

INF8405 – Informatique Mobile

**TP3 –** Outdoor Fishing

Groupe 01

Cloutier, Frédéric 1633375

Lam, Ba Samson 1671028

Vuong, Sylvester 1635535

Soumis à : Berquez, Fabien

28 Avril 2017

Introduction

Dans les années passées, nous avons constaté une croissance importante dans le développement des applications mobiles. Ce progrès fut marqué par la naissance des téléphones intelligents. Il serait donc intéressant d’étudier sur la technologie des téléphones mobiles. Plus précisément, nous allons nous intéresser dans le développement des applications mobiles.

Dans le cadre de ce projet, nous devons concevoir une application mobile à titre éducatif. Le but du travail est d’aider à l’apprentissage avancé pour les étudiants, notamment dans l’utilisation de service et des réseaux sans fils, le partage de données par NFC et l’utilisation de capteurs.

À cette fin, nous concevons une application de partage d’informations de pêches. L’application permet à l’utilisateur de prendre en photo une prise de poissons lors d’une activité de pêche. À ce moment, la localisation de l’appareil ainsi que plusieurs données provenant des capteurs sont enregistrées. La photo et les données sont ensuite stockées dans une base de données sur *Firebase*. L’utilisateur peu visionner les entrées faites par les autres utilisateurs via une carte *Google Maps* sous forme de marqueurs situées à la localisation de l’appareil au moment de la prise de données. ADD STUFF PHOTOS,NFC,SHARE SHIT, SCROLLING VIEW

Pour réaliser ce projet, il faut d’abord conceptualiser les différentes activités de l’application. Donc, ce rapport décrit tous les aspects techniques reliés à la conception, énumère les difficultés rencontrées lors de celle-ci. Enfin, des critiques et suggestions sont proposées.

Présentation technique

**Base de données**

En premier temps, toutes les données et les photos pertinentes à l’application sont stocké dans une base de données *Firebase*. Voici à quoi ressemble la structure des données :

PHOTO FIREBASE

Fig. 1 : Structure des données sur *Firebase*

~~On retrouve à la racine une liste des différents groupes. Sous chacune d’elles, on retrouve le profil de l’organisateur (~~*~~m\_manager~~*~~), les informations de lieu et de temps reliés à l’évènement (~~*~~meeting~~*~~), une liste des différents lieux proposés par l’organisateur (~~*~~placeList~~*~~), et une liste des utilisateurs appartenant au groupe (~~*~~usersList~~*~~). Pour plus de détails, accéder directement à la base de donnée publique via :~~ [~~https://tp2firebaseproject-e31e0.firebaseio.com/~~](https://tp2firebaseproject-e31e0.firebaseio.com/)

Les photos, quant à elles, sont simplement stockées dans un module *Storage* sur *Firebase*.

**Structure du projet**

Il y a 8 classes d’activité, soit *AppCompatPreferenceActivity, BatteryActivity, BeamActivity, FishActivity, GalleryActivity, MainActivity, MapsActivity* et *SettingsActivity*. Chacune est décrite plus en détail dans les sections qui suivent. Ensuite, les POJOs utilisé pour la modélisation de la problématique sont *EnvSensorEntry, FishEntry, FishDTO, Group*. Finalement, la classe *Singleton* se charge principalement de toute interactions avec la base de données *Firebase* et *BatteryReceiver* se charge de recevoir le pourcentage de batterie restante de l’appareil.

Du coté des ressources, les deux fichiers à souligner sont *strings.xml* et *pref\_general.xml*. Le premier contient tous les valeurs constantes de chaînes de caractères utilisés à travers l’application et le deuxième sert à sauvegarder le nom de l’utilisateur.

**MainActivity**

L’activité d’entrée est *MainActivity*. Lors de sa création, elle instancie un objet singleton *UserSingleton*. Ce dernier se charge de toutes les interactions avec la base de données *Firebase* et est utilisé principalement dans les activités *CalendarActivity*, *MapActivity*, *PlaceActivity* et *PreferencesActivity*. *MainActivity* s’occupe de prendre en entrée le pseudonyme de l’utilisateur ainsi que sa photo de profil. Elle permet aussi de changer de pseudonyme, ainsi que d’accéder à une activité de préférences (*PreferencesActivity*). Le pseudonyme et la photo sont stockés sur *Firebase* via *UserSingleton*.

