

Firebase



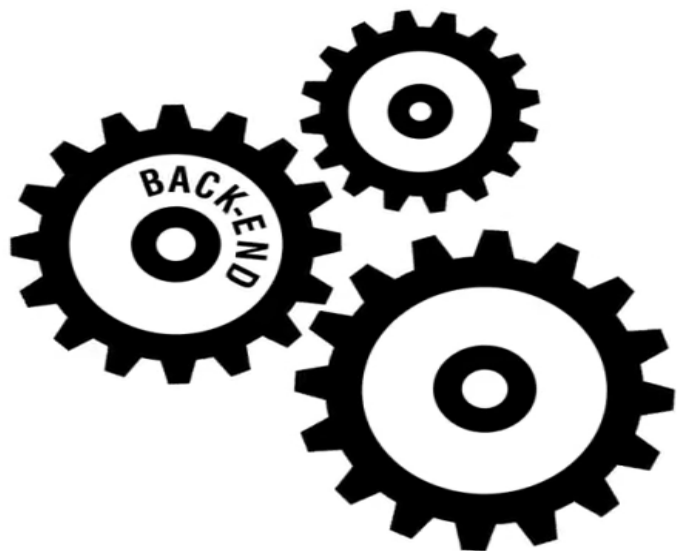
Plateforme Firebase – Pourquoi

Pourquoi

Supposez que l'on voudrait créer une application interactive en temps réel
Exemple application de chat



Plateforme Firebase – Pourquoi



Database
API/Http Requests
API/Authentication
Analytics
Bug Report
Push notification
Chat
. . .

Plateforme Firebase – Pourquoi

- Cela coute beaucoup d'argent, et de temps
- Beaucoup d'effort de maintenance
- Contre productif quand on veut créer une application rapidement
- Etc.



Plateforme Firebase – Pourquoi

- Plateforme google qui nous permet de créer des applications « scalables » (capables d'être mises à l'échelle)
- BDD déjà construite et prête à être utilisée
- API d'authentification facile à utiliser
- Offre l'analyse sémantique
- Offre les rapports de bug
- Tout ce qu'il faut pour avoir une infrastructure solide pour construire des applications avec le moindre coût possible



Database
API/Http Requests
API/Authentication
Analytics
Bug Report
Push notification
Chat
...

Plateforme Firebase – Pourquoi

- Plateforme google qui nous permet de créer des applications « scalables » (capables d'être mises à l'échelle)
- BDD déjà construite et prête à être utilisée
- API d'authentification facile à utiliser
- Offre l'analyse sémantique
- Offre les rapports de bug
- Tout ce qu'il faut pour avoir une infrastructure solide pour construire des applications avec le moindre coût possible



Database
API/Http Requests
API/Authentication
Analytics
Bug Report
Push notification
Chat
...

