Présentation de l'environnement

1. Objectifs

Les objectifs de ce chapitre sont les suivants :

* Avoir la culture minimum nécessaire pour comprendre l'environnement Android.
* Connaitre la structure de la plateforme Android.
* Comprendre ce qu'est JetPack
* Connaitre la structure d'une application.
* Connaitre l'environnement de développement d'une application Android.
* Savoir créer et exécuter un projet Android.

1. Présentation

1.1 Les grandes dates pour Android et Kotlin

Android est le nom d'une startup créée en 2003, Andy Rubin un des deux cofondateurs, est un ancien ingénieur de chez Apple. Cette startup est à l’origine du travail sur le système d'exploitation Android qui était exclusivement dédié aux téléphones et tablettes. En 2005, Google rachète cette startup et son système d'exploitation dans le but de créer un système d'exploitation compatible avec tous les appareils mobiles. En janvier 2007, Apple sort son iPhone révolutionnaire ; en novembre, Google crée le Open Handset Alliance, un consortium de plusieurs dizaines d'entreprises, dans le but d'accélérer le développement de son système d'exploitation.

En 2011, présentation de Kotlin à la conférence JVM Langage Summit par JetBrains.

En 2014, Google s'associe avec Jetbrain, L'IDE Android Studio basé sur l'IDE IntelliJ IDEA devient l'IDE officiel pour les développements Android.

A partir de 2015, plusieurs grosses failles de sécurité ont été découverte, Google a réagi en proposant des mises à jour régulières du système d'exploitation aux utilisateurs et en repensant les outils proposés aux développeurs Android.

En 2017, officialisation du support de Kotlin pour la plateforme Android lors de la célèbre conférence du Google i/o.

En 2019, toujours lors de la conférence [Google I/O](https://fr.wikipedia.org/wiki/Google_I/O), Kotlin devient officiellement le [langage de programmation](https://fr.wikipedia.org/wiki/Langage_de_programmation) recommandé par [Google](https://fr.wikipedia.org/wiki/Google) pour le développement des applications Android.

Le 3 septembre 2019, arrivée d'Android 10 avec l'API 30 qui propose androidx et architecture component comme modèle de développement vivement recommandé par google.

Aujourd'hui, Android détient plus de 80 % des parts de marché et Kotlin est utilisé principalement pour les nouveaux projets (96 % selon JetBrains).

* 1. Les avantages de Kotlin

Voici les principaux avantages de ce langage :

* Moins verbeux : le nombre de lignes de code d’un programme Kotlin est nettement plus faible qu’un même programme développé en Java.
* Plus explicite : moins de lignes de code avec un code qui s’autocommente davantage, c’est-à-dire que le rôle des instructions est plus clair.
* Pratique : moins verbeux, moins explicite, donc plus pratique.
* Moins d’erreurs avec un système de gestion des NullPointerException séduisant.

En résumé, il est plus productif.

* 1. La plateforme Android

Android est un système d'exploitation basé sur le noyau Linux. Le noyau Linux est solide en termes de sécurité, déjà très éprouvé, et simplifie la création de pilote, car le système de pilote Linux est bien connu des fabricants de périphériques dans le monde entier. Le noyau Linux contient donc les pilotes pour gérer par exemple le son, l'affichage, le Bluetooth, le Wi-Fi, etc.

Android propose des interfaces permettant aux constructeurs des différents composants matériels de développer leurs pilotes de manière standardisée. Ainsi tous les composants matériels fonctionnent relativement de la même manière et cela facilite énormément la fabrication de téléphone.

Android contient de nombreuses librairies en C et C++ utilisant ces pilotes au fonctionnement standardisé. Ces librairies contiennent les fonctionnalités de base pour les différentes actions à réaliser avec le matériel. Ces librairies sont essentiellement utilisées par Android Runtime (ART).

ART est l'environnement d'exécution des applications. Concrètement, ART crée une machine virtuelle pour chaque application, ainsi, si l'application plante, c'est la machine virtuelle de l'application qui s'arrêtera et non l'ensemble des applications. Ce système permet de protéger le système d'exploitation face aux applications problématiques.

