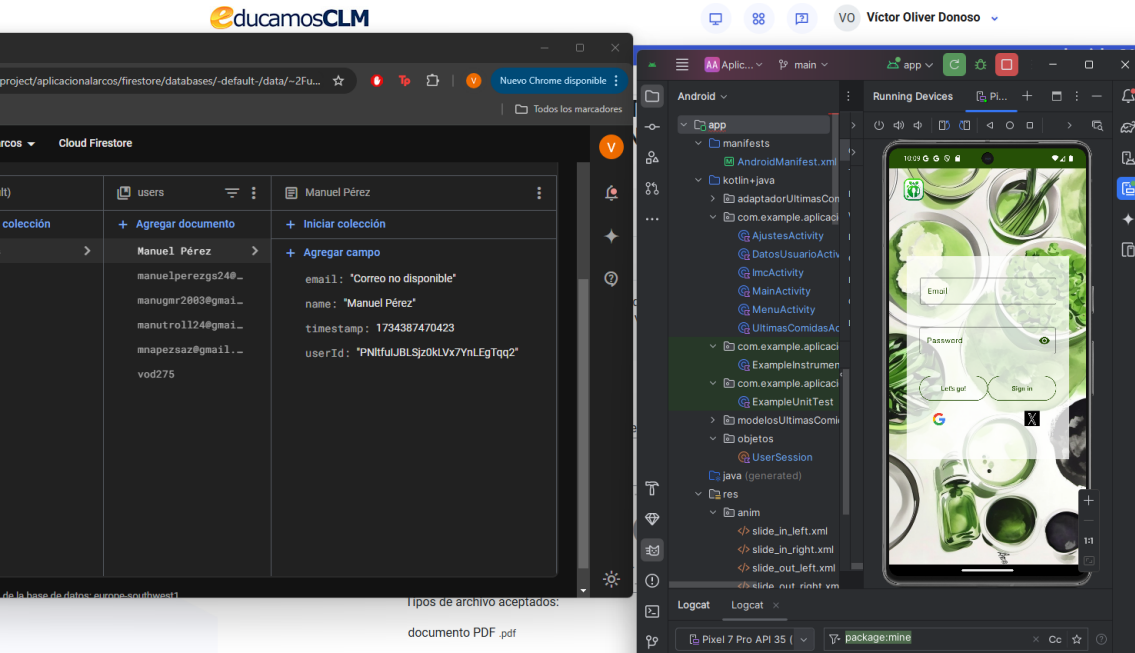
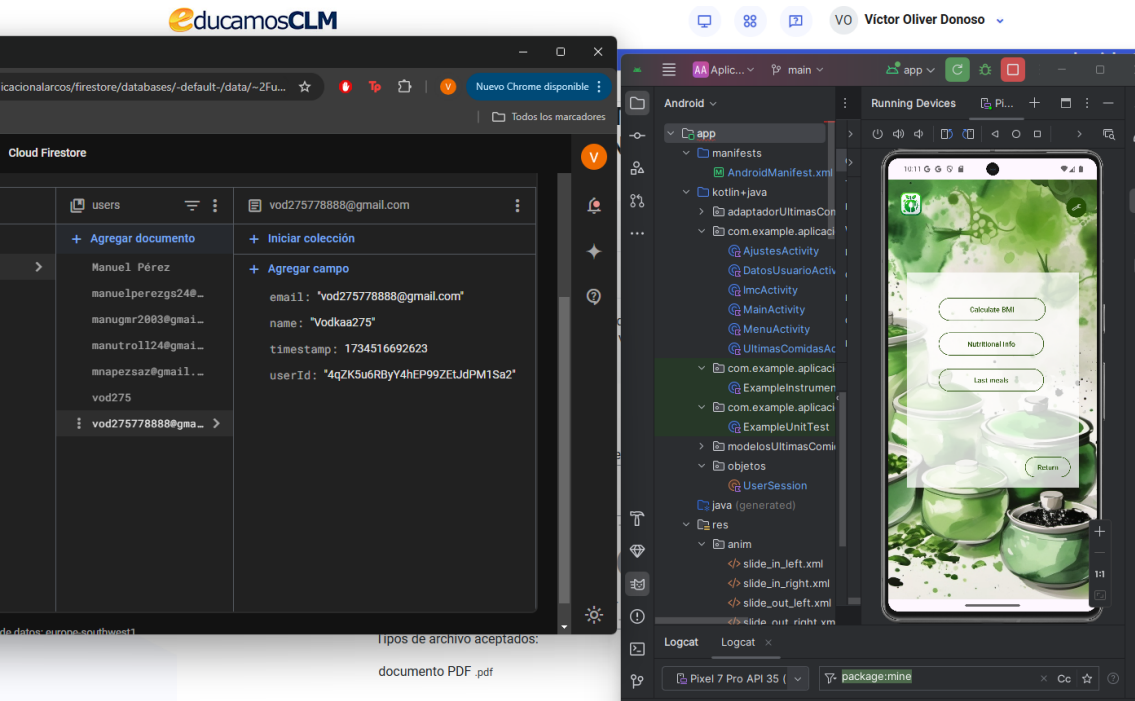


Hecho por Manuel Pérez y Víctor Oliver.

