
FORMATION FLUTTER N°1
FLUTTER
RÉVOLUTION
Créez de Magnifiques Applications
pour iOS & Android avec Flutter

1. Bienvenue
    1.1 Mes débuts avec Ionic
    1.2 Ma découverte de Flutter
    1.3 Pourquoi ce cours ?

2. Qu'est-ce que Flutter ?
    2.1 Sa rapidité de développement
    2.2 Son interface utilisateur très flexible
    2.3 Des performances natives
    2.4 Qui utilise Flutter ?

3. Qu'est-ce que Dart ?
    3.1 Le Dart et les autres langages
    3.2 Le SDK Dart
    3.3 Quelques exemples du Dart

4. Installer Flutter sur macOS
    4.1 Installer le SDK Flutter
    4.2 Configurer le SDK Flutter
        4.2.1 Méthode 1: zsh (recommandée)
        4.2.2 Méthode 2: bash (ancienne méthode)
    4.3 Configuration de Xcode
    4.4 Configurer Android Studio

5. Installer Flutter sur Windows
    5.1 Installer le SDK Flutter sur Windows
    5.2 Configurer le SDK Flutter sur Windows
    5.3 Configurer Android Studio

6. Configurer Visual Studio Code
    6.1 Installer et configurer VS Code sur Mac
    6.2 Installer et configurer VS Code sur Windows
    6.3 Créer une application en partant de zéro

7. Pour aller plus loin
    7.1 À propos de Flutter Révolution
    7.2 La création de la Flutter Académie
    7.3 Les différentes offres

1. Bienvenue

Bonjour à tous et bienvenue dans **FLUTTER RÉVOLUTION**!

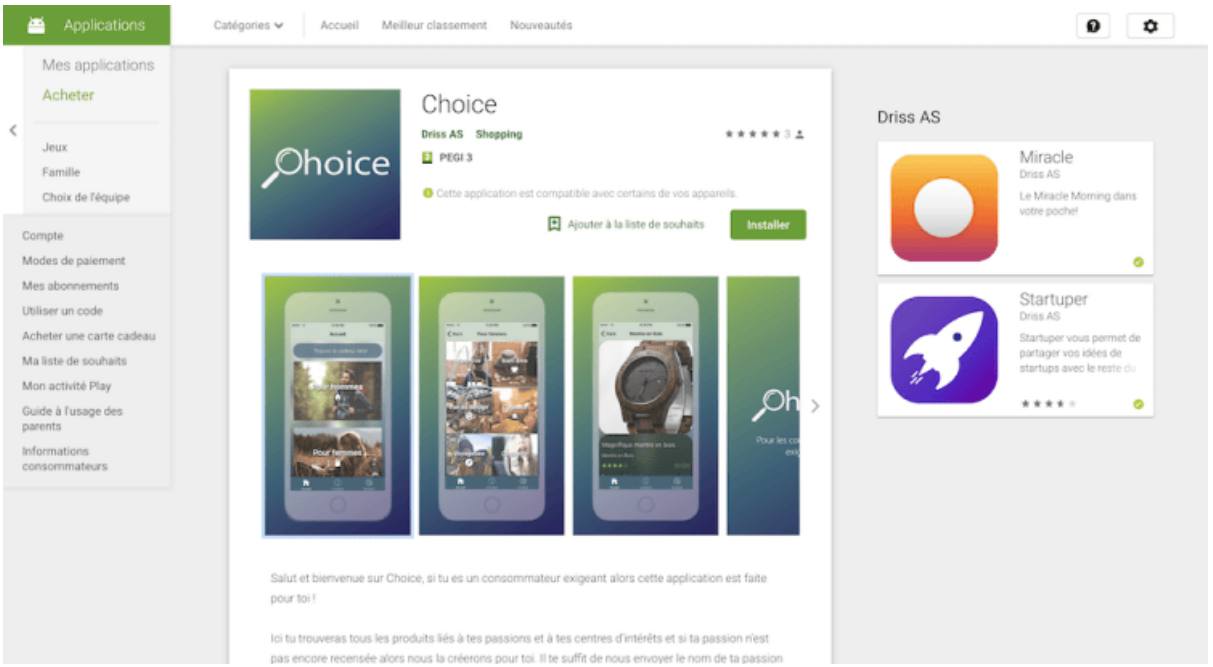
Je vous remercie tout d'abord de m'avoir fait confiance et de me rejoindre dans ce cours qui j'en suis sûr **changera votre vision** du développement mobile.

Dans cette première vidéo, je vais vous parler de mon **parcours de développeur** et de ma découverte de Flutter.

1.1 Mes débuts avec Ionic

Si vous connaissez un peu mon parcours, vous savez que je suis **développeur mobile** depuis **2017** après avoir commencé par le développement web.

J'ai commencé à créer mes premières applications avec le **framework Ionic** qui à l'époque m'a permis de démarrer assez rapidement.



Choice, ma toute première application mobile

J'ai par la suite étendu mes connaissances et proposé **mes services** à d'autres clients, puis finalement, j'ai **créé des cours**.

À travers ma chaîne **YouTube** et mon **blog**, vous êtes maintenant des milliers à suivre mes vidéos et mes **tutoriels gratuits** sur le développement mobile.

Mon activité a pu ainsi prospérer grâce à mes **cours en ligne** payants et à mes prestations de cours particuliers.

Ionic utilise le framework **Angular de Google** qui est basé sur les technologies **Web** et permet de coder des applications avec les langages **HTML, CSS et TypeScript**.

Le framework Ionic permet lui aussi de développer pour iOS et Android

Le framework **le plus connu** cependant à l'époque où j'ai commencé à me former était **React JS** de Facebook.

Il avait l'avantage d'être utilisé par de grandes startups dont **Facebook** et **Instagram** par exemple, ce qui lui confie une certaine notoriété.

J'ai par la suite longtemps justifié mon choix du framework Ionic en invoquant sa **simplicité**.

En fait Ionic n'a jamais était aussi **populaire** que React... Mais je suis toujours parti du principe que si j'avais pu **démarrer de zéro** avec Ionic, les autres le pouvait aussi.

1.2 Ma découverte de Flutter

J'avais déjà annoncé à plusieurs reprises que je passerai un jour ou l'autre à **React JS** pour m'adapter à la forte demande du **marché**.

Pourtant, vous êtes en train de lire un **cours sur Flutter**... Alors pourquoi ?

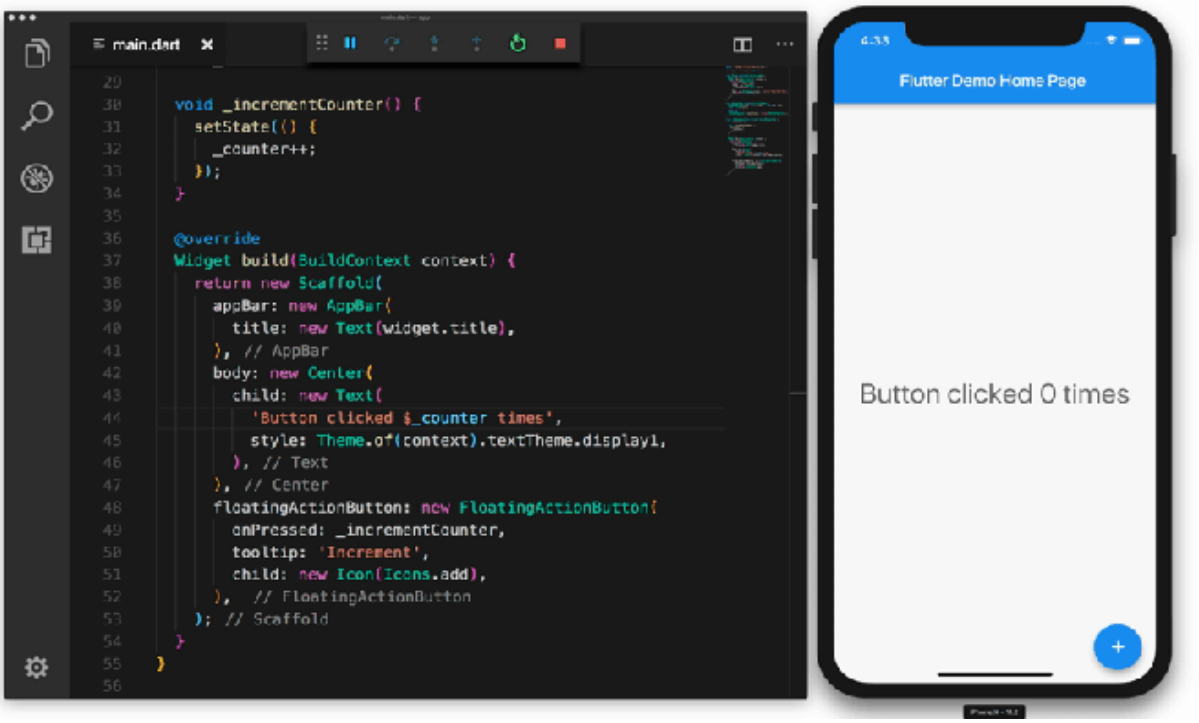
Je pourrais passer très longtemps à répondre à cette question, le **développement mobile est mon quotidien** et je baigne en permanence dans toute l'actualité des startups.

Mais pour faire simple, c'est parce que **Flutter** représente selon moi une véritable **révolution**...

Ce n'est pas qu'un simple **jeu de mot** pour nommer cette formation, c'est véritablement ce que j'ai **ressenti en testant pour la première fois Flutter**.

Flutter permet de développer **très rapidement** par rapport à Ionic ou React, il est directement orienté vers le développement **natif** sur émulateur **iOS et Android**.

En fait, Flutter est un **SDK** et non un framework comme React, ce qui veut dire qu'il s'adapte en fait aux environnements de développement d'Android et iOS via leurs logiciels respectifs.



Flutter avec un émulateur iOS s'actualise instantanément

Concrètement, avec **un seul code** (le **Dart**) et juste en enregistrant son fichier, on actualise son application sur iOS et Android.

Cette simple différence par rapport aux frameworks JavaScript est en fait monstrueuse, pas pour nos applications directement, mais pour la **rapidité de développement**.

On développe tout simplement **plus vite** sur Flutter que sur n'importe quel **autre framework** multiplateforme.

1.3 Pourquoi ce cours ?

J'ai donc décidé en **juillet 2020** de démarrer un nouveau cours débutant sur Flutter.

Pour aider la **communauté francophone** du monde entier à exploiter au maximum le superbe outil que représente Flutter.

Dans cette formation, nous allons reprendre toutes les **bases de Flutter** et du **Dart** en repartant vraiment **de zéro**.

Le but pour moi est de vous permettre d'aller **le plus loin possible** dans le développement de vos applications mobiles avec Flutter.

On commencera avec la prise en main des **bases** de Flutter et de la maîtrise de son **langage Dart**.

Je vous proposerai aussi plusieurs chapitres consacrés aux **interfaces utilisateurs** que propose Flutter qui sont à la pointe des **animations** sur smartphone.

Enfin, on terminera par un ensemble de **tutoriels ciblés** sur les **fonctionnalités** les plus utiles des applications modernes.

2. Qu’est-ce que Flutter ?

Flutter est en fait un **SDK** et non un framework de développement mobile. Qu'est-ce qu'un SDK par rapport à un framework et qu'est-ce que cela représente pour nos applications ?

Selon **Wikipédia**, la définition officielle d'un Framework est la suivante:

*“En programmation informatique, un **framework** désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d’une partie d’un logiciel.”*

https://fr.wikipedia.org/wiki/Framework

Concrètement, un **framework** comme Angular ou React permet de **créer de A à Z** une application mobile par exemple.

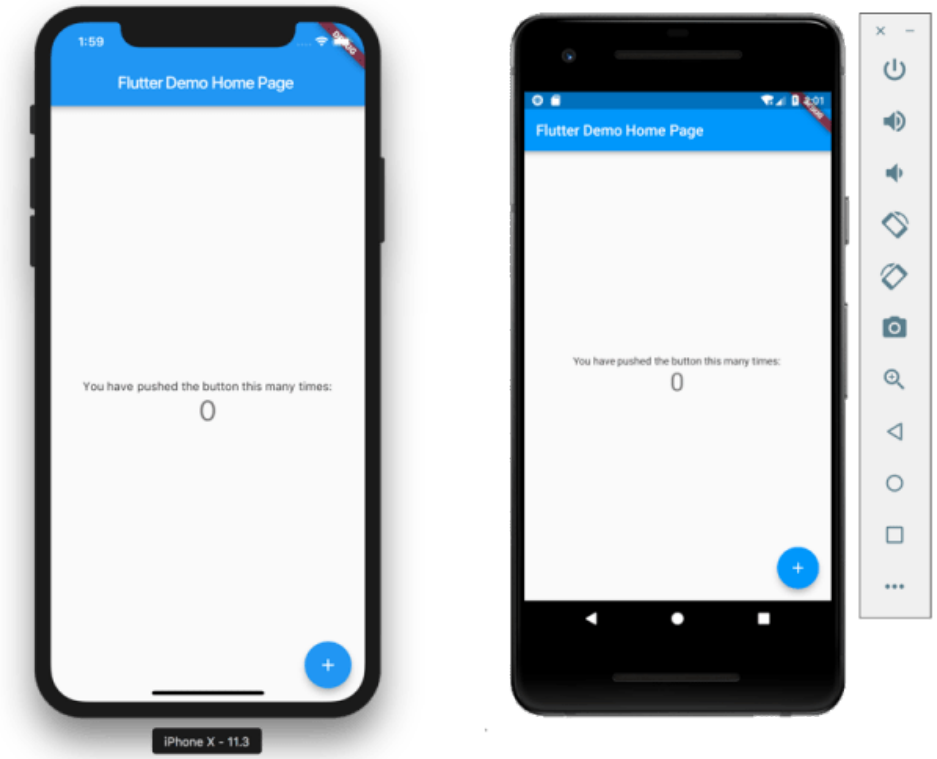
Il nous faut cependant passer par une **phase d'adaptation** aux appareils mobile lors de la création de chaque version iOS et Android.

Flutter en revanche est un **SDK**, ce qui veut dire littéralement **Kit de Développement** (*Software Development Kit*).

Google a conçu le **SDK Flutter** pour s'adapter aux deux environnements de développements iOS et Android: **Xcode et Android Studio**.

Le SDK Flutter vient donc s'installer sur votre machine en **complément** de ces deux logiciels (avec Android qui comprend aussi son propre SDK).

Donc Flutter nous permet de travailler depuis un **éditeur de code** et de compiler pour les deux plateformes en même temps.



Flutter permet de lancer deux émulateurs iOS et Android en même temps

Voyons donc maintenant quels sont les **points essentiels** à connaître sur l'utilisation de Flutter.

2.1 Sa rapidité de développement

Le premier **point essentiel** encore une fois de Flutter est la rapidité de développement qu'il nous offre.

En fait, Flutter nous offre une rapidité de développement que l'on retrouve avec le **développement natif** d'application.

Sur **Xcode** lorsqu'on code avec le **Swift** et qu'on teste son application en quelques secondes avec un simulateur **d'iPhone**.

Mais aussi sur **Android Studio** avec **Java ou Kotlin** lorsqu'on teste son application sur un émulateur **Android**.

En fait, avec Flutter, on utilise un **éditeur de code** (comme Visual Studio Code dont nous détaillerons la configuration) pour travailler sur nos fichiers **Dart**.

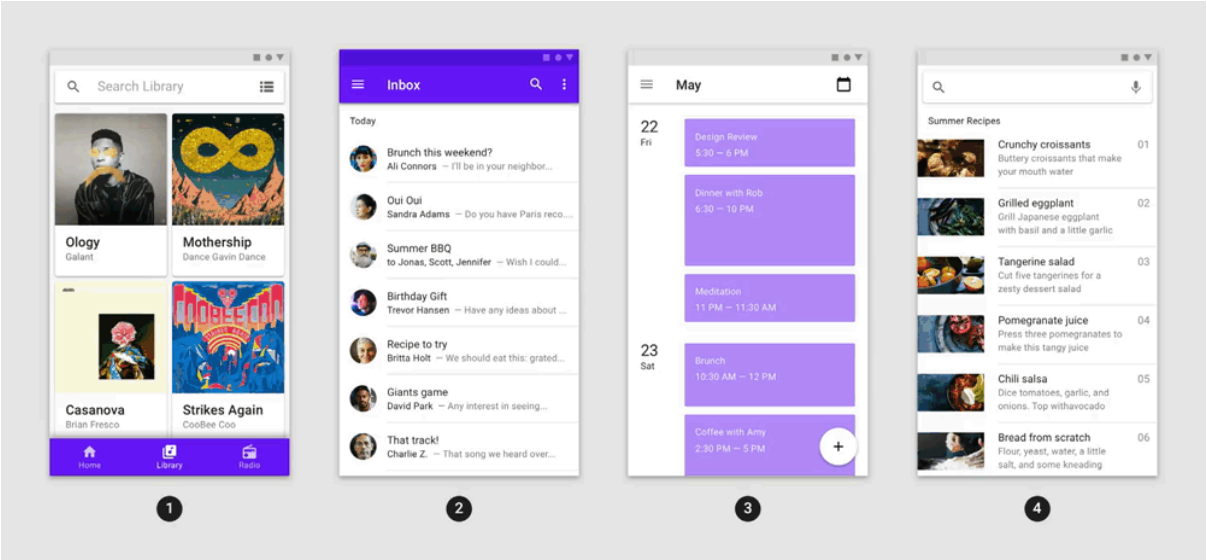
À chaque **enregistrement** de nos fichiers, grâce à l'extension Flutter de Visual Studio, notre **application s'actualise** sur notre émulateur.

2.2 Son interface utilisateur très flexible

Le deuxième point essentiel de Flutter est son **interface utilisateur** ou UI Design vraiment **révolutionnaire**.

En fait, la plupart des autres frameworks ou outils de développement mobile permettent de recréer des **animations modernes**.

Mais Flutter a vraiment mis une **priorité** sur ces animations pour les rendre très **accessibles**.



Nous verrons par exemple comment créer avec les **Widgets Flutter** toute sorte d'affichage pour nos applications.

Mais nous utiliserons aussi l'immense **bibliothèque d'animations** et d'interactions que Flutter propose pour **dynamiser** nos applications.

À la différence d'autres frameworks de développement mobile, ces interactions sont **facilement accessibles**.

C'est la raison pour laquelle la grande majorité des **applications** développées avec Flutter sont très **intuitives** et dynamiques.

Vous pouvez en retrouver quelques **exemples** sur le site de Flutter: https://flutter.dev/showcase

2.3 Des performances natives

Je vous ai parlé à plusieurs reprises de la rapidité de **développement** que propose Flutter, mais parlons maintenant de la **rapidité de nos applications** elles-mêmes.

Il est évident que **Google a conçu Flutter** pour proposer la meilleure rapidité d'exécution possible pour ses applications, mais à quel point une application peut-elle être **plus rapide ?**

En fait, quand on parle de **performances natives,** on parle notamment de l'accès aux fonctionnalités natives d'un smartphone.

Avec Ionic, nous utilisions **Cordova JS** pour accéder aux fonctionnalités natives d'un appareil mobile, comme **l'appareil photo** ou la **localisation**.

Avec Flutter, qui je vous le rappelle est un **SDK**, on accède beaucoup plus rapidement aux **fonctionnalités natives**.

Google prétend même que les performances sont **équivalentes au développement natif** d'application avec Swift et Java notamment.

À ce stade, c'est assez **difficile à vérifier**, car les derniers modèles d'iPhone sont déjà extrêmement performants et les différences de performance se calculent en millisecondes.

Même les applications de Facebook comme **Instagram** développées avec React proposent des excellentes performances.

Nous pouvons donc **faire confiance à Google** en ce qui concerne les performances et le maintien de celles-ci dans la durée.

2.4 Qui utilise Flutter ?

Le SDK Flutter est **le plus récent** de tous les outils logiciels du monde du développement mobile.

Comme je vous l'ai dit, je me suis spécialisé pendant des années sur le framework Ionic mais qui était lui-même basé sur **Angular**, un autre framework de **Google**.

L'avantage d'utiliser un **framework mature** est que l'on retrouve beaucoup de **ressources sur internet** pour nous faciliter l'apprentissage.

Sur **Flutter**, les cours en ligne et tutoriels en français restent **plutôt rares** pour le moment, d'où cette formation.

En ce qui concerne les **entreprises** qui utilisent Flutter, les plus connus se comptent pour le moment sur les doigts d'une main.

Vous retrouverez toutes les **startups** qui l'utilisent sur le site de Flutter: https://flutter.dev/showcase

Mais pas d'inquiétude, à la **fin de ce cours**, vous ferez peut-être partie des futures startups ou développeurs **sélectionnés par Google** pour mettre en avant Flutter 😉

3. Qu'est-ce que Dart ?

Contrairement à ce qu'on peut penser, le projet du langage **Dart** chez Google est **plus ancien** que le projet **Flutter**.

Il a été développé chez Google au départ pour combler les défauts du **JavaScript** et avait pour objectif de le remplacer en tant que langage phare du web.

Flutter n'est venu dans un deuxième temps pour trouver une **nouvelle utilité** au langage Dart resté inconnu du grand public.

Aujourd'hui, avec le langage Dart et Flutter, nous pouvons créer des **applications mobiles**, des **sites web** et des **logiciels de bureau**.

Dart est donc un langage très **puissant** et **polyvalent**, car il permet de gérer le côté **design** avec Flutter mais également la **connexion** avec **Firebase**.

Bref, le langage Dart vous offrira de nombreux **choix** en termes de développement et vous ouvrira aussi des portes pour votre **carrière**.

3.1 Le Dart et les autres langages

Le Dart, on va le voir, n'est pas si nouveau que ça, mais vient parfaitement **s'adapter aux besoins** des développeurs **d'applications**.

Le Dart est un **langage orienté objet** comme le sont la plupart des langages de développement logiciel (C#, Swift, ou Objective C).

On sera amené par exemple à le **comparer** au **JavaScript** qui fait tourner la majorité des autres frameworks multiplateformes.

Contrairement à Angular par exemple, on développe **quasiment 100%** de notre **code** avec le **Dart**.

Sur les frameworks JavaScript comme **Angular et React**, on développe avec les **langages web** classiques:

- **HTML**: Pour le contenu de notre application (texte, image, vidéo, boutons)
- **CSS**: Pour styliser nos pages et ajouter des animations
- **JavaScript**: Pour dynamiser nos applications et les connecter aux bases de données

Avec Ionic, on développe en HTML, CSS et JS

Avec Flutter, **98%** de votre application sera **codée en Dart**, qu'il s'agisse de la partie **design** ou de la connexion à vos bases de données Firebase.

C'est à la fois un **avantage et un inconvénient**, car dans le cas des langages web cela facilite l'apprentissage au début.

Le langage **HTML** par exemple est **très facile** à prendre en main, et on peut progressivement styliser nos pages avec le CSS puis les dynamiser avec le JS.

Avec Flutter et **Dart** c'est différent, on doit obligatoirement **comprendre** et **maîtriser** le Dart pour afficher du contenu.

L'avantage est qu'une fois le Dart maîtrisé, on peut **tout coder** avec ce même langage jusqu'à la fin du développement de notre application.

Nous allons maintenant parler plus en détail du **SDK du Dart** dont je vous laisse le site web: https://dart.dev/

3.2 Le SDK Dart

Tout comme Flutter, Dart est proposé par Google **sous forme de SDK** à télécharger et installer sur votre machine.

Il se télécharge **automatiquement** avec l'installation de **Flutter** sur votre ordinateur.

Nous détaillerons les **étapes d'installation** sur Mac et Windows dans les **prochains chapitres**.

Celui-ci contient toutes les **bibliothèques Dart** que nous utiliserons pour développer nos applications.

3.3 Quelques exemples du Dart

Reprenons **quelques exemples** de code Dart pour comprendre les spécificités de ce langage.

Ces exemples sont tirés de la **documentation de Google** sur le site https://dart.dev/#try-dart.

On commence avec la traditionnelle fonction **Hello Word** qui permet d'afficher ce texte dans la console:

```
void main() {
 print('Hello, World!');
}
```

Le langage Dart a toujours besoin d'une **fonction principale *main()*** pour exécuter le reste de son code.

On retrouvera ainsi pour chacun de nos codes Dart **une unique fonction** main() qui appelle d'autres fonctions à son tour.

Ici, on utilise la fonction d'affichage ***print()*** pour afficher dans la **console** le message: "*Hello, World!*".

On peut également afficher afficher d'autres **chaînes de caractères** avec une simple fonction print():

```
void main() {
 print('a single quoted string');
 print("a double quoted string");
 // Strings can be combined with the + operator.
 print('cat ' + 'dog');
 // Triple quotes define a multi-line string.
 print('''triple quoted strings
are for multiple lines''');
}
```

On utilisera généralement cette fonction ***print()*** pour **afficher** ou récupérer des **informations** à certains endroits de notre code pour vérifier son bon fonctionnement.

On peut également déclarer des **constantes** et les intégrer à notre fonction ***print()*** avec la syntaxe **$constante**:

```
void main() {
 // Dart supports string interpolation.
 const pi = 3.14;
 print('pi is $pi');
 print('tau is ${2 * pi}');
}
```

On finit par voir un exemple de déclaration de fonction en Dart pour **afficher** un **nombre** dans la **console**:

```
// Define a function.
void printInteger(int aNumber) {
 print('The number is $aNumber.'); // Print to console.
}

// This is where the app starts executing.
void main() {
 var number = 42; // Declare and initialize a variable.
```

```
 printInteger(number); // Call a function.
}
```

On déclare la fonction ***printInteger()*** en lui indiquant le type de paramètre à prendre (ici ***int*** représente les variables de types nombres).

Ensuite, dans notre fonction ***main()***, on peut faire appel à cette fonction ***printInteger()*** en lui indiquant un paramètre de type nombre.

4. Installer Flutter sur macOS

Dans ce chapitre, nous allons voir comment installer le **SDK Flutter** sur votre **Mac** de A à Z.

Une distinction est faite entre les **systèmes d'exploitation**, car les méthodes de téléchargement sont différentes entre Mac et PC.

Sur **MacOS**, vous avez la possibilité d'installer **Xcode** et **Android Studio** et vous serez donc capable de développer pour **iOS et Android**.

