ECOLE DES MINES DE L'INDUSTRIE
ET DE LA GÉOLOGIE

Développement Mobile

2025-2026

Chapitre 1: Introduction au développement mobile

1. Introduction au développement mobile

Aujourd'hui, plus de 6,8 milliards d'utilisateurs de smartphones dans le monde. Le développement mobile consiste à créer des applications conçues pour fonctionner sur des systèmes d'exploitation mobiles comme Android et iOS.

Chaque plateforme possède ses propres environnements, langages et outils de développement, ce qui peut représenter un défi. C'est dans ce contexte que sont nées les technologies multiplateformes, telles que Flutter, permettant de développer une seule base de code pour plusieurs environnements.

Les applications mobiles représentent un canal majeur pour :

- La communication (WhatsApp, Messenger)
- Les services (banques, santé, transport)
- Le divertissement (jeux, streaming, réseaux sociaux)

Les entreprises adoptent une **stratégie mobile-first** pour atteindre leurs clients.

1.1 Évolution du développement mobile

- Première génération (2000-2010) : Applications natives simples
- Deuxième génération (2010-2015) : Explosion des apps natives iOS/Android.
- Troisième génération (2015 à aujourd'hui) :
  - Applications hybrides
  - Frameworks cross-platform performants (Flutter, React Native).
- Tendance actuelle : applications multiplateformes et progressive web apps (PWA).

1.2 Types de développement mobile

Type d'application	Technologies	Avantages	Inconvénients / Limites
Applications natives	Android : Java/Kotlin iOS : Swift/Objective	Performances optimales Accès complet au matériel du smartphone	Coûts plus élevés (nécessité de deux équipes distinctes)
Applications hybrides	HTML, CSS, JS encapsulé (Ionic, Cordova)	Code unique pour plusieurs plateformes	Performances parfois inférieures
Applications cross-platform	• Flutter (Google) • React Native (Meta)	• Productivité accrue • Communauté active • Performances proches du natif	Peut nécessiter des ajustements spécifiques pour chaque plateforme
Progressive Web Apps (PWA)	Applications web évoluées, installables depuis un navigateur	Légères Pas besoin de passer par les stores	Accès restreint aux fonctionnalités matérielles

2. Environnement et systèmes d'exploitation

2.1 Définition d'un système d'exploitation

Un système d'exploitation (OS) est le logiciel de base qui permet à un appareil (ordinateur, smartphone, tablette) de fonctionner. Il gère :

- Les ressources matérielles (processeur, mémoire, batterie, capteurs, caméra, etc.).
- Les interactions entre l'utilisateur et le matériel (via une interface graphique).
- L'exécution des programmes et applications.

Sans système d'exploitation, un smartphone n'est qu'un matériel inerte.

2.2 Les principaux OS mobiles

Actuellement, le marché est dominé par deux systèmes :

Android (Google)

- Basé sur un noyau Linux.
- Open-source, mais maintenu par Google.
- Large part de marché (plus de 70%).
- Très répandu en Afrique, notamment au Niger grâce à des smartphones abordables.
- Langages natifs : Java et Kotlin.



iOS (Apple)

- Exclusif aux iPhones et iPads.
- Système fermé, optimisé pour le matériel Apple.
- Réputé pour sa sécurité et sa fluidité.
- Langages natifs : Swift et Objective-C.



2.3 Comparaison Android vs iOS

Critère	Android	iOS
Part de marché	Majoritaire (70%)	25%
Ouverture	Open-source, adaptable par les fabricants	Système fermé, réservé à Apple
Langages natifs	Java, Kotlin	Swift, Objective-C
Matériel	Multimarques (Samsung, Tecno, Infinix,)	Exclusivement iPhone/iPad
Distribution des apps	Google Play Store, APK externes	App Store uniquement

Coût moyen des appareils	Accessible, grande variété de prix	Plus élevé

2.4 Environnement de développement mobile

Pour développer des applications mobiles, il faut :

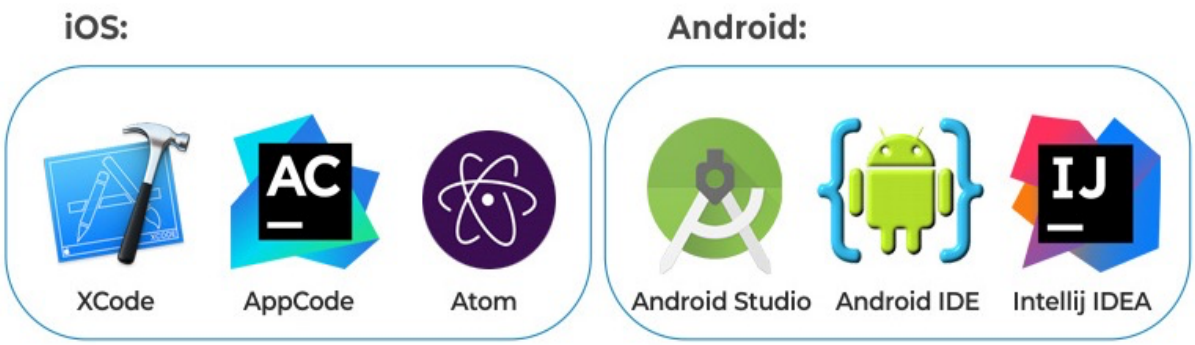
- Un système d'exploitation poste de travail : Windows, Linux, macOS.
- Un environnement de développement intégré (IDE) : Android Studio ou Visual Studio Code.
- SDK (Software Development Kit) : outils et bibliothèques nécessaires pour compiler et exécuter les applications.
- Émulateurs ou smartphones physiques pour tester les applications.

