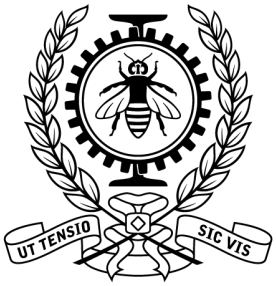

POLYTECHNIQUE
MONTRÉAL

LE GÉNIE
EN PREMIÈRE CLASSE

INF8405 – Informatique mobile

TP2 – Application d'organisation d'événements

Soumis à : **Fabien Berquez**

Elyes Kriaa 1768882

Mostafa Yousefi 1823736

Louise Piotte 1683988

23 mars 2017

Introduction

Pour le deuxième TP de la cour de programmation mobile, On nous a mis en charge de créer une application ayant des composantes utilisées fréquemment dans les applications. Le but était d'apprendre à utiliser les capteurs du téléphone, de connecter notre application à une base de données et d'utiliser Google Maps.

Pour atteindre ce but, l'application à faire était une application qui permet d'organiser un évènement pour plusieurs personnes.

Décision technique

Packages

Nous avons cinq packages pour notre code :

- Activities : contient les activités qui représentent des pages de l'interface.
- Dao : les interactions avec la base de données
- infoWindow : les classes ayant rapport aux « infoWindow » des markers pour Google Maps
- model : Notre instance et notre représentation des données dans l'application pour être utilisé ailleurs. Cherche les données avec les classes dao dans le package du même nom.
- Services : classes gérant la création et la gestion des services pour notre application (par exemple le GPS)

Logique générale

Voici structure générale de notre application du point de vu logique.

- On débute de l'activité « MainActivity », ou on prend le nom, le groupe et la photo de l'utilisateur. De plus, dans cette activité on se charge de la connexion à base de données et à sa lecture initiale en plus de l'initialisation des services de surveillance de network, de GPS et de batterie.
- Après avoir confirmé, l'application envoi l'utilisateur a l'activité contenant la carte Google. On lit les objets de modèles qui ont été remplis par les classes dao et on met l'information pertinente dans les markers de la carte tout en installant notre propre implémentation du « infoWindow » spécialisé.
- Ensuite on attend les inputs de l'utilisateur avec différents observateurs. Dépendant de quelle action est prise, on va updater les données de la base de données ou ajouté un évènement au calendrier du téléphone.

Plan de la base de données


```
roups" : {
Groupe" : {
"events" : {
  "description" : {
    "coordinate" : {
      "latitude" : 0,
      "longitude" : 0
    },
    "dateEnd" : "string",
    "dateStart" : "string",
    "eventName" : "string",
    "going" : {"id" : "string",...},
    "maybe" : {"id" : "string",...},
    "notgoing" : {"id" : "string",...},
    "picture" : "string"
  }
},
"lieux" : {
            "string":{
                        "coordinate" : {
                          "latitude" : 0,
                          "longitude" : 0
                                    }
    "pictureURI" : "string",
    "name" : "string"
                      }
          },
"users" : {
  "string" : {
    "coordinate" : {
      "latitude" : 0,
      "longitude" : 0
    },
    "organisateur" : "false",
    "pictureURI" : "string",
    "username" : "string"
  }
}


roupsNames" : {
id" : "string"
```


Pour notre base de données entre les cellulaires, nous avons utilisé le système « noSQL » de « Firebase ». Nous avons fait ce choix car nos données étaient relativement petites et limité en nombre. De plus, ce système offre un système d'observateurs qui permet de recevoir les update immédiatement, plutôt que notre application les demande explicitement à chaque fois. Voici notre structure de base de données.

À la racine, nous avons groups qui contient les groups et un liste des nom des groups pour pouvoir les identifier sans downloader toute la base de données.

Goups est diviser en plusieurs groupes qui utilisent leurs nom comme id.

Ce groupe contient la liste des lieux, l'évènement et les utilisateurs qui font partis du groupe. Encore une fois, leurs noms est leur ID.

Lieu et évènement

Il est important de noter que les lieux sont détruits après la construction de l'évènement pour éviter que d'avoir deux markers à la place sur la carte. On conserve les coordonnées dans l'évènement.

Services

Les services ainsi que les « BroadcastReceiver » nous ont permis d'implémenter les parties de détection de changement de localisation de l'utilisateur. La localisation est détectée grâce à un « LocationListener » et ceci même lorsque l'application marche en arrière-plan grâce à un Flag « START_STICKY ». Ce service fonctionne avec deux types de fournisseurs d'informations, à savoir la localisation grâce au GPS ainsi que le réseau WIFI. Le rôle du « BroadcastReceiver » ici est de détecté la disponibilité de la connectivité à internet « *NetworkStatusService* ». Ce dernier signale l'interruption de la connectivité à l'utilisateur et

mets en arrêt le service de détection de localisation afin d'éviter un crash de l'application. L'autre « BroadcastReceiver », sert à mettre à jour les paramétrages de délai de mise-à-jour de la réception de la localisation en cas de batterie faible du téléphone « *BatteryStatusService ».*

Markers

Un aspect intéressant de notre design est comment nous traitons nos markers. Nous avons utilisé l'option de personnaliser les markers de Google Maps que l'API offre. Par contre, cet API ne permet qu'un design par carte. Alors nous avons mis une option dans le « snippet » du marker. Selon cette option, nous avons identifié le type de marker (« user », « location », « event ») et nous avons utilisé un « layout » différent dans chaque cas. De plus, les markers de Google Maps ne permettent pas de mettre de boutons dedans car les markers ne sont que des images. Alors nous avons mis l'observateur pour les clicks sur tout le marker plutôt que simplement sur le bouton.

Difficultés

Une des grandes difficultés que nous avons rencontrées est apprendre à utiliser tous les librairies et modules externes différents qui constituent notre application. Ils étaient nombreux et nous n'avions pas utilisé un seul d'eu au paravent.

Une autre difficulté que nous avons rencontrée est connecter et structurer notre base de donnée. Au début de la construction de notre application, nous avons eu plusieurs problèmes de connexion avec « Firebase ». En effet, seulement un de nous réussissions à nous identifier correctement à notre projet.

Enfin la dernière difficulté que nous avons eue était la longueur de l'Application. Nous avons dû mettre un grand nombre de journées de travail pour faire fonctionner correctement toutes les nombres fonctionnalité de l'application donc, nous avions de la difficulté à rentrer cela dans notre horaire, même avec la semaine de relâche.

Critique et suggestions

Comme il a été mentionné dans la section difficultés, ce TP était très long. Pour diminuer la charge de travail tout en gardant la partie apprentissage du TP, on pourrait simplement couper la partie de lieu et de passer immédiatement à la section évènement.

Une autre solution serait de répartir la charge. De mettre un peu plus de sections du TP2 dans le TP1 (par exemple, une table de meilleurs scores pour introduire « Firebase » dans le TP1 pour qu'une partie du temps à passer à l'apprendre soit mieux rependu à travers la session.