Université Assane SECK de Ziguinchor

Unité de Formation et de Recherche des Sciences et Technologies

Département d'Informatique

Développement d'applications dans Android Studio

L3 – 2I

Mars 2023

Papa-alioune.cisse@univ-zig.sn

Résumé :

1 - Composants d'une application Android

Sous Android, une application est un ensemble de composants qui peuvent être de 4 types :

- Les activités (Activity) sont les éléments constitutifs principaux d'une application. Elles sont normalement associées à une vue graphique faite pour être affichée en plein écran. Pour simplifier, nous pourrons considérer qu'une activité se réduit à la notion de fenêtre. C'est le composant principal que nous manipulerons dans le cadre de ce cours.
- Les services (Service) sont des tâches de fond n'étant pas associées à une vue utilisateur.
- Les récepteurs de Broadcast (Broadcast Receiver) permettent à l'application d'être notifiée de certains événements (ex : batterie faible, l'écran a été tourné, ...) mais ne sont pas associés à une vue graphique comme le serait une Activité.
- ~~Les fournisseurs de contenu (Content Provider) sont des composants permettant de gérer l'accès à des informations de manière centralisée, comme le fait le gestionnaire de contacts par exemple~~.

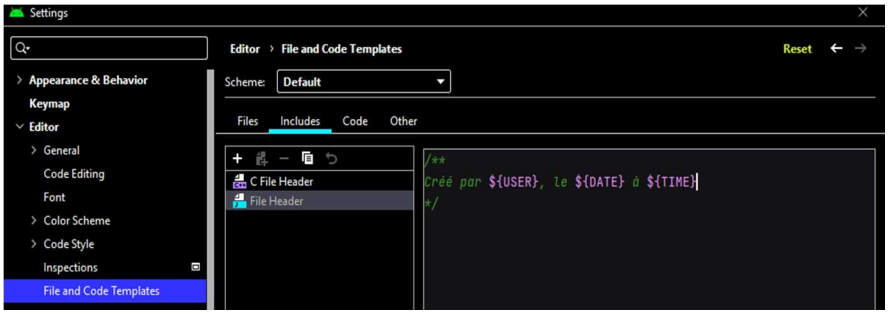
2 - Créer un projet dans Android Studio

Un projet dans Android Studio contient tout ce qui définit votre espace de travail pour une application, du code source et des actifs, au test du code et à la création de configurations. Lorsque vous démarrez un nouveau projet, Android Studio crée la structure nécessaire pour tous vos fichiers et les rend visibles dans la fenêtre **Projet** sur le côté gauche de l'IDE. S'il n'y est visible, cliquez sur **View > Tool Windows > Project** pour le faire apparaitre. Cette page fournit un aperçu des composants clés de votre projet.

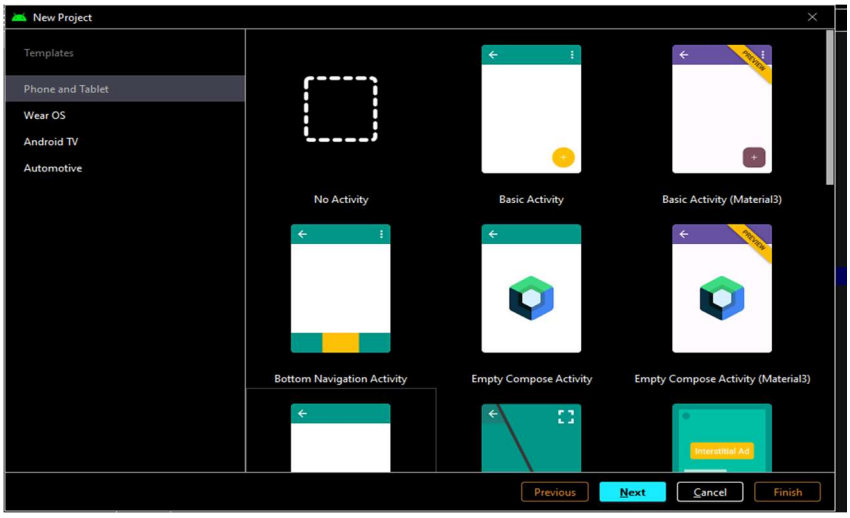
Nous démarrons cette partie avec quelques configurations de l'environnement de développement :

