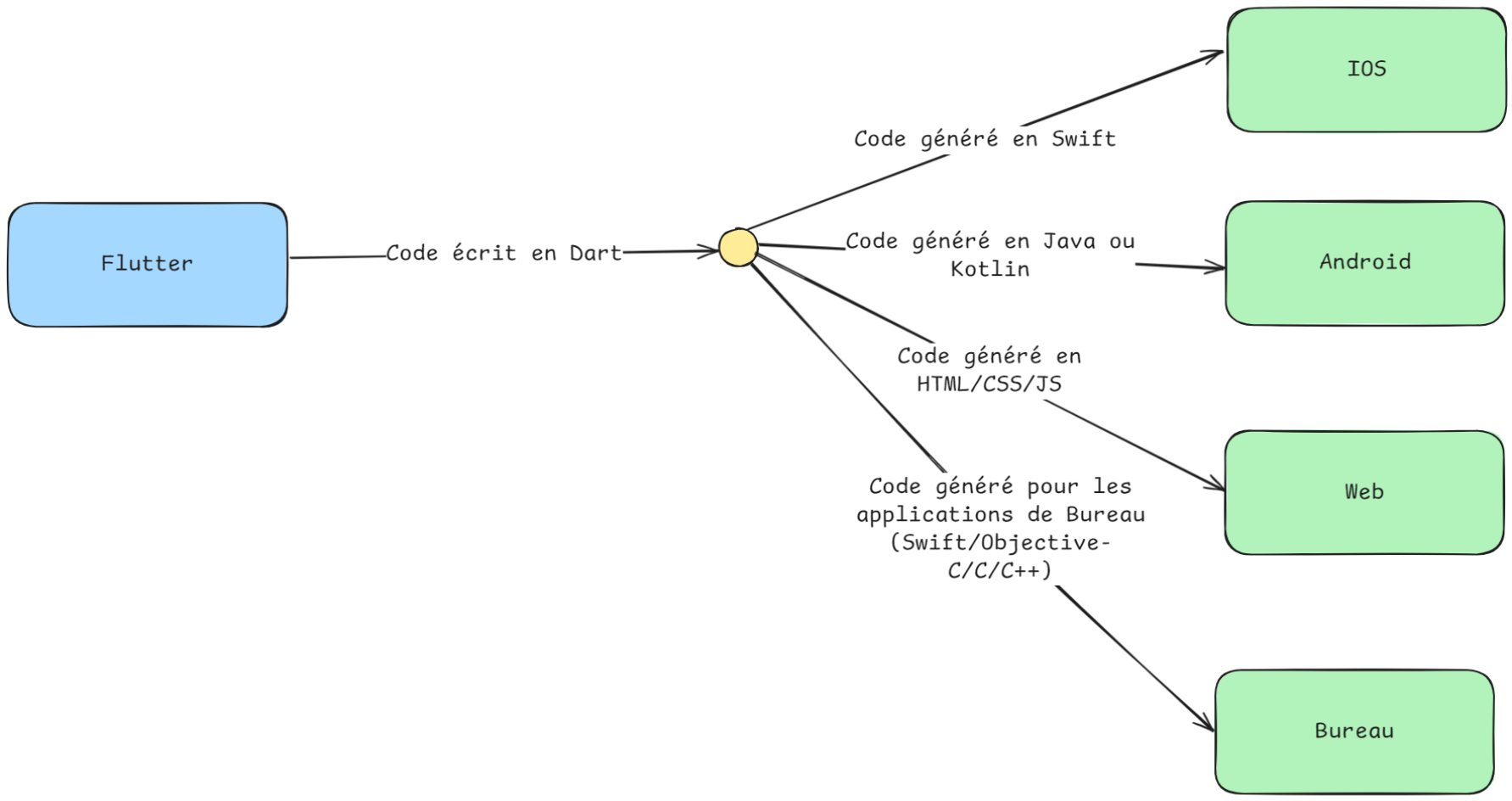
DEVELOPPEMENT D'APPLICATION MOBILE

Chapitre 3 : Découverte de Flutter

Sommaire

Introduction

Flutter est un Framework open source qui permet de créer des applications multiplateformes compilées nativement avec une unique base de code.



