Développer une première application mobile en Flutter

Présentation du projet

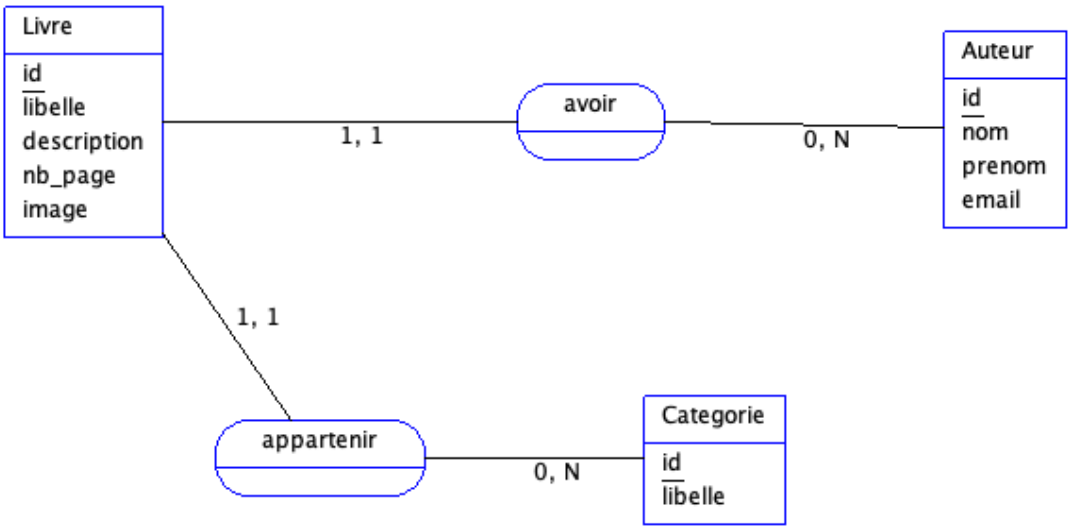
Notre projet « bibliotheca » est une application mobile qui permettra de gérer une bibliothèque. Nous pourrons gérer les entités suivantes :

- Les livres
- Les catégories de livre et
- Les auteurs des livres

Pour chacune de ces entités, nous allons créer pas-à-pas des écrans qui nous permettrons de faire les actions telles que **ajouter**, **modifier**, **supprimer** et **lire la liste** de ces différentes entités.

Ce projet se veut assez complet, il sera traité en plusieurs étapes. Dans un premier temps, nous allons créer la structure générale de notre application et ensuite, on ajoutera une petite base de données locale qui nous permettra d'enregistrer nos données de manière persistante.

Le schéma du modèle conceptuel de données de notre application est le suivant :



Le dictionnaire de données :

Nom	ID	Type	Taille	Utilisé	Entité
id	id_Livre	BIGINT	0	☑	Livre
libelle	libelle_Livre	VARCHAR	255	☑	Livre
description	description_Livre	VARCHAR	500	☑	Livre
nb_page	nb_page_Livre	INT	3	☑	Livre
id	id_Categorie	BIGINT	0	☑	Categorie
libelle	libelle_Categorie	VARCHAR	255	☑	Categorie
id	id_Auteur	BIGINT	0	☑	Auteur
nom	nom_Auteur	VARCHAR	255	☑	Auteur
prenom	prenom_Auteur	VARCHAR	255	☑	Auteur
email	email_Auteur	VARCHAR	500	☑	Auteur
image	image_Livre	VARCHAR	100	☑	Livre
		BIGINT	0	☐	

Création du projet Flutter

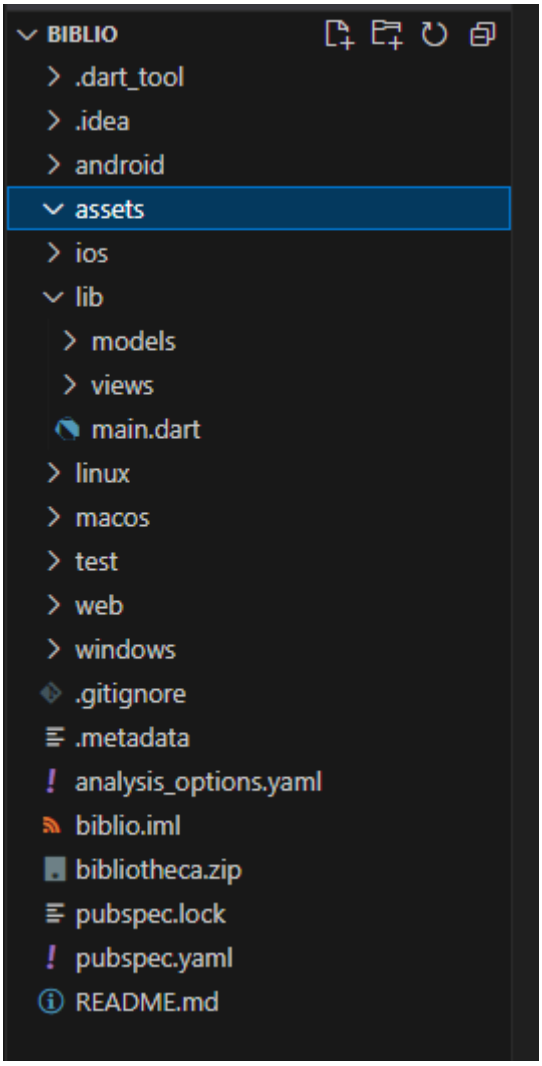
Pour commencer, vous devez créer un projet Flutter. Vous pouvez le faire en tapant la commande suivant dans un terminal :

```
flutter create bibliotheca
```

Une fois le projet créé, vous pouvez l'ouvrir avec votre éditeur

Mise en place de l'architecture

Organisation et mise en place de l'architecture de notre projet.

