Android – Concepts fondamentaux du développement d'applications Android

1. Présentation d'Android

Android est un système d'exploitation qui était destiné, initialement, aux appareils photo numériques. Aujourd'hui, il est également utilisé comme OS pour les tablettes, les téléphones portables, les objets connectés…

Les évolutions du système sont rapides, la version 8 est apparue en août 2017. Pour aider les développeurs à faire leur choix concernant la version de la plateforme Android, Google fournit, tous les quinze jours, des statistiques sur la répartition du taux d'installation de chaque version, voir https://developer.android.com/about/dashboards/index.html. Ces données permettent d'aider le développeur à choisir, en toute connaissance de cause, la version minimale sur laquelle devra s'exécuter son application.

Remarque : *plus la version minimale d'Android requise par l'application est faible, plus le nombre d'appareils, et donc d'utilisateurs et potentiellement d'acheteurs, pouvant exécuter l'application sera important. Mais en contrepartie, l'application disposera de moins d'API du SDK que les versions plus récentes. C'est donc au développeur de trouver le meilleur compromis entre les fonctionnalités requises du SDK et l'étendue du public visé*.

Le tableau suivant présente l'implantation des versions jusqu'en février 2018.

Version de la plateforme Android	**Date**	**Nom**	**Niveau d'API**
1.0	09/2008	Apple Pie	1
1.1	02/2009	Banana Split	2
1.5	04/2009	Cupcake	3
1.6	09/2009	Donut	4
2.0	10/2009	Eclair	5
2.0.1			6
2.1			7
2.2	05/2010	Froyo	8
2.3	12/2010	Gingerbread	9
2.3.3			10
3.0	02/2011	Honeycomb	11
3.1			12
3.2			13
4.0.1	10/2011	Ice Cream Sandwich	14
4.0.3			15
4.1	07/2012	Jelly Bean	16
4.2.1			17
4.3			18
4.4	10/2013	KitKat	19
5.0	11/2014	Lollipop	21
5.1			22
6.0	10/2015	Marshmallow	23
7.0	08/2016	Nougat	24
7.1	12/2016		25
8.0	08/2017	Oreo	26
8.1			27

Android assure une compatibilité ascendante entre les versions du système. Néanmoins, cette profusion de versions différentes, ainsi que la multitude d'appareils ayant des caractéristiques différentes (taille d'écran, résolution d'écran, mémoire…), résumé par le terme « fragmentation », est le principal problème pour les développeurs Android.

2. Architecture d'Android

Android s'appuie sur un noyau Linux. Les dernières versions d'Android sont basées sur la version 4.4 de Linux. Ce noyau prend en charge la gestion des couches basses, telles que les processus, la gestion de la mémoire, les droits utilisateurs et la couche matérielle.

Au-dessus de ce noyau, figure la couche des principales bibliothèques principales du système. Celles-ci, de bas niveau, sont écrites en C et/ou en C++. Elles fournissent des services essentiels tels que la gestion de l'affichage 2D et 3D, un moteur de base d données SQLite, la lecture et l'enregistrement audio et vidéo, un moteur de navigateur web…

Les fonctionnalités offertes par ces bibliothèques sont ensuite reprises et utilisées par la couche supérieure sous forme de bibliothèques Java. Celles-ci fournissent des bibliothèques et composants réutilisables spécifiques à des domaines particuliers (ex : bibliothèques de BDD, de téléphonie, de localisation géographique…).

La couche de plus haut niveau est celle des applications. Ces applications sont celles fournies par défaut, comme l'application d'accueil (dénommée également bureau), l'application de lancement d'applications, le navigateur web, l'application de téléphone ou celles téléchargées sur le magasin d'applications officiel (Play Store anciennement nommé Android Market).

Par défaut, chaque application s'exécute dans une machine virtuelle Java elle-même confinée dans un processus Linux dédié. Cette machine virtuelle est spécifique à la plate-forme Android et spécialisée pour les environnements embarqués. Jusqu'à la version 4 d'Android, l'environnement d'exécution s'appuyait sur la machine virtuelle **Dalvik**. Depuis la version 5, Android s'appuie sur une nouvelle machine virtuelle nommée ART (Android Run Time). Elle possède des spécificités telles que :

- *AOT (Ahead of Time) Compilation* : permet de compiler une application à l'installation et non plus à l'exécution (Dalvik).
- Amélioration du Garbage Collector)
- Amélioration du développement et du débogage d'applications

Pour résumer, l'architecture d'Android se compose essentiellement de cinq parties distinctes :

- **Application** : représente l'ensemble des applications fournies avec Android
- **Framework Android** : représente le framework permettant aux développeurs de créer des applications en accédant à l'ensemble des API et fonctionnalités disponibles sur le téléphone (fournisseur de contenu, gestionnaire de ressources, gestionnaire de notification, gestionnaire d'activités..).
- **Bibliothèques** : Android dispose de bibliothèques utilisées dans les différentes composantes du système.
- **Machine virtuelle** nommée actuellement **ART**.
- **Linux Kernel** : le noyau Linux fournit l'interface avec le matériel et gère les ressources et les processus Android.

La compilation du source Java d'une application Android produit du bytecode. Ce code est embarqué dans un fichier APK (Android PacKage).
Une application peut être déployée sur un terminal via Play Store. Cependant, lors de la phase de développement, les applications sont transférées sur un terminal au moyen d'un câble USB. Il faut alors préparer le terminal pour le développement en activant l'option Débogage USB dans l'option Applications/Développement de l'application Paramètres.
Les applications Android s'exécutent sur un système contraint en termes de ressources matérielles : mémoire disponible, consommation de batterie, différence d'affichage, stockage disponible…

3. Environnement de développement

3.1. Environnement Java

Le premier prérequis au développement Android est l'installation d'un environnement Java, particulièrement le JDK (Java Development Kit). Pour installer l'environnement Java, il faut télécharger et installer la dernière version de JDK sur le site http://www.oracle.com/technetwork/java/javase/downloads.

Remarque : la version actuelle (février 2018) est le JDK 9.0.4.

3.2. Android Studio

Android Studio est l'environnement de développement spécifique à Android. *Nous allons travailler avec cet IDE sous Windows.*
Pour télécharger la dernière version d'Android Studio, rendez-vous à l'adresse suivante : https://developer.android.com/sdk/installing/studio.html

Pendant l'installation, noter l'emplacement où sera installé le kit de développement d'Android (SDK : Software Development Kit) car ce sera demandé lors de l'installation d'Android Studio.

Remarque : ce support de cours est basé sur Android Studio v3.0.1. C'est la dernière version actuelle (fév. 2018).

