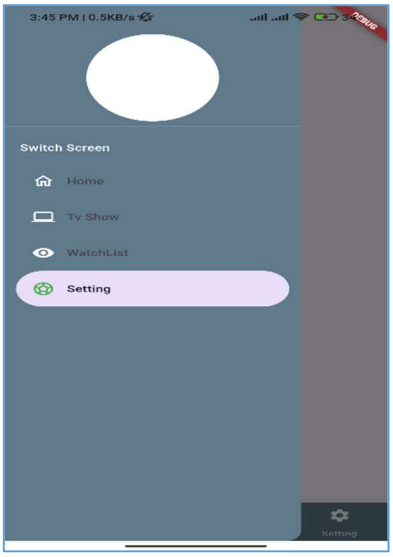
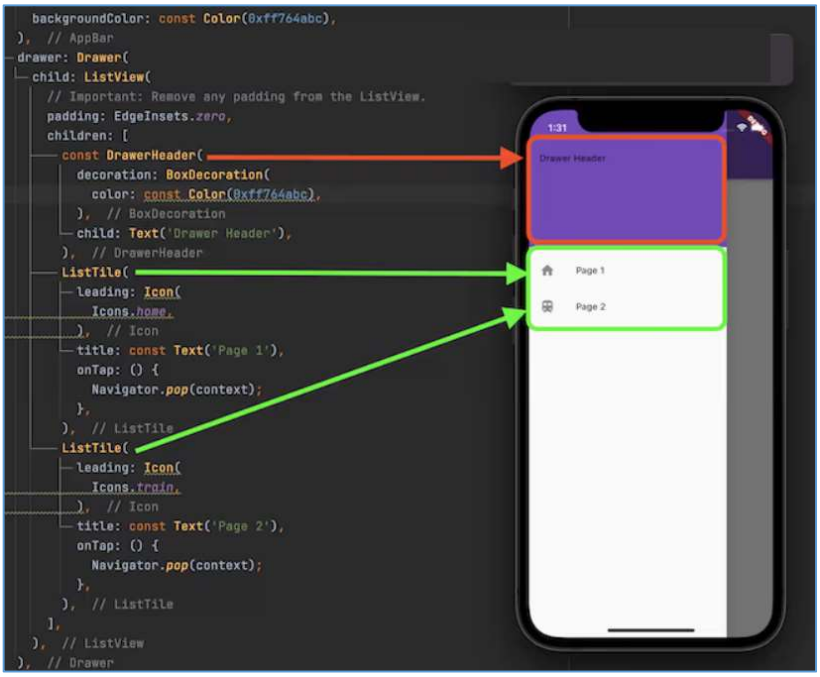
Atelier 3 Flutter
Drawer et Bottom navigation bar



I. Le Drawer

Les applications mobiles qui utilisent Material Design ont deux options principales pour la navigation. Ces navigations sont les onglets et les tiroirs (Drawer). Un Drawer est une option alternative pour les onglets car parfois les applications mobiles n'ont pas suffisamment d'espace pour prendre en charge les onglets.

Un Drawer est un écran latéral invisible. Il s'agit d'un menu coulissant de gauche qui contient généralement des liens importants dans l'application et occupe la moitié de l'écran lorsqu'il est affiché.



Le Drawer de navigation peut être utilisé comme une option alternative au widget TabBar. Il est recommandé d'utiliser un Drawer lorsque vous avez au moins cinq pages à parcourir. Si votre application comporte plusieurs pages, la navigation à l'intérieur de la barre d'onglets rend l'expérience utilisateur moins intuitive.

Pour ajouter un Drawer dans Flutter, vous devez d'abord utiliser MaterialApp dans votre projet. Ensuite, le widget Drawer peut être ajouté au widget Scaffold.

1. UserAccountsDrawerHeader

Il s'agit d'un en-tête Material Design Drawer qui identifie l'utilisateur de l'application.

