FireBase (II)

Insertar Documento desde tu App

Cuando insertamos un documento en Firebase, podemos hacerlo de dos formas:

- Dejando que Firebase autogenere la ID
- Dándole la ID nosotros.

Para darle nosotros la ID, basta con hacer:

```
val enterprise = Enterprise( id = "D3",  nombre = "Softcom",  localizacion = "Vitoria")
db.collection( collectionPath = "EmpresaBD")
    .document( documentPath = enterprise.id)
    .set(enterprise)
    .addOnSuccessListener { result ->
        Log.d( tag = "Firestore",  msg = "Enterprise added: $enterprise")
    }
    .addOnFailureListener { exception ->
        Log.e( tag = "Firestore",  msg = "Error when inserting enterprises",  tr = exception)
    }
```

En este ejemplo extraemos la id del POJO Enterprise. Esto es por simplicidad, pero en realidad provoca que el Documento en Firebase aparezca con un campo extra ID. Esto se debe a que, al fin y al cabo, Enterprise también tiene un atributo ID.

Por tanto, recuerda que las ID de Firebase deberían de tratarse por separado del resto de los campos del Documento.

Para dejar la autogeneración, basta con hacer:

```
val enterprise = Enterprise( id = "D3",  nombre = "Softcom",  localizacion = "Vitoria")
db.collection( collectionPath = "EmpresaBD")
    .add(enterprise)
    .addOnSuccessListener { result ->
        Log.d( tag = "Firestore",  msg = "Enterprise added: $enterprise")
    }
    .addOnFailureListener { exception ->
        Log.e( tag = "Firestore",  msg = "Error when inserting enterprises",  tr = exception)
    }
```

Eliminar Documento desde tu App

Cuando eliminamos un documento en Firebase, podemos hacerlo de dos formas:

- Eliminándolo directamente por su ID
- Realizando una búsqueda del Documento por otro criterio, y borrándolo entonces.

Para hacerlo por ID, basta con hacer:

```
val id = "D3"

db.collection( collectionPath = "EmpresaBD")
    .document( documentPath = id)
    .delete()
    .addOnSuccessListener { result ->
        Log.d( tag = "Firestore", msg = "Enterprise deleted, ID: $id")
    }
    .addOnFailureListener { exception ->
        Log.e( tag = "Firestore", msg = "Error when inserting enterprises", tr = exception)
    }
```

Mientras que para el borrado por otro criterio, hacemos primero una búsqueda:

```
val name = "Softcom"

db.collection( collectionPath = "EmpresaBD")
    .whereEqualTo( field = "nombre", value = name)
    .get()
    .addOnSuccessListener { querySnapshot ->
        for (document in querySnapshot.documents) {
            document.reference.delete()
                .addOnSuccessListener {
                    Log.d( tag = "Firestore", msg = "Enterprise deleted: $name")
                }.addOnFailureListener { exception ->
                    Log.e( tag = "Firestore", msg = "Error when inserting enterprises", tr = exception)
                }
        }
    }.addOnFailureListener { exception ->
        Log.e( tag = "Firestore", msg = "Error when inserting enterprises", tr = exception)
    }
```

Recuerda: un delete () ordena un borrado, pero no hay forma de saber si ha borrado o no. Es por esto que primero se hace una búsqueda del documento a borrar y si existe, se borra.