

Préambule

Le système d'exploitation *Android* est actuellement l'OS le plus utilisé dans le monde faisant tourner des smartphones, tablettes, montres connectées, liseuses électroniques, télévisions interactives, et bien d'autres. C'est un système, open source qui utilise le noyau Linux. Il a été créé par *Android, Inc.* qui fut rachetée par Google en 2005. Le développement d'applications pour *Android* s'effectue en Java en utilisant des bibliothèques spécifiques.

Le but de ce tutoriel est de vous familiariser avec l'esprit de développement *Android* et ses bibliothèques. Nous introduirons les concepts de bases de création d'application en mettant en oeuvre quelques fonctionnalités simples. Ce tutoriel n'est en aucun cas exhaustive, le potentiel des applications *Android* est beaucoup plus ample, les exemples cités dans ce document ne devront pas brider votre imagination ni votre curiosité.

Sur le site officiel (<http://developer.android.com/develop/index.html>) pour les développeurs Android vous trouverez la documentation des classes (<http://developer.android.com/reference/packages.html>), des tutoriels (<http://developer.android.com/training/index.html>) ainsi que les lignes directrices (<http://developer.android.com/distribute/index.html>) pour préparer une distribution *Google Play*.

1 Installation de l'IDE

Dans cette section nous allons décrire la procédure d'installation d'un environnement de développement *Android*.

 Attention : Il faut exécuter les étapes dans l'ordre cité ci-dessous.

1. Téléchargez (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>) le dernier JDK (Java Development Kit) que vous pouvez trouver sur le site d'*Oracle*¹.

1. Ce tutoriel a été réalisé avec JDK8u65

2. Désinstallez des éventuelles versions antérieures du *JDK*.
3. Installez le nouveau *JDK*.
4. Téléchargez (<http://developer.android.com/sdk/index.html>) *Android Studio*. Il contient l'environnement de développement, le *SDK (Software Development Kit) Android* avec la dernière version de la plateforme, ainsi qu'un émulateur.
5. Lancez l'exécutable pour démarrer l'installation et suivez le wizard ².

2 Configuration de l'IDE

Installation des paquets supplémentaires et des mises à jours

1. Lancez *Android Studio*.
2. Nous commencerons par nous assurer que nous possédons tout ce qu'il faut pour développer. Dans la page de démarrage, sélectionnez *Configure -> SDK Manager*. Dans le gestionnaire (fig.1) vous verrez la version du *SDK* installé (avec les mises jour disponibles) et aussi la version de l'*API (Application Programming Interface)* installée et la version du *OS* pour laquelle elle vous permettra de développer. Installez les éventuelles mises à jour. Assurez vous de cocher au moins un *System Image* pour l'émulateur.
3. Dans l'onglet *SDK Tools* assurez vous d'avoir au moins
 - *Android SDK Build Tools*
 - *Android SDK Tools*
 - *Android SDK Platform Tools*
 - *Android Support Library*
 - *Android Support Repository*
 - *Google Repository*
 - *Google Play Services*
4. Quand vous aurez terminé, cliquez *Apply* pour lancez les installations des éléments supplémentaires.

2. Si le répertoire Java n'est pas détecté automatiquement, il faudrait définir une variable d'environnement `JAVA_HOME` qui indique le répertoire où vous avez installé le *JDK* (ex : `C:\Program Files\Java\jdk1.7.0_21`)

