# Sistema de alertas PcComponentes II
# PcComponentes Alert System II

Por
Sergio Díaz Renedo


UNIVERSIDAD COMPLUTENSE
MADRID

Grado en Ingeniería del Software
Facultad de Informática

Mercedes García Merayo

Madrid, 2021–2022

# Agradecimientos

*A la Universidad Complutense de Madrid y a la Facultad de Informática por su plan de estudios, profesores y las instalaciones que nos han proporcionado para contribuir a nuestro desarrollo personal y profesional durante el Grado.*

*Y, en particular, a Mercedes por darme la oportunidad de trabajar en este proyecto y por su implicación en el mismo. Muchas gracias.*

**Sergio Díaz**

# Índice general

**Página**

# Resumen

Este documento tiene como objetivo informar del desarrollo del proyecto que ha dado lugar al sistema de alertas de PcComponentes [14] II.

El propósito de este trabajo es ampliar las funcionalidades del proyecto original, ofreciendo al usuario la posibilidad de registrarse con proveedores externos como Google y Facebook y tener un mayor control a la hora de realizar filtros sobre los componentes en venta de la web. Dichas mejoras, unidas a cambios en la interfaz, buscan mejorar la usabilidad de la aplicación y la retención de usuarios.

PCComponentes es una tienda de informática, electrónica y telefonía constituida en el año 2005, con dos tiendas físicas situadas en Murcia y Madrid y cuyo negocio es mayoritariamente online. Debido a su posición como líder del sector en España, al gran volumen de diferentes artículos a la venta y a las habituales rebajadas de los mismos, es interesante disponer de un sistema de alertas que nos avise cuando un producto baje de precio a una cantidad máxima que especifiquemos, característica de la que no dispone la propia web.

Por ello, este trabajo de fin de grado consistirá en la ampliación de funcionalidades de la aplicación realizada en el curso 2019-2020.

# Abstract

The intention of this document is to inform on the development of PCComponentes II Alert System.

The purpose of this work is to expand the functionalities of the original project, offering the user the possibility of registering in the application with external providers, such as Google and Facebook, and giving more control filtering the components for sale on the web. These improvements, together with changes in the interface, seek to improve the usability of the application and user retention.

PCComponentes is a computer, electronics and telephony store established in 2005, with two physical stores located in Murcia and Madrid, and whose business is mostly online. Due to its position as the leader in these sectors in Spain, it's interesting to have an alert system that notifies us when prices are below a certaint quantity, a feature that the website itself does not have.

For this reason, this project consists of expanding the functionalities of the application developed on 2019-2020.

# Capítulo 1

# Introducción

Sistema de Alertas PCComponentes fue un trabajo de fin de grado realizado durante el curso 2019-2020 que tenía como objetivo la extracción de datos de la web de PCComponentes, concretamente la de sus artículos a la venta, por medio de web scrapping. Este web scrapping se implementó en Python y se lanzaba diariamente desde una máquina de Amazon AWS.

El almacenamiento de estos datos se guardaba en la plataforma de Google Firebase, cuyas funcionalidades explicaremos detalladamente en este documento. La base de datos de Firebase es NoSQL, por lo que existe una limitación a la hora de realizar consultas complejas de cruce de datos que no existiría en caso de almacenarse en una base de datos SQL.

La aplicación fue realizada en Android, centrándose en una usabilidad sencilla y de poca carga para los dispositivos, que permitía el registro mediante correo electrónico y contraseña, el filtrado de artículos mediante el nombre, el guardado de artículos en una lista de seguimiento, la elección de un precio máximo esperado de cada artículo para recibir notificaciones de ofertas en caso de bajar a dicho umbral, y un sistema de notificaciones que incluía el envío de correos electrónicos y de notificaciones push desde el propio dispositivo.

En este trabajo de fin de grado se ampliarán las funcionalidades de la aplicación, manteniendo el sistema de alertas y de seguimiento inmutable.

## 1.1. Objetivos

El objetivo de este proyecto es ampliar las funcionalidades originales de la aplicación "Sistema de alertas PCComponentes" [16], permitiendo el login directo mediante proveedores externos como pueden ser Google y Facebook y aumentando las posibilidades a la hora de recorrer y filtrar el listado de componentes de la base de datos, implementado

el filtrado por tipo de componente además de por su nombre. También se mejorará la interfaz gráfica para hacer más sencilla la navegación por la aplicación.

# Capítulo 2

# Análisis de requisitos

Se detallará a continuación los requisitos necesarios para el funcionamiento del proyecto:

## 2.1. Requisitos

### 2.1.1. Registro mediante correo electrónico y contraseña

- Se requiere conexión a Internet
- Se debe introducir un correo electrónico con un formato válido. Si el correo electrónico no existe no se podrá confirmar el registro
- El correo electrónico no puede coincidir con el de un usuario actualmente registrado, independientemente del proveedor
- La contraseña debe tener al menos 6 caracteres

### 2.1.2. Registro usando una cuenta de Google

- Se requiere conexión a Internet
- Se debe dar permisos de lectura del nombre, email y foto de perfil a la aplicación

### 2.1.3. Registro usando una cuenta de Facebook

- Se requiere conexión a Internet
- Dependiendo de la versión de móvil es posible que sea necesario tener la aplicación de Facebook instalada
- Se debe dar permisos de lectura del nombre, email y foto de perfil a la aplicación

### 2.1.4. Login

- Se requiere conexión a Internet
- Si el login se realiza introduciendo un correo electrónico y contraseña, la cuenta debe haber sido confirmada previamente por el usuario desde el email enviado automáticamente en el momento del registro
- Si el login se realiza mediante Google o Facebook no se requieren requisitos adicionales a los del registro

### 2.1.5. Aplicación

- Se requiere conexión a Internet
- El dispositivo debe ser compatible con el sistema operativo Android [17]
- Los productos seguidos por el usuario serán almacenados en un listado personal en la base de datos
- El usuario podrá modificar el precio de aviso para un producto seguido o dejar de seguirlo
- El buscador permitirá al usuario filtrar los productos por precio y alfabéticamente, de mayor a menor y de menor a mayor en ambos casos, además de filtrar por un nombre o categoría del artículo introduciendo un texto en el buscador de la aplicación
- El usuario podrá borrar su cuenta y datos asociados permanentemente

# Capítulo 3

# Entornos de Desarrollo

En este apartado se detallarán las aplicaciones, plataformas y lenguajes de programación utilizados para el desarrollo del proyecto:

## 3.1. Aplicaciones y plataformas

En este punto se detallarán qué aplicación y plataformas se han usado para el desarrollo e implementación del proyecto:

### 3.1.1. Android Studio

Android Studio [1] es el entorno oficial de desarrollo integrado de aplicaciones para Android. Anunciado por Google en 2013, su primera versión estable se lanzó en diciembre de 2014, reemplazando a Eclipse como el IDE oficial de desarrollo de aplicaciones Android. Soporta diversos tipos de lenguajes, siendo Kotlin [10] y Java [7] las preferencias en el desarrollo de apps.

### 3.1.2. Firebase

Firebase [3] es una plataforma para agilizar y simplificar el desarrollo de aplicaciones. Lanzada en 2011 y adquirida por Google en 2014, ha ido adquiriendo popularidad a la vez que añadía nuevas funcionalidades, convirtiéndose en la plataforma más importante para el desarrollo de aplicaciones móviles y web, soportando plataformas como Android, iOS [6] y Unity [19], entre otras.

### 3.1.3. Meta Platforms

Meta Platforms [12] es la empresa matriz de Facebook, Instagram, Whatsapp y otras subsidiarias, que ofrece un servicio de integración de aplicaciones móviles a sus servicios mediante Meta for Developers [11].

## 3.2. Lenguajes de Programación

Para el desarrollo del proyecto se han usado dos lenguajes de programación:

### 3.2.1. Kotlin

Kotlin es un lenguaje de programación orientado a objetos de código abierto. Fue desarrollado por JetBrains en 2010, anunciado en 2011 y liberado en 2012 bajo una licencia Apache 2. En 2017 fue nombrado por Google como el lenguaje oficial para el desarrollo en Android [9], compartiendo reconocimiento con el lenguaje Java, siendo estos dos lenguajes interoperables entre ellos, lo que permite tener una aplicación con parte de código en Java y parte en Kotlin, realizando llamadas entre ellos.

### 3.2.2. Java

Java es un lenguaje de programación creado en 1991 por Sun Microsystems y comercializado en 1995. Desde entonces se ha convertido en uno de los lenguajes de programación más importantes, siendo el lenguaje líder en la programación orientada a objetos y uno de los dos lenguajes oficiales para el desarrollo de aplicaciones Android.

## 3.3. Base de Datos

Se hace uso de la base de datos de Firebase, denominada Firebase Realtime Database [4]. Es una base de datos NoSQL alojada en la nube e integrada en la plataforma Firebase, que almacena sus datos en formato JSON [8]. Permite acceder a los datos tanto con conexión como sin ella, ya que permite la persistencia de los datos en disco y su sincronización automática en caso de existir conexión a Internet.

Los datos se almacenan en colecciones y documentos. Una colección, por ejemplo, podrían ser los 'Procesadores' y los documentos serían cada modelo de procesador a la venta. A su vez, los documentos pueden tener colecciones y éstas más documentos asociados, a una profundidad máxima de 100 niveles.