Sur **Windows,** le logiciel **Xcode** d'Apple n'est **pas disponible** et vous ne pourrez pas développer ou publier sur iOS.

Vous pouvez néanmoins faire par exemple tourner une **machine virtuelle** sur votre PC avec macOS.

Passons maintenant aux **étapes d'installation** de Flutter sur macOS dont je vous laisse la documentation: https://flutter.dev/docs/get-started/install/macos

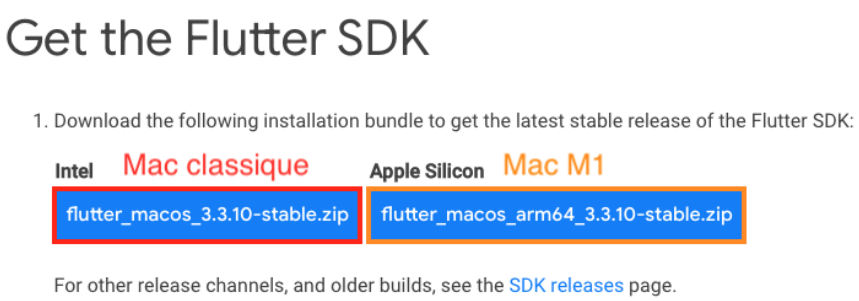
Dans la **vidéo**, je ferai quelques **pauses** pour traiter des **versions Mac M1** ou M2, avec les **puces Apple Silicon**.

L'installation est quasiment **identique**, et demande la plupart du temps de simplement **télécharger** une **version Apple Silicon** d'un logiciel.

4.1 Installer le SDK Flutter

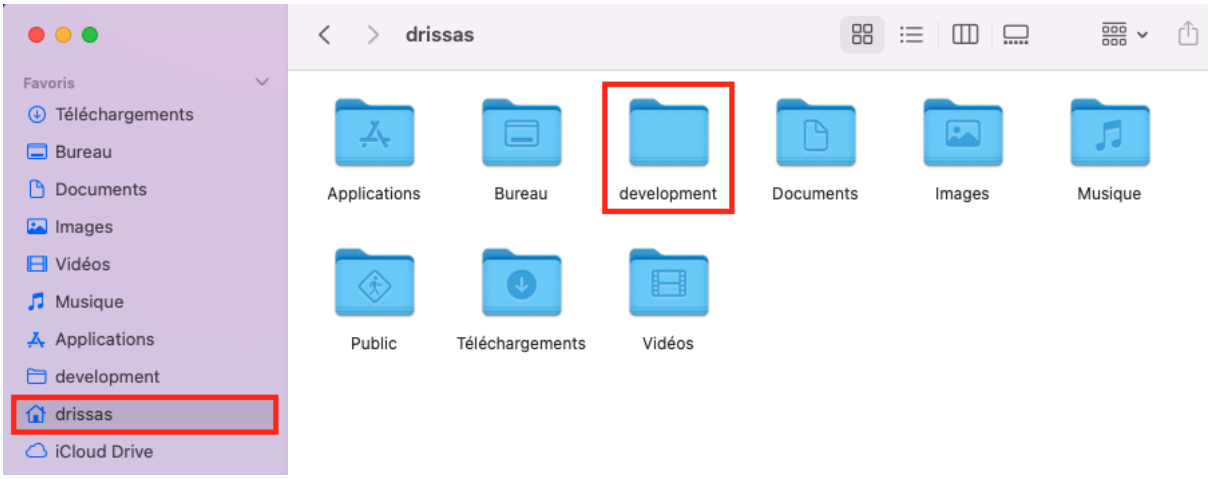
Commencez par **télécharger le fichier ZIP** contenant tout le SDK Flutter à installer sur votre ordinateur:
https://flutter.dev/docs/get-started/install/macos#get-sdk

Vous retrouverez dans cette section, **deux boutons de téléchargement** correspondant aux deux versions du **SDK Flutter** pour Mac:



Nous allons ensuite reprendre toutes les étapes de l'installation et suivre au maximum les **conseils** que propose la **documentation de Flutter**.

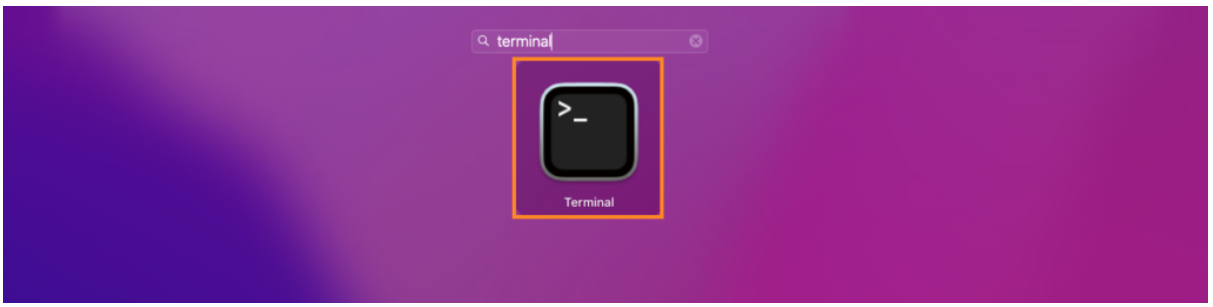
Ainsi, je vous invite à créer comme ils le recommandent un **dossier "*development*"** à la racine de votre **session utilisateur**:



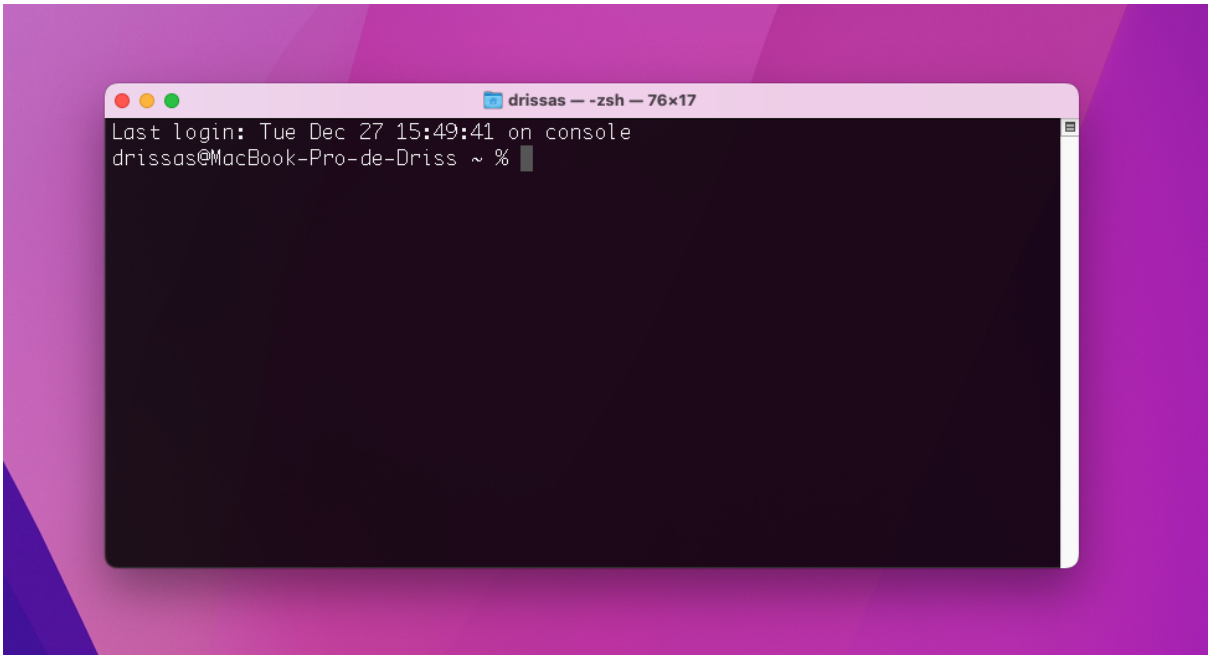
Par exemple, dans le dossier de ma **session** "drissas", je crée le dossier "***development***".

Ce nouveau dossier "development" contiendra le **SDK de Flutter** ainsi que toutes vos **futures applications** Flutter.

Pour **configurer** définitivement le **SDK Flutter** sur votre Mac, nous allons aussi utiliser le **Terminal**, cherchez donc le logiciel et ouvrez-le:



Une fois ouvert, votre terminal devrait s'ouvrir au **dossier racine** de votre **session utilisateur**:



Vous pouvez ensuite vous **rendre** dans le dossier ***development*** que vous devriez avoir créé précédemment:

```
cd development
```

Une fois situé dans le dossier ***development*** dans votre terminal, nous allons pouvoir **dézipper** le fichier téléchargé sur le site de Flutter.

Vous pouvez le **dézipper** manuellement ou utiliser la **commande** suivante pour dézipper le fichier ***flutter_macos_3.3.10-stable.zip*** (*attention le nom du fichier change avec les versions*):

```
unzip flutter_macos_3.3.10-stable.zip
```

Prenez soin auparavant de situer votre **fichier zip** dans le **dossier *development***.

Pour les **Mac M1** équipés des puces Apple Silicon, vous pouvez également entrer la **commande suivante**:

```
sudo softwareupdate --install-rosetta --agree-to-license
```

Cela vous permettra d'installer **Rosetta** qui est un **processus de traduction** pour exécuter et de convertir des **instructions Intel** sur les **Mac M1**.

Voilà pour **l'installation** du SDK Flutter, nous allons maintenant faire en sorte de le **configurer** pour un fonctionnement optimal.

4.2 Configurer le SDK Flutter

Nous venons donc **d'installer le SDK Flutter** sur notre Mac et nous avons ainsi accès aux **commandes Flutter** pour créer de nouvelles applications par exemple.

Néanmoins, nous avons besoin de **configurer l'adresse de notre SDK** pour qu'il soit accessible partout sur notre Mac.

Le **SDK** se trouve précisément dans le **dossier "*bin"*** de notre **dossier "flutter"**, nous allons donc stocker dans notre Mac **l'adresse exacte** de ce dossier.

On utilise la **commande '*pwd*'** pour **trouver** l'adresse complète du dossier '***flutter/bin***'.

Vous pouvez ainsi utiliser les **commandes** ci-dessous pour déclarer et **afficher** cette **nouvelle adresse**:

```
export PATH="$PATH:`pwd`/development/flutter/bin"
echo $PATH
```

Copiez et mettez l'adresse affichée par le terminal **de côté** pour pouvoir la **réutiliser** plus tard.

Je vais vous proposer **deux méthodes** pour déclarer votre **variable globale** d'accès au SDK Flutter.

4.2.1 Méthode 1: zsh (recommandée)

Je commence par vous proposer la **méthode** la **plus récente**, qui utilise **l'interpréteur de commande zsh** pour Mac.

Reprenez votre **terminal** et entrez la commande:

```
vim $HOME/.zshrc
```

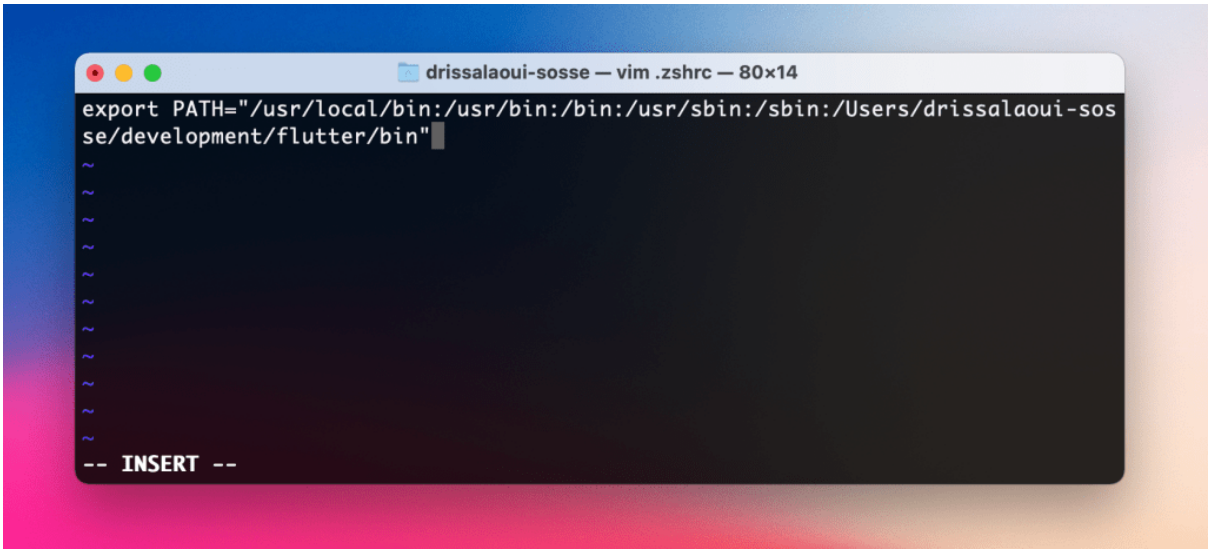
Dans le **fichier** qui s'affiche dans votre terminal, vous allez pouvoir **déclarer l'adresse du SDK Flutter** et pouvoir ainsi y accéder depuis tout votre ordinateur.

Pour **modifier** votre fichier, tapez sur la **touche [I]** et vous devriez voir apparaître le texte ***-- INSERT --*** tout en bas.

Reprenez donc **l'adresse** de votre dossier ***flutter/bin*** et **déclarez la variable** suivante:

```
export PATH="L_ADRESSE_DU_DOSSIER_FLUTTER_BIN"
```

Voilà par **exemple** ce que donne le rendu de mon **fichier zshrc:**



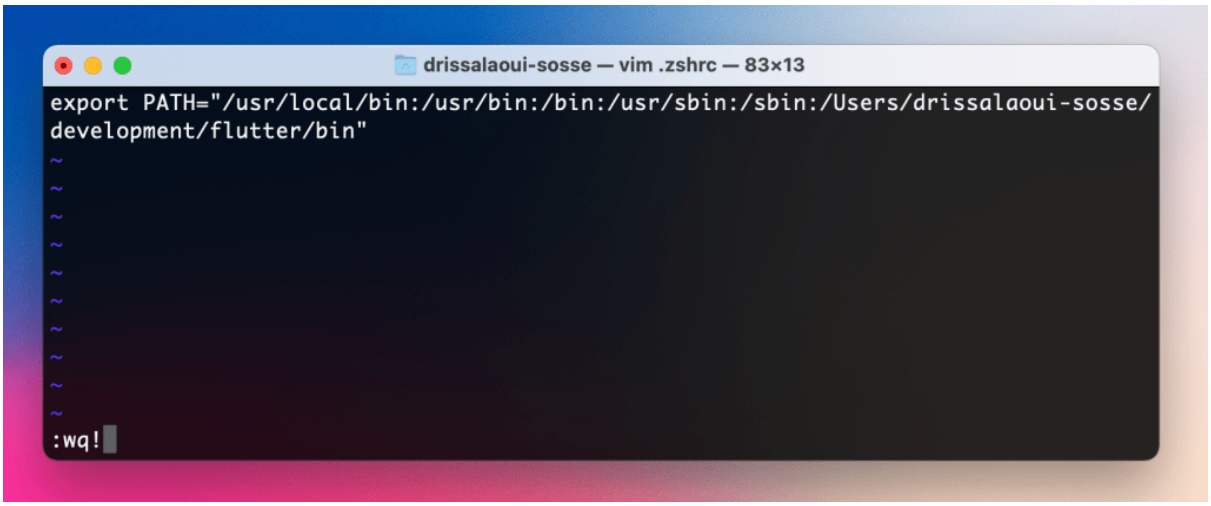
Pour **quitter** l'édition du fichier, entrez:

```
Ctrl + C
```

Puis tapez dans le terminal les caractères suivants pour quitter et **enregistrer votre fichier**:

```
:wq!
```

Cette commande (**write and quite**) vous permet de bien enregistrer votre fichier, devrait être visible en bas de votre terminal:



Une fois revenu à votre terminal habituel, n'oubliez pas de **REDÉMARRER** votre logiciel avant de tester que la **méthode zsh** a bien **fonctionné**.

Pour tester que votre **variable PATH** a bien été mise à jour, vous pouvez entrez la commande suivante pour **l'afficher**:

```
echo $PATH
```

Vous pouvez aussi **tester** par exemple la commande ***flutter doctor*** pour vérifier si la commande flutter est accessible (ne faites pas encore attention au résultat de la commande):

```
flutter doctor
```

Je vous montre maintenant la **deuxième méthode,** pour ceux que ça intéresse.

4.2.2 Méthode 2: bash (ancienne méthode)

Si vous le souhaitez, vous pouvez aussi utiliser la **méthode bash** qui est similaire, mais **moins recommandée** pour Mac à ce jour.

Nous allons utiliser le fichier **bash_profile** pour stocker cette adresse de manière définitive.

Entrez les **deux commandes** suivantes pour **créer** puis **ouvrir** le fichier **bash_profile** (vous devez vous trouver dans votre dossier racine utilisateur):

```
touch .bash_profile
open .bash_profile
```

Un **document texte** vierge devrait s'ouvrir, vous pouvez alors lui **entrer** la ligne de code suivante:

```
export PATH=L_ADRESSE_DU_DOSSIER_FLUTTER_BIN
```

Par **exemple** voilà ce que j'entre dans **mon fichier** bash_profile:

```
export
PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Users/dris
salaoui-sosse/development/flutter/bin
```

Enregistrez ensuite votre fichier texte et **quittez l'éditeur** TextEdit, entrez ensuite la **commande source** suivante pour appliquer les modifications.

```
source .bash_profile
```

Vous pouvez ensuite **tester** par exemple la commande ***flutter doctor*** pour tester si la commande flutter est accessible (ne faites pas encore attention au résultat de la commande):

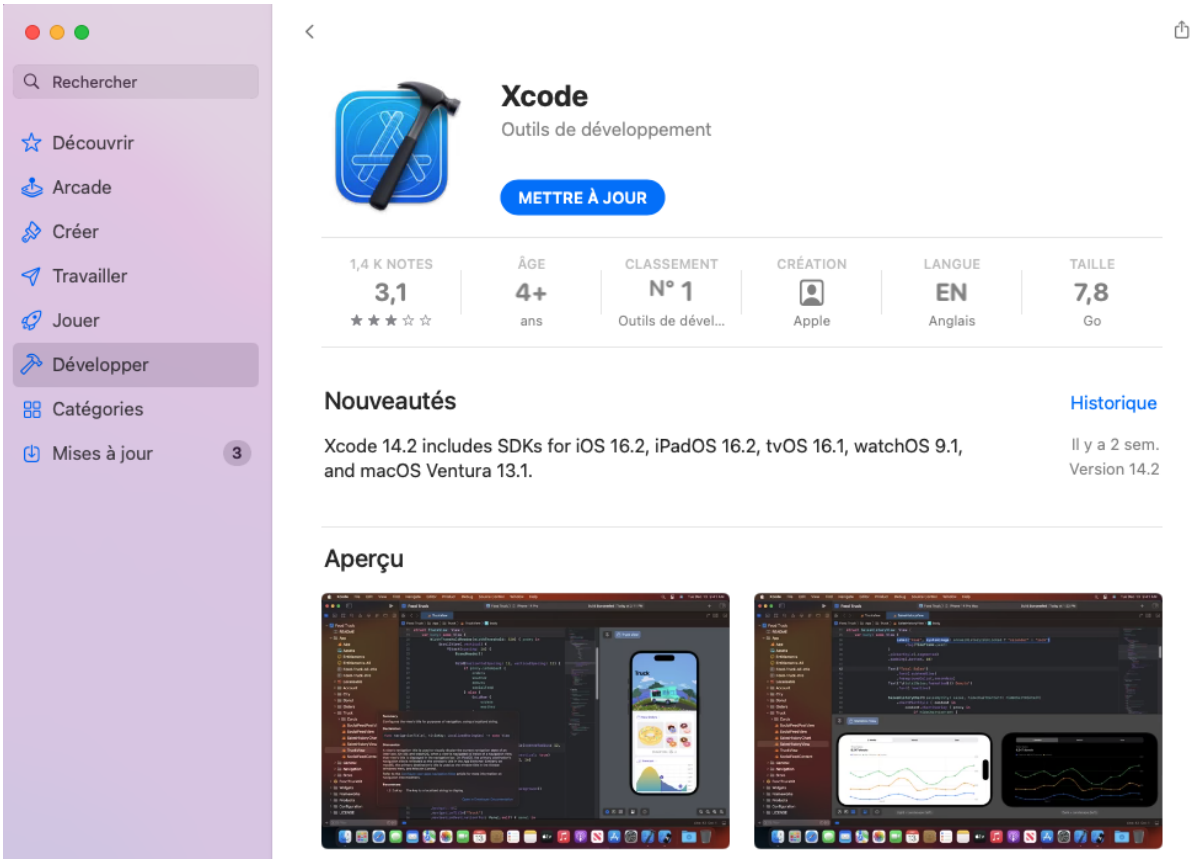
```
flutter doctor
```

On passe maintenant à la **configuration de Xcode** et du lancement de notre émulateur d'iPhone.

4.3 Configuration de Xcode

Nous allons à présent **installer** le logiciel de développement d'Apple **Xcode** pour accéder aux **émulateurs d'iPhone**.

Vous pouvez **télécharger Xcode** gratuitement en vous rendant sur le **Mac App Store**: https://apps.apple.com/fr/app/xcode/id497799835?mt=12



Xcode est un logiciel assez **volumineux** qui demande chaque année une **mise à jour**.

Votre version de **macOS** doit aussi **être à jour** pour permettre à Xcode de vous faire utiliser les dernières **versions d'iOS**.

Si ce n'est pas déjà fait, mettez donc **à jour** votre version de **macOS** et installez **Xcode**.

Une fois le logiciel installé sur votre Mac, entrez les **commandes** suivantes pour **configurer Xcode** sur votre terminal:

```
sudo xcode-select --switch
/Applications/Xcode.app/Contents/Developer
sudo xcodebuild -runFirstLaunch
```

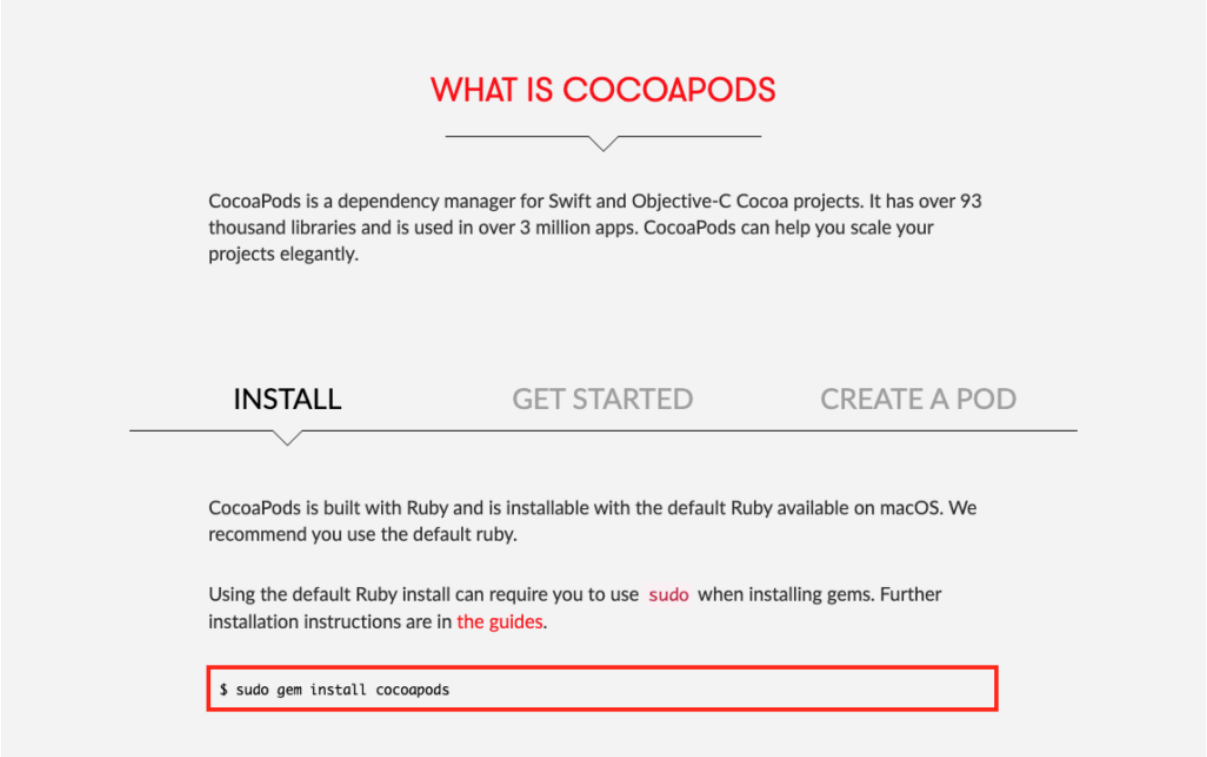
Le mot clé '**sudo**' vous indique que vous devez entrer votre **mot de passe** dans votre terminal pour valider cette action.

Une fois ces deux commandes entrées, vous pouvez lancer un premier **simulateur d'iPhone** avec la commande **open**:

```
open -a Simulator
```

Vous pouvez **changer d'appareil iOS** en faisant un **clic-droit** sur l'icône de votre émulateur dans le dock et sélectionnez par exemple un **iPhone 14 Pro Max**.

