



UTT

UNIVERSIDAD TECNOLÓGICA DE TIJUANA

GOBIERNO DE BAJA CALIFORNIA

TEMA:

Proyecto Android Studio - ChatExample

PRESENTADO POR:

Maciel Leyva Ariana Lizeth

GRUPO:

9A

MATERIA:

Desarrollo para Dispositivos Inteligentes

PROFESOR:

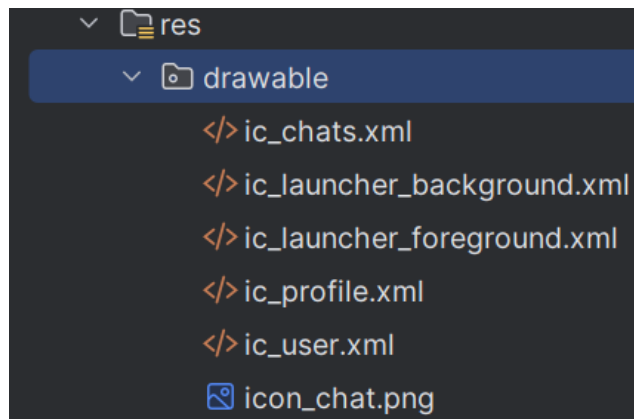
Ray Brunett Parra Galaviz

Tijuana, Baja California, 29 de octubre del 2024

“Chat Example”

Para realizar el proyecto de chatExample en Android Studio, hice lo siguiente:

Primero se comenzó realizando una interfaz para colocar 3 iconos, uno para profile, otro para users y el último de chats, estos los sacamos del catálogo de imágenes de vector que ya tiene incluido Android Studio, una vez que seleccionamos los iconos, los vamos a colocar en la carpeta res/drawable, tal como se ve en la siguiente captura de pantalla.



El siguiente paso fue crear el código para poder visualizar esos íconos en la interfaz que está incluida por defecto también. El código lo coloqué en el archivo que viene por default cuando se crea un proyecto que es el activity_main.xml, en ese archivo está el siguiente código que se observa en la imagen que está debajo.

```

LoginMailActivity.kt  activity_main.xml  activity_options_login.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:app="http://schemas.android.com/apk/res-auto"
3      xmlns:tools="http://schemas.android.com/tools"
4      xmlns:android="http://schemas.android.com/apk/res/android"
5      android:id="@+id/main"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".MainActivity"
9  >
10     <RelativeLayout
11         android:id="@+id/toolbarRL"
12         android:layout_width="match_parent"
13         android:layout_height="55dp"
14     >
15         <TextView
16             android:id="@+id/tv_title"
17             android:text="Title"
18             android:textStyle="bold"
19             android:layout_centerInParent="true"
20             android:layout_width="wrap_content"
21             android:layout_height="wrap_content"
22         />

```

```

23         <View
24             android:layout_alignParentBottom="true"
25             android:background="#ccc"
26             android:layout_width="match_parent"
27             android:layout_height="1dp"
28         />
29     </RelativeLayout>
30
31     <FrameLayout
32         android:id="@+id/fragmentFL"
33         android:layout_above="@+id/view"
34         android:layout_below="@+id/toolbarRL"
35         android:layout_width="match_parent"
36         android:layout_height="match_parent"
37     />

```

```

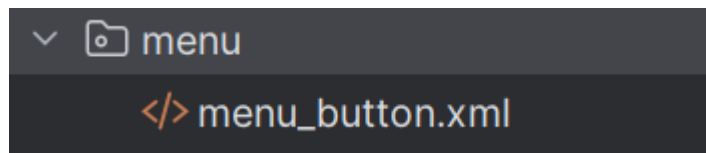
39  <View
40      android:id="@+id/view"
41      android:background="#ccc"
42      android:layout_above="@+id/bottomNV"
43      android:layout_width="wrap_content"
44      android:layout_height="1dp"
45  />
46
47  <com.google.android.material.bottomnavigation.BottomNavigationView
48      android:id="@+id/bottomNV"
49      android:layout_alignParentBottom="true"
50      app:menu="@menu/menu_button"
51      android:layout_width="match_parent"
52      android:layout_height="wrap_content"
53  />
54  </RelativeLayout>
55

```

Cómo se puede observar en las 3 imágenes anteriores, se muestra el código del activity_main.xml, el cual contiene varios componentes como lo son el RelativeLayout, TextView, View, FrameLayout y un BottomNavigationView hasta el final del código, y a cada componente se le colocaron los atributos necesarios para que el acomodo de los elementos en la interfaz y en el dispositivo virtual se vean bien.