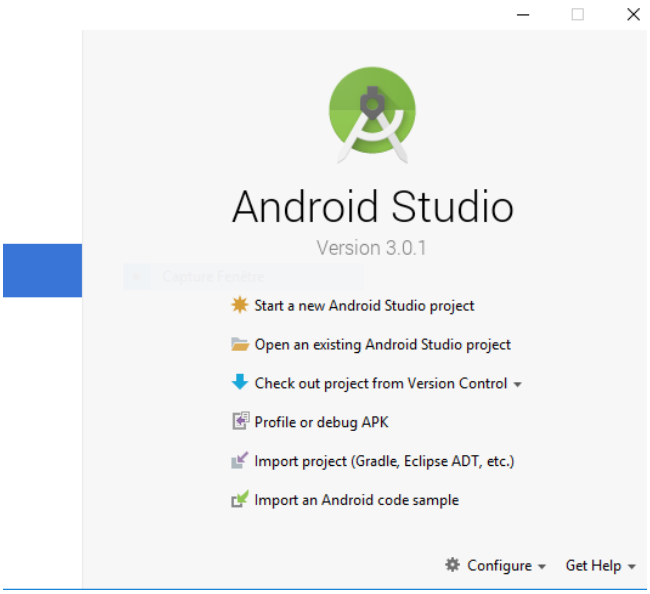
Une fois le téléchargement terminé, installer l'IDE en suivant l'assistant d'installation. Si le répertoire d'installation du JDK n'est pas trouvé par l'assistant, une boîte de dialogue vous demande son emplacement et vous fournit le lien pour installer le kit de développement.
Il vous sera demandé de préciser les éléments à installer. Vérifier que tous les éléments listés sont cochés, à savoir : **Android Studio, Android SDK, Android Virtual Device.**

Une fois l'installation terminée, vous pouvez exécuter l'application Android Studio et obtenir le résultat suivant :

Les options disponibles sur cet écran d'accueil permettent de :

- Créer un nouveau projet
- Ouvrir un projet existant
- Récupérer un projet depuis un gestionnaire de version (GitHub, CVS, Git, Google Cloud, Mercurial, subversion)
- Profile ou debugger un APK
- Importer un projet
- Importer un simple code Android.

La partie gauche de cette fenêtre comportera la liste des projets Android Studio. La première fois, elle est vide.



3.3. SDK Android

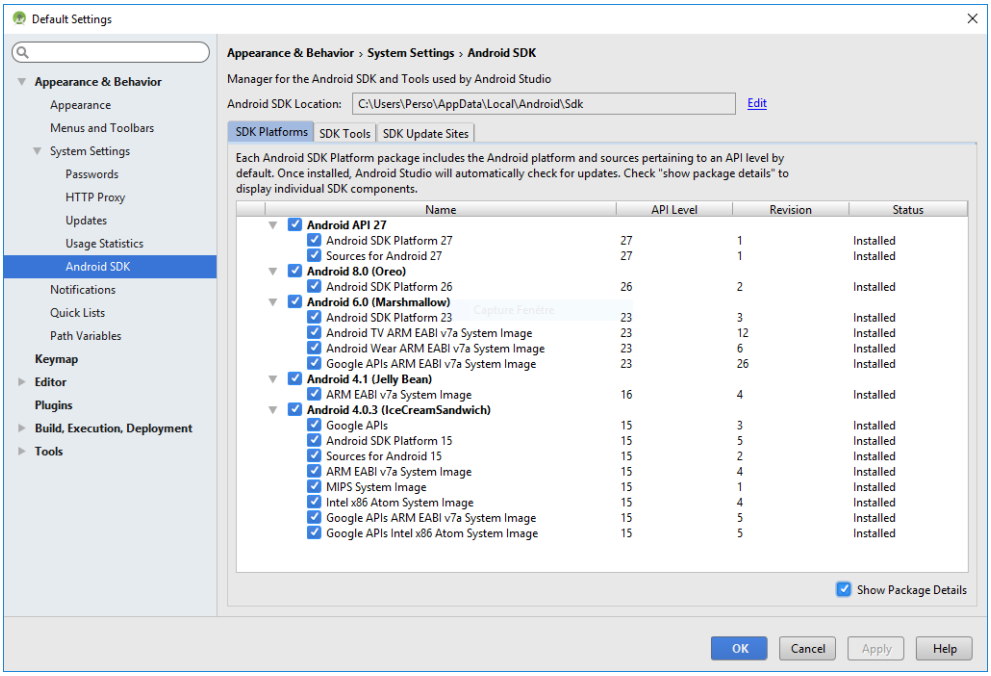
Android SDK contient tous les outils nécessaires pour créer une application Android. Il permettra de mettre à jour le SDK, d'installer de nouvelles versions d'Android ou de mettre à jour des versions déjà installées.
Les sections :

- **SDK Platforms** : L'écran de l'application affiche une liste des packages téléchargeables et indique, pour chaque élément, s'il est déjà installé, partiellement installée ou non installé et s'il doit être mis à jour. Par défaut, seule la dernière version d'Android disponible est installée.
  L'option **Show Package Details**, située en bas à droite de la liste, si elle est cochée, affiche le détail des éléments installés ou qui peuvent l'être.
- **SDK Tools** : présente la liste des outils fournis par Google pour assister le développeur. On y retrouve, entre autres, les éléments suivants :
  - *Android Emulator* : l'émulateur Android qui permet de simuler le fonctionnement d'un appareil Android sur le poste de développement.
  - *Android Support Library* : ensemble de bibliothèques de support fournies par Google pour gérer les problèmes de fragmentation.
  - *Google Play services* : services proposés par Google (Google Maps, géo-localisation…) qui peuvent être utilisés par le développeur.

La configuration du SDK est réalisée depuis l'écran d'accueil d'Android Studio à l'aide du raccourci Configure.

A l'ouverture, le programme se connecte automatiquement aux serveurs de Google pour télécharger les dernières informations sur les packages disponibles.

L'écran suivant présente cette liste de packages et indique, pour chaque élément, son statut (déjà installé ou pas et s'il doit être mis à jour).



Une fois les packages sélectionnés, le téléchargement commence. *Soyez patients, cela peut prendre un certain temps.*

4. Concepts de base d'Android

4.1. Les activités (activity)

Une activité est la composante principale d'une application Android. Elle représente l'implémentation et les interactions des différentes interfaces de l'application (formulaire de saisie, vidéo…). Chaque écran présenté à l'utilisateur est une sous-classe de la classe Activity du framework Android.
Deux règles importantes concernent les activités :

- Une seule activité est en cours d'exécution à un instant t,
- Une application, pour être distribuée, doit contenir obligatoirement au moins une activité.

Chaque activité doit être inscrite dans le fichier AndroidManifest.xml, sinon une erreur sera déclenchée lors du lancement de l'application.

4.2. Android Package

Pour installer et distribuer une application, il faut créer un APK (Android PacKage). L'APK est le binaire représentant une application, il doit avoir un nom unique et doit être signé par son propriétaire.
Chaque package contient :

- Le code source de l'application compilé en bytecode (.dex)
- Les ressources (images, vidéo…) embarquées
- Les assets
- Les certificats
- Le fichier manifeste.