- **Le thème.** Pour configurer le thème principal, rendez-vous dans le menu ***File*** et cliquer sur ***Settings***. Sur la fenêtre qui s'ouvre, cliquez sur le menu ***Apparence & Behaviour***, puis sur ***Apparence***. A droite de la fenêtre, choisissez le thème que vous voulez avec le champ ***Theme***. Cliquez sur ***ok*** pour appliquer les changements.
- **Style du code.** Si vous le souhaitez, Vous pouvez modifier l'ensemble des règles d'indentation, de placement, de style ou de génération de code. Après ***File*** et ***Settings***, allez cette fois-ci dans le menu ***Editor***, puis ***Code Style*** et choisissez votre langage (***Java*** par exemple). A vous de jouer pour le reste.

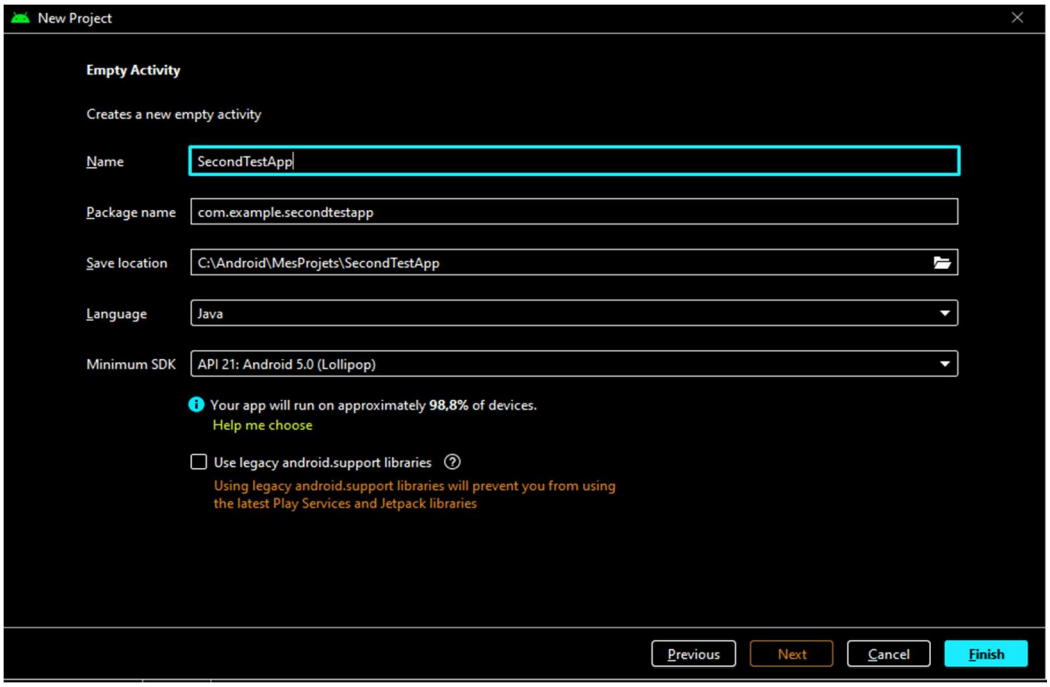
- **En-têtes des fichiers.** Lorsque vous créez un nouveau fichier dans votre projet, un en-tête est automatiquement ajouté avec des paramètres tels que le créateur du fichier, la date de création, etc. Si vous souhaitez créer vos propres en-têtes, allez cette fois-ci dans ***Editor***, puis ***File and Code Templates***. A droite de la fenêtre, choisissez l'onglet ***Includes***, puis ***File Header*** et renseigner votre en-tête comme dans la figure suivante : (${USER} donne l'utilisateur courant, ${DATE} donne la date courante et ${TIME} l'heure courante).



