

Atelier 1

Matière : ATELIER FRAMEWORK CROSS-PLATFORM

DSI3

Enseignants : S. Hadhri & M. Hadiji

I. PRÉSENTATION DE FLUTTER

Flutter est un framework Google qui permet de développer des applications esthétiques compilées de manière native pour les mobiles, le Web et les ordinateurs de bureau, à partir d'une seule base de code. Le site officiel de Flutter est : <https://flutter.dev/>

Flutter est dévoilé par Google pour la première fois en 2015 et la première version stable est publiée en Décembre 2018. Il utilise le langage de programmation DART qui est également une création de Google, inventé en 2011. Le site officiel de Dart est : <https://dart.dev/>

Initialement utilisé pour concevoir des applications cross plateforme, Flutter permet maintenant de produire des applications web et aussi des applications desktop pour Windows, MacOS et Linux. Ces applications s'adaptent aussi bien aux systèmes d'exploitation mobiles qu'à ceux intégrés au web, ordinateurs et TV connectés.

Le choix de Flutter se justifie par les nombreux avantages qui le caractérisent. Il est par exemple simple d'utilisation. C'est un outil pratique et très facile à manipuler. Il fournit à ses utilisateurs une documentation complète et propose de nombreux avantages :

- La qualité de ses widgets (il propose une panoplie de widgets qui facilitent la construction d'interface) ;
- La rapidité d'exécution (grâce à la fonctionnalité Hot Reload du langage DART, le build des applications est très rapide, ce qui rend quasiment invisible le temps de compilation) ;
- Les outils de suivi (il propose des mesures de suivi très simples) ;
- La complétude (contrairement à ses concurrents, Flutter est un outil autonome et qui bug très rarement).
- L'environnement Google avec Firebase
- Une communauté présente et à l'écoute

Remarque :

Il est possible d'ajouter des packages dans les applications Flutter grâce à l'outil Pub. Il s'agit d'un gestionnaire de packages pour le langage de programmation Dart, contenant des bibliothèques et des packages réutilisables pour Flutter, AngularDart et les programmes Dart : <https://pub.dev/>

II. INSTALLATION FLUTTER

Les étapes d'installation Flutter sont détaillées dans l'adresse web suivante :

<https://docs.flutter.dev/get-started/install/windows>

1. Obtenir le SDK Flutter

1. Télécharger le bundle d'installation suivant pour obtenir la dernière version stable du SDK Flutter :
https://storage.googleapis.com/flutter_infra_release/releases/stable/windows/flutter_windows_3.13.4-stable.zip
2. Extraire le fichier zip et placer le contenu dans l'emplacement d'installation souhaité pour le SDK Flutter (par exemple C:\Flutter\Flutter_SDK).

2. Mettre à jour le path

Sous variables utilisateur des variables d'environnement, vérifier s'il existe une entrée appelée Path et ajouter le chemin complet à C:\Flutter\Flutter_SDK\bin en utilisant ; comme séparateur des valeurs existantes.

NB :

- Il est impératif de fermer et rouvrir toutes les fenêtres de console existantes pour que ces modifications prennent effet.
- Pour vérifier que Flutter et dart sont installés, taper dans l'invite de commande :
`where flutter dart`
- Pour voir s'il existe des dépendances de plate-forme nécessaires pour terminer la configuration, exécuter la commande suivante :
`flutter doctor`

3. Configuration pour Android

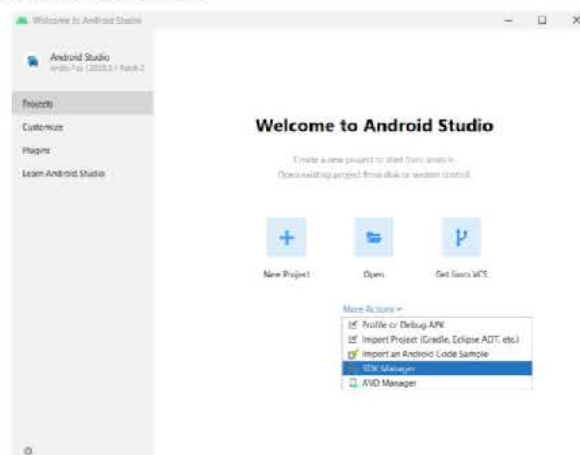
Flutter s'appuie sur une installation complète d'Android Studio pour fournir ses dépendances de la plateforme Android. Cependant, il est possible d'écrire des applications Flutter dans certains éditeurs ;

3.1. Installer Android Studio

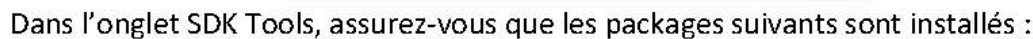
1. Télécharger et installer Android Studio.
2. Démarrer Android Studio et passez par « l'assistant de configuration d'Android Studio ». Cela installe le dernier Android SDK, Android SDK Command-line Tools, and Android SDK Build-Tools, qui sont requis par Flutter lors du développement pour Android.