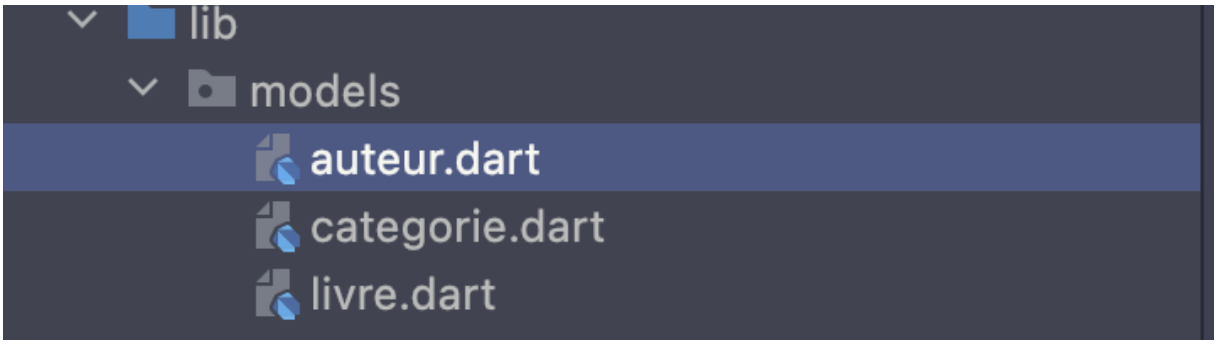


Voici les dossiers dans lesquels nous travaillerons :

- **assets** : ce dossier contiendra nos images, fichier de police, etc.
- **lib** : c'est le dossier principal de notre projet. Ce dossier contiendra tout le code Dart qu'on écrira.
- **models** : le dossier models contiendra les déclarations des entités métiers de notre projet.
- **views** : ici, on organisera les fichiers qui gèreront l'affichage de nos écrans.

Création des entités métiers

Création des classes de nos différentes entités **livre**, **auteur** et **categorie**.



Contenu du fichier auteur.dart

```dart
class Auteur {
  int? id;
  String? nom;
  String? prenoms;
  String? email;

  Auteur({this.id, this.nom, this.prenoms, this.email});
}
```

Contenu du fichier categorie.dart :

```dart
class Categorie {
  int? id;
  String? libelle;

  Categorie({this.id, this.libelle});
}
```

Contenu du fichier livre.dart :

```dart
class Livre {
  int? id;
  String? libelle;
  String? description;
  int? nbPage;
  String? image;
  int? categorieId;
  int? auteurId;

  Livre(
      {this.id,
      this.libelle,
      this.description,
      this.nbPage,
      this.image,
```

```
      this.auteurId,
      this.categorieId});
}
```

Nous venons de créer nos entités métiers. Vous pouvez remarquer que ces entités sont simplement des classes Dart. Nous nous contenterons, pour l'instant, de déclarer les attributs qui composent chaque entité ainsi qu'un constructeur avec des paramètres nommés facultatifs.

Les écrans de notre application

Notre application comportera 7 écrans ou pages :

1. Un écran de menu (accueil)
2. Un écran pour lister les catégories
3. Un écran pour éditer (créer ou modifier) une catégorie
4. Un écran de liste des auteurs
5. Un écran d'édition d'auteur (création et modification)
6. Un écran pour lister les livres
7. Et enfin, un écran pour éditer un livre.

Création des écrans

Ecran d'accueil

Avant de commencer, nous allons faire un peu de menage dans code existant. Nous allons donc nous intéresser au contenu du dossier **lib**. Dans ce dossier, nous avons un fichier **main.dart**. Il s'agit du fichier de démarrage d'une application Flutter. Nous allons supprimer tout le contenu de ce fichier.

Dans le dossier « views », nous allons créer un fichier qui se nommera **home_page.dart**. Voici le code de ce fichier :

```
import 'package:flutter/material.dart';

class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) => Scaffold();
}
```

Ce fichier contient deux classes :

- La première est **HomePage** qui hérite (extends) de la classe **StatefulWidget,**
- et une autre **_HomePageState** qui hérite de la classe **State<HomePage>**.