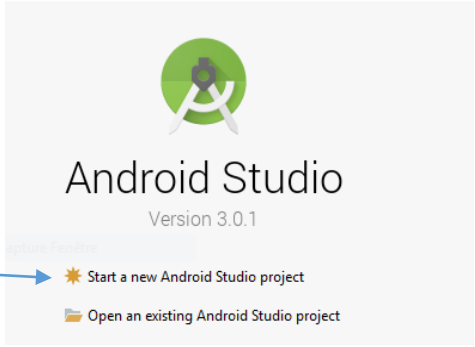
Android Studio fournit un outil qui s'appelle `aapt` (`Android Asset Packaging Tool`). Ce dernier permet de créer, visualiser et modifier des archives Android-APK. On peut générer deux types d'APK :

- Un APK signé.
- Un APK non signé. Ce format peut être utilisé pour tester l'application mais ne peut être publié sur Google Play.

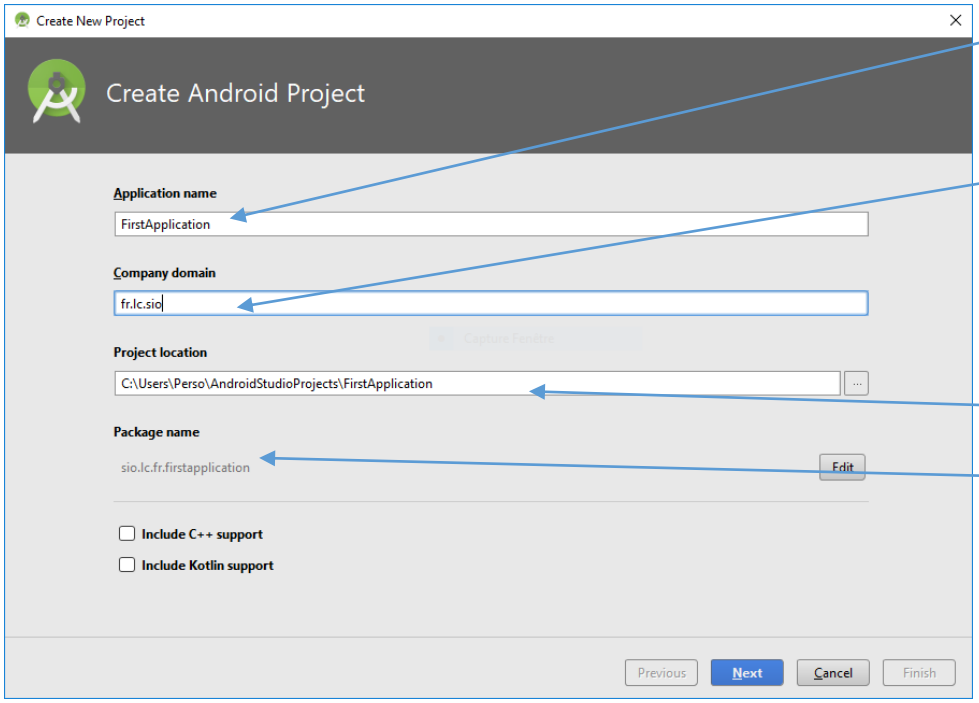
Durant la phase de développement, les applications sont transférées sur le terminal physique via un câble USB. Il faut alors préparer le terminal pour le développement en activant l'option Débogage USB dans l'option `Application/Développement` de l'application `Paramètres`.

5. Création d'une première application

Pour créer un nouveau projet sous Android Studio, lancer Android Studio et cliquer sur l'option : Start a new Android Studio Project.



Un premier écran s'affiche et permet de saisir des informations sur le projet :



Application name : correspond au nom qui sera affiché pour l'application lorsqu'elle sera installée sur un terminal Android.

Company domain : nom de domaine de la société. Cette information sert à construire le nom du package de l'application. Ce nom sera repris à chaque exécution de l'assistant de création de projet.

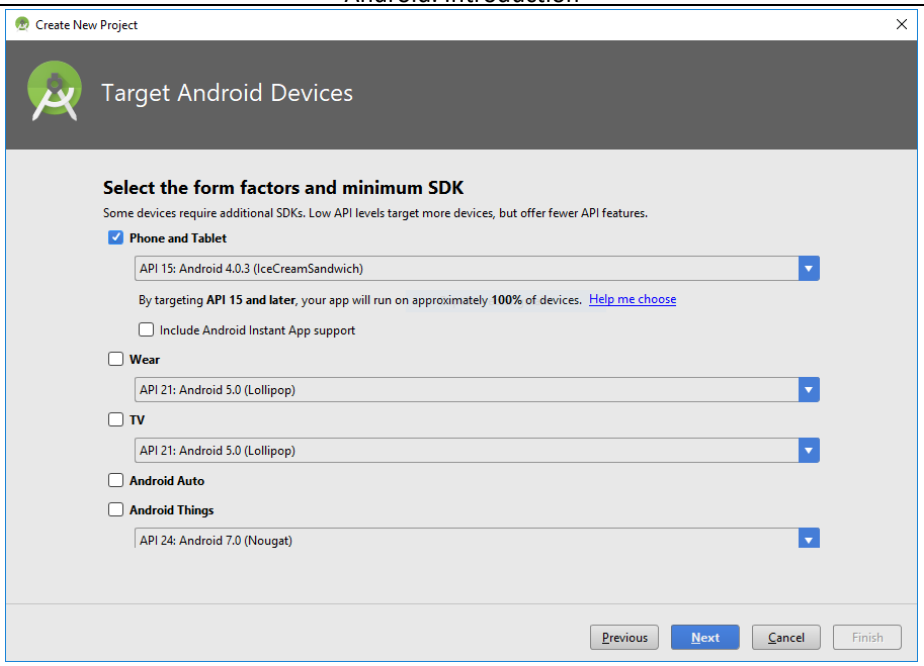
Project location : répertoire où sera stocké le projet.

Package name : généré automatiquement à partir du nom et du domaine inversé. Ce nom doit être unique. Par défaut, il est en lecture seule mais il est possible de le modifier en cliquant sur le bouton : `Edit`.