La Figura 3.1 muestra una representación de la base de datos del proyecto:

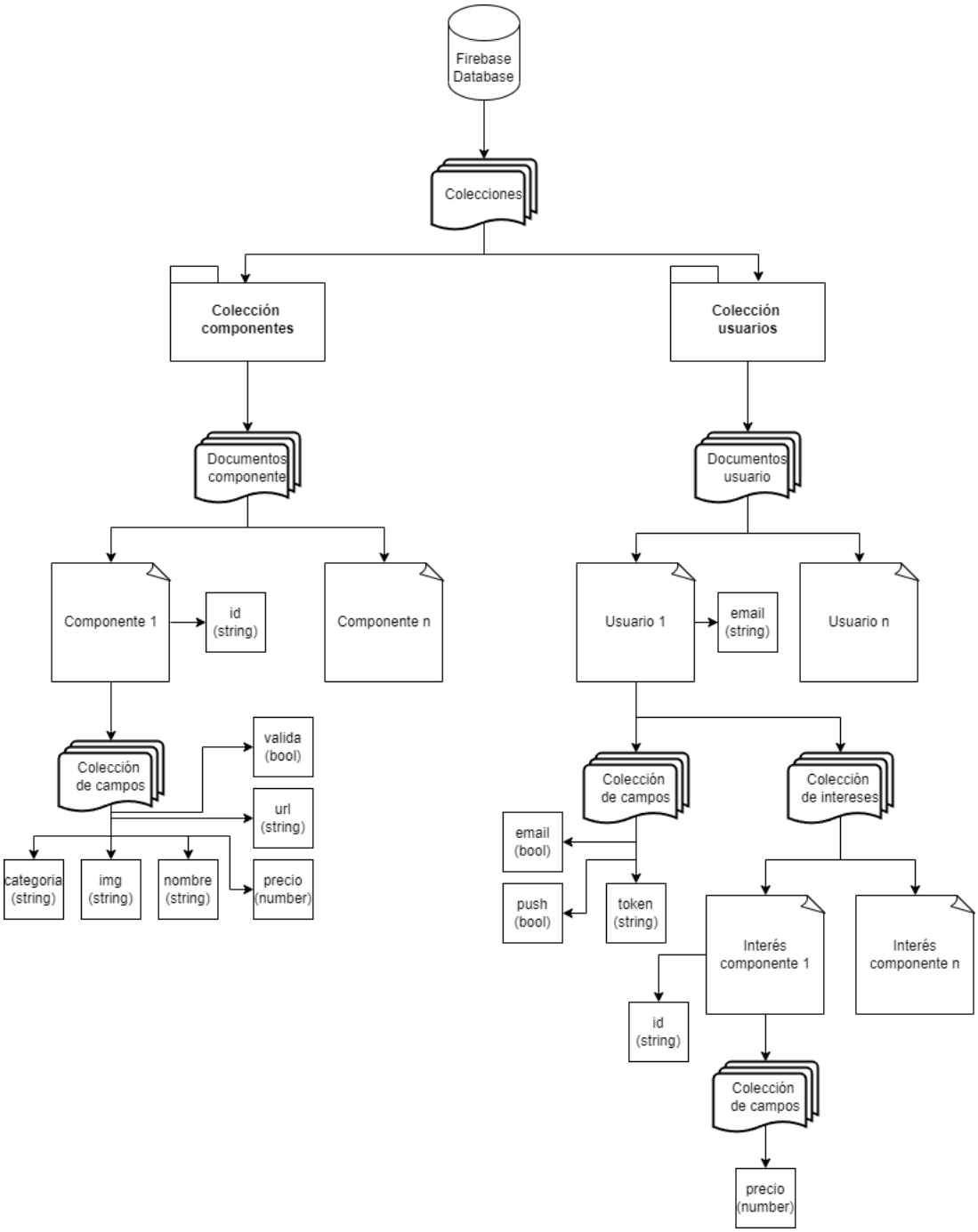



Figura 3.1: Base de datos Firebase

## 3.4. Control de Versiones

Se ha decidido hacer uso de la plataforma GitHub [5] para almacenar el código del proyecto. GitHub, que hace uso de la tecnología Git [18], permite almacenar diferentes versiones del código, comparar los cambios entre versiones y revertir a versiones anteriores en ca-

so de necesidad. Por ello, se han creado un repositorio público, como se observa en la Figura 3.2, que almacenará el código de la aplicación móvil.

- URL: https://github.com/Sergidia/PcComponentesTFGII

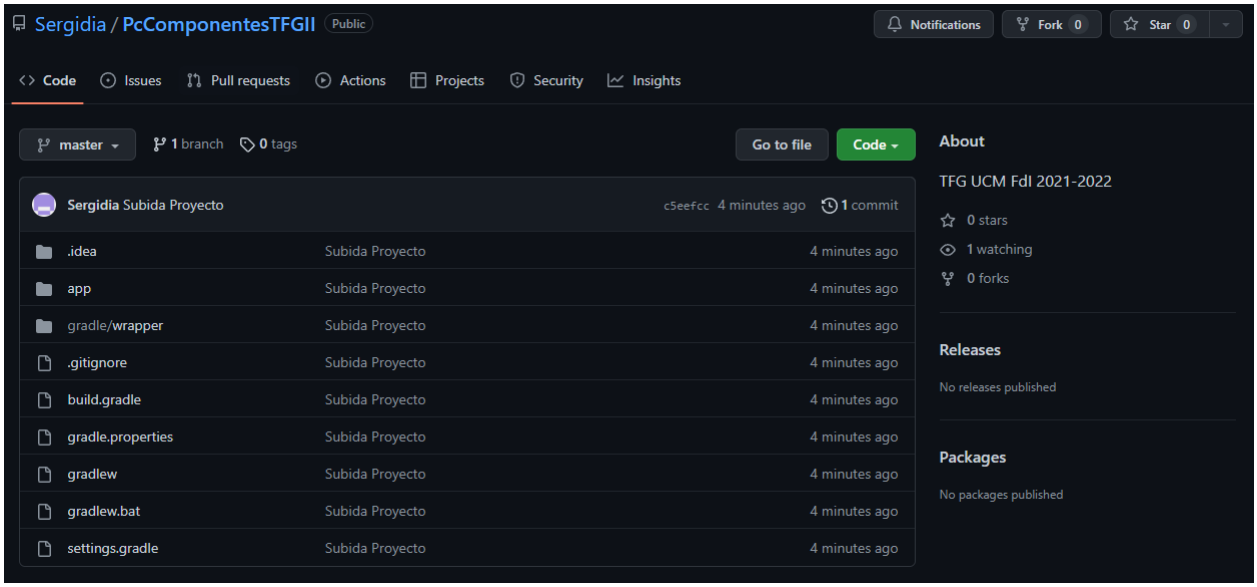



Figura 3.2: Repositorio del código de la aplicación

# Capítulo 4

# Diseño e Implementación

En este apartado se detallarán las diferentes fases de diseño e implementación del proyecto, a fin de comprender las decisiones tomadas durante el desarrollo y con el objetivo de que sirva como un breve manual para replicar su implementación:

## 4.1. Fase de análisis

Se plantearon propuestas de mejora de la aplicación a partir del listado de "Trabajo Futuro" del proyecto original, escogiendo las mejoras relacionadas con la funcionalidad de la aplicación, tales como aumentar los proveedores aceptados para el registro y las mejoras en la búsqueda de componentes.

## 4.2. Fase de investigación

Al tener conocimientos previos en desarrollo en Android Studio, con lenguajes como Java y Kotlin, además de haber desarrollado con Firebase, la investigación se centró en la integración de las autenticaciones mediante Google y Facebook en aplicaciones móviles, requiriendo esta última de una herramienta propia que debía ser configurada y vinculada con Firebase y la aplicación, mediante hashes de clave que explicaremos con detenimiento más adelante.

## 4.3. Fase de Instalación y Configuración

Se describirán los pasos realizados para la configuración de los entornos de desarrollo y la implementación del proyecto:

**Paso 1 - Instalación de Android Studio:** Para el desarrollo de la aplicación móvil se instaló el programa de código abierto Android Studio.

**Paso 2 - Descarga del proyecto original de GitHub:** Se partirá de la última versión subida de la defensa del proyecto de 2020.

**Paso 3 - Creación de un proyecto en Firebase:** Para mantener la estructura del proyecto original era necesario crear un proyecto en Firebase, que agiliza la gestión de autenticación de usuarios para distintos tipos de proveedores, permite el envío de notificaciones push a los usuarios que tengan configurada esa opción dentro de la aplicación y contiene una base de datos no relacional, donde cargaremos tanto la información de los componentes recuperados de PCComponentes como el listado de seguimiento de los usuarios de la aplicación.

Para ello accedemos a la web de Firebase[1] e iniciamos sesión con una cuenta de Google.

Una vez iniciada sesión creamos un proyecto haciendo click en "Crear proyecto", tal y como se aprecia en la Figura 4.1, y completamos el formulario.

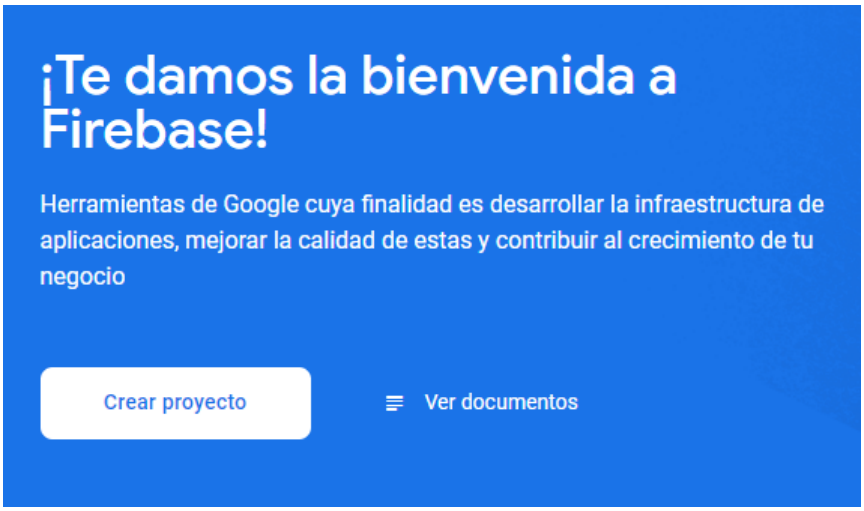



Figura 4.1: Creación del proyecto

Una vez creado el proyecto y accedido a su página principal (Figura 4.2), haremos click en "Agregar app"para vincular nuestra aplicación de Android Studio con Firebase, seleccionando la plataforma sobre la que vamos a desarrollar (Android, en este caso).


