Coté labo – Application médicaments pour le laboratoire GSB

Mission 1/4 : Android Studio, API et machines virtuelles

Propriétés	Description
Intitulé long	Comprendre le fonctionnement d'une machine virtuelle connectée à l'environnement de développement Android Studio
Formation concernée	BTS Services Informatiques aux Organisations
Matière	Bases de la programmation
Présentation	L'objectif est de comprendre le déploiement d'un paquet *apk* dans un environnement de développement Android.
Notions	D1.1 – Analyse de la demande A1.1.1 Analyse du cahier des charges d'un service à produire D4.1 – Conception et réalisation d'une solution applicative A4.1.6 Gestion d'environnements de développement et de test **Savoir-faire** Utiliser un environnement de développement **Savoirs associés** Fonctions d'un environnement de développement Normes de développement
Pré-requis	Utilisation d'un environnement de développement standard (Visual Studio, Netbeans, Eclipse, etc.) Débogage Gestion de versions (git)
Outils	Android Studio, Android Virtual Devices
Mots-clés	Android Studio, AVD, API, console Logcat
Durée	2 heures
Auteur(es)	Fabrice Missonnier, relecture Hervé Le Guern et Yann Barrot
Version	v 1.0
Date de publication	mai 2019

Android Studio est un environnement de développement intégré (IDE) utilisable sous Windows, Linux et Mac OS pour programmer des applications dans le langage Java pour le système d'exploitation Android. Il est gratuit et fourni par Google en version stable depuis la fin de l'année 2014.

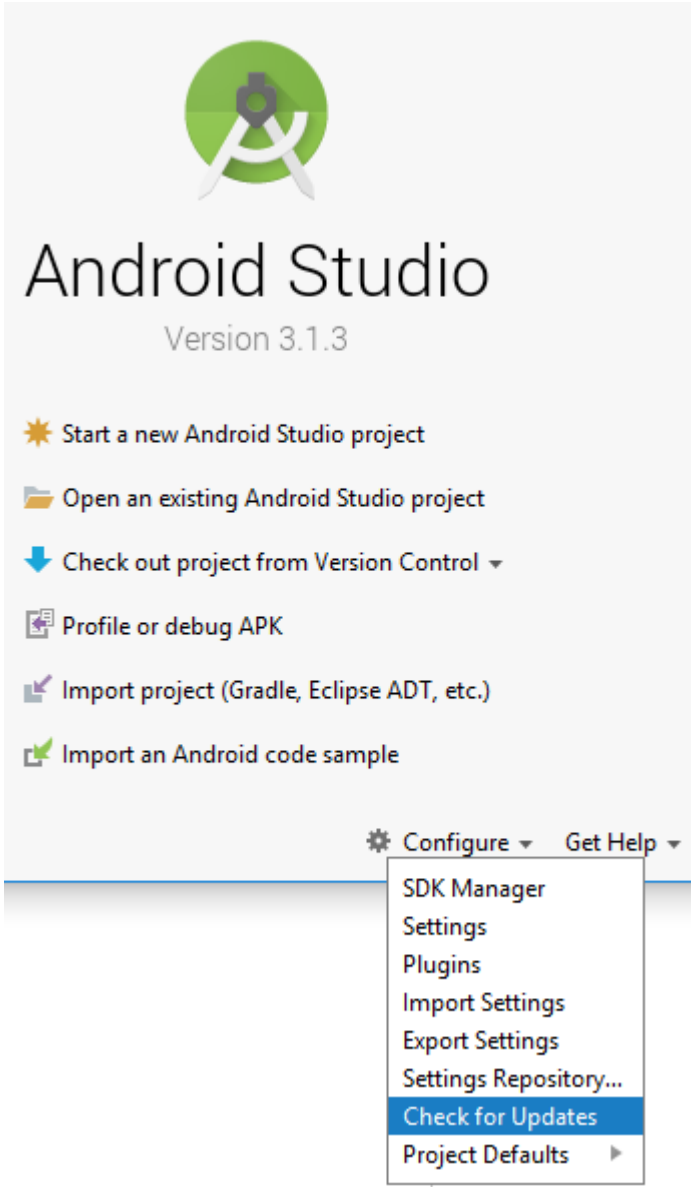
Cet IDE récent évolue sans cesse. Il est souvent demandé de télécharger de lourdes mises à jour.

Lorsque le projet est correctement configuré, développer en Java est strictement équivalent aux autres environnements que vous connaissez déjà. C'est l'exécution sur une machine cible Android qui diffère de l'usage standard. L'objectif de cette mission est de comprendre le fonctionnement de l'exécution d'un programme.

