(peluquero, fecha, hora, servicio, usuario);
    }
}

//Creamos la cita
private void createCita(final String peluquero, final String fecha, final String hora, final
String servicio, final String usuario) {
    StringRequest stringRequest = new StringRequest(
        Request.Method.POST,
        URL2,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                Toast.makeText(PeluqueroActivity.this, response,
                Toast.LENGTH_SHORT).show();
            }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Toast.makeText(PeluqueroActivity.this, "Hubo un error al registrar la cita",
                Toast.LENGTH_SHORT).show();
            }
        }
    );
}

){
    //Enviamos los datos a la BD
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        Map<String, String> params = new HashMap<>();
        //Tienen que ser el mismo nombre del param que en Visual Studios
        params.put("cita", peluquero + " " + hora + " " + fecha);
        params.put("servicio", servicio);
        params.put("username", usuario);

        return params;
    }
};

requestQueue.add(stringRequest);
}
}

```

Con el método **createCita** registramos la cita, que al igual que con el método **createUser**, obtenemos los datos de los EditText y los enviamos con **getParams** y el método **put**.

A través del método **getMonthFormat** le daremos formato al **Dialog** de fechas, en mi caso he decidido ponerlo con su nombre y no con su número.

El método **getTodayDate** nos permite obtener el día actual para que así comience el calendario justo en ese momento.

PerfilActivity:

```
public class PerfilActivity extends AppCompatActivity implements View.OnClickListener{

    TextView introduzcaView, cit, serv;
    Button btnCancelar;
    String usuario;

    RequestQueue requestQueue;

    //URL del PHP para registrar datos
    private static final String URL = "http://192.168.0.10/barber/buscarcita.php";

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

        //TextViews
        introduzcaView = findViewById(R.id.introduzcaView);
        cit = findViewById(R.id.cita);
        serv = findViewById(R.id.servicio);

        //Buttons
        btnCancelar = findViewById(R.id.btnCancelar);

        //Volley
        requestQueue = Volley.newRequestQueue(this);

        //Obtenemos el nombre de usuario del otro activity
        Bundle extras = getIntent().getExtras();
        if (extras != null) {
            usuario = extras.getString("usuario");
        }

        verCita();

        btnCancelar.setOnClickListener(new View.OnClickListener() {
            @Override
```

```

        public void onClick(View v) {
            cancelarCita(usuario);

        }
    });
}

//Método para visualizar la cita
private void verCita(){
    String URL = "http://192.168.0.10/barber/buscarcitas.php?username="+usuario;
    JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(
        Request.Method.GET,
        URL,
        null,
        new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                String cita, servicio;
                try {
                    cita = response.getString("cita");
                    servicio = response.getString("servicio");

                    cit.setText(cita);
                    serv.setText(servicio);

                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {

            }
        }
    );

    //Solicitud al RequestQueue
    requestQueue.add(jsonObjectRequest);
}

//Cuando clicquemos en el botón cancelar, se borra la cita
@Override
public void onClick(View v) {
    int id = v.getId();

    if (id == R.id.btnCancelar) {

```

```

        //Obtenemos el nombre del usuario
        Bundle datoEnviado = getIntent().getExtras();
        String usuario = null;

        if(datoEnviado!=null){
            usuario = datoEnviado.getString("usuario");
        }

        cancelarCita(usuario);
    }
}

//Método para que el usuario pueda cancelar su cita
private void cancelarCita(final String user) {
    String URL2 = "http://192.168.0.10/barber/cancelar.php";
    StringRequest stringRequest = new StringRequest(
        Request.Method.POST,
        URL2,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                Toast.makeText(PerfilActivity.this, response, Toast.LENGTH_LONG).show();
                //Limpiamos los TextViews
                cit.setText("No tienes ninguna cita pendiente");
                serv.setText("");
            }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Toast.makeText(PerfilActivity.this, "Hubo un error y no se pudo cancelar",
                Toast.LENGTH_LONG).show();
            }
        }
    ){
        protected Map<String, String> getParams() throws AuthFailureError {
            Map<String, String> params = new HashMap<>();
            params.put("username", user);
            return params;
        }
    };

    requestQueue.add(stringRequest);
}
}

```


En esta clase, a través del objeto **Bundle** traemos el dato referente al nombre de usuario, lo que haremos con éste dato será colocarlo en un **TextView**.

Con el método **verCita** obtenemos los datos de la cita y el servicio del usuario logeado.

El método **cancelarCita** nos permite borrar la cita de la Base de datos.

ProductosActivity:

```
public class ProductosActivity extends AppCompatActivity {

    TextView tvCantProductos;
    Button btnVerCarro;
    RecyclerView rvListaProductos;
    AdaptadorProductos adaptador;

    List<Producto> listaProductos = new ArrayList<>();
    List<Producto> carroCompras = new ArrayList<>();

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

        tvCantProductos = findViewById(R.id.tvCantProductos);
        btnVerCarro = findViewById(R.id.btnVerCarro);
        rvListaProductos = findViewById(R.id.rvListaProductos);
        rvListaProductos.setLayoutManager(new GridLayoutManager(ProductosActivity.this,
1));

        listaProductos.add(new Producto("1", "Cera", "Fijador para el pelo", 10.0));
        listaProductos.add(new Producto("2", "Aceite para barba", "Aceite para suavizar la
barba", 20.0));
        listaProductos.add(new Producto("3", "Corte de pelo", "Podrá seleccionar hora y fecha",
15.0));
        listaProductos.add(new Producto("4", "Corte de pelo y barba", "Podrá seleccionar hora y
fecha", 25.0));
        listaProductos.add(new Producto("5", "Corte de barba", "Podrá seleccionar hora y
fecha", 15.0));

        adaptador = new AdaptadorProductos(ProductosActivity.this, tvCantProductos,
btnVerCarro, listaProductos, carroCompras);
        rvListaProductos.setAdapter(adaptador);
    }
}
```

Esta clase obtiene los datos de la clase Productos, y se crea un Adaptador para unir los datos. Se visualizarán los productos de los ArrayLists que añadiremos en ésta misma clase.

AdaptadorProductos:

```

public class AdaptadorProductos extends RecyclerView.Adapter<AdaptadorProductos.ProductosViewHolder> {

    Context context;
    TextView tvCantProductos;
    Button btnVerCarro;
    List<Producto> listaProductos;
    List<Producto> carroCompra;

    public AdaptadorProductos(Context context, TextView tvCantProductos, Button btnVerCarro,
    List<Producto> listaProductos, List<Producto> carroCompra) {
        this.context = context;
        this.tvCantProductos = tvCantProductos;
        this.btnVerCarro = btnVerCarro;
        this.listaProductos = listaProductos;
        this.carroCompra = carroCompra;
    }

