Séance Android n°1/2 : Installation de l'environnement, premier projet et déploiement d'application

Android, Quésaco ?

Android est un système d'exploitation pour support mobile, soit *OS Mobile* (*Operating System Mobile*). Il a été développé par une start-up du même nom, racheté par Google en août 2005. Il est basé sur un noyaux linux allégé, ceci afin de répondre aux contraintes liées aux appareils mobiles (batterie, capacité mémoire, capacité traitement,...). La nature open source du *framework* Android et la facilité à reprendre son code source a permis son succès sur le marché des smartphones et autres supports tactiles.
Un environnement de développement Android est généralement constitué du *SDK* (*System Development Kit*) Android intégré à un *IDE* (*Integrated Development Environment*) Eclipse ou NetBean.
Concrètement, le cours de programmation Android va vous permettre de développer des applications pour smartphone Android (Samsung Galaxy, Htc, Sony, etc.).

Cette première séance est dédié à l'installation de l'environnement de développement Android (TD 1) ainsi qu'à la création de projets et au déploiement d'applications (TP 1).

TD 1 : Installation de l'environnement de développement

TP 1 : Création de projets et déploiement d'applications

TD 1 : Environnement de développement

Temps estimés : **30min**.

Dificulté : ∗

Pour installer votre environnement de développement Android, vous pouvez soit télécharger l'environnement complet à partir du site officiel [1], soit télécharger seulement le *SDK* Android en utilisant un *IDE* existant comme Eclipse ou Netbeans. Ceci dit, le plugin Android d'Eclipse n'est plus maintenu, il est donc préférable d'utiliser Android Studio.
Jusqu'à présent, le langage Java était utilisé pour programmer sous Android. Cependant, *Google* a officialisé le support du langage Kotlin pour le développement d'application mobile Android, le 17 mai 2017, à la conférence G I/O [3]. En outre, il est possible d'utiliser du C ou C++ si cela est nécessaire pour l'application (exemple : application utilisant de manière intensive le CPU).
Ce TD détaille comment vérifier la bonne installation de l'environnement et comment faire les dernières mises à jour.

A. Vérification et mise(s) à jour

Cette partie décrit comment vérifier la configuration du *SDK* pour Android Studio et comment télécharger les dernières versions d'Android.

1] Vérification de la configuration du SDK

1. Dans Android Studio, allez dans Windows>Preferences (sur mac Android Studio>Preferences ou ADT>Preferences).
2. Choisissez Android dans le menu de gauche.
3. Vérifiez l'emplacement du dossier SDK Location, il doit correspondre à l'endroit où vous avez placé le SDK (exemple : /Users/macha/Library/Android/sdk).

2] Mise(s) à jour des plateformes et codes exemples

1. Ouvrez une fenêtre Android SDK Manager en cliquant sur l'icône avec une flèche blanche vers le bas (cf. Figure 1), situé dans la barre d'outils Android Studio.
2. Allez dans la partie des téléchargements.
3. Sélectionnez, les packages et codes exemple de la dernière version d'Android (*Samples for SDK, Google APIs, Source for Android SDK* de la version 4.4).
4. Lancez les téléchargements en cliquant sur Install packages.