I. Installation de l'environnement de travail

Il faut une version d'Android Studio à jour pour pouvoir travailler sur cette mission.

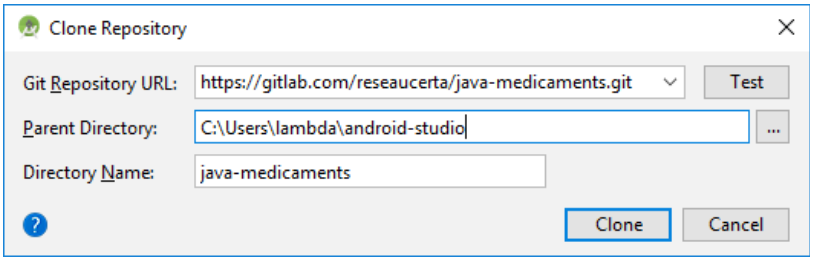
1- Lancer l'environnement de développement et faire les mises à jour (onglet *Check for Updates*).



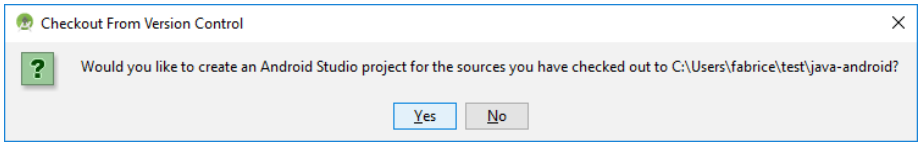
2- Récupérer le code source en local sur la machine de développement (sur la fenêtre de démarrage, *Check Out Project from Version Control*) à l'adresse

https://gitlab.com/reseaucerta/java-medicaments.git

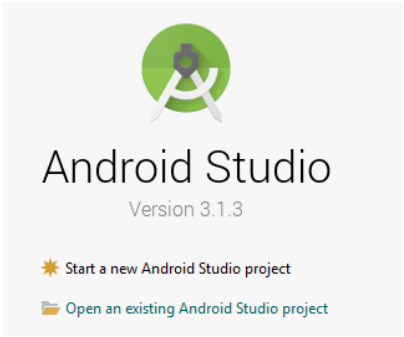
Attention : le nom du répertoire ne doit pas contenir d'espaces

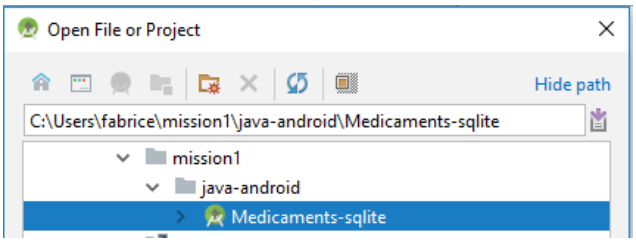


Le projet cloné étant déjà un projet Android Studio, il ne faut répondre *No* au message suivant :

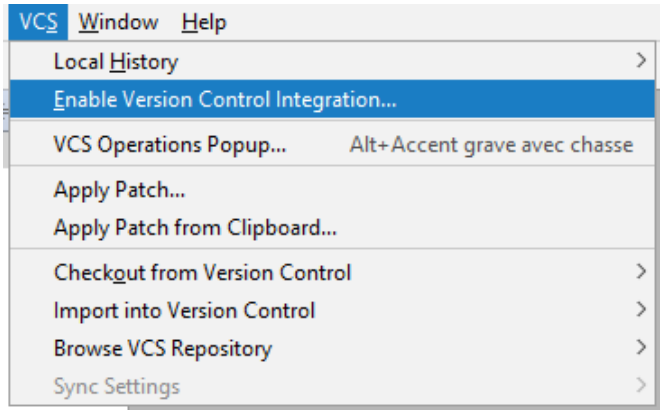


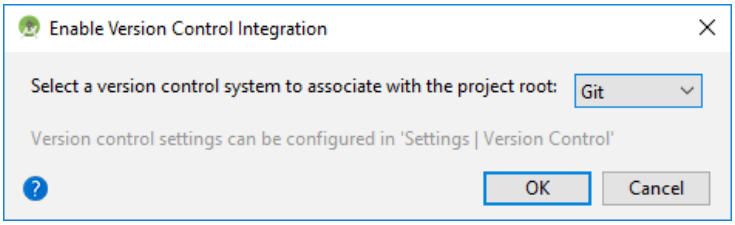
3- Cliquer sur *Open an existing Android Studio project* et ouvrir le projet cloné dans le répertoire (il apparait avec le logo . Des téléchargements sont nécessaires pour mettre à niveau les logiciels.