    @NonNull
    @Override
    public ProductosViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        View v = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.item_rv_productos, null, false);
        return new AdaptadorProductos.ProductosViewHolder(v);
    }

    @Override
    public void onBindViewHolder(@NonNull final ProductosViewHolder productosViewHolder, final int i) {
        productosViewHolder.tvNomProducto.setText(listaProductos.get(i).getNomProducto());
        productosViewHolder.tvDescripcion.setText(listaProductos.get(i).getDescripcion());
        productosViewHolder.tvPrecio.setText(""+listaProductos.get(i).getPrecio());

        productosViewHolder.cbCarro.setOnCheckedChangeListener(new
        CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                if(productosViewHolder.cbCarro.isChecked() == true) {
                    tvCantProductos.setText(""+(Integer.parseInt(tvCantProductos.getText().toString().trim()) + 1));
                    carroCompra.add(listaProductos.get(i));
                } else if(productosViewHolder.cbCarro.isChecked() == false) {
                    tvCantProductos.setText(""+(Integer.parseInt(tvCantProductos.getText().toString().trim()) - 1));
                    carroCompra.remove(listaProductos.get(i));
                }
            }
        });

        //Al pulsar el Button Ver Carro nos enviará al pantalla CarroCompras
        btnVerCarro.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(context, CarroCompra.class);
                intent.putExtra("CarroCompras", (Serializable) carroCompra);
                context.startActivity(intent);
            }
        });
    }

    //Cuenta el total de productos
    @Override
    public int getItemCount() {

```

```

        return listaProductos.size();
    }

    public class ProductosViewHolder extends RecyclerView.ViewHolder {

        TextView tvNomProducto, tvDescripcion, tvPrecio;
        CheckBox cbCarro;

        public ProductosViewHolder(@NonNull View itemView) {
            super(itemView);

            tvNomProducto = itemView.findViewById(R.id.tvNomProducto);
            tvDescripcion = itemView.findViewById(R.id.tvDescripcion);
            tvPrecio = itemView.findViewById(R.id.tvPrecio);
            cbCarro = itemView.findViewById(R.id.cbCarro);
        }
    }
}

```

Con esta clase se nos muestran los datos de los productos como son el nombre, la descripción y el precio. También se contará el total de los productos que hemos seleccionado, a través del método **getItemCount**.

AdaptadorCarroCompras:

```

public class AdaptadorCarroCompras extends
RecyclerView.Adapter<AdaptadorCarroCompras.ProductosViewHolder> {

    Context context;
    List<Producto> carroCompra;
    TextView tvTotal;
    double total = 0;

    public AdaptadorCarroCompras(Context context, List<Producto> carroCompra, TextView
tvTotal) {
        this.context = context;
        this.carroCompra = carroCompra;
        this.tvTotal = tvTotal;

        for(int i = 0 ; i < carroCompra.size() ; i++) {
            total = total + Double.parseDouble(""+carroCompra.get(i).getPrecio());
        }

        tvTotal.setText(""+total);
    }

    @NonNull
    @Override
    public ProductosViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int
i) {
        View v =
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.item_rv_carro_compras, null,

```

```

false);
    return new AdaptadorCarroCompras.ProductosViewHolder(v);
}

@Override
public void onBindViewHolder(@NonNull final ProductosViewHolder
productosViewHolder, final int i) {
    productosViewHolder.tvNomProducto.setText(carroCompra.get(i).getNomProducto());
    productosViewHolder.tvDescripcion.setText(carroCompra.get(i).getDescripcion());
    productosViewHolder.tvPrecio.setText(""+carroCompra.get(i).getPrecio());

}

@Override
public int getItemCount() {
    return carroCompra.size();
}

public class ProductosViewHolder extends RecyclerView.ViewHolder {

    TextView tvNomProducto, tvDescripcion, tvPrecio;

    public ProductosViewHolder(@NonNull View itemView) {
        super(itemView);

        tvNomProducto = itemView.findViewById(R.id.tvNomProducto);
        tvDescripcion = itemView.findViewById(R.id.tvDescripcion);
        tvPrecio = itemView.findViewById(R.id.tvPrecio);
    }
}
}

```

Con el método **AdaptadorCarroCompras** veremos los productos y el precio, por último se verá el precio final que se sumará con el parámetro **tvTotal**.

Para obtener la cantidad de cada producto y su precio lo obtenemos con el método **onBindViewHolder**.

UbicacionActivity:

```
public class UbicacionActivity extends AppCompatActivity implements
OnMapReadyCallback {

    private GoogleMap mMap;
    VideoView videoView;

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

        //El fragment para visualizar Google Maps
        SupportMapFragment mapFragment = (SupportMapFragment)
        getSupportFragmentManager()
            .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    //Método para ajustar Google Maps
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        LatLng barber = new LatLng(36.548342029303136, -4.628322789650115);
        mMap.addMarker(new MarkerOptions().position(barber).title("BarberShop"));
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(barber, 18));
        mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
    }
}
```

Con el objeto **SupportMapFragment** visualizaremos el Mapa de Google y con el método **onMapReady** podemos ajustar la ubicación con las coordenadas, el zoom que queremos y el tipo de mapa que mostraremos.

Producto:

```
public class Producto implements Serializable {

    String idProducto;
    String nomProducto;
    String descripcion;
    double precio;

    public Producto(String idProducto, String nomProducto, String descripcion, double precio)
    {
        this.idProducto = idProducto;
        this.nomProducto = nomProducto;
        this.descripcion = descripcion;
        this.precio = precio;
    }

    public String getIdProducto() {
        return idProducto;
    }

    public void setIdProducto(String idProducto) {
        this.idProducto = idProducto;
    }

    public String getNomProducto() {
        return nomProducto;
    }

    public void setNomProducto(String nomProducto) {
        this.nomProducto = nomProducto;
    }

    public String getDescripcion() {
        return descripcion;
    }

    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }

    public double getPrecio() {
        return precio;
    }

    public void setPrecio(double precio) {
        this.precio = precio;
    }
}
```

Esta clase **representa a los productos**, para poder obtener sus atributos como son el precio, nombre, descripción e ID.

AndroidManifest:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.proyecto">
    <!--
        The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
        Google Maps Android API v2, but you must specify either coarse or fine
        location permissions for the "MyLocation" functionality.
    -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/Theme.Proyecto"
        android:usesCleartextTraffic="true">
        <activity android:name=".PerfilActivity"></activity>
        <activity android:name=".CarroCompra" />
        <activity android:name=".ProductosActivity" />
        <activity android:name=".UbicacionActivity" />
        <!--
            The API key for Google Maps-based APIs is defined as a string resource.
            (See the file "res/values/google_maps_api.xml").
            Note that the API key is linked to the encryption key used to sign the APK.
            You need a different API key for each encryption key, including the release key that
            is used to
            sign the APK for publishing.
            You can define the keys for the debug and release targets in src/debug/ and
            src/release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyBv1cu4-8Uf2WKfASh-Tq9I1Bd2Dtvh968" />
        <meta-data
            android:name="com.google.android.gms.ads.APPLICATION_ID"
            android:value="ca-app-pub-6591920118654710~4540089974" />

        <activity android:name=".HubActivity" />
        <activity android:name=".PeluqueroActivity" />
        <activity android:name=".LoginActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