Je vous propose également une étape complémentaire qui consiste à installer le **gestionnaire de dépendance** pour iOS, **CocoaPods**: https://cocoapods.org/



Il vous suffit d'entrer la **commande suivante** pour télécharger **CocoaPods** sur macOS qui nous sera indispensable pour le développement sur **iOS**:

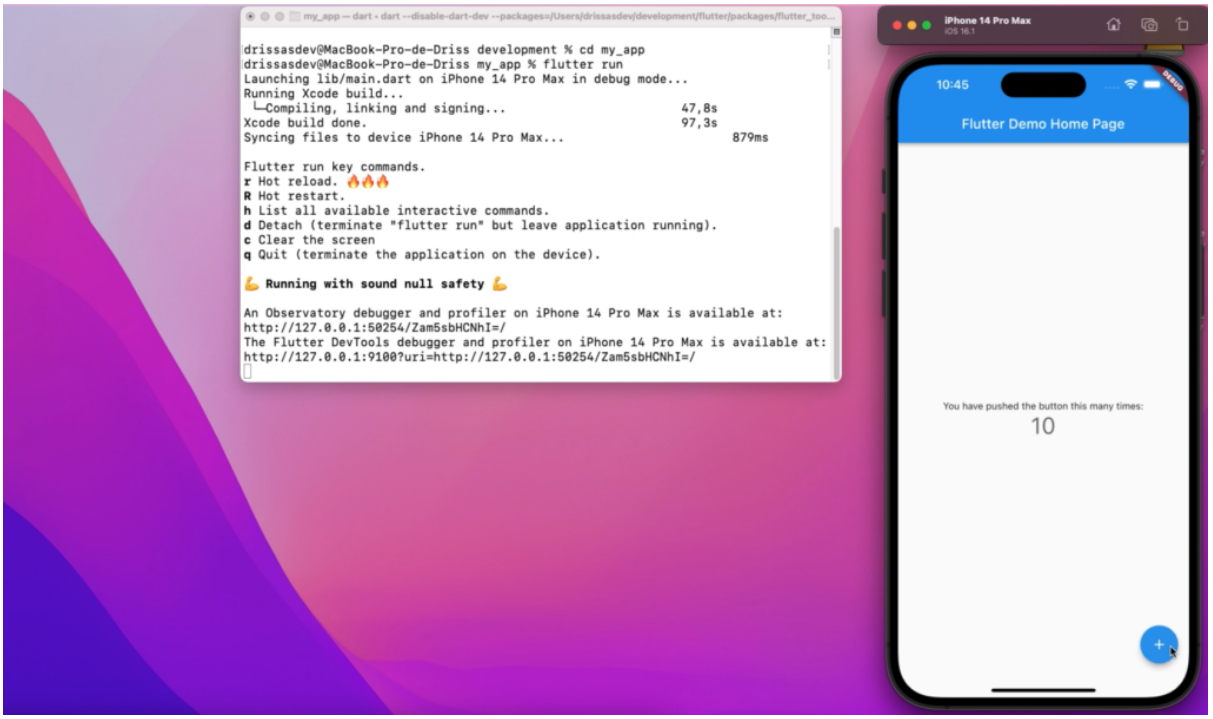
```
sudo gem install cocoapods
```

Rassurez-vous, nous n'aurons que rarement **besoin de le manipuler** directement, **Flutter** prendra en compte l'essentiel du travail.

Si vous voulez **créer** une **première application** rapidement et la **lancer** dans votre émulateur, entrez les **commandes suivantes**:

```
flutter create my_app
cd my_app
flutter run
```

Voilà alors le **résultat** de votre émulateur iOS qui fonctionne et avec lequel nous allons pouvoir tester nos **applications Flutter**:



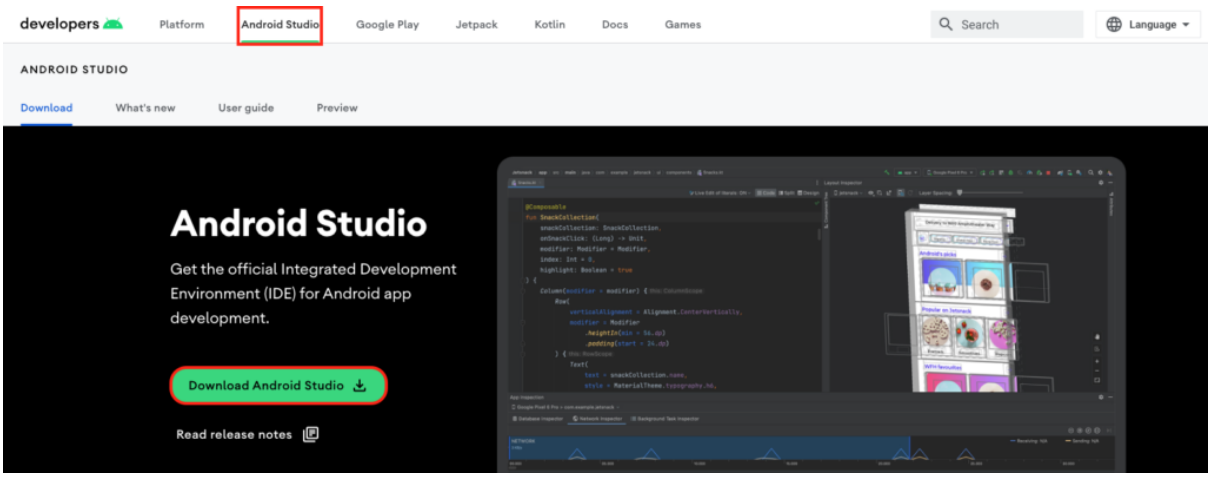
Nous détaillerons toutes ces **étapes** dans la leçon dédiée à **Visual Studio Code** et toutes les plateformes disponibles.

4.4 Configurer Android Studio

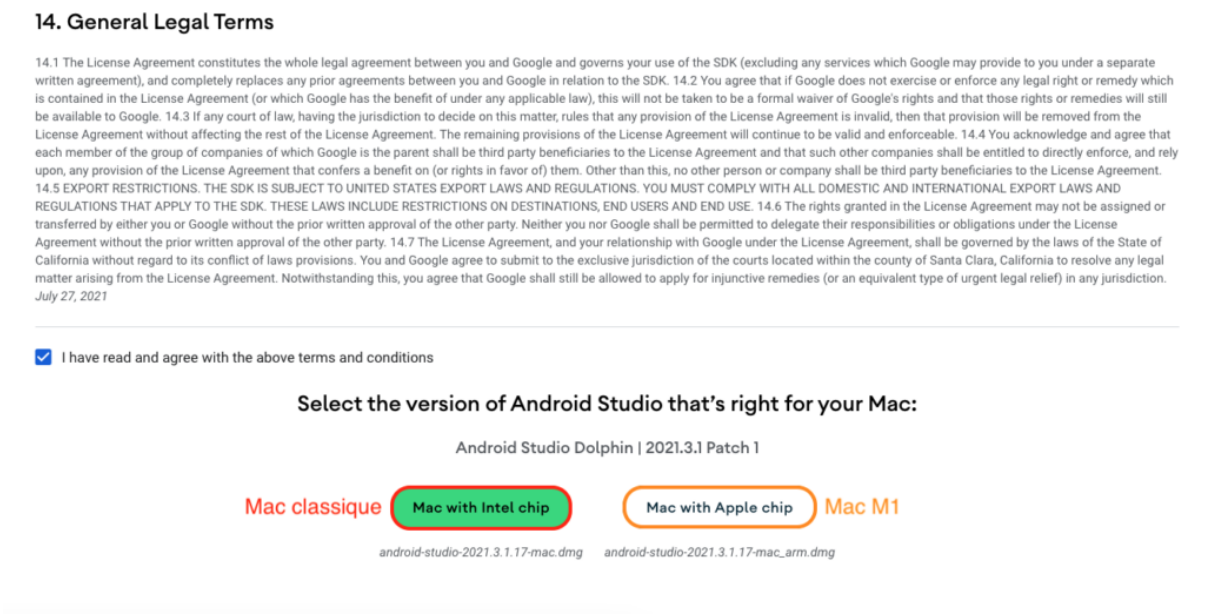
La configuration d'**Android Studio** est plus **rapide**, mais **pas moins technique** que pour Xcode.

Je vous invite encore une fois à vous référer en parallèle à cette leçon à la **documentation de Flutter:**
https://flutter.dev/docs/get-started/install/macos#android-setup

Vous pouvez notamment commencer par **télécharger Android Studio** sur Mac à cette adresse: https://developer.android.com/studio



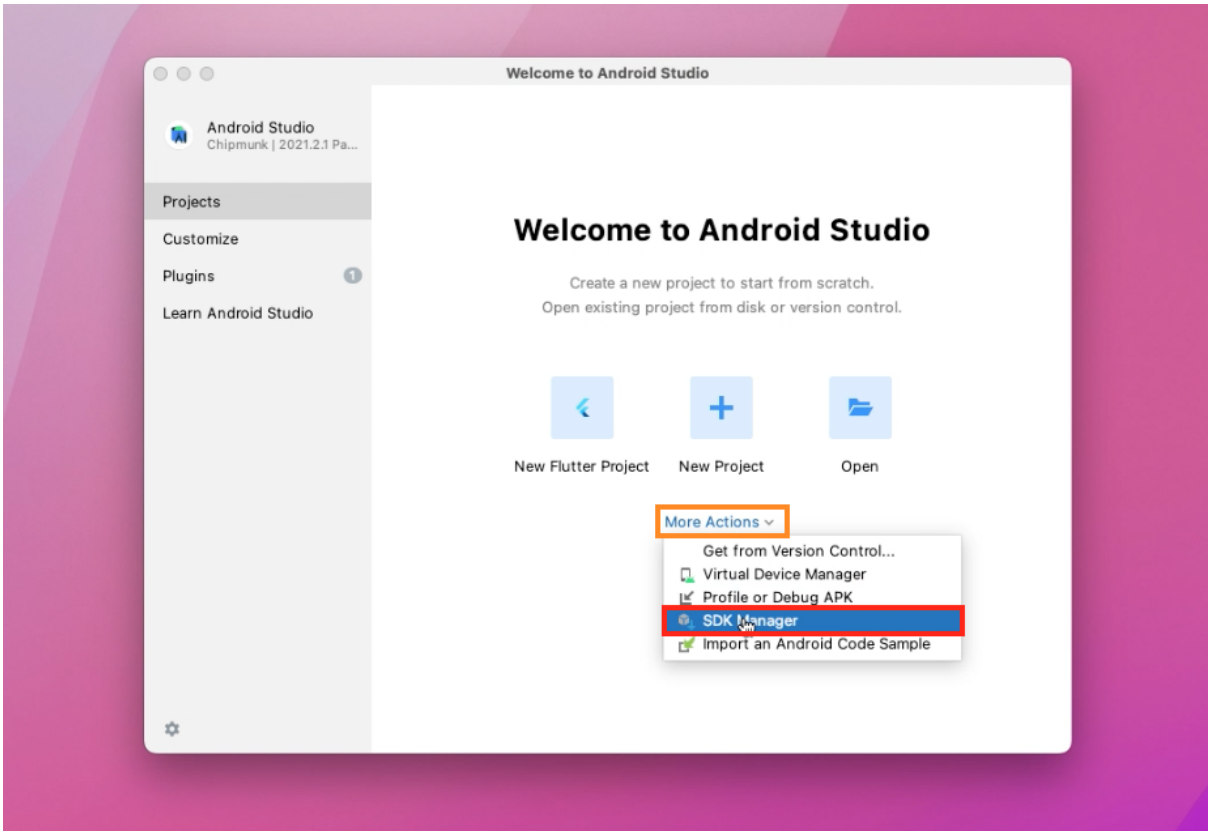
Choisissez alors la **version du logiciel** à télécharger, **Mac classique** ou **Mac M1**:



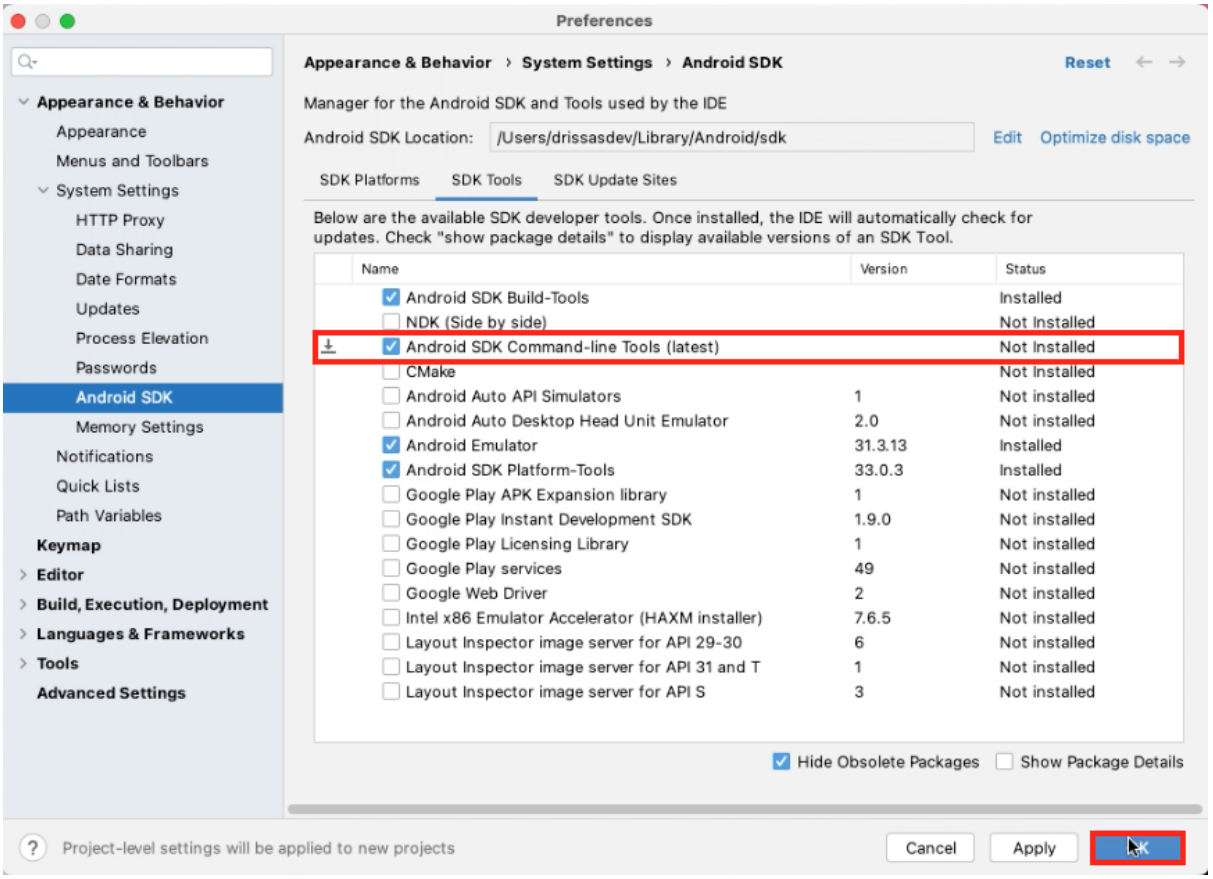
Lancez ensuite le **fichier téléchargé** pour démarrer **l'installation** d'Android Studio.

Vous devriez voir également les différents **SDK supplémentaires** qui vont être **installés** par la même occasion.

Une fois installé, **lancez Android Studio** une première fois et ouvrez les paramètres **SDK Managers**:

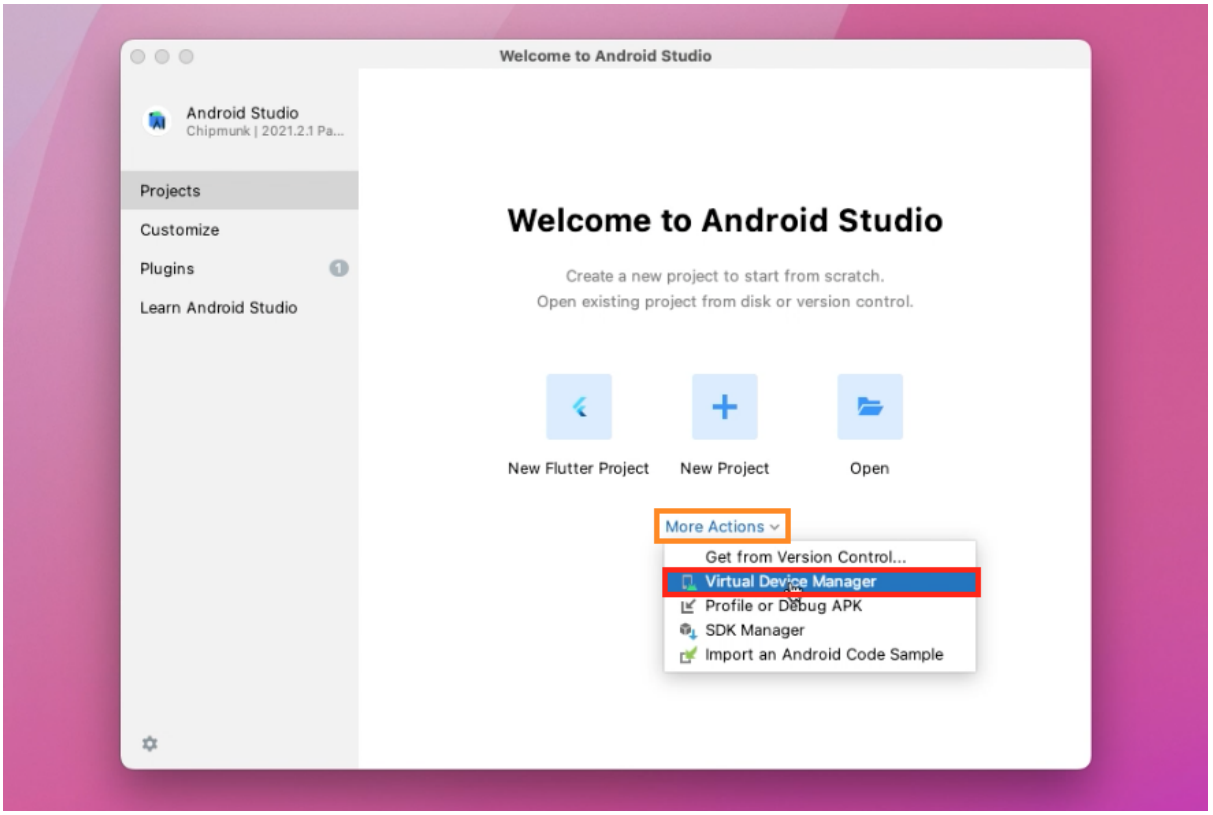


Vous pouvez de cette manière **vérifier** la présence du **SDK Android Build Tool** et surtout ajouter le "**Android SDK Command-line Tools**" qui est indispensable:

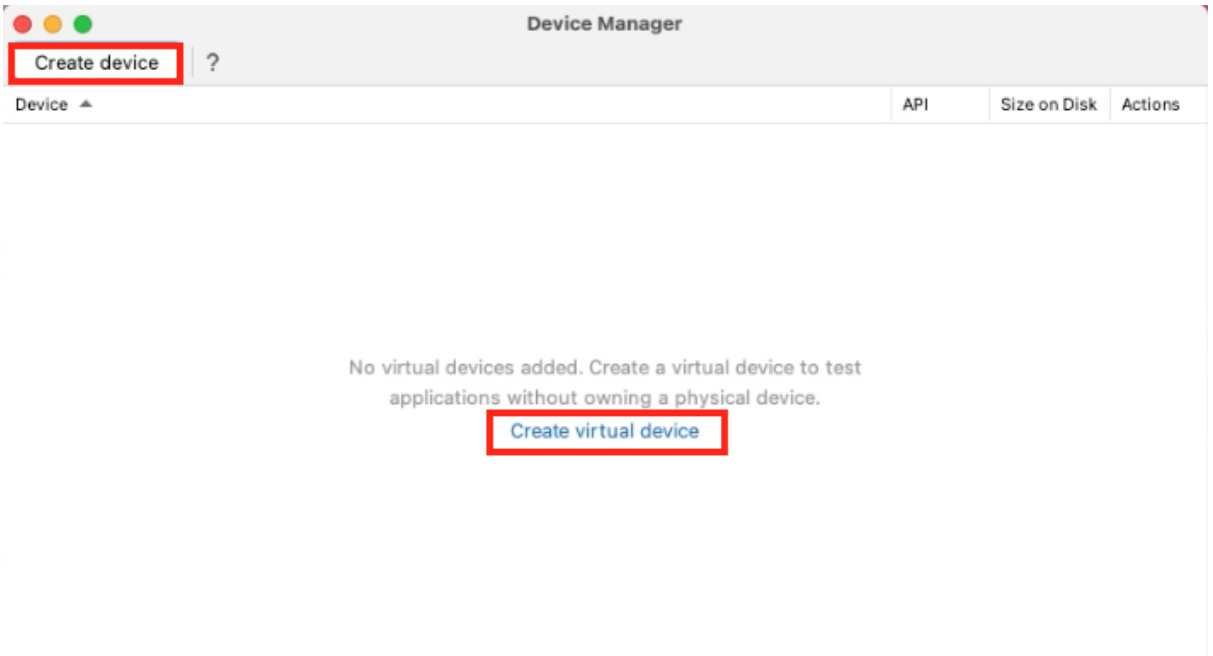


Si jamais il **manque** l'un des outils ou SDK, **cochez-le pour l'installer** sur votre Mac.

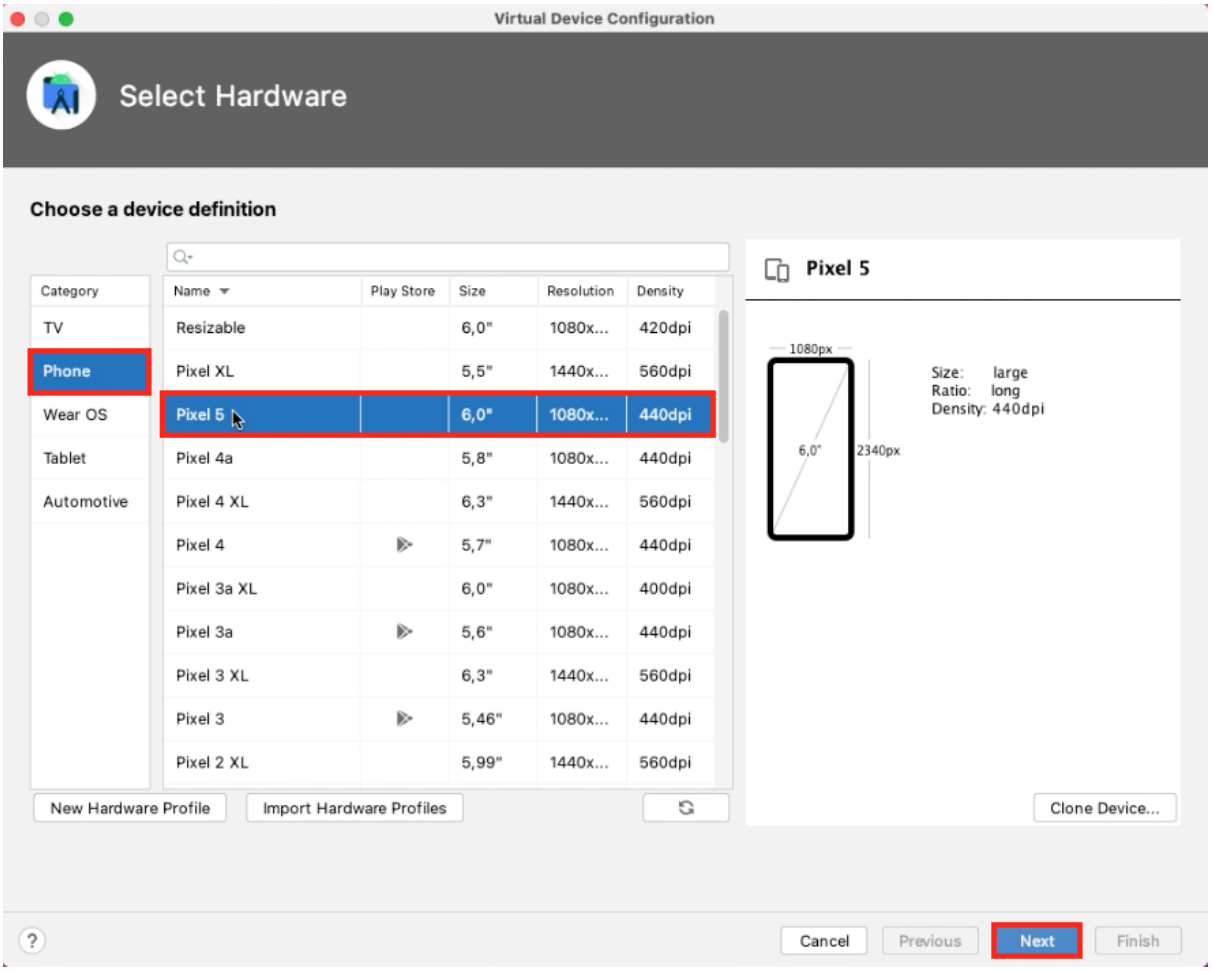
Passons maintenant à la **configuration** d'un **premier émulateur Android**, pour cela, ouvrez le menu **Virtual Device Manager**:



L'interface vous proposera alors de **créer un nouvel appareil virtuel**:



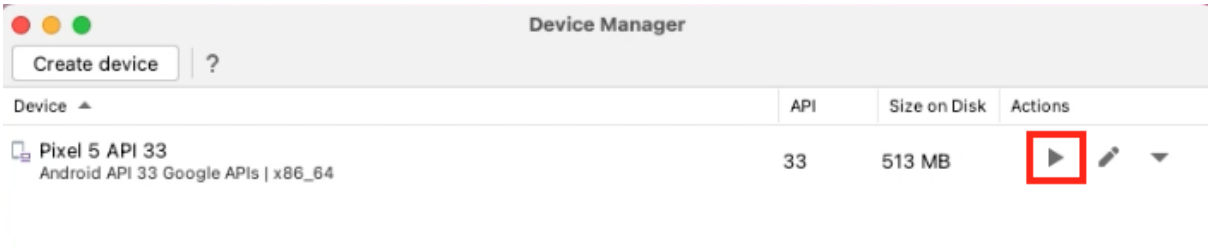
Sélectionnez dans la **catégorie Phone** le modèle de votre choix, par exemple un **Pixel 5** de Google:



Installez et téléchargez la **dernière version d'Android** sur cet émulateur pour terminer la configuration de cet appareil.

Une fois **l'émulateur** Android **téléchargé** et installé, il devrait apparaître dans le tableau de vos **machines virtuelles**.

Vous pouvez cliquer sur le **bouton Play** pour **lancer** cet émulateur Android.



Un **Pixel 5** devrait donc **apparaître** sur votre Mac, dans lequel nous allons pouvoir **lancer notre première application** Flutter.