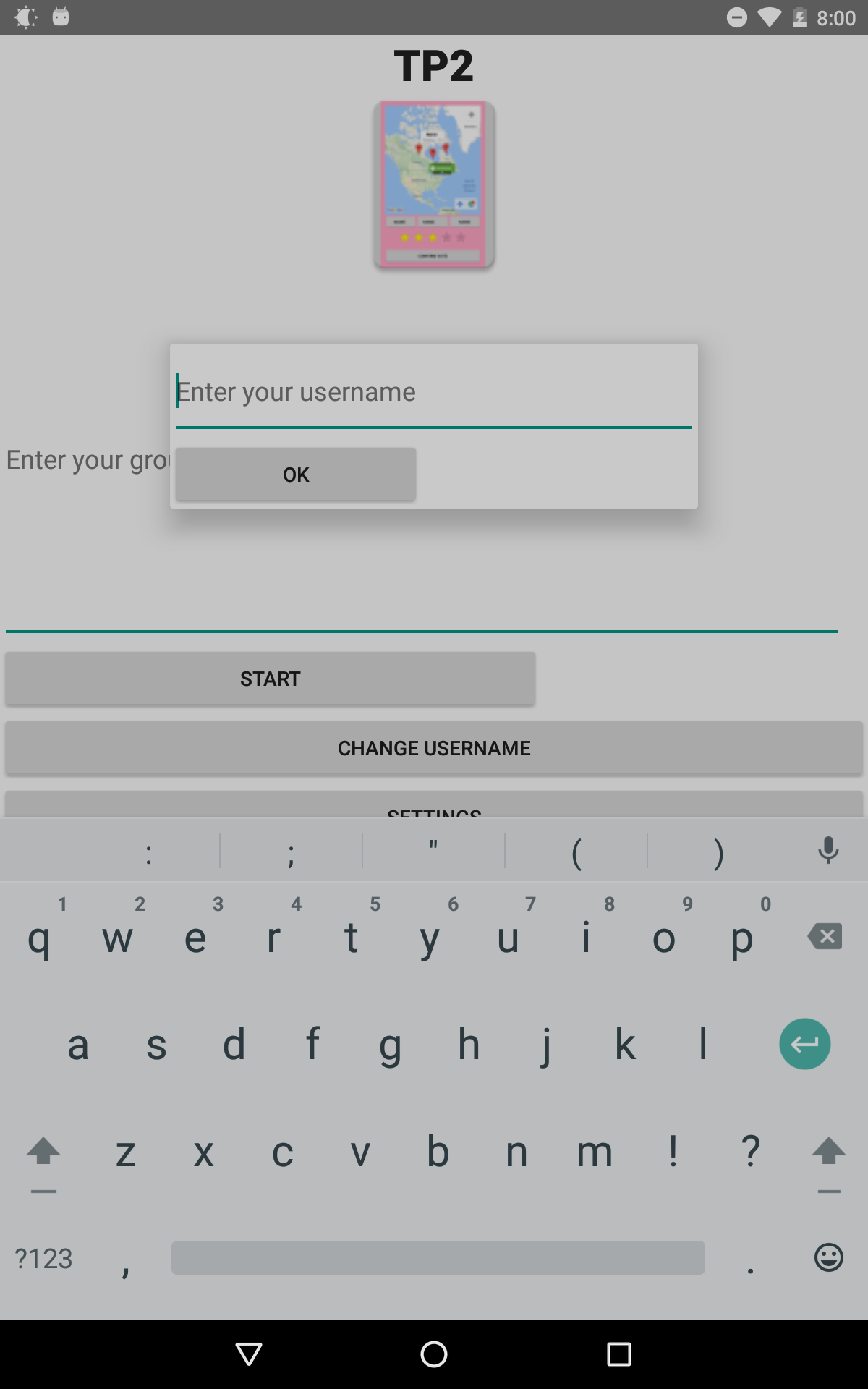