Plateforme Firebase – Pourquoi

Develop & test your app



Realtime Database



Cloud Functions



Cloud Firestore



Authentication



Hosting



Cloud storage



Performance Monitoring



Test Lab for Android

Plateforme Firebase – Pourquoi

Grow & engage your audience



Google Analytics



Cloud Messaging



Predictions



Dynamic Links



Remote Config



Invites



App Indexing



AdMob

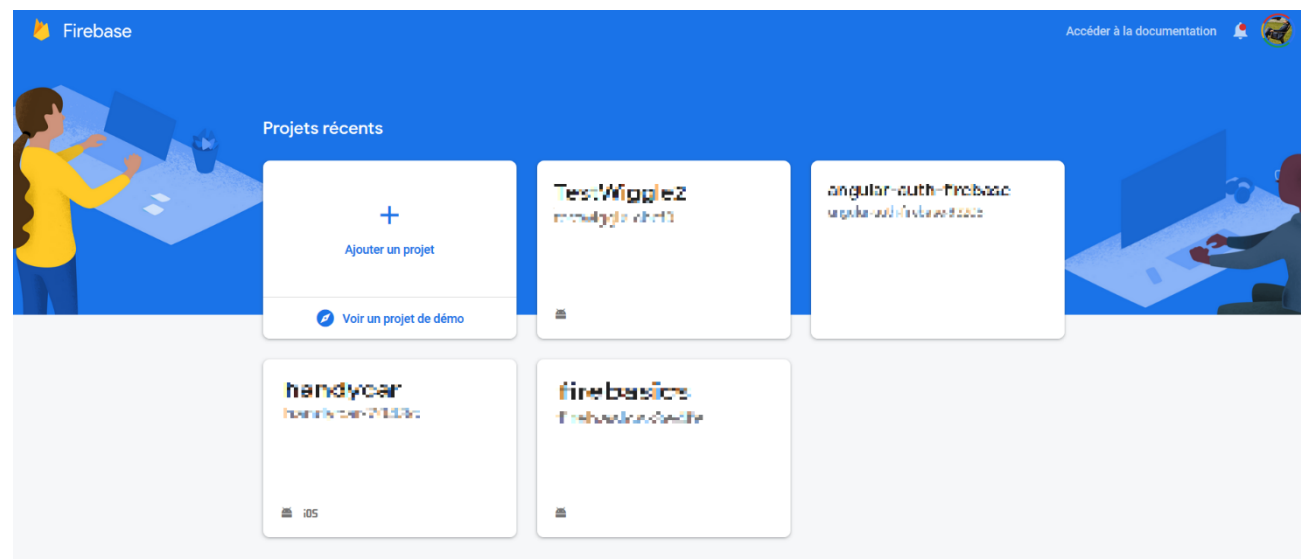
Plateforme Firebase – Pourquoi

Facile à intégrer sur



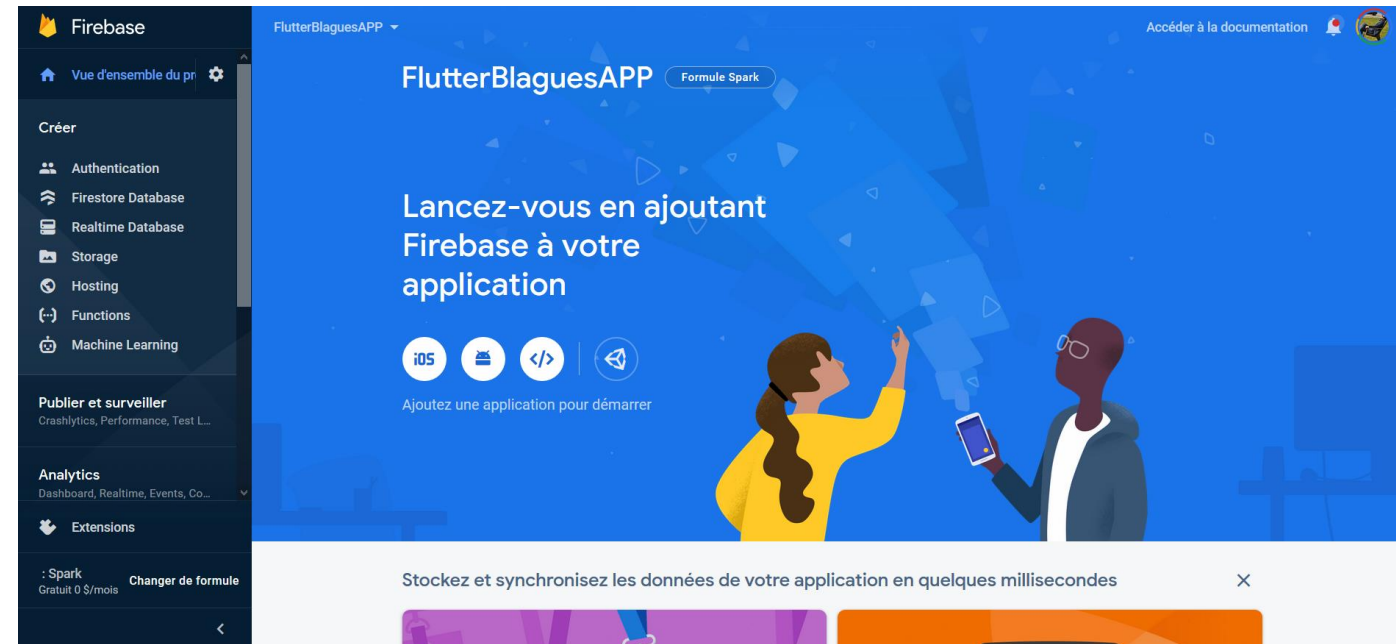
Plateforme Firebase – mise en place de Firestore dans un projet

