TP 4

- # null safety

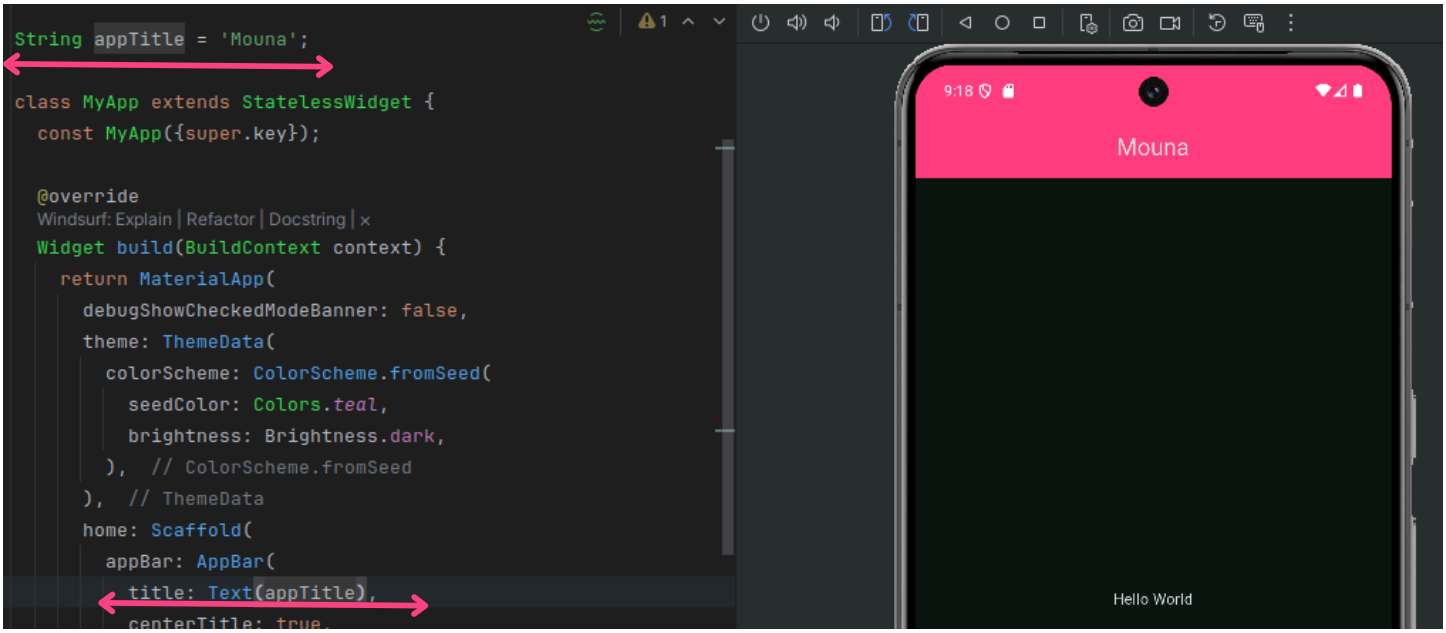
Dart's **null safety** rend les types **non nullables par défaut**. Une variable de type String ne peut pas être null. Si une valeur peut être absente, vous devez le dire avec **?** => **String?**

Non-nullable vs nullable

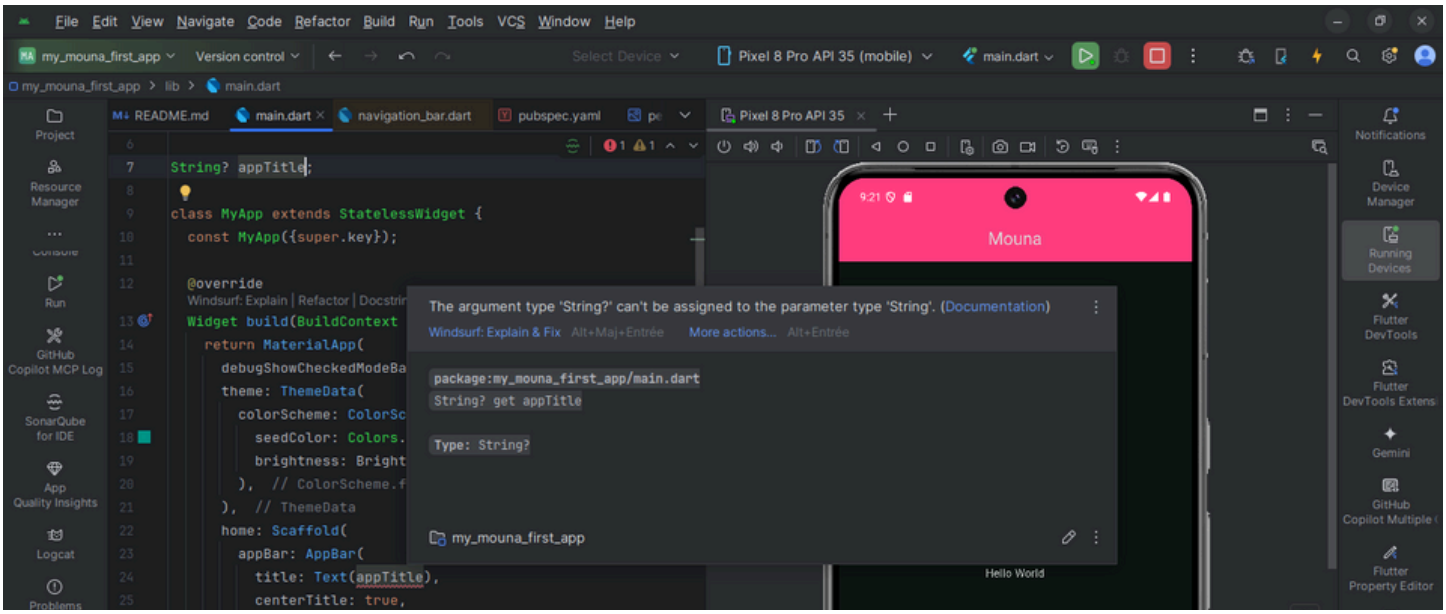
String name = 'Rita'; **=> must be non-null**

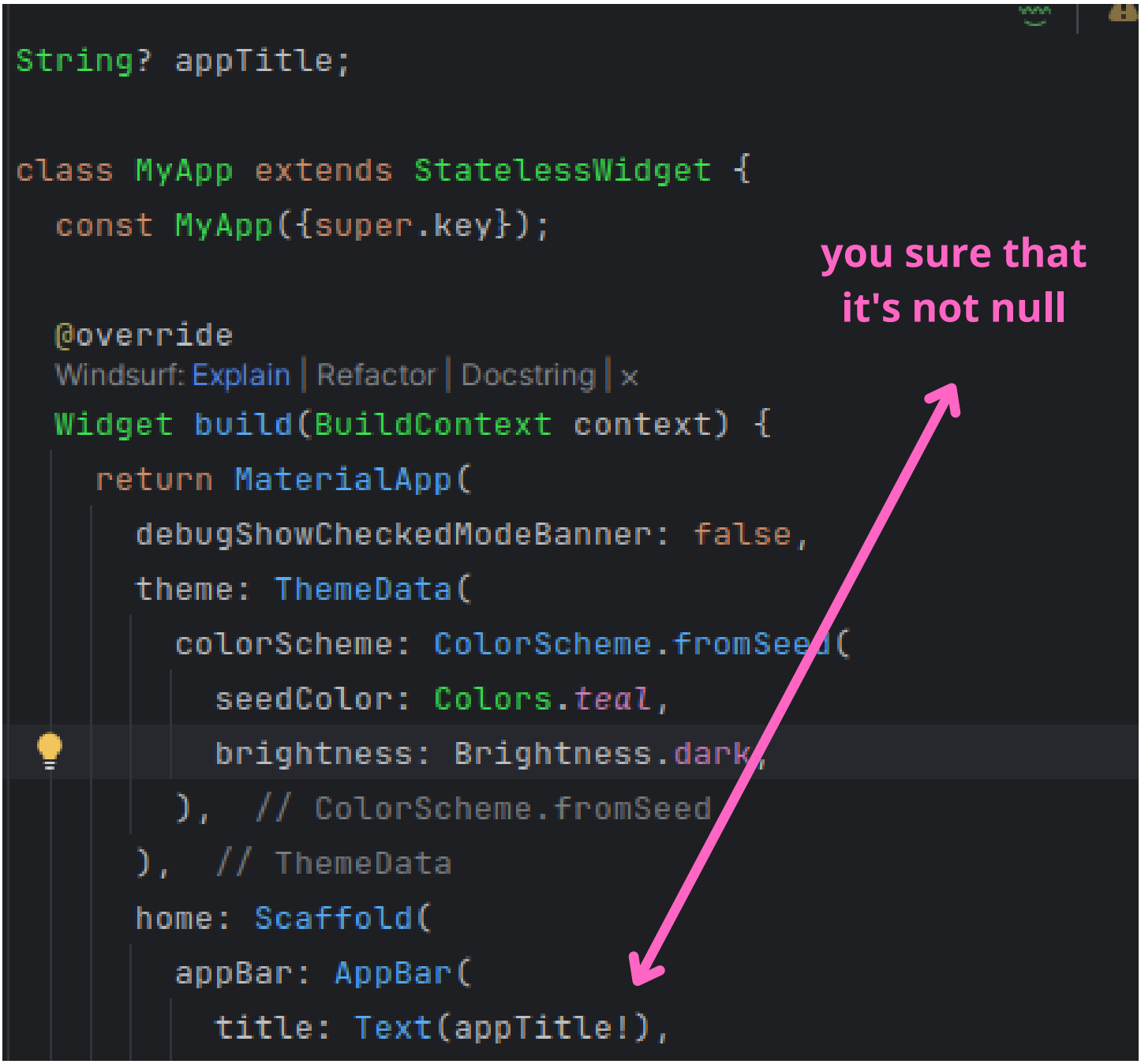
String? nickname; **=> may be null**

- Make nullable: **T?**
- Guard: **if (x == null) return;**
- Default: **x ?? default**
- Assign if null: **x ??= y**
- Force unwrap (avoid unless proven): **!**