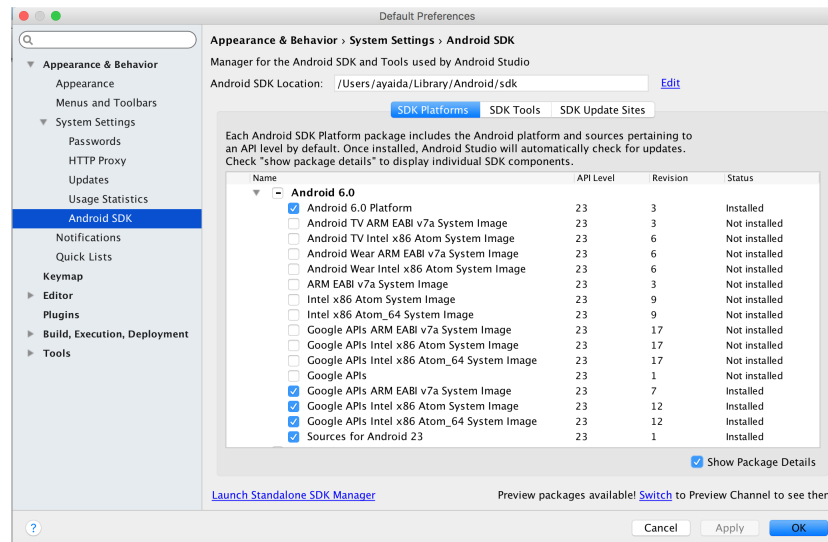


FIGURE 1 – SDK Manager : Dans cet exemple, il existe une mise à jour disponible pour l'API 23 qui permet un développement pour Android 6.0

3 Notre première application Android

Création d'un projet et d'une application "Hello World"

1. Dans le menu *Quick Start*, sélectionnez *Start a new Android Studio Project*, et renseignez les informations comme dans la figure 2a.
2. Cliquez sur *Next* puis remplissez les champs comme dans la figure 2b.

Application name : c'est le nom qui va apparaître dans la liste des applications sur l'appareil et dans le *Play Store*.

Company domain : c'est un qualifiant qui apparaîtra dans le nom du package.

Package name : il est utilisé comme identifiant de l'application, il permet de considérer différentes versions d'une application comme étant une même application. Il doit être unique parmi tous les packages installés sur le système.

Minimum required SDK : c'est la version Android la plus ancienne sur laquelle l'application peut tourner. Il faut éviter de remonter

trop en arrière, ça réduirait les fonctionnalités que vous pourriez donner à votre application³.

3. Cliquez sur [Next](#). Nous arrivons à la création d'une activité (un écran avec une interface graphique). Sélectionnez [Blank Activity](#) (fig. 3a) et cliquez [Next](#).
4. Renseignez les champs comme dans la figure 3b. Vous pourriez choisir l'utilisation de fragments, mais pour faire simple nous poursuivrons sans fragments. Chaque activité dispose d'un *layout* qui définit la façon dont les composants seront disposés sur l'écran. Une activité peut être divisée en portions (ou fragments) chacune ayant son propre *layout*. La notion de fragment a été introduite pour favoriser la ré-utilisabilité de morceaux d'activité (un fragment peut être défini une fois et réutilisé dans plusieurs activités).
5. Cliquez sur [Finish](#), le projet est créé.

