

# **Firestore**

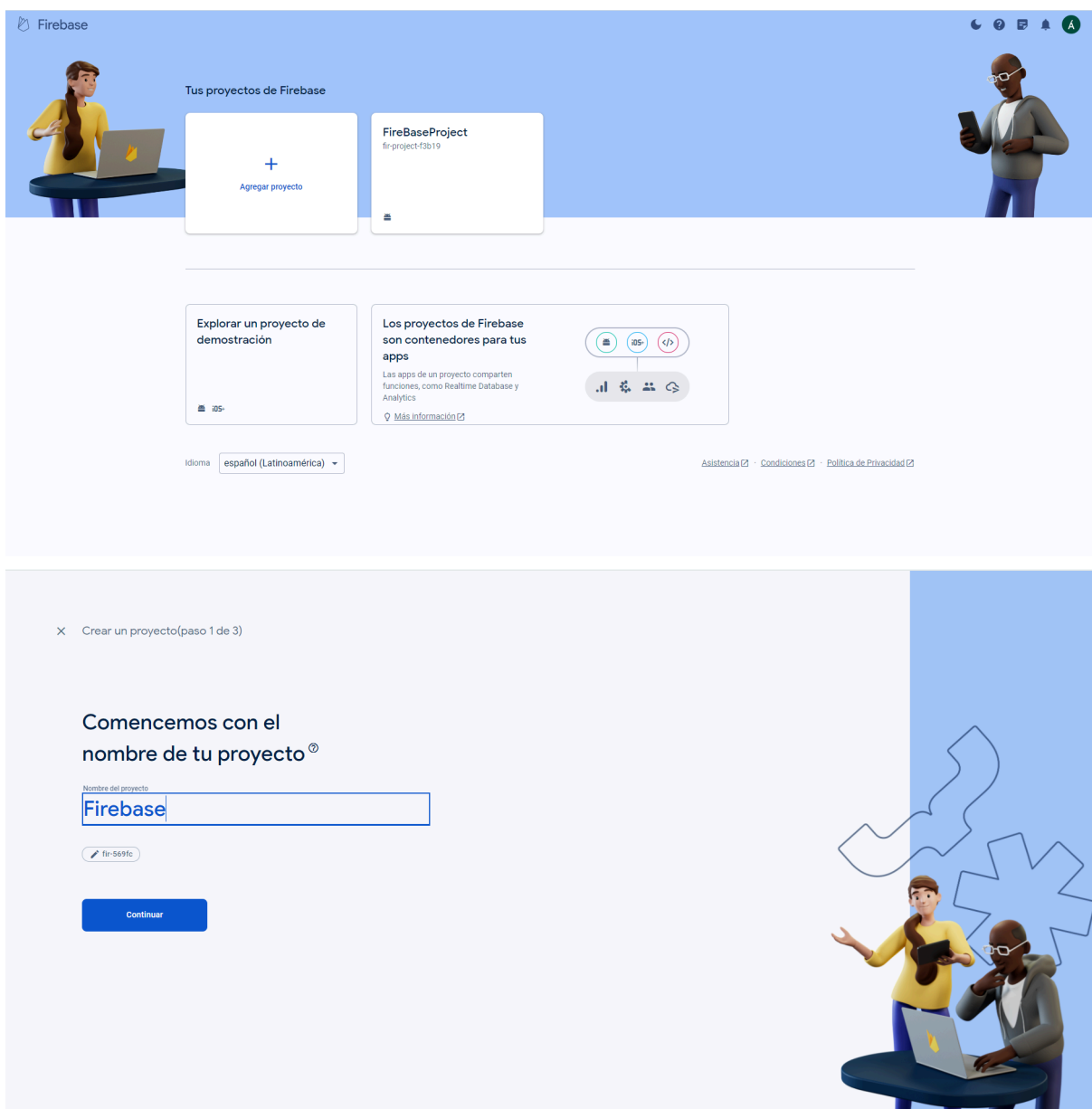
**Álvaro Castilla Loaiza**

**2º DAM**

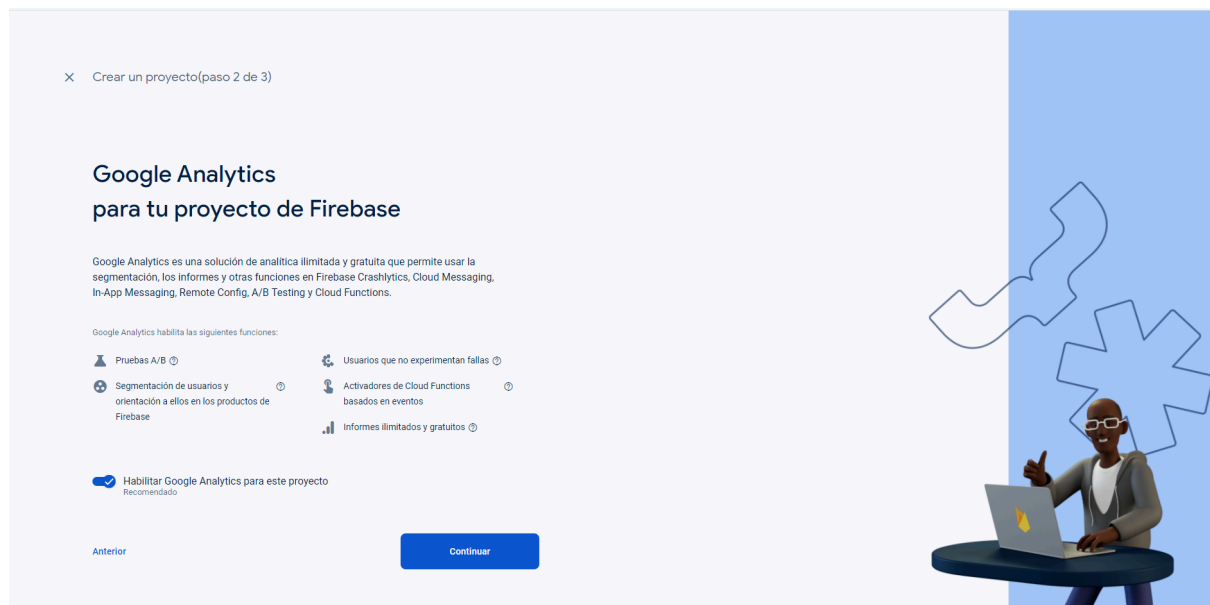
Firestore es una plataforma creada por Google que proporciona diversas herramientas como: bases de datos en tiempo real, autenticación de usuarios, almacenamiento en la nube, entre otras.