Pour créer un projet dans AS, allez dans **File -> New -> New Project**. La fenêtre qui apparait permet de choisir le type d'appareils cibles de votre application (Phone and Tablette, TV, Auto, etc.), et un type d'activité à ajouter à la création du projet. Même si vous créez votre projet sans activité (***Empty Activity***), une activité dénommé ***MainActivity*** sera ajoutée dans tous les cas : il s'agit de l'activité principale, correspondant par exemple à la fonction main d'un programme C/Java.



Au clic sur Suivant, la fenêtre qui s'affiche permet de configurer le projet en précisant :



- **Le « *Name* » de l'application**. Ce sera le nom qui apparaîtra en dessous de l'icône de l'application sur l'écran d'accueil du téléphone, et dans la barre de titre de l'application. Il peut contenir des caractères non ASCII.

- **Le « *Package name* »**. Il permet de déterminer quel nom de paquetage utiliser pour votre application et, par la suite, de distinguer votre application d'une autre application qui porterait le même nom. Par convention, la notation inverse est utilisée. Par exemple, si vous travaillez dans la société *UASZ*, vous pourriez préciser *com.android.uasz.nom_projet*.

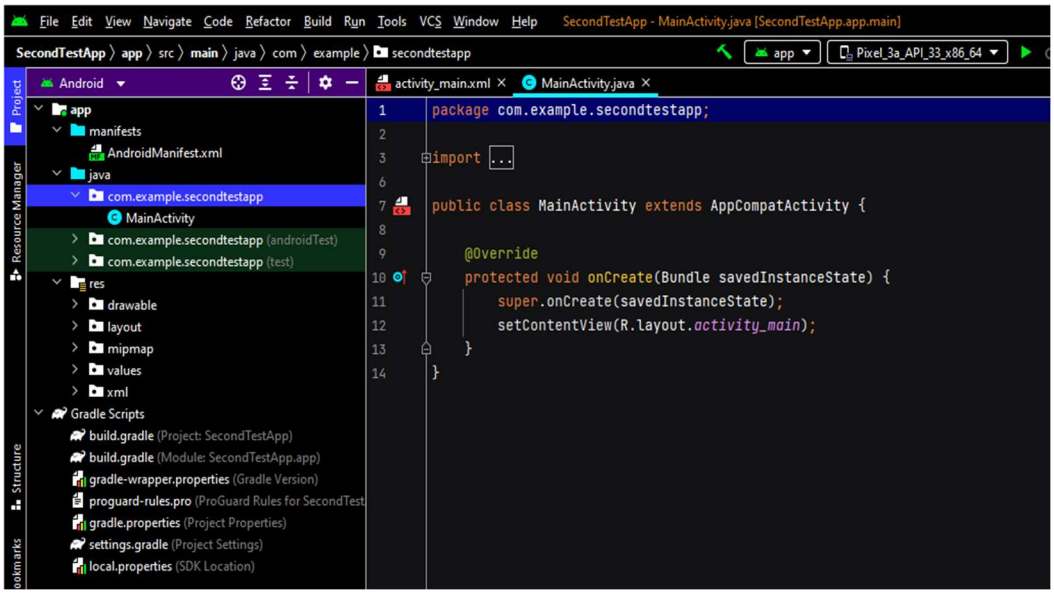
- **Le « *Save location* »**. C'est le répertoire de sauvegarde qui contiendra tout le contenu de votre projet.

- **Le « *Langage* »**. Actuellement, vous avez deux langages dans AS pour développer des applications Android : Java et Kotlin.

- **Le « *Minimum API Level* »** est la version minimale de SDK à utiliser pour le projet. Utiliser une API élevée (donc récente) vous permet de bénéficier des dernières fonctionnalités proposées par Android. Toutefois, les anciens appareils présents sur le marché qui ne sont pas à jour ne pourront pas faire fonctionner votre application.