Tout ce qui vient d'être présenté relève du bas niveau, les développeurs d'applications mobiles interagissent rarement avec ces différents éléments. Ce dont ont besoin les développeurs pour développer des applications c'est de l'ensemble complet des fonctionnalités du système d'exploitation fourni par le Framework Android.

Ce Framework offre des fonctionnalités de haut niveau simplifiant ainsi la réutilisation des composants et services de base.

Ce Framework constitue les outils de construction qui sont utilisés pour construire les applications.

Parmi ces outils, certains permettent :

* De créer des interfaces graphiques avec des composants pratiques (affichage de texte, saisie de texte, bouton, liste…),
* De gérer les ressources de l'application facilement,
* De gérer la cinématique des différents écrans d'une application,
* Et bien plus encore…

1. Android JetPack

Android Jetpack est une suite de bibliothèques permettant aux développeurs et aux développeuses d'avoir des applications avec une architecture de qualité, de suivre les meilleures pratiques et produire des fonctionnalités qui fonctionnent de manière cohérente sur toutes les versions, appareils Android sans s'occuper de l'adaptabilité afin de se concentrer uniquement sur le code essentiel qui les intéresse.

1. Processus de construction d'une application

Le processus de construction pour une application de base suit ces étapes :

Compilation des sources

Le compilateur convertit le code source en fichiers DEX (Dalvik Executable), qui incluent le bytecode qui s'exécute sur les appareils Android, et les ressources utilisées par l'application.

L'outil de packaging APK combine les fichiers DEX en un seul fichier APK. Cependant, avant que votre application puisse être installée et déployée sur un appareil Android, le fichier APK doit être signé.

Signer l'APK

Il est nécessaire de signer l'APK dans le but d'identifier l'auteur et ainsi de sécuriser l'application. Seul l'auteur pourra apporter des mises à jour sur l'application et décider des interactions de l'application vers d'autres applications.

À la fin du processus de génération, l'auteur dispose d'un fichier APK de débogage ou d'un fichier APK de version qu'il peut utiliser pour déployer, tester ou publier l'application.

1. Environnement de développement
   1. Présentation

L'environnement minimum pour développer une application Android est composé des éléments suivants :

* Un SDK Android
* Android Studio
* Un émulateur
  1. SDK Android

Le kit de développement d'Android est un ensemble complet d'outils de développement.

Il inclut un débogueur, des bibliothèques logicielles, un émulateur, de la documentation, des exemples de code et des tutoriels.

Android Studio est l'IDE officiel permettant d'utiliser le SDK Android.

* 1. Android Studio
     1. Présentation

Android Studio est l’environnement de développement officiel pour développer des applications Android. Il est complet et parfaitement adapté aux développements d'application native Android.

Il est basé sur IntelliJ IDEA, qui est un IDE reconnu et apprécié par les développeurs pour la variété de ses outils permettant de simplifier et d’améliorer la productivité de leurs développements.

Android Studio permet principalement d’utiliser le SDK Android et d'éditer les fichiers Kotlin et de configuration XML d'une application Android.

Android Studio a vu sa première version stable sortir en 2014. C'est cet IDE qui est vivement recommandé pour les développements Android.

Android Studio et le SDK sont téléchargeables à cette adresse :   
<https://developer.android.com/studio/>

* 1. Gradle
     1. Présentation

Gradle est un moteur de production permettant de faire une multitude d'actions sur le cycle de vie d'un projet telles que gérer les dépendances ; gérer le processus de compilation et de génération des applications. Dans ce livre, Gradle ne sera utilisé que pour sa partie gestionnaire de dépendances. En effet, Gradle permet d'ajouter des libraires de manière très simple au projet.

* + 1. Gestion des dépendances

Lors de la création d'un nouveau projet, vous pouvez apercevoir dans l'explorateur de projet une section nommée "Gradle Scripts", cette section contient différents projets permettant de configurer Gradle. Les seuls fichiers intéressants pour utiliser ce livre et découvrir le développement Android sont les fichiers "build.gradle" et le fichier gradle.properties dans le cas ou vous souhaitez utilisez un proxy lors de vos développements.

C'est dans les fichiers "build.gradle" que l'on définit les dépendances de l'application et plus particulièrement dans la partie dependencies{…}.