## **Crear BDD en Firestore**

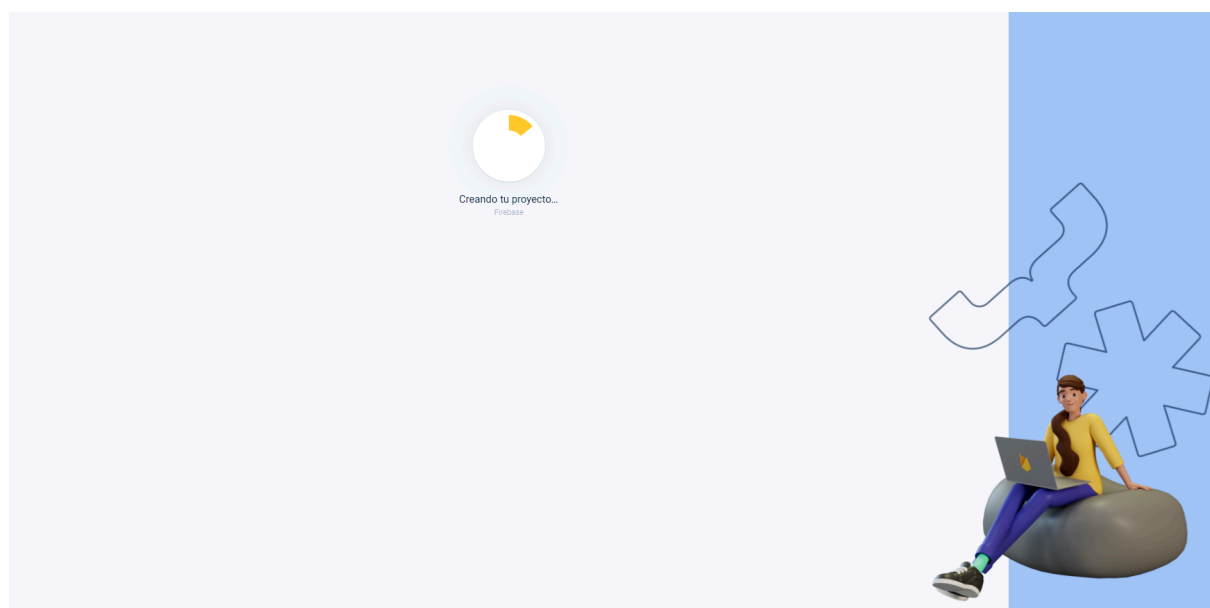
Lo primero es entrar en [Firestore](#) y crear un nuevo proyecto.