si nous disons flutter cette valeur **peut être null**





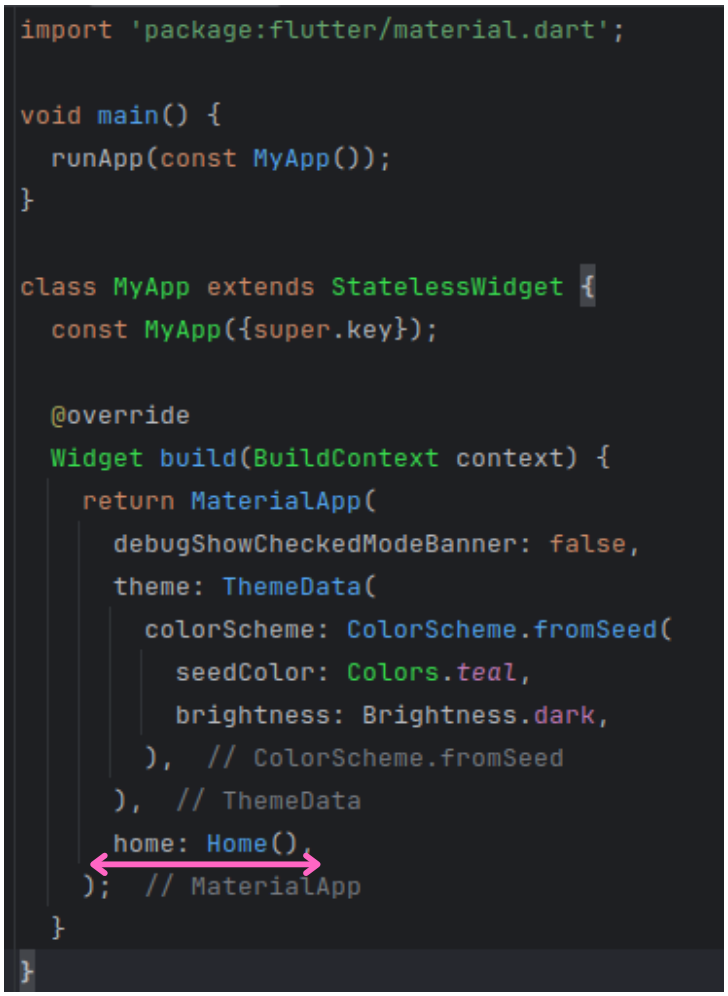
- **the statefull :** l'écran peut rafraîchir

- **the stateless :** cela ne peut pas rafraîchir l'écran
- **the setstate :** pour rafraîchir l'écran, il appelle le **widget statefull**

nous allons créer une nouvelle classe et celle-ci s'appellera la première … nous utiliserons un widget statefull

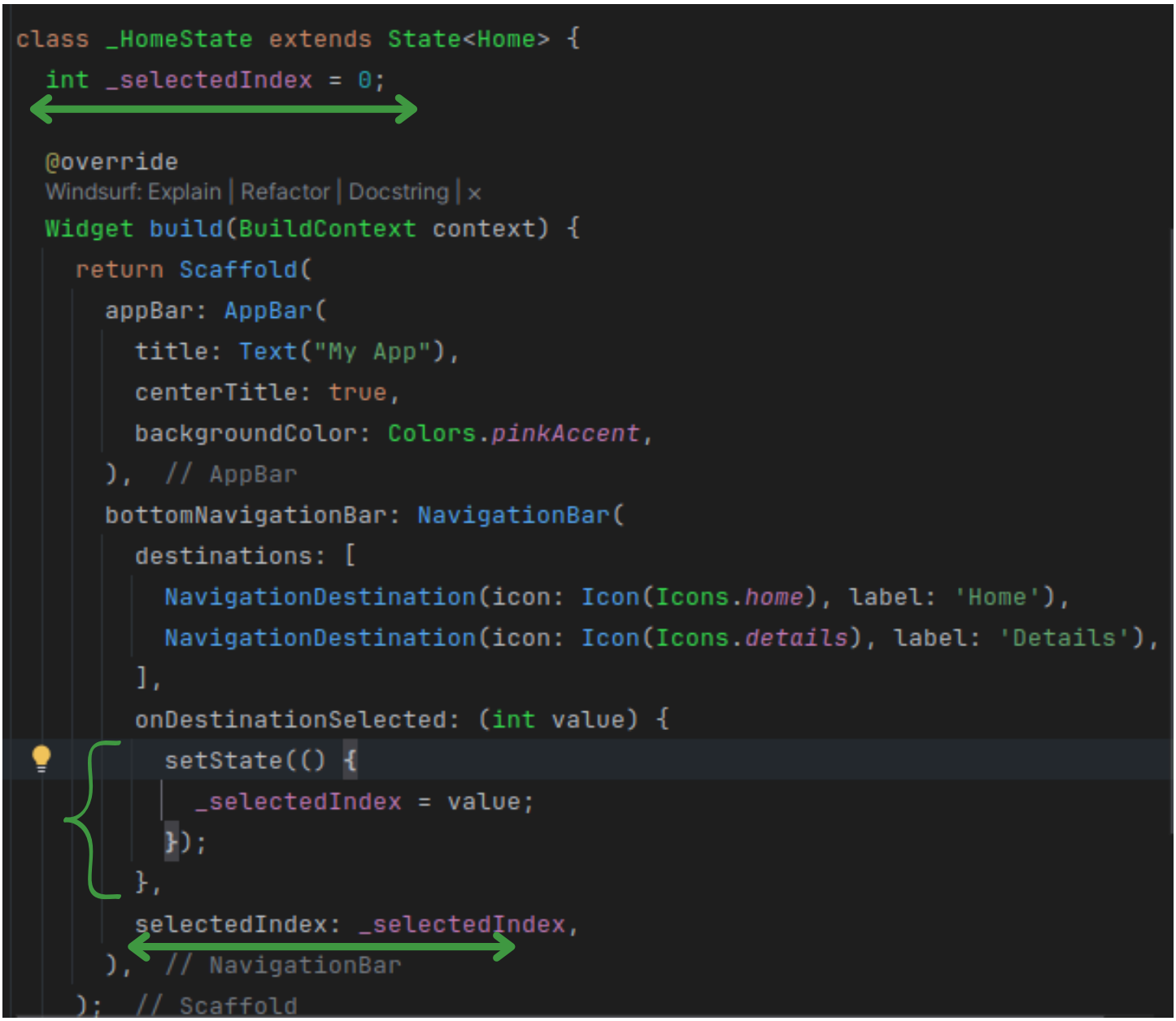
- sous la premiere classe je vais en créer un autre : **clic stl+ tab**
- changer le nom avec: **Home**
- mettre le Scaffold de 1ère classe dans le retun du 2ème
- dans la 1ère classe fais un appel au 2ème classe

```
  const Home({super.key});
  @override
  State<Home> createState() => _HomeState();
}

class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("My App"),
        centerTitle: true,
        backgroundColor: Colors.pinkAccent,
      ),  // AppBar
      bottomNavigationBar: NavigationBar(
        destinations: [
          NavigationDestination(icon: Icon(Icons.home), label: 'Home'),
          NavigationDestination(icon: Icon(Icons.details), label: 'Details'),
        ],
        onDestinationSelected: (int value) {
          print('select $value');
        },
      ),  // NavigationBar
      body: Center(child: Text('Hello World')),
    );  // Scaffold
  }
```