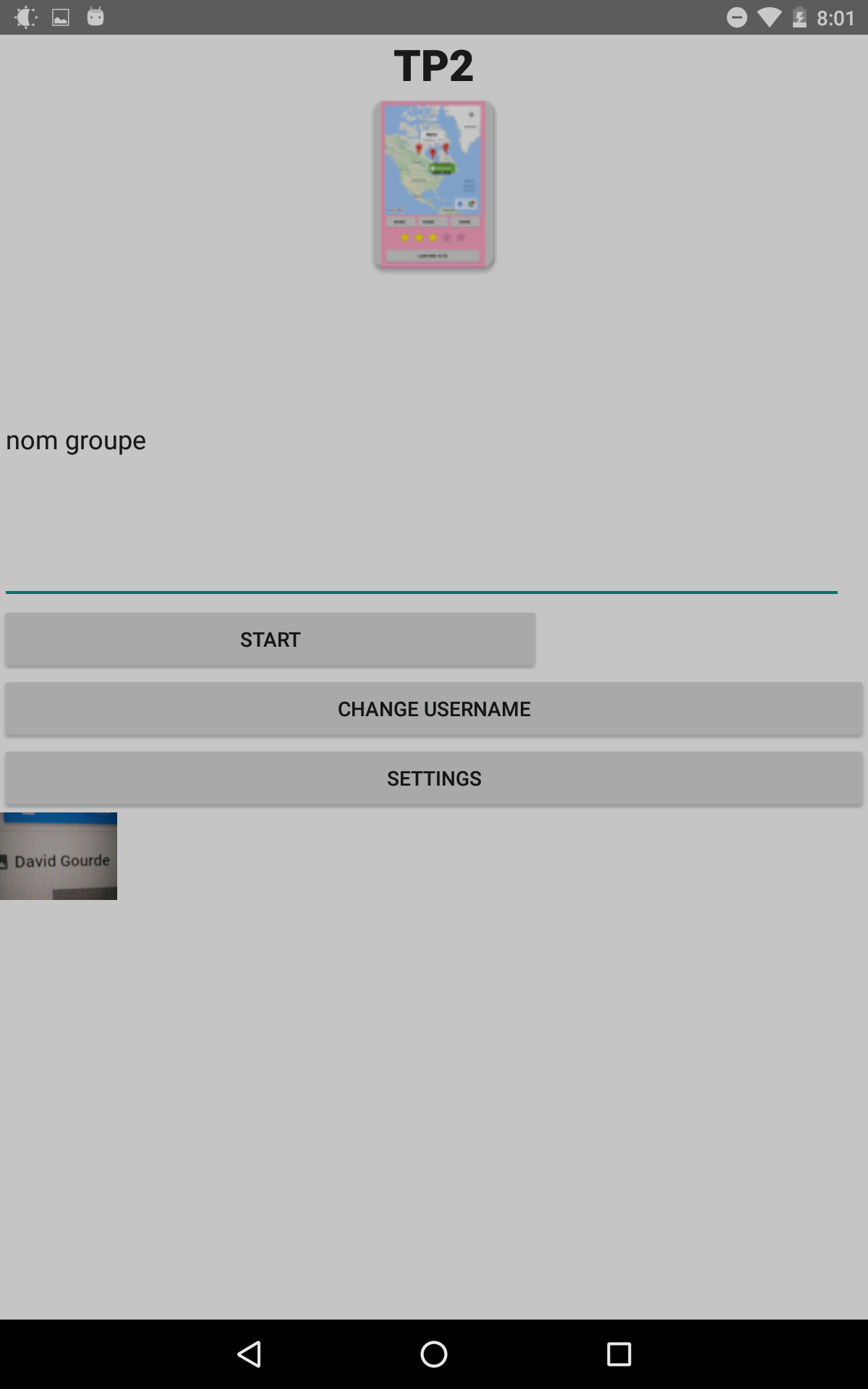