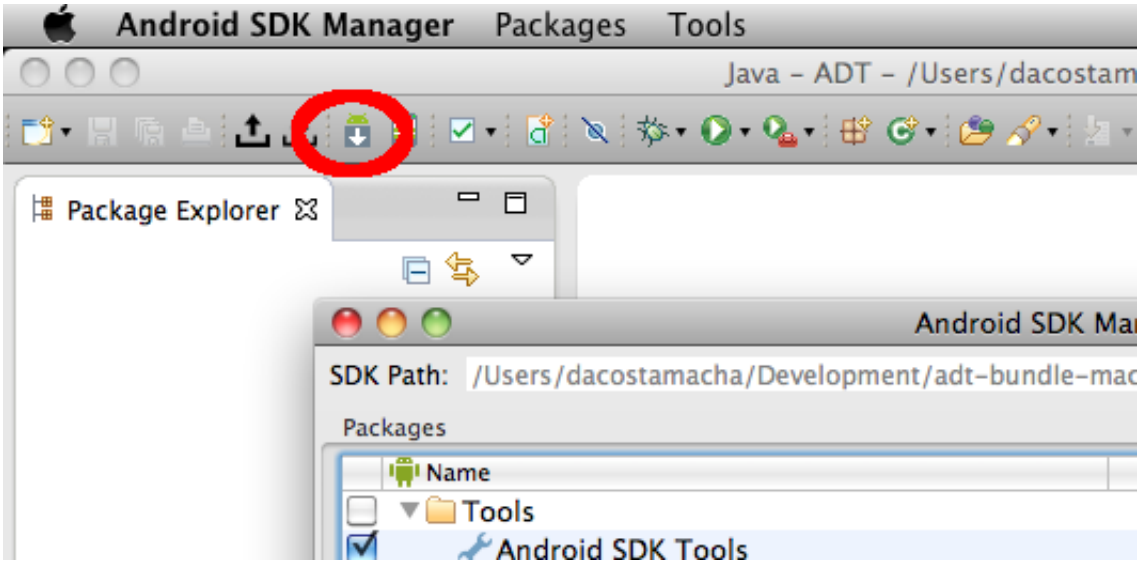

FIGURE 1 – Android SDK Manager

TP 1 : Création de projets et déploiement d'applications

Temps : **1h**

Dificulté : **

Le TP qui suit permet de faire ses premiers pas dans le développement d'application mobile, au travers d'une application de type "Helloworld". De plus, il vous est proposé d'explorer les applications exemples du SDK (*Samples for SDK*, package téléchargé précédemment).

A. Création d'applications

1] Application HelloWord

Ici, il est indiqué comment créer un premier projet avec l'utilitaire **Android Studio**. Cependant, il est également possible de créer un projet en ligne de commande.

1. Créez un nouveau Projet d'Application Android (New>Android Application Project).
2. Laissez vous guider par l'utilitaire, Voici un exemple de configuration d'un projet :
   'Project name' *MyFirstApp*,
   'Target SDK' *Android 4.4*,
   'Application name' *MyFirstApp*,
   'Package name' *com.example.myfirstapp*,
   'Created Activity' *MyFirstApp*
   Puis sélectionnez Next.
3. Personnalisez brièvement le logo de l'application.
4. Changez le nom proposé par défaut d'une **Activity**, en MyMainActivity.java. Vous avez la possibilité de choisir un modèle d'activité, choisissez **Blank Activity** pour ce premier projet.

Remarque : si vous ne comprenez pas la subtilité de tous ces éléments générés lors de la création du projet, ne vous inquiétez pas. Il vous faudra explorer votre projet afin de vous familiariser avec un package, une *Activity*, un fichier XML, etc.

2] Application existante

1. Créez un projet Android à partir des exemples proposés par **Android Studio**.

2. Déployez votre projet (voir partie B).
3. Essayez de repérez des éléments graphiques et faites le lien avec le code source.

B. Déploiement d'une application

1] Sur émulateur

1. Ouvrez l'Android Virtual Device Manager (*AVD Manager*) en cliquant sur l'icône représentant un smartphone (cf. Figure 2).
2. Faite New, pour créer un nouveau émulateur.
3. Nommez le, par exemple, *MyNexus*.
4. Choisissez un *Nexus* pour 'Device'.
5. Vérifiez les autres champs, et cliquez ok.
6. Dans la fenêtre AVD Manager, sélectionnez l'avd créé (*MyNexus*), faite Start puis Launch.

Remarque : le lancement de l'émulateur prend quelques minutes. Vous pouvez le laisser ouvert durant toute la séance et le fermer seulement quand vous avez terminé de développer. Le fermer après chaque test d'application reviendrait à éteindre votre téléphone après l'utilisation d'une application...

2] Sur support

Le déploiement d'une application Android sur un support consiste à installer un fichier .APK (acronyme de *Android PacKage*) sur un support donné. Un fichier .APK pour un téléphone Android est ce qu'est un .EXE pour un PC, un .DMG pour un Mac OS X, un .APP pour un smartphone iOS, etc [6].
Habituellement, l'installation d'une application Android se fait via l'*Android Market* ou *Play Store*. Cependant, pour des applications Bêta ou *underground* il n'est pas possible de passer par la voie normale où l'installation de l'application se fait de manière invisible pour l'utilisateur. Ce qui suit décrit comment installer manuellement une application, cela se fait en 2 étapes :

* autoriser les sources inconnus sur le support (attention à n'installer que des applications de confiance)
* télécharger l'application sur le support via le *cloud* ou via câble USB

Il est également possible d'exécuter l'application d'un projet **Android Studio** directement sur le support, il suffit d'autoriser le *debuggage* USB sur ce dernier et d'installer les *drivers* sur le PC.

