

Firestore & NoSQL

Développement d'applications mobiles

Daniel Schreurs

SQL : Connecting Data

Ce qu'on souhaite modéliser



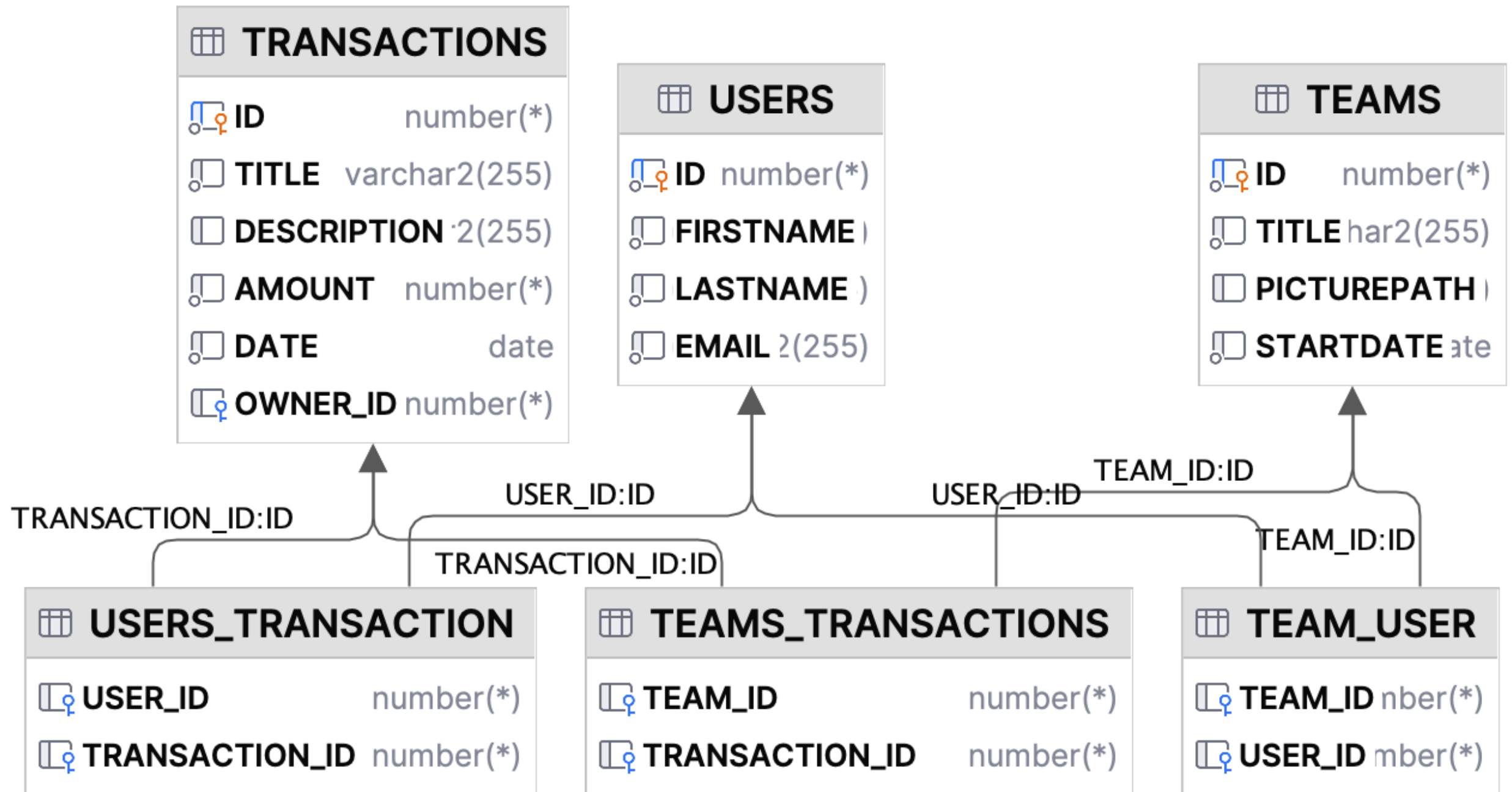
- Les utilisateurs sont membres de groupes
- Chaque groupe est constitué d'utilisateurs
- Au sein d'un groupe, ajouter des transactions :
- Des montants
- Qui concernent d'autres utilisateurs du même groupe



Petit rappel

- **Première Forme Normale (1FN)**
 - Élimination des valeurs multiples.
 - Chaque attribut contient une seule valeur.
- **Deuxième Forme Normale (2FN)**
 - Respect de la 1FN.
 - Tous les attributs non-clés dépendent de la clé primaire.
- **Troisième Forme Normale (3FN)**
 - Respect de la 2FN.
 - Élimination des dépendances transitives.
- **Forme Normale de Boyce-Codd (BCNF)**
 - Respect de la 3FN.
 - Clés candidates déterminent tous les attributs non-clés.