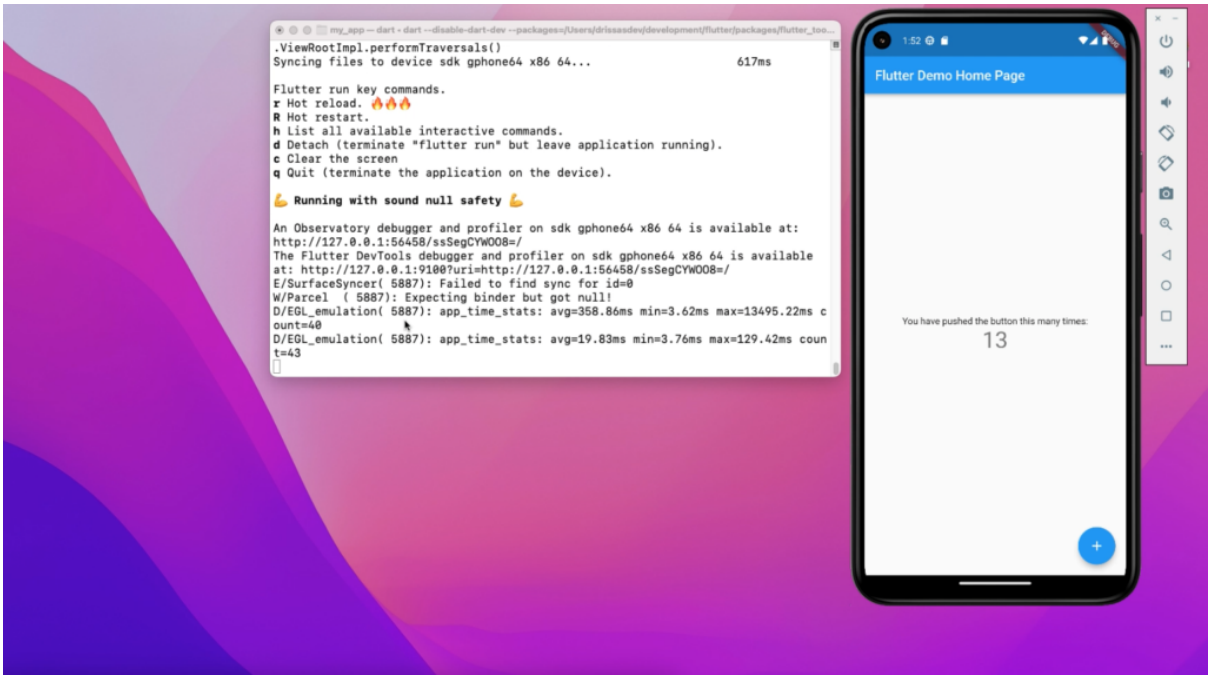
Reprenez votre **terminal** et entrez la commande suivante pour **accepter les licences Android:**

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

Si vous n'avez pas encore créé d'**application Flutter**, vous pouvez en **créer une rapidement** et la lancer dans votre **émulateur Android** avec les commandes suivantes:

```
flutter create my_app
cd my_app
flutter run
```

Votre **application Flutter** devrait ainsi se lancer dans un **Pixel 5** avec l'apparence et le design d'une **application Android** classique.



Voilà pour **l'installation de Flutter** sur votre **Mac**.

5. Installer Flutter sur Windows

Dans ce tutoriel, nous allons voir comment **installer Flutter** sur un PC **Windows**.

La procédure est légèrement **différente** que sur un **Mac**, notamment en ce qui concerne la déclaration des **variables d'environnement**.

En ce qui concerne **Android Studio,** le logiciel est **identique** sur Windows et Mac, donc pas de différence majeure à ce niveau-là.

En revanche, sur Windows, vous n'avez **pas accès** au **logiciel Xcode** et donc au développement sur **iOS**.

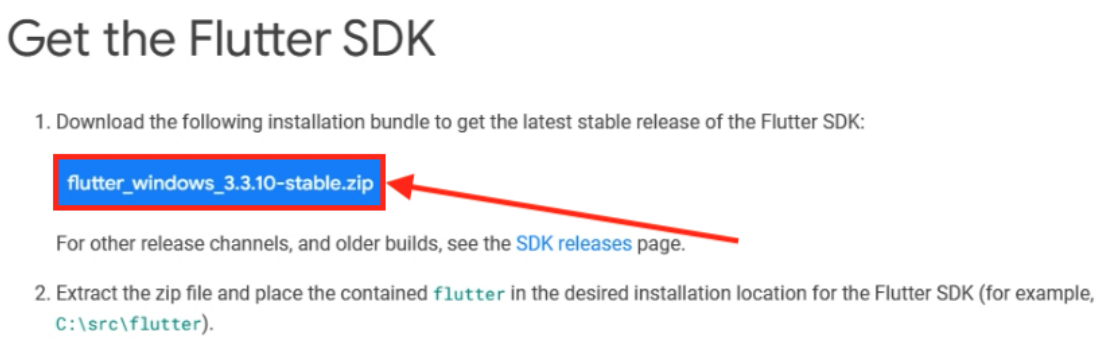
Vous ne pourrez pas tester vos applications sur un émulateur **d'iPhone** ou **d'iPad**.

Pour cela, vous devrez soit **installer macOS** sur votre PC ou le faire tourner sur une **machine virtuelle** avec un logiciel adapté.

5.1 Installer le SDK Flutter sur Windows

Pour commencer, nous allons télécharger et installer le **SDK Flutter** sur notre PC **Windows**.

Vous pouvez vous rendre sur le **site de Flutter** pour **télécharger** la dernière version du SDK:
https://flutter.dev/docs/get-started/install/windows#get-the-flutter-sdk



Vous pouvez également utiliser la **commande git** si vous l'avez installé sur votre ordinateur pour télécharger la dernière **version** du SDK Flutter.

```
git clone https://github.com/flutter/flutter.git -b stable
```

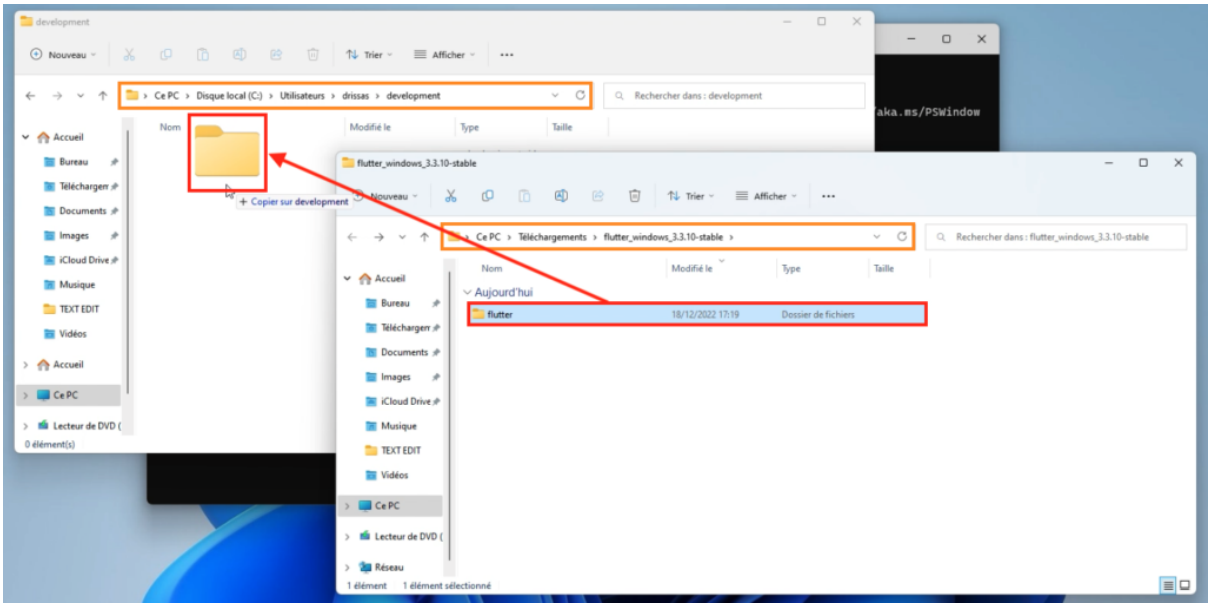
Je vous invite, quelle que soit la méthode que vous utilisez, de placer le **SDK Flutter** dans un nouveau **dossier** "***development***".

Pour créer ce nouveau dossier à la racine de votre session utilisateur, ouvrez le logiciel "**Console**" ou "**Invite de commande**".

Une fois ouvert, entrez la commande ***mkdir*** pour **créer le dossier** development et ***start*** pour **l'ouvrir** dans votre explorateur de fichier.

```
mkdir development
start development
```

Une fois que ce dossier ***development*** est ouvert, je vous invite à **transférer le contenu dézippé** du SDK Flutter dans ce nouveau dossier.



Maintenant que votre **SDK Flutter** se trouve dans votre dossier ***development***, vous pouvez vous rendre dans le sous dossier ***flutter/bin***:

```
cd development/flutter/bin
```

C'est en effet dans ce **sous-dossier** que se trouve notre **commande flutter** qui nous permettra de créer et déployer nos applications.

Vous pouvez par exemple tester la **commande d'analyse** "***flutter doctor***" qui vous indique si tout est bien installé sur votre ordinateur.

```
flutter doctor
```

Pour le moment, la console devrait vous indiquer **l’absence du SDK Android** sur votre PC.

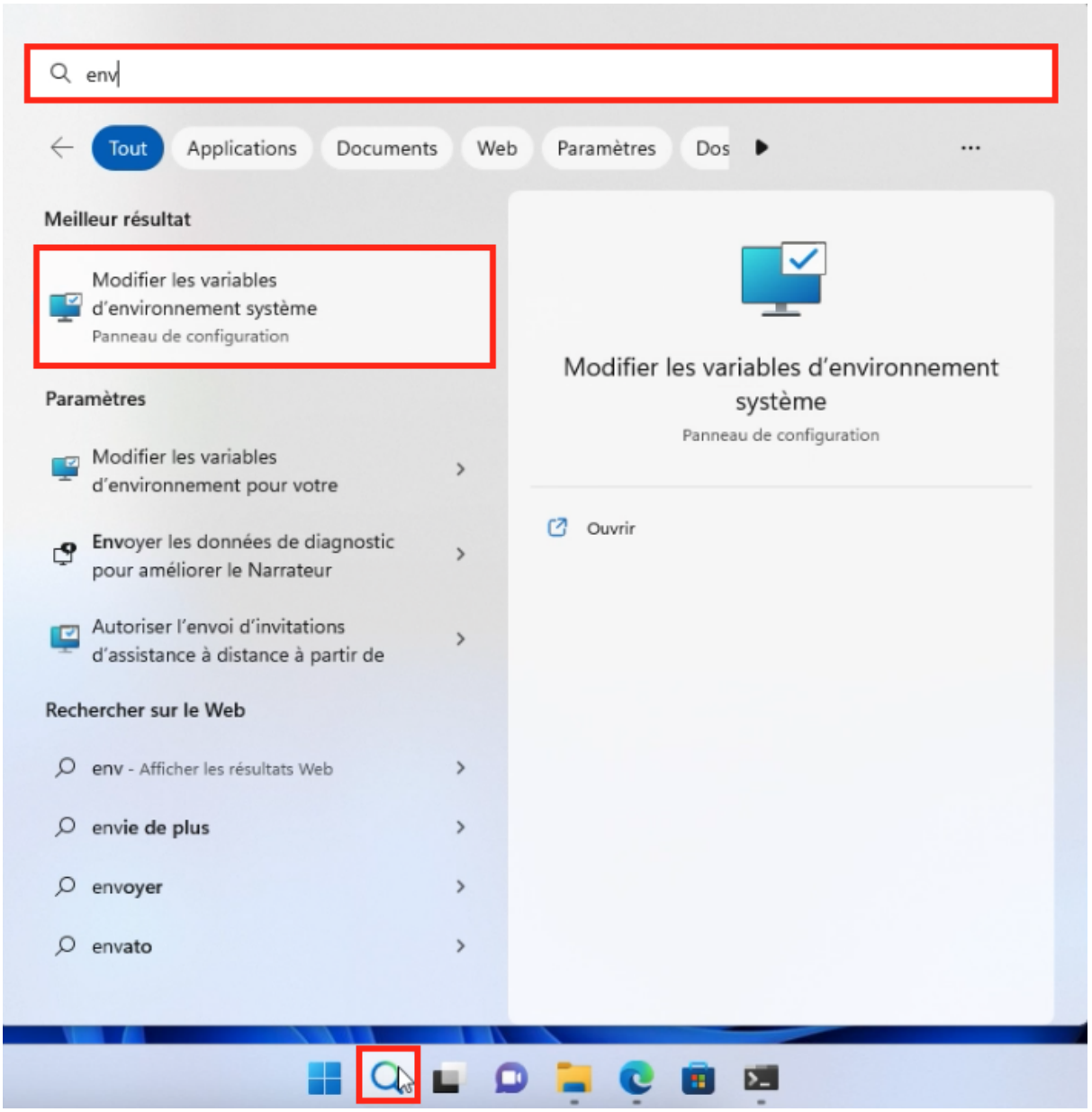
5.2 Configurer le SDK Flutter sur Windows

Nous sommes donc capables d'utiliser la **commande Flutter** et accéder à tout le SDK.

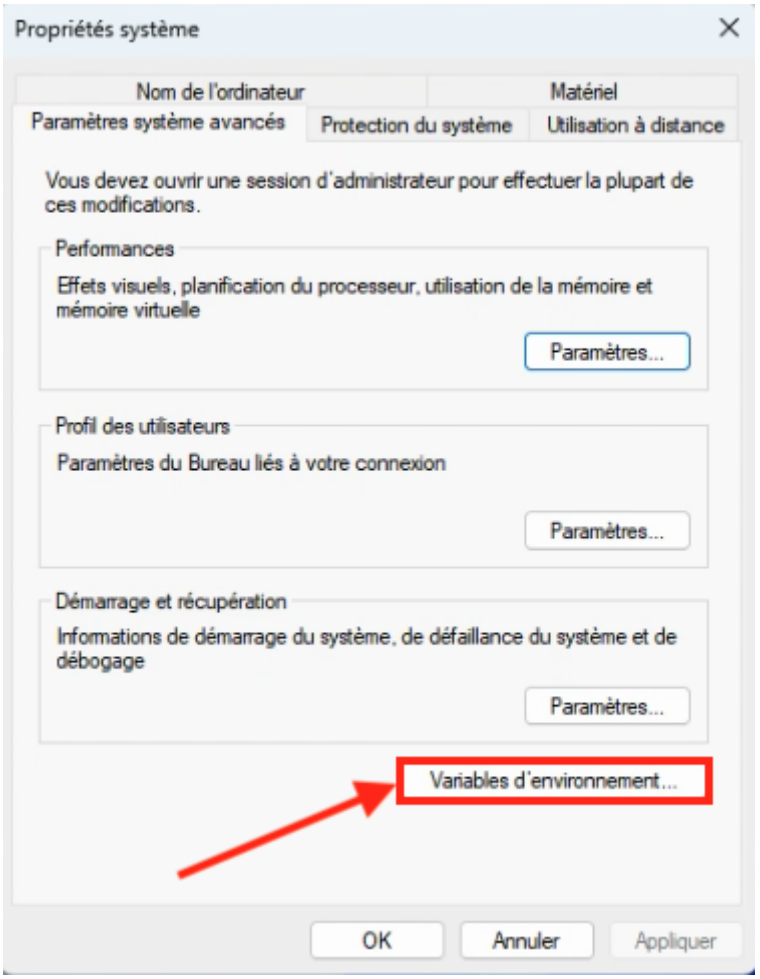
Mais pour l'instant, nous ne pouvons le faire qu'**en nous rendant** dans ce **sous-dossier bin**.

Nous pouvons déclarer et mettre à jour nos **variables d'environnements** pour pouvoir accéder à la commande Flutter depuis **n'importe où**.

Pour cela, tapez dans la barre de recherche Windows "***env***" et cliquez sur le résultat "***Modifier les variables d'environnement système***":



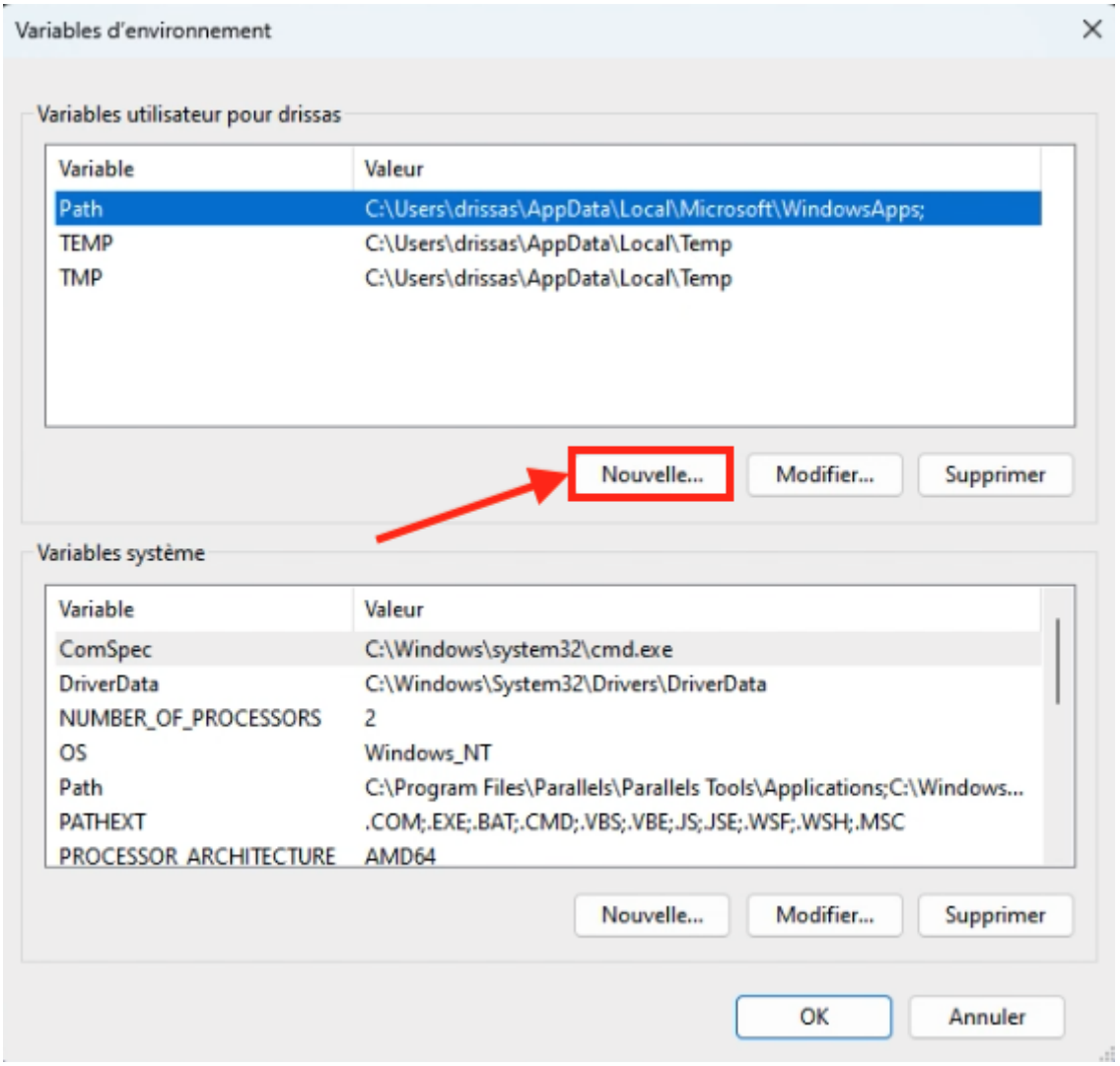
Une fois le **panneau de configuration** ouvert, cliquez sur le bouton "***Variables d'environnement***" en bas de la page:



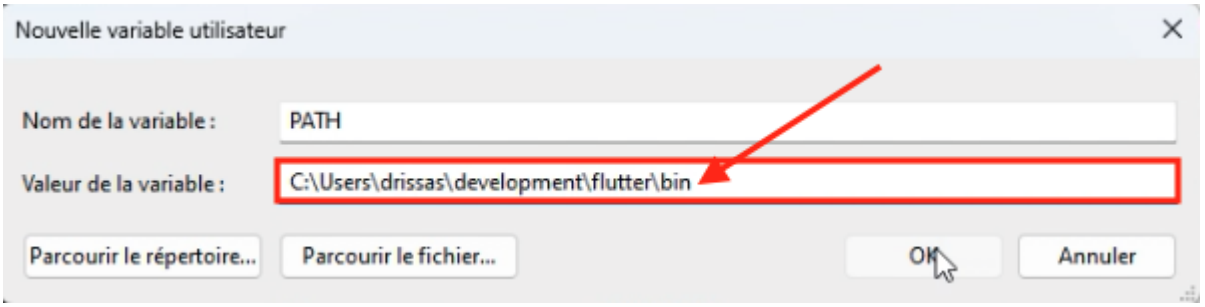
Il devrait alors vous afficher les **variables d'environnement**, déjà présentes sur votre PC.

Si la variable **PATH** ou **Path** existe déjà, cliquez dessus et sur **Modifier** pour la mettre à jour.

Si elle **n'apparaît pas encore**, cliquez sur "**Nouvelle**":



Indiquez alors le nouveau de votre variable “**PATH**” et donnez-lui l’adresse de votre **répertoire flutter/bin** de votre disque:



Cliquez ensuite sur **OK** et fermer votre panneau de configuration et **RELANCER votre terminal**.

Lorsque vous entrerez la **commande** suivante **echo** suivante:

```
echo %PATH%
```

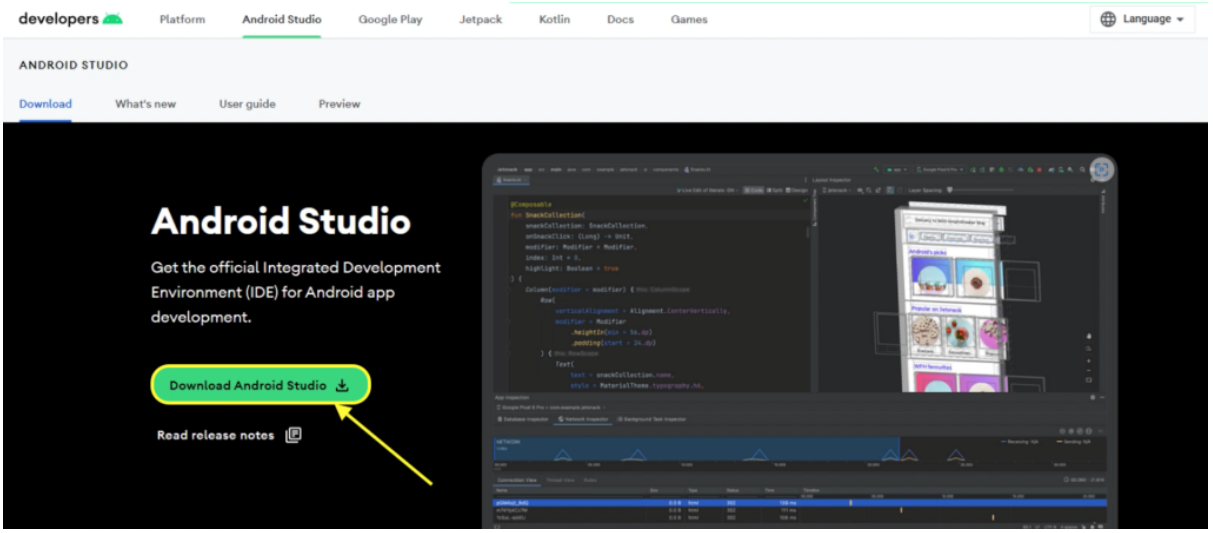
Votre console devrait vous **afficher** toutes les **variables d'environnement** de votre PC.

Elles sont séparées par des points virgules et la **dernière** devrait être l'adresse de votre **SDK Flutter flutter/bin**.

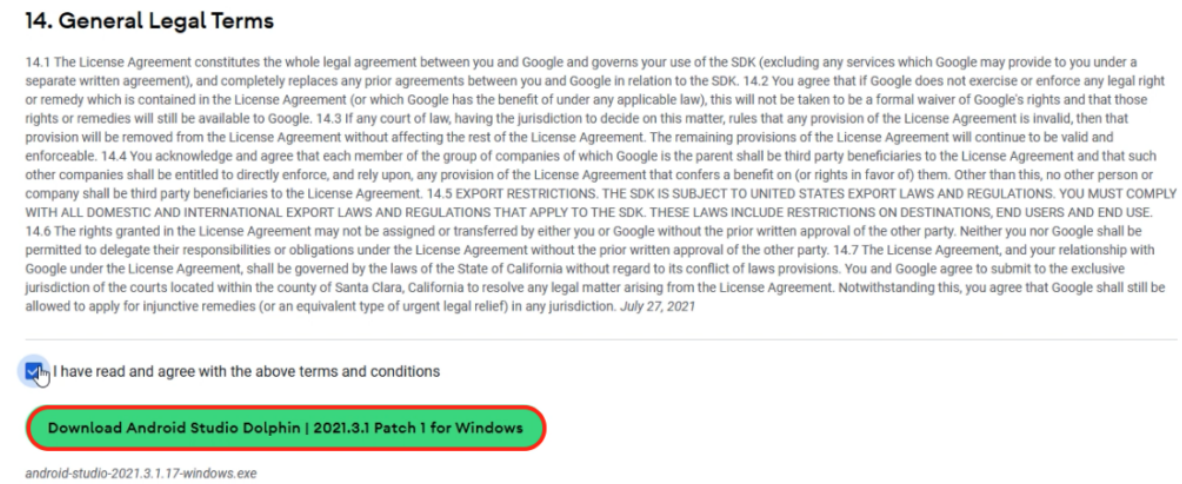
Vous avez donc normalement accès à la **commande Flutter** depuis **tout votre PC** pour créer par exemple de nouvelles applications.