Posteriormente se creó otro archivo en la carpeta menú que se encuentra dentro de la carpeta res, a ese archivo lo nombramos como menu_button.xml, ahí declaramos los íconos que usamos anteriormente como item, en donde se colocaron los atributos de ID, el título (nombre) para cada uno y la ubicación de donde se encuentran, que como mencioné antes, estos están en la carpeta drawable.

```
LoginMailActivity.kt  </> menu_button.xml  </> activity_main.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/and
3      <item
4          android:id="@+id/item_profile"
5          android:title="Profile"
6          android:icon="@drawable/ic_profile"
7      />
8      <item
9          android:id="@+id/item_users"
10         android:title="Users"
11         android:icon="@drawable/ic_user"
12     />
13     <item
14         android:id="@+id/item_chats"
15         android:title="Chats"
16         android:icon="@drawable/ic_chats"
17     />
18 </menu>
```



Lo siguiente fue crear 3 códigos más, llamados fragment_chats.xml, fragment_profile.xml y fragment_users.xml, en ellos se colocó un TextView para que, al momento de ejecutar la aplicación, cuando demos clic a alguno de los iconos, nos muestre el título correspondiente, por ejemplo, si damos clic en el ícono de chats, el texto “Chats” también aparezca en el dispositivo. En el TextView colocamos atributos como el layout_width, layout_height, text (ahí se colocó el título), textStyle que sirve para dar otro estilo al texto, textSize para asignar un tamaño y gravity que es la posición de donde queremos que aparezca ese texto.

```
</> fragment_chats.xml × </> fragment_profile.xml </> fragment_users.xml </> activity_r
1 <?xml version="1.0" encoding="utf-8"?>
2   <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     tools:context="fragments.FragmentChats">
7
8     <!-- TODO: Update blank fragment layout -->
9     <TextView
10       android:layout_width="match_parent"
11       android:layout_height="match_parent"
12       android:text="CHATS"
13       android:textStyle="bold"
14       android:textSize="25dp"
15       android:gravity="center"
16     />
17   </FrameLayout>
```

```

</> fragment_chats.xml  </> fragment_profile.xml × </> fragment_users.xml  </> activity_n
1  <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
2      xmlns:tools="http://schemas.android.com/tools"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      tools:context="fragments.FragmentProfile">
6
7      <!-- TODO: Update blank fragment layout -->
8      <TextView
9          android:layout_width="match_parent"
10         android:layout_height="match_parent"
11         android:text="PROFILE"
12         android:textStyle="bold"
13         android:textSize="25dp"
14         android:gravity="center"
15     />
16 </FrameLayout>

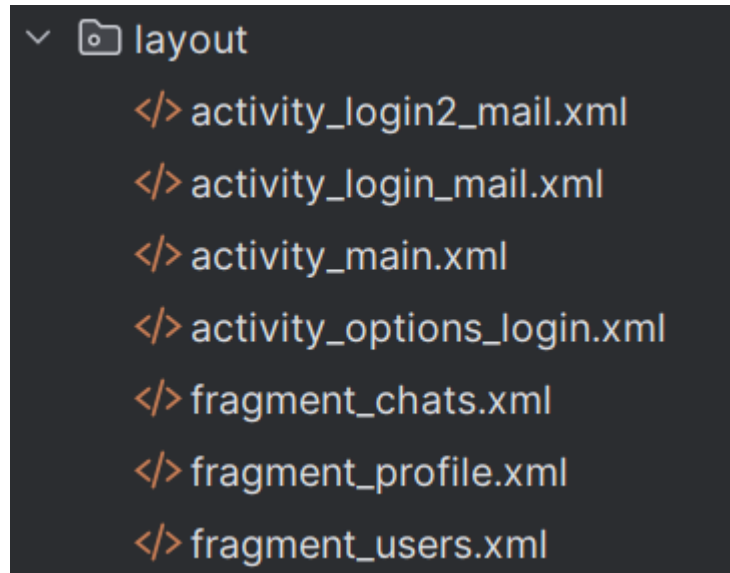
```

```

</> fragment_chats.xml  </> fragment_profile.xml  </> fragment_users.xml × </> activity_n
1  <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
2      xmlns:tools="http://schemas.android.com/tools"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      tools:context="fragments.FragmentUsers">
6
7      <!-- TODO: Update blank fragment layout -->
8      <TextView
9          android:layout_width="match_parent"
10         android:layout_height="match_parent"
11         android:text="USERS"
12         android:textStyle="bold"
13         android:textSize="25dp"
14         android:gravity="center"
15     />
16
17 </FrameLayout>