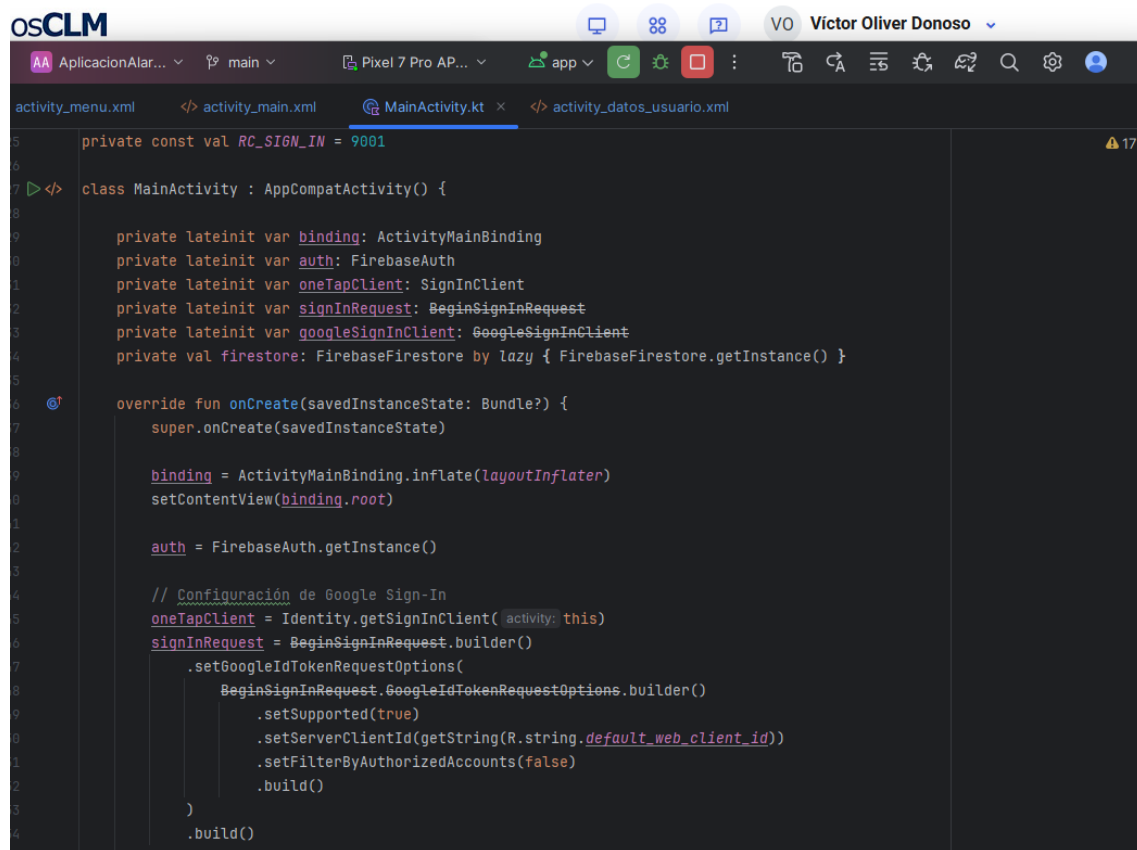
Aquí vemos que no hemos iniciado sesión



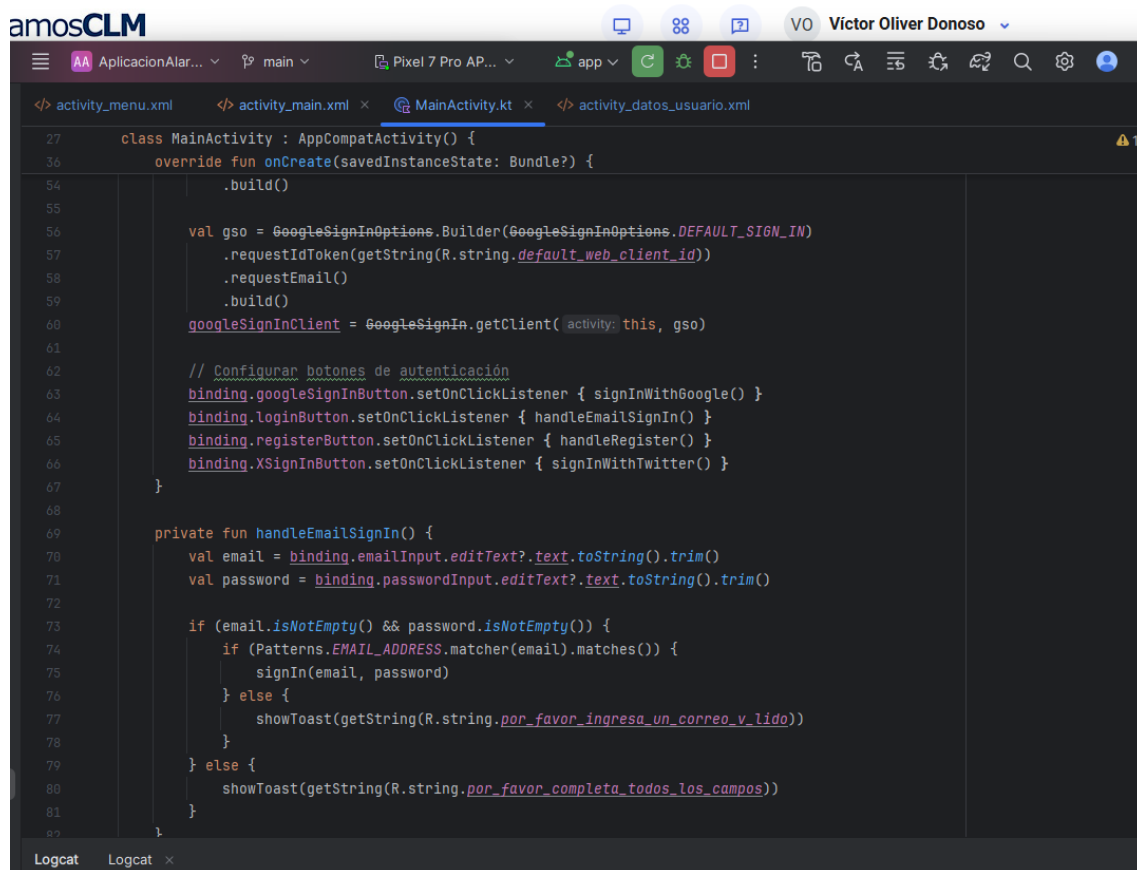
Yaqui iniciamos sesión con Google y se guarda el usuario vod27578888@gmail.com



