

CJ PRODUCTIONS

PROYECTO INTEGRADO

2020/2021

**Carlos José Martínez
Sánchez**



Índice

Información.....	3
Descripción en información de la app.....	3
Motivación.....	3
Objetivos del proyecto.....	4
Módulos utilizados.....	4
Investigación.....	5
Análisis.....	6
Usuarios potenciales.....	6
Requisitos del software.....	7
Planificación.....	9
Legislación, encuesta y presupuesto.....	10
Estructura general del programa.....	16
Diagrama del programa.....	16
Tipo de desarrollo.....	16
Casos de uso.....	17
Librerías utilizadas.....	20
Diseño.....	20
Base de datos.....	22
Estructura de la base de datos.....	22
Código Fuente.....	25
Control de excepciones y errores.....	26
Activitys.....	27
Layouts.....	52
Pruebas.....	65
Caja Negra.....	65

Caja Blanca.....	66
Anexo.....	69
Bibliografía.....	75

Información

Descripción e información de la App

CJ Productions es una aplicación creada con el objetivo de acercar mi empresa ficticia de videojuegos (CJ Productions) a los usuarios que lo deseen.

La aplicación está destinada a estar en la Google Play de forma gratuita y sin anuncios (al menos por ahora) y está formada por un apartado de atención al cliente, la cual tiene como finalidad mantener una conversación entre el usuario de la aplicación y los distintos asistentes de atención al cliente, también consta de un registro e inicio de sesión de usuarios tanto por correo y contraseña como por el sistema de autenticación de Google.

Otra característica de la aplicación es el apartado de Novedades, lugar donde se recogerán las distintas noticias relacionadas con la empresa, también contiene otro apartado en el que podremos ver un listado de los videojuegos que ha desarrollado la empresa y que al pulsar en cualquiera de ellos nos lleve a la web donde poder comprarlo.

Por último, consta de un apartado “Sobre nosotros” con el objetivo de saber un poco más de la empresa.

Motivación

Este proyecto es el resultado de estos 2 últimos años del Ciclo Superior de Desarrollo de Aplicaciones Multiplataforma, proyecto en el que intentamos reunir todo lo que se ha aprendido durante el ciclo.

Mi motivación para decidirme por la informática fue principalmente por los videojuegos, de pequeño me sorprendía todo lo que ofrecían y la gran variedad que había.

Así que aprovechando que para la asignatura de Empresas e Iniciativa Emprendedora llegué a crear una empresa de videojuegos, decidí que ese sería el punto de partida del proyecto de fin de grado.

Objetivos del proyecto

El objetivo de la aplicación es acercar la empresa ficticia de videojuegos a los usuarios, es por ello que su diseño sea intuitivo y amigable, de fácil uso y comprensión.

Módulos utilizados

Primer curso.

- Sistemas informáticos:
 - Instalación y configuración de Android Studio 4.1.0 y los diversos programas necesarios.
- Bases de Datos:
 - Creación de una base de datos en Firebase Firestore
 - Acceso, modificación y creación de los datos dentro de la base de datos.
- Programación:
 - Programación en Lenguaje Kotlin (que tiene su base en Java).
- Lenguaje de marcas y sistemas de gestión de información:
 - Programación en lenguaje XML.
- Entornos de desarrollo:
 - Documentación interna del proyecto.
 - Pruebas de caja negra y caja blanca
 - Diagramas de flujo.
- Formación y Orientación laboral:
 - Investigación sobre la privacidad de los datos de los usuarios que se quieran registrar en la App.

Segundo curso

- Programación multimedia de dispositivos móviles:
 - Uso de Android Studio.
- Programación de servicios y procesos:
 - Creación de un chat en tiempo real entre 2 personas.
- Desarrollo de interfaces:
 - Programación de las interfaces de la aplicación.
 - Uso de los distintos conceptos de Usabilidad.
- Empresa e iniciativa emprendedora
 - La aplicación ha sido basada en el proyecto de empresa.
 - Estudio de mercado para ver cómo han realizado las empresas sus aplicaciones.

Investigación

- **VideoView**

Se ha usado un VideoView para mostrar un video de fondo en la pantalla principal de la aplicación. La forma de cómo hacerlo la saqué de un tutorial de YouTube, dicho tutorial es este: [Tutorial](#)

Solo se ha recogido la información de la parte del video de fondo y al estar el ejemplo en Java, ha sido necesaria la conversión a Kotlin.

- **RecyclerView Horizontal**

Se usa en la App para mostrar el apartado de novedades como el de Comprar productos. Para implementar este tipo de RecyclerView se ha mirado la documentación de android, la cual es esta: [Documentación](#)

- **Seleccionar imágenes desde la galería o directamente de la cámara.**

Se usa en la App para seleccionar una imagen de perfil que luego será subida a la base de datos.

En cuanto a los servicios utilizados podemos destacar el siguiente.

- **Firebase**

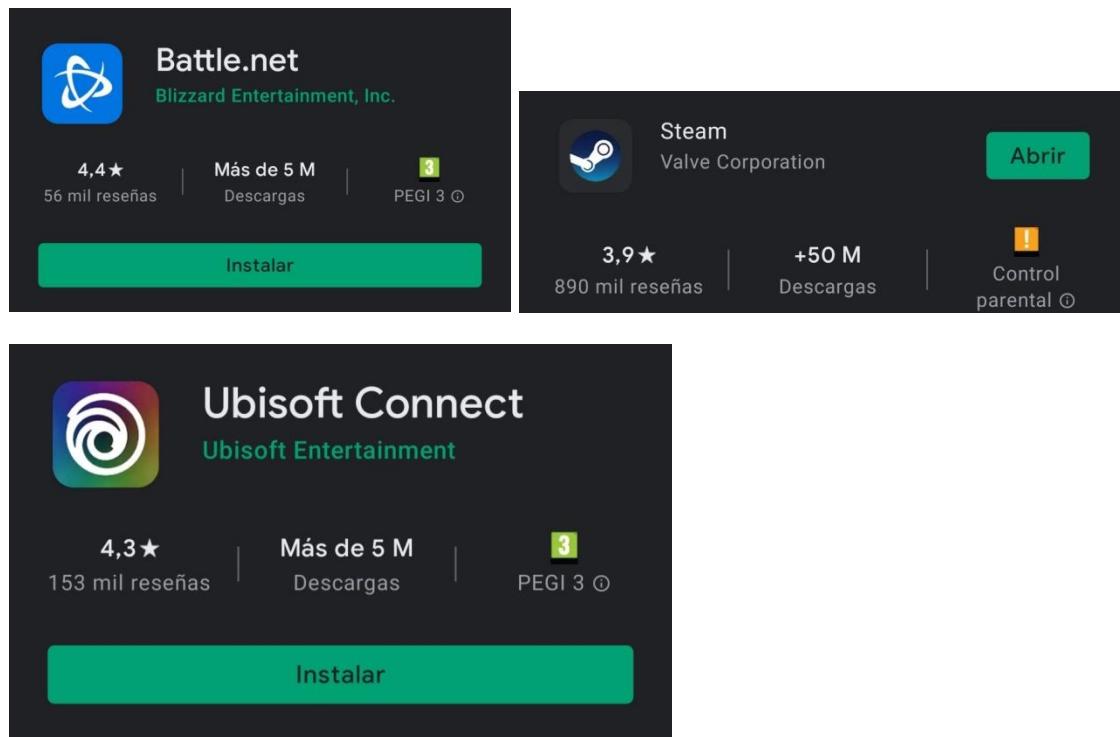
Se ha utilizado Firebase como base de datos para gestionar casi el 100% de la aplicación, aunque he tenido que aprender su funcionamiento desde 0 y para ello me he basado en la propia documentación de Google: [Documentación](#)

Análisis

Usuarios potenciales

La aplicación no va destinada a un rango de edad concreto, más bien está orientada a las personas que les gusta el contenido de la empresa y quieren estar al tanto de las últimas novedades.

He estado investigando aplicaciones similares y he encontrado las siguientes:



De todas las que he mostrado, la más completa sin duda es la de Steam (su cantidad de descargas es la clara muestra), aunque todas ofrecen cosas muy similares a mi aplicación.

También basándonos en el número de descargas, considero que tengo una cantidad de público más que aceptable.

Requisitos del software

Para la creación de la aplicación.

Herramientas necesarias:

- Android Studio 4.1.0



Descargada desde esta página: [Pagina](#)

La documentación de Android studio es la siguiente: [Documentación](#)

- Firebase



Firebase es una base de datos propiedad de Google, por lo que es necesaria una cuenta de Google para poder usarla, con ella se puede llevar un control de usuarios, así como guardar información de los mismos. Su documentación se puede encontrar en el siguiente [enlace](#).

- GitHub



Usado para mantener un control de versiones de la aplicación.

Herramientas necesarias para el funcionamiento de la aplicación.

Requerimientos funcionales

- El usuario puede acceder con normalidad a casi todo el contenido de la aplicación salvo a la sección de atención al cliente, para poder acceder será necesario iniciar sesión.
- En el menú principal se muestra un pequeño video de fondo junto con un botón, la idea es que la pantalla se quede lo más limpia posible, centrando la atención que en el video y en un lateral se encuentran las distintas funciones de la aplicación. Estas opciones son:
 - Poder registrarse mediante un correo y una contraseña
 - Iniciar sesión con el correo y contraseña previamente creado o usando una cuenta de Google.
 - Una vez iniciada sesión aparecerá un botón en la pantalla donde podremos ver nuestro perfil.
 - Una sección de atención al cliente, en el que se iniciará un chat en tiempo real con un administrador aleatoriamente seleccionado.
 - Una sección de novedades en las que se mostrará una lista con todas las novedades sobre los videojuegos de la empresa y demás contenido.
 - Un apartado de comprar productos en el que se nos muestra una lista con todos los videojuegos que la empresa ha creado y que al hacer click en cualquiera de ellos te lleva a la página donde poder comprarlo.
 - Por último, un apartado con la finalidad de conocer un poco más acerca del proyecto.

Requisitos no funcionales

- El registró para nuevos usuarios.
- En el apartado de ver perfil se nos mostrarán los datos del usuario, estos datos se extraen de la base de datos, incluso su imagen de perfil.
- Según las normas de usabilidad, los botones son intuitivos y se podrán acceder a ellos de forma sencilla.

Planificación.

Este ha sido el seguimiento del desarrollo del proyecto integrado.

	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Semana 7	Semana 8	Semana 9	Semana 10
- Informarme sobre Firebase										
- Seguimiento 1										
- Realizar los primeros bocetos										
- Seguimiento 2										
- Creación de la pantalla principal										
- Seguimiento 3										
- Creación de la pantalla de inicio de sesión.										
- Creación de la base de datos.										
- Seguimiento 4										
- Retoque en el inicio de sesión.										
- Creación de ver perfil.										
- Seguimiento 5										
- Creación del chat en tiempo real.										
- Seguimiento 6										
- Implementación de Novedades										
- Retoque en el chat										
- Seguimiento 7										
- Implementación de Comprar productos										
- Seguimiento 8										
- Implementación de Sobre nosotros										
- Seguimiento 9										
- Documentación y manual de la App										
- Seguimiento 10										

Legislación, encuesta y presupuesto

Legislación vigente relacionada con las aplicaciones.

A la hora de lanzar una aplicación, esta debe de cumplir una serie de requisitos legales. Estos requisitos son los siguientes:

- Permisos y licencia de uso

Se le debe de notificar al usuario los permisos tanto para acceder a cualquier contenido de la aplicación, así como de los pagos. También se debe de desarrollar unas condiciones de uso las cuales deberán de aceptar los usuarios de la App.

- Derechos propios y terceros

Es obligatorio disponer de licencias de todos los recursos que se van a utilizar, de no ser así puede acarrear diversos problemas.

- Privacidad y geo localización

Recoger información es en muchas ocasiones indispensable para el correcto funcionamiento de la App, pero el usuario debe de elegir y configurar su privacidad. Es por eso que si nuestra App dispone de geo localización o acceso a la cámara, se le debe de notificar al usuario y contar con su aceptación para poder hacer uso.

- Cookies

Aspecto necesario que el usuario debe de aceptar para el correcto funcionamiento de la App, mediante un aviso informativo con la información básica de las mismas.

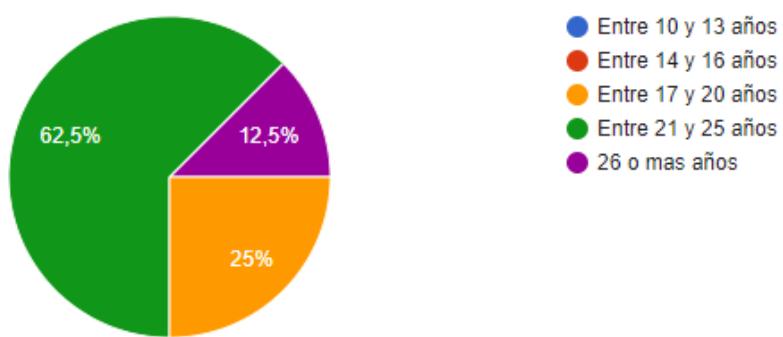
También se ha realizado **una encuesta** a través de Google Forms para conocer un poco el feedback de los usuarios a la hora de usar nuestra App. El formulario se puede realizar haciendo click en el siguiente [enlace](#)

The screenshot shows a Google Forms survey titled "CJ Productions". The title is in a large, bold, dark font at the top. Below it, there's a welcome message: "Bienvenido al formulario sobre la aplicación CJ Productions. Esperemos que estés disfrutando de la aplicación." A note below that says "Nos gustaría conocer tu opinión sobre la aplicación para saber que cosas mejorar. Es por eso que te pedimos que rellenes esta pequeña encuesta." The background of the form features a silhouette of a person holding a flashlight against a purple and pink sunset sky with mountains.

Se han realizado 8 encuestas y estos han sido los resultados

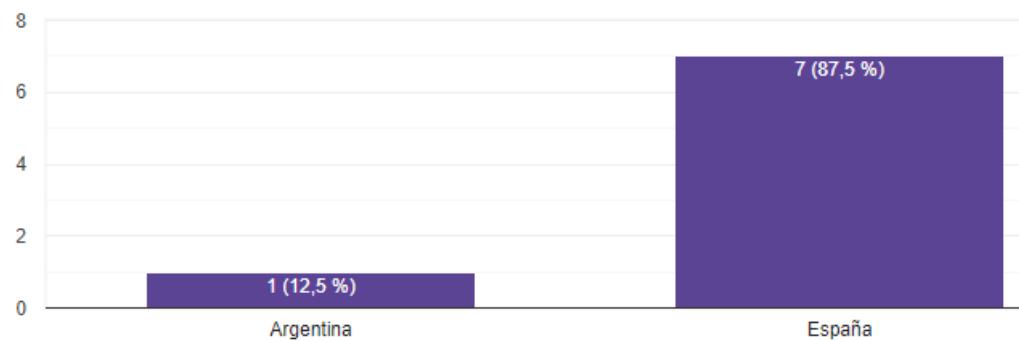
Edad

8 respuestas



País

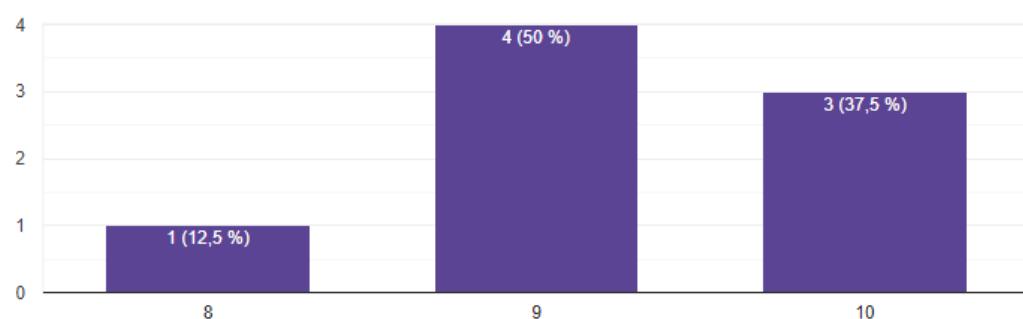
8 respuestas



Valoración de la aplicación

Puntuación de la aplicación

8 respuestas



¿Te ha parecido fácil de usar?



8 respuestas



Si su respuesta anterior fue negativa, indique los motivos. En caso contrario puede saltarse esta pregunta

0 respuestas

Aún no hay respuestas para esta pregunta.

¿Qué es lo que mas te ha gustado de la aplicación?

8 respuestas

Es intuitiva

Es muy limpia

Los apartados

Interfaz

Facilidad de usar

Es muy intuitiva

Es fácil de usar si nunca antes has usado una app como esta

¿Tienes alguna sugerencia para mejorar la aplicación?

3 respuestas

No

Presupuesto

De primeras la aplicación será totalmente gratuita, sin anuncios ni nada parecido. Aunque no se descarta que en un futuro de coloquen anuncios y/o un apartado en el que los usuarios voluntariamente puedan realizar una donación.

En caso de llegar a ese punto, partimos de que la empresa está creada como sociedad limitada, aunque vamos a explicar los pasos para crearla.

Los trámites para crear una Sociedad Limitada consisten desde la solicitud de la denominación social, el otorgamiento de la escritura pública de constitución e inscripción en el Registro Mercantil, y complementariamente el alta en las administraciones tributarias y, en este caso, de Seguridad Social.

Como **requisitos** para crear una Sociedad Limitada tenemos:

1. **Certificado Negativo de Denominación Social:** Es una confirmación de reserva de la denominación que se desea en el que se debe de especificar que la denominación está disponible para su uso en una sociedad nueva. Esto se solicita en el Registro Mercantil Central.
2. **Capital social:** Se debe de disponer de 3.000€ como capital social mínimo para crear una Sociedad Limitada. Este puede ser en efectivo o incluir inmuebles, mobiliario o cualquier otro bien evaluable económicamente.
3. **Abrir una cuenta bancaria:** Los socios deben de abrir una cuenta en cualquier entidad bancaria, para el que se podrá pedir un documento que acredite que se está conformando una Sociedad Limitada.
4. **Tener DNI o NIE:** Los socios de la Sociedad Limitada y el/los administrador(es) deben contar con número de DNI o NIE.

Ahora vamos a indicar los **pasos** a seguir para poder crear la Sociedad Limitada de una forma simplificada.

1. **Solicitar el nombre de la sociedad.** (Importante que no esté siendo utilizado por otra Sociedad Limitada).
2. **Abrir la cuenta bancaria de la Sociedad.** con el mínimo de 3.000€ antes mencionado.

3. **Redactar los estatutos sociales.** Aquí me quiero detener, ya que los Estatutos Sociales de la Sociedad Limitada contienen la estructura, funcionamiento interno y normativas sobre las que se regirá la Sociedad. Este documento se incorpora al registro público y debe de contener entre otras cosas, la identificación de la modalidad de la Sociedad de Responsabilidad Limitada, el domicilio social, capital social y participaciones en que se divide.
4. **Escritura pública de la constitución de la sociedad.** Todos los socios deben acudir a la notaría para formalizar la Escritura de la Constitución.
5. **Obtener el NIF (provisional) de la sociedad.** Al ser provisional, este tiene una validación de 6 meses y después debe de cambiarse por el NIF definitivo una vez terminados todos los trámites.
6. **Alta en el Impuesto de Actividades Económicas.**
7. **Declaración del IVA o censal.** Se debe de cumplir el modelo 036 el cual se puede consultar en la web de la [Agencia Tributaria de España](#).
8. **Inscripción en el Registro Mercantil provincial.** Aquí deberemos hacer uso del NIF provisional que obtuvimos previamente junto con una copia autorizada de la escritura de la constitución de la Sociedad.
9. **Adquirir el NIF definitivo** en la oficina de Hacienda

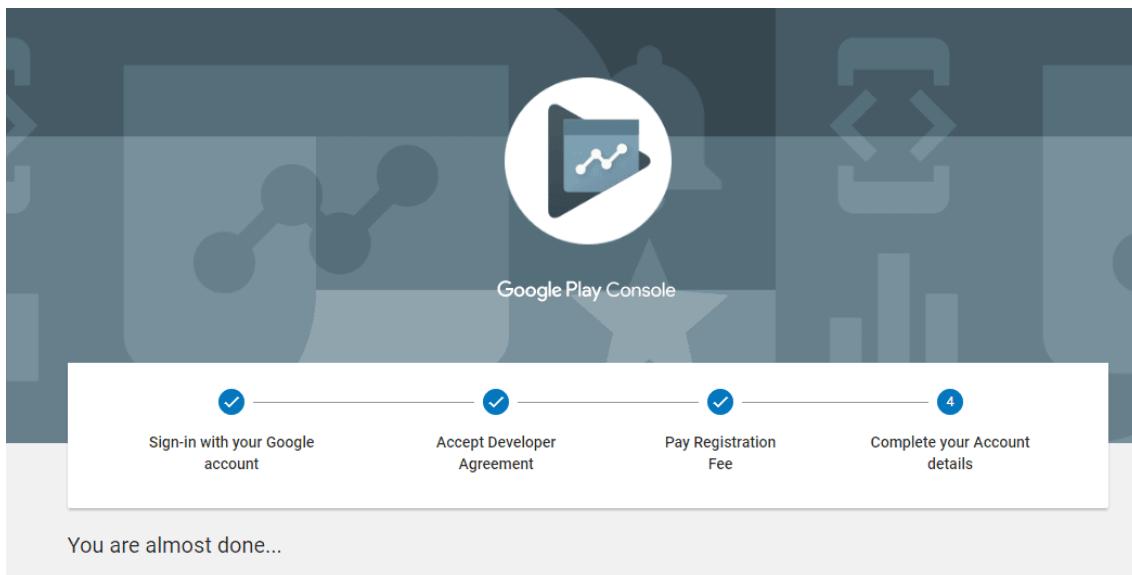
Publicar la aplicación en la Google Play

Puesto que una forma de recaudar dinero es mediante la publicación de la aplicación en la tienda de aplicaciones, es preferible explicar el proceso en este apartado.

Para publicar la aplicación hay que seguir una serie de pasos:

- Registrarse con una cuenta de desarrollador

Primero hay que registrarse en el programa de desarrolladores de Google para poder tener acceso a la consola de desarrollador. Deberemos de completar nuestro perfil incluyendo un único pago de 25 dólares americanos.



Una vez llenado todo el perfil de desarrollador con datos tales como el nombre, dirección y el correo electrónico, podremos pasar al siguiente paso que es el de crear el perfil de nuestra aplicación.

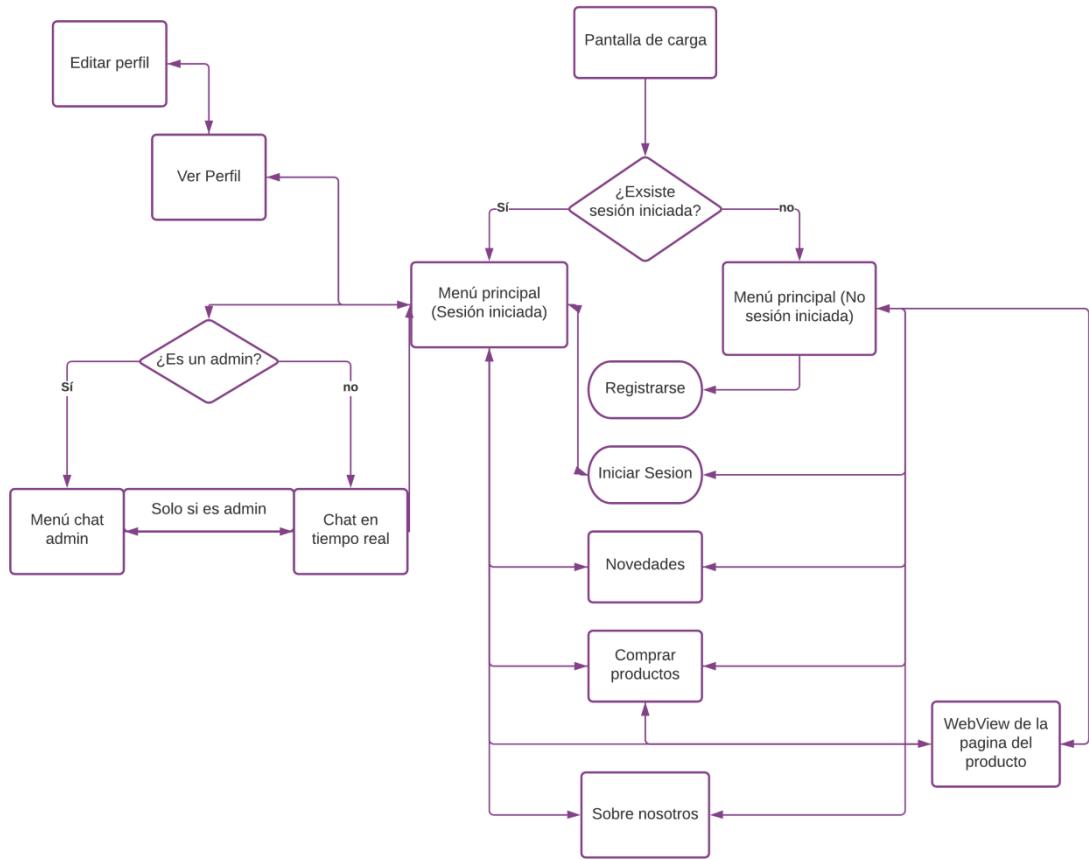
- Crear el perfil de la aplicación.

Deberemos registrar una nueva aplicación y se nos solicitarán datos como el idioma por defecto, el nombre de la aplicación, una descripción de la misma, también deberemos añadir capturas y/o un video de YouTube en el que mostraremos el contenido de nuestra aplicación y el ícono de la aplicación.

También podremos añadir etiquetas de búsqueda, aunque es opcional pero si muy recomendable.

Estructura general del programa

Diagrama del programa



Tipo de desarrollo

Modelo incremental

Se ha utilizado el modelo incremental debido a que une el modelo en cascada con la idea de ir incrementando poco a poco las funcionalidades del programa.

Al principio se empezó con el menú principal y poco a poco se le fueron añadiendo las opciones de inicio de sesión, chat,... Así hasta terminar el desarrollo de la aplicación.