Schéma de la base de données



Requête SQL





Join'ing creates Clarity!



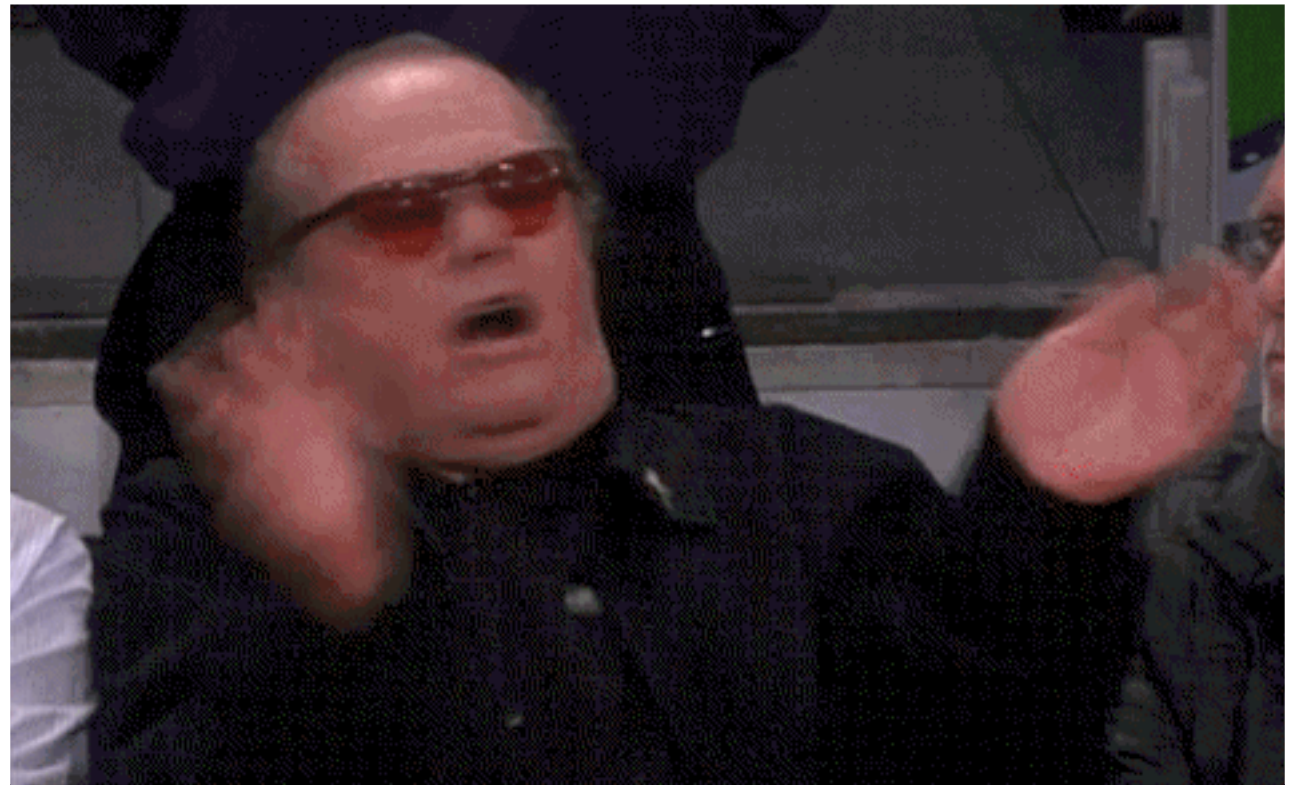
```
select FIRSTNAME, LASTNAME, TITLE  
from USERS  
inner join USER_DAM.TEAM_USER TU on USERS.ID = TU.USER_ID  
inner join USER_DAM.TEAMS T on T.ID = TU.TEAM_ID;
```

Requête SQL

Résultat

	 FIRSTNAME		 LASTNAME		 TITLE	
1	Ada		Lovelace		 Code Magicians	
2	Alan		Turing		 Code Magicians	
3	Grace		Hopper		 Code Magicians	
4	Linus		Torvalds		 Web Weavers	
5	Margaret		Hamilton		 Web Weavers	
6	Tim		Berners-Lee		 Web Weavers	
7	Ada		Lovelace		 Game Gurus	
8	Grace		Hopper		 Game Gurus	
9	Margaret		Hamilton		 Game Gurus	
10	Ada		Lovelace		 nouveau groupe	
11	Grace		Hopper		 nouveau groupe	
12	Margaret		Hamilton		 nouveau groupe	

NoSQL 🧘



Points forts

- Flexibilité
- Évolutivité
 - Load balancing
 - Changer la structure en cours de route 🤯
- Données non structurées
 - Lectures très rapides
 - Requêtes simplistes !

