Atelier 9 :
Gestion d'une base de données avec Firebase & Flutter

Objectifs:

- comprendre les étapes de création & de configuration d'un projet en utilisant la plateforme Firebase.
- Configurer et exécuter les opérations CRUD sur une BD Firebase.

I. Création d'un projet Firebase

La première étape pour cet Altier, consiste à la création d'un projet en utilisant Firebase pour cela, veuillez suivre les étapes suivantes :

1. Commencer par vous rendre sur le site officiel de Firebase et spécialement sur la console Firebase disponible à l'adresse suivante: : https://console.firebase.google.com
2. On commence par créer un nouveau projet Firebase : cliquez sur le bouton "*Ajouter un projet*" pour créer un nouveau projet.
3. Donnez un nom à votre projet Firebase,:Firebase Crud Article
4. désactiver l'option : « Activer Google Analytics pour ce projet »

5. Valider la création du projet.

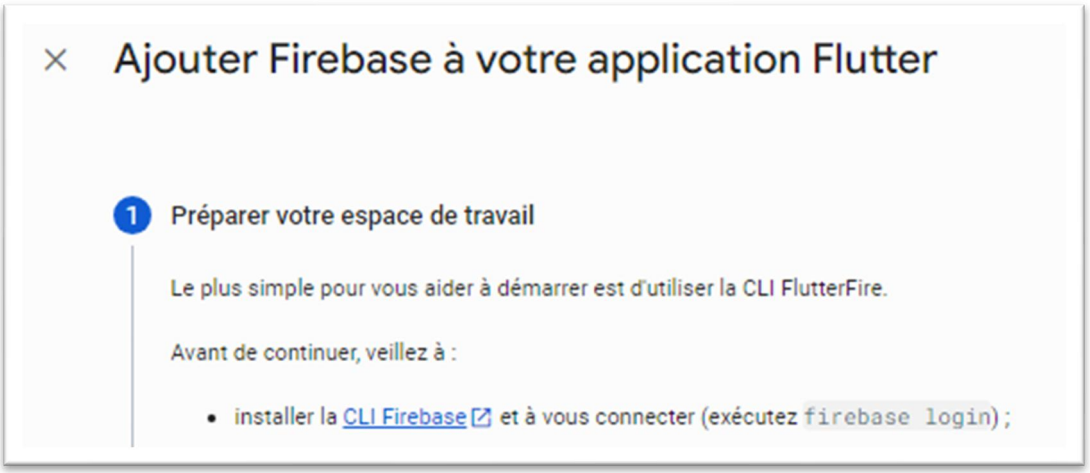
II. Téléchargement & installation de CLI FLUTTER :

Pour l'utilisation de Firebase avec Framework Flutter, un assistant a été implémenté spécialement pour automatiser l'opération de configuration de tout le projet flutter :

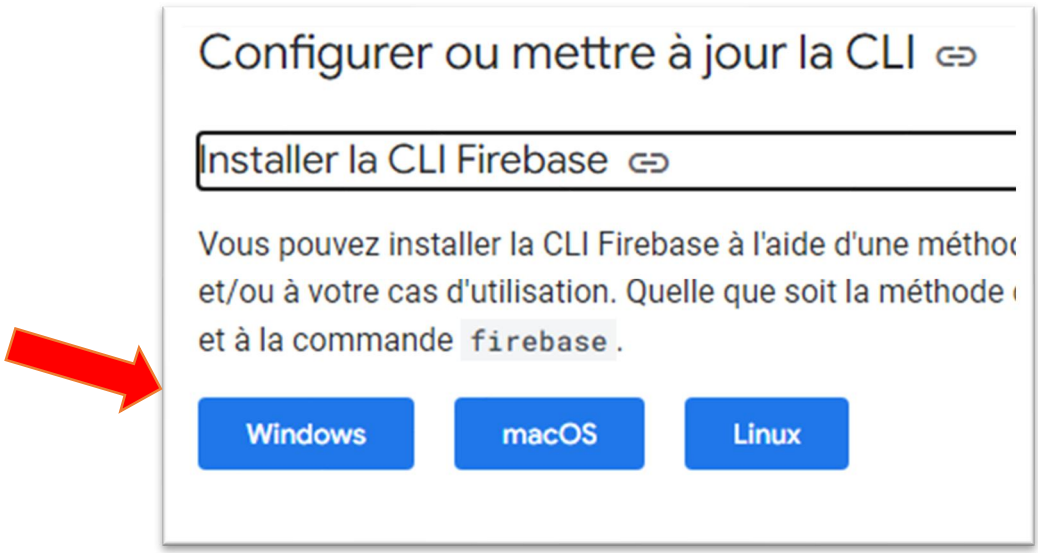
1. commencer par clique sur l'icone d'intégration de votre créer avec flutter.

