
Atelier 1 Flutter : Installation et mise en place



Introduction

Flutter est le SDK mobile de Google permettant de créer des applications natives iOS et Android, de bureau (Windows, Linux, macOS) et Web à partir d'une seule base de code. Lors de la création d'applications avec Flutter, tout se tourne vers les widgets – les blocs avec lesquels les applications Flutter sont construites. Ce sont des éléments structurels livrés avec un ensemble de fonctionnalités spécifiques à la conception matérielle et de nouveaux widgets peuvent également être composés à partir de widgets existants. Le processus de composition de widgets ensemble est appelé composition. L'interface utilisateur de l'application est composée de nombreux widgets simples, chacun gérant une tâche particulière. C'est la raison pour laquelle les développeurs Flutter ont tendance à considérer leur application Flutter comme un arbre de widgets.

Android Studio

Télécharger :

https://developer.android.com/studio?utm_source=android-studio&hl=fr

Plate-forme	Package Android Studio	Taille	Somme de contrôle SHA256
Windows (64 bits)	android-studio-2024.1.1.11-windows.exe Recommandé	1,2 Go	eaf9fcec291e4be5b0b2f2fdcef15bfcc60df7303243c9ec0b00cd2cb7b37a72

Pour créer des simulateurs Android, pour tester vos applications, rendez-vous à la page d'ouverture d'Android studio, où vous allez pouvoir cliquer sur More Actions.

Dans un premier temps, cliquez sur “SDK manager”, pour finir de paramétrer votre simulateur et pouvoir l'utiliser sans devoir toujours passer par Android Studio :

Sélectionnez une version du système d'exploitation (idéalement la dernière) ;

Dans SDK Tools, vérifiez que les éléments suivants sont cochés :

Android SDK Build-Tools ;

Android SDK Command-line Tools ;

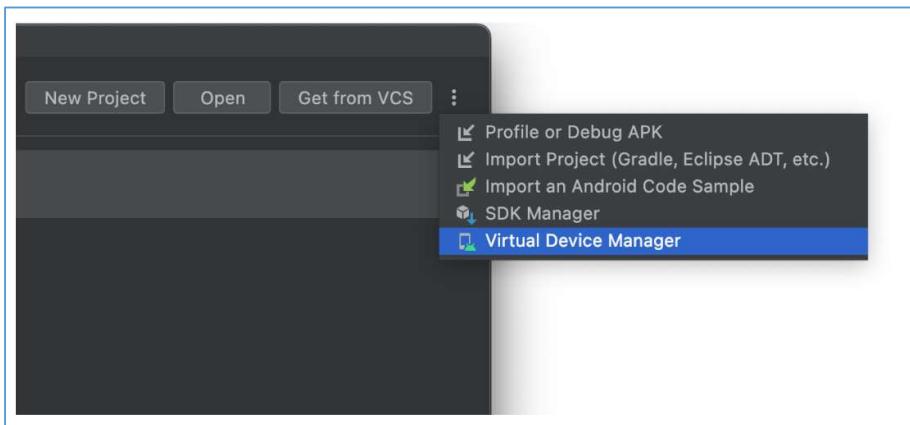
Android Emulator ;

Android SDK Platform Tools.

Cliquez sur “ok” pour installer

Pour créer un simulateur, retournez dans le menu More Actions et cliquez sur “Virtual Device manager”. Vous n'avez alors plus qu'à choisir le type d'appareil que vous souhaitez créer ainsi que la version d'Android sur laquelle il va fonctionne. Une fois cela fait, l'appareil va être installé et vous pourrez l'utiliser depuis votre éditeur de code, pour tester votre application.

Un appareil virtuel Android (AVD, Android Virtual Device) est une configuration qui définit les caractéristiques d'un téléphone ou d'une tablette Android, ou d'un appareil Wear OS, Android TV ou Automotive OS, que vous souhaitez simuler dans Android Emulator. Le Gestionnaire d'appareils est un outil que vous pouvez lancer depuis Android Studio et qui vous aide à créer et à gérer des AVD.



Après avoir ouvert un projet, sélectionnez View > Tool Windows > Device Manager (Afficher > Outils Windows > Gestionnaire d'appareils) dans la barre de menu principale, puis cliquez sur Create device (Créer un appareil).

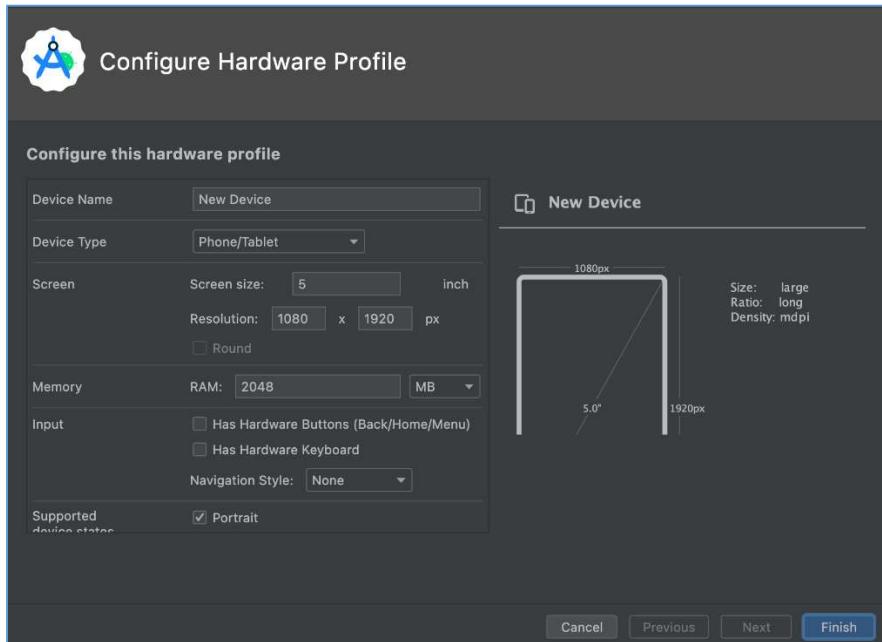
Le Gestionnaire d'appareils fournit des profils matériels prédéfinis pour les appareils courants, afin que vous puissiez les ajouter facilement à vos définitions AVD. Si vous devez définir un autre appareil, vous pouvez créer un profil matériel.

Vous pouvez définir un nouveau profil matériel depuis le début ou copier un profil matériel comme point de départ. Les profils matériels pré chargés ne sont pas modifiables.

Pour créer un profil matériel depuis le début :

Dans la fenêtre Select Hardware (Sélectionner le matériel), cliquez sur New Hardware Profile (Nouveau profil matériel).

Dans la fenêtre Configure Hardware Profile (Configurer le profil matériel), modifiez les propriétés du profil matériel selon vos besoins.



Remarque :

On peut choisir l'alternative de physical device à la place d'un émulateur.

Référence : <https://developer.android.com/codelabs/basic-android-kotlin-compose-connect-device#2>

Pour permettre à Android Studio de communiquer avec votre appareil Android, vous devez activer le débogage USB dans les paramètres des options du développeur de l'appareil.

Pour afficher les options de développeur et activer le débogage USB :

Sur votre appareil Android, appuyez sur Paramètres > À propos du téléphone.

Appuyez sept fois sur Numéro de version.

Revenez à Paramètres, puis appuyez sur Système > Options du développeur.

Si vous ne voyez pas les options pour les développeurs, appuyez sur Options avancées.

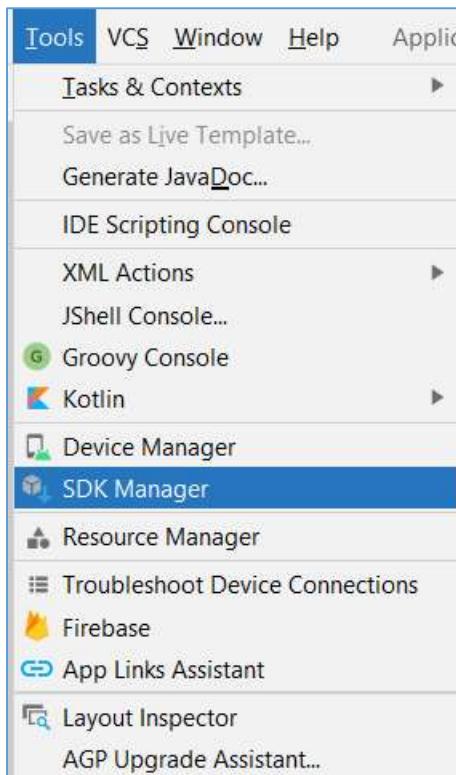


Appuyez sur Options du développeur, puis appuyez sur la bascule de débogage USB pour l'activer.



Dans Android Studio sur Windows, vous devez installer un pilote de périphérique USB avant de pouvoir exécuter votre application sur un périphérique physique.

Dans Android Studio, cliquez sur Outils > Gestionnaire de SDK. La boîte de dialogue Préférences > Paramètres système d'apparence et de comportement > Android SDK s'ouvre.