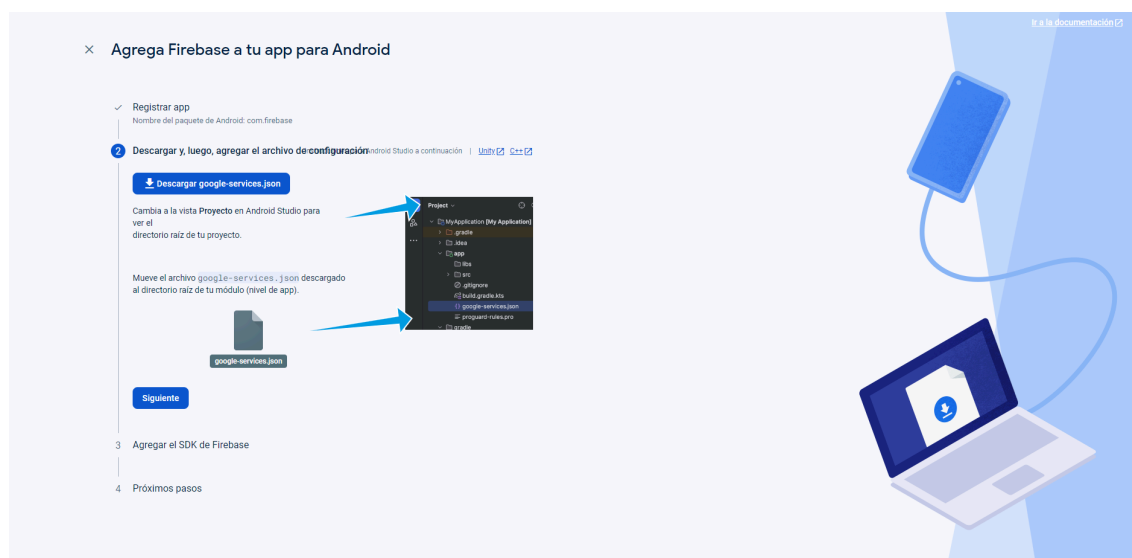
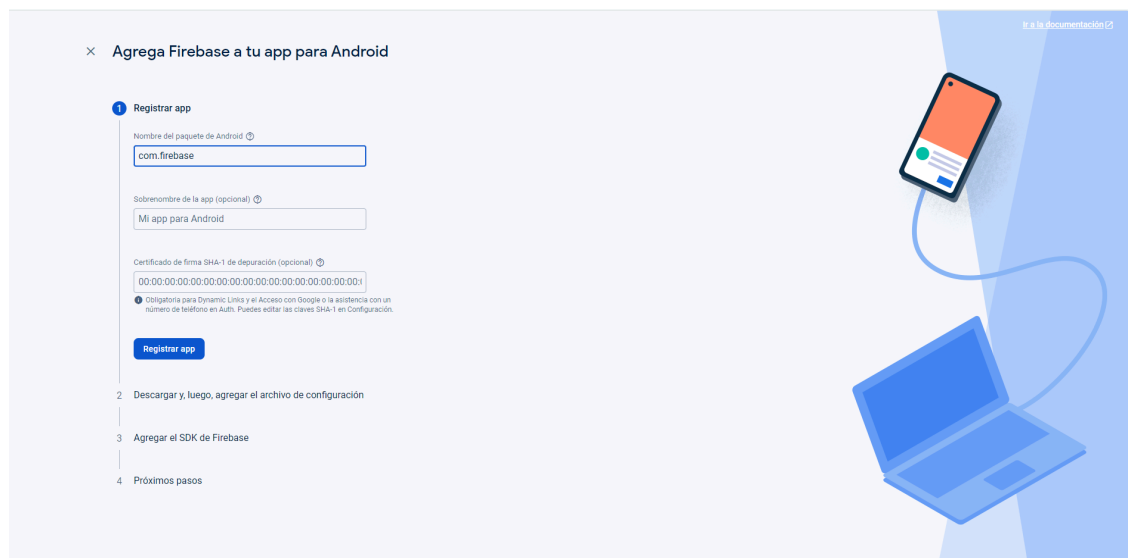
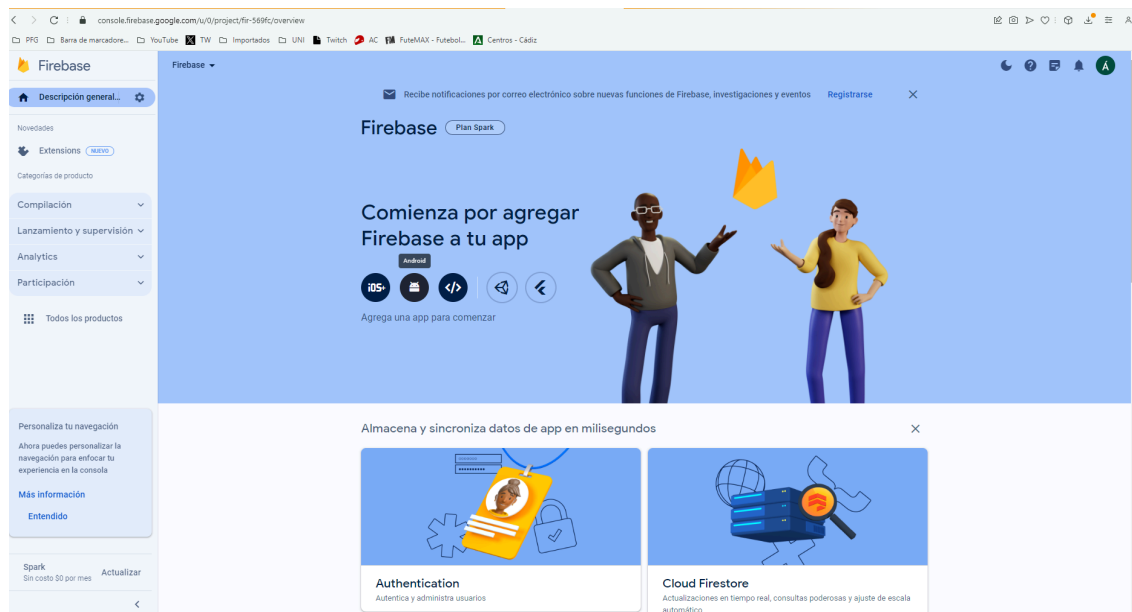
Le pondremos un nombre al proyecto y seguidamente activaremos [Google Analytics](#).



Tras esto crearemos el proyecto.



Tras la creación del proyecto, selecciona Android  
Tras esto crearemos el proyecto.