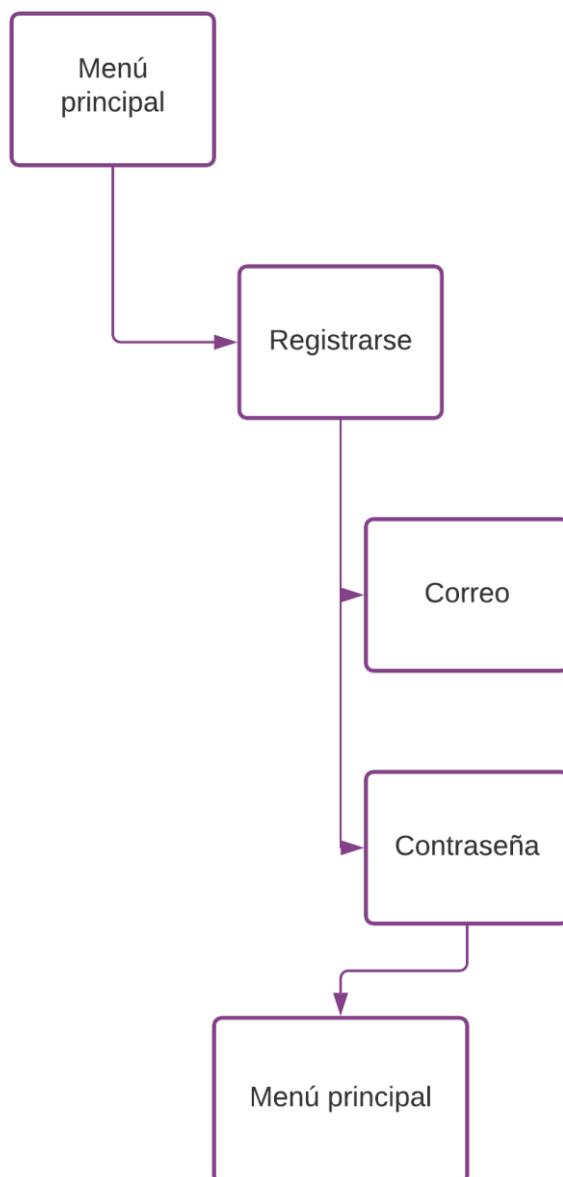
Casos de uso

Aquí vamos a recoger algunos diagramas que hará la aplicación.

Registrar usuario

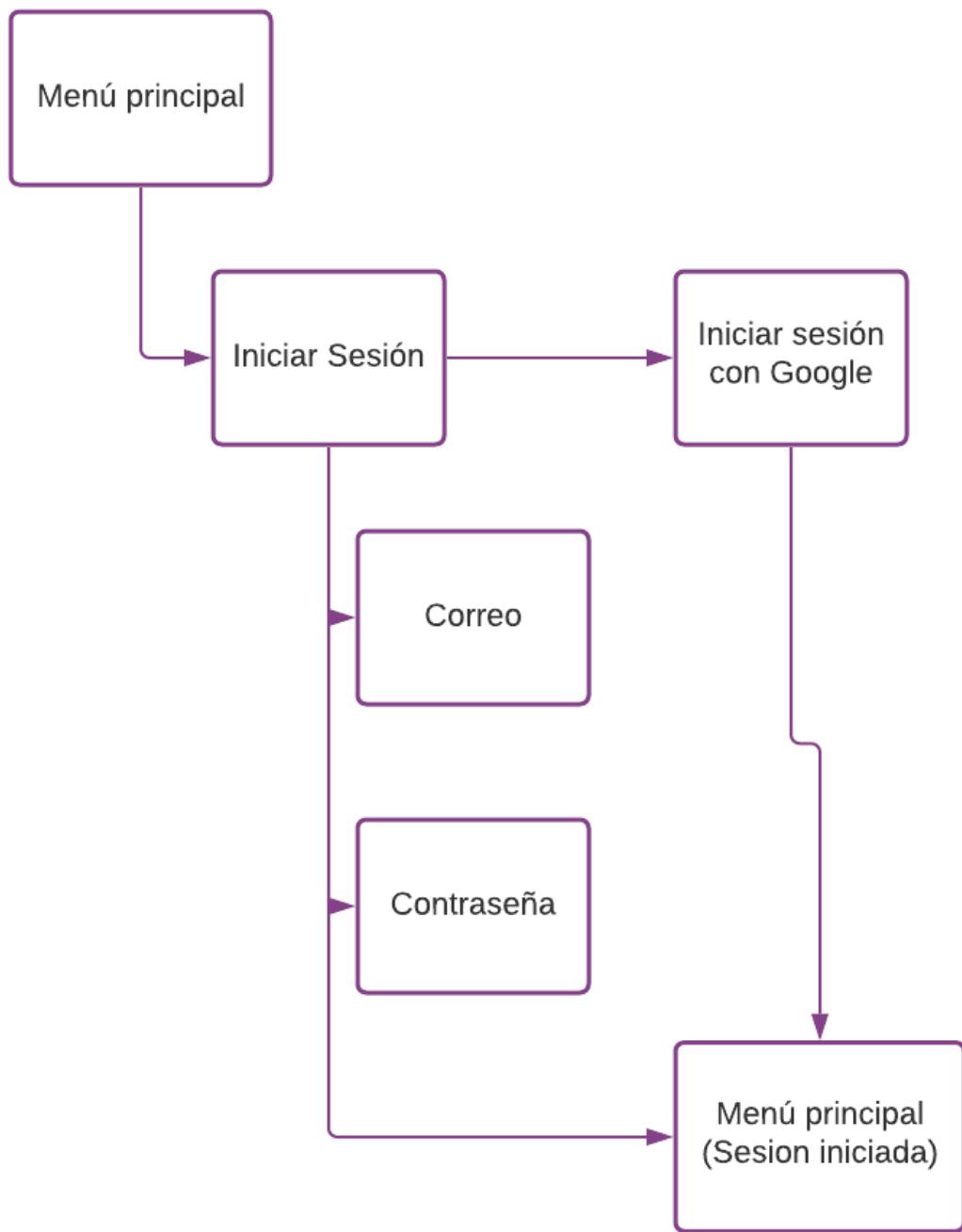
Como tal no es necesario que el usuario se registre, sin hacerlo podrá acceder a casi todo el contenido de la aplicación. A lo único que no podrá acceder es a la sección Atención al Cliente, sección en la que es obligatorio iniciar sesión puesto que el identificador principal es el correo.

El diagrama para el registro del usuario es el siguiente



Inicio de sesión

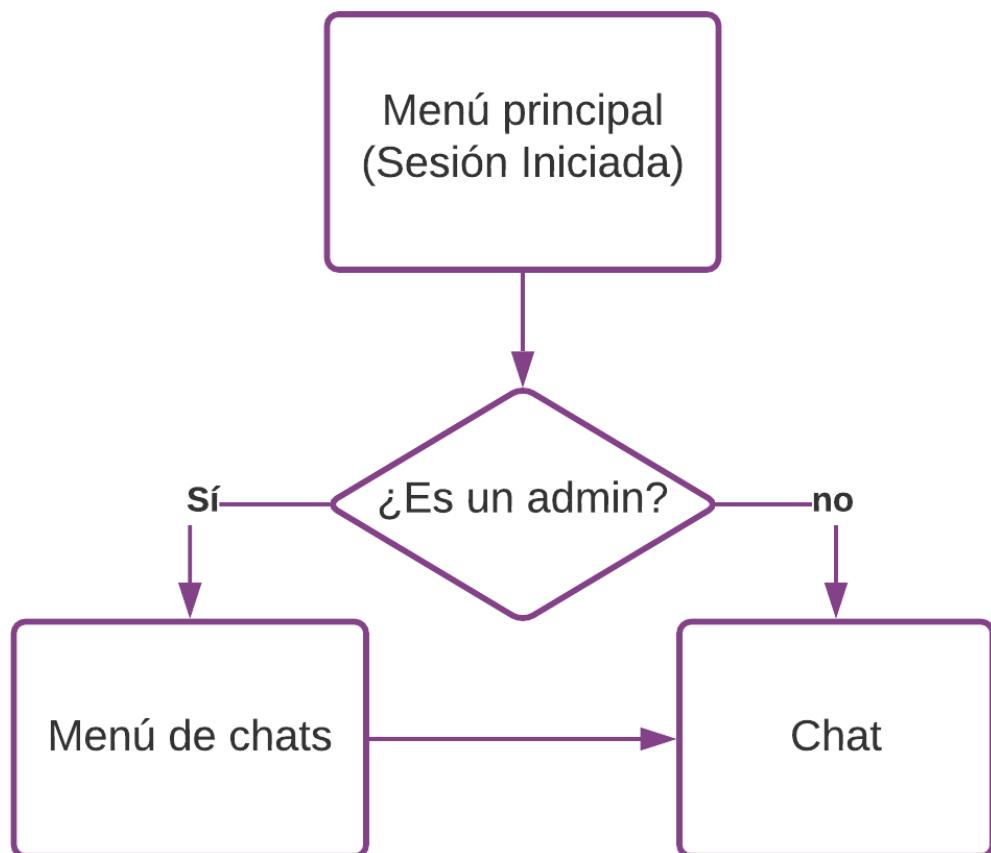
El usuario si quiere acceder sobre todo a la Atención al Cliente, deberá de iniciar sesión usando la cuenta de Google o bien mediante un correo y contraseña. Una vez iniciada sesión en el menú principal nos aparecerá un botón con el que editar nuestro perfil. El diagrama es el siguiente:



Atención al cliente

Si el usuario que ha iniciado sesión es un administrador, se abrirá una ventana en el que se mostrará un listado de chats de parte de todos los usuarios y al hacer click en cualquiera de ellos nos llevará al chat indicado.

Si es un usuario normal, se le dirige directamente al chat y se le asignará un administrador aleatorio con el que poder chatear.



Librerías utilizadas

Firebase

Las librerías de Firebase utilizadas son las siguientes:

```
'com.google.firebaseio:firebase-bom:27.1.0'  
'com.google.firebaseio:firebase-analytics-ktx'  
'com.google.firebaseio:firebase-auth-ktx:20.0.3'  
'com.google.android.gms:play-services-auth:19.0.0'  
'com.google.firebaseio:firebase-firebase-store-ktx:22.1.2'  
'com.google.firebaseio:firebase-storage-ktx'
```

Picasso

```
'com.squareup.picasso:picasso:2.71828'
```

Dialogo de alerta

```
'com.github.d-max:spots-dialog:1.1 @aar'
```

Diseño

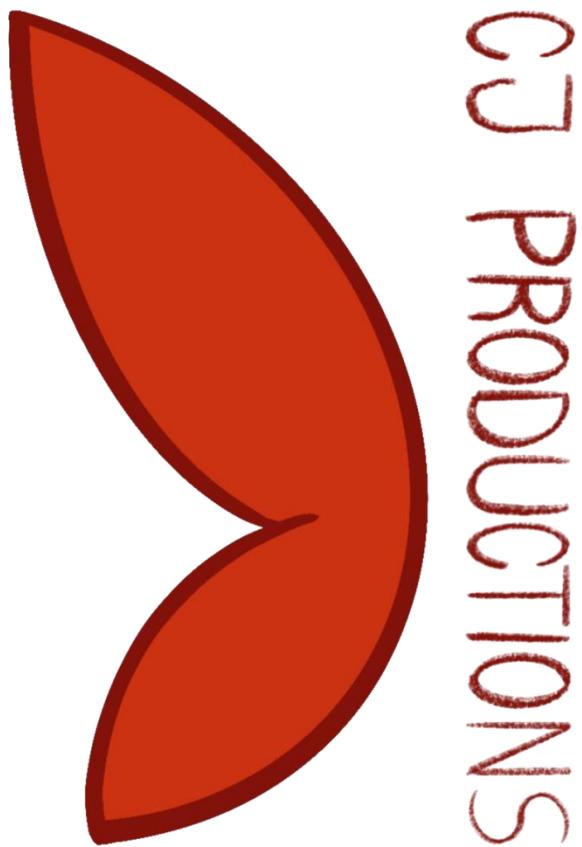
Principios de Usabilidad

A continuación se detallan los principios de usabilidad que he encontrado en mi aplicación:

1. Adecuación entre el sistema real y el mundo real
2. Libertad y control por el usuario
3. Consistencia y estándares.
4. Prevención de errores.
5. Estética y diseño minimalista.
6. Ayuda y documentación.

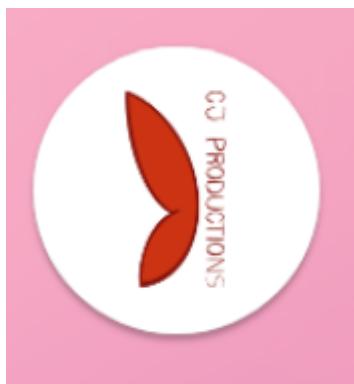
Logotipo

El logotipo original es el siguiente:



Este logotipo es el que se ha usado para el ícono de la aplicación pero redimensionado para que se pueda mostrar en todos los dispositivos

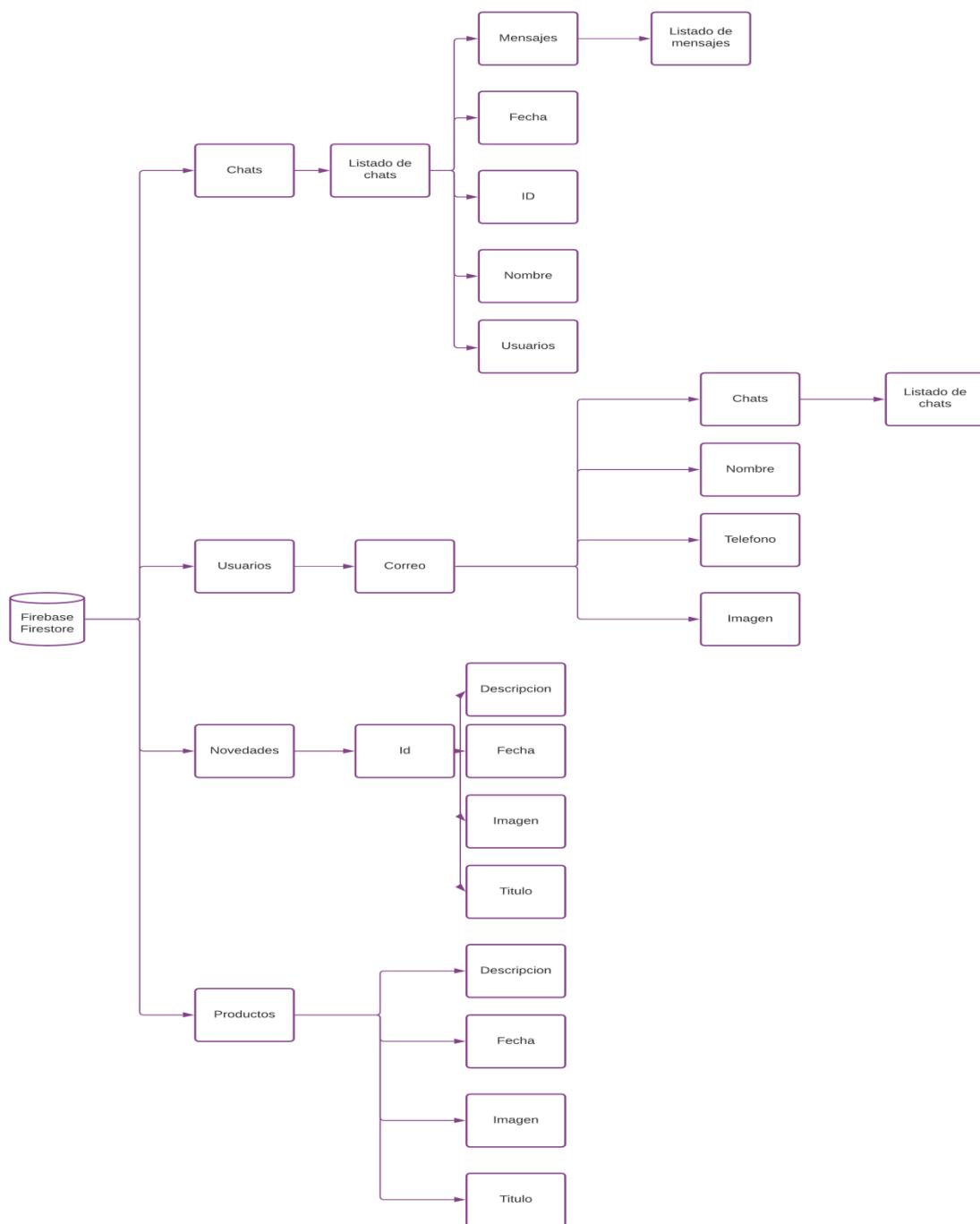
El ícono de la aplicación es el siguiente:



Base de datos

Firebase Authentication proporciona servicios para registrar y autenticar usuarios. Se usará la autenticación por correo y contraseña o usando una cuenta de Google.

La estructura de la base de datos es la siguiente:



Firebase Authentication

The screenshot shows the Firebase Authentication interface under the 'Users' tab. It displays a list of users with columns for Identifier, Providers, Date Created, Date Last Accessed, and User ID. The list includes entries for 'carlosjmsanchez@gmail.com' (Email), 'pepe@gmail.com' (Email), 'carlitos123@gmail.com' (Email), and 'carlosjmsanchezglobo@gmail.com' (Google). A search bar at the top allows filtering by email or phone number. A button to 'Agregar usuario' (Add user) is visible.

Firebase Firestore

The screenshot shows the Cloud Firestore interface. On the left, a sidebar lists collections: Chats, Novedades, Productos, and Usuarios. The 'Chats' collection is selected. In the main area, a document named '9ea34b64-6f30-49f5-ac9f-89c005782986' is expanded, showing its subcollection 'messages'. Below this, an 'Agregar campo' (Add field) section is shown with fields for 'fecha', 'id', 'name', and a 'users' array containing two entries: 'carlosjmsanchezglobo@gmail.com' and 'carlosjmsanchez@gmail.com'. The status bar at the bottom indicates the location is 'eur3 (europe-west)'.

The screenshot shows another view of the Cloud Firestore interface. The 'Novedades' collection is selected. A document named 'UWP1FME47UnHtmoepuLJ' is expanded, showing fields for 'descripcion', 'fecha', 'imagen', and 'titulo'. The 'descripcion' field contains a promotional message about the release of a game demo. The 'fecha' field is set to 'Noticia del 27-05-2020'. The 'imagen' field points to a file named 'novedades/daysGoneFechaDemo.png'. The 'titulo' field is 'Fecha de publicación de la demo'.

The screenshot shows a MongoDB interface with three panels. The left panel displays a sidebar with collections: Chats, Novedades, Productos (selected), and Usuarios. The middle panel shows the 'Productos' collection with a document titled 'JueYKBM7iVdSvyR1mbGx'. The right panel shows the document's fields: descripción, enlace, imagen, and nombre.

+ Iniciar colección	+ Agregar documento	+ Iniciar colección
Chats	JueYKBM7iVdSvyR1mbGx	+ Agregar campo
Novedades		descripción: "Adéntrate en un mundo devastado por la vegetación con el objetivo de sobrevivir"
Productos		enlace: "https://store.steampowered.com/app/1259420/Days_Gone/"
Usuarios		imagen: "productos/logo_genesis.jpg"
		nombre: "Days Gone"

The screenshot shows a MongoDB interface with three panels. The left panel displays a sidebar with collections: Chats, Novedades, Productos, and Usuarios (selected). The middle panel shows the 'Usuarios' collection with documents: carlitos123@gmail.com, carlosjmsanchez@gmail.com (selected), carlosjmsanchezglobo@gmail.com, and pepe@gmail.com. The right panel shows the document's fields: Nombre and Telefono.

+ Iniciar colección	+ Agregar documento	+ Iniciar colección
Chats	carlitos123@gmail.com	Chats
Novedades	carlosjmsanchez@gmail.com	+ Agregar campo
Productos	carlosjmsanchezglobo@gmail.com	Nombre: "Carlos"
Usuarios	pepe@gmail.com	Telefono: "654565676"

Storage

The screenshots show the Google Cloud Storage interface for the bucket `gs://cjproductions-pi2020.appspot.com`. Each screenshot displays a list of files and folders with columns for Nombre (Name), Tamaño (Size), Tipo (Type), and Modificación más reciente (Last modified).

- Screenshot 1:** Shows the root directory with three empty folders: `novedades/`, `productos/`, and `usuarios/`. A blue "Subir archivo" (Upload file) button is visible.
- Screenshot 2:** Shows the `novedades/` folder containing one file: `daysGoneFechaDemo.png` (6.6 MB, image/png, 27 may. 2021).
- Screenshot 3:** Shows the `productos/` folder containing one file: `logo_genesis.jpg` (1.74 MB, image/jpeg, 25 may. 2021).
- Screenshot 4:** Shows the `usuarios/` folder containing three empty folders: `CcrbPjaHh7WCvtkiGCICChskEPXZ2/`, `bv4RdoLsjHd6E6s70z1qlgGJITB2/`, and `pfxReWRLkFuihbWYDbhiZmpKEM2/`.

Cada carpeta de usuarios almacena su imagen de perfil respectivamente

Código fuente

El código se encuentra en mi GitHub, el código para acceder es el siguiente:

https://github.com/W0lfMeN/CJ_Productions

Control de excepciones y errores

Los mensajes que van a aparecer serán principalmente de alerta para notificar al usuario de los posibles errores que pueden suceder a lo largo de la aplicación. Algunos mensajes son los siguientes:

- Si no hay sesión iniciada, al momento de acceder a la sección Atención al cliente saldrá un mensaje diciendo que debe de iniciar sesión para poder acceder.
- A la hora de iniciar sesión con Google, si ocurre algún problema en el proceso se le notificará al usuario de que ha ocurrido un error durante el inicio de sesión.
- Al igual que el apartado anterior, se mostrará un error si ha habido un error durante el inicio de sesión por correo o por contraseña, principalmente será porque no se ha introducido correctamente alguno de los campos como los son el correo y contraseña.
- Durante el registro de usuarios, se notificará al usuario en caso de que los dos campos de contraseña no coincidan, en el caso de que el correo no sea válido o ya se encuentra registrado en la base de datos.

Activity

MainActivity.kt

```
 MainActivity.kt
 1 package com.example.cjproductions
 2
 3 import android.content.Context
 4 import android.content.Intent
 5 import android.content.SharedPreferences
 6 import android.net.Uri
 7 import android.os.Bundle
 8 import android.view.Menu
 9 import android.view.MenuInflater
10 import android.view.MenuItem
11 import android.view.View
12 import android.widget.Toast
13 import android.app.AlertDialog
14 import android.appCompat.AppCompatActivity
15 import com.example.cjproductions.atencionCliente.activities.ChatActivity
16 import com.example.cjproductions.atencionCliente.admin.MenuPrincipalAdmin
17 import com.example.cjproductions.atencionCliente.models.chat
18 import com.example.cjproductions.comprarProductos.MenuPrincipalComprarProductos
19 import com.example.cjproductions.comprarProductos.activity.ProductosActivity
20 import com.example.cjproductions.login.InicioSesion
21 import com.example.cjproductions.login.Registrarse
22 import com.example.cjproductions.novedades.MenuPrincipalNovedades
23 import com.example.cjproductions.perfilUsuarios.PerfilUsuarios
24 import com.example.cjproductions.sobreNosotros.SobreNosotros
25 import com.google.firebaseio.firebaseio.ktx.firebaseio
26 import com.google.firebaseio.firebaseio.ktx.firebaseio
27 import kotlinx.android.synthetic.main.activity_main.*
28 import java.time.LocalDateTime
29 import java.time.format.DateTimeFormatter
30 import java.util.*
31
32
33 class MainActivity : AppCompatActivity() {
34
35     /**
36      * Variable que almacenará los correos pertenecientes a los desarrolladores de la aplicación
37      * Se usará para saber si la cuenta iniciada es de un desarrollador o no
38      * Para saber eso, se comparará el contenido del array con el email perteneciente al archivo de persistencia
39      */
40     private val correosDesarrolladores = arrayOf("carlosjmsanchez@gmail.com")
41
42     override fun onCreate(savedInstanceState: Bundle?) {
43         //Thread.sleep(4000)
44         setTheme(R.style.Theme_CJProductions)
45
46         setContentView(R.layout.activity_main)
47
48         //setSupportActionBar(toolbar)
49
50         setTheme(R.style.Theme_CJProductions)
51
52         super.onCreate(savedInstanceState)
53         setContentView(R.layout.activity_main)
54
55         //Llamamos al método que se encarga de comprobar si existe contenido en el archivo de preferencias
56         comprobarArchivoPreferencias()
57
58         //Llamamos al método donde estarán todos los listeners de la ventana principal
59         ponerListeners()
60
61         //Llamamos al método que se encarga de poner y reproducir el video de fondo del activity
62         iniciarBackgroundVideo()
63     }
64
65     /**
66      * Método que añade los listeners que necesitamos en este activity
67      */
68     private fun ponerListeners() {
69         //Botón Saber Más
70
71         btComprar.setOnClickListener { v: View? ->
72             //Aqui se llamará a la pantalla de comprar producto
73             //Asignamos los parámetros que se van a transferir al activity
74             val nombre = "Days Gone"
75             val enlace = "https://store.steampowered.com/app/1259420/Days_Gone/"
76
77             val intent: Intent = Intent(packageContext, this, ProductosActivity::class.java)
78             intent.putExtra("enlace", enlace)
79             intent.putExtra("nombre", nombre)
80             startActivity(intent)
81         }
82
83         btComprar.setOnLongClickListener { v: View? ->
84             Toast.makeText(context, "Pulsa para obtener mas información del videojuego", Toast.LENGTH_SHORT).show()
85             true
86         }
87
88         //Botón Ver Perfil
89         mainPerfil.setOnLongClickListener { v: View? ->
90             Toast.makeText(context, "Pulsa para ver y configurar tu perfil", Toast.LENGTH_SHORT).show()
91             true
92         }
93
94     }
95 }
```

```
86     mainBtPerfil.setOnLongClickListener { v: View! ->
87         Toast.makeText( context, "Pulsa para ver y configurar tu perfil", Toast.LENGTH_SHORT).show()
88         true
89     }
90
91     mainBtPerfil.setOnClickListener { v: View! ->
92         val intent: Intent = Intent( packageContext, perfilusuarios::class.java )
93         startActivity(intent)
94     }
95
96 }
97
98 /**
99 * Función que se encarga de buscar e iniciar el video de fondo del activity
100 */
101 private fun iniciarBackgroundVideo() {
102     //Instanciamos la ruta del video de fondo usando la clase Uri
103     videoPrincipal.setVideoURI(Uri.parse(unString: "android.resource://$(packageName)://${R.raw.video_principal}"))
104
105     //con esto se inicia el video
106     videoPrincipal.start()
107
108     /**
109      * Añadimos un listener al video. Este bloque de código entrará cuando finalice el video
110      * En concreto, cuando termine el video lo que hará será volver a reproducirlo.
111      * Haciendo que se reproduzca en bucle
112      */
113     videoPrincipal.setOnCompletionListener() { mediaPlayer: MediaPlayer! ->
114         videoPrincipal.start()
115     }
116 }
117
118 /**
119 * Este método se ejecuta cuando se vuelve a la ventana principal
120 * desde cualquier otra activity.
121 *
122 * Lo que hace es reproducir otra vez el video de fondo.
123 * De no hacer esto, la pantalla se quedaría en blanco
124 * cuando se vuelve desde otra activity
125 */
126 override fun onRestart() {
127     super.onRestart()
128     iniciarBackgroundVideo()
129
130     //Aquí se va a comprobar si existe o no contenido en el archivo de preferencias
131
132     //Aquí se va a comprobar si existe o no contenido en el archivo de preferencias
133     //Se hace así para evitar modificar la visibilidad del botón ver perfil desde otro activity
134     comprobarArchivoPreferencias()
135
136 /**
137 * Método que añade el ícono de las opciones del menu en el activity
138 */
139 override fun onCreateOptionsMenu(menu: Menu?): Boolean {
140     val inflate: MenuInflater = menuInflater
141     inflate.inflate(R.menu.menu_principal, menu)
142     return true
143 }
144
145 /**
146 * Método que les da funcionalidad a los botones del menu
147 */
148 override fun onOptionsItemSelected(item: MenuItem): Boolean {
149     when (item.itemId) {
150
151         //Al pulsar se llamará a la clase de Atención Al cliente
152         R.id.atencioncliente -> {
153             val prefs: SharedPreferences = getSharedPreferences("com.example.cjproductions.PREFERENCIAS", Context.MODE_PRIVATE)
154
155             /**
156             * Si el usuario del archivo de persistencia es distinto de null
157             * quiere decir que hay una cuenta iniciada. En caso contrario no dejará iniciar el activity
158             * mostrando un mensaje informando del problema
159
160             * En caso de existir un correo en el archivo de persistencia,
161             * este se comparará con el array en el que están contenidos los correos de los administradores.
162
163             * Si el correo coincide con uno de los que hay en el array, quiere decir que es un admin
164             * y por lo tanto iniciará el activity que muestra el menú de chats del admin
165
166             * Por el contrario, si no coincide con ninguno, quiere decir que es un usuario normal
167             * y se llamará al método sendMessage para crear el chat
168
169             if (prefs.getString(key = "email", defValue = null) != null) {
170                 if (!correoDesAdministradores.contains(prefs.getString(key = "email", defValue = null).toString()))
171                     val intent = Intent( packageContext, this, MenuPrincipalAdmin::class.java )
172                     startActivity(intent)
173                 else {
174                     showalert("Si cierra la ventana del chat, tendrá que volver a entrar...")
175                 }
176             }
177         }
178     }
179 }
```

```

173         showAlertDialog("Si cierra la ventana del chat, tendrá que volver a entrar...")
174     } else
175         Toast.makeText(context: this, "Debe de iniciar sesión para acceder", Toast.LENGTH_SHORT).show()
176     }
177 }
178
179 //Al pulsarse se llamará a la clase de Novedades
180 R.id.Novedades -> {
181     val intent: Intent = Intent( packageName: this, MenuPrincipalNovedades::class.java)
182     startActivity(intent)
183 }
184
185 //Al pulsarse se llamará a la clase de Comprar_productos
186 R.id.ComprarProductos -> {
187     val intent: Intent = Intent( packageName: this, MenuPrincipalComprarProductos::class.java)
188     startActivity(intent)
189 }
190
191 //Al pulsarse se llamará a la clase de Sobre_nosotros
192 R.id.SobreNosotros -> {
193     val intent: Intent = Intent( packageName: this, SobreNosotros::class.java)
194     startActivity(intent)
195 }
196
197 R.id.mainBtnIniciarSesion -> {
198     val intent: Intent = Intent( packageName: this, InicioSesion::class.java)
199     startActivityForResult(intent, requestCode: 100)
200     //Este método llama al activity y cuando termine llama a onActivityResult
201     //para trabajar con los datos enviados de dicha activity
202 }
203
204 R.id.mainBtnRegistrarse -> {
205     val intent: Intent = Intent( packageName: this, Registrarse::class.java)
206     startActivityForResult(intent, requestCode: 100)
207 }
208
209 /*
210 R.id.mainBtnVerPerfil ->
211 */
212
213 }
214
215 return super.onOptionsItemSelected(item)
216 }
217
218 /**
219 * Método que es llamado automáticamente cuando termina el activity Iniciar_sesion
220 * Lo que hace es retornar el correo que ha sido introducido en el activity para guardarla
221 * en el archivo de preferencias
222 */
223 override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
224     super.onActivityResult(requestCode, resultCode, data)
225     if (requestCode == 100 & resultCode == RESULT_OK) {
226         if (data != null) {
227             editarPersistenciaDatos(data.getStringExtra( name: "email"))
228
229             //Aquí tenemos que acceder al botón de Ver_Perfil
230             mostrarOcultarBtnVerPerfil( valor: 1 )
231         }
232     }
233 }
234
235 /**
236 * Método que añade el correo al fichero de persistencia
237 */
238 fun editarPersistenciaDatos(email: String?) {
239     val sharedpreferences: Editor? = getSharedPreferences("com.example.cjproductions.PREFERENCIAS", Context.MODE_PRIVATE).edit()
240     if (prefs != null) {
241         prefs.putString("email", email)
242         prefs.apply()
243     }
244 }
245
246 /**
247 * Método que muestra o oculta el botón de iniciar_sesion según el parámetro
248 *
249 * 1 para MOSTRAR el botón
250 * 0 para OCULTAR el botón
251 */
252 fun mostrarOcultarBtnVerPerfil(valor: Int) {
253     if (valor == 1)
254         mainBtnPerfil.visibility = View.VISIBLE
255
256     if (valor == 0)
257         mainBtnPerfil.visibility = View.INVISIBLE
258 }