Figura 4.2: Vincular una aplicación

[1]https://console.firebase.google.com/

Completaremos los pasos solicitados y copiaremos un fichero JSON generado por Firebase en la raíz del proyecto, que nos servirá para conectar la aplicación con la base de datos.

Una vez conectado Firebase con la aplicación podremos hacer uso de las utilidades que ofrece, concretamente para este proyecto la utilidad "Authentication", que es una base de datos de usuarios, y "Firestore Database", que es la base de datos no relacional que utilizaremos para almacenar los componentes y el listado de seguimiento de los usuarios.

**Paso 4 - Configurar Firebase para permitir el inicio de sesión con Google:** Para ello deberemos ir al menú "Authentication", "Sign-in method" y activar el proveedor de inicio de sesión de Google, como se muestra en la Figura 4.3.

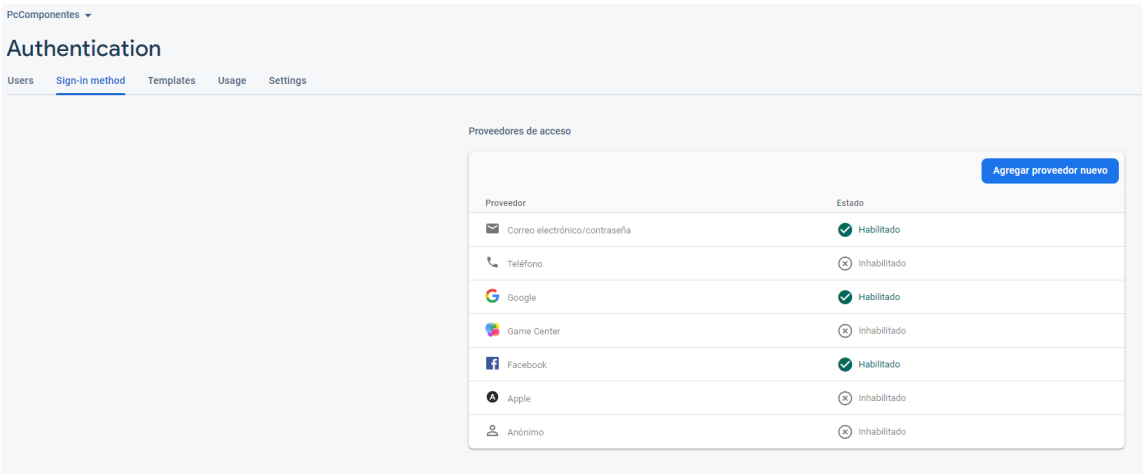



Figura 4.3: Proveedores de autenticación

Se nos pedirá agregar la huella digital del certificado SHA para permitir el login mediante Google, tal y como se muestra en la Figura 4.4. Esta huella digital se agregará en la ventana "Configuración del Proyecto" (Figura 4.5).

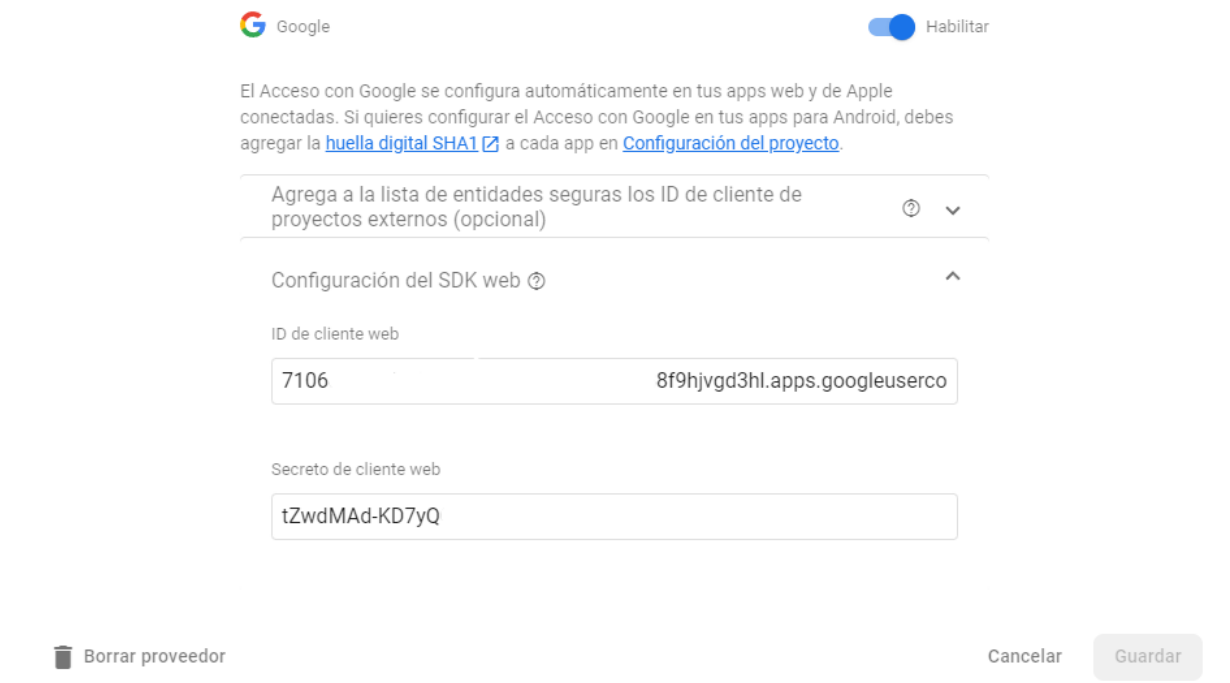



Figura 4.4: Configuración del SDK Web Google

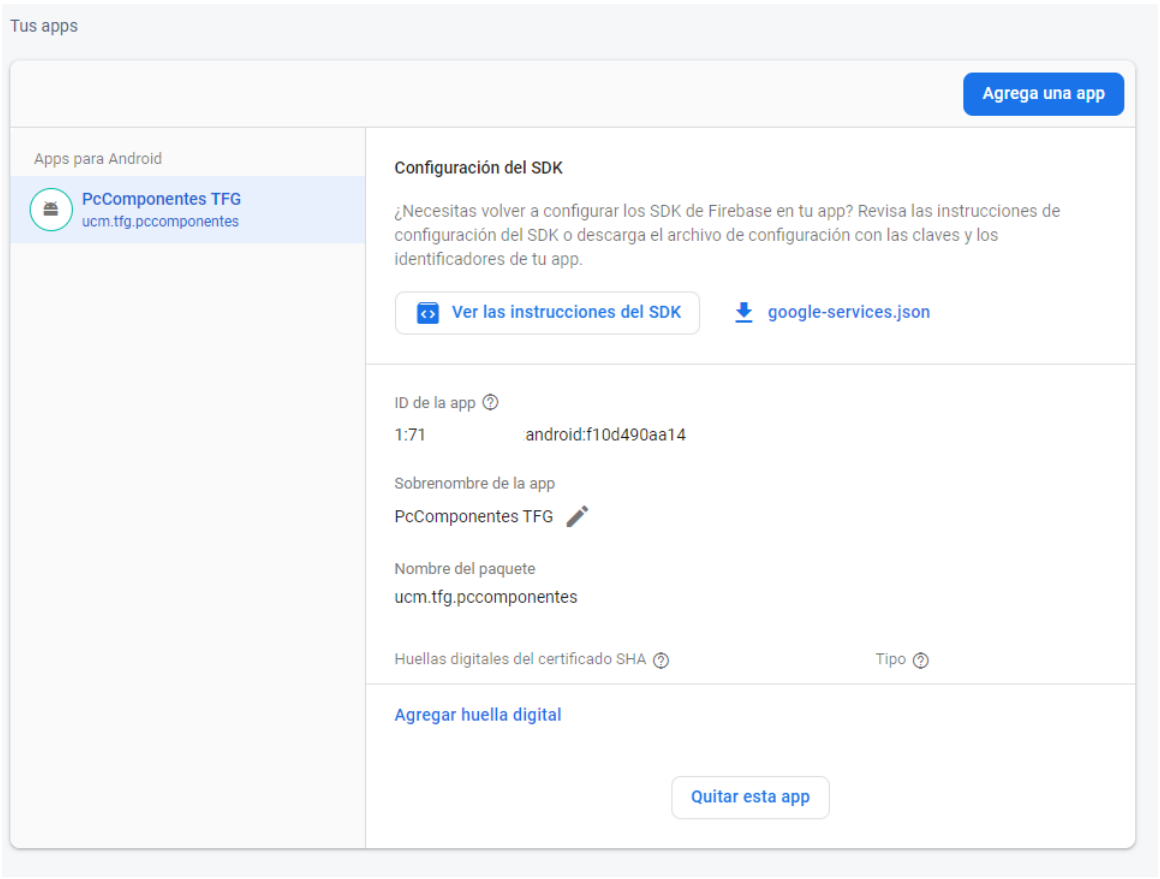



Figura 4.5: Configuración del Proyecto - Agregar huella digital

Para obtenerla, debemos ir a Android Studio, pestaña "Gradle", carpeta "Task" y abrir el fichero "sigingReport". Esto generará en la consola de Android Studio la información referente a la huella digital, como se observa en la Figura 4.6, copiaremos la clave SHA1 y la pegaremos en el formulario de Firebase (Figura 4.7):

```
Alias: AndroidDebugKey
MD5: E2:4B:90:F5:47:94:54
SHA1: B2:53:BB:2A:F4:0F:C8
SHA-256: 99:40:9C:E1:57
Valid until: jueves, 16 de noviembre de 2051
----------
```