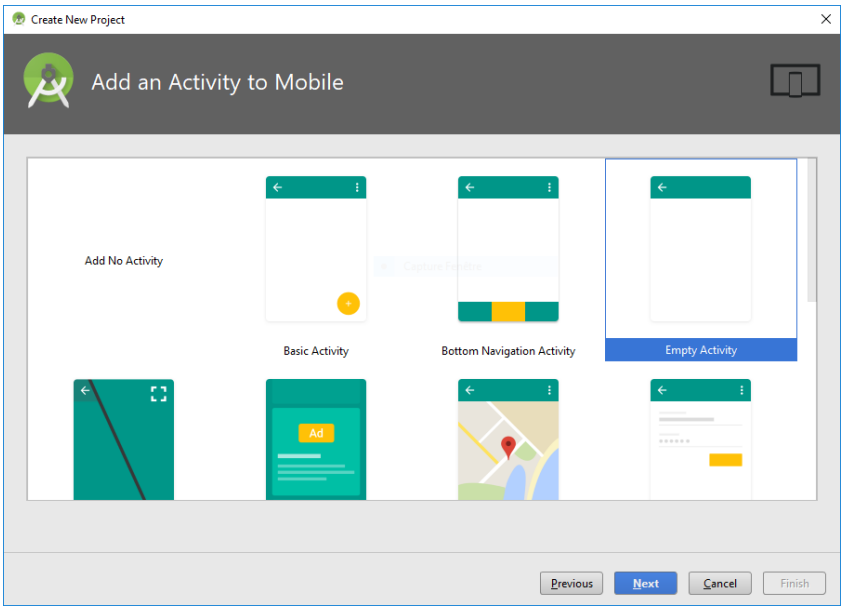
Cliquer sur le bouton `Next`, une fois toutes les informations renseignées.

Nous obtenons l'écran suivant qui permet de choisir les types de terminaux sur lesquels l'application fonctionnera. On y trouve : smartphones et tablettes (un seul type d'appareil), télévisions Android, objets connectés (Wear), Android Auto, autres domaines où la plateforme Android est présente.
Pour chaque type d'appareil, il faut choisir le SDK minimal requis, cela correspond à la version d'Android minimale dont l'appareil doit être équipé pour être autorisé à installer l'application.

Nous allons choisir uniquement Phone et Tablet. On choisira le SDK qui présente la plus grande couverture d'appareils cibles (API15 : Android 4.0.3).



L'écran suivant affiche les informations permettant de choisir le type de l'activité à créer. Les choix possibles sont :

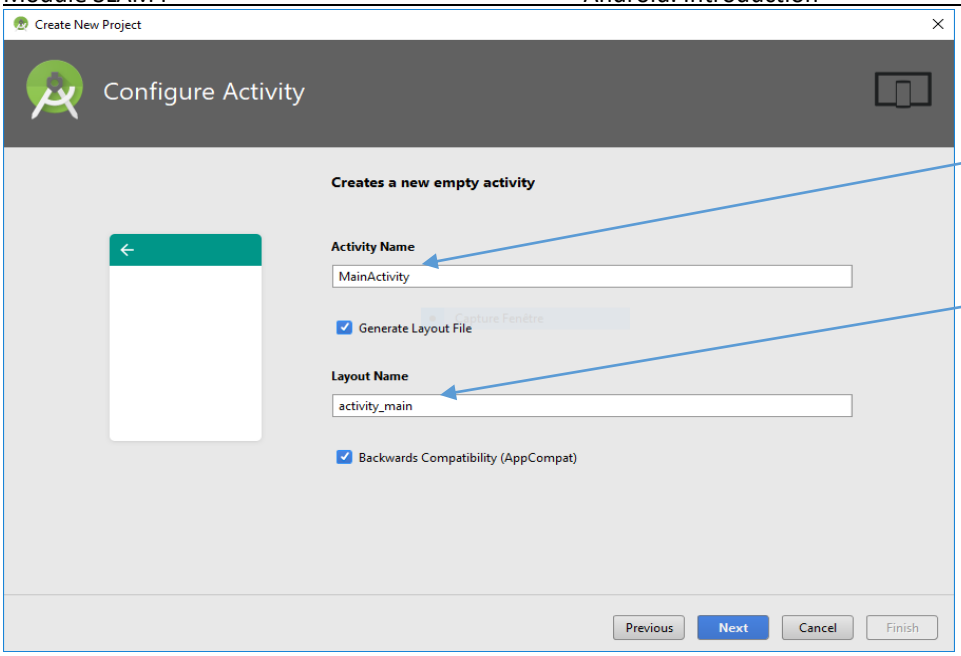


Add No Activity : ne pas ajouter d'activité par défaut
Basic Activity : activité à ajouter dans sa forme la plus simple
Bottom Navigation Activity : l'activité comportera une barre de navigation (en bas de l'écran), ce qui inclut toute la logique de traitement de la barre de navigation.
Empty Activity : une activité sera ajoutée dans une forme basique.
Fullscreen Activity : une activité intégrant le code permettant de passer l'application en mode « plein écran » (sans barre de notification) sera ajoutée.
Google AdMob Ads Activity : une activité, incluant le code nécessaire à la gestion des publicités du réseau AdMob (régie publicitaire de Google) sera ajoutée.

Google Maps Activity : les éléments nécessaires à l'intégration de Google Maps seront ajoutés.
Login Activity : une activité intégrant un mécanisme de connexion sera ajoutée.

Choisir Empty ou Basic Application et cliquer sur Next.

L'assistant demande de saisir quelques informations pour l'activité qui sera créée :



génère le fichier **MainActivity.java**

génère le fichier de layout (l'interface en XML)

génère **activity_main.xml** correspondant à la description de l'interface.

Cliquer sur le bouton Next. Une fois, les composants nécessaires installés, cliquer sur le bouton `Finish`. Le projet va être alors construit. Cela prend un certain temps et nécessite une connexion réseau.

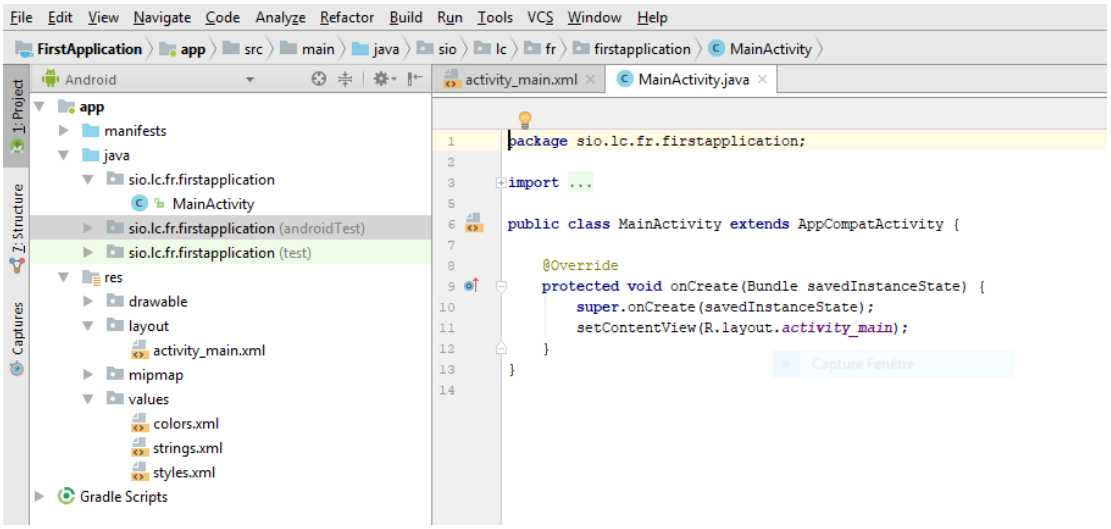
6. Structure du projet

La structure d'un projet Android est la suivante :

manifests : contient le fichier AndroidManifest.xml

java : fichiers sources

res : ressources de l'application. Ce répertoire contient l'ensemble des éléments dont a besoin l'application pour fonctionner. Les images sont classées dans `res/drawable`.