2. Depuis la fenêtre qui s'ouvre ,on doit installer la CLI Firebase : une application qui sera utilisée en mode terminal pour configurer les applications a relié avec votre projet Firebase.

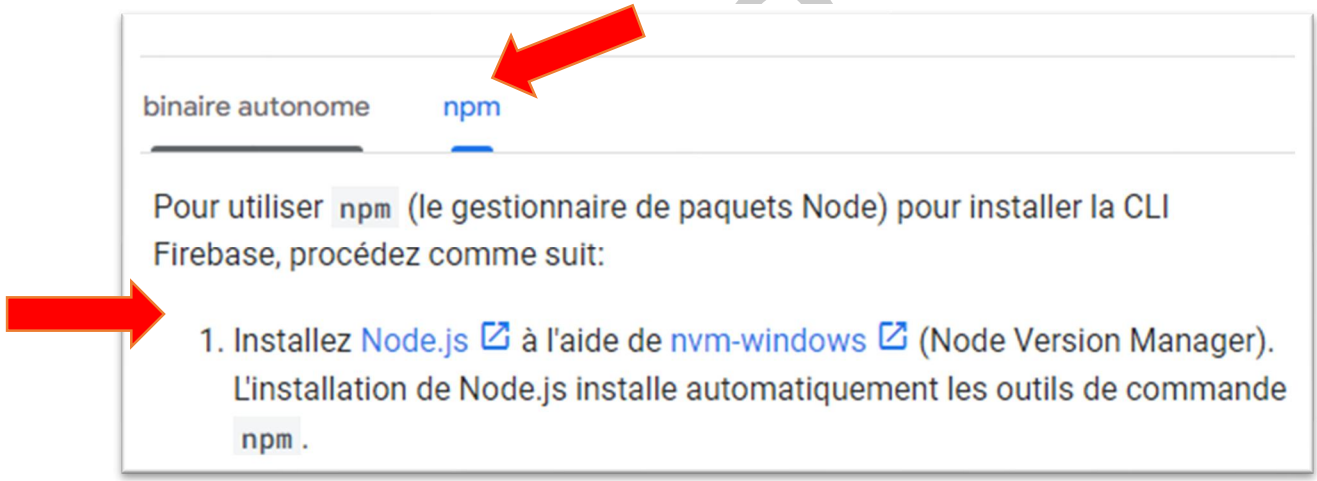
Pour cela il faudra :



- Accéder au site suivant ou cliquer sur le lien CLI Firebase
- Sélectionner votre système d'exploitation utilisé pour le développement (Windows je suppose 😊) :

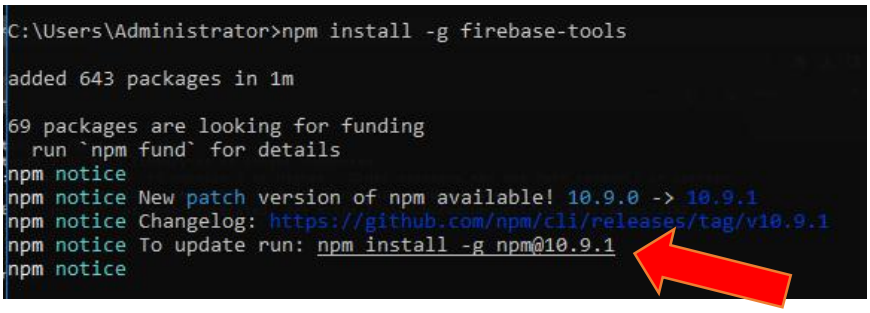


- Sélectionner la méthode d'installation de CLI Firebase en utilisant NPM, pour qu'il doit être installé et intégré au système.



