

Faculté des technologies de l'information et de la communication

Département informatique appliquée



UNIVERSITÉ DES
MASCAREIGNES
SAVOIR, C'EST POUVOIR

Travaux pratique

TP3

Réalisé par : Nathan Carlinot RANDRIAMIHAJA

Année scolaire : 2023 – 2024

A.Container widget

Structure du Code :

1. Importation des Packages :

La première ligne du script importe le package `flutter/material.dart`. Ce package offre des widgets Dart ainsi que des widgets conformes au thème Material Design. Ces widgets seront utilisés pour construire l'interface utilisateur de notre application.

2. Fonction main() :

La fonction `main()` est le point d'entrée de tout projet Flutter. Elle indique où l'exécution de l'application doit commencer. Dans ce script, `main()` est utilisé pour appeler la fonction `runApp()`, qui lance l'application.

3. Fonction runApp() :

- Argument : `Container(color: Colors.green)`

Explication :

- `runApp()` nécessite un widget en argument, qui sera utilisé comme mise en page de l'application. Dans cet exemple, le widget fourni est un `Container`.
- `Container` est un widget de base qui peut contenir d'autres widgets. Il est configuré ici pour avoir une couleur verte, définie par `color: Colors.green`. Cette propriété peut être modifiée pour changer la couleur du conteneur.

Ce premier script établit les bases d'une application Flutter en important les bibliothèques nécessaires, en définissant la fonction principale `main()`, et en lançant une application avec un conteneur de couleur verte. Ce script simple sert d'introduction à la création d'interfaces utilisateur avec Flutter et peut être utilisé comme point de départ pour des développements plus complexes.

A screenshot of the Visual Studio Code interface. The left sidebar shows the file structure with 'main.dart' selected. The main editor window displays the code for 'main.dart'. The code imports 'package:flutter/material.dart' and sets the background color of a Container widget to Colors.green. The bottom right corner shows a mobile phone icon with a solid green screen, indicating the app is running on an Android emulator. The status bar at the bottom right shows 'Android-x64 emulator'.

```
// importing this package gives us the dart widgets
// as well as the Material Theme widgets
import 'package:flutter/material.dart';

// the main() function is the starting point for every Flutter project
void main() {
  // calling this method (you guessed it) runs our app
  runApp(
    // runApp() takes any widget as an argument.
    // This widget will be used as the layout.
    // We will give it a Container widget this time.
    Container(
      color: Colors.green, // <-- change this
    ),
  );
}
```

Voici alors le changement de couleur :

A screenshot of the Visual Studio Code interface, identical to the previous one but with a red background color instead of green. The code in the editor has been modified to set the Container's color to Colors.red. The mobile phone icon in the bottom right shows a solid red screen. The status bar at the bottom right shows 'Android-x64 emulator'.

```
// importing this package gives us the dart widgets
// as well as the Material Theme widgets
import 'package:flutter/material.dart';

// the main() function is the starting point for every Flutter project
void main() {
  // calling this method (you guessed it) runs our app
  runApp(
    // runApp() takes any widget as an argument.
    // This widget will be used as the layout.
    // We will give it a Container widget this time.
    Container(
      color: Colors.red, // <-- change this
    ),
  );
}
```

B. Text widget

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "STEP_03".
- Editor:** The "main.dart" file is open, displaying the code for a Flutter application. The title bar says "step_03".
- Terminal:** Shows the command "Flutter run key commands." followed by a list of hot reload and quit commands.
- Output:** Shows the message: "A Dart VM Service on sdk gphone64 x86 64 is available at: http://127.0.0.1:56726/B91HvG8Jrk=/ The Flutter DevTools debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:9100?uri=http://127.0.0.1".
- Device View:** An iPhone X simulator is running the application, displaying the text "Bonjour, Flutter!".

Ce deuxième script constitue une progression dans le développement d'une application Flutter en introduisant la structuration du code à l'aide de widgets personnalisés et en utilisant des composants plus avancés tels que `MaterialApp` et `Scaffold`. L'objectif est de créer une interface utilisateur plus élaborée avec un AppBar et un corps de page distinct.

Structure du Code :

1. Importation des Packages :

- La première ligne du script importe le package `flutter/material.dart`, tout comme dans le premier script.

2. Fonction main() :

- La fonction `main()` est le point d'entrée de l'application, appelant la méthode `runApp()` avec le widget `MyApp()` comme argument.

3. Classe MyApp :

- Type : `StatelessWidget`

Explication :

- `MyApp` est définie comme une classe qui étend `StatelessWidget`. Un `StatelessWidget` est utilisé lorsqu'une partie de l'interface utilisateur n'a pas besoin de changer au fil du temps.

- La méthode `build()` est implémentée pour définir la structure de l'interface utilisateur de l'application. Cette méthode retourne un widget `MaterialApp`.
- `MaterialApp` est utilisé pour configurer l'application avec le thème Material. Il contient un widget `Scaffold` en tant que page principale.
- `Scaffold` est un conteneur pour la structure de base de l'interface utilisateur, avec des propriétés telles que `appBar` (barre d'applications) et `body` (corps de la page).
- La barre d'applications (`AppBar`) contient un widget `Text` définissant le titre.
- Le corps de la page est défini par la méthode `myWidget()`

4. Fonction myWidget() :

Explication :

- `myWidget()` retourne un simple widget `Text` avec le texte "Hello, World!".

Ce script étend la complexité de l'application en introduisant des widgets plus avancés tels que `MaterialApp` et `Scaffold`. Il illustre la structuration du code en utilisant des widgets personnalisés et commence à créer une interface utilisateur plus sophistiquée avec une barre d'applications et un corps de page distinct contenant du texte statique.

Voici la combinaison de ces deux scripts :

The screenshot shows the Android Studio interface with the following details:

- EXPLORER:** Shows the project structure with files like main.dart, lib/main.dart.lib, STEP_03, and various build configurations for different platforms (linux, macos, web, windows).
- OPEN EDITORS:** Shows the main.dart file open in the editor.
- main.dart:**

```

1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   @override
9   Widget build(BuildContext context) {
10     return MaterialApp(
11       home: Scaffold(
12         appBar: AppBar(
13           title: Text("Exploring Widgets"),
14         ), // AppBar
15         body: myWidget(),
16       ), // Scaffold
17     ); // MaterialApp
18   }
19 }
20
21 Widget myWidget() {
22   return Container(
23     color: Colors.green,
24     child: Center(
25       child: Text(
26         "Hello, World!",
27       ),
28     ),
29   );
30 }
31 
```
- PROBLEMS:** Shows some Dart VM Service logs related to the Flutter DevTools.
- OUTPUT:** Shows logs from the Flutter DevTools.
- DEBUG CONSOLE:** Shows logs from the Flutter DevTools.
- TERMINAL:** Shows logs from the Flutter DevTools.
- PORTS:** Shows logs from the Flutter DevTools.
- DEVICE:** Shows a smartphone emulator displaying the application with the title "Exploring Widgets" and the text "Hello, World!".
- SIDE BAR:** Shows various icons for device controls (power, volume, camera, search, etc.) and a sidebar with tabs like "powershell" and "git".

Ensuite j'ai ajouté la deuxième fonction Text en utilisant une Column pour organiser verticalement les deux textes. Compilez ce script pour voir comment ces modifications concernent l'interface utilisateur de l'application Flutter.

J'ai intégré une nouvelle fonction `myWidget()` qui affiche un autre texte. L'objectif est de démontrer comment incorporer de nouveaux éléments dans la structure existante de l'application Flutter.

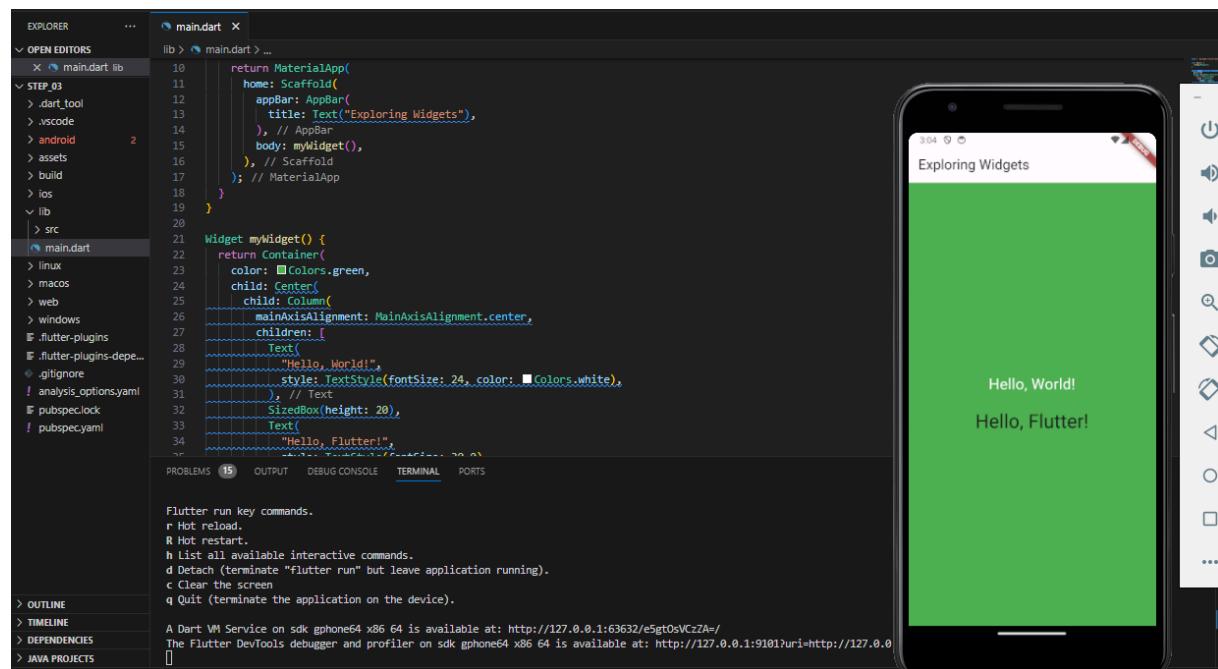
5. Evolution du Code_1 :

Ajout de la Troisième Fonction myWidget() :

Explication :

- La nouvelle fonction `myWidget()` est ajoutée à la suite du script. Elle crée un conteneur coloré avec un fond vert, centré verticalement et horizontalement.
- A l'intérieur du conteneur, un widget `Column` est utilisé pour organiser verticalement les deux widgets `Text`. Le premier affiche "Hello, World!" avec une taille de police de 24, et le deuxième affiche "Hello, Flutter!" avec une taille de police de 30.
- Un espace vertical (`SizedBox`) de 20 unités est ajouté entre les deux textes pour une mise en page plus aérée.

Cette intégration réussie de la troisième fonction `myWidget()` enrichit l'interface utilisateur en affichant deux textes distincts de manière organisée. Le script continue à illustrer la construction progressive d'une application Flutter en combinant des widgets et en ajoutant des fonctionnalités supplémentaires. Ces modifications contribuent à la compréhension du développement d'interfaces utilisateur évolutives avec Flutter.