Les animations dans le répertoire `res/anim`.
Les fichiers de description de l'interface dans le répertoire `res/layout`.

6.1. Manifeste

Le fichier `AndroidManifest.xml`, situé à la racine de tout projet Android, est une description complète de l'application. Ce fichier joue le rôle d'interface entre le système Android et l'application. Tout composant faisant partie de l'application doit être déclaré dans ce fichier pour être pris en compte.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="sio.lc.fr.firstapplication">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
```

```
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

La balise <application> : les composants doivent être définis entre la balise <application> et </application>.
La balise <activity> : spécifie la classe qui implémente une activité avec ses intentions.

En plus des déclarations qui figurent dans l'exemple ci-dessus, on peut trouver également :

- Les permissions
- uses-configuration : précise les spécificités de navigation supportées par l'applications (clavier physique, stylet…) ;
- uses-feature : indique les ressources matérielles indispensables à l'utilisation de l'application (audio, bluetooth, caméra, GPS…)
- supports-screens : spécifie les dimensions d'écrans supportées par une application (smallScreens, normalScreens, largeScreens, xlargeScreens).

6.2. Le fichier res/values/strings.xml

Toute ressource/attribut de type chaîne (@string) est renseigné dans le fichier res/values/strings.xml.

```
<resources>
    <string name="app_name">FirstApplication</string>
</resources>
```

6.3. Composants graphiques et agencement

Les composants graphiques de l'interface ainsi que leur agencement dans la vue sont décrits dans le fichier res/layout/activity_main.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="sio.lc.fr.firstapplication.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Les composants graphiques sont agencés à l'aide de conteneurs. Android utilise des gestionnaires de disposition : les layout managers. Android propose les layouts suivants :

- **LinearLayout** : conteneur qui place les widgets les uns à la suite des autres. Si l'attribut android :orientation="vertical", les composants sont placés les uns en dessous des autres (de haut en bas). Si l'attribut android :orientation="horizontal", les composants sont mis les uns à côté des autres (de gauche vers la droite).

- **RelativeLayout** : conteneur qui place les composants les uns par rapport aux autres (au-dessus, en-dessous, à côté…) ou par rapport au conteneur lui-même (en haut, en bas, à gauche, à droite).
- **TableLayout** : les éléments sont placés comme dans un tableau, dans des conteneurs `tableRow`.
- **ScrollView** : il va permettre de faire du scrolling sur des objets occupant une place plus importante que la hauteur de l'écran.
- **ConstraintLayout** : les composants sont agencés de façon relative aux autres à l'aide de contraintes de positionnement.

Les attributs `android :layout_width` et `android :layout_height` permettent de définir le remplissage de l'espace du conteneur par le composant :

- `wrap_content` : le widget occupe la place nécessaire
- `match_parent` : le widget occupe tout l'espace disponible dans son conteneur.

L'attribut `android :layout_gravity` spécifie l'alignement du composant par rapport à son parent. Par défaut, les composants sont alignés à partir du haut_gauche du conteneur. Les valeurs possibles sont :

- `left` : le composant sera positionné à gauche.
- `Right` : le composant sera positionné à droite.
- `center_horizontal` : le composant sera positionné au centre dans la dimension horizontale.
- `center_vertical` : le composant sera centré dans la dimension verticale.
- `center` : le composant sera positionné au centre dans son conteneur.
- `top` : le composant sera positionné en haut de son conteneur.
- `bottom` : le composant sera positionné en bas de son conteneur.

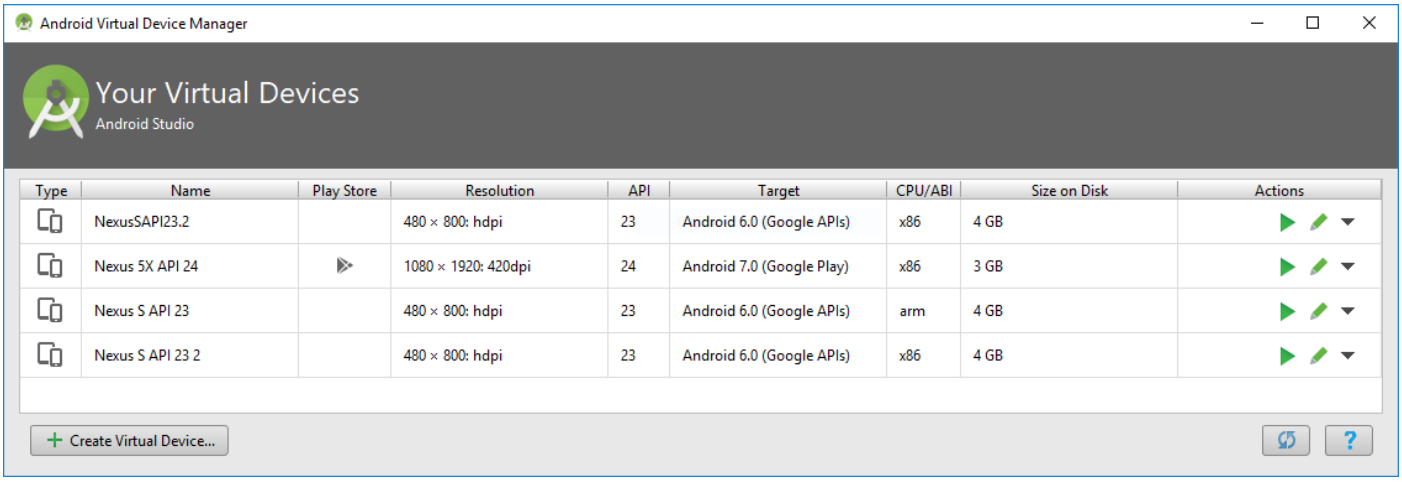
Il est possible de combiner ces valeurs à l'aide du caractère |.

L'attribut `android :layout_weight` attribue un poids à un composant dans le cas où plusieurs widgets doivent se partager un espace commun. Par défaut, il a la valeur 0.