Aquí la lógica de guardado de en firestore en el main



```
1 private const val RC_SIGN_IN = 9001
2
3 class MainActivity : AppCompatActivity() {
4
5     private lateinit var binding: ActivityMainBinding
6     private lateinit var auth: FirebaseAuth
7     private lateinit var oneTapClient: SignInClient
8     private lateinit var signInRequest: BeginSignInRequest
9     private lateinit var googleSignInClient: GoogleSignInClient
10    private val firestore: FirebaseFirestore by lazy { FirebaseFirestore.getInstance() }
11
12    override fun onCreate(savedInstanceState: Bundle?) {
13        super.onCreate(savedInstanceState)
14
15        binding = ActivityMainBinding.inflate(layoutInflater)
16        setContentView(binding.root)
17
18        auth = FirebaseAuth.getInstance()
19
20        // Configuración de Google Sign-In
21        oneTapClient = Identity.getSignInClient(this)
22        signInRequest = BeginSignInRequest.builder()
23            .setGoogleIdTokenRequestOptions(
24                BeginSignInRequest.GoogleIdTokenRequestOptions.builder()
25                    .setSupported(true)
26                    .setServerClientId(getString(R.string.default_web_client_id))
27                    .setFilterByAuthorizedAccounts(false)
28                    .build()
29            )
30            .build()
31    }
32}
```



```
27 class MainActivity : AppCompatActivity() {
28
29     override fun onCreate(savedInstanceState: Bundle?) {
30         .build()
31
32         val gso = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
33             .requestIdToken(getString(R.string.default_web_client_id))
34             .requestEmail()
35             .build()
36         googleSignInClient = GoogleSignIn.getClient(this, gso)
37
38         // Configurar botones de autenticación
39         binding.googleSignInButton.setOnClickListener { signInWithGoogle() }
40         binding.loginButton.setOnClickListener { handleEmailSignIn() }
41         binding.registerButton.setOnClickListener { handleRegister() }
42         binding.XSignInButton.setOnClickListener { signInWithTwitter() }
43     }
44
45     private fun handleEmailSignIn() {
46         val email = binding.emailInput.editText?.text.toString().trim()
47         val password = binding.passwordInput.editText?.text.toString().trim()
48
49         if (email.isNotEmpty() && password.isNotEmpty()) {
50             if (Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
51                 signIn(email, password)
52             } else {
53                 showToast(getString(R.string.por_favor_ingresa_un_correo_v_lido))
54             }
55         } else {
56             showToast(getString(R.string.por_favor_completa_todos_los_campos))
57         }
58     }
59 }
```

```

class MainActivity : AppCompatActivity() {
    27
    83
    84     private fun handleRegister() {
    85         val email = binding.emailInput.editText?.text.toString().trim()
    86         val password = binding.passwordInput.editText?.text.toString().trim()
    87
    88         if (email.isNotEmpty() && password.isNotEmpty()) {
    89             register(email, password)
    90         } else {
    91             showToast(getString(R.string.por_favor_completa_todos_los_campos))
    92         }
    93     }
    94
    95     private fun signIn(email: String, password: String) {
    96         auth.signInWithEmailAndPassword(email, password)
    97         .addOnCompleteListener(this) { task ->
    98             if (task.isSuccessful) {
    99                 val user = auth.currentUser
    100                 updateUI(user)
    101             } else {
    102                 Log.w( tag: "Login", msg: "signInWithEmail:failure", task.exception)
    103                 showToast("Error: ${task.exception?.message}")
    104                 updateUI( user: null)
    105             }
    106         }
    107     }
    108
    109     private fun register(email: String, password: String) {
    110         auth.createUserWithEmailAndPassword(email, password)
    111         .addOnCompleteListener(this) { task ->
    112             if (task.isSuccessful) {

```

```

class MainActivity : AppCompatActivity() {
    27
    107
    108
    109     private fun register(email: String, password: String) {
    110         auth.createUserWithEmailAndPassword(email, password)
    111         .addOnCompleteListener(this) { task ->
    112             if (task.isSuccessful) {
    113                 val user = auth.currentUser
    114                 showToast(getString(R.string.usuario_registrado_exitosamente))
    115                 updateUI(user)
    116             } else {
    117                 val errorMessage = task.exception?.message
    118                 ?: getString(R.string.error_desconocido_intenta_nuevamente)
    119                 Log.w( tag: "Register", msg: "createUserWithEmail:failure", task.exception)
    120                 showToast(getString(R.string.error_al_registrar) + errorMessage)
    121                 updateUI( user: null)
    122             }
    123         }
    124     }
    125
    126     private fun signInWithGoogle() {
    127         oneTapClient.beginSignIn(signInRequest)
    128         .addOnSuccessListener(this) { result ->
    129             try {
    130                 signInLauncher.launch(IntentSenderRequest.Builder(result.pendingIntent).build())
    131             } catch (e: android.content.IntentSender.SendIntentException) {
    132                 Log.e( tag: "GoogleSignIn", msg: "Couldn't start One Tap UI:" + e.localizedMessage)
    133                 val signInIntent = googleSignInClient.signInIntent
    134                 startActivityForResult(signInIntent, RC_SIGN_IN)
    135             }
    136         }
    137     }
    138
    139     1

```

```
class MainActivity : AppCompatActivity() {

    private fun signInWithTwitter() {
        val provider = OAuthProvider.newBuilder("twitter.com")
        auth.startActivityForResultSignInWithProvider(this, provider.build())
        .addOnCompleteListener { task ->
            if (task.isSuccessful) {
                val user = auth.currentUser
                updateUI(user)
            } else {
                Log.e(tag: "TwitterSignIn", msg: "Error al autenticar con Twitter", task.exception)
                showToast(getString(R.string.error_al_autenticar_con_twitter))
            }
        }
    }

    private val signInLauncher =
        registerForActivityResult(ActivityResultContracts.StartIntentSenderForResult()) { result ->
            if (result.resultCode == RESULT_OK) {
                val credential = oneTapClient.getCredentialFromIntent(result.data)
                val idToken = credential.googleIdToken
                if (idToken != null) {
                    val firebaseCredential = GoogleAuthProvider.getCredential(idToken, null)
                    auth.signInWithCredential(firebaseCredential)
                    .addOnCompleteListener(this) { task ->
                        if (task.isSuccessful) {
                            val user = auth.currentUser
                            updateUI(user)
                        } else {
                            showToast(getString(R.string.error_al_autenticar_con_google))
                        }
                    }
                }
            }
        }
    }
```

