

ANDROID DEVELOPMENT – S3

LISBON, FEBRUARY 2025

ERASMUS+



Bună ziua

DATABASES



The screenshot shows a web application window titled "Lista de Alunos". It contains a table with the following columns: ID, Nome Próprio, Apelido, Endereço de Correio Electrónico, and Telefone da Empresa. The first row is highlighted in blue and contains the value "p" in the "Apelido" column. A red box highlights the "Total" row in the "ID" column.

ID	Nome Próprio	Apelido	Endereço de Correio Electrónico	Telefone da Empresa
(Novo)		p		
Total				

DATABASES

Firebase Cloud Firestore



Cloud Firestore

Android Studio: Tools » Firebase » Cloud Firestore

Assistant

Firebase

← Firebase › Cloud Firestore

Get started with Cloud Firestore [Java]

Cloud Firestore is a flexible, scalable database from Firebase and Google Cloud. It keeps your data in sync across client apps through realtime listeners and offers offline support so you can build responsive apps that work regardless of network latency or internet connectivity.

[Launch in browser](#)

- 1 Connect your app to Firebase

Connect to Firebase
- 2 Add Cloud Firestore to your app

Add the Cloud Firestore SDK to your app

NOTE: After adding the SDK, here are some other helpful configurations to consider:

- Do you want an easier way to manage library versions?
You can use the [Firebase Android BoM](#) to manage your Firebase library versions and ensure that your app is always using compatible library versions.

To use Cloud Firestore, you need to create the database in the [Firebase console](#).

✓ **Cloud Firestore**

Store and sync your app data with this flexible, scalable NoSQL cloud-hosted database. [More info.](#)

📺 [Get started with Cloud Firestore](#)

📺 [Get started with Cloud Firestore \[Java\]](#)



***Your Firebase Android app
has been created in Firebase.***

It's ready for you to connect to your Android Studio project!

Connect

CLOUD FIRESTORE DATABASE

Cloud Firestore

Realtime updates, powerful queries, and automatic scaling

Create database

Ask Gemini



Create database

1 Set name and location — 2 Secure rules

Database ID

(default)

Location

eur3 (Europe)

i Your location setting is where your Cloud Firestore data will be stored

Set name and location — 2 Secure rules

After you define your data structure, you will need to write rules to secure your data.

[Learn more](#)

☐ Start in **production mode**

Your data is private by default. Client read/write access will only be granted as specified by your security rules.

☒ Start in **test mode**

Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if
        request.time < timestamp.date(2025, 2, 21);
    }
  }
}
```

! The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days

Cancel

Create



 > users > pferraz@live.co... More in Google Cloud 		
 (default)	 users  	 pferraz@live.com 
+ Start collection	+ Add document	+ Start collection
users >	pferraz@live.com >	+ Add field age: "48" name: "Pedro"

Conect to DB

- FirebaseFirestore db =
Firestore.getInstance();

Query (see assistant)

```
db.collection(collectionPath: "users") CollectionReference
    .get() Task<QuerySnapshot>
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override no usages
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : task.getResult()) {
                    Log.d(TAG, msg: document.getId() + " => " + document.getData());
                }
            } else {
                Log.w(TAG, msg: "Error getting documents.", task.getException());
            }
        }
    });
```

Queries

```
db.collection( collectionPath: "users") CollectionReference
    .whereEqualTo( field: "nome", value: "Pedro") Query
    .get() Task<QuerySnapshot>
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override no usages
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : task.getResult()) {
                    Log.d(TAG, msg: document.getId() + " => " + document.getData());
                }
            } else {
                Log.d(TAG, msg: "Error getting documents: ", task.getException());
            }
        }
    });
```

Query by ID

```
DocumentReference docRef = db.collection( collectionPath: "users").document( documentPath: "pferraz@live.com");
docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override no usages
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        if (task.isSuccessful()) {
            DocumentSnapshot document = task.getResult();
            if (document.exists()) {
                // Document found
                Log.d( tag: "Firestore", msg: "Document data: " + document.getData());
            } else {
                // No such document
                Log.d( tag: "Firestore", msg: "No such document");
            }
        } else {
            // Task failed
            Log.d( tag: "Firestore", msg: "get failed with ", task.getException());
        }
    }
});
```

Add entry

```
/*add data*/
Map<String, Object> usr = new HashMap<>();
usr.put( k: "name", v: "Pedro");
usr.put( k: "surname", v: "Ferraz");
usr.put( k: "age", v: 48);

db.collection( collectionPath: "Users").document( documentPath: "pedro.ferraz@live.com") DocumentReference
    .set(usr) Task<Void>
    .addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override no usages
        public void onSuccess(Void aVoid) {
            Log.d(TAG, msg: "DocumentSnapshot successfully written!");
        }
    })
    .addOnFailureListener(new OnFailureListener() {
        @Override no usages
        public void onFailure(@NonNull Exception e) {
            Log.w(TAG, msg: "Error writing document", e);
        }
    })
});
```

Get properties

```
HashMap<String, Object> hashMap = (HashMap<String, Object>) document.getData();
Log.d( tag: "Firestore", msg: "Name: " + hashMap.get("name"));
```





MULȚUMESC

THANK YOU

OBRIGADO

SEE YOU TOMORROW!