3 - Arborescence d'un projet

Une fois le projet créé, l'arborescence des fichiers du projet ressemble à celui-ci :



- **Le répertoire manifests.** Ce répertoire contient généralement un seul fichier : le fichier *AndroidManifest.xml*. Ce fichier est la carte d'identité de votre application. Il permet entre autres de préciser le nom de l'application, l'icône à utiliser, quelle activité lancer au démarrage, etc. Il contient également les ***permissions*** de votre application. En effet, Lorsque l'on développe une application Android, on peut vouloir exploiter des fonctionnalités du système (caméra, accès à internet, etc.) ou des fonctionnalités qui concernent la vie privée de l'utilisateur (Ex : accès à ses contacts). Dans ces cas-là, on devra annoncer, dans le ***Manifest*** des ***permissions***. De cette façon, votre application demandera explicitement l'autorisation à l'utilisateur avant d'accéder à ces fonctionnalités.

- **Le répertoire java.** Il contient 3 sous dossiers. Seul le premier nous intéresse dans le cadre de ce cours. Il contient l'ensemble des classes Java de l'application, ce qui se limite au fichier ***MainActivity.java*** dans cet exemple. Les 2 autres dossiers sont des outils mis à disposition pour développer une stratégie de tests de l'application.

- **Le répertoire res.** Il contient l'ensemble des ressources du programme :
  - Toutes les images ou assimilés sont dans le dossier ***drawable***. Il existe toutefois une exception : les icônes. Celles-ci sont des images que l'on fournira volontiers en plusieurs résolutions afin de permettre une bonne adaptation aux différentes densités d'écran. Elles sont alors placées dans le dossier ***mipmap***.
  - Le dossier ***layout*** contient l'ensemble des éléments structurant les interfaces graphiques de l'application (ce sont des fichiers XML). Dans cet exemple, cela ce réduit à ***activity_main.xml***

- Le dossier ***values*** contient divers fichiers XML complémentaires permettant de configurer l'application. En particulier, on notera le fichier ***strings.xml***, dont l'usage permet de réaliser « aisément » une application multi-lingue ; le fichier ***styles.xml*** pour définir les styles des interfaces graphiques ; le fichier ***colors.xml*** pour définir les couleurs à utiliser ; etc.

- **Gradle Scripts.** Gradle est un outil qui permet d'effectuer, entre autres, toutes les tâches complexes nécessaires à la compilation. C'est aussi le gestionnaire de dépendances de l'application.

4 - Focus sur le code de « MainActivity »

Dans le manifest, il est déjà établi que ***MainActivity*** est l'activité principale. C'est-à-dire qu'une instance de cette activité est lancée au démarrage de l'application.