```

Estos 3 archivos también los ubicamos en la carpeta layout que está dentro de la carpeta res, tal como se observa en la imagen de abajo.



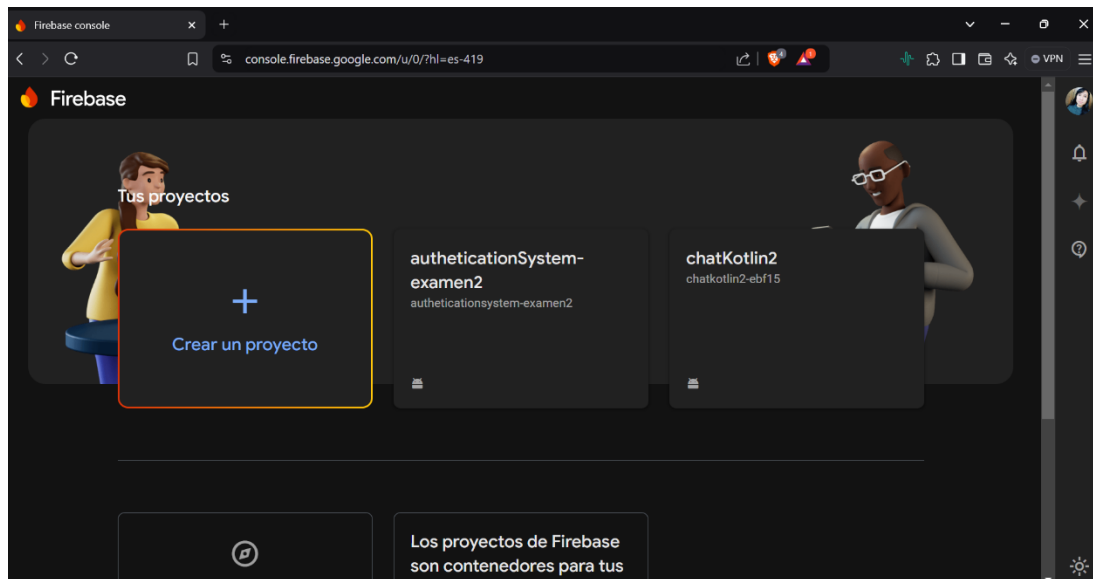
Después en el archivo de strings.xml declaré todos los iconos que utiliza mi aplicación mediante su etiqueta que es definir un nombre y este se coloca en los archivos xml de forma directa ya que Android Studio los detecta con mayor facilidad cuando están declarados en este archivo.

```
1  <resources>
2      <string name="app_name">chatExample</string>
3      <string name="item_profile">Profile</string>
4      <string name="item_users">Users</string>
5      <string name="item_chats">Chats</string>
6      <string name="tv_title">Title</string>
7
8      <string name="txt_login">Login</string>
9      <string name="et_email">Email</string>
10     <string name="et_password">Password</string>
11     <string name="btn_ingresar">Ingresar</string>
12     <string name="txt_no_tienes_cuenta">No tienes una Cuenta?</string>
13     <string name="tv_registrarme">Registrarme</string>
14
15     <!-- TODO: Remove or change this placeholder text -->
16     <string name="hello_blank_fragment">Hello blank fragment</string>
17 </resources>
```

Y este es el resultado final que se obtuvo luego de haber realizado todos los códigos anteriores.