Figura 4.6: Obtener huella digital SHA1

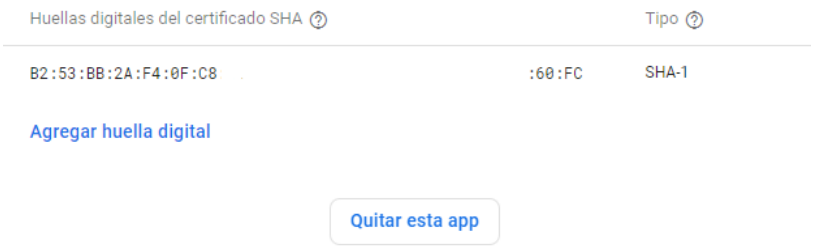



Figura 4.7: Huella digital agregada

Una vez realizada esta configuración podremos desarrollar la integración con "Google Authentication"

**Paso 5 - Instalación de OpenSSL [13] y JDK [2]:** Para hacer parte de la integración de Facebook con nuestra app, Meta for Developers requiere la introducción de Hashes de clave de Android del entorno de desarrollo sobre el que se va a ejecutar la aplicación. La instrucción que devuelve esta clave requiere de librerías contenidas en estos proyectos.

**Paso 6 - Creación del proyecto en Meta for Developers:** Para configurar los permisos a nivel de aplicación necesarios para la integración del login con Facebook, es necesario el registro en esta herramienta, que además nos permite tener dos entornos (desarrollo y producción) y crear usuarios ficticios de Facebook con los que realizar las pruebas.

Requerirá también de una configuración adicional en Firebase, como hicimos anteriormente para el proveedor de Google, que servirá para vincular el proyecto de Meta for Developers, Firebase y la aplicación.

Primero accederemos a la web de Meta for Developers[2], iniciaremos sesión con una cuenta de Facebook y navegaremos a la pestaña "Mis Aplicaciones", donde pulsaremos "Crear aplicación"(Figura 4.8) y rellenaremos un formulario con los datos del mismo:

[2]https://developers.facebook.com/

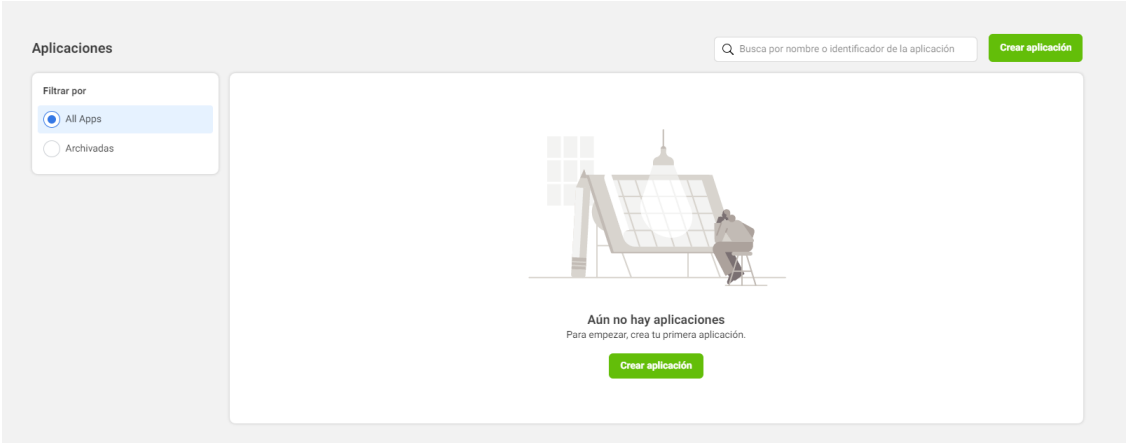



Figura 4.8: Crear proyecto

Una vez finalizada la creación del proyecto accederemos a la ventana mostrada en la Figura 4.9:

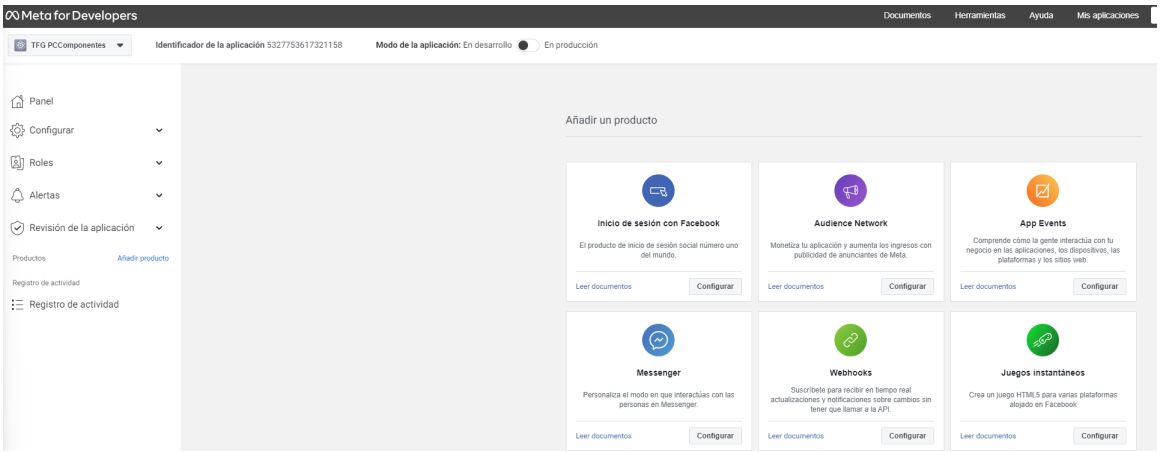



Figura 4.9: Proyecto creado

Donde pulsaremos en “Configurar” y después en “Información básica”, que abrirá la siguiente vista de la Figura 4.10:

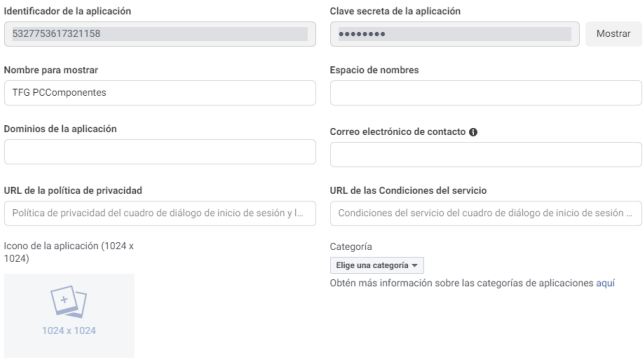



Figura 4.10: Datos de la aplicación

En esta ventana podremos obtener los datos del "Identificador de la aplicación" y la 'Clave secreta de la aplicación", que solicitará Firebase para activar el proveedor de autenticación de Facebook, tal y como se muestra en la Figura 4.11:

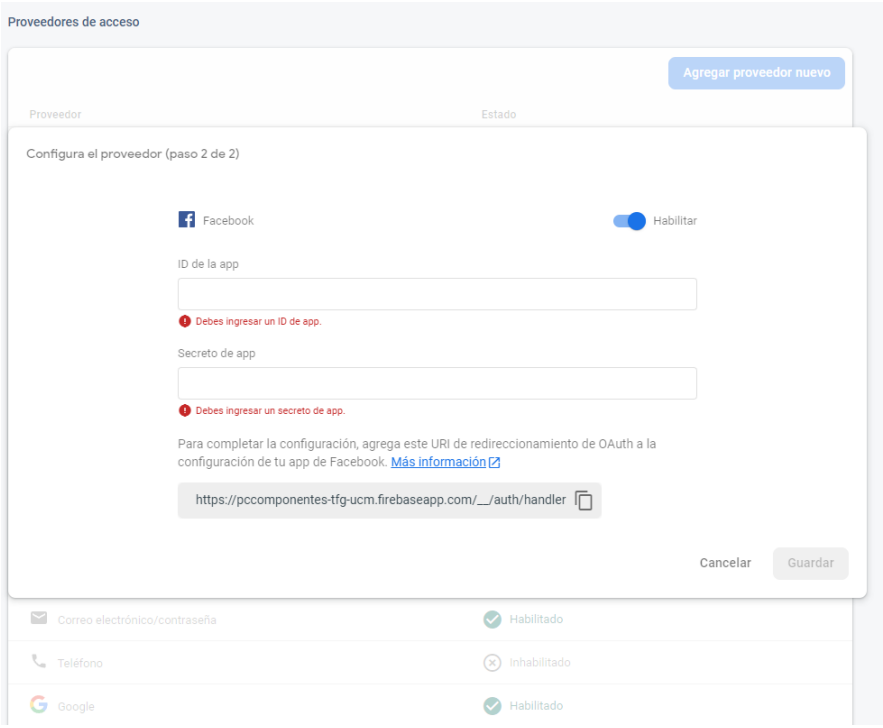



Figura 4.11: Configuración en Firebase del proveedor Facebook

Una vez introducidos esos dos datos en Firebase, volveremos a 'Meta for Developers" y haremos click en el botón "Añadir un producto", seleccionando la opción 'Inicio de sesión con Facebook". Elegiremos 'Android" y se abrirá el siguiente formulario mostrado en la Figura 4.12:

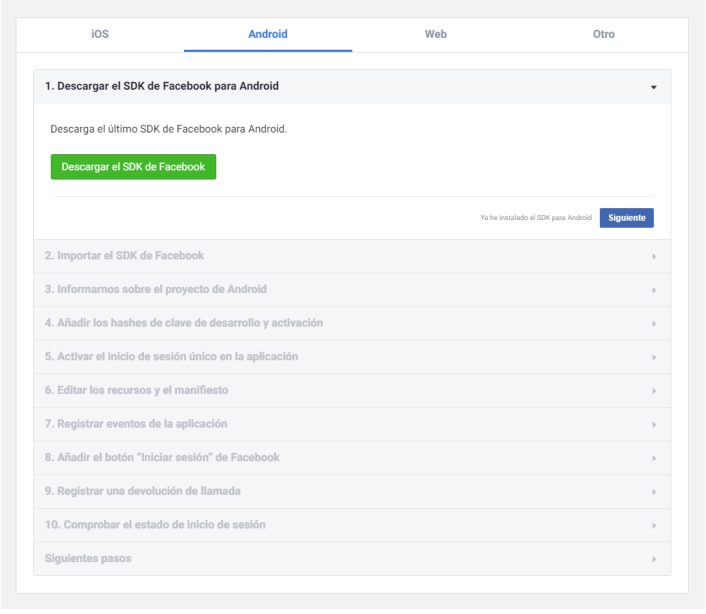



Figura 4.12: Configuración del login con Facebook en un proyecto Android

Lo rellenaremos siguiendo las indicaciones, donde se nos pedirá la clave de desarrollo hash comentada en la fase 5. Para recuperarla debemos abrir la consola de Windows como administrador, posicionarnos en la carpeta bin donde se encuentre instalada nuestra versión de JDK, y escribir la instrucción mostrada en la Figura 4.13, indicando el directorio de la carpeta openssl. Se nos solicitará un contraseña, que será "android":

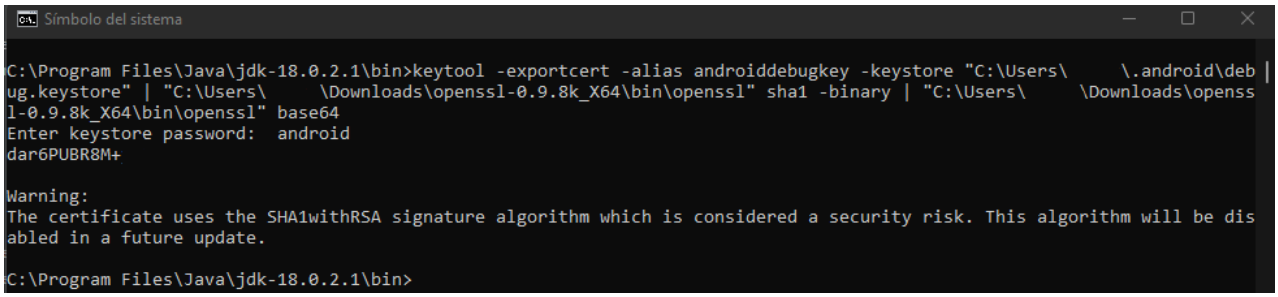



Figura 4.13: Obtención de la clave de desarrollo

Una vez completado el formulario aparecerá en el menú de la izquierda el desplegable "Inicio de sesión con Facebook", que pulsaremos para posteriormente seleccionar "Configurar", y que abrirá la siguiente ventana mostrada en la Figura 4.14:

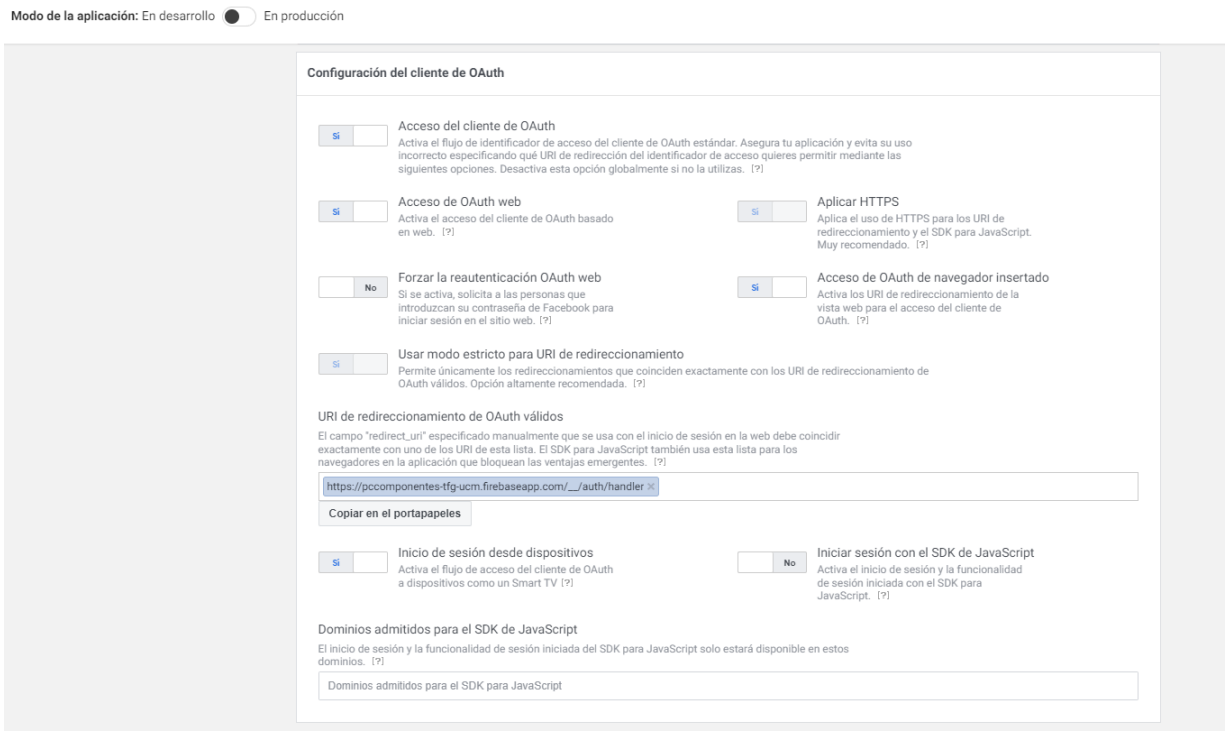



Figura 4.14: Configuración del cliente de OAuth

Activaremos las opciones mostradas en la figura anterior. Por último, copiaremos la URI de direccionamiento de OAuth mostrada durante la configuración de Firebase para Facebook en el campo "URI de direccionamiento de OAuth válidos" de Meta for Developers para finalizar la vinculación.

Si la configuración ha sido exitosa podremos ir al desplegable “Roles”, “Usuarios de prueba” y crear usuarios ficticios para hacer pruebas de desarrollo, como se muestra en la Figura 4.15.

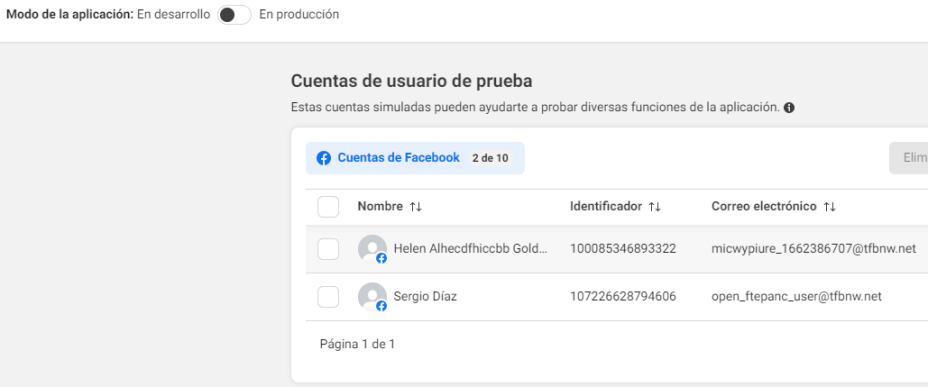



Figura 4.15: Cuentas de usuario de prueba

Estos usuarios se crean con un correo electrónico e identificador aleatorio, pero podremos elegir tanto el nombre como la contraseña y acceder con esas credenciales a Facebook, donde accederemos a un perfil completamente funcional.

Una vez configurado el entorno se comenzó con el desarrollo de la aplicación.

## 4.4. Fase de Diseño

### 4.4.1. Registro y autenticación de usuarios

En la vista de login, manteniendo el diseño de la vista original e inspirándose en el login de la web de PCComponentes, se añadieron los botones para los proveedores adicionales implementados en este proyecto.

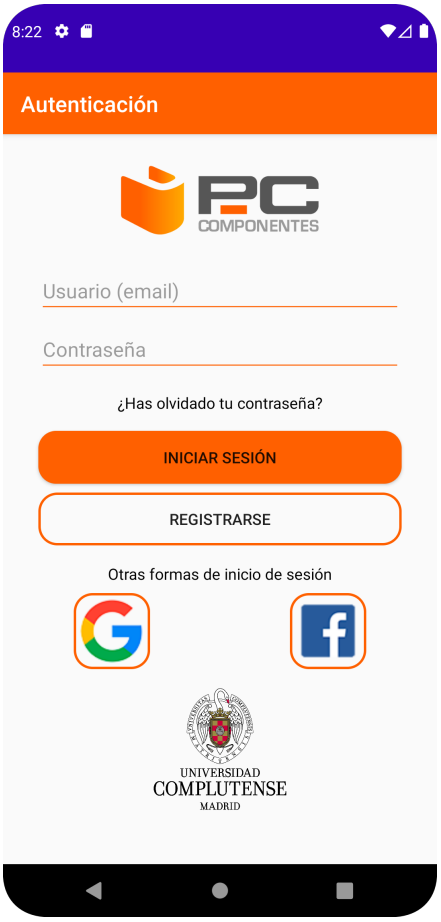



Figura 4.16: Login

En caso de que el usuario no se haya autenticado anteriormente, aparecerá una ventana emergente con el registro según el proveedor del botón que haya pulsado, abriéndose, en el caso de Google, una ventana emergente para elegir el usuario con el que se quiere iniciar sesión y, en el caso de Facebook, una ventana de la aplicación de Facebook o del navegador web integrado para dar los permisos para el registro.