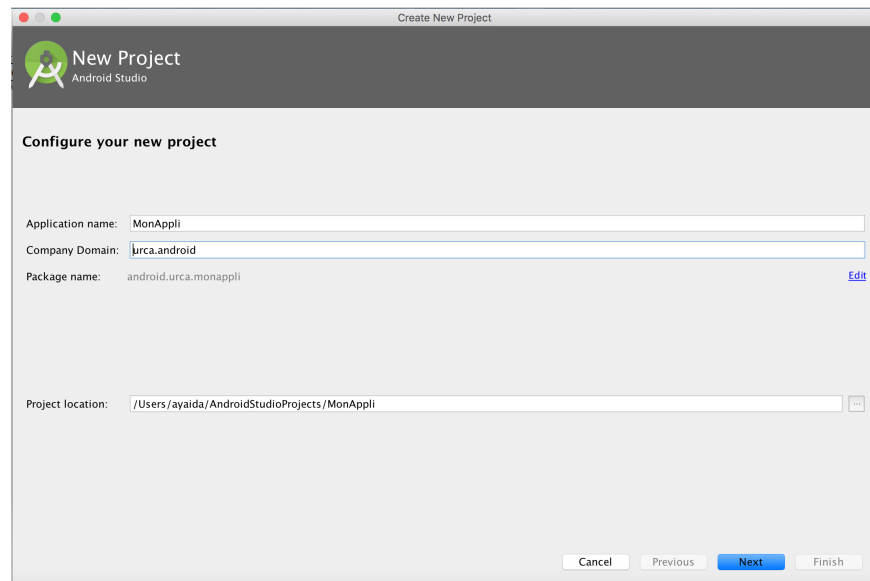
4 Exécution de l'application

4.1 Sur un émulateur

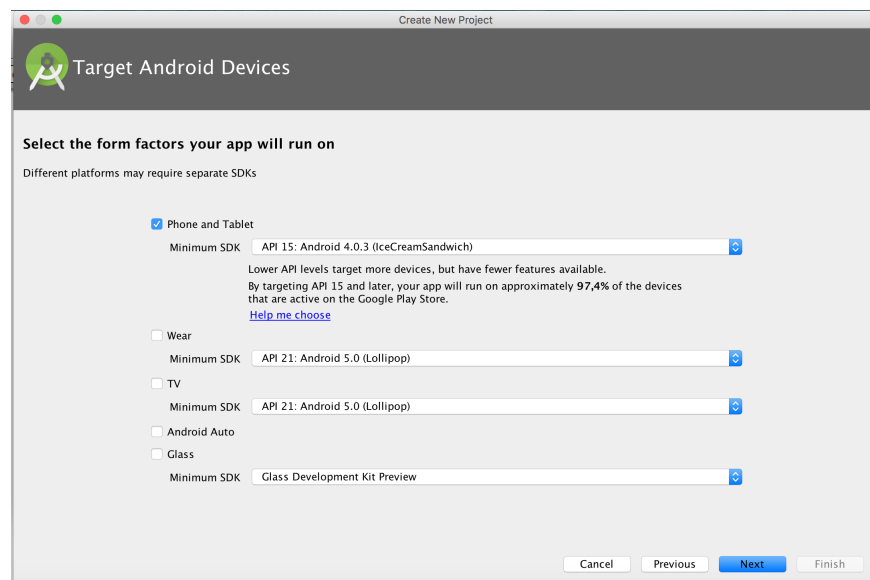
Un émulateur permet de reproduire le comportement d'un appareil réel d'une façon virtuelle. L'utilisation d'un émulateur nous évite d'avoir à charger à chaque fois l'application dans un appareil pour la tester. On pourra ainsi lancer l'application dans l'*IDE* et elle s'exécutera sur un appareil virtuel appelé *Android Virtual Device AVD* qui émule le comportement d'un téléphone, une tablette ou autre.

Pour configurer un émulateur, allez dans [Tools > Android > AVD Manager](#), un émulateur existe par défaut, mais vous pourriez rajouter d'autres appareils en cliquant sur [Create Virtual Device](#). Dans l'exemple de la figure 4a nous rajoutons une tablette *Nexus 10* avec une extra haute résolution (*xhdpi*). Nous sélectionnons ensuite le processeur qui sera émulé (fig. 4b). En cochant [Show downloadable system images](#), vous pouvez télécharger d'autres images systèmes avec d'autres versions *Android*. En cliquant sur [Next](#) Vous avez ensuite la possibilité de configurer d'autres paramètres.

3. Il est possible d'activer certaines fonctionnalités (non essentielles) de votre application uniquement quand elle tourne sur une version qui les supporte. Pour plus d'informations consulter cette page : [<http://developer.android.com/sdk/index.html>]