La quatrième fonction myWidget() est ajoutée au script existant pour démontrer l'utilisation du widget Padding. L'objectif est de montrer comment appliquer des marges autour d'un widget pour améliorer la mise en page de l'interface utilisateur

6. Evolution du Code_2 :

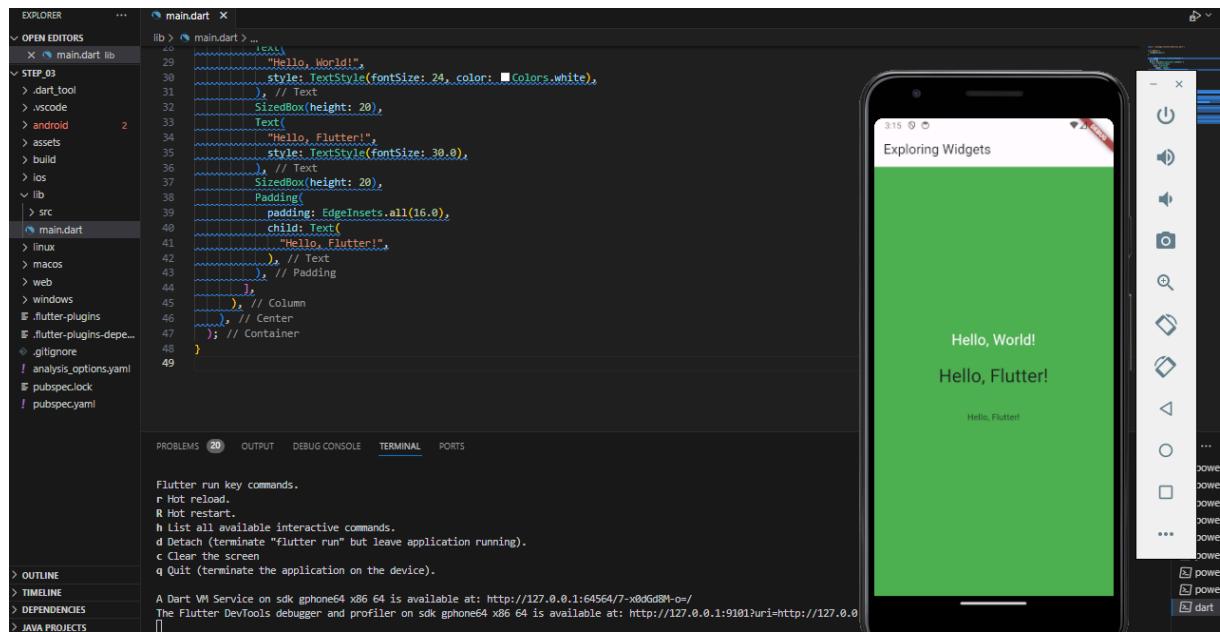
Ajout de la Quatrième Fonction myWidget() :

Explications :

La nouvelle fonction myWidget() utilise le widget Padding pour ajouter des marges autour du widget Text. Les marges sont définies à 16 pixels de tous les côtés (EdgeInsets.all(16.0)).

L'utilisation de Padding permet de créer un espace vide autour du texte, améliorant ainsi la lisibilité et l'apparence générale de l'interface utilisateur.

L'ajout de la quatrième fonction myWidget() démontre comment appliquer des marges avec le widget Padding pour améliorer la disposition des éléments dans l'interface utilisateur. Ces ajustements contribuent à une conception plus esthétique et fonctionnelle de l'application Flutter en évolution. Le script continue de servir d'exemple progressif pour comprendre les différents composants et concepts de Flutter.



La cinquième fonction `myWidget()` est ajoutée au script existant, introduisant le widget `RaisedButton` pour créer un bouton interactif. L'objectif est de démontrer l'incorporation d'éléments interactifs dans une application Flutter.

7. Evolution du Code_3 :

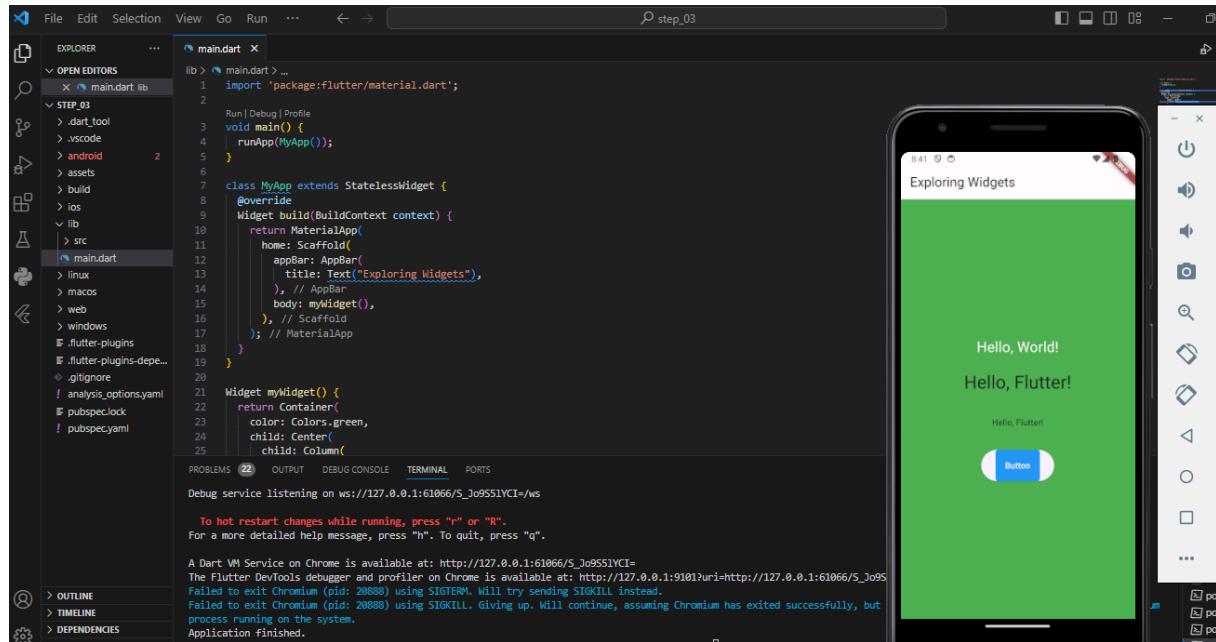
Ajout de la Cinquième Fonction myWidget() :

Explications :

- La nouvelle fonction `myWidget()` utilise le widget `RaisedButton` pour créer un bouton. Ce bouton affiche le texte "Button" et un fond bleu, une élévation de 4,0 pixels et une couleur de surbrillance jaune.

- La propriété `onPressed` du bouton est actuellement configurée pour ne rien faire (`// faire quelque chose`). Elle peut être remplacée par une fonction qui sera exécutée lorsque le bouton est enfoncé.

L'ajout de la cinquième fonction `myWidget()` introduit un élément interactif avec le widget `RaisedButton`. Cela montre comment intégrer des éléments tels que des boutons dans une application Flutter pour permettre des interactions utilisateur. Le script continue à évoluer en démontrant divers widgets et leur utilisation dans la construction d'interfaces utilisateur avec Flutter.



Nous avons ajouté deux nouveaux boutons à cette étape : un ElevatedButton qui a une couleur de fond bleue et un InkWell qui a une surbrillance personnalisée en vert.

Le bouton surélevé, qui ressemble à un bouton matériel, a une couleur de fond bleue. Il gère l'effet d'encre lorsqu'on appuie dessus avec de l'encre.

InkWell est un widget qui affiche une surbrillance personnalisée en réponse au toucher. Il simule le comportement d'un bouton avec une surbrillance personnalisée.

Je peux modifier le texte et les actions des boutons en fonction de l'application que vous utilisez.

The screenshot shows the VS Code interface with the main.dart file open in the editor. The code defines a green-themed UI with a central button labeled "Hello, Flutter!". An iPhone Xd4 emulator is running in the background, showing the same green screen with the text "Hello, World!" and "Hello, Flutter!". The VS Code status bar indicates "Ln 82, Col 1 Spaces: 2 UTF-8 CRLF".

```

52     ),
53   ), // BoxDecoration
54   child: Padding(
55     padding: EdgeInsets.all(16.0),
56   ),
57   child: Text(
58     'Hello, World!',
59     style: TextStyle(color: Colors.white),
60   ),
61   ), // Padding
62   ), // ElevatedButton
63   SizedBox(height: 20),
64   InkWell(
65     onTap: () {
66       // do something
67     },
68   ),
69   splashColor: Colors.green, // Couleur de surbrillance personnalisée
70   child: Container(
71     padding: EdgeInsets.all(16.0),
72     child: Text(
73       'Hello, Flutter!',
74       style: TextStyle(color: Colors.black),
75     ),
76   ),
77   ), // InkWell
78 )
79 ),
80 ),
81 );
82 )
83 )

```

Nous avons ajouté un nouveau widget nommé `TextField` dans cette mise à jour. Ce widget est souvent utilisé pour la saisie de données et permet à l'utilisateur d'entrer du texte. En utilisant la propriété `decoration`, nous avons modifié l'apparence du champ de texte, qui comprend la bordure : `InputBorder.none` pour supprimer la bordure par défaut du champ de texte et `hintText` pour afficher un texte d'invitation lorsqu'il est vide.

L'objectif de cette modification est de permettre à l'utilisateur d'interagir avec votre application en entrant du texte.

The screenshot shows the VS Code interface with the main.dart file open in the editor. The code now includes a `TextField` with a placeholder "Write something here". An iPhone Xd4 emulator is running in the background, showing the updated green screen with the text input field and the placeholder text. A virtual keyboard is visible at the bottom. The VS Code status bar indicates "Ln 83, Col 1 Spaces: 2 UTF-8 CRLF".

```

52     ),
53   ), // ElevatedButton
54   SizedBox(height: 20),
55   // Ajout du bouton InkWell avec surbrillance personnalisée
56   InkWell(
57     onTap: () {
58       // Action à effectuer lorsqu'on appuie sur le bouton
59     },
60   ),
61   splashColor: Colors.green, // Couleur de surbrillance personnalisée
62   child: Container(
63     padding: EdgeInsets.all(16.0),
64     child: Text(
65       'Hello, World!',
66       style: TextStyle(color: Colors.white),
67     ),
68   ),
69   ), // Container
70   ),
71   ), // Column
72   ),
73   ), // Center
74   );
75 )
76 )
77 )
78 )
79 )
80 )
81 )
82 )
83 )

```

« Write something here » indique que l'utilisateur peut écrire, voici le résultat :

The screenshot shows the VS Code interface with the following details:

- File Explorer (Left):** Shows the project structure with files like `main.dart`, `lib`, `assets`, and `build`.
- Code Editor (Top):** Displays the `main.dart` file content in Dart. The code creates a button labeled "Button" with the text "oui je suis à la maison!".
- Terminal (Bottom Left):** Shows log output from the Flutter emulator, including CPU usage statistics for EGL emulation.
- Output (Bottom Right):** Shows the text "maison Maison maisonette".
- Flutter Device View (Right):** Shows a green screen with a blue button labeled "Button". The text "oui je suis à la maison!" is displayed above the button.

J'ai ajouté des commentaires dans cette explication pour expliquer le nouveau widget `ListView.builder`. Cela permet de créer une liste dynamique d'éléments avec une hauteur et une marge autour spécifiées. Le widget à afficher pour chaque élément de la liste est choisi à l'aide de la fonction `itemBuilder`.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure with files like `main.dart`, `lib/main.dart`, and `lib/src/main.dart`.
- Code Editor:** Displays the `main.dart` file containing Dart code for a Flutter application.
- Terminal:** Shows command-line output related to frame tracking and a perfecto trigger.
- Output:** Shows the application's UI on an Android device, titled "Exploring Widgets". The screen displays a blue button labeled "Button" and a list of 14 rows labeled "Row 0" through "Row 14".

```
File Edit Selection View Go Run ... step_03

EXPLORER
OPEN EDITORS
STEP_03
> .dart_tool
> .vscode
> android
> assets
> build
> ios
> lib
> src
main.dart
> linux
> macos
> web
> windows
> flutter-plugins
> flutter-plugins-deps...
> .gitignore
> analysis_options.yaml
> pubspec.lock
> pubspec.yaml

main.dart x
lib > main.dart > myWidget
73   button',
74   style: TextStyle(color: Colors.black),
75   ), // Text
76   ), // Container
77   ), // InkyList
78
79   SizedBox(height: 20),
80
81   // Ajout du widget ListView.builder
82   Expanded(
83     child: ListView.builder(
84       padding: EdgeInsets.all(16.0),
85       itemExtent: 30.0,
86       itemBuilder: (BuildContext context, int index) {
87         return Text('Row $index');
88       },
89     ), // ListView.builder
90   ), // Expanded
91   ],
92   ), // Column
93   ), // Center
94   ); // Container
95
96
97
98
99
100
101
102

PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL PORTS

W/FrameTracker( 8369): Missed SF frame:UNKNOWN: 88, 95476, 0, CUJ=1IME_INSETS_ANIMATION:1@com.example.myartist
W/FrameTracker( 8369): Missed SF frame:UNKNOWN: 88, 95498, 0, CUJ=1IME_INSETS_ANIMATION:1@com.example.myartist
W/FrameTracker( 8369): Missed SF frame:UNKNOWN: 88, 95506, 0, CUJ=1IME_INSETS_ANIMATION:1@com.example.myartist
W/FrameTracker( 8369): Missed SF frame:UNKNOWN: 88, 95514, 0, CUJ=1IME_INSETS_ANIMATION:1@com.example.myartist
W/FrameTracker( 8369): Missed SF frame:UNKNOWN: 88, 95523, 0, CUJ=1IME_INSETS_ANIMATION:1@com.example.myartist
W/FrameTracker( 8369): Missed SF frame:UNKNOWN: 88, 95545, 0, CUJ=1IME_INSETS_ANIMATION:1@com.example.myartist
W/FrameTracker( 8369): Missed SF frame:UNKNOWN: 88, 95568, 0, CUJ=1IME_INSETS_ANIMATION:1@com.example.myartist
W/FrameTracker( 8369): Missed SF frame:UNKNOWN: 88, 95590, 0, CUJ=1IME_INSETS_ANIMATION:1@com.example.myartist
V/PerfettoTrigger( 8369): Triggering /system/bin/trigger_perfetto com.android.telemetry.interaction-jank-monitor-69
Application finished.

OUTLINE
TIMELINE
DEPENDENCIES
```

