Séance Android 1 / 2

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 ainsi qu'à la création d'un premier projet et au déploiement de l'application relative à ce dernier (TD 1). Puis, l'exploration de l'univers Android est proposé à travers des exercices d'expérimentation (TP 1).

TD 1 : Installation de l'environnement de développement

TP 1 : Exploration et expérimentation de l'univers Android

TD 1 : Installation de l'environnement de développement

Temps estimés : **2h**.
Dificulté : ∗

L'installation de l'environnement de développement comprend également la création d'un premier projet et le déploiement d'application.

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. Vous pouvez aussi suivre cet article afin de faire l'installation [3].
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 [4]. 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 (AS) et comment télécharger les dernières versions d'Android.

1] Vérification de la configuration du SDK

1. Dans Android Studio, allez dans File>Settings (sur mac Android Studio>Preferences ou ADT>Preferences).
2. Choisissez Android SDK dans le menu de gauche, dans Appearance & Behaviour > System Settings.
3. Vérifiez l'emplacement du dossier SDK Location, il doit correspondre à l'endroit où AS a 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 vers le bas (cf. Figure 1), situé dans la barre d'outils Android Studio.
2. Allez dans la partie des téléchargements.
3. Observez les détails de téléchargements en cochant Show Package details

Les exercices suivants permettent de faire vos premiers pas dans le développement d'application mobile, au travers d'une application de type "Hello World". Ensuite, le déploiement de cette application est détaillé selon deux pratiques. La première se fait sur l'émulateur ; la seconde, à privilégier se fait directement sur téléphone.

B. 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 (Start a new Android Studio project).
2. Laissez vous guider par l'utilitaire, Voici un exemple de configuration d'un projet :
   'Application name' *MyFirstApp*,
   'Company Domain (Package name)' *macha.fr*,
   Cochez Include Kotlin support
   Puis Next
   Cochez 'Phone and Tablet' avec en version. minimum supportée : *API 15 : Android 4.0*,
   Puis sélectionnez Next, Finish.
3. Vous avez la possibilité de choisir un modèle d'activité, choisissez Empty Activity pour ce premier projet.

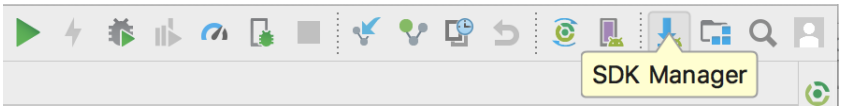

FIGURE 1 – Barre de navigation Android Studio avec l'icône 'Android SDK Manager'

4. Remarquez le nom proposé par défaut pour une **Activity** : **MainActivity**, c'est l'une des conventions de nommage adopté par les développeurs Android.

Remarque : si vous ne comprenez pas la subtilité de tous les 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.

C. 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 Create Virtual Device, pour créer un nouvel émulateur.
3. Choisissez un modèle de téléphone en fonction de votre résolution d'ordinateur, par exemple le *Nexus*.
4. Sélectionnez un System Image, gardez celui recommandé, par exemple *API 27*.
5. Nommez le, par exemple, *MyNexus*.
6. Vérifiez les autres champs, et cliquez ok.
7. Dans la fenêtre AVD Manager, sélectionnez l'avd créé (*MyNexus*), pour le lancer cliquez sur l'icône *play*.

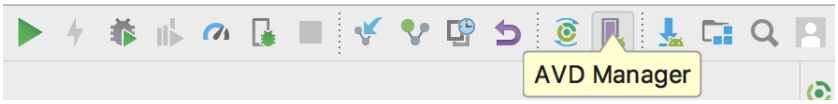

FIGURE 2 – Barre de navigation d'Android Studio avec l'icône 'Android Virtual Device Manager'

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

[8].
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.

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 : /MyFirstApp/app/build/outputs/apk/debug
4. Branchez le téléphone à votre ordinateur via le câble 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.

TP 1 : Exploration et expérimentation de l'univers Android

Ici, il vous est propos d' :
— explorer l'arborescence d'un projet
— expérimenter le cycle de vie d'une *Activity*
— analyser une application existante

A. Exploration de l'arborescence d'un projet

Temps : **1h**
Dificulté : ∗

1. Observez le contenu du dossier app/ : Manifest, java, res, graddle
2. Remarquez le lien avec l'interface graphique, dans le fichiers **MainActivity.kt**.
3. Quel fichier modifier pour changer le texte affiché l'écran ? Modifiez pour afficher « Hello Kotlin !».
4. Remplacez l'agencement principale (situ la racine du fichier activity_main.xml) ConstraintLayout par un LinearLayout.
5. Quel fichier modifier pour ajouter des éléments graphiques l'écran ? Ajoutez du texte.
6. Dans le fichier des ressources relatif aux images, res/drawable, ajoutez une image de votre choix.
7. Ajoutez l'image en dessous du texte.
8. Personnalisez brièvement le logo, les couleurs de l'application.