5.3 Configurer Android Studio

Pour terminer, nous allons maintenant **installer et configurer** le logiciel de développement **Android Studio**: https://developer.android.com/studio

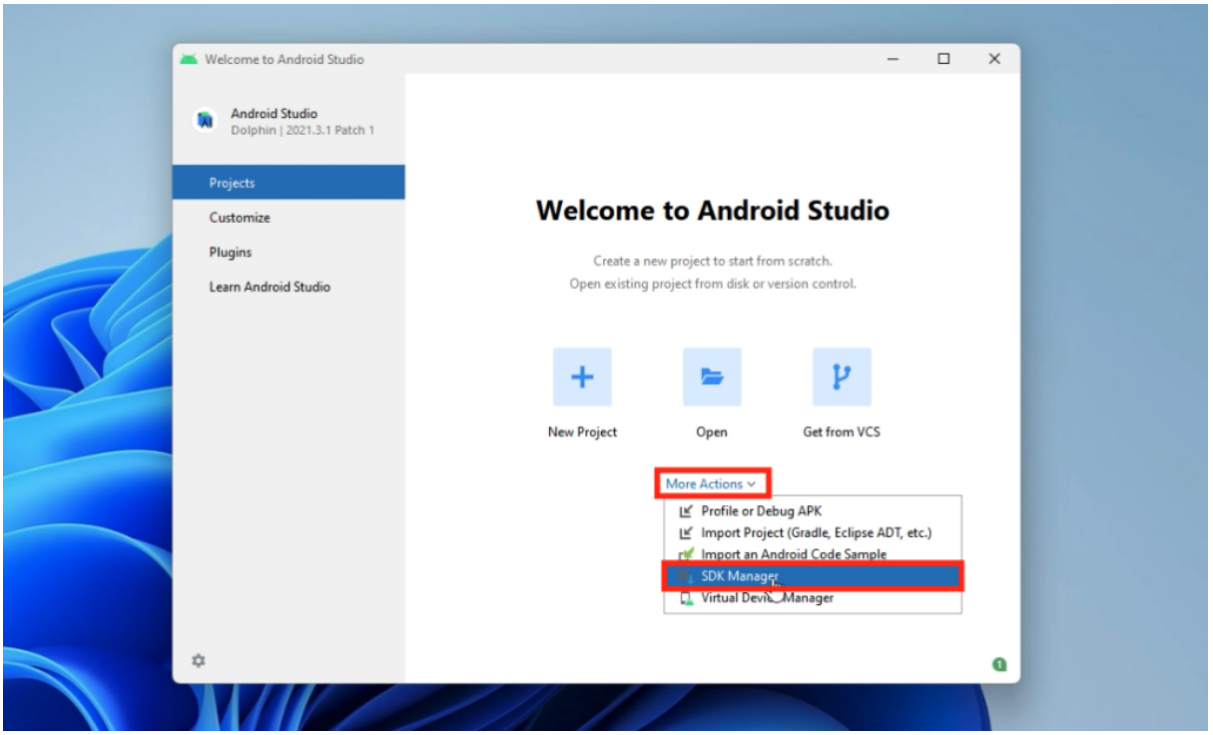


Cliquez sur le bouton "**Download Android Studio**", puis acceptez les conditions d'utilisation avant de télécharger le **patch Windows**:



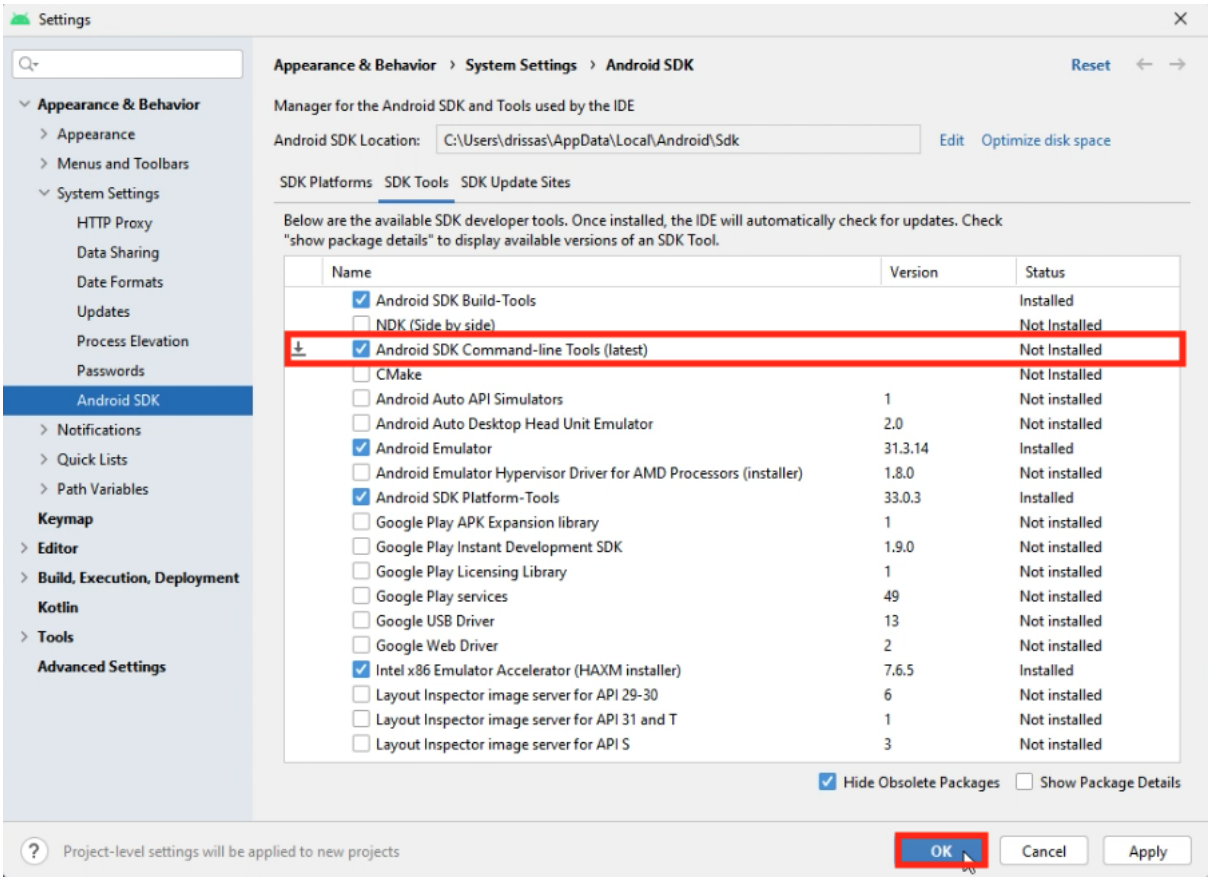
La **procédure** d'installation est la **même** que pour **Mac** et elle est classique, donc pas d'inquiétude à ce niveau-là.

Une fois que le logiciel Android Studio est **téléchargé et installé**, ouvrez-le une première fois.



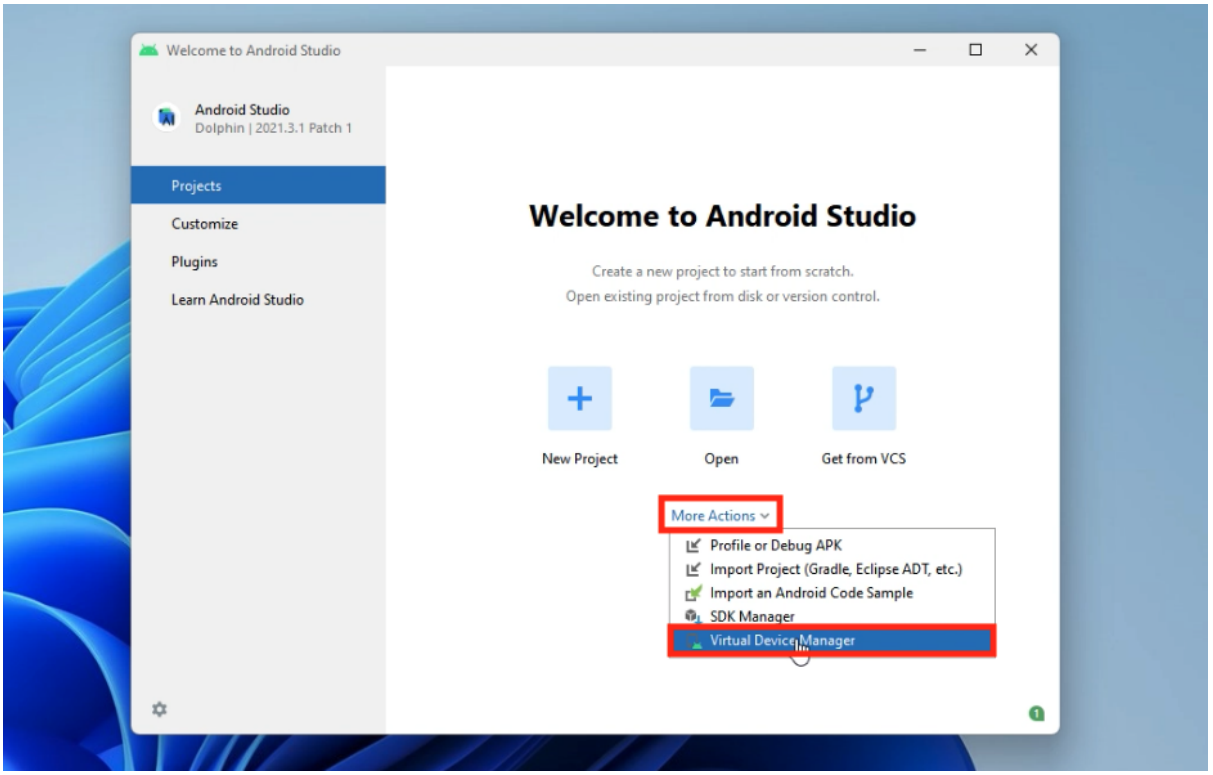
Sur le bouton "Configure"**More Actions**", cliquez sur **SDK Manager** pour vérifier la présence de **tous les SDK** dont nous allons avoir besoin par la suite.

Nous utiliserons **Android SDK Build-Tools**, **Android Emulator** et **Android SDK Platform-Tools** et surtout **Android SDK Command-line Tools** qui n'est pas présent par défaut:



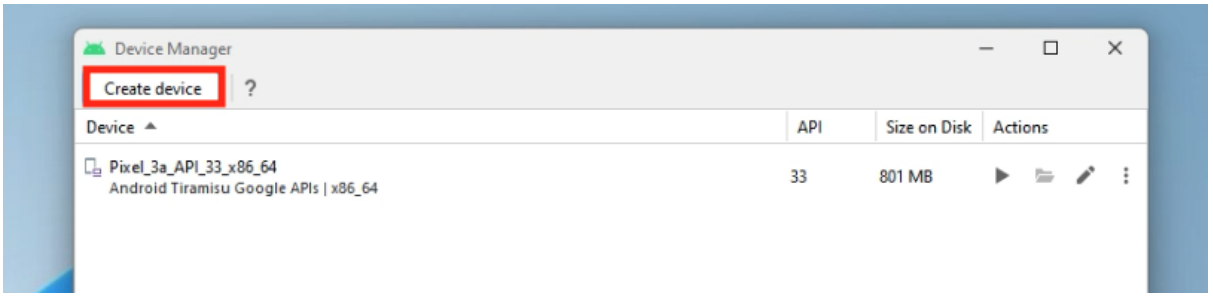
S'il en **manque un**, cliquez dessus et **installez-le** pour être à jour.

Ensuite, revenez sur l'**écran d'accueil** d'Android Studio et ouvrez cette fois-ci le **Virtual Device Manager**:

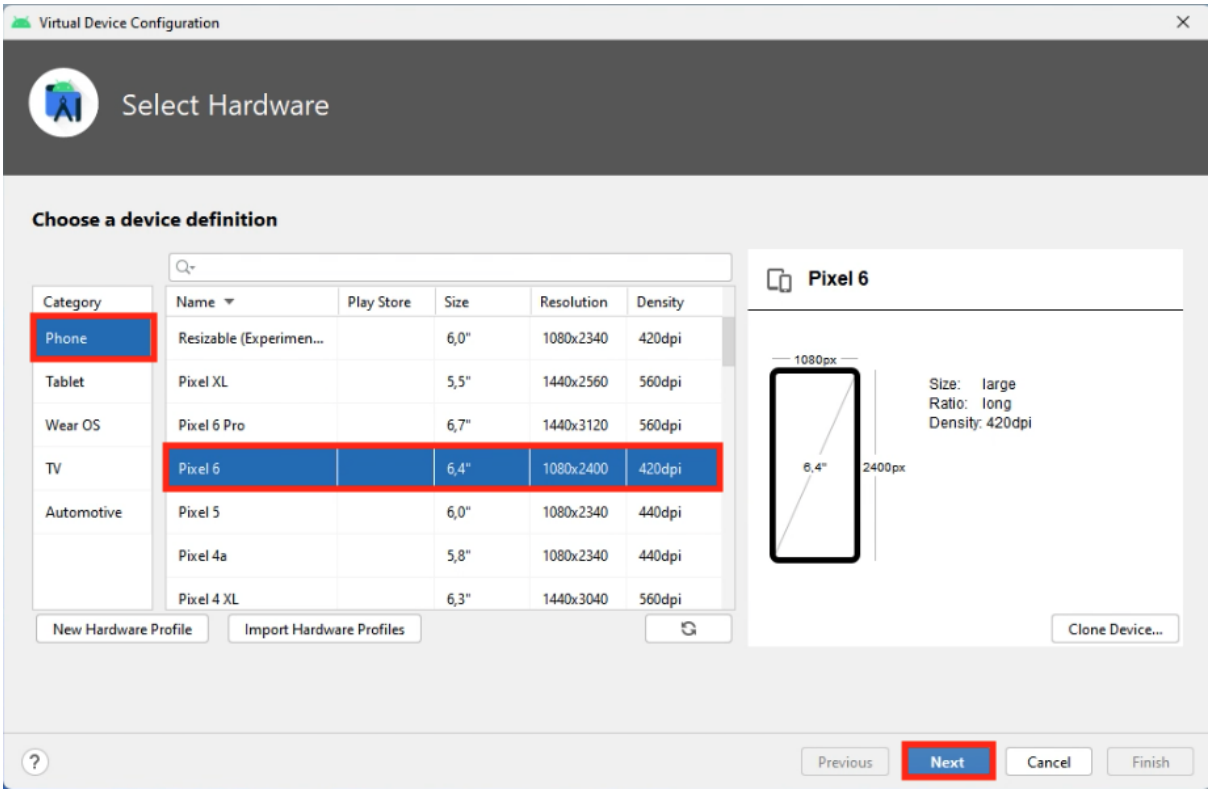


C'est le **gestionnaire des émulateurs Android** (Android Virtual Device: AVD) qui nous permettra de les **lancer** ou d'en **créer** de nouveaux.

Vous pouvez donc cliquer sur "**Create Device**" pour créer notre **premier émulateur** Android:

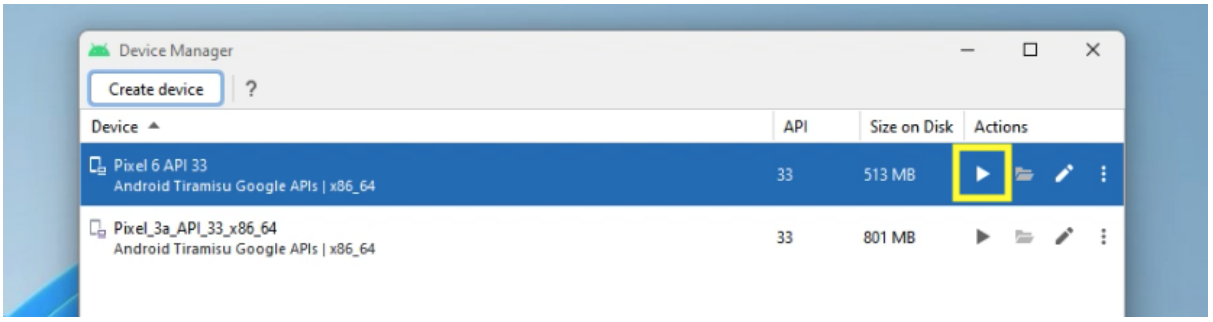


Sélectionnez ensuite le **modèle d'Android** que vous souhaitez reproduire, personnellement, je conserve le choix du **Pixel 6** proposé par défaut:



Installez ensuite la **dernière version d'Android** et validez toutes les étapes de la création votre premier émulateur.

Une fois créé, il devrait apparaître dans votre **tableau de bord**, vous n'avez plus qu'à le lancer en cliquant sur **l'icône play**:



Pendant que votre émulateur Android **s'ouvre** et se charge correctement, ouvrez à nouveau votre **console Windows**.

Rendez-vous dans votre **répertoire de développement**:

```
cd development
```

Puis **créez** votre **nouvelle application Flutter** avec la commande:

```
flutter create my_app
```

Un **nouveau dossier** devrait apparaître dans votre répertoire ***development***, contenant tout le code de votre **application Flutter**.

Avant de lancer votre application dans l'émulateur Android, exécutez une **dernière fois** la commande:

```
flutter doctor
```

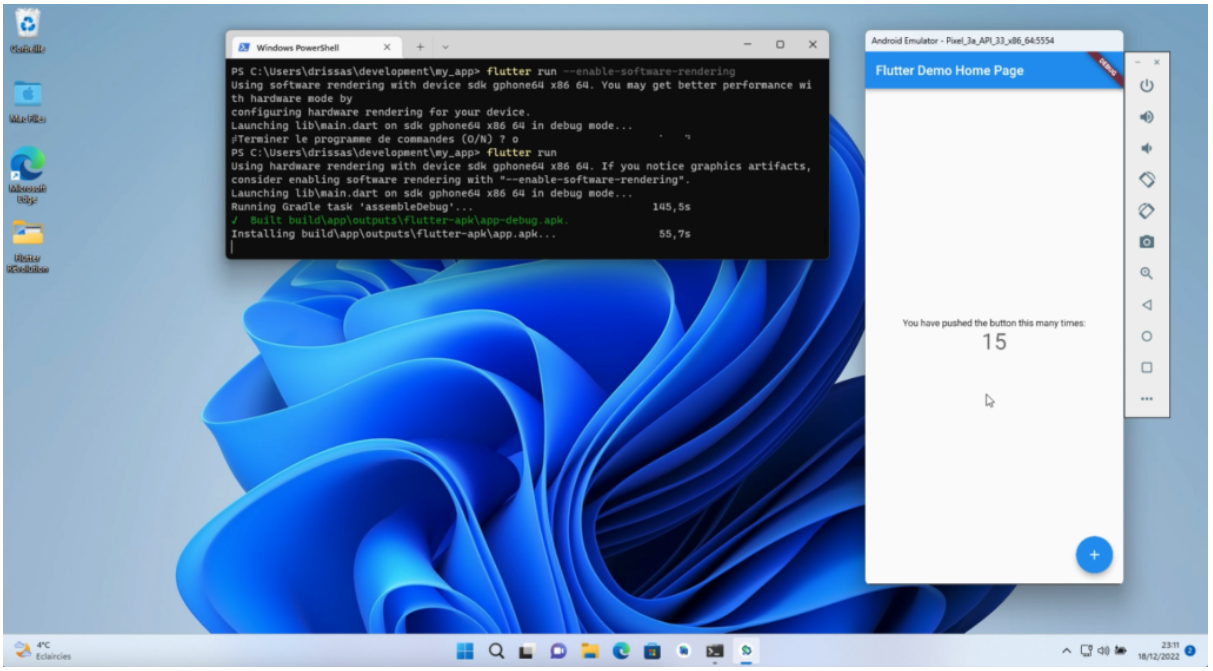
Pour repérer d'éventuelles **erreurs du SDK** et par exemple **accepter les conditions d'utilisation** du **SDK d'Android**, avec la commande suivante:

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

Une fois toutes les conditions d'utilisation acceptées, vous pouvez lancer la **commande run** pour lancer votre application:

```
flutter run
```

Patientez quelques instants pour voir votre application **se lancer** dans votre **émulateur Android**.



Voilà pour **l'installation de Flutter** sur **Windows**, nous terminerons la configuration de notre PC dans le prochain tuto dédié à Visual Studio Code.

6. Configurer Visual Studio Code

Dans ce chapitre, nous allons voir comment **installer et configurer** l'éditeur de code **Visual Studio** pour créer nos applications avec Flutter.

Flutter propose en effet **différentes solutions** pour développer avec Dart et Flutter, j'ai choisi d'utiliser Visual Studio Code, car le logiciel est **disponible** sur toutes les plateformes **Mac et PC**.

Nous verrons ensuite comment **créer une nouvelle application Flutter** en partant de zéro et l'ouvrir dans votre émulateur **iOS ou Android** depuis Visual Studio Code (VSC).

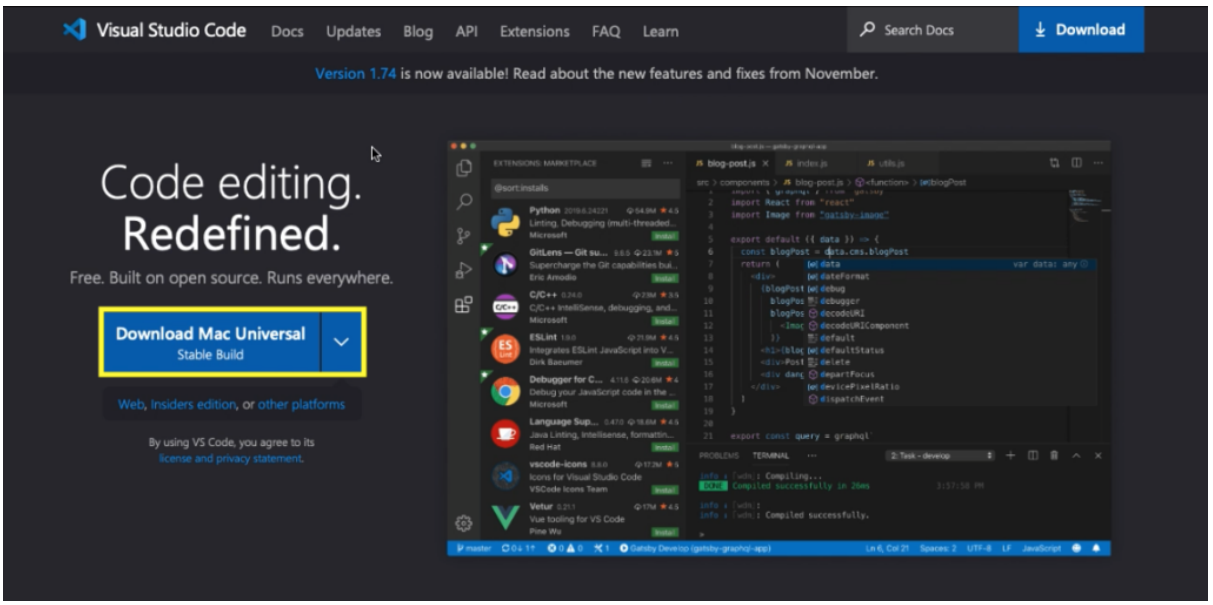
6.1 Installer et configurer VS Code sur Mac

Flutter propose une page consacrée à la configuration d'un **éditeur de code**: https://flutter.dev/docs/get-started/editor?tab=vscode

J'ai choisi **VSC** car Flutter y a créé des **extensions** qui transforment l'éditeur en véritable **logiciel** de développement très **complet**.

Nous n'aurons même **plus besoin d'ouvrir Xcode ou Android Studio** pour lancer nos applications dans un émulateur.

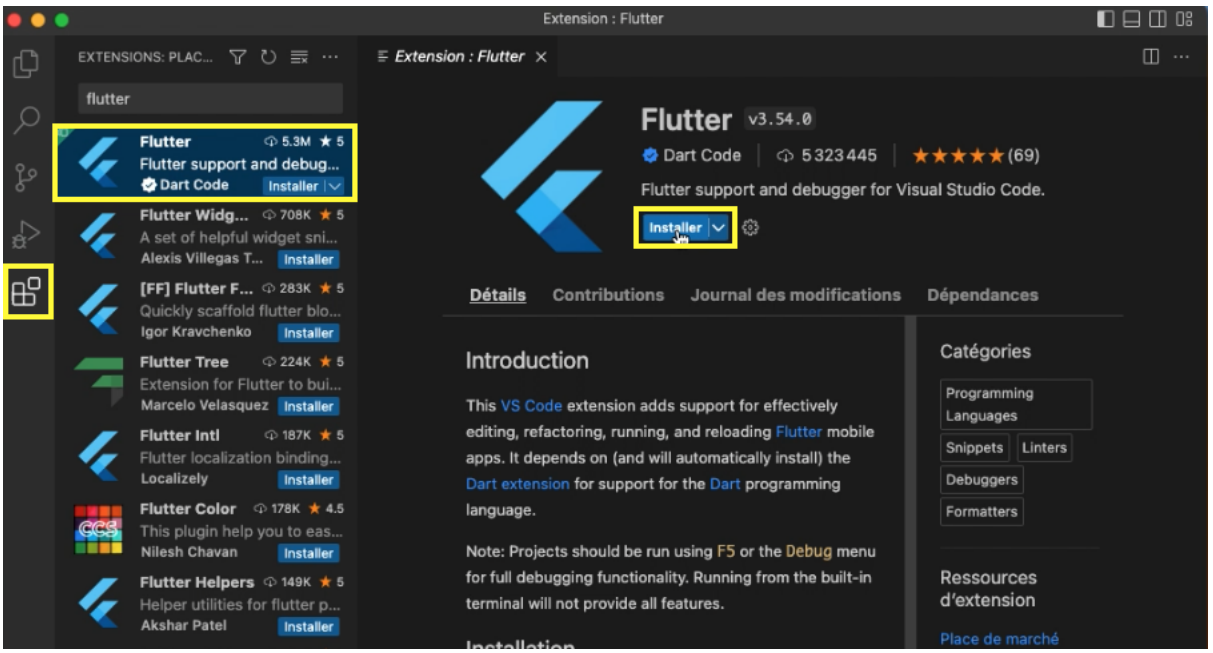
Vous pouvez commencer par **télécharger le logiciel VSC** depuis le site: https://code.visualstudio.com/



Une fois le logiciel installé et lancé sur votre ordinateur, nous allons installer les **extensions Dart et Flutter** de VSC.