7. Préparation d'un terminal virtuel

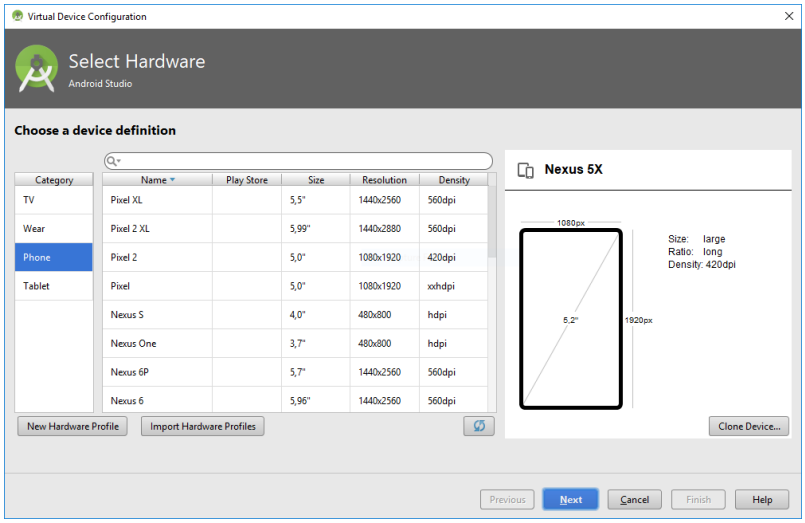
Les outils Android incluent un émulateur, c'est-à-dire un logiciel permettant de simuler un téléphone ou une tablette Android. Il permet de créer une large gamme d'émulateurs afin de simuler différents types de téléphones, de tailles d'écrans et de résolutions. Chaque configuration est stockée dans un AVD (Android Virtual Device) créé avec l'Android SDK et AVD Manager. Sous Android Studio, le menu Tools => Android => AVD Manager ou l'icône sous forme de smartphone dans la barre d'outils permet de gérer les AVD.

Si un ou plusieurs terminaux sont déjà configurés, le gestionnaire les présente sous forme de liste. Sinon, on peut en créer à l'aide du bouton Create Virtual Device :



Cliquer sur le bouton `Create Virtual Device`.

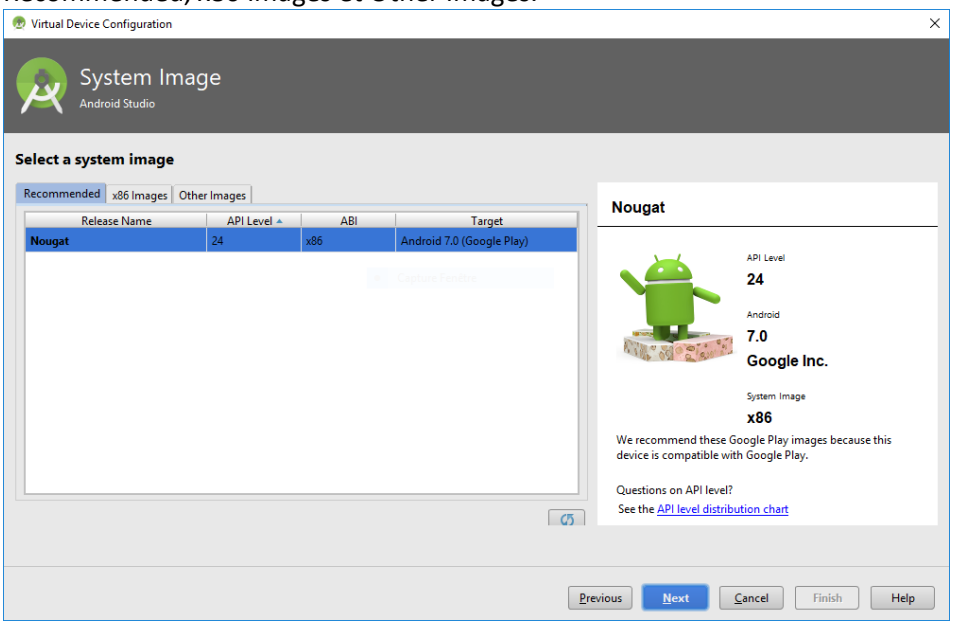
L'écran de l'assistant va présenter la liste des différents profils de terminaux, classés par catégorie : TV, Wear, Phone, Tablet.



Sélectionner, dans la catégorie Phone, un modèle de terminal Nexus. Par exemple : Nexus 5.

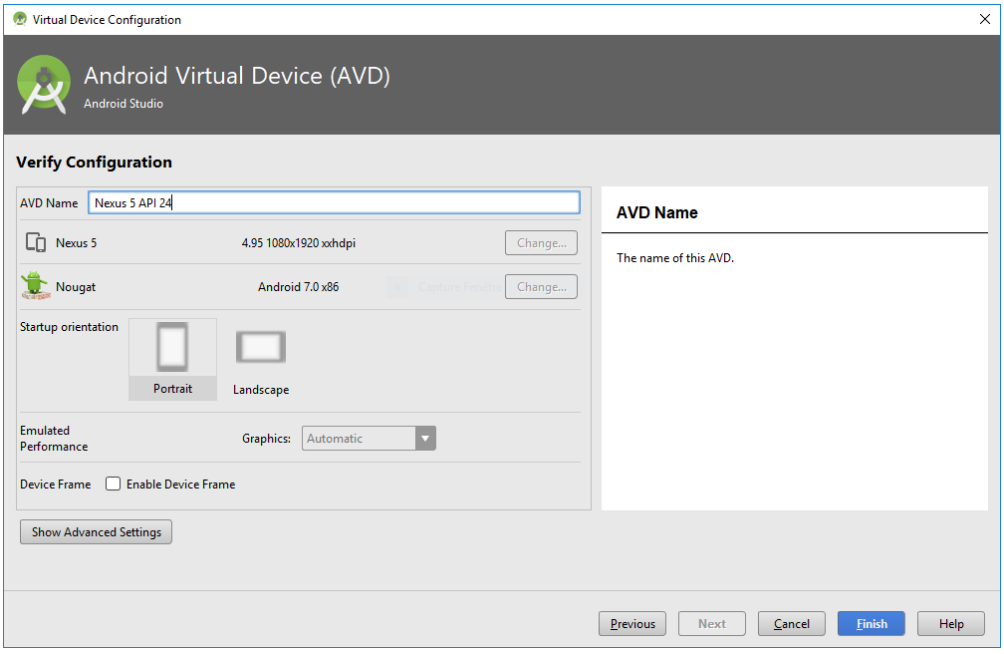
Cliquer sur le bouton `Next`.

L'écran suivant affiche la liste des versions Android organisée en trois onglets : Recommended, x86 Images et Other Images.



Si l'image système n'est pas présente sur le disque, un lien Download est alors présent à côté du nom de la Release, il faut alors télécharger l'image avant de pouvoir cliquer sur le bouton `Next`.