Exemple

Reprenons l'application précédente. Créez le Widget /lib/**widget**/**mydrawer.dart**


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

class MyDrawer extends StatelessWidget {
  const MyDrawer({super.key});

  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: ListView(
        children: const [
          UserAccountsDrawerHeader(
            accountName: Text('Mohamed Tounsi',
                style: TextStyle(fontWeight: FontWeight.bold, fontSize: 17.0)),
            accountEmail: Text('mohamed.tounsi@gmail.com'),
            currentAccountPicture: CircleAvatar(
```

```
              backgroundImage: NetworkImage(
                  "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcRr6stJk0RCFjmfPs0XSVF6RhXl84VDRx5V5g&s"),
            ),
            decoration: BoxDecoration(color: Colors.deepPurpleAccent),
          ),
        ],
      ),
    );
  }
}
```

Explication

Dans Flutter, ListView est une liste déroulante de widgets disposés linéairement. Il affiche ses enfants les uns après les autres dans le sens du défilement, c'est-à-dire vertical ou horizontal.

ListView est le widget de défilement le plus couramment utilisé. Il affiche ses enfants les uns après les autres dans le sens du défilement. Dans l'axe transversal, les enfants doivent remplir le ListView.

UserAccountsDrawerHeader contient les options accountName, accountEmail, currentAccountPicture et decoration.

CircleAvatar est tout simplement un cercle dans lequel on peut ajouter une couleur d'arrière-plan, une image d'arrière-plan ou du texte. Il représente généralement un utilisateur avec son image ou ses initiales. Bien que l'on puisse créer un widget similaire à partir de zéro, ce widget est utile dans le développement rapide d'une application.

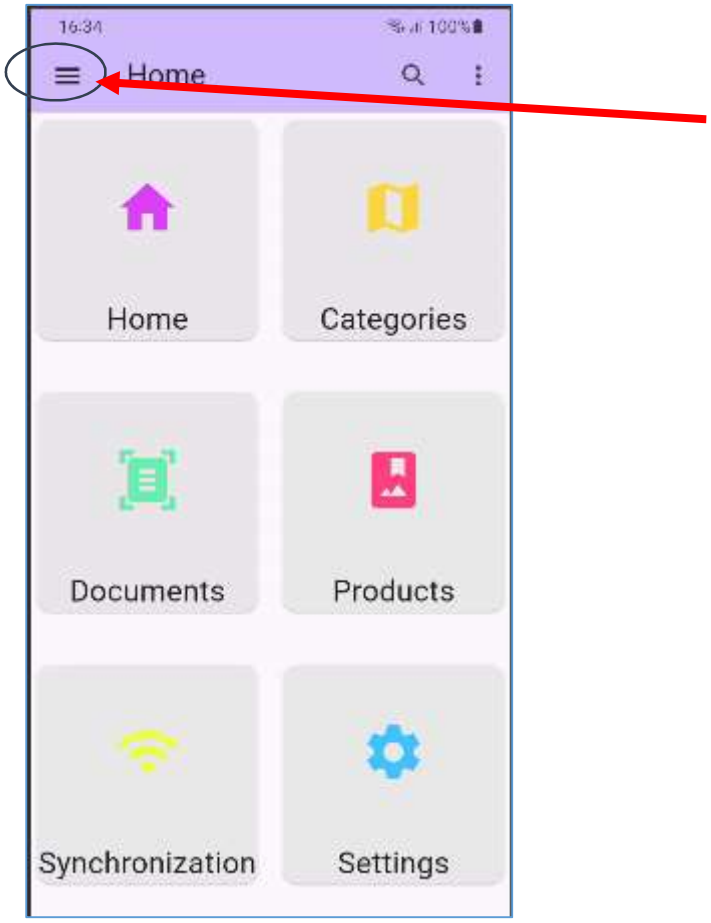
BoxDecoration est un widget intégré dans l'API flutter. À un niveau de base, il décrit comment une boîte doit être peinte à l'écran. La forme de la boîte n'a pas besoin d'être simplement un rectangle ou un carré, elle peut également être encerclée. Il est livré avec plusieurs propriétés.

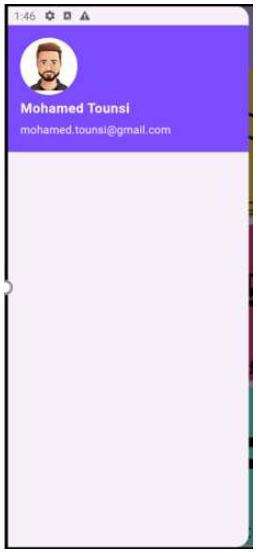
Il faut faire appel à MyDrawer dans main.dart à travers le mot clé drawer.