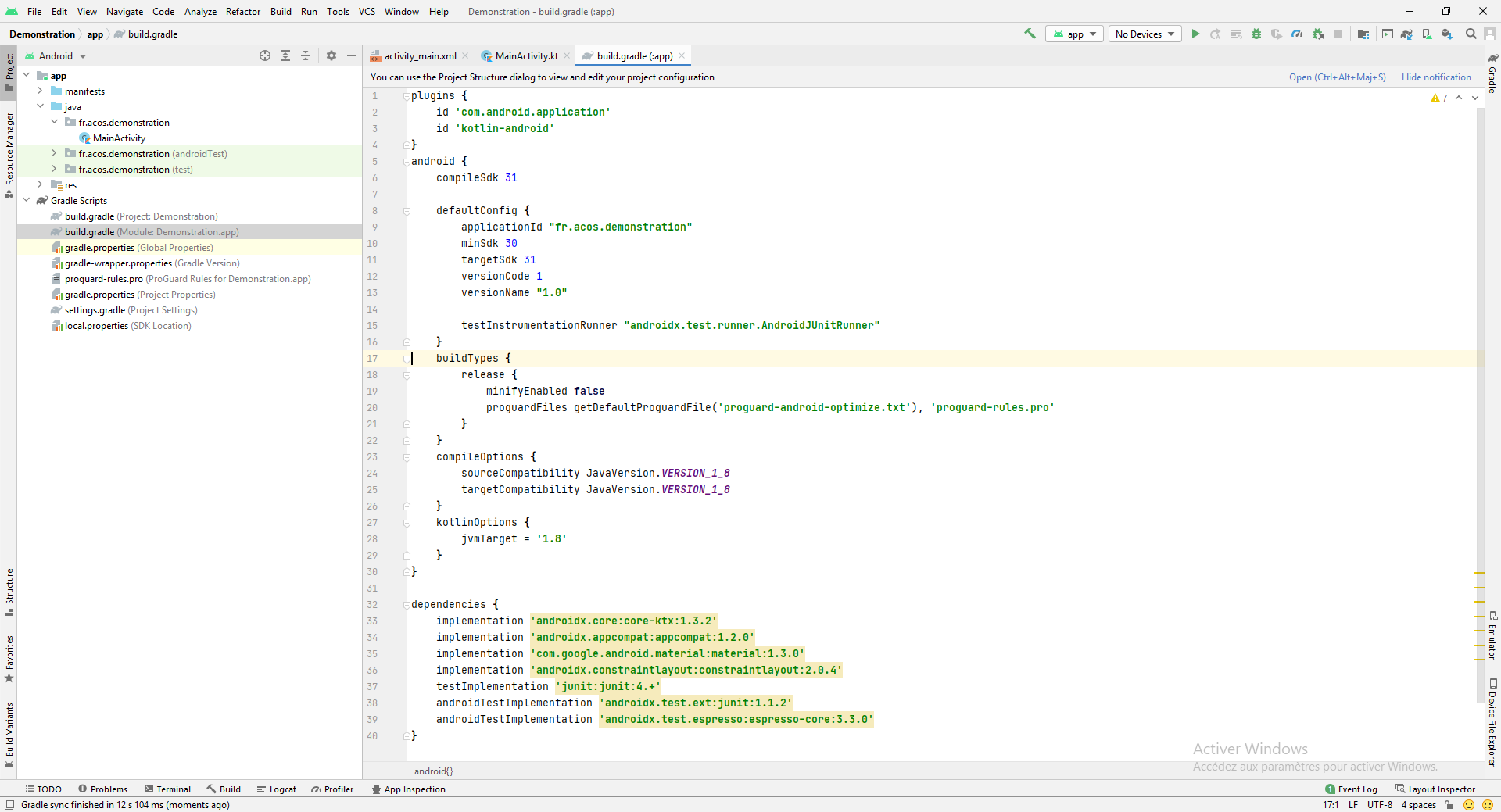


Figure Apercu du fichier build.gradle permettant de géré les dépendances

Pour configurer un serveur proxy avec gradle il vous suffit de modifier le fichier gradle.properties comme ceci :