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

void main() {
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(
          seedColor: Colors.teal,
          brightness: Brightness.dark,
        ),  // ColorScheme.fromSeed
      ),  // ThemeData
      home: Home(),
    );  // MaterialApp
  }
}
```

maintenant nous avons le **widget statefull** et nous pouvons **le réinitialiser ou nous pouvons rafraîchir l'écran** pour changer la barre de navigation



int _selectedIndex = 0; => **pour savoir** quelle distination est sélectionnée (Home / details)

onDestinationSelected: (int value) { => **Callback** exécuté quand une destination du menu est sélectionnée.

setState(() { => **Met à jour l'état** du widget pour refléter le changement.

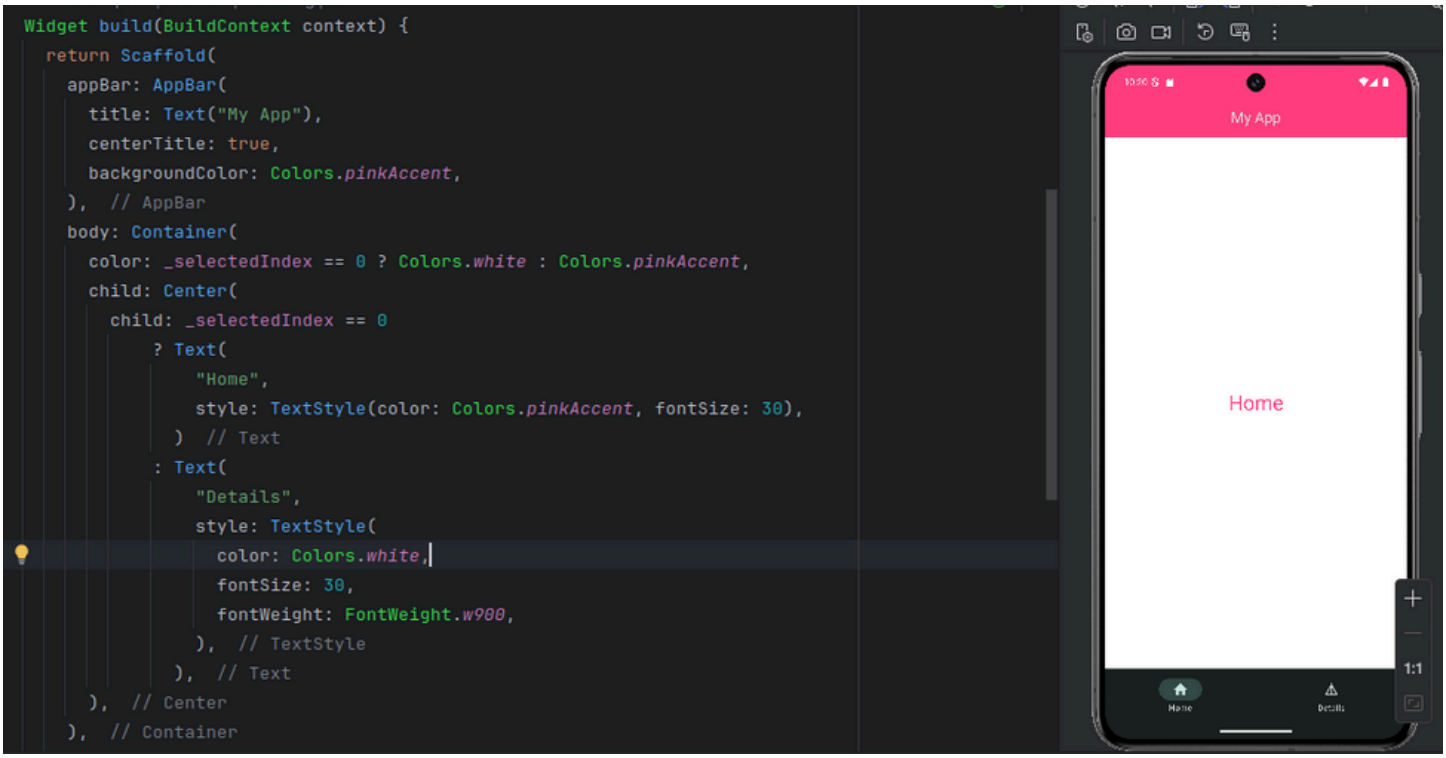
_selectedIndex = value; => **Enregistre** l'index sélectionné dans la variable d'état.

});

},

selectedIndex: _selectedIndex, => **Indique quelle destination** est actuellement sélectionnée.

=> si nous voulons changer le corps de chaque destination:



- ## split your widgets

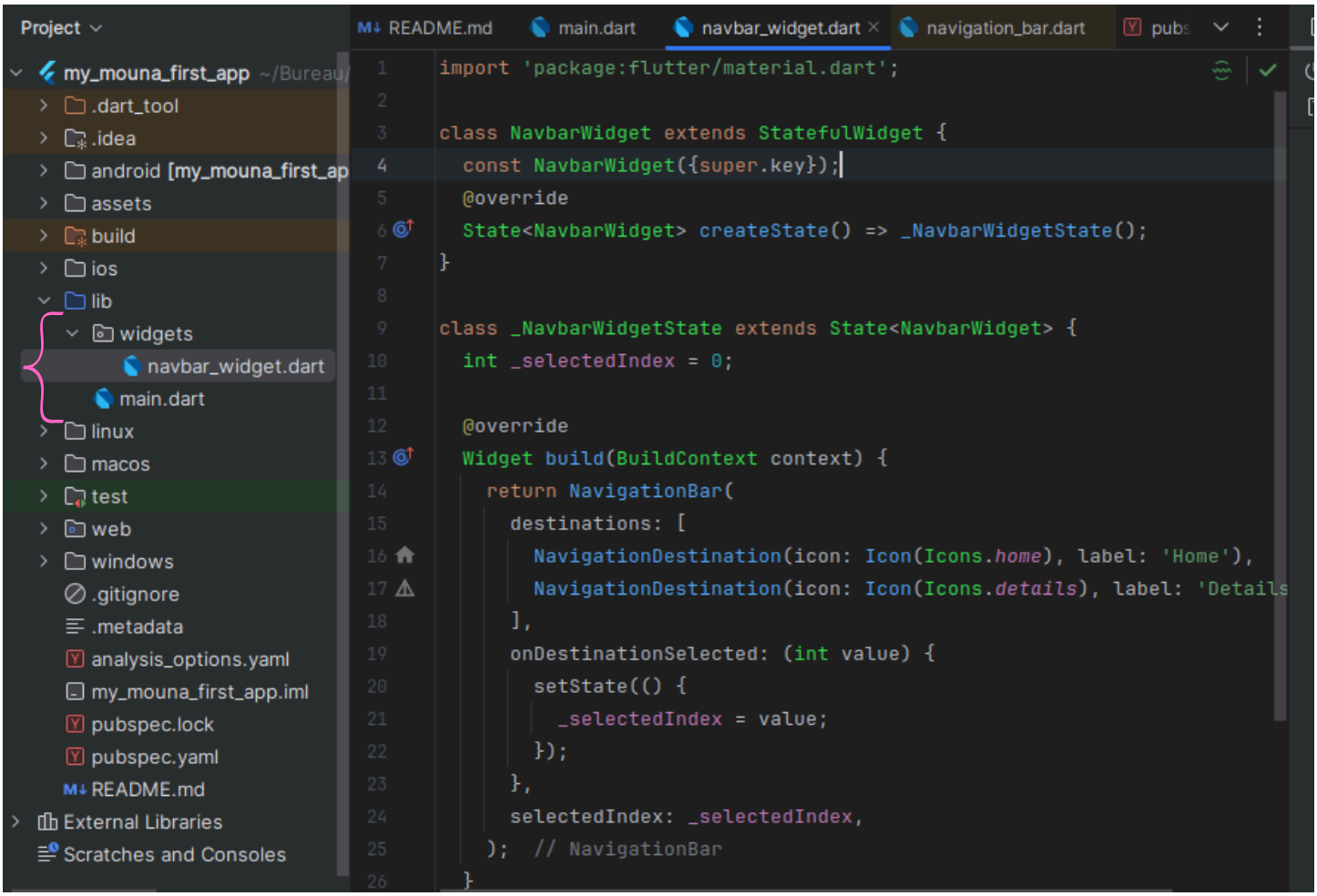
donc avec **flutter** l'objectif est de créer **le plus petit widget possible** donc vous devrez diviser vos widgets dans **différents fichiers**

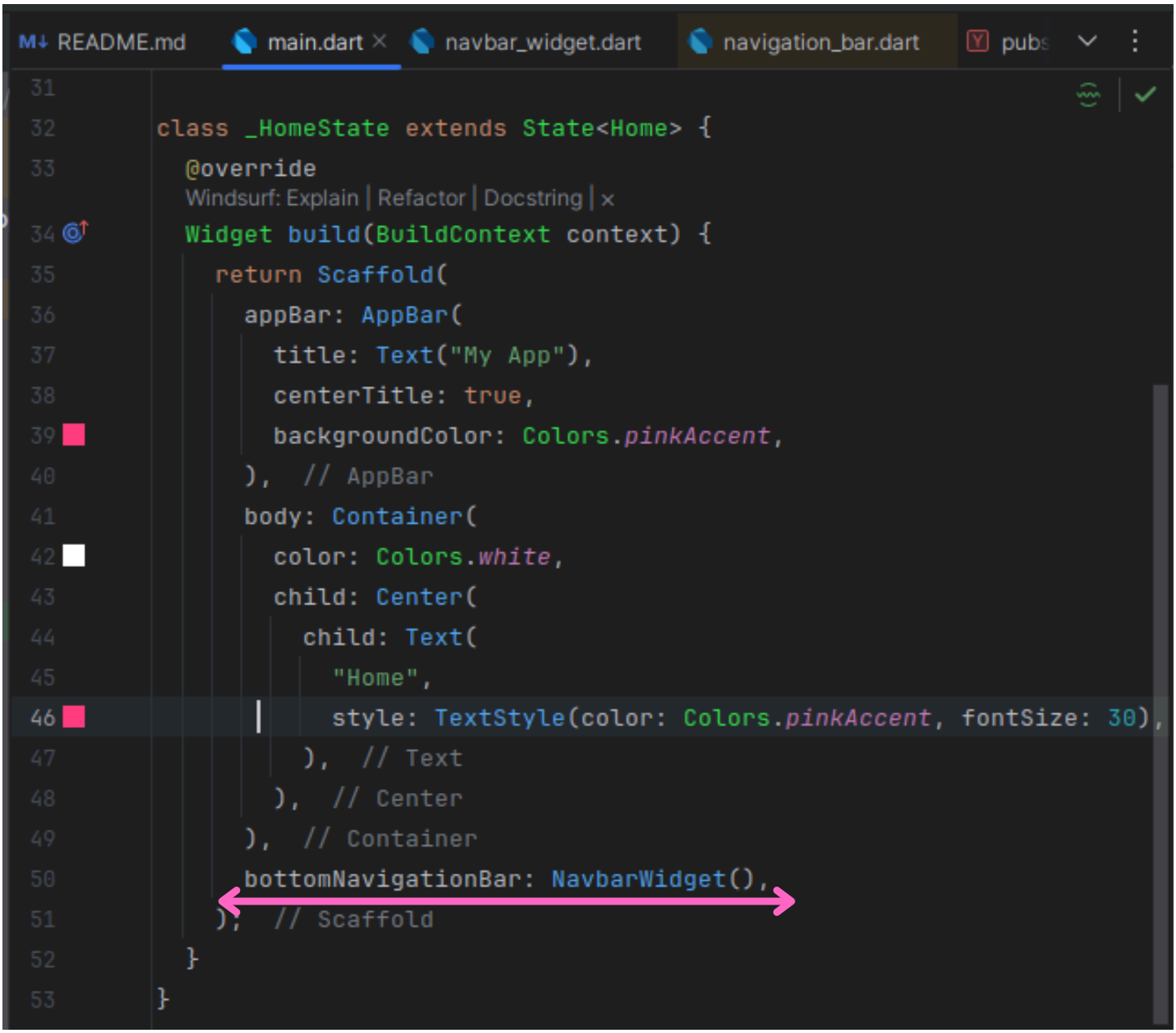
==> la raison est par exemple lorsque vous cliquez sur **setState** et que vous actualisez ce bouton pour simplement actualiser l'écran, vous n'avez pas besoin **de tout rafraîchir**

parce qu'en ce moment, lorsque vous cliquez dessus, cela va aller à la **build** et rafraîchir le **material app**, rafraîchir le **scaffold** et **tout rafraîchir**, **ce qui ralentira votre application .**

====> donc ce que vous devez faire est divisé en différents widgets:

- aller à l'intérieur de la **lib** et **créer nouveau dossier** ; **widgets** => ce sera un widget divisé