```

```
mainBtPerfil.visibility = View.INVISIBLE
}

/**
 * Este metodo nos permite saber si hay una sesion iniciada comprobando el contenido de la persistencia
 *
 * Si por defecto es null, quiere decir que no hay datos, por lo cual no hay sesion iniciada
 * por lo que llamo al metodo de ocultar el boton para cerrar sesion pasandole el parametro correcto
 *
 * Por el contrario si es distinto de null, quiere decir que hay una sesion iniciada
 * y mostrara el boton de cerrar sesion llamando al metodo correspondiente
 */
private fun comprobarArchivoReferencias() {
    val prefs: SharedPreferences = getSharedPreferences("com.example.cjproductions.PREFERENCIAS", Context.MODE_PRIVATE)

    if (prefs.getString(key = "email", defValue = null) == null) {
        mostrarOcultarBtVerPerfil(val: 0)
    } else {
        mostrarOcultarBtVerPerfil(val: 1)
    }
}

/**
 * Metodo que gestiona y crea del chat bajo la vista del usuario
 */
private fun gestionChatUsuario() {
    val db = FirebaseFirestore

    //guardamos en la variable 'user' el correo que se encuentre en la persistencia
    val prefs: SharedPreferences = getSharedPreferences("com.example.cjproductions.PREFERENCIAS", Context.MODE_PRIVATE)
    val user = prefs.getString(key = "email", defValue = null).toString()

    //Damos valores a las variables que se necesitan para crear el objeto Chat
    val chatId = UUID.randomUUID().toString()
    val otherUser = correosDeAdministradores.random() //se asigne un desarrollador aleatorio
    val users = listOf(user, otherUser)
    val laFecha = LocalDateTime.now().format(DateTimeFormatter.ofPattern(pattern = "dd-MM-yyyy HH:mm")).toString()

    //Creamos el objeto Chat con sus valores
    val chat = Chat(
        id = chatId,
        name = "Chat con $user",
        users = users,
        fecha = laFecha
    )

    //Añadimos la colección del Chat tanto en una colección principal, como en una colección dentro de cada usuario que intervienen
    db.collection(collectionPath = "Chats").document(chatId).set(chat)
    db.collection(collectionPath = "Usuarios").document(user).collection(collectionPath = "Chats").document(chatId).set(chat)
    db.collection(collectionPath = "Usuarios").document(otherUser).collection(collectionPath = "Chats").document(chatId).set(chat)

    //Iniciamos el activity pasando los datos importantes como son el id, el usuario y el número del administrador
    val intent = Intent(packageContext = this, ChatActivity::class.java)
    intent.putExtra(name = "chatId", chatId)
    intent.putExtra(name = "user", user)
    intent.putExtra(name = "admin", otherUser)
    startActivity(intent)
}

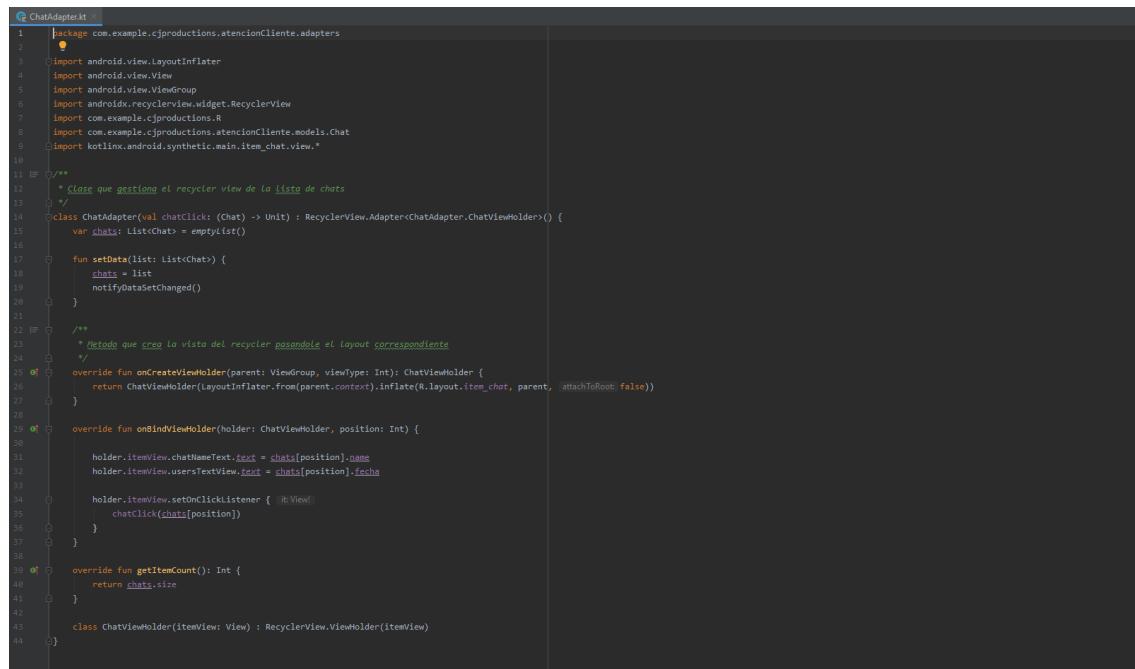
/**
 * Función que genera un mensaje de alerta en la pantalla
 * con el texto que se le pase por parametro
 */
private fun showAlert(mensaje: String) {
    val builder = AlertDialog.Builder(context = this)
    builder.setMessage(mensaje)
    builder.setPositiveButton("SI") { _, _ ->
        gestionChatUsuario()
    }
    builder.setNegativeButton("NO") { view, _ ->
        view.dismiss()
    }
    val dialog = builder.create()
    dialog.show()
}
}
```

ChatActivity.kt

```
1 package com.example.cjproductions.atencionClientes.activities
2
3 import android.os.Bundle
4 import androidx.appcompat.app.AlertDialog
5 import androidx.appcompat.app.AppCompatActivity
6 import androidx.recyclerview.widget.LinearLayoutManager
7 import com.example.cjproductions.R
8 import com.example.cjproductions.atencionCliente.adapters.MessageAdapter
9 import com.example.cjproductions.atencionCliente.models.Message
10 import com.google.firebaseio.firebaseio.Query
11 import com.google.firebaseio.firebaseio.ktx.firebaseio
12 import com.google.firebaseio.ktx.firebaseio
13 import kotlin.android.synthetic.main.activity_chat.*
14
15 class ChatActivity : AppCompatActivity() {
16     private var chatId = ""
17     private var user = ""
18
19     private var db = Firebase.firebaseio
20
21     override fun onCreate(savedInstanceState: Bundle?) {
22         super.onCreate(savedInstanceState)
23         setContentView(R.layout.activity_chat)
24
25         var admin = "" //Esta variable almacenará el nombre del administrador seleccionado
26
27         intent.getStringExtra("chatId")?.let { chatId = it }
28         intent.getStringExtra("user")?.let { user = it }
29
30         //Esta variable solo será usada si el activity es llamado desde un usuario y no un admin
31         intent.getStringExtra("admin")?.let { admin = it }
32
33         /*
34             Si el admin es distinto de "" quiere decir que es un usuario es el que ha iniciado sesión,
35             y por tanto mostrará en el título del activity el nombre del administrador que ha sido
36             seleccionado.
37         */
38         if (admin != "") {
39             db.collection("Userios").document(admin).get().addOnCompleteListener { task<DocumentSnapshot> {
40                 title = it.result?.get("Nombre").toString()
41             }
42         }
43
44         if (!chatId.isNotEmpty() && user.isNotEmpty()) {
45
46             if (chatId.isNotEmpty() && user.isNotEmpty()) {
47                 initViews()
48             }
49
50             /**
51             * Function que inicializa la vista
52             */
53             private fun initViews() {
54                 messagesRecyclerView.layoutManager = LinearLayoutManager(context = this)
55                 messagesRecyclerView.adapter = MessageAdapter(user)
56
57                 sendMessageButton.setOnClickListener { view: View? ->
58                     sendMessage()
59                 }
60
61                 val chatRef = db.collection("Chats").document(chatId)
62
63                 chatRef.collection("messages").orderBy("dob", Query.Direction.ASCENDING)
64                     .get()
65                     .addOnSuccessListener { messages ->
66                         val listMessages = messages.toObjects(Message::class.java)
67                         (messagesRecyclerView.adapter as MessageAdapter).setListData(listMessages)
68                     }
69
70                 chatRef.collection("messages").orderBy("dob", Query.Direction.ASCENDING)
71                     .addSnapshotListener { messages, error ->
72                         if (error == null) {
73                             messages?.let { it: QuerySnapshot? ->
74                                 val listMessages = it.toObjects(Message::class.java)
75                                 (messagesRecyclerView.adapter as MessageAdapter).setListData(listMessages)
76                             }
77                         }
78                     }
79
80             }
81             /**
82             * Function que envía el mensaje
83             */
84             private fun sendMessage() {
85                 val message = Message()
86                 message = messageTextField.text.toString()
87                 from = user
88             }
89         }
90     }
91 }
```

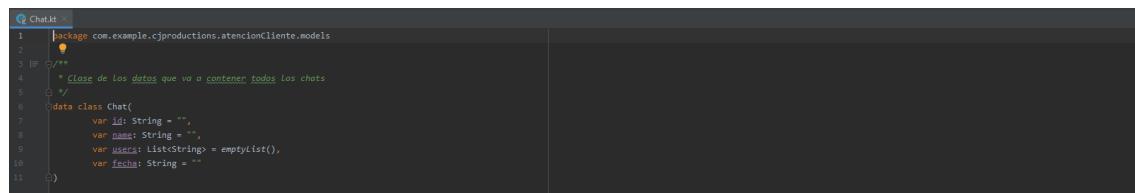
```
19
20     db.collection( collectionPath: "chats" ).document( chatId ).collection( collectionPath: "messages" ).document().set( message )
21
22     messageTextField.setText( "" )
23 }
24
25 /**
26 * Función que se ejecuta cuando se pulse el botón de retroceder
27 * al momento de finalizar el activity
28 */
29 override fun onBackPressed() {
30     val correosDesarrolladores = arrayListOf<String>("carlosjmsanchez@gmail.com")
31
32     //Este if evitará que aparezca el dialogo de cerrar ventana cuando el correo que hay es de un admin
33     if (!correosDesarrolladores.contains(user))
34         showAlertDialog("Si cierra la ventana del chat, tendrá que volver a entrar...")
35     else
36         finish()
37 }
38
39 /**
40 * Función que genera un mensaje de alerta en la pantalla
41 * con el texto que se le pase por parámetro
42 */
43 private fun showAlertDialog(mensaje: String) {
44     val builder = AlertDialog.Builder( context: this )
45     builder.setMessage( mensaje )
46     builder.setPositiveButton("SI") { _, _ ->
47         finish()
48     }
49     builder.setNegativeButton("NO") { view, _ ->
50         view.dismiss()
51     }
52     val dialog = builder.create()
53     dialog.show()
54 }
55 }
```

ChatAdapter.kt



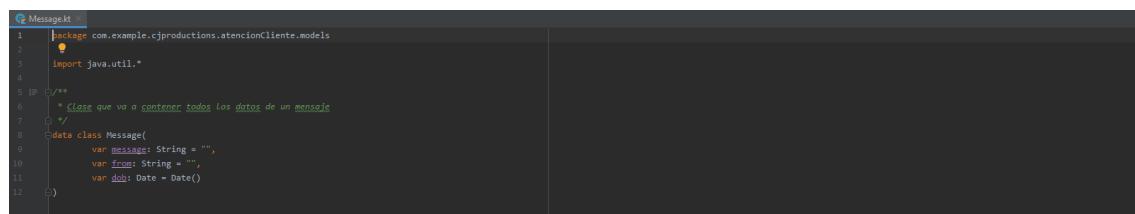
```
ChatAdapter.kt
1 package com.example.cjproductions.atencionCliente.adapters
2
3 import android.view.LayoutInflater
4 import android.view.View
5 import android.view.ViewGroup
6 import androidx.recyclerview.widget.RecyclerView
7 import com.example.cjproductions.R
8 import com.example.cjproductions.atencionCliente.models.Chat
9 import kotlin.android.synthetic.main.item_chat.view.*
10
11 /**
12  * Clase que gestiona el recycler view de la lista de chats
13  */
14 class ChatAdapter(val chatClick: (Chat) -> Unit) : RecyclerView.Adapter<ChatAdapter.ChatViewHolder>() {
15     var chats: List<Chat> = emptyList()
16
17     fun setData(list: List<Chat>) {
18         chats = list
19         notifyDataSetChanged()
20     }
21
22     /**
23      * Método que crea la vista del recycler pasando el layout correspondiente
24      */
25     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ChatViewHolder {
26         return ChatViewHolder(LayoutInflater.from(parent.context).inflate(R.layout.item_chat, parent, attachToRoot: false))
27     }
28
29     override fun onBindViewHolder(holder: ChatViewHolder, position: Int) {
30
31         holder.itemView.chatNameText.text = chats[position].name
32         holder.itemView.userTextview.text = chats[position].fecha
33
34         holder.itemView.setOnClickListener { v: View? ->
35             chatClick(chats[position])
36         }
37     }
38
39     override fun getItemCount(): Int {
40         return chats.size
41     }
42
43     class ChatViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView)
```

Chat.kt



```
Chat.kt
1 package com.example.cjproductions.atencionCliente.models
2
3 /**
4  * Clase de los datos que va a contener todos los chats
5  */
6 data class Chat(
7     var id: String = "",
8     var name: String = "",
9     var users: List<String> = emptyList(),
10    var fecha: String = "")
11
```

Message.kt



```
Message.kt
1 package com.example.cjproductions.atencionCliente.models
2
3 import java.util.*
4
5 /**
6  * Clase que va a contener todos los datos de un mensaje
7  */
8 data class Message(
9     var message: String = "",
10    var from: String = "",
11    var date: Date = Date())
12
```

MessageAdapter.kt

```
MessageAdapter.kt
1 package com.example.cjproductions.attencionCliente.adapters
2
3 import android.view.LayoutInflater
4 import android.view.View
5 import android.view.ViewGroup
6 import androidx.recyclerview.widget.RecyclerView
7 import com.example.cjproductions.attencionCliente.models.Message
8 import com.example.cjproductions.R
9 import kotlin.android.synthetic.main.item_message.view.*
10
11 /**
12  * Clase que gestiona el recycler view de los mensajes
13  */
14
15 class MessageAdapter(private val user: String) : RecyclerView.Adapter<MessageAdapter.MessageViewHolder>() {
16
17     private var messages: List<Message> = emptyList()
18
19     fun setData(list: List<Message>) {
20         messages = list
21         notifyDataSetChanged()
22     }
23
24     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MessageViewHolder {
25         return MessageViewHolder(LayoutInflater.from(parent.context).inflate(R.layout.item_message, parent, attachToRoot: false))
26     }
27
28     override fun onBindViewHolder(holder: MessageViewHolder, position: Int) {
29         val message = messages[position]
30
31         if (user == message.from) {
32             holder.itemView.myMessageLayout.visibility = View.VISIBLE
33             holder.itemView.otherMessageLayout.visibility = View.GONE
34
35             holder.itemView.myMessageTextview.text = message.message
36         } else {
37             holder.itemView.myMessageLayout.visibility = View.GONE
38             holder.itemView.otherMessageLayout.visibility = View.VISIBLE
39
40             holder.itemView.othersMessageTextview.text = message.message
41         }
42     }
43
44     override fun getItemCount(): Int {
45         return messages.size
46     }
47
48     class MessageViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView)
49 }
```

MenuPrincipalAdmin.kt

```
1 package com.example.cjproductions.atencionCliente.admin
2
3 import android.content.Context
4 import android.content.Intent
5 import android.content.SharedPreferences
6 import android.os.Bundle
7 import androidx.appcompat.app.AppCompatActivity
8 import androidx.recyclerview.widget.LinearLayoutManager
9 import com.example.cjproductions.R
10 import com.example.cjproductions.atencionCliente.activities.ChatActivity
11 import com.example.cjproductions.atencionCliente.adapters.ChatAdapter
12 import com.example.cjproductions.atencionCliente.models.Chat
13 import com.google.firebaseio.firebaseio.ktx.firebaseio
14 import com.google.firebaseio.ktx.Firebase
15 import kotlin.android.synthetic.main.activity_menu_principal_admin.*
16
17 class MenuPrincipalAdmin : AppCompatActivity() {
18     private var user = ""
19
20     private var db = firebase.firebaseio
21
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24         setContentView(R.layout.activity_menu_principal_admin)
25
26         title = "Atención al cliente"
27
28         //guardamos en la variable 'user' el correo que se encuentre en la persistencia
29         val prefs: SharedPreferences = getSharedPreferences("com.example.cjproductions.PREFERENCIAS", Context.MODE_PRIVATE)
30         user = prefs.getString("key_email", defaultValue null).toString()
31
32         //si el usuario no es nulo, iniciamos la vista
33         if (user.isNotEmpty()) {
34             initViews()
35         }
36     }
37
38     /**
39      * Función que inicializa la vista de la ventana
40      * Mostrando los chats que tendremos disponibles
41      *
42      * El admin no puede iniciar conversación, solo pueden hacerlo
43      * los usuarios que no sean admins
44      */
45     private fun initViews() {
46
47         private fun initViews() {
48             listChatsRecyclerView.layoutManager = LinearLayoutManager(context: this)
49             listChatsRecyclerView.adapter =
50                 ChatAdapter { Chat ->
51                     chatSelected(Chat)
52                 }
53
54             val userRef = db.collection("collectionPath: \"Usuarios\"").document(user)
55
56             userRef.collection("collectionPath: \"Chats\").get().addOnSuccessListener { Chats ->
57                 val listChats = Chats.toObjects(Chat::class.java)
58
59                 (listChatsRecyclerView.adapter as ChatAdapter).setData(listChats)
60             }
61
62             //Se añade este listener para capturar cualquier cambio en la base de datos y lo muestra en la pantalla
63             userRef.collection("collectionPath: \"Chats\").
64                 addSnapshotListener { Chats, error ->
65                     if (error == null) {
66                         Chats.let { it: QuerySnapshot
67                             val listChats = it.toObjects(Chat::class.java)
68
69                             (listChatsRecyclerView.adapter as ChatAdapter).setData(listChats)
70                         }
71                     }
72                 }
73             }
74
75             /**
76              * Función que inicia un nuevo activity con el chat que hemos seleccionado
77              * Pasando como parámetros el id del chat y el usuario
78              */
79             private fun chatSelected(chat: Chat) {
80                 val intent = Intent(packageName: this, ChatActivity::class.java)
81                 intent.putExtra(name: "chatId", chat.id)
82                 intent.putExtra(name: "user", user)
83                 startActivity(intent)
84             }
85         }
86     }
87 }
```

MenuPrincipalComprarProductos.kt

```
1 package com.example.cjproductions.comprarProductos
2
3 import android.content.Intent
4 import android.os.Bundle
5 import androidx.appcompat.app.AppCompatActivity
6 import androidx.recyclerview.widget.LinearLayoutManager
7 import com.example.cjproductions.
8 import com.example.cjproductions.comprarProductos.activity.ProductosActivity
9 import com.example.cjproductions.comprarProductos.adapter.ProductosAdapter
10 import com.example.cjproductions.comprarProductos.modelo.Productos
11 import com.google.firebase.firestore.FirebaseFirestore
12 import kotlinx.android.synthetic.main.activity_menu_principal_comprar_productos.*
13
14 class MenuPrincipalComprarProductos : AppCompatActivity() {
15     private var db = FirebaseFirestore.getInstance()
16
17     override fun onCreate(savedInstanceState: Bundle?) {
18         super.onCreate(savedInstanceState)
19         setContentView(R.layout.activity_menu_principal_comprar_productos)
20
21         title = "Comprar productos"
22
23         initViews()
24     }
25
26     private fun initViews() {
27         listComprarRecycler.layoutManager = LinearLayoutManager(context = this)
28         listComprarRecycler.adapter =
29             ProductosAdapter { productos -
30                 productosSelected(productos)
31             }
32
33         db.collection( collectionPath: "Productos").get().addOnSuccessListener { productos ->
34             val listaDeProductos = productos.toObjects(Productos::class.java)
35
36             (listComprarRecycler.adapter as ProductosAdapter).setData(listaDeProductos)
37         }
38
39         //Se añade este listener para capturar cualquier cambio en la base de datos y lo muestra en la pantalla
40         db.collection( collectionPath: "Productos")
41             .addSnapshotListener { productos, error -
42                 if (error == null) {
43                     productos?.let { it: QuerySnapshot -
44                         products?.let {
45                             val listaDeProductos = it.toObjects(Productos::class.java)
46
47                             (listComprarRecycler.adapter as ProductosAdapter).setData(listaDeProductos)
48                         }
49                     }
50                 }
51             }
52     }
53
54     private fun productosSelected(productos: Productos) {
55         val intent = Intent( packageContext, this, ProductosActivity::class.java)
56         intent.putExtra( name: "enlace", productos.enlace)
57         intent.putExtra( name: "nombre", productos.nombre)
58         startActivity(intent)
59     }
60
61 }
62
```

Productos.kt

```
1 package com.example.cjproductions.comprarProductos.modelo
2
3 data class Productos(
4     val nombre: String = "",
5     val imagen: String = "",
6     val descripcion: String = "",
7     val enlace: String = ""
8 )
```

ProductosActivity.kt

```
1 package com.example.cjproductions.comprarProductos.activity
2
3 import android.graphics.Bitmap
4 import androidx.appcompat.app.AppCompatActivity
5 import android.os.Bundle
6 import android.webkit.WebChromeClient
7 import android.webkit.WebResourceRequest
8 import android.webkit.WebView
9 import android.webkit.WebViewClient
10 import com.example.cjproductions.R
11 import kotlinx.android.synthetic.main.activity_productos.*
12
13 class ProductsActivity : AppCompatActivity() {
14
15     private var enlace: String = ""
16
17     override fun onCreate(savedInstanceState: Bundle?) {
18         super.onCreate(savedInstanceState)
19         setContentView(R.layout.activity_productos)
20
21         intent.getStringExtra("enlace")?.let { enlace = it }
22         intent.getStringExtra("nombre")?.let { titulo = it }
23
24         //Swipe Refresh
25         swipeRefresh.setOnRefreshListener {
26             webView.reload()
27         }
28
29         webView.webChromeClient = object : WebChromeClient() {
30
31             override fun onPageStarted(view: WebView?, url: String?, favicon: Bitmap?) {
32                 super.onPageStarted(view, url, favicon)
33
34                 swipeRefresh.isRefreshing = true
35             }
36
37             override fun onPageFinished(view: WebView?, url: String?) {
38                 super.onPageFinished(view, url)
39
40                 swipeRefresh.isRefreshing = false
41             }
42
43             val settings = webView.settings
44             settings.javaScriptEnabled = true
45
46             webView.loadUrl(enlace)
47         }
48
49         override fun onBackPressed() {
50             if (webView.canGoBack()) {
51                 webView.goBack()
52             } else {
53                 super.onBackPressed()
54             }
55         }
56     }
57 }
```