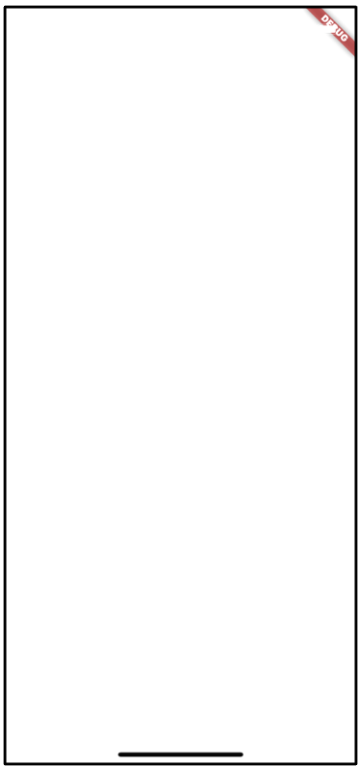
Ce sont ces deux classes qui se chargeront de dessiner l'interface de notre page d'accueil. Nous avons choisi d'utiliser une page qui hérite de la classe StatefulWidget parce que nous aurons besoin de changer l'état de notre écran en l'actualisant. Nous construirons l'architecture de notre écran dans le widget **Scaffold.**

Après avoir créé le premier écran, il faut informaer Flutter que la page **HomePage** constitu la première page de l'application. Pour cela, on édite le fichier main.dart et on ajoute le code suivant :

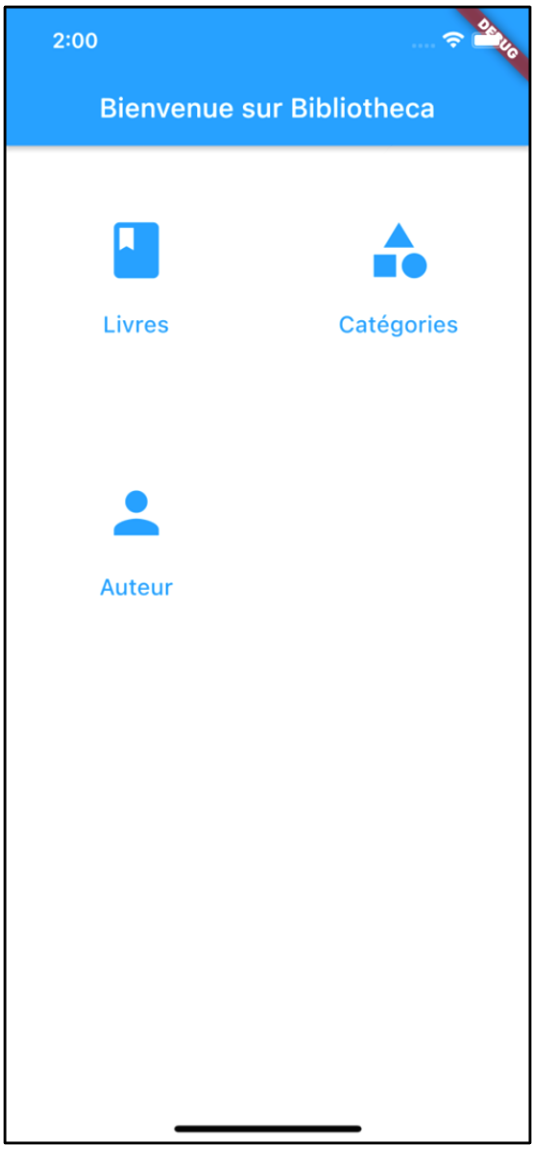
```
import 'package:bibliotheca/views/home_page.dart';
import 'package:flutter/material.dart';

void main() {
  runApp(
    MaterialApp(
      title: "Bibliotheca",
      theme: ThemeData(primaryColor: Colors.blue),
      home: const HomePage(),
    ),
  );
}
```

Analysons ce code. On remarque que la méthode main fait appel à la méthode runApp (c'est cette méthode qui permet le lancement de l'application). Ensuite, cette dernière prend en paramètre Un widget (la classe MaterialApp). Dans ce widget, nous définissons un titre pour notre application, un thème (où l'on précise la couleur principale de notre application et enfin la première page qui sera ouverte lorsque notre application aura démarré.

A ce stade nous pouvons lancer notre application. Quand vous lancez l'application, vous obtenez une page blanche.



Le rendu final sera celui-ci :



Le code complet de la page est le suivant :

```dart
import 'package:bibliotheca/views/liste_auteur.dart';
import 'package:bibliotheca/views/liste_categorie.dart';
import 'package:bibliotheca/views/liste_livre.dart';
import 'package:flutter/material.dart';

class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) => Scaffold(
        appBar: AppBar(
          title: const Text("Bienvenue sur Bibliotheca"),
        ),
        body: GridView(
          gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
            crossAxisCount: 2,
          ),
          children: [
            MaterialButton(
              textColor: Colors.blue,
              onPressed: () {
                Navigator.push(context,
                    MaterialPageRoute(builder: (_) => const ListeLivrePage()));
              },
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: const [
                  Icon(Icons.book, size: 50),
                  SizedBox(height: 20),
                  Text(
                    "Livres",
                    style: TextStyle(fontSize: 17),
                  ),
                ],
              ),
            ),
            MaterialButton(
              textColor: Colors.blue,
              onPressed: () {
                Navigator.push(context,
                    MaterialPageRoute(builder: (_) => const ListeCategorie()));
              },
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: const [
                  Icon(Icons.category, size: 50),
                  SizedBox(height: 20),
                  Text(
                    "Catégories",
                    style: TextStyle(fontSize: 17),
                  ),
                ],
```

```
            ),
          ),
          MaterialButton(
            textColor: Colors.blue,
            onPressed: () {
              Navigator.push(context,
                  MaterialPageRoute(builder: (_) => const ListeAuteur()));
            },
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: const [
                Icon(Icons.person, size: 50),
                SizedBox(height: 20),
                Text(
                  "Auteur",
                  style: TextStyle(fontSize: 17),
                ),
              ],
            ),
          ),
        ],
      ),
    );
}
```