2.5 Approches de developpement d'applications mobiles

2.5.1 Dévéloppement Native(Apps natives)

Les apps natives sont pour ainsi dire la manière classique d'implémenter une application.

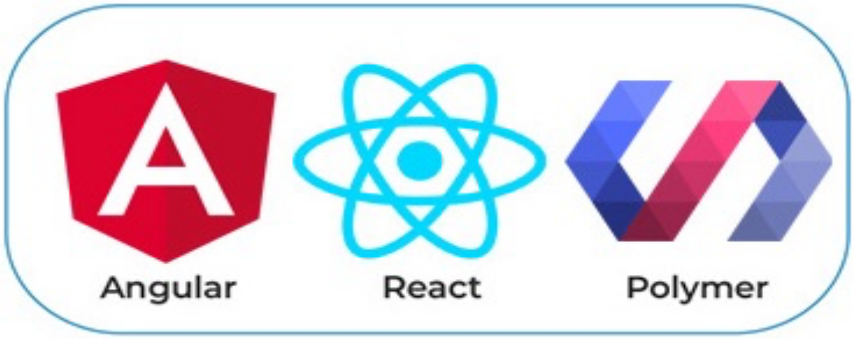
Programmées en Objective-C pour iOS, en Java pour Android et en C# ou VB.Net pour Windows Phone 7, les apps natives peuvent être mises à disposition des utilisateurs via les app stores correspondants.



2.5.2 Dévéloppement Web Progressive (Apps web)

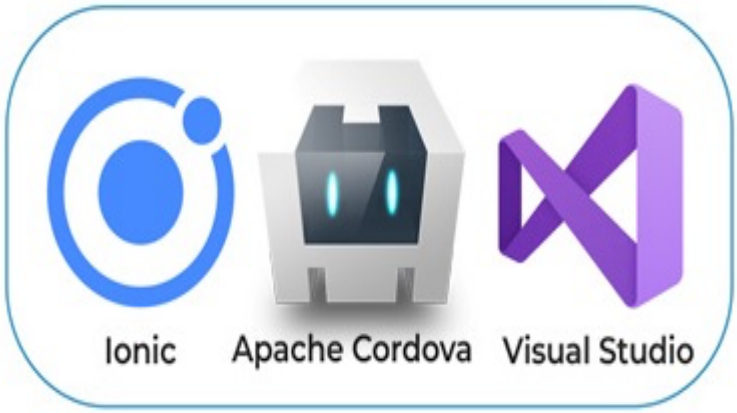
HTML5, CSS et JavaScript permettent d'ores et déjà de créer des pages web à peine différentes des applications de bureau.

Cette approche de développement mobile est un nom relativement nouveau sur le marché. Dans une certaine mesure, les applications Web progressives ressemblent à des applications hybrides. Elles sont basés sur des technologies Web et fonctionnent dans un navigateur.



2.5.3 Développement Hybride

Une application **hybride** combine du code web (HTML, CSS, JavaScript) intégré dans un conteneur natif. Elle s'exécute via une WebView.