Cette mise à jour a remplacé le précédent ListView.builder par un nouveau qui utilise des ListTile pour afficher des éléments de liste structurés avec un titre. Lorsqu'on appuie sur un élément de la liste, la fonction onTap permet de sélectionner une action à effectuer.

```

lib > main.dart > ...
90     itemBuilder: (BuildContext context, int index) {
91       return ListTile(
92         title: Text('Row $index'),
93         onTap: () {
94           // Action à effectuer lorsqu'on appuie sur la ListTile
95         },
96       ); // ListTile
97     }, // ListView.builder
98   ), // Expanded
99
100  // Explication du nouveau ListView.builder avec ListTile :
101  // Le widget ListView.builder est utilisé pour construire une liste d'éléments dynamique.
102  // 'itemBuilder' est une fonction callback appellée pour chaque élément de la liste,
103  // et elle renvoie le widget à afficher pour cet élément.
104  // Ici, nous utilisons des ListTile pour afficher des éléments de liste structurés avec un titre.
105  // 'onTap' est une fonction callback appellée lorsqu'on appuie sur un élément de la liste.
106  // ...
107  ),
108  ), // Column
109  ), // Center
110  ); // Container
111 }
112 
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

A Dart VM Service on sdk gphone64 x86_64 is available at: http://127.0.0.1:53640/_Bm03KUav88=/
The Flutter DevTools debugger and profiler on sdk gphone64 x86_64 is available at: http://127.0.0.1:9102?uri=http://127.0.0.1
D/EGL_emulation(9880): app_time_stats: avg=29352.00ms min=9.21ms max=58694.79ms count=2
D/EGL_emulation(9880): app_time_stats: avg=250.80ms min=14.34ms max=4839.44ms count=22
D/EGL_emulation(9880): app_time_stats: avg=45.97ms min=3.35ms max=651.54ms count=26
D/EGL_emulation(9880): app_time_stats: avg=76.57ms min=8.32ms max=759.07ms count=18
D/EGL_emulation(9880): app_time_stats: avg=96.69ms min=18.91ms max=1000.78ms count=17
D/EGL_emulation(9880): app_time_stats: avg=17.32ms min=3.72ms max=54.20ms count=43
D/EGL_emulation(9880): app_time_stats: avg=134.93ms min=35.49ms max=754.40ms count=8
D/EGL_emulation(9880): app_time_stats: avg=52.47ms min=4.34ms max=869.79ms count=27

OUTLINE TIMELINE DEPENDENCIES JAVA PROJECTS

Et voici mon résultat final après avoir arranger l'interface :

```

lib > main.dart > ...
43   );
44   ),
45   child: Padding(
46     padding: const EdgeInsets.all(16.0),
47     child: Text(
48       'Button',
49       style: TextStyle(color: Colors.black),
50     ), // Text
51   ), // Padding
52   ), // ElevatedButton
53
54   SizedBox(height: 20),
55
56   // Ajout du bouton InkWell avec surbrillance personnalisée
57   InkWell(
58     onTap: () {
59       // Action à effectuer lorsqu'on appuie sur le bouton
60     },
61     splashColor: Colors.green, // Couleur de surbrillance personnalisée
62     child: Container(
63       padding: EdgeInsets.all(16.0),
64       child: Text(
65         'Button',
66         style: TextStyle(color: Colors.black),
67       ), // Text
68     ), // Container
69   ), // InkWell
70 
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Flutter run key commands.
r Hot reload.
R Hot restart.
h List all available interactive commands.
d Detach (terminate "Flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).

A Dart VM Service on sdk gphone64 x86_64 is available at: http://127.0.0.1:55222/yCpIEY14NN0=/
The Flutter DevTools debugger and profiler on sdk gphone64 x86_64 is available at: http://127.0.0.1:9102?uri=http://127.0.0.1

OUTLINE TIMELINE DEPENDENCIES JAVA PROJECTS

SETUP

Le code crée une application Flutter de base avec une structure matérielle comprenant une barre d'applications et un corps de page. Actuellement, la méthode myLayoutWidget() renvoie un simple widget texte, mais vous pouvez la personnaliser pour créer des mises en page plus complexes en ajoutant d'autres widgets.

The screenshot shows the VS Code interface with the main.dart file open in the editor. The code defines a StatelessWidget named MyApp that returns a MaterialApp with a Scaffold containing a Centered Text widget. The emulator on the right shows the app running on an iPhone X, displaying the text "Hello world!".

```

lib > main.dart > MyApp > build
3 // entry point for the app,
4 // the >> operator is shorthand for {} when there is only one line of code
5 void main() >> runApp(MyApp());
6
7 // the root widget of our application
8 class MyApp extends StatelessWidget {
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      home: Scaffold(
13        appBar: AppBar(
14          title: Text("Building layouts"),
15        ), // AppBar
16        body: myLayoutWidget(),
17      ), // Scaffold
18    ); // MaterialApp
19  }
20
21 // replace this method with code in the examples below
22 Widget myLayoutWidget() {
23   return Text("Hello world!");
24 }
25

```

Cette mise à jour a modifié la méthode myLayoutWidget() pour retourner un widget Padding. Ce widget ajoute de l'espace (padding) autour du widget enfant, qui est actuellement un simple widget Text avec le texte "Hello world!". La valeur EdgeInsets.all(8.0) indique une marge de chaque côté du texte de 8 pixels.

The screenshot shows the VS Code interface with the main.dart file open in the editor. The code now includes a call to myLayoutWidget() which returns a Padding widget with a child Text("Hello world!"). The emulator on the right shows the app running on an iPhone X, displaying the text "Hello world!" with padding around it.

```

lib > main.dart > runApp(MyApp());
3 void main() >> runApp(MyApp());
4
5 class MyApp extends StatelessWidget {
6   @override
7   Widget build(BuildContext context) {
8     return MaterialApp(
9       home: Scaffold(
10         appBar: AppBar(
11           title: Text("Building layouts"),
12         ), // AppBar
13         body: myLayoutWidget(),
14       ), // Scaffold
15     ); // MaterialApp
16   }
17
18 Widget myLayoutWidget() {
19   return Padding(
20     padding: EdgeInsets.all(8.0),
21     child: Text("Hello world!"),
22   ); // Padding
23 }
24

```

Cette mise à jour a modifié la méthode myLayoutWidget() pour retourner un widget Center. Le widget Center place son unique enfant au centre de l'écran. De plus, la police du texte a été augmentée en utilisant le style TextStyle avec la propriété fontSize : 30.

File Edit Selection View Go Run ... ↶ ↷ ⚡ step_03

EXPLORER

OPEN EDITORS

STEP_03

main.dart lib

lib > main.dart ...

```
4
5 class MyApp extends StatelessWidget {
6   @override
7   Widget build(BuildContext context) {
8     return MaterialApp(
9       home: Scaffold(
10         appBar: AppBar(
11           title: Text("Building layouts"),
12         ), // AppBar
13         body: myLayoutWidget(),
14       ), // Scaffold
15     ); // MaterialApp
16   }
17 }
18
19 Widget myLayoutWidget() {
20   return Center(
21     child: Text(
22       "Hello world!",
23       style: TextStyle(fontSize: 30),
24     ), // Text
25   ); // Center
26 }
```

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Flutter run key commands.

- r Hot reload.
- R Hot restart.
- h List all available interactive commands.
- d Detach (terminate "flutter run" but leave application running).
- c Clear the screen
- q Quit (terminate the application on the device).

A Dart VM Service on sdk gphone64 x86 64 is available at: http://127.0.0.1:61600/ajoin?xsgye=/
The Flutter DevTools debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:9101?uri=http://127.0.0.1

LN 27, Col 1 SPACES: 2 UTF-8 CR/LF

OUTLINE

TIMELINE

DEPENDENCIES

JAVA PROJECTS

2 6 0 9 0

La méthode `myLayoutWidget()` utilise maintenant le widget `Align` pour positionner son enfant (`Text`) en haut et au centre de l'écran dans cette version mise à jour. La propriété d'alignement est `alignment`. Selon `topCenter`, le texte doit être aligné au centre de l'écran supérieur.

File Edit Selection View Go Run ... ← → 🔍 step_03

EXPLORER OPEN EDITORS 1 unsaved

main.dart

lib > main.dart > ...

```
5  <class MyApp extends StatelessWidget {  
6    @override  
7    Widget build(BuildContext context) {  
8      return MaterialApp(  
9        home: Scaffold(  
10          appBar: AppBar(  
11            title: Text("Building layouts"),  
12          ), // AppBar  
13          body: myLayoutWidget(),  
14        ), // Scaffold  
15      ); // MaterialApp  
16    }  
17  }  
18  
19  Widget myLayoutWidget() {  
20    return Align(  
21      alignment: Alignment.topCenter,  
22      child: Text(  
23        "Hello",  
24        style: TextStyle(fontSize: 30),  
25      ), // Text  
26    ); // Align  
27  }  
28}
```

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Flutter run key commands.

- r Hot reload.
- R Hot restart.
- b List all available interactive commands.
- d Detach (terminate "flutter run" but leave application running).
- c Clear the screen
- q Quit (terminate the application on the device).

> OUTLINE

> TIMELINE

> DEPENDENCIES

> JAVA PROJECTS

A Dart VM Service on sdk_phone_x64 x86_64 is available at: http://127.0.0.1:65235/gQ4451KnFlk=</p>

I/Choreographer(5812): Skipped 32 frames! The application may be doing too much work on its main thread.

The Flutter DevTools debugger and profiler on sdk_phone_x64 x86_64 is available at: http://127.0.0.1:9101?uri=http://127.0.0.1

La méthode `myLayoutWidget()` dans cette version mise à jour utilise un widget `Container` pour englober le texte "Hello". Le conteneur a des propriétés telles que le bord, la couverture, l'alignement, la largeur, la hauteur et la décoration pour définir diverses caractéristiques visuelles.

- marges : créez une marge autour du récipient.
 - Ajouter du remplissage à l'intérieur du récipient.
 - alignement : aligner les éléments du conteneur.
 - La largeur et la hauteur du récipient sont déterminées par eux.
 - Décoration : Définit un style visuel, dans ce cas une couleur de fond verte avec une bordure.

Dans cette mise à jour, la méthode myLayoutWidget() utilise un widget Row pour aligner horizontalement les icônes Icons.home. Dans cette configuration, les icônes sont répétées quatre fois.