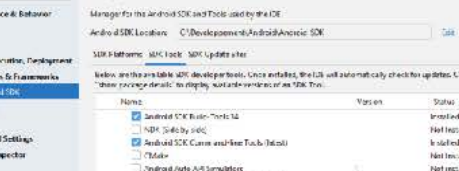
NB :

Si vous avez une installation différente du Android SDK, il faudrait la lier à Android Studio : More Actions → SDK Manager, cliquez ensuite sur Edit de l'option Android SDK Location afin de sélectionner le chemin du SDK..



- Android SDK Platform
- Google Play ... System Image



- 
- The screenshot shows the 'Settings' dialog in Android Studio, with the 'Languages & Frameworks' tab selected. The 'Android SDK' section is expanded, showing a list of installed and available SDK components. The 'Tools' section is also expanded, showing the 'Android SDK Platform-Tools' and 'Android SDK Build-Tools' are installed. The 'System' section is also expanded, showing the 'Android Studio' and 'Android SDK' are installed. The 'Android Studio' section is also expanded, showing the 'Android Studio' and 'Android SDK' are installed.
- | Name | Version | Status |
|---|---------|---------------|
| <input checked="" type="checkbox"/> Android Studio Build-Tools 14 | | Installed |
| <input checked="" type="checkbox"/> Android Studio 4.0.0 | | Not installed |
| <input checked="" type="checkbox"/> Android SDK Command Line Tools (x86_64) | | Not installed |
| <input type="checkbox"/> CMake | | Not installed |
| <input type="checkbox"/> Android Auto AVI Simulators | | Not installed |
| <input checked="" type="checkbox"/> Android Auto Desktop Head Unit Controller | 2.0 | Not installed |
| <input checked="" type="checkbox"/> Android Emulator | 32.3.0 | Installed |
| <input checked="" type="checkbox"/> Android Emulator Hypervisor Driver (x86_64) | 26.0 | Not installed |
| <input checked="" type="checkbox"/> Android SDK Platform-Tools | 34.0.4 | Installed |
| <input checked="" type="checkbox"/> Google Play APK Expansion Library | + | Not installed |
| <input checked="" type="checkbox"/> Google Play In-App Billing Development SDK | 6.0 | Not installed |
| <input checked="" type="checkbox"/> Google Play Licensing - binary | - | Not installed |
| <input checked="" type="checkbox"/> Google Play Services | 49 | Not installed |
| <input checked="" type="checkbox"/> Google Play Services | 15 | Installed |
| <input checked="" type="checkbox"/> Google Web Linker | 2 | Not installed |
| <input checked="" type="checkbox"/> Intel X86 Emulator Accelerator (x86_64) (installed) | 7.6.5 | Installed |
| <input checked="" type="checkbox"/> Layered In-App Billing - In-App Billing | 6 | Not installed |
| <input checked="" type="checkbox"/> Layered In-App Billing - In-App Billing | 3 | Not installed |
- ☒ Hide Obsolete Packages

- ### 3.2. Accepter les licences Android

AFCP

1. S'assurer qu'une version de Java est installée et que la variable d'environnement JAVA_HOME est définie sur le dossier du JDK.

NB :

Les versions 2.2 et supérieures d'Android Studio sont livrées avec un JDK, cela devrait donc déjà être fait.

2. Ouvrir une fenêtre de console et exécutez la commande suivante pour commencer à signer des licences.
`flutter doctor --android-licenses`
3. Lire attentivement les termes de chaque licence avant de les accepter.
4. Une fois que toutes les licences sont acceptées, exécuter à nouveau `flutter doctor` pour vérifier que Flutter est prêt à l'utilisation.

III. CONFIGURER ANDROID STUDIO

Il est possible de créer des applications avec Flutter en utilisant n'importe quel éditeur de texte combiné aux outils de ligne de commande. Cependant, il est recommandé d'utiliser Android studio comme éditeur auquel il faut ajouter 2 plugins Flutter et Dart: Android Studio offre une expérience IDE complète et intégrée pour Flutter (Android Studio, version 2020.3.1 -Arctic Fox ou ultérieure)
<https://developer.android.com/studio>

Pour installer les plugins Flutter et Dart, utiliser les instructions suivantes :

1. Ouvrir les préférences du plugin (Fichier > Paramètres > Plugins).
2. Sélectionner Marketplace, sélectionner le plugin Flutter et cliquer sur Installer.

IV. CRÉER UNE PREMIÈRE APPLICATION FLUTTER AVEC ANDROID STUDIO

Dans cette partie, nous allons voir comment créer une nouvelle application Flutter à partir de modèles, l'exécuter et faire l'expérience d'un rechargement à chaud (hot reload) après avoir apporté des modifications à l'application.

1. Créer l'application

1. Ouvrir l'IDE et sélectionner Nouveau projet Flutter.
2. Sélectionner Flutter, vérifiez le chemin du SDK Flutter avec l'emplacement du SDK. Cliquer ensuite sur Suivant.
3. Entrer un nom de projet (par exemple, my_app).
4. Sélectionner Application comme type de projet.
5. Cliquer sur Terminer.