Points faibles

Tout se paye...

- La mise à jour des données
 - Il faut mettre à jour toutes les données en double
 - Risque d'oublier des mises à jour... 🤯
- Redondance des données
- Ce qui paraît simple ne l'est pas au final

Concrètement

Tout va bien

+ Commencer une collection

teams >

users

+ Ajouter un document

1ZPAir3V3ZxdEyTMY0WE >

RHta1A1kGBRfNye1fRd7

aSnY02YqMrzAr51bfRHo

up9NwPKroIkNUIINKXb07

+ Commencer une collection

+ Ajouter un champ

2 "Networking"

title: 🌐 Web Weavers"

transactions

0

amount: 20

concerns

0

email: "alan@example.com"

firstName: "Alan"

lastName: "Turing"

toto: "titi"

1

email: "linus@example.com"

firstName: "Linus"

lastName: "Torvalds"

2

email: "tim@example.com"

firstName: "Tim"

lastName: "Berners-Lee"

date: 21 août 2023 à 00:00:00 UTC+2

description: "Achats préventifs pour la soirée"

owner

email: "tim@example.com"

firstName: "Tim"

lastName: "Berners-Lee"

title: "Pansements Ampoules"

1

amount: 50

(default)

users

alan@example.com

+ Commencer une collection

+ Ajouter un champ

email: "alan@example.com"

firstName: "Alan"

lastName: "Turing"