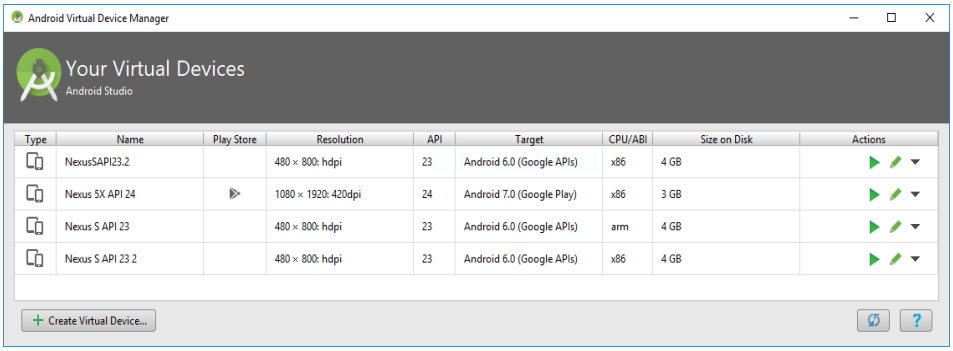
L'écran suivant permet de modifier tous les paramètres du terminal :



AVD Name : nom du terminal tel qu'il sera affiché dans la liste.
Modèle du terminal émulé.
Version d'Android exécutée par le terminal.
Orientation de l'écran du terminal au démarrage.
Emulated Performance : l'option Graphics indique si la carte graphique de l'ordinateur hôte doit être utilisée (Hardware) ou pas (Software). L'entrée automatic laisse le choix au SE.
Device Frame : ajouter ou non un habillage en forme de smartphone à la fenêtre de l'émulateur.

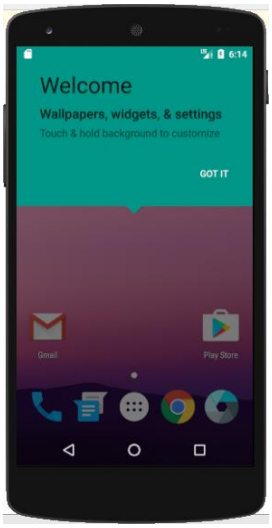
Cliquer sur le bouton `Finish`.

Une fois le terminal créé, l'écran présentant la liste des terminaux est affiché à nouveau en y ajoutant le nouveau terminal virtuel. Pour chaque terminal, il y a plusieurs actions possibles :



- démarrer le terminal
- Modifier les paramètres du terminal
- La liste déroulante permet de dupliquer le terminal, de supprimer les données utilisateurs qu'il renferme, d'ouvrir un explorateur à l'endroit où l'image est stockée, de visualiser un résumé de la configuration du terminal.

Cliquer sur l'icône Launch pour démarrer le terminal virtuel.



Contrairement aux terminaux physiques, l'application Play Store n'est pas disponible sur l'émulateur. La distribution de l'application ne peut donc se réaliser que sur un terminal physique. D'autre part, tous les tests faisant appel à des manipulations par geste ne peuvent être émulés.

Les terminaux virtuels sont plus lents en exécution que les terminaux physiques et fonctionnent dans une situation idéale : pas d'interruption causée par un appel, un SMS…

AVD n'est pas le seul logiciel d'émulation de terminaux Android.

8. Configurer un terminal physique

Tout terminal physique sous Android peut être utilisé pour tester une application.
L'outil ADB (Android Debug Bridge) permet de faire le lien entre le terminal physique et l'ordinateur de développement. Le terminal physique est relié à l'aide d'un câble USB.

Pour utiliser un appareil physique, il faut installer le driver de l'appareil sur l'ordinateur. Cette opération dépend à la fois du système d'exploitation et de la marque de l'appareil.

Ensuite, il faut indiquer au terminal Android qu'il autorise le débogage via USB :

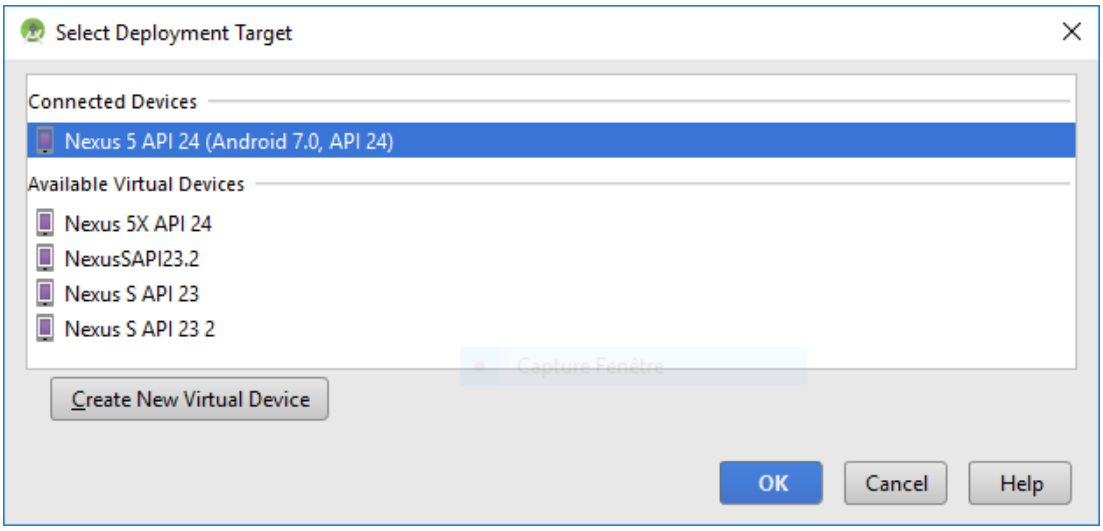
- Depuis les paramètres du système, accéder à l'écran : **A propos du téléphone**
- Positionner la liste pour voir le **Numéro du Build**
- Taper x fois sur cette ligne jusqu'à ce que le message : « il vous reste 3 tapes pour devenir développeur », puis un autre vous informe : « Vous êtes maintenant développeur ».
- Une fois le message obtenu, revenir à l'écran précédent. Dans la rubrique Système apparaît une nouvelle entrée : **Options pour les développeurs**.
- Sélectionner cette option. Un écran va s'afficher. Dans la rubrique **Débogage**, activer le débogage USB.

Le terminal physique Android est alors prêt à être utilisé dans le cadre du développement Mobile.