```

```

    </activity>
    <activity android:name=".RegistroActivity" />
</application>

</manifest>

```

En este archivo daremos permiso para que la aplicación se conecte a Internet y tenga acceso a la localización. También añadimos la **API** de Google Maps, para poder usarlo.

Build.gradle:

```

plugins {
    id 'com.android.application'
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.example.proyecto"
        minSdkVersion 22
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'androidx.gridlayout:gridlayout:1.0.0'
}

```



```
//Tener acceso a los servicios de Google
implementation 'com.google.android.gms:play-services-ads:20.1.0'
implementation 'com.google.android.gms:play-services-maps:17.0.1'
testImplementation 'junit:junit:4.13.2'
androidTestImplementation 'androidx.test.ext:junit:1.1.2'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
implementation 'com.pierfrancescosoffritti.androidyoutubeplayer:core:10.0.5'
implementation "androidx.recyclerview:recyclerview:1.2.0"
// For control over item selection of both touch and mouse driven selection
implementation "androidx.recyclerview:recyclerview-selection:1.1.0"

//Volley
implementation 'com.android.volley:volley:1.2.0'
}
```

En el archivo **build.gradle app** implementamos **Volley** para tener una conexión con **PHP** y la Base de Datos **MYSQL**. También se han implementado los servicios de **Google**, para poder tener acceso a **Google Maps**.

9.2. Desarrollo de los archivos PHP

Para poder conectarnos a la **Base de Datos** desde la aplicación necesitamos los archivos **PHP** que harán de intermediarios.

Dbbarber.php:

```
<?php

$mysql = new mysqli(
    "localhost",
    "root",
    "",
    "barber"
);

if ($mysql->connect_error){
    die("Error al conectarse" . $mysql->connect_error);
}
```

Figura 13: dbbarber.php

Este archivo nos va a permitir conectarnos a la Base de Datos.

Register.php:

```

<?php

require_once("dbbarber.php");

$nombre = $_POST['nombre'];
$apellidos = $_POST['apellidos'];
$usuario = $_POST['usuario'];
$password = $_POST['password'];
$email = $_POST['email'];
$phone = $_POST['phone'];

$query = "INSERT INTO usuarios (nombre, apellidos, usuario, password, email, phone)
VALUES ('$nombre', '$apellidos', '$usuario', '$password', '$email', '$phone')";

$verificar_correo = mysqli_query($mysql, "SELECT * FROM usuarios
WHERE email='$email' or usuario='$usuario'");

if(mysqli_num_rows($verificar_correo) > 0) {
    echo "ERROR! El email o usuario ya está en uso";

    exit();
}

if(empty($nombre) || empty($apellidos) || empty($usuario)
|| empty($password) || empty($email) || empty($phone)) {

    // SI ALGUNA VARIABLE ESTA VACIA MUESTRA ERROR
    //echo "Se deben llenar los dos campos";
    echo "ERROR! Alguno de los campos está vacío";

} else {

    $query = "INSERT INTO usuarios (nombre, apellidos, usuario, password, email, phone)
VALUES ('$nombre', '$apellidos', '$usuario', '$password', '$email', '$phone')";
$result = $mysql->query($query);

    if($result === TRUE) {

        echo "Registrado con éxito";

    }

}

mysqli_close($mysql);
?>

```

Comprobamos si los campos que nos llega a la base están vacíos, si lo están se mostrará un mensaje de que alguno de los campos está vacío. También se chequeará que el usuario o el email ya estén registrados, si fuera el caso, se mostrará el mensaje de error.

Con la sentencia **INSERT INTO** registraremos los datos en al **Base de Datos**. En caso de que no haya ningún error se mostrará el mensaje de “Registrado con éxito”.

Validateuser.php:

```

<?php
    include 'dbbarber.php';
    $usuario= $_POST['usuario'];
    $password= $_POST['password'];

    $sentencia=$mysql->prepare("SELECT * FROM usuarios WHERE usuario=? AND password=?");
    $sentencia->bind_param('ss',$usuario, $password);
    $sentencia->execute();

    $resultado = $sentencia->get_result();
    if ($fila = $resultado->fetch_assoc()) {
        echo json_encode($fila,JSON_UNESCAPED_UNICODE);
    }
    $sentencia->close();
    $mysql->close();
?>

```

La sentencia **SELECT – WHERE** comprobará que los parámetros usuario y password que llegue desde la aplicación estén dentro de la Base de Datos, en caso correcto se iniciará sesión.

Cita.php:

```

<?php

if($_SERVER['REQUEST_METHOD'] == 'POST'){

    require_once("dbbarber.php");

    $cita = $_POST['cita'];
    $servicio = $_POST['servicio'];
    $username = $_POST['username'];

    $query = "INSERT INTO citas (cita, servicio, username) VALUES ('$cita', '$servicio', '$username')";

    $verificar_cita = mysqli_query($mysql, "SELECT * FROM citas WHERE cita='$cita'");

    if(mysqli_num_rows($verificar_cita) > 0) {
        echo "ERROR! Esa cita no está disponible";

        exit();
    }

    $query = "INSERT INTO citas (cita, servicio, username) VALUES ('$cita', '$servicio', '$username')";
    $result = $mysql->query($query);

    if ($result === TRUE){
        echo "La cita se creó correctamente";
    }else{
        echo "Error al crear la cita";
    }

    $mysql->close();
}

```

Se verificará que la cita no esté repetida a través de la sentencia **SELECT – WHERE**, donde si la cita es igual se nos mostrará un mensaje de error.

En caso de que no esté ya registrada la cita usaremos la sentencia **INSERT INTO** para registrar los datos de la cita que nos lleguen desde la aplicación.

