Préparation pour Android

Renommer votre projet

Nous allons commencer par donner un nom à notre `package` avant qu'il ne soit publié sur le `Play Store`.

Pour ce faire, nous allons aller dans le fichier `AndroidManifest.xml` dans le dossier `android/app/src/main` :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="votre.nom.de.paquet.par.défaut">
…
```

Copiez le nom du `package` et faites une recherche globale dans le projet puis un `replace`.

Remplacez tous les anciens noms par le nouveau nom choisi dans tout le projet.

Cela doit remplacer le nom dans les fichiers suivants `build.gradle`, `AndroidManifest.xml`, `MainActivity.java` et `AndroidManifest.xm` (dans le dossier `profile`).

Modification du fichier `AndroidManifest.xml`

Dans le dossier `android/app/src/main` nous allons ajouter une permission supplémentaire pour que l'application puisse accéder à `Internet` :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.projet_dyma_end">
  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCAT
ION" />
…
```

Toujours dans le même fichier nous allons modifier la valeur `android:label` qui est très importante car **c'est le nom de l'application que l'utilisateur verra** :

```
…
<application
    android:name="io.flutter.app.FlutterApplication"
```

```
    android:label="Dyma Trip"
…
```

Signer son application

Pour publier sur le `Play Store` il faut signer votre application avec une clé.

Il faut commencer par générer la clé avec cette commande pour `Mac OS` et `Linux` :

```
keytool -genkey -v -keystore ~/key.jks -keyalg RSA -keysize 2048 -val
idity 10000 -alias key
```

Ou cette commande sur `Windows` :

```
keytool -genkey -v -keystore c:/Users/USER_NAME/key.jks -storetype JK
S -keyalg RSA -keysize 2048 -validity 10000 -alias key
```

Il faut ensuite rentrez l'ensemble des informations demandées par le terminal.

Rentrez des mots de passe beaucoup plus sécurisés que dans la vidéo, d'au moins 8 caractères avec majuscule, chiffre et caractère spécial. Il vous en sera demandé deux : un pour le `store` et un pour la clé privée.

Vous aurez ensuite un fichier `key.jks` qui contient une clé privée. **Il faut absolument qu'elle reste secrète, ne la publiez pas dans `GitHub` par exemple**.

Il faut ensuite créer un fichier `key.properties` dans le dossier `android` :

```
storePassword=<VOTRE MOT DE PASSE 1>
keyPassword=<VOTRE MOT DE PASSE 2>
keyAlias=key
storeFile=<emplacement de la clé>
```

Vous devez renseigner dans `storeFile` le chemin vers le fichier `key.jks`.

Le fichier `key.properties` doit également rester privé : mettez le dans votre fichier `.gitignore` pour ne pas le suivre avec `Git`.

Ajoutez sous `Android Related` :

```
**/android/key.properties
```

Modification de `android/app/build.gradle`

Il faut maintenant modifier le fichier `android/app/build.gradle` pour pouvoir utiliser la clé pour le `build` de votre application `Android`.

Il faut donc remplacer :

```
android {
```

Par :

```
def keystoreProperties = new Properties()
def keystorePropertiesFile = rootProject.file('key.properties')
  if (keystorePropertiesFile.exists()) {
    keystoreProperties.load(new FileInputStream(keystorePropertiesFil
e))
  }

android {
```

Il faut ensuite remplacer :

```
buildTypes {
  release {
    // TODO: Add your own signing config for the release build.
    // Signing with the debug keys for now,
    // so `flutter run --release` works.
    signingConfig signingConfigs.debug
  }
}
```

Par :

```
signingConfigs {
  release {
    keyAlias keystoreProperties['keyAlias']
    keyPassword keystoreProperties['keyPassword']
    storeFile file(keystoreProperties['storeFile'])
    storePassword keystoreProperties['storePassword']
  }
}
buildTypes {
```

```
    release {
      signingConfig signingConfigs.release
    }
  }
```