+ Commencer une collection

teams

users >

+ Ajouter un document

ada@example.com

alan@example.com >

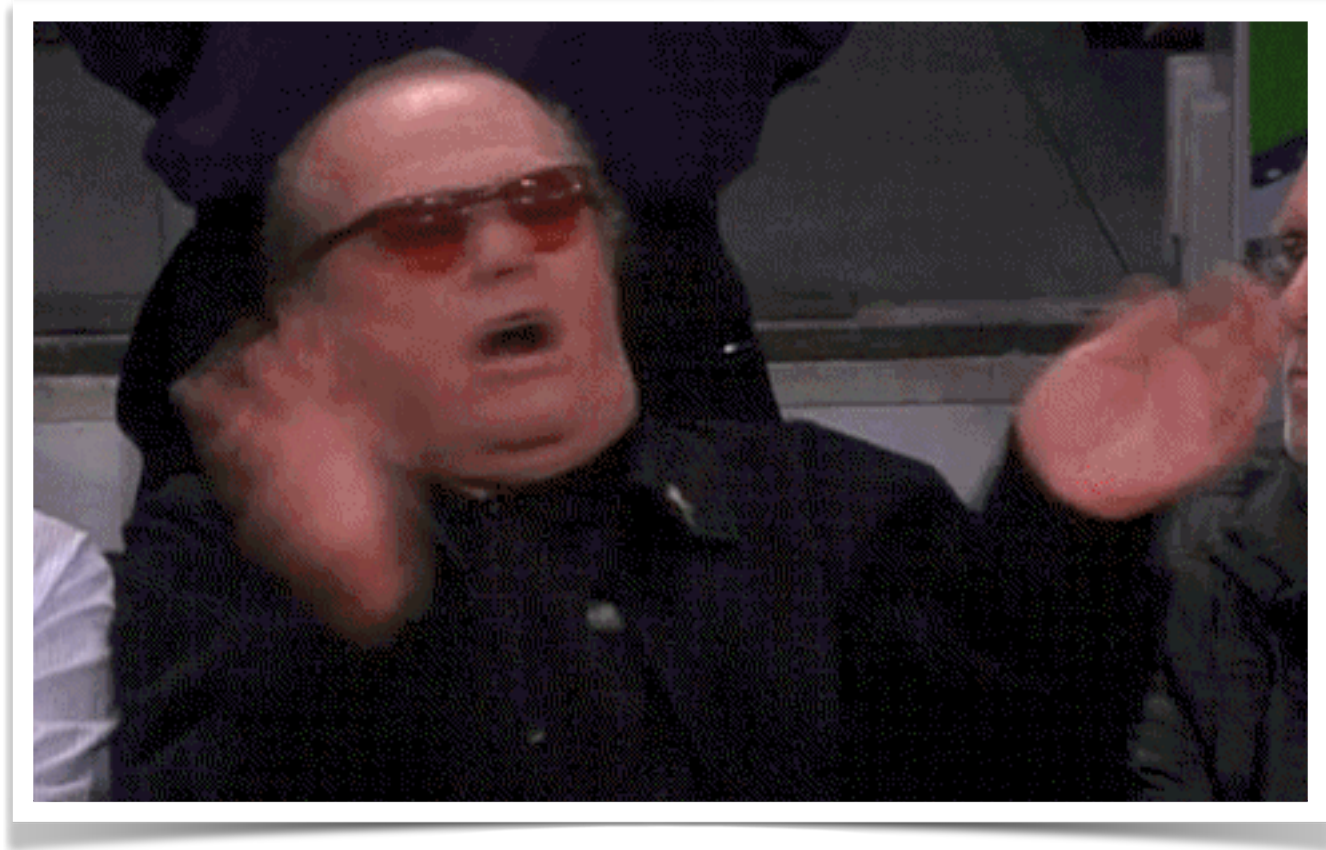
grace@example.com

linus@example.com

margaret@example.com

tim@example.com

Mais c'est quoi ce truc ?

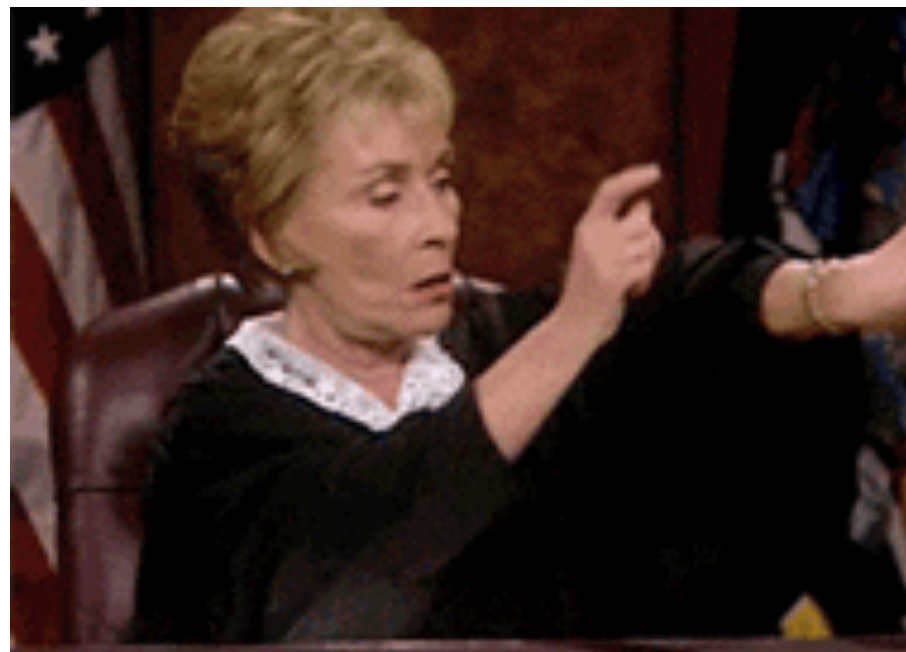
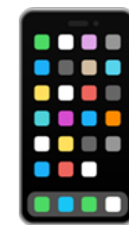


Mais c'est quoi ce truc ?

- Lecture ou mise à jour ?
 - 700 lectures pour 1 mise à jour
- Tant pis pour les mises à jour... mais :
 - Requête beaucoup plus simple
 - Moins de Temps de Latence