Buscarcitas.php:

```
<?php
if($_SERVER['REQUEST_METHOD'] == 'GET'){

    require_once("dbbarber.php");

    $username = $_GET['username'];

    $query ="SELECT cita, servicio FROM citas WHERE username = '$username'";
    $result = $mysql->query($query);

    if ($mysql->affected_rows > 0){
        while($row = $result->fetch_assoc()){
            $array = $row;
        }

        echo json_encode($array);
    }else{
        echo "No se encontró ninguna columna";
    }

    $result->close();
    $mysql->close();
}
```

Figura 17: buscarcita.php

Obtenemos los datos de la cita a través del nombre de usuario con la sentencia **SELECT – WHERE**.

Cancelar.php:

```
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {

    require_once("dbbarber.php");
    $username = $_POST['username'];

    $query = "DELETE FROM citas WHERE username = '$username'";
    $result = $mysql->query($query);

    if ($mysql->affected_rows > 0) {
        if ($result == TRUE){
            echo "Se canceló su cita correctamente";
        }
    }else{
        echo "No se canceló, porque no tienes ninguna cita pendiente";
    }

    $mysql->close();
}
```

Figura 18: cancelar.php

Cancelaremos la cita usando el nombre de usuario y la sentencia **DELETE – WHERE**, se mostrará un mensaje de que se canceló la cita correctamente y si no es el caso, se nos mostrará un mensaje de error en caso de que no hay ninguna cita.

10. Manual de Usuario

1. La primera pantalla que aparecerá será la de Login:

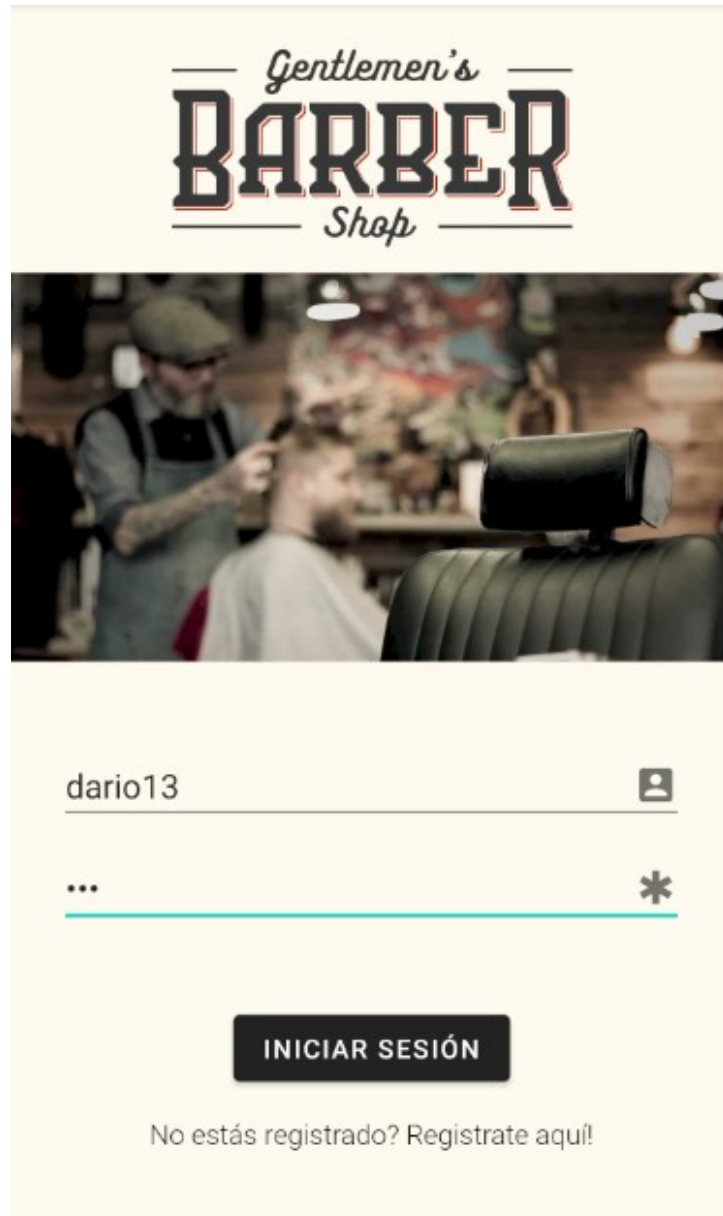


Figura 19: Prueba Login

Se colocará el nombre de usuario y la contraseña, una vez colocados se clicará en el botón Iniciar Sesión y se nos dirige a la siguiente pantalla.

En caso de no estar registrado, se clicla en “*No estás registrado? Regístrate aquí!*” y se pasará a la pantalla de registro.



Figura 20: Usuario o contraseña incorrectos

Se mostrará un mensaje de error en caso de no estar registrado o alguno de los datos sea erróneo.

2. En la pantalla de registro hay que rellenar los campos Nombre, Apellidos, Nombre de usuario, Contraseña, Email y Teléfono. Si alguno de los campos no ha sido rellenado se mostrará el siguiente mensaje:

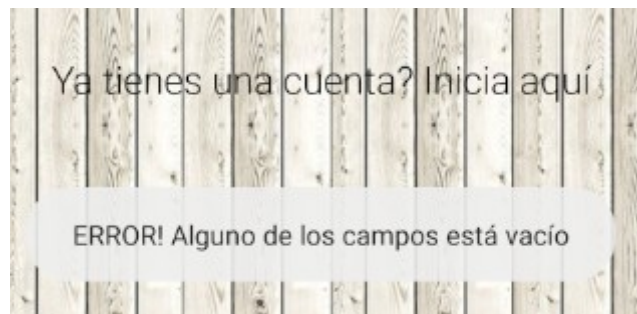


Figura 21: Alguno de los campos está vacío

Si el usuario o email ya está registrado se mostrará un mensaje de error como el que aparece en la siguiente imagen:

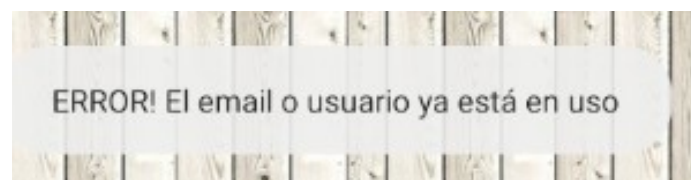
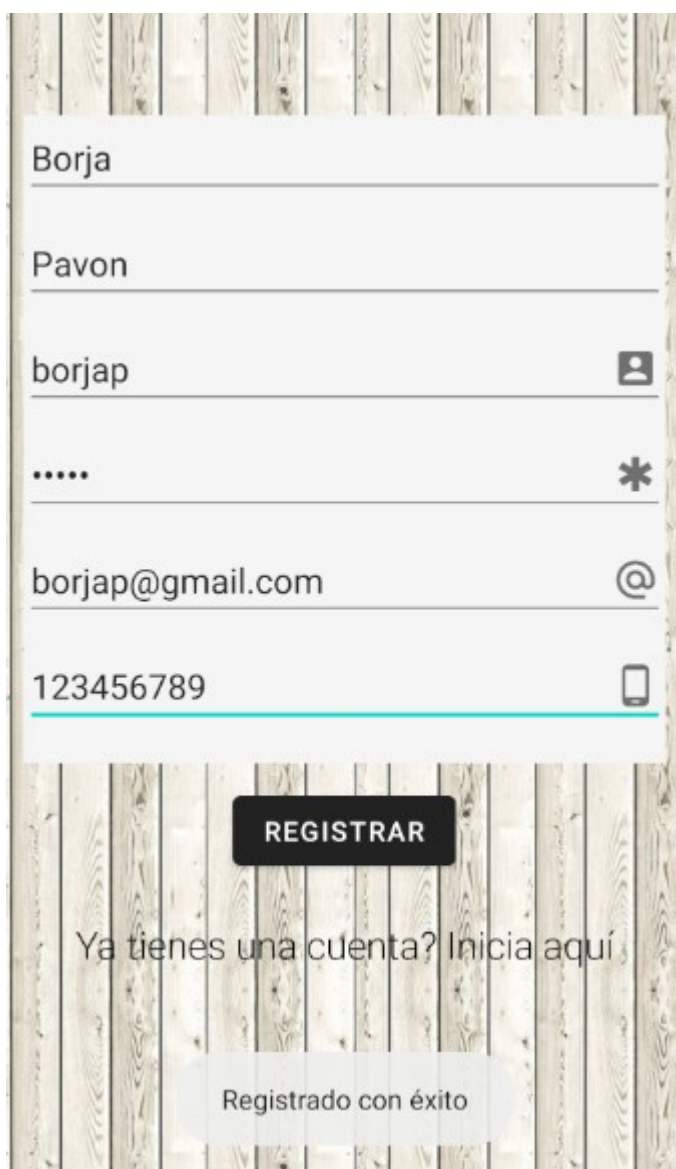


Figura 22: El email o usuario ya está en uso

Si todos los campos han sido rellenos y no están repetidos se registrará el nuevo usuario.



Borja

Pavon

borjap

....

borjap@gmail.com

123456789

REGISTRAR

Ya tienes una cuenta? Inicia aquí

Registrado con éxito

Figura 23: Registrado con éxito

3. Una vez logeado el usuario se nos dirige a una pantalla intermedia, donde se permite seleccionar la acción que queremos realizar.



Figura 24: Se muestra el nombre de usuario

4. Si se clicca en el botón Pedir Cita se seleccionará el peluquero que se quiera, el tipo de servicio, el horario y la fecha de la cita.

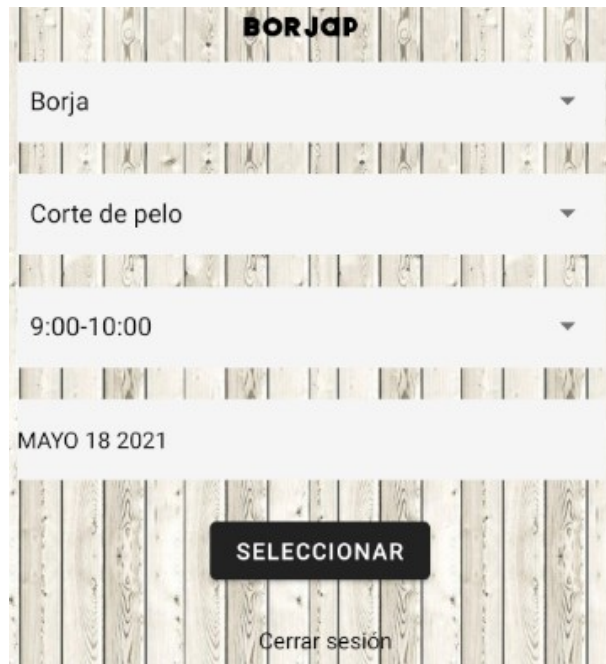


Figura 25: Selección de cita

Una vez se clique en el botón Seleccionar se registrará la cita.

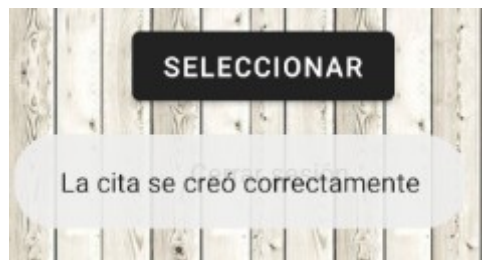


Figura 26: La cita se creó correctamente

En el caso de que la cita no esté disponible se nos mostrará un mensaje de error:



Figura 27: Esa cita no está disponible

5. Si se selecciona Ver reservas se mostrará la fecha, hora, el peluquero y el servicio seleccionado.



Figura 28: Se muestra la cita

Si se quiere cancelar la cita, se pulsará el botón Cancelar cita. Si hubiera una cita se mostrará los siguiente:



Figura 29: Se canceló correctamente

Al no haber cita y clicar en botón de Cancelar se muestra el mensaje “*No se canceló, porque no tienes ninguna cita pendiente*”.