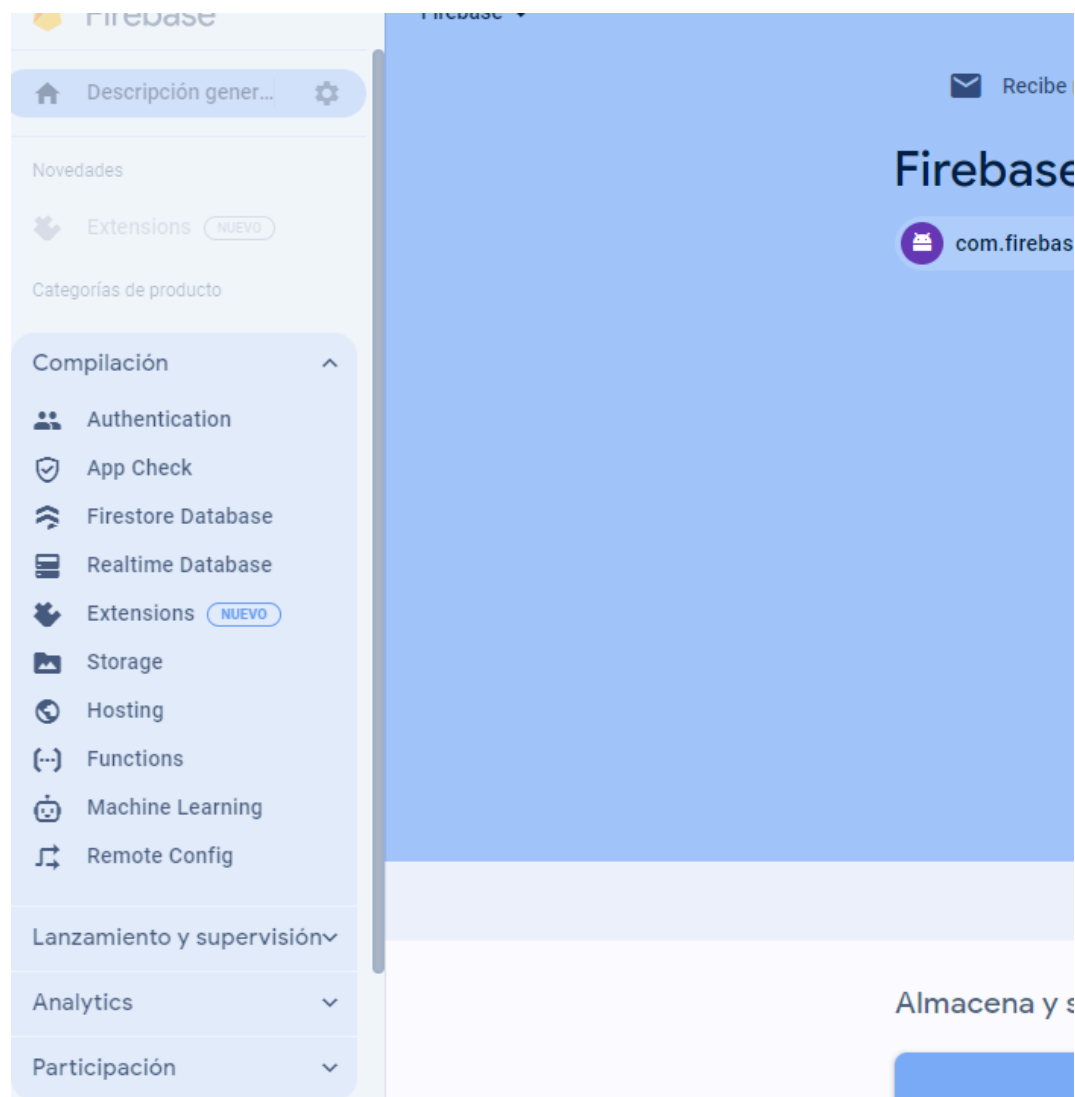


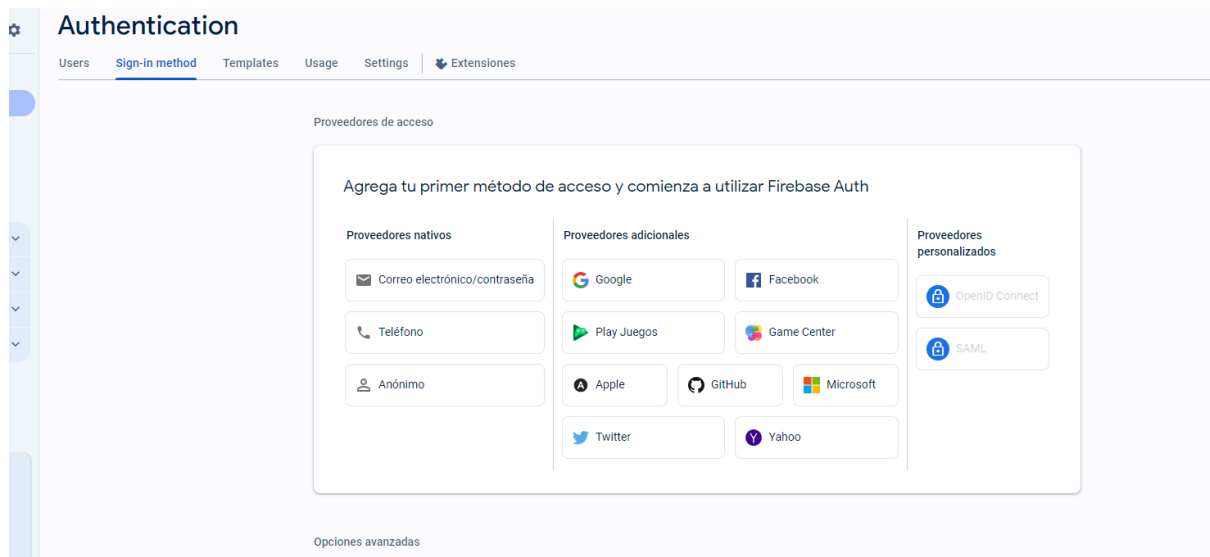
Descarga el archivo **google-services.json** y pégalo en la carpeta /app

Este equipo > Disco local (C:) > Usuarios > AlvaroPC > AndroidStudioProjects > FireBaseProject > start > app					
Nombre	Fecha de modificación	Tipo	Tamaño		
build	01/02/2024 19:29	Carpeta de archivos			
src	01/02/2024 18:54	Carpeta de archivos			
.gitignore	01/02/2024 18:54	Documento de te...	1 KB		
build.gradle.kts	01/02/2024 18:54	Archivo KTS	4 KB		
google-services.json	01/02/2024 19:58	Archivo JSON	1 KB		
proguard-rules.pro	01/02/2024 18:54	Archivo PRO	1 KB		

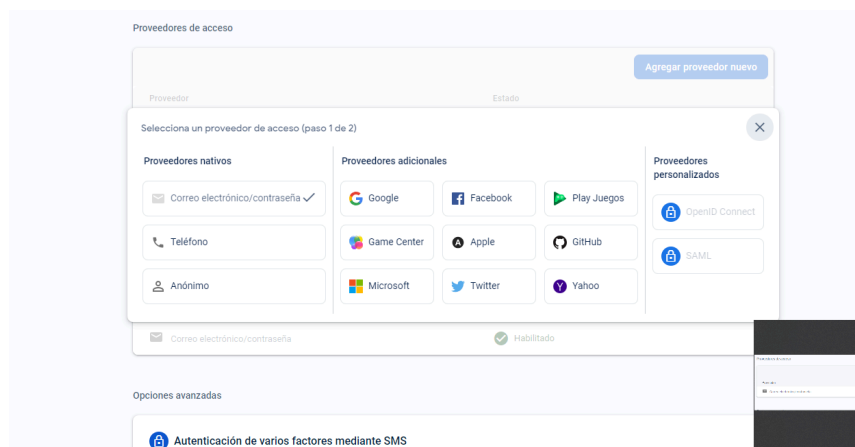
Tras esto habilitaremos la autenticación y la base de datos.