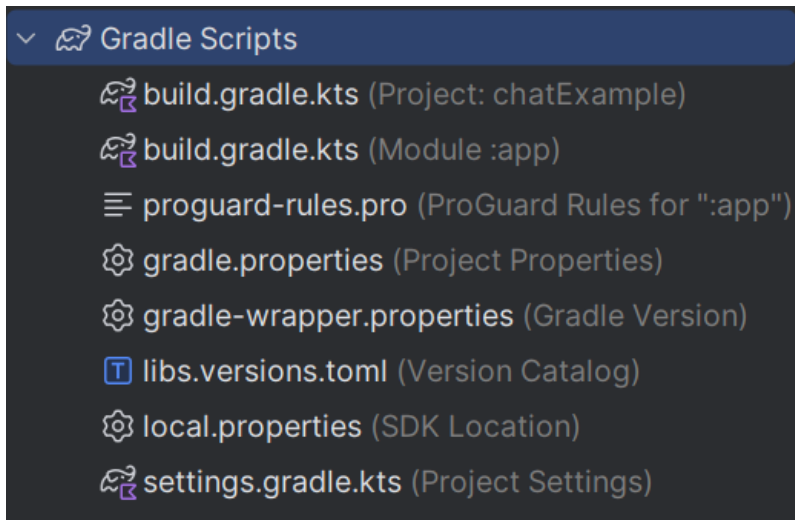


El siguiente paso es utilizar e integrar Firebase a la aplicación en Android Studio para establecer la conexión con esa plataforma, para ello nos dirigimos a la página oficial y buscamos la opción ir a la consola, ahí vamos a crear un proyecto nuevo, el cual nombré como chatKotlin2, esperamos unos segundos a que la aplicación se haya creado en Firebase. Luego de haberse creado nos preguntará a qué aplicación la queremos integrar, para Android o iOS, daremos clic en el icono de Android y ahí nos pedirá algunos datos para poder agregar Firebase a nuestra app de Android Studio, estos datos son el nombre del paquete de Android, un sobrenombre de la app que es opcional, y el certificado de firma SHA-1 de depuración que es opcional igual pero en mi caso sí coloqué el certificado de firma y el nombre del paquete.