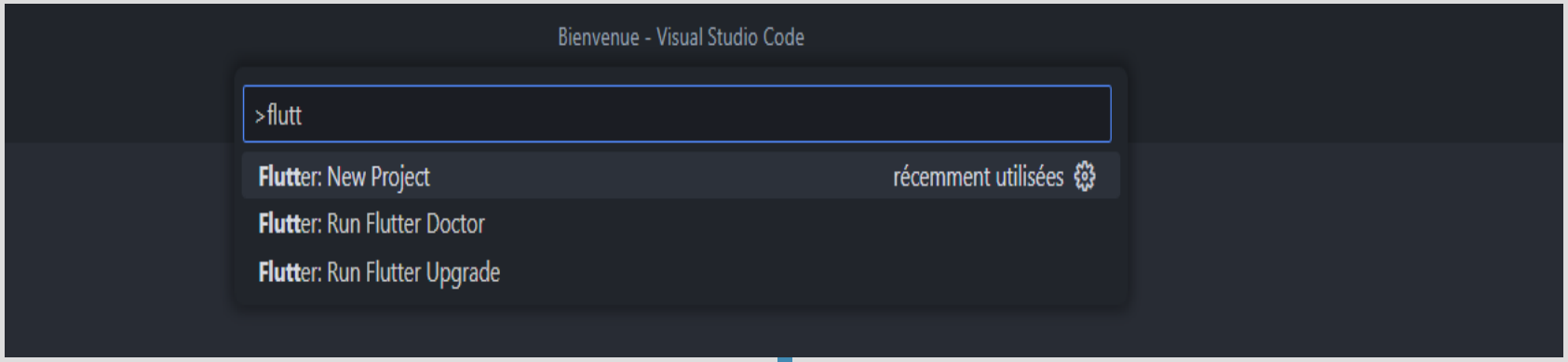
Particularités

- une seule base de code,
- performance native,
- widgets personnalisables,
- hot reload (changement visible instantanément),
- communauté active,
- écosystème riche,
- support multiplateforme,
- design uniforme sur toutes les plateformes,
- possibilité de développer des applications embarquées.

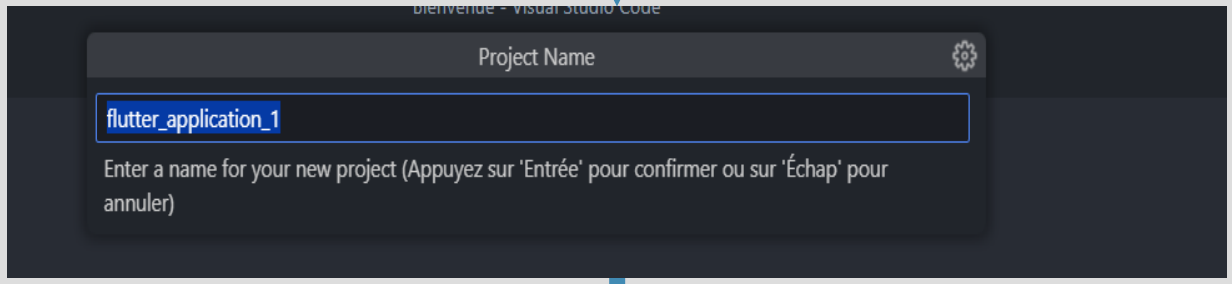
Pour plus d'informations visiter: Site Web de Flutter, Documentation de Flutter.

Création d'un projet Flutter avec VSCode

Dans VSCode exécuter $Ctrl + Maj + P$ sur Windows ou $Command + Maj + P$ sur Mac.