1. Avoir un compte google
2. Se connecter à Firebase :
<https://firebase.google.com/>
3. Cliquez sur le bouton « get started »
4. Ajouter un projet



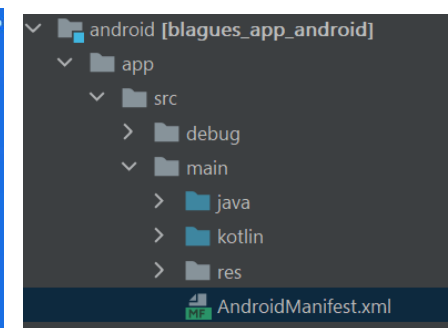
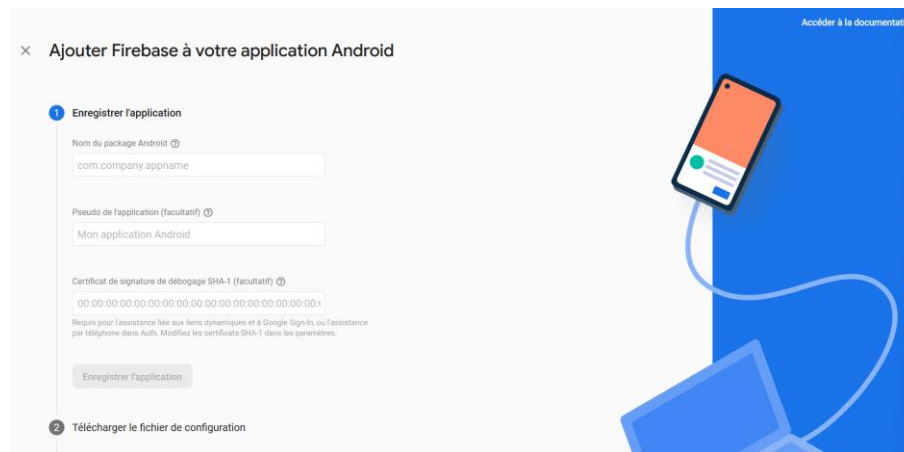
Plateforme Firebase – mise en place de Firestore dans un projet

1. Avoir un compte google
2. Se connecter à Firebase :
<https://firebase.google.com/>
3. Cliquez sur le bouton « get started »
4. Ajouter un projet

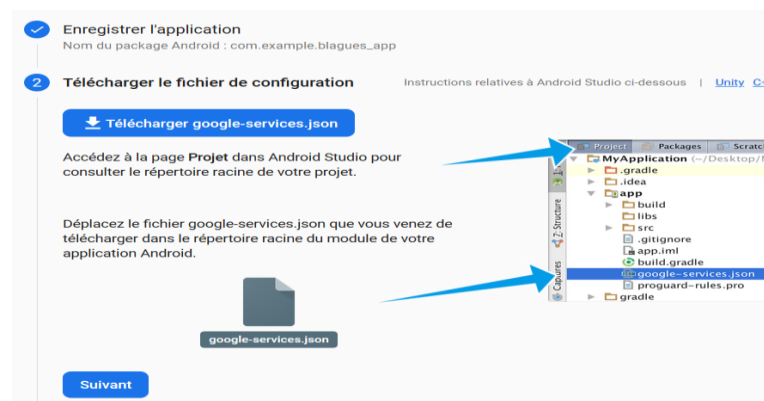


Plateforme Firebase – mise en place de Firestore dans un projet

1. Avoir un compte google
2. Se connecter à Firebase :
<https://firebase.google.com/>
3. Cliquez sur le bouton « get started »
4. Ajouter un projet
5. Ajouter Firebase à l'application Android

