

TP 4

• null safety

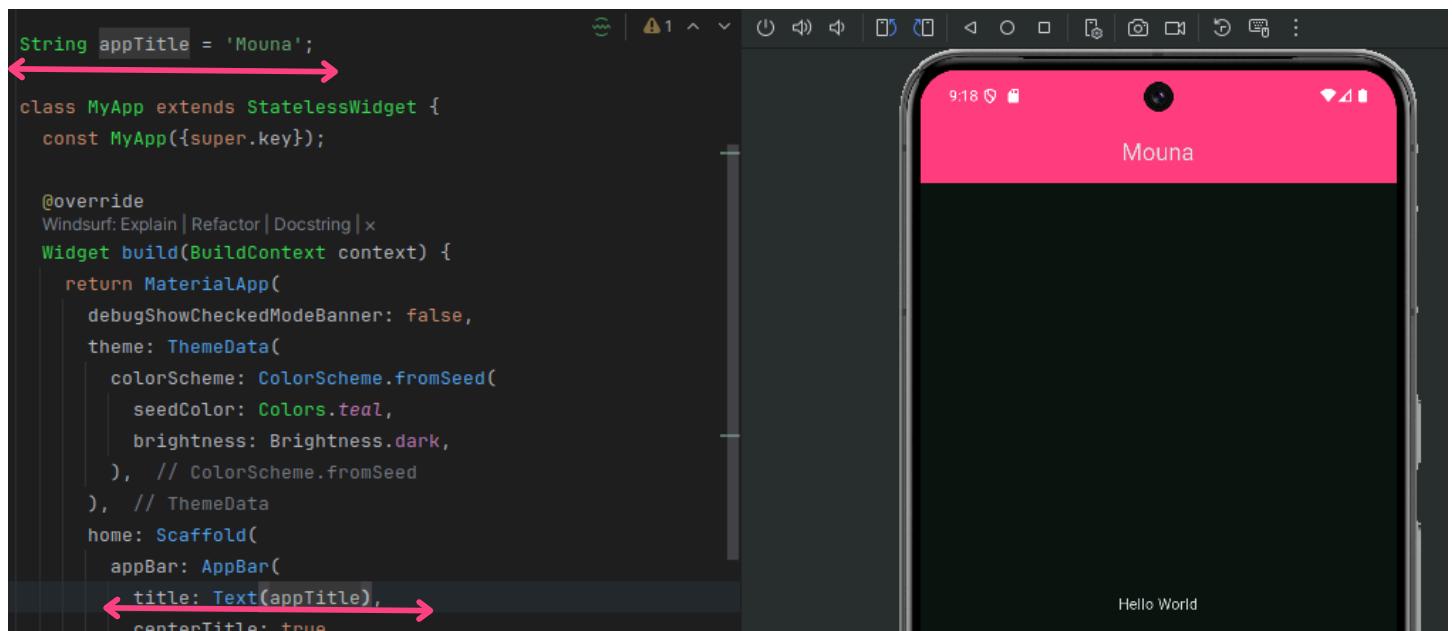
Dart's **null safety** rend les types **non nullables par défaut**. Une variable de type String ne peut pas être null. Si une valeur peut être absente, vous devez le dire avec **? => String?**