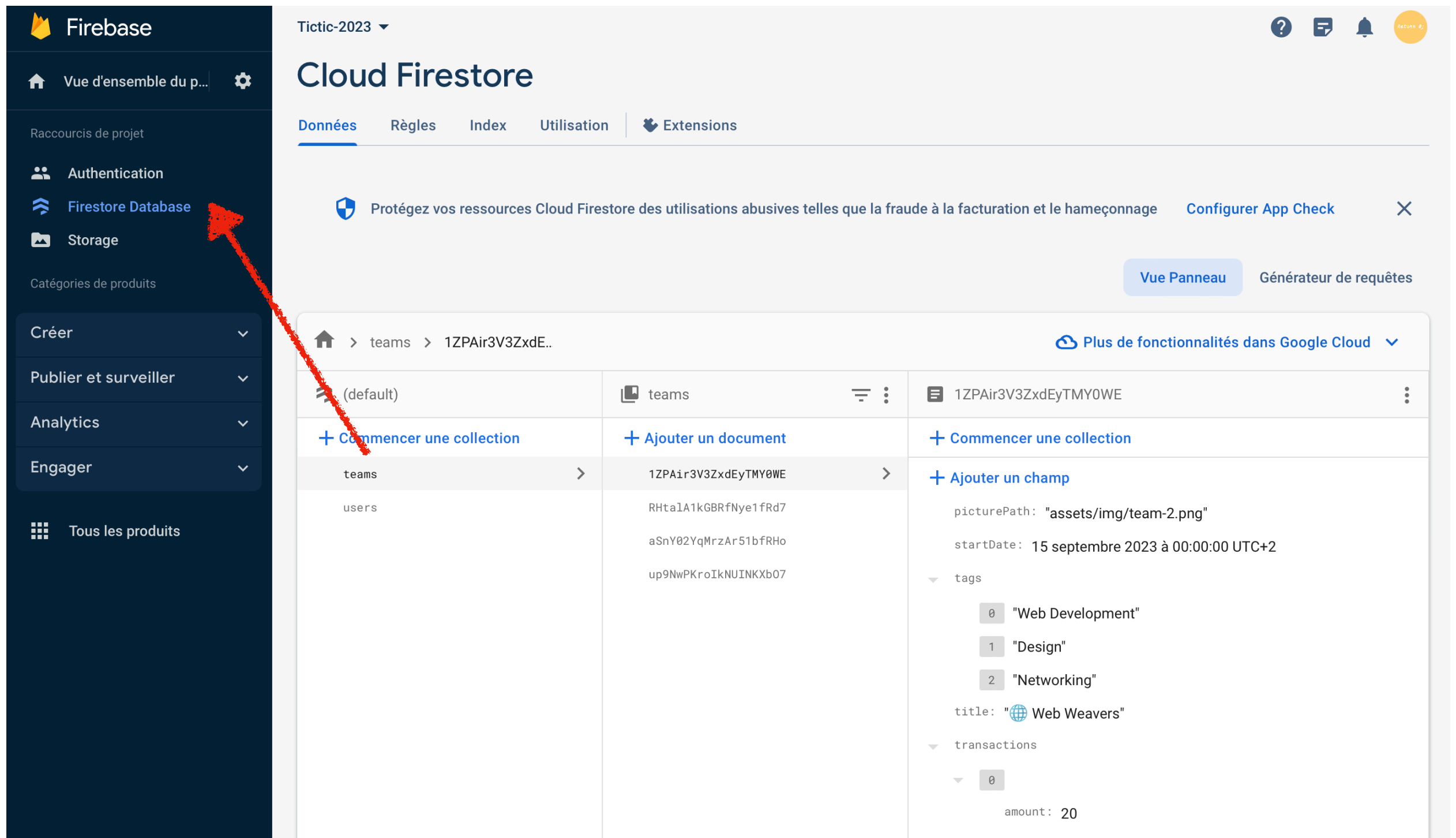


Et donc avec Flutter



Step by step

Activer les modules sur Firebase



The screenshot displays the Firebase console interface for a project named 'Tictic-2023'. The left sidebar contains the following navigation options:

- Home icon: Vue d'ensemble du p...
- Settings icon: [gear]
- Raccourcis de projet
- Authentication
- Firestore Database** (highlighted with a red arrow)
- Storage
- Catégories de produits
- Créer (dropdown)
- Publier et surveiller (dropdown)
- Analytics (dropdown)
- Engager (dropdown)
- Tous les produits

The main content area is titled 'Cloud Firestore' and includes tabs for 'Données', 'Règles', 'Index', 'Utilisation', and 'Extensions'. A notification banner at the top states: 'Protégez vos ressources Cloud Firestore des utilisations abusives telles que la fraude à la facturation et le hameçonnage' with a 'Configurer App Check' link.

Below the notification, there are buttons for 'Vue Panneau' and 'Générateur de requêtes'. The breadcrumb path is 'home > teams > 1ZPAir3V3ZxdE..'. The interface is divided into three columns:

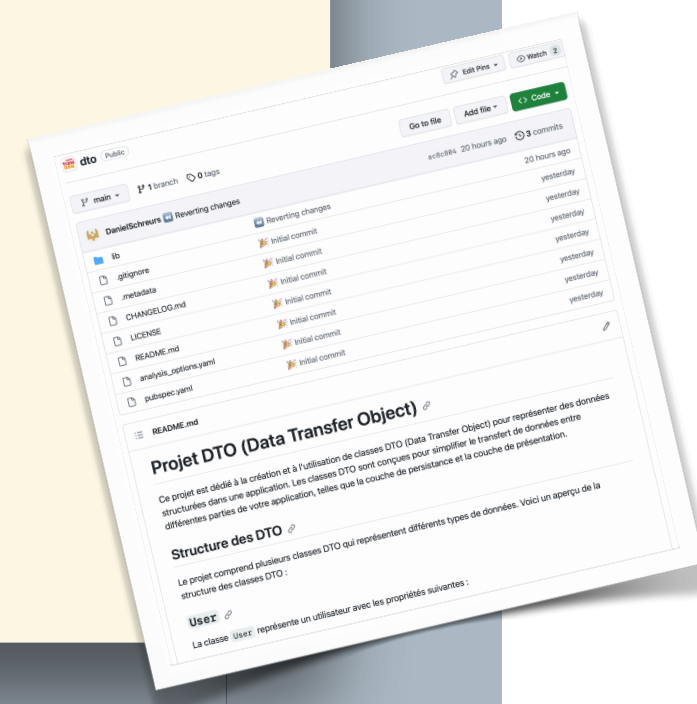
- Left Column:** Shows the '(default)' database with a '+ Commencer une collection' button. Below it, a list of collections includes 'teams' and 'users'.
- Middle Column:** Shows the 'teams' collection with a '+ Ajouter un document' button. Below it, a list of documents includes '1ZPAir3V3ZxdEyTMY0WE' and others with their IDs.
- Right Column:** Shows the document '1ZPAir3V3ZxdEyTMY0WE' with a '+ Commencer une collection' button. Below it, a list of fields includes 'picturePath', 'startDate', 'tags', 'title', and 'transactions'.