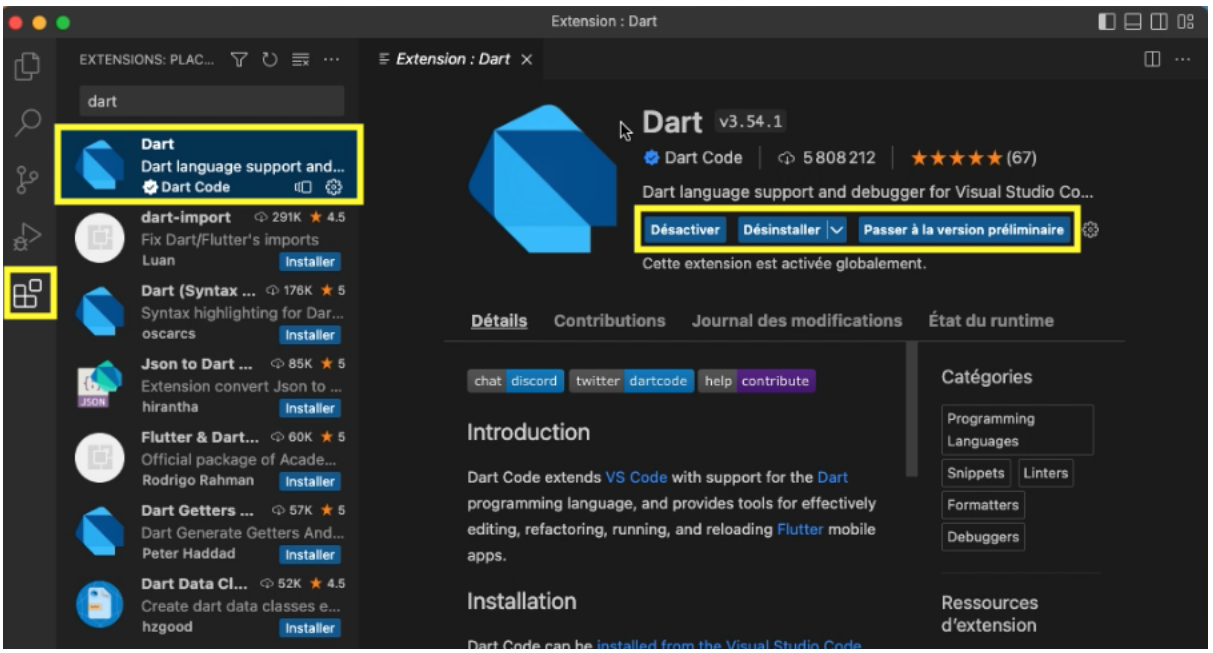
Ouvrez le **menu des extensions** à gauche de votre logiciel, puis tapez dans la **barre de recherche** "Flutter".

Installez ensuite le premier **plugin** qui s'affiche qui doit avoir été **édité par "Dart Code"**.



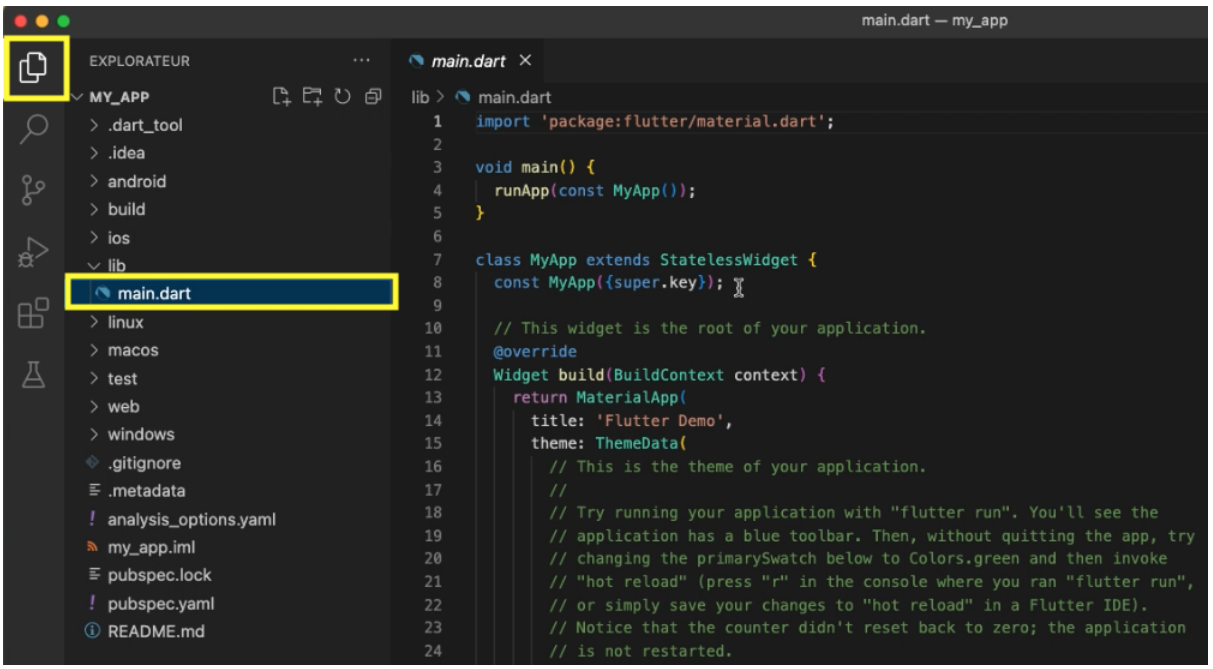
Lorsque l'extension **Flutter** s'installe, l'extension **Dart** doit, elle aussi, être **installé automatiquement.**

Dans le doute, vérifiez qu'elle est bien installée en cherchant **l'extension Dart** du développeur "Dart Code".



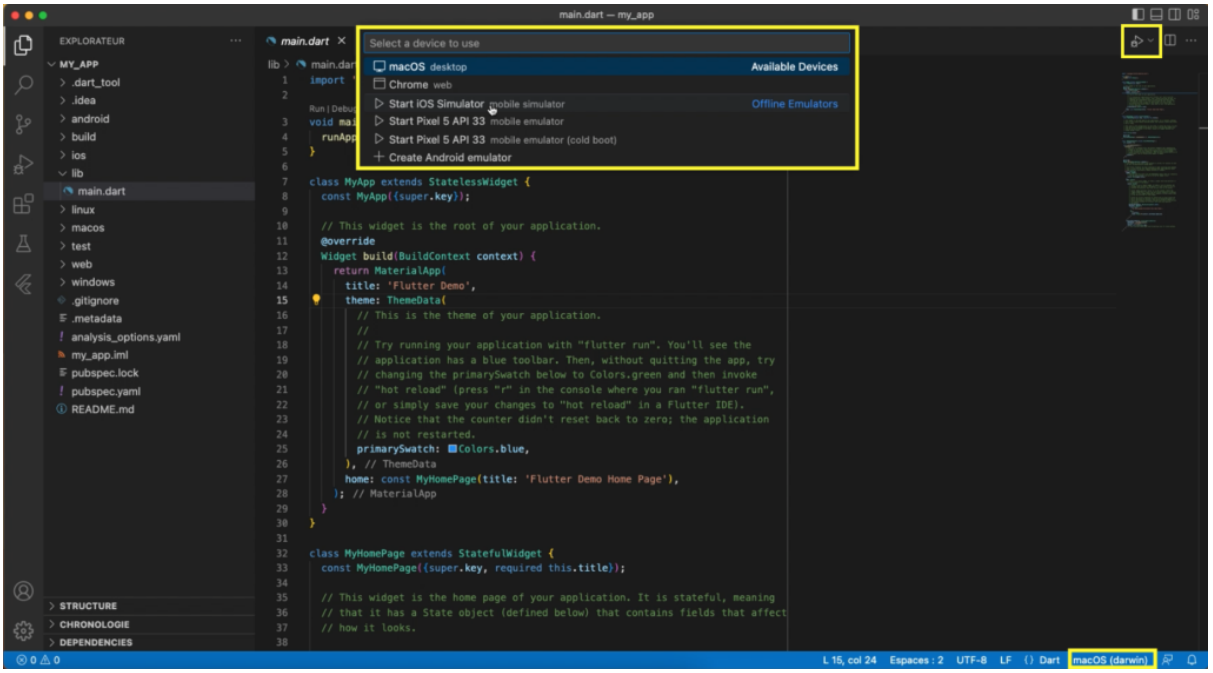
Maintenant que les **deux extensions** sont **installées**, nous allons **tester la fonctionnalité** la plus intéressante de VSC.

Commencez par **ouvrir le fichier principal** de votre application, le fichier ***main.dart*** dans le dossier ***lib***:



Maintenant que ce **fichier est ouvert**, c'est celui-ci que VSC choisira **d'exécuter et de lancer** dans votre émulateur.

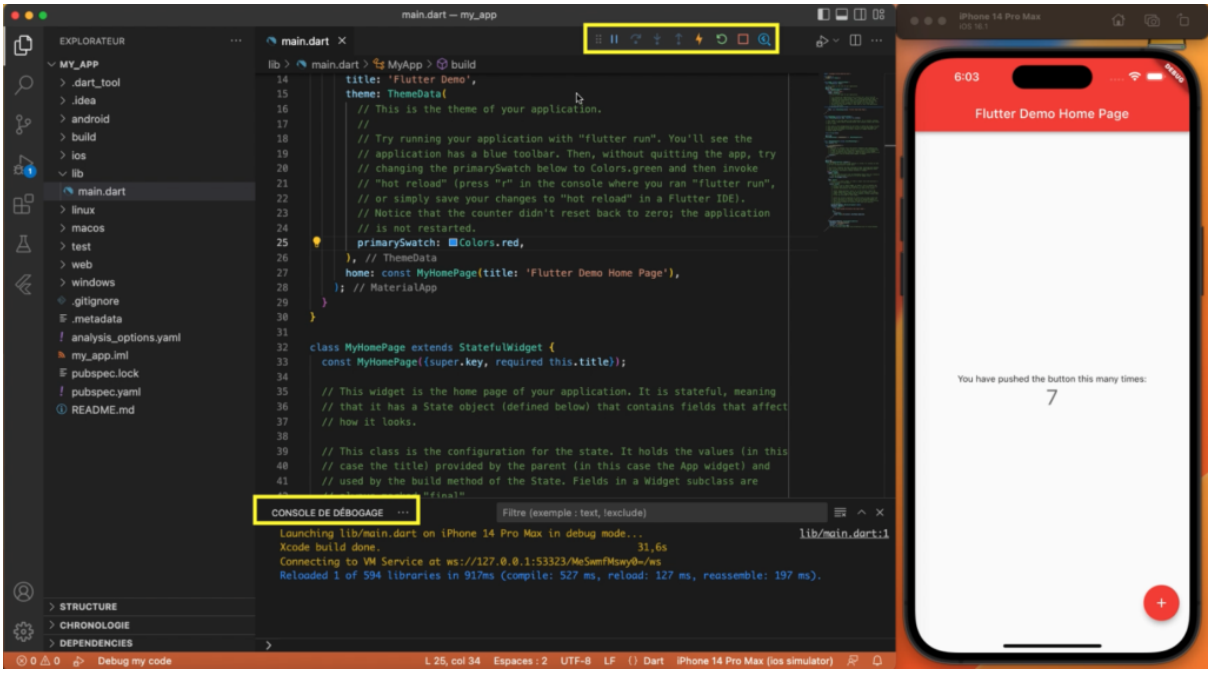
VSC devrait vous proposer de **choisir un émulateur iOS ou Android** selon votre configuration.



Je choisis de lancer mon application sur un **émulateur iOS** et il me lance mon dernier appareil ouvert, un **iPhone 14 Pro Max**.

Après un **petit temps de chargement,** mon application s'ouvre dans mon émulateur.

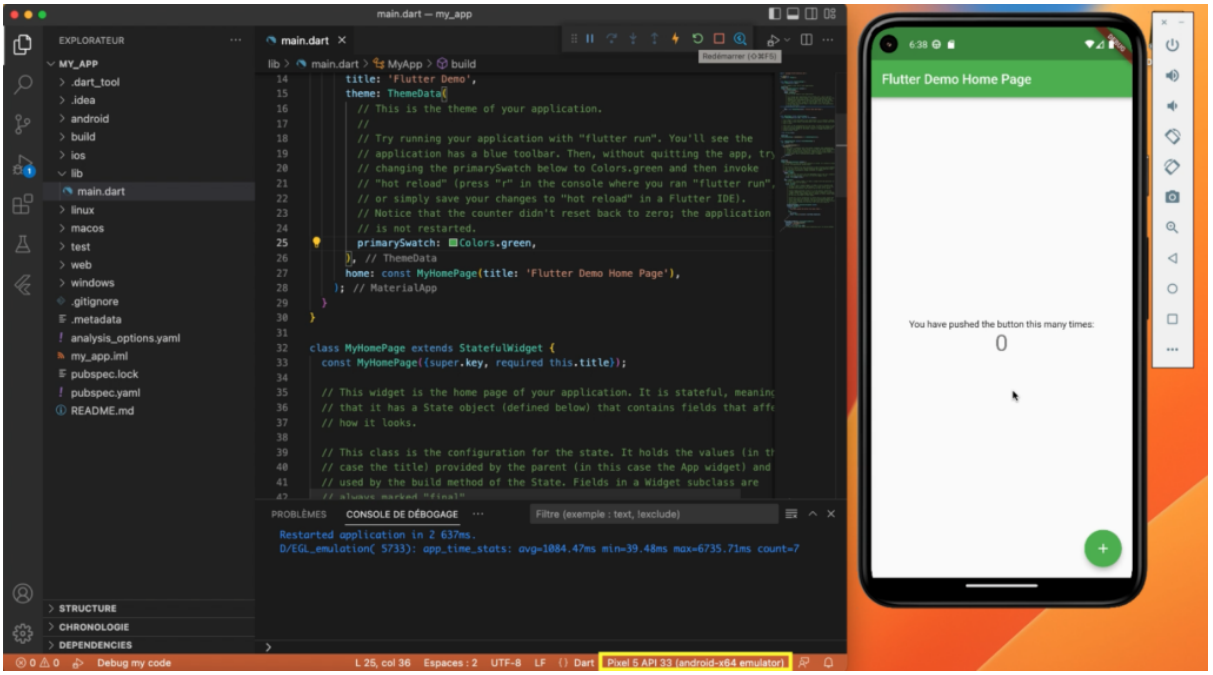
Vous avez accès à une **série de boutons** pour **rafraîchir** manuellement votre application ou bien **arrêter** son exécution.



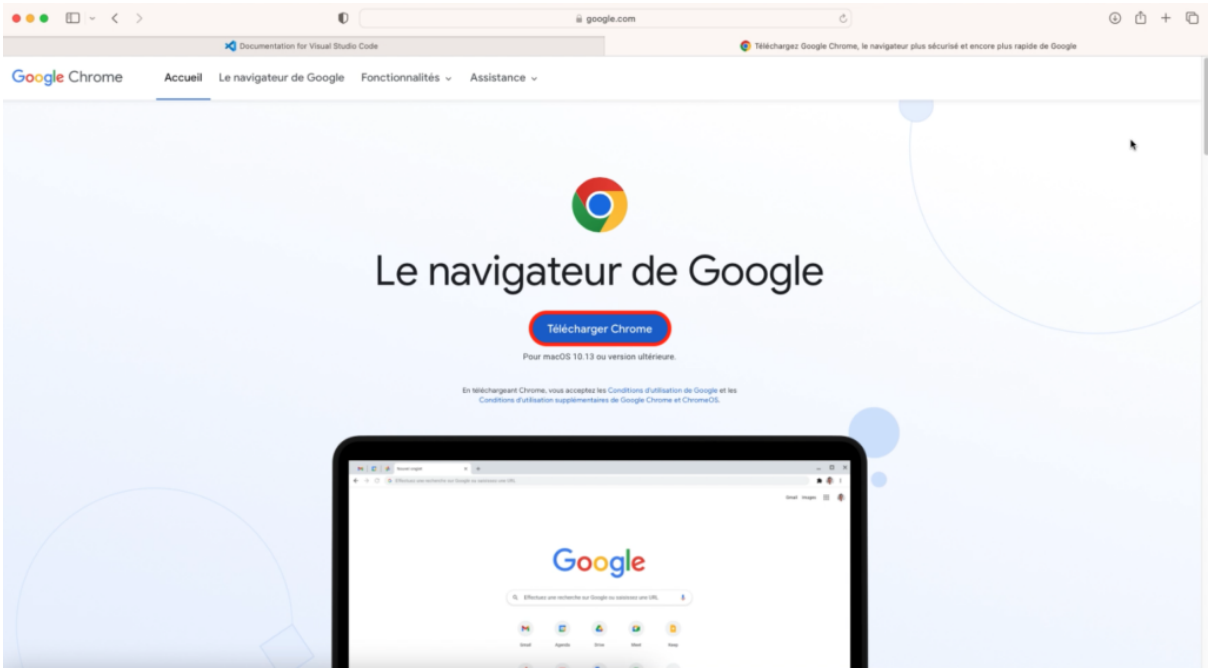
Mais normalement dès que vous **enregistrez votre fichier** main.dart, votre application devrait **s'actualiser en direct**, voilà toute la magie de Flutter!

Vous pouvez également lancer un **émulateur Android** depuis Visual Studio Code en cliquant sur "**Start Pixel Emulator**".

Une fois que celui-ci est chargé, vous pouvez **lancer** votre application sur Android en restant positionné dans le fichier **main.dart**:



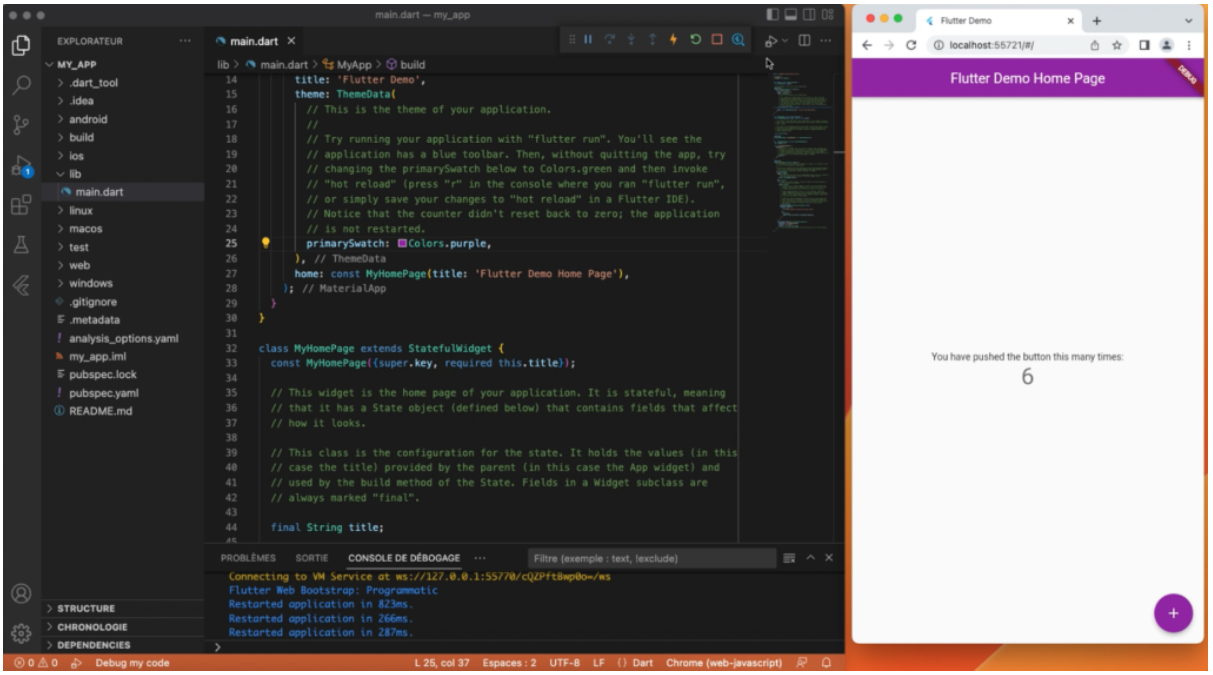
Pour la version web de votre application, n'oubliez pas de télécharger Google Chrome: https://www.google.com/intl/fr_fr/chrome/



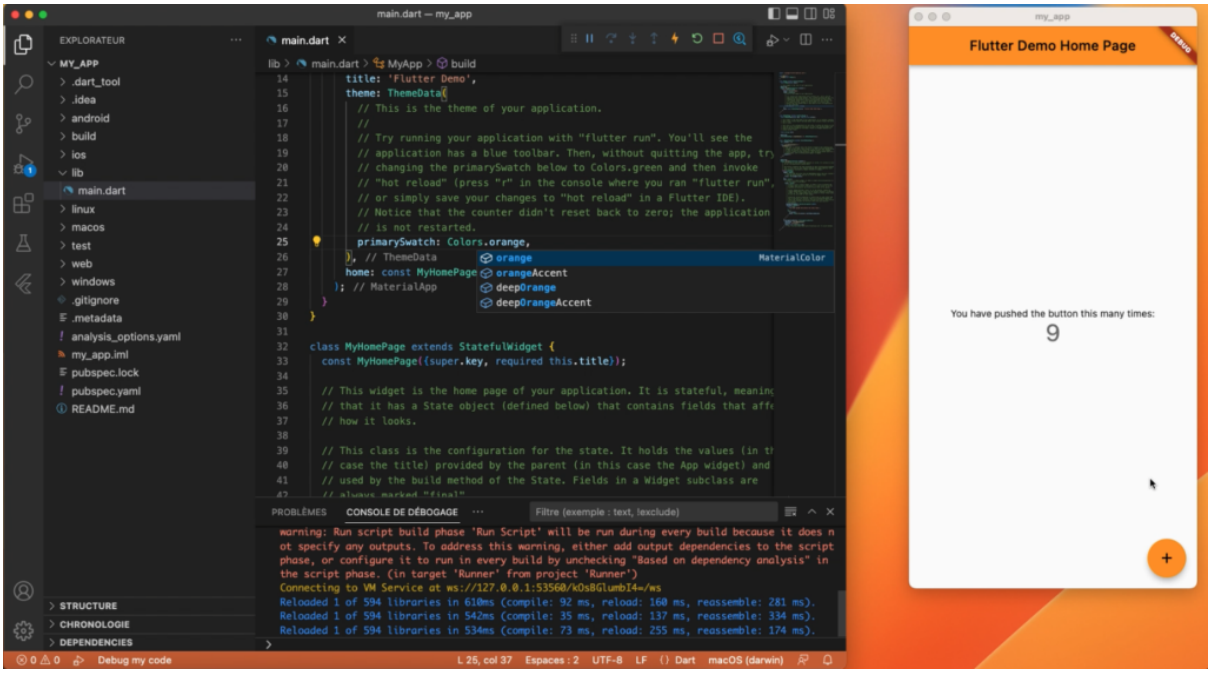
```
drissasdev@MacBook-Pro-de-Driss ~ % flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.3.9, on macOS 13.1 22C5033e darwin-x64, locale
    fr-FR)
[✓] Android toolchain - develop for Android devices (Android SDK version 33.0.1)
[✓] Xcode - develop for iOS and macOS (Xcode 14.1)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2021.2)
[✓] VS Code (version 1.74.1)
[✓] VS Code (version 1.74.1)
[✓] Connected device (2 available)
[✓] HTTP Host Availability

• No issues found!
```

Vous pouvez alors cibler la **version web** de votre application et la lancer **en local** sur votre ordinateur:



La méthode est identique pour la **version macOS** de votre application, en ciblant la version **macOS (darwin)**:

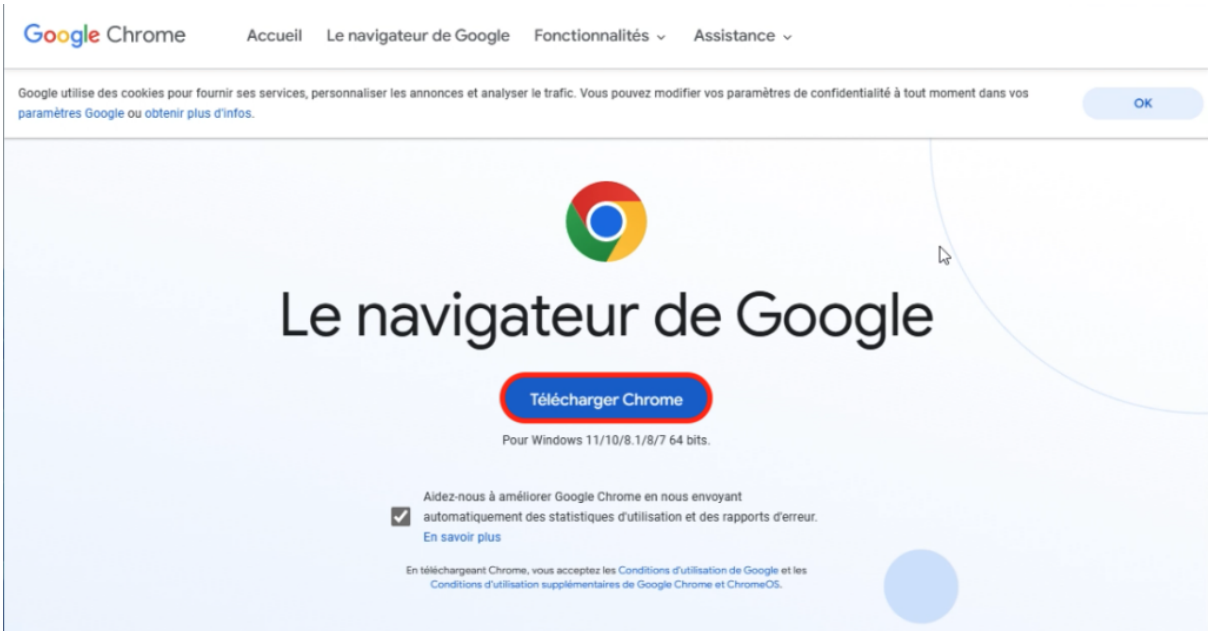


Voilà pour toutes les **plateformes à cibler** sur Mac: application **iOS et Android**, site **web** et logiciel de **bureau** macOS.