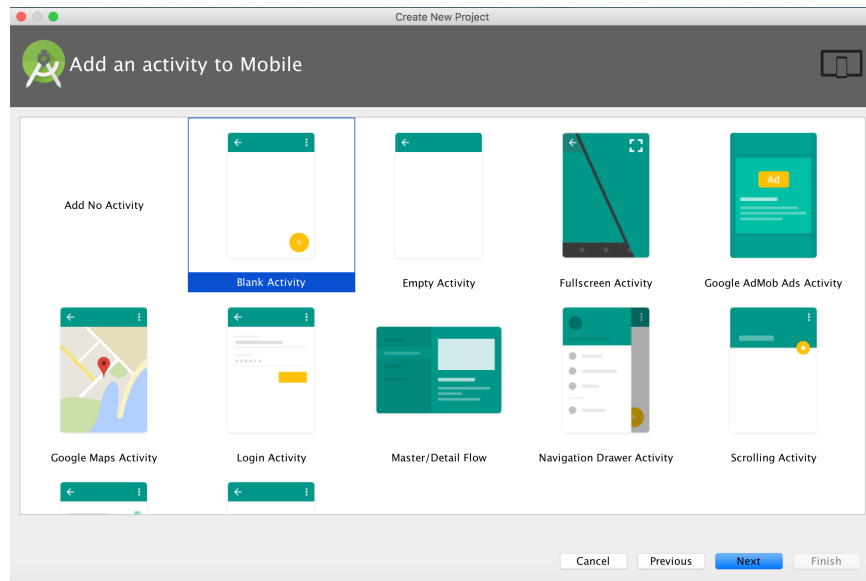


(a) Configuration du projet

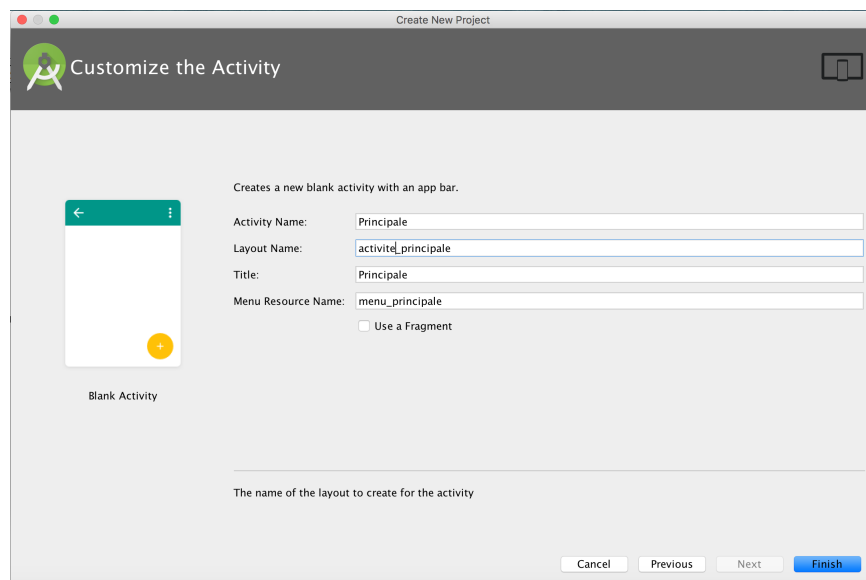


(b) Choix des cibles

FIGURE 2 – Création d'un projet



(a) Ajout d'une activité



(b) Personnalisation de l'activité


FIGURE 3 – Création d'une activité

Notez qu'à la création de l'appareil sa résolution vous est signalée. Ceci est important à noter pour l'intégration d'images dans l'application.

Pour lancer l'exécution sur l'émulateur, appuyez sur le bouton d'exécution (fig. 5) et sélectionnez l'émulateur sur lequel vous souhaitez lancer l'application. Vous pouvez cocher *Use same device for future launches* pour éviter d'avoir à sélectionner l'appareil à chaque lancement. L'émulateur se lance, ça peut prendre quelques minutes soyez patients. Rassurez-vous, vous n'aurez pas à le relancer à chaque fois que vous compilez votre projet, laissez-le ouvert et à chaque fois que vous compilez et relancez votre application, elle pourra être chargée dans l'émulateur en cours (il vous le proposera parmi les *Running devices*).