Fig. 3 : Vue de *MainActivity*

**PreferencesActivity**

Cette activité permet à l’utilisateur de changer le temps en secondes entre les mises à jour de sa position géographique. Elle permet aussi de quitter un groupe et de consulter le calendrier avec les évènements enregistrés. Lorsque l’utilisateur quitte un groupe, son profil est retiré de *usersList* dans le groupe correspondant dans la base de données. Cette activité est aussi disponible à travers un bouton *Settings* (  ) au coin supérieur droit de la plupart des autres activités.

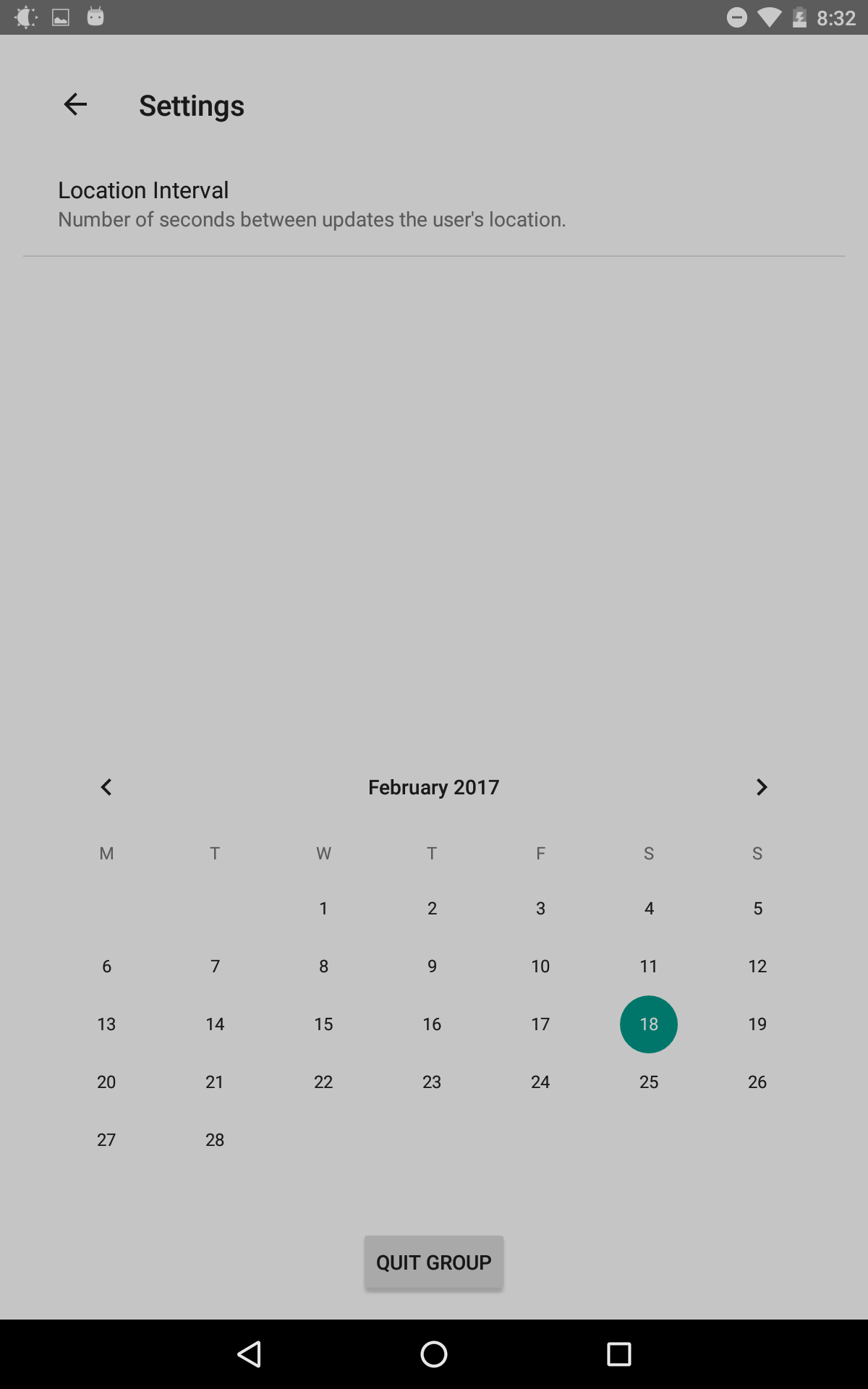


Fig. 4 : Vue de *PreferencesActivity*

**MapActivity**

Cette activité survient lorsque l’utilisateur s’est identifié et appuie sur *Start* à partir de *MainActivity*. Si l’utilisateur est l’organisateur du groupe, il peut ajouter un marqueur sur la carte *Google Maps* en appuyant longuement dessu*s*. Lorsqu’il a placé trois marqueurs, il peut débuter le vote pour les autres membres du groupe.