```
// Importation du package Flutter Material
import 'package:flutter/material.dart';
import 'package:myflutterapplication/screens/menu.dart';
import 'package:myflutterapplication/widgets/myappbar.dart';
import 'package:myflutterapplication/widgets/mydrawer.dart';

// Fonction principale qui lance l'application Flutter
void main() {
  // Lance l'application en exécutant MyApp
  runApp(const MyApp());
```

```dart
}

// Définition de la classe stateless MyApp
class MyApp extends StatelessWidget {
  // Constructeur constant avec une clé facultative
  const MyApp({super.key});

  // Ce widget est la racine de l'application
  @override
  Widget build(BuildContext context) {
    // Retourne un MaterialApp configuré
    return MaterialApp(
      // Titre de l'application
      title: 'Flutter Application',
      // Thème de l'application avec une palette de couleurs personnalisée
      theme: ThemeData(
        // Utilisation d'un ColorScheme basé sur une couleur de départ
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        // Activation de Material Design 3
        useMaterial3: true,
      ),
      // Désactivation du bandeau "DEBUG"
      debugShowCheckedModeBanner: false,
      // Définition de la page d'accueil de l'application
      home: const Scaffold(
        // AppBar de l'application
        appBar:  Myappbar(),
        // Corps du Scaffold
        body: Menu(),
        drawer : MyDrawer(),
      ),
    );
  }
}
```

Résultat





2. ListTile

ListTile contient une à trois lignes de texte facultativement flanquées d'icônes ou d'autres widgets. Il contient le titre ainsi que leader ou suivi des icônes.

Les hauteurs des widgets de début et de fin sont contraintes en fonction de la spécification Material. Une exception est faite pour les ListTiles d'une ligne pour l'accessibilité.

ListTile est généralement utilisée dans les ListViews ou disposées en colonnes dans le Drawer et les Cards.

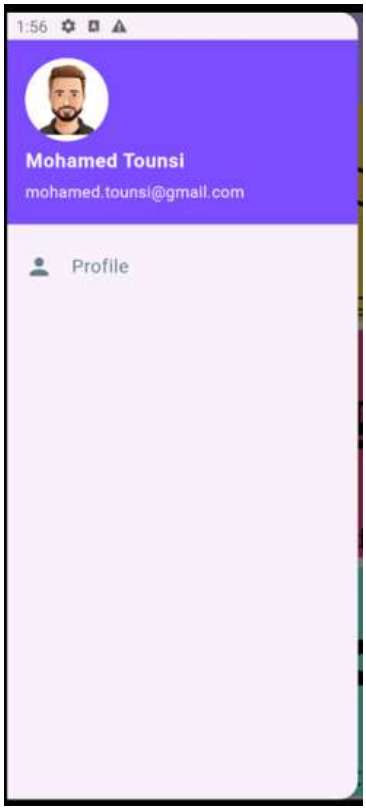
Exemple

lib > widgets > mydrawer.dart

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

class MyDrawer extends StatelessWidget {
  const MyDrawer({super.key});

  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: ListView(
        children: [
          const UserAccountsDrawerHeader(
            accountName: Text('Mohamed Tounsi',
                style: TextStyle(fontWeight: FontWeight.bold, fontSize: 17.0)),
            accountEmail: Text('mohamed.tounsi@gmail.com'),
            currentAccountPicture: CircleAvatar(
              backgroundImage: NetworkImage(
                  "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcRr6stJk0RCFjmfPs0XSVF6RhXl84VDRx5V5g&s"),
            ),
            decoration: BoxDecoration(color: Colors.deepPurpleAccent),
          ),
          ListTile(
              leading:
              const Icon(Icons.person, color: Colors.blueGrey),
              textColor: color: Colors.blueGrey,
              title: const Text('Profile'),
              onTap: () { print('Profile');  },
              ),
        ],

      ),
    );
  }
}
```

Remarque : Dans `children: [` nous avons supprimé `const` car elle n'est plus valable avec `onTap()`

Résultat



On va créer une classe choice et lui faire appel dans ListView en mettant ListTile dans une boucle.

lib > widgets > mydrawer.dart

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

class MyDrawer extends StatelessWidget {
  const MyDrawer({super.key});