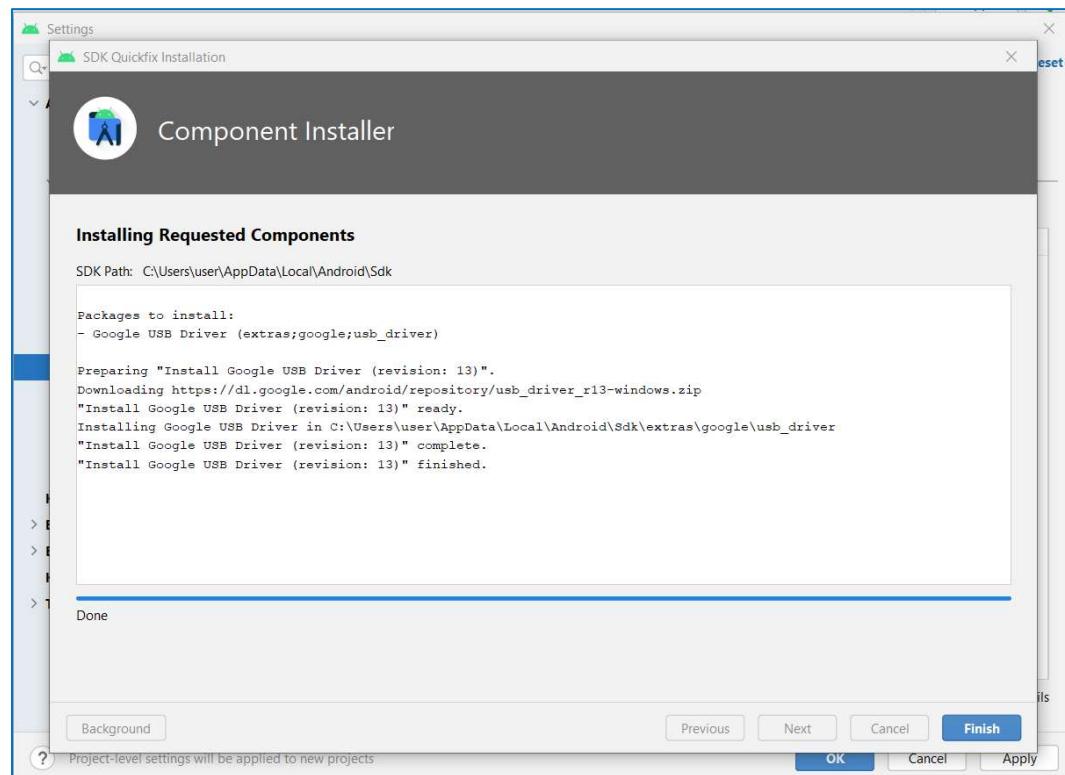
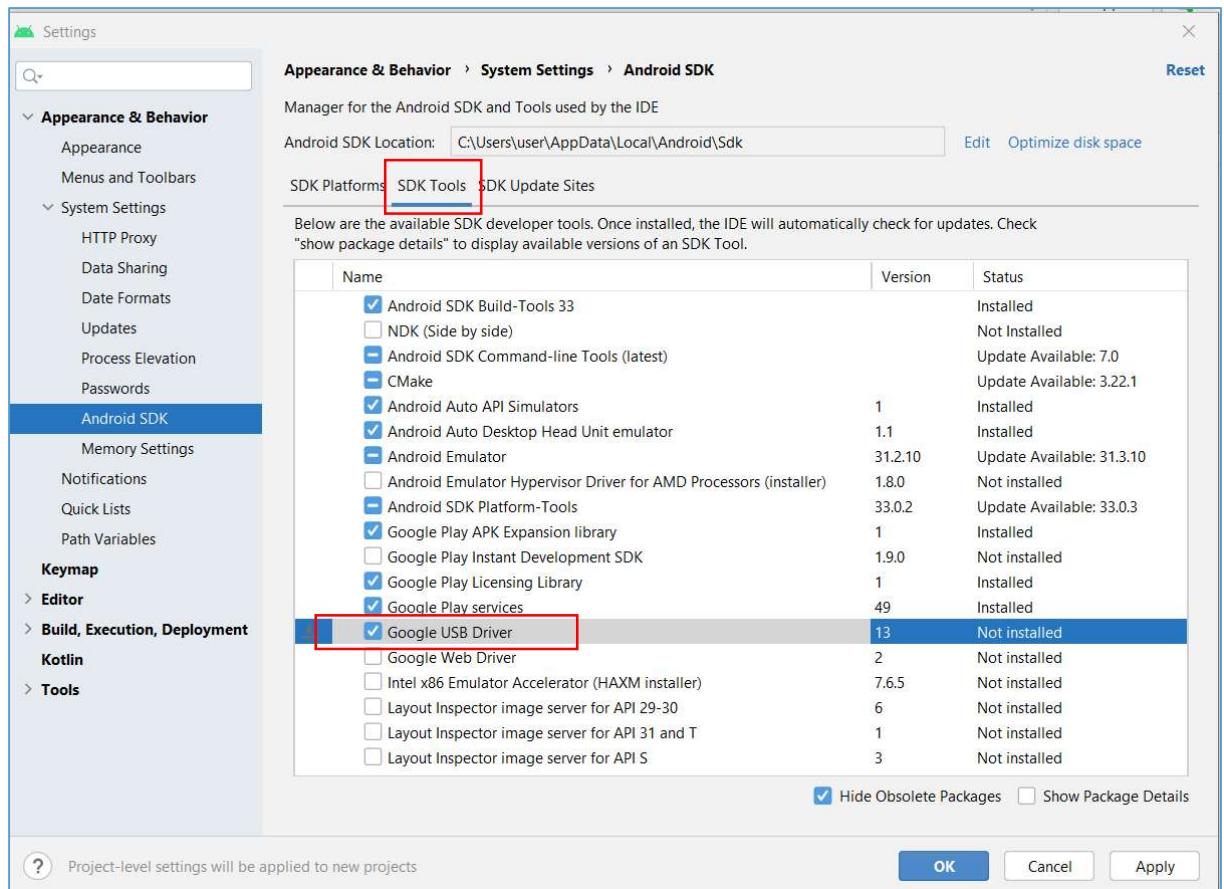
Ou raccourci



SDK Tools

Cliquez sur l'onglet Outils SDK.

Sélectionnez Pilote USB Google, puis cliquez sur OK.





Device Manager

Virtual Physical

Visual Studio Code

Télécharger et installer VSCode en visitant le lien :

<https://code.visualstudio.com/Download>

The screenshot shows the Visual Studio Code download page. At the top, there's a navigation bar with links for Visual Studio Code, Docs, Updates, Blog, API, Extensions, FAQ, and Learn. A search bar and a 'Download' button are also present. A banner at the top indicates 'Version 1.90 is now available! Read about the new features and fixes from May.' Below the banner, the main heading is 'Download Visual Studio Code' with the subtext 'Free and built on open source. Integrated Git, debugging and extensions.' There are three main download sections: 'Windows' (Windows 10, 11), 'Linux' (Debian, Ubuntu, Red Hat, Fedora, SUSE), and 'Mac' (macOS 10.15+). Each section includes icons for the respective operating system, download links, and supported architectures (x64, Arm64, etc.).

Flutter

Télécharger :

<https://docs.flutter.dev/get-started/install/windows/mobile?tab=download>

Download then install Flutter

To install Flutter, download the Flutter SDK bundle from its archive, move the bundle to where you want it stored, then extract the SDK.

1. Download the following installation bundle to get the latest stable release of the Flutter SDK.

[flutter_windows_3.22.2-stable.zip](#)

Ne décompressez pas le fichier d'installation tout de suite après l'avoir téléchargé. Il est d'abord conseillé de créer un dossier « development » dans vos documents. C'est dans celui-ci que vous allez

placer votre dossier Flutter, ainsi que les applications que vous allez créer. Ce n'est qu'alors que vous pouvez le décompresser.

Une fois cela fait, vous pouvez entrer cette suite de commande pour vous assurer que tout fonctionne correctement :

```
cd development  
cd flutter  
cd bin  
flutter doctor
```

```
D:\development\flutter\bin>flutter doctor  
  
Welcome to Flutter! - https://flutter.dev  
  
The Flutter tool uses Google Analytics to anonymously report feature usage statistics and basic crash reports. This data is used to help improve Flutter tools over time.  
  
Flutter tool analytics are not sent on the very first run. To disable reporting, type 'flutter config --no-analytics'. To display the current setting, type 'flutter config'. If you opt out of analytics, an opt-out event will be sent, and then no further information will be sent by the Flutter tool.  
  
By downloading the Flutter SDK, you agree to the Google Terms of Service. The Google Privacy Policy describes how data is handled in this service.  
  
Moreover, Flutter includes the Dart SDK, which may send usage metrics and crash reports to Google.  
  
Read about data we send with crash reports:  
https://flutter.dev/docs/reference/crash-reporting  
  
See Google's privacy policy:  
https://policies.google.com/privacy  
  
To disable animations in this tool, use  
'flutter config --no-cli-animations'.
```

```
Doctor summary (to see all details, run flutter doctor -v):  
[V] Flutter (Channel stable, 3.22.2, on Microsoft Windows [version 10.0.19045.4412], locale fr-FR)  
[V] Windows Version (Installed version of Windows is version 10 or higher)  
[V] Android toolchain - develop for Android devices (Android SDK version 34.0.0)  
[V] Chrome - develop for the web  
[V] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.4.1)  
[V] Android Studio (version 2023.3)  
[V] VS Code (version 1.90.1)  
[V] Connected device (3 available)  
[V] Network resources  
  
• No issues found!  
The Flutter CLI developer tool uses Google Analytics to report usage and diagnostic data along with package dependencies, and crash reporting to send basic crash reports. This data is used to help improve the Dart platform, Flutter framework, and related tools.  
  
Telemetry is not sent on the very first run. To disable reporting of telemetry, run this terminal command:  
  
  flutter --disable-analytics  
  
If you opt out of telemetry, an opt-out event will be sent, and then no further information will be sent. This data is collected in accordance with the Google Privacy Policy (https://policies.google.com/privacy).
```

Actuellement, il vous est possible d'accéder aux commandes de Flutter, uniquement si vous vous trouvez dans le dossier bin. L'étape suivante va donc consister à importer ces commandes, afin de les rendre accessibles partout depuis le terminal. Pour cela, voici la marche à suivre :

Dans la barre de recherche de Windows, commencez à chercher le terme "environnement". Un résultat du type « Modifier les variables d'environnement système » devrait apparaître en premier.

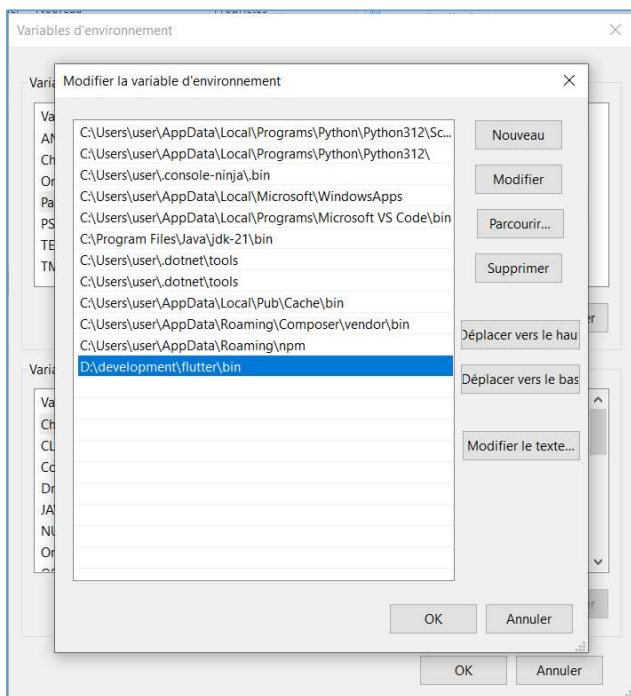
En bas de la boîte de dialogue, cliquez sur le bouton « variable d'environnement ».

Plusieurs variables seront déjà configurées dans la partie « utilisateur », et vous allez pouvoir cliquer sur « nouvelle » et entrer les valeurs suivantes :

Nom de la variable : PATH ;

Valeur de la variable : Le chemin de votre fichier bin, que vous pouvez trouver en cliquant sur « Parcourir le répertoire ».

Cliquez sur « ok » pour valider les changements et fermez la fenêtre.



Fermez le terminal si ce n'était pas le cas, puis rouvrez le dans n'importe quel emplacement et entrez la commande flutter doctor. Si le message s'affiche correctement, c'est que votre variable a bien été exportée.

```
C:\Users\user>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.22.2, on Microsoft Windows [version 10.0.19045.4412], locale fr-FR)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Chrome - develop for the web
[✓] Visual Studio - Develop Windows apps (Visual Studio Community 2022 17.4.1)
[✓] Android Studio (version 2023.3)
[✓] VS Code (version 1.90.1)
[✓] Connected device (3 available)
[✓] Network resources

• No issues found!
```

Remarques :

En cas d'erreurs :

- I. Un problème avec les licences Android.

```
Doctor summary (to see all details, run flutter doctor -v):
[!] Flutter (Channel stable, 2.10.4, on Microsoft Windows [version 10.0.19041.746], locale en-US)
[!] Android toolchain - develop for Android devices (Android SDK version 32.1.0-rc1)
    ! Some Android licenses not accepted. To resolve this, run: flutter doctor --android-licenses
```

Ce problème est assez courant et est atténué en exécutant la commande suivante dans CMD.

```
flutter doctor --android-licenses
```

Lorsque vous y êtes invité, saisissez-y à toutes les invites pour accepter les licences.

```
5 of 7 SDK package licenses not accepted. 100% Computing updates...
```

```
Review licenses that have not been accepted (y/N)? y
```

L'exécution à nouveau de la commande Flutter Doctor montre que le problème est résolu.