Remarques :

les fichiers XMLs placés dans le dossier layout/, peuvent être dits manuellement ou bien avec l'éditeur graphique.

communment ces fichiers XMLs, contiennent les éléments graphiques d'un cran

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 cran, lecteur mp3 [2]).

Note : Un fichier de projet R.java est un index contenant tous les identifiants des ressources du projet. Il est automatiquement généré par **Android Studio**. Cette classe est utilisé dans le code source comme réfrence vers les ressources incluses dans le projet.

L'environnement de développement Android comprends un outil permettant un développeur graphiste, de designer un cran 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 généré du code xml.

B. Expérimentation du cycle de vie d'une *Activity*

Temps : **1h**
Dificulté : $**$

1. Faites hriter votre **Activity** principale, avec la classe **AnkoLogger** de la bibliothque *Anko*.
2. Affichez un premier message d'information depuis la mthode *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 la console Logcat.
5. Décrivez un scénario utilisateur permettant de mettre en évidence un maximum de méthode du cycle de vie. Pour chaque étape du scénario, mentionner la méthode correspondante.

Remarque : Aidez-vous de la diapositive sur le cycle de vie d'une **Activity**, dans la présentation "1-ABCdAndroid.pdf". Aussi la partie A de la présentation "3-OutilsduDéveloppeur.pdf" est nécessaire, associe avec la partie sur la bibliothèque *Anko* de la prsentation sur "2-Kotlin.pdf".

C. Analyse d'application existante

Temps : **1h**
Dificulté : $**$

1. Créez un projet Android partir des exemples proposs par **Android Studio** : Import an Android code sample depuis la fenêtre d'accueil d'**Android Studio**.
2. Explorez les projets existants puis choisissez en un.
3. Dployez votre projet.
4. Essayez de repérez des éléments graphiques et faites le lien avec le code source.
5. Réalisez des modifications (nom des menus, couleurs, etc).

D. Expérimentation sur la persistance des données

Temps : **1h**
Dificulté : ∗ ∗ ∗

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

E. Gestion du clique sur un bouton

Temps : **1h**
Dificulté : ∗ ∗ ∗

1. Ajoutez une image vectorielle dans l'cran principale.
2. Affichez un message rapide lorsque l'utilisateur clique sur l'image (cf. "Afficher un message rapide" dans cet article [5]).

Remarque : Pour cette exercice, la partie A de la présentation "5-InterfaceUtilisateurInteractive.pdf" est nécessaire.

F. Passage de données entre écran

Temps : **1h**
Dificulté : ∗ ∗ ∗

1. Placez deux champs d'édition, de type nombre, ainsi qu'un bouton dans l'écran principale.
2. Ajouter un second écran, il est affiché lors de l'interaction avec le bouton et affiche l'addition des deux champs d'édition.

Remarque : Pour cette exercice, la partie B de la présentation "5-InterfaceUtilisateurInteractive.pdf" est nécessaire.

Références du cours ABC d'Android

Cette formation s'adresse aux développeurs souhaitant se lancer dans le développement d'applications mobiles en Kotlin sur Android. Une bonne connaissance de la programmation orientée objet est nécessaire pour en tirer le meilleur profit. Des connaissances en programmation fonctionnelle est un plus.

L'ABC d'Android a pour objectif de présenter les fondamentaux du développement d'application mobile Android.

Dans un premier temps, vous découvrirez pourquoi choisir Android, son historique et les particularités liées à sa nature open source.

Dans un second temps, vous appréhenderez comment installer et configurer l'environnement de développement Android Studio.

Puis, les principes de programmation sont approfondis via l'appréhension de l'architecture d'un projet et des principales composantes d'Android : 'Activity' et 'Fragment'.

Enfin, le déploiement d'un premier projet est envisagé. Ce dernier permet d'étudier les briques essentielles à la création d'applications, à savoir l'arborescence d'un projet Android et ces fichiers clés.

A l'issue de ce thème, vous serez capable de créer une première application simple.

Le livre de Nazim BENBOURAHLA publié par les Éditions ENI : [7]

Quelques liens du site de référence developer.android.com : [6]

A. Plateforme Android

— Système ouvert
— 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. Installer un environnement de développement android. https://www.chillcoding.com/blog/2016/08/03/android-studio-installation/.

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

[5] Macha da Costa. Utiliser la bibliothèque anko dans un projet android. https://www.chillcoding.com/blog/2017/10/09/utiliser-anko-kotlin-android/.

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

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

[8] 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.