

## TP1

# Prise en main d'Android Studio

License DAWM  
Pr. EL AZAMI-Année 2023

### Introduction

Ce premier TP est une initiation à l'environnement de développement Android Studio que vous utiliserez au cours de ce module. Il a pour but de vous familiariser avec cet environnement et d'illustrer les concepts du SDK Android, au travers du développement d'une application mobile très simple.

### Création d'une application

Lancez l'environnement Android Studio. Vous devez obtenir la fenêtre suivante (figure 1), dans laquelle vous sélectionnerez l'option *Start a new Android Studio Project*. Les projets récemment utilisés apparaîtront ultérieurement dans la partie gauche de la fenêtre, ce qui vous facilitera leur ouverture par la suite.

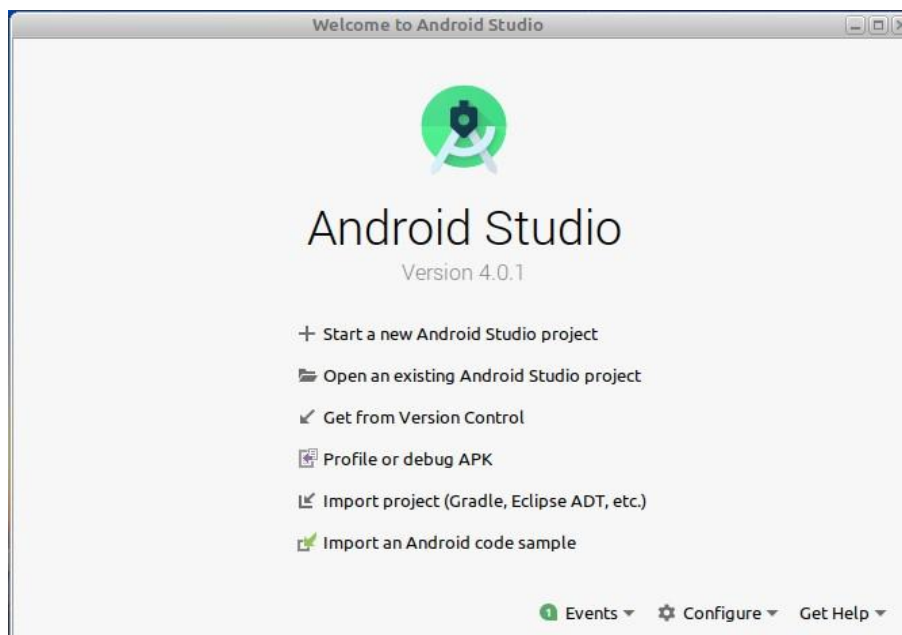


Figure 1 – La fenêtre de lancement initiale d'Android Studio

La première étape consiste ensuite à sélectionner le type d'activité qui sera utilisé pour l'activité initiale de votre application. Comme vous le constatez sur la figure 2, vous disposez d'un choix assez large, qui dépend du type de périphérique que vous ciblez. Pour ce TP, nous utiliserons la *Empty Activity* pour téléphones et tablettes, qui correspond à un écran très simple.

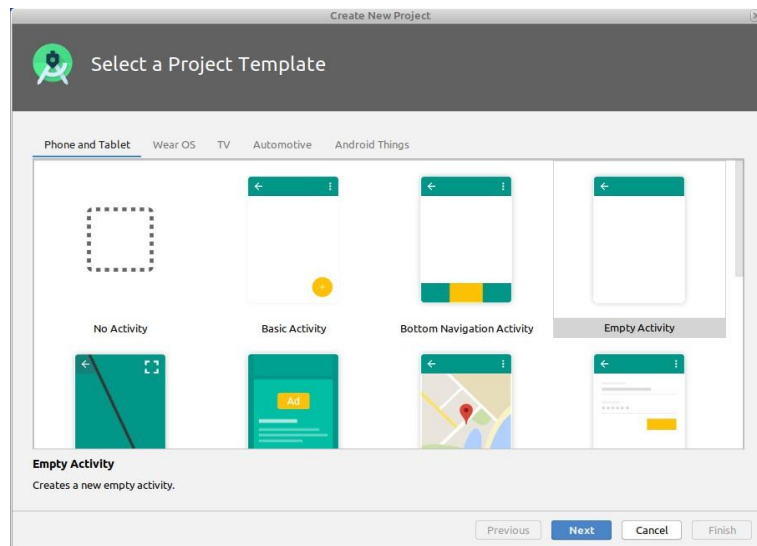


Figure 2 – La première étape de la création d'un nouveau projet

La seconde étape consiste à configurer votre nouveau projet; parmi les paramètres à configurer, qui apparaissent sur l'image de la figure 3, doivent être précisés :