- II. Visual Studio - develop for Windows

X Visual Studio not installed; this is necessary for Windows development.

Download at <https://visualstudio.microsoft.com/downloads/>.

Please install the "Desktop development with C++" workload, including all of its default components.

```
Doctor summary (to see all details, run flutter doctor -v):
[!] Flutter (Channel stable, 2.10.0, on Microsoft Windows [Version 10.0.22000.434], locale en-US)
[!] Android toolchain - develop for Android devices (Android SDK version 32.1.0-rc1)
[!] Chrome - develop for the web
[!] Visual Studio - develop for Windows
    ✘ Visual Studio not installed; this is necessary for Windows development.
        Download at https://visualstudio.microsoft.com/downloads/.
        Please install the "Desktop development with C++" workload, including all of its default components
[!] Android Studio (version 2020.3)
[!] VS Code (version 1.64.2)
[!] Connected device (4 available)
[!] HTTP Host Availability

! Doctor found issues in 1 category.
```

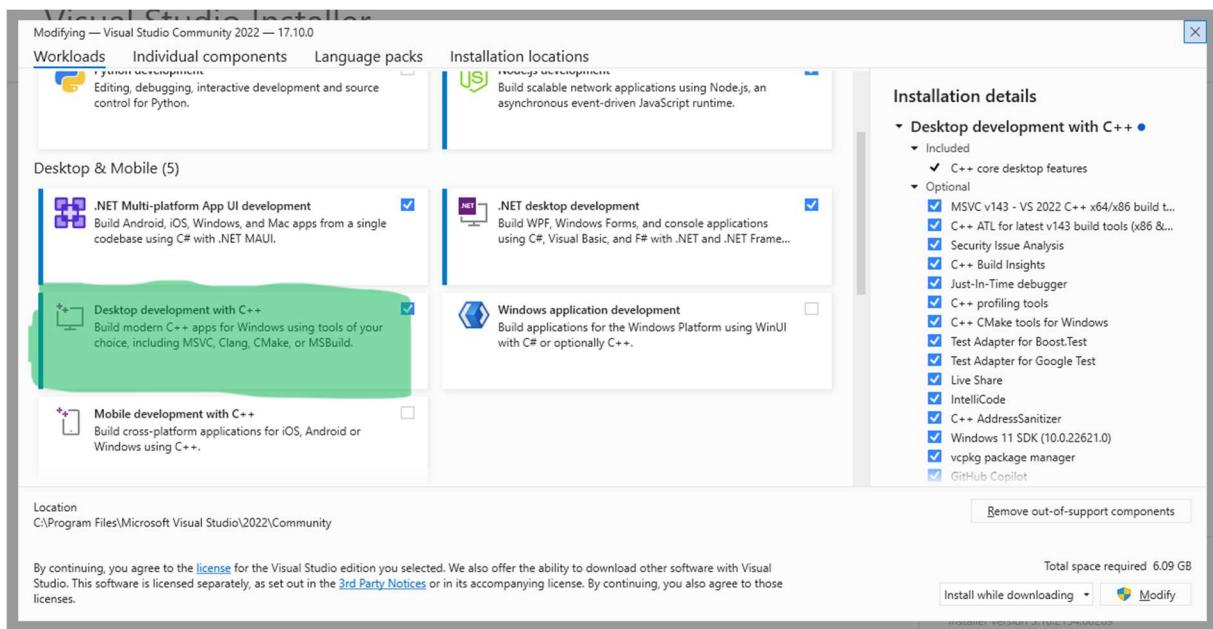
Il faut installer Visual Studio. Car l'application Flutter s'exécute dans un "runner". Sous Windows, ce programme d'exécution est écrit en C++ et utilise la bibliothèque Win32 de Microsoft pour créer la fenêtre de votre application, ce qui nécessite la compilation de Visual Studio.

Pour cela :

1. Installer Visual Studio :

Téléchargez et installez Visual Studio (pas Visual Studio Code) à partir du site officiel :
<https://visualstudio.microsoft.com/fr/downloads/?cid=learn-onpage-download-install-visual-studio-page-cta>

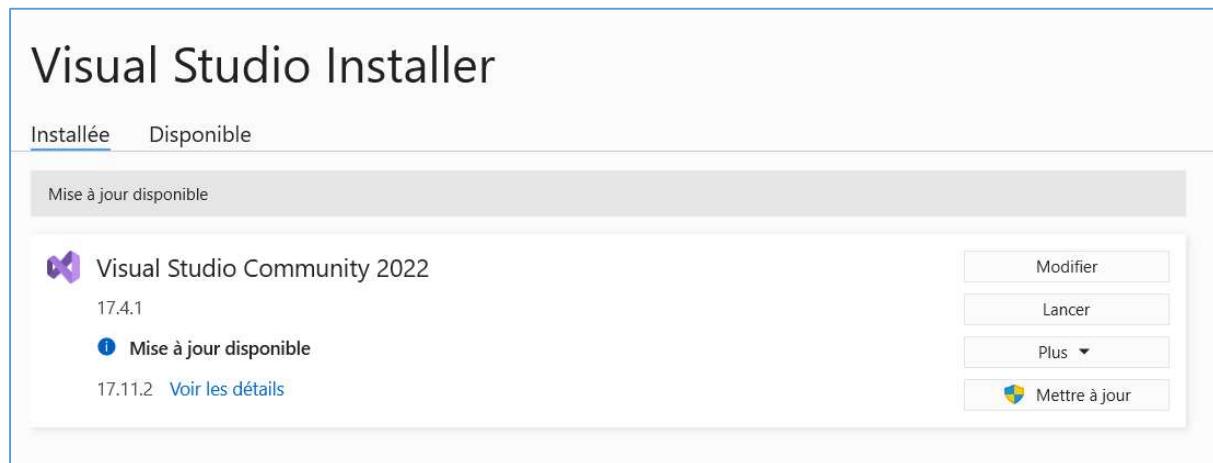
Lors de l'installation, assurez-vous de sélectionner "Desktop development with C++", car Flutter nécessite les outils de build C++ pour la compilation des applications Windows.



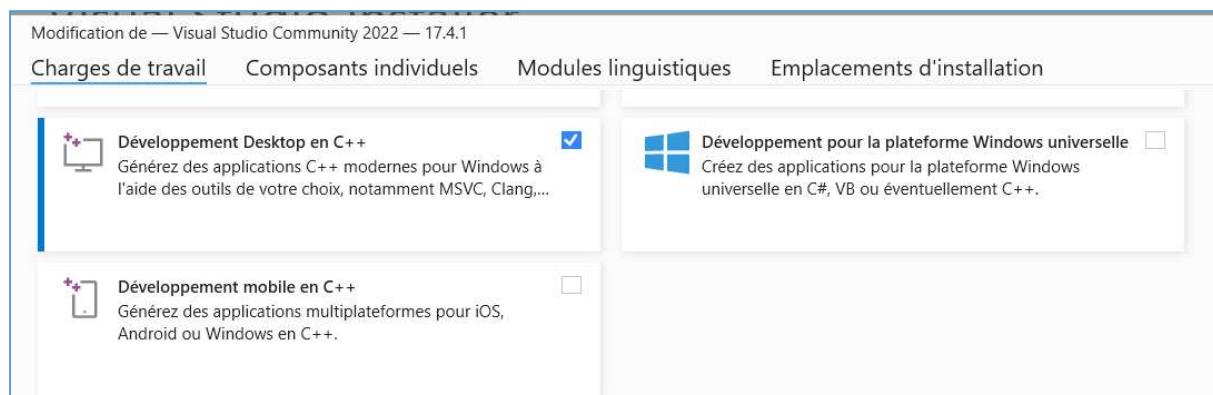
2. Configurer Visual Studio :

Si Visual Studio est déjà installé, mais que vous n'avez pas sélectionné les workloads nécessaires, vous pouvez les ajouter en ouvrant Visual Studio Installer :

Cliquez sur "Modify" à côté de votre installation actuelle de Visual Studio.



Sélectionnez "Desktop development with C++".



Détails de l'installation

- ▼ Développement Desktop en C++
 - ▼ Inclus
 - ✓ Fonctionnalités de bureau de base C++
 - ▼ Facultatif
 - ✓ MSVC V143 - VS 2022 C++ x64/x86 Build T...
 - ✓ C++ ATL pour dernière version de Build To...
 - Kit de développement logiciel (SDK) Wind...
 - ✓ Débogueur juste-à-temps
 - ✓ Outils de profilage C++
 - ✓ Outils C++ CMake pour Windows
 - ✓ Adaptateur de test pour Boost.Test
 - ✓ Adaptateur de test pour Google Test
 - ✓ Live Share
 - ✓ IntelliCode
 - ✓ AddressSanitizer C++
 - C++ MFC pour dernière version de Build T...
 - Modules C++ pour Build Tools v143 (x64/x...
 - Kit de développement logiciel (SDK) Wind...
 - Prise en charge de C++/CLI pour Build Too...
 - Outils C++ Clang pour Windows (15.0.1 —)
 - ✓ Diagnostics JavaScript
 - Incredibuild - Accélération de build
 - Kit de développement logiciel (SDK) Wind...
 - ✓ Kit SDK Windows 10 (10.0.19041.0)

3. Relancer flutter doctor :

Après avoir installé ou configuré Visual Studio correctement, relancez la commande flutter doctor dans votre terminal pour vérifier que l'erreur est résolue.

Les extensions VScode

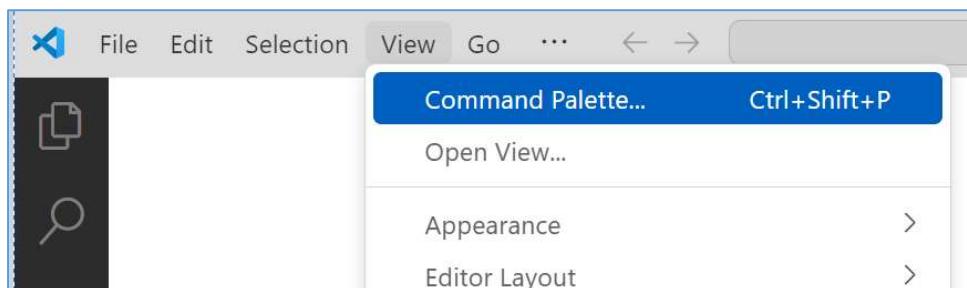
The screenshot shows the Visual Studio Code interface with the Extensions sidebar open. A search bar at the top contains the text "flutter". Below it, the "Flutter" extension by Dart Code is listed, showing a download count of 8.5M and a rating of 4.5 stars from 84 reviews. An "Install" button is visible next to the extension name.