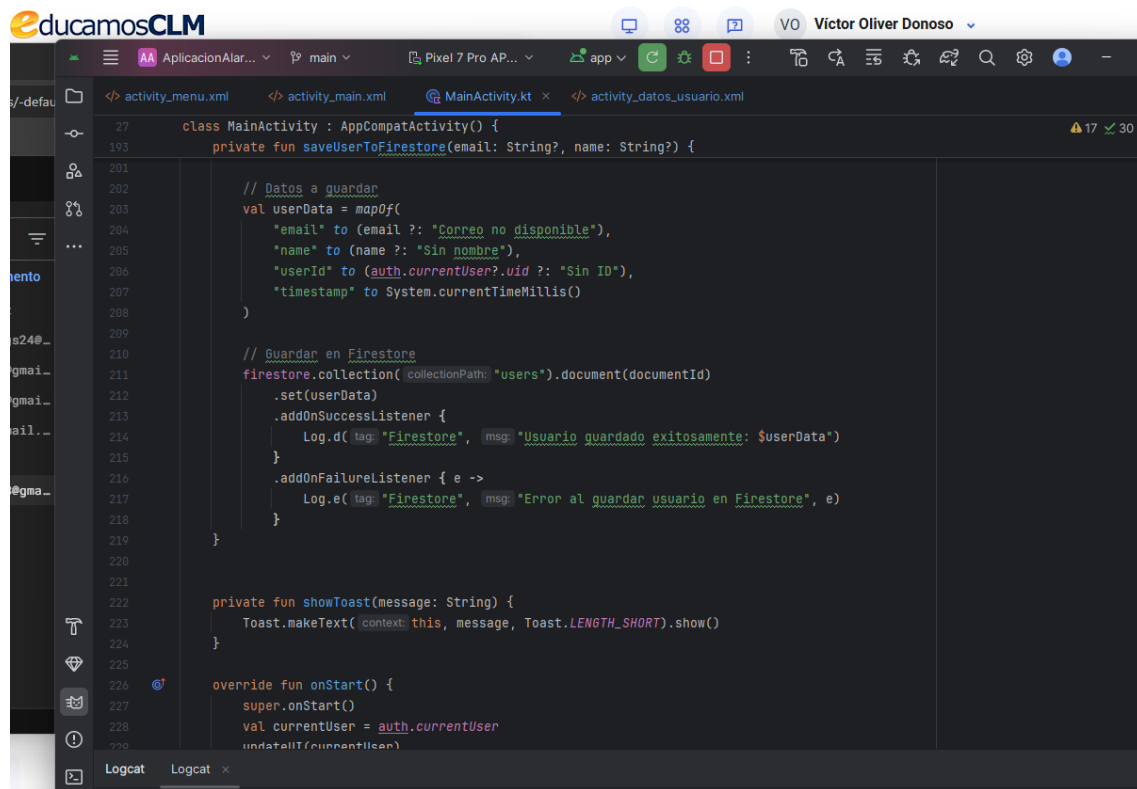
```
class MainActivity : AppCompatActivity() {

    private fun updateUI(user: FirebaseUser?) {
        if (user != null) {
            val userEmail = user.email
            val userName = user.displayName ?: "Usuario sin nombre"
            saveUserToFirestore(userEmail, userName)

            showToast(getString(R.string.bienvenido, userName))
            val intent = Intent(packageContext, this, MenuActivity::class.java)
            startActivity(intent)
            finish()
        } else {
            showToast(getString(R.string.por_favor_inicia_sesi_n_o_req_strate))
        }
    }

    private fun saveUserToFirestore(email: String?, name: String?) {
        // Determinar el identificador del documento
        val documentId = when {
            !email.isNullOrEmpty() -> email.replace(oldValue: " ", newValue: "_") // Reemplazar espacios por _
            !name.isNullOrEmpty() -> name.replace(oldValue: " ", newValue: "_") // Reemplazar espacios por _
            auth.currentUser?.uid != null -> auth.currentUser!!.uid
            else -> "UnknownUser_${System.currentTimeMillis()}"
        }

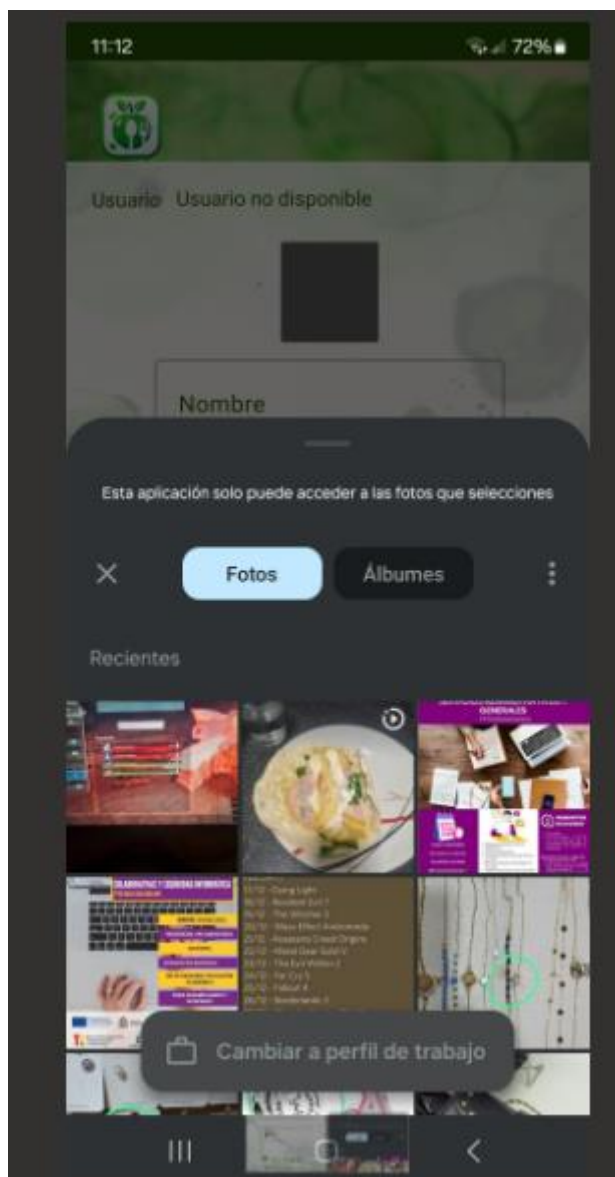
        // Datos a guardar
        val userData = mapOf(
            "email" to (email ?: "Correo no disponible"),
            "name" to (name ?: "Sin nombre")
        )
    }
}
```



```
27 class MainActivity : AppCompatActivity() {
193 private fun saveUserToFirestore(email: String?, name: String?) {
201
202     // Datos a guardar
203     val userData = mapOf(
204         "email" to (email ?: "Correo no disponible"),
205         "name" to (name ?: "Sin nombre"),
206         "userId" to (auth.currentUser?.uid ?: "Sin ID"),
207         "timestamp" to System.currentTimeMillis()
208     )
209
210     // Guardar en Firestore
211     firestore.collection( collectionPath: "users").document(documentId)
212         .set(userData)
213         .addOnSuccessListener {
214             Log.d( tag: "Firestore", msg: "Usuario guardado exitosamente: $userData")
215         }
216         .addOnFailureListener { e ->
217             Log.e( tag: "Firestore", msg: "Error al guardar usuario en Firestore", e)
218         }
219 }
220
221
222 private fun showToast(message: String) {
223     Toast.makeText( context: this, message, Toast.LENGTH_SHORT).show()
224 }
225
226 @
227 override fun onStart() {
228     super.onStart()
229     val currentUser = auth.currentUser
230     updateUI(currentUser)
231 }
232 }
```