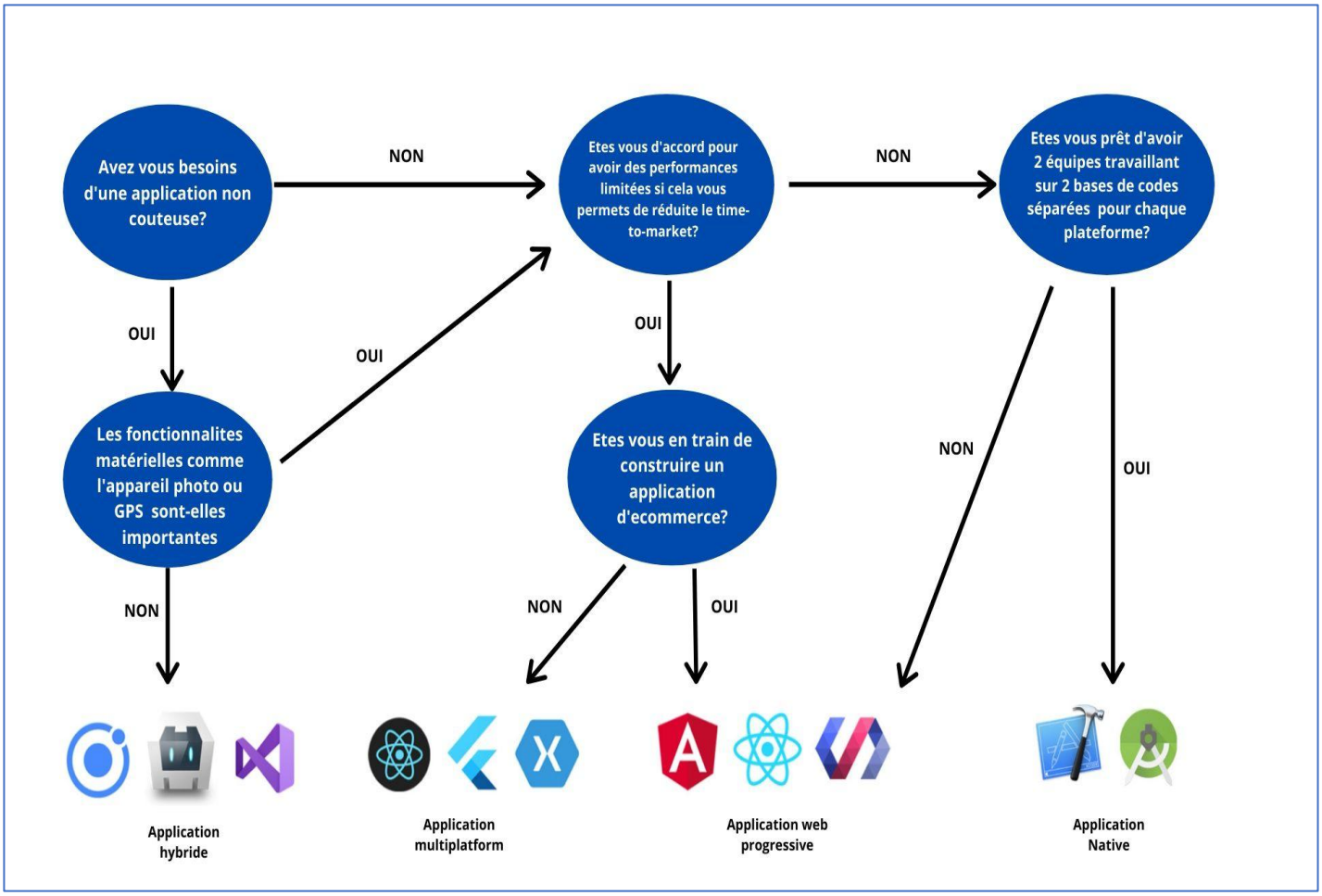


2.5.4 Développement multiplateforme

Une application **multiplateforme** est développée **avec un langage unique** (comme Dart, C#, etc.) mais compilée en code natif pour chaque plateforme (Android, iOS…).



2.5.5 En resumé





- ❑ Sa rapidité de développement
- ❑ Son interface utilisateur très flexible
- ❑ Des performances natives (Performance proche du natif)
- ❑ Développement multiplateforme avec un seul code source
- ❑ Hot Reload pour tester rapidement les changements.
- ❑ Grande bibliothèque de widgets personnalisables.
- ❑ En entreprise, cela représente un gain de temps et de coûts, car un seul projet peut être déployé sur plusieurs plateformes.



3. Présentation de l'écosystème Android

3.1 Qu'est-ce qu'Android ?

Android est un système d'exploitation mobile open-source, développé initialement par la société Android Inc., puis racheté par Google en 2005.

La première version commerciale a été lancée en 2008. Aujourd'hui, Android est le système mobile le plus utilisé dans le monde, notamment en Afrique où il occupe plus de 80% de parts de marché grâce à son accessibilité et sa diversité d'appareils.

3.2 Architecture d'Android

L'architecture d'Android repose sur plusieurs couches principales :

1. **Noyau Linux**
   - Gère le matériel (CPU, mémoire, batterie, capteurs, caméra).
   - Fournit une base stable et sécurisée.
1. **Bibliothèques natives**
   - Ensemble de composants pour gérer les graphiques (OpenGL), bases de données (SQLite), navigation web, etc.
1. **Android Runtime (ART)**
   - Machine virtuelle qui exécute le code des applications.
   - Remplace Dalvik (ancienne VM).
   - Optimisé pour les performances et la gestion de la mémoire.
1. **Framework d'applications**
   - Fournit les services et API permettant aux développeurs de créer leurs applications.
   - Exemple : gestion des activités, services, ressources, notifications.
1. **Applications**
   - Inclut les apps natives (Téléphone, Messages, Paramètres).
   - Les apps installées par l'utilisateur via le Google Play Store ou des fichiers APK.

3.3 L'écosystème Android pour les développeurs

L'écosystème Android ne se limite pas au système d'exploitation : il comprend un ensemble d'outils et de services qui facilitent le développement :

- Android Studio : IDE officiel pour créer des applications Android.
- SDK Android : bibliothèques et outils de compilation.
- ADB (Android Debug Bridge) : outil de communication entre le PC et l'appareil.
- AVD (Android Virtual Device) : permet de simuler différents modèles de smartphones.
- Google Play Services : fournit des API pour la géolocalisation, Google Maps, Firebase, notifications push, .
- Google Play Console : plateforme pour publier, gérer et analyser les applications distribuées.

3.4 Avantages d'Android dans le contexte professionnel

- Ouverture : possibilité de développer et distribuer facilement des applications (via Play Store ou fichiers APK).
- Diversité matérielle : disponible sur une large gamme de smartphones et tablettes à tous les prix.
- Flexibilité : possibilité de personnaliser fortement l'interface et les fonctionnalités.
- Communauté : énorme communauté de développeurs, documentation et support abondants.

4. Installation des outils de développement (Android Studio, SDK, Flutter SDK, émulateurs)

4.1 Objectif

Mettre en place un environnement complet de développement Flutter afin que chaque participant puisse créer, exécuter et tester des applications mobiles sur Android (émulateur et/ou smartphone physique).