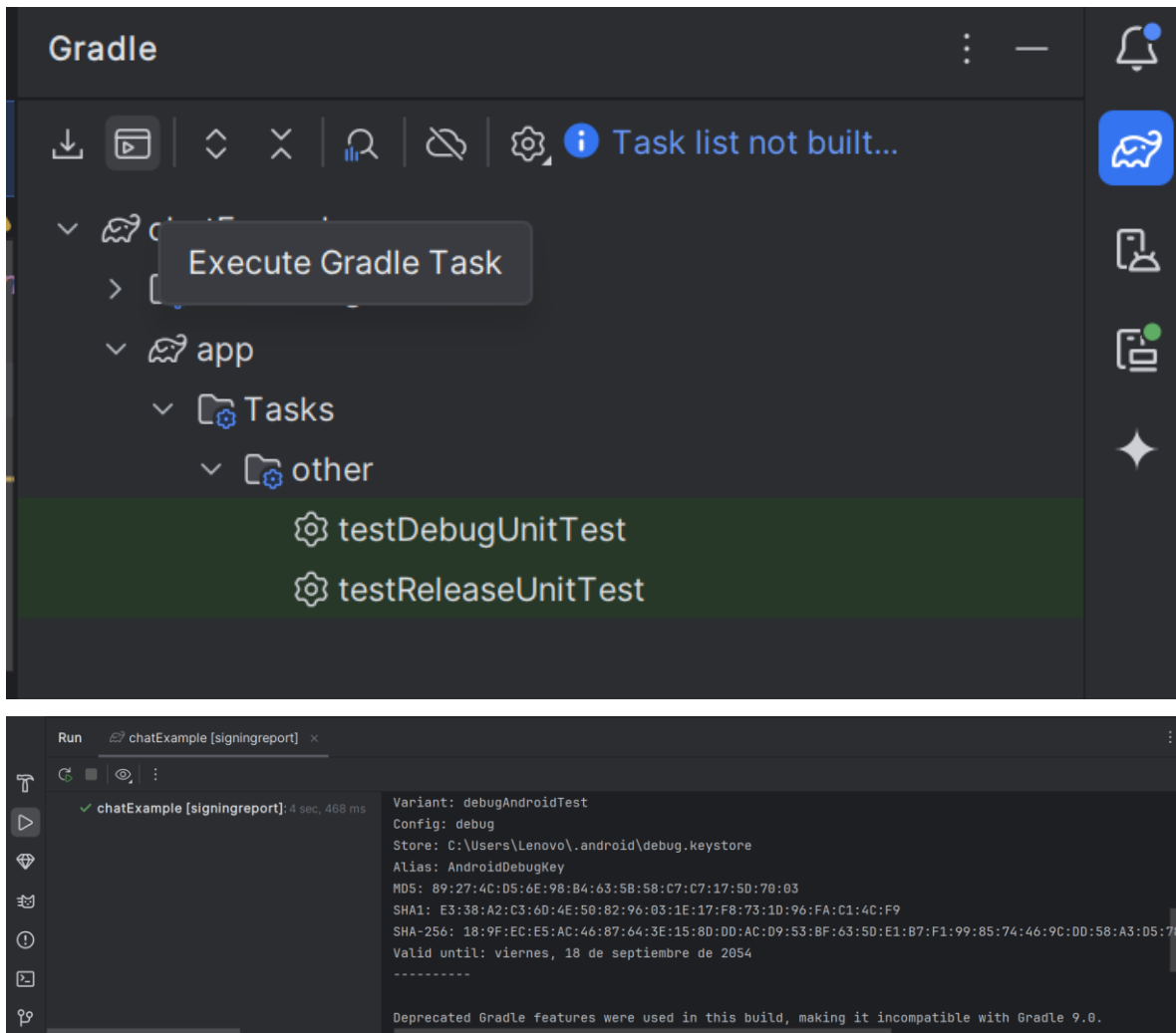
The image shows the 'Agrega Firebase a tu app para Android' (Add Firebase to your Android app) dialog. The 'Registrar app' (Register app) step is active. The 'Nombre del paquete de Android' (Android package name) field is filled with 'com.company.appname'. The 'Sobrenombre de la app (opcional)' (App nickname (optional)) field is filled with 'Mi app para Android'. The 'Certificado de firma SHA-1 de depuración (opcional)' (Optional SHA-1 debug signing certificate) field is filled with a placeholder value. A note at the bottom states: 'Obligatorio para Dynamic Links y el Acceso con Google o la compatibilidad con un número de teléfono en Auth. Puedes editar las claves SHA-1 en Configuración.' (Required for Dynamic Links and Google Sign-in or compatibility with a phone number in Auth. You can edit the SHA-1 keys in Settings.)

Para poder obtener el nombre del paquete, tuve que ir al proyecto de Android Studio, y en la carpeta de Gradle Scripts, busqué un archivo llamado build.gradle.kts (app), en él vienen características adicionales que también necesita el proyecto y que por default ya los incluye, como los plugins (complementos), dependencias, configuración por default, entre otros parámetros, lo que buscamos es el nombre del paquete de Android, este lo podemos tomar del script que dice android que es el que se puede ver en la imagen de abajo que viene siendo el com.example.chatexample, ese lo vamos a colocar en Firebase.

```
android {  
    namespace = "com.example.chatexample"  
    compileSdk = 34  
  
    defaultConfig {  
        applicationId = "com.example.chatexample"  
        minSdk = 24  
        targetSdk = 34  
        versionCode = 1  
        versionName = "1.0"  
  
        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"  
    }  
}
```




Para poder tener la firma SHA-1 nos vamos a ir al icono que dice Gradle que se encuentra en la parte derecha, y seguido vamos a dar clic en el icono que es de una consola (Execute Gradle Task) y ahí vamos a colocar el comando gradle signingreport, damos un enter, y revisamos en Run que es el icono de un triángulo que está de lado izquierdo y ahí es donde nos arroja el certificado de firma SHA-1 tal como se ve en las imágenes.




Luego de haber colocado ambos datos, damos clic en la opción Guardar y en la configuración del proyecto en la página de Firebase se pueden visualizar los datos y la información detallada del proyecto chatKotlin2, ahí nos muestra el nombre, el ID que se asigna por default, número de proyecto que también Firebase lo asigna por default y una clave de API web que es un token que igual se genera ahí mismo y más para abajo viene otro ID que es de la app, el sobrenombre que se colocó automáticamente y el certificado SHA-1 que es el que obtuvimos con los pasos anteriores, en las siguientes imágenes se puede observar esa parte.