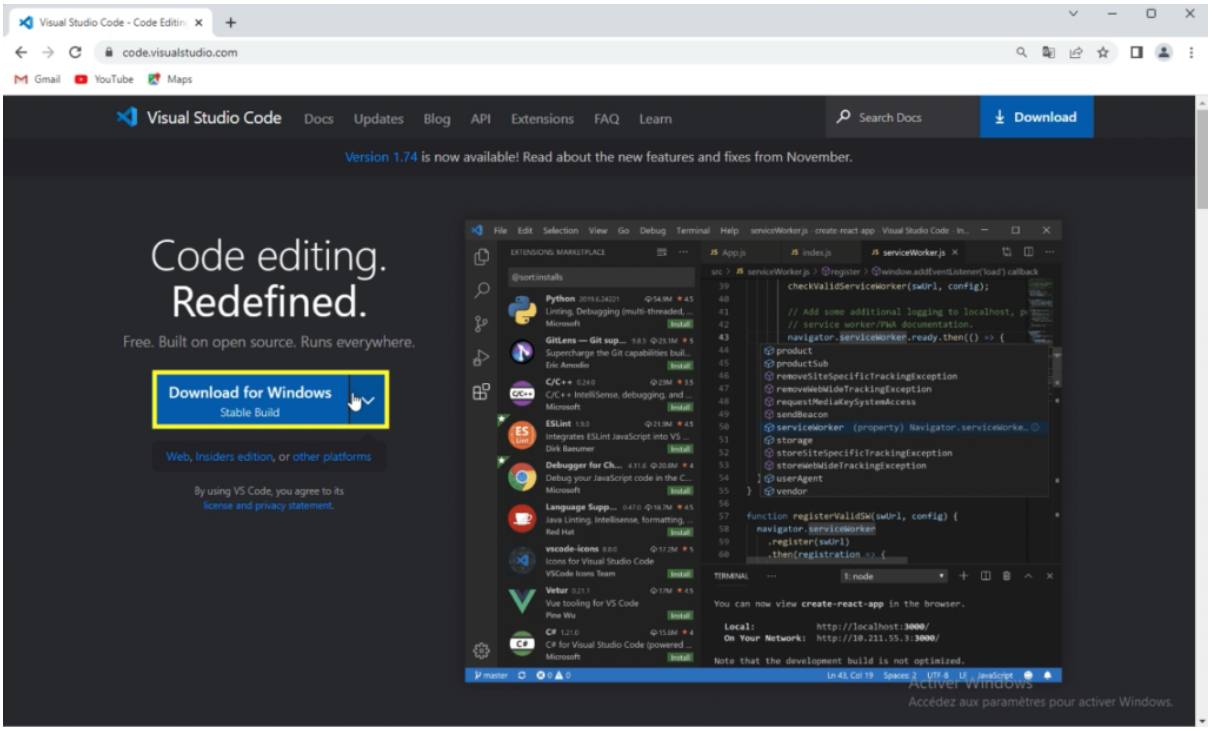
6.2 Installer et configurer VS Code sur Windows

Passons maintenant à la **configuration** de **Visual Studio Code** sur **Windows**, très proche, mais avec des étapes légèrement différentes.

Commencez par télécharger **Google Chrome** pour lancer votre application en **version web**: https://www.google.com/intl/fr_fr/chrome/

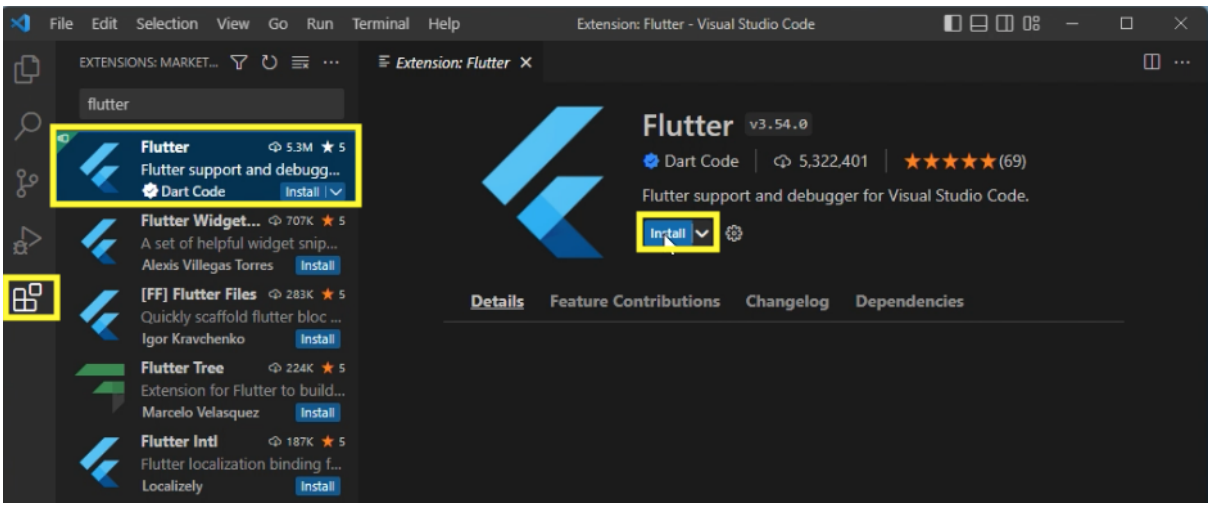


Continuez en téléchargeant le logiciel Visual Studio Code pour Windows: https://code.visualstudio.com/

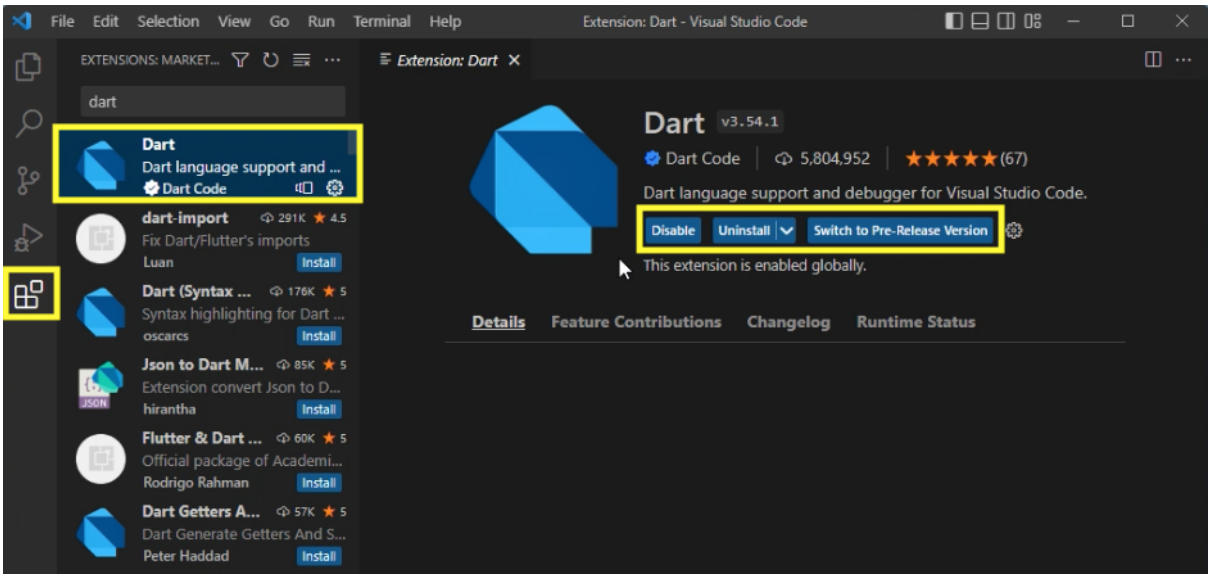


Une fois que le **logiciel est installé**, rendez-vous dans la **section** dédiée aux **extensions** et cherchez "***flutter***".

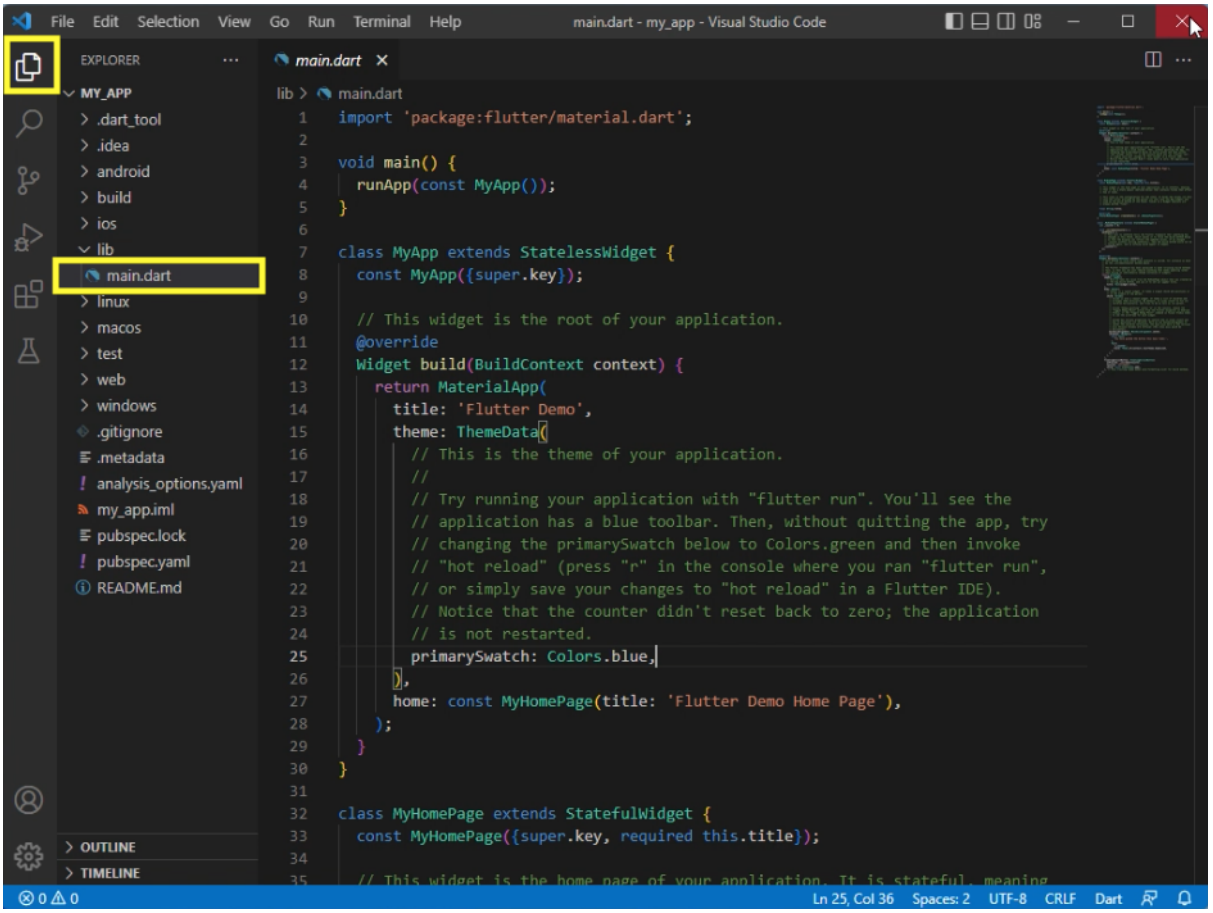
Installer alors la première **extension "Flutter"** éditée par "**Dart Code**":



Cela vous installera aussi automatiquement **l'extension "Dart**":

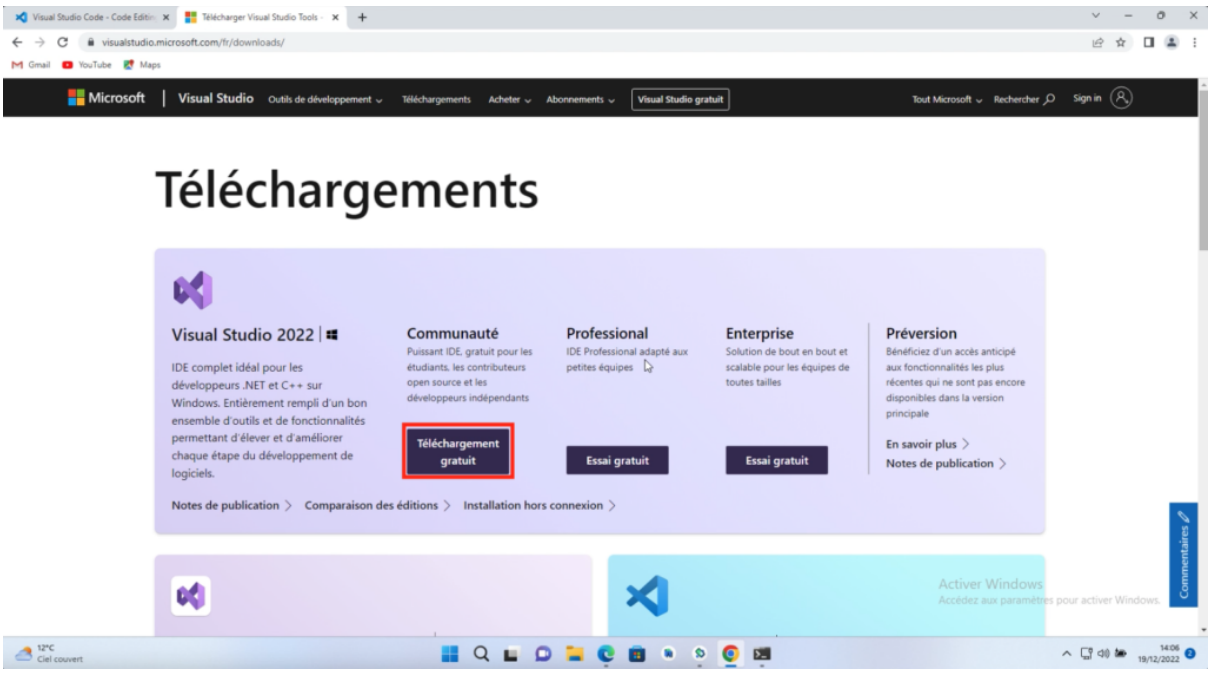


Vous pouvez alors ouvrir votre **dossier** d'application "***my_app***" avec Visual Studio Code et ouvrir le fichier **main.dart**:



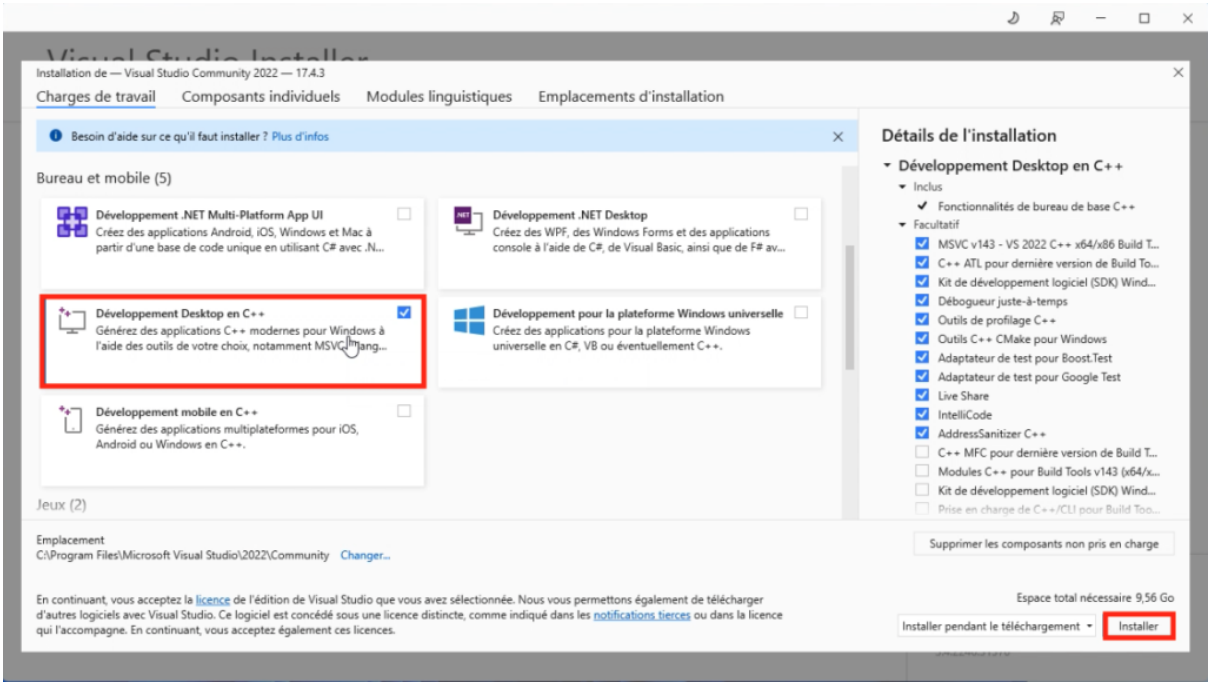
Avant de tester les différentes plateformes, je vous invite à **télécharger** en complément le logiciel "**Visual Studio**" de **Microsoft**: https://visualstudio.microsoft.com/fr/

Il s'agit du logiciel de **développement logiciel** de Microsoft accessible gratuitement:



Il vous permettra d'installer des outils de développement **logiciel C++** pour **Windows**.

Lors de la **phase d'installation** du logiciel, cliquez sur la case "***Développement Desktop C++***" pour ajouter les **modules nécessaires**:

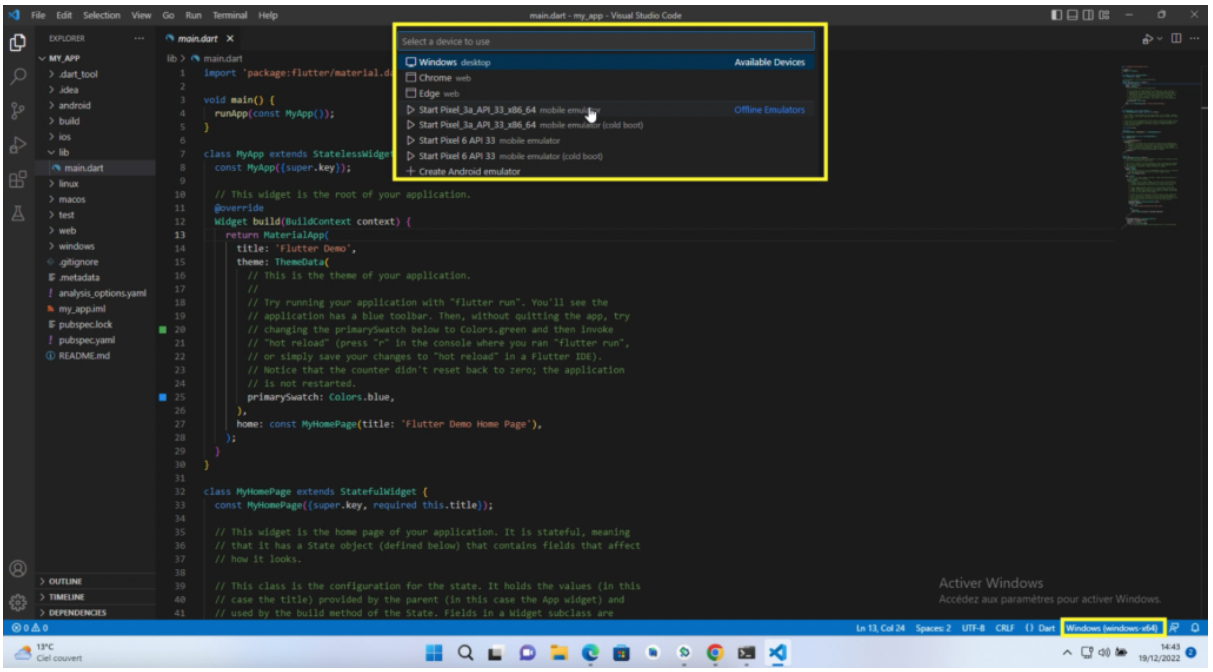


La commande "***flutter doctor***" devrait maintenant vous confirmer la **bonne configuration** de votre ordinateur:

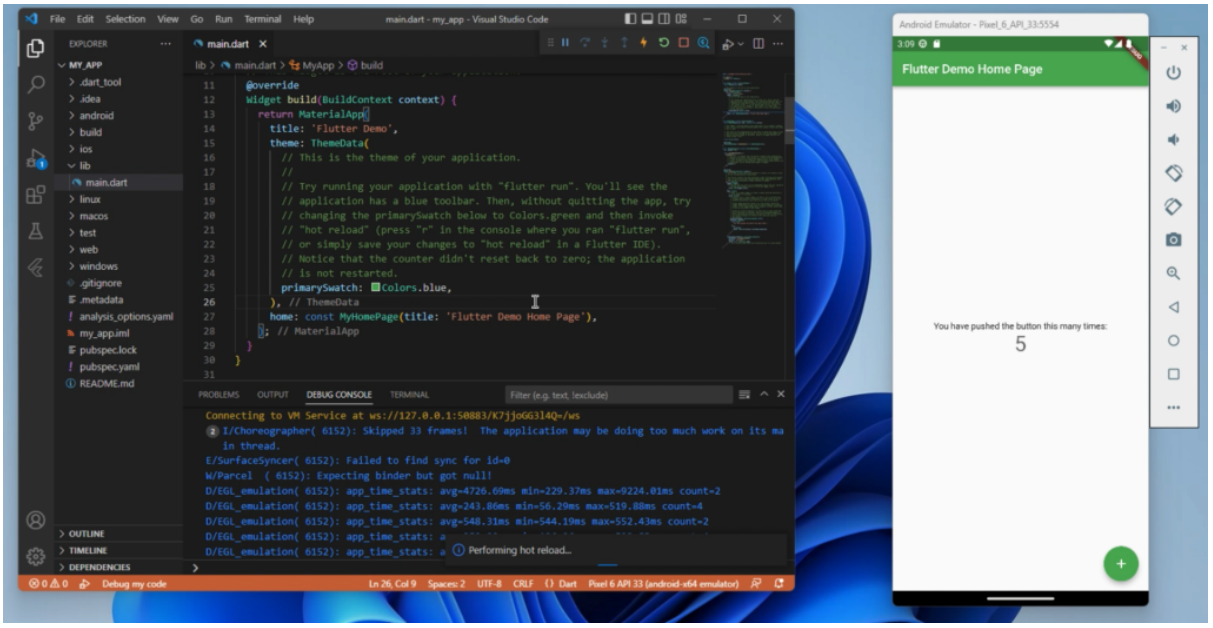
```
PS C:\Users\drissas> flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.3.10, on Microsoft Windows [version 10.0.22621.525], locale fr-FR)
Checking Android licenses is taking an unexpectedly long time...[✓] Android toolchain - develop for Android devices (And
roid SDK version 33.0.1)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop for Windows (Visual Studio Community 2022 17.4.3)
[✓] Android Studio (version 2021.3)
[✓] VS Code (version 1.74.1)
[✓] Connected device (4 available)
[✓] HTTP Host Availability

• No issues found!
PS C:\Users\drissas>
```

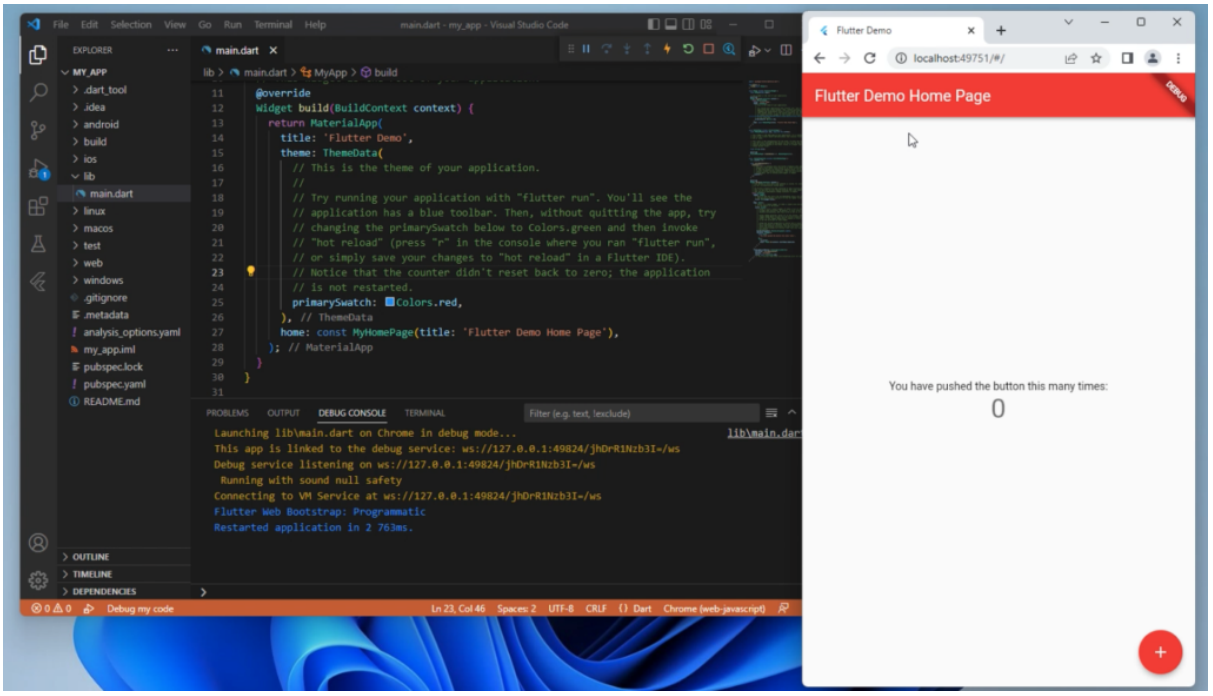
Vous pouvez alors sélectionner une **cible** pour votre **application Flutter:**



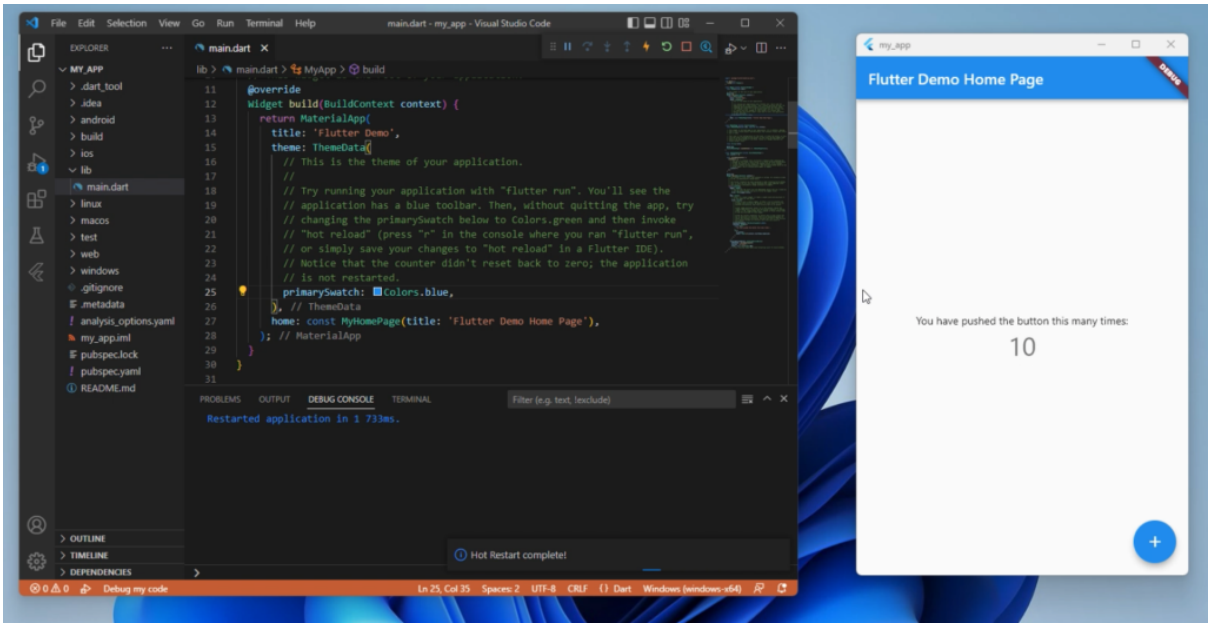
Par exemple, lancer un **Pixel de Google** puis lancer votre application dans cet **émulateur Android:**



Lancez également la **version web** de votre application avec la cible "***Chrome (web-javascript)***":



Et enfin la version de **bureau** avec la cible **Windows** (***windows-x64***):



Voilà pour toutes les **cibles possibles** de développement **Flutter** avec **Windows**.