The 'tags' field is expanded, showing an array of strings: 'Web Development', 'Design', and 'Networking'. The 'title' field is 'Web Weavers'. The 'transactions' field is expanded, showing an array of objects with 'amount' values.

Step by step

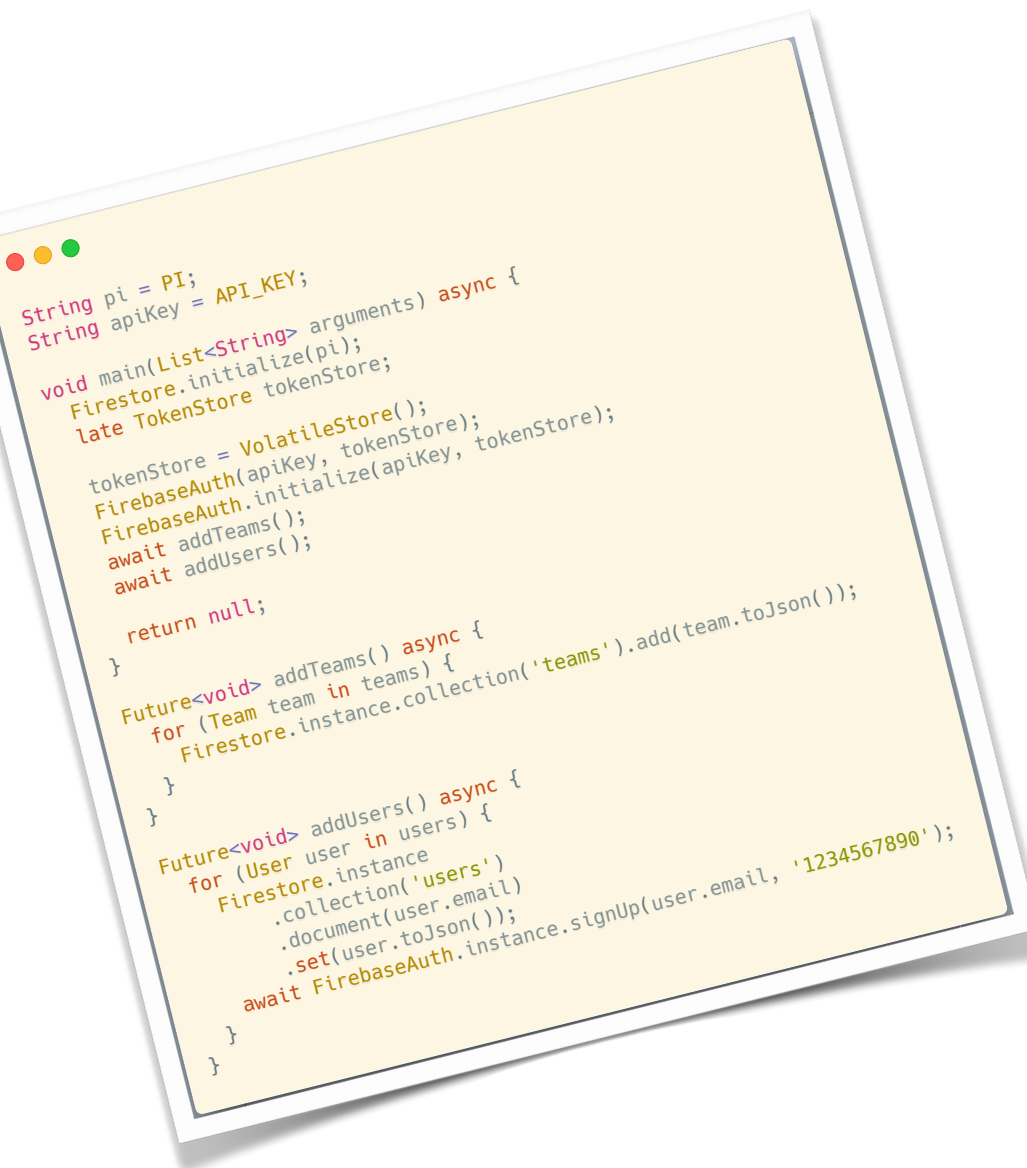
Écrire des classes DTO

```
class User {  
    final String firstName; final String lastName; final String email;  
    const User({  
        required this.firstName, required this.lastName, required this.email,  
    });  
  
    factory User.fromJson(Map<String, dynamic> json) {  
        return User(  
            firstName: json['firstName'],  
            lastName: json['lastName'],  
            email: json['email'],  
        );  
    }  
  
    Map<String, Object?> toJson() {  
        return {  
            'firstName': firstName,  
            'lastName': lastName,  
            'email': email,  
        };  
    }  
}
```
