Apps para Android

 chatKotlin2
com.example.chatexample

Configuración del SDK

¿Necesitas volver a configurar los SDK de Firebase en tu app? Revisa las instrucciones de configuración del SDK o descarga el archivo de configuración con las claves y los identificadores de tu app.

 Ver las instrucciones del SDK

[📄 google-services.json](#)

ID de la app ⓘ

1:432777327046:android:f79cef85eeb323c99002c7

Sobrenombre de la app

chatKotlin2 ✎

Nombre del paquete

com.example.chatexample

Huellas digitales del certificado SHA ⓘ	Tipo ⓘ
e3:38:a2:c3:6d:4e:50:82:96:03:1e:17:f8:73:1d:96:fa:c1:4c:f9	SHA-1

[Agregar huella digital](#)

chatKotlin2 ▾ Configuración de proyecto

[General](#) [Cloud Messaging](#) [Integraciones](#) [Cuentas de servicio](#) [Privacidad de los datos](#) [Usuarios y permisos](#)

Tu proyecto

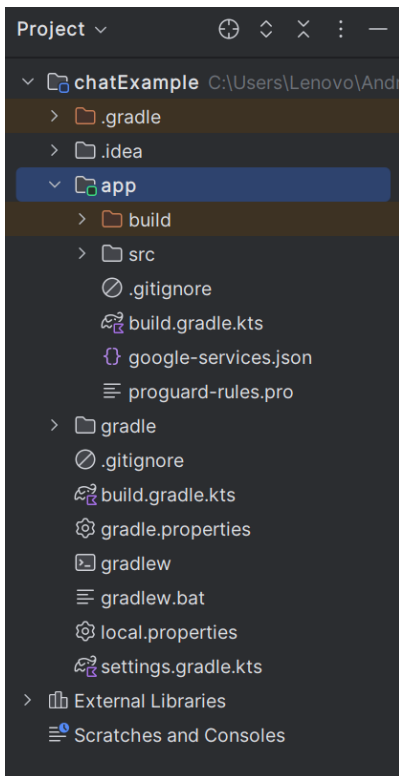
Nombre del proyecto	chatKotlin2 ✎
ID del proyecto ⓘ	chatkotlin2-ebf15
Número de proyecto ⓘ	432777327046
Clave de API web ⓘ	AlzaSyA76lG3jPqIMNPsEzof5FcqaVvXxE0Ywdg

Entorno

Este parámetro de configuración permite personalizar tu proyecto para las diferentes etapas del ciclo de vida de la app

Tipo de entorno	Sin especificar ✎
-----------------	-------------------

Otro paso que nos indica Firebase es descargar un archivo tipo json llamado google-services y ahí mismo nos dice en donde debemos guardarlo, cambiamos a la opción Project, buscamos la carpeta app, luego la carpeta src que está dentro de app y ahí vamos a ubicar el json que nos generó la plataforma de Firebase, ese archivo también nos ayuda al funcionamiento de la aplicación.