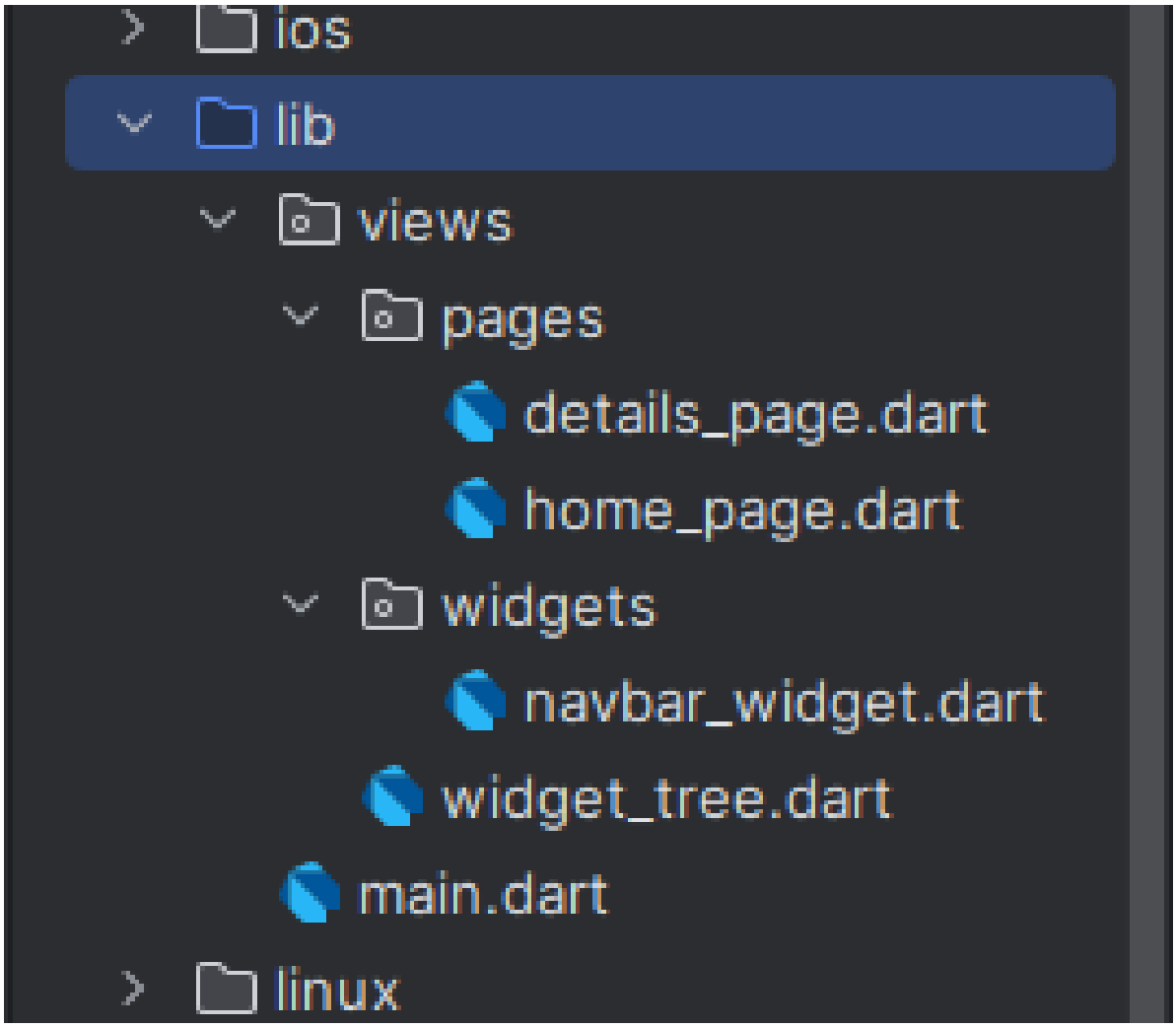

Project
my_mouna_first_app ~/Bureau/
.dart_tool
.idea
android [my_mouna_first_ap
assets
build
ios
lib
widgets
navbar_widget.dart
main.dart
linux
macos
test
web
windows
.gitignore
.metadata
analysis_options.yaml
my_mouna_first_app.iml
pubspec.lock
pubspec.yaml
README.md
External Libraries
Scratches and Consoles
README.md
main.dart
navbar_widget.dart
navigation_bar.dart
import 'package:flutter/material.dart';
class NavbarWidget extends StatefulWidget {
const NavbarWidget({super.key});
@override
State<NavbarWidget> createState() => _NavbarWidgetState();
}
class _NavbarWidgetState extends State<NavbarWidget> {
int _selectedIndex = 0;
@override
Widget build(BuildContext context) {
return NavigationBar(
destinations: [
NavigationDestination(icon: Icon(Icons.home), label: 'Home'),
NavigationDestination(icon: Icon(Icons.details), label: 'Details
],
onDestinationSelected: (int value) {
setState(() {
_selectedIndex = value;
});
},
selectedIndex: _selectedIndex,
); // NavigationBar
}



maintenant si nous voulons avoir à l'intérieur de **scaffold** un corps different dans chaque destination => comment avons-nous la valeur qui se trouve dans le **widget de barre de navigation** ? comment nous avons accès à cette information **(_selectedIndex)?**

- # widget tree

donc maintenant nous avons un **problème**, nous avons les **données à l'intérieur du widget** de la **barre de navigation**, mais nous devons aussi avoir **les données à l'intérieur** de la **main**, donc d'abord ce que nous ferons, c'est séparer **scaffold**

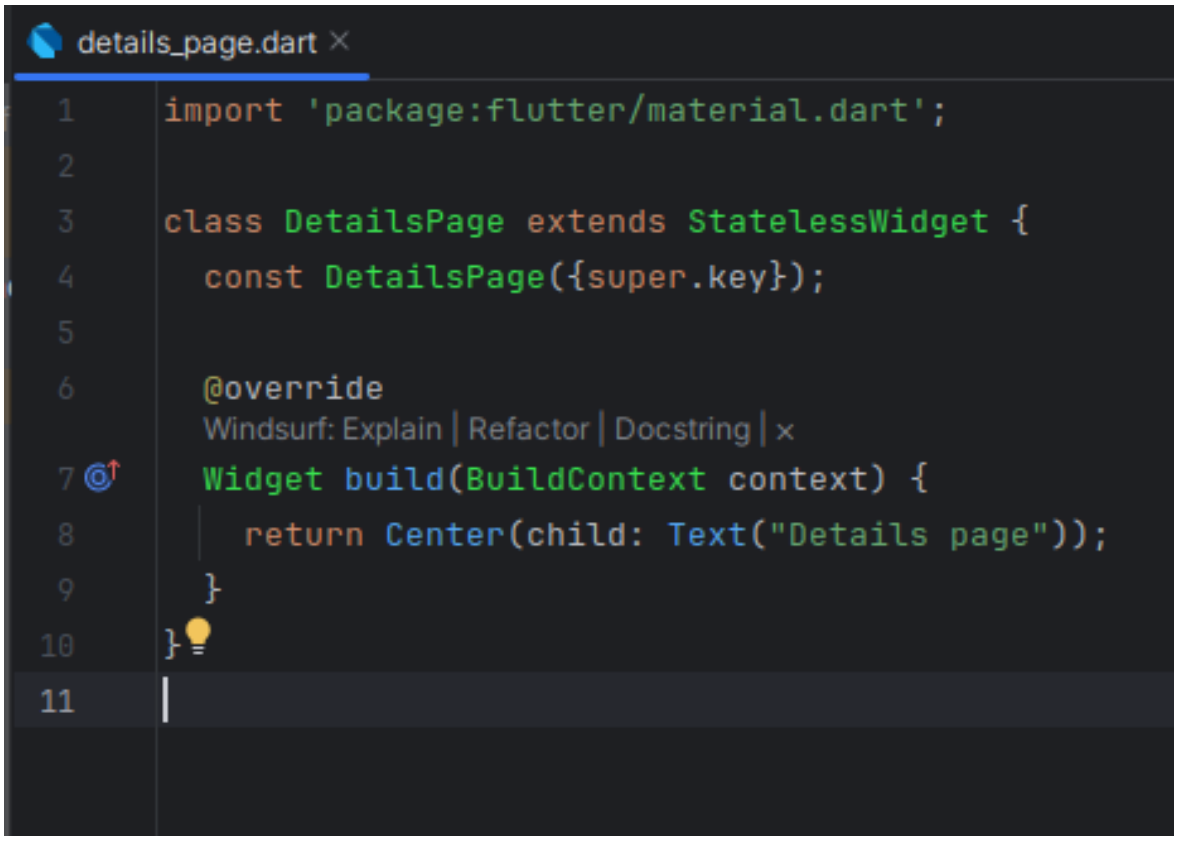

ios
lib
views
pages
details_page.dart
home_page.dart
widgets
navbar_widget.dart
widget_tree.dart
main.dart
linux

widget_tree.dart

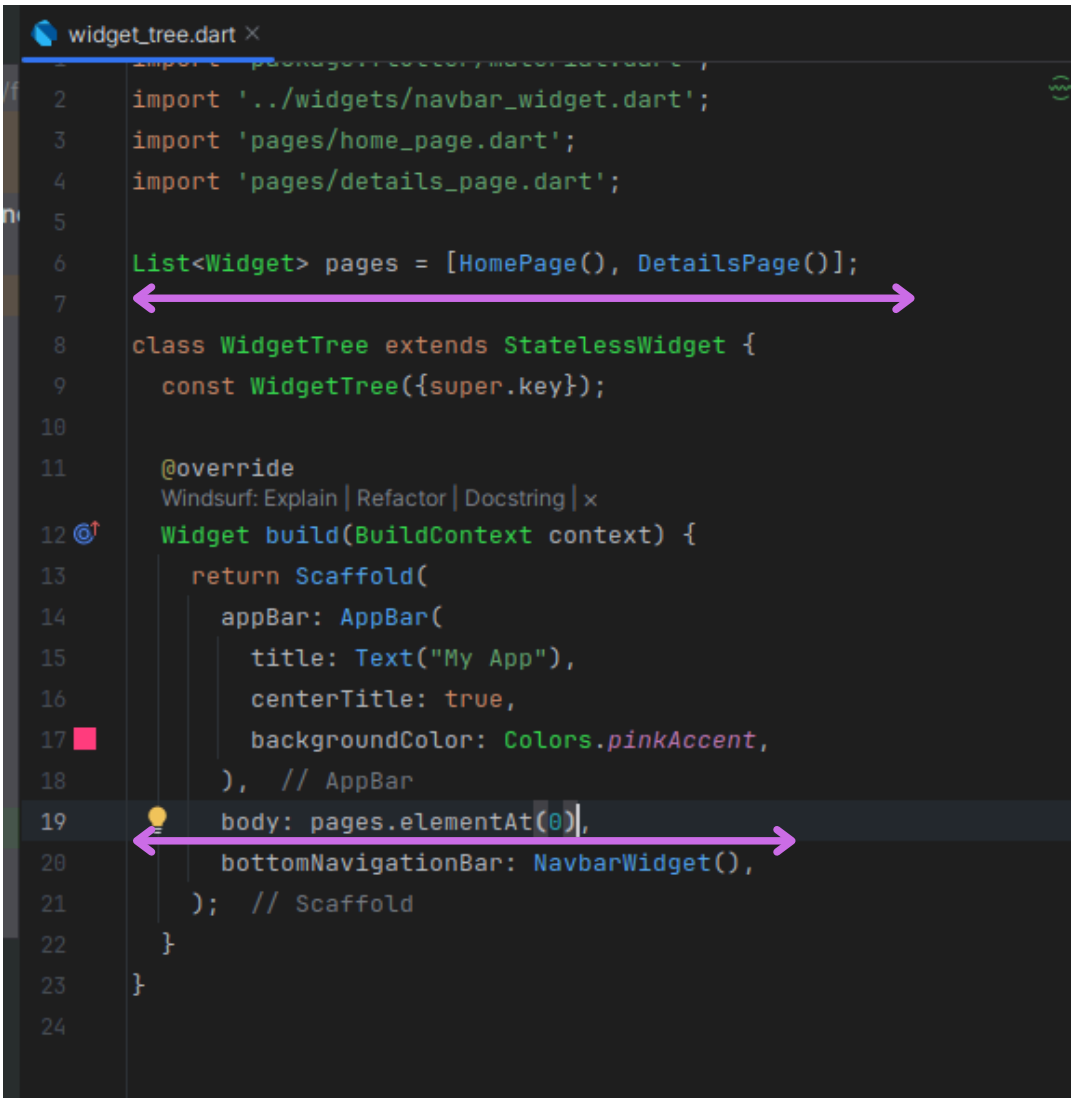
```dart
import 'package:flutter/material.dart';
import '../widgets/navbar_widget.dart';