4.2 Outils nécessaires

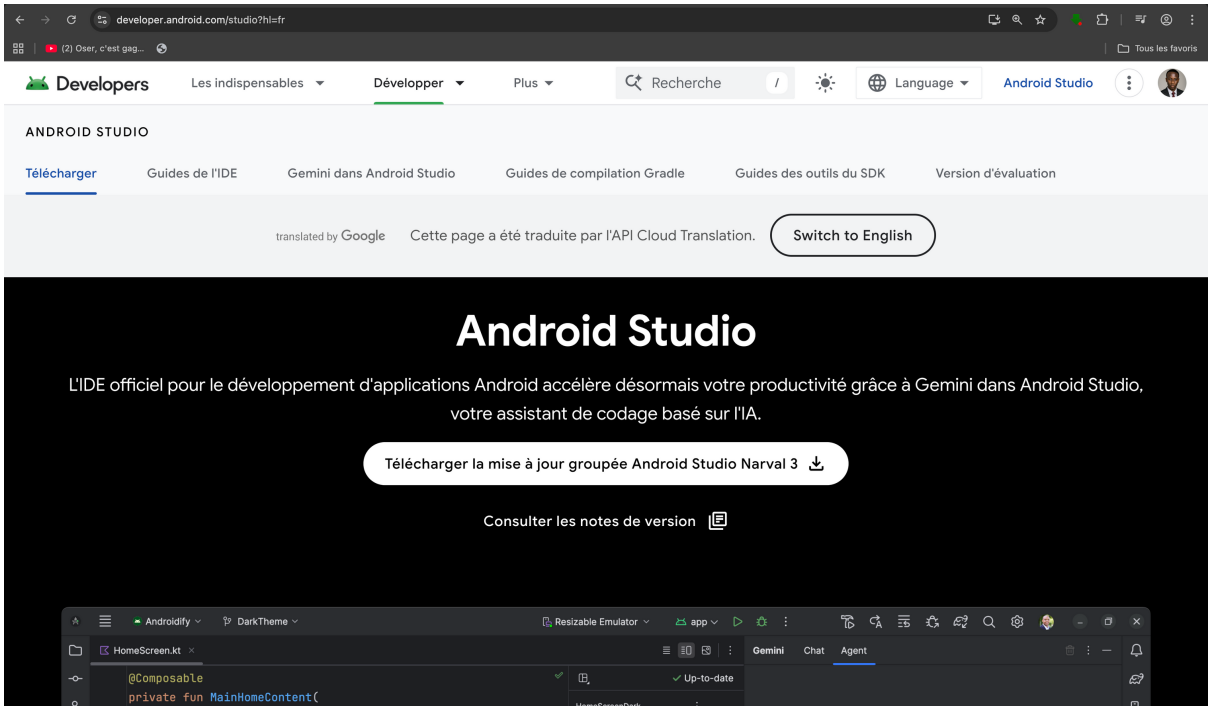
1. **Flutter SDK**
   - C'est le framework principal qui contient toutes les bibliothèques, outils et commandes nécessaires pour créer des applications Flutter.
2. **Dart SDK**
   - Inclus dans Flutter, il permet de compiler et exécuter le langage Dart.
3. **Android Studio**
   - IDE officiel recommandé par Google.
   - Fournit le SDK Android, les outils de compilation, et le gestionnaire d'émulateurs (AVD).
4. **Visual Studio Code (optionnel)**
   - IDE léger qui peut être utilisé à la place d'Android Studio.
   - Nécessite l'installation des extensions **Flutter** et **Dart**.
5. **Émulateurs Android (AVD)**
   - Permettent de simuler différents modèles de smartphones virtuels.
6. **Smartphone Android physique (optionnel)**
   - Pour tester directement les applications sur un appareil réel.
   - Nécessite d'activer le mode développeur et le débogage USB.