The screenshot shows the Visual Studio Code interface with a Flutter project. The left sidebar lists files like `main.dart.lib`, `STEP_03`, and `analysis_options.yaml`. The main editor area contains the `main.dart` file with code for a home screen. The preview window shows the resulting application on an iPhone X simulator. The bottom right corner displays the iOS control center.

```
EXPLORER
OPEN EDITORS
STEP_03
> dart tool
> vscode
> android
> assets
> build
> ios
> lib
> linux
> macos
> web
> windows
flutter-plugins
flutter-plugins-deps...
.githignore
analysis_options.yaml
pubspec.lock
pubspec.yaml

OUTLINE
TIMELINE
DEPENDENCIES
JAVA PROJECTS
2.0.0 11 %0

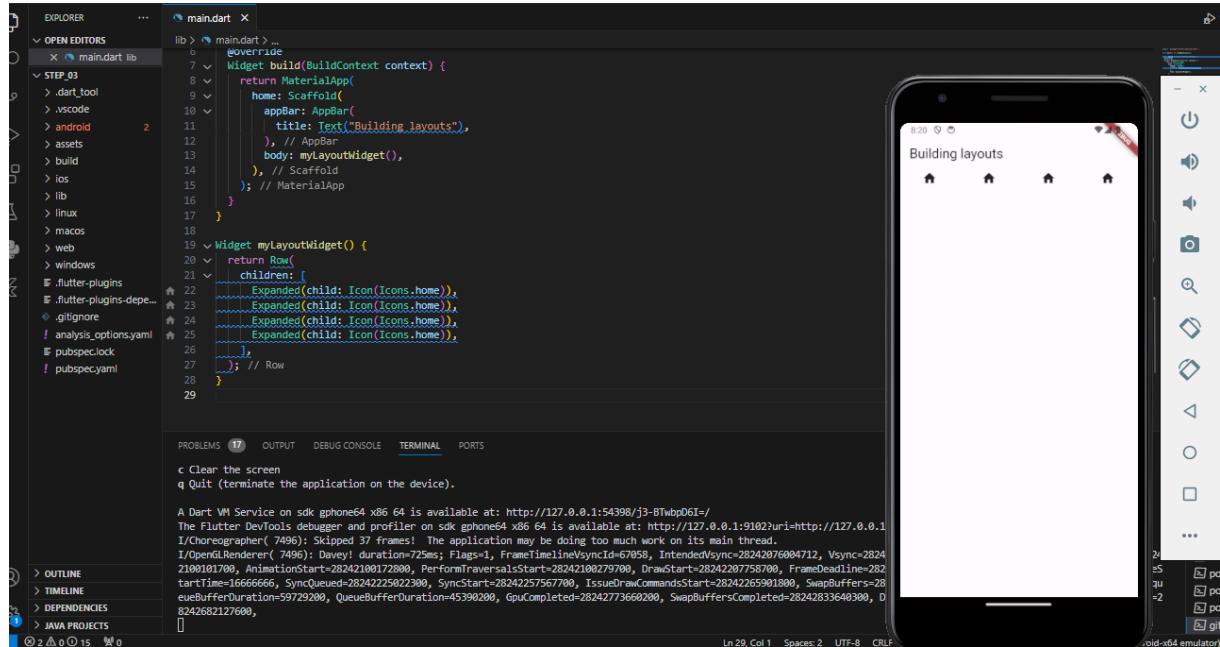
main.dart x
lib > main.dart > ...
  > > override
  7 > > > Widget build(BuildContext context) {
  8 > > >   return MaterialApp(
  9 > > >     home: Scaffold(
 10 > > >       appBar: AppBar(
 11 > > >         title: Text("Building layouts"),
 12 > > >       ), // AppBar
 13 > > >       body: myLayoutWidget(),
 14 > > >     ), // Scaffold
 15 > >   ); // MaterialApp
 16 > }
 17 >
 18 >
 19 > > > Widget myLayoutWidget() {
 20 > > >   return Row(
 21 > > >     children: [
 22 > > >       Icon(Icons.home),
 23 > > >       Icon(Icons.home),
 24 > > >       Icon(Icons.home),
 25 > > >       Icon(Icons.home),
 26 > > >     ],
 27 > >   ); // Row
 28 >
 29 >

PROBLEMS (13) OUTPUT DEBUG CONSOLE TERMINAL PORTS
Flutter run key commands.
r Hot reload.
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).

A Dart VM Service on sdk gphone64 x86 64 is available at: http://127.0.0.1:50681/Z8gjw07nz04=
The Flutter DevTools debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:9101?uri=http://127.0.0.1
I/Choreographer( 6143): Skipped 40 frames! The application may be doing too much work on its main thread.
[]

29. Col 1 Spaces: 2 UTF-8 CRLF
old-x84 emulator)
```

Dans cette mise à jour, la méthode myLayoutWidget() utilise un widget Row où chaque icône est enveloppée dans un widget Expanded. Cela permet à chaque icône d'occuper la moitié de l'espace horizontal disponible dans la rangée.



```

EXPLORER ... main.dart
OPEN EDITORS STEP_03
> dart_tool
> vscode
> android 2
> assets
> build
> ios
> lib
> linux
> macos
> web
> windows
> flutter-plugins
> flutter-plugins-deps
> .gitignore
> analysis_options.yaml
> pubspec.lock
> pubspec.yaml

main.dart
6     @override
7     Widget build(BuildContext context) {
8         return MaterialApp(
9             home: Scaffold(
10                 appBar: AppBar(
11                     title: Text("Building layouts"),
12                     // AppBar
13                 ),
14                 body: myLayoutWidget(),
15             ), // Scaffold
16         ); // MaterialApp
17     }
18
19     Widget myLayoutWidget() {
20         return Row(
21             children: [
22                 Expanded(child: Icon(Icons.home)),
23                 Expanded(child: Icon(Icons.home)),
24                 Expanded(child: Icon(Icons.home)),
25                 Expanded(child: Icon(Icons.home)),
26             ],
27         ); // Row
28     }
29

```

PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL PORTS

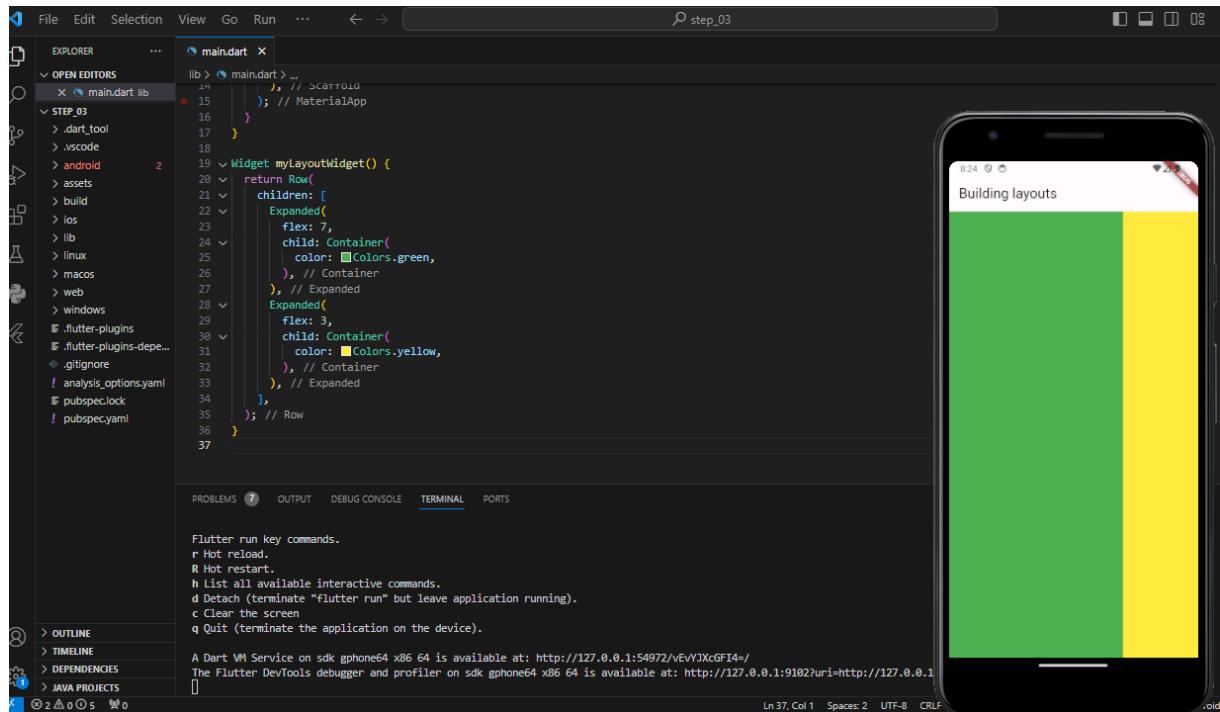
c Clear the screen
q Quit (terminate the application on the device).

A Dart VM Service on sdk gphone64 x86 64 is available at: http://127.0.0.1:54398/j3-BTwbpD6I=/ The Flutter DevTools debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:9102?url=http://127.0.0.1:1/Choreographer/7496; Skipped 37 frames! The application may be doing too much work on its main thread.
I/OpenGLRenderer(7496): Davey! duration=725ms; Flags=1, FrameTimelineVSyncId=67058, IntendedVSync=28242076004712, Vsync=282420101700, AnimationStart=28242100172800, PerformTraversalsStart=28242100279700, DrawStart=2824220758700, FrameDeadline=2824220766666, SyncQueued=2824225022300, SyncStart=28242257567700, IssueDrawCommandsStart=28242265901800, SwapBuffers=28242682127600, QueueBufferDuration=45390200, GpuCompleted=28242773660200, SwapBuffersCompleted=28242833648300, D

OUTLINE TIMELINE DEPENDENCIES JAVA PROJECTS

Ln 29 Col 1 Spaces: 2 UTF-8 CR/LF

Dans cette mise à jour, la méthode myLayoutWidget() utilise un widget Row contenant deux conteneurs enveloppés dans des widgets Expanded. Le premier conteneur a une flexibilité de 7 et le deuxième a une flexibilité de 3. Cela signifie que le premier conteneur occupera soixante-dix pour cent de l'espace horizontal disponible, tandis que le deuxième conteneur occupera trente pour cent.



```

File Edit Selection View Go Run ... ← → ⌂ step_03
EXPLORER ...
OPEN EDITORS STEP_03
> dart_tool
> vscode
> android 2
> assets
> build
> ios
> lib
> linux
> macos
> web
> windows
> flutter-plugins
> flutter-plugins-deps
> .gitignore
> analysis_options.yaml
> pubspec.lock
> pubspec.yaml

main.dart
14     ), // Scaffold
15 ); // MaterialApp
16
17
18
19     Widget myLayoutWidget() {
20         return Row(
21             children: [
22                 Expanded(
23                     flex: 7,
24                     child: Container(
25                         color: Colors.green,
26                     ), // Container
27                 ), // Expanded
28                 Expanded(
29                     flex: 3,
30                     child: Container(
31                         color: Colors.yellow,
32                     ), // Container
33                 ), // Expanded
34             ],
35         ); // Row
36     }
37

```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Flutter run key commands.
r Hot reload.
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen.
q Quit (terminate the application on the device).

A Dart VM Service on sdk gphone64 x86 64 is available at: http://127.0.0.1:54972/vEVXcGFI4=/ The Flutter DevTools debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:9102?url=http://127.0.0.1:1/

Ln 37, Col 1 Spaces: 2 UTF-8 CR/LF

Dans cette mise à jour, la méthode myLayoutWidget() empile les icônes Icons.home avec un widget Stack. Les icônes sont par défaut superposées dans le coin supérieur gauche du Stack.

```

File Edit Selection View Go Run ...
main.dart ...
lib > main.dart > ...
15   ); // MaterialApp
16 }
17 }
18
19 Widget myLayoutWidget() {
20   return Stack(
21     children: [
22       Icon(Icons.home),
23       Icon(Icons.home),
24       Icon(Icons.home),
25       Icon(Icons.home),
26     ],
27   ); // Stack
28 }

```

PROBLEMS 13 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Flutter run key commands.