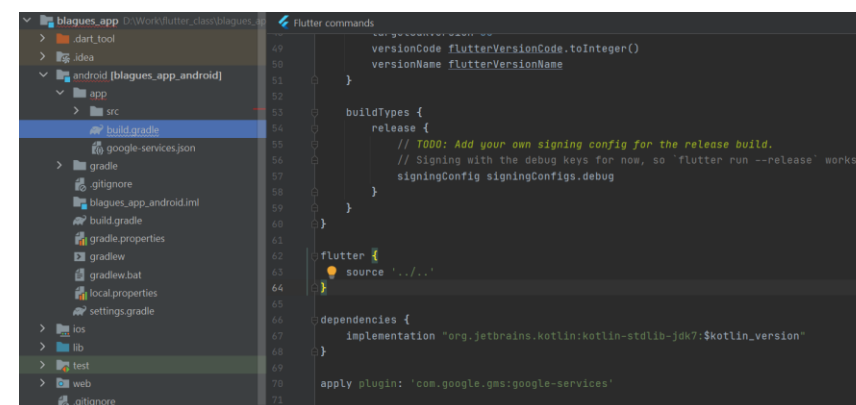
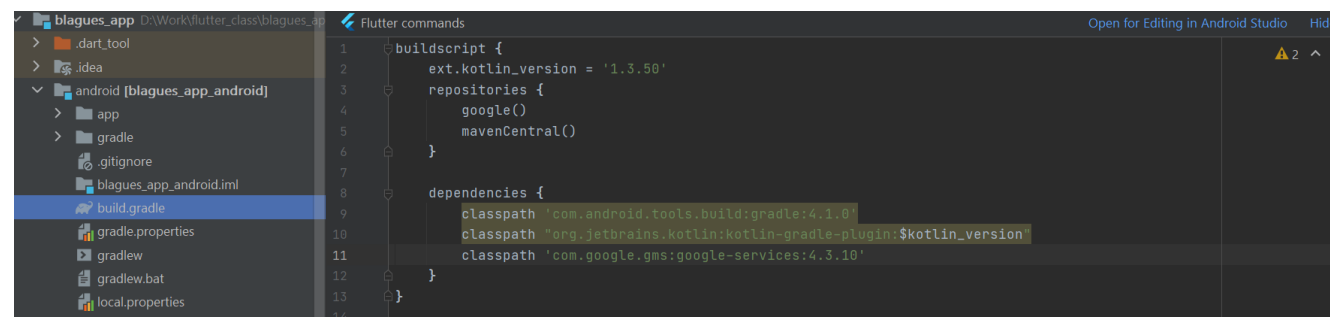
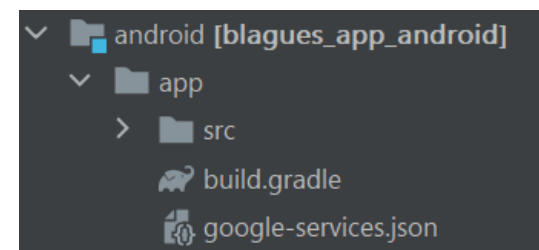


```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.blagues_app">
```



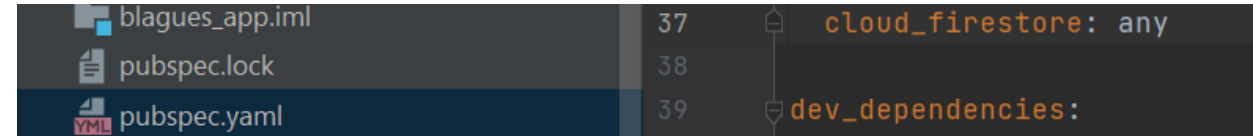
Plateforme Firebase – mise en place de Firestore dans un projet

1. Avoir un compte google
2. Se connecter à Firebase :
<https://firebase.google.com/>
3. Cliquez sur le bouton « get started »
4. Ajouter un projet
5. Ajouter Firebase à l'application Android