Analysons ce code :

Dans la classe **_HomePageState** nous avons la méthode **build** qui retourne un widget **Scafold**.

Scafold : ce widget est un widget assez courant qui nous permet de facilement construire un écran en respectant la structure générale d'un écran d'application mobile. Ce widget a plusieurs propriétés telles que

- **appBar** : qui nous permet de construire la barre supérieure bleue de la page, comme le montre la capture précédente.

```
...
appBar: AppBar(
  title: const Text("Bienvenue sur Bibliotheca"),
),
...
```

- **body** : nous avons également la propriété body qui permet de construire « le corps » de notre écran, la partie en bas de la barre de titre (appBar).

```
...
body: GridView(
          gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
            crossAxisCount: 2,
          ),
          children: [
            MaterialButton(
              textColor: Colors.blue,
              onPressed: () {
                Navigator.push(context,
```

```
                    MaterialPageRoute(builder: (_) => const
ListeLivrePage()));
                },
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: const [
                    Icon(Icons.book, size: 50),
                    SizedBox(height: 20),
                    Text(
                      "Livres",
                      style: TextStyle(fontSize: 17),
                    ),
                  ],
                ),
              ),
              MaterialButton(
                textColor: Colors.blue,
                onPressed: () {
                  Navigator.push(context,
                      MaterialPageRoute(builder: (_) => const
ListeCategorie()));
                },
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: const [
                    Icon(Icons.category, size: 50),
                    SizedBox(height: 20),
                    Text(
                      "Catégories",
                      style: TextStyle(fontSize: 17),
                    ),
                  ],
                ),
              ),
              MaterialButton(
                textColor: Colors.blue,
                onPressed: () {
                  Navigator.push(context,
                      MaterialPageRoute(builder: (_) => const ListeAuteur()));
                },
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: const [
                    Icon(Icons.person, size: 50),
                    SizedBox(height: 20),
                    Text(
                      "Auteur",
                      style: TextStyle(fontSize: 17),
                    ),
                  ],
                ),
              ),
            ],
          ),
...
```

Dans ce code nous avons comme body un widget **GridView**. C'est ce widget qui nous permet d'afficher les éléments comme une grille.

Nous avons utilisé deux propriétés de ce widget. Le **gridDelegate** et le **children**.

gridDelegate permet de paramétrer la grille. Ici, nous avons signifié que nous voulons deux colonnes de widget par ligne.

```
...
gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
  crossAxisCount: 2,
),
...
```

children permet de déclarer la liste d'enfant qui sera afficher. Notez que les enfants auront plus ou moins le même aspect. Nous avons donc déclaré les widgets enfants avec la même architecture comme suit :

```
MaterialButton(
  textColor: Colors.blue,
  onPressed: () {},
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
```

```
    children: const [
      Icon(Icons.book, size: 50),
      SizedBox(height: 20),
      Text(
        "Livres",
        style: TextStyle(fontSize: 17),
      ),
    ],
  ),
),
```

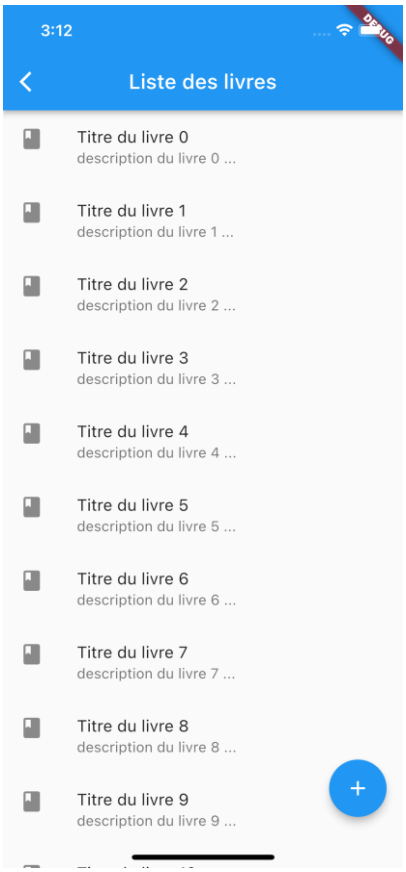
Comme chaque enfant doit être cliquable, nous avons commencé par déclarer un widget bouton **MaterialButton**. A ce bouton, nous avons donné une couleur bleue à tous les widgets qu'il contient dans sa propriété **child** et enfin une propriété **onPressed** qui est une fonction dont le code sera exécutée lorsque l'utilisateur clique sur le bouton. Dans la propriété **child** de notre bouton, nous avons déclaré une **Column**.