A Dart VM Service on sdk gphone64 x86 64 is available at: http://127.0.0.1:5398/vF6vu8ipqQEw/
The Flutter DevTools debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:9102?uri=http://127.0.0.1

Ln 29, Col 1 Spaces: 2 UTF-8 CRLF

Cette mise à jour place l'image en bas à droite du Stack et ajoute du texte ('Baaaaaa') dans le coin inférieur droit avec une police de caractères de taille 30.

```

File Edit Selection View Go Run ...
main.dart ...
lib > main.dart > ...
15   ); // MaterialApp
16 }
17 }
18
19 Widget myLayoutWidget() {
20   return Stack(
21     // any unpositioned children (i.e., our text) will be aligned at the bottom right
22     alignment: Alignment.bottomRight,
23     children: [
24       // first child in the stack is on the bottom
25       Image.asset('images/sheep.jpg'), // <-- image
26
27       // second child in the stack
28       Padding(
29         padding: EdgeInsets.all(16.0),
30         child: Text(
31           'Baaaaaa', // <-- text
32           style: TextStyle(fontSize: 30),
33         ), // Text
34       ), // Padding
35     ],
36   ); // Stack
37 }

```

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Flutter run key commands.

A Dart VM Service on sdk gphone64 x86 64 is available at: http://127.0.0.1:56440/RCoKLeZgJlgw/
The Flutter DevTools debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:9102?uri=http://127.0.0.1

Ln 38, Col 1 Spaces: 2 UTF-8 CRLF

Normalement l'image devrait se montrer, mais il ya eu probleme.

Dans cette version, la méthode myLayoutWidget() utilise un widget Column pour organiser plusieurs widgets Row et Text. Chaque Row contient une combinaison d'icônes et de texte.

The screenshot shows the VS Code interface with the main.dart file open. The code uses a Column widget with three children, each being a Row widget. The first Row contains an icon and text ('BEAMS'). The second Row contains text ('description...'). The third Row contains text ('EXPLORE BEAMS') and an arrow forward icon. The iPhone X simulator on the right shows the resulting UI with rounded corners and a purple header.

```

main.dart
15   lib > main.dart > ...
16     ...
17   }
18
19   Widget myLayoutWidget() {
20     return Column(
21       children: [
22         Row(
23           children: [
24             Icon(Icons.favorite),
25             Text('BEAMS'),
26           ],
27         ), // Row
28         Text('description...'),
29         Row(
30           children: [
31             Text('EXPLORE BEAMS'),
32             Icon(Icons.arrow_forward),
33           ],
34         ), // Row
35       ],
36     ); // Column
37   }
38

```

L'ensemble est enfermé dans un conteneur violet avec des bordures arrondies avec ce code, qui ajoute des styles et des éléments spécifiques à chaque rangée. Pour voir les nouveaux effets de mise en page, copiez et collez ce code dans votre projet Flutter.

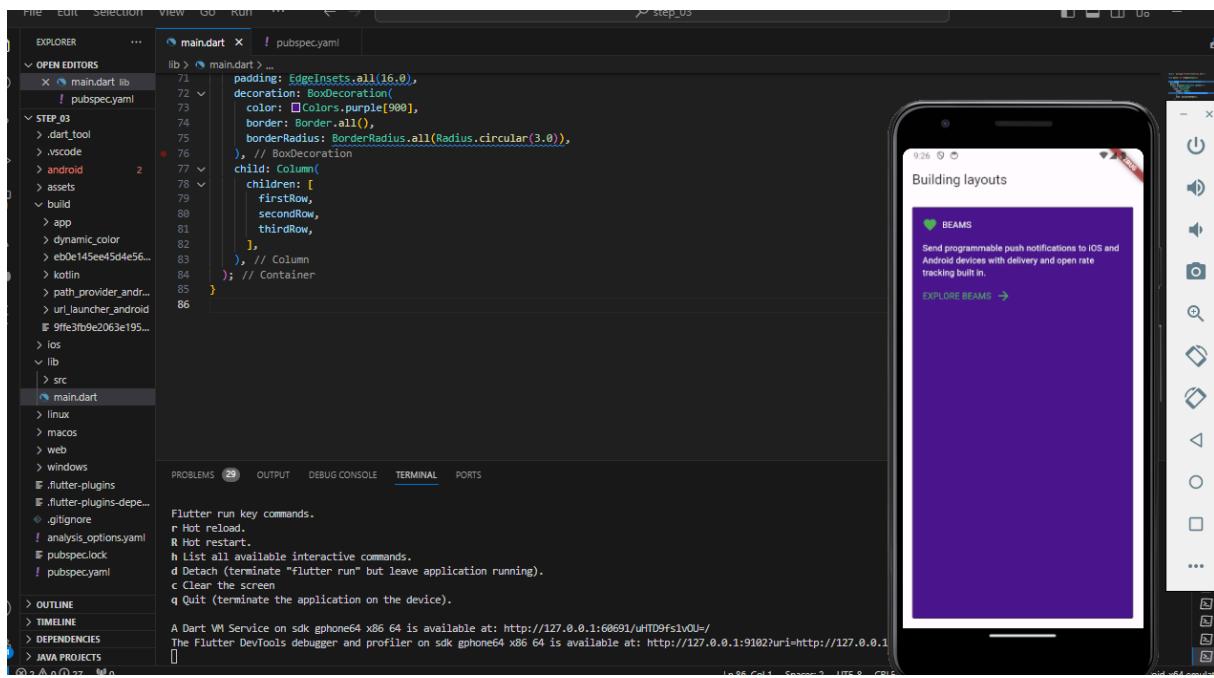
The screenshot shows the VS Code interface with the main.dart file open. The code now includes TextStyle and Padding widgets within the Row children. The iPhone X simulator on the right shows the updated UI with rounded corners and purple styling.

```

main.dart
60   lib > main.dart > ...
61     ...
62   Row(
63     children: [
64       Text(
65         'EXPLORE BEAMS',
66         style: TextStyle(
67           color: Colors.green,
68         ), // TextStyle
69       ), // Text
70       Padding(
71         padding: EdgeInsets.only(left: 8.0),
72         child: Icon(
73           Icons.arrow_forward,
74           color: Colors.green,
75         ), // Icon
76       ), // Padding
77     ],
78   ); // Row
79   ...
80   ...
81   ...
82   ...
83   ...
84   ...
85   ...
86   ...
87   ...
88   ...
89   ...
90   ...
91   ...
92 }
93

```

J'ai extrait chaque rangée dans des widgets différents dans ce code. La méthode myLayoutWidget() utilise ensuite ces widgets pour rendre le code plus modulaire et lisible.



```
lib > main.dart > pubspec.yaml
  71   padding: EdgeInsets.all(16.0),
  72   decoration: BoxDecoration(
  73     color: Colors.purple[900],
  74     border: Border.all(),
  75     borderRadius: BorderRadius.all(Radius.circular(3.0)),
  76   ), // BoxDecoration
  77   child: Column(
  78     children: [
  79       firstRow,
  80       secondRow,
  81       thirdRow,
  82     ],
  83   ), // Column
  84 } // Container
  85 }
  86 
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Flutter run key commands.

r Hot reload.

R Hot restart.

h List all available interactive commands.

d Detach (terminate "flutter run" but leave application running).

c Clear the screen

q Quit (terminate the application on the device).

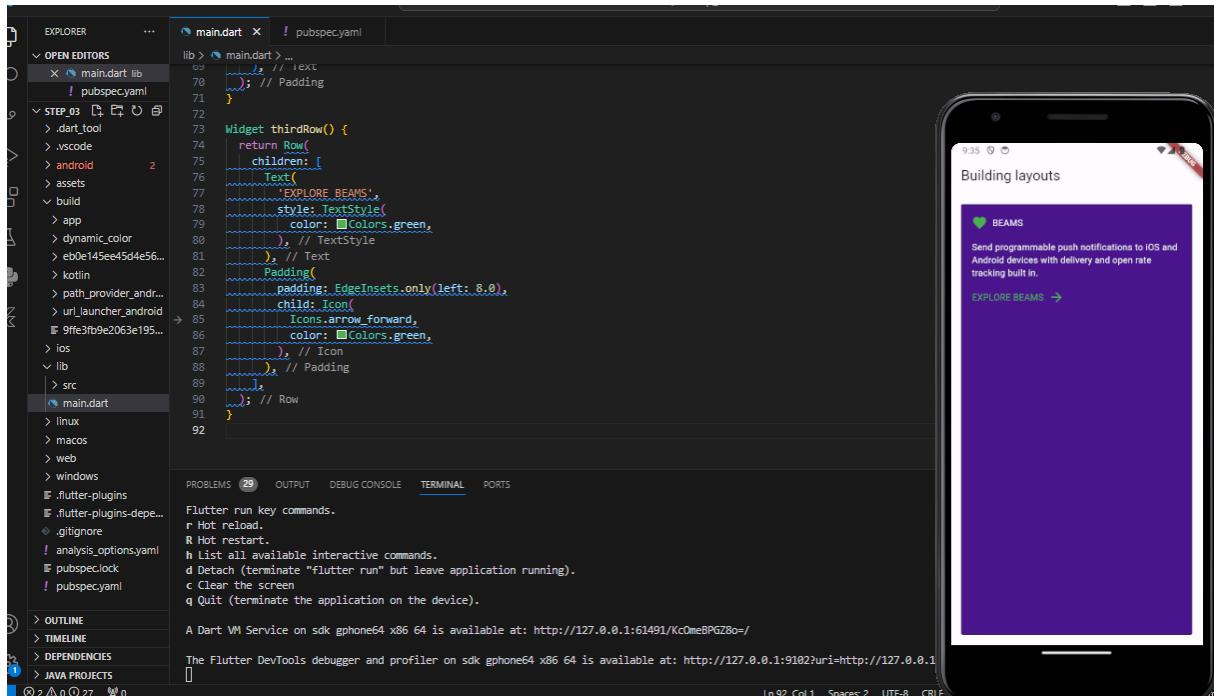
A Dart VM Service on sdk gphone64 x86 64 is available at: http://127.0.0.1:60691/uHTD9fs1VOU=

The Flutter DevTools debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:9102?uri=http://127.0.0.1:

J'ai changé les méthodes de création de chaque rangée en (firstRow(), secondRow(), thirdRow()). Cette méthode rend le code plus modulaire et facilite la lecture et la gestion des différentes parties de votre UI.

La création d'une colonne de trois rangées (firstRow, secondRow et thirdRow) complète le code dans cette section.

J'ai ajouté la classe FirstRow récemment créée et j'ai remplacé la fonction firstRow() par FirstRow() dans la méthode myLayoutWidget().



```
lib > main.dart > pubspec.yaml
  69   padding: EdgeInsets.only(left: 8.0),
  70   ); // Padding
  71 }
  72 
  73 Widget thirdRow() {
  74   return Row(
  75     children: [
  76       Text(
  77         'EXPLORE BEAMS',
  78         style: TextStyle(
  79           color: Colors.green,
  80         ), // TextStyle
  81       ),
  82       Padding(
  83         padding: EdgeInsets.only(left: 8.0),
  84         child: Icon(
  85           Icons.arrow_forward,
  86           color: Colors.green,
  87         ), // Icon
  88       ),
  89     ],
  90   ); // Row
  91 }
  92 
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Flutter run key commands.

r Hot reload.

R Hot restart.

h List all available interactive commands.

d Detach (terminate "flutter run" but leave application running).

c Clear the screen

q Quit (terminate the application on the device).

A Dart VM Service on sdk gphone64 x86 64 is available at: http://127.0.0.1:61491/KcOMeBPGZ8o=

The Flutter DevTools debugger and profiler on sdk gphone64 x86 64 is available at: http://127.0.0.1:9102?uri=http://127.0.0.1:

La fonction myLayoutWidget() produit un widget Container avec une colonne (Column) avec trois rangées (FirstRow, SecondRow et ThirdRow).

Les classes FirstRow, SecondRow et ThirdRow sont des widgets sans statut qui représentent chacune une rangée particulière dans la colonne. Chaque rangée a sa structure et son style spécifiés.