Plateforme Firebase – mise en place de Firestore dans un projet

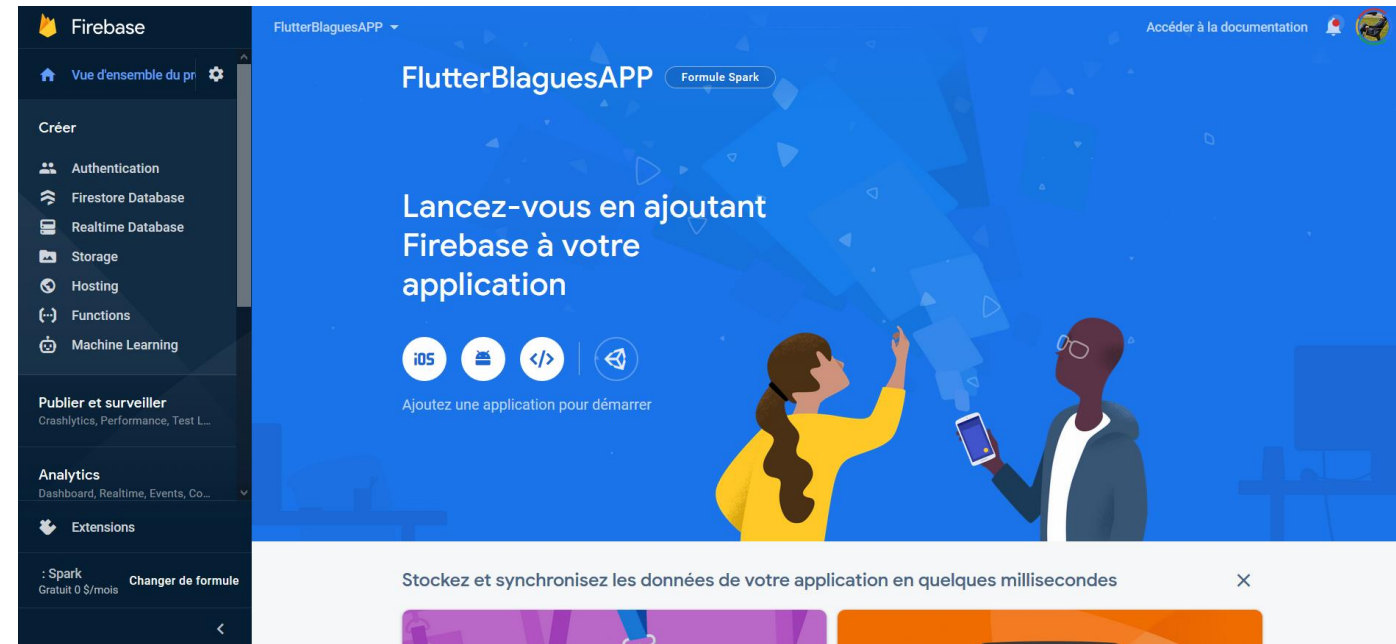
1. Ajouter la dépendance `cloud_firestore` dans Le `pubspec.yaml`



```
blagues_app.iml 37 cloud_firestore: any
pubspec.lock      38
pubspec.yaml     39 dev_dependencies:
```

Plateforme Firebase – Création d'une base de données

1. Choisir la zone du serveur Firestore



Plateforme Firebase – Création d'une base de données

1. Choisir la zone du serveur Firestore

Créer une base de données

✓ Règles de sécurité pour Cloud Firestore

2 Définir l'emplacement Cloud Firestore

Le paramètre d'emplacement définit l'endroit où sont stockées vos données Cloud Firestore.

⚠ Une fois défini, cet emplacement ne peut plus être modifié. Ce paramètre s'appliquera également à votre bucket Cloud Storage par défaut.