ProductosAdapter.kt

```
1 package com.example.cjproductions.comprarProductos.adapter
2
3 import android.view.LayoutInflater
4 import android.view.View
5 import android.view.ViewGroup
6 import androidx.recyclerview.widget.RecyclerView
7 import com.example.cjproductions.R
8 import com.example.cjproductions.comprarProductos.modelo.Productos
9 import com.google.firebase.storage.FirebaseStorage
10 import com.squareup.picasso.Picasso
11 import kotlinx.android.synthetic.main.item_producto.view.*
12
13 class ProductosAdapter(val productoClick: (Productos) -> Unit) : RecyclerView.Adapter<ProductosViewHolder>() {
14
15     var listProductos: List<Productos> = emptyList()
16     private val storage = FirebaseStorage.getInstance().reference
17
18     fun setData(list: List<Productos>) {
19         listProductos = list
20         notifyDataSetChanged()
21     }
22
23     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ProductosViewHolder {
24         return ProductosViewHolder(LayoutInflater.from(parent.context).inflate(R.layout.item_producto, parent, attachToRoot: false))
25     }
26
27     override fun onBindViewHolder(holder: ProductosViewHolder, position: Int) {
28         val perfilReferencia = storage.child(listProductos[position].imagen)
29
30         holder.itemView.tvTitulo.text = listProductos[position].nombre
31         holder.itemView.tvDescripcion.text = listProductos[position].descripcion
32
33         //coloca la imagen desde el almacenamiento de firebase
34         if (perfilReferencia != null) {
35             perfilReferencia.downloadUrl.addOnSuccessListener { url: Uri -
36                 Picasso.get().load(url).resize( targetWidth: 300, targetHeight: 300).centerCrop().into(holder.itemView.logoImageView)
37             }
38         }
39
40         holder.itemView.setOnClickListener { v: View! -
41             productoClick(listProductos[position])
42         }
43     }
44
45     override fun getItemCount(): Int {
46         return listProductos.size
47     }
48
49     class ProductosViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView)
50 }
```

Novedades.kt

```
1 package com.example.cjproductions.novedades.modelo
2
3 data class Novedades(
4     val titulo:String = "",
5     val descripcion:String = "",
6     val imagen:String = "",
7     val fecha:String = ""
8 )
```

NovedadesAdapter.kt

```
NovedadesAdapter.kt
1 package com.example.cjproductions.novedades.adapter
2
3 import android.view.LayoutInflater
4 import android.view.View
5 import android.view.ViewGroup
6 import androidx.recyclerview.widget.RecyclerView
7 import com.example.cjproductions.R
8 import com.example.cjproductions.novedades.modelo.Novedades
9 import com.google.firebase.storage.FirebaseStorage
10 import com.squareup.picasso.Picasso
11 import kotlin.android.synthetic.main.item_novedad.view.*
12
13 class NovedadesAdapter : RecyclerView.Adapter<NovedadesViewHolder>() {
14
15     private val storage = FirebaseStorage.getInstance().reference
16     private var listaNovedades: List<Novedades> = emptyList()
17
18     fun setData(lista: List<Novedades>) {
19         listaNovedades = lista
20         notifyDataSetChanged()
21     }
22
23     /**
24      * Creamos la vista
25      */
26     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): NovedadesViewHolder {
27         return NovedadesViewHolder(LayoutInflater.from(parent.context).inflate(R.layout.item_novedad, parent, attachToRoot: false))
28     }
29
30
31     /**
32      * Aquí ponemos lo que queremos mostrar en cada celda del recycler view
33      */
34     override fun onBindViewHolder(holder: NovedadesViewHolder, position: Int) {
35         val perfilReferencia = storage.child(listaNovedades[position].imagen)
36
37         holder.itemView.tvTituloNovedad.text = listaNovedades[position].titulo
38         holder.itemView.tvDescripcionNovedad.text = listaNovedades[position].descripcion
39         holder.itemView.tvFechaNovedad.text = listaNovedades[position].fecha
40
41         //Coloca la imagen desde el almacenamiento de firebase
42         if (perfilReferencia != null) {
43             perfilReferencia.downloadUrl.addOnSuccessListener { url: Uri? ->
44                 Picasso.get().load(url).resize( targetWidth: 500, targetHeight: 400).centerCrop().into(holder.itemView.imagenNovedad)
45             }
46         }
47     }
48
49     override fun getItemCount(): Int {
50         return listaNovedades.size
51     }
52
53     class NovedadesViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView)
54 }
```

MenuPrincipalNovedades.kt

```
1  package com.example.cjproductions.novedades
2
3  import android.os.Bundle
4  import androidx.appcompat.app.AppCompatActivity
5  import androidx.recyclerview.widget.LinearLayoutManager
6  import androidx.recyclerview.widget.RecyclerView
7  import com.example.cjproductions.R
8  import com.example.cjproductions.novedades.adapter.NovedadesAdapter
9  import com.example.cjproductions.novedades.modelo.Novedades
10 import com.google.firebase.firestore.FirebaseFirestore
11 import kotlinx.android.synthetic.main.activity_menu_principal_novedades.*
12
13 class MenuPrincipalNovedades : AppCompatActivity() {
14
15     private var db = FirebaseFirestore.getInstance()
16
17     override fun onCreate(savedInstanceState: Bundle?) {
18         super.onCreate(savedInstanceState)
19         setContentView(R.layout.activity_menu_principal_novedades)
20
21         iniciarVista()
22     }
23
24     /**
25      * Funcion que se encarga de cargar la vista del activity
26      */
27     private fun iniciarVista() {
28
29         /*
30          Seleccionamos el tipo de layout y asignamos el adapter
31         */
32         recyclerViewNovedades.layoutManager = LinearLayoutManager(context = this, RecyclerView.HORIZONTAL, false)
33         recyclerViewNovedades.adapter = NovedadesAdapter()
34
35         db.collection( collectionPath: "Novedades").get().addOnSuccessListener { novedades ->
36             val listaDeNovedades = novedades.toObjects(Novedades::class.java)
37
38             (recyclerViewNovedades.adapter as NovedadesAdapter).setListData(listaDeNovedades)
39         }
40
41         //Se añade este listener para capturar cualquier cambio en la base de datos y lo muestra en la pantalla
42         db.collection( collectionPath: "Novedades")
43             .addSnapshotListener { novedades, error ->
44                 if (error == null) {
45                     ...
46                 }
47             }
48
49         if (error == null) {
50             novedades?.let {
51                 val listaDeNovedades = it.toObjects(Novedades::class.java)
52
53                 (recyclerViewNovedades.adapter as NovedadesAdapter).setListData(listaDeNovedades)
54             }
55         }
56     }
57 }
```

InicioSesion.kt

```
1 package com.example.cjproductions.login
2
3 import android.content.Intent
4 import android.net.Uri
5 import android.os.Bundle
6 import android.util.Patterns
7 import android.widget.Toast
8 import androidx.appcompat.app.AlertDialog
9 import androidx.appcompat.app.AppCompatActivity
10 import com.example.cjproductions.R
11 import com.google.android.gms.auth.api.signin.GoogleSignIn
12 import com.google.android.gms.auth.api.signin.GoogleSignInAccount
13 import com.google.android.gms.auth.api.signin.GoogleSignInClient
14 import com.google.android.gms.auth.api.signin.GoogleSignInOptions
15 import com.google.android.gms.common.api.ApiException
16 import com.google.android.gms.tasks.Task
17 import com.google.firebase.auth.AuthCredential
18 import com.google.firebase.auth.FirebaseAuth
19 import com.google.firebase.auth.GoogleAuthProvider
20 import com.google.firebaseio.firestore.FirebaseFirestore
21 import com.google.firebaseio.ktx.firebaseio
22 import com.google.firebaseio.storage.StorageReference
23 import com.google.firebaseio.storage.ktx.storage
24 import kotlinx.android.synthetic.main.activity_editar_usuarios.*
25 import kotlinx.android.synthetic.main.activity_inicio_sesion.*
26 import java.util.regex.Pattern
27
28
29 class InicioSesion : AppCompatActivity() {
30
31     private val db = FirebaseFirestore.getInstance()
32     lateinit var storageReference: StorageReference
33
34     override fun onCreate(savedInstanceState: Bundle?) {
35         super.onCreate(savedInstanceState)
36         setContentView(R.layout.activity_inicio_sesion)
37         title = "Iniciar Sesión" //Cambia el título de la ventana
38
39         //-----
40         val storage= Firebase.storage
41         storageReference=storage.reference
42         //-----
43
44         ponerListeners()
45
46
47
48     private fun ponerListeners() {
49         btLogin.setOnClickListener { v: View ->
50             if (!comprobar()) return@setOnClickListener //Se comprueba que todos los campos están correctos antes de continuar
51
52             FirebaseAuth.getInstance().signInWithEmailAndPassword(
53                 etEmail.text.toString(),
54                 etContrasena.text.toString()
55             ).addOnCompleteListener { task: Task<AuthResult>
56
57                 if (task.isSuccessful) {
58                     Toast.makeText(context, "Se ha iniciado sesión correctamente", Toast.LENGTH_LONG).show()
59                     val returnIntent = Intent()
60                     returnIntent.putExtra("email", etEmail.text.toString().trim())
61                     setResult(Activity.RESULT_OK, returnIntent)
62                     finish()
63                 } else {
64                     showAlert("Ha ocurrido un error durante el inicio de sesión")
65                 }
66             }
67         }
68
69         btLogin.setOnLongClickListener { v: View ->
70             Toast.makeText(context, "Pulsa para iniciar sesión", Toast.LENGTH_SHORT).show()
71             true //setOnLongClickListener
72         }
73
74
75     //Botón registrarse con Google
76
77     btRegistrarGoogle.setOnClickListener { v: View ->
78         Toast.makeText(context, "Pulsa para registrarte usando una cuenta de correo electrónico", Toast.LENGTH_SHORT).show()
79         true //setOnLongClickListener
80     }
81
82     btRegistrarGoogle.setOnClickListener { v: View ->
83         val googleConf: GoogleSignInOptions = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
84             .requestIdToken("749715430420-crae0g5dk887gm045ectad1rnzj8o.apps.goog...")
85             .requestEmail()
86             .build()
87     }
88 }
```

```

17     .build()
18
19     val googleClient: GoogleSignInClient = GoogleSignIn.getClient(this, googleConf)
20
21     googleClient.signOut() //se ejecuta por si existe por si existe alguna cuenta iniciada
22
23     startActivityForResult(googleClient.signInIntent, requestCode: 200)
24 }
25
26
27 //BOTON RESTABLECER CONTRASEÑA
28
29 btRestablecerContrasena.setOnClickListener { v: View? ->
30     val intent = Intent(packageContext: this, RestablecerPassword::class.java)
31     startActivity(intent)
32 }
33
34
35
36 /**
37 * Este metodo esta principalmente por el inicio de sesion mediante google
38 * Aqui se realiza el inicio de sesion, controlando que pueda haber un error o capturandolo
39 */
40 override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
41     super.onActivityResult(requestCode, resultCode, data)
42     if (requestCode == 200) {
43         val task: Task<GoogleSignInAccount> = GoogleSignIn.getSignedInAccountFromIntent(data)
44
45         try {
46             val account: GoogleSignInAccount? = task.getResult(ApiException::class.java)
47
48             if (account != null) {
49                 val credential: AuthCredential = GoogleAuthProvider.getCredential(account.idToken, accessToken: null)
50
51                 FirebaseAuth.getInstance().signInWithCredential(credential).addOnCompleteListener { it: Task<AuthResult>?
52
53                     if (it.isSuccessful) {
54
55                         val returnIntent = Intent()
56                         returnIntent.putExtra("name": "email", account.email)
57                         setResult(RESULT_OK, returnIntent)
58
59                         Toast.makeText(context: this, "Se ha iniciado sesion correctamente", Toast.LENGTH_LONG).show()
60
61                         Toast.makeText(context: this, "Se ha iniciado sesion correctamente", Toast.LENGTH_LONG).show()
62
63                         //Creamos el documento en firestore
64                         val textoDefault: String = "No proporcionado"
65
66                         /**
67                         * Este bloque de codigo realiza lo siguiente:
68                         * Primero obtiene la informacion de la base de datos
69                         * correspondiente al email.
70
71                         * Una vez termina (it.isComplete), si el resultado es distinto de true,
72                         * quiere decir que no existe ese correo en la base de datos, por lo que procede a crearlo.
73
74                         * Por el contrario, si es True, quiere decir que ese email ya esta en la base de datos
75                         * y por lo tanto no hace nada ya que no queremos modificarlo
76                         */
77                         db.collection(collectionPath: "Usuarios").document(account.email.toString()).get().addOnCompleteListener { it: Task<DocumentSnapshot>?
78                             if (it.isComplete) {
79
80                                 if (it.getResult()?.exists() != true) {
81
82                                     subirImagenDefecto() //Se sube la imagenes por defecto
83
84                                     db.collection(collectionPath: "Usuarios").document(account.email.toString())
85                                         .set(hashMapOf("Nombre" to textoDefault, "Telefono" to textoDefault))
86
87                                 }
88
89                             }
90                         }
91                         finish()
92                     } else {
93                         showAlert(mensaje: "Ha ocurrido un error durante el inicio de sesion")
94                     }
95                 }
96             }
97         } catch (e: ApiException) {
98             showAlert(mensaje: "No se ha podido recuperar la cuenta")
99         }
100     }
101 }
102
103 /**
104 */

```

```

1.1 /**
1.2 * Función que comprueba que los campos de texto no están vacíos
1.3 */
1.4
1.5 private fun comprobar(): Boolean {
1.6     var email = etEmail.text.toString().trim()
1.7     var contraseña = etContraseña.text.toString().trim()
1.8
1.9     if (!isOk(email)) {
1.10         etEmail.error = "Rellene el campo %s".format(...args: "Email")
1.11         return false
1.12     }
1.13
1.14     if (!isOk(contraseña)) {
1.15         etContraseña.error = "Rellene el campo %s".format(...args: "Contraseña")
1.16         return false
1.17     }
1.18
1.19     if (contraseña.length < 6) {
1.20         showAlert("La contraseña tiene que tener mínimo 6 caracteres")
1.21         return false
1.22     }
1.23
1.24     //Comprueba que el email introducido es válido (ojo, puede no existir pero ser válido)
1.25     val patronEmail: Pattern = Patterns.EMAIL_ADDRESS
1.26
1.27     if (!patronEmail.matcher(email).matches()) {
1.28         showAlert("El email no tiene un formato válido")
1.29         return false
1.30     }
1.31
1.32     //Si llega hasta aquí quiere decir que todos los campos son válidos
1.33     return true
1.34 }
1.35
1.36 /**
1.37 * Función que comprueba si una cadena está vacía
1.38 * Dicha cadena se pasa como parámetro
1.39 */
1.40 private fun isOk(cadena: String): Boolean {
1.41     return !cadena.isEmpty()
1.42 }
1.43
1.44 private fun showAlert(message: String) {
1.45     val builder = AlertDialog.Builder(context = this)
1.46     builder.setTitle("ERROR")
1.47     builder.setMessage(message)
1.48
1.49     builder.setPositiveButton(text: "Aceptar", listener: null)
1.50     val dialog = builder.create()
1.51     dialog.show()
1.52 }
1.53
1.54 /**
1.55 * Método que subirá la imagen que ha elegido el usuario a la base de datos
1.56 */
1.57 private fun subirImagenDefecto() {
1.58     val referencia = storageReference.child(pathString: "usuarios/" + FirebaseAuth.getInstance().currentUser.email.toString() + "/profile.jpg")
1.59     referencia.putFile(Uri.parse(unString: "android.resource://${packageName}/${R.mipmap.user_default}"))
1.60
1.61     /*
1.62     .addOnSuccessListener {
1.63         @Override
1.64         fun onSuccess(taskSnapshot: UploadTask.TaskSnapshot) {
1.65             referencia.downloadUrl.addOnSuccessListener {
1.66                 @Override
1.67                 fun onSuccess(url: Uri) {
1.68                     Picasso.get().load(url).into(editorImagen)
1.69                 }
1.70             }
1.71         }
1.72     }
1.73 }
1.74

```

Registrarse.kt

```
1 package com.example.cjproductions.login
2
3 import android.net.Uri
4 import android.os.Bundle
5 import android.util.Patterns
6 import android.widget.Toast
7 import androidx.appcompat.app.AlertDialog
8 import androidx.appcompat.app.AppCompatActivity
9 import com.example.cjproductions.R
10 import com.google.firebase.auth.FirebaseAuth
11 import com.google.firebase.firestore.FirebaseFirestore
12 import com.google.firebaseio.ktx.firebaseio
13 import com.google.firebase.storage.StorageReference
14 import com.google.firebase.storage.ktx.storage
15 import kotlinx.android.synthetic.main.activity_editar_usuarios.*
16 import kotlinx.android.synthetic.main.activity_inicio_sesion.*
17 import kotlinx.android.synthetic.main.activity_registrarse.*
18 import kotlinx.android.synthetic.main.activity_registrarse.etContrasena
19 import kotlinx.android.synthetic.main.activity_registrarse.etEmail
20 import java.util.regex.Pattern
21
22
23 class Registrarse : AppCompatActivity() {
24
25     private val db = FirebaseFirestore.getInstance()
26     lateinit var storageReference: StorageReference
27
28     override fun onCreate(savedInstanceState: Bundle?) {
29         super.onCreate(savedInstanceState)
30         setContentView(R.layout.activity_registrarse)
31
32         title = "Registrarse"
33
34         //-----
35         val storage= Firebase.storage
36         storageReference=storage.reference
37         //-----
38
39         ponerListeners()
40     }
41
42     private fun ponerListeners() {
43
44         //Boton registrarse
45
46         btRegistrarse.setOnLongClickListener { (View) ->
47             if (!comprobar()) return@setOnLongClickListener true
48
49             btRegistrarse.setOnClickListener { (View) ->
50                 if (!comprobar()) return@setOnClickListener
51
52                 FirebaseAuth.getInstance().createUserWithEmailAndPassword(
53                     etEmail.text.toString(),
54                     etContrasena.text.toString()
55                 ).addOnCompleteListener { (Task<AuthResult>)
56
57                     if (it.isSuccessful) {
58                         val textoDefault: String = "No proporcionado"
59
60                         subirImagenDefecto() //se sube la imagen por defecto
61
62                         Toast.makeText(context, "Usuario registrado correctamente", Toast.LENGTH_LONG).show()
63
64                         //Se crea la colección junto con el registro
65                         db.collection("Usuarios").document(etEmail.text.toString()).set(hashMapOf("Nombre" to textoDefault, "Telefono" to textoDefault))
66
67                         onBackPressed()
68                     } else {
69                         showAlert("Ha ocurrido un error durante la creación del usuario")
70                     }
71                 }
72             }
73         }
74     }
75
76     /**
77      * Función que comprueba que los campos de texto no están vacíos
78     */
79     private fun comprobar(): Boolean {
80
81         var email = etEmail.text.toString().trim()
82         var contraseña = etContrasena.text.toString().trim()
83         var contraseña2 = etContrasena2.text.toString().trim()
84
85         if (!isOk(email)) {
86             etEmail.setError("Rellene el campo %s",format("%s", "Email"))
87             return false
88         }
89
90         if (!contraseña.equals(contraseña2)) {
91             etContrasena.setError("Las contraseñas no coinciden")
92             etContrasena2.setError("Las contraseñas no coinciden")
93             return false
94         }
95
96         return true
97     }
98
99     private fun isOk(texto: String): Boolean {
100         return !Patterns.EMAIL_ADDRESS.matcher(texto).matches()
101     }
102 }
```

```

36     etEmail.error = "Rellene el campo %s",format( ...arg0: "email")
37     return false
38   }
39   if (!isOk(contrasena)) {
40     etContrasena.error = "Rellene el campo %s",format( ...arg0: "Contraseña")
41     return false
42   }
43   if (!isOk(contrasena2)) {
44     etContrasena2.error = "Rellene el campo %s",format( ...arg0: "Confirmar Contraseña")
45     return false
46   }
47
48   //Comprueba que los dos campos de las contraseñas son iguales
49   if (contrasena == contrasena2) {
50     showAlert("Las contraseñas no son iguales")
51     return false
52   }
53
54   if (contrasena.length < 6) {
55     showAlert("La contraseña tiene que tener mínimo 6 caracteres")
56     return false
57   }
58
59   //Comprueba que el email introducido es válido (ojo, puede no existir pero ser válido)
60   val patronEmail: Pattern = Patterns.EMAIL_ADDRESS
61
62   if (!patronEmail.matcher(etEmail.text).matches()) {
63     showAlert("El email no tiene un formato válido")
64     return false
65   }
66
67   //SI llega hasta aquí quiere decir que todos los campos son válidos
68   return true
69 }
70
71 /**
72 * Función que comprueba si una cadena está vacía
73 * Ojalá la cadena se pasé como parámetro
74 */
75 private fun isOk(cadena: String): Boolean {
76   return !cadena.isEmpty()
77 }
78
79 /**
80 * Función que genera un mensaje de alerta en la pantalla
81 * con el texto que se le pase por parámetro
82 */
83 private fun showAlert(mensaje: String) {
84   val builder = AlertDialog.Builder(context, this)
85   builder.setTitle("ERROR")
86   builder.setMessage(mensaje)
87   builder.setPositiveButton(text: "Aceptar", listener: null)
88   val dialog = builder.create()
89   dialog.show()
90 }
91
92 /**
93 * Método que subió la imagen que ha elegido el usuario a la base de datos
94 */
95 private fun subirImagenDefecto() {
96   val referencia = storageReference.child(pathString: "usuarios/${etEmail.text.trim()}/profile.jpg")
97   referencia.putFile(Uri.parse(unString: "android.resource://${packageName}/${R.mipmap.user_default}"))
98   /*
99    * .addOnSuccessListener {
100    *   @Override
101    *   fun onSuccess(taskSnapshot: UploadTask.TaskSnapshot) {
102    *     referencia.downloadUrl.addOnSuccessListener {
103    *       @Override
104    *       fun onSuccess(uri: Uri) {
105    *         Picasso.get().load(uri).into(editarImagen)
106    *       }
107    *     }
108    *   }
109    * }
110  */
111 }

```

RestablecerPassword.kt

```
1 package com.example.cjproductions.login
2
3 import android.os.Bundle
4 import android.util.Patterns
5 import android.widget.Toast
6 import androidx.appcompat.app.AlertDialog
7 import androidx.appcompat.app.AppCompatActivity
8 import com.example.cjproductions.R
9 import com.google.firebase.auth.FirebaseAuth
10 import kotlin.android.synthetic.main.activity_restablecer_password.*
11 import java.util.regex.Pattern
12
13 class RestablecerPassword : AppCompatActivity() {
14     override fun onCreate(savedInstanceState: Bundle?) {
15         super.onCreate(savedInstanceState)
16         setContentView(R.layout.activity_restablecer_password)
17
18         title = "Restablecer contraseña"
19
20         ponerListeners()
21     }
22
23     private fun ponerListeners() {
24         btRestablecerContrasena.setOnClickListener { v: View? ->
25             if (!comprobarEmail()) return@setOnClickListener
26
27             cambiarContrasena()
28         }
29     }
30
31     /**
32      * Función que comprueba que el campo del email no esté vacío y sea válido
33      * (Puede no existir pero si es válido)
34      */
35     private fun comprobarEmail(): Boolean {
36
37         if (!isOk(etRestablecerContrasena.text.toString())) {
38             etRestablecerContrasena.error = "Rellene el campo %s".format("Email")
39             return false
40         }
41
42         //Comprueba que el email introducido es valido (ojo, puede no existir pero es valido)
43         val patronEmail: Pattern = Patterns.EMAIL_ADDRESS
44
45         if (!patronEmail.matcher(etRestablecerContrasena.text.toString()).matches())
46
47             if (!patronEmail.matcher(etRestablecerContrasena.text.toString()).matches()) {
48                 showAlert("El email no tiene un formato válido")
49                 return false
50             }
51
52             return true
53         }
54
55         /**
56          * Función que comprueba si una cadena está vacía
57          * Dicha cadena se pasa como parámetro
58          */
59         private fun isOk(cadena: String): Boolean {
60             return !cadena.isEmpty()
61         }
62
63         /**
64          * Función que genera un mensaje de alerta en la pantalla
65          * con el texto que se le pase por parámetro
66          */
67         private fun showAlert(mensaje: String) {
68             val builder = AlertDialog.Builder(context = this)
69             builder.setTitle("AVISO")
70             builder.setMessage(mensaje)
71             builder.setPositiveButton(text = "Aceptar", listener = null)
72             val dialog = builder.create()
73             dialog.show()
74         }
75
76         /**
77          * Función que se encarga de enviar un correo al email introducido
78          */
79         private fun cambiarContrasena() {
80
81             FirebaseAuth.getInstance().sendPasswordResetEmail(etRestablecerContrasena.text.toString()).addOnCompleteListener { task: Task<Void>?
82
83                 if (task.isSuccessful) {
84                     Toast.makeText(context = this, "Se ha enviado un correo para restablecer la contraseña", Toast.LENGTH_SHORT).show()
85                 } else {
86                     showAlert("No se ha podido enviar el correo para restablecer contraseña")
87                 }
88             }
89         }
90     }
```