4- Le code source du projet n'est pas, par défaut, suivi par git. Il faut donc sélectionner ce gestionnaire de version (menu *VCS / Enable Version Control Integration*).


VCS
Window
Help
Local History
Enable Version Control Integration...
VCS Operations Popup...
Alt+Accent grave avec chasse
Apply Patch...
Apply Patch from Clipboard...
Checkout from Version Control
Import into Version Control
Browse VCS Repository
Sync Settings

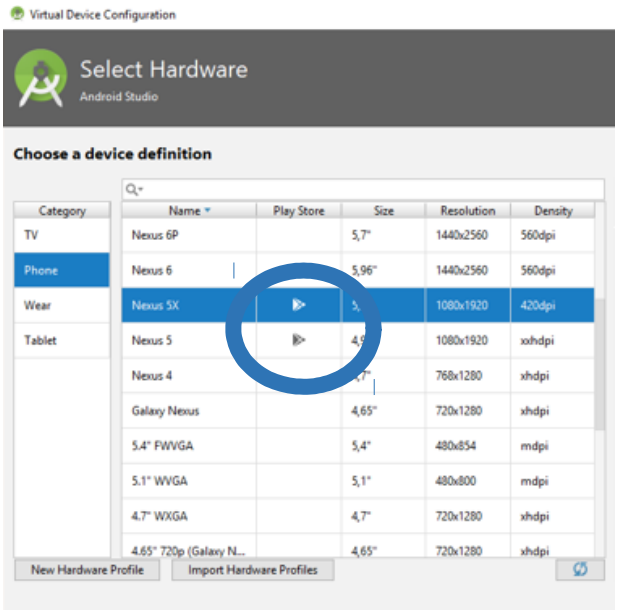

Enable Version Control Integration
Select a version control system to associate with the project root:
Git
Version control settings can be configured in 'Settings | Version Control'
OK
Cancel

II. Comment exécuter un programme sous Android Studio ?

L'exécution du code est plus complexe à mettre en œuvre que sur une interface de développement classique.

Puisqu'on programme pour des machines installées avec un système d'exploitation Android, il faut forcément lancer l'application sur un ordinateur Android distant (un émulateur ou un smartphone directement connecté en USB sur l'ordinateur de développement). Android Studio fournit l'utilitaire AVD, émulateur intégré qui gère des machines virtuelles.

5- Créer une machine virtuelle avec l'utilitaire AVD (*Tools*, *AVD Manager*, *Create Virtual Device*). L'application étant destinée aux smartphones, il est préférable, pour les tests, d'installer un téléphone. Cette machine doit intégrer le *Play Store* de Google :



Quelle version des API utiliser ?

Le système d'exploitation Android est, lui-aussi, en constante évolution. Lorsqu'on programme des applications qui lui sont destinées, il faut configurer le compilateur pour qu'il produise les fichiers binaires en fonction de la version d'Android installée sur le smartphone ou la tablette.

On retrouve cette configuration dans le script `build.gradle` du projet où l'on spécifie la version minimale des API (*Application Programming interface*). Les API Java sont des composants logiciels précompilés et fournis par Google. Dans notre projet, voici un extrait de `build.gradle` :

```
android {
    compileSdkVersion 27
    defaultConfig {
        applicationId "com.gsb.javamedicaments"
        minSdkVersion 14
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
```

Le lien entre le niveau des API et la version d'Android est spécifié à l'adresse
https://source.android.com/setup/start/build-numbers

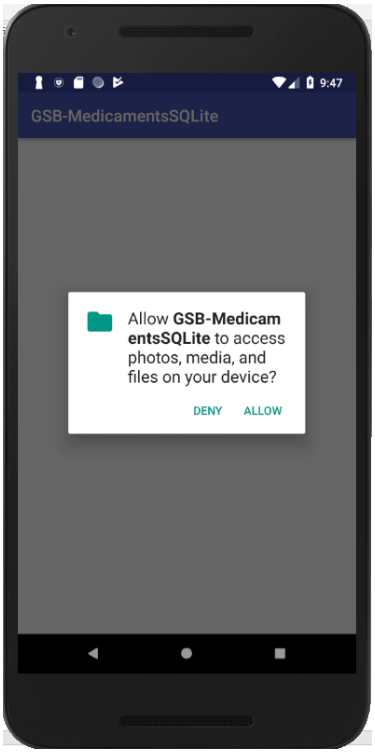
Code name	Version	API level
Oreo	8.1.0	API level 27
Oreo	8.0.0	API level 26
Nougat	7.1	API level 25
Nougat	7.0	API level 24
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21