Below the main interface, a detailed view of the Flutter extension page is shown in a browser window. The page title is "Extension: Flutter". It features the Flutter logo, the version "v3.90.0", and developer information: "Dart Code" and "dartcode.org". It also displays the download count "8,549,475" and the rating "★★★★★ (84)". A brief description states: "Flutter support and debugger for Visual Studio Code." At the bottom of this page view, there is an "Install" button and a "Dart Code" logo.

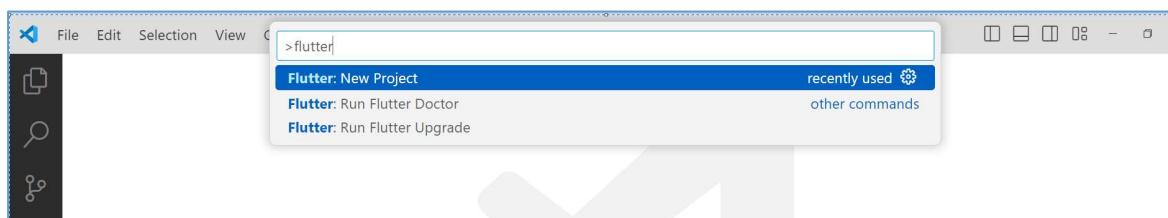
Créer un nouveau projet

Pour créer un nouveau projet Flutter à partir du modèle d'application de démarrage Flutter :

Accédez à Affichage > Palette de commandes... .



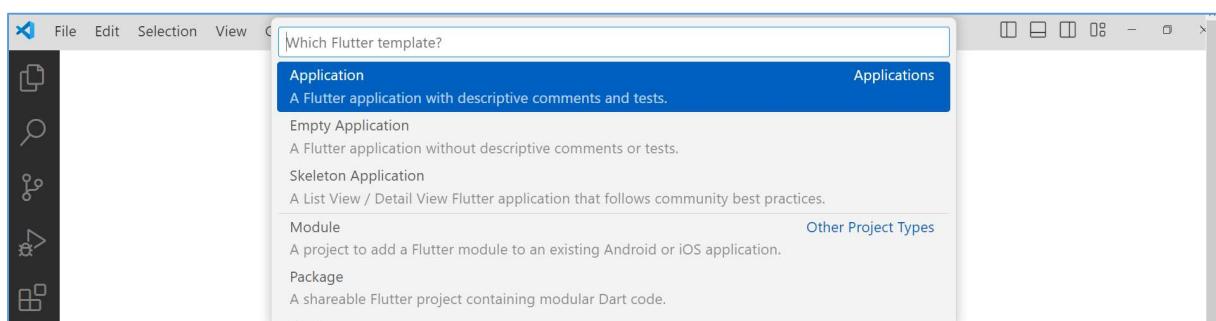
Vous pouvez également appuyer sur CTRL SHIFT P

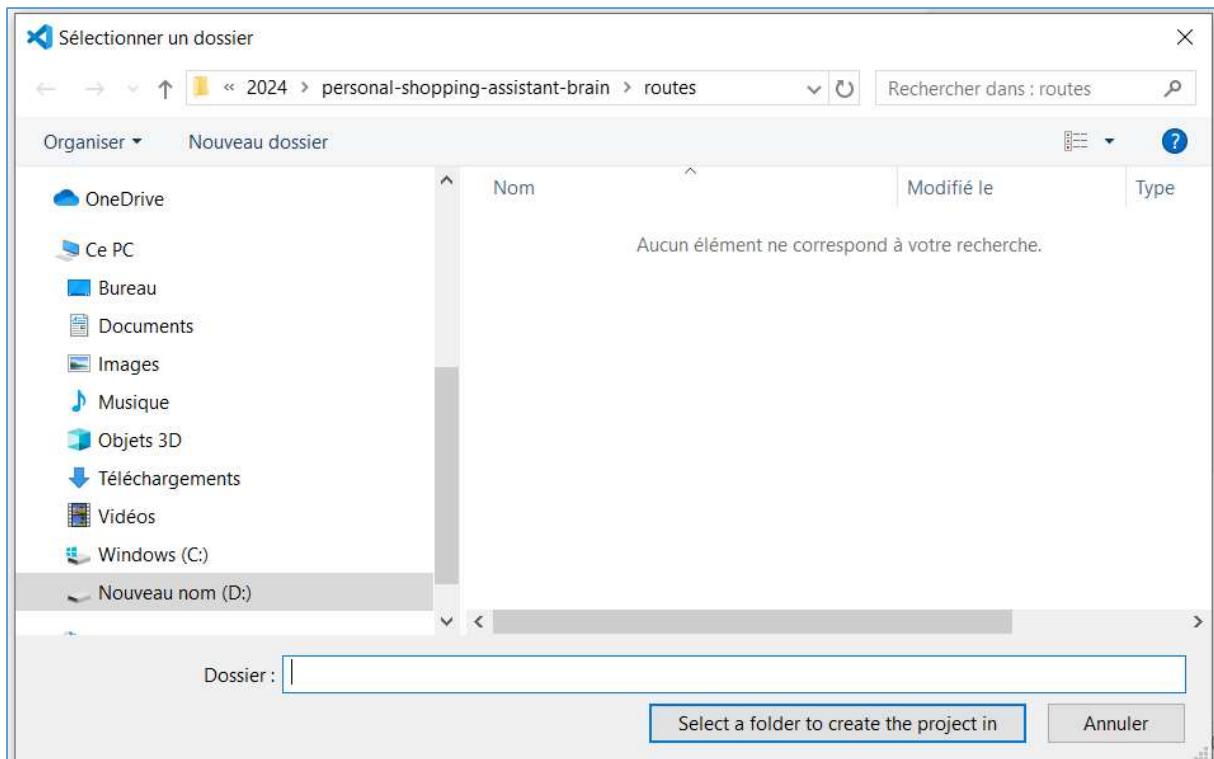


Taper flutter

Sélectionnez le Flutter : Nouveau projet.

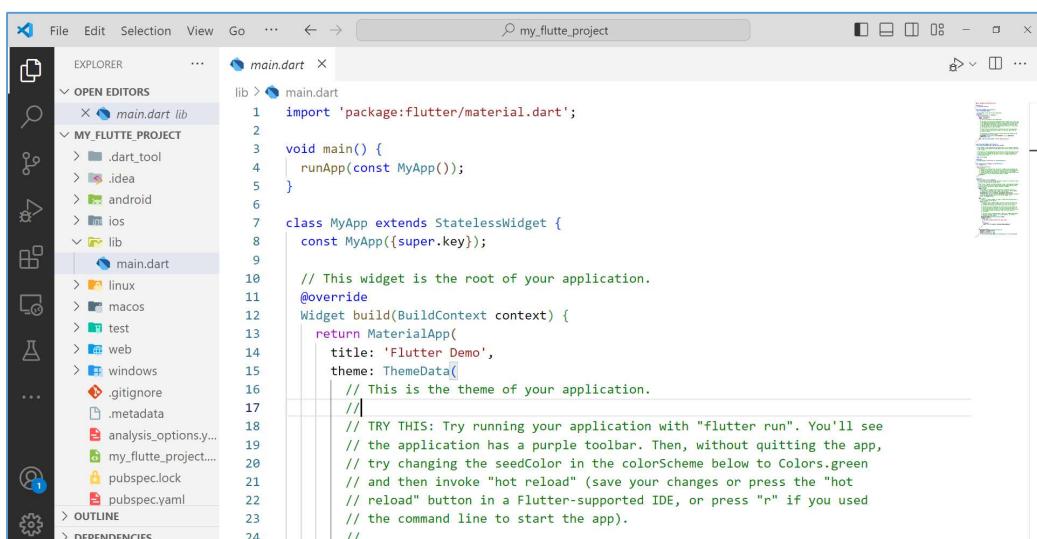
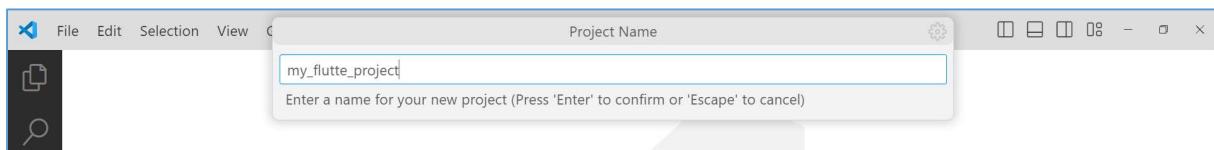
Sélectionnez Application.

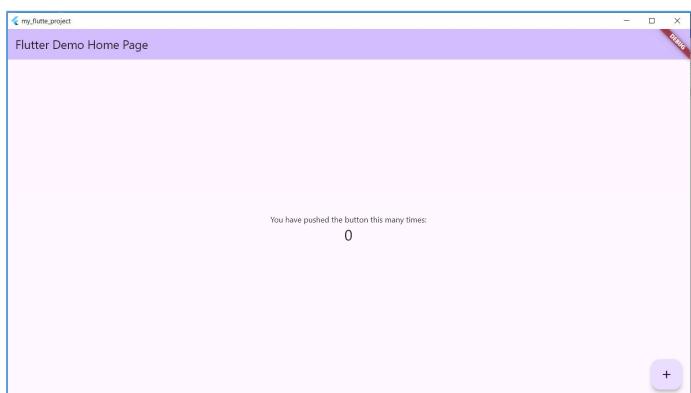
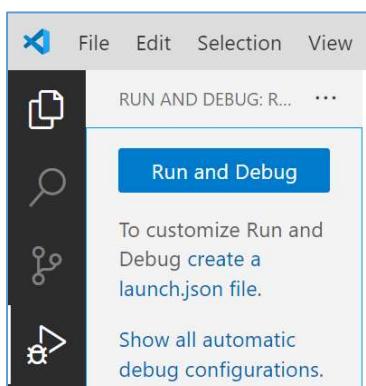
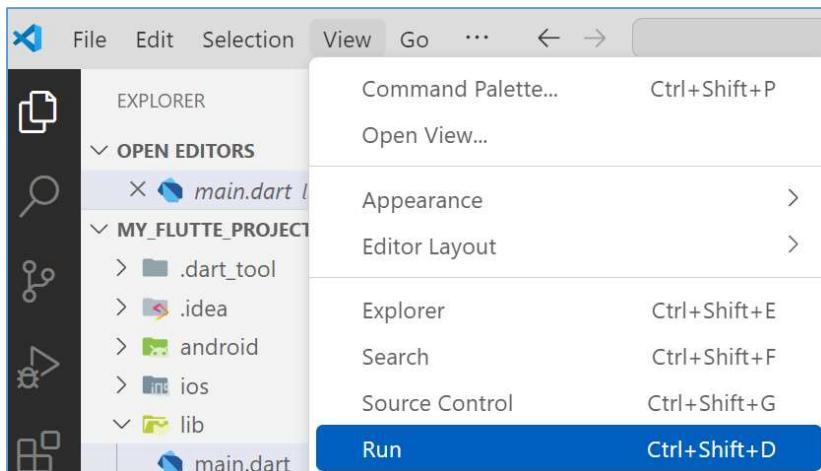




Sélectionnez un emplacement de projet.