6.3 Créer une application en partant de zéro

Je vous montre à nouveau comment **créer une application Flutter** en partant de zéro.

L'idée ici est de vous montrer la **procédure à suivre** lorsqu'on utilise **VSC** pour coder ses applications Flutter.

Vous devez comme toujours ouvrir votre **terminal** et vous rendre dans le dossier "***development***" qui contient votre SDK Flutter et vos autres applications.

```
cd developmment
```

Une fois que vous vous trouvez dans ce **dossier**, vous pouvez entrer la commande "**flutter create ...**" pour créer une nouvelle application Flutter avec son **nom**:

```
flutter create my_flutter_app
```

Cette étape doit être effectuée depuis votre logiciel **Terminal ou Console**, et non dans Visual Studio.

Vous avez le choix à partir de là d'**ouvrir votre nouvelle application dans Visual Studio Code** et de suivre les procédures précédentes pour lancer votre application dans un émulateur.

Sinon, vous pouvez aussi **rester dans votre terminal**, vous rendre dans le dossier de votre application et la lancer avec la commande "**flutter run**":

```
cd my_flutter_app
flutter run
```

Sinon, je vous conseille de n'utiliser le **Terminal ou la Console** que pour **créer de nouvelles applications**, ce qui ne devrait pas arriver si souvent.

Ensuite, je vous invite à travailler **exclusivement** avec **Visual Studio Code** qui intègre son propre **terminal**.

Cela vous **facilitera** votre travail de développeur en ayant à ouvrir que le logiciel **Visual Studio Code** lorsque vous voulez améliorer votre application.

7. Pour aller plus loin

Bienvenue dans cette vidéo consacrée à ma deuxième formation **Flutter Académie**.

Si vous consultez cette vidéo, c'est que vous êtes **membre de Flutter Révolution** ou que vous envisagez sérieusement de le **devenir**.

Je vais tâcher de vous expliquer dans cette **courte vidéo**, pourquoi j'ai créé un **programme plus avancé** et comment il pourra vous aider.

7.1 À propos de Flutter Révolution

Revenons rapidement sur la **Flutter Révolution** et ses objectifs de départ en tant que formation **accessible aux débutants**.

Le but de Flutter Révolution, est de permettre à n'importe quelle personne de **découvrir Flutter**, de développer sa **première application** et de la **publier sur iOS et Android**.

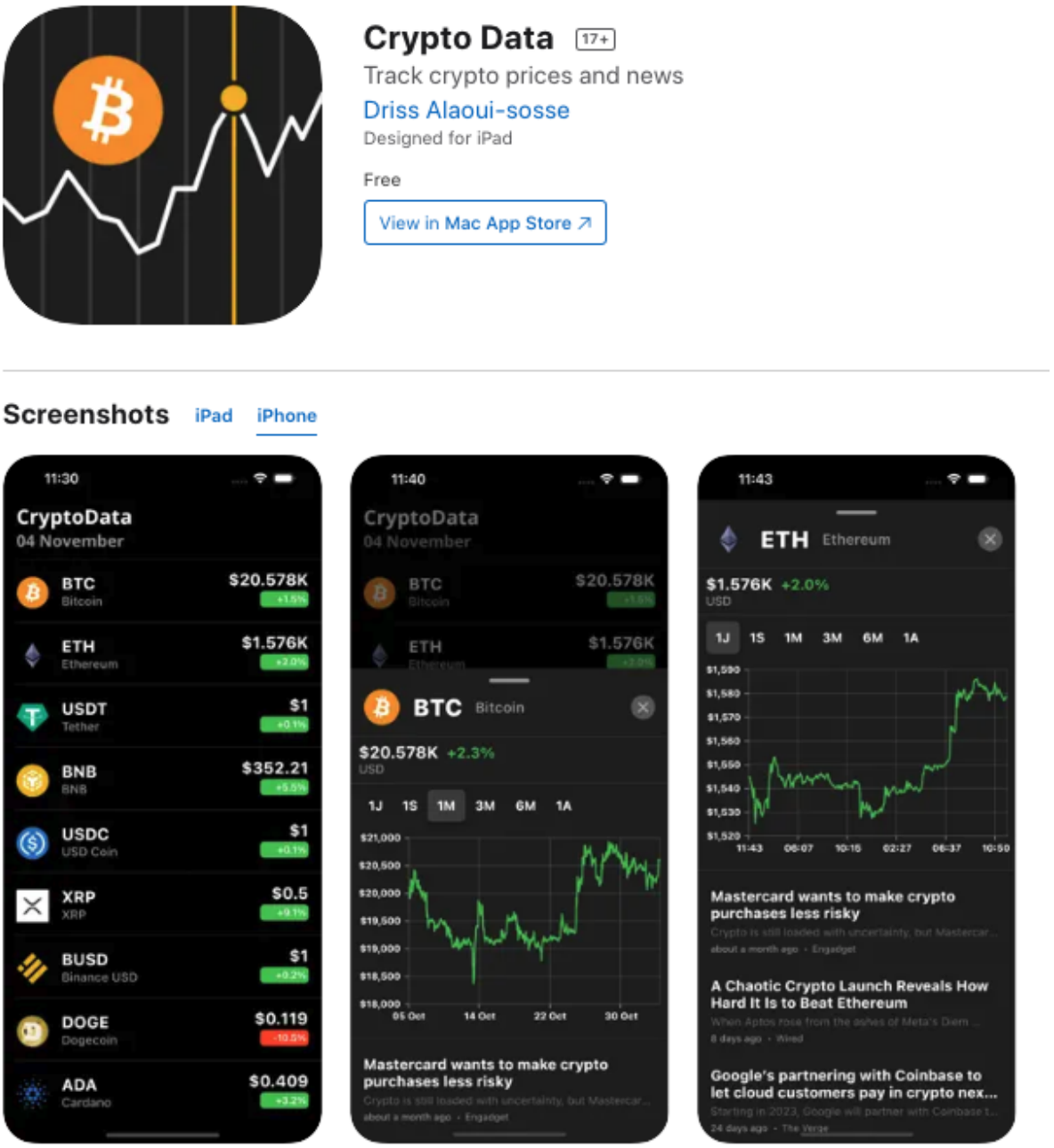
La formation vous donne les **bases indispensables** pour créer une **application mobile** Flutter pour iOS et Android.

Je reviens sur les **bases de Flutter** et du **Dart**, j'aborde la connexion à **Firebase** et à d'autres services tierce comme **Google Maps**.

Même la monétisation est présente avec **Stripe** et les **achats intégrés** à la fin de la formation.

Mais le véritable objectif de la formation, c'est de vous amener à une **première publication** de votre application sur les **stores d'Apple et de Google**.

Le **dernier chapitre** est donc consacré aux étapes de **publication** sur **l'AppStore** et le **Play Store**, avec toute la configuration nécessaire.



Avec cette **base de compétence**, il est évidemment possible **d'aller plus loin** avec Flutter et de développer des applications encore **plus avancées**.

J'avais notamment en tête toute une **liste de fonctionnalités** qui m'était très régulièrement demandé, comme les **notifications** push, la **traduction**, l'architecture **Bloc**, etc.

C'est pourquoi j'ai eu l'idée de créer une **deuxième formation Flutter**, **complémentaire** à la première et surtout **plus avancée** dans son contenu.

7.2 La création de la Flutter Académie

Quant à la création de la **Flutter Académie**, l'idée a mis du temps à **germer dans mon esprit** pour me **convaincre** de me lancer dans une deuxième formation.

Le **temps** à consacrer à une nouvelle formation étant le **principal frein** de mon côté, compte tenu du **travail à fournir** pour chaque tuto.

Les tutoriels Flutter **plus avancés** demandant naturellement **plus de travail**, j'étais frileux de me lancer dans une deuxième formation.

Ce qui m'a convaincu, c'est de créer non pas plusieurs **petits chapitres** comme dans la Flutter Révolution, mais plutôt des **gros modules** bien distincts.

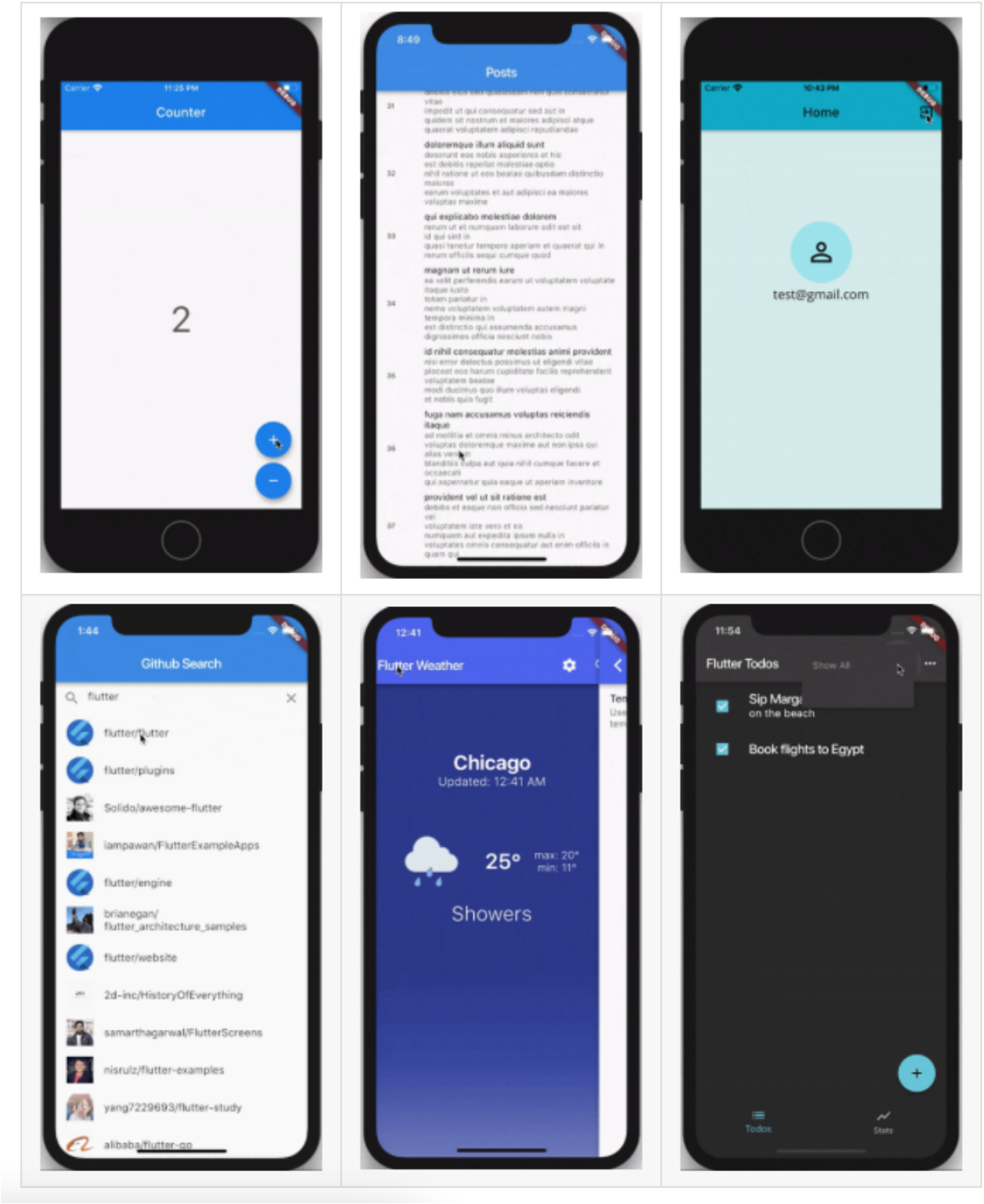
Vous retrouvez actuellement **trois modules** dans la **Flutter Académie** que sont:

1. Flutter Power
2. Firebase Expert
3. Bonus Flutter

Le premier module **Flutter Power**, comme son nom l'indique, consiste à vous faire **exploiter** tout le **potentiel de Flutter.**

Maîtriser les **bases de Flutter** peut se faire en **quelques jours**, mais approfondir certaines notions vous demandera beaucoup **plus de temps**.

Il s'agit par exemple de **l'architecture** d'un projet Flutter avec le modèle de **conception BLoC**, ou encore la maîtrise des **animations** plus avancées.



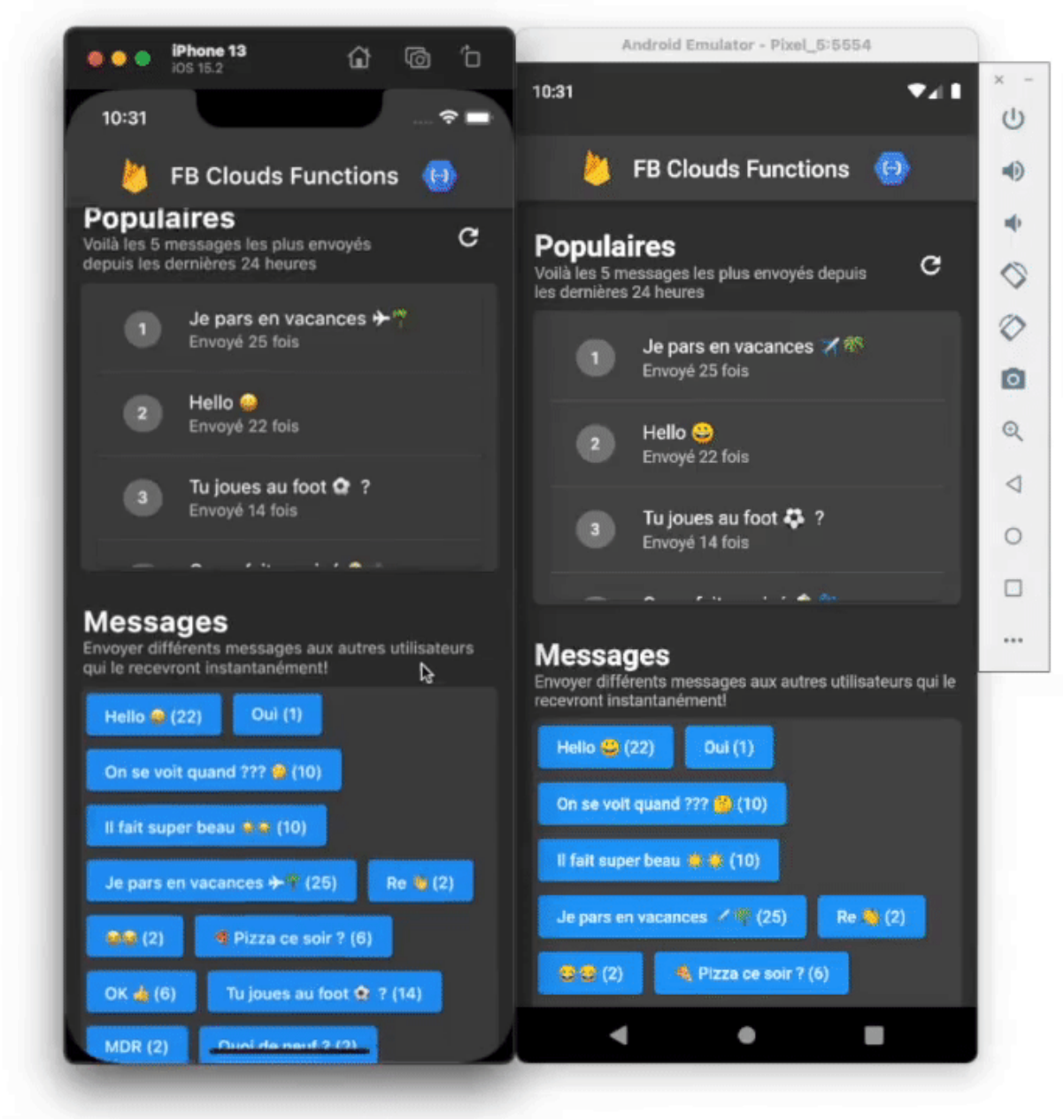
L'idée est donc d'aller beaucoup **plus loin** dans tout ce que propose Flutter, et d'exploiter tout le **pouvoir de Flutter** en tant qu'outil multiplateforme.

Le **deuxième module** était, lui aussi, tout aussi naturel et concernait l'autre grande domaine du développement mobile, le **backend Firebase**.

Même si dans **Flutter Révolution**, je consacre un **chapitre entier** aux bases de **Firebase** et que j'utilise ses services dans de **nombreux tutos**, je n'ai fait qu'effleurer le potentiel de Firebase.

Ce deuxième module **Firebase Expert** à ainsi pour objectif de vous faire **découvrir** et maîtriser d'autres **services de cloud** vraiment très intéressant.

On parle notamment des notifications push avec le service **Cloud Messaging**, de l'exécution de code en back-end avec les **Clouds Functions** ou encore du contrôle à distance d'une application avec **Remote Config**.



Vous l'avez compris, **Firebase** propose de **nombreux services** qu'il serait dommage de ne pas maîtriser pour les intégrer à Flutter.

Ce module fera donc de vous un **expert de Firebase** et vous permettra de développer des **applications ultra-avancées** et dynamiques avec Flutter.

J'aurais pu m'arrêter à ces **deux modules** vraiment **fondamentaux**, mais j'ai tenu à rajouter un ensemble de **contenu bonus**.

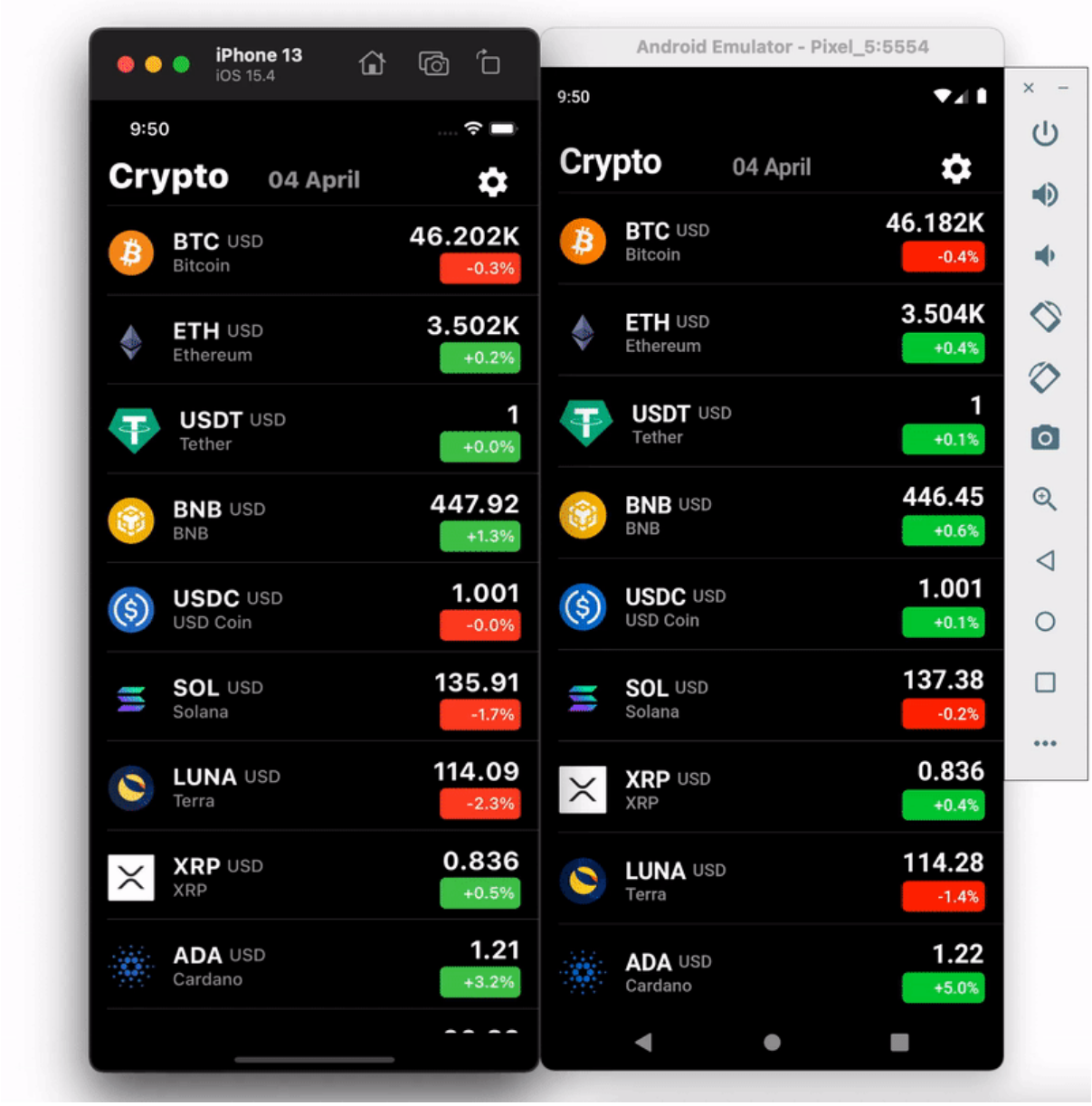
Ce **troisième module** regroupe à la fois des **tutos Flutter**, sur la création de **jeux vidéo** par exemple, et du contenu plus **marketing**.

L'idée est de vous donner des **outils complémentaires** pour faire de vous un développeur Flutter **polyvalent** avec un **angle à 360°**.

On aborde les fonctionnalités de **réseaux sociaux**, l'utilisation de **Flutter Flame** et l'utilisation du **marketing digital** pour les applications mobiles.

Je vous propose également un nouveau concept **d'atelier live** (disponible en **rediffusion**) où je développe avec vous des **applications plus avancées**.

L'application de **cryptomonnaie** est d'ailleurs le premier exemple que j'ai pris pour **l'atelier n°1** de la formation.



Bref, ce troisième **module bonus** est vraiment un concentré de pleins de **connaissances indispensables** aux **développeurs Flutter ambitieux**.

Voilà donc pour le contenu de la **Flutter Académie**, qui ne cessera d'ailleurs pas de **s'enrichir au fil du temps**.

7.3 Les différentes offres

Je vais être honnête et **totalement transparent** avec vous, lorsque j'ai lancé la **Flutter Académie** en début 2022, je ne proposait **aucun tarif particulier** pour les membres de Flutter Révolution.

Je considérais que seulement les **membres** de cette **première formation** rejoindraient le deuxième cours plus avancé.

J'ai alors lancé la Flutter Académie pendant **plusieurs mois** avant de me rendre compte que les **nouveaux arrivants** étaient **perdus**.

Certains me découvrait sur **YouTube** et **commençaient** par la **Flutter Académie** sans avoir appris les bases avec Flutter Révolution.

Ce n'était alors **pas suffisamment clair** pour tout le monde au niveau de la bonne **marche à suivre**.

J'ai en conséquence décidé de créer **deux offres spéciales**, pour tous les **futurs membres** et les encourager à prendre les formations dans **le bon ordre**.

Vous avez aujourd'hui la possibilité de prendre les **deux formations en même temps**, à un tarif vraiment très intéressant, disponible uniquement depuis la page de paiement de **Flutter Révolution**.

Pour ceux qui **hésitent**, pas de problème, je vous propose également **une offre valable à vie** pour les membres de **Flutter Révolution** qui leur permet de rejoindre la Flutter Académie à un **tarif préférentiel**.

Pour **résumer**, le **plus intéressant financièrement** est de prendre les **deux formations en même temps** depuis la page de commande de Flutter Révolution.

Si vous hésitez et vous **préférez attendre** pour prendre la deuxième formation et suivre déjà la Flutter Révolution, vous disposerez d'un **lien spécial** depuis votre **espace membre**.

Ce lien de paiement vous permettra de **rejoindre la Flutter Académie** à un **tarif privé** et uniquement accessible **aux membres** de Flutter Révolution.

Voilà les **deux options** que je vous propose aujourd'hui pour **aller plus loin** dans le développement mobile avec Flutter et **devenir un véritable expert**.

J'espère que ces **offres** vous conviendront et vous aideront à **franchir le pas** dans ce monde incroyable du développement mobile avec **Flutter**.

Je vous dis à très vite et surtout n'oubliez pas de **rêver très grand**!