- Cliquer sur Installer Node.js et accéder au site https://nodejs.org/fr
- Télécharger la dernière version LTS disponible.
- Installer nodeJs
- Redémarrer votre machine
- Tester dans un Terminal la commande suivante : **nodejs –version**

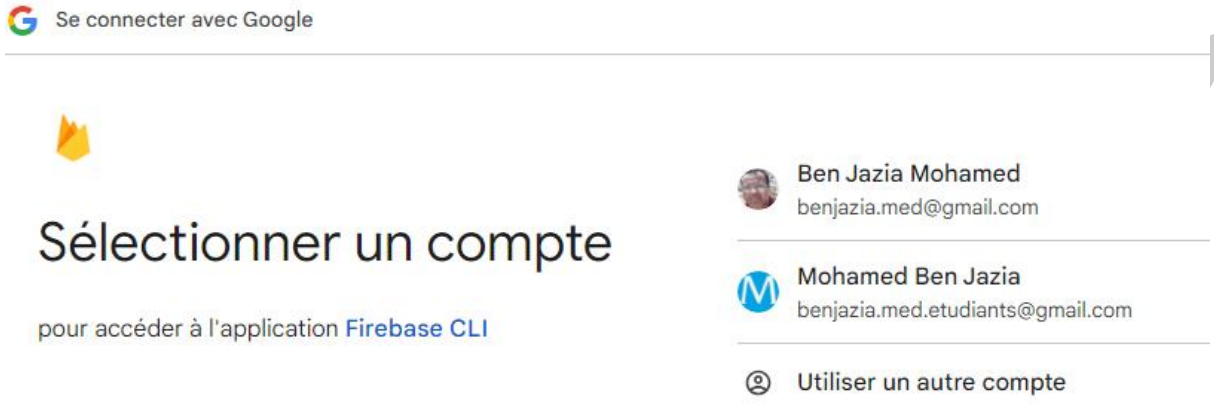
4. Après installation avec succès, une notice vous informe de l'existence d'une nouvelle version de mise à jour :



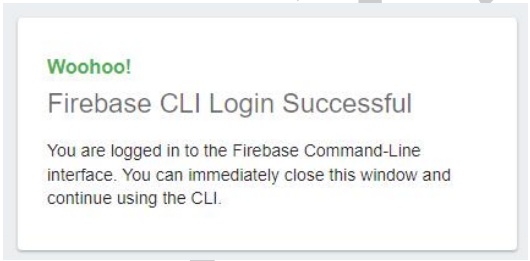
5. l'installation de CLI Firebase est Terminé.
6. on doit se connecter au Firebase en ligne , pour cela taper la commande suivante :

firebase login

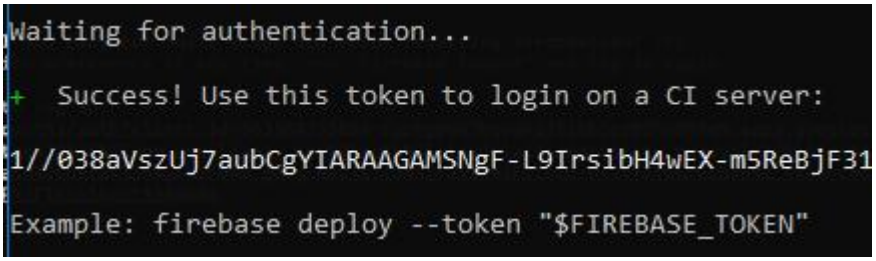
- une fenêtre web vous invite à vous connecter avec google
- Vous devez utiliser le même compte google que celui utilisé avec Firebase pendant la création du projet



7. Suivez les étapes de l'assistant , enfin un message vous informe du succès de l'authentification :



- Apres la connexion le message précédant vous informe que vous et reconnecté au service de Firebase en ligne et vous pouvez utilisé le CLI FIREBASE avec la commande **firebase**



NB : les étapes de téléchargement de NPM et de Flutter tools se fait une seule fois, tant que vous n'avez pas changer de pc ou bien formatter.

III. Configuration Firebase avec application flutter :

1. Commencer par revenir à page web du site console.firebase.com.
2. cliquer sur suivant :

3. taper la commande suivante dans un terminal :

dart pub global activate flutterfire_cli

4. il faudra ajouté le chemin indiqué dans la variable PATH de votre système

```
Building package executables... (15.2s)
Built flutterfire_cli:flutterfire.
Installed executable flutterfire.
Warning: Pub installs executables into C:\Users\Administrator\AppData\Local\Pub\Cache\bin, which is not on your path.
You can fix that by adding that directory to your system's "Path" environment variable.
A web search for "configure windows path" will show you how.
```

5. la dernière commande pour lancer la configuration du notre projet Flutter :

flutterfire configure --project=fir-crud-article

NB : uniquement cette commande doit être utilisé dans le cas de création d'un nouveau projet

Il nous reste maintenant de sélectionner avec le clavier les os à relier avec notre projet Firebase, comme indiquer ci-dessous :