PerfilUsuarios.kt

```
1 package com.example.cjproductions.perfilusuarios
2
3 import android.app.AlertDialog
4 import android.content.Context
5 import android.content.Intent
6 import android.content.SharedPreferences
7 import android.os.Bundle
8 import android.widget.Toast
9 import androidx.appcompat.app.AppCompatActivity
10 import com.example.cjproductions.R
11 import com.google.firebase.auth.FirebaseAuth
12 import com.google.firebase.firestore.FirebaseFirestore
13 import com.google.firebase.ktx.firebaseio
14 import com.google.firebase.storage.StorageReference
15 import com.google.firebase.storage.ktx.storage
16 import com.squareup.picasso.Picasso
17 import dmax.dialog.SpotsDialog
18 import kotlinx.android.synthetic.main.activity_perfil_usuarios.*
19
20 class PerfilUsuarios : AppCompatActivity() {
21
22     private val db = FirebaseFirestore.getInstance()
23     lateinit var storageReference: StorageReference
24     private lateinit var email:String
25
26     override fun onCreate(savedInstanceState: Bundle?) {
27         val dialog: AlertDialog = SpotsDialog.Builder()
28             .setContext(this)
29             .setMessage("Cargando perfil")
30             .setCancelable(false)
31             .build()
32
33         dialog.show()
34
35         super.onCreate(savedInstanceState)
36         setContentView(R.layout.activity_perfil_usuarios)
37
38         title = "Mi perfil"
39
40         ponerListeners()
41
42         llenarCampos()
43
44         //-----
45         val storage= Firebase.storage
46
47         val storage= Firebase.storage
48         storageReference=storage.reference
49         //-----
50
51         val prefs: SharedPreferences? = getSharedPreferences("com.example.cjproductions.PREFERENCIAS", Context.MODE_PRIVATE)
52         if (prefs != null) {
53             email=prefs.getString(key="email", defaultValue=null).toString()
54         }
55         //-----
56
57         cargarImagen()
58
59         dialog.dismiss()
60     }
61
62     private fun ponerListeners() {
63
64         //BOTON CERRAR SESION
65
66         btCerrarSesion.setOnLongClickListener { view ->
67             Toast.makeText(context, "Pulsa para cerrar sesión", Toast.LENGTH_SHORT).show()
68             true /*setOnLongClickListener*/
69         }
70
71         btCerrarSesion.setOnClickListener { view ->
72             FirebaseAuth.getInstance().signOut()
73             Toast.makeText(context, "Sesión cerrada correctamente", Toast.LENGTH_SHORT).show()
74
75             borrarPreferencias() //borra el correo que hay en el archivo de preferencias
76
77             finish() //Hace finalizar la actividad, volviendo automáticamente al main activity
78         }
79
80         //BOTON EDITAR PERFIL
81
82         btEditarImagen.setOnLongClickListener { view ->
83             Toast.makeText(context, "Pulsa para editar tu perfil", Toast.LENGTH_SHORT).show()
84             true /*setOnLongClickListener*/
85         }
86
87     }
88 }
```

```

17     }
18
19     btEditarImagen.setOnClickListener { (v: View) {
20         val intent = Intent(packageContext, EditarUsuarios::class.java)
21         startActivity(intent)
22     }
23
24     //BOTON CAMBIAR CONTRASEÑA
25
26     btCambiarContrasena.setOnLongClickListener { (v: View) {
27         Toast.makeText(context, "Pulsa para cambiar la contraseña", Toast.LENGTH_SHORT).show()
28         true
29     }
30
31     btCambiarContrasena.setOnOnClickListener { (v: View) {
32         cambiarContrasena()
33     }
34
35 }
36
37 /**
38 * Función que se encarga de llenar los datos del activity con la información correspondiente
39 */
40 private fun llenarCampos() {
41
42     val prefs: SharedPreferences? = getSharedPreferences("com.example.cjproductions.PREFERENCIAS", Context.MODE_PRIVATE)
43     //Rellenar el campo de correo
44     if (prefs != null) {
45         tvCorreo.setText(prefs.getString(key_email, null).toString())
46
47         //Ahora rellenamos los demás campos
48         db.collection(collectionPath("usuarios")).document(tvCorreo.text.toString()).get().addOnCompleteListener { (task: Task<DocumentSnapshot>)
49             tvTitulo.setText(task.result?.get("Nombre").toString())
50             tvTelefono.setText(task.result?.get("Telefono").toString())
51         }
52
53     }
54
55 }
56
57 private fun cargarImagen() {
58
59     val perfilReferencia = storageReference.child(pathString("usuarios/${email}/profile.jpg"))
60
61
62     val perfilReferencia = storageReference.child(pathString("usuarios/${email}/profile.jpg"))
63
64     //Coloca la imagen desde el almacenamiento de firebase
65     if (perfilReferencia != null) {
66         perfilReferencia.downloadUrl.addOnSuccessListener { (uri: Uri) {
67             Picasso.get().load(uri).resize(targetWidth: 150, targetHeight: 150).centerCrop().into(imagenUsuario)
68         }
69     }
70
71
72 /**
73 * Método que borra el contenido del fichero de persistencia
74 */
75 fun borrarPreferencias() {
76     val prefs: SharedPreferences.Editor? = getSharedPreferences("com.example.cjproductions.PREFERENCIAS", Context.MODE_PRIVATE).edit()
77     if (prefs != null) {
78         prefs.clear()
79         prefs.apply()
80     }
81
82 }
83
84 /**
85 * Se implementa este método para actualizar la información
86 * cuando se vuelve desde el activity de editar usuarios
87 */
88 override fun onRestart() {
89     val dialog: AlertDialog = SpotsDialog.Builder()
90         .setContext(this)
91         .setMessage("Cargando perfil")
92         .setCancelable(false)
93         .build()
94
95     dialog.show()
96
97     super.onRestart()
98
99     llenarCampos()
100    cargarImagen()
101
102    dialog.dismiss()
103 }
104
105 private fun cambiarContrasena() {
106
107     super.onRestart()
108
109     llenarCampos()
110    cargarImagen()
111
112     dialog.dismiss()
113 }
114
115 private fun cambiarContrasena() {
116
117     val prefs: SharedPreferences = getSharedPreferences("com.example.cjproductions.PREFERENCIAS", Context.MODE_PRIVATE)
118
119     FirebaseAuth.getInstance().sendPasswordResetEmail(prefs.getString(key_email, null).toString()).addOnCompleteListener { (task: Task<Void>)
120
121         if (task.isSuccessful) {
122             Toast.makeText(context, "Se ha enviado un correo para restablecer la contraseña", Toast.LENGTH_SHORT).show()
123         } else {
124             showAlert(mensaje: "No se ha podido enviar el correo para restablecer contraseña")
125         }
126     }
127
128
129 private fun showAlert(mensaje: String) {
130     val builder = AlertDialog.Builder(context)
131     builder.setTitle(ERROR)
132     builder.setMessage(mensaje)
133     builder.setPositiveButton(text: "Aceptar", listener: null)
134     val dialog = builder.create()
135     dialog.show()
136 }
137
138
139
140
141
142
143
144

```

EditarUsuarios.kt

```
EditorUsuarios.kt
1 package com.example.cjproductions.perfilusuarios
2
3 import android.Manifest
4 import android.content.Context
5 import android.content.Intent
6 import android.content.SharedPreferences
7 import android.content.pm.PackageManager
8 import android.graphics.Bitmap
9 import android.net.Uri
10 import android.os.Bundle
11 import android.provider.MediaStore
12 import android.view.LayoutInflater
13 import android.widget.Toast
14 import androidx.appcompat.AlertDialog
15 import androidx.appcompat.AppCompatActivity
16 import android.core.app.ActivityCompat
17 import android.core.content.ContextCompat
18 import com.example.cjproductions.R
19 import com.google.firebase.firestore.FirebaseFirestore
20 import com.google.firebase.ktx.firebaseio
21 import com.google.firebase.storage.StorageReference
22 import com.google.firebase.ktx.storage
23 import com.squareup.picasso.Picasso
24 import kotlinx.android.synthetic.main.activity_editar_usuarios.*
25 import kotlinx.android.synthetic.main.dialogo_editar_imagen.*
26 import java.io.ByteArrayOutputStream
27
28
29 class EditorUsuarios : AppCompatActivity() {
30
31     private val db = FirebaseFirestore.getInstance()
32     lateinit var storageReference: StorageReference
33
34     private val CAMERA_ACTION_CODE: Int = 201
35     private val GALERIA_ACTION_CODE: Int = 501
36
37     private lateinit var nombre: String
38     private lateinit var telefono: String
39
40     private lateinit var email:String
41
42     override fun onCreate(savedInstanceState: Bundle?) {
43         super.onCreate(savedInstanceState)
44         setContentView(R.layout.activity_editor_usuarios)
45
46         //...
47
48         title = "Editar Perfil"
49         ponerListeners()
50
51         //...
52         val prefs: SharedPreferences? = getSharedPreferences("com.example.cjproductions.PREFERENCIAS", Context.MODE_PRIVATE)
53         if (prefs != null) {
54             email=prefs.getString( key="email", defValue="null").toString()
55         }
56
57         //...
58         val storage= Firebase.storage
59         storageReference=storage.reference
60
61
62         cargarImagen()
63     }
64
65     private fun cargarImagen() {
66         val perfilReferencia = storageReference.child( pathString: "usuarios/$email/profile.jpg")
67
68         //Coloca la imagen desde el almacenamiento de firebase
69         perfilReferencia.downloadUrl.addOnSuccessListener { url: Uri?
70             Picasso.get().load(it).resize( targetWidth: 150, targetHeight: 150).centerCrop().into(editorImagen)
71         }
72     }
73
74     private fun ponerListeners() {
75         btGuardarDatos.setOnLongClickListener { view: View?
76             val prefs: SharedPreferences? = getSharedPreferences("com.example.cjproductions.PREFERENCIAS", Context.MODE_PRIVATE)
77             prefs.edit().putString(key="nombre", value= nombre).apply()
78             Toast.makeText( context: this, "Pulsa para guardar los datos", Toast.LENGTH_SHORT).show()
79             true
80         }
81
82         btGuardarDatos.setOnClickListener { view: View?
83             val prefs: SharedPreferences? = getSharedPreferences("com.example.cjproductions.PREFERENCIAS", Context.MODE_PRIVATE)
84
85             if (prefs != null) {
86
87                 //Comprobar campo Nombre
88                 //Se hace por si el usuario decide eliminar el nombre
89             }
90         }
91     }
92
93 }
```

```
    //Comprobar campo Nombre
    //Se hace por si el usuario decide eliminar el nombre
    if (etEditarNombre.getText().toString() == "")
        nombre = "No proporcionado"
    else
        nombre = etEditarNombre.getText().toString()

    //Comprobar campo Telefono
    //Se hace por si el usuario decide eliminar el telefono
    if (etEditarTelefono.getText().toString() == "")
        telefono = "No proporcionado"
    else
        telefono = etEditarTelefono.getText().toString()

    db.collection( collectionPath: "UserLog" ).document( prefs.getString( key: "email", defaultValue: null ).toString() )
        .set( hashMapOf( "Nombre" to nombre, "Telefono" to telefono ) )

    Toast.makeText( context: this, text: "Datos actualizados correctamente", Toast.LENGTH_SHORT ).show()
    finish() //Cierra el activity
}

btEditarImagen.setOnLongClickListener { v: View!
    Toast.makeText( context: this, "Pulsa para editar la foto de perfil", Toast.LENGTH_SHORT ).show()
    true //SetOnLongClickListener
}

btEditarImagen.setOnClickListener { v: View!
    val dialogView = LayoutInflater.from( context: this ).inflate(R.layout.dialogo_editar_imagen, root: null )

    val mBuilder = AlertDialog.Builder( context: this ).setView( dialogView ).setTitle("Elegir foto!")

    val mAlertDialog = mBuilder.show()

    mAlertDialog.dialogAtCamera.setOnClickListener { v: View!
        Toast.makeText( context: this, text: "Se ha pulsado camera", Toast.LENGTH_SHORT ).show()
        accionCamara()
    }

    mAlertDialog.dismiss()
}

mAlertDialog.dialogAtDefecto.setOnClickListener { v: View!
```

```
        }

        mAlertDialog.dialogoBtDefecto.setOnClickListener { it:View ->
            editarImagen.setImageURI(Uri.parse(unString))
            val imagen = Uri.parse(unString)
                "android.resource://$packageName/$R.mipmap.user_default")}

            subirImagen(imagen)

            mAlertDialog.dismiss()
        }

        mAlertDialog.dialogoBtGaleria.setOnClickListener { it:View ->
            accionGaleria()

            mAlertDialog.dismiss()
        }
    }
}

private fun accionGaleria() {
    val i = Intent(Intent.ACTION_OPEN_DOCUMENT)
    i.addCategory(Intent.CATEGORY_OPENABLE)
    i.type = "image/*"
    startActivityForResult(i, GALERIA_ACTION_CODE)
}

/*
 * Notando que llamard a todo lo relacionado con la camara
 */
private fun accionCamara() {
    permisoCamara() //Cuando termina esto quiere decir que se ha hecho la foto o denegado permisos
}

private fun permisoCamara() {
    if (ContextCompat.checkSelfPermission(context this, Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED)
        ActivityCompat.requestPermissions(activity this, arrayOf(Manifest.permission.CAMERA), requestCode 101)
    else {
        abrirCamara()
    }
}

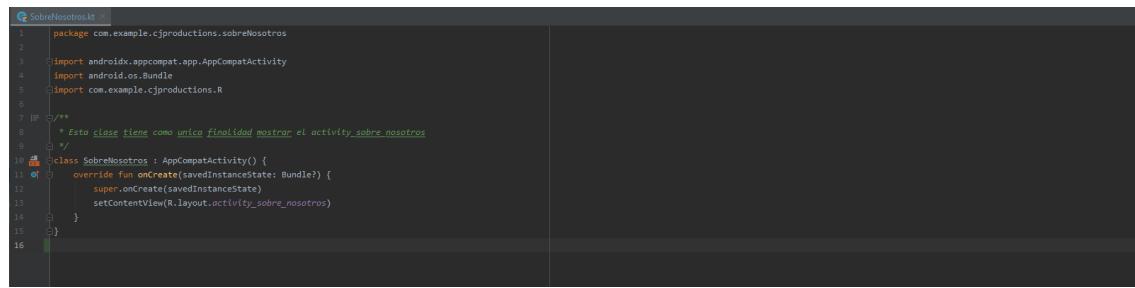
override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<String>, grantResults: IntArray) {
    if (requestCode == 101) {
        if (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            abrirCamara()
        }
    }
}
```

```

70     if (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
71         abrirCamara()
72     } else {
73         Toast.makeText(context, text: "Se necesitan los permisos de cámara", Toast.LENGTH_SHORT).show()
74     }
75 }
76
77 private fun abrirCamara() {
78     val camera = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
79     if (camera.resolveActivity(packageManager) != null)
80         startActivityForResult(camera, CAMERA_ACTION_CODE)
81     else
82         Toast.makeText(context, text: "El dispositivo no soporta esta opción", Toast.LENGTH_SHORT).show()
83 }
84
85 override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
86     super.onActivityResult(requestCode, resultCode, data)
87
88     /**
89      * DE PARTE DE LA CÁMARA
90      */
91     if (requestCode == CAMERA_ACTION_CODE && resultCode == RESULT_OK) {
92         var datos: Bundle? = data?.extras
93         var fotoBitmap: Bitmap = datos?.get("data") as Bitmap
94
95         var uri: Uri = getImageURI(context, fotoBitmap)
96
97         editarImagen.setImageURI(uri)
98
99         if (data!=null){
100             if ($email!="-1"){
101                 if (uri != null) {
102                     subirImagen(uri)
103                 }
104             }
105         }
106     }
107
108     /**
109      * DE PARTE DE LA GALERIA
110      */
111     if(data!=null){
112         val uri: Uri? = data.data // mnt/sdcard/images/image.jpg por ejemplo
113
114         if (data!=null){
115             val uri: Uri? = data.data // mnt/sdcard/images/image.jpg por ejemplo
116             if($email!="-1"){
117                 subirImagen(uri)
118             }
119         }
120     }
121
122     /**
123      * Método que subió la imagen que ha elegido el usuario a la base de datos
124      */
125     private fun subirImagen(uri: Uri?) {
126         val referencia = storageReference.child(pathString: "usuarios/$email/profile.jpg")
127
128         if(referencia==null){
129
130             if (uri != null) {
131                 referencia.putFile(uri).addOnCompleteListener { it: Task<UploadTask.TaskSnapshot>{
132                     if (it.isSuccessful) {
133                         cargarImagen()
134                     } else {
135                         Toast.makeText(context, text: "Se ha producido un error", Toast.LENGTH_LONG).show()
136                     }
137                 }
138             }
139         }
140
141     /**
142      * Método que convierte un objeto de tipo Bitmap a un objeto Uri
143      * Retornando dicho objeto
144      */
145     private fun getImageURI(context: Context, inImage: Bitmap): Uri {
146         val bytes = ByteArrayOutputStream()
147         inImage.compress(Bitmap.CompressFormat.JPEG, quality: 100, bytes)
148         val path = MediaStore.Images.Media.insertImage(context.contentResolver, inImage, title: "Title", description: null)
149         return Uri.parse(path)
150     }
151 }

```

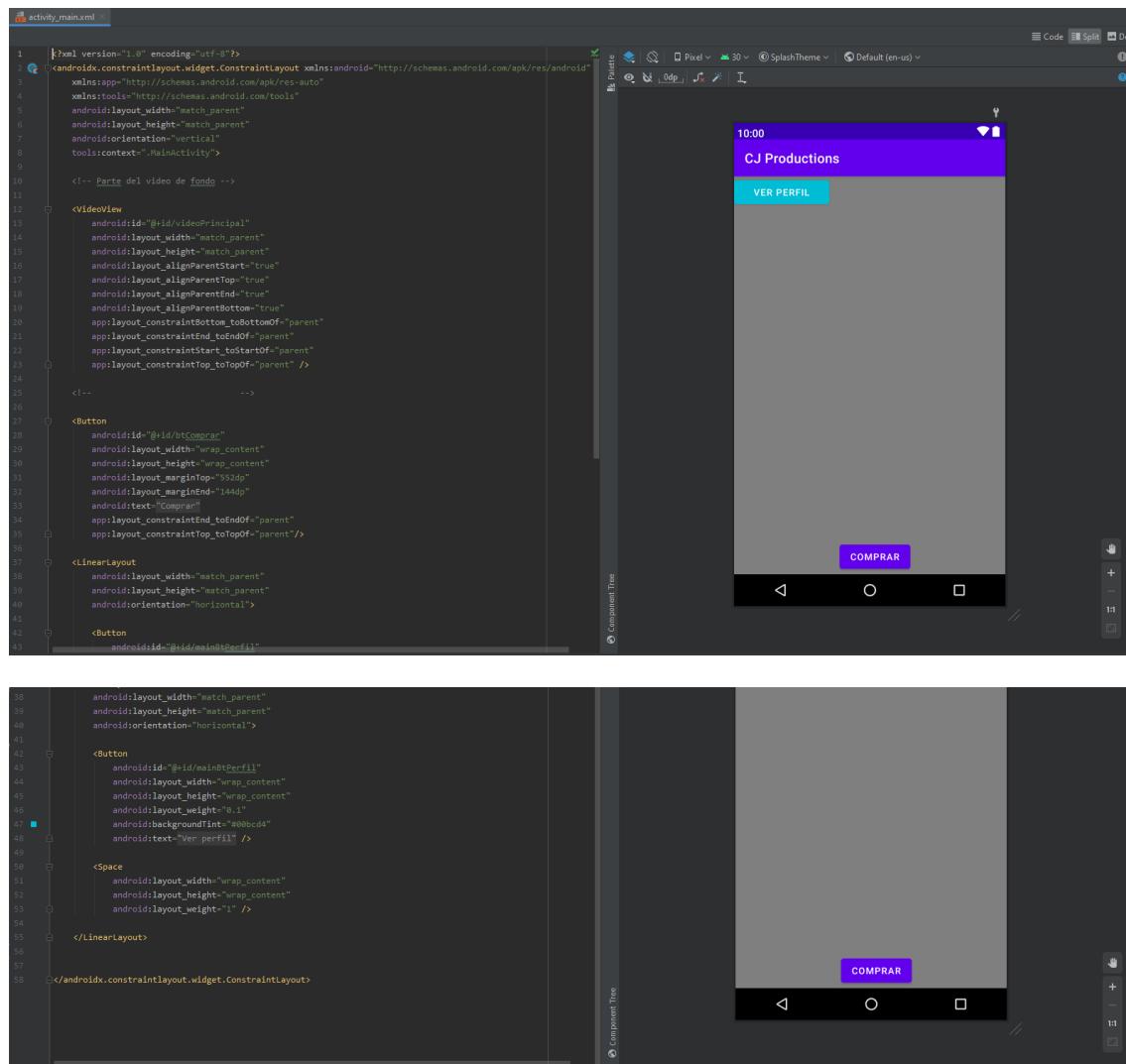
SobreNosotros.kt



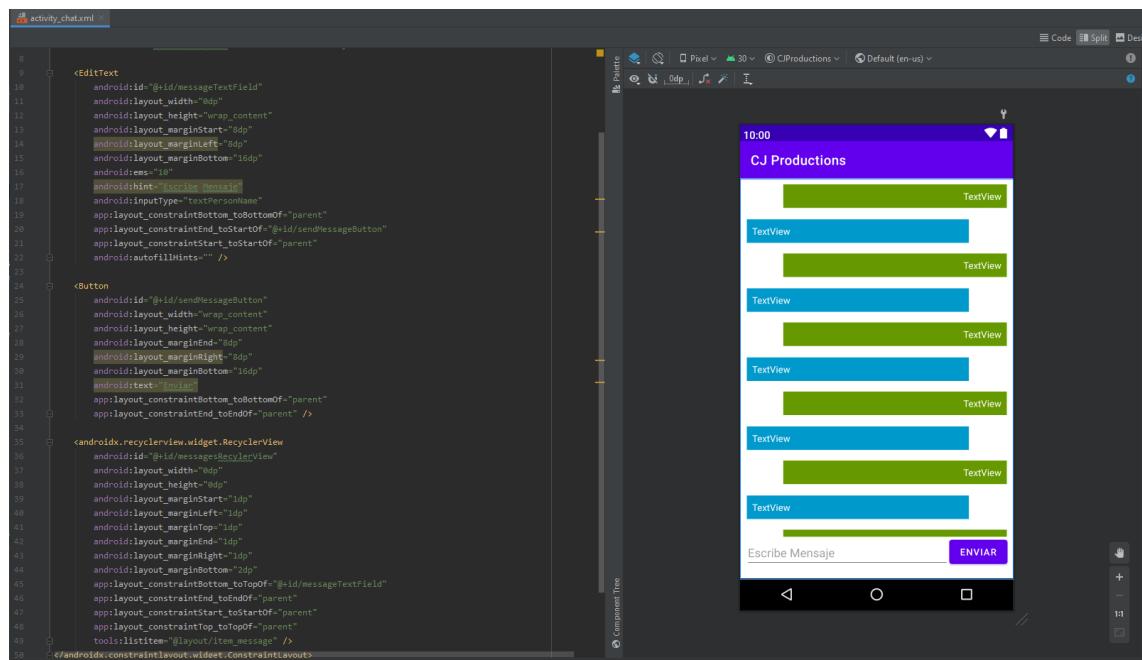
```
1 package com.example.cjproductions.sobreNosotros
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5 import com.example.cjproductions.R
6
7 /**
8 * Esta clase tiene como única finalidad mostrar el activity_sobre_nosotros
9 */
10 class SobreNosotros : AppCompatActivity() {
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         setContentView(R.layout.activity_sobre_nosotros)
14     }
15 }
```