Column est un widget qui permet d'afficher un ensemble de widget de façon verticale. Ici, nous avons affiché une icône et un text (Ex: Livres) qui sont tous les deux espacés de 20 pixels avec le widget **SizedBox**. Ainsi en répétant ce code, bien sûr en prenons soin de changer l'icône et le text du bouton à chaque fois, nous avons pu construire les autres boutons qui composent notre petit menu

Nous allons maintenant créer les pages de liste des livres, catégories et auteurs sur quoi l'utilisateur est censé être redirigé quand il clique sur un bouton du menu.

Ecran de liste des livres

Cet écran servira à lister les livres qui seront enregistrés par l'utilisateur. Nous allons créer un nouveau fichier **liste_livre.dart** toujours dans le dossier **views** pour y afficher la liste. Dans ce fichier, nous créons également une classe qui va s'appeler **ListLivrePage**. Elle sera chargée de construire un écran qui devrait ressembler à ça :



Le code pour produire cet écran est le suivant :

```dart
import 'package:flutter/material.dart';

class ListLivrePage extends StatefulWidget {
  const ListLivrePage({Key? key}) : super(key: key);

  @override
  State<ListLivrePage> createState() => _ListLivrePageState();
}

class _ListLivrePageState extends State<ListLivrePage> {
  @override
  Widget build(BuildContext context) => Scaffold(
        appBar: AppBar(
          title: const Text("Liste des livres"),
        ),
        floatingActionButton: FloatingActionButton(
          child: const Icon(Icons.add),
          onPressed: () {},
        ),
        body: ListView.builder(
          itemCount: 20,
          itemBuilder: (context, i) => ListTile(
            leading: const Icon(Icons.book),
            title: Text("Titre du livre $i"),
            subtitle: Text("description du livre $i ..."),
            onTap: () {},
          ),
        ),
      );
```

```
}
```

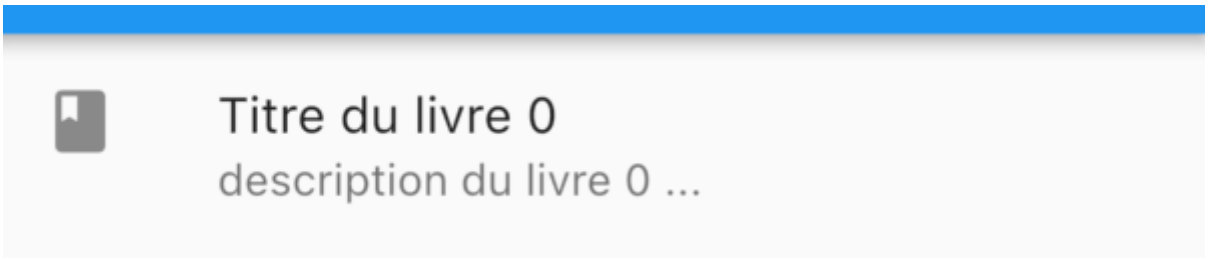
Dans le **body**, nous avons un widget **ListView.builder**. ListView.builder permet de créer une liste verticale de widget comme le widget Column. Mais la différence est que **ListView.builder** permet de créer une liste scrollable (défilante). Ainsi lorsque les éléments de la liste dépassent la surface visible de l'écran du téléphone, l'utilisateur pourra faire glisser la liste de bas en haut pour faire afficher les autres éléments contenus dans la liste. Nous avons utilisé différentes propriétés du widget **ListView.builder** dont **itemCount** et **itemBuilder**.

itemCount permet de dire au composant ListView qui nous voulons afficher 20 éléments dans la liste. Le paramètre **itemCount** attend une valeur entière ou nulle c'est à dire une valeur de type **int?**. Quand vous lui passez une valeur nulle, cela signifie que nous voulons une liste avec un nombre infini d'éléments.

itemBuilder permet de construire chaque élément de notre liste. En fait, ici, on lui fournit une fonction anonyme qui attend les paramètres **context** et l'**indice** de l'élément courant de la liste à construire. Cette fonction doit retourner un widget. Ainsi, on lui retourne un widget **ListTile**.

```
...
itemBuilder: (context, i) => ListTile(
  leading: const Icon(Icons.book),
  title: Text("Titre du livre $i"),
  subtitle: Text("description du livre $i ..."),
  onTap: () {},
),
...
```

Le widget **ListTile** permet de construire chaque élément de la liste.



Nous avons utilisé le paramètre

leading qui permet de déclarer l'icône livre à gauche de la ligne. Nous y avons mis une icône **leading: const Icon(Icons.book)**.

title permet d'afficher le titre du livre **title: Text(« Titre du livre $i »)**.

subTitle permet d'afficher la description du livre en sous-titre **subtitle: Text(« description du livre $i … »)**.