Step by step

(Un petit script pour générer des données 🧘)



 **load_data_to_firestore** Public Edit Pins Watch 2

main 1 branch 0 tags Go to file Add file Code

 DanielSchreurs  Reverting changes	969a865 20 hours ago	 4 commits
 .idea	 Initial commit	yesterday
 lib	 Reverting changes	20 hours ago
 .gitignore	 Initial commit	yesterday
 CHANGELOG.md	 Initial commit	yesterday
 README.md	 Initial commit	yesterday
 analysis_options.yaml	 Initial commit	yesterday
 load_data_to_firestore.iml	 Initial commit	yesterday
 pubspec.lock	 Initial commit	yesterday
 pubspec.yaml	 Initial commit	yesterday

 **README.md** 

Load Data to Firestore

Ce projet a pour objectif de publier deux collections de données fictives dans une base de données Firestore. Les deux collections sont les suivantes :

Collection "users"

La collection "users" représente une série d'utilisateurs fictifs. Voici un exemple de données fictives pour cette collection :

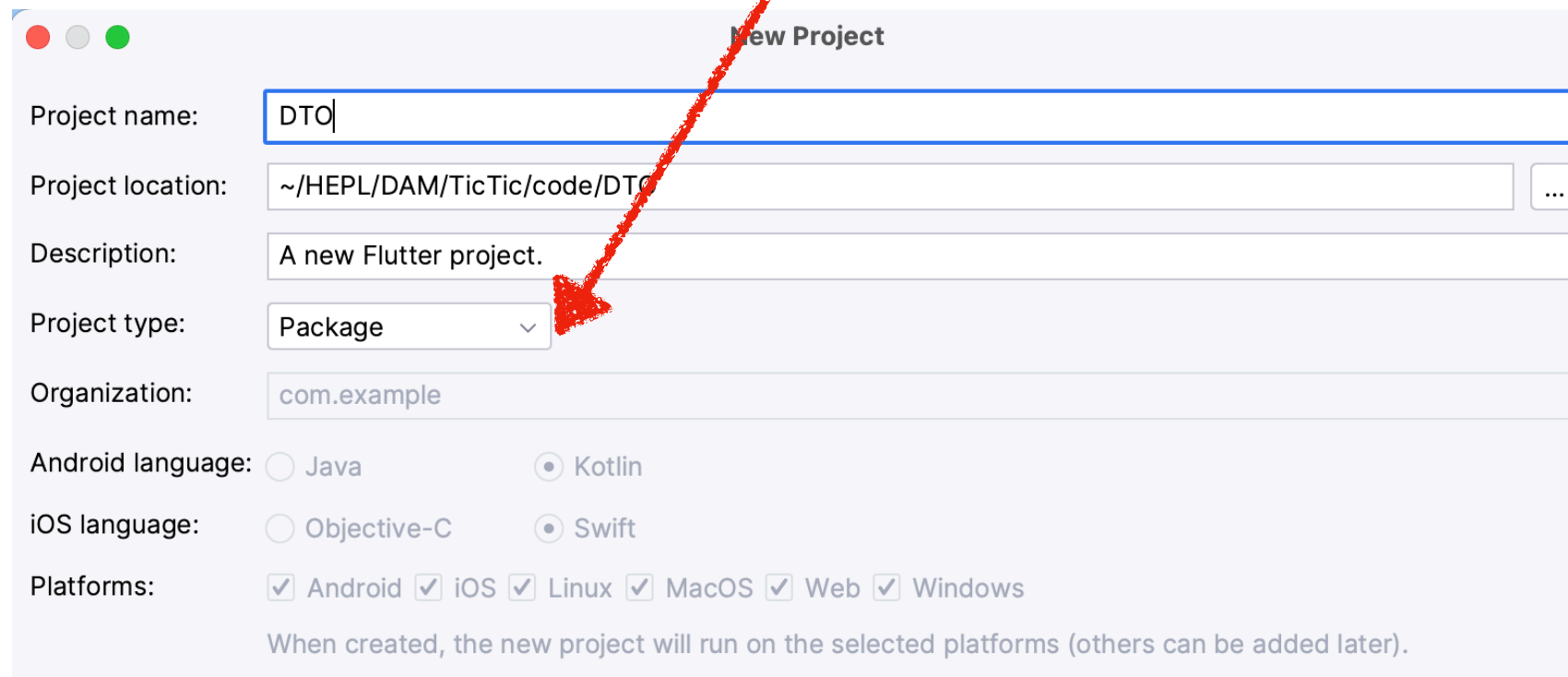
```
import 'package:dto/user.dart';
```



Step by step

Écrire des classes DTO (Objet de transfert de données)

- Objet de transfert de données
- Des classes très simples qu'on peut **sérialiser** et **désérialiser**
- Dans un projet Flutter à package Flutter



The screenshot shows the 'New Project' dialog box with the following fields and options:

- Project name: DTO
- Project location: ~/HEPL/DAM/TicTic/code/DTO
- Description: A new Flutter project.
- Project type: Package (selected, indicated by a red arrow)
- Organization: com.example
- Android language: ☐ Java ☒ Kotlin
- iOS language: ☐ Objective-C ☒ Swift
- Platforms: ☒ Android ☒ iOS ☒ Linux ☒ MacOS ☒ Web ☒ Windows

When created, the new project will run on the selected platforms (others can be added later).

Step by step

Initialiser Firebase



```
Future<void> main() async {  
  initializeDateFormatting();  
  WidgetsFlutterBinding.ensureInitialized();  
  await Firebase.initializeApp(  
    options: DefaultFirebaseOptions.currentPlatform,  
  );  
  await FirebaseAuth.instance.setLanguageCode("fr");  
  runApp(const MyApp());  
}
```

Step by step

Déclarer un flux de données



```
late final Stream<DocumentSnapshot<dto_user.User>>? _userStream;  
late final Stream<QuerySnapshot>? _teamsStream;
```


Step by step