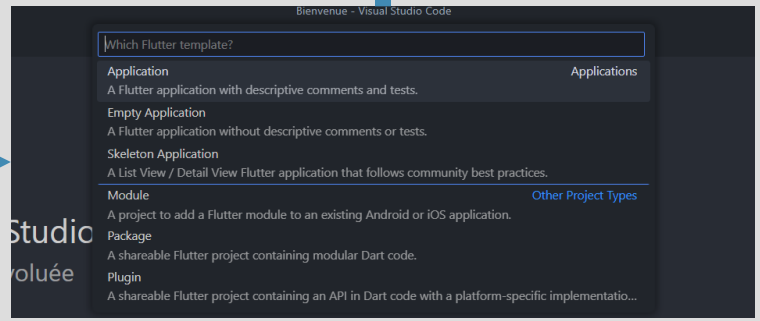
1



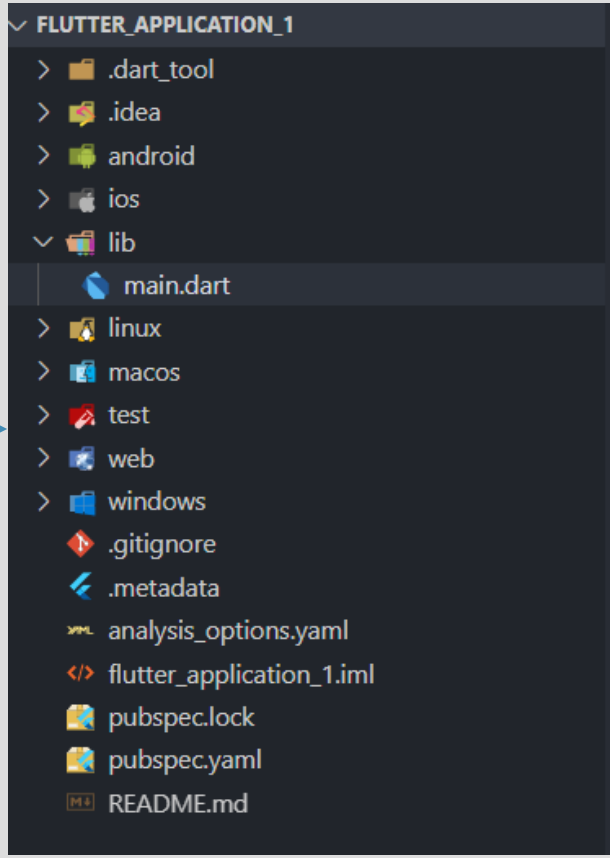
2

Choix du dossier de sauvegarde du projet

3

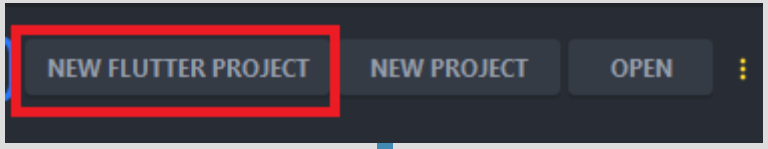


4