Non-nullable vs nullable

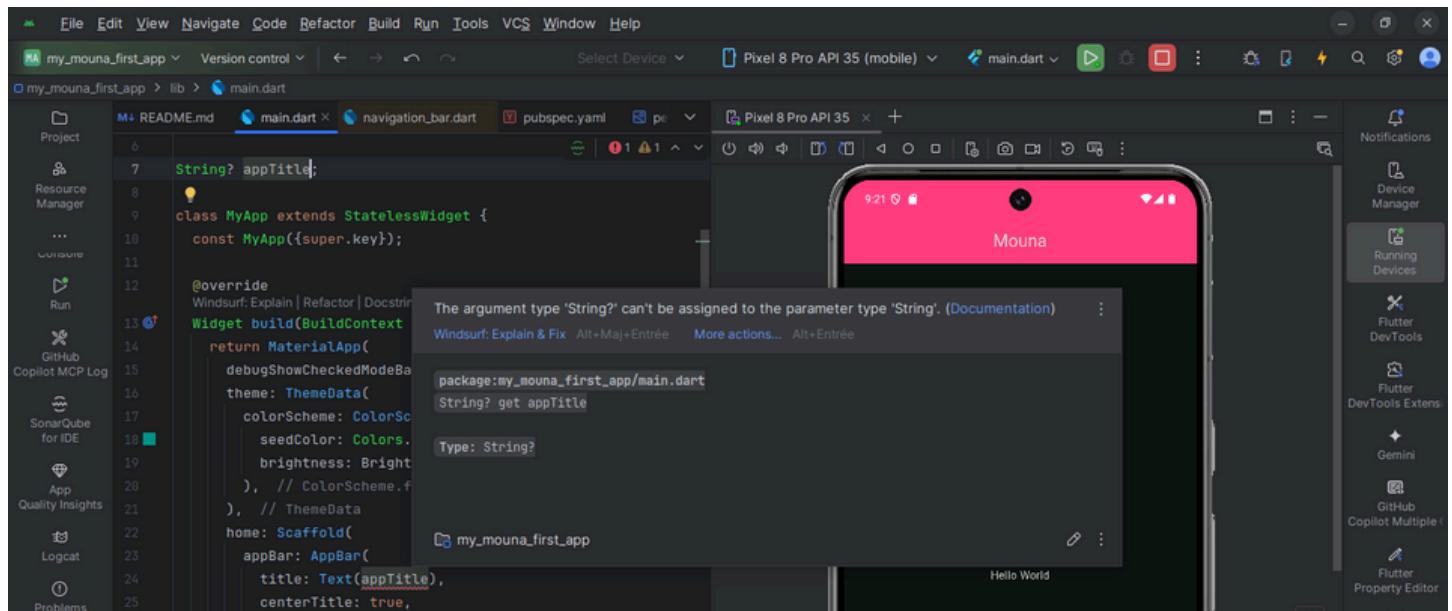
String name = 'Rita'; => must be non-null

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**



```
String? appTitle;
```

```
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: Scaffold(  
        appBar: AppBar(  
          title: Text(appTitle!),
```

you sure that
it's not 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 un 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
    Windsurf: Explain | Refactor | Docstring | x
    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:

The screenshot shows a code editor on the left with Dart code for a Flutter application. The code defines a scaffold with an appBar containing a title 'My App'. The body contains a container with a child that is a center. Inside the center, there is a child that is a text. The text has a condition: if _selectedIndex == 0, it displays 'Home' in white color, 30px font size, and bold weight. If not, it displays 'Details' in white color, 30px font size, and bold weight. On the right, a mobile device screen displays the app with the title 'My App'. Below the title, the word 'Home' is visible in black text. At the bottom of the screen, there are two tabs: 'Home' and 'Details', with 'Home' being the active tab.

```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("My App"),
      centerTitle: true,
      backgroundColor: Colors.pinkAccent,
    ), // AppBar
    body: Container(
      color: _selectedIndex == 0 ? Colors.white : Colors.pinkAccent,
      child: Center(
        child: _selectedIndex == 0
          ? Text(
              "Home",
              style: TextStyle(color: Colors.pinkAccent, fontSize: 30),
            ) // Text
          : Text(
              "Details",
              style: TextStyle(
                color: Colors.white,
                fontSize: 30,
                fontWeight: FontWeight.w900,
              ), // TextStyle
        ), // Text
      ), // Center
    ), // Container
  );
}
```