Le code de cette activité générée se réduit à quelques lignes :

```
package com.example.secondtestapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

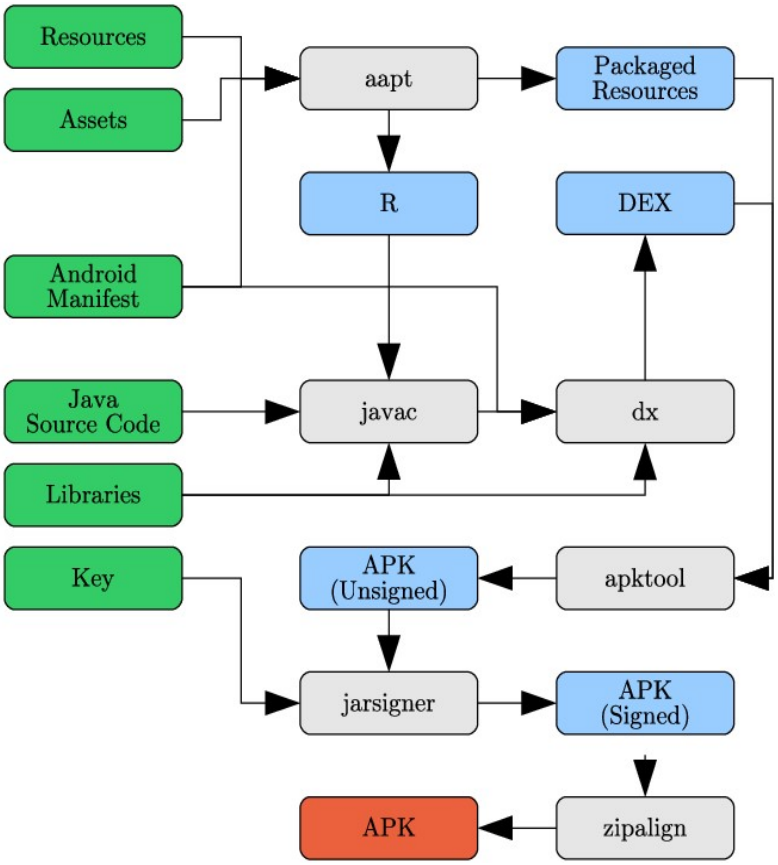
Nous reviendront de façon détaillée sur les activités dans le chapitre suivant. Pour le moment, il faut comprendre juste qu'une activité est une classe qui :

- hérite d'une classe appelée ***AppCompactActivity*** ;
- est sans constructeur, car une activité est lancée via une intention (***Intent***) sur laquelle on reviendra en détails. Cependant, noter que pour lancer une activité et passer d'une activité à une autre, on utilise des objets appelés ***Intent***.
    - Les ***Intent*** dits explicites dirigent vers une activité (c'est-à-dire un écran) spécifique (exemple : affichage d'une photo vers un écran lecteur d'images).

- Les ***Intent*** dits génériques ouvrent un lien de redirection disponible pour toute activité capable d'« *attraper* » ce lien (exemple : partage d'une photo vers différents réseaux sociaux disponibles).

- a un cycle de vie dont la première méthode appelée est ***onCreate*** (elle joue un peu le rôle du constructeur d'une activité). La méthode ***onCreate*** attend en paramètre un Bundle. Il est utilisé lorsque l'activité a été interrompue et que sa mémoire a été récupérée. Il contient alors les informations qui étaient précédemment présentes dans les différents éléments de champs de saisie et qui ont été sauvegardées. Il est nécessaire de retransmettre ce paramètre à la méthode ***onCreate*** telle que définie dans la classe parent. La seconde instruction présente dans la méthode ***onCreate*** permet de charger un ***Layout*** pour construire le visuel de l'activité. Ici c'est ***activity_main*** qui est chargé. Le gabarit ***activity_main*** qui a été généré est un fichier XML constitué de deux objets (vous pouvez l'ouvrir pour voir) : un conteneur principal de type **Constraint Layout** et un **TextView** qui affiche le texte « Hello World ! ».

5 - Processus de compilation et d'exécution d'une application Android



Une application Android comprend généralement des fichiers de code source (répertoire **java**), des fichiers de ressources (répertoire **res**, le manifeste et d'autres ressources comme les images et les Drawables), des librairies externes (dépendances).