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

  @override
  Windsurf: Explain | Refactor | Docstring | ×
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("My App"),
        centerTitle: true,
        backgroundColor: Colors.pinkAccent,
      ),  // AppBar
      bottomNavigationBar: NavbarWidget(),
    );  // Scaffold
  }
}
```

home_page.dart

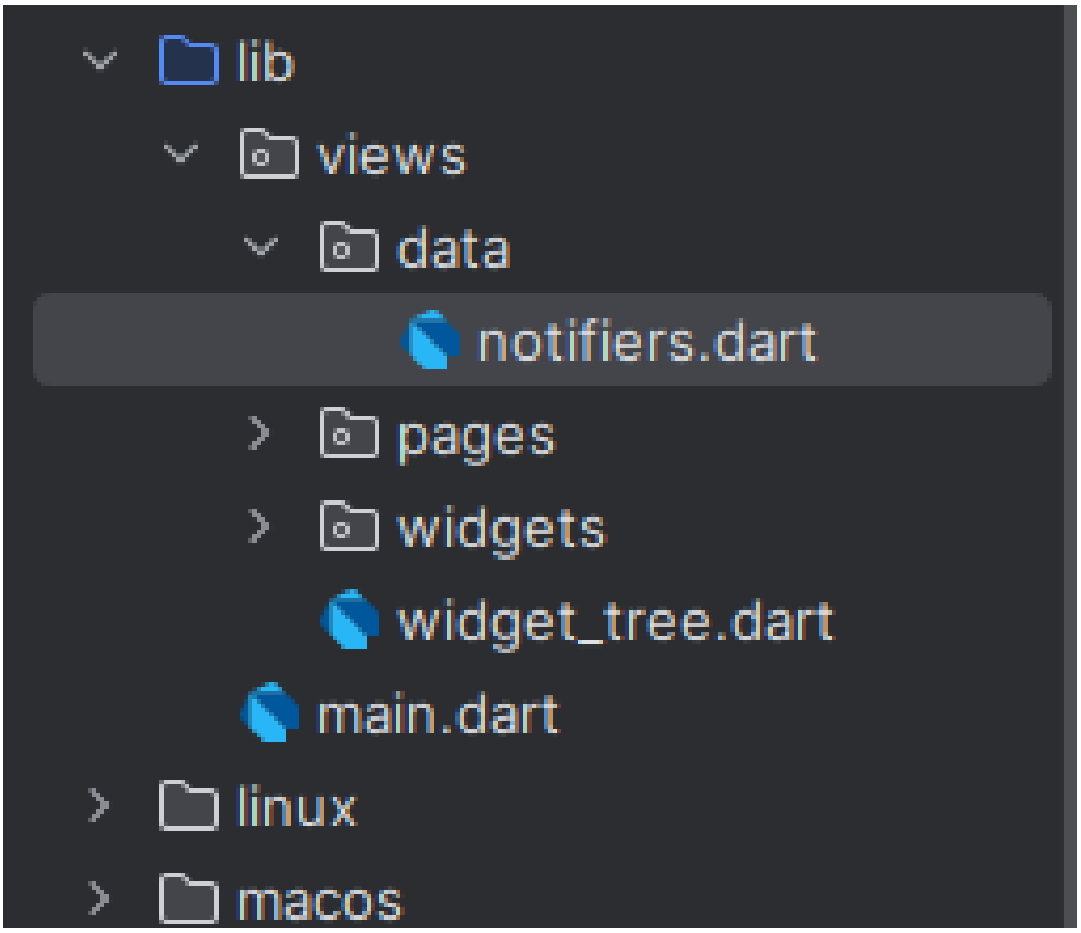
```dart
import 'package:flutter/material.dart';

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

  @override
  Windsurf: Explain | Refactor | Docstring | ×
  Widget build(BuildContext context) {
    return Center(child: Text("Home page"));
  }
}
```



=> donc nous avons séparé le **widget Scaffold et les 2 pages** dans des fichiers différents maintenant comment accéder aux deux pages dans le **body** ?



d'accord, alors comment pouvez-vous trouver toutes les données **de n'importe où** dans votre application ..... Pour l'instant, nous avons **seulement l'index sélectionné disponible dans le widget de la barre de navigation**, mais nous en avons besoin à **l'intérieur de la vue du widget** afin d'afficher soit le premier, soit le deuxième élément .

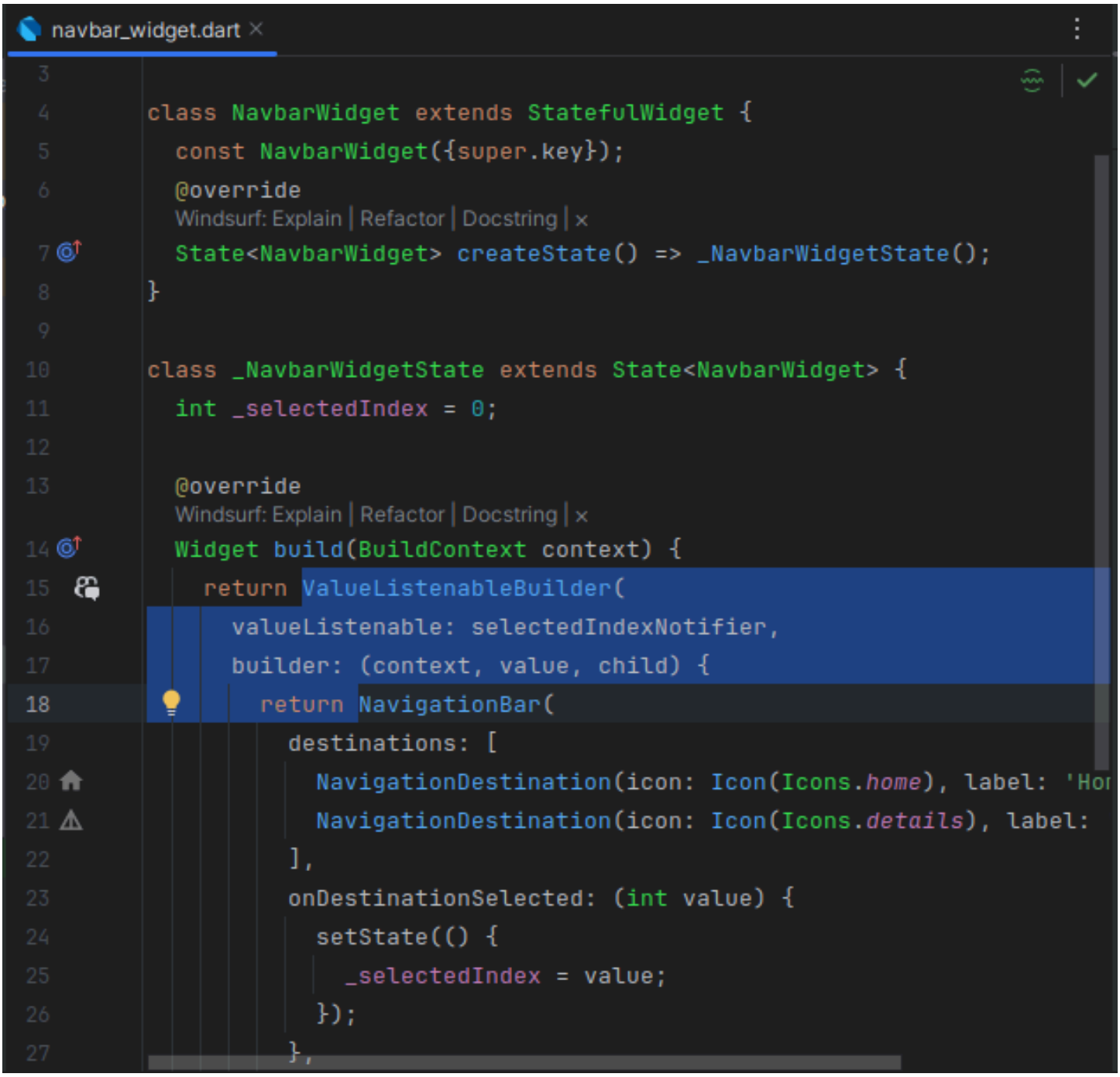


lorsque vous utilisez des **notificateurs** ou lorsque vous souhaitez mettre **tous les données au même endroit** **on dois utiliser :**

- **valueNotifier :** tenir les données (hold the data)
- **valueListenableBuilder:** il écoutera les données donc si les données changent, cela changera et dans ce cas vous n'avez pas besoin de **setState()**

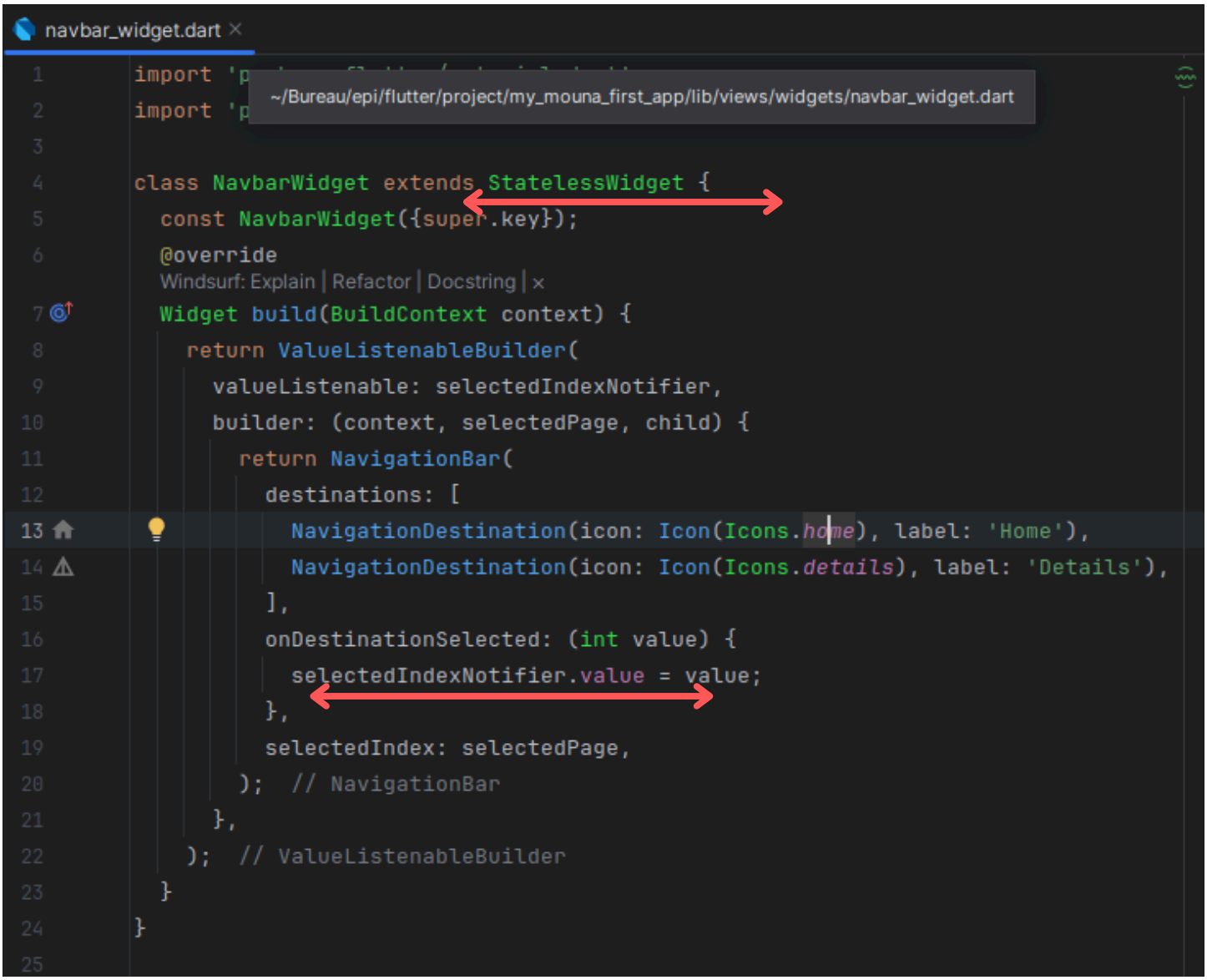
notifiers.dart

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

ValueNotifier<int> selectedIndexNotifier = ValueNotifier(0);