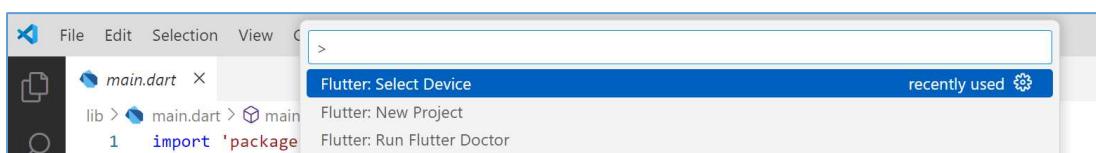
Entrez le nom de votre projet souhaité.

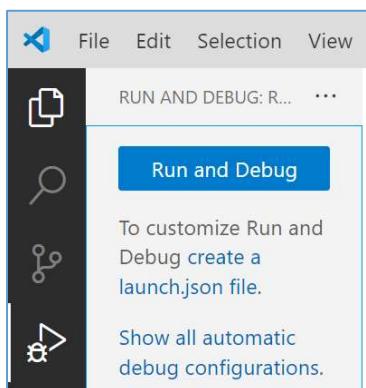
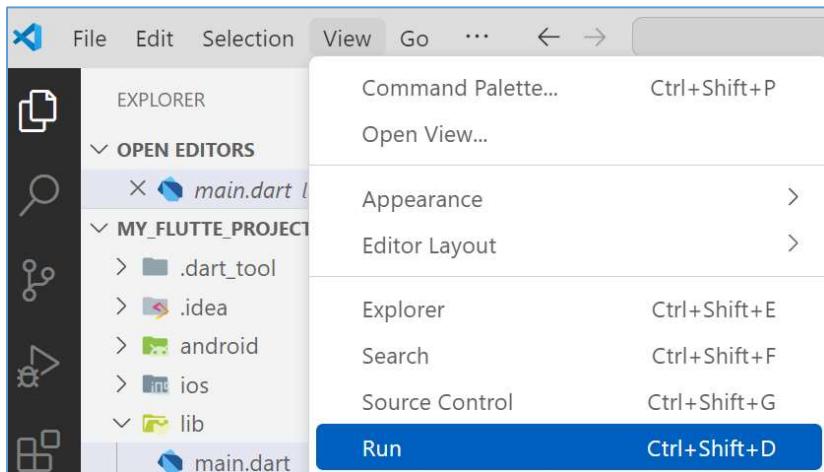
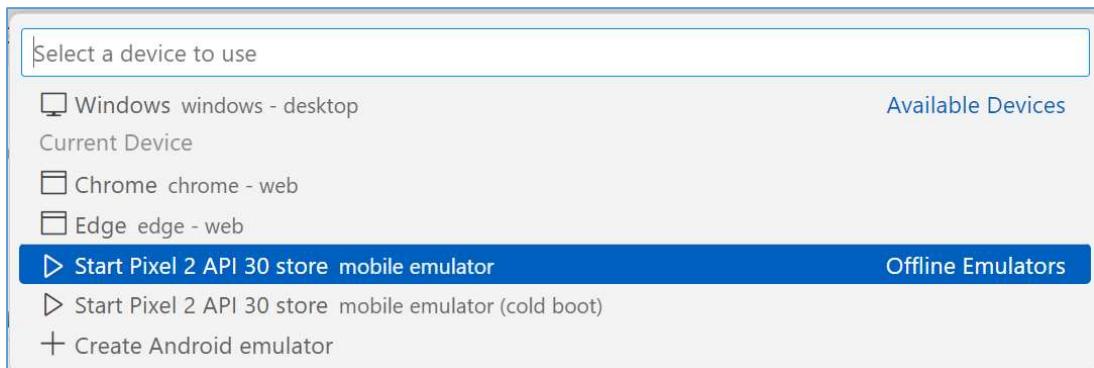




Maintenant on va voir le résultat dans l'émulateur.

CTRL SHIFT P





Cliquez sur Exécuter > Démarrer le débogage. Vous pouvez également appuyer sur F5. La barre d'état devient orange pour indiquer que vous êtes dans une session de débogage.

The screenshot shows the Android Studio interface. On the left is the tool palette with icons for variables, watch, call stack, and breakpoints. The main area displays the code for `main.dart`:

```
lib > main.dart > main
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10 // This widget is the root of your application.
11 @override
12 Widget build(BuildContext context) {
```

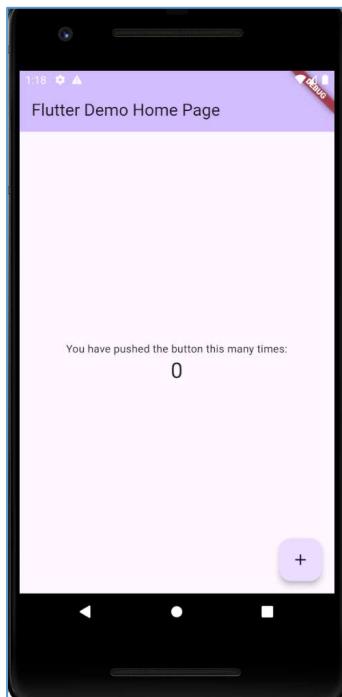
The debug console at the bottom shows:

```
Launching lib/main.dart on sdk gphone x86 in debug mode...
Flutter: Running Gradle task 'assembleDebug'.
```

Après un moment :

The screenshot shows the debug console output after the application has started:

```
Launching lib/main.dart on sdk gphone x86 in debug mode...
✓ Built build/app/outputs/flutter-apk/app-debug.apk
Connecting to VM Service at ws://127.0.0.1:53618/zvuAe89tRxs=/ws
```



Ecrire la première application

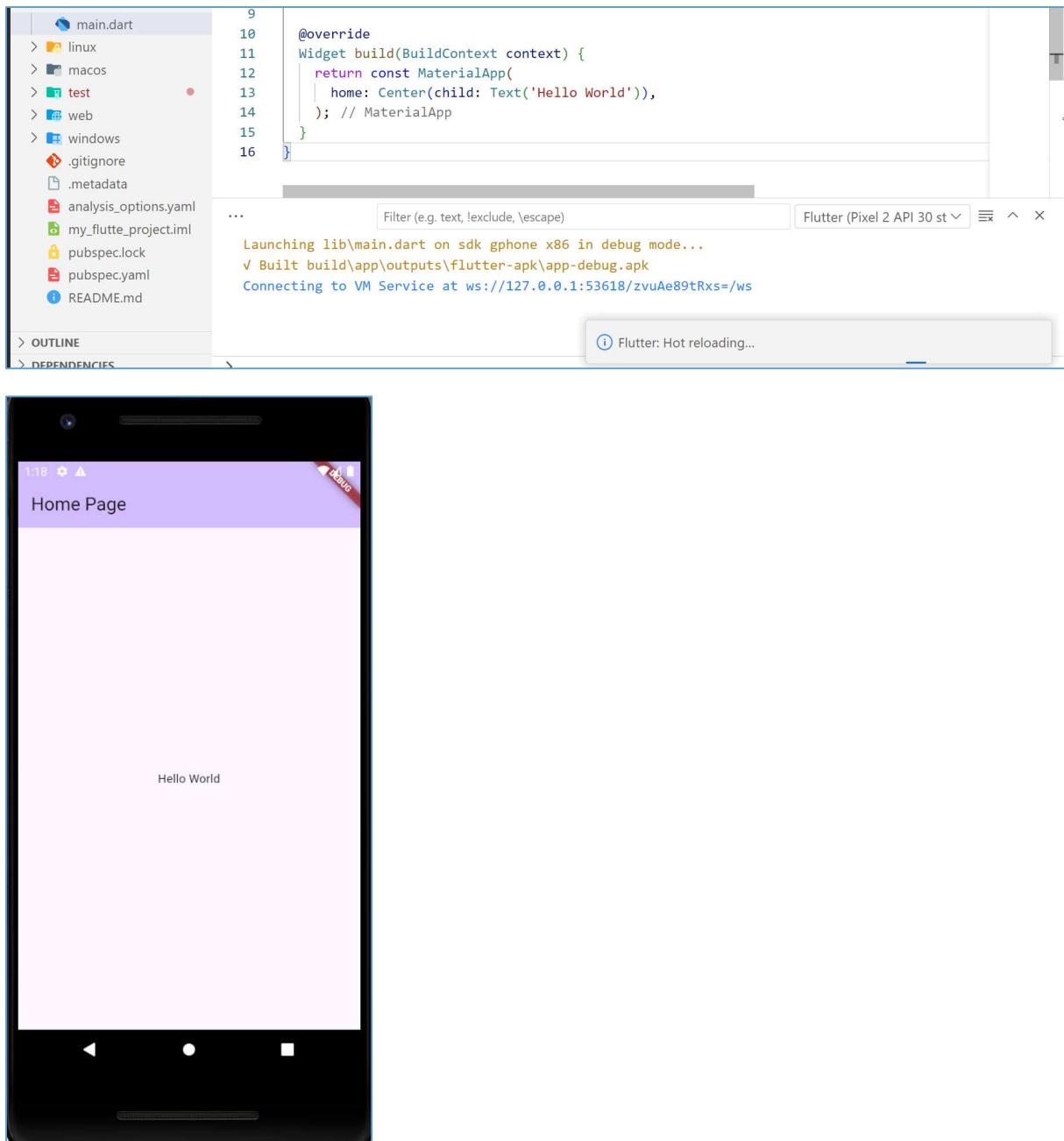
lib > main.dart

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

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

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Application',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: Scaffold(
        appBar: AppBar(
          backgroundColor: Theme.of(context).colorScheme.inversePrimary,
          title: const Text('Home Page'),
        ),
        body: const Center(
          child: Text(
            'Hello World',
          ),
        ),
      ),
    );
  }
}
```



