

## TP 4

### • null safety

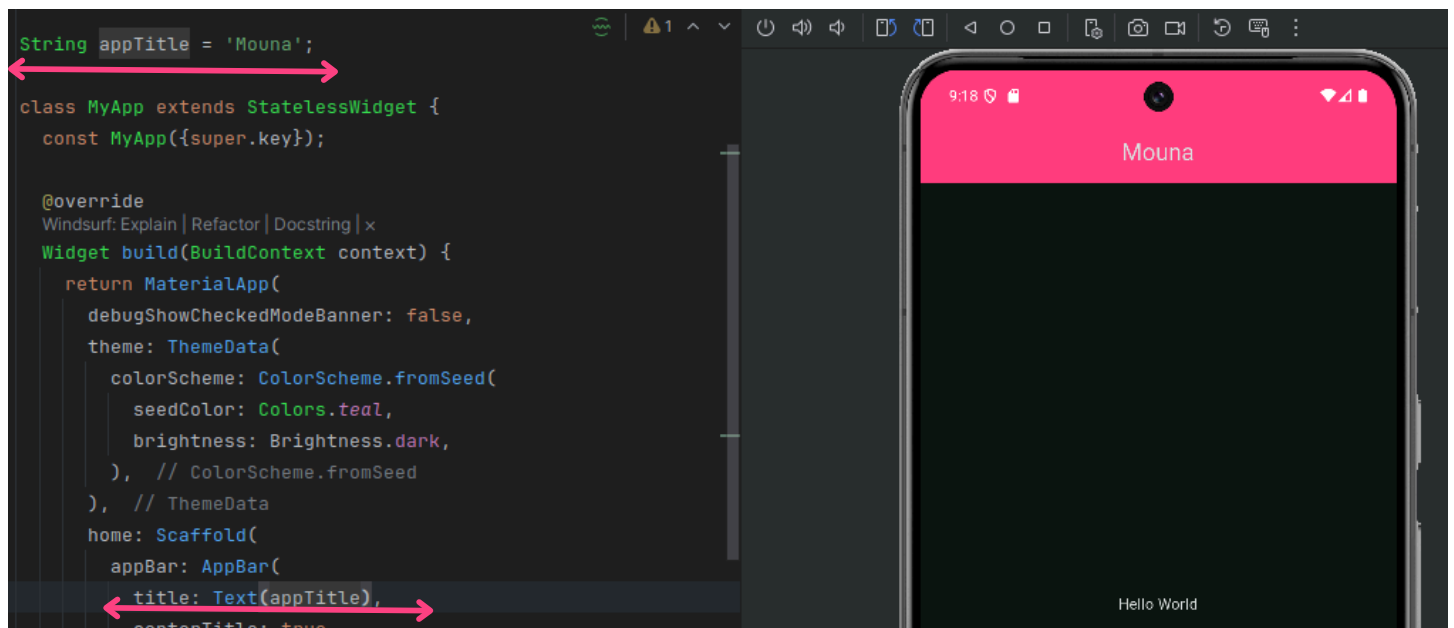
Dart's **null safety** rend les types **non nullables par défaut**. Une variable de type String ne peut pas être null. Si une valeur peut être absente, vous devez le dire avec **?** => **String?**