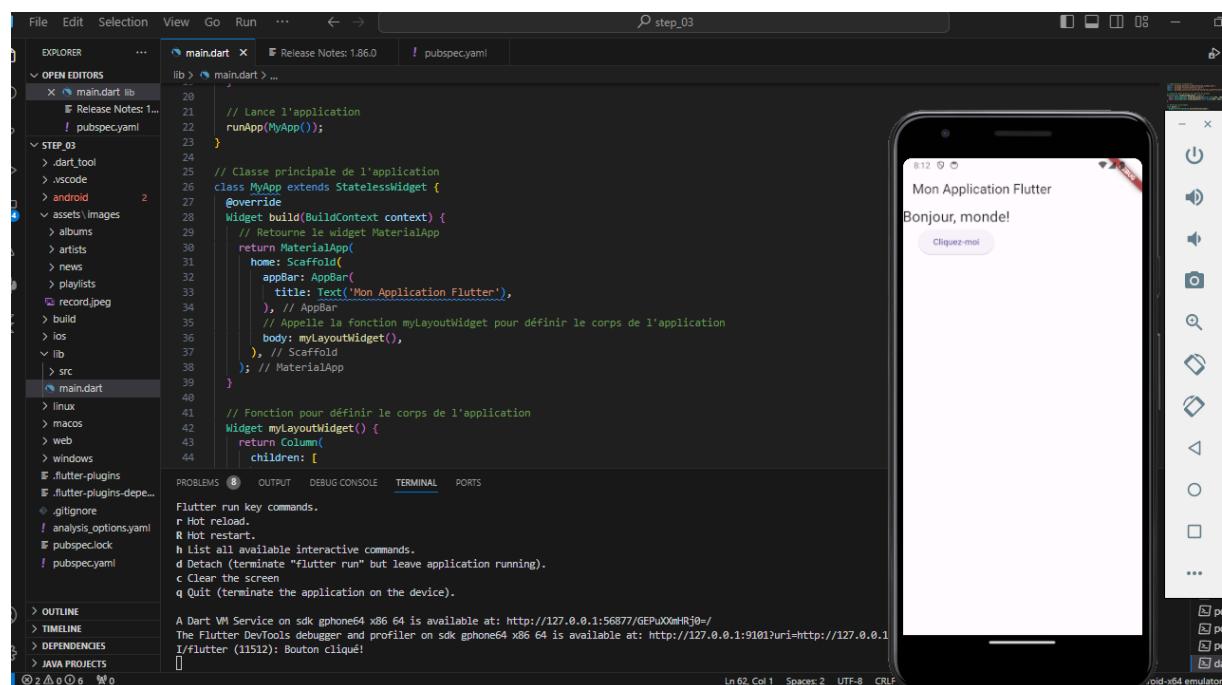
En somme, ce code crée une application Flutter de base avec une interface utilisateur simple composée d'une colonne avec trois rangées.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like main.dart, pubspec.yaml, and various build configurations.
- Code Editor:** Displays the main.dart code. The code defines a Row with three children: a Text widget for "EXPLORE BEAMS", a Padding widget containing an Icon, and another Padding widget containing an Icon. The Text and first Padding widget have specific styles (text color green, padding left 8.0).
- Terminal:** Shows Dart VM service and DevTools URLs.
- Output:** Shows build logs, including "A Dart VM Service on sdk_gphone64_x86_64 is available at: http://127.0.0.1:63883/mW8...".
- Flutter Preview:** A mobile phone icon displays the running Flutter application titled "Building layouts". The screen shows a purple background with a blue header bar and a white content area containing the text "EXPLORE BEAMS" and two green icons.

C. First steps with Flutter - Part 3: Responding to user input

Ce code Flutter crée une application pour les environnements de bureau avec une fenêtre personnalisable. L'interface utilisateur est créée par la classe `MyApp` avec une barre d'applications et une fonction `myLayoutWidget` pour organiser un texte de salutation et un bouton dans une colonne. Un message est imprimé dans la console lorsque le bouton est cliqué. Pour les applications de bureau, la taille de la fenêtre est ajustée automatiquement.



J'ai ajouté un nouveau widget appelé `MyWidget`, qui avait son propre état (`_MyWidgetState`) et une mise en page spécifique, ce qui a étendu l'application Flutter. Le widget `MyWidget` a été incorporé dans la structure principale de l'interface utilisateur de `MyApp`. Par conséquent, notre application comprend désormais une hiérarchie d'interface utilisateur plus complexe avec des éléments réutilisables pour une gestion efficace de l'état et de la mise en page.

The screenshot shows the VS Code interface with the main.dart file open in the editor. The code defines a MyWidget class that extends StatelessWidget. It contains a build method that returns a Container widget with a Text child containing 'Contenu de MyWidget'. The main function initializes the application and runs it on a desktop platform. The right side of the screen shows an iPhone X simulator running the application. The app's title is 'Mon Application Flutter' and its content is 'Bonjour, monde!'. A button labeled 'Cliquez-moi' has the placeholder text 'Contenu de MyWidget'.

```

lib > main.dart ...
53 Class _MyWidgetState extends State<MyWidget> {
54   @override
55   Widget build(BuildContext context) {
56     // Remplacez le retour par le layout souhaité pour MyWidget
57     return Container(
58       // Ajoutez votre mise en page spécifique pour MyWidget ici
59       child: Text('Contenu de MyWidget'),
60     ); // Container
61   }
62 }
63
64 // Fonction principale
65 void main() {
66   WidgetsFlutterBinding.ensureInitialized();
67
68   // Si l'application est exécutée sur une plate-forme de bureau
69   if (UniversalPlatform.isDesktop) {
70     setDesktopWindow();
71   }
72
73   // Lance l'application
74   runApp(MyApp());
75 }
76
77
78
79
80
81
82

```

Un titre, un texte initial, un bouton et un champ de texte sont présents dans l'interface utilisateur de l'application Flutter. Lorsque le bouton est pressé, le texte est mis à jour dans le champ de texte et le texte est mis à jour en fonction de la saisie de l'utilisateur.

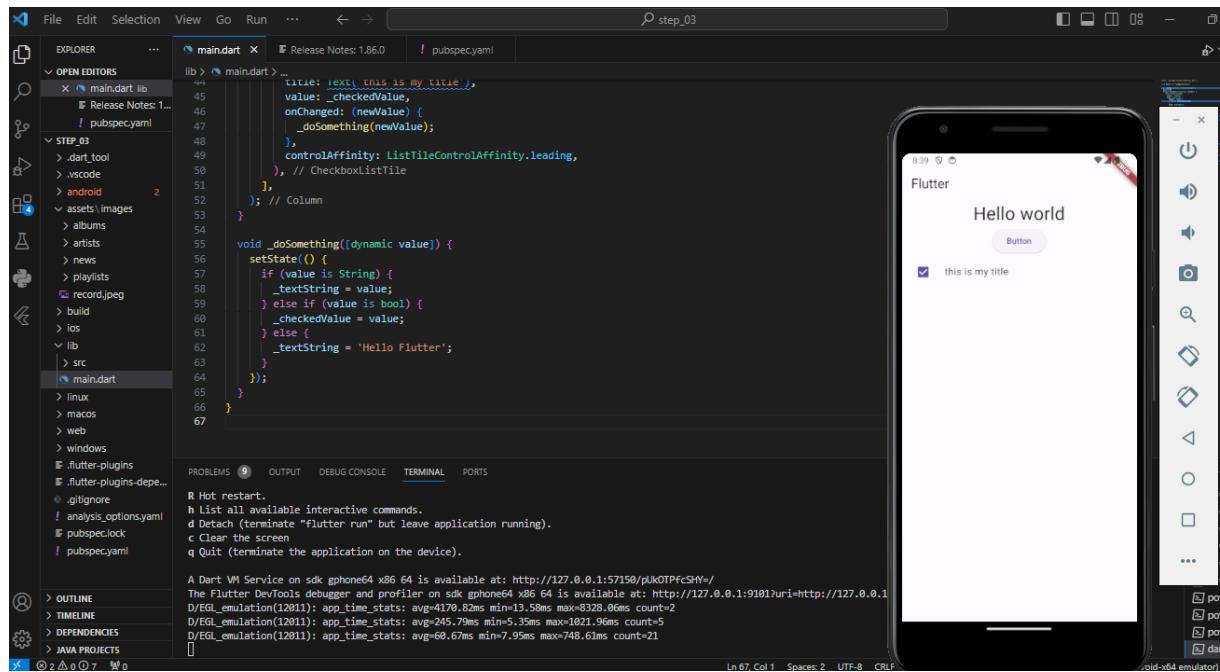
The screenshot shows the VS Code interface with the main.dart file open in the editor. The code now includes a TextEditingController variable named _textString. The build method creates a Column with a Text widget and an ElevatedButton. The button's onPressed callback calls a _doSomething() function. The TextField's onChanged callback also calls _doSomething() with the new text value. The right side of the screen shows an iPhone X simulator running the application. The app's title is 'Flutter' and its content is 'Hello world'. A button labeled 'Button' is present. The text field contains the placeholder 'Hello world'.

```

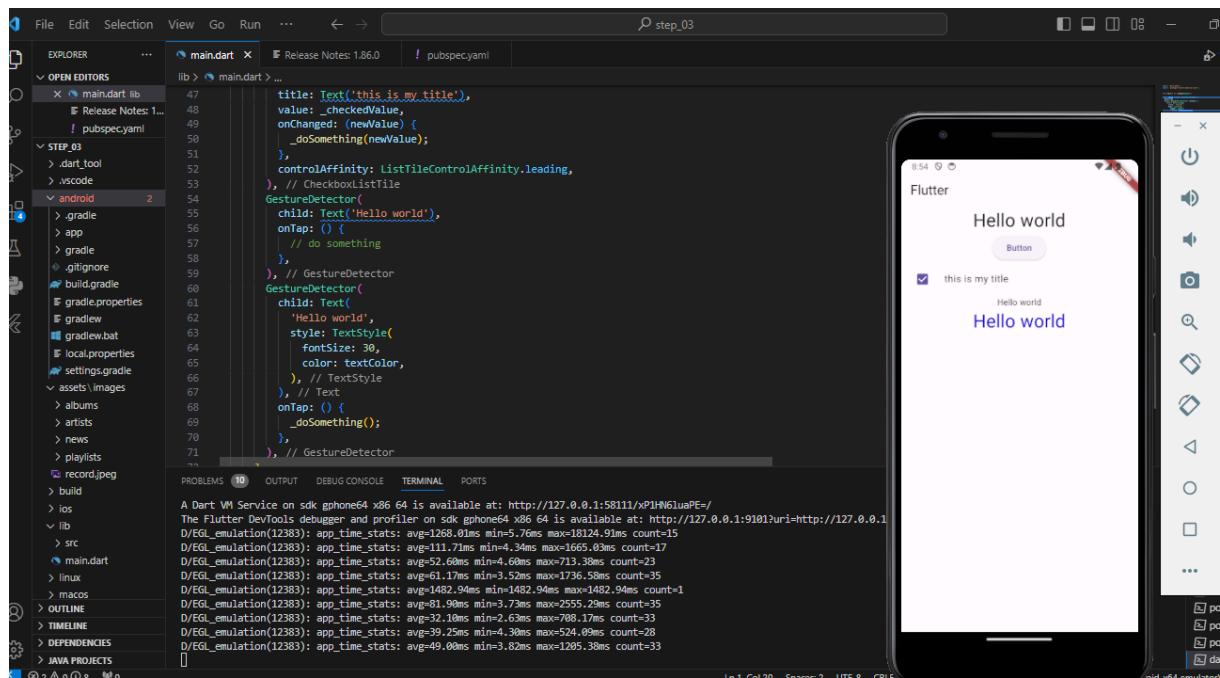
lib > main.dart ...
26   String _textString = 'Hello world';
27
28   @override
29   Widget build(BuildContext context) {
30     return Column(
31       children: [
32         Text(
33           _textString,
34           style: TextStyle(fontSize: 30),
35         ), // Text
36         ElevatedButton(
37           onPressed: () {
38             _doSomething();
39           },
40           child: Text('Button'),
41         ), // ElevatedButton
42         TextField(
43           onChanged: (text) {
44             _doSomething(text);
45           },
46         ), // TextField
47       ],
48     ); // Column
49   } // build
50
51
52
53
54
55

```

J'ai ajouté le CheckboxListTile dans _MyWidgetState avec la fonctionnalité correspondante. La méthode _doSomething a été mise à jour pour gérer la mise à jour du texte et la valeur de la case à cocher en fonction du type de valeur fourni.