Logcat Logcat x

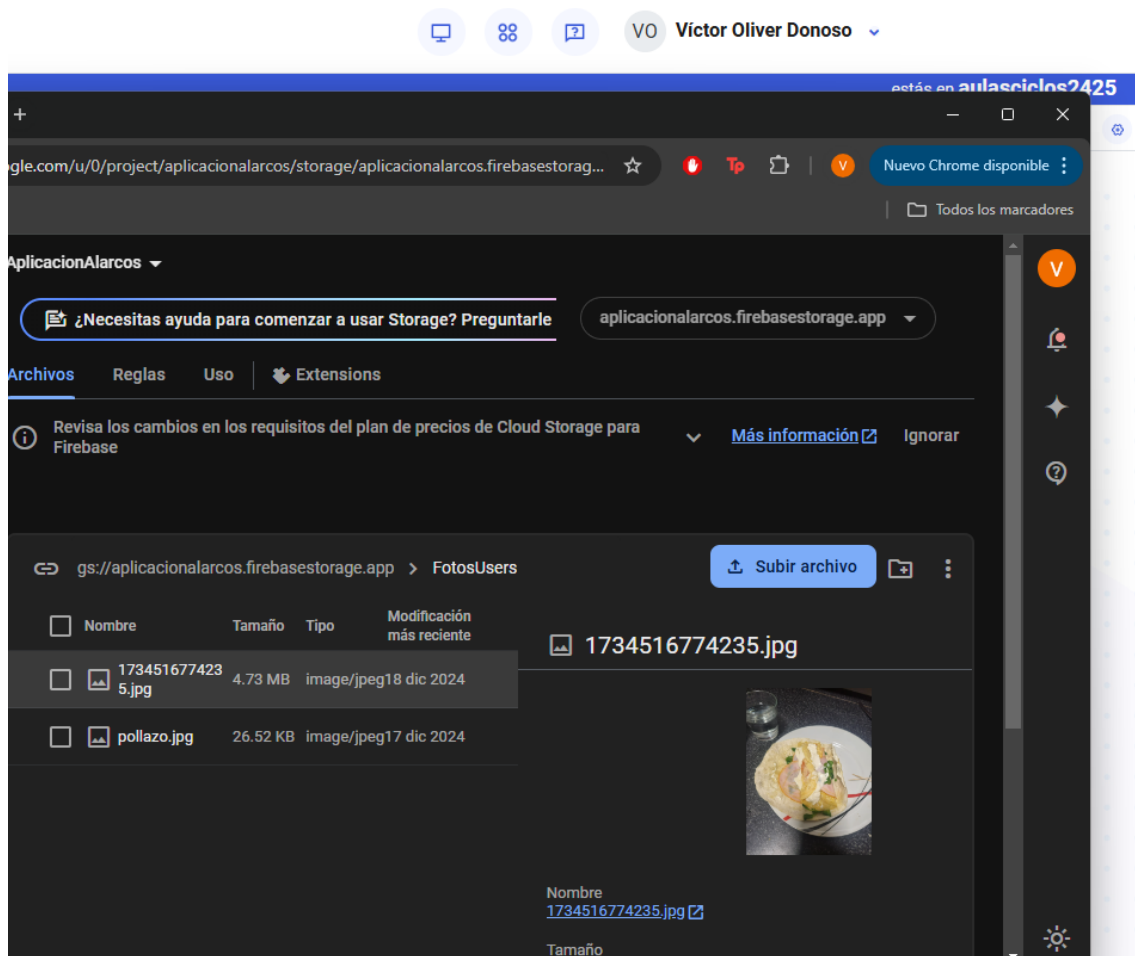
Y aquí guardamos la imagen de perfil en el storage(lo hemos hecho con nuestro móvil, en este caso el de victor, haciendo capturas, por que no teníamos fotos en el emulador)