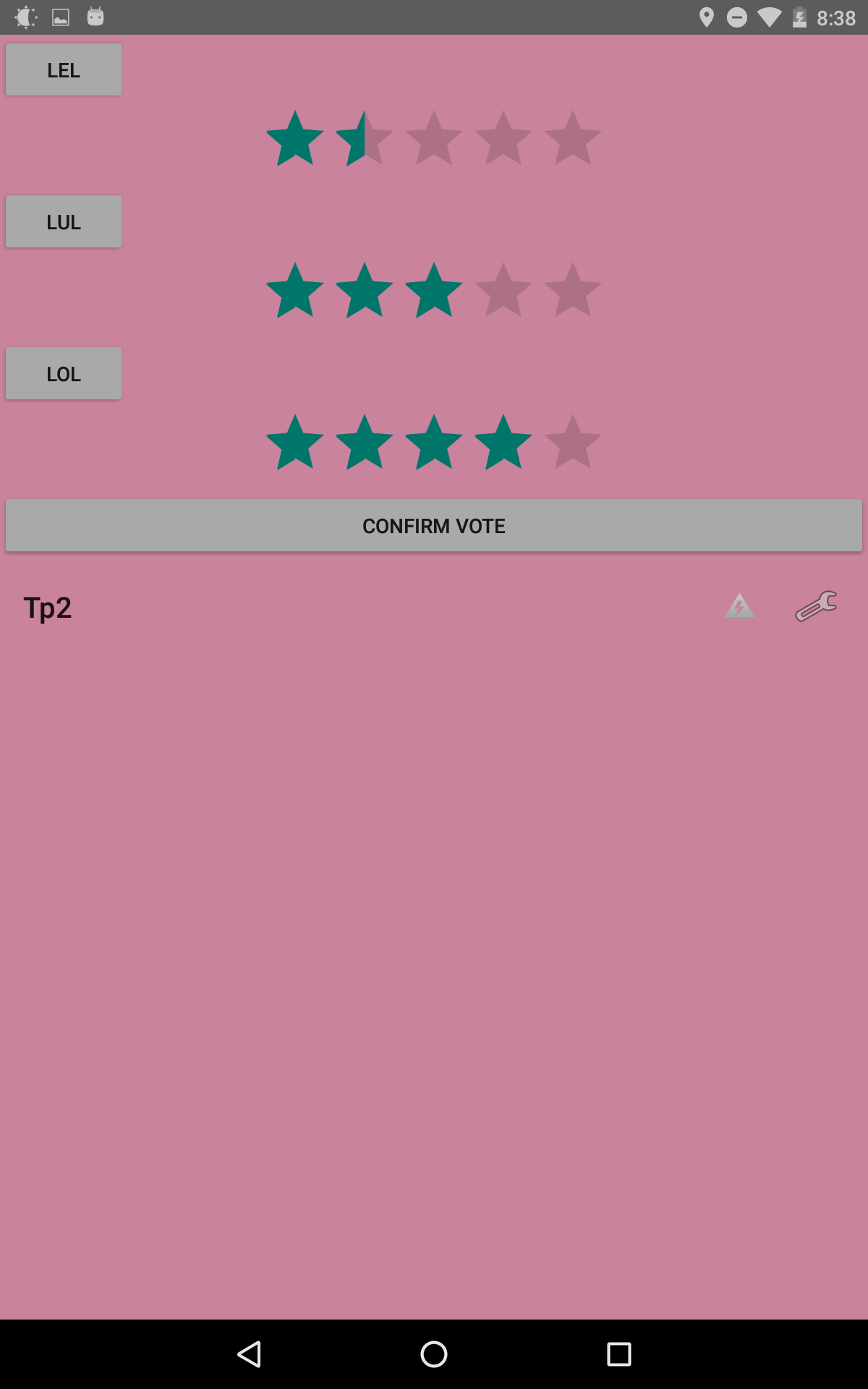