Le choix pour ce projet a été de configurer la compilation avec les dernières API définies à ce jour (Android Oreo, version 8.1.0, API level 27). Les autres lignes de configuration sont décrites sur la documentation officielle
https://developer.android.com/guide/topics/manifest/uses-sdk-element

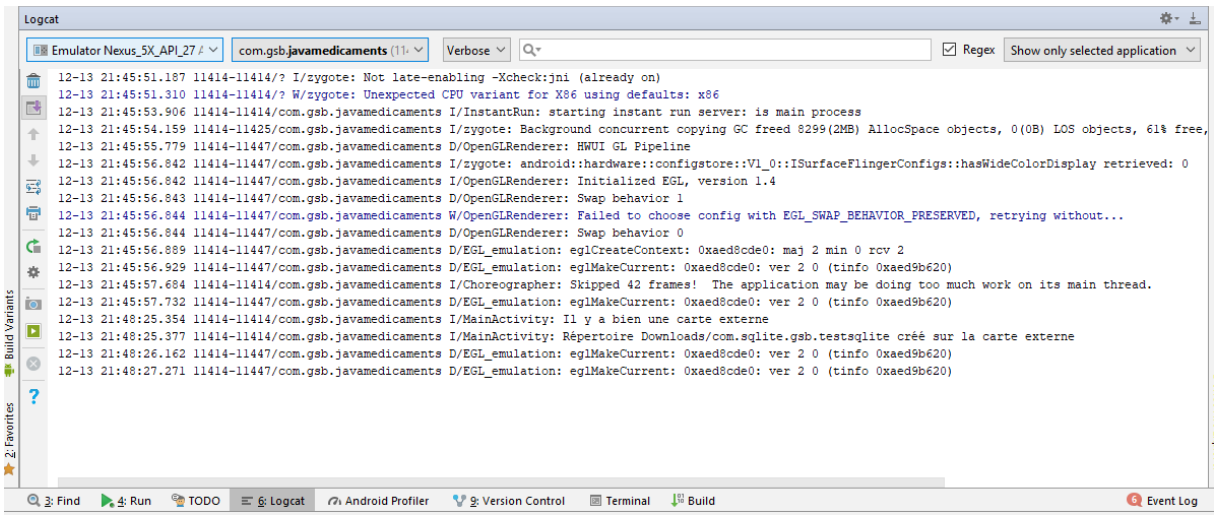
6- Avant de compiler le projet, synchroniser *Gradle* afin de télécharger les bonnes versions des API Android (*Menu Files*, *Sync Project WithGradle Files*).

Les principaux répertoires utilisés par une application Android

7- Compiler le code (Menu *Build*, *Makeproject*) et l'exécuter (Menu *Run*, puis *Run*). Après installation de l'*apk* sur le smartphone cible, accepter l'accès à la carte mémoire externe :



Lorsqu'on exécute le code sur la machine virtuelle, les évènements de débogage qui proviennent de la machine destination sont consignés dans la console *Logcat* dans la fenêtre en bas de l'IDE :



C'est dans cette console que l'on voit les piles d'erreurs générées par Java ainsi que les « traces » que l'on peut placer dans le code source. Pour éviter de récupérer les *logs* de toutes les applications qui tournent sur la machine Android, choisir en haut, dans le menu déroulant *Show only selected application*.

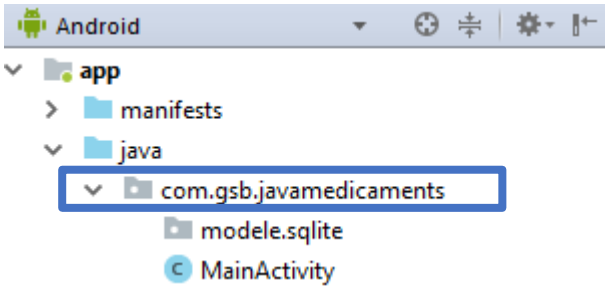
Ce projet, lorsqu'il est exécuté, est déployé sous forme d'*apk* sur le smartphone cible. Comme une application téléchargée sur le *Play Store*, le logiciel a plusieurs répertoires qui lui sont propres pour fonctionner.

Le premier est placé sur la mémoire interne du smartphone (dans le répertoire `/data/data/`). Le second correspond à la mémoire interne partagée, dont le système de fichier est monté dans le répertoire `/storage`.