#### Non-nullable vs nullable

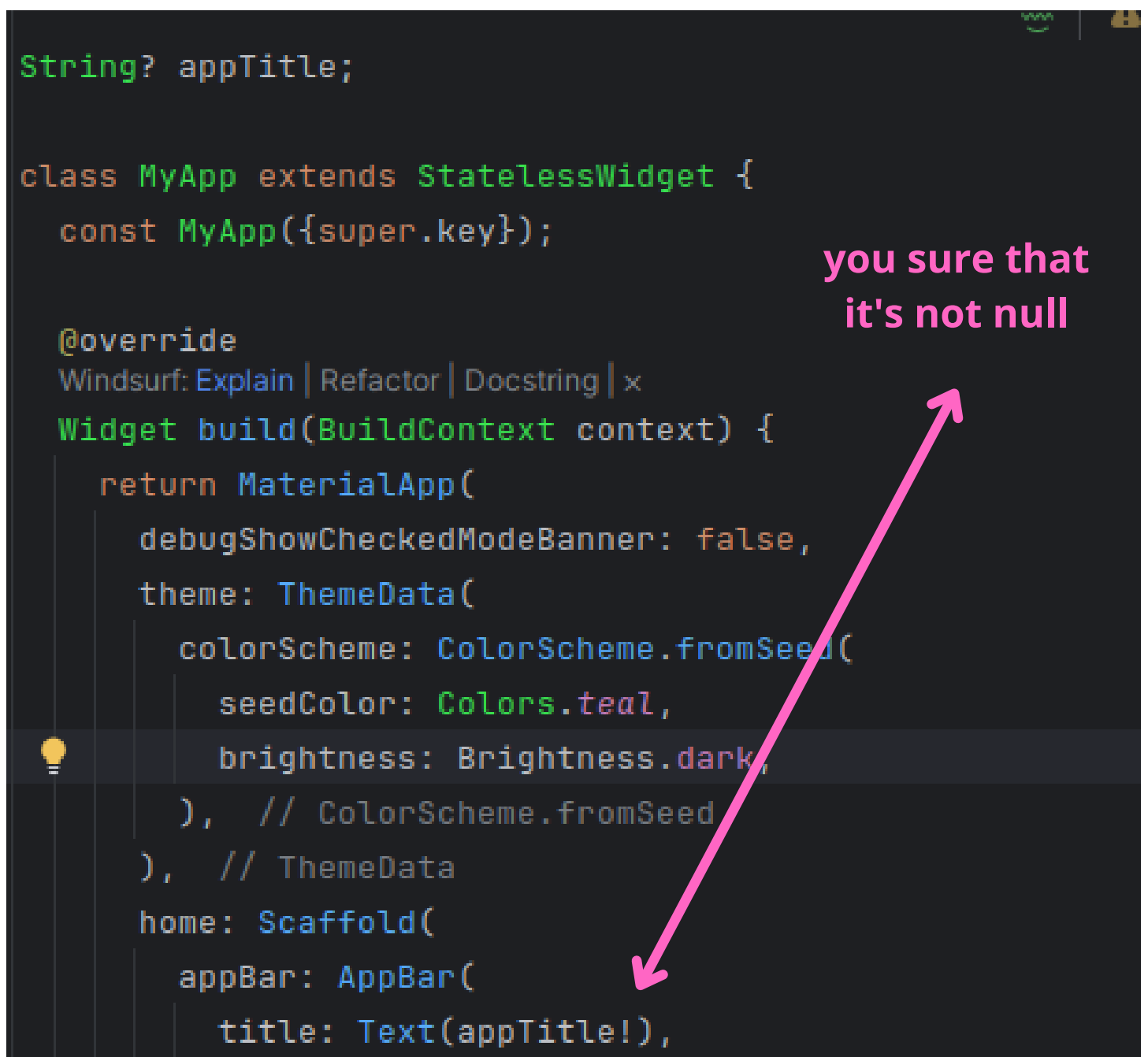
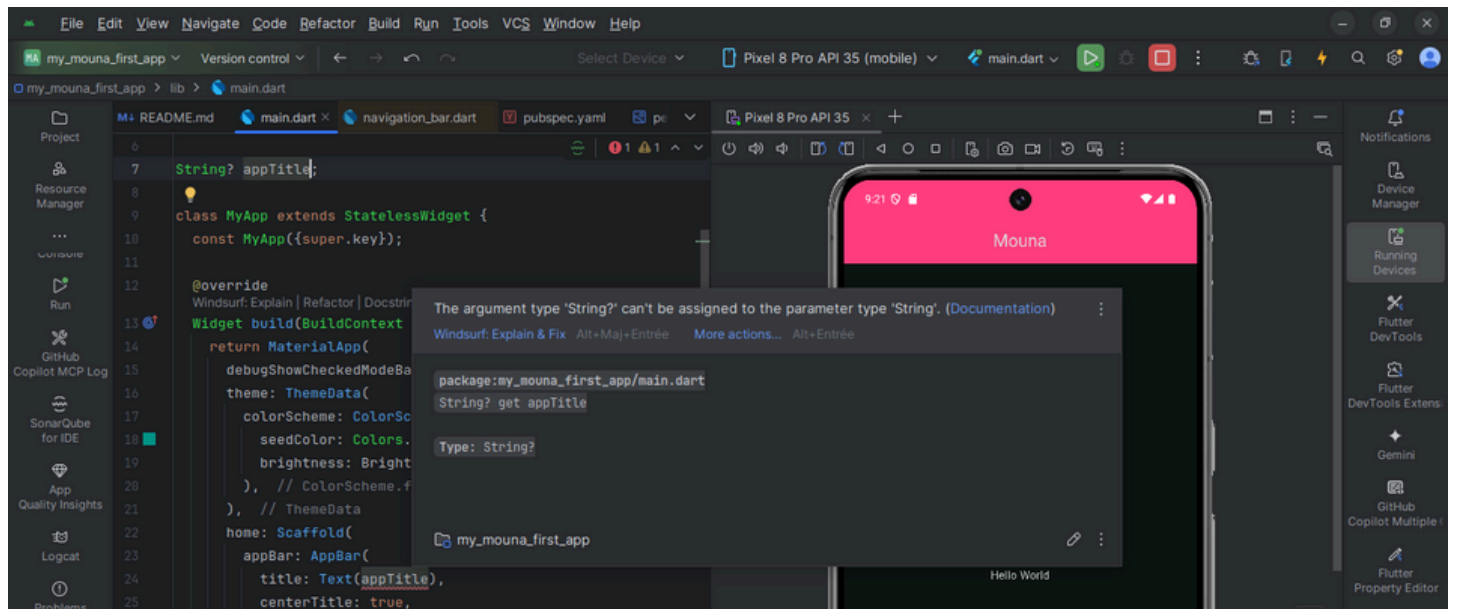
`String name = 'Rita';` => **must be non-null**