Layouts

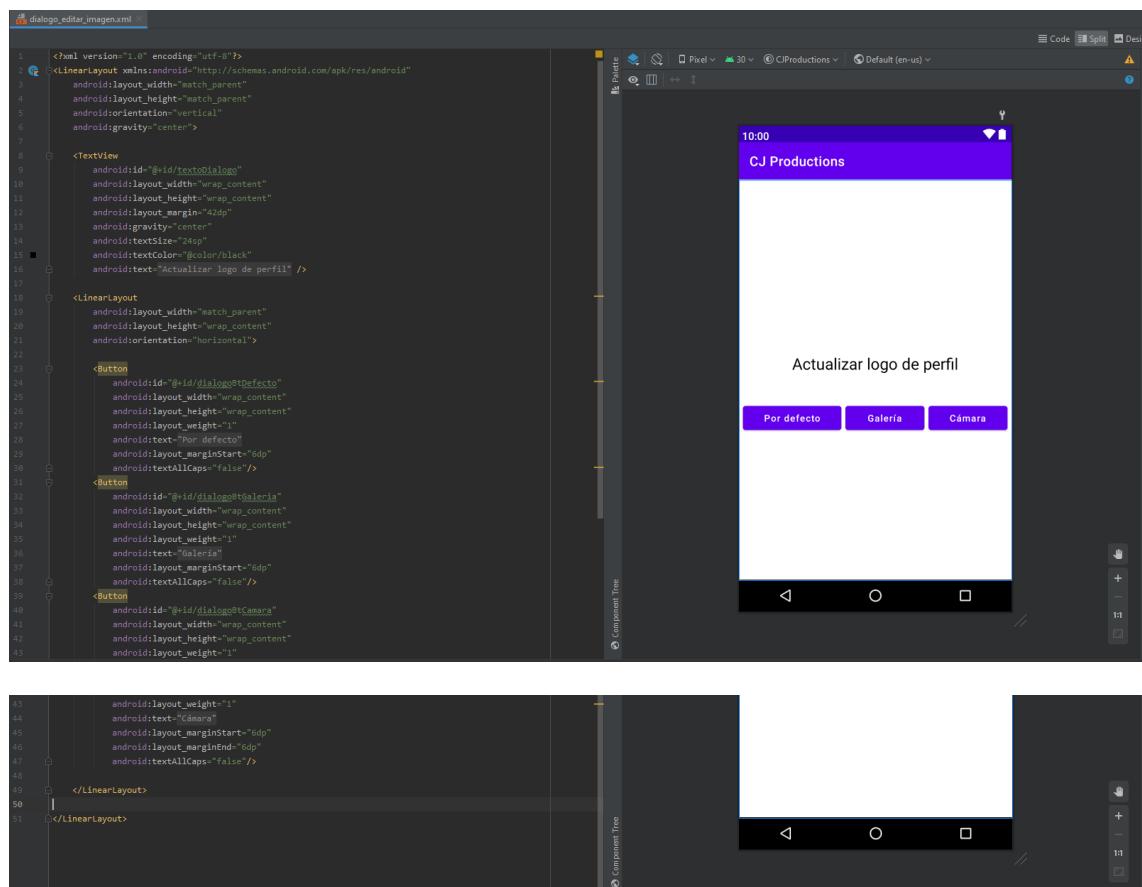
activity_main.xml



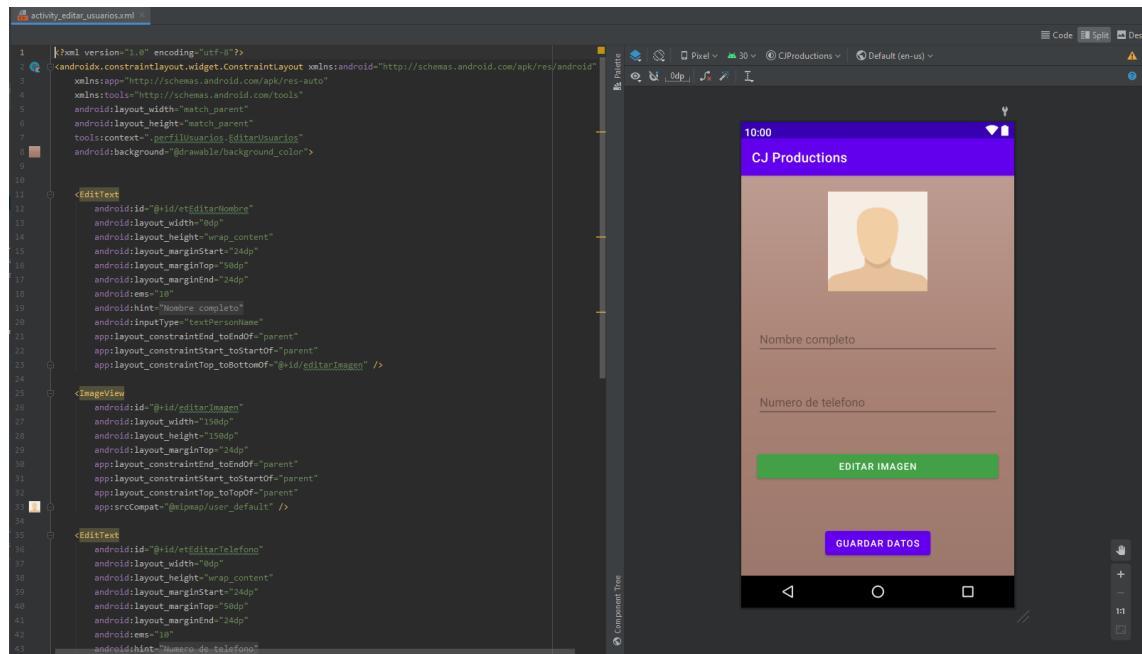
activity_chat.xml



dialogo_editar_imagen.xml



activity_editar_usuarios.xml



The screenshot shows the Android Studio interface with the XML code for the activity. The XML code defines a ConstraintLayout with various views like EditText, ImageView, and Buttons. The preview window shows a purple header with the text 'CJ Productions'. Below the header is a placeholder image of a person. There are two text input fields labeled 'Nombre completo' and 'Número de teléfono'. Below the inputs are two buttons: a green 'EDITAR IMAGEN' button and a blue 'GUARDAR DATOS' button. The bottom of the screen has standard Android navigation icons.

```
<?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/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".perfijousuarios.EditarUsuarios"
    android:background="@drawable/background_color">

    <EditText
        android:id="@+id/etEditorNombre"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="24dp"
        android:ems="10"
        android:hint="Nombre completo"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editarImagen" />

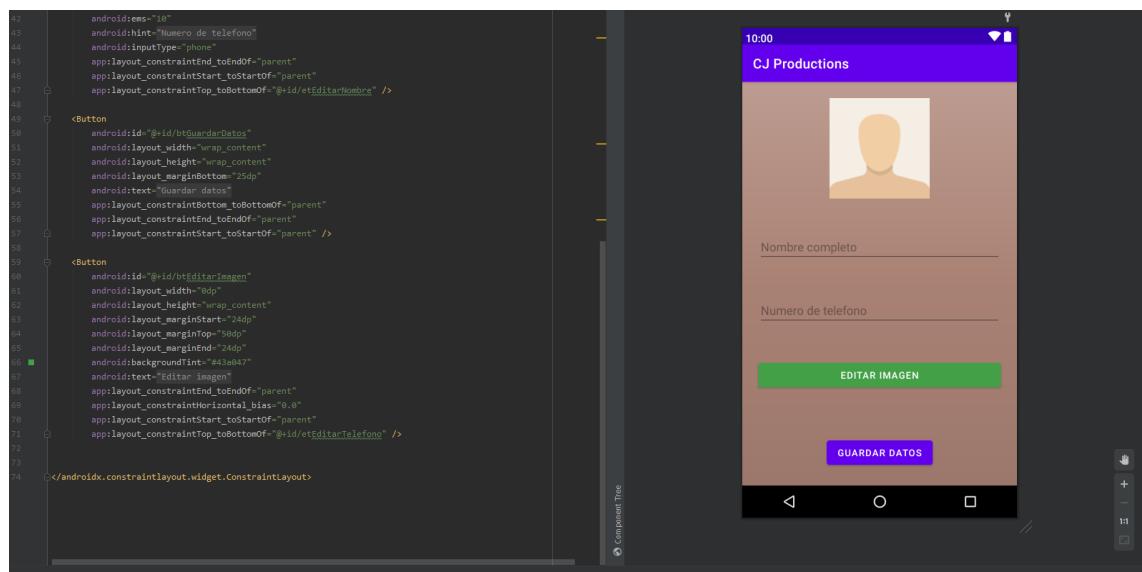
    <ImageView
        android:id="@+id/editarImagen"
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:layout_marginTop="50dp"
        android:layout_constraintEnd_toEndOf="parent"
        android:layout_constraintStart_toStartOf="parent"
        android:layout_constraintTop_toTopOf="parent"
        android:srcCompat="@{elpmap/user_default}" />

    <EditText
        android:id="@+id/etEditorTelefono"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="24dp"
        android:ems="10"
        android:hint="Número de teléfono"
        android:inputType="phone"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etEditorNombre" />

    <Button
        android:id="@+id/btGuardarDatos"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="25dp"
        android:text="Guardar datos"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/btEditarImagen"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="24dp"
        android:background="@+color/colorPrimaryDark"
        android:text="Editar imagen"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etEditorTelefono" />

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



This screenshot shows the same XML code as the first one, but the preview window displays a different state. The placeholder image of the person is now larger and centered. The text input fields are also visible. The buttons are present but appear to be in a different state or color. The bottom navigation icons are visible at the bottom of the screen.

```
<?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/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".perfijousuarios.EditarUsuarios"
    android:background="@drawable/background_color">

    <EditText
        android:id="@+id/etEditorNombre"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="24dp"
        android:ems="10"
        android:hint="Nombre completo"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editarImagen" />

    <ImageView
        android:id="@+id/editarImagen"
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:layout_marginTop="50dp"
        android:layout_constraintEnd_toEndOf="parent"
        android:layout_constraintStart_toStartOf="parent"
        android:layout_constraintTop_toTopOf="parent"
        android:srcCompat="@{elpmap/user_default}" />

    <EditText
        android:id="@+id/etEditorTelefono"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="24dp"
        android:ems="10"
        android:hint="Número de teléfono"
        android:inputType="phone"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etEditorNombre" />

    <Button
        android:id="@+id/btGuardarDatos"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="25dp"
        android:text="Guardar datos"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/btEditarImagen"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="24dp"
        android:background="@+color/colorPrimaryDark"
        android:text="Editar imagen"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etEditorTelefono" />

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

activity_inicio_sesion.xml

The screenshot shows the Android Studio interface with the XML code for the login screen and its corresponding preview.

XML Code:

```

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

    <androidx.cardview.widget.CardView
        android:layout_width="2dp"
        android:layout_alignParentBottom="true"
        android:useCompatPadding="true"
        android:layout_margin="20dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="10dp"
            android:orientation="vertical">

            <androidx.cardview.widget.CardView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="10dp"
                android:padding="10dp"
                android:background="#5E81B1B1"
                android:elevation="2dp">

                <EditText
                    android:id="@+id/etEmail"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_margin="10dp"
                    android:background="#FFFFFF"
                    android:hint="Email"
                    android:inputType="textEmailAddress"
                    android:textColor="#FFFFFF" />
            </androidx.cardview.widget.CardView>

                <EditText
                    android:id="@+id/etContrasena"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_margin="10dp"
                    android:background="#FFFFFF"
                    android:hint="Contraseña"
                    android:inputType="textPassword"
                    android:textColor="#FFFFFF" />
            </androidx.cardview.widget.CardView>

                <Button
                    android:id="@+id/btEstablecerContrasena"
                    android:layout_width="match_parent"
                    android:layout_height="45dp"
                    android:background="#FFFFFF"
                    android:text="He olvidado mi contraseña" />
            </Button>

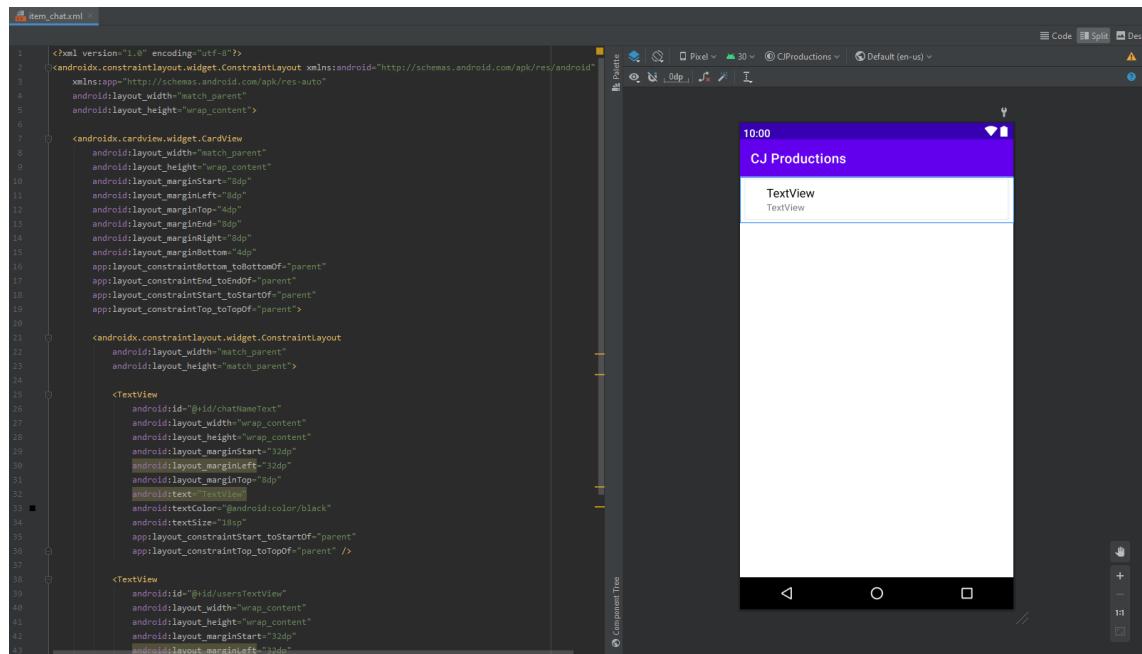
                <Button
                    android:id="@+id/btIniciarLogin"
                    android:layout_width="match_parent"
                    android:layout_height="45dp"
                    android:background="#B4C4F9"
                    android:text="Iniciar sesión" />
            </Button>

                <Space
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content" />
            </LinearLayout>
        </androidx.cardview.widget.CardView>
    </RelativeLayout>

```

Preview:

item_chat.xml



```
<?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">

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="4dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="4dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

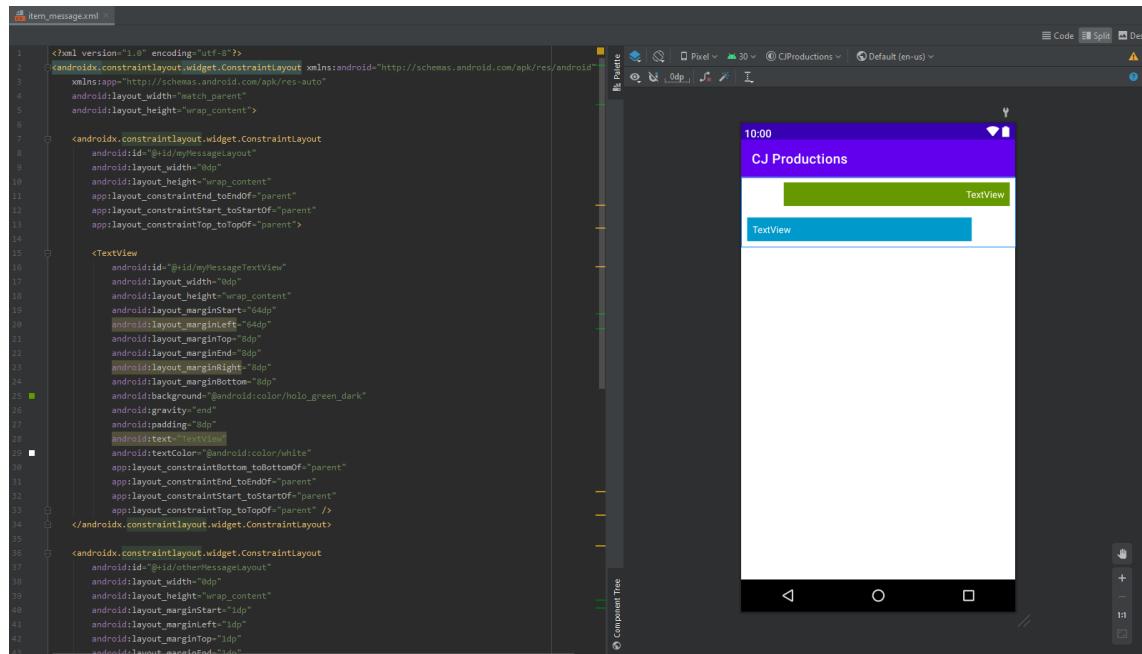
        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/chatNameText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="32dp"
                android:layout_marginLeft="32dp"
                android:layout_marginTop="8dp"
                android:layout_marginEnd="32dp"
                android:layout_marginRight="32dp"
                android:text="TextView"
                android:textColor="@android:color/black"
                android:textSize="16sp"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <TextView
                android:id="@+id/usersTextView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="32dp"
                android:layout_marginLeft="32dp" />
        
    

```

item_message.xml



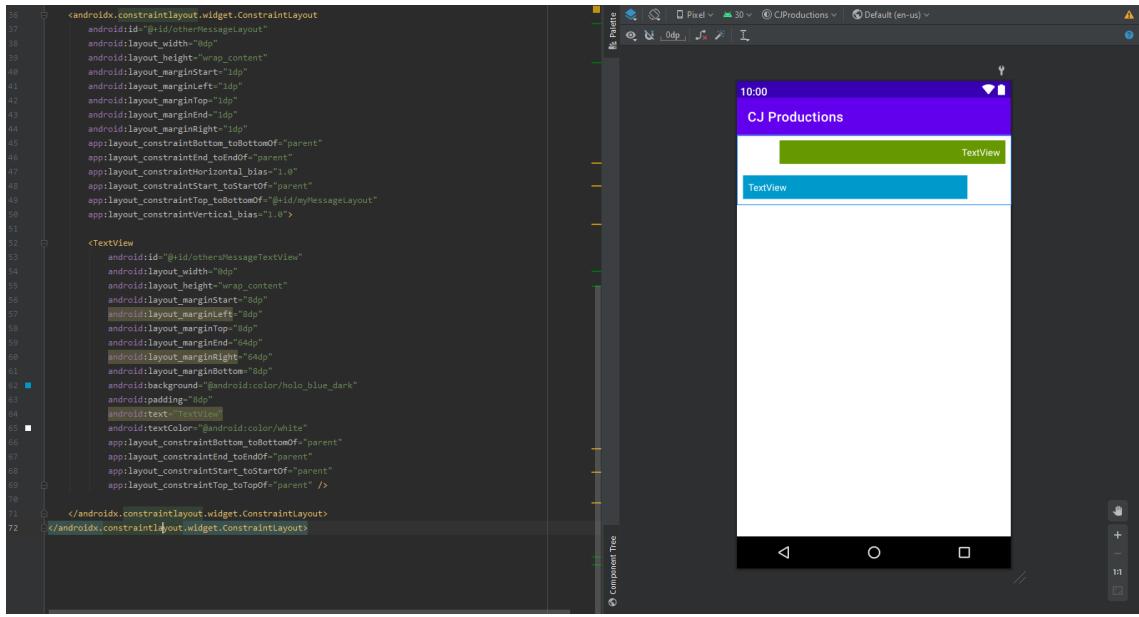
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/myMessageLayout"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

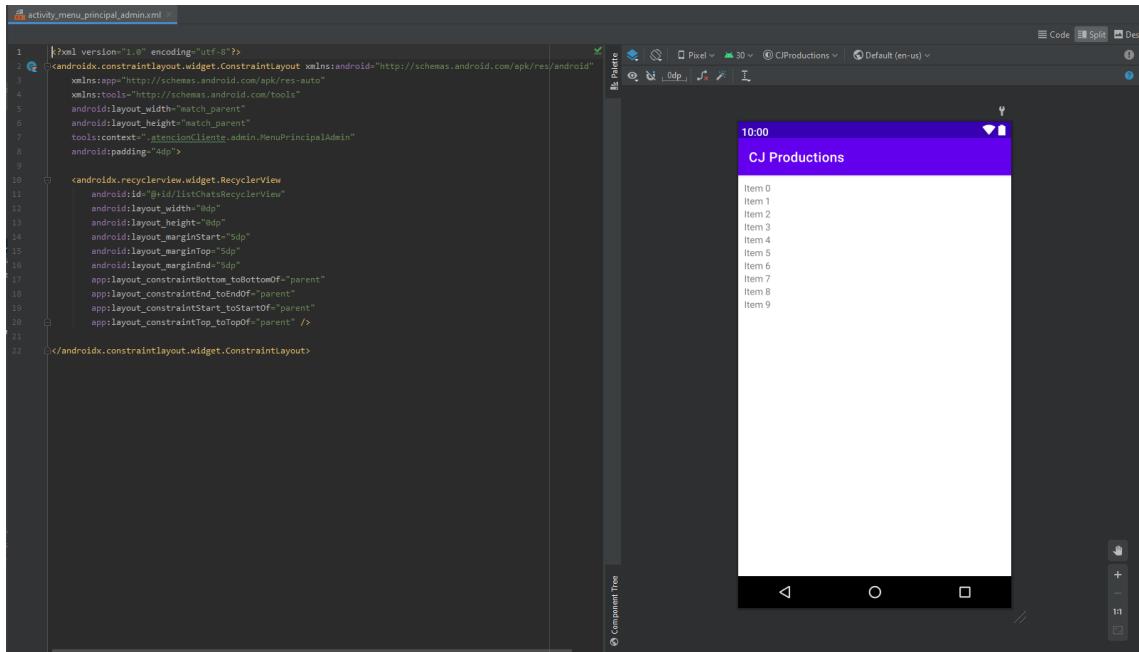
        <TextView
            android:id="@+id/myMessageText"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="64dp"
            android:layout_marginLeft="64dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="0dp"
            android:layout_marginRight="0dp"
            android:layout_marginBottom="8dp"
            android:background="@android:color/holo_green_dark"
            android:gravity="end"
            android:padding="8dp"
            android:text="TextView"
            android:textColor="@android:color/white"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
    

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/otherMessageLayout"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginRight="16dp" />

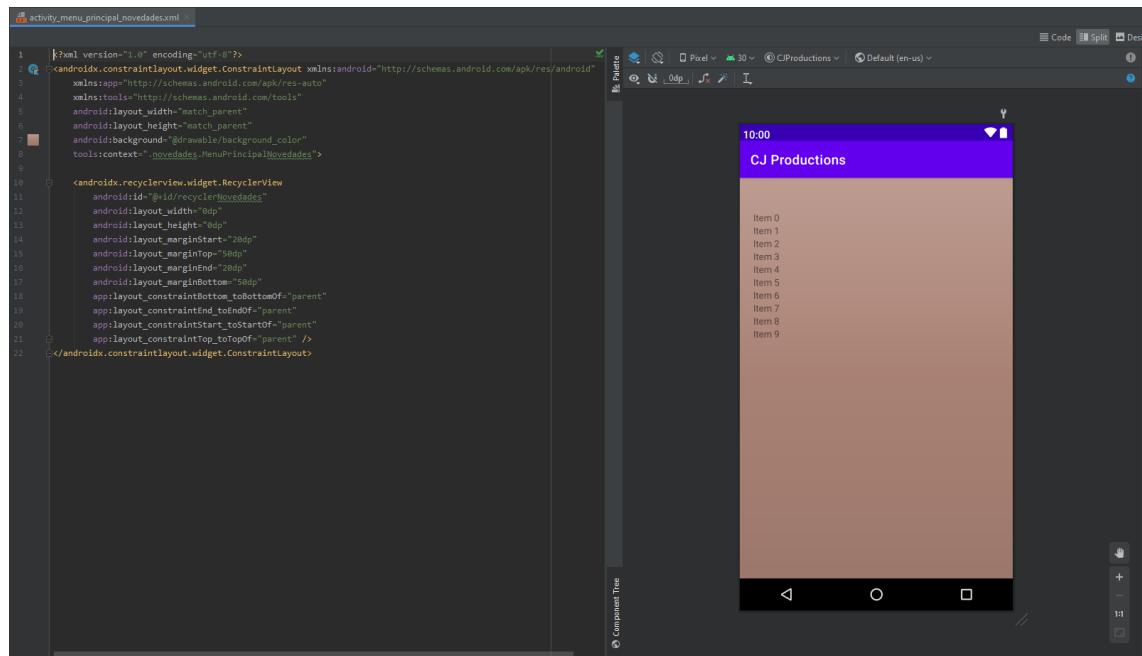
```



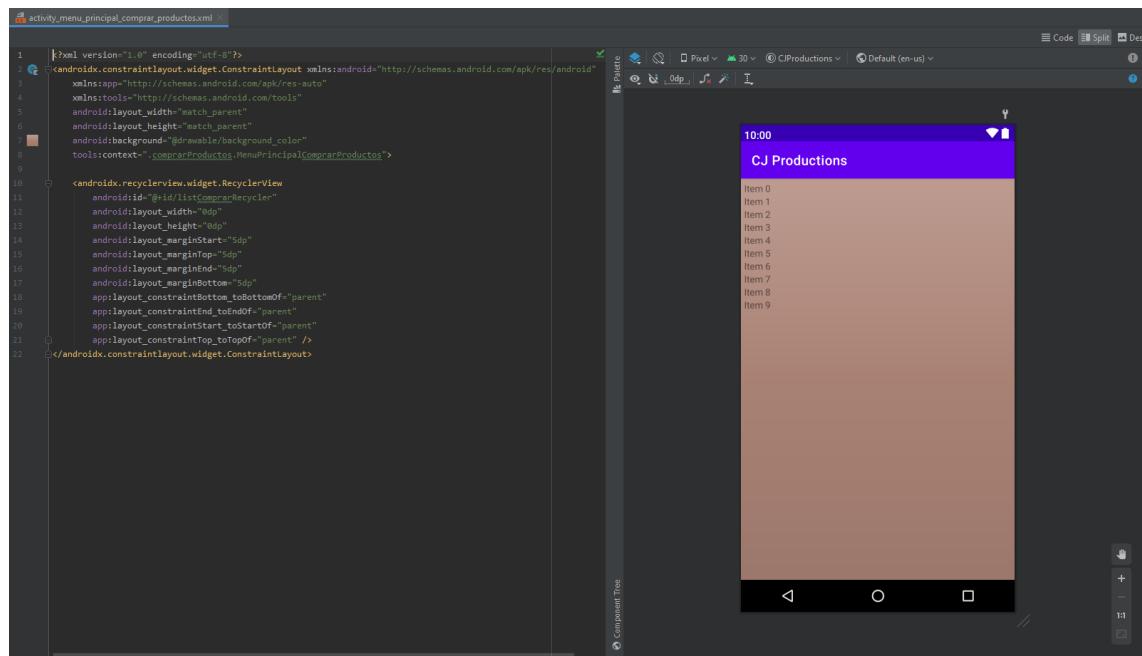
`activity_menu_principal_admin.xml`



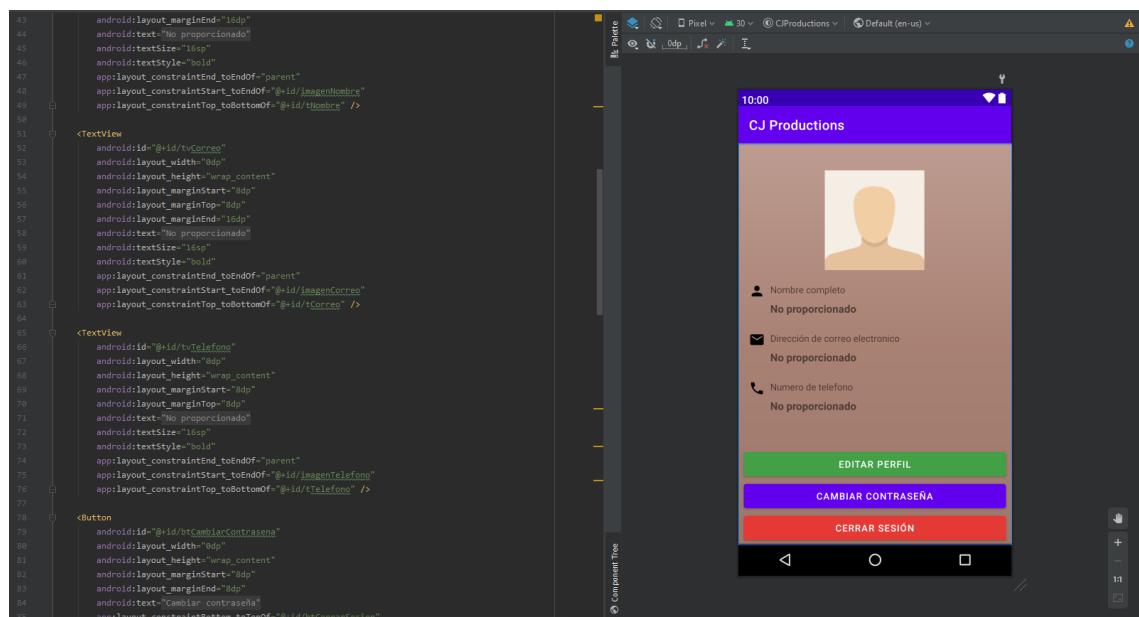
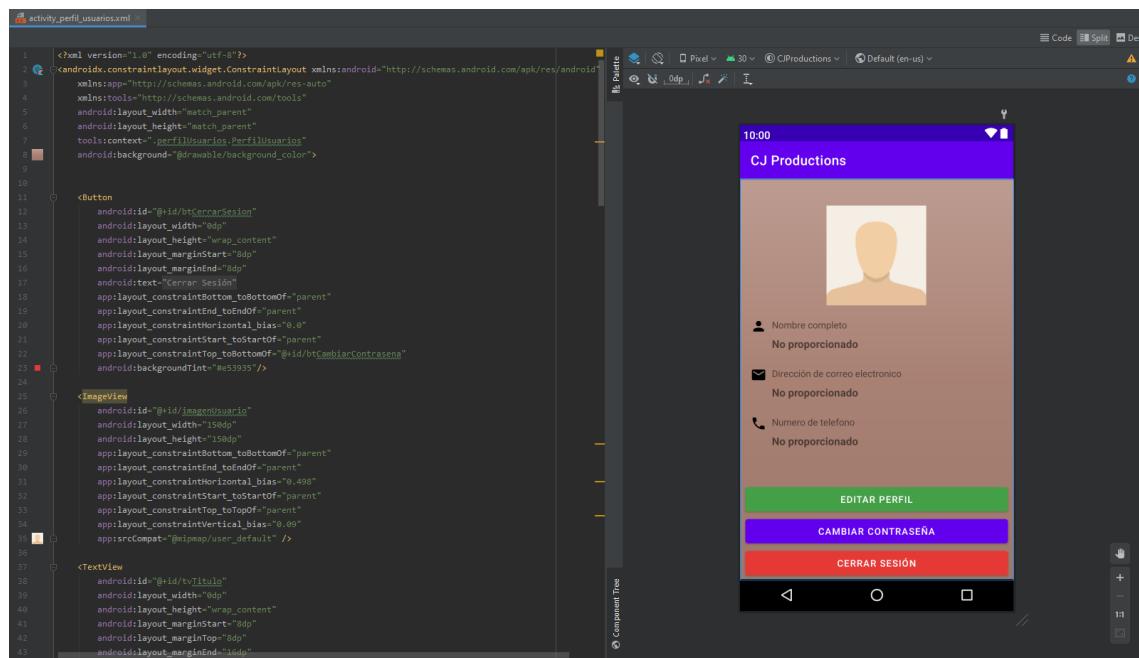
activity_menu_principal_novedades.xml



activity_menu_principal_comprar_productos.xml



activity_perfil_usuarios.xml