4.3 Étapes d'installation

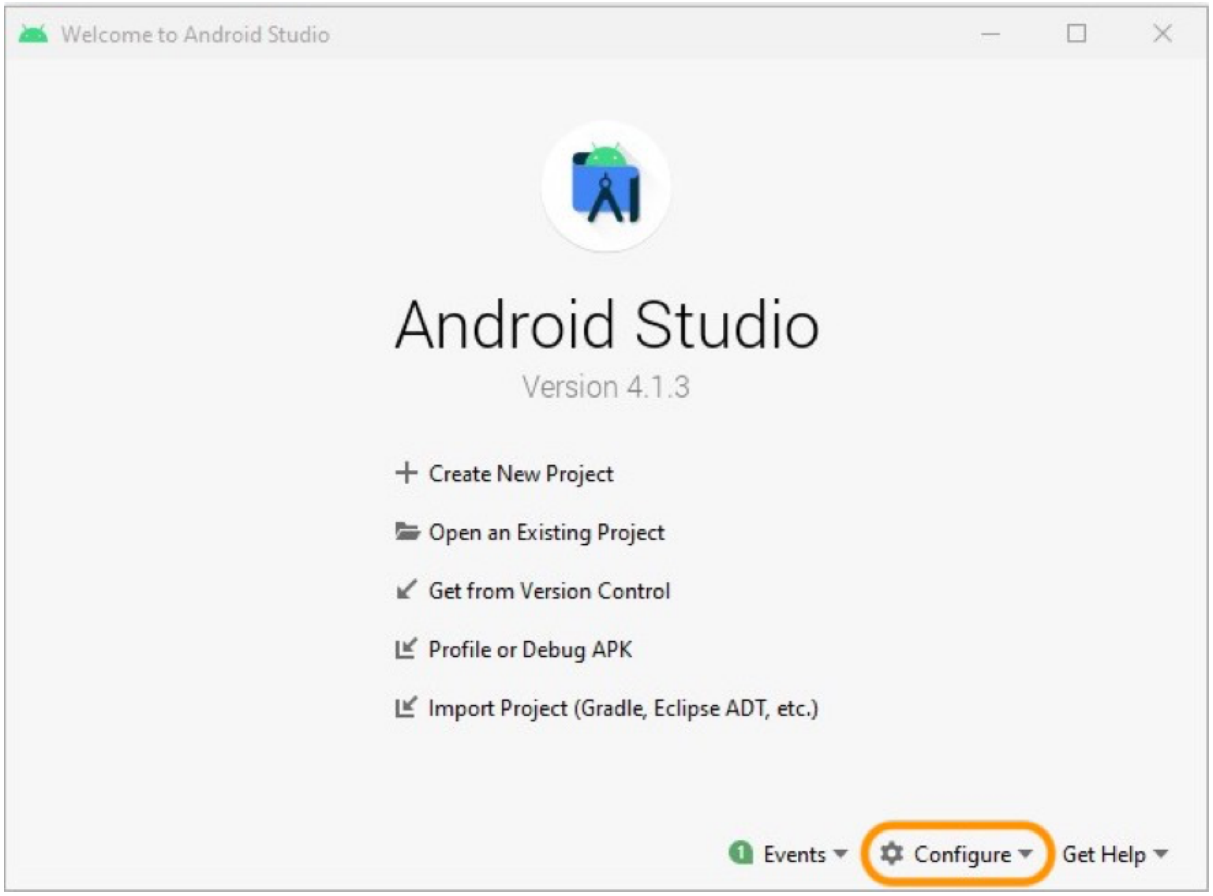
4.3.1 Installer Android Studio

✓ **Etape 1 :** Télécharger et installer Android Studio depuis: https://developer.android.com/studio.

La page détecte automatiquement votre système d'exploitation, et propose de télécharger l'archive compatible.



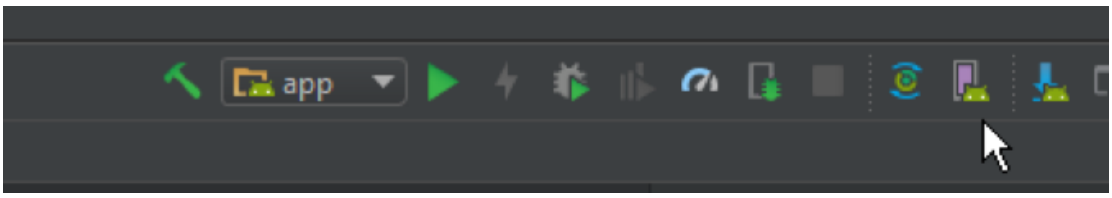
En Suivant les instructions d'installation, nous parvenons à lancer l'application. Nous voyons apparaître une fenêtre. Il ne reste plus qu'à configurer l'environnement en cliquant sur Configure.



✓ **Étape 2 :** Configuration de l'émulateur Android

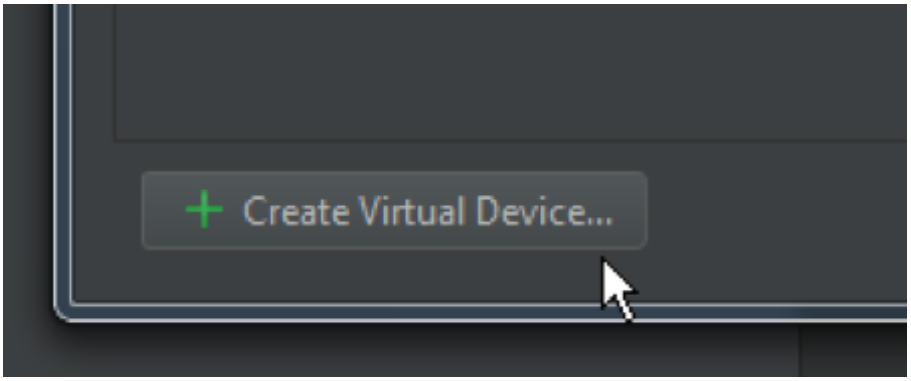
2.a. Pour Configurer l'émulateur Android, on suit les étapes ci-dessous :

- Dans Android Studio, repérez la **barre d'outils supérieure**.
- Cliquez sur l'icône **AVD Manager** (ou via le menu : *Tools > Device Manager*).
- Une fenêtre de gestion des appareils virtuels s'ouvre.



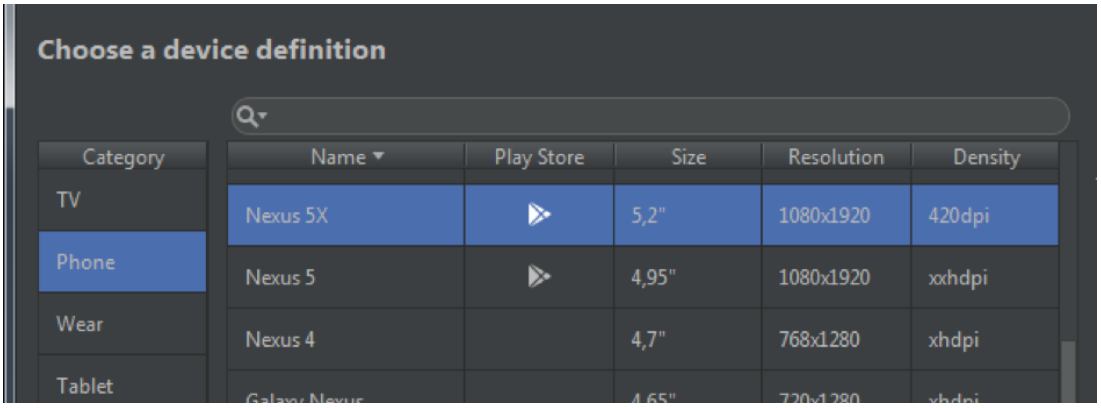
2.b Dans la fenêtre AVD Manager :

- Cliquez sur le bouton **Create Virtual Device** situé en bas à gauche.
- Cela permet de lancer l'assistant de création d'un nouvel émulateur.