`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 première classe je vais en créer une autre : **clic stl+ tab**
- changer le nom avec: **Home**
- mettre le Scaffold de 1ère classe dans le return 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  
  }  
}
```

```

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

```

class _HomeState extends State<Home> {
  int _selectedIndex = 0;

  @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) {
          setState(() {
            _selectedIndex = value;
          });
        },
        selectedIndex: _selectedIndex,
      ), // NavigationBar
    ); // Scaffold
  }
}

```

`int _selectedIndex = 0;` => **pour savoir** quelle destination 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.

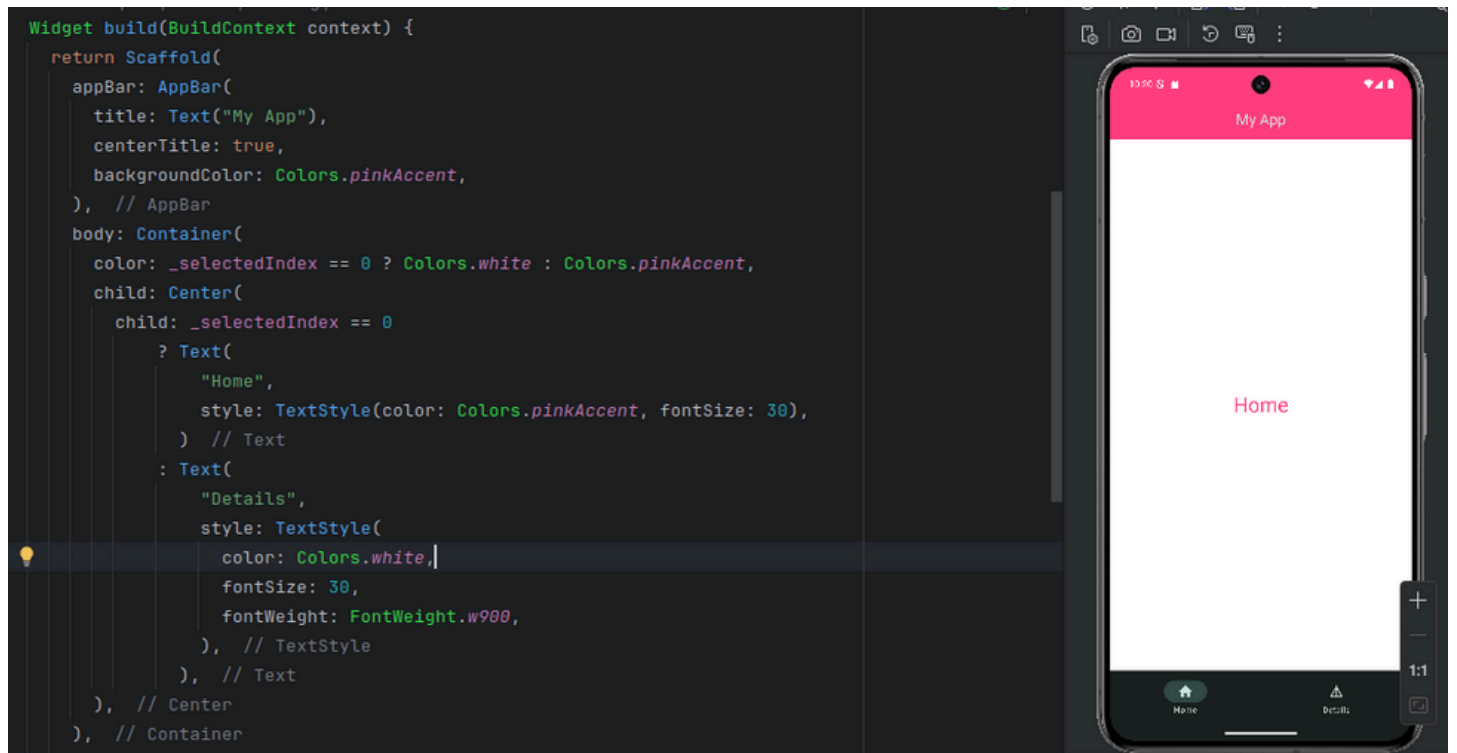
`_selectedIndex = value;` => **Enregistre** l'index sélectionné dans la variable d'état.