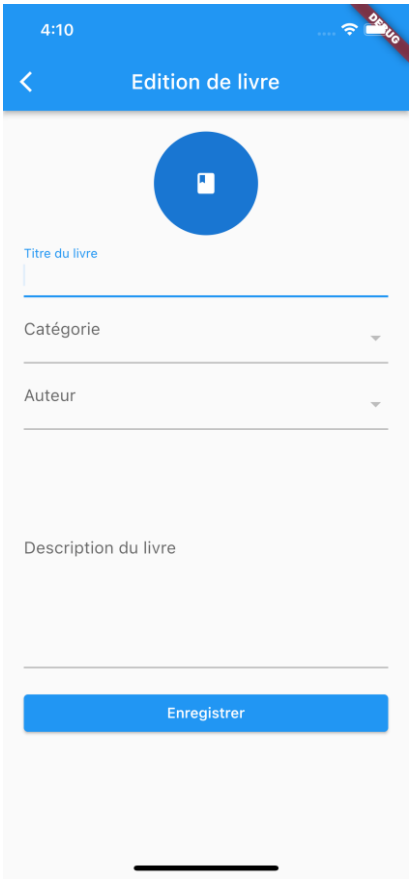
Nous avons aussi le bouton qui nous permettra d'ajouter un livre dans notre liste. Il s'agit du widget **FloatingActionButton** qui est un élément du widget Scafold.

```
...
floatingActionButton: FloatingActionButton(
  child: const Icon(Icons.add),
  onPressed: () {},
),
...
```

FloatingActionButton est un bouton. Il prend comme paramètre un widget enfant (child) et une fonction anonyme (onPressed). Vous l'aurez deviné, **child** permet de définir l'icône plus (+) et **onPressed** permet de déclarer la fonction qui sera exécutée quand l'utilisateur cliquera sur le bouton.

Ecran d'édition d'un livre

Cet écran est un formulaire qui permettra de renseigner le titre, la description, l'image de notre livre et également l'auteur de ce livre. Nous allons créer un nouveau fichier dans le dossier **views** **edition_livre.dart** qui comportera le code de l'écran **EditionLivre** Voici à quoi ressemble cet écran :




```dart
import 'package:bibliotheca/models/auteur.dart';
import 'package:bibliotheca/models/livre.dart';
import 'package:flutter/material.dart';

class EditionLivre extends StatefulWidget {
  const EditionLivre({Key? key}) : super(key: key);

  @override
  State<EditionLivre> createState() => _EditionLivreState();
}

class _EditionLivreState extends State<EditionLivre> {
  @override
  Widget build(BuildContext context) => Scaffold(
        appBar: AppBar(
          title: const Text("Edition de livre"),
        ),
        body: Form(
          child: ListView(
            padding: const EdgeInsets.all(20),
            children: [
              MaterialButton(
                onPressed: () {},
                child: const CircleAvatar(
```

```
                radius: 50,
                child: Icon(Icons.book),
              ),
            ),
            TextFormField(
              decoration: const InputDecoration(labelText: "Titre du livre"),
            ),
            DropdownButtonFormField<Categorie>(
              items: [
              DropdownMenuItem(
                  value: Categorie(id: 1),
                  child: Text("Catégorie 1"),
                ),
                DropdownMenuItem(
                  value: Categorie(id: 2),
                  child: Text("Catégorie 2"),
                ),
              ],
              onChanged: (value) {},
              decoration: const InputDecoration(labelText: "Catégorie"),
            ),
            DropdownButtonFormField<Auteur>(
              items: [
                DropdownMenuItem(
                  value: Auteur(id: 1),
                  child: Text("Auteur 1"),
                ),
                DropdownMenuItem(
                  value: Auteur(id: 2),
                  child: Text("Auteur 2"),
                ),
              ],
              onChanged: (value) {},
              decoration: const InputDecoration(labelText: "Auteur"),
            ),
            TextFormField(
              maxLines: 10,
              decoration:
                  const InputDecoration(labelText: "Description du livre"),
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {},
              child: const Text("Enregistrer"),
            ),
          ],
        ),
      ),
    );
}
```

Dans cet écran, nous avons plusieurs champs de saisi. Nous avons utilisé les widgets **TextFormField**, **DropdownButtonFormField** et **ElevatedButton**.

TextFormField permet de construire un champ de saisi. Nous avons utilisé le paramètre **decoration** pour lui assigner une **labelText** (Le text descriptif qui s'affiche dans le champ).

```
TextFormField(
  maxLines: 10,
```

```
  decoration:
      const InputDecoration(labelText: "Description du livre"),
),
```

NB: le paramètre **maxLines** permet de donner le nombre maximum de ligne que le champ doit accepter.