9. Lancement de l'application

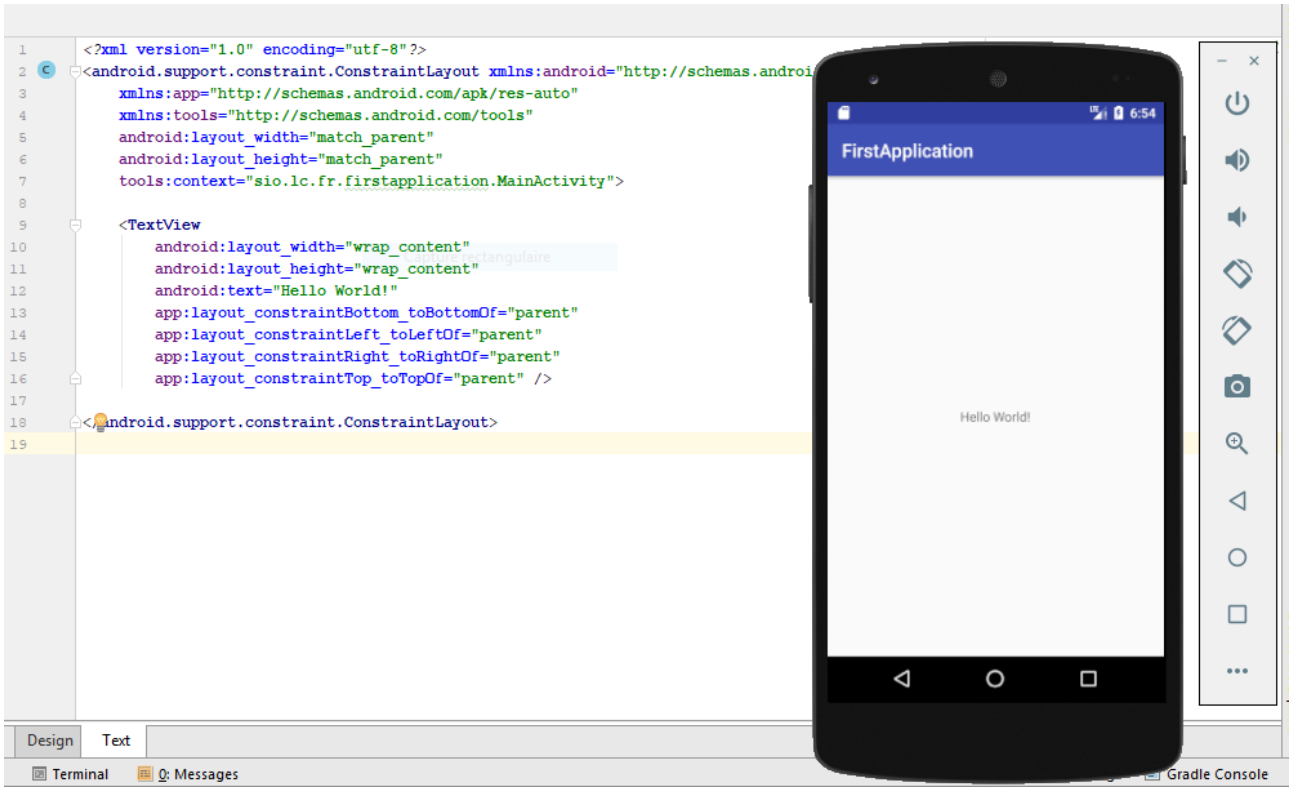
Une fois le terminal virtuel/physique prêt, on peut lancer l'exécution de l'application à l'aide du menu `Run => Run app`. Il est possible également d'utiliser l'icône de lancement (flèche verte) dans la barre d'outils ou alors d'appuyer simultanément sur les touches `Shift+F10`.

Une fenêtre pop-up présente la liste des terminaux Android physiques ou virtuels détectés.



Sélectionner un terminal et cliquer sur le bouton `OK`.

L'application est alors automatiquement lancée.



10. Outils de debug

Android Studio fournit un ensemble d'outils pour aider les développeurs. Nous allons présenter deux d'entre eux.

10.1. Les messages Toast

Les messages Toast sont de simples messages qu'on affiche à l'écran. Un message Toast prend la forme d'une fenêtre indépendante qui affiche un message sans mise en forme, durant une période plus ou moins courte.
Pour afficher un message Toast, on appelle la méthode suivante :

```
static Toast makeText(Context context, CharSequence text, int duree)
```

Cette méthode construit le message de type Toast. Les paramètres de cette méthode sont :

- `Context context` : contexte dans lequel s'exécute l'application. Context est une interface implémentée par plusieurs classes du framework, la plus utilisée étant la classe Activity.
- `CharSequence text` : chaîne de caractère à afficher.

- Int duration : durée d'affichage du message. Deux valeurs sont possibles : `Toast.LENGTH_SHORT` ou `Toast.LENGTH_LONG`.

Une fois le message construit, il sera affiché à l'aide de la méthode :

```
void show()
```

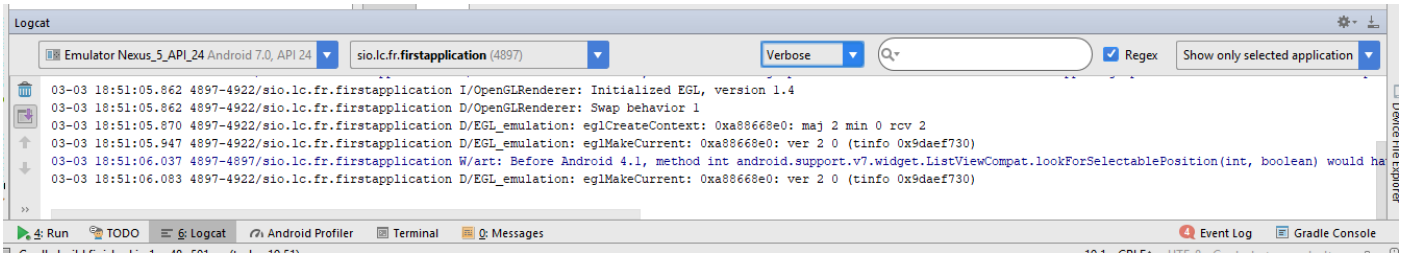
Exemple :

```
Toast.makeText(this, "Message de trace", Toast.LENGTH_LONG.show() ;
```

10.2. Journalisation : Logcat

Android fournit un mécanisme de journalisation appelé Logcat et une classe Java permettant de journaliser des messages d'information, de debug, d'avertissement, d'erreurs et d'assert.

Logcat permet de visualiser, en temps réel, le fichier de logs du terminal connecté. Pour afficher ce journal, il faut sélectionner l'onglet Logcat en bas de la fenêtre principale de l'environnement de développement Android Studio.



Le framework Android propose la classe Log qui permet d'écrire dans le fichier LogCat ses propres messages de log. A chaque niveau d'importance du log correspond une méthode :

```
Log.v(String tag, String message)    =>  verbeux
Log.d(String tag, String message)    => Debug
Log.i(String tag, String message)    => Information
Log.w(String tag, String message)    => Warning
Log.e(String tag, String message).   => Error
```

Le premier paramètre est une étiquette qui est affichée en regard du message. Une convention consiste à définir pour chaque classe de l'application une chaîne de caractère statique TAG qui contient le nom de l'activité.