• 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

The screenshot shows a Flutter project named "my_mouna_first_app" in an IDE. The project structure on the left includes folders for android, assets, build, ios, lib (with widgets), test, web, windows, .gitignore, .metadata, analysis_options.yaml, my_mouna_first_app.iml, pubspec.lock, pubspec.yaml, README.md, External Libraries, and Scratches and Consoles. The code editor on the right displays the file "navbar_widget.dart". The code defines a "NavbarWidget" class that extends "StatefulWidget". It contains a constructor "const NavbarWidget({super.key});", an overridden "createState" method returning a "_NavbarWidgetState" object, and a state class "._NavbarWidgetState" that extends "State<NavbarWidget>". The state class initializes "_selectedIndex = 0", overrides the "build" method to return a "NavigationBar" widget with destinations for 'Home' and 'Details', and handles the "onDestinationSelected" callback by updating the state's selected index. A pink bracket on the left side groups the "lib/widgets/navbar_widget.dart" and "main.dart" files.

```
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;

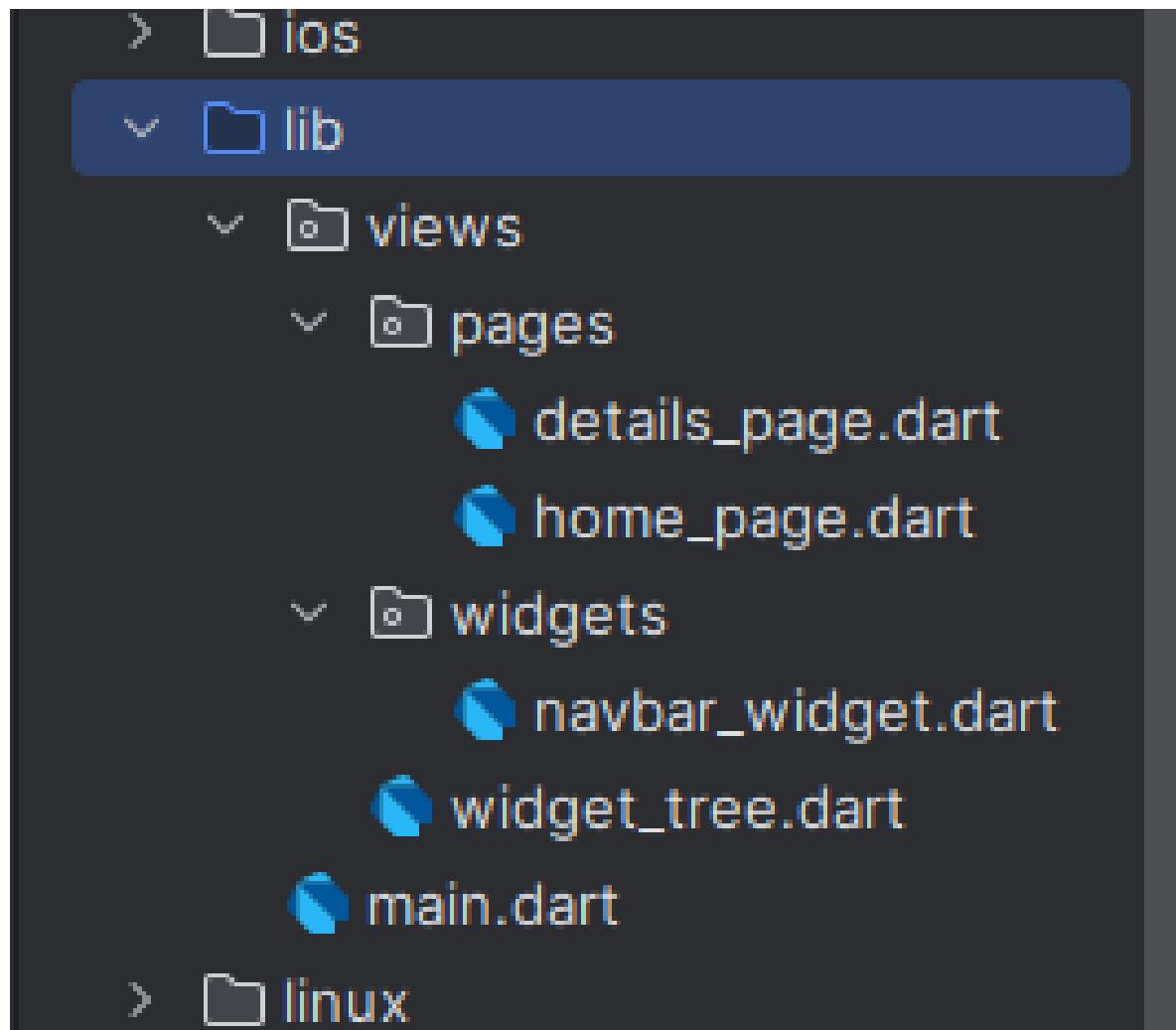
    @override
    Widget build(BuildContext context) {
        return NavigationBar(
            destinations: [
                NavigationDestination(icon: Icon(Icons.home), label: 'Home'),
                NavigationDestination(icon: Icon(Icons.details), label: 'Details')
            ],
            onDestinationSelected: (int value) {
                setState(() {
                    _selectedIndex = value;
                });
            },
            selectedIndex: _selectedIndex,
        ); // NavigationBar
    }
}
```

```
31
32     class _HomeState extends State<Home> {
33
34     @override
35     Widget build(BuildContext context) {
36
37         return Scaffold(
38             appBar: AppBar(
39                 title: Text("My App"),
40                 centerTitle: true,
41                 backgroundColor: Colors.pinkAccent,
42             ), // AppBar
43             body: Container(
44                 color: Colors.white,
45                 child: Center(
46                     child: Text(
47                         "Home",
48                         style: TextStyle(color: Colors.pinkAccent, fontSize: 30),
49                     ), // Text
50                     ), // Center
51                     ), // Container
52                     bottomNavigationBar: NavbarWidget(),
53     ); // Scaffold
54 }
```