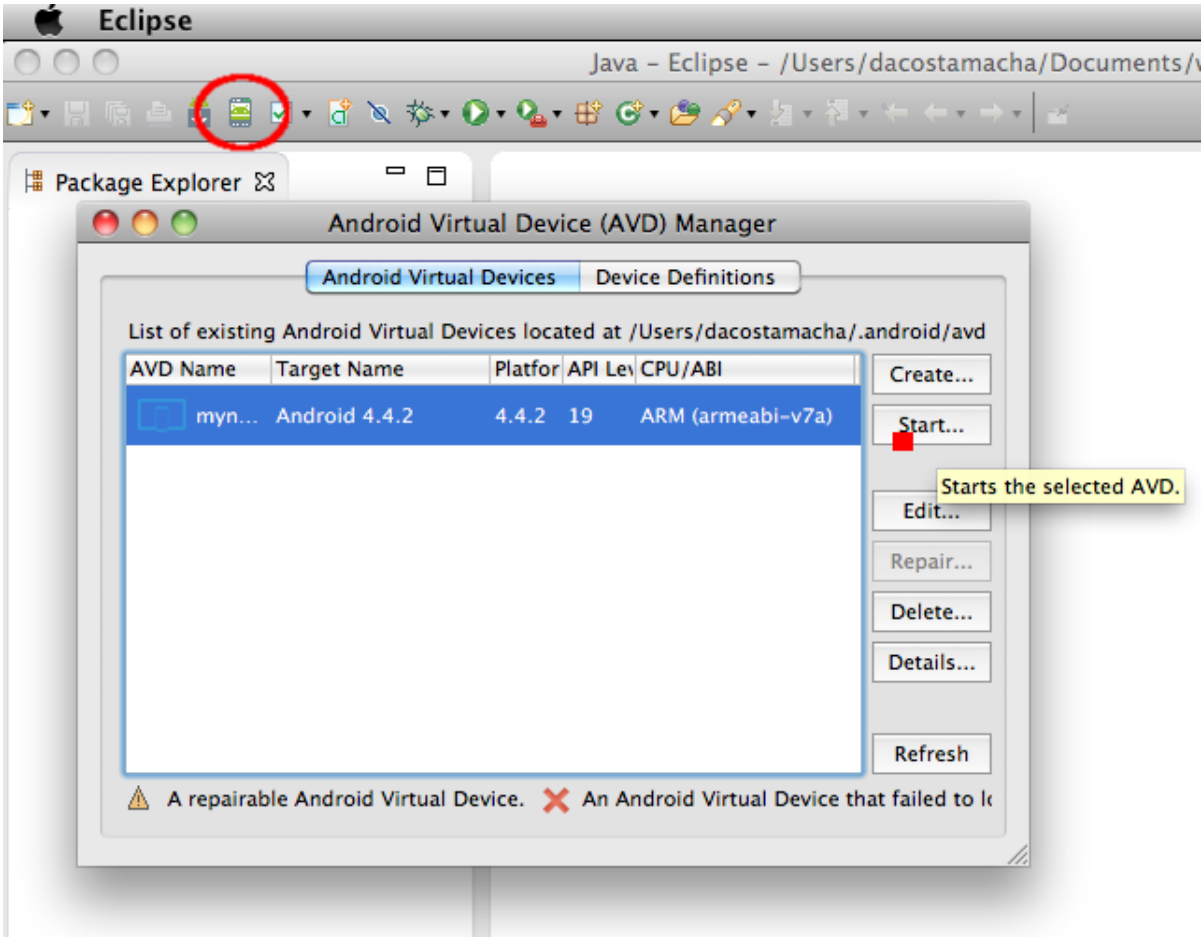

FIGURE 2 – Android Virtual Device Manager

1. Sur le smartphone, allez dans Menu>Paramètres>Application et cochez Sources inconnues.
2. Sur le téléphone, allez dans Menu>Paramètres>Application>Développement et cochez Débogage USB.
3. Sur votre ordinateur, identifiez l'emplacement du .APK : /Development/MyApps/MyFirstApp/bin
4. Branchez le téléphone à votre ordinateur via le cable USB.
5. A présent, vous pouvez directement exécuter votre projet via Android Studio, sur votre téléphone.
6. Ou bien, vous pouvez installer manuellement, une application tiers dont vous détenez le .APK : Sur le téléphone, en sélectionnant dans la barre de notification Connecté avec un câble USB, puis Activer le périphérique de stockage. Enfin déplacer le .APK depuis l'ordinateur.