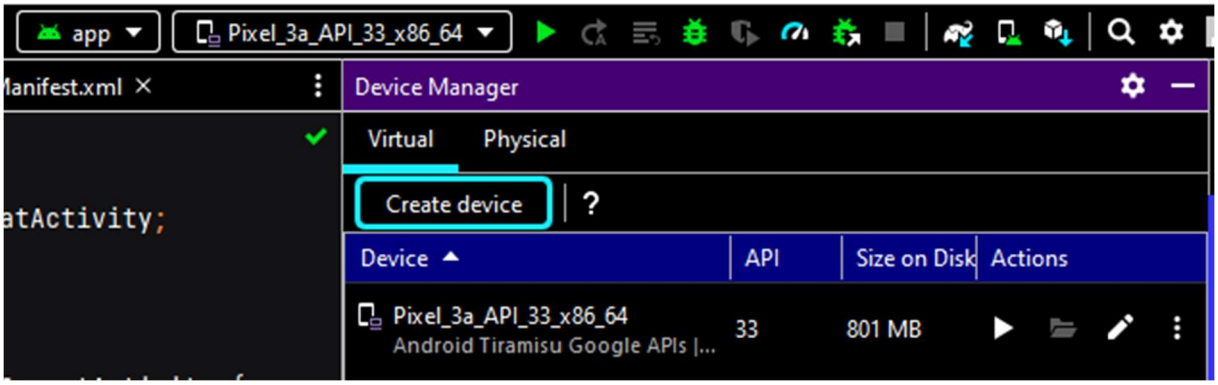
1. Le processus de ***build*** commence par la compilation des fichiers de ressources par ***Android Asset Packaging Tool (aapt)***. Un fichier appelé **R.java** (on l'appelle la classe R) est également créé à cette étape, ce qui permet de référencer des ressources et d'y accéder à partir de fichiers sources Java. La classe R contient des classes internes dont les noms correspondent aux types de ressources (id, drawable, layout, etc.) et est constituée à partir des fichiers placés dans les sous répertoire du répertoire **res**. Une propriété est créée pour :
   - Chaque image placée dans le dossier drawable-xxxx
   - Chaque identificateur défini dans les fichiers xml (objets d'interface, constantes)
   - Chaque fichier placé dans les répertoires xml, raw, etc.
2. Un compilateur JAVA compile tous les fichiers Java de votre projet et génère les fichiers **.class** correspondants.
3. L'outil ***dx*** convertit tous les fichiers de classe (fichiers .class) de votre projet en **Bytecode** pour la machine virtuelle **Dalvik** (fichiers .dex = exécutables Dalvik).
4. Toutes les ressources compilées, les ressources non compilées, les fichiers ***.dex*** sont packagés au format ***.apk*** (fichier de package d'application Android). Le résultat est un fichier dénommé ***Project-debug.apk***.
5. Vous devez signer votre application, ce qui aboutira à ***Project-release.apk***.
6. Une application Android est déployée via un **fichier .apk**. Ce fichier est l'équivalent du .jar en Java. Il contient l'application, ses dépendances et son *Manifest*. Cependant, la manière classique de diffuser une application n'est pas de partager un .apk directement, mais de le déposer sur le ***play store***.

6 - Lancer l'application sur un émulateur

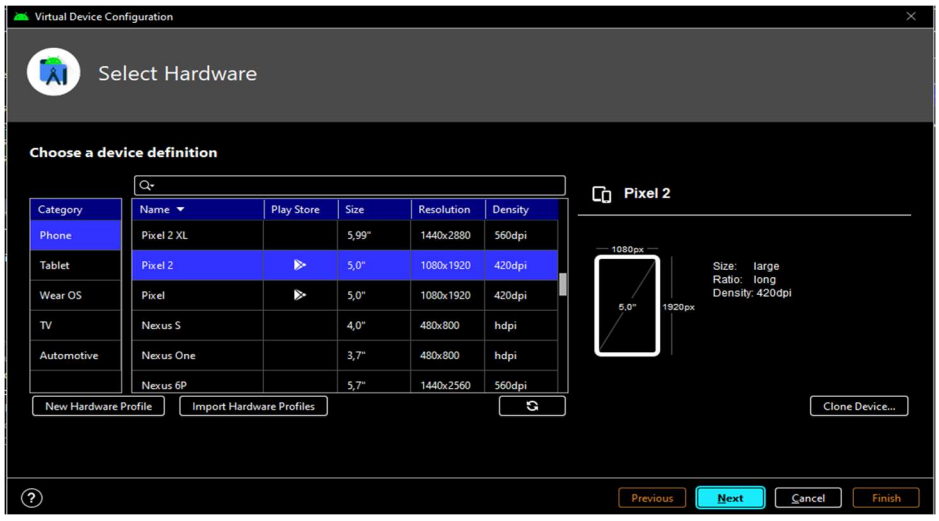
Un émulateur Android est un périphérique virtuel Android (AVD, Android Virtual Device) qui représente un périphérique Android spécifique. Vous pouvez utiliser un émulateur Android en tant que plate-forme cible pour exécuter et tester vos applications Android sur votre PC.