## Autenticación





Agregaremos la opción de correo electrónico



Tras activar la opción agregaremos un proveedor nuevo anónimo. Esto da la opción de que un usuario pueda utilizar la aplicación sin necesidad de iniciar sesión, pero sus datos no se guardarán si cambia de teléfono o borra la aplicación.

Podemos agregar muchas más formas de iniciar sesión pero por ahora solo activaremos esas dos.

## Base de datos

The image shows two screenshots of the Google Cloud platform interface. The top screenshot is a sidebar menu from the Google Cloud console, with 'Firestore Database' selected. The bottom screenshot is the 'Cloud Firestore' landing page, which features a 'Crear base de datos' button and a 'Más información' section with links to '¿Cómo empiezo?', '¿Cuánto costará Cloud Firestore?', and '¿Cómo me puede ayudar'. The landing page also includes a video titled 'Introducing Cloud Firestore'.

**Firestore Database**

Novedades

Extensions **NUEVO**

Categorías de producto

Compilación

- Authentication
- App Check
- Firestore Database**
- Realtime Database
- Extensions **NUEVO**
- Storage
- Hosting
- Functions
- Machine Learning
- Remote Config

Lanzamiento y supervisión

Analytics

Participación

**Más información**

- ¿Cómo empiezo?  
Ver los documentos
- ¿Cuánto costará Cloud Firestore?  
Ver los precios
- ¿Cómo me puede ayudar

**Cloud Firestore**

Actualizaciones en tiempo real, consultas potentes y ajuste de escala automático

**Crear base de datos**

**Más información**

- ¿Cómo empiezo?  
Ver los documentos
- ¿Cuánto costará Cloud Firestore?  
Ver los precios

**Introducing Cloud Firestore**

Ver más ta... Compartir

Accederemos a Firestore Database y crearemos una base de datos.

### Crear base de datos

1 Establece el nombre y la ubicación

2 Reglas de seguridad

ID de la base de datos

(default)

Ubicación

eur3 (Europe)

La configuración de ubicación es el lugar donde se almacenarán tus datos de Cloud Firestore

! No podrás cambiar la ubicación después de configurarla. Además, si esta es tu primera base de datos, esta configuración de la ubicación será la de tu bucket predeterminado de Cloud Storage.

Más información

Cancelar

Siguiente

Cambiaremos la ubicación a Europa y le daremos a siguiente.

### Crear base de datos

✓ Establece el nombre y la ubicación

2 Reglas de seguridad

Después de definir la estructura de tus datos, debes crear reglas para protegerlos.  
[Más información](#)

☒ Iniciar en modo de producción

Tus datos son privados de forma predeterminada. El acceso de lectura/escritura de los clientes solo se otorgará como se indica en tus reglas de seguridad.

☐ Comenzar en modo de prueba

Para permitir una configuración rápida, los datos se abren de forma predeterminada. Sin embargo, debes actualizar las reglas de seguridad en un plazo de 30 días a fin de habilitar el acceso de lectura/escritura a largo plazo para los clientes.

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
  }
}
```

Se denegarán todas las operaciones de lectura y escritura de terceros.

Cancelar

Crear

Dejaremos esta pantalla por defecto y crearemos la base de datos.

Cuando ya se haya creado, accederemos al apartado reglas y pegaremos lo siguiente:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow create: if request.auth != null;
      allow read, update, delete: if request.auth != null && resource.data.userId == req
    }
  }
}
```

Ya habríamos terminado la configuración de Firebase y estaríamos listos para empezar a programar.

## ***Crear la aplicación***

### **Dependencias**

Tendremos que agregar las siguientes dependencias:

```
plugins {
    // ...

    // Add the dependency for the Google services Gradle plugin
    id("com.google.gms.google-services") version "4.4.0" apply false
}
```

```
plugins {
    id("com.android.application")
    // Add the Google services Gradle plugin
    id("com.google.gms.google-services")
    ...
}

dependencies {
    // Import the Firebase BoM
    implementation(platform("com.google.firebase:firebase-bom:32.7.1"))

    // TODO: Add the dependencies for Firebase products you want to use
    // When using the BoM, don't specify versions in Firebase dependencies
    implementation("com.google.firebase:firebase-analytics")

    // Add the dependencies for any other desired Firebase products
    // https://firebase.google.com/docs/android/setup#available-libraries
}
```

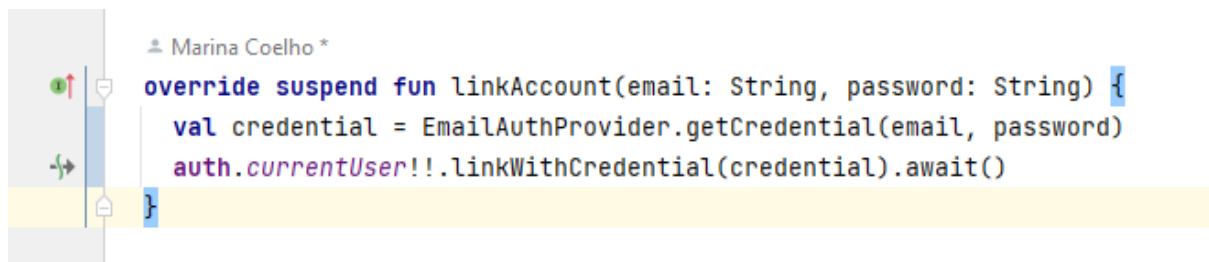


Tras agregar las dependencias empezaremos a diseñar nuestra aplicación. En este ejemplo utilizaremos una aplicación de notas.