Cette application Flutter affiche du texte, un bouton, une case à cocher et réagit à certaines interactions de l'utilisateur en modifiant aléatoirement le texte, la valeur de la case à cocher et la couleur du texte. En réponse à ces interactions, la fonction principale _doSomething met à jour l'état de l'application.



Avec ce prochain code , il y a une application à deux écrans (deux pages) :

- La première page (`MyWidget`) affiche du texte, un bouton élevé, une case à cocher, des gestes (taps) sur du texte, un changement aléatoire de la couleur du texte, et un bouton pour naviguer vers la deuxième page.
- La deuxième page (`SecondScreen`) affiche un bouton élevé qui, lorsqu'il est appuyé, ramène l'utilisateur à la première page.
- La navigation entre les pages est réalisée à l'aide de `Navigator.push` et `Navigator.pop` .

The screenshot shows the Android Studio interface with the main.dart file open in the editor. The code defines a StatelessWidget named SecondScreen. It contains an ElevatedButton labeled "Go back to first screen", a Text widget with the placeholder "this is my title", and a Text widget with the placeholder "Hello world". The application is running on an iPhone X simulator, which displays the first screen of the Flutter app. The screen shows the text "Hello world", a button labeled "Go to second screen", and a checkbox labeled "this is my title". The status bar indicates the time is 9:02.

```

lib> <main.dart> ...
105 > CLASS SecondScreen extends StatelessWidget {
106   @override
107   Widget build(BuildContext context) {
108     return Scaffold(
109       appBar: AppBar(title: Text("Second screen")),
110       body: Center(
111         child: ElevatedButton(
112           onPressed: () {
113             _goBackToFirstScreen(context);
114           },
115           child: Text(
116             'Go back to first screen',
117             style: TextStyle(fontSize: 24),
118           ),
119           // Text
120           ), // ElevatedButton
121           ), // Center
122         ); // Scaffold
123       }
124       void _goBackToFirstScreen(BuildContext context) {
125         Navigator.pop(context);
126       }
127     }
128

```

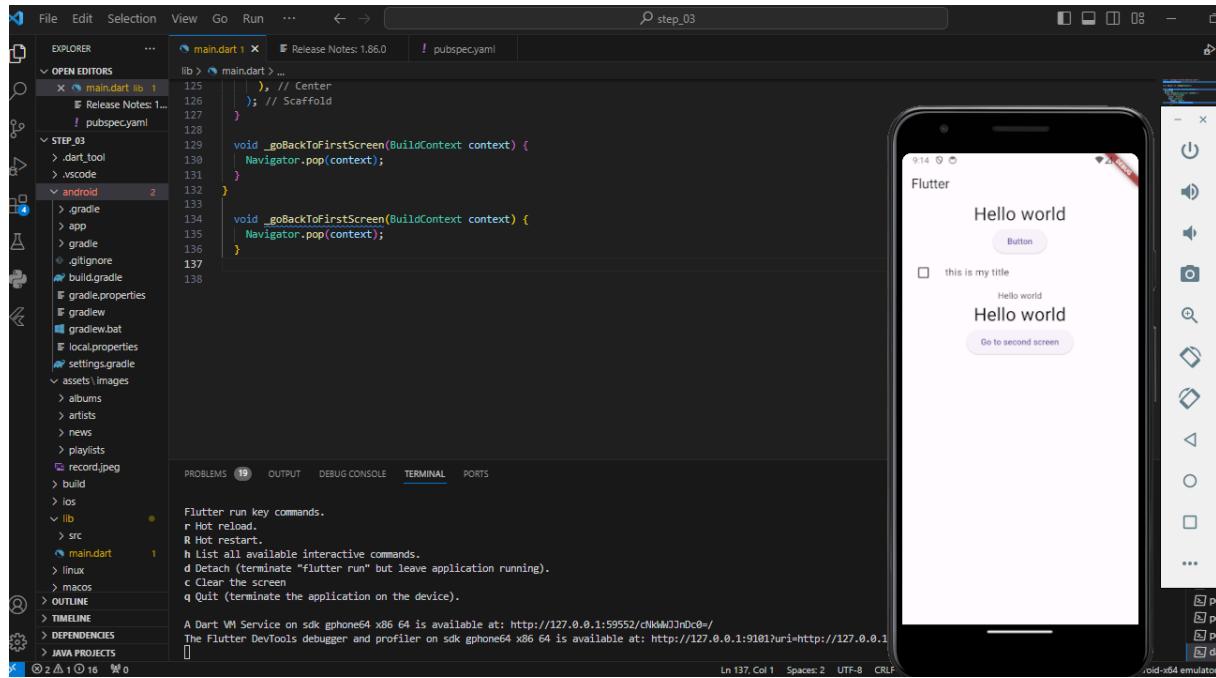
The screenshot shows the Android Studio interface with the main.dart file open in the editor. The code is identical to the previous screenshot, defining the SecondScreen StatelessWidget. The application is now running on an iPhone X simulator, showing the second screen of the Flutter app. This screen has a single button labeled "Go back to first screen". The status bar indicates the time is 9:03.

```

lib> <main.dart> ...
105 > CLASS SecondScreen extends StatelessWidget {
106   @override
107   Widget build(BuildContext context) {
108     return Scaffold(
109       appBar: AppBar(title: Text("Second screen")),
110       body: Center(
111         child: ElevatedButton(
112           onPressed: () {
113             _goBackToFirstScreen(context);
114           },
115           child: Text(
116             'Go back to first screen',
117             style: TextStyle(fontSize: 24),
118           ),
119           // Text
120           ), // ElevatedButton
121           ), // Center
122         ); // Scaffold
123       }
124       void _goBackToFirstScreen(BuildContext context) {
125         Navigator.pop(context);
126       }
127     }
128

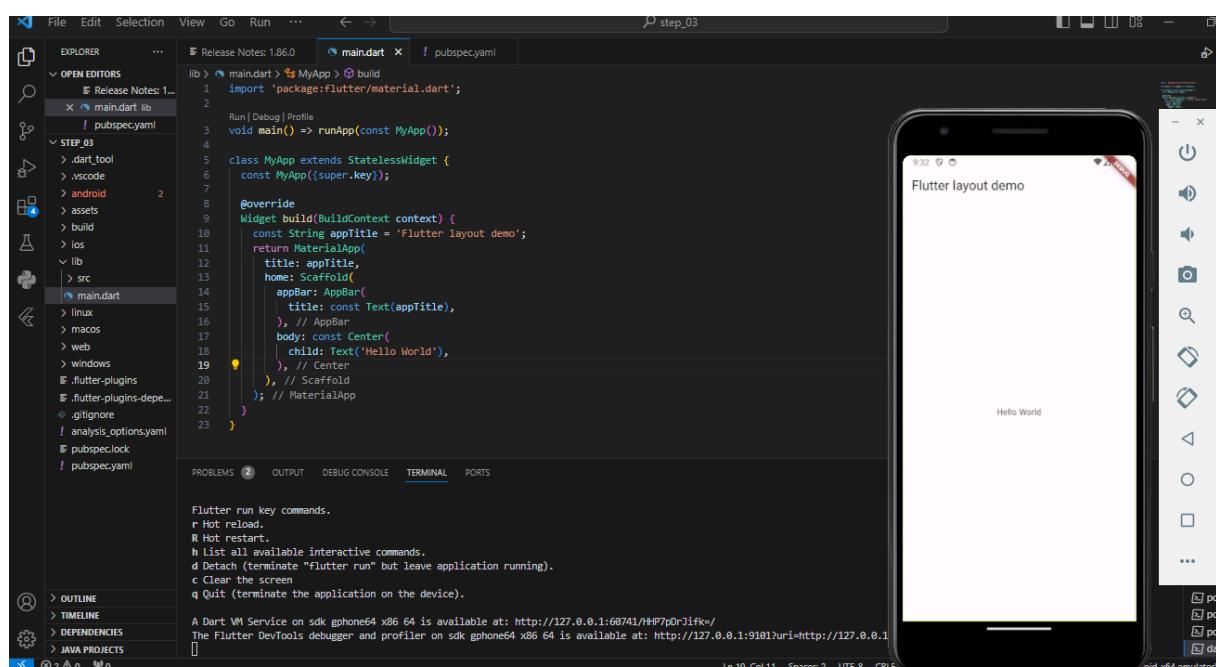
```

Ce prochain code Flutter crée une deuxième page (`SecondScreen`) qui prend un texte en paramètre lors de son initialisation. Cette page affiche un bouton élevé, et lorsque ce bouton est pressé, elle utilise `Navigator.pop` pour retourner à la première page. La classe `SecondScreen` a été modifiée pour déclarer que le paramètre `key` est requis (`required`).

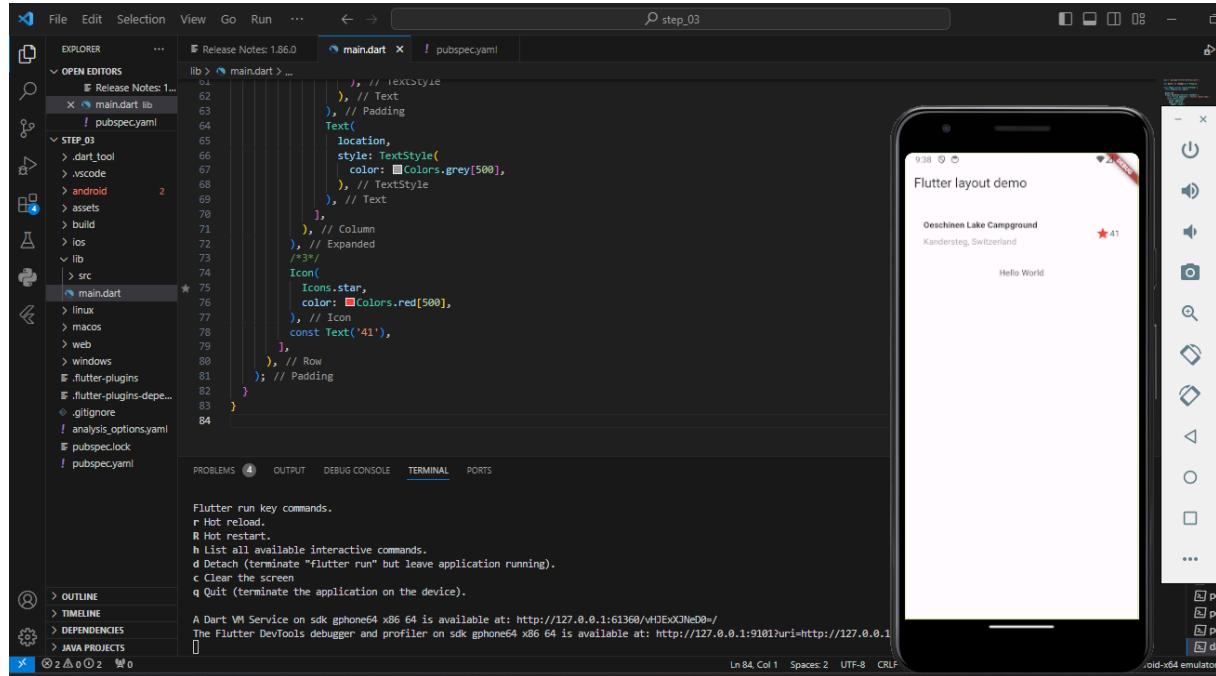


D. Build a Flutter layout

Ce code crée une application Flutter avec une fenêtre contenant une barre d'applications ayant le titre "Flutter layout demo" et un corps affichant le texte "Hello World" centré. L'output visuel sera une interface utilisateur simple avec ces éléments.