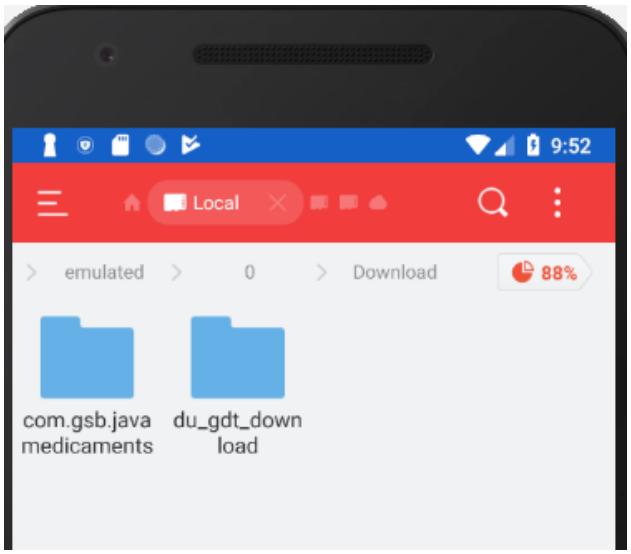
Deux répertoires de travail pour l'application en cours d'exécution sont automatiquement créés lors du déploiement de *l'apk*

- sur la mémoire interne (non accessible par un utilisateur), `/data/user/0/com.gsb.javamedicaments/`, lien symbolique vers `/data/data/com.gsb.javamedicaments/`
- sur la mémoire interne, dans le cas d'une machine virtuelle (répertoire partagé, donc visible par tous les utilisateurs) `/storage/emulated/0/Download/com.gsb.javamedicaments/`

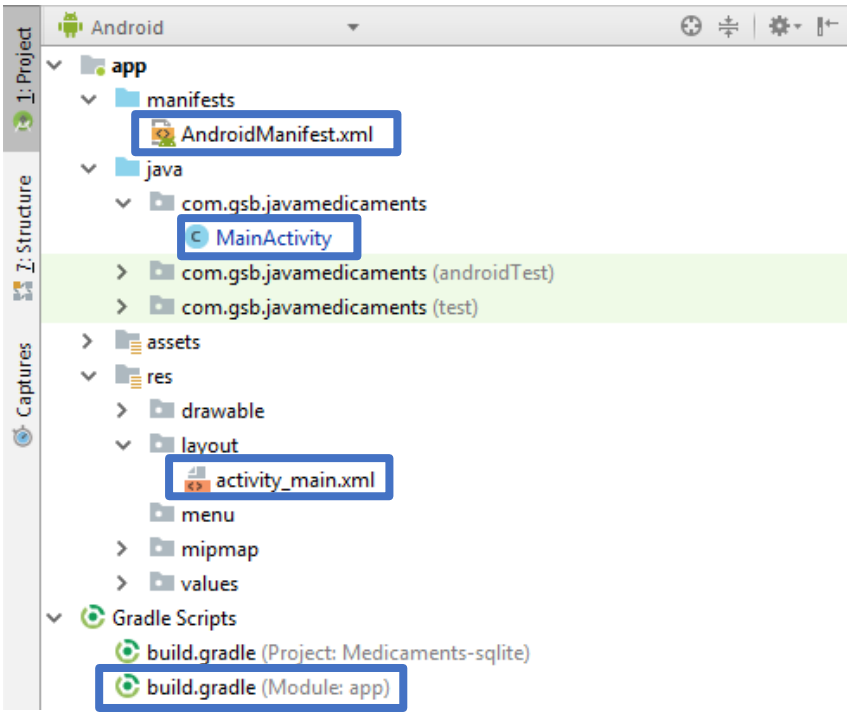
`com.gsb.javamedicaments` représente le nom du package sous Android Studio :



8- Pour voir si les fichiers ont bien été copiés, installer un explorateur de fichiers sur le smartphone (*ES explorer* dans Play Store par exemple). Par défaut, l'application crée les fichiers dans le répertoire download de la carte externe du téléphone.



Sous Android Studio, ouvrir les fichiers `AndroidManifest.xml`, `MainActivity.java`, `activity_main.xml` et `build.gradle` :



9- Quelle est l'utilité de ces différents fichiers ?

Dans la fenêtre *logcat*, la ligne `I/MainActivity: Il y a bien une carte externe` s'affiche.

10- Retrouver dans le code source l'instruction qui permet d'écrire cette ligne.