Primero empezaremos habilitando el inicio de sesión con Firebase. Lo haremos de la siguiente manera.

## Autenticación

Crearemos un archivo AccountService y agregaremos esta función para enlazar un correo electrónico y una contraseña a una cuenta anónima.

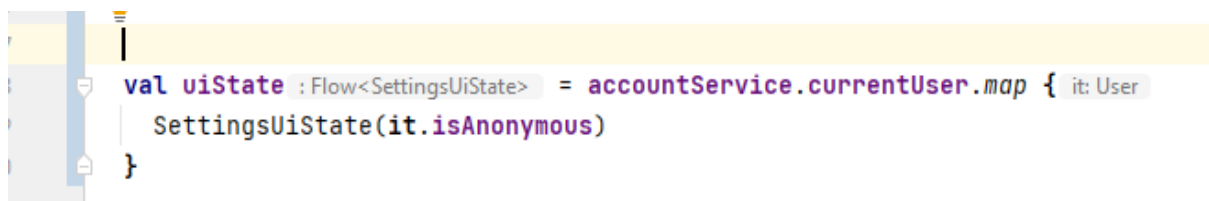


```
Marina Coelho *
override suspend fun linkAccount(email: String, password: String) {
    val credential = EmailAuthProvider.getCredential(email, password)
    auth.currentUser!!.linkWithCredential(credential).await()
}
```

Tras esto, en el viewmodel de la pantalla de registro pondremos lo siguiente:

```
launchCatching { this: CoroutineScope
    accountService.linkAccount(email, password)
    openAndPopUp(SETTINGS_SCREEN, SIGN_UP_SCREEN)
}
```

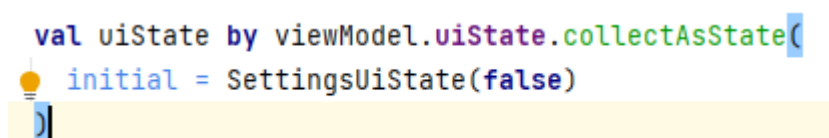
Tras esto accederemos al SettingsViewModel, en el que pondremos lo siguiente:



```
val uiState : Flow<SettingsUiState> = accountService.currentUser.map { it: User
    SettingsUiState(it.isAnonymous)
}
```

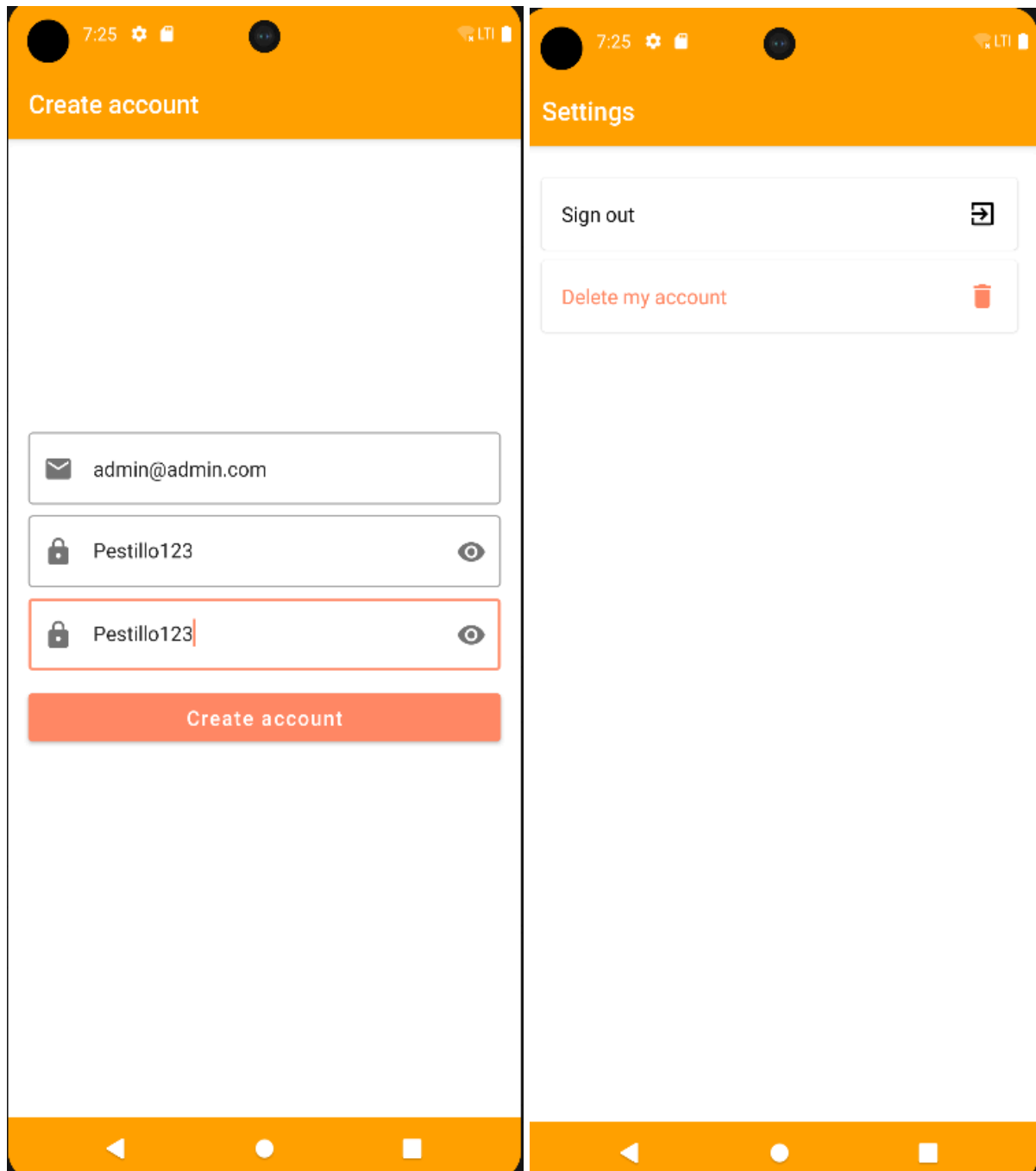
Esto se pone para que el ViewModel esté escuchando siempre el estado de la cuenta, para cambiar los botones o no.