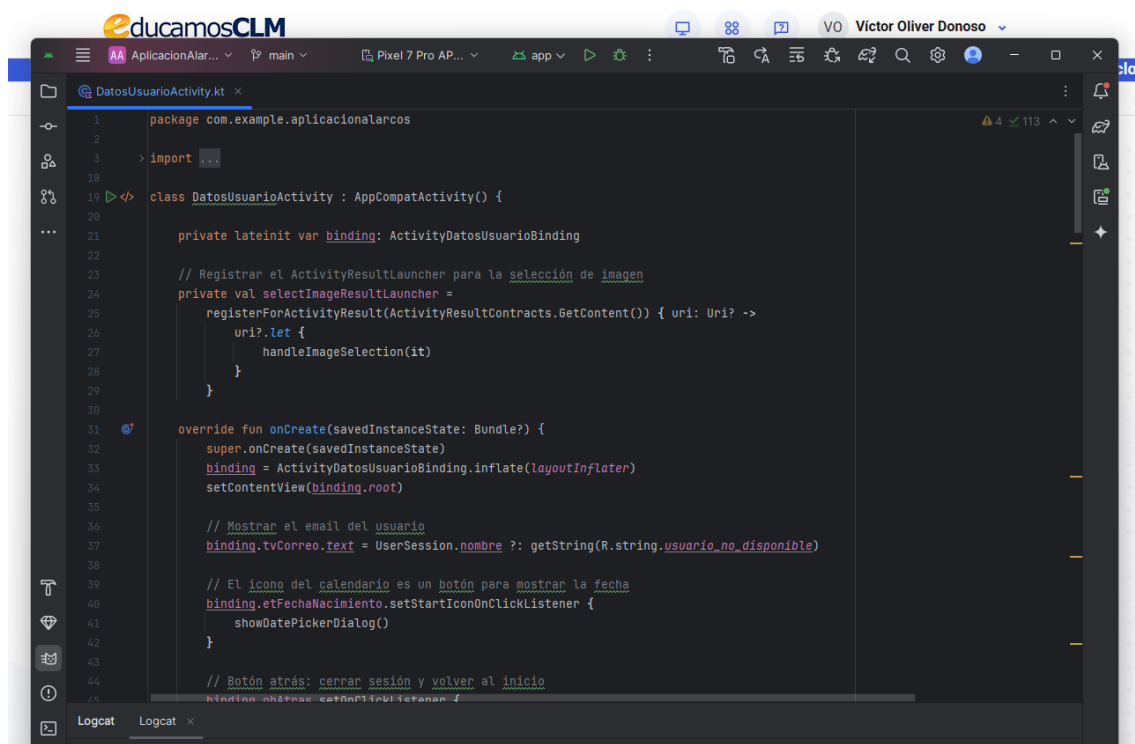
Hay otra captura más abajo

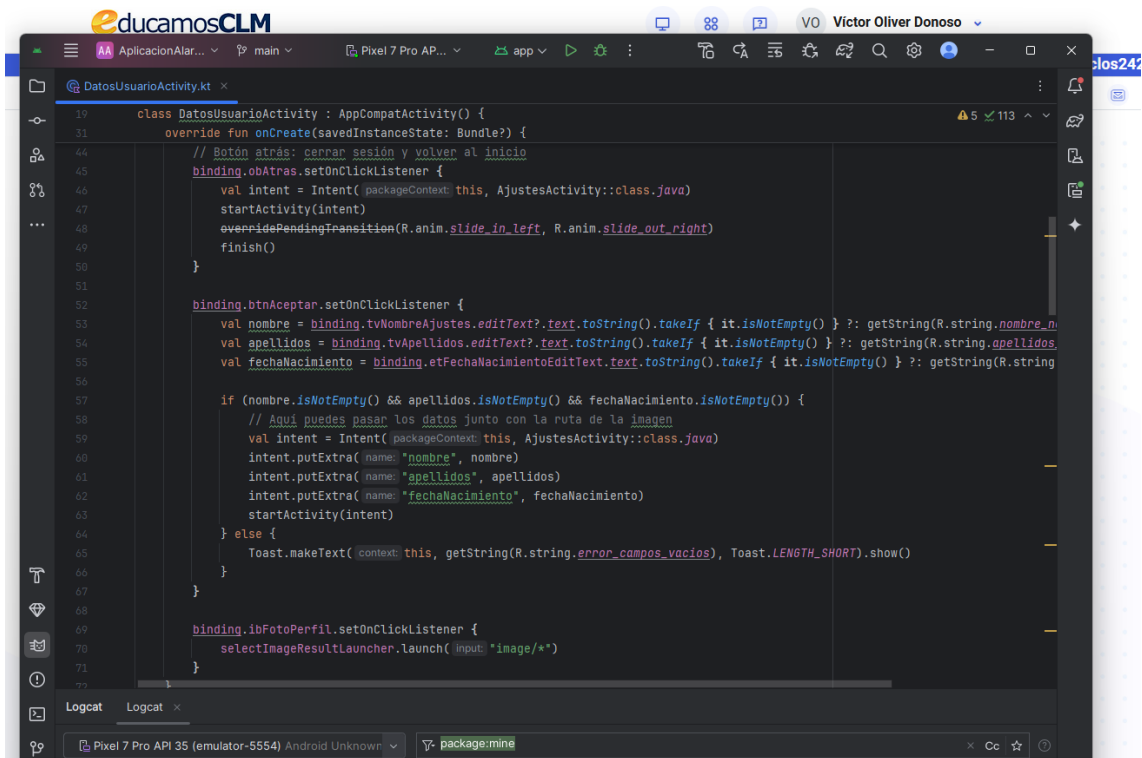


Hay otra captura más abajo

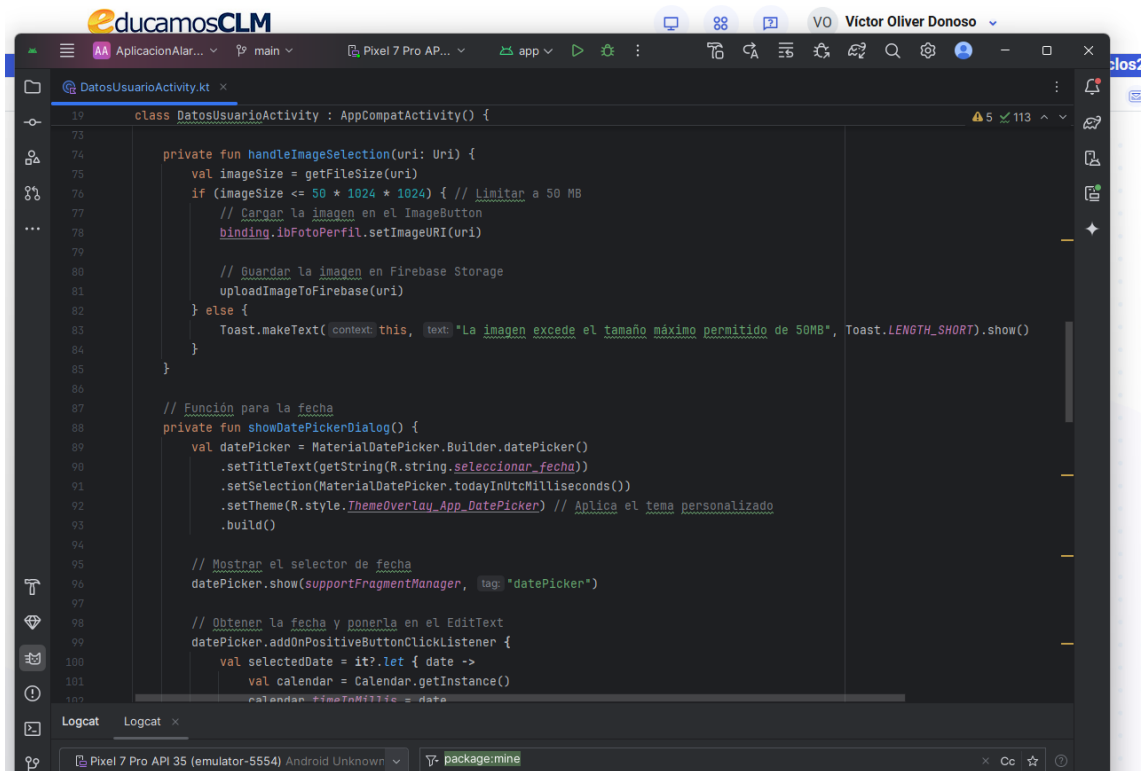


Logica del storage

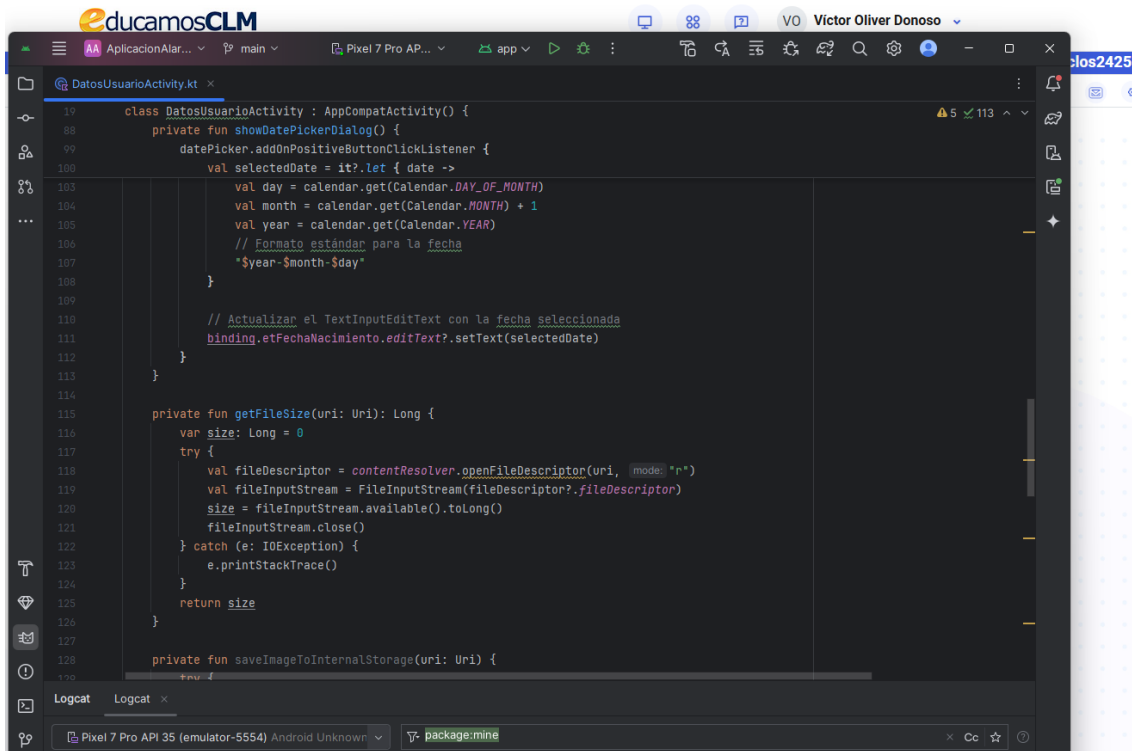




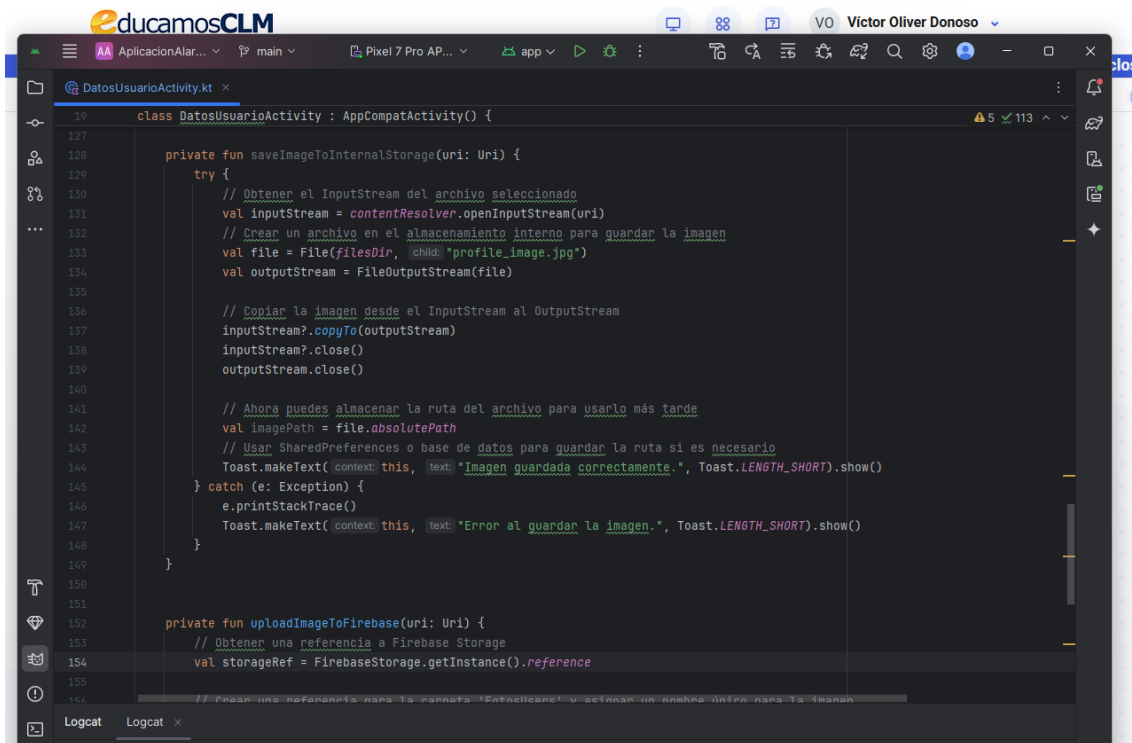
```
19 class DatosUsuarioActivity : AppCompatActivity() {
31     override fun onCreate(savedInstanceState: Bundle?) {
44         // Botón atrás: cerrar sesión y volver al inicio
45         binding.obAtras.setOnClickListener {
46             val intent = Intent(packageContext, this, AjustesActivity::class.java)
47             startActivity(intent)
48             overridePendingTransition(R.anim.slide_in_left, R.anim.slide_out_right)
49             finish()
50         }
51
52         binding.btnAceptar.setOnClickListener {
53             val nombre = binding.tvNombreAjustes.editText?.text.toString().takeIf { it.isNotEmpty() }?: getString(R.string.nombre_n
54             val apellidos = binding.tvApellidos.editText?.text.toString().takeIf { it.isNotEmpty() }?: getString(R.string.apellidos
55             val fechaNacimiento = binding.etFechaNacimiento.editText?.text.toString().takeIf { it.isNotEmpty() }?: getString(R.string
56
57             if (nombre.isNotEmpty() && apellidos.isNotEmpty() && fechaNacimiento.isNotEmpty()) {
58                 // Aquí puedes pasar los datos junto con la ruta de la imagen
59                 val intent = Intent(packageContext, this, AjustesActivity::class.java)
60                 intent.putExtra(name="nombre", nombre)
61                 intent.putExtra(name="apellidos", apellidos)
62                 intent.putExtra(name="fechaNacimiento", fechaNacimiento)
63                 startActivity(intent)
64             } else {
65                 Toast.makeText(context, this, getString(R.string.error_campos_vacios), Toast.LENGTH_SHORT).show()
66             }
67
68
69             binding.ibFotoPerfil.setOnClickListener {
70                 selectImageResultLauncher.launch(input="image/*")
71             }
72         }
73     }
74 }
```



```
73
74 private fun handleImageSelection(uri: Uri) {
75     val imageSize = getFileSize(uri)
76     if (imageSize <= 50 * 1024 * 1024) { // Limitar a 50 MB
77         // Cargar la imagen en el ImageButton
78         binding.ibFotoPerfil.setImageURI(uri)
79
80         // Guardar la imagen en Firebase Storage
81         uploadImageToFirebase(uri)
82     } else {
83         Toast.makeText(context, this, text="La imagen excede el tamaño máximo permitido de 50MB", Toast.LENGTH_SHORT).show()