- *Name* : le nom de l'application. Utilisez le nom Application00 qui apparaîtra dans tout le reste de l'énoncé. A noter que le nom des applications doit commencer par une majuscule et qu'il faut éviter qu'il corresponde au nom d'une application disponible dans *Google Play*;
- *Package name* : l'interface vous propose le nom com.example.application00. Il n'est pas utile ici de le modifier;
- *Save location* : par défaut, vos projets sont créés dans le dossier AndroidStudioProjects situé à la racine de votre compte. Il n'est pas nécessaire ici de modifier cette manière de faire.
- *Language* : l'interface vous propose par défaut le langage Kotlin. Choisissez ici Java;
- *Minimum SDK* : Vous avez ici la possibilité de choisir le sdk minimum avec lequel votre application sera compatible. L'interface vous indique que le SDK proposé par défaut (API 16) sera compatible avec environ 99,8% des périphériques Android en circulation. Conservez donc cette valeur.

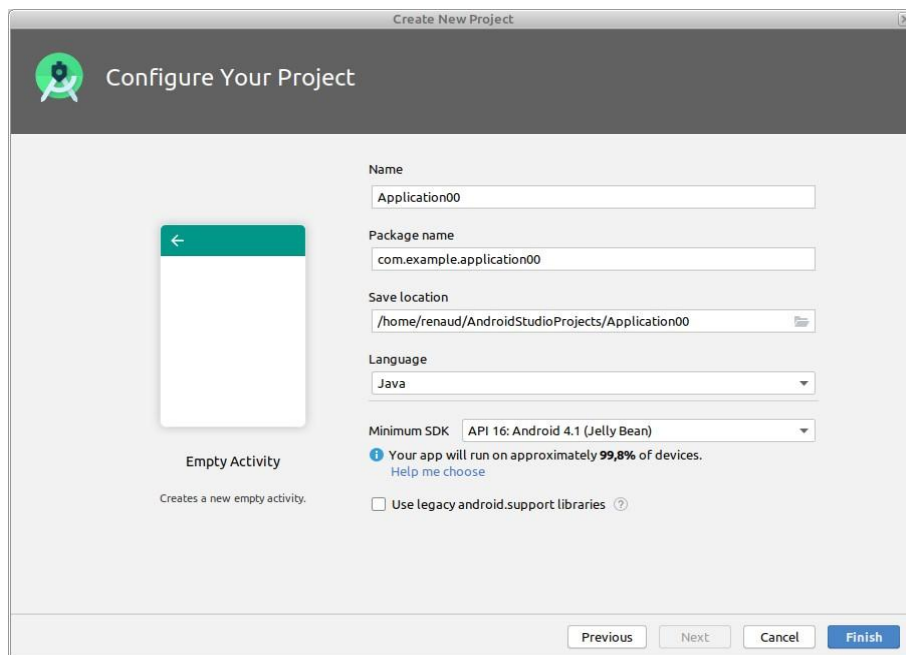


Figure 3 – La seconde étape de la création d'un nouveau projet

Après validation de ce second écran de configuration, vous voyez s'ouvrir l'IDE d'Android Studio, avec l'ensemble des fichiers générés par défaut (figure 4). L'interface est initialement découpée en deux parties adjacentes, l'une pour l'éditeur (partie droite), l'autre pour l'arborescence du projet (partie gauche). Le nombre de zones présentes dans l'IDE évoluera évidemment en fonction de vos actions et de vos choix.