También añadiremos la variable en el settingsScreen



```
val uiState by viewModel.uiState.collectAsState(
    initial = SettingsUiState(false)
)
```

Ahora probaremos si podemos acceder mediante un correo y una contraseña.



Vemos que cuando creamos una cuenta con nuestro correo, al acceder a Settings los botones han cambiado.

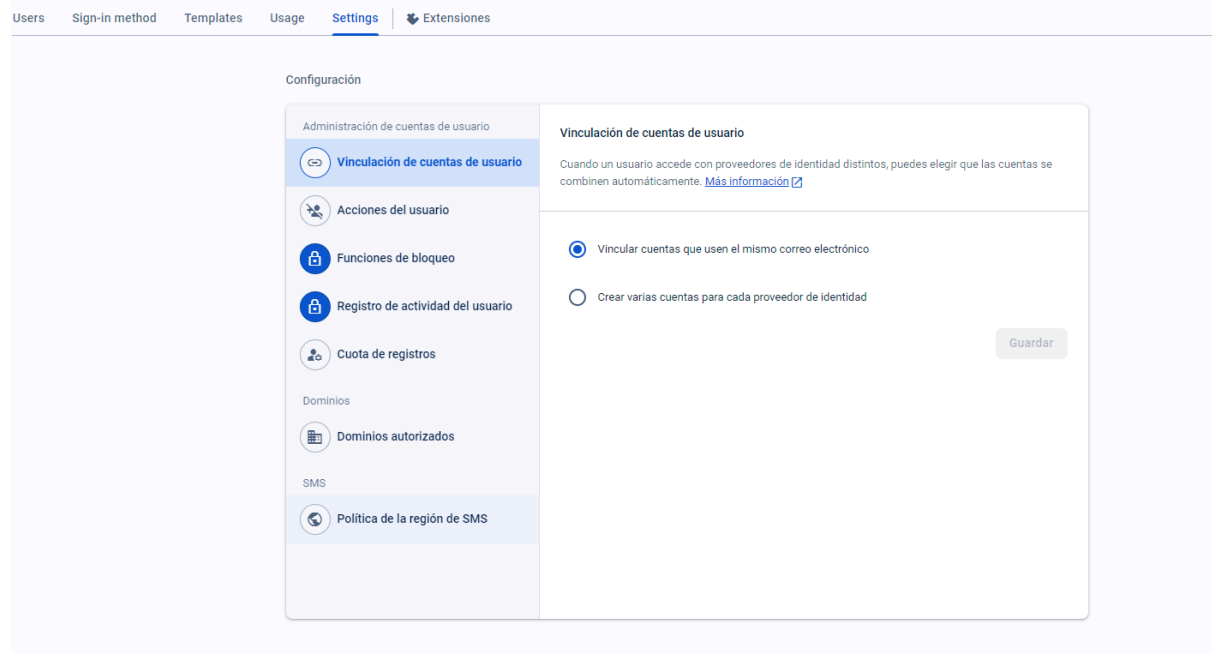
**Importante:** si al intentar acceder nos salta el siguiente error

*the given sign-in provider is disabled for this firebase project. enable it in the firebase console, under the sign in method tab of the Auth section [Please verify the new email before changing email]*

Tendremos que hacer lo siguiente:

Accedemos a Firebase y al apartado de autenticación.

## Authentication



En la configuración nos iremos a *Acciones del usuario* y desmarcaremos la tercera casilla.

Ya habríamos terminado con la autenticación

## Base de datos en la nube

Utilizaremos un flow para que vaya leyendo de la base de datos de Firebase:

```
Marina Coelho +1 *  
@OptIn(ExperimentalCoroutinesApi::class)  
override val tasks: Flow<List<Task>>  
    get() =  
        auth.currentUser.flatMapLatest { user ->  
            firestore.collection(TASK_COLLECTION).whereEqualTo(USER_ID_FIELD, user.id).dataObjects()  
        }
```

Tras esto actualizaremos la variable tasks en el TasksViewmodel.

```
val tasks : Flow<List<Task>> = storageService.tasks
```