Ahora continuamos con los códigos que faltan para que la aplicación funcione, para ello hice otros archivos en la carpeta de layout, uno llamado `activity_options_login.xml` y otro llamado `activity_login2_mail.xml`, en el de `activity_login2_mail.xml` hice un login muy sencillo que es para el usuario ingrese sus credenciales que es su correo y una contraseña, para eso coloqué varios elementos en código como `TextView`, `TextInputLayout`, `EditText` y un `MaterialButton`, a cada componente le agregué los atributos necesarios para que tengan un diseño discreto y a la misma vez que se vea bien.

```
</> activity_login2_mail.xml × LoginMailActivity.kt </> activity_options_login.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.appcompat.widget.LinearLayoutCompat
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:id="@+id/main"
7     android:layout_width="match_parent"
8     android:layout_height="match_parent"
9     tools:context=".LoginMailActivity"
10    android:gravity="center"
11    android:orientation="vertical"
12 >
13    <TextView
14        android:text="Login"
15        android:textAllCaps="true"
16        android:textSize="25dp"
17        android:layout_width="wrap_content"
18        android:layout_height="wrap_content"
19    />
```

```
<com.google.android.material.textfield.TextInputLayout
    android:layout_marginTop="20dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    >
    <EditText
        android:id="@+id/et_email"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
    />
</com.google.android.material.textfield.TextInputLayout>

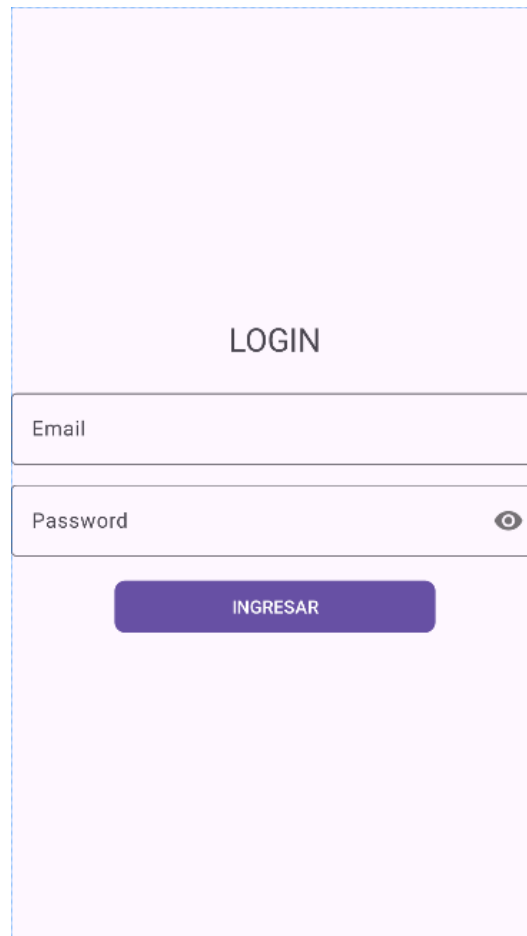
<com.google.android.material.textfield.TextInputLayout
    android:layout_marginTop="10dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:passwordToggleEnabled="true"
    >
```

```

41     <EditText
42         android:id="@+id/et_password"
43         android:hint="Password"
44         android:inputType="textPassword"
45         android:layout_width="match_parent"
46         android:layout_height="wrap_content"
47     />
48 </com.google.android.material.textfield.TextInputLayout>
49
50 <com.google.android.material.button.MaterialButton
51     android:id="@+id/btn_ingresar"
52     android:text="Ingresar"
53     android:textAllCaps="true"
54     android:layout_marginTop="15dp"
55     app:cornerRadius="8dp"
56     android:layout_width="250dp"
57     android:layout_height="wrap_content"
58 />
59
60 </androidx.appcompat.widget.LinearLayoutCompat>

```

Así es como se mira en la interfaz del celular en Android Studio, el cual es un login que contiene el título al centro, seguido de dos EditText y dos TextInputLayout que es el nombre que tiene cada recuadro para ingresar datos (Email y Password) y hasta abajo un botón centrado con el texto "Ingresar".

A login form with a light purple background. At the top, the word "LOGIN" is centered in a dark grey font. Below it are two input fields: the first is labeled "Email" and the second is labeled "Password" with a small eye icon to its right. At the bottom, there is a purple button with the text "INGRESAR" in white, centered.

Y en el archivo de `activity_options_login.xml` hice un código pequeño con el propósito de que cuando el usuario dé clic a alguno de los dos botones lo envíe directamente al login para que pueda iniciar sesión, para eso solamente agregué dos botones y un ícono de mensaje que descargué de internet el cual coloqué en la carpeta de `drawable` ya que ahí se deben guardar todas las imágenes, iconos y fondos que queramos utilizar.

