Annexe 2 – Mise en place de Firebase

Pour mettre en œuvre *Firebase Cloud Message* vous devez avoir un compte *Google* et vous devrez tester votre solution sur un smartphone ou un émulateur possédant le *Google Play Store*. Nous allons, dans un premier temps, créer le projet *Firebase* et l'ajouter, ainsi que les dépendances nécessaires dans notre application *Android*.

1. La première étape consiste à créer un projet sur la console *Firebase* : https://console.firebase.google.com/. Vous saisirez un nom de projet, par exemple « DMA Labo1 », vous pouvez désactiver *Google Analytics* nous n'allons pas en avoir besoin.

2. Après quelques instants votre projet va s'ouvrir dans le navigateur, vous cliquerez sur l'icône Android pour pouvoir ajouter *Firebase* à votre application. La seule information nécessaire est le nom du package de votre application, par exemple «ch.heigvd.iict.dma.labo1»

3. Vous téléchargerez ensuite le fichier *.json* de configuration et l'ajouterez, comme indiqué, dans votre projet sur *Android Studio*. Vous ajouterez également les dépendances indiquées dans les deux fichiers *gradles* de votre projet.

4. La configuration que nous venons d'effectuer est générique aux différents services offerts par *Firebase*, pour utiliser plus spécifiquement les Firebase Cloud Messaging, vous devrez ajouter les librairies spécifiques. Les dépendances relatives à *Firebase* dans le fichier *gradle* de votre module *app* devront être les suivantes :

   ```
   // Firebase messaging
   implementation(platform("com.google.firebase:firebase-bom:x.y.z"))
   implementation("com.google.firebase:firebase-messaging")
   implementation("com.google.firebase:firebase-messaging-ktx")
   ```

   Vous noterez l'absence de numéro de version pour les deux librairies de *firebase-messaging*. En raison du nombre important de librairies et des différentes versions de celles-ci, il deviendrait très difficile de gérer les éventuelles incompatibilités entre elles, le *BoM*[1] (Bill of Materials) permet de gérer à notre place ces dépendances, en tant que développeur n'allons décider que du numéro de version du *BoM*, celui-ci chargera alors les versions stables et compatibles entre elles des librairies listées.

5. La configuration de votre projet *Android Studio* est à présent finalisée, vous pouvez ajouter les deux fonctionnalités :
   a. Le *Service* permettant de traiter les messages reçus
   b. Le code pour enregistrer votre app sur un topic de distribution dans votre *Activité* principale

Pour envoyer un message push, nous devons passer par les serveurs de *Google*. Jusqu'à récemment l'envoi passait par une simple requête *POST* sur un *endpoint* d'une *API* avec un

[1] Plus d'informations sur le BoM: https://firebase.google.com/docs/android/learn-more?authuser=1#bom

token fixe d'identification, cela pouvait aisément être réalisé en ligne de commande avec une requête *curl*. Depuis récemment, l'envoi se déroule via une identification *OAuth2*, nécessitant deux appels successifs à deux services différents : 1) pour l'obtention d'un *token* temporaire à l'aide d'une clé privée, et 2) l'appel à l'*API* d'envoi du message. Cette nouvelle *API* est beaucoup plus compliquée à utiliser en ligne de commande, on va préférer passer par un logiciel utilisant des librairies *Firebase* « haut niveau ».
La mise en place de cette nouvelle API sort du cadre de ce laboratoire, nous vous fournissons pour cela un petit logiciel *Java* (sous la forme d'un jar exécutable : *DMAMessagePusher.jar*) utilisable en ligne de commande, permettant de réaliser plus facilement l'authentification et l'envoi du message. En cas d'intérêt, nous pouvons vous fournir le code source de cet outil.

Pour utiliser l'outil, vous devrez tout de même posséder une clé privée, liée à votre compte *Google* (le même que celui utilisé pour créer le projet *Firebase*), permettant de réaliser l'envoi de messages *push*. Nous allons voir à présent comment créer cette clé :

1. Sur la console *Firebase*, https://console.firebase.google.com/, ouvrez, si ce n'est pas déjà fait, le projet créé à l'étape précédente. Dans le menu latéral, cliquez sur la roue dentée et ouvrez les « Paramètres du projet », puis aller sur l'onglet « Comptes de service ».

2. Vous cliquerez alors sur « Générez une nouvelle clé privée », un nouveau fichier *.json* va être téléchargé par notre navigateur, celui-ci contient, votre clé privée. Si tout s'est déroulé correctement, votre clé aura automatiquement les permissions requises pour l'envoi de messages push.

3. Placer ce fichier *json* à un emplacement adéquat sur votre ordinateur, vous créerez ensuite la variable d'environnement GOOGLE_APPLICATION_CREDENTIALS avec comme valeur le chemin absolu vers ce fichier *.json*.

4. Ouvrez un nouveau terminal pour vous assurer que la variable d'environnement soit bien prise en compte. Dans ce terminal, déplacez-vous dans le dossier dans lequel se trouve l'outil *DMAMessagePusher.jar* et lancez la commande suivante (vous devez avoir au minimum *Java 8* dans votre *PATH*) :
   > java -jar DMAMessagePusher.jar

5. En cas de succès, la commande retournera le code 0. Sur *Powershell*, vous pouvez contrôler le code de retour avec la commande suivante :
   > echo $LASTEXITCODE

6. Le logiciel s'utilise en ligne de commande, avec les paramètres *--topic* et *--data*
   Pour spécifier un topic de distribution différent de *all*, voici un exemple de commande :
   > java -jar DMAMessagePusher.jar --topic monTopic

   Les données contenues dans le message envoyé sont une *Map* de *clés/valeurs*, par défaut nous envoyons uniquement la paire « *message=test* », vous pouvez définir

vos propres paires de *clés/valeurs* avec le paramètre *data*, celui-ci peut être utilisé une ou plusieurs fois :

```
> java -jar DMAMessagePusher.jar --data "key1=value1" --data "key2=value2"
```

Il est bien entendu possible d'utiliser conjointement --topic et –data

```
> java -jar DMAMessagePusher.jar --topic mTopic --data "k1=v1" --data "k2=v2"
```

Les messages en lien avec *SLF4J* peuvent être ignorés.

7. Une fois que vous aurez mis en place le *Service* de réception des messages sur votre application *Android* avec un affichage de ceux-ci dans les *Logs* et que vous vous serez enregistré à un topic de distribution, vous pourrez utiliser l'outil pour vérifier la bonne réception des messages envoyés.