```
9
10
11 @override
12 Widget build(BuildContext context) {
13   return const MaterialApp(
14     home: Center(child: Text('Hello World')),
15   ); // MaterialApp
16 }
```

... Filter (e.g. text, !exclude, \escape) Flutter (Pixel 2 API 30 st ▾)

Launching lib\main.dart on sdk gphone x86 in debug mode...
✓ Built build\app\outputs\flutter-apk\app-debug.apk
Connecting to VM Service at ws://127.0.0.1:53618/zvuAe89tRxs=/ws

Flutter: Hot reloading...

11:18 Home Page

Hello World

Dans Flutter, tout est un widget et en utilisant des widgets prédéfinis, on peut créer des widgets définis par l'utilisateur, tout comme en utilisant int, float, double, nous pouvons créer un type de données défini par l'utilisateur. Dans Flutter, il existe trois types de widgets

Stateless Widget

Stateful Widget

Inherited Widget

Dans cet exemple, nous avons utilisé le widget Stateless , l'application Material, le centre et le widget texte.

Widget sans état : dans Flutter StatelessWidget, les widgets ne peuvent pas changer d'état. Dans StatelessWidget, il existe une méthode (fonction) appelée build qui est responsable du dessin des composants sur l'écran, appelé une seule fois. Pour redessiner un widget sans état, il faut créer une nouvelle instance du widget sans état.

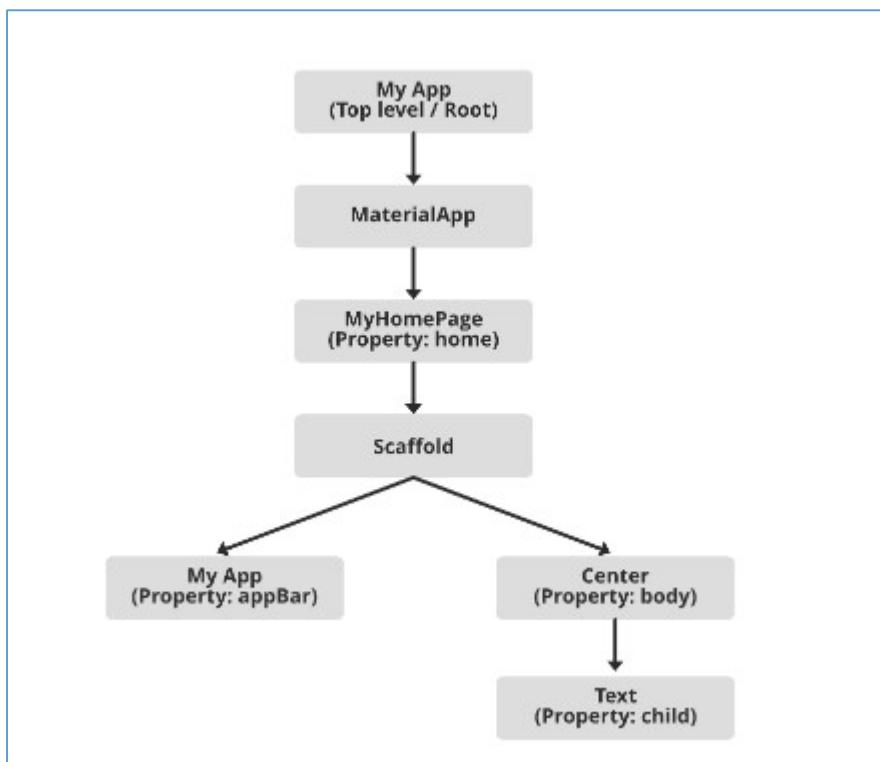
MaterialApp : C'est également un widget fourni par Flutter Team, qui suit le schéma de conception matérielle de Google, MaterialApp est une classe qui a divers arguments nommés comme home : dans lequel nous passons le widget qui doit être affiché sur l'écran d'accueil d'une application.

Center Widget : Center est également un widget prédéfini par Flutter Team, qui prend un autre widget dans son argument enfant. En utilisant Center Widget comme son nom l'indique, il affichera Widget dans son argument enfant dans Center.

Widget de texte : le widget de texte est également prédéfini par Flutter Team, qui est utilisé pour afficher du texte. Créons maintenant une application Hello World en utilisant Flutter.

Build : C'est une méthode responsable du dessin des composants sur l'écran. Elle prend un BuildContext comme argument qui contient des informations sur le widget qui doit être affiché et dans quel ordre il doit être peint sur l'écran.