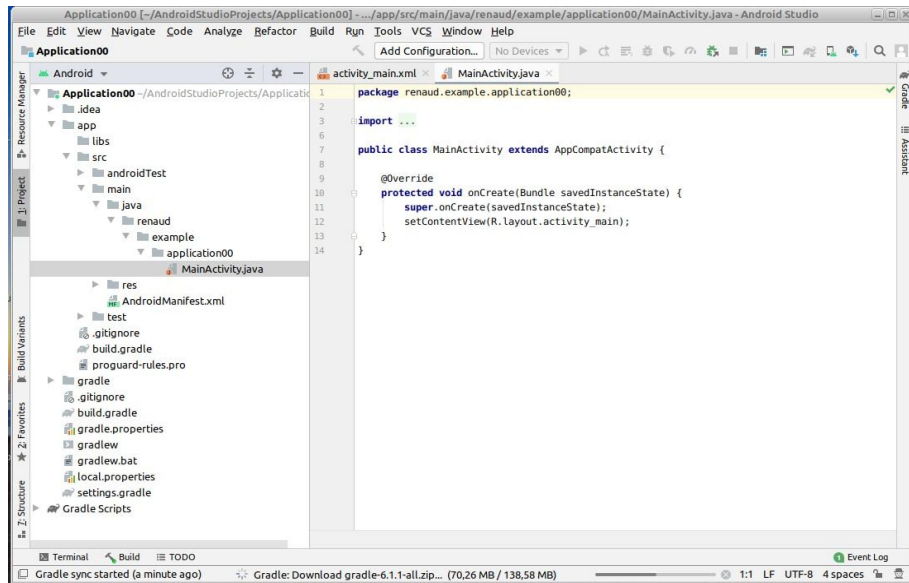



Figure 4 – Vue initiale de l'IDE après création du nouveau projet.

## Compilation et lancement de l'application

La création d'un nouveau projet via Android Studio génère une application par défaut, compilable et exécutable. Son exécution produit la création d'une seule activité et de son écran associé; cette application affiche simplement le texte Hello world!

### Compilation


Pour compiler uniquement votre application, plusieurs choix s'offrent à vous :

- utiliser le menu Build->Make Project;
- utiliser le raccourci Ctrl+F9;
- utiliser le bouton de compilation .

Choisissez l'option qui vous convient et compilez l'application. Vous devez voir apparaître dans la barre de notification de la fenêtre (en bas ...) le message Gradle build running durant la phase de compilation, qui peut prendre quelques secondes.

### Exécution

Pour lancer votre application, vous avez également plusieurs choix :

- utiliser le menu Run->Run app;
- utiliser le raccourci Maj+F10;
- utiliser le bouton de lancement .

Notez que si l'application n'a pas été compilée, une demande d'exécution lancera préalablement la phase de compilation ...

Choisissez l'option qui vous convient et lancez l'exécution de cette application. Si c'est la première fois que vous ouvrez Android Studio, vous voyez apparaître le message qui est présent dans la figure 5. Il signifie qu'aucun périphérique n'est connecté à votre machine, ce qui empêche l'exécution de l'application. Pour

pallier ce problème, vous avez le choix entre connecter un périphérique physique (téléphone, tablette) ou créer un périphérique virtuel, dont les étapes sont décrites ci-après. Par la suite, si Android Studio détecte plusieurs possibilités pour lancer l'application (différents types de périphériques virtuels existant et/ou de périphériques physiques connectés), il vous demandera de choisir celui à utiliser.

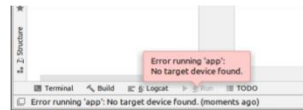



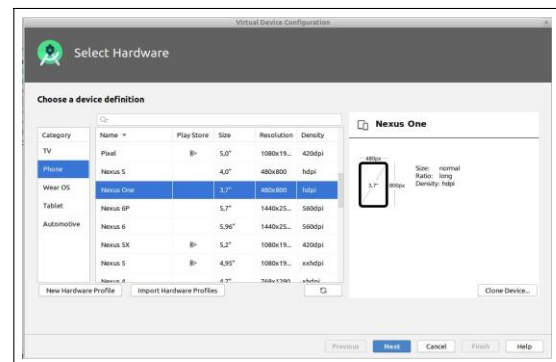
Figure 5 – Message d'erreur si aucun périphérique, virtuel ou physique, n'est connecté.

## Création d'un périphérique virtuel

Pour créer un périphérique virtuel, appuyez sur le bouton  du AVD Manager (*Android Virtual Device*). Si aucun périphérique virtuel n'existe, la fenêtre visible dans la figure 6a apparaît, vous permettant de lancer la création d'un nouvel AVD. Après avoir cliqué sur *Create Virtual Device...*, vous êtes amenés à choisir l'un des AVD disponibles dans le kit de développement (figure 6b); choisissez le **NEXUS One** puis validez.



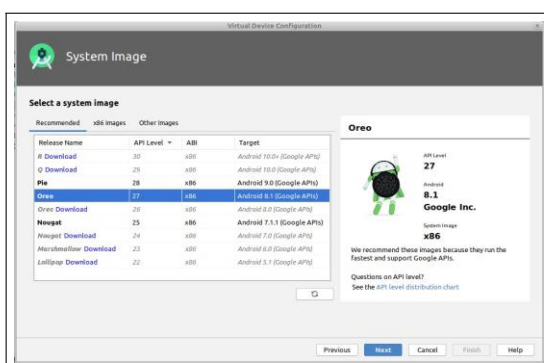
(a) Vue de l'activité principale.