Pour lancer l'application sur un émulateur, il faut cliquer sur le bouton de l'encadré 2. Cependant, il faut d'abord s'assurer d'avoir déjà ajouté un émulateur. Si un émulateur est ajouté, vous le verrez dans l'encadré 1 de la figure ci-dessus. Sinon vous verrez à la place la mention « No Device ». S'il n y a pas d'émulateur, vous pouvez en ajouter un en cliquant sur l'encadré 3 de la figure ci-dessus.

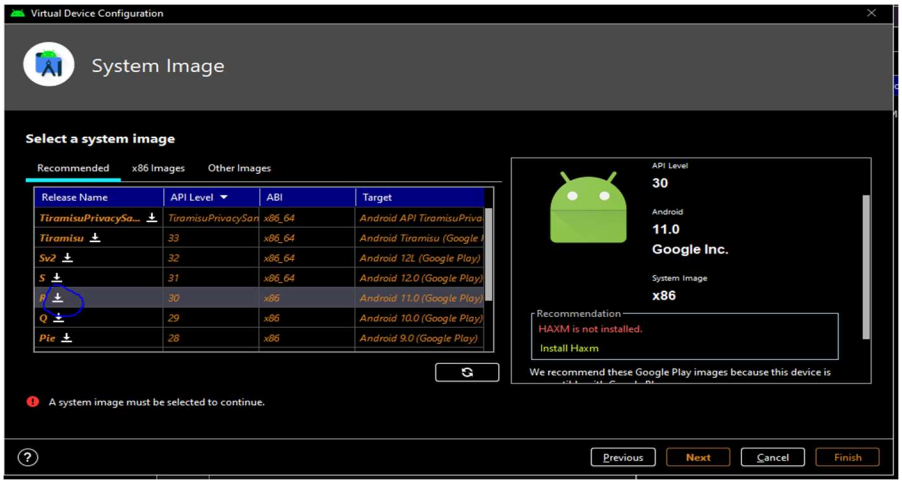
Pour mon cas, il y a déjà un émulateur ajouté comme vous pouvez le voir dans l'encadré 1 de la figure ci-dessus et sur la figure suivante, après clic sur l'encadré 3.



Pour ajouter un nouvel émulateur, cliquer sur le bouton « Create device ». Sur la fenêtre qui s'ouvre, vous pouvez choisir le type d'équipement, la résolution et cliquer ensuite sur « Next ».



Après clic sur « Next », la fenêtre ci-dessous s'ouvre, Il est possible que vous deviez installer une image système pour pouvoir aller plus loin : cliquez pour cela sur le lien (encadré en bleu) à côté de l'image que vous souhaitez installer.

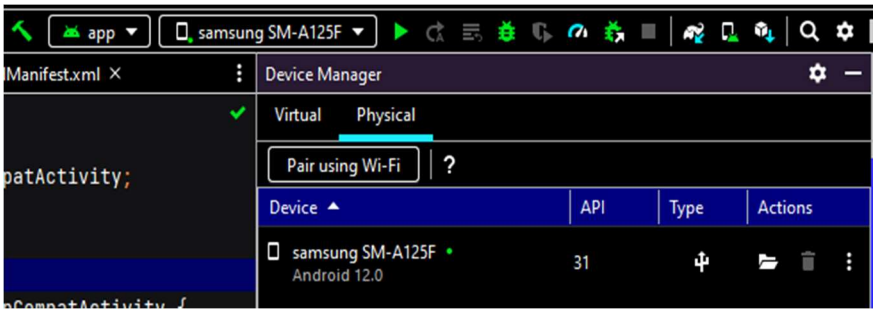


L'écran **Component Installer** s'affichera ensuite, le temps de télécharger les fichiers nécessaires (qui peuvent être volumineux).

7 - Lancer l'application sur un équipement réel

Afin de permettre à Android Studio de communiquer avec votre téléphone par exemple, il est nécessaire d'activer le mode « Options développeur » sur votre smartphone. Pour cela, selon votre smartphone, vous pouvez voir sur internet comment faire. Après l'avoir ajouté, allez dans **Paramètres**, sélectionnez **Options pour les développeurs** (il est tout en bas dans **Paramètres**), puis activez le **débogage USB**. Branchez ensuite votre smartphone sur un port USB de votre ordinateur.

Après cela, cliquez sur l'encadré 3 de la figure plus haut. Sur la fenêtre qui s'ouvre, allez dans l'onglet « Physical » pour voir que votre smartphone est ajouté dans les supports physiques comme sur la figure suivante (pendant ce temps, vérifiez votre smartphone et acceptez une éventuelle demande de communication avec votre ordinateur).



Il vous est possible maintenant de choisir votre téléphone comme « device » pour lancer l'application.