```

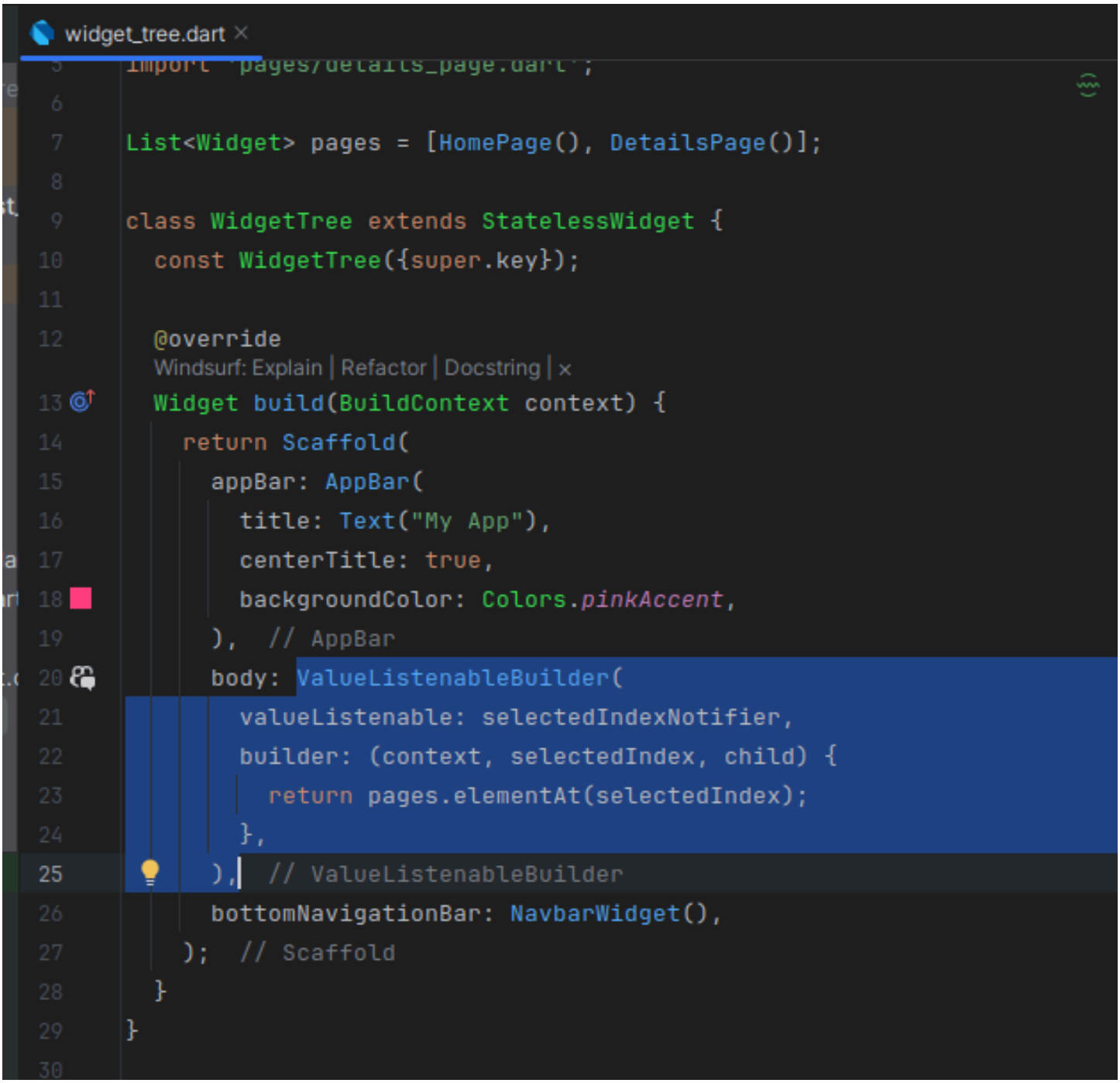


maintenant parce que nous avons le ValueListenableBuilder dans la navigation Nous n'avons pas besoin du setState, _selectedIndex

=> donc on va les supprimer



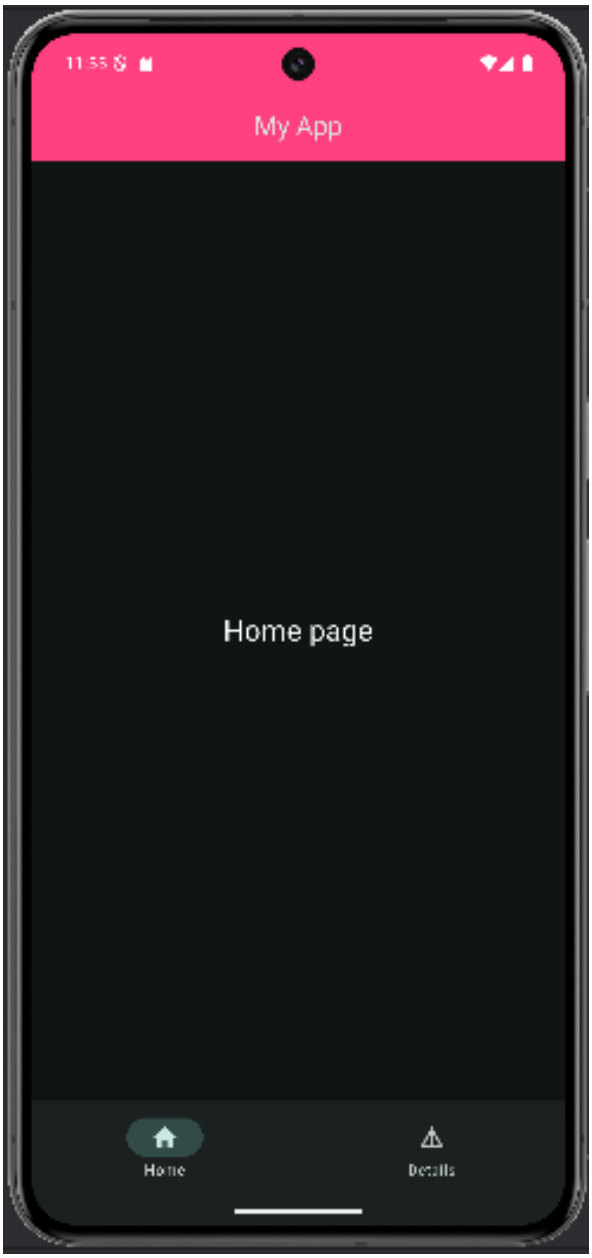
=> maintenant nous pouvons accéder **selectedIndexNotifier** **de n'importe où dans l'application** juste en utilisant ce **ValueListenableBuilder ()**

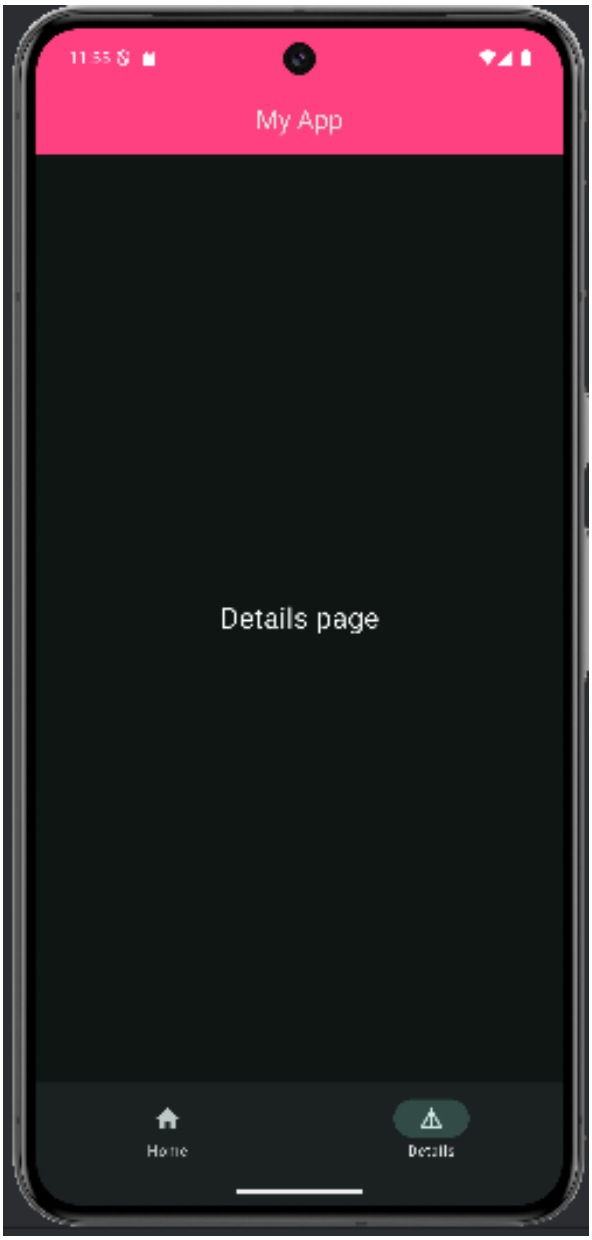


maintenant si nous cliquons sur :

