Formulário - Firestore

```java
/* Variável de ambiente com chave
* GOOGLE_APPLICATION_CREDENTIALS=<pathname do ficheiro json com chave>
*/

GoogleCredentials credentials = GoogleCredentials.getApplicationDefault();

FirestoreOptions options = FirestoreOptions
        .newBuilder()
        .setDatabaseId("db-name")
        .setCredentials(credentials)
        .build();
Firestore db = options.getService();

/*
* Exemplo de classes
*/

public class User {
    public String first;
    public String last;
    public int born;
    public Location location;
}

public class Location {
    public String city;
    public String street;
}

/*
* Inserir/Atualizar com map
*/

CollectionReference colRef = db.collection("Users");
DocumentReference docRef = colRef.document("Bill-Gates");

HashMap<String, Object> map = new HashMap<String, Object>();
map.put("first", "Bill");
map.put("last", "Grates");
map.put("born", 1955);

ApiFuture<WriteResult> result = docRef.set(map); // Overwrites a document
result.get(); // create a new document if it doesn't exist

// update 1 or more fields
map.put("last", "Gates");
result = docRef.update(map); // generates an error if the document doesn't exist
```

```java
result.get();

// update a specific field
result = docRef.update("last", "Gates");
result.get();

/*
* Inserir documentos a partir de objetos
*/

CollectionReference colRef = db.collection("Users");
User user = new User();

user.first = "Bill";
user.last = "Gates";
user.born = 1955;

DocumentReference docRef = colRef.document("DocId - " + user.first + user.last);

ApiFuture<WriteResult> result = docRef.set(user);
result.get();

/*
* Listagem de documentos de uma coleção
*/

CollectionReference colRef = db.collection("Users");
Iterable<DocumentReference> docs = colRef.listDocuments();

for (DocumentReference docRef : docs) {
    ApiFuture<DocumentSnapshot> docFut = docRef.get();
    DocumentSnapshot docSnap = docFut.get();
    // docSnap.getData();
}

/*
* Ler campo ou objeto a partir de um documento
*/

String first = "Bill";
String last = "Gates";

DocumentReference docRef = db.collection("Users").document("DocId - " + first +
last);
ApiFuture<DocumentSnapshot> docFut = docRef.get();
DocumentSnapshot docSnap = docFut.get();

// read a field from a document
String first = docSnap.getString("first");

// read an object from a document
User user = docSnap.toObject(User.class);
```

```java
String first = user.first;

/*
* Apagar campos e documentos
*/

DocumentReference docRef = db.collection("Users").document("DocId - BillGates");

// delete a field
ApiFuture<WriteResult> result = docRef.update("born", FieldValue.delete());
result.get();

// delete a document
ApiFuture<WriteResult> result = docRef.delete();
result.get();

/*
* Query simples
*/

Query query = db.collection("Users").whereEqualTo("born", 1955);
Query query = db.collection("Users").whereGreaterThan("born", 1955);
Query query = db.collection("Users").whereLessThan("born", 1955);

ApiFuture<QuerySnapshot> querySnap = query.get();

// iterate over the results
for (DocumentSnapshot doc : querySnap.get().getDocuments()) {
    // doc.getString("first");
}

/*
* Interrogações simples de campos complexos
*/

FieldPath fp = FieldPath.of("location", "city");
Query query = db.collection("Users").whereEqualTo(fp, "Lisboa");
ApiFuture<QuerySnapshot> querySnap = query.get();

/*
* Interrogações compostas com índice composto
*/

FieldPath fp = FieldPath.of("location", "city");

Query query = db.collection("Users")
    .whereLessThan("born", 1955)
    .whereEqualTo(fp, "Lisboa");

ApiFuture<QuerySnapshot> querySnap = query.get();
```