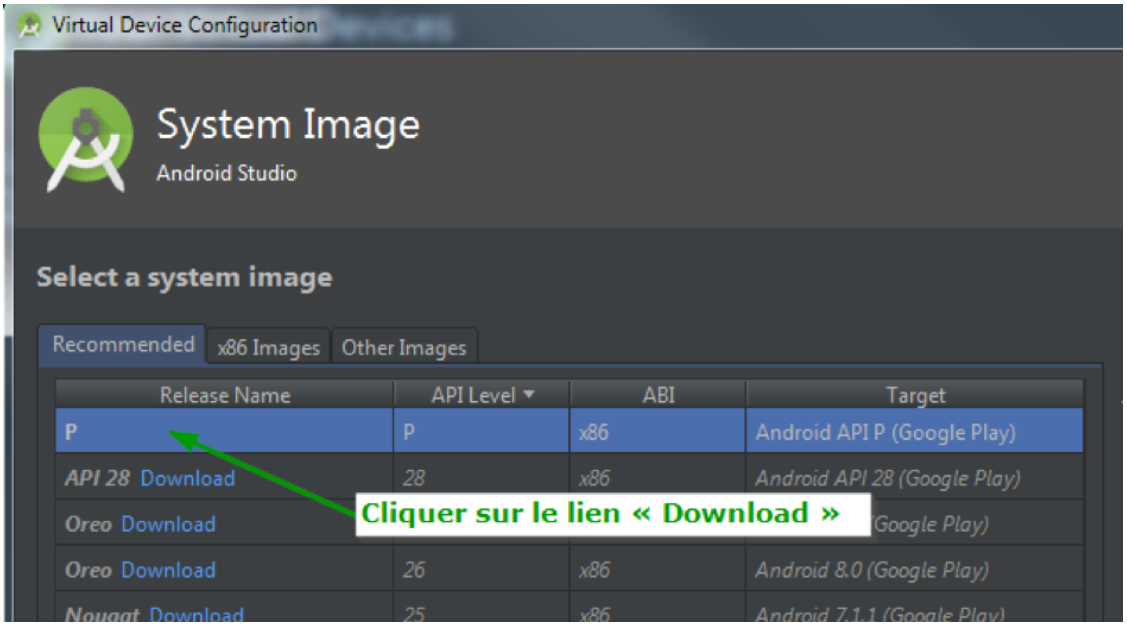
2.c Choisir un modèle de mobile :

- Une liste de modèles de smartphones est proposée.
- Sélectionnez un modèle recommandé, par exemple **Pixel 5**, car il est stable et compatible avec la plupart des images système.
- Cliquez sur **Next** pour continuer.



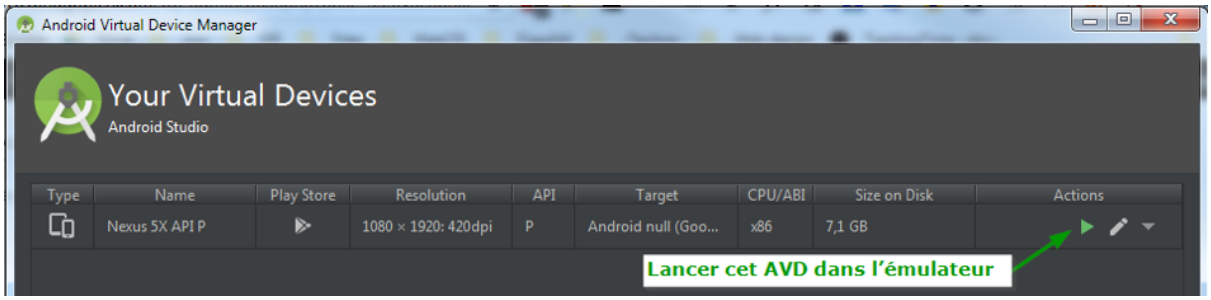
2.d. Télécharger et sélectionner une image système :

- L'assistant propose plusieurs versions d'Android (API Level).
- Sur la première ligne marquée **Recommended**, cliquez sur **Download** pour installer l'image système correspondante.
- Une fois le téléchargement terminé, sélectionnez l'image système et cliquez sur **Next**.



2.e Configurer et lancer l'émulateur :