## For more details on how to configure your build environment visit  
# http://www.gradle.org/docs/current/userguide/build\_environment.html  
#  
# Specifies the JVM arguments used for the daemon process.  
# The setting is particularly useful for tweaking memory settings.  
# Default value: -Xmx1024m -XX:MaxPermSize=256m  
# org.gradle.jvmargs=-Xmx2048m -XX:MaxPermSize=512m -XX:+HeapDumpOnOutOfMemoryError -Dfile.encoding=UTF-8  
#  
# When configured, Gradle will run in incubating parallel mode.  
# This option should only be used with decoupled projects. More details, visit  
# http://www.gradle.org/docs/current/userguide/multi\_project\_builds.html#sec:decoupled\_projects  
# org.gradle.parallel=true  
#Mon Nov 15 11:26:21 CET 2021  
systemProp.http.proxyHost=XXX.XXX.XXX.XXX  
systemProp.https.proxyHost=XXX.XXX.XXX.XXX  
systemProp.https.proxyPort=XXXX  
systemProp.http.proxyPort=XXXX

1. Application Android
   1. Création d'un nouveau projet

Pour créer un nouveau projet, suivre les étapes suivantes :

* File -> New -> New Project

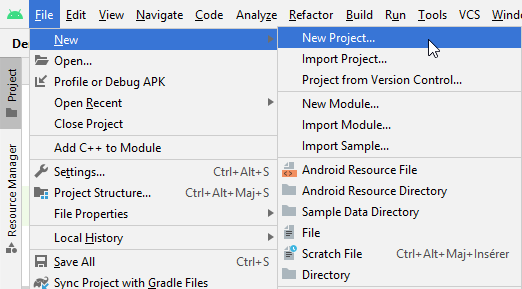


Figure Créer un nouveau projet

* Choisir le type de projet à créer

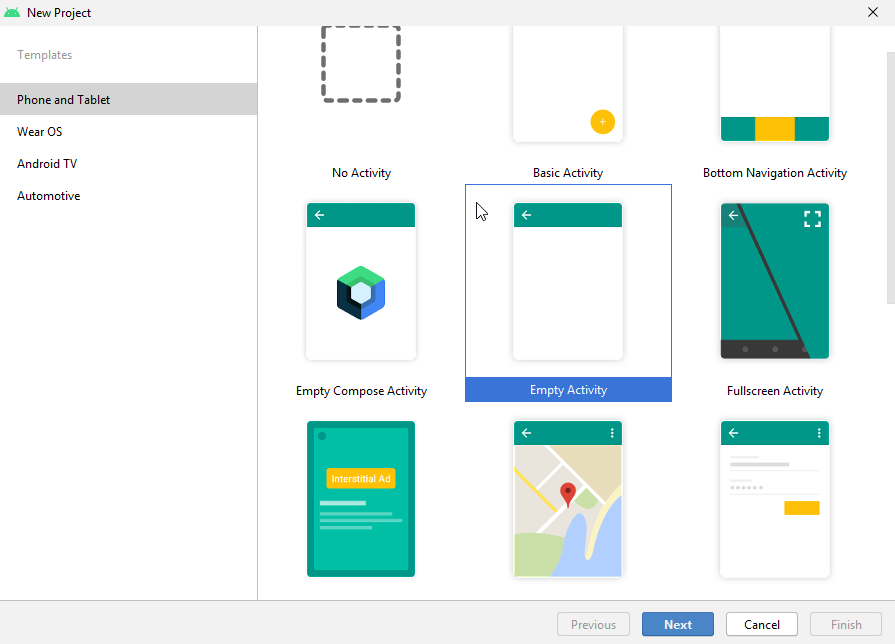


Figure Empty Activity représente une application vierge.

* Saisir les informations nécessaires pour la création du projet puis appuyer sur Finish.