C. Exploration de l'arborescence de votre projet

1. Quel fichier modifier pour changer le texte affiché à l'écran ? Modifiez pour afficher « Hello Kotlin ! ».
2. Quel fichier modifier pour ajouter des éléments graphiques à l'écran ? Ajoutez du texte.
3. Dans le fichier des ressources relatif aux images, res/drawable, ajoutez une image de votre choix.
4. Ajoutez l'image à côté du texte.
5. Ajoutez un bouton qui change le texte initial en un autre texte.

Remarques :

les fichiers XMLs peuvent être édités manuellement ou bien avec l'éditeur graphique.

les dossiers drawable-ldpi/mdpi | hdpi contiennent les images adaptés aux 3 types d'écrans (hdpi : images pour smartphone, xhdpi : images pour tablette, mdpi : images pour petit écran, lecteur mp3 [2]).

Note : Un fichier de projet R.java est un index contenant tous les identifiants des ressources du projet. Cette classe est utilisée dans le code source comme référence vers les ressources incluses dans le projet.

L'environnement de développement Android comprends un outil permettant à un développeur graphiste, de designer un écran à partir de glisser/déposer d'éléments graphiques classés dans une bibliothèque vers une maquette d'écran. L'intérêt de la manipulation n'est pas seulement pour la conception mais aussi pour le développement. En effet, l'outil génère du code xml.

D. Expérimentation sur le cycle de vie d'une *Activity*

1. Faites hériter votre *Activity* principale, avec la classe *AnkoLogger* de la bibliothèque *Anko*.
2. Affichez un premier message d'information depuis la méthode *onCreate()*.
3. Surchargez, les méthodes du cycle de vie d'une *Activity* afin qu'elles affichent des messages d'informations.
4. Analysez les messages de votre application lors de son exécution via Android Monitor.
5. Décrivez un scénario utilisateur permettant de mettre en évidence un maximum de méthode du cycle de vie. Pour chaque étape du scénario ainsi inventé, mentioner la méthode correspondante.

E. Expérimentation sur la persistence des données

1. Ajoutez un composant graphique de type *NumberPicker*, configurez-le avec une valeur minimum et maximum.
2. Testez votre application sur émulateur ou téléphone, modifiez le chiffre puis faites passer votre téléphone du mode portrait au mode paysage. Que se passe t-il ?
3. Implémentez la persistence de la donnée sélectionnée dans le *NumberPicker* dans le cas de figure évoqué dans le point précédent. Notamment à l'aide des méthodes *onSaveInstanteState(Bundle outState)* et *onRestoreInstanteState(Bundle inState)*.

Références du cours ABC d'Android

Le livre de Nazim BENBOURAHLA publié par les Éditions ENI : [5]

Quelques liens du site de référence developer.android.com : [4]

A. Plateforme Android

— Présentation
— Pourquoi choisir Android ?
— Historique
— Chemin d'une Application

B. Environnement de Développement

— Environnement Java ou C/C++
— Android Studio
— SDK Android
— Déploiement

C. Principes de Programmation

— Architecture Android
— Composantes Android
— Cycle de vie d'une Activity
— Manifeste

D. Premier Projet

— Création
— Arborescence
— Fichiers clés
— Déploiement

Webographie

[1] Android. Site officiel android. http://developer.android.com.

[2] Android. Supporting multiple screens. http://developer.android.com/guide/practices/screens_support.html.

[3] Macha da Costa. Introduction à kotlin. https://www.chillcoding.com/blog/2017/07/11/android-kotlin-introduction/.

[4] developer.android. Application fundamentals. https://developer.android.com/guide/components/fundamentals.html.

[5] Editions ENI. Android 5 les fondamentaux du développement d'applications java. www.editions-eni.fr/.

[6] Guiding Tech. What are apk files and how to install them. http://www.guidingtech.com/10352/what-are-android-apk-files-how-to-install-them/, 2010.