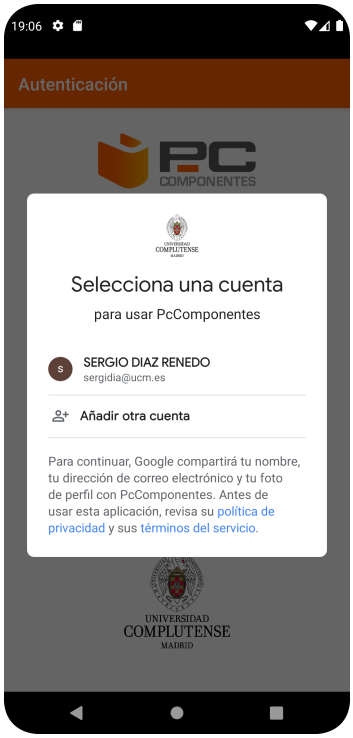



Figura 4.17: Login Google

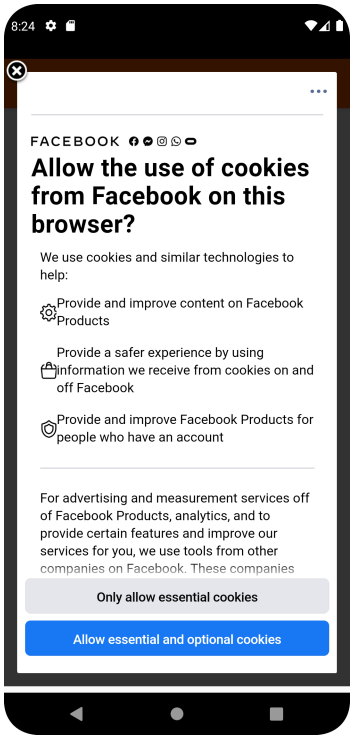



Figura 4.18: Login Facebook - Aceptar cookies

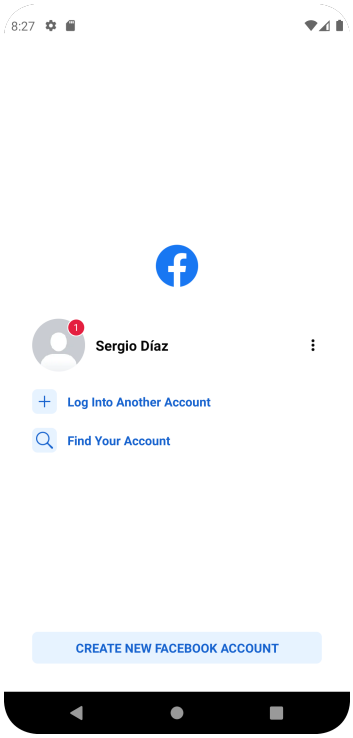



Figura 4.19: Login Facebook - Seleccionar cuenta existente

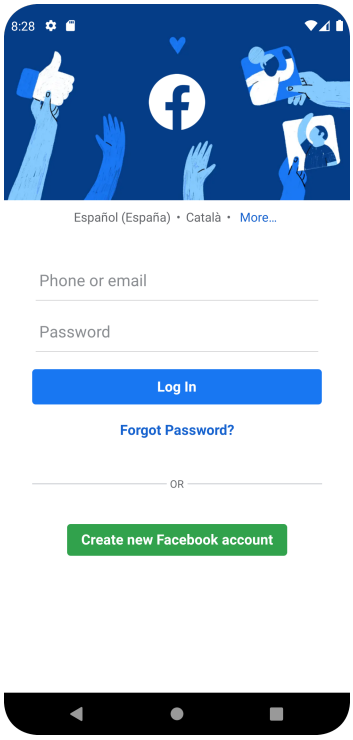



Figura 4.20: Login Facebook - Iniciar sesión en una nueva cuenta

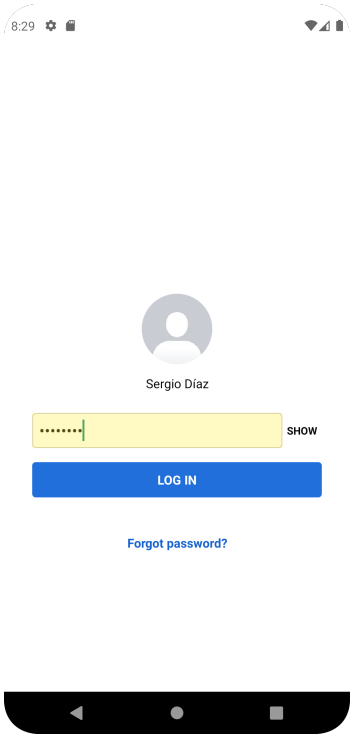



Figura 4.21: Login Facebook

Al finalizar el registro para estos dos proveedores, como no se requiere de confirmación por correo electrónico, se hará login automáticamente.

Para cualquier usuario registrado mediante Google o Facebook, el tipo de notificación predeterminado para el listado de seguimiento será por correo, puesto que es el único dato que tenemos visible. Posteriormente, el usuario podrá modificar sus preferencias de notificación para permitir usar su token y enviar notificaciones push con las alertas.

La autenticación de usuarios mediante Google y Facebook es instantánea y no requiere de interacción por parte del usuario mientras exista una sesión previa abierta, al igual que ocurría con el login mediante correo electrónico.

### 4.4.2. Perfil de usuarios

En esta vista se permite cambiar las preferencias de notificación, la contraseña y dar de baja una cuenta, independientemente del proveedor con el que se haya iniciado sesión, tal y como se ve en la Figura 4.22.

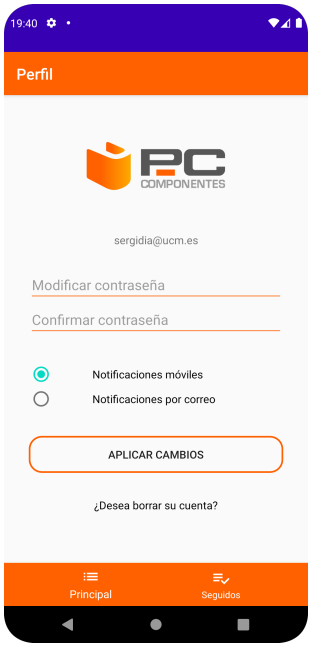



Figura 4.22: Perfil

### 4.4.3. Lista de componentes y de seguimiento

En estas dos vistas se ha mejorado el filtrado de componentes, añadiendo un botón "Switch" que permite cambiar el modo de búsqueda, filtrando por nombre o categoría, como se muestra en la Figura 4.23. Además, como veremos más adelante, se ha añadido un menú desplegable que nos permitirá navegar por la aplicación y realizar otro tipo de filtros, como son la ordenación por precio, nombre y categoría.

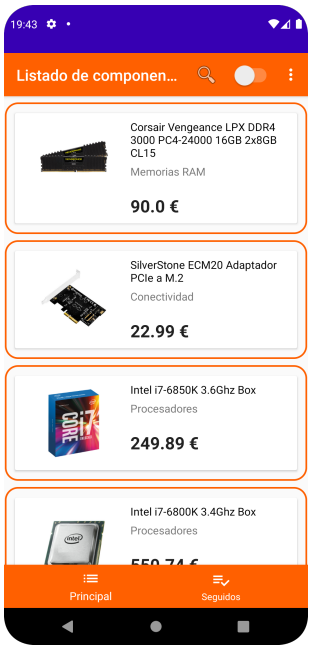



Figura 4.23: Listado de componentes

## 4.5. Fase de Desarrollo

En esta sección analizaremos el código de la aplicación, tanto de su *back-end* como de su *front-end*:

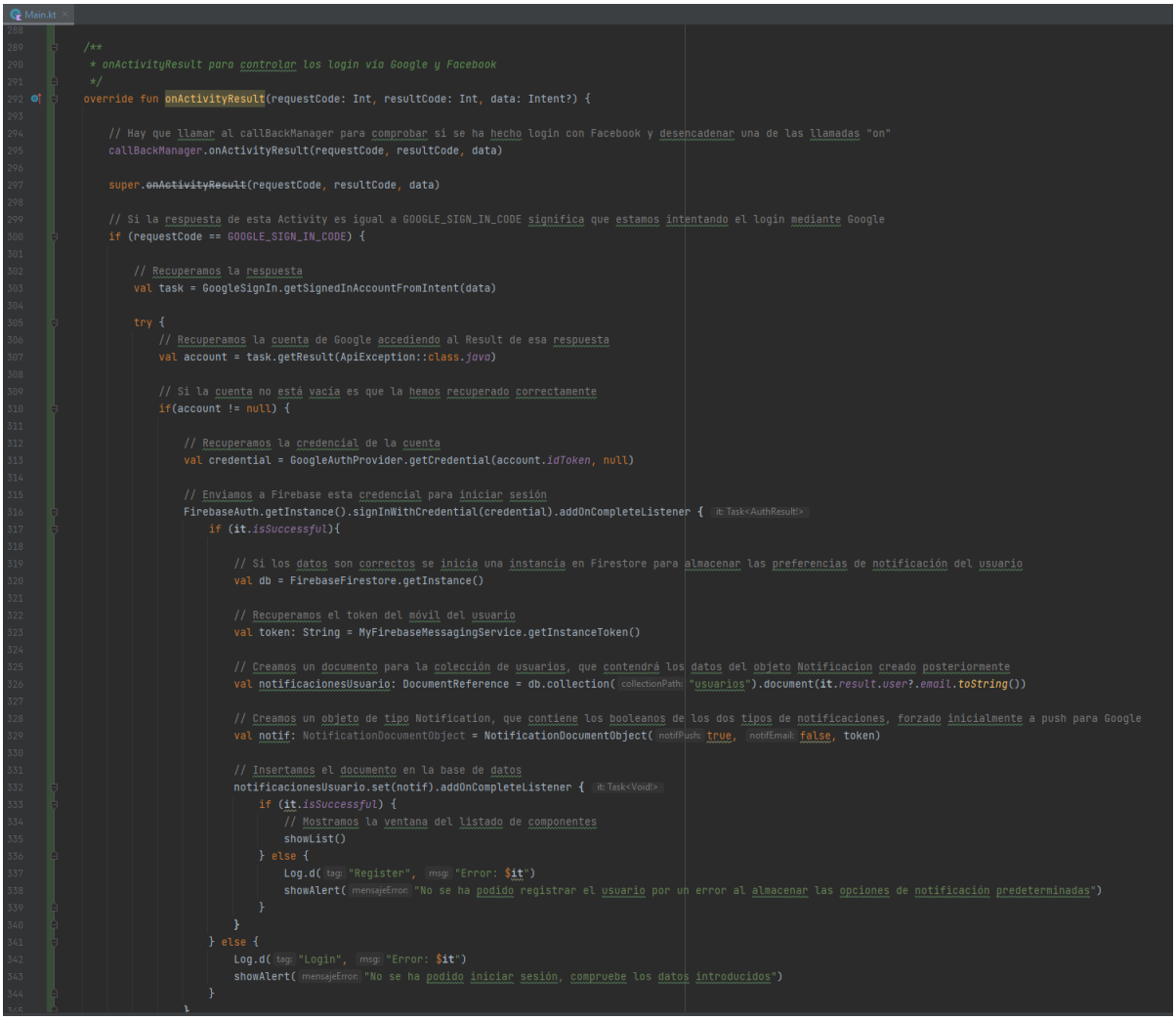
### Back-End

Esta parte se encarga de la lógica de la aplicación, cuyo resultado será mostrado visualmente por el front-end.

- **Login Google:** Creamos el cliente de autenticación de Google, recibimos la credencial que haya seleccionado el usuario y se la enviamos a Firebase para autenticar al usuario, como se muestra en la Figura 4.24 y Figura 4.25.

```
Main.kt

    /**
     * Este Listener implementa el botón de 'Login mediante Google'
     */
    googleButton.setOnClickListener {  it: View!

        // Creamos una constante de configuración para almacenar datos del inicio de sesión
        val googleConf = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestIdToken("710686798944-s7ivkbgop56ba0nsur8a88f9hjvgd3hl.apps.goog...") // Token asociado a nuestra app. Está almacenado en google-services.json
            .requestEmail()
            .build()

        // Creamos el cliente de autenticación de Google con la configuración anterior
        val googleClient = GoogleSignIn.getClient( activity: this, googleConf)

        googleClient.signOut() // Este logout es por si tenemos más de una cuenta de Google asociada a nuestro dispositivo Android

        // Mostramos la pantalla de autenticación de Google. GOOGLE_SIGN_IN_CODE es un código que hemos implementado manualmente y que esperaremos
        // como salida del proceso de autenticación para confirmar su éxito. Si devuelve algo que no sea GOOGLE_SIGN_IN_CODE ha ocurrido un error
        startActivityForResult(googleClient.signInIntent, GOOGLE_SIGN_IN_CODE)
    }
```

Figura 4.24: Login Google parte 1



```kotlin
/**
 * onActivityResult para controlar los login vía Google y Facebook
 */
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {

    // Hay que llamar al callBackManager para comprobar si se ha hecho login con Facebook y desencadenar una de las llamadas "on"
    callBackManager.onActivityResult(requestCode, resultCode, data)

    super.onActivityResult(requestCode, resultCode, data)

    // Si la respuesta de esta Activity es igual a GOOGLE_SIGN_IN_CODE significa que estamos intentando el login mediante Google
    if (requestCode == GOOGLE_SIGN_IN_CODE) {

        // Recuperamos la respuesta
        val task = GoogleSignIn.getSignedInAccountFromIntent(data)

        try {
            // Recuperamos la cuenta de Google accediendo al Result de esa respuesta
            val account = task.getResult(ApiException::class.java)

            // Si la cuenta no está vacía es que la hemos recuperado correctamente
            if(account != null) {

                // Recuperamos la credencial de la cuenta
                val credential = GoogleAuthProvider.getCredential(account.idToken, null)

                // Enviamos a Firebase esta credencial para iniciar sesión
                FirebaseAuth.getInstance().signInWithCredential(credential).addOnCompleteListener {  it: Task<AuthResult!> 
                    if (it.isSuccessful){

                        // Si los datos son correctos se inicia una instancia en Firestore para almacenar las preferencias de notificación del usuario
                        val db = FirebaseFirestore.getInstance()

                        // Recuperamos el token del móvil del usuario
                        val token: String = MyFirebaseMessagingService.getInstanceToken()

                        // Creamos un documento para la colección de usuarios, que contendrá los datos del objeto Notificacion creado posteriormente
                        val notificacionesUsuario: DocumentReference = db.collection( collectionPath: "usuarios").document(it.result.user?.email.toString())

                        // Creamos un objeto de tipo Notification, que contiene los booleanos de los dos tipos de notificaciones, forzado inicialmente a push para Google
                        val notif: NotificationDocumentObject = NotificationDocumentObject( notifPush: true,  notifEmail: false, token)

                        // Insertamos el documento en la base de datos
                        notificacionesUsuario.set(notif).addOnCompleteListener {  it: Task<Void!> 
                            if (it.isSuccessful) {
                                // Mostramos la ventana del listado de componentes
                                showList()
                            } else {
                                Log.d( tag: "Register",  msg: "Error: $it")
                                showAlert( mensajeError: "No se ha podido registrar el usuario por un error al almacenar las opciones de notificación predeterminadas")
                            }
                        }
                    } else {
                        Log.d( tag: "Login",  msg: "Error: $it")
                        showAlert( mensajeError: "No se ha podido iniciar sesión, compruebe los datos introducidos")
                    }
                }
```

Figura 4.25: Login Google parte 2

- **Login Facebook:** Creamos una instancia de Login de Facebook y damos permisos de lectura únicamente para el email del usuario. Si la instancia de Facebook devuelve que el login se ha hecho correctamente, recuperamos la credencial y se la enviamos a Firebase para autenticar al usuario, tal y como se muestra en la Figura 4.26.

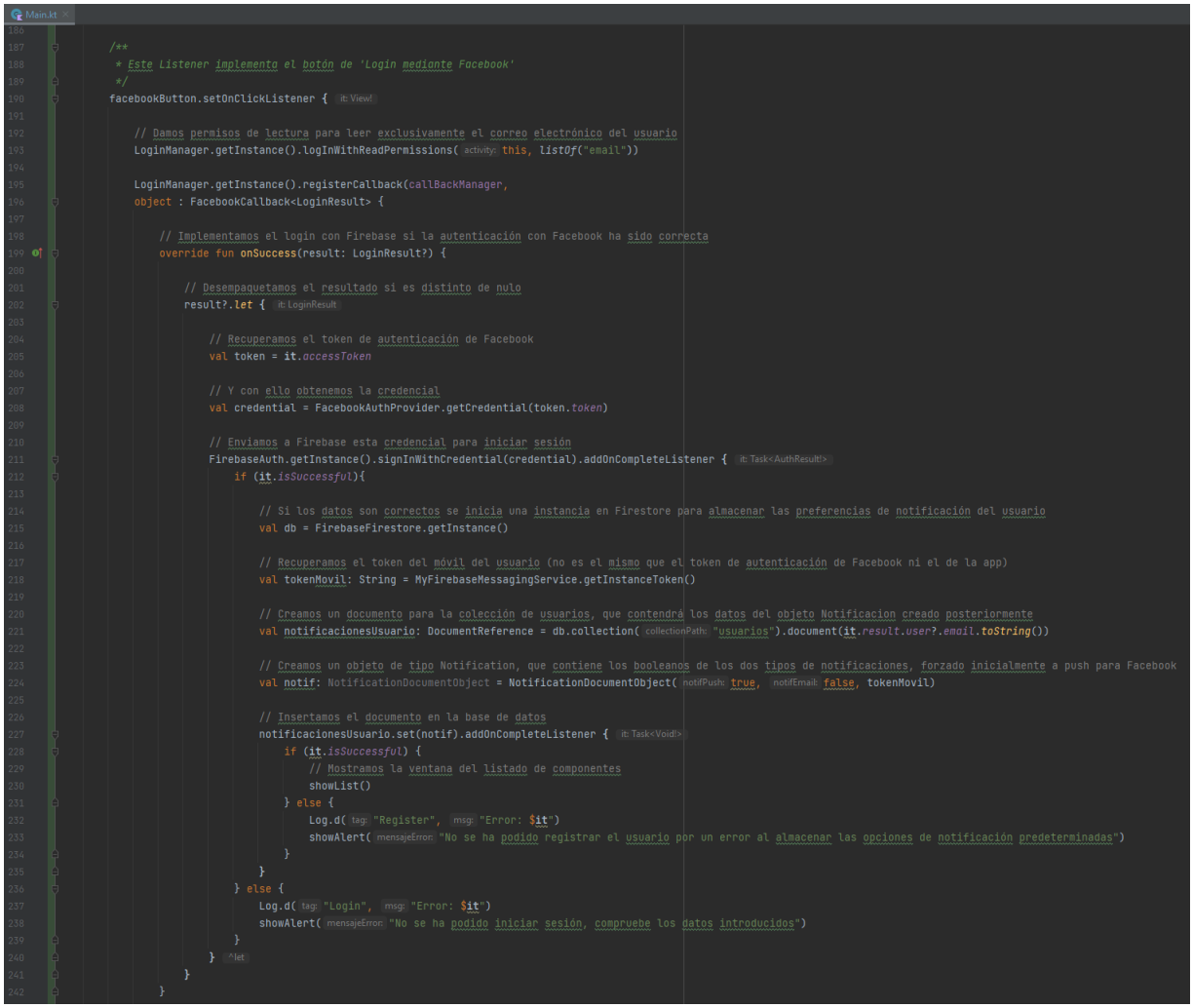



Figura 4.26: Login Facebook

- **Perfil:** Se recupera el proveedor con el que se ha autenticado el usuario para gestionar los posibles cambios que solicite el usuario.
- **Listado de artículos y seguimiento:** Recuperamos los componentes de la base de datos, sin filtros en una primera ejecución.