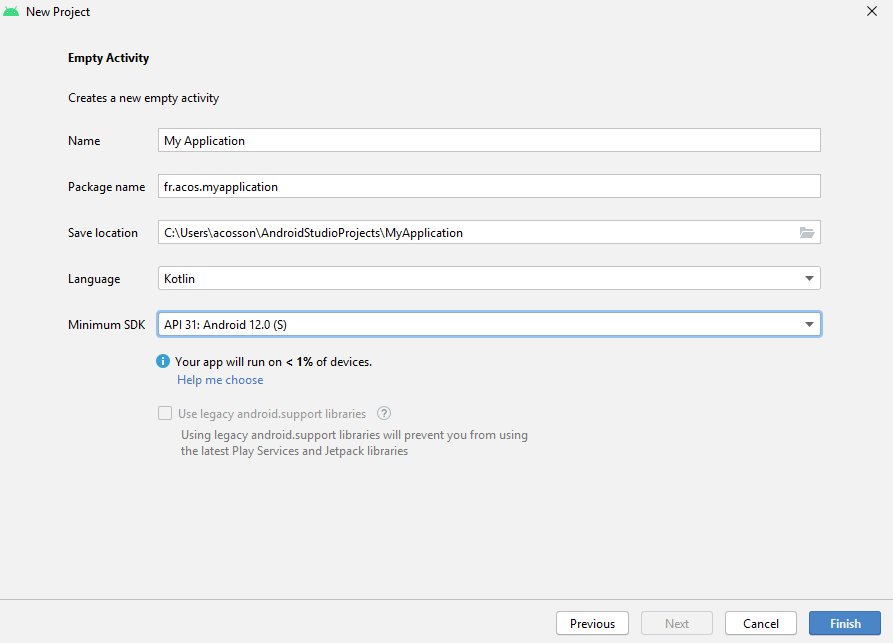


Figure Ecran final avant création de l'application

* Parmi les paramètres de configuration du projet, il est intéressant de voir qu'il est possible de choisir le langage de programmation à utiliser pour le développement et il est aussi possible de choisir la version de l'API à utiliser, il est préférable de choisir une version en Release.
  1. Structure d'une application

Après avoir créer un nouveau projet une architecture de base est proposée.

Une image contenant texte

Description générée automatiquement

Figure Architecture de base après création de l'application

Le dossier manifests contient un fichier nommé AndroidManifest.xml, celui-ci décrit le contenu de l'application, les permissions nécessaires au bon fonctionnement de l'application ainsi que quelques éléments de style.

Le dossier java contient toute la logique applicative, des contrôleurs, en passant pas les composants d'accès aux données ainsi que les classes modèles et tests unitaires.

Le package fr.acos.super\_application contient toute la logique applicative sans les tests unitaires.

Le package fr.acos.super\_application (androidTest) contient les tests unitaires devant être exécutés sur l'appareil mobile.

Le package fr.acos.super\_application (test) contient les tests unitaires pouvant être exécutés sur la machine hôte du développeur.

Le dossier res contient toutes les ressources statiques de l'application, les ressources statiques de l'application doivent obligatoirement se trouver dans un sous dossier du dossier res.

Le dossier drawable contient les images de l'application.

Le dossier layout contient les ihms de l'application.

Le dossier mipmap contient les icones de lancement de l'application.

Le dossier values contient des fichiers xml contenant eux même des informations sous forme de clé/valeur.

Le dossier Gradle Scripts contient les fichiers permettant de définir la configuration du compilateur et des dépendances nécessaires à l'application.

* 1. Exécution d'une application

Pour exécuter une application, il est nécessaire d'avoir un appareil mobile disponible, soit réel soit un émulateur.

Pour créer un émulateur, il est nécessaire de se rendre dans l'outils AVD Manager dans l'onglet Tools d'Android Studio.

Pour connecter un appareil mobile Android il est nécessaire de mettre le téléphone en mode développeur et d'installer le pilote du téléphone sur votre ordinateur.

Une fois un appareil réel ou émulé disponible il faut cliquer sur Run 'app' dans l'onglet Run.

1. Conclusion

Dans ce chapitre il a été vu que l'environnement Android / Android Studio / Kotlin est unique et relativement jeune mais offre des outils performants et spécialement dédiés au développement d'applications Android. L'architecture, les pratiques et les librairies se standardisent de plus en plus dans l'objectif d'offrir la meilleure productivité et le meilleur confort aux développeuses et développeurs pour des applications performantes.