Flutter TD2

Pour commencer

Créez une nouvelle application flutter et construisez un #StatelessWidget# simple qui construit l'arbre suivant:

```
MaterialApp(title:'TD2')
|_home:Scaffold
        |_appBar: AppBar(...)
        |_body: Center(...)
```

Utilisation d'un thème

Après avoir étudié le fichier *mytheme.dart* disponible sur Celene, utilisez le thème fournit dans votre application pour modifier le style du titre de l'AppBar et du texte du body en utilisant *headline6* et *headline1* du thème sombre. Il faut pour cela ajouter *google_fonts: ^3.0.1* aux dépendances dans le fichier *pubspec.yaml*.

BottomNavigationBar

Ajoutez maintenant une *BottomNavigationBar* composée de trois *icons* à votre *Scaffold*. La liste des icons disponibles se trouve ici.

Interactivité

Faites fonctionner votre *BottomNavigationBar* en créant trois *StatelessWidgets* basiques (un texte centré avec une couleur de fond).

Ajout d'un modèle

Dans un dossier *models*, ajoutez un fichier *task.dart* qui contient la classe suivante:

```
class Task {
  int id;
  String title;
  List<String> tags;
  int nbhours;
  int difficuty;
  String description;

  Task({required this.id,required this.title,required this.tags,required
this.nbhours,required this.difficuty,required this.description});

  static List<Task> generateTask(int i){
    List<Task> tasks=[];
    for(int n=0;n<i;n++){
      tasks.add(Task(id: n, title: "title $n", tags: ['tag $n','tag
${n+1}'], nbhours: n, difficuty: n, description: '$n'));
```

```
    }
    return tasks;
  }
}
```

Une première ListView

Le premier widget de la *BottomNavigationBar* doit afficher dans une *ListView* les *Tasks* générée par la méthode static *generateTask* de la classe *Task*. Pour chaque *Task*, on veut afficher son titre, son index et ses tags. Utilisez le cours sur les *ListView* disponible sur Celene ainsi que la documentation(ListView,ListTitle).

Création d'une API

Ajoutez à votre projet le fichier *tasks.json*. Nous allons construire une petite API nous permettant d'obtenir les *tasks* de ce fichier. Pour cela, il va falloir utiliser la programmation asynchrone.

```
class MyAPI{

  Future<List<Task>> getTasks() async{
    await Future.delayed(Duration(seconds: 1));
    final dataString = await _loadAsset('assets/json/tasks.json');
    final Map<String,dynamic> json = jsonDecode(dataString);
    if (json['tasks']!=null){
      final tasks = <Task>[];
      json['tasks'].forEach((element){
        tasks.add(element);
      });
      return tasks;
    }else{
      return [];
    }
  }

  Future<String> _loadAsset(String path) async {
    return rootBundle.loadString(path);
  }
}
```

Mise en place d'une ListView utilisant l'API

Pour cette ListView, utilisez un *FuturBuilder* (doc). Utilisez le second widget de la *BottomNavigationBar*.

Utilisation d'une API REST

Pour le troisième widget de la *BottomNavigationBar*, utilisez une API Rest (par exemple https://jsonplaceholder.typicode.com/) pour construire une ListView. Pour cela, consultez la documentation flutter qui explique comment effectuer des requêtes http grâce au package http (doc flutter, doc http).

Pensez à ajouter ce package à vos dépendances dans le fichier *pubspec.yaml*.

Un peu d'interactivité

En utilisant la documentation flutter se trouvant ici, faites en sorte que le click sur un *ListTile* permette d'afficher une nouvelle "page" contenant toutes les informations de la *task*.