En savoir plus

Zone Cloud Firestore

nam5 (us-central)

Si vous activez Cloud Firestore, vous ne pourrez pas utiliser Cloud Datastore avec ce projet, notamment depuis l'application App Engine associée

Annuler Activer

Plateforme Firebase – lecture des données de la base de données firestore

1. Pour commencer à utiliser le package Cloud Firestore dans votre projet, importez-le en haut de vos fichiers de projet

```
import 'package:cloud_firestore/cloud_firestore.dart';
```

2. Pour créer une nouvelle instance Firestore, appelez le getter d'instance sur FirebaseFirestore :

```
FirebaseFirestore firestore = FirebaseFirestore.instance;
```

Plateforme Firebase – lecture des données de la base de données firestore: Collections & Documents

1. Cloud Firestore est une base de données NoSQL orientée document
2. Les données sont stockées dans les *documents*, qui sont organisés en *collections*
 - Chaque *document* contient un ensemble de paires clé / valeur
 - Tous les documents doivent être conservés dans des collections
 - Les documents peuvent contenir des objets imbriqués et sous – *collections*
3. Un document représentant un utilisateur *alovelace* pourrait ressembler à ceci



 alovelace

```
first : "Ada"
```

```
last : "Lovelace"
```

```
born : 1815
```

4. Les objets complexes et imbriqués dans un document sont appelés cartes

 alovelace

```
name :
```

```
first : "Ada"
```

```
last : "Lovelace"
```

```
born : 1815
```

Plateforme Firebase – lecture des données de la base de données firestore: Collections & Documents

5. On peut avoir une collection *utilisateurs* pour contenir les différents utilisateurs, chacun représenté par un document

```
utilisateurs
├── alovelace
│   ├── first : "Ada"
│   ├── last  : "Lovelace"
│   └── born   : 1815
└── aturing
    ├── first : "Alan"
    ├── last  : "Turing"
    └── born   : 1912
```

6. Une collection ne peut pas contenir directement des champs bruts avec des valeurs, et ne peut pas contenir d'autres collections
7. Chaque document dans Cloud Firestore est identifié de manière unique par son emplacement dans la base de données
8. Pour faire référence à cet emplacement dans votre code, vous pouvez créer une *référence*

```
DocumentReference alovelaceDocumentRef = db.collection("users").document("alovelace");
DocumentReference alovelaceDocumentRef = db.document("users/alovelace");
```

Plateforme Firebase – lecture des données de la base de données firestore: Collections & Documents

9. Vous pouvez également créer des références à des *collections*

```
CollectionReference usersCollectionRef = db.collection("users");
```

10. Exemples

chambres

ROOMA

name : "my chat room"

messages

message1

from : "alex"

msg : "Hello World!"

message2

...

roomB

...

```
DocumentReference messageRef = db
    .collection("rooms").document("roomA")
    .collection("messages").document("message1");
```

Plateforme Firebase – lecture des données de la base de données firestore: Collections & Documents

11. Pour lire une collection ou un document une fois, appelez les méthodes *Query.get* ou *DocumentReference.get*

```
@override
Widget build(BuildContext context) {
  CollectionReference users = FirebaseFirestore.instance.collection('users');

  return FutureBuilder<DocumentSnapshot>(
    future: users.doc(documentId).get(),
    builder:
      (BuildContext context, AsyncSnapshot<DocumentSnapshot> snapshot) {

        if (snapshot.hasError) {
          return Text("Something went wrong");
        }

        if (snapshot.hasData && !snapshot.data!.exists) {
          return Text("Document does not exist");
        }

        if (snapshot.connectionState == ConnectionState.done) {
          Map<String, dynamic> data = snapshot.data!.data() as Map<String, dynamic>;
          return Text("Full Name: ${data['full_name']} ${data['last_name']}");
        }

        return Text("loading");
      },
  );
}
```

Plateforme Firebase – lecture des données de la base de données firestore: Collections & Documents

12. Pour créer un document, appelez le méthode add

```
// Create a CollectionReference called users that references the firestore collection  
CollectionReference users = FirebaseFirestore.instance.collection('users');
```

```
Future<void> addUser() {  
    // Call the user's CollectionReference to add a new user  
    return users  
        .add({  
            'full_name': fullName, // John Doe  
            'company': company, // Stokes and Sons  
            'age': age // 42  
        })  
        .then((value) => print("User Added"))  
        .catchError((error) => print("Failed to add user: $error"));  
}
```

Plateforme Firebase – lecture des données de la base de données firestore: Collections & Documents