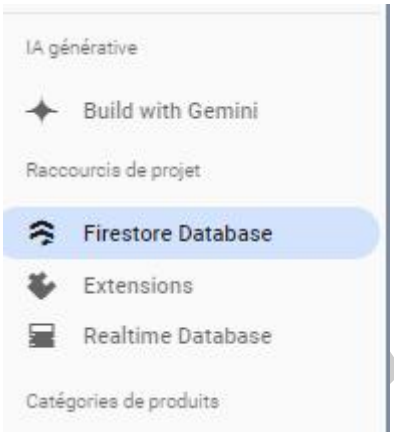
```
-project=fir-crud-article
i Found 3 Firebase projects. Selecting project fir-crud-article.
? Which platforms should your configuration support (use arrow keys & space to select)? ›
✓ android
✓ ios
✓ macos
✓ web
✓ windows
```

Le flutter cli crée un fichier de configuration : lib\ firebase_options.dart

6. pour la configuration d'Android , le cli vous demande le nom de package de l'application android à générer ; vous devez accéder au fichier :**android\app\build.gradle** , vous copiez **nameSpace**
7. il suffit d'installer le plugin firebase_core,

IV. Création d'une base de données Firebase :

1. A partir de menu principal de votre projet sur console.firebase.com, sélectionner créer Firestore Database :



2. Cliquer sur « créer une base de données »

3. Sélectionner le mode TEST : pour la création rapide de la bd

4. Créer une collection appelée : articles , par exemple

5. Ajouter des documents à collection : chaque article est un document., en choisissant id auto générée.

V. Insertion d'un Article :

Pour sauvegarder un article, on pourra utiliser le formulaire déjà implémenter dans le TP de Formulaire .

Après la validation et la récupération des données on fera appel à la méthode suivante :

```
final CollectionReference collectionArticles =
    FirebaseFirestore.instance.collection('articles');
```

```
void addArticle() async {
   try {
     await collectionArticles.add({
       'title': 'Introduction à Firestore avec Flutter',
       'qte': 300,
     });
     print("Article ajouté avec succès !");
     ScaffoldMessenger.of(context).showSnackBar(
       const SnackBar(content: Text('Article ajouté avec succès !')),
     );
   } catch (e) {
     print("Erreur lors de l'ajout de l'article : $e");
   }
 }
```

VI. Récupération des données(documents) :

1. commencer par déclarer un objet pour pouvoir récupérer une instance de la base de données en ligne :

```
final CollectionReference collectionArticles =
      FirebaseFirestore.instance.collection('articles');
```

2. implémenter la fonction suivante : getAllArticles() : permettant de récupérer la liste des articles depuis la base de données :

```
void fetchAllDocuments1() async {
   try {
     QuerySnapshot snapshot = await collectionArticles.get();
     for (var doc in snapshot.docs) {
       print("Article: [ Libellé = "+ doc['lib'] + "Quantité= "+
doc['qte']);
     }
   } catch (e) {
     print("Erreur : $e");
   }
 }
```

3. on essaye maintenant d'implémenter une nouvelle version de récupération des articles en utilisant une liste de Map :

On commence par implémenter une méthode future :

```
Future<List<Map<String, dynamic>>> fetchAllDocuments2() async
{
    QuerySnapshot snapshot = await collectionArticles.get();
    return snapshot.docs
        .map((doc) => doc.data() as Map<String, dynamic>)
        .toList();
 }
```

4. intégrer la fonction dans un widget FuturBuilder :

```
FutureBuilder<List<Map<String, dynamic>>>(
   future: fetchAllDocuments2(),
   builder: (context, snapshot) {
      if (snapshot.connectionState == ConnectionState.waiting) {
          return const Center( child: CircularProgressIndicator(),);
      }
      if (snapshot.hasError) {
          return Center(child: Text("Erreur : ${snapshot.error}"),);
      }
      if (!snapshot.hasData || snapshot.data!.isEmpty) {
          return const Center(child: Text("Aucun document trouvé."),);
     }
      final documents = snapshot.data!;
      return ListView.builder(
             itemCount: documents.length,
             itemBuilder: (context, index) {
                 final article = documents[index];
                    return Card(
                         color: Colors.amber,
                        child: ListTile(
                          title: Text(article['lib'] ?? "Sans titre"),
                          subtitle: Text(
                              article['qte'].toString() ?? "Pas de contenu"),
                        ),
                      );
                  },
               );
            },
          ),
        ) ;
```