DropdownButtonFormField permet de construire un champ à choix multiple ou un champ déroulant. Dans ce champ, nous listerons les catégories et aussi la liste des auteurs pour les assigner au livre que nous voulons éditer. Vous devez préciser le type des objets gérés dans le champ (Ex : DropdownButtonFormField**<Categorie>**). Il contient des paramètres tels que

- **items** qui attend une liste de **DropDownMenuItem** des éléments à afficher dans la liste.
- **onChanged** qui est une fonction anonyme avec un paramètre (ici nommer « value »). Cette fonction est exécutée à chaque fois que l'utilisateur sélectionne un élément dans la liste. La valeur renvoyée peut être interceptée grâce au paramètre de la fonction anonyme (ici nommer « value »).
- **decoration** qui nous permet d'ajouter un **labelText** au champ.

```
DropdownButtonFormField<Categorie>(
  items: [
  DropdownMenuItem(
      value: Categorie(id: 1),
      child: Text("Catégorie 1"),
    ),
    DropdownMenuItem(
      value: Categorie(id: 2),
      child: Text("Catégorie 2"),
    ),
  ],
  onChanged: (value) {},
  decoration: const InputDecoration(labelText: "Catégorie"),
),
```

ElevatedButton est un autre widget qui permet de construire un bouton avec un aspect de surélévation du bouton. Nous avons utilisé le paramètre

- **onPressed** pour intercepter le clique de l'utilisateur sur le bouton et produit un traitement.
- **child** pour déclarer le text qui s'affiche dans le bouton.

```
ElevatedButton(
  onPressed: () {},
  child: const Text("Enregistrer"),
),
```

Ecran de liste des livres

```
import 'package:bibliotheca/views/edition_livre.dart';
import 'package:flutter/material.dart';

class ListeLivrePage extends StatefulWidget {
  const ListeLivrePage({Key? key}) : super(key: key);

  @override
  State<ListeLivrePage> createState() => _ListeLivrePageState();
}

class _ListeLivrePageState extends State<ListeLivrePage> {
  @override
  Widget build(BuildContext context) => Scaffold(
        appBar: AppBar(
          title: const Text("Liste des livres"),
        ),
        floatingActionButton: FloatingActionButton(
          child: const Icon(Icons.add),
          onPressed: () {
            Navigator.push(context,
                MaterialPageRoute(builder: (_) => const EditionLivre()));
          },
        ),
        body: ListView.builder(
          itemCount: 20,
          itemBuilder: (context, i) => ListTile(
            leading: const Icon(Icons.book),
            title: Text("Titre du livre $i"),
            subtitle: Text("description du livre $i ..."),
            onTap: () {},
          ),
        ),
      );
}
```

NB: **Navigator.push** permet d'ouvrir un nouvel écran.

```
Navigator.push(context,
  MaterialPageRoute(builder: (_) => const EditionLivre())
);
```

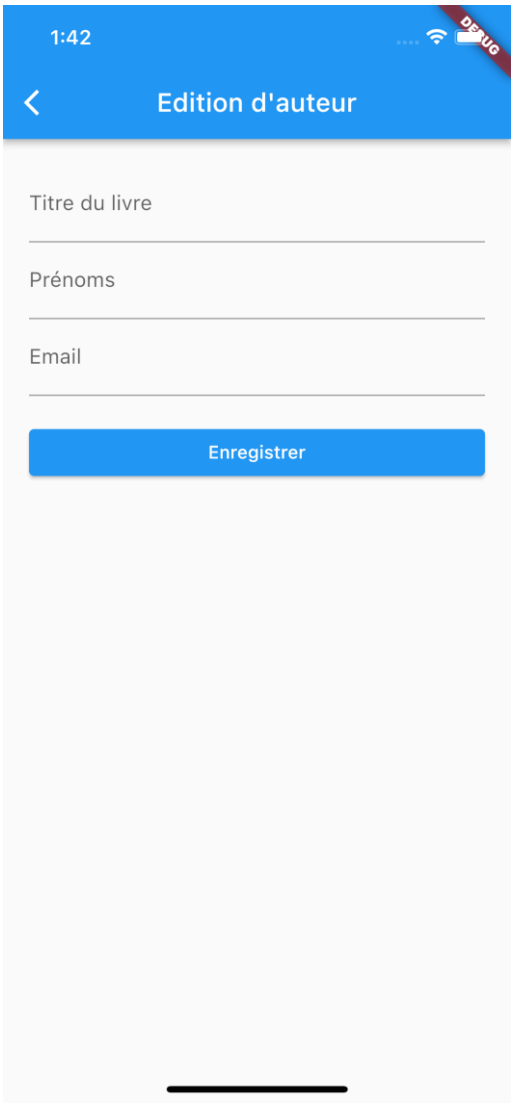
Faisons pareil pour les autres fonctionnalités de notre application.