(b) Vue de l'activité secondaire

Figure 6 – Vues des deux activités de l'application à développer

Vous devez ensuite choisir l'image système qui sera installée sur l'AVD. Sur la figure 7a, le choix a été fait sur la version 8.1 du système. Notez que la version du système choisi peut ne pas être présente sur votre machine (indication), auquel cas un téléchargement est requis, qui peut s'avérer long au vu de la taille de certaines versions. Il est conseillé, pour ce module, d'éviter de télécharger trop de versions différentes, qui n'apporteront pas grande chose à vos développements.



(a) Vue de l'activité principale.



(b) Vue de l'activité secondaire

Figure 7 – Vues des deux activités de l'application à développer

Une dernière étape de configuration apparaît enfin après validation de l'image système (figure 7 b), concernant un paramétrage plus fine de l'AVD. A ce stade, validez sans chercher à aller plus loin. Vous pouvez enfin relancer votre application, qui par défaut sera lancée sur le seul périphérique disponible existant. En cas de présence de plusieurs choix, vous serez amenés à sélectionner le périphérique que vous souhaitez utiliser.

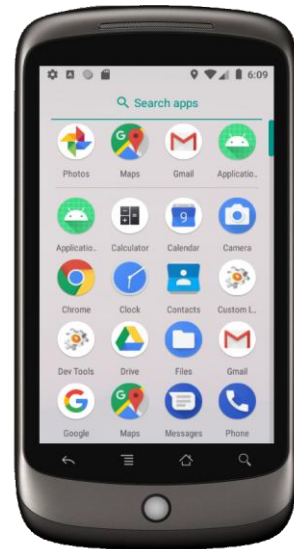
La figure 8a illustre le lancement d'un périphérique virtuel et l'exécution de votre application sur celui-ci (8b). Vous pouvez ensuite interagir avec l'AVD comme vous le feriez avec un périphérique réel, en utilisant la souris. Notez que votre application est bien présente dans la liste des applications disponibles sur le périphérique (figure 8 c ).



(a) Vue de l'émulateur après Initialisation d'android



(b) Vue de l'application.



(c) L'application dans la liste des applications

Figure 8 – Différentes vues de l'émulateur (ici celui du NEXUS One).

Vous pouvez suivre certaines informations liées à l'exécution de votre application dans une console d'exécution, qui peut être ouverte dans la partie inférieure de l'IDE (figure 9).

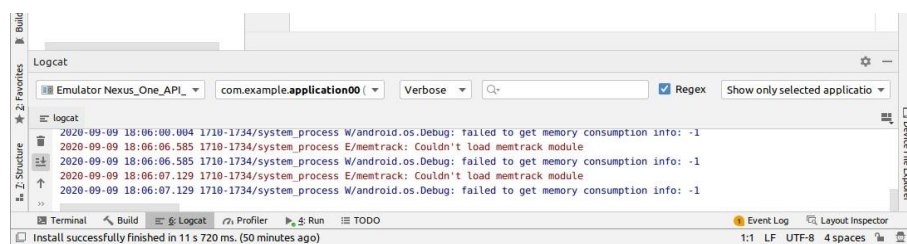


Figure 9 – Vue de la partie inférieure de l'IDE, avec la console de l'émulateur.

**Remarque importante :** Gardez l'émulateur ouvert (ou réduit) après l'avoir lancé, de manière à ne pas avoir de délais d'attente longs dès que vous voulez tester une nouvelle modification de votre application.

## Exercice 1

Modifiez le nom de la classe de l'activité principale en la renommant sous l'intitulé `ActivitePrincipale`. Notez que le nom de la classe et le nom du fichier Java doivent être modifiés en conséquence. Vous pouvez effectuer ces modifications soit dans le fichier Java, soit en changeant le nom du fichier Java et en utilisant les fonctions de refactoring disponibles dans l'IDE :

- petite icône ampoule qui apparaît dans le cas d'une modification directe du nom de la classe;
- clic sur le bouton droit de la souris sur le nom du fichier Java présent dans l'arborescence en cas de modification du nom du fichier, suivi de `Refactor->Rename`.

Notez qu'il est également nécessaire de modifier le fichier `AndroidManifest.xml` pour tenir compte du changement de nom si vous utilisez la première option ...

Testez les deux possibilités en vérifiant à chaque fois que toutes les modifications sont correctes, puis compilez et testez votre application.

## Exercice 2

Il s'agit à présent de changer le message qui apparaît lors de l'exécution de l'application. Dans le fichier `activity_main.xml`, le message `Hello world!` apparaît dans la balise `TextView`. Vous pouvez dès lors modifier son contenu, en fonction de vos souhaits. Compilez et testez ...