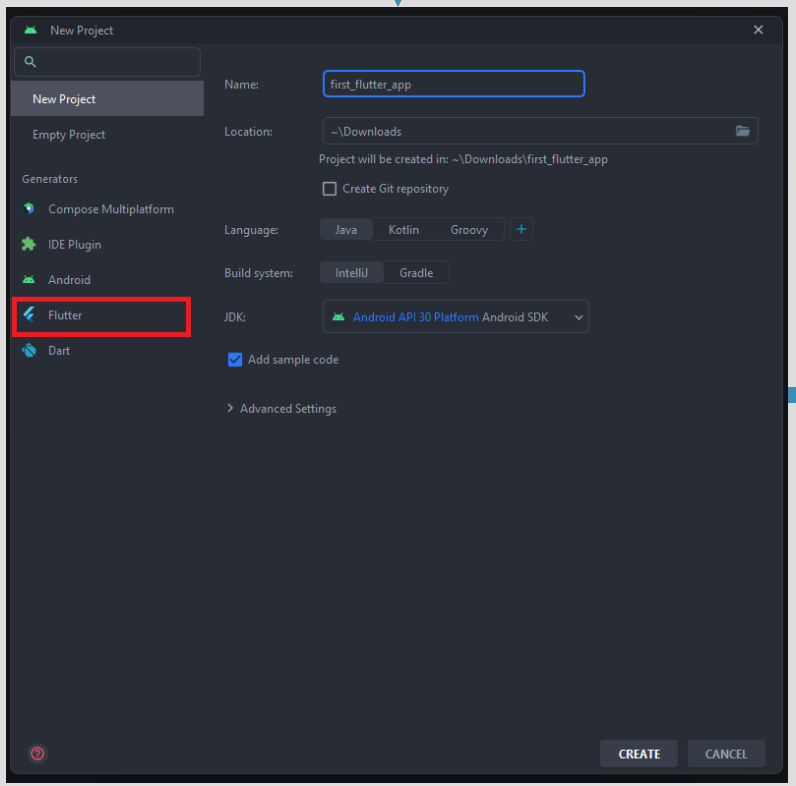


Création d'un projet Flutter avec Android Studio

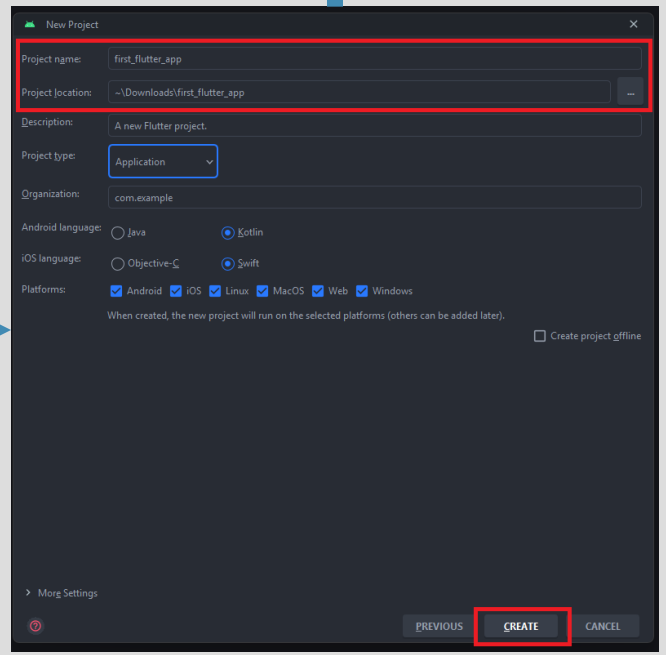
Ouvrir Android Studio et suivre les étapes ci-dessous:



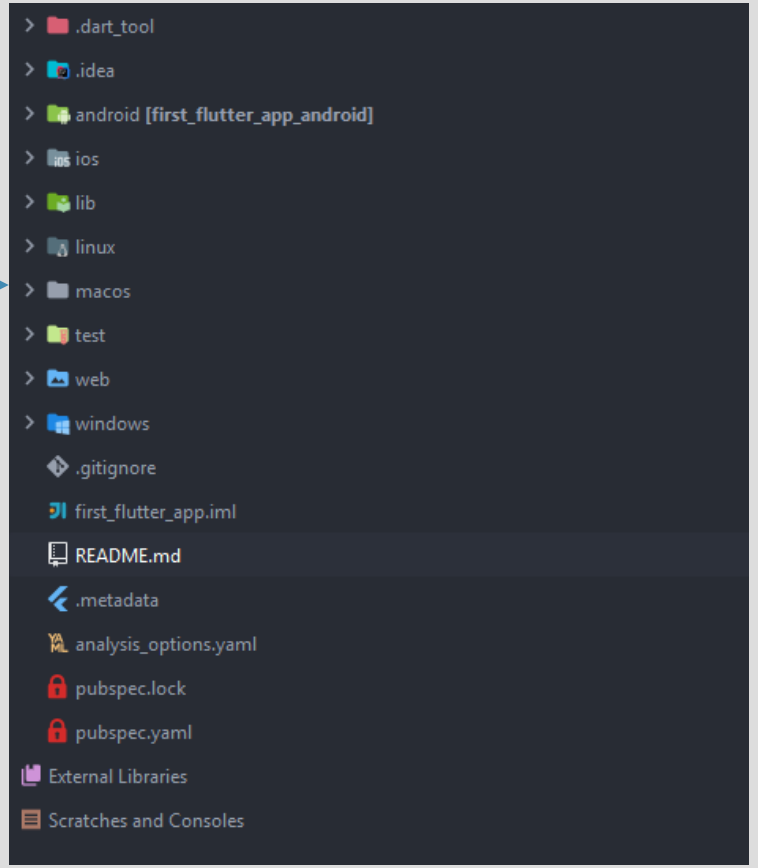
1



2



3

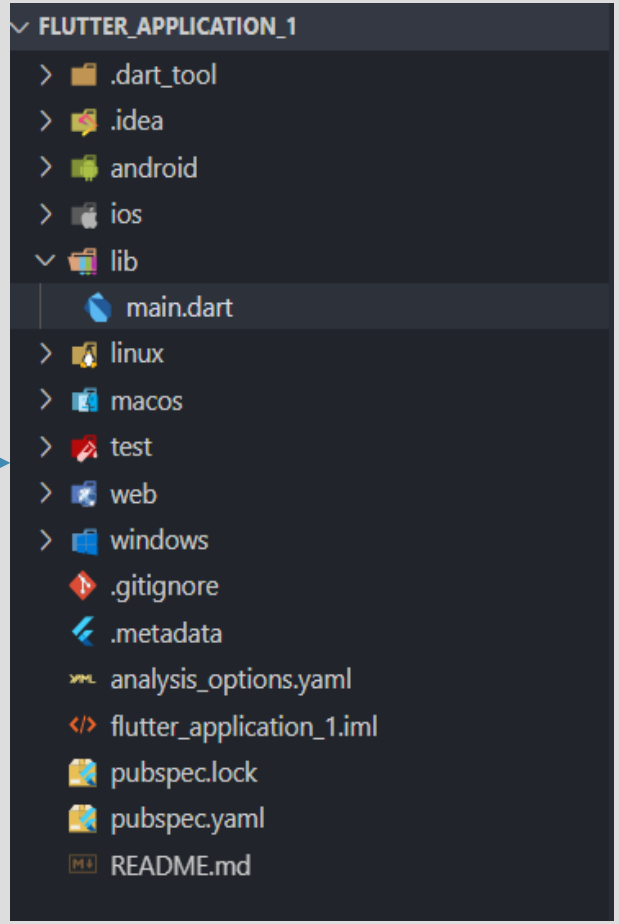


Création d'un projet Flutter en ligne de commande

Ouvrir le terminal et saisir la commande *flutter create nom_du_projet*.

```
Creating project first_app...
Resolving dependencies in `first_app`...
Downloading packages...
Got dependencies in `first_app`.
Wrote 129 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev

In order to run your application, type:

  $ cd first_app
  $ flutter run

Your application code is in first_app\lib\main.dart.
```

Fichiers du projet



Par défaut, un nouveau projet créé contient le code natif de toutes les plateformes supportées Android, iOS, Linux, MacOS, Windows et Web.

Il existe des commandes pour désactiver certaines plateformes si besoin avec *flutter config --no-enable-xxx*

Pour plus d'informations visiter Désactiver des plateformes.

Arborescence d'un projet Flutter

- .dart_tool → Contient des éléments utiles pour l'analyseur du code Dart
- .idea → Contient des informations utiles pour l'IDE Android Studio
- android → Code natif de l'application Mobile Android
- ios → Code natif de l'application Mobile iOS
- lib → **Dossier du code Flutter c'est le principal dossier de travail**
- linux → Code natif de l'application Desktop Linux
- macos → Code natif de l'application Desktop MacOs
- test → Dossier des tests de l'application
- web → Code natif de l'application Web
- windows → Code natif de l'application Desktop Windows
- .gitignore → Contient la liste des fichiers que Git doit ignorer
- .metadata → Contient les propriétés du projet Flutter
- analysis_options.yaml → Contient les paramètres de l'analyseur du code Dart
- first_app.iml → Informations sur le projet, utiles pour les IDE comme Android Studio
- pubspec.lock / pubspec.yaml → Gestion des propriétés et modules du projet comme la description et les modules
- README.md → Fichier de description du projet

StatefulWidget vs StatelessWidget

Tout élément (bouton, images, texte…) de l'interface utilisateur dans Flutter est un Widget.

Les Widgets sont classés en deux catégories principales : **StatelessWidget** et **StatefulWidget**.

StatefulWidget

Widget qui possède un état mutable. Il peut être reconstruit en fonction des valeurs de variables. Ils peuvent être considérés comme Widgets dynamiques.

- Mutable: peut être redessiné.
- Idéal pour les éléments réactifs comme le compteur, les cases à cocher.

VS

StatelessWidget

Widget sans état, ils restent identiques après création.
Ils peuvent être considérés comme Widgets statiques.

- Immuable: sans état réactif.
- Ne réagit pas aux événements qui modifient l'interface utilisateur comme un texte.