```

3   android:layout_constraintBottom_toTopOf="@+id/btCerrarSession"
4   android:layout_constraintEnd_toEndOf="parent"
5   android:layout_constraintStart_toStartOf="parent"
6   android:layout_constraintTop_toBottomOf="@+id/btEditarImagen"
7   android:layout_constraintVertical_bias="1.0" />
8
9   <TextView
10    android:id="@+id/tNombre"
11    android:layout_width="0dp"
12    android:layout_height="wrap_content"
13    android:layout_marginStart="8dp"
14    android:layout_marginEnd="16dp"
15    android:text="Nombre completo"
16    android:layout_constraintEnd_toEndOf="parent"
17    android:layout_constraintStart_toEndOf="@+id/imagenNombre"
18    android:layout_constraintTop_toTopOf="@+id/imagenNombre" />
19
20   <TextView
21    android:id="@+id/tCorreo"
22    android:layout_width="0dp"
23    android:layout_height="wrap_content"
24    android:layout_marginStart="8dp"
25    android:layout_marginEnd="16dp"
26    android:text="Dirección de correo electrónico"
27    android:layout_constraintEnd_toEndOf="parent"
28    android:layout_constraintStart_toEndOf="@+id/imagenCorreo"
29    android:layout_constraintTop_toTopOf="@+id/imagenCorreo" />
30
31   <ImageView
32    android:id="@+id/imagenNombre"
33    android:layout_width="wrap_content"
34    android:layout_height="wrap_content"
35    android:layout_marginStart="16dp"
36    android:layout_constraintBottom_toBottomOf="parent"
37    android:layout_constraintStart_toStartOf="parent"
38    android:layout_constraintTop_toBottomOf="@+id/imagenUsuario"
39    android:layout_constraintVertical_bias="0.0590000012" />
40    app:srcCompat="@drawable/ic_person_black_24dp" />
41
42   <ImageView
43    android:id="@+id/imagenCorreo"
44    android:layout_width="wrap_content"
45    android:layout_height="wrap_content"
46    android:layout_marginStart="16dp"
47    android:layout_marginTop="24dp"
48    android:layout_constraintStart_toStartOf="parent"
49    android:layout_constraintTop_toBottomOf="@+id/tvCorreo"
50    android:layout_constraintBottom_toBottomOf="@+id/tvCorreo" />
51
52   <TextView
53    android:id="@+id/tTelefono"
54    android:layout_width="0dp"
55    android:layout_height="wrap_content"
56    android:layout_marginStart="8dp"
57    android:layout_marginEnd="16dp"
58    android:text="Número de teléfono"
59    android:layout_constraintEnd_toEndOf="parent"
60    android:layout_constraintStart_toEndOf="@+id/imagenTelefono"
61    android:layout_constraintTop_toTopOf="@+id/imagenTelefono" />
62
63   <Button
64    android:id="@+id/btEditarImagen"
65    android:layout_width="0dp"
66    android:layout_height="wrap_content"
67    android:layout_marginStart="8dp"
68    android:layout_marginEnd="16dp"
69    android:text="Editar perfil"
70    android:layout_constraintBottom_toBottomOf="@+id/btCerrarSession"
71    android:layout_constraintEnd_toEndOf="parent"
72    android:layout_constraintStart_toStartOf="parent"
73    android:layout_constraintTop_toBottomOf="@+id/tvTelefono"
74    android:layout_constraintVertical_bias="0.59000003" />
75    android:backgroundTint="#E438A7" />
76
77   </androidx.constraintlayout.widget.ConstraintLayout>

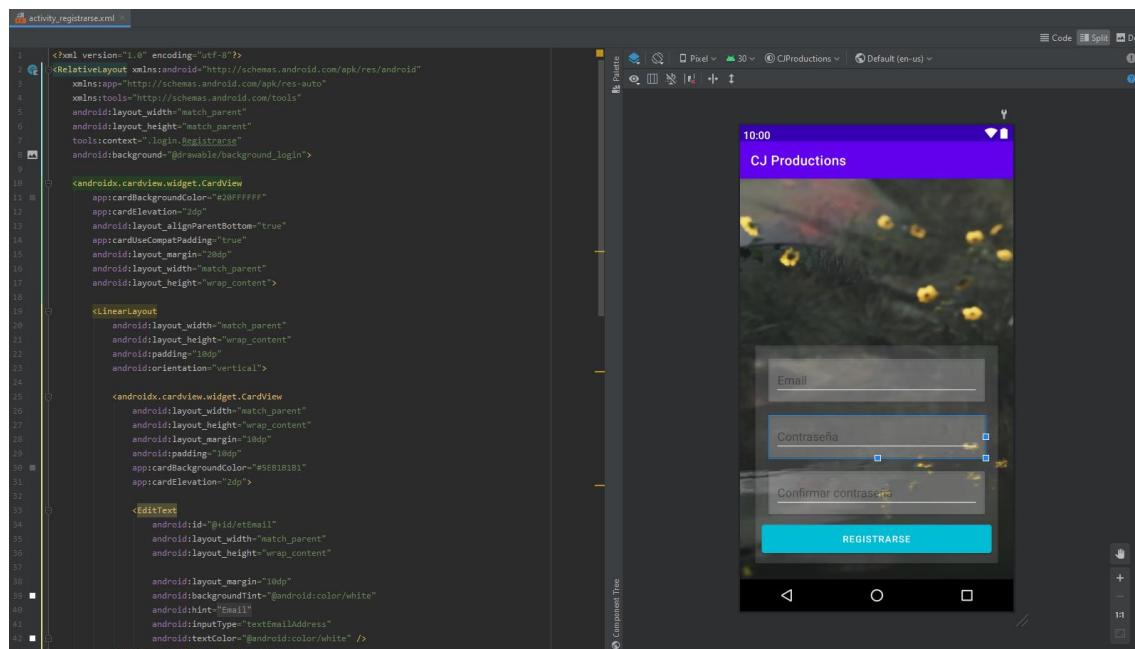
```

```

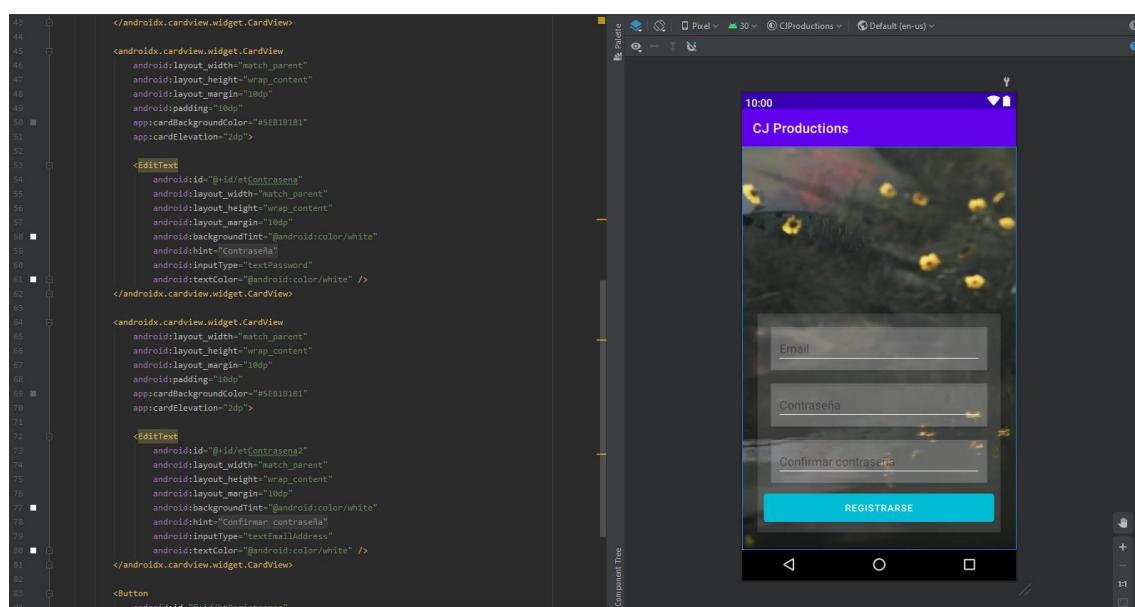
137   android:layout_height="wrap_content"
138   android:layout_marginStart="16dp"
139   android:layout_marginTop="24dp"
140   android:layout_constraintStart_toStartOf="parent"
141   android:layout_constraintTop_toBottomOf="@+id/tvTitulo"
142   app:srcCompat="@drawable/ic_email_black_24dp" />
143
144   <ImageView
145    android:id="@+id/imagenTelefono"
146    android:layout_width="wrap_content"
147    android:layout_height="wrap_content"
148    android:layout_marginStart="16dp"
149    android:layout_marginTop="24dp"
150    android:layout_constraintStart_toStartOf="parent"
151    android:layout_constraintTop_toBottomOf="@+id/tvCorreo"
152    android:layout_constraintBottom_toBottomOf="@+id/tvCorreo" />
153
154   <TextView
155    android:id="@+id/tTelefono"
156    android:layout_width="0dp"
157    android:layout_height="wrap_content"
158    android:layout_marginStart="8dp"
159    android:layout_marginEnd="16dp"
160    android:text="Número de teléfono"
161    android:layout_constraintEnd_toEndOf="parent"
162    android:layout_constraintStart_toEndOf="@+id/imagenTelefono"
163    android:layout_constraintTop_toTopOf="@+id/imagenTelefono" />
164
165   <Button
166    android:id="@+id/btCerrarSession"
167    android:layout_width="0dp"
168    android:layout_height="wrap_content"
169    android:layout_marginStart="8dp"
170    android:layout_marginEnd="16dp"
171    android:text="Cerrar sesión"
172    android:layout_constraintBottom_toBottomOf="@+id/btEditarImagen"
173    android:layout_constraintEnd_toEndOf="parent"
174    android:layout_constraintStart_toStartOf="parent"
175    android:layout_constraintTop_toBottomOf="@+id/tvTelefono"
176    android:layout_constraintVertical_bias="0.59000003" />
177    android:backgroundTint="#E438A7" />
178
179   </androidx.constraintlayout.widget.ConstraintLayout>

```

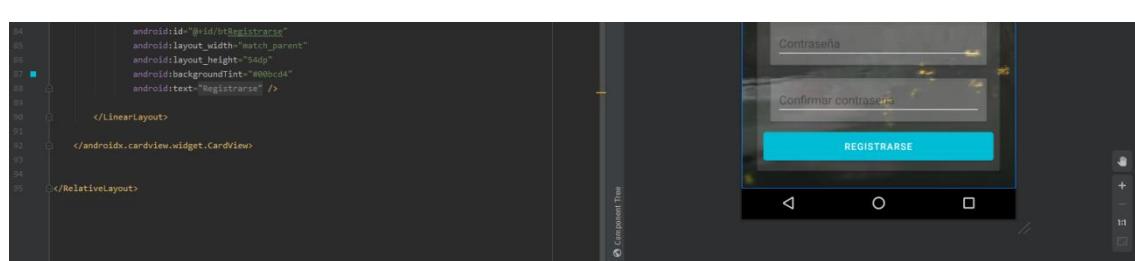
activity_Registrarse.xml



The screenshot shows the Android Studio interface with the XML code for the registration activity. The code defines a relative layout containing three card views for input fields and a button. The first card view contains an email input field. The second contains a password input field. The third contains a confirm password input field. A blue button at the bottom right is labeled 'REGISTRARSE'. The preview window shows the layout on a device with a purple header and a background image of yellow flowers.



This screenshot shows the same layout as above, but the password input field ('etContrasena') is highlighted with a blue selection bar, indicating it is selected or focused.



This screenshot shows the layout with the confirm password input field ('etConfirmarContrasena') highlighted with a blue selection bar.

```
<?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"
    tools:context=".login.Registrarse"
    android:background="@drawable/background_login">

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:layout_margin="20dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="10dp"
            android:orientation="vertical">

            <androidx.cardview.widget.CardView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="10dp"
                android:padding="10dp"
                app:cardBackgroundColor="#5E81B1B1"
                app:cardElevation="2dp">

                <EditText
                    android:id="@+id/etEmail"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"

                    android:layout_margin="10dp"
                    android:backgroundTint="@android:color/white"
                    android:hint="Email"
                    android:inputType="textEmailAddress"
                    android:textColor="@android:color/white" />
            </androidx.cardview.widget.CardView>

            <androidx.cardview.widget.CardView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="10dp"
                android:padding="10dp"
                app:cardBackgroundColor="#5E81B1B1"
                app:cardElevation="2dp">

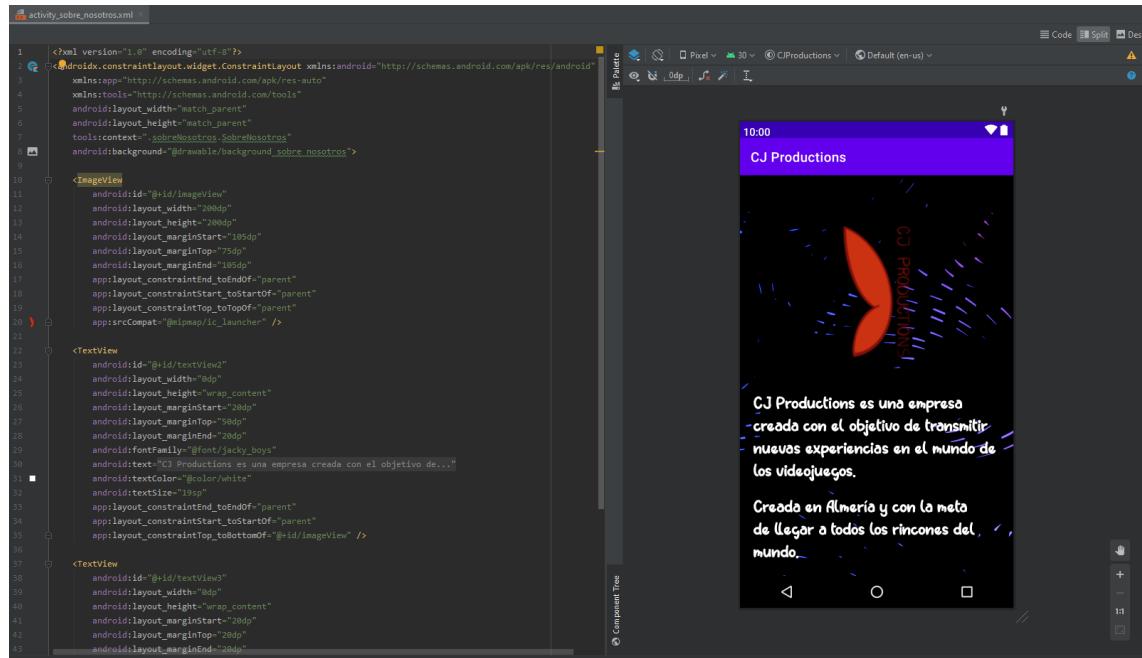
                <EditText
                    android:id="@+id/etContrasena"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_margin="10dp"
                    android:backgroundTint="@android:color/white"
                    android:hint="Contraseña"
                    android:inputType="textPassword"
                    android:textColor="@android:color/white" />
            </androidx.cardview.widget.CardView>

            <androidx.cardview.widget.CardView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="10dp"
                android:padding="10dp"
                app:cardBackgroundColor="#5E81B1B1"
                app:cardElevation="2dp">

                <EditText
                    android:id="@+id/etConfirmarContrasena"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_margin="10dp"
                    android:backgroundTint="@android:color/white"
                    android:hint="Confirmar contraseña"
                    android:inputType="textEmailAddress"
                    android:textColor="@android:color/white" />
            </androidx.cardview.widget.CardView>

            <Button
                android:id="@+id/btRegistrarse"
                android:layout_width="match_parent"
                android:layout_height="50dp"
                android:backgroundTint="#0000cd"
                android:text="Registrarse" />
        </LinearLayout>
    </androidx.cardview.widget.CardView>
</RelativeLayout>
```

activity_sobre_nosotros.xml



```
<?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="match_parent"
    tools:context=".sobreNosotros_SobreNosotros"
    android:background="@drawable/background_sobre_nosotros">

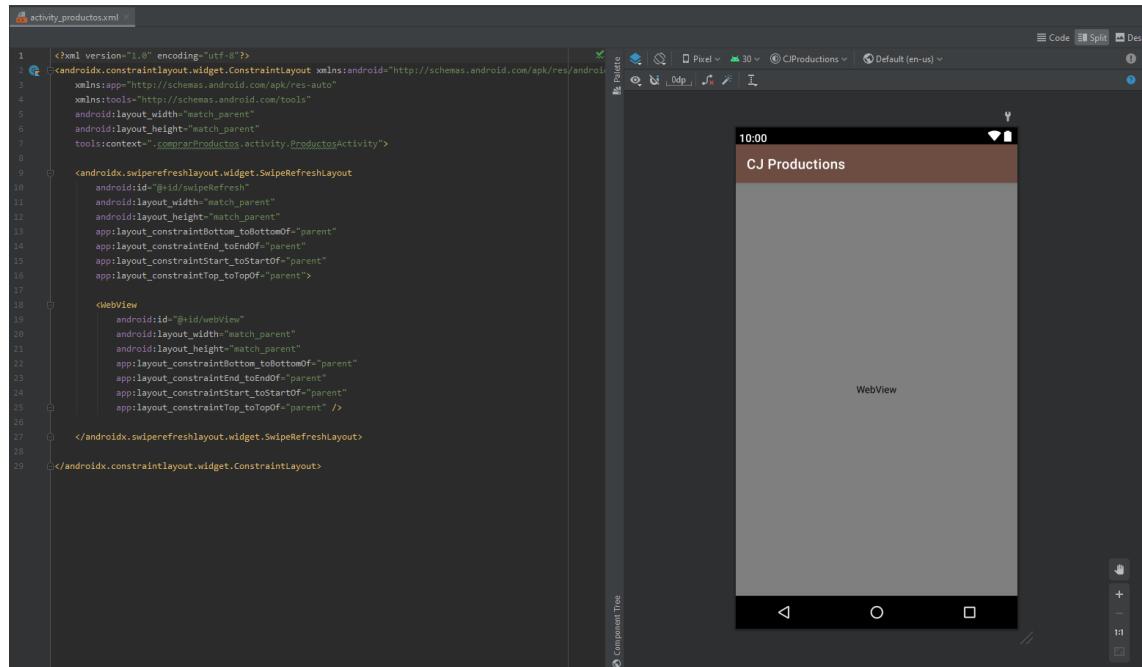
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_marginStart="105dp"
        android:layout_marginTop="75dp"
        android:layout_marginEnd="105dp"
        android:layout_constraintEnd_toEndOf="parent"
        android:layout_constraintStart_toStartOf="parent"
        android:layout_constraintTop_toTopOf="parent"
        android:srcCompat="@mipmap/ic_launcher" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="20dp"
        android:fontFamily="#font/Jucky_boys"
        android:text="CJ Productions es una empresa creada con el objetivo de..."
        android:textColor="@color/white"
        android:textSize="18sp"
        android:layout_constraintEnd_toEndOf="parent"
        android:layout_constraintStart_toStartOf="parent"
        android:layout_constraintTop_toBottomOf="@+id/imageView" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="20dp"
        android:layout_marginBottom="20dp"
        android:layout_marginBottom="20dp"
        android:layout_marginEnd="20dp" />

```

Activity_products.xml

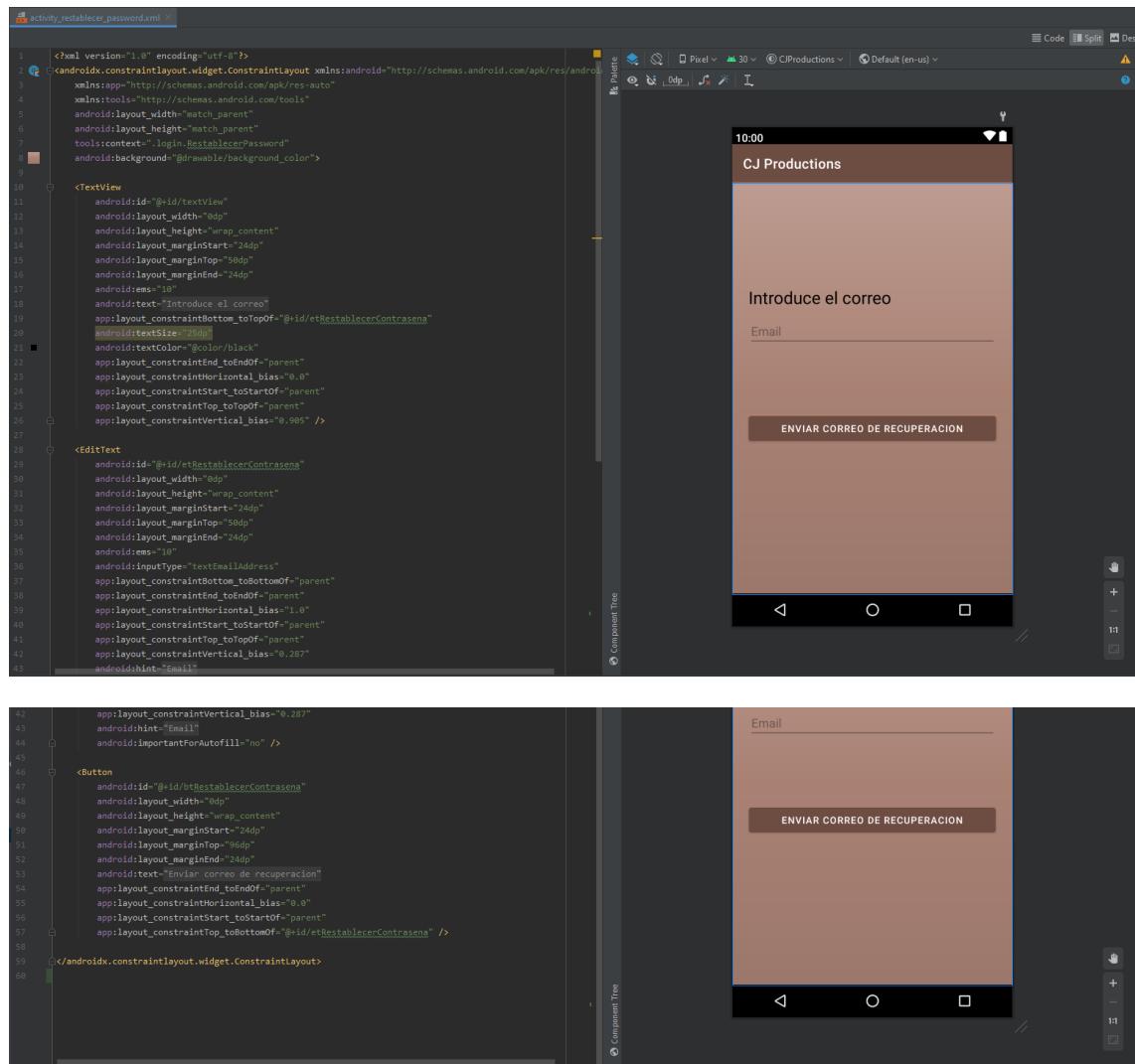


```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".compraproduc.activity_ProductosActivity">

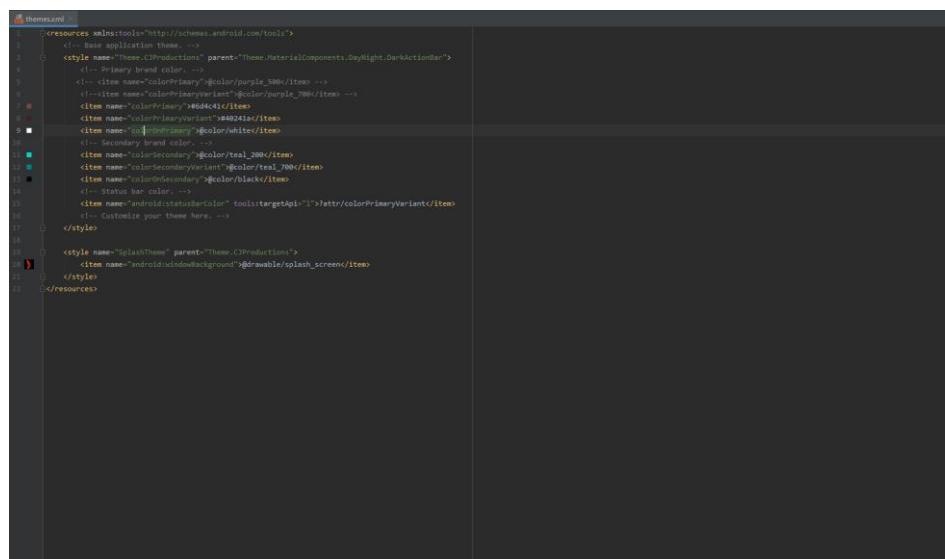
    <androidx.swiperefreshlayout.widget.SwipeRefreshLayout
        android:id="@+id/swipeRefreshLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_constraintBottom_toBottomOf="parent"
        android:layout_constraintEnd_toEndOf="parent"
        android:layout_constraintStart_toStartOf="parent"
        android:layout_constraintTop_toTopOf="parent">

        <WebView
            android:id="@+id/webView"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_constraintBottom_toBottomOf="parent"
            android:layout_constraintEnd_toEndOf="parent"
            android:layout_constraintStart_toStartOf="parent"
            android:layout_constraintTop_toTopOf="parent" />
    </androidx.swiperefreshlayout.widget.SwipeRefreshLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