También lo actualizaremos en el TaskScreen y cambiaremos la LazyRow para que en ella se muestre la variable tasks

```

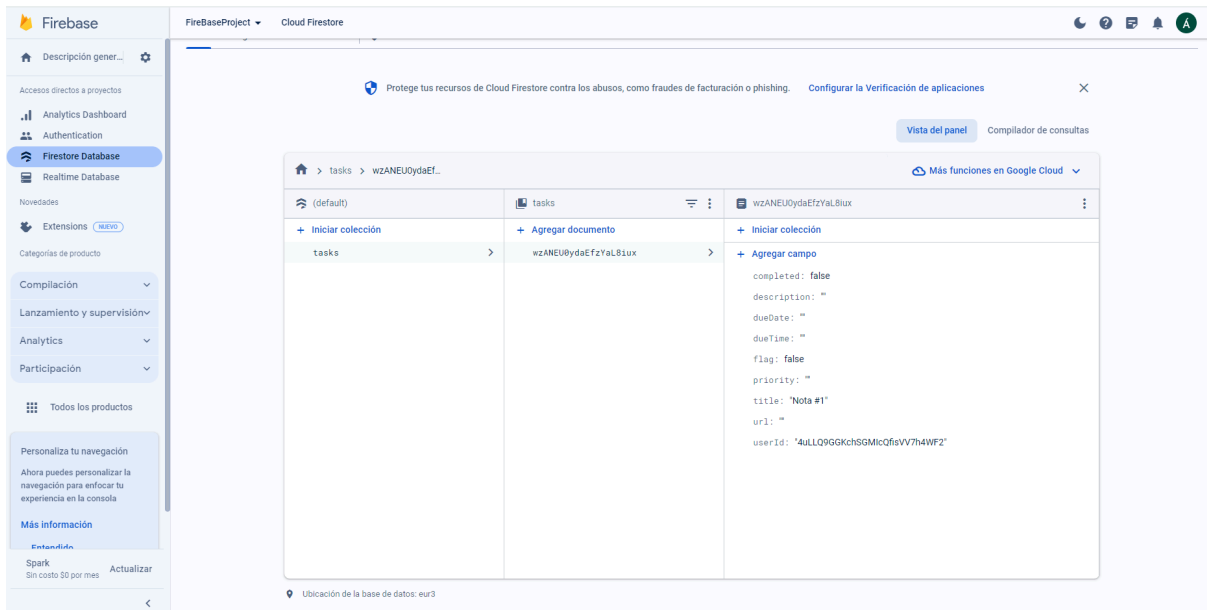
val tasks = viewModel
    .tasks
    .collectAsStateWithLifecycle(emptyList())

LazyColumn { this: LazyListScope
    items(tasks.value, key = { it.id }) { this: LazyItemScope taskItem ->
        TaskItem(
            task = taskItem,
            options = listOf(),
            onCheckChange = { onTaskCheckChange(taskItem) },
            onActionClick = { action -> onTaskActionClick(openScreen, taskItem, action) }
        )
    }
}

```

Ya podremos añadir notas y ver los documentos en la base de datos de Firebase.





## Rendimiento

Para monitorizar el rendimiento tendremos que crear una trace y asignarla a un proceso.

```
inline fun <T> trace(name: String, block: Trace.() -> T): T = Trace.create(name).trace(block)

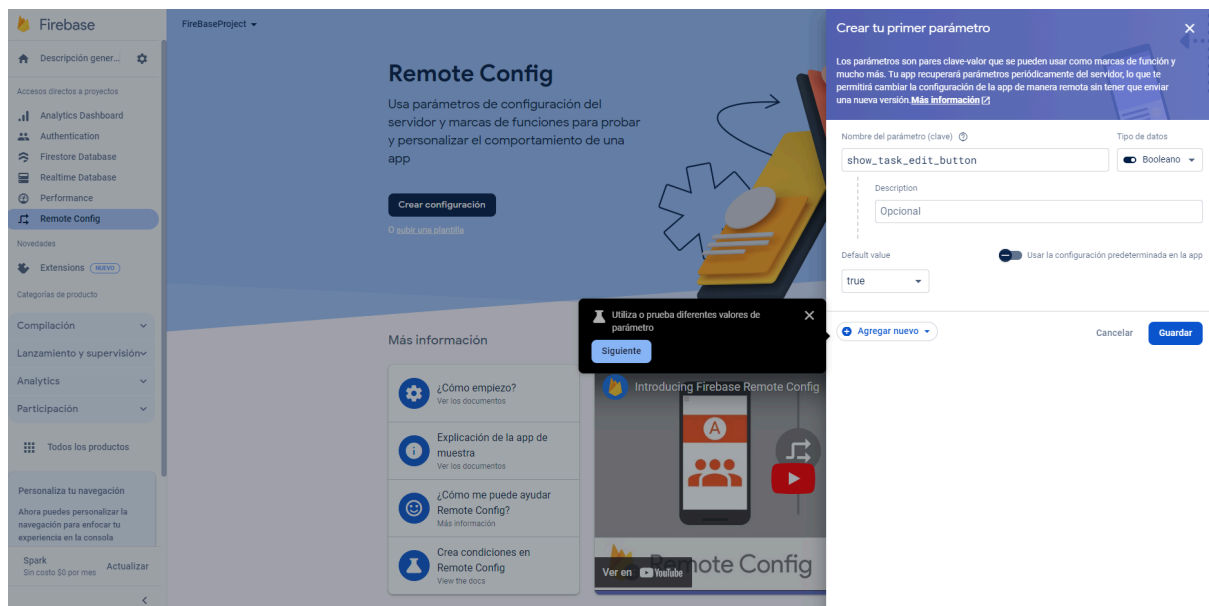
override suspend fun linkAccount(email: String, password: String): Unit =
    trace(LINK_ACCOUNT_TRACE) { this: Trace
        val credential = EmailAuthProvider.getCredential(email, password)
        auth.currentUser!!.linkWithCredential(credential).await()
    }
```

Nos meteremos en el apartado de rendimiento en Firebase y podremos ver los seguimientos personalizados



## Configuración remota

Primero iremos a Firebase y entraremos en el apartado de configuración remota.



Crearemos la configuración e iremos a Android Studio.

```
new ""

override suspend fun fetchConfiguration(): Boolean {
    return remoteConfig.fetchAndActivate().await()
}

Vladimir Kryachko *
override val isShowTaskEditButtonConfig: Boolean
    get() = remoteConfig[SHOW_TASK_EDIT_BUTTON_KEY].asBoolean()

fun loadTaskOptions() {
    val hasEditOption = configurationService.isShowTaskEditButtonConfig
    options.value = TaskActionOption.getOptions(hasEditOption)
}
```

```

val options by viewModel.options
LazyColumn { this: LazyListScope
    items(tasks.value, key = { it.id }) { this: LazyItemScope taskItem ->
        TaskItem(
            task = taskItem,
            options = options,
            onCheckChange = { onTaskCheckChange(taskItem) },
            onActionClick = { action -> onTaskActionClick(openScreen, taskItem, action) }
        )
    }
}
}

```

Ya habríamos terminado nuestra aplicación.