Il existe d'autres types de widgets comme:

- les widgets de mise en page (LayoutWidgets): colonnes, lignes…
- les widgets interactifs comme les boutons,…
- les widgets visuels comme les images…
- les widgets de style et de décoration comme l'alignement, la marge….

Pour plus d'informations voir Catalogue des widgets.

Corps d’un écran Flutter

```dart
// Squelette d'un widget Stateless
import 'package:flutter/material.dart';

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: Scaffold(
        body: Center(
          child: Text('Hello World!'),
        ),
      ),
    );
  }
}
```

VS

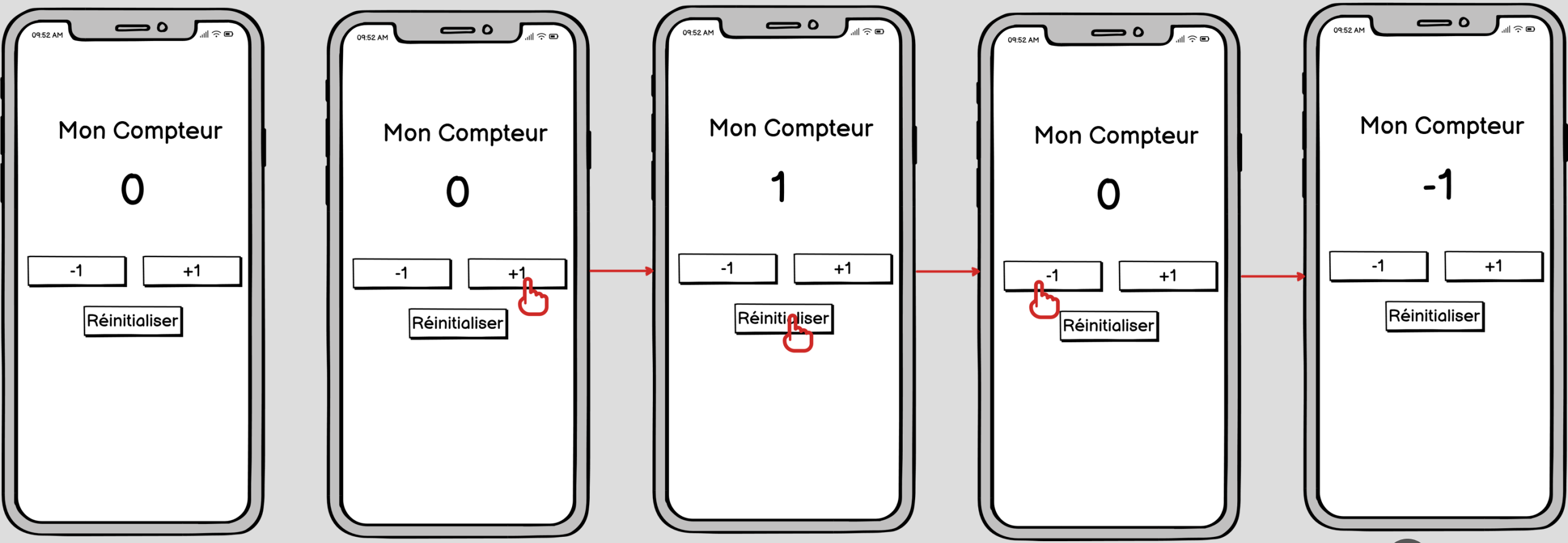
```dart
// Squelette d'un widget Stateful
import 'package:flutter/material.dart';

class MainApp extends StatefulWidget {
  @override
  const MainApp({super.key});

  @override
  State<MainApp> createState() => _MainAppState();
}

class _MainAppState extends State<MainApp> {
  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: Scaffold(
        body: Center(
          child: Text('Hello World!'),
        ),
      ),
    );
  }
}
```

Le widget *Scaffold* est le widget de base qui permet de structurer l’affichage d’un écran.

TP 2: Compteur manuel

Créer une application de compteur manuel avec les fonctionnalités suivantes:

- Un bouton $+1$ pour ajouter 1 au compteur.
- Un bouton -1 pour retirer 1 au compteur.
- Un bouton $Réinitialiser$ pour mettre le compteur à 0.



FIN DU CHAPITRE 3