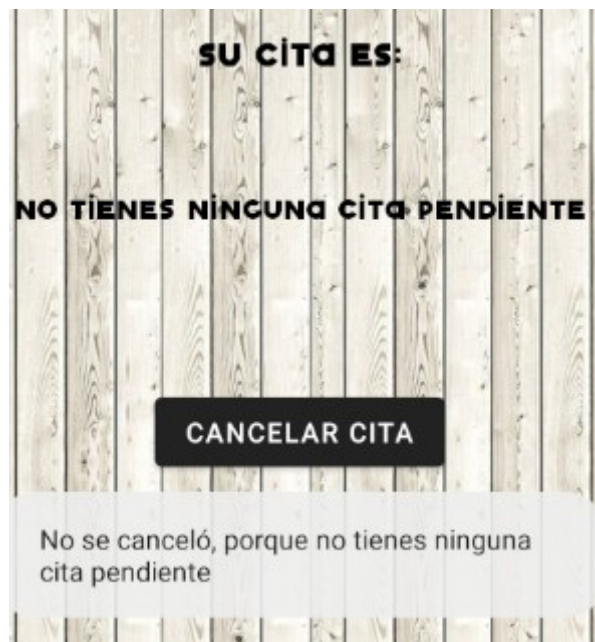
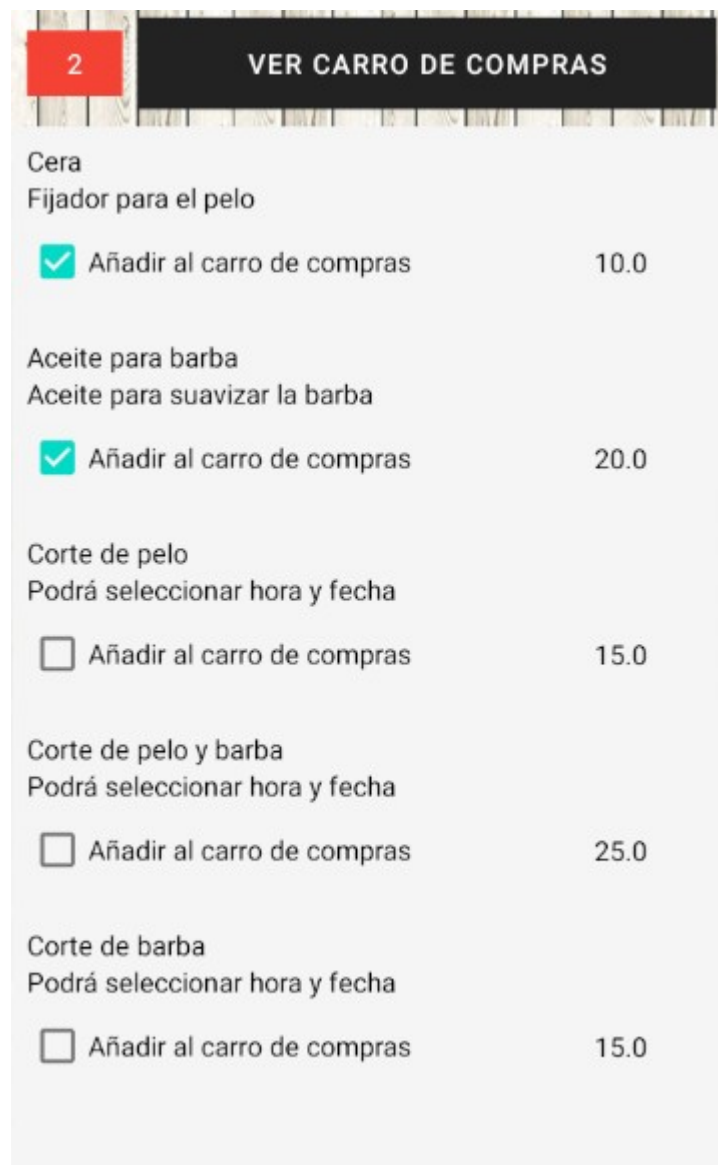


Figura 30: No se canceló

6. Seleccionando Comprar productos aparecerá una lista de productos que podemos seleccionar, si se selecciona algún producto se añadirá al contador.



2 VER CARRO DE COMPRAS	
Cera Fijador para el pelo	
<input checked="" type="checkbox"/> Añadir al carro de compras	10.0
Aceite para barba Aceite para suavizar la barba	
<input checked="" type="checkbox"/> Añadir al carro de compras	20.0
Corte de pelo Podrá seleccionar hora y fecha	
<input type="checkbox"/> Añadir al carro de compras	15.0
Corte de pelo y barba Podrá seleccionar hora y fecha	
<input type="checkbox"/> Añadir al carro de compras	25.0
Corte de barba Podrá seleccionar hora y fecha	
<input type="checkbox"/> Añadir al carro de compras	15.0

Figura 31: Selección de productos

Clicando en el botón Ver carro de compras se mostrará una nueva pantalla donde aparecerán los productos seleccionados y el precio total.

[illegible]

Figura 32: Productos seleccionados

7. La última opción a elegir sería Ubicación, que mostrará un vídeo de la peluquería y la ubicación de ésta misma en un mapa de Google Maps.



Figura 33: Ubicación

11. Puesta en marcha

Al tratarse de una aplicación móvil hay varias formas de ponerla en marcha, una de ellas es subirla a una plataforma como Google Play o a una página web desde el cual se pueda descargar.

Primero debemos prepararla para el lanzamiento. Para ello, debemos configurar la app quitando las llamadas de Log y el atributo **android:debuggable** del archivo de manifiesto. Se deberá proporcionar los valores para los atributos **android:versionCode** y **android:versionName**.

Compilaremos y firmaremos una versión de actualización de la app. Seguidamente se probará antes de distribuirla.

Se prepararán los servidores y servicios remotos de los que depende la aplicación.

Una vez esté preparada para su lanzamiento debemos, podemos lanzarla a Google Play donde tendremos que publicar la versión de lanzamiento de la aplicación. También deberemos decidir a que público está dirigida y en que países se va a publicar. Google Play nos permite realizar promociones con capturas o vídeos de la propia aplicación.

Una vez publicada nuestra aplicación en Google Play o en una página web, se descargará y Android se encargará de instalarla en nuestro dispositivo.

En el caso de ésta aplicación se necesitará cambiar las URLs, ya que usamos el localhost. A mejorar sería obtener un servidor para que cualquier persona pudiera usar la aplicación si necesidad de cambiar las URLs.

Cuando la aplicación esté instalada se pulsará en el icono de la aplicación y se abrirá la pantalla de Login.

Para ello en Android Studios iremos a Build/Generate Signed Bundle/APK.