13. *CollectionReference* et *DocumentReference* fournissent une méthode *snapshots()* qui renvoie un *Stream* :

```
Stream collectionStream = FirebaseFirestore.instance.collection('users').snapshots();
Stream documentStream = FirebaseFirestore.instance.collection('users').doc('ABC123').snapshots();
```

14. Une fois renvoyé, vous pouvez vous abonner aux mises à jour via la méthode *listen()*

```
class _UserInformationState extends State<UserInformation> {
  final Stream<QuerySnapshot> _usersStream = FirebaseFirestore.instance.collection('users').snapshots();

  @override
  Widget build(BuildContext context) {
    return StreamBuilder<QuerySnapshot>(
      stream: _usersStream,
      builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
        if (snapshot.hasError) {
          return Text('Something went wrong');
        }

        if (snapshot.connectionState == ConnectionState.waiting) {
          return Text("Loading");
        }

        return ListView(
          children: snapshot.data!.docs.map((DocumentSnapshot document) {
            Map<String, dynamic> data = document.data()! as Map<String, dynamic>;
            return ListTile(
              title: Text(data['full_name']),
              subtitle: Text(data['company']),
            );
          }).toList(),
        );
      },
    );
  }
}
```

```
FirebaseFirestore.instance
  .collection('users')
  .snapshots(includeMetadataChanges: true)
```

Plateforme Firebase – lecture des données de la base de données firestore: Collections & Documents

13. Lors de l'exécution d'une requête, Firestore renvoie soit un *QuerySnapshot*, soit un *DocumentSnapshot*.
13. Un *QuerySnapshot* est renvoyé à partir d'une requête de collection et vous permet d'inspecter la collection, par exemple le nombre de documents qu'elle contient, donne accès aux documents de la collection, affiche les modifications depuis la dernière requête et plus encore.

```
FirestoreFirestore.instance
  .collection('users')
  .get()
  .then((QuerySnapshot querySnapshot) {
    querySnapshot.docs.forEach((doc) {
      print(doc["first_name"]);
    });
  });
```


Plateforme Firebase – lecture des données de la base de données firestore: Collections & Documents

13. Un *DocumentSnapshot* est renvoyé à partir d'une requête ou en accédant directement au document. Même si aucun document n'existe dans la base de données, un instantané sera toujours renvoyé
- Pour déterminer si le document existe, utilisez la propriété *exists*
 - Si le document existe, vous pouvez en lire les données en appelant la méthode *data*, qui renvoie un `Map<String, dynamic>`, ou null s'il n'existe pas

```
Firestore.instance
  .collection('users')
  .doc(userId)
  .get()
  .then((DocumentSnapshot documentSnapshot) {
    if (documentSnapshot.exists) {
      print('Document exists on the database');
    }
  });
```

```
Firestore.instance
  .collection('users')
  .doc(userId)
  .get()
  .then((DocumentSnapshot documentSnapshot) {
    if (documentSnapshot.exists) {
      print('Document data: ${documentSnapshot.data()}');
    } else {
      print('Document does not exist on the database');
    }
  });
```

Plateforme Firebase – lecture des données de la base de données firestore: Collections & Documents

13. Cloud Firestore offre des fonctionnalités avancées pour interroger les collections. Les requêtes fonctionnent à la fois avec des lectures ponctuelles ou des abonnements aux modifications

Filtration

```
Firestore.instance
  .collection('users')
  .where('age', isGreaterThan: 20)
  .get()
  .then(...);
```

```
Firestore.instance
  .collection('users')
  .where('language', arrayContainsAny: ['en', 'it'])
  .get()
  .then(...);
```

Limitation

```
Firestore.instance
  .collection('users')
  .limit(2)
  .get()
  .then(...);
```

```
Firestore.instance
  .collection('users')
  .orderBy('age')
  .limitToLast(2)
  .get()
  .then(...);
```

Ordonner

```
Firestore.instance
  .collection('users')
  .orderBy('age', descending: true)
  .get()
  .then(...);
```

Curseurs de début et de fin

```
Firestore.instance
  .collection('users')
  .orderBy('age')
  .orderBy('company')
  .startAt([4, 'Alphabet Inc.'])
  .endAt([21, 'Google LLC'])
  .get()
  .then(...);
```