maintenant si nous voulons avoir à l'intérieur de **scaffold** un corps différent 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**



widget_tree.dart × Not applicable for the main.dart configuration

```
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 ×

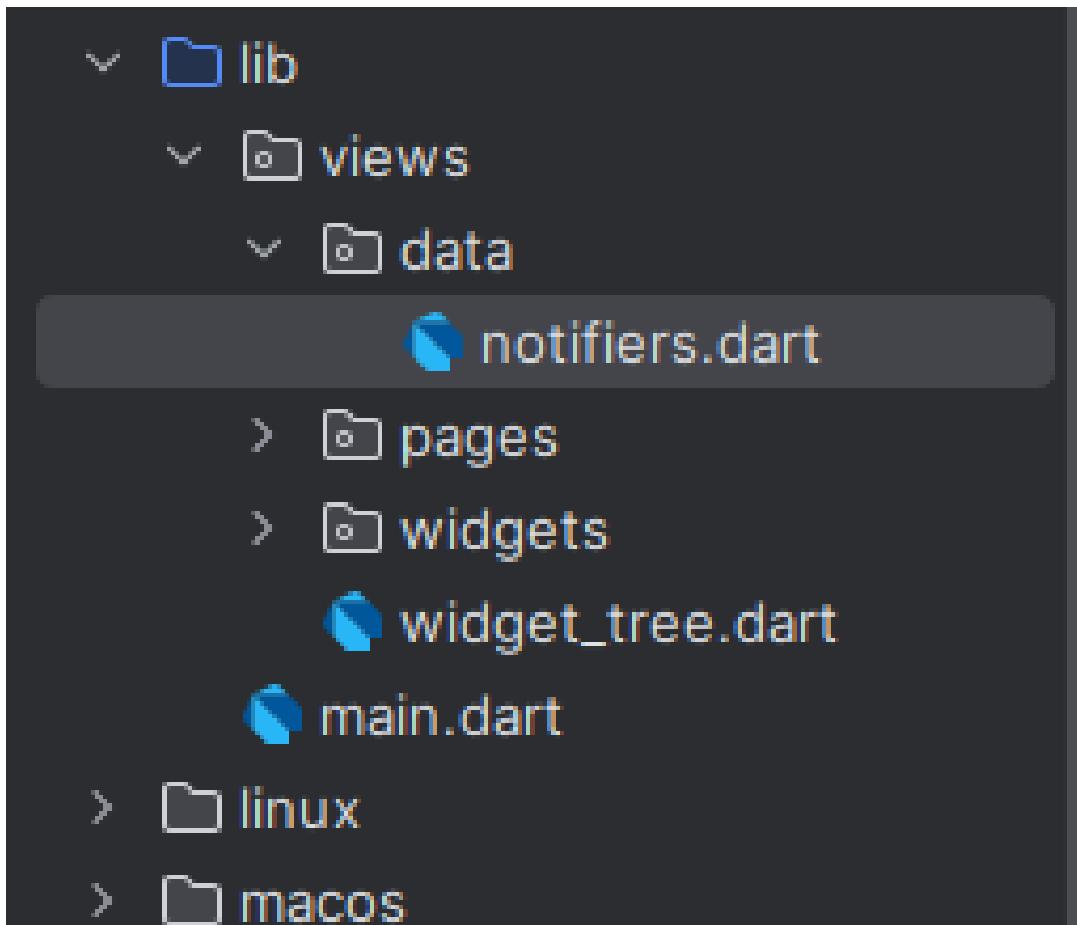
```
1 import 'package:flutter/material.dart';
2 
3 class HomePage extends StatelessWidget {
4     const HomePage({super.key});
5 
6     @override
7     Widget build(BuildContext context) {
8         return Center(child: Text("Home page"));
9     }
10 }💡
11 
```