`});`

`},`

`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

1 import 'package:flutter/material.dart';  
2  
3 class NavbarWidget extends StatefulWidget {  
4 const NavbarWidget({super.key});  
5  
6 @override  
7 State<NavbarWidget> createState() => \_NavbarWidgetState();  
8 }  
9  
10 class \_NavbarWidgetState extends State<NavbarWidget> {  
11 int \_selectedIndex = 0;  
12  
13 @override  
14 Widget build(BuildContext context) {  
15 return NavigationBar(  
16 destinations: [  
17 NavigationDestination(icon: Icon(Icons.home), label: 'Home'),  
18 NavigationDestination(icon: Icon(Icons.details), label: 'Details'),  
19 ],  
20 onDestinationSelected: (int value) {  
21 setState(() {  
22 \_selectedIndex = value;  
23 });  
24 },  
25 selectedIndex: \_selectedIndex,  
26 ); // NavigationBar  
27 }  
28 }

```
M README.md main.dart x navbar_widget.dart navigation_bar.dart pub
31
32 class _HomeState extends State<Home> {
33   @override
34   Widget build(BuildContext context) {
35     return Scaffold(
36       appBar: AppBar(
37         title: Text("My App"),
38         centerTitle: true,
39         backgroundColor: Colors.pinkAccent,
40       ), // AppBar
41       body: Container(
42         color: Colors.white,
43         child: Center(
44           child: Text(
45             "Home",
46             style: TextStyle(color: Colors.pinkAccent, fontSize: 30),
47           ), // Text
48         ), // Center
49       ), // Container
50       bottomNavigationBar: NavBarWidget(),
51     ), // Scaffold
52   }
53 }
```

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

Not applicable for the main.dart configuration

widget\_tree.dart ×

```
1 import 'package:flutter/material.dart';
2 import '../widgets/navbar_widget.dart';
3
4 class WidgetTree extends StatelessWidget {
5   const WidgetTree({super.key});
6
7   @override
8   Widget build(BuildContext context) {
9     return Scaffold(
10       appBar: AppBar(
11         title: Text("My App"),
12         centerTitle: true,
13         backgroundColor: Colors.pinkAccent,
14       ), // AppBar
15       bottomNavigationBar: NavBarWidget(),
16     ); // Scaffold
17   }
18 }
19
```

