TP Flutter 5AL

Objectif

Pour ce TP, il vous est demandé de développer une application proposant :

- une page avec une liste de posts (un post possède un titre et une description) qui vient de votre base de données Firestore
- une page de détail d'un poste avec la possibilité de modifier un post existant.
- un bouton flottant amenant à une page permettant de créer un nouveau post
- Que ce soit pour la récupération des posts, la modification ou la création d'un nouveau post, tous les états potentiels doivent être gérés (vide, succès, erreur, chargement)

D'où viennent les données ?

Je ne peux pas vous demander de développer pour ce TP votre propre API car je ne peux juger que votre niveau en Flutter.

Cependant, nous avons justement vu dans notre cours, grâce à l'architecture Bloc - Repository Pattern que la source des données ne devrait pas jouer sur le reste de l'application.

L'idée de l'application est donc de fournir un **faux data source** qui fournit de fausses données avec de faux temps d'attente.

Par exemple :

- imaginons que nous avons bien une classe `Post`
- imaginons que nous avons mis en place notre interface `PostsDataSource`
- alors notre fausse implémentation pourrait donner ça :

```
class FakePostsDataSource extends PostsDataSource {
  final List<Post> _fakePosts = const [
    Post(id: '1', title: 'Post 1', description: 'Description of
    Post(id: '2', title: 'Post 2', description: 'Description of
    Post(id: '3', title: 'Post 3', description: 'Description of
  ];

  @override
  Future<List<Post>> getAllPosts() async {
    await Future.delayed(const Duration(seconds: 1));
    return _fakePosts;
  }

  @override
  Future<Post> createPost(Post postToAdd) async {
    await Future.delayed(const Duration(seconds: 1));

    /// Add the post to the list
  }

  @override
  Future<Post> updatePost(Post newPost) async {
    await Future.delayed(const Duration(seconds: 1));

    /// Update the post in the list
```

```
  }
}
```

💡 Avec un faux data source comme celui-ci, nous pouvons quand même tester tous les cas possible :

- le chargement
- le succès
- le succès vide
- l'erreur

Dans votre rendu final, je m'attends uniquement à un faux data source qui ne renvoie que des cas de succès, mais l'app doit pouvoir gérer les autres cas.

Rendu final

Vous devez déposer votre code final sur MyGES à la date demandée sur MyGES.

Je m'attends à 2 choses :

- le code zippé (obligatoire pour l'école)
- **un lien vers un depôt public avec votre code**
  - mettez le simplement en commentaire lorsque vous mettez le dossier zippé 👍
  - je dois y avoir accès donc ça doit être disponible publiquement

⚠️ Pour lancer votre application, je dois avoir à faire que ces commandes :

```
git clone <votre_depôt>
cd <votre_dossier>
```

```
flutter pub get
flutter pub run
```

Barème

Critère	Points
Respect du pattern Bloc	5
Respect du pattern Repository	5
Gestion des états	5
Qualité générale : - Expérience utilisateur - Navigation, - respect des bonnes pratiques et conventions - etc.	5