Ensuite, nous avons utilisé la propriété home de MaterialApp, qui à son tour contient le widget Scaffold. Le widget Scaffold contient tout l'écran de l'application. Nous avons utilisé la propriété appBar qui prend le widget AppBar comme objet. Et à son tour, le widget AppBar contient « GFG » comme titre. Ensuite, nous avons le corps, qui est à nouveau la propriété du MaterialApp. Le centre est l'objet du corps et son enfant est le widget Texte qui lit « Hello World ».



Des extensions VSC utiles pour Flutter

Flutter : facilite l'écriture du code de Flutter



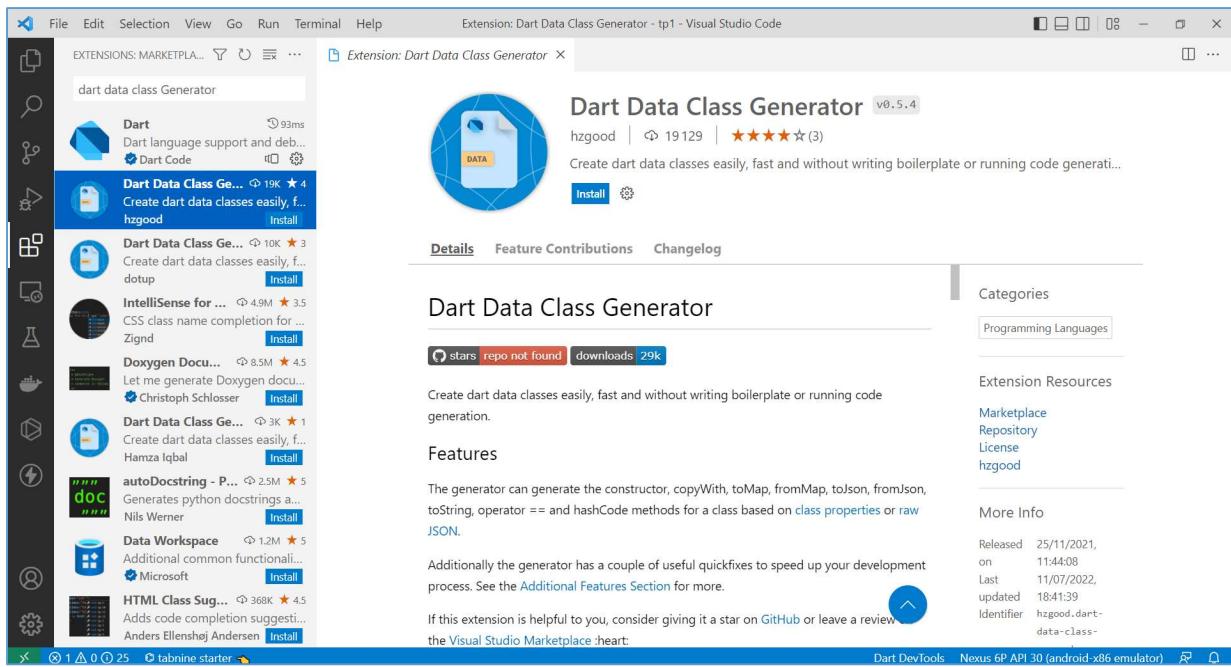
Flutter Widget Snippets



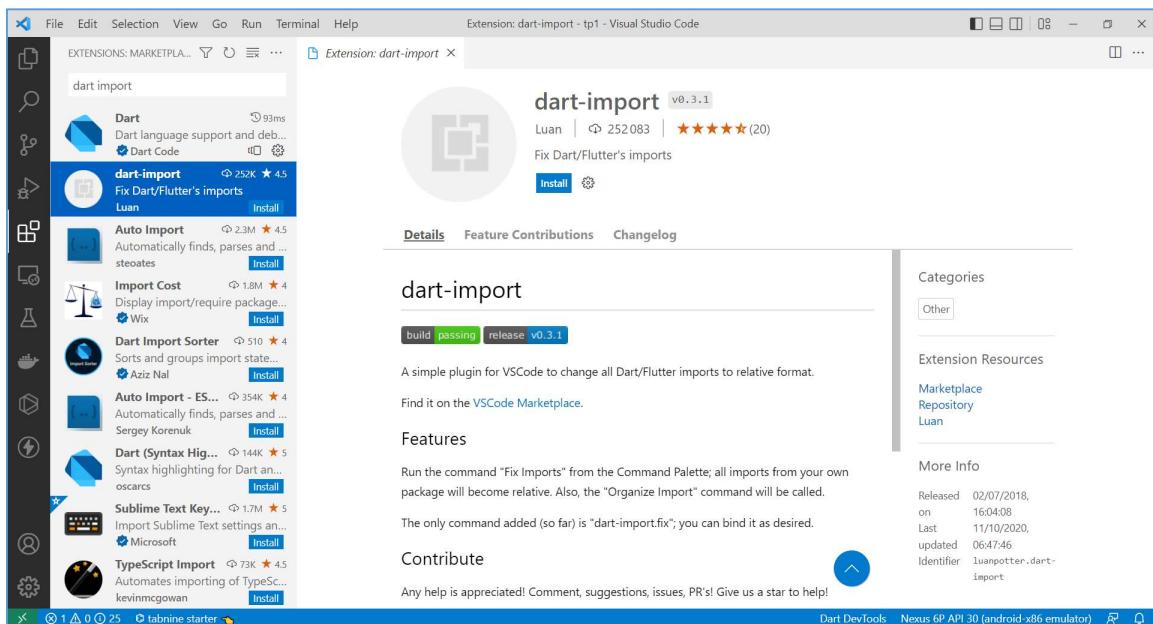
Dart



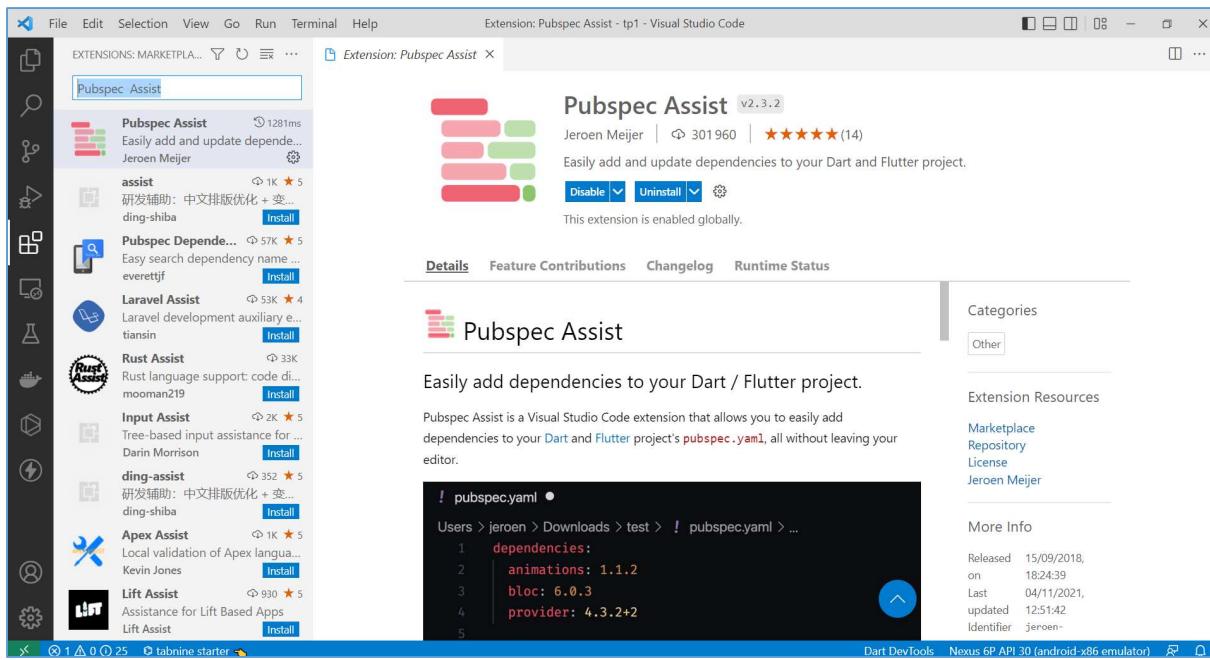
Dart Data Class Generator : plus de facilité lors de la création du Modal class



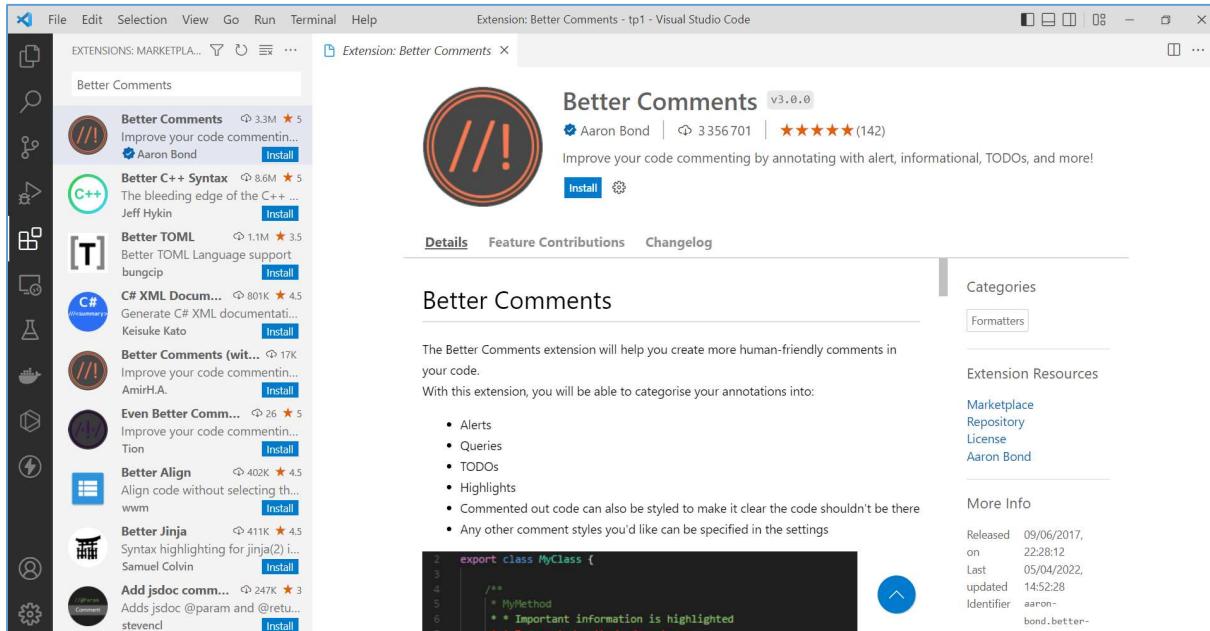
dart import : automatiquement ajoute les imports manquants



Pubspec Assist : sert à l'ajout d'une bibliothèque avec CTRL + SHIFT + P puis saisir le nom de la bibliothèque (ou une partie de son nom)



Better Comments : meilleure visualisation pour les commentaires



Error Lens : meilleure visualisation des erreurs dans le code

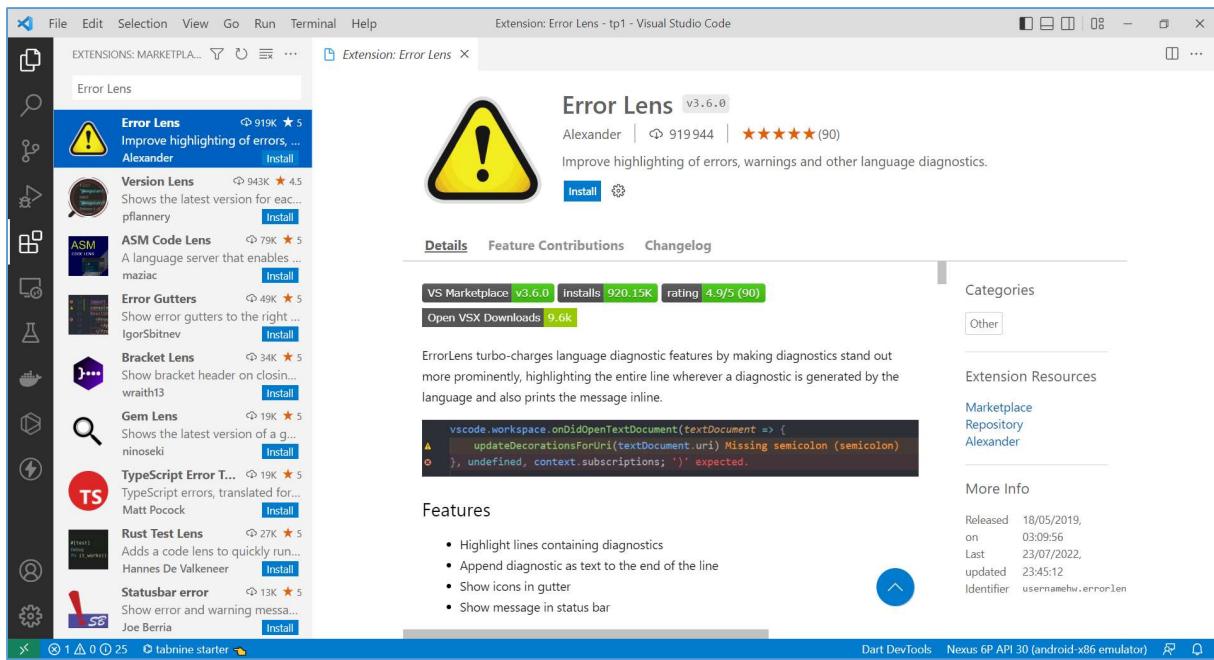
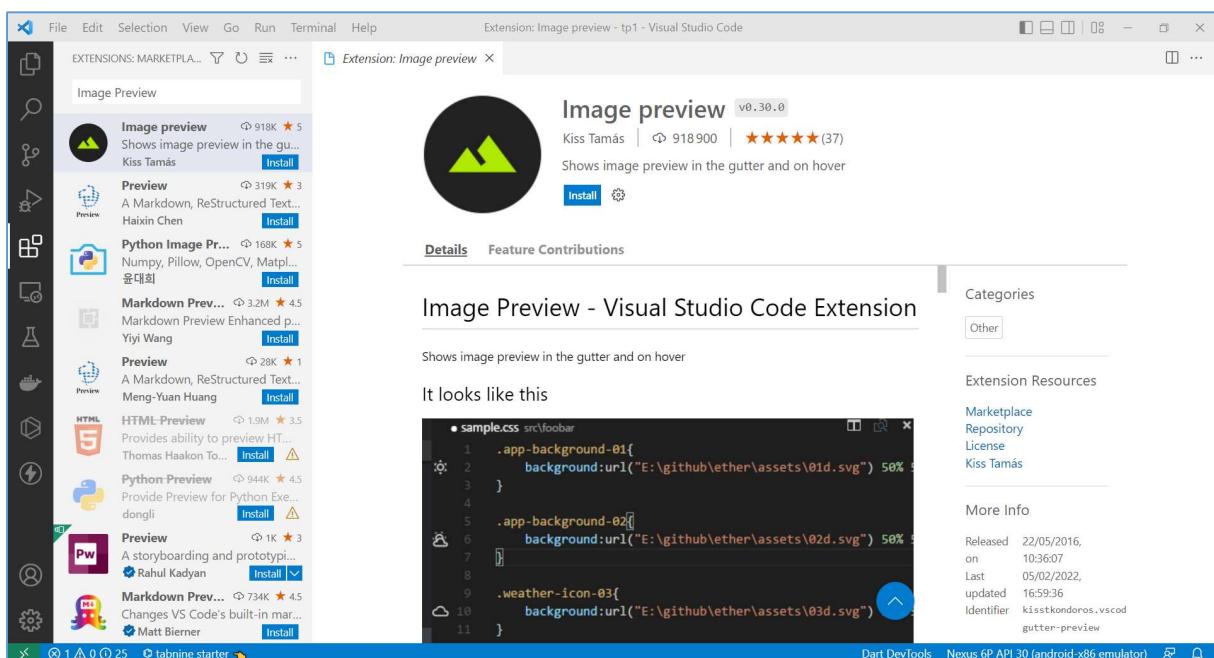
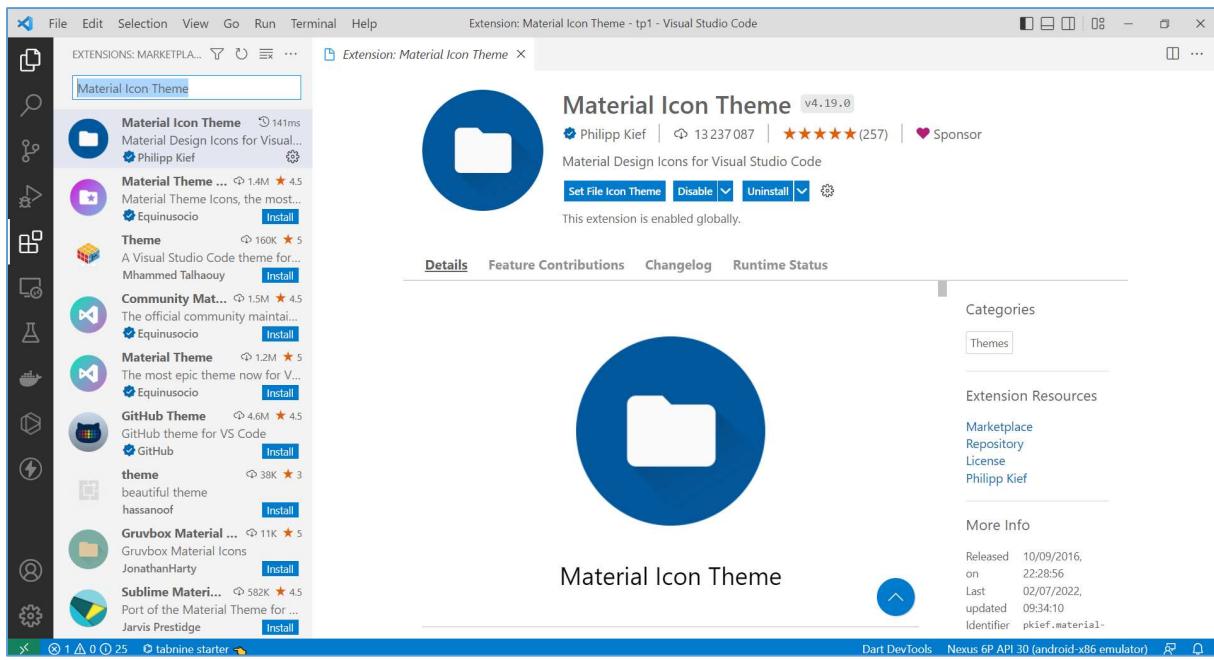


Image Preview : Affiche les images / icônes à gauche du code



Material Icon Theme : ajoute des icônes devant les fichiers dans l'arborescence de l'explorateur VSC

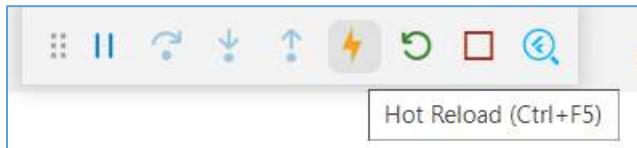


Le Hot Reload

La fonction de recharge à chaud de Flutter vous aide à expérimenter rapidement et facilement, à créer des interfaces utilisateur, à ajouter des fonctionnalités et à corriger les bugs. Le recharge à chaud fonctionne en injectant des fichiers de code source mis à jour dans la machine virtuelle Dart (VM) en cours d'exécution. Une fois que la machine virtuelle a mis à jour les classes avec les nouvelles versions des champs et des fonctions, le Framework Flutter reconstruit automatiquement l'arborescence des widgets, vous permettant de visualiser rapidement les effets de vos modifications.



Cette option doit être active dans VSC



Le but est qu'une fois on écrit le code et on enregistre directement le résultat est affiché instantanément.

MAIS ATTENTION ! Il faut que le main du programme est appelé dans un Widget pour que le Hot Reload fonctionne.

C'est quoi widget en Flutter ?

Les **widgets Flutter** sont les principaux composants d'une application. Ils donnent l'apparence en fonction de leur configuration et de leur état. Dans **Flutter**, tout ce qui apparaît dans l'interface utilisateur s'appelle **widget** et ils héritent tous de la classe **Widget**.