  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: ListView(
        children: [
          const UserAccountsDrawerHeader(
            accountName: Text('Mohamed Tounsi',
                style: TextStyle(fontWeight: FontWeight.bold, fontSize: 17.0)),
            accountEmail: Text('mohamed.tounsi@gmail.com'),
            currentAccountPicture: CircleAvatar(
              backgroundImage: NetworkImage(
                  "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcRr6stJk0RCFjmfPs0XSVF6RhXl84VDRx5V5g&s"),
            ),
            decoration: BoxDecoration(color: Colors.deepPurpleAccent),
          ),
          // Ajout des ListTile en utilisant les objets Choice
          ...choices.map((Choice choice) {
            return ListTile(
              leading: Icon(choice.icon, color: Colors.blueGrey),
```

```
              textColor: Colors.blueGrey,
              title: Text(choice.title),
              onTap: () {
                print(choice.title);
              },
            );
          }),
        ],
      ),
    );
  }
}

class Choice {
  const Choice({required this.title, required this.icon});
  final String title;
  final IconData icon;
}

const List<Choice> choices = <Choice>[
  Choice(title: 'Profile', icon: Icons.home),
  Choice(title: 'Subscribe', icon: Icons.directions),
  Choice(title: 'Items', icon: Icons.format_list_bulleted_outlined),
  Choice(title: 'Contact', icon: Icons.email_rounded),
  Choice(title: 'Settings', icon: Icons.settings),
  Choice(title: 'Exit', icon: Icons.exit_to_app),
];
```

Explication

Le ListView dans le Drawer est peuplé par les ListTile créés dynamiquement en parcourant la liste des Choice.

choices est une liste d'objets de type Choice.

map prend une fonction anonyme (Choice choice) { ... } comme argument. Cette fonction est appliquée à chaque élément de la liste choices.

Pour chaque élément choice de la liste choices, la fonction est exécutée.

À l'intérieur de la fonction, chaque objet choice est transformé en un widget ListTile.

Le ListTile est configuré avec les propriétés de l'objet choice, comme icon pour l'icône et title pour le texte.

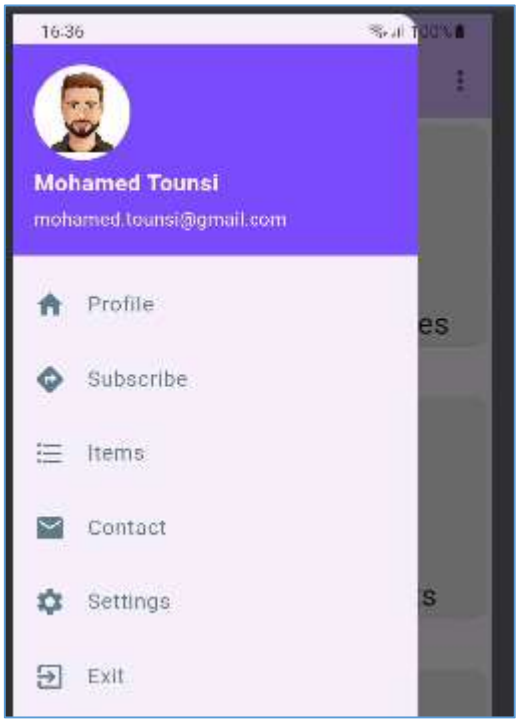
La fonction map retourne un Iterable de ListTile, où chaque ListTile correspond à un élément Choice de la liste d'origine.

L'opérateur ... en Dart est appelé l'opérateur de décomposition (ou "spread operator"). Il est utilisé pour insérer les éléments d'une collection (comme une liste ou un ensemble) dans une autre collection. Dans le

contexte de ce code, il permet d'ajouter les éléments générés par map directement dans le ListView comme s'ils faisaient partie de la liste originale.

L'opérateur ... est essentiel pour insérer chaque ListTile généré dans le children du ListView. Cela permet de créer une interface utilisateur dynamique en ajoutant des widgets à partir de données structurées.

Résultat



3. AboutListTile