details nous avons le detail

home vous avons home

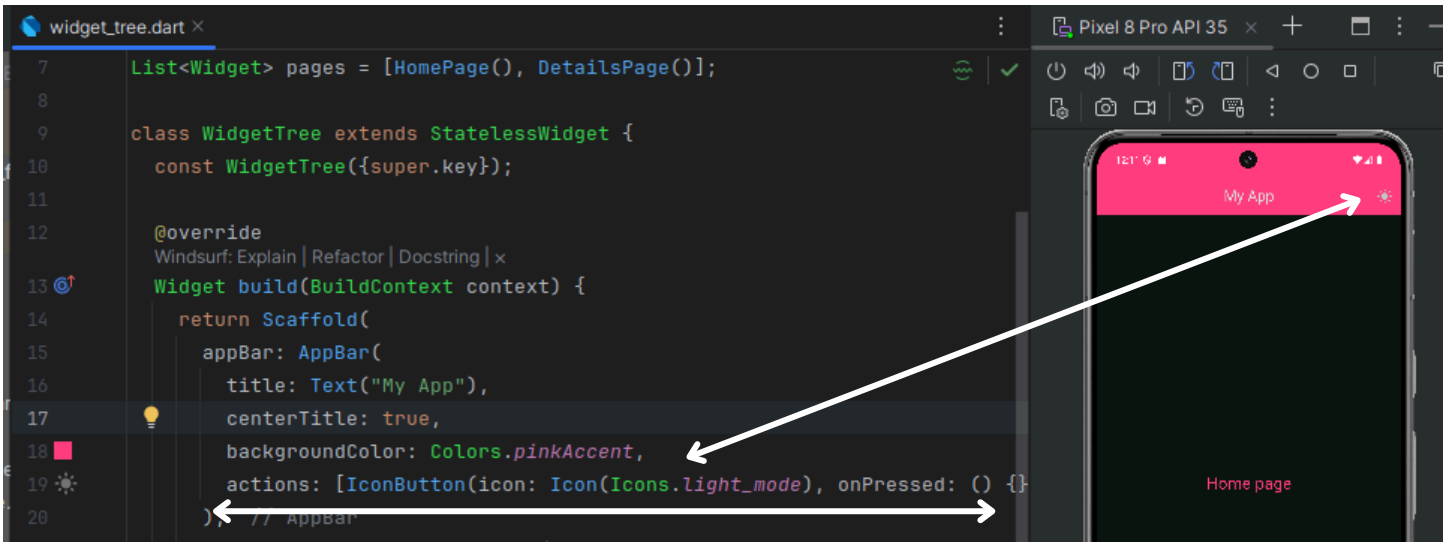


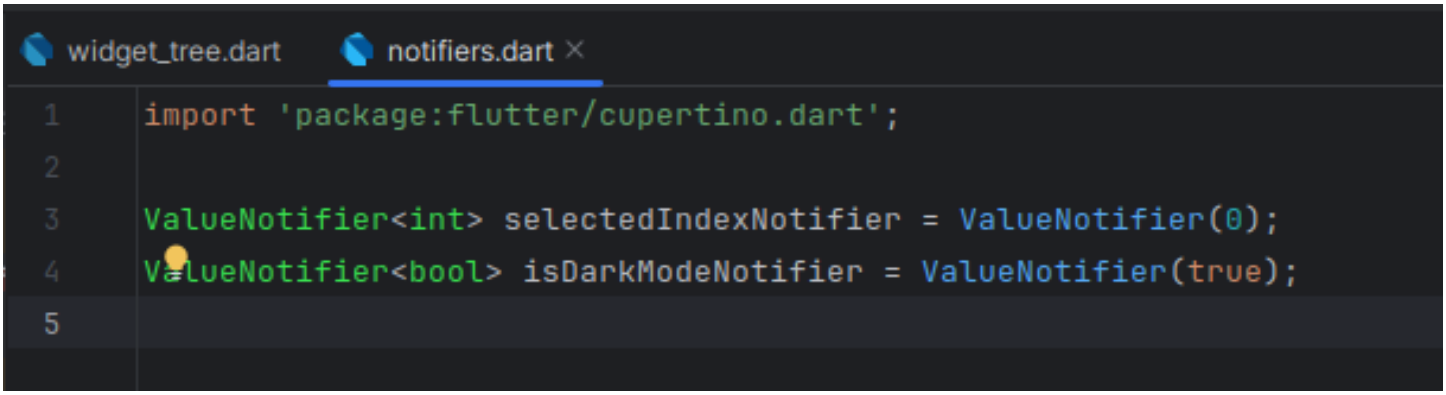


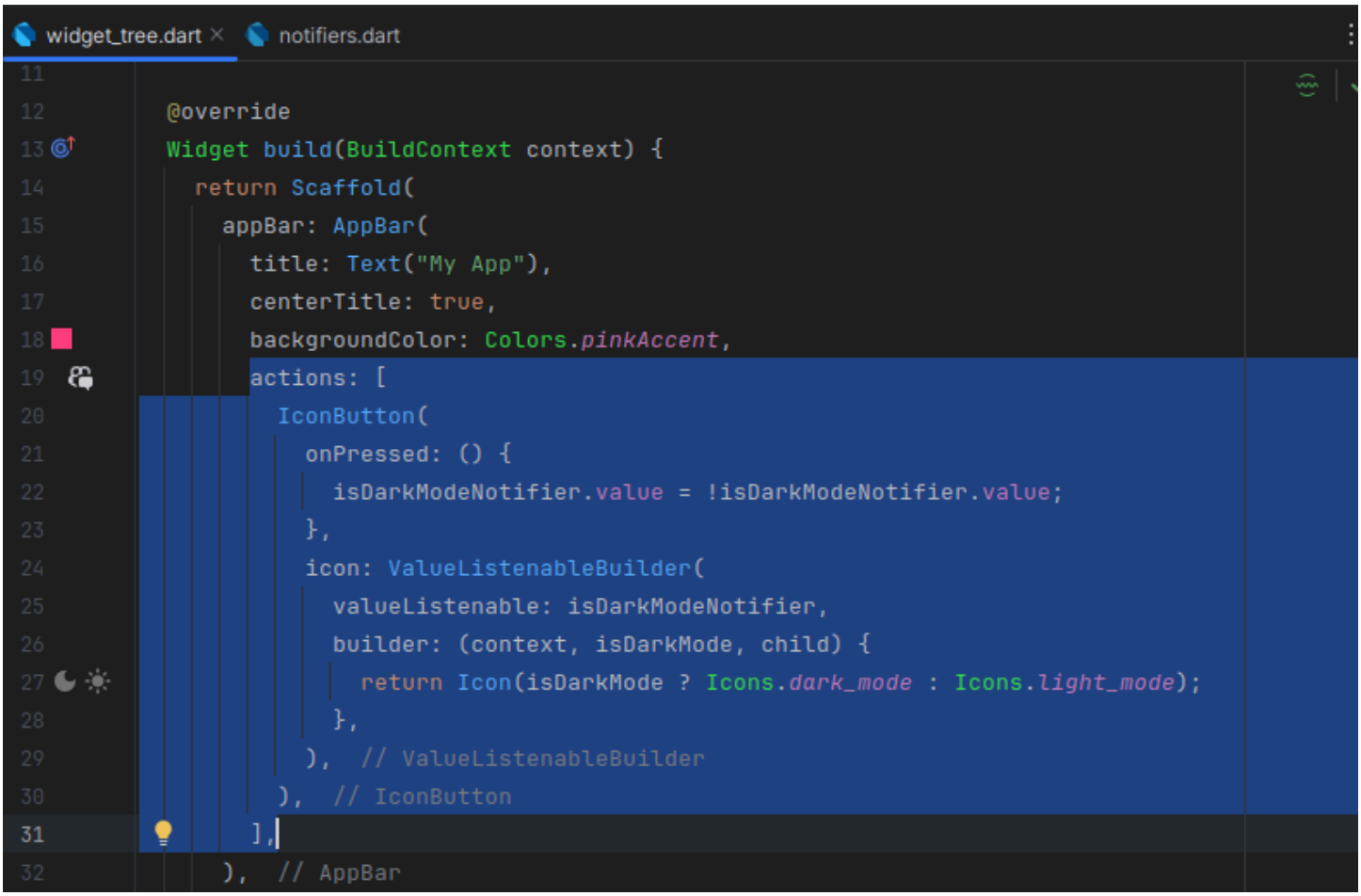
- ## User le Notifier pour le mode sombre ou clair

donc on vas :

- supprimer **: brightness: Brightness.dark** dans **MaterialApp**
- **ajouter un icon dans l'appBar**
- **créer un nouveau notifier**