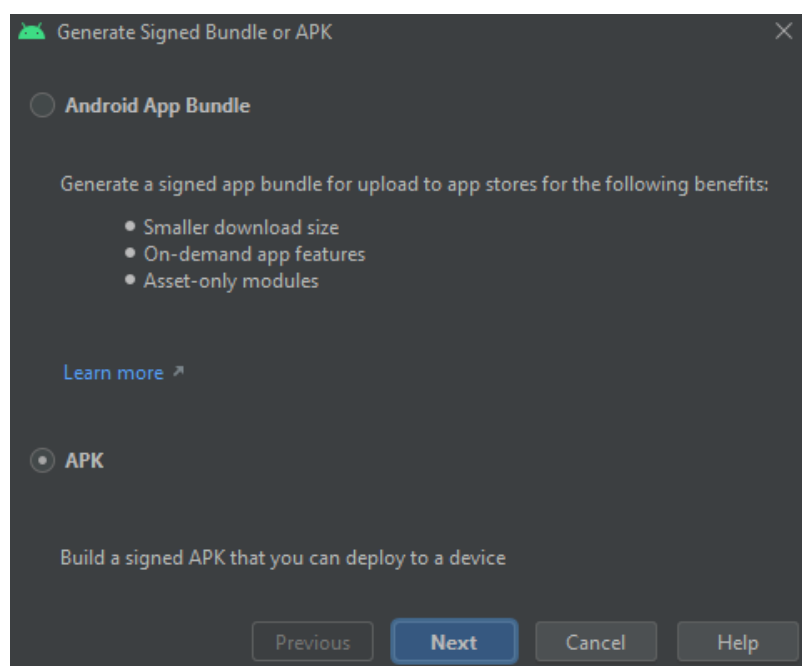


Figura 34: Generar APK

Nos aparecerá una nueva pantalla, donde se puede seleccionar para crear una APK o directamente un Bundle para subir a Google Play. En mi caso elijo APK ya que no la voy a subir a Google de momento.

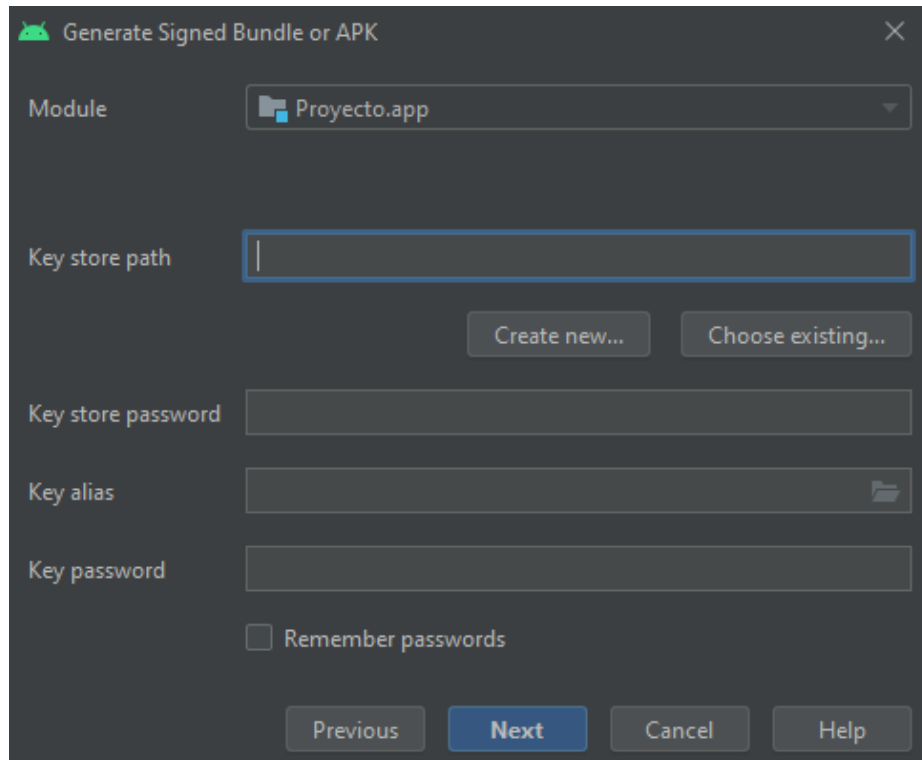


Figura 35: Generar Key store

Se pulsa en el botón Create new para crear una nueva Key store y aparece una nueva pantalla donde debemos rellenar los siguientes campos:

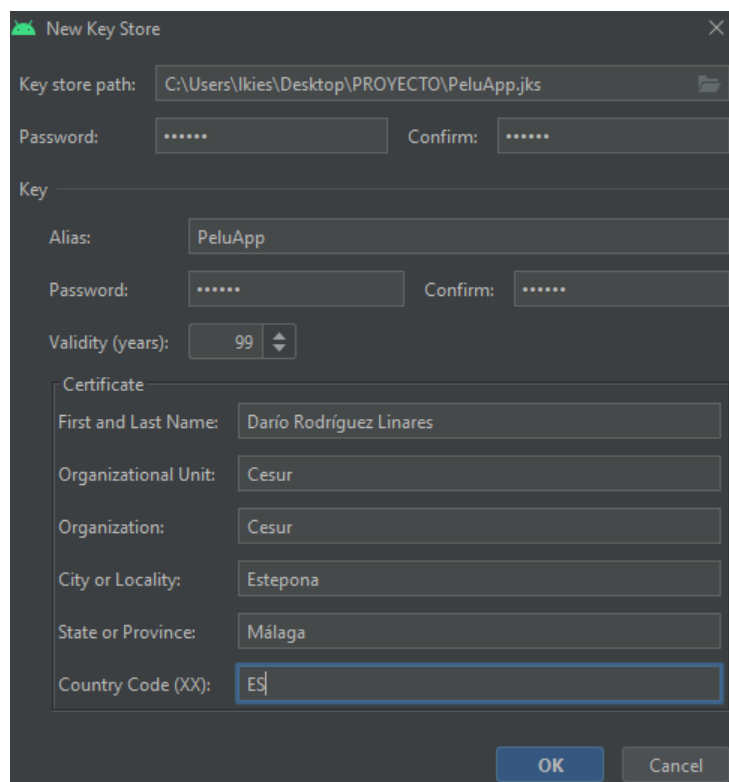


Figura 36: Rellenar los datos para la Key store

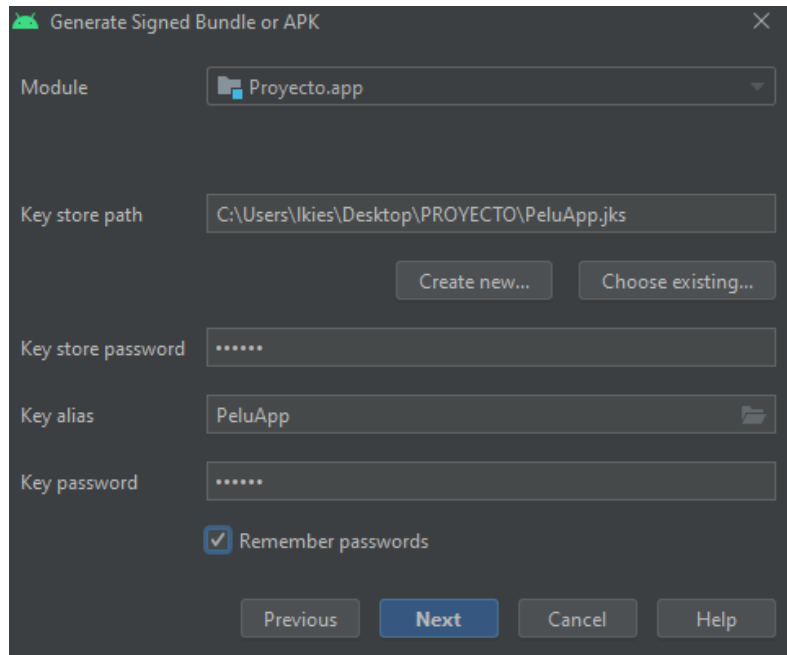


Figura 37: Se han rellenado los campos

Haremos click en Next y nos llevará a otra pantalla, donde se seleccionará release que sería la APK final, ya que debug sería una versión de prueba. Además debemos marcar V2 para ya crea la APK definitiva y firmada.

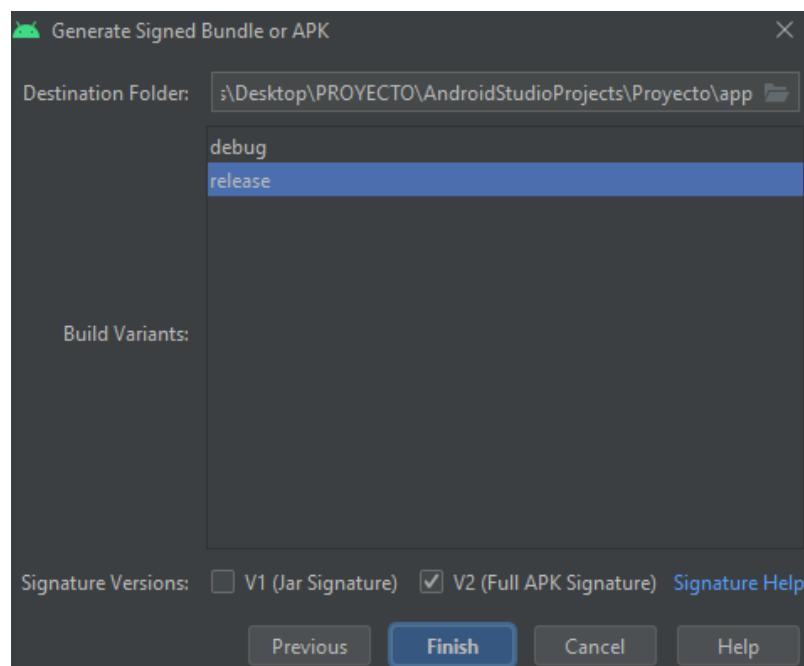


Figura 38: Generación de APK

Una vez se clique en Finish se generará la aplicación y solo habría que instalarla en un dispositivo Android.

12. Mejoras

Hay muchas posibilidades de mejor, y voy a exponer algunas de las más evidentes:

1. Es muy engorroso el hecho de elegir una cita ya que no es posible saber cual es la que está disponible sin intentar registrar una previamente. Una solución sería usar Firebase, que permite crear una serie de SlotTimes, pero en mi caso he tenido problemas y he decidido realizar el proyecto con MySQL, debido al poco tiempo que tenía para realizar el proyecto.
2. A la tienda le haría falta finalizar la compra y enviar los datos a la Base de Datos, así implementar una forma para enviar los productos. También haría falta poder seleccionar más de un mismo producto, ya que ahora solo permite seleccionar solo uno por cada producto.
3. En la ubicación se podría mejorar para poder abrir el mapa de Google y que nos lleve hasta la ubicación.
4. Otra de las mejoras posibles sería poder enviar correos electrónicos automáticamente cuando se registren las citas para confirmarlas.
5. Quizás la mayor mejora sería crear una pantalla donde los administradores o peluqueros pudieran confirmar o cancelar citas, también elegir los días festivos u restricciones de horario.
6. Una última mejora podría ser que cuando se seleccione uno de los servicios, la hora cambie dependiendo de éste, ya que cada servicio conlleva un tiempo distinto.
7. Hacer que las conexiones sean más seguras.

Posiblemente hay más mejoras que se escapan de mi conocimiento, pero creo que con éstas la aplicación escalaría a un nivel superior.

13. Conclusiones

Por una parte, he asentado los conocimientos aprendidos durante el curso, además de adquirir muchos más conocimientos, como es el caso del lenguaje PHP, el cual no se ha tocado durante el curso.

Por otra parte, creo que el proyecto que quería realizar era muy ambicioso para los conocimientos y tiempo que tenía, ya que para realizar algunas funciones hacía falta conocimiento algo más avanzados, tanto de código como de aplicaciones, plataformas, etc.

Para las aplicaciones Android es casi obligatorio usar Firebase, ya que es una plataforma que permite realizar funciones de manera muy sencilla. En mi caso he tenido muchos problemas con la plataforma Firebase.

Espero poder seguir con este proyecto para poder mejorarlo con lo que ya he comentado previamente, ya que es una aplicación que me ilusiona y que se le puede dar un gran uso.

Por último, aunque soy consciente de los fallos que puede tener la aplicación, creo que he hecho un buen trabajo con el tiempo y los conocimientos que tenía. Voy a seguir mejorando la aplicación para aprender y tener un proyecto completo que poder enseñar a una empresa para demostrar mis conocimientos.

14. Anexos.

Anexo 1. Enlace APK: <https://drive.google.com/drive/folders/1whD6-uwllpZU5JJsRTK2pL4e-N8BOPvN?usp=sharing>

Anexo 2. Enlace Proyecto Android Studios: https://drive.google.com/drive/folders/1RkY894iXN_8r5T5CWSaeR3tUtkHjDOPB?usp=sharing

Anexo 3. Enlace Archivos PHP: <https://drive.google.com/drive/folders/1ZsOTilwKddUKR4SudT3pTbbpxXViXp8W?usp=sharing>

15. Bibliografía

<https://www.masquenegocio.com/2017/07/19/desarrollar-app-espana/#:~:text=As%C3%AD%20el%20precio%20medio%20para,aproximadamente%2035%20euros%20por%20hora.>

<https://developer.android.com/studio/publish?hl=es>

<https://es.stackoverflow.com/>

<https://developer.android.com/studio>

<https://firebase.google.com/>

<https://www.youtube.com/>