Parfois, vous devrez peut-être afficher des informations supplémentaires sur l'application, telles que sa version, sa politique de confidentialité, son site Web officiel, etc. Flutter dispose d'un widget dédié appelé AboutListTile que vous pouvez afficher dans le Drawer.

Pour afficher le AboutListTile dans le Drawer :

. Ajoutez le widget AboutListTile à la fin et à l'intérieur de ListView (où vous avez des éléments ListTile pour les pages)

- À l'intérieur de AboutListTile, ajoutez l'icône et les paramètres enfants et ajoutez le widget Texte à l'intérieur de l'enfant

- Dans AboutListTile, ajoutez le paramètre applicationName et indiquez le nom de l'application

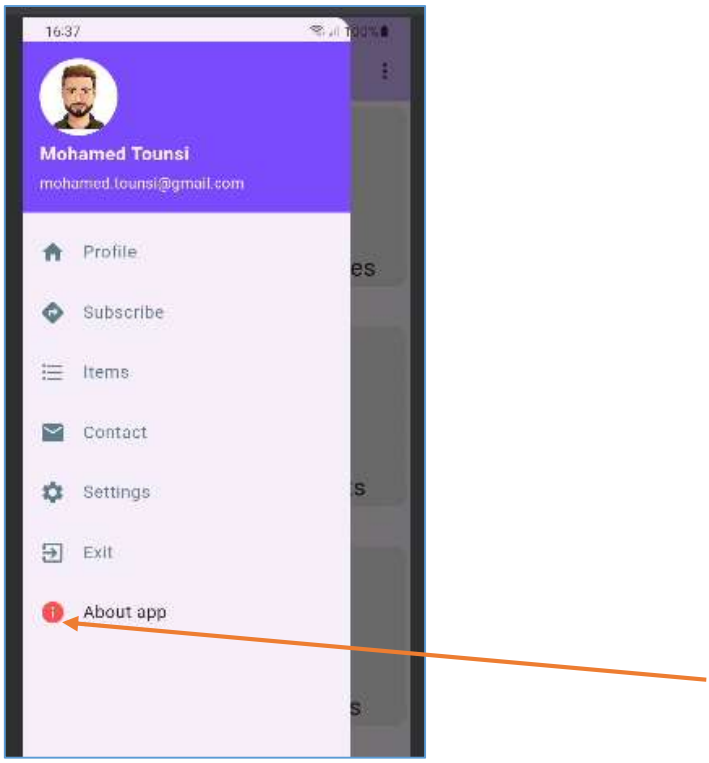
- Dans AboutListTile, ajoutez le paramètre applicationVersion et fournissez la version actuelle de l'application

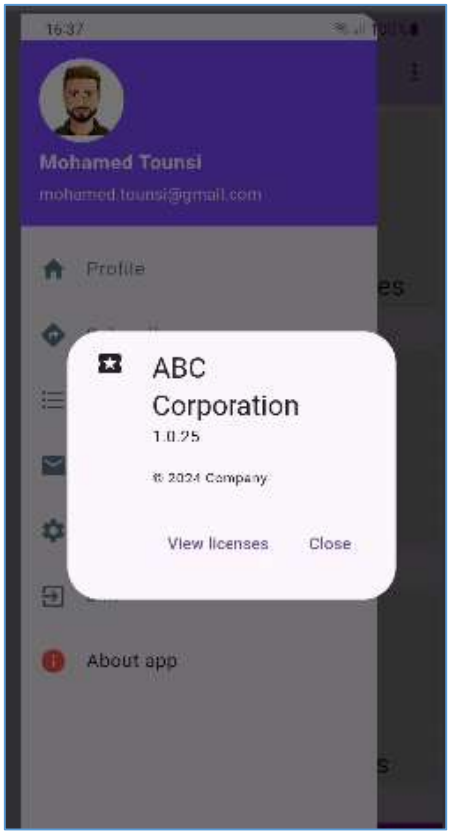
Exemple

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

class MyDrawer extends StatelessWidget {
  const MyDrawer({super.key});

  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: ListView(
        children: [
          const UserAccountsDrawerHeader(
            accountName: Text('Mohamed Tounsi',
                style: TextStyle(fontWeight: FontWeight.bold, fontSize: 17.0)),
            accountEmail: Text('mohamed.tounsi@gmail.com'),
            currentAccountPicture: CircleAvatar(
              backgroundImage: NetworkImage(
                  "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcRr6stJk0RCFjmfPs0XSVF6RhXl84VDRx5V5g&s"),
            ),
            decoration: BoxDecoration(color: Colors.deepPurpleAccent),
          ),
          // Ajout des ListTile en utilisant les objets Choice
          ...choices.map((Choice choice) {
            return ListTile(
              leading: Icon(choice.icon, color: Colors.blueGrey),
              textColor: Colors.blueGrey,
              title: Text(choice.title),
              onTap: () {
                print(choice.title);
              },
            );
          }),
          const AboutListTile(
            icon: Icon(
              Icons.info,
              color: Colors.redAccent,
            ),
            applicationIcon: Icon(
              Icons.local_play,
            ),
            applicationName: 'ABC Corporation',
            applicationVersion: '1.0.25',
            applicationLegalese: '© 2024 Company',
            child: Text('About app'),
          ),
        ],
      ),
    );
  }
}
```

```dart
class Choice {
  const Choice({required this.title, required this.icon});
  final String title;
  final IconData icon;
}

const List<Choice> choices = <Choice>[
  Choice(title: 'Profile', icon: Icons.home),
  Choice(title: 'Subscribe', icon: Icons.directions),
  Choice(title: 'Items', icon: Icons.format_list_bulleted_outlined),
  Choice(title: 'Contact', icon: Icons.email_rounded),
  Choice(title: 'Settings', icon: Icons.settings),
  Choice(title: 'Exit', icon: Icons.exit_to_app),
];
```