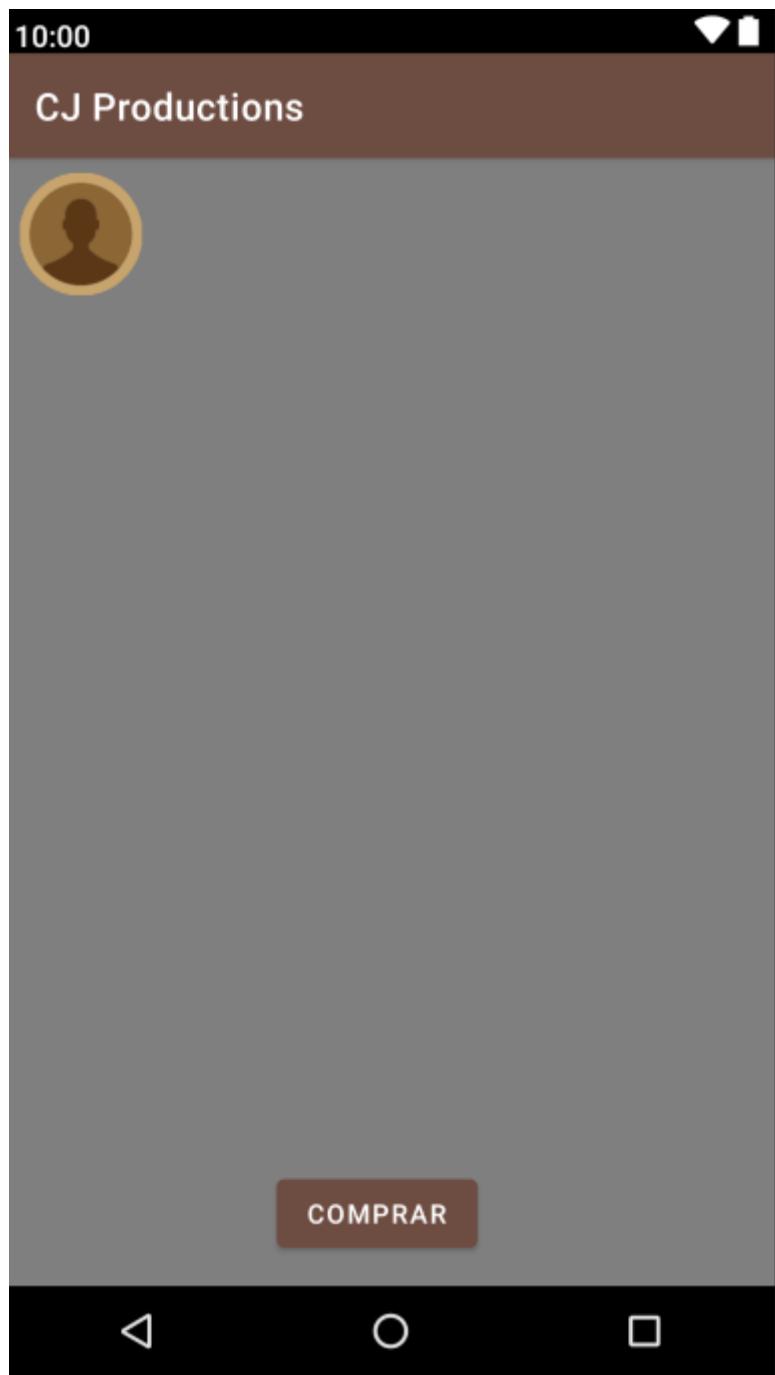
activity_restablecer_password.xml



En estos últimos días se ha cambiado el color de la interfaz para hacerla más amigable, he aquí el themes.xml



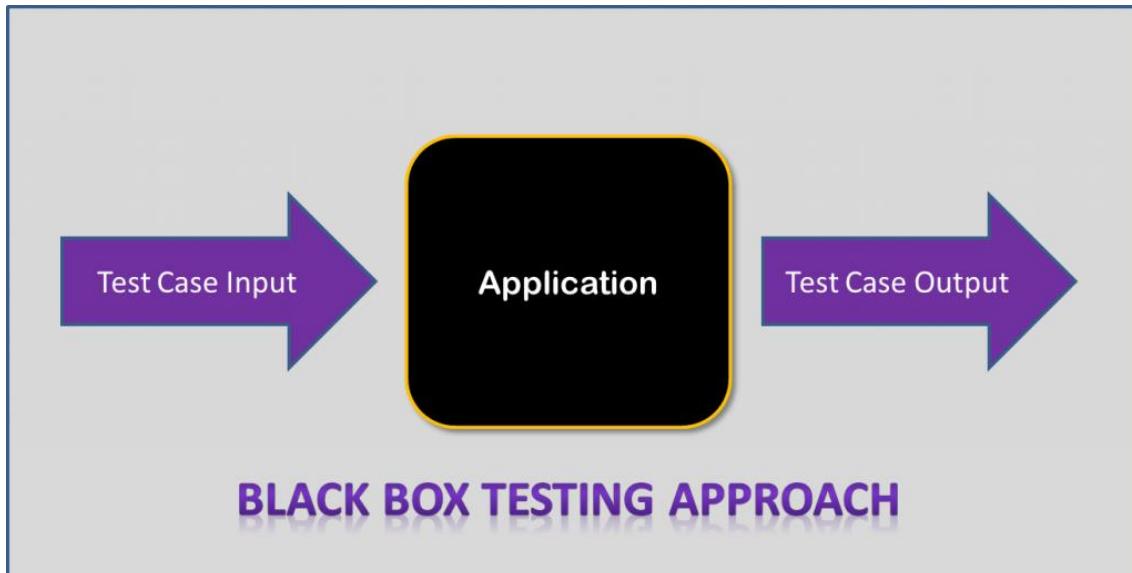
Por último, se ha cambiado el icono de Ver perfil que aparece en el mainActivity.tk, he aquí su nuevo aspecto



Pruebas

Voy a mostrar unas cuantas pruebas que se han realizado para comprobar el correcto funcionamiento de la aplicación.

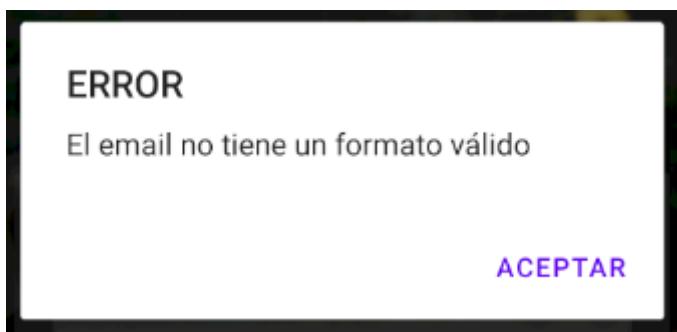
Caja Negra



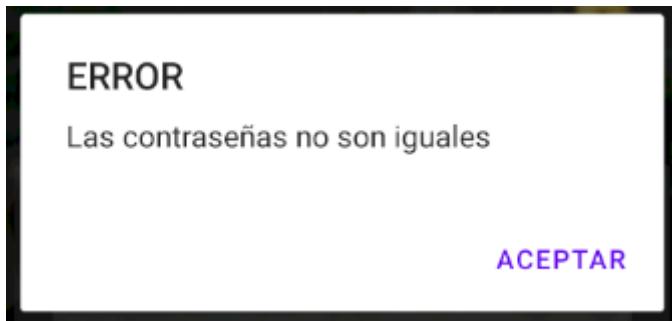
Un ejemplo de esta prueba la podemos encontrar en el apartado de registro o inicio de sesión.

En el registro, se debe de introducir un correo electrónico y 2 contraseñas que deben de coincidir.

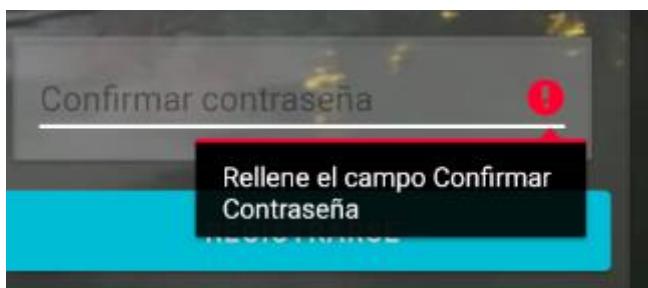
Si el formato del correo introducido es incorrecto la aplicación lo notificará y no dejará continuar. El mensaje que nos aparece es este



De la misma forma, si las 2 contraseñas no coinciden nos aparecerá el siguiente mensaje

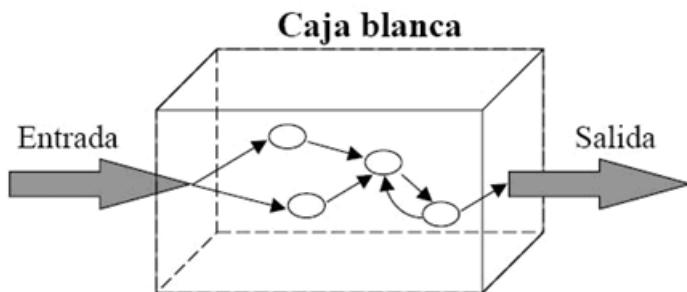


Y por ultimo, en caso de que haya algún campo vacío nos aparecerá lo siguiente



Una vez todos los campos estén correctos, el usuario se registrará correctamente.

Caja blanca



Para este caso (y acompañando al anterior punto) he decidido explicar cómo se realiza el proceso de validación de los campos de registro.

A continuación se mostrará el método que se encarga de esa función

```
/*
 * Función que comprueba que los campos de texto no estén vacíos
 */
private fun comprobar(): Boolean {
    var email = etEmail.text.toString().trim()
    var contraseña = etContrasena.text.toString().trim()
    var contraseña2 = etContrasena2.text.toString().trim()

    if (!isOk(email)) {
        etEmail.error = "Rellene el campo %s".format(...args: "Email")
        return false
    }
    if (!isOk(contraseña)) {
        etContrasena.error = "Rellene el campo %s".format(...args: "Contraseña")
        return false
    }
    if (!isOk(contraseña2)) {
        etContrasena2.error = "Rellene el campo %s".format(...args: "Confirmar Contraseña")
        return false
    }

    //Comprueba que los dos campos de las contraseñas son iguales
    if (!(contraseña == contraseña2)) {
        showAlert("Las contraseñas no son iguales")
        return false
    }

    if (contraseña.length < 6) {
        showAlert("La contraseña tiene que tener mínimo 6 caracteres")
        return false
    }

    //Comprueba que el email introducido es valido (ojo, puede no existir pero ser valido)
    val patronEmail: Pattern = Patterns.EMAIL_ADDRESS

    if (!patronEmail.matcher(email).matches()) {
        showAlert("El email no tiene un formato válido")
        return false
    }

    //Si llega hasta aquí quiere decir que todos los campos son validos
    return true
}
```

Primero de todo, si cualquiera de los 3 campos están vacíos (esa comprobación la hace el método `isOk` que devuelve verdadero en caso de SI se haya escrito algo) es cuando muestra el mensaje de que dicha cadena en cuestión está vacía y debe de ser rellenada.

Una vez comprobado eso, pasa a comprobar que los campos de contraseña tengan una longitud mínima de 6 caracteres (en caso de no ser así mostrará

una alerta igual que a las dos mencionadas en el apartado de la Caja Negra, pero con un mensaje adaptado al error).

Tras eso, se comprueba que las contraseñas son iguales. Y por último se comprueba que el email tiene un formato valido. Ojo, solo comprueba que tenga un formato valido, que sea válido no quiere decir que ese email exista.

Si todo lo anterior está correcto, quiere decir que todos los campos están correctos y se procederá al registro de sesión.

Anexos

Anexo 1

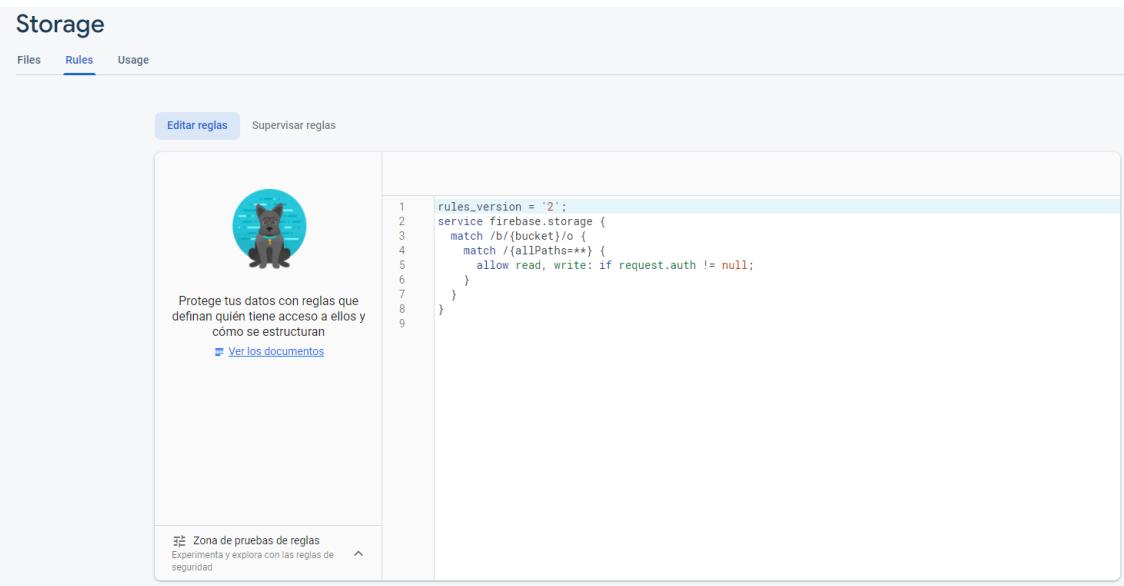
Manual del Usuario y el Administrador

https://drive.google.com/file/d/1_T8lISGi6yT2RKXciLsgi5KvW2wSiHYt/view?usp=sharing

Conclusiones finales

Problemas encontrados y soluciones

1. Debido a que no he tenido prácticas, al principio me resultó bastante complicado realizar la aplicación, sobre todo porque casi todo lo he tenido que buscar en internet ya sea en YouTube o en otras páginas.
2. Respecto a Firebase ha ocurrido lo mismo que en el apartado anterior, todo lo he tenido que buscar por mi cuenta, pero gracias a la documentación de Firebase y algunos tutoriales no me ha resultado complicado aprender su funcionamiento.
3. Otro problema fue con los recyclerView que capturan los datos directamente desde la base de datos. Resulta que si no hay sesión iniciada, no se puede acceder al Storage de Firebase ni para modificar contenido ni para leerlo puesto que la regla que hay lo impide. La regla en cuestión es esta



Y este es el error que provoca

```
W/e.cjproduction: Accessing hidden method Lsun/misc/Unsafe;->getObject(Ljava/lang/Object;)Ljava/lang/Object; (greylist, linking, allowed)
E/StorageUtil: error getting token java.util.concurrent.ExecutionException: com.google.firebaseio.internal.api.FirebaseNoSignedInUserException: Please sign in before trying to get a token.
W/NetworkRequest: no auth token for request
E/StorageException: StorageException has occurred.
User does not have permission to access this object.
Code: 13023 HttpResult: 403
E/StorageException: ( "code": 403, "message": "Permission denied. Could not perform this operation" )
java.io.IOException: ( "error": ( "code": 403, "message": "Permission denied. Could not perform this operation" ))
at com.google.firebaseio.storage.network.NetworkRequest.parseErrorResponse(NetworkRequest.java:434)
at com.google.firebaseio.storage.network.NetworkRequest.parseErrorResponse(NetworkRequest.java:451)
at com.google.firebaseio.storage.network.NetworkRequest.processResponseStream(NetworkRequest.java:442)
at com.google.firebaseio.storage.network.NetworkRequest.performRequest(NetworkRequest.java:272)
at com.google.firebaseio.storage.network.NetworkRequest.performRequest(NetworkRequest.java:286)
at com.google.firebaseio.storage.internal.ExponentialBackoffSender.sendWithExponentialBackoff(ExponentialBackoffSender.java:70)
at com.google.firebaseio.storage.internal.ExponentialBackoffSender.sendWithExponentialBackoff(ExponentialBackoffSender.java:62)
at com.google.firebaseio.storage.internal.ExponentialBackoffSender.getDownloadUrlInternal(ExponentialBackoffSender.java:50)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1167)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:641)
at java.lang.Thread.run(Thread.java:919)
V/FirebaseAuth: Inactivity, disconnecting from the service
```

Para solucionarlo, es necesario modificar dicha regla para que se permita al menos, leer contenido aunque el usuario no esté registrado. Para ello la propia documentación de Firebase nos indica cómo hacerlo.

Privado autenticado

En ciertos casos, es posible que desee que todos los usuarios autenticados de su aplicación puedan ver los datos, pero no los usuarios no autenticados. Dado que la variable `request.auth` es `null` para todos los usuarios no autenticados, todo lo que tiene que hacer es verificar que la variable `request.auth` exista para requerir autenticación:

```
// Require authentication on all internal image reads
match /internal/{imageId} {
  allow read: if request.auth != null;
}
```



También me guié de StackOverflow para poder saber más acerca de este error.

La forma de solucionarla es la siguiente, basta con modificar dicho código:

The screenshot shows the Firebase Storage Rules editor. At the top, there are tabs for 'Files', 'Rules' (which is selected), and 'Usage'. Below the tabs, there are buttons for 'Editar reglas' (Edit rules) and 'Supervisar reglas' (Monitor rules). The main area contains a code editor with the following content:

```
rules_version = '2';
service firebase.storage {
  match /{b}/{bucket} {
    match /{allPaths=**} {
      //Se ha comentado la parte en la que se requiere que el usuario se autentique
      allow read, write /*: if request.auth != null*/;
    }
  }
}
```

To the left of the code editor, there is a placeholder image of a dog and a note: 'Protege tus datos con reglas que definen quién tiene acceso a ellos y cómo se estructuran'. Below the code editor, there is a section titled 'Zona de pruebas de reglas' (Rule testing zone) with the sub-instruction 'Experimenta y explora con las reglas de seguridad'.

Tras esto esperamos unos minutos a que Firebase se actualice y listo, ya está el problema solucionado:

The screenshot shows a news article titled 'Fecha de publicación de la demo' (Release date of the demo). The article features a thumbnail image of a person playing a guitar in a forest. The text of the article is as follows:

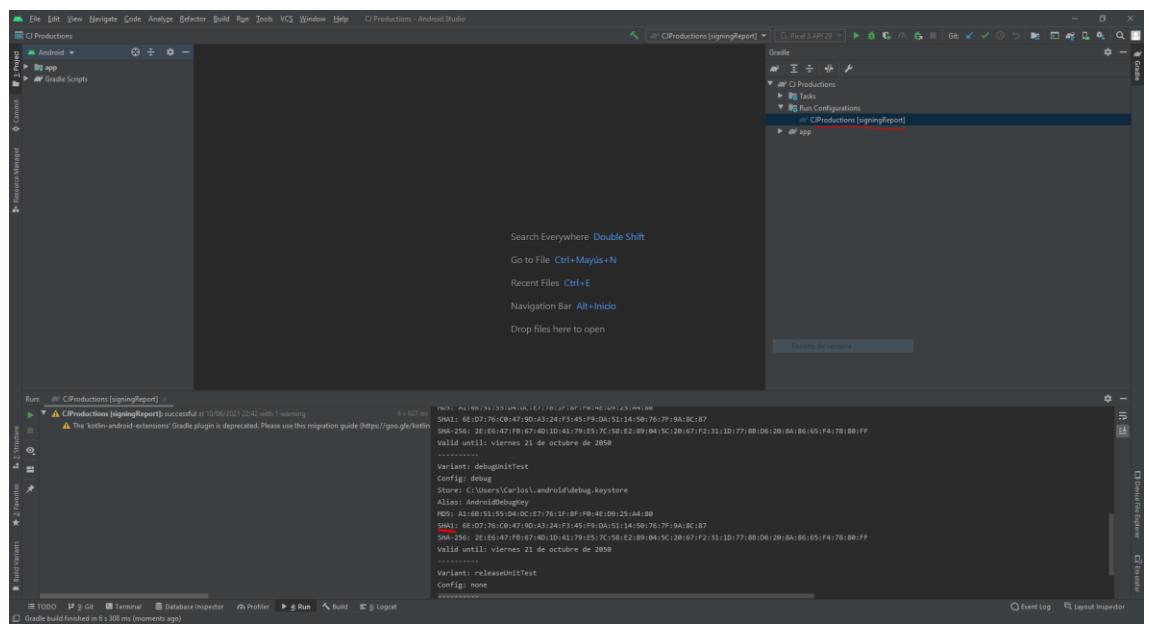
Tras varios meses de espera, nos complace anunciar que la demo del videojuego "Days Gone" será publicada el día 17 de junio de forma gratuita. Esperamos que os guste y nos haría ilusión saber vuestras opiniones.

Noticia del 27-05-2020

4. El inicio de sesión con Google funciona correctamente en el emulador, pero al momento de hacer uso de esa función en el dispositivo físico no funciona. He investigado y este error se debe a que la huella digital está caducada.

Aquí se resumen los pasos para solucionar este problema:

1. Nos dirigimos al Gradle que se encuentra en la parte derecha de Android Studio y le damos al desplegable SigningReport, donde veremos una huella digital.



2. Ahora abrimos la terminal de Windows y nos vamos a la carpeta de instalación de Java.

```

Símbolo del sistema
Microsoft Windows [Versión 10.0.19042.985]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Carlos>cd "C:\Program Files\Java"

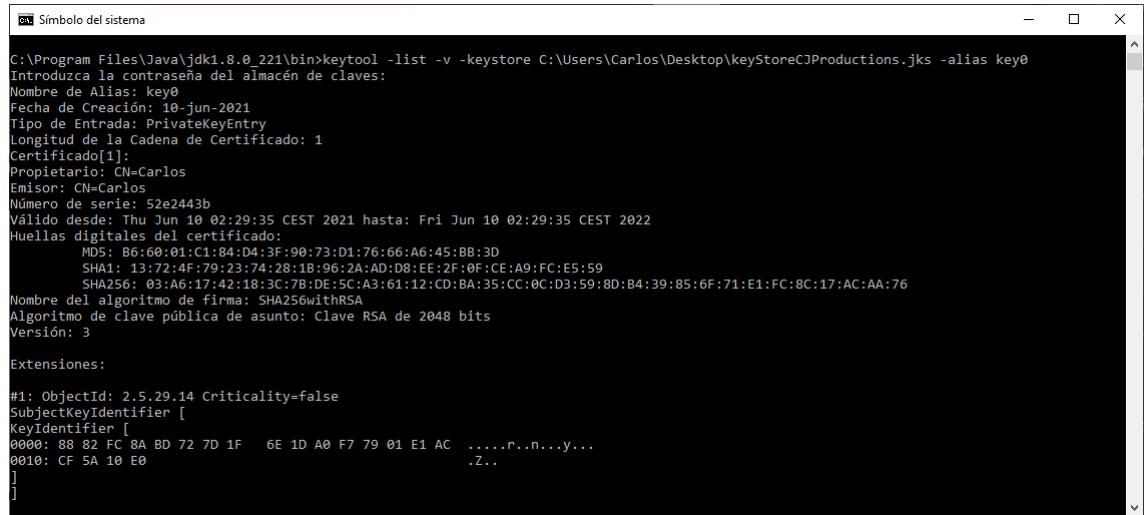
C:\Program Files\Java>cd "jdk1.8.0_221\bin"

C:\Program Files\Java\jdk1.8.0_221\bin>

```

3. Una vez en la carpeta, ponemos el siguiente código:

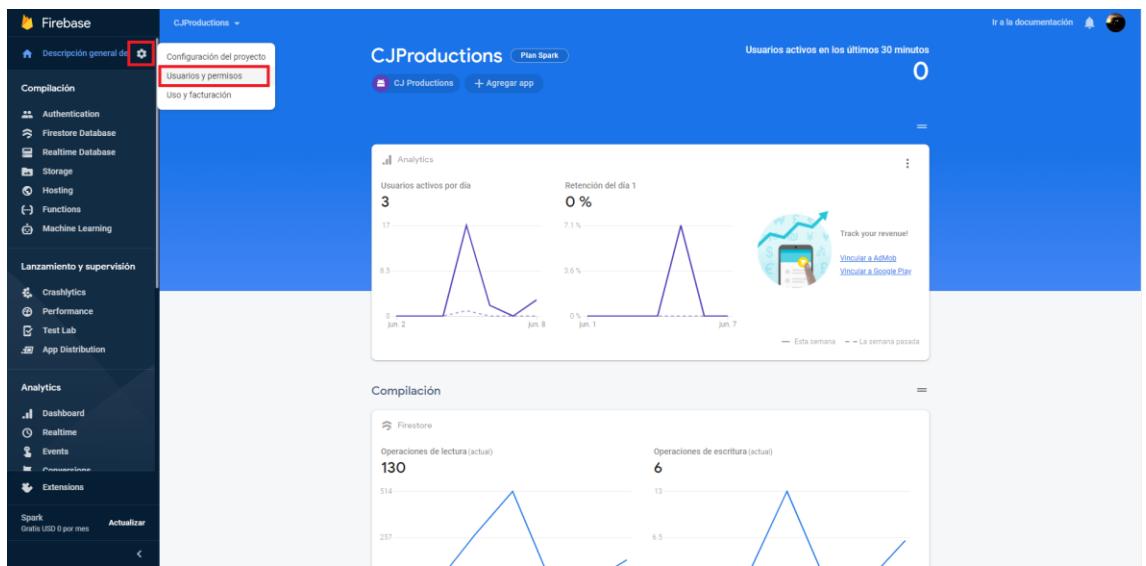
```
keytool -list -v -keystore <nombre_keystore> -alias <nombre_alias>
```



```
C:\Program Files\Java\jdk1.8.0_221\bin>keytool -list -v -keystore C:\Users\Carlos\Desktop\keyStoreCJProductions.jks -alias key0
Introduzca la contraseña del almacén de claves:
Nombre de Alias: key0
Fecha de Creación: 10-jun-2021
Tipo de Entrada: PrivateKeyEntry
Longitud de la Cadena de Certificado: 1
Certificado[1]:
Propietario: CN=Carlos
Emisor: CN=Carlos
Número de serie: 52e2443b
Válido desde: Thu Jun 10 02:29:35 CEST 2021 hasta: Fri Jun 10 02:29:35 CEST 2022
Huellas digitales del certificado:
    MD5: 86:60:01:c1:84:04:3F:90:73:D1:76:66:A6:45:BB:3D
    SHA1: 13:72:4F:79:23:74:28:1B:96:2A:AD:D8:EE:2F:0F:CE:A9:FC:E5:59
    SHA256: 03:A6:17:42:18:3C:7B:DE:5C:3A:61:12:CD:BA:35:CC:0C:D3:59:8D:B4:39:85:6F:71:E1:FC:8C:17:AC:AA:76
Nombre del algoritmo de firma: SHA256withRSA
Algoritmo de clave pública de asunto: Clave RSA de 2048 bits
Versión: 3

Extensiones:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 88 82 FC 8A BD 72 7D 1F   6E 1D A0 F7 79 01 E1 AC  ....r..n...y...
0010: CF 5A 10 E0               .Z..
]
]
```

4. Una vez hecho esto, tendremos otra huella digital SHA1, esa es la que tenemos que colocar en FireBase.



5. Ahora en “General” y al final encontraremos un apartado donde añadir la huella digital nueva. La añadimos y ya está el problema solucionado.

The screenshot shows the Firebase Project Configuration page for a project named "CJProductions". The "General" tab is selected. The page includes sections for "Tu proyecto" (Project details) and "Configuración pública" (Public configuration). In the "Tus apps" section, there is a configuration for an "Android" app with the package name "com.example.cjproductions". A specific SHA-1 certificate fingerprint is highlighted with a red box: "13:72:4F:79:23:74:28:1B:9E:2A:AD:D8:EE:2F:0F:CE:A9:FC:E5:59".

Bibliografía

- Web que me ha resultado de mucha utilidad
<https://stackoverflow.com/>
- Android Studio
<https://developer.android.com/docs>
<https://developer.android.com/>
- Información sobre como abrir una Sociedad Limitada
<https://institutocajasol.com/como-crear-una-sociedad-limitada/>
- Canal de Píldoras Informáticas en YouTube
<https://www.youtube.com/user/pildorasinformaticas>
- Video que ha sido usado de fondo en el MainActivity.kt
<https://www.youtube.com/watch?v=6Pm4sRdPioM&t=12s>
- Documentación sobre crear notificaciones Push
<https://developer.android.com/reference/androidx/core/app/NotificationCompat.Builder?hl=es>

<https://developer.android.com/training/notify-user/build-notification?hl=es#kotlin>

<https://developer.android.com/training/notify-user/expanded>

Fin