Ecran de liste des auteurs

```
import 'package:bibliotheca/views/edition_auteur.dart';
import 'package:flutter/material.dart';

class ListeAuteur extends StatefulWidget {
  const ListeAuteur({Key? key}) : super(key: key);

  @override
  State<ListeAuteur> createState() => _ListeAuteurState();
}

class _ListeAuteurState extends State<ListeAuteur> {
  @override
```

```
  Widget build(BuildContext context) => Scaffold(
        appBar: AppBar(
          title: const Text("Liste des auteurs"),
        ),
        floatingActionButton: FloatingActionButton(
          child: const Icon(Icons.add),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (_) => const EditionAuteur(),
              ),
            );
          },
        ),
        body: ListView.builder(
          itemCount: 20,
          itemBuilder: (context, i) => ListTile(
            leading: const Icon(Icons.book),
            title: Text("Titre du livre $i"),
            subtitle: Text("description du livre $i ..."),
            onTap: () {},
          ),
        ),
      );
}
```

Ecran édition des auteurs




```dart
import 'package:flutter/material.dart';

class EditionAuteur extends StatefulWidget {
  const EditionAuteur({Key? key}) : super(key: key);

  @override
  State<EditionAuteur> createState() => _EditionAuteurState();
}

class _EditionAuteurState extends State<EditionAuteur> {
  @override
  Widget build(BuildContext context) => Scaffold(
        appBar: AppBar(
          title: const Text("Edition d'auteur"),
        ),
        body: Form(
          child: ListView(
            padding: const EdgeInsets.all(20),
            children: [
              TextFormField(
                decoration: const InputDecoration(labelText: "Titre du livre"),
              ),
```

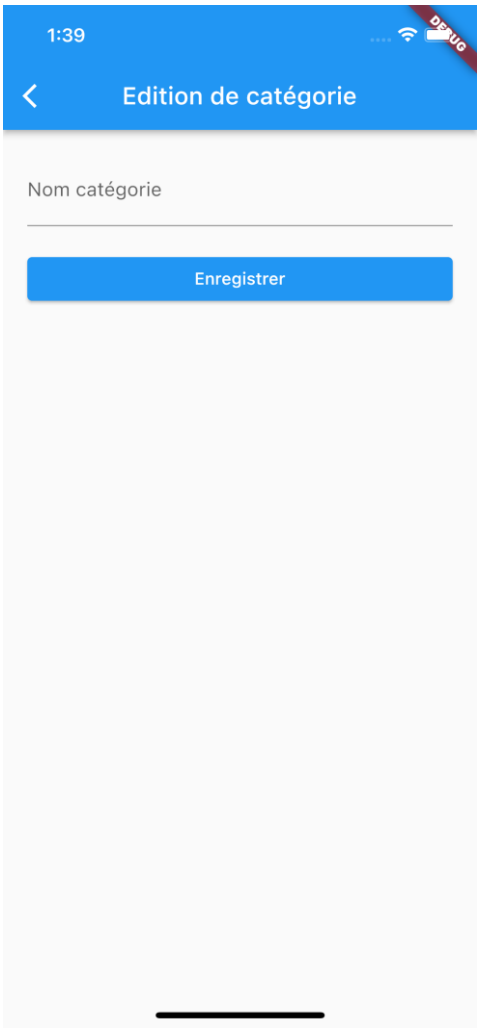
```
              TextFormField(
                decoration: const InputDecoration(labelText: "Prénoms"),
              ),
              TextFormField(
                decoration: const InputDecoration(labelText: "Email"),
              ),
              const SizedBox(height: 20),
              ElevatedButton(
                onPressed: () {},
                child: const Text("Enregistrer"),
              ),
            ],
          ),
        ),
      );
}
```

Ecran de liste des catégories


```
import 'package:bibliotheca/views/edition_categorie.dart';
import 'package:flutter/material.dart';

class ListeCategorie extends StatefulWidget {
  const ListeCategorie({Key? key}) : super(key: key);
```

```
  @override
  State<ListeCategorie> createState() => _ListeCategorieState();
}

class _ListeCategorieState extends State<ListeCategorie> {
  @override
  Widget build(BuildContext context) => Scaffold(
        appBar: AppBar(
          title: const Text("Liste des catégories"),
        ),
        floatingActionButton: FloatingActionButton(
          child: const Icon(Icons.add),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (_) => const EditionCategorie(),
              ),
            );
          },
        ),
        body: ListView.builder(
          itemCount: 20,
          itemBuilder: (context, i) => ListTile(
            leading: const Icon(Icons.book),
            title: Text("Titre de la catégorie $i"),
            onTap: () {},
          ),
        ),
      );
}
```

Ecran d'édition catégorie




```dart
import 'package:flutter/material.dart';

class EditionCategorie extends StatefulWidget {
  const EditionCategorie({Key? key}) : super(key: key);

  @override
  State<EditionCategorie> createState() => _EditionCategorieState();
}

class _EditionCategorieState extends State<EditionCategorie> {
  @override
  Widget build(BuildContext context) => Scaffold(
        appBar: AppBar(
          title: const Text("Edition de catégorie"),
        ),
        body: Form(
          child: ListView(
            padding: const EdgeInsets.all(20),
            children: [
              TextFormField(
                decoration: const InputDecoration(labelText: "Nom catégorie"),
              ),
```

```
              const SizedBox(height: 20),
              ElevatedButton(
                onPressed: () {},
                child: const Text("Enregistrer"),
              ),
            ],
          ),
        ),
      );
}
```

Dans la prochaine étape, on ajoutera des traitements pour enregistrer nos entités et aussi faire un certain nombre d'action sur eux à l'aide d'une base de données locale.