La manière de gérer et modifier le contenu de ce texte va cependant cacher les possibilités offertes par Android de gérer plusieurs langues, en centralisant par exemple les chaînes constantes dans le fichier **strings.xml** (dossier `values` du dossier `res`) et ses dérivées. Il est donc plus judicieux de déclarer cette chaîne de texte dans le fichier `strings.xml` et d'établir un lien entre la variable ainsi définie dans le fichier `main_activity.xml`.

1. créez une variable `app message` dans le fichier `strings.xml`, en l'initialisant avec le texte « Coucou le Monde ! »;
2. modifiez le `TextView` présent dans le fichier `main_activity.xml`, pour établir un lien vers cette variable. Compilez et testez.

## Exercice 3

Vous allez à présent internationaliser votre application, en proposant 2 versions du texte qui y apparaît, l'une en anglais, l'autre en français. On rappelle que les textes, libellés, etc. sont définis dans le fichier `strings.xml` qui est utilisé par défaut. Ce fichier se trouve dans le dossier `values` de votre application. Lorsque plusieurs versions linguistiques sont présentes, ce fichier doit être dupliqué et placé dans un dossier portant le nom `values-xx`, le `xx` étant remplacé par un identifiant représentant la langue (`fr` pour le français, `en` pour l'anglais, etc.).

Plutôt que de modifier manuellement l'arborescence de l'application, Android Studio propose un éditeur permettant, d'une part de définir la traduction des données figurant dans `strings.xml` et d'autre part, de générer l'arborescence nécessaire en fonction du ou des langue(s) choisie(s).

**Application :** cliquez avec le bouton droit sur le fichier `strings.xml` et sélectionnez `Open Translations Editor`. Vous voyez apparaître les différentes entrées figurant dans le fichier `strings.xml`. En cliquant sur l'icône représentant la Terre (`Add Locale`), vous pouvez ajouter une nouvelle langue et saisir la traduction des entrées à votre guise. Lorsque cela est fait, vous devez voir apparaître dans la fenêtre de l'arborescence un nouveau fichier `strings.xml`, accompagné du drapeau de la langue choisie et de l'extension de cette langue entre parenthèses. Il ne vous reste plus qu'à compiler et tester l'application sur votre émulateur, en testant différents choix de langues sur celui-ci (sur l'émulateur du Nexus One, démarrez l'**application** `Custom Locals` et choisissez la langue que vous souhaitez).

## Exercice 4

On souhaite modifier l'icône de lancement de l'application. Dans un premier temps, récupérez l'image disponible avec le sujet et rangez-la quelque part sur votre compte; elle représente une icône Android en 3D, avec une résolution importante.

Sous Android Studio, cliquez sur le bouton droit sur le dossier `app` et sélectionnez l'option `New->ImageAsset`. Cela a pour effet d'ouvrir un petit éditeur qui vous permet de sélectionner une image, d'y appliquer quelques effets simples et de générer des icônes avec différents niveaux de détails (figure 10). Lorsque vous êtes content de votre icône, il ne vous reste plus qu'à la sauvegarder, en la renommant `android3d` par exemple (pour éviter d'écraser l'icône précédente). Notez que l'image de départ doit avoir une résolution suffisante pour que l'éditeur d'icône puisse générer les différents niveaux de détails. En cas de résolution insuffisante, les niveaux hauts seront plus ou moins pixélisés ...

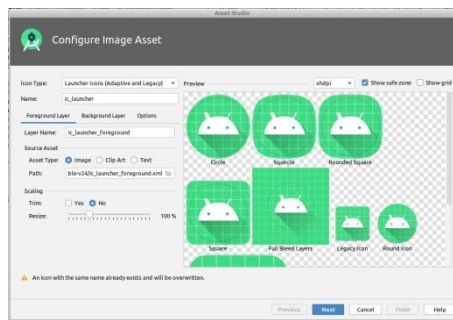


Figure 10 – Vue initiale de la fenêtre de l'éditeur d'icônes.

Il vous reste à modifier le fichier `AndroidManifest.xml` pour utiliser la nouvelle icône, compiler et enfin tester.

A noter que vous pouvez supprimer toutes les icônes non utilisées en cliquant droit sur le dossier qui les contient et en sélectionnant l'option delete. Effectuer cette opération sur `ic_launcher.png` et notez au passage que l'IDE vérifie, avant suppression, que ces icônes ne sont plus utilisées dans votre application.