Los componentes que añadí fue un `ImageView` para visualizar la imagen que descargué, y dos `MaterialButton` con su respectivo texto cada uno que es realizar login con correo electrónico o con una cuenta de Google, el código se puede observar en las siguientes imágenes.

```
</> activity_options_login.xml × MainActivity.kt ×
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.appcompat.widget.LinearLayoutCompat
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:id="@+id/main"
7     android:layout_width="match_parent"
8     android:layout_height="match_parent"
9     tools:context=".options.LoginActivity"
10    android:orientation="vertical"
11    android:gravity="center"
12    >
13    <ImageView
14        android:layout_width="80dp"
15        android:layout_height="80dp"
16        android:src="@drawable/icon_chat"
17    />
```

```
19    <com.google.android.material.button.MaterialButton
20        android:id="@+id/optionEmail"
21        android:text="OptionEmail"
22        android:textAllCaps="true"
23        android:layout_marginTop="35dp"
24        android:layout_width="250dp"
25        android:layout_height="wrap_content"
26    />
27    <com.google.android.material.button.MaterialButton
28        android:id="@+id/optionGoogle"
29        android:text="OptionGoogle"
30        android:textAllCaps="true"
31        android:layout_width="250dp"
32        android:layout_height="wrap_content"
33    />
34 </androidx.appcompat.widget.LinearLayoutCompat>
```

Y la interfaz quedó sencilla, pero se ve bien, coloqué los atributos que consideré necesarios para los botones y para la imagen, en la imagen de abajo se observa el resultado.



Finalmente, realicé el código en MainActivity, en el cuál declaré funciones para redirigir al login, otras 3 funciones para los iconos de users, profile y chats, invocando los iconos también y hasta el principio del código algunos parámetros que son de Firebase que son necesarios para su funcionamiento. El código se puede ver en las imágenes de a continuación.

```

</> activity_options_login.xml  MainActivity.kt  x
1  package com.example.chatexample
2
3  > import androidx.appcompat.app.AppCompatActivity
15
16  </> class MainActivity : AppCompatActivity() {
17      private lateinit var binding: ActivityMainBinding
18      private lateinit var firebaseAuth: FirebaseAuth
19
20      override fun onCreate(savedInstanceState: Bundle?) {
21          super.onCreate(savedInstanceState)
22          binding = ActivityMainBinding.inflate(layoutInflater)
23          enableEdgeToEdge()
24
25          setContentView(binding.root)
26
27          firebaseAuth = FirebaseAuth.getInstance()
28
29          if(firebaseAuth.currentUser == null) {
30              gotoLogin()
31          }
32
33          ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
34              val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
35              v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
36              insets
37          }
38      }
39  }

```

```

</> activity_options_login.xml  MainActivity.kt  x
16  class MainActivity : AppCompatActivity() {
20      override fun onCreate(savedInstanceState: Bundle?) {
37          binding.bottomNV.setOnItemClickListener { item ->
38              when(item.itemId){
39                  R.id.item_profile -> {
40                      // Visualizar el Item PROFILE
41                      viewFragmentProfile()
42                      true
43                  }
44                  R.id.item_users -> {
45                      // Visualizar el Item USERS
46                      viewFragmentUsers()
47                      true
48                  }
49                  R.id.item_chats -> {
50                      // Visualizar el Item CHATS
51                      viewFragmentChats()
52                      true
53                  }
54                  else -> {
55                      false
56                  }
57              }
58          }
59      }

```

```

60     private fun goToLogin(){
61         startActivity(Intent(applicationContext, optionsLoginActivity::class.java))
62     }
63
64     private fun viewFragmentProfile() {
65         binding.tvTitle.text = "Profile"
66         val fragment = FragmentProfile()
67         val fragmentTransaction = supportFragmentManager.beginTransaction()
68         fragmentTransaction.replace(binding.fragmentFL.id, fragment)
69         fragmentTransaction.commit()
70     }

```

```

71     private fun viewFragmentUsers() {
72         binding.tvTitle.text = "Users"
73         val fragment = FragmentUsers()
74         val fragmentTransaction = supportFragmentManager.beginTransaction()
75         fragmentTransaction.replace(binding.fragmentFL.id, fragment)
76         fragmentTransaction.commit()
77     }
78     private fun viewFragmentChats() {
79         binding.tvTitle.text = "Chats"
80         val fragment = FragmentChats()
81         val fragmentTransaction = supportFragmentManager.beginTransaction()
82         fragmentTransaction.replace(binding.fragmentFL.id, fragment)
83         fragmentTransaction.commit()
84     }
85 }

```

Por último, ejecuté la aplicación con el emulador que tiene incluido Android Studio con el dispositivo virtual que elegimos, y el resultado es exitoso, ya que la aplicación funciona correctamente, no tiene ningún error.