home\_page.dart ×

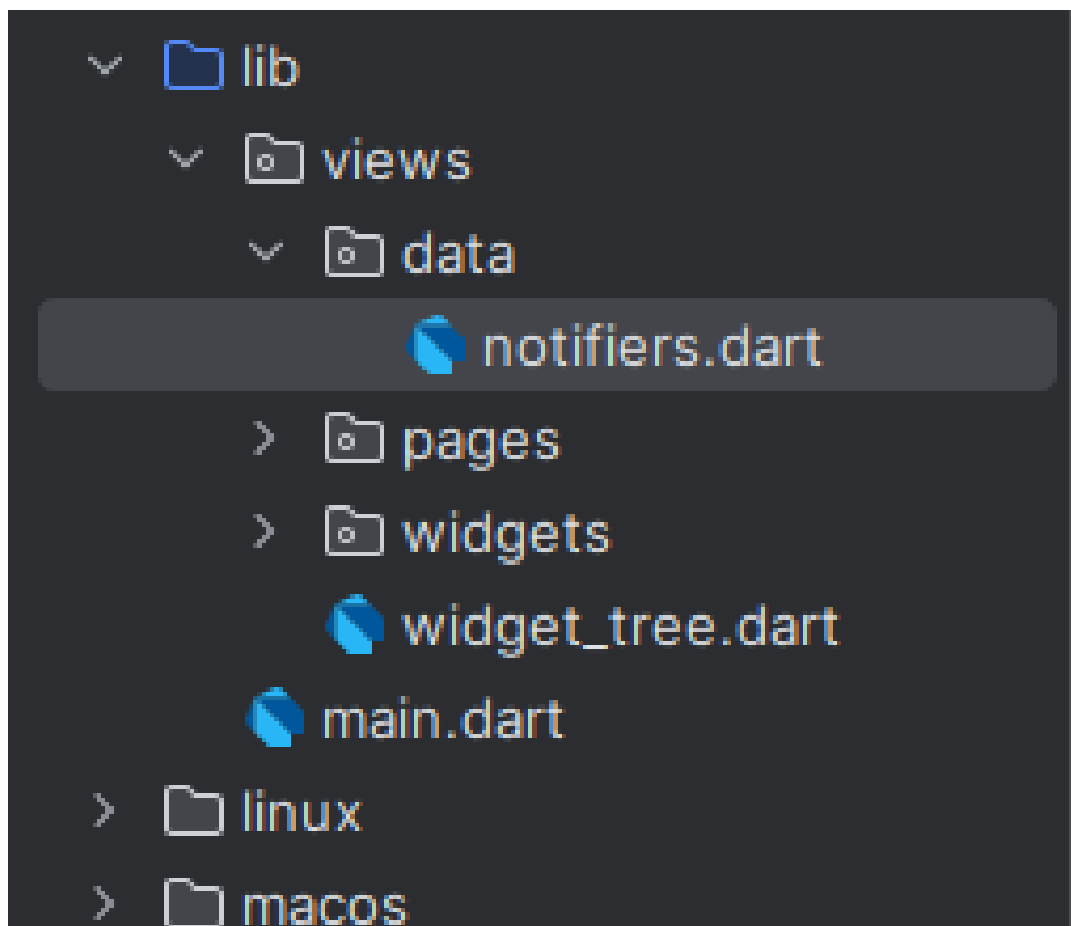
```
1 import 'package:flutter/material.dart';
2
3 class HomePage extends StatelessWidget {
4   const HomePage({super.key});
5
6   @override
7   Widget build(BuildContext context) {
8     return Center(child: Text("Home page"));
9   }
10 }
11
```

```
details_page.dart x
1  import 'package:flutter/material.dart';
2
3  class DetailsPage extends StatelessWidget {
4    const DetailsPage({super.key});
5
6    @override
7    Widget build(BuildContext context) {
8      return Center(child: Text("Details page"));
9    }
10  }
11  |
```

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

```
widget_tree.dart x
1 import 'package:flutter/material.dart';
2 import '../widgets/navbar_widget.dart';
3 import 'pages/home_page.dart';
4 import 'pages/details_page.dart';
5
6 List<Widget> pages = [HomePage(), DetailsPage()];
7
8 class WidgetTree extends StatelessWidget {
9   const WidgetTree({super.key});
10
11   @override
12   Widget build(BuildContext context) {
13     return Scaffold(
14       appBar: AppBar(
15         title: Text("My App"),
16         centerTitle: true,
17         backgroundColor: Colors.pinkAccent,
18       ), // AppBar
19       body: pages.elementAt(0),
20       bottomNavigationBar: NavBarWidget(),
21     ); // Scaffold
22   }
23 }
24
```

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 doit 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 x
1  import 'package:flutter/cupertino.dart';
2
3  ValueNotifier<int> selectedIndexNotifier = ValueNotifier(0);
4
```

```
navbar_widget.dart x
3
4 class NavBarWidget extends StatefulWidget {
5   const NavBarWidget({super.key});
6   @override
7   State<NavBarWidget> createState() => _NavBarWidgetState();
8 }
9
10 class _NavBarWidgetState extends State<NavBarWidget> {
11   int _selectedIndex = 0;
12
13   @override
14   Widget build(BuildContext context) {
15     return ValueListenableBuilder(
16       valueListenable: selectedIndexNotifier,
17       builder: (context, value, child) {
18         return NavigationBar(
19           destinations: [
20             NavigationDestination(icon: Icon(Icons.home), label: 'Home'),
21             NavigationDestination(icon: Icon(Icons.details), label: 'Details'),
22           ],
23           onDestinationSelected: (int value) {
24             setState(() {
25               _selectedIndex = value;
26             });
27           },
28         );
29       },
30     );
31   }
32 }
```

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

=> donc on va les supprimer