NB :

Les commandes ci-dessus créent un répertoire de projet Flutter appelé my_app qui contient une application de démonstration simple qui utilise Material Components.

Le code de l'application créée se trouve dans lib/main.dart.

2. Configurer l'émulateur Android

Pour préparer l'exécution et le test de l'application Flutter sur l'émulateur Android, suivre ces étapes :

1. Activer l'accélération de la machine virtuelle sur l'ordinateur.

2. Lancer Android Studio, cliquer sur l'icône AVD Manager et sélectionner Créer un périphérique virtuel...
 - Dans les anciennes versions d'Android Studio, vous devez plutôt lancer Android Studio > Tools > Android > AVD Manager et sélectionner Create Virtual Device.... (Le sous-menu Android n'est présent que dans un projet Android.)
 - Si aucun projet n'est ouvert, choisir Configurer > AVD Manager et sélectionner Créer un périphérique virtuel...
3. Choisir une définition de périphérique et sélectionner Suivant.
4. Sélectionner une ou plusieurs images système pour les versions d'Android souhaitées, puis sélectionner Suivant. Une image x86 ou x86_64 est recommandée.
5. Sous Performances émulées, sélectionner Matériel - GLES 2.0 pour activer l'accélération matérielle.
6. Vérifier que la configuration AVD est correcte et sélectionner Terminer.
7. Dans Android Virtual Device Manager, cliquer sur Exécuter dans la barre d'outils. L'émulateur démarre et affiche le canevas par défaut pour la version du système d'exploitation et l'appareil sélectionné.

3. Configurer un appareil Android réel

Pour préparer l'exécution et le test de votre application Flutter sur un appareil Android, il faudrait avoir un appareil Android exécutant Android 4.1 (API niveau 16) ou supérieur.

1. Activer les options de développeur et le débogage USB sur votre appareil.
 - a. Ouvrir l'application "Paramètres" sur l'appareil Android.
 - b. Défiler vers le bas et recherchez l'option "À propos du téléphone" ou "À propos de l'appareil". Sélectionner-la.
 - c. Rechercher la section "Numéro de build" ou "Version logicielle". Appuyer plusieurs fois rapidement (généralement 7 fois) sur cette entrée. Entrer le code PIN ou le mot de passe de l'appareil si nécessaire : Un message indiquant que les options de développement ont été activées est alors affiché.
 - d. Retourner aux "Paramètres". Chercher la nouvelle section appelée "Options pour les développeurs" ou "Options de développement".
 - e. Ouvrir cette section et rechercher l'option "Débogage USB". Activer-la en basculant le commutateur.
2. À l'aide d'un câble USB, brancher l'appareil Android à l'ordinateur. Si un message s'affiche, autoriser l'ordinateur à accéder à l'appareil.
3. Dans le terminal, exécutez la commande `Flutter Devices` pour vérifier que Flutter reconnaît l'appareil Android connecté. Par défaut, Flutter utilise la version du SDK Android sur laquelle est basé l'outil adb.

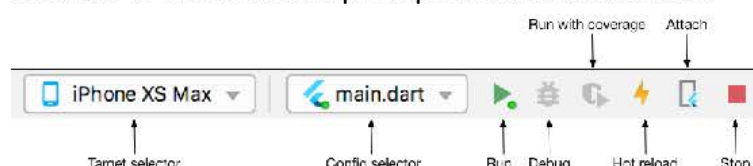
NB :

Si l'appareil n'est pas détecté, installer le driver relatif au constructeur :

<https://developer.android.com/studio/run/oem-usb>

4. Exécuter l'application

1. Localiser la barre d'outils principale d'Android Studio :



2. Dans target selector, sélectionner un appareil Android pour exécuter l'application. Si aucun n'est répertorié comme disponible, sélectionner Outils > Gestionnaire AVD et créez-en un.
3. Cliquer sur l'icône d'exécution dans la barre d'outils ou appeler l'élément de menu Run > Run.

⇒ Une fois la création de l'application terminée, l'application de démarrage se lance sur l'appareil.



5. Essayer le hot reload

Flutter offre un cycle de développement rapide avec Stateful Hot Reload, la possibilité de recharger le code d'une application en cours d'exécution sans redémarrer ni perdre l'état de l'application. Modifier la source de l'application, indiquer à l'IDE utilisé qu'on souhaite recharger à chaud et observer la modification dans l'émulateur ou l'appareil réel.

1. Ouvrir lib/main.dart.
2. Changer la chaîne

Fichier lib/main.dart
'You have pushed the button this many times'
'You have clicked the button this many times'

NB :

N'arrêter pas l'application. Laisser la s'exécuter.

3. Enregistrer les modifications : appeler Enregistrer tout ou cliquer sur l'éclair Recharger à chaud ⚡. La nouvelle chaîne s'affiche dans l'application en cours d'exécution presque immédiatement.