Récupérer le flux de données (team)

```
@override
void initState() {
  super.initState();
  if (FirebaseAuth.instance.currentUser != null) {
    _teamsStream = FirebaseFirestore.instance
      .collection('teams')
      .where('users',
        arrayContains: FirebaseAuth.instance.currentUser!.email)
      .withConverter<Team>(
        fromFirestore: (snapshot, _) => Team.fromJson(snapshot.data()!),
        toFirestore: (team, _) => team.toJson(),
      )
      .snapshots();
  }
}
```

Step by step

Récupérer le flux de données (user)



```
@override
void initState() {
  super.initState();
  if (FirebaseAuth.instance.currentUser != null) {
    _userStream = FirebaseFirestore.instance
      .collection('users')
      .withConverter<dto_user.User>(
        fromFirestore: (snapshot, _) =>
          dto_user.User.fromJson(snapshot.data()!),
        toFirestore: (user, _) => user.toJson(),
      )
      .doc(FirebaseAuth.instance.currentUser!.email)
      .snapshots();
  }
}
```

Step by step

Utiliser ce flux



```
StreamBuilder<DocumentSnapshot<dto_user.User>>(
  stream: _userStream,
  builder: (BuildContext context, AsyncSnapshot<DocumentSnapshot<dto_user.User>> snapshot) {
    if (snapshot.hasError ||
        snapshot.connectionState ==
            ConnectionState.waiting) {
      return const HomeTitle(text: 'Bienvenue !');
    }
    return HomeTitle(
      text:
        'Hey, ${snapshot.data?.data().firstName}!');
  },)
```

Step by step

Utiliser ce flux



```
StreamBuilder<QuerySnapshot>(
  stream: _teamsStream,
  builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
    if (snapshot.connectionState == ConnectionState.waiting) {
      return const Center(
        child: CircularProgressIndicator());
    }
    if (snapshot.hasError) {
      return const Text('Oups, une erreur est survenue !');
    }
    return Column(
      children: snapshot.data!.docs
        .map((DocumentSnapshot document) {
          Team team = document.data()! as Team;
          return TeamOverview(team: team);
        }).toList(),
    );
  },
),
```


Sources

Sources et ressources à consulter

- [Get to know Cloud Firestore](#)
- [The Firebase Database For SQL Developers](#)
- [NoSQL data modeling](#)
- firebase.google.com
- [Firebase : Podcast](#)