4.2 Sur un appareil réel

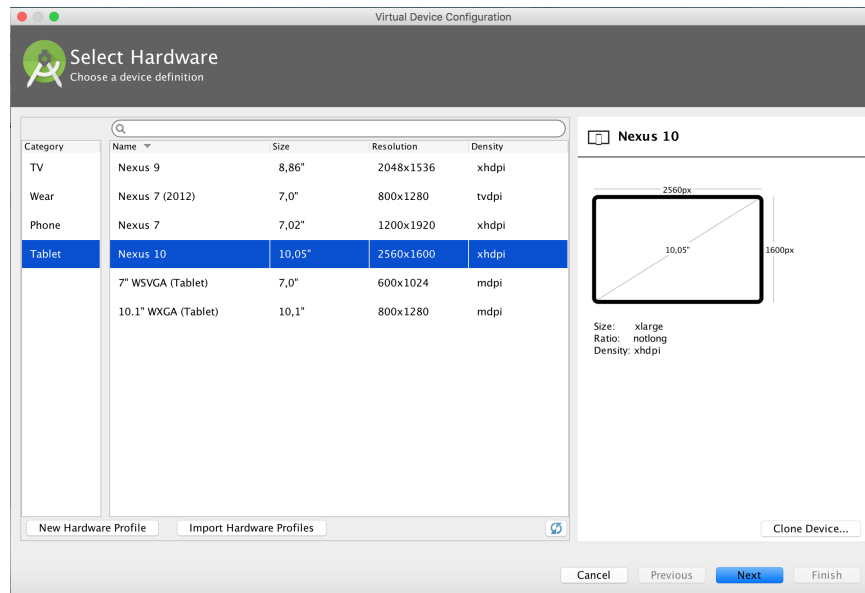
Connectez l'appareil par câble *USB* à l'ordinateur et installez le pilote si nécessaire. Activez l'option de débogage *USB* sur votre appareil (dans les paramètres, sous développement ou option de développement). Lancez l'application depuis *Android Studio* comme précédemment. Si on vous demande de choisir l'appareil, sélectionnez *Choose a running device*, puis votre téléphone ou tablette. *Android Studio* installera l'application sur votre appareil et la lancera.

 Une fois que votre application est compilée, un fichier *.apk* est créé dans le dossier *app\build\outputs\apk* de votre répertoire de travail. C'est l'exécutable de votre application. C'est ce fichier que vous devez déployer pour distribuer votre application. Le contenu de ce fichier peut être inspecté à l'aide de n'importe quel logiciel standard de compression/décompression de fichiers.

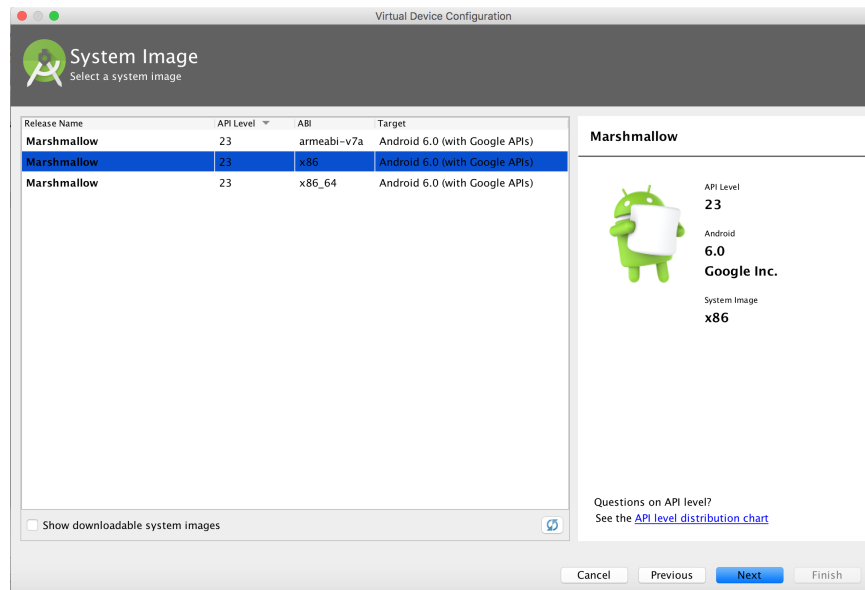
5 Se repérer dans le projet

La figure 6 montre les principaux éléments de l'interface *Android Studio*. Tout projet *Android* doit respecter une hiérarchie bien précise qui permettra au compilateur de retrouver les différents éléments et ressources lors de la génération de l'application. Cette hiérarchie favorise la modularité des applications *Android*.