  Si el usuario filtra por nombre o precio, se modifican las querys y se vuelve a imprimir el resultado, siempre de 10 en 10 componentes para no saturar el hilo de la aplicación, como se muestra en la Figura 4.27 y en la Figura 4.28.

```java
MainActivity.java
                }
            });
    }

    // ------------------------------------------------------------------------------------------------
    // ------------------------------------ INICIO LISTA NAVIGATION DRAWER ----------------------------
    // ------------------------------------------------------------------------------------------------

    /**
     * Inicialización de la lista de componentes recuperando los 10 primeros de la base de datos ordenándolos por nombre de A a Z
     */
    private void iniciarListaComponentesNombreAZ() {

        Log.d( tag: "Listado Main",  msg: "Se inicializa la lista de componentes por nombre AZ");

        db.collection( collectionPath: "componentes") CollectionReference
                .orderBy( field: "nombre", Query.Direction.ASCENDING) Query
                .limit(10)
                .get() Task<QuerySnapshot>
                .addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {

                    @Override
                    public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
                        for (QueryDocumentSnapshot document : queryDocumentSnapshots) {

                            addComponente(document);
                        }
                        cargarRecyclerView();
                        isLoading = false;
                    }
                });
    }
```

Figura 4.27: Ordenación de componentes por nombre de A a Z

```java
SeguidosView.java
    /**
     * Comparador manual por nombre ordenado de menor a mayor
     */
    public class ordenacionNombreAZ implements Comparator<Item> {
        public int compare(Item izq, Item der) {
            return izq.getNombre().compareTo(der.getNombre());
        }
    }
```

Figura 4.28: Ordenación de seguimiento por nombre de A a Z

Como se observa, las ordenaciones se realizan de forma completamente diferente. En la vista del listado principal de componentes este filtrado se realiza con consultas a la base de datos de Firebase, mientras que en el caso de la lista de seguimiento se realiza ordenando un "ArrayList".

Esto es así porque, como se mostró en la Figura 3.1, la lista de seguimientos es una colección de documentos, cuyos nombres son los identificadores del componente al que hacen referencia, con un único atributo: el precio de notificación al usuario. Por tanto, para extraer todos los datos de los componentes que sigue un usuario, se recorre la colección de seguimiento almacenando los nombres de los documentos (ids de los componentes), luego se hace una consulta a la base de datos de Firebase para extraer los documentos del listado de componentes que coincidan con dichos ids, y estos documentos se guardan en un "ArrayList" para mostrar sus datos por pantalla.

Lo que nos permite, en el caso de la vista de seguimiento, filtrar por nombre o categoría y extraer todos los componentes que contengan el texto que hayamos

introducido, mientras que en el caso de la vista del listado general de componentes las limitaciones de Firebase impiden ese tipo de filtrado, y las búsquedas por nombre y categoría se hacen buscando componentes que empiecen por el texto introducido, no que lo contengan.

### Front-End

Esta parte está dedicada a la representación de la información devuelta por el back-end de un modo visual para el usuario:

- **Login:** El login está compuesto de un pequeño formulario que solicita el correo electrónico y la contraseña del usuario, dos botones de inicio de sesión y registro, una opción de ¿has olvidado tu contraseña? Y dos botones adicionales que permiten el login y registro mediante Google y Facebook.
- **Listado de componentes y seguidos:** Se ha añadido una ventana de navegación lateral, llamada *Navigation Drawer*, que se puede expandir pulsando a la izquierda de la pantalla y arrastrando hacia la derecha, y que ofrece opciones de navegación por la aplicación, tal y como se puede comprobar en la Figura 4.29.


Figura 4.29: Navigation Drawer

- **Perfil:** Se ha añadido la misma ventana de navegación lateral que al listado de componentes y de seguidos.

# Capítulo 5

# Trabajo Futuro

Analizando el estado actual del proyecto, se pueden proponer varias ideas a desarrollar en un futuro que se van a tratar en las próximas secciones.

## 5.1. Modo desconectado

Actualmente, la aplicación requiere de conexión permanente para su uso, ya que es necesario conectarnos a Firebase para recuperar los componentes almacenados de la base de datos. Sería posible, sin embargo, permitir la opción al usuario de descargar el contenido de la base de datos en el almacenamiento de su dispositivo, para permitir guardar listas de seguimiento que se sincronizarían con Firebase al recuperar la conexión.

## 5.2. Login con el proveedor PCComponentes

Al igual que hemos hecho con otros proveedores como Google y Facebook, sería posible el uso de la credencial de PCComponentes para iniciar sesión en la aplicación, integrando el perfil de la web e incluso el carrito de compra, lo que permitiría de forma más sencilla monetizar la aplicación con enlaces referidos.

## 5.3. Monetización con enlaces referidos

Un enlace referido es una forma de monetización no intrusiva que no afecta al precio del usuario final, en el que el proveedor paga un porcentaje del precio de compra de cada artículo referido a la empresa que promociona su sitio web. Actualmente, la aplicación redirige a la web de PCComponentes cuando se pincha en los detalles de un artículo. PCComponentes dispone de un programa de afiliación [15], en el que, en caso de ser

aceptado, permite modificar los enlaces añadiendo una cadena de texto final al mismo que identifica inequívocamente a la empresa afiliada, y con la que recibe ese porcentaje fijado de comisión si un usuario compra un artículo desde el enlace referido.

## 5.4. Versión iOS

La aplicación sólo es compatible con dispositivos Android pero sería posible el desarrollo de la aplicación para iOS, ya que tanto Firebase como Meta for Developers son compatibles con este sistema operativo.

## 5.5. Cambio de Base de Datos

Firebase, al disponer de una base de datos NoSQL, limita mucho a la hora de realizar consultas sobre el contenido almacenado, lo que impide implementaciones de filtros complejos que sí permite la web de PCComponentes. Cambiar la base de datos a una SQL solventaría esta limitación.

# Capítulo 6

# Conclusiones

En primer lugar, las diferentes características de implementación de la autenticación mediante Google y Facebook, que requieren cumplir una serie de requisitos de seguridad en nuestra aplicación, han permitido ampliar la experiencia con el uso de huellas digitales y claves de desarrollo.

Por otro lado, el modificar una aplicación ya existente y funcional para añadirle nuevas características y mejorar alguna de las existentes, ha reforzado la importancia de los comentarios en los códigos, que han agilizado mucho el desarrollo.

En referencia a los cambios de interfaz, la creación de mock-ups ayudó a pensar una distribución de los elementos de la pantalla antes de empezar con el desarrollo, evitando complicaciones durante el mismo.

En definitiva, este proyecto ha ayudado a reforzar los conocimientos de programación aprendidos durante la carrera y la integración entre distintas plataformas.

# Capítulo 7

# Conclusions

First of all, the different implementation characteristics of authentication through Google and Facebook, which require meeting a series of security requirements in our application, have allowed me to expand my experience with the use of fingerprints and development keys.

On the other hand, modifying an existing and functional application to add new features and improve some of the existing ones has reinforced the importance of code comments, which have greatly speeded up the development.

In reference to the interface changes, mock-ups helped to think about a distribution of the elements on the screen before starting the development, avoiding complications during it.

In conclussion, this project has helped to reinforce the programming knowledge learned during the degree and the integration between different platforms.

# Índice de figuras

# Bibliografía


[1] Android studio. https://developer.android.com/studio.

[2] Descarga de java development kit. https://www.oracle.com/java/technologies/downloads/.

[3] Firebase. https://firebase.google.com/?hl=es.

[4] Firebase realtime database. https://firebase.google.com/docs/database.

[5] Github. https://github.com/.

[6] ios. https://www.apple.com/es/ios/.

[7] Java | Oracle. https://www.java.com/es/.

[8] JSON. https://www.json.org/json-es.html.

[9] Kotlin, declarado como lenguaje oficial por Google para desarrollos Android. https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/.

[10] Kotlin Programming Language. https://kotlinlang.org/.

[11] Meta for developers. https://developers.facebook.com/?locale=es_ES.

[12] Meta platforms. https://about.facebook.com/es/meta/.

[13] Openssl. https://www.openssl.org/.

[14] PcComponentes.com | Tienda de Informática y Tecnología online. https://www.pccomponentes.com/.

[15] Programa de afiliados PcComponentes. https://www.pccomponentes.com/soporte/cual-es-la-comision-en-el-programa-de-afiliados-de-pccomponentes.html/.

[16] Sistema de alertas pccomponentes. https://eprints.ucm.es/id/eprint/62072/.

[17] Sistema Operativo Android. https://www.android.com/intl/es_es/.

[18] Software de control de versiones git. https://git.kernel.org/pub/scm/git/git.git/.

[19] Unity. https://unity.com/es.

PASCAL
ENERO 2018
Ult. actualización 26 de septiembre de 2022
LaTeX lic. LPPL & powered by TEFLON CC-ZERO