84     }
85 }
86
87 // Función para la fecha
88 private fun showDatePickerDialog() {
89     val datePicker = MaterialDatePicker.Builder.datePicker()
90     .setTitleText(getString(R.string.seleccionar_fecha))
91     .setSelection(MaterialDatePicker.todayInUtcMilliseconds())
92     .setTheme(R.style.ThemeOverlay_App_DatePicker) // Aplica el tema personalizado
93     .build()
94
95     // Mostrar el selector de fecha
96     datePicker.show(supportFragmentManager, tag="datePicker")
97
98     // Obtener la fecha y ponerla en el EditText
99     datePicker.addOnPositiveButtonClickListener {
100         val selectedDate = it?.let { date ->
101             val calendar = Calendar.getInstance()
102             calendar.timeInMillis = date
103         }
104     }
105 }
```



```
19 class DatosUsuarioActivity : AppCompatActivity() {
88     private fun showDatePickerDialog() {
89         datePicker.addOnPositiveButtonClickListener {
90             val selectedDate = it?.let { date ->
103                 val day = calendar.get(Calendar.DAY_OF_MONTH)
104                 val month = calendar.get(Calendar.MONTH) + 1
105                 val year = calendar.get(Calendar.YEAR)
106                 // Formato estándar para la fecha
107                 "$year-$month-$day"
108             }
109
110             // Actualizar el TextInputEditText con la fecha seleccionada
111             binding.etFechaNacimiento.editText?.setText(selectedDate)
112         }
113     }
114
115     private fun getFileSize(uri: Uri): Long {
116         var size: Long = 0
117         try {
118             val fileDescriptor = contentResolver.openFileDescriptor(uri, mode: "r")
119             val fileInputStream = FileInputStream(fileDescriptor?.fileDescriptor)
120             size = fileInputStream.available().toLong()
121             fileInputStream.close()
122         } catch (e: IOException) {
123             e.printStackTrace()
124         }
125         return size
126     }
127
128     private fun saveImageToInternalStorage(uri: Uri) {
129         +pu 4
130     }
131 }
```



```
127 class DatosUsuarioActivity : AppCompatActivity() {
128     private fun saveImageToInternalStorage(uri: Uri) {
129         try {
130             // Obtener el InputStream del archivo seleccionado
131             val inputStream = contentResolver.openInputStream(uri)
132             // Crear un archivo en el almacenamiento interno para guardar la imagen
133             val file = File(filesDir, "profile_image.jpg")
134             val outputStream = FileOutputStream(file)