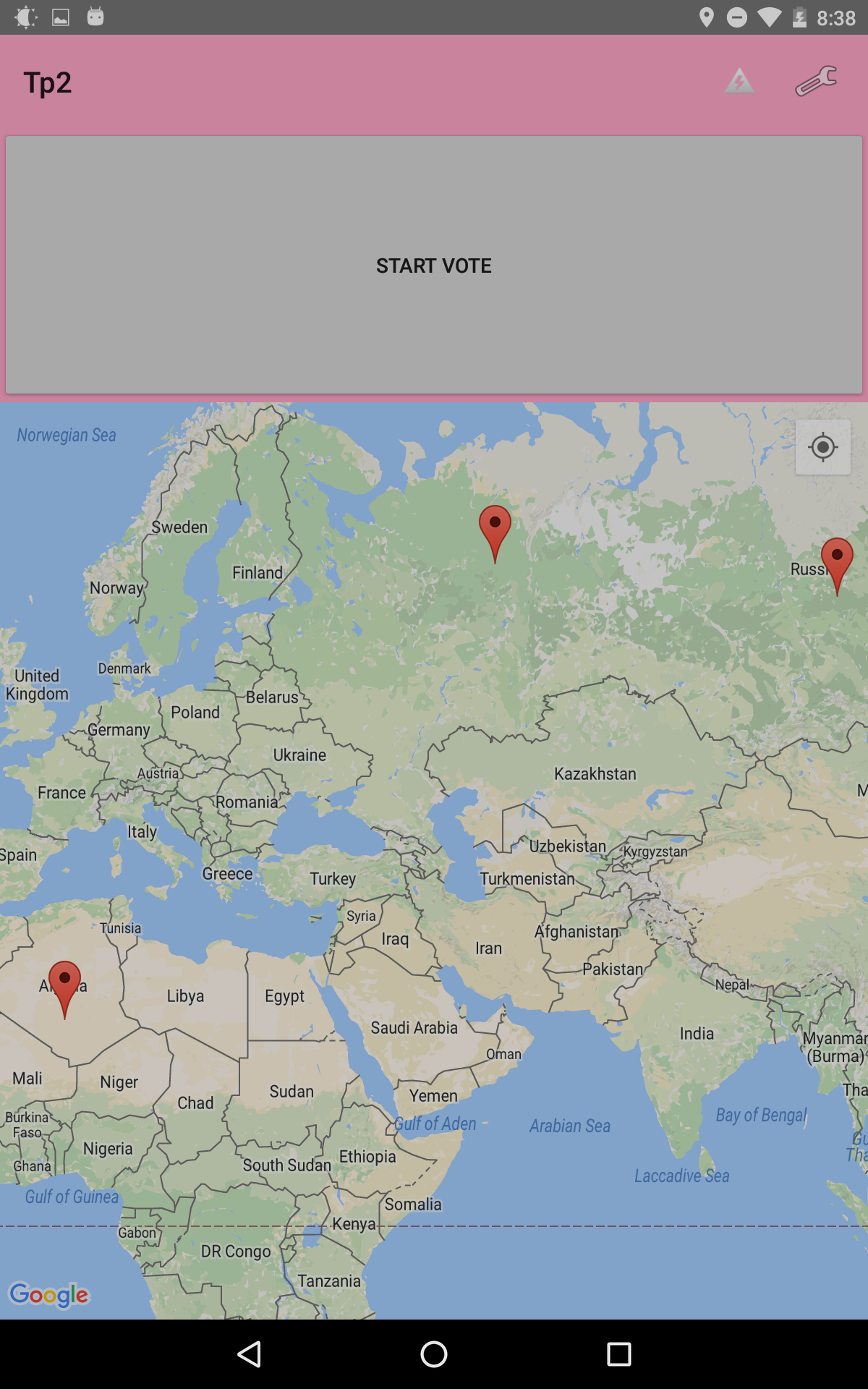


