TP1 – Mise en place de l'environnement de développement

1) Clonage du dépôt

Pour cloner le dépôt, il faut tout d'abord se placer dans le répertoire où nous voulons placer les sources.

Il faut ensuite taper la commande :

git clone https://github.com/gxfab/LO52_A2020

Cette dernière permettra d'initialiser un répertoire local contenant les sources du dépôt distant.

2) Modification du fichier de configuration

Pour mettre en place l'utilisateur qui interagira avec Git, il faut taper les deux commandes suivantes :

git config --global user.name = "Nicolas Lépy"

git config --global user.email = "lepynicolas@orange.fr"

3) Création de la branche

Le clonage du dépôt a créé un sous-dossier contenant les sources. Il faut tout d'abord se déplacer dans ce dossier.

Ensuite, il faut réaliser la commande suivante :

git branch SidawyCem_LepyNicolas

Cette dernière va permettre de créer une nouvelle branche avec le nom donné en paramètre. La branche est pour l'instant seulement locale.

Enfin, il faut sélectionner cette nouvelle branche en tant que branche active. Pour cela, il faut taper la commande git

git checkout SidawyCem_LepyNicolas

On peut ensuite réaliser un *git push* qui va envoyer et créer la branche sur le dépôt distant.

4) Initialisation d'Android Studio

La configuration du proxy et l'installation du SDK pour Android Studio furent nécessaire.

5) Création des deux activités

L'affichage des éléments dans une activité est essentiellement géré dans un fichier .xml associé qui décrit quels sont les éléments et comment ils sont placés.

Pour ajouter un bouton, on peut utiliser l'interface graphique que propose Android Studio, qui fonctionne par drag-and-drop, puis édition des propriétés, tout comme directement éditer le fichier xml. Une balise *Button* est créée et contient les paramètres du bouton, notamment ses propriétés d'affichage (s'il est étiré, le plus condensé possible), mais aussi son identifiant et son texte.

On peut glisser le bouton dans une balise *LinearLayout* qui permet de ranger automatiquement les éléments de l'interface de manière horizontale ou verticale.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Open"
        android:onClick="btnOpen"
        />
</LinearLayout>
```

Pour appeler une fonction lorsque l'utilisateur clique sur un bouton, il faut ajouter la propriété *android:onClick* dans le bouton dans laquelle on va spécifier le nom de la fonction à appeler, qui sera présente dans le fichier java associé à l'activité.

Pour ouvrir une deuxième activité à partir de la première, il faut aller dans le fichier java associé à l'activité. On peut créer la fonction que l'on a défini dans l'action du clic sur le bouton, avec la signature : *public void btnOpen(View view)*.

A l'intérieur, nous allons créer un objet *Intent*, utilisé pour communiquer entre plusieurs briques d'une application. Dans cet objet Intent, nous spécifions quelle activité nous souhaitons ouvrir. La fonction *startActivity* est enfin appelée en passant notre *Intent* en paramètre, qui va se charger de créer et d'afficher à l'écran la deuxième activité. La fonction est donc la suivante :

```
public void btnOpen(View view) {
        Intent intent = new Intent(this, HelloWorldActivity.class);
        startActivity(intent);
}
```

6) Publication du projet

Afin de publier le projet, il faut commencer par utiliser la commande *git add* pour ajouter les nouveaux fichiers du projet à notre branche locale dans git.

Une fois les fichiers ajoutés, on crée un commit avec la commande *git commit -a "Message de commit"* qui va contenir tous les fichiers qui ont été modifiés.

A noter que Android Studio a créé un fichier .gitignore qui permet de spécifier les dossiers, des extensions… à ne pas inclure dans le commit. Tous les fichiers générés à la compilation par exemple ne sont pas inclus dans les fichiers à envoyer, et n'encombrent pas le dépôt distant.

Enfin, on réalisera un *git push* pour publier notre commit dans le dépôt distant.