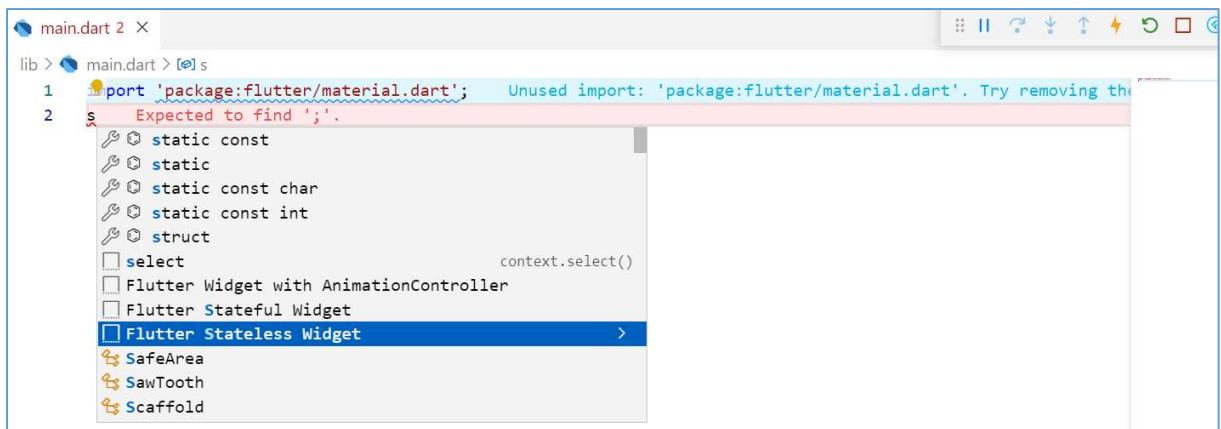
Chaque widgets Flutter peut être sans état (Stateless Widget) ou avec état (Stateful Widget). La principale différence est la possibilité de restituer les widgets au moment de l'exécution. Le widget sans état ne sera utilisé qu'une seule fois et est immuable. Par contre un widget avec état peut être utilisé plusieurs fois en fonction du changement de son état interne.

On peut utiliser les raccourcis offerts par le Snippets comme par exemple :

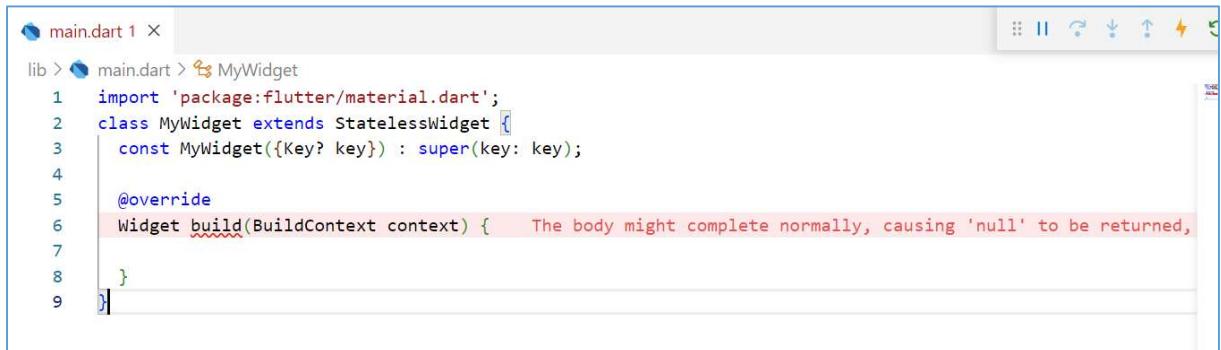
Taper s



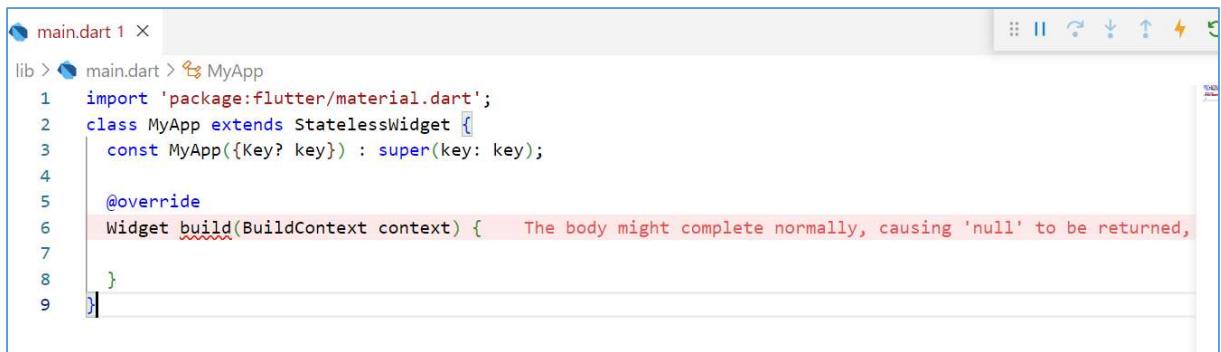
Choisir par exemple :



Ce qui donne



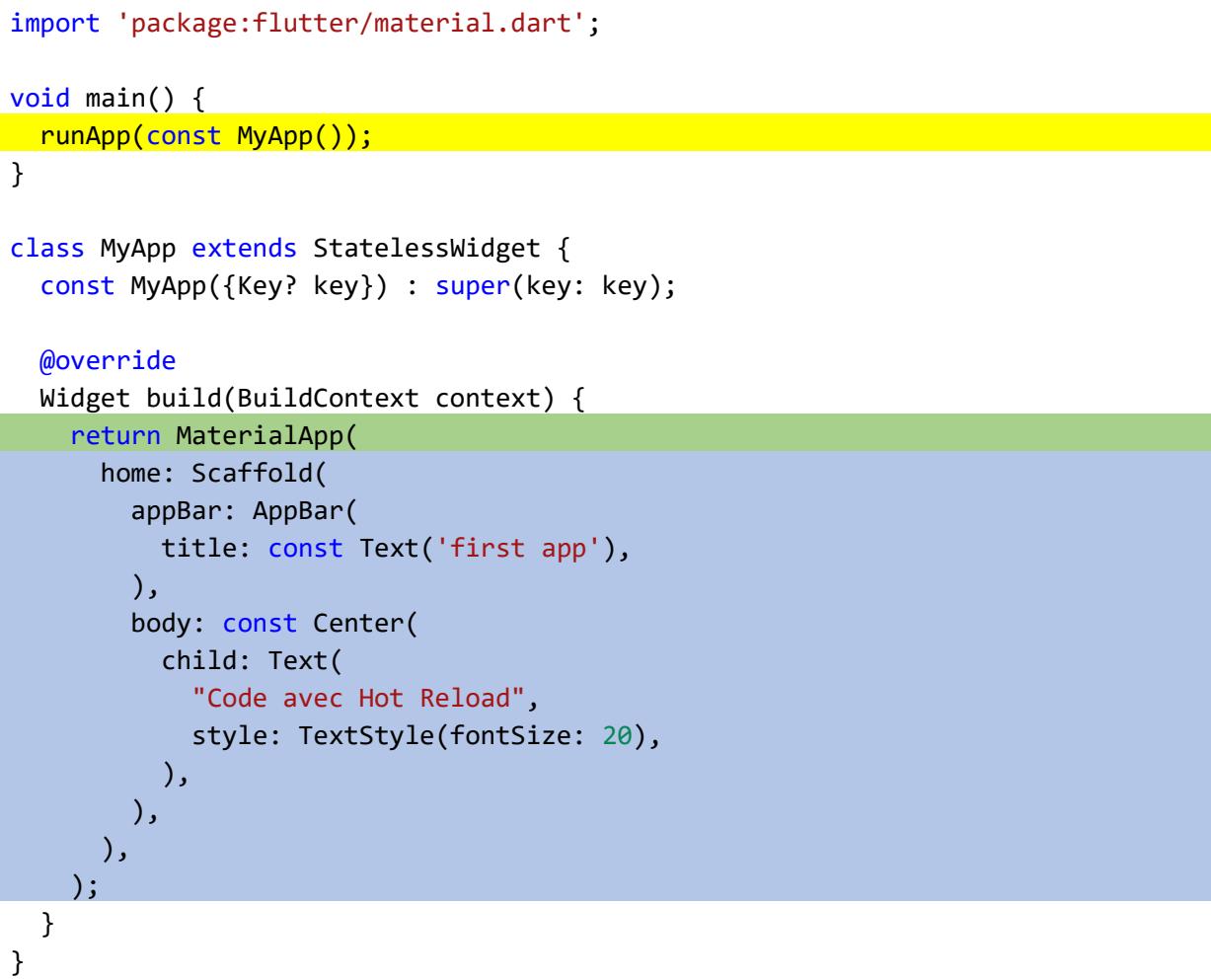
Donner le nom MyApp à la place de MyWidget avant de perdre la sélection.



The screenshot shows a code editor window with a file named 'main.dart' open. The code defines a class 'MyApp' that extends 'StatelessWidget'. The 'build' method is annotated with '@override'. A red squiggly underline is under the word 'Widget' in 'Widget build(BuildContext context) {'. A tooltip above the underline reads: 'The body might complete normally, causing 'null' to be returned,'. The code editor has a toolbar at the top with various icons.

```
lib > main.dart 1 <
lib > main.dart > MyApp
1 import 'package:flutter/material.dart';
2 class MyApp extends StatelessWidget {
3   const MyApp({Key? key}) : super(key: key);
4
5   @override
6   Widget build(BuildContext context) {    The body might complete normally, causing 'null' to be returned,
7
8   }
9 }
```

Le code deviendra



The screenshot shows the completed code for the 'main.dart' file. It includes the 'MaterialApp' widget, which is highlighted in green. The 'MaterialApp' widget contains a 'Scaffold' with an 'AppBar' and a 'Center' child containing a 'Text' widget with the text 'Code avec Hot Reload'. The code editor has a toolbar at the top with various icons.

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

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text('first app'),
        ),
        body: const Center(
          child: Text(
            "Code avec Hot Reload",
            style: TextStyle(fontSize: 20),
          ),
        ),
      ),
    );
  }
}
```

Pour prendre en compte ce widget il faut pour la première fois faire



Puis plus besoin de le faire, et pour chaque enregistrement le changement est instantanément fait :