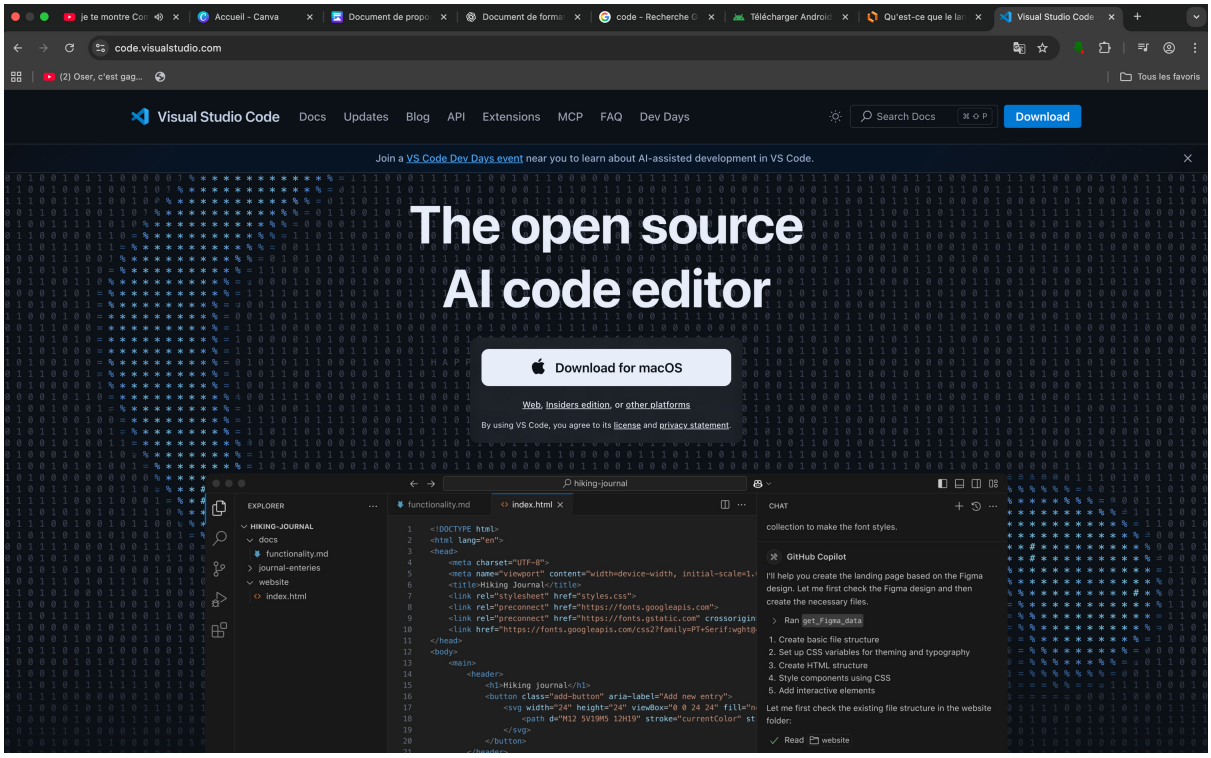
- Dans l'écran de configuration finale, vous pouvez personnaliser certains paramètres (Nom de l'appareil virtuel, Taille de la mémoire (RAM), Orientation de l'écran (portrait ou paysage))
- Cliquez sur **Finish** pour créer l'émulateur.
- Dans AVD Manager, cliquez sur le bouton ▶ Launch à côté de l'appareil pour démarrer l'émulateur.



2. Installation de Visual Studio code

✓ **Etape 1:** Télécharger l'installateur

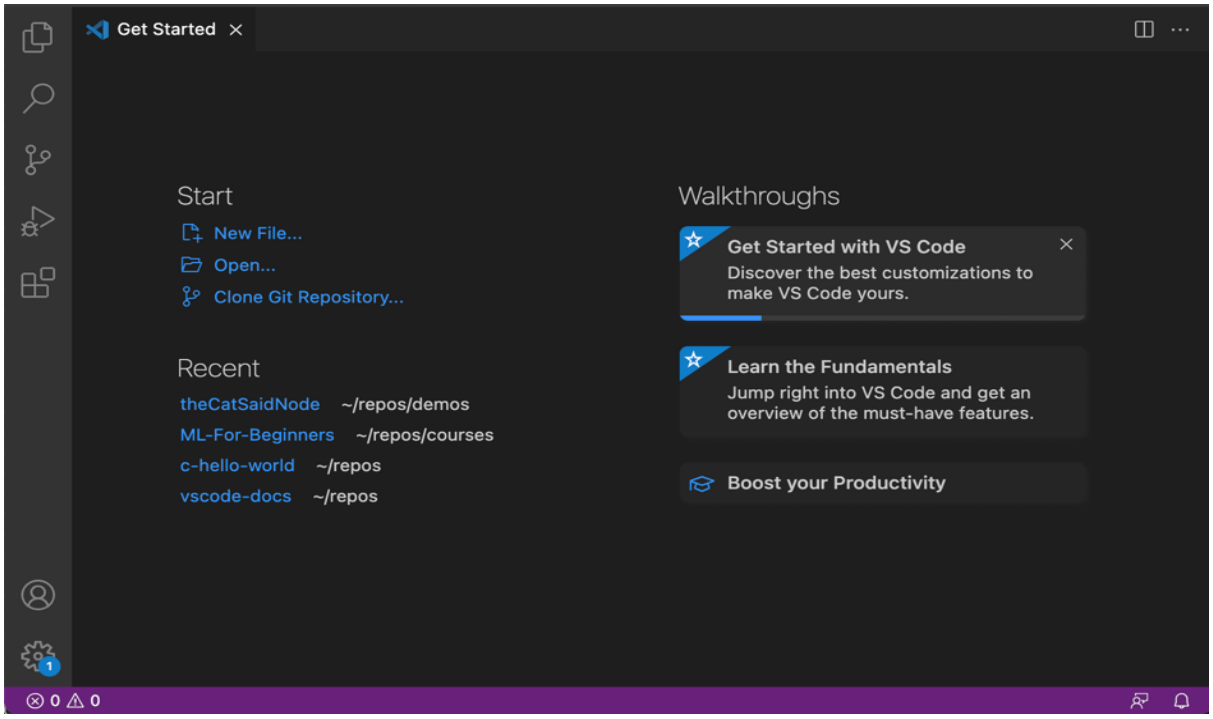
- Rendez-vous sur le site officiel : https://code.visualstudio.com
- Cliquez sur **Download for Windows**.
- Choisissez la version adaptée à votre processeur :
- **User Installer 64-bit** → recommandé pour la plupart des ordinateurs récents.
- **User Installer 32-bit** → pour les anciens PC.
- **ARM** → pour les machines équipées de processeurs ARM.