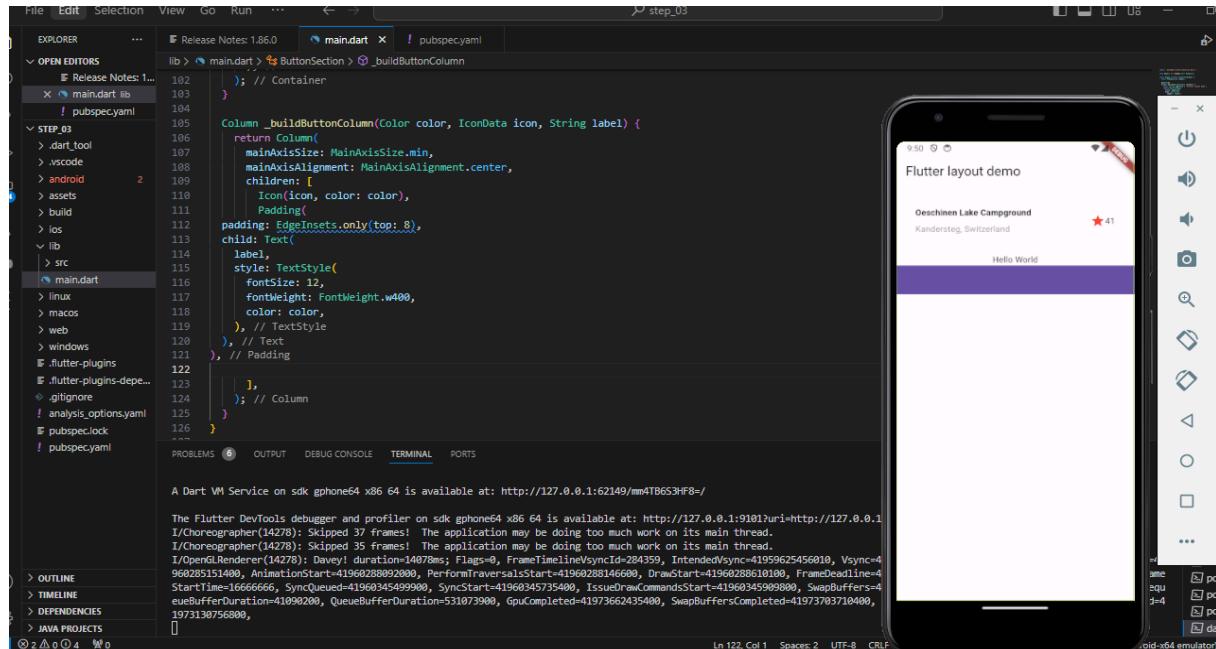


Avant le texte "Hello World", l'application affiche maintenant une section de titre avec le nom "Oeschinen Lake Campground" et l'emplacement "Kandersteg, Switzerland". La section de titre contient également le texte "41" et une icône d'étoile rouge.

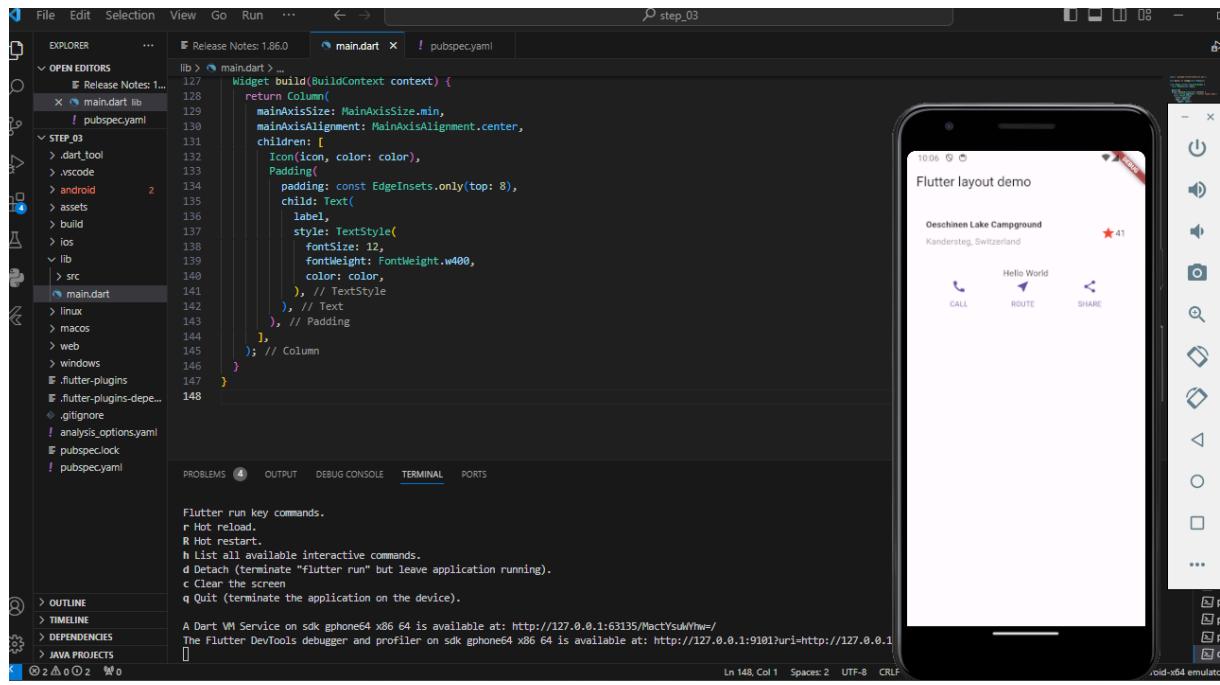


L'application affiche maintenant une section de boutons sous le texte "Hello World". Trois boutons, "CALL", "ROUTE" et "SHARE", sont présents dans cette section de boutons.

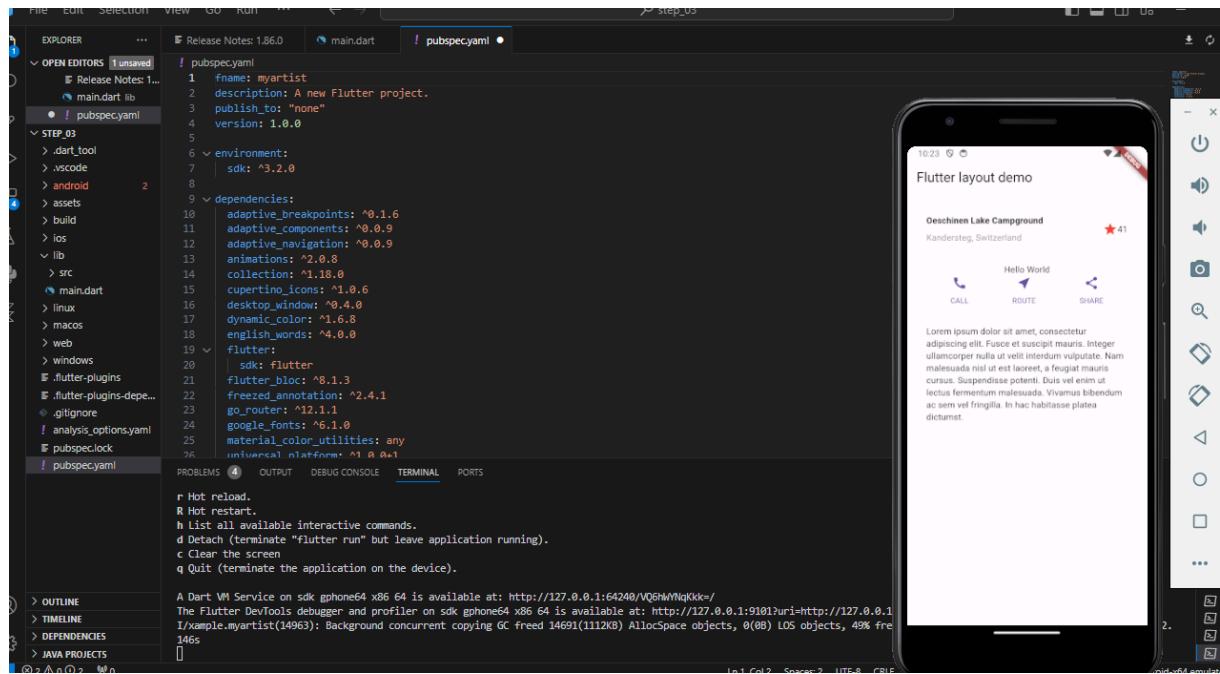
Avec cette modification, la classe `ButtonSection` utilise désormais la classe `ButtonWithText` récemment créée pour créer des boutons contenant des icônes et du texte. Après avoir intégré ces modifications, vous pouvez lancer votre application.



La structure générale de l'application est conservée avec ces modifications, car la classe `ButtonSection` utilise maintenant un `SizedBox` pour englober les boutons.



L'interface utilisateur a maintenant une nouvelle section de texte (`TextSection`) grâce à ces modifications.



Cette modification a ajouté une nouvelle section d'image à l'interface utilisateur.

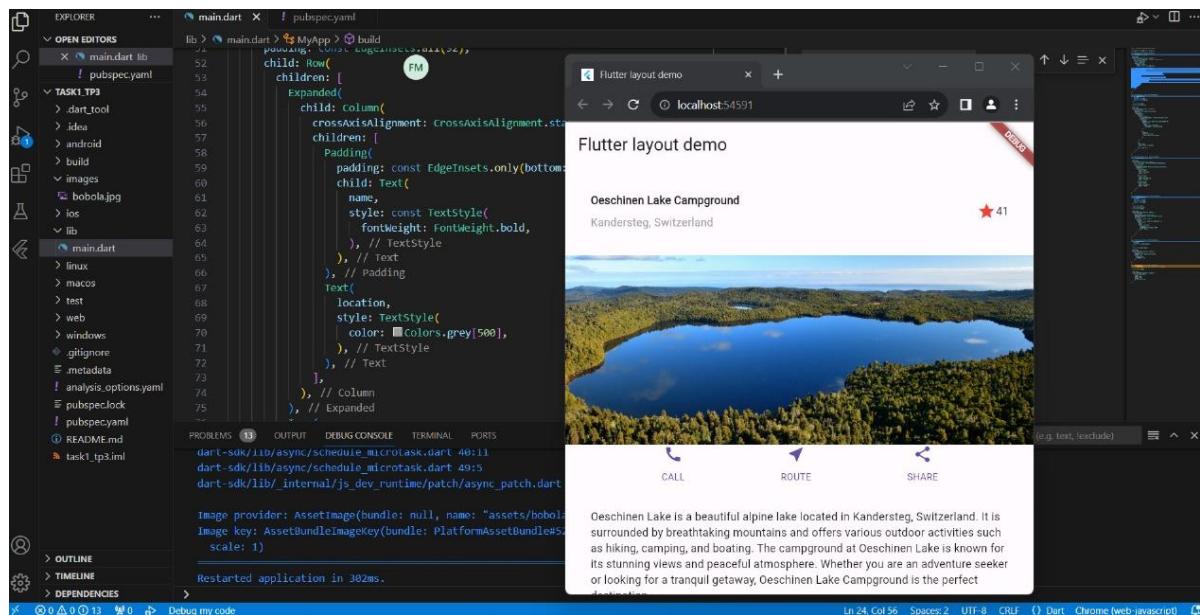


Table de Matière

| | |
|---|----|
| A.Container widget..... | 1 |
| 1. Importation des Packages :..... | 1 |
| 2. Fonction main() : | 1 |
| 3. Fonction runApp() :..... | 1 |
| B. Text widget | 3 |
| 1. Importation des Packages :..... | 3 |
| 2. Fonction main() : | 3 |
| 3. Classe MyApp : | 3 |
| 4. Fonction myWidget() :..... | 4 |
| 5. Evolution du Code_1 :..... | 5 |
| 6. Evolution du Code_2 :..... | 6 |
| 7. Évolution du Code_3 :..... | 6 |
| C. First steps with Flutter - Part 3: Responding to user input..... | 19 |
| D.Build a Flutter layout..... | 23 |