Résultat





II. Bottom navigation bar

La barre de navigation inférieure de Flutter peut contenir plusieurs éléments tels que des étiquettes de texte, des icônes ou les deux. Il permet à l'utilisateur de naviguer rapidement entre les vues de niveau supérieur d'une application. Si nous utilisons un écran plus grand, il est préférable d'utiliser une barre de navigation latérale.

Dans l'application Flutter, nous définissons généralement la barre de navigation inférieure en conjonction avec le widget Scaffold. Le widget Scaffold fournit un argument Scaffold.bottomNavigationBar pour définir la barre de navigation inférieure. Il est à noter que seul l'ajout de BottomNavigationBar n'affichera pas les éléments de navigation. Il est nécessaire de définir la propriété BottomNavigationItems pour Items qui accepte une liste de widgets BottomNavigationItems.

Créer le fichier /lib/**widgets**/**mybottomnavbar.dart**


```dart
// Importation du package Flutter Material
import 'package:flutter/material.dart';

// Définition de la classe stateless MyBottomNavigation
```

```dart
class MyBottomNavigation extends StatelessWidget {
  // Constructeur constant avec une clé facultative
  const MyBottomNavigation({super.key});

  @override
  Widget build(BuildContext context) {
    // Retourne un widget BottomNavigationBarWidget
    return const BottomNavigationBarWidget();
  }
}

// Définition de la classe stateful BottomNavigationBarWidget
class BottomNavigationBarWidget extends StatefulWidget {
  // Constructeur constant avec une clé facultative
  const BottomNavigationBarWidget({super.key});

  @override
  // Création de l'état associé à ce widget
  State<BottomNavigationBarWidget> createState() =>
      _BottomNavigationBarWidgetState();
}

// Classe état pour BottomNavigationBarWidget
class _BottomNavigationBarWidgetState
    extends State<BottomNavigationBarWidget> {
  // Index de l'élément actuellement sélectionné dans la barre de navigation
  int _selectedIndex = 0;

  // Méthode appelée lorsque l'utilisateur tape sur un élément de la barre de
navigation
  void _onItemTapped(int index) {
    // Mise à jour de l'état avec le nouvel index sélectionné
    setState(() {
      _selectedIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    // Retourne un BottomNavigationBar
    return BottomNavigationBar(
      // Liste des éléments de la barre de navigation
      items: const <BottomNavigationBarItem>[
        // Élément de la barre de navigation pour la page d'accueil
        BottomNavigationBarItem(
          icon: Icon(Icons.home),
          label: 'Home',
          backgroundColor: Colors.purple,
        ),
        // Élément de la barre de navigation pour les documents
        BottomNavigationBarItem(
          icon: Icon(Icons.add_chart),
```

```
          label: 'Documents',
          backgroundColor: Colors.teal,
        ),
        // Élément de la barre de navigation pour les produits
        BottomNavigationBarItem(
          icon: Icon(Icons.all_inbox_sharp),
          label: 'Products',
          backgroundColor: Colors.pink,
        ),
        // Élément de la barre de navigation pour les paramètres
        BottomNavigationBarItem(
          icon: Icon(Icons.settings),
          label: 'Settings',
          backgroundColor: Colors.green,
        ),
      ],
      // Index de l'élément actuellement sélectionné
      currentIndex: _selectedIndex,
      // Couleur de l'élément sélectionné
      selectedItemColor: Colors.amber[800],
      // Appel de la méthode lorsque l'utilisateur tape sur un élément
      onTap: _onItemTapped,
    );
  }
}
```

Explication

Dans ce code, il est approprié que BottomNavigationBarWidget soit un widget StatefulWidget.