```
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 ×
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), // Yellow dot icon here
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 ×
1 import 'package:flutter/cupertino.dart';
2
3 ValueNotifier<int> selectedIndexNotifier = ValueNotifier(0);
4
```

The image shows a code editor window with the title 'notifiers.dart'. The code is as follows:

```
import 'package:flutter/cupertino.dart';
ValueNotifier<int> selectedIndexNotifier = ValueNotifier(0);
```

```
(navbar_widget.dart) : X
```

```
3
4     class NavbarWidget extends StatefulWidget {
5         const NavbarWidget({super.key});
6         @override
7         Windsurf: Explain | Refactor | Docstring | x
8         State<NavbarWidget> createState() => _NavbarWidgetState();
9
10    class _NavbarWidgetState extends State<NavbarWidget> {
11        int _selectedIndex = 0;
12
13        @override
14        Windsurf: Explain | Refactor | Docstring | x
15        Widget build(BuildContext context) {
16            return ValueListenableBuilder(
17                valueListenable: selectedIndexNotifier,
18                builder: (context, value, child) {
19                    return NavigationBar(
20                        destinations: [
21                            NavigationDestination(icon: Icon(Icons.home), label: 'Home'),
22                            NavigationDestination(icon: Icon(Icons.details), label: 'Details'),
23                        ],
24                        onDestinationSelected: (int value) {
25                            setState(() {
26                                _selectedIndex = value;
27                            });
28                    },
29                ),
30            );
31        }
32    }
33}
```

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

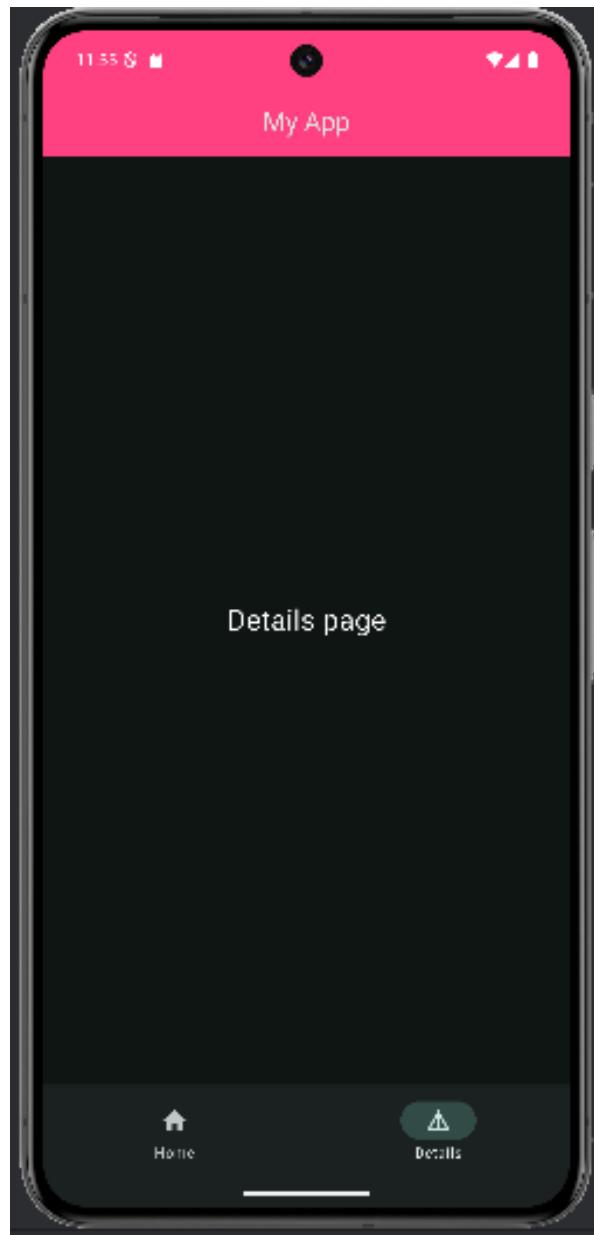
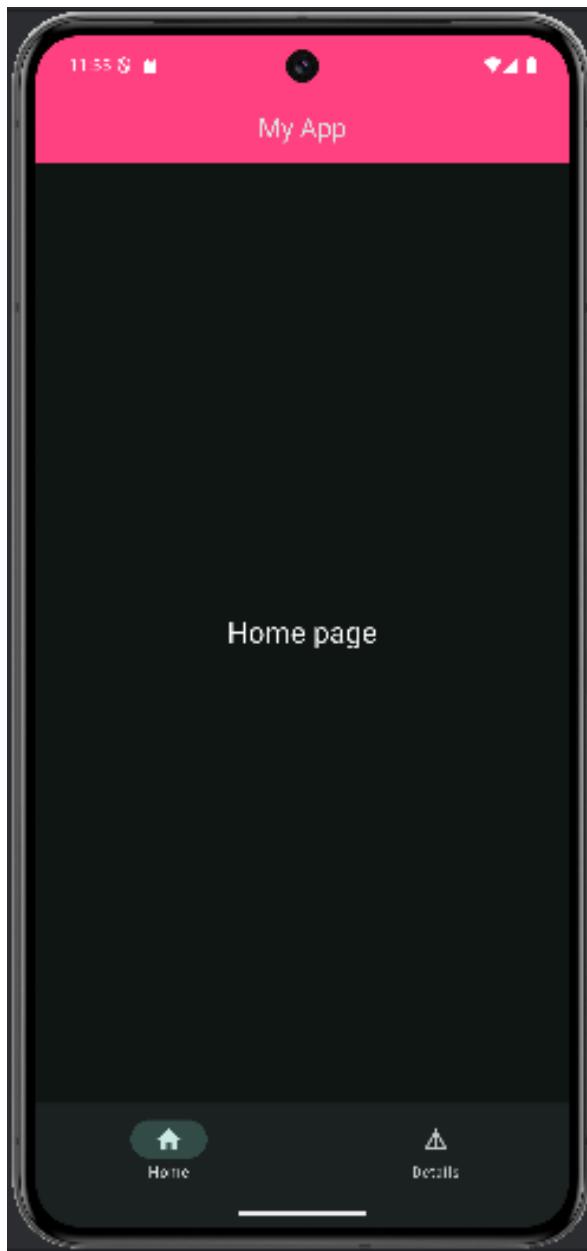
=> maintenant nous pouvons accéder **selectedIndexNotifier** de n'importe où dans l'**application** juste en utilisant ce **ValueListenableBuilder ()**