Fig. 5 : Ajout de marqueur et vote

Ensuite, lorsque le lieu est déterminé, l’organisateur peut consulter le résultat des votes pour choisir un lieu et définir les informations de la rencontre tel que l’heure et la date dans une nouvelle activité *CalendarActivity*. Ceux-ci vont être consultables dans un *InfoWindowAdapter* au-dessus du marqueur sur la carte *Google Maps*. De plus, les listes de participation des utilisateurs sont affichées dans le même *InfoWindowAdapter* (Go, Maybe, Declined).

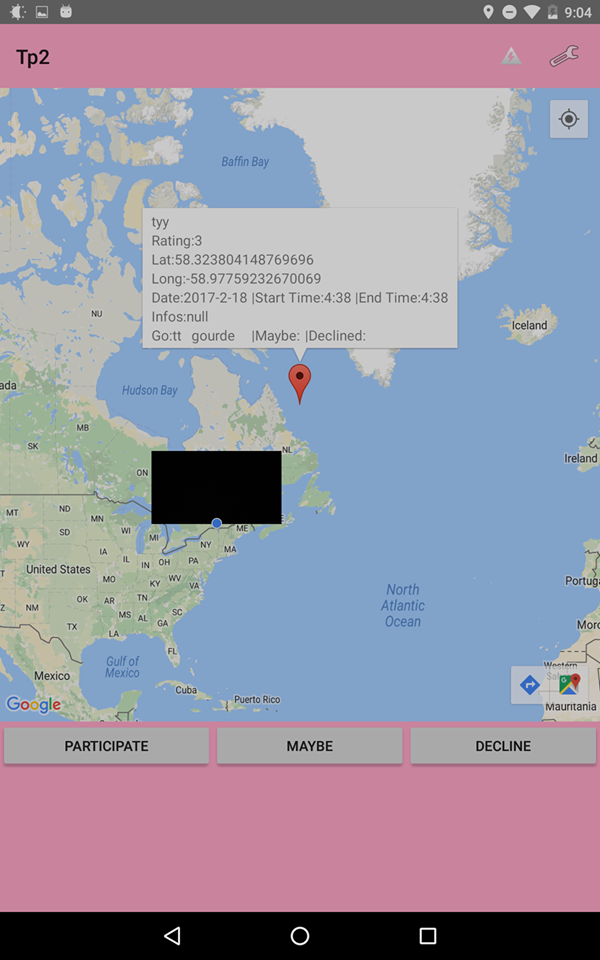
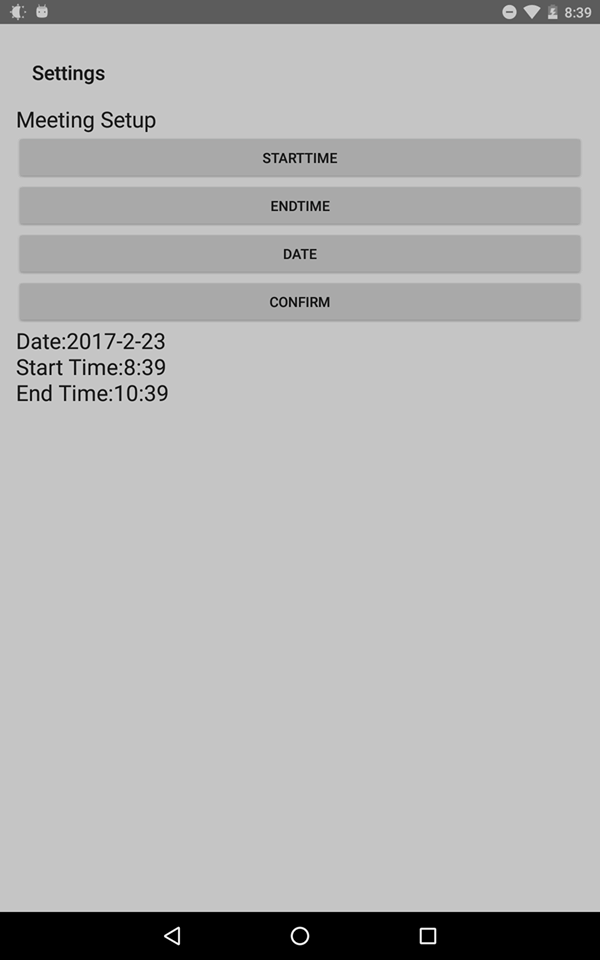
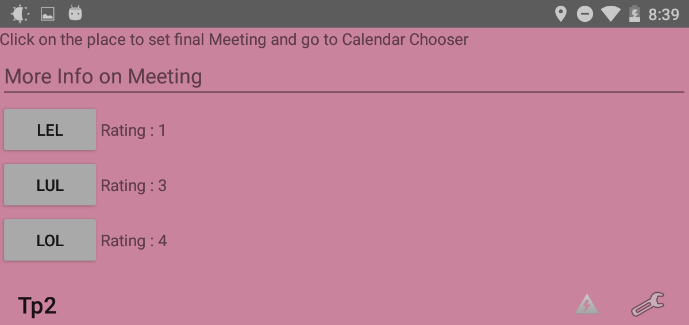


Fig. 6 : CalendarActivity et Informations de la rencontre

 Fig. 7 : Choix du lieu après les votes

**PlaceActivity**

Dans l’activité précédente, soit *MapActivity,* l’organisateur peut appuyer longuement sur la carte *Google Maps* pour créer un marqueur représentant un lieu. Cette action est gérée par un *OnMapLongClickListener* de l’API *GoogleMaps*. Lorsque l’action est detectée, l’application lance l’activité *PlaceActivity*. Dans celle-ci, les coordonnées de latitude et longitude de l’emplacement sur la carte sont affichées, et on demande à l’organisateur d’entrer le nom du lieu ainsi que de choisir une photo.