```
navbar_widget.dart x
1 import 'package:flutter/material.dart';
2 import 'package:flutter/widgets.dart';
3
4 class NavBarWidget extends StatelessWidget {
5     const NavBarWidget({super.key});
6     @override
7     Widget build(BuildContext context) {
8         return ValueListenableBuilder(
9             valueListenable: selectedIndexNotifier,
10             builder: (context, selectedPage, child) {
11                 return NavigationBar(
12                     destinations: [
13                         NavigationDestination(icon: Icon(Icons.home), label: 'Home'),
14                         NavigationDestination(icon: Icon(Icons.details), label: 'Details'),
15                     ],
16                     onDestinationSelected: (int value) {
17                         selectedIndexNotifier.value = value;
18                     },
19                     selectedIndex: selectedPage,
20                 ); // NavigationBar
21             },
22         ); // ValueListenableBuilder
23     }
24 }
25
```

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

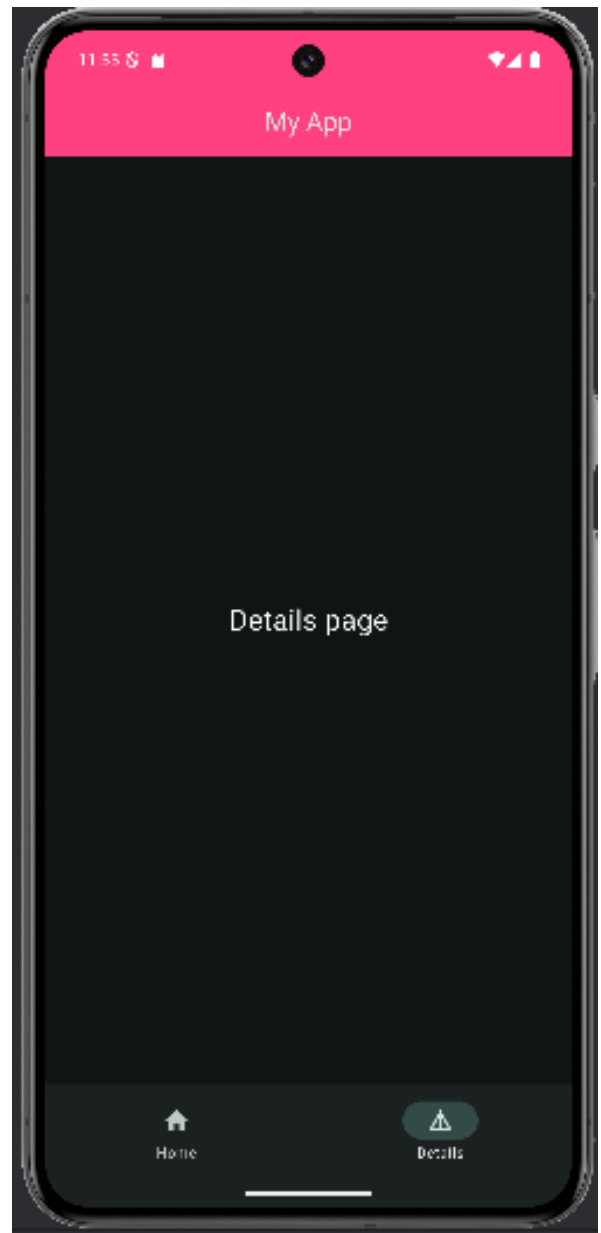
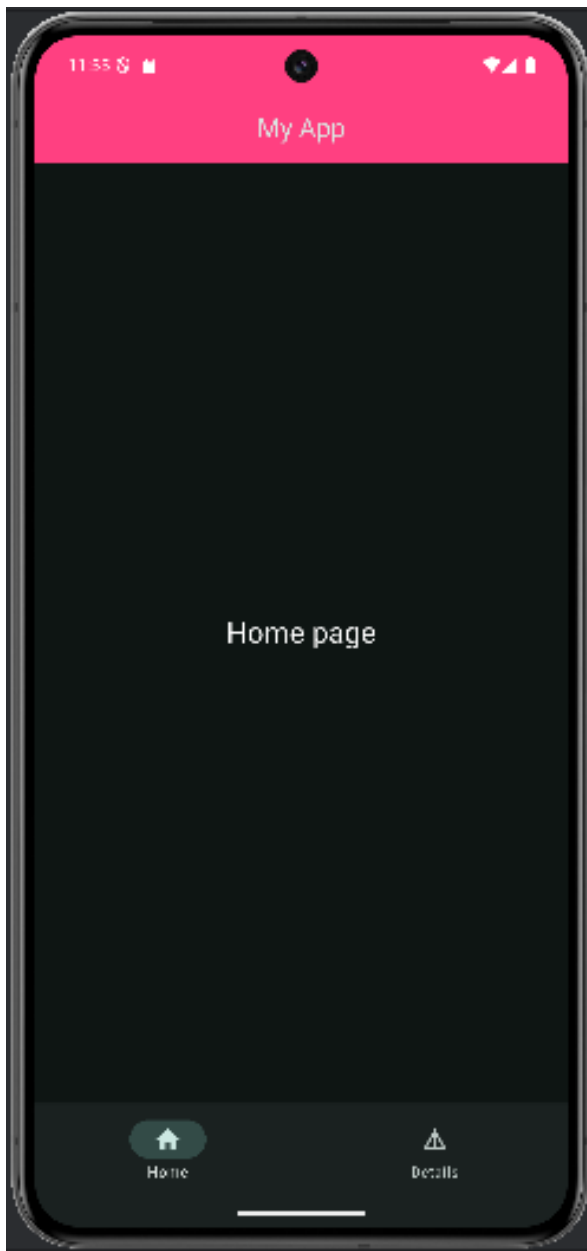
```
widget_tree.dart x
5 import 'pages/details_page.dart';
6
7 List<Widget> pages = [HomePage(), DetailsPage()];
8
9 class WidgetTree extends StatelessWidget {
10   const WidgetTree({super.key});
11
12   @override
13   Widget build(BuildContext context) {
14     return Scaffold(
15       appBar: AppBar(
16         title: Text("My App"),
17         centerTitle: true,
18         backgroundColor: Colors.pinkAccent,
19       ), // AppBar
20       body: ValueListenableBuilder(
21         valueListenable: selectedIndexNotifier,
22         builder: (context, selectedIndex, child) {
23           return pages.elementAt(selectedIndex);
24         },
25       ), // ValueListenableBuilder
26       bottomNavigationBar: NavbarWidget(),
27     ); // Scaffold
28   }
29 }
30
```

maintenant si nous cliquons sur :

**details** nous avons le detail

**home** vous avons home

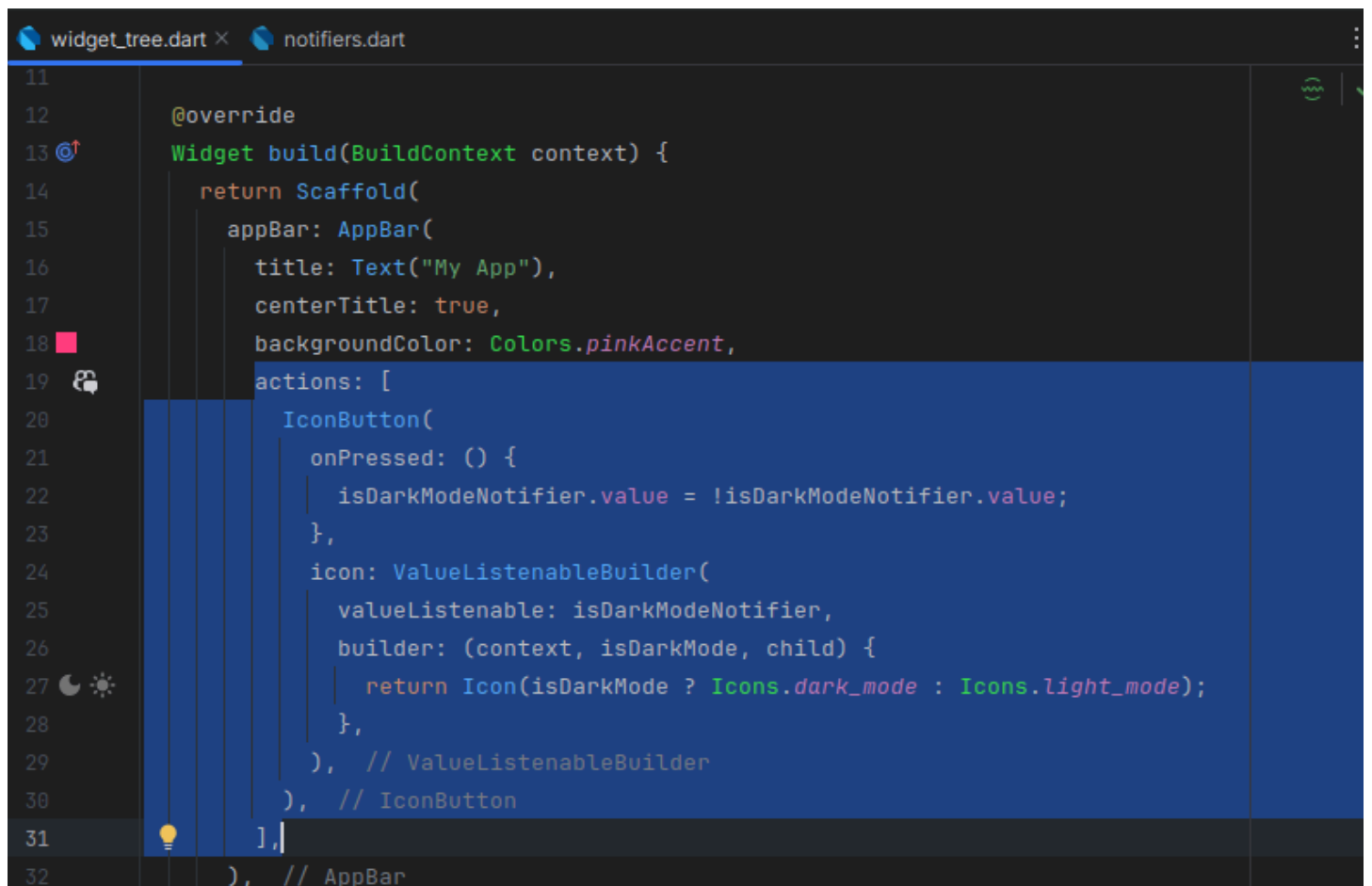
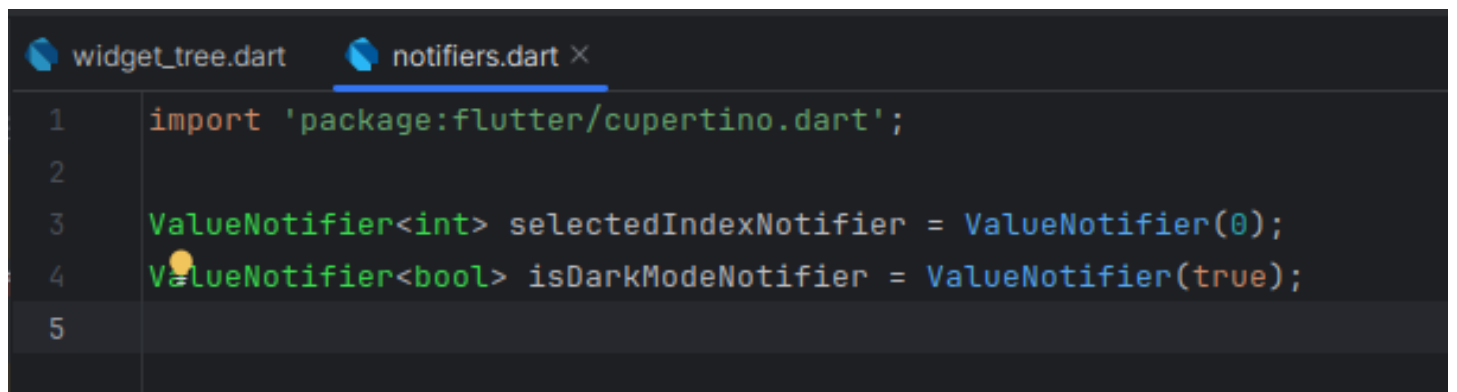




- 
- User le Notifier pour le mode sombre ou clair

donc on vas :

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



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

on vas faire le même chose pour le brightness :

```
main.dart x
13 State<MyApp> createState() => _MyAppState();
14 }
15
16 class _MyAppState extends State<MyApp> {
17   @override
18   Windsurf: Explain | Refactor | Docstring | x
19   Widget build(BuildContext context) {
20     return ValueListenableBuilder(
21       valueListenable: isDarkModeNotifier,
22       builder: (context, isDarkMode, child) {
23         return MaterialApp(
24           debugShowCheckedModeBanner: false,
25           theme: ThemeData(
26             colorScheme: ColorScheme.fromSeed(
27               seedColor: Colors.teal,
28               brightness: isDarkMode ? Brightness.dark : Brightness.light,
29             ), // ColorScheme.fromSeed
30           ), // ThemeData
31           home: WidgetTree(),
32         ); // MaterialApp
33       },
34     ); // ValueListenableBuilder
35 }
```