La classe _BottomNavigationBarWidgetState contient une variable _selectedIndex qui garde la trace de l'élément actuellement sélectionné dans le BottomNavigationBar. Lorsque l'utilisateur tape sur un élément, la méthode _onItemTapped est appelée pour mettre à jour l'index sélectionné via setState(). Cette mise à jour de l'état entraîne le rafraîchissement de l'interface utilisateur pour refléter la sélection actuelle.

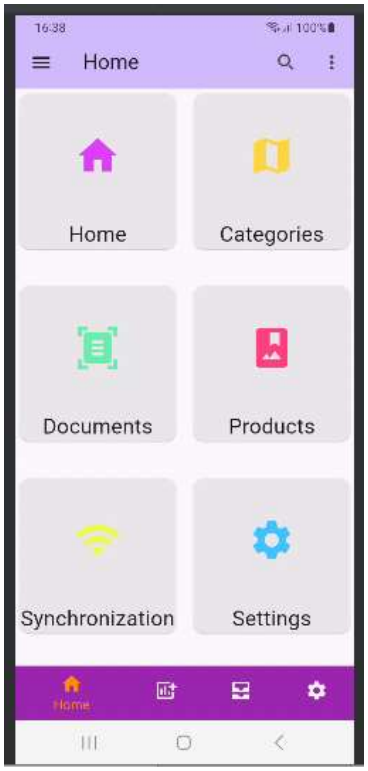
L'appel à setState() dans _onItemTapped permet au widget de se reconstruire avec le nouvel index sélectionné. Cela n'est possible qu'avec un widget StatefulWidget, car un StatelessWidget ne peut pas maintenir ou gérer un état interne mutable.

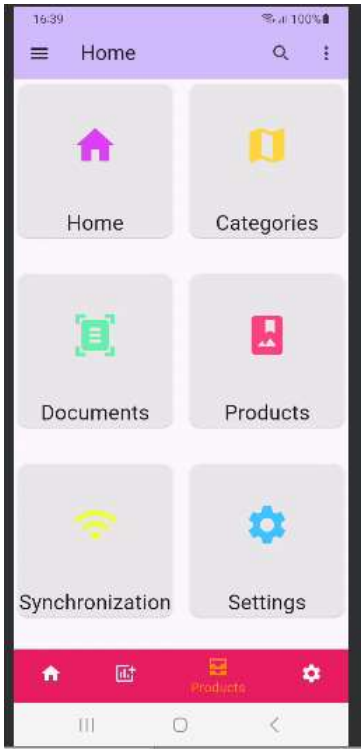
Faire appel à `MyBottomNavigation()`

```
// Importation du package Flutter Material
import 'package:flutter/material.dart';
import 'package:myflutterapplication/screens/menu.dart';
import 'package:myflutterapplication/widgets/myappbar.dart';
import 'package:myflutterapplication/widgets/mybottomnavbar.dart';
```

```dart
import 'package:myflutterapplication/widgets/mydrawer.dart';

// Fonction principale qui lance l'application Flutter
void main() {
  // Lance l'application en exécutant MyApp
  runApp(const MyApp());
}

// Définition de la classe stateless MyApp
class MyApp extends StatelessWidget {
  // Constructeur constant avec une clé facultative
  const MyApp({super.key});

  // Ce widget est la racine de l'application
  @override
  Widget build(BuildContext context) {
    // Retourne un MaterialApp configuré
    return MaterialApp(
      // Titre de l'application
      title: 'Flutter Application',
      // Thème de l'application avec une palette de couleurs personnalisée
      theme: ThemeData(
        // Utilisation d'un ColorScheme basé sur une couleur de départ
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        // Activation de Material Design 3
        useMaterial3: true,
      ),
      // Désactivation du bandeau "DEBUG"
      debugShowCheckedModeBanner: false,
      // Définition de la page d'accueil de l'application
      home: const Scaffold(
        // AppBar de l'application
        appBar:  Myappbar(),
        // Corps du Scaffold
        body: Menu(),
        drawer : MyDrawer(),
        bottomNavigationBar: MyBottomNavigation(),
      ),
    );
  }
}
```

Résultat


16:38
100%
Home
Home
Categories
Documents
Products
Synchronization
Settings
Home


16:39
100%
Home
Home
Categories
Documents
Products
Synchronization
Settings
Products