✓ **Etape 2 :** Lancer l'installation

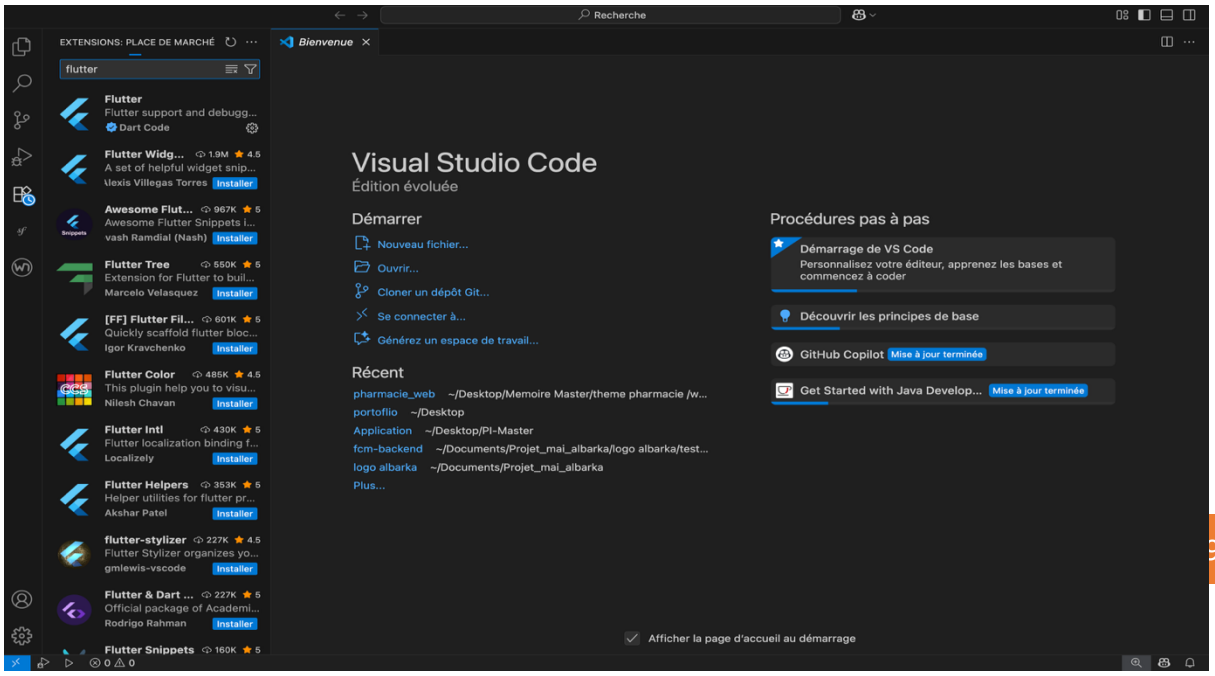
- Ouvrez le fichier téléchargé (VSCodeUserSetup-x64.exe ou équivalent).
- Suivez l'assistant d'installation en cliquant sur **Next**.
- Acceptez les conditions d'utilisation.
- Sélectionnez le dossier d'installation par défaut (recommandé).
- Cochez les options suivantes :
  - **Add to PATH (recommended)** → permet d'utiliser VS Code depuis le terminal.
  - **Register Code as an editor for supported file types.**

Cliquez sur Install pour lancer l'installation.



✓ Etape 3 : Premier lancement et configuration

- Une fois l'installation terminée, lancez **Visual Studio Code**.
- Installez les **extensions nécessaires** pour Flutter :
  a. Cliquez sur l'icône **Extensions** (à gauche ou Ctrl+Shift+X).
  b. Recherchez et installez :
    - **Flutter** (officiel)
    - **Dart**
- Redémarrez VS Code pour activer les extensions.



3. Installer Flutter SDK sous Windows

- ✓ **Étape 1 :** Télécharger le SDK de Flutter depuis le site officiel : https://flutter.dev ;
- ✓ **Étape 2 :** Décompresser le dossier dans C:\src\flutter (Windows) ;
- ✓ **Étape 3 :** Ajoutez Flutter à votre **PATH ;**

Si vous souhaitez exécuter les commandes Flutter dans la console Windows ordinaire, procédez comme suit pour ajouter Flutter à la variable d'environnement **PATH** :

- Dans la barre de recherche du menu Démarrer, entrez **" environnement "** et sélectionnez **" Modifier les variables d'environnement "** de votre compte.
- Sous Variables utilisateur, vérifiez s'il existe une entrée appelée Path
- Si l'entrée existe, ajoutez le chemin complet à **flutter\bin** en utilisant ; comme séparateur des valeurs existantes.
- Si l'entrée n'existe pas, créez une nouvelle variable utilisateur nommée Path avec le chemin complet vers **flutter\bin** comme valeur.
- Après fermer et rouvrir toutes les fenêtres de console existantes pour que ces modifications prennent effet.

En fin nous pouvons tester notre installation en exécutant la commande suivante pour voir s'il y a des dépendances que nous devons installer pour compléter la configuration.

Sur le terminal taper la commande: flutter doctor

Réponse :

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.0.5, on macOS 12.5.1 21G83 darwin-x64, locale en-FR)
[✓] Android toolchain - develop for Android devices (Android SDK version 31.0.0)
[✓] Xcode - develop for iOS and macOS (Xcode 13.4.1)
[✗] Chrome - develop for the web (Cannot find Chrome executable at
 /Applications/Google Chrome.app/Contents/MacOS/Google Chrome)
 ! Cannot find Chrome. Try setting CHROME_EXECUTABLE to a Chrome executable.
[✓] Android Studio (version 2020.3)
[✓] VS Code (version 1.71.0)
[✓] Connected device (1 available)
[✓] HTTP Host Availability
! Doctor found issues in 1 category.
```