À la création du projet, *Android Studio* crée automatiquement des dossiers



(a) Choix de l'appareil



(b) Choix du processeur

FIGURE 4 – Création d'un AVD

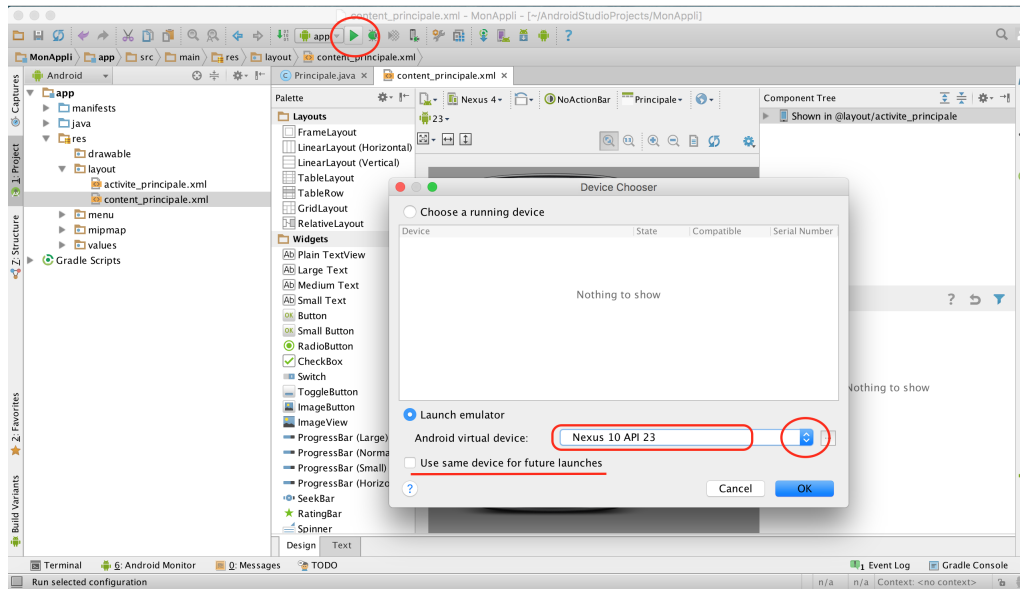


FIGURE 5 – Exécution de l'application

pour contenir les fichiers de code *Java*, les fichiers *XML*, et les fichiers multimédias.

L'explorateur de projet vous permettra de naviguer dans ces dossiers.

Les dossiers que nous utiliserons le plus sont *java* et *res*. Le premier contient le code *Java* qui définit le comportement de l'application (situé dans le répertoire de votre projet sous *app\src\main*) et le second comporte des sous-dossiers (dans *app\src\main\res*) où sont stockés les ressources qui définissent l'interface de l'application (l'apparence).

La séparation entre fonctionnalité et apparence est un point essentiel de la philosophie *Android*.

Le code de la classe principale de l'application (*Principale.java*) est situé dans le sous-dossier *android.urca.monappli* de *java*. C'est dans le dossier *java* que seront enregistrées toutes les classes que nous allons créer dans ce projet⁴.

4. Vous remarquerez qu'une classe *ApplicationTest* est créée automatiquement par *Android Studio*. Cette classe vous permet d'écrire un code pour tester les différentes fonctionnalités de votre application plutôt que de les tester "à la main". Pour plus d'informations sur la création de scénario de test consulter cette page : <http://developer.android.com/training/activity-testing/activity-basic-testing.html>.

Par ailleurs, tout ce qui touche à l'interface utilisateur sera intégré dans les sous dossiers de *res*, dont voici une brève description :