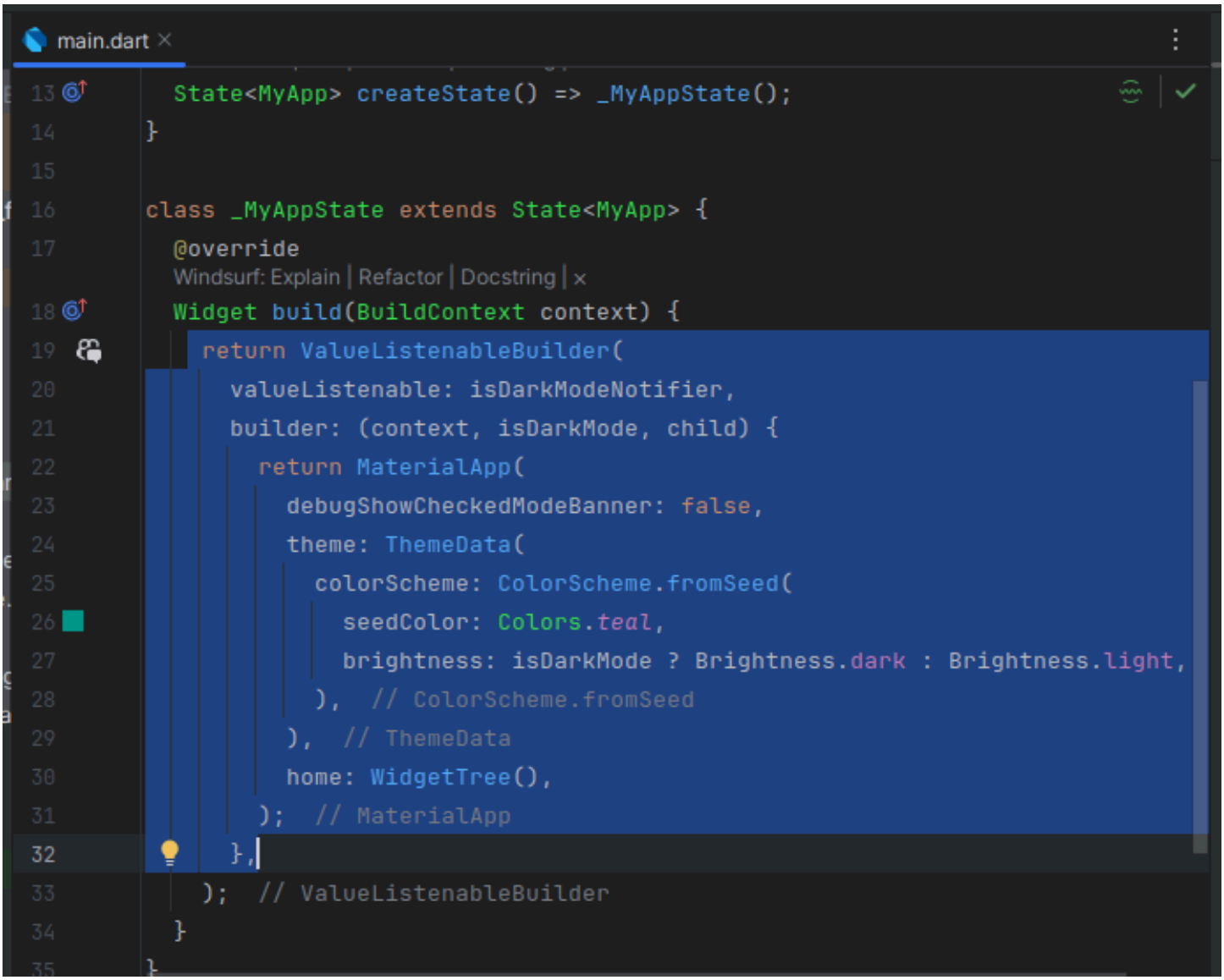

widget_tree.dart
List<Widget> pages = [HomePage(), DetailsPage()];
class WidgetTree extends StatelessWidget {
const WidgetTree({super.key});
@override
Windsurf: Explain | Refactor | Docstring | ×
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text("My App"),
centerTitle: true,
backgroundColor: Colors.pinkAccent,
actions: [IconButton(icon: Icon(Icons.light_mode), onPressed: () {}
Pixel 8 Pro API 35
My App
Home page

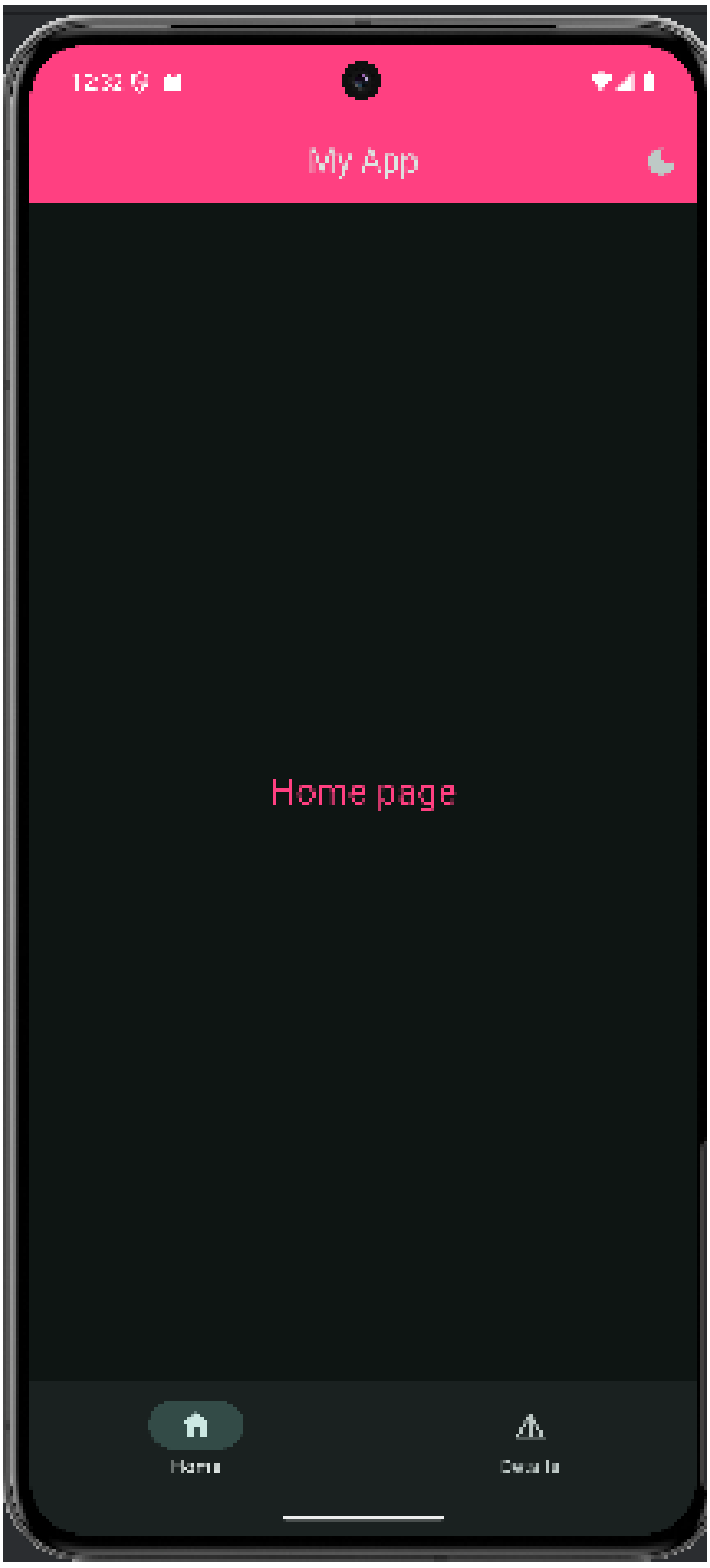

widget_tree.dart
notifiers.dart
import 'package:flutter/cupertino.dart';
ValueNotifier<int> selectedIndexNotifier = ValueNotifier(0);
ValueNotifier<bool> isDarkModeNotifier = ValueNotifier(true);

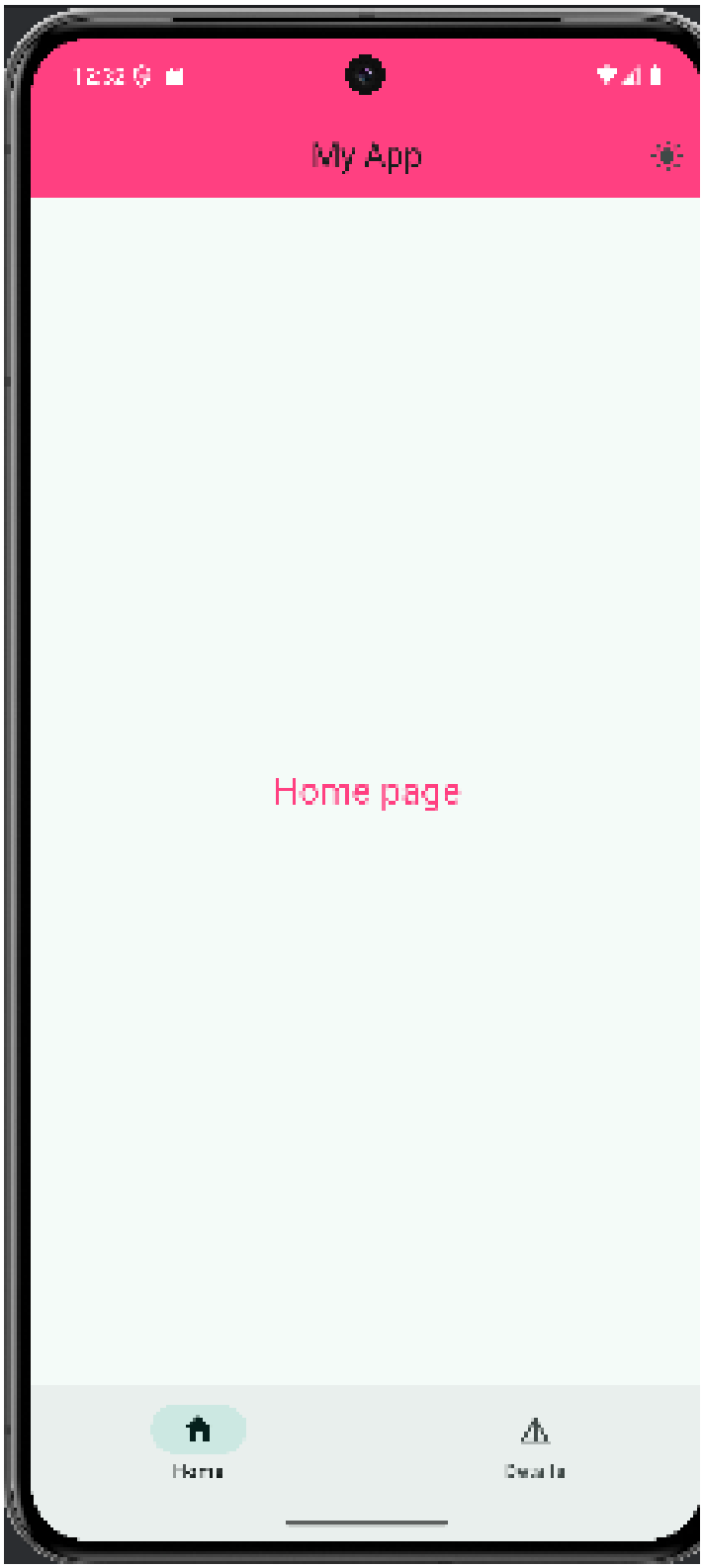

widget_tree.dart
notifiers.dart
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text("My App"),
centerTitle: true,
backgroundColor: Colors.pinkAccent,
actions: [
IconButton(
onPressed: () {
isDarkModeNotifier.value = !isDarkModeNotifier.value;
},
icon: ValueListenableBuilder(
valueListenable: isDarkModeNotifier,
builder: (context, isDarkMode, child) {
return Icon(isDarkMode ? Icons.dark_mode : Icons.light_mode);
},
), // ValueListenableBuilder
), // IconButton
],
), // AppBar

=> maintenant quand on clique sur les icônes, ils changent

on vas faire le même chose pour le brightness :




My App
Home page
Home
Details


My App
Home page
Home
Details