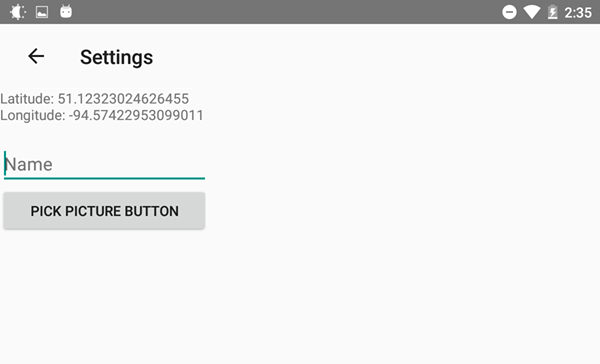


Fig. 8 : PlaceActivity

**BatteryActivity**

Cette activité est accessible à partir de plusieurs autres activités qui possèdent le bouton  au coin supérieur droit de leur vue. Cette activité affiche le pourcentage de batteries restant de l’appareil et donne l’option à l’utilisateur de passer en mode économie d’énergie. Le cas échéant, l’intervalle de temps entre deux envois de la localisation est alors augmenté à 60 secondes plutôt que 5 secondes. Cette valeur est stockée dans *preferences.xml*.

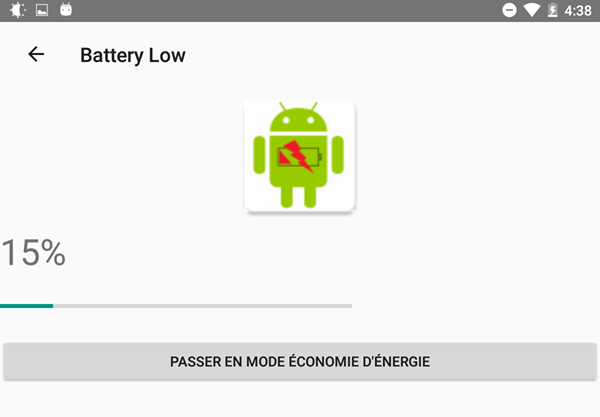
****

Fig. 9 : BatteryActivity

Difficultés rencontrées

L’une des difficultés rencontrées était la modélisation de la problématique. La structure des données que nous avons utilisées a souvent changé durant le développement du projet et la structure finale est complètement différente de celle de départ.

Une autre difficulté réside dans l’utilisation de la base de données Firebase. Cette base de données gratuite nous limite dans les types d’objets donc nous pourrions sauvegarder. Par exemple, nous ne pouvons pas envoyer des collections spécifiques à Java. De plus, l’information envoyée doit être sérialisable et idéalement, convertit sous-forme JSON. De plus, la documentation est limitée. Nous constatons aussi que l’utilisation d’un *string* de longueur moins de deux ayant un chiffre numérique comme caractère (ex : « 1 ») semble des fois ne pas être bien sauvegardé lors de l’appel *dataSnapshot.getvalue(NameOfClass.class)*. Nous pensons que le problème réside dans la confusion de la lecture de donnée entre un *string* ayant un entier comme élément avec la numérotation (ou bien l’index) lorsque nous tentons d’affecter notre liste dans Firebase. Par ce fait même, nous devons s’assurer que l’utilisateur n’entre pas un entier comme nom d’utilisateur ou bien comme nom du groupe.

Une dernière difficulté se retrouve dans l’utilisation du Google Maps API. Pour le travail, nous devons implémenter plusieurs *Listener*. Cela rend notre code très lourd pour la lecture. Certainement, nous aurions pu séparer cela en classe, mais le travail n’était pas assez complexe pour y procéder ainsi. Dans un projet à plus grande complexité, la programmation orientée objet sera donc de mise.

Critiques et suggestions

Puisque l’utilisation d’une base de données est importante pour la réalisation du travail, nous suggérons que l’école fournisse aux étudiants une base de données beaucoup mieux robuste et stable. Firebase étant une base de données gratuite, ses capacités et ses fonctionnalités sont ainsi limités, quoique dans le cadre du travail, nos besoins sont aussi limités. Toutefois, une base de données beaucoup plus complète et offerte par l’université pourrait bénéficier les étudiants à mettre en pratique leur connaissance en configuration et interaction avec une base de données.

Autrement, ce travail pratique permet les étudiants à se familiariser avec des outils et d’API gratuits. Certainement, le Google Maps est fréquemment utilisé; les étudiants en tireront bénéfices.

//TODO Autre critique ou suggestions..