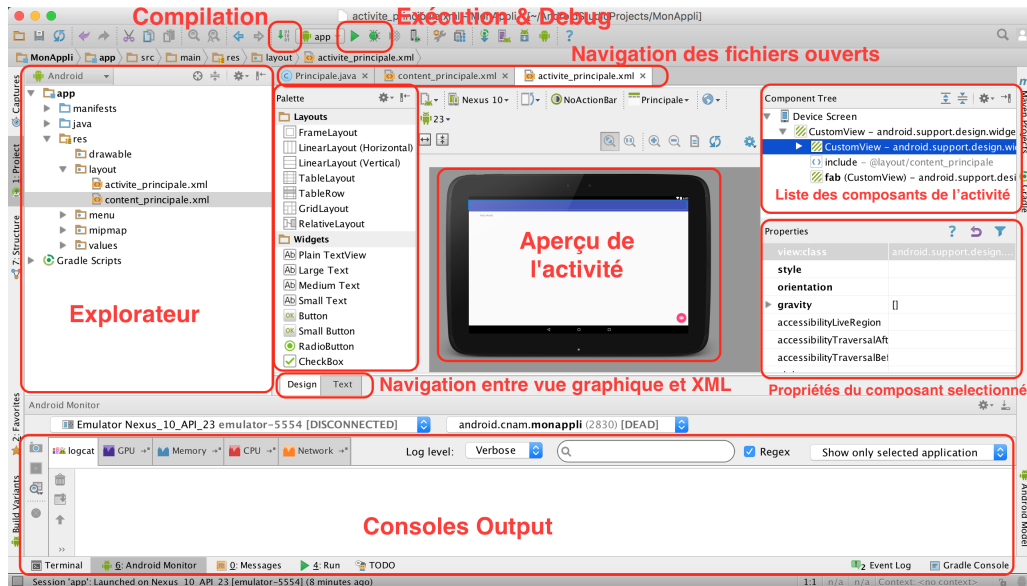


FIGURE 6 – Aperçu de l'interface Android Studio

layout : regroupe les fichiers *XML* qui définissent la disposition des composants sur l'écran. Il contient déjà, dès la création du projet, le *layout* de l'activité principale que nous avons créée. Deux fichiers *XML* sont créés par *Android Studio* : *activite_principale.xml* et *content_principale.xml*. Le premier définit l'apparence générale de l'activité : sa disposition, sa taille, sa barre d'outil, éventuellement des boutons d'action flottant ainsi que son *layout* qui n'est autre que *content_principale.xml*. Ce sera donc ce dernier que nous devrons manipuler pour disposer les composants de l'activité et créer notre interface graphique.

drawable-**** : contient tout élément qui peut être dessiné sur l'écran : images (en *PNG* de préférence), formes, animations, transitions, etc.. Cinq dossiers *drawable* permettent aux développeurs de proposer des éléments graphiques pour tout genre d'appareil *Android* en fonction de sa résolution. En manipulant correctement ces dossiers on peut ainsi créer des applications avec une interface qui s'adapte à chaque résolution d'écran avec un seul fichier *.apk*.

- dpi : low-resolution dots per inch. Pour des images destinées à des écrans de basse résolution ($\sim 120dpi$)
- mdpi : pour des écrans de moyenne résolution ($\sim 160dpi$)
- hdpi : pour des écrans de haute résolution ($\sim 240dpi$)
- xhdpi : pour des écrans ayant une extra haute résolution ($\sim 320dpi$)
- xxhdpi : pour des écrans ayant une extra extra haute résolution ($\sim 480dpi$)

menu : contient les fichiers *XML* définissant les menus.

mipmap : contient les images de l'icône de votre applications sous différentes résolutions.

values : contient les fichiers *XML* qui définissent des valeurs constantes (des chaînes de caractères, des dimensions, des couleurs, des styles etc.).

gradle : *Android Studio* utilise un système qu'on appelle *Gradle* pour compiler et générer les applications. Pour fonctionner le *Gradle* a besoin d'un script qui définit les règles de compilation et génération (configuration et dépendances). *Android Studio* crée ainsi un script *gradle* pour chaque module (*build.gradle (Module :app)*) du projet ainsi qu'un script pour le projet entier (*build.gradle (Project : MonAppli)*)⁵. Dans le *build.gradle* de l'application on définit entre autre la version du *SDK* utilisée pour la compilation, la version minimale du *SDK* nécessaire pour faire tourner l'application (rétro-compatibilité), l'identifiant de l'application (le nom du package), etc⁶.

Vous trouverez également dans le dossier *manifests* du projet un fichier nommé *AndroidManifest.xml*. Ce fichier est obligatoire dans tout projet *Android*, et doit toujours avoir ce même nom. Ce fichier permet au système de reconnaître l'application.



On peut accéder à une description rapide des classes ou des attributs *XML* en sélectionnant leur nom dans l'éditeur et appuyant sur *Ctrl+Q*.

5. *Android Studio* sépare projet et module. Le module c'est une application ou une librairie. En général un projet contient un seul module de préférence quand il s'agit d'applications, mais peut contenir plusieurs modules quand on développe des librairies.

6. Le *Gradle* est l'équivalent du *gcc* en langage *C* et le *gradle.build* l'équivalent du *makefile*.