```
widget_tree.dart ×
5 import 'pages/details_page.dart';
6
7 List<Widget> pages = [HomePage(), DetailsPage()];
8
9
10 class WidgetTree extends StatelessWidget {
11   const WidgetTree({super.key});
12
13   @override
14     Windsurf: Explain | Refactor | Docstring | ×
15   Widget build(BuildContext context) {
16     return Scaffold(
17       appBar: AppBar(
18         title: Text("My App"),
19         centerTitle: true,
20         backgroundColor: Colors.pinkAccent,
21       ), // AppBar
22       body: ValueListenableBuilder(
23         valueListenable: selectedIndexNotifier,
24         builder: (context, selectedIndex, child) {
25           return pages.elementAt(selectedIndex);
26         },
27       ), // ValueListenableBuilder
28       bottomNavigationBar: NavbarWidget(),
29     ); // Scaffold
30   }
31 }
```

maintenant si nous cliquons sur :

details nous avons le detail

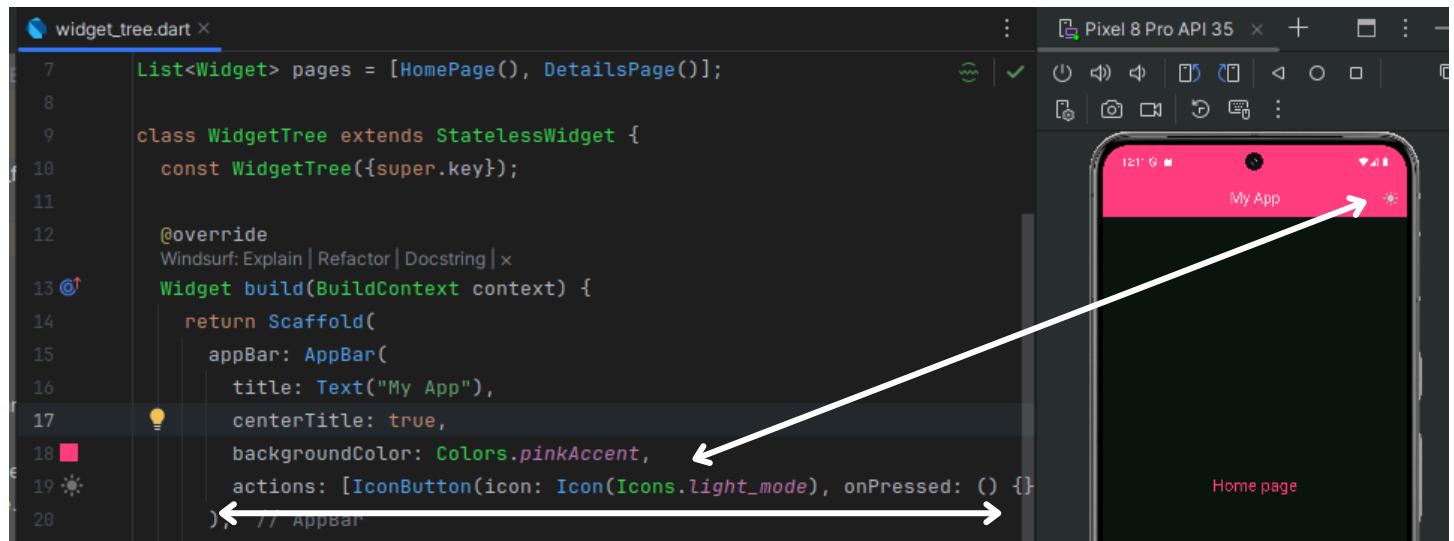
home nous avons home



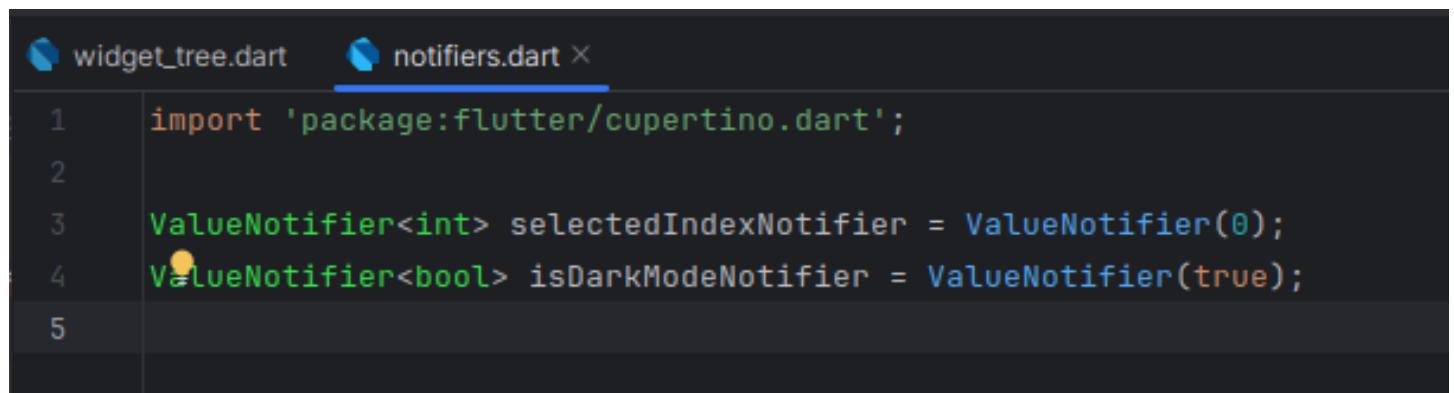
- **User le Notifier pour le mode sombre ou clair**

donc on vas :

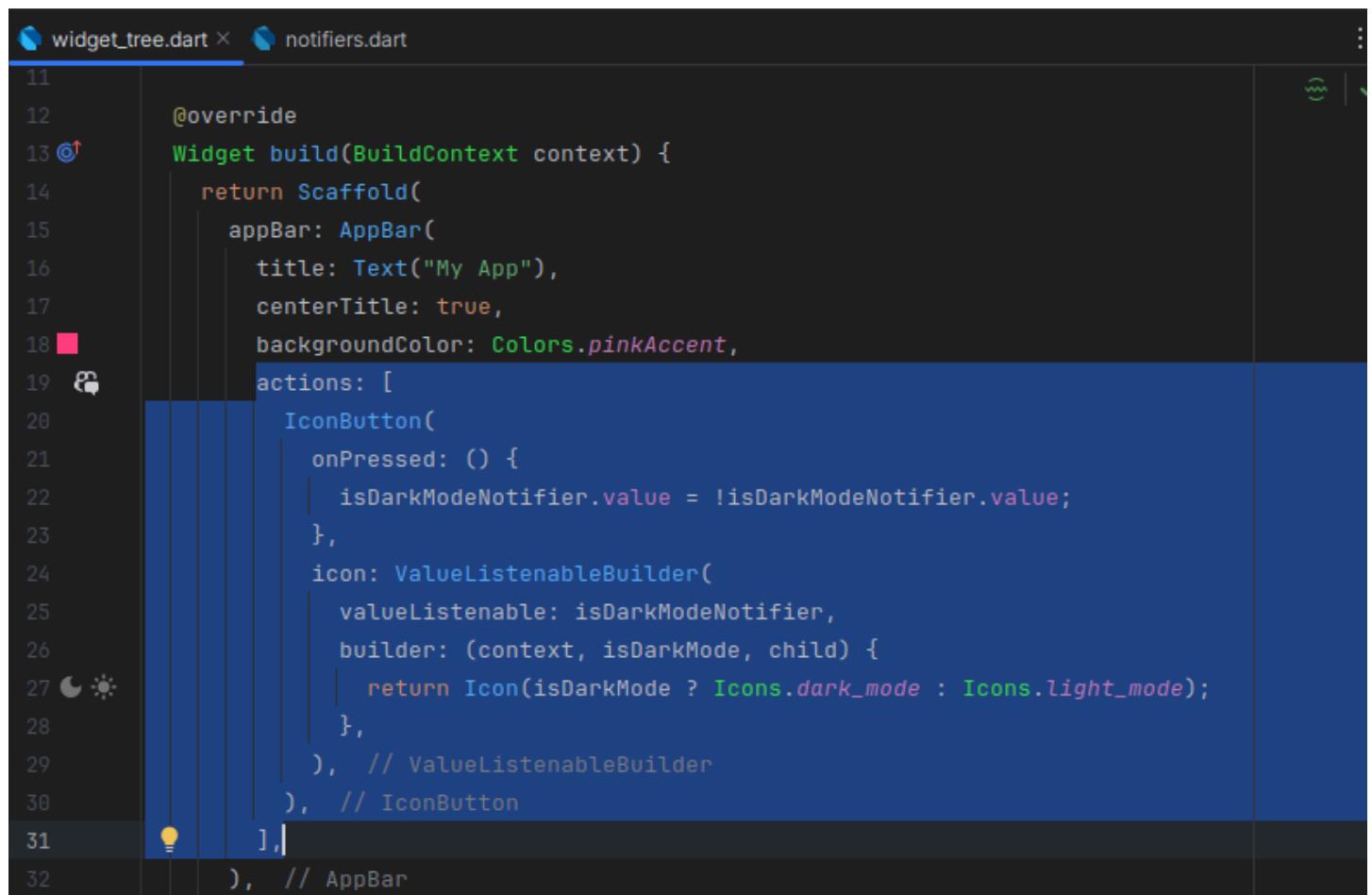
- supprimer : **brightness: Brightness.dark** dans **MaterialApp**
- ajouter un icon dans l'AppBar
- créer un nouveau notifier



```
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         actions: [IconButton(icon: Icon(Icons.light_mode), onPressed: () {})], →
20       ), // AppBar
21   }, // Scaffold
22 }
23
24 void main() {
25   runApp(WidgetTree());
26 }
```



```
1 import 'package:flutter/cupertino.dart';
2
3 ValueNotifier<int> selectedIndexNotifier = ValueNotifier(0);
4 ValueNotifier<bool> isDarkModeNotifier = ValueNotifier(true);
5
```



```
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         actions: [
20           IconButton(
21             onPressed: () {
22               isDarkModeNotifier.value = !isDarkModeNotifier.value;
23             },
24             icon: ValueListenableBuilder<
25               bool>(
26               valueListenable: isDarkModeNotifier,
27               builder: (context, isDarkMode, child) {
28                 return Icon(isDarkMode ? Icons.dark_mode : Icons.light_mode);
29               },
30             ), // ValueListenableBuilder
31           ), // IconButton
32         ],
33       ), // AppBar
34     );
35   }
36 }
```

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

on vas faire le même chose pour le brightness :

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