135
136             // Copiar la imagen desde el InputStream al OutputStream
137             inputStream?.copyTo(outputStream)
138             inputStream?.close()
139             outputStream.close()
140
141             // Ahora puedes almacenar la ruta del archivo para usarlo más tarde
142             val imagePath = file.absolutePath
143             // Usar SharedPreferences o base de datos para guardar la ruta si es necesario
144             Toast.makeText(context, this, "Imagen guardada correctamente.", Toast.LENGTH_SHORT).show()
145         } catch (e: Exception) {
146             e.printStackTrace()
147             Toast.makeText(context, this, "Error al guardar la imagen.", Toast.LENGTH_SHORT).show()
148         }
149     }
150
151     private fun uploadImageToFirebase(uri: Uri) {
152         // Obtener una referencia a Firebase Storage
153         val storageRef = FirebaseStorage.getInstance().reference
154         // Crear una referencia para la imagen "profile_image.jpg" y guardar la ruta de la imagen
155     }
156 }
```

educamosCLM

VO Victor Oliver Donoso

AplicacionAlar... main... Pixel 7 Pro AP... app

DatosUsuarioActivity.kt

```
149 class DatosUsuarioActivity : AppCompatActivity() {
150
151     private fun uploadImageToFirebase(uri: Uri) {
152         // Obtener una referencia a Firebase Storage
153         val storageRef = FirebaseStorage.getInstance().reference
154
155         // Crear una referencia para la carpeta 'FotosUsers' y asignar un nombre único para la imagen
156         val imagesRef = storageRef.child(pathString: "FotosUsers/${System.currentTimeMillis()}.jpg")
157
158         // Subir la imagen
159         imagesRef.putFile(uri)
160
161         .addOnSuccessListener {
162             // Subida exitosa
163             Toast.makeText(context: this, text: "Imagen subida exitosamente", Toast.LENGTH_SHORT).show()
164
165             // Obtener la URL de la imagen subida
166             imagesRef.downloadUrl.addOnSuccessListener { downloadUri ->
167                 // Esta es la URL de la imagen subida en Firebase Storage
168                 val imageUrl = downloadUri.toString()
169                 // Puedes guardar la URL donde la necesites, por ejemplo en la base de datos o en SharedPreferences
170             }
171         }
172         .addOnFailureListener { e ->
173             // Error al subir la imagen
174             Toast.makeText(context: this, text: "Error al subir la imagen: ${e.message}", Toast.LENGTH_SHORT).show()
175         }
176     }
177
178 }
179
```

Logcat Logcat