



Master 2 Informatique STL Insta

Application Android

---

# Développement d'un client Android de l'application web Halitran

---

*Auteurs :*

M. Ladislav HALIFA

M. Patrick TRAN

*Encadrants :*

M. Gregory POTDEVIN

7 novembre 2016

# Table des matières

<b>1</b>	<b>Présentation de l'application</b>	<b>1</b>
1.1	Idée générale . . . . .	1
1.2	Fonctionnalités . . . . .	2
<b>2</b>	<b>Implémentation</b>	<b>11</b>
2.1	Requettes HTTP . . . . .	11
2.2	Gestion des sessions . . . . .	11
2.3	Activités et Fragments . . . . .	12
2.4	Liste de messages . . . . .	13
<b>3</b>	<b>Difficultés</b>	<b>15</b>
3.1	Problèmes rencontrés et solutions . . . . .	15
3.2	Améliorations . . . . .	16

# Chapitre 1

## Présentation de l'application

### 1.1 Idée générale

Nous avons choisi d'implémenter une application Android reprenant les fonctionnalités du site web [Halitran](#) , une application web développée dans un précédent projet. Cette application est un réseau social de type Twitter, basée sur une API Rest, permettant à ses utilisateurs de partager des messages courts. L'application Android offre ainsi un accès direct au service en ligne. Le client web est accessible à l'adresse suivante :

[http://vps197081.ovh.net:8080/HALIFA\\_TRAN/](http://vps197081.ovh.net:8080/HALIFA_TRAN/)

## 1.2 Fonctionnalités

Nous avons fait le choix d'incorporer à l'application les fonctionnalités dans l'ordre suivant :

- **Inscription** : l'utilisateur doit pouvoir s'inscrire depuis l'application,

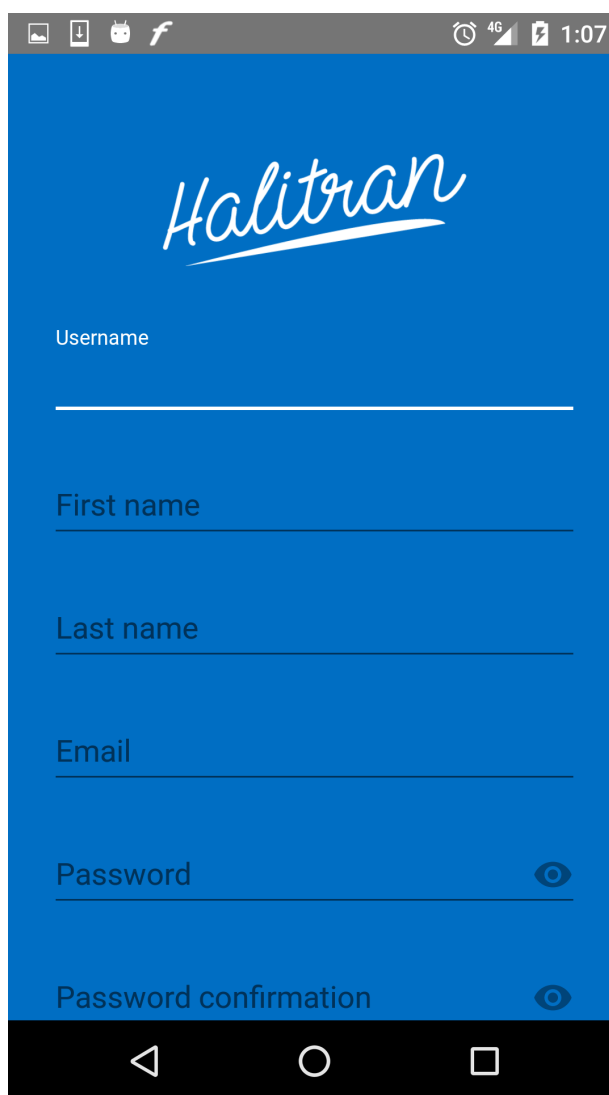


FIGURE 1.1 – Vue d'inscription

Cette vue permet à l'utilisateur de se créer un compte utilisateur sur le site en saisissant ses informations dans les champs textuels, puis en cliquant sur le bouton de validation (non visible sur cette capture d'écran)

- **Connexion/Déconnexion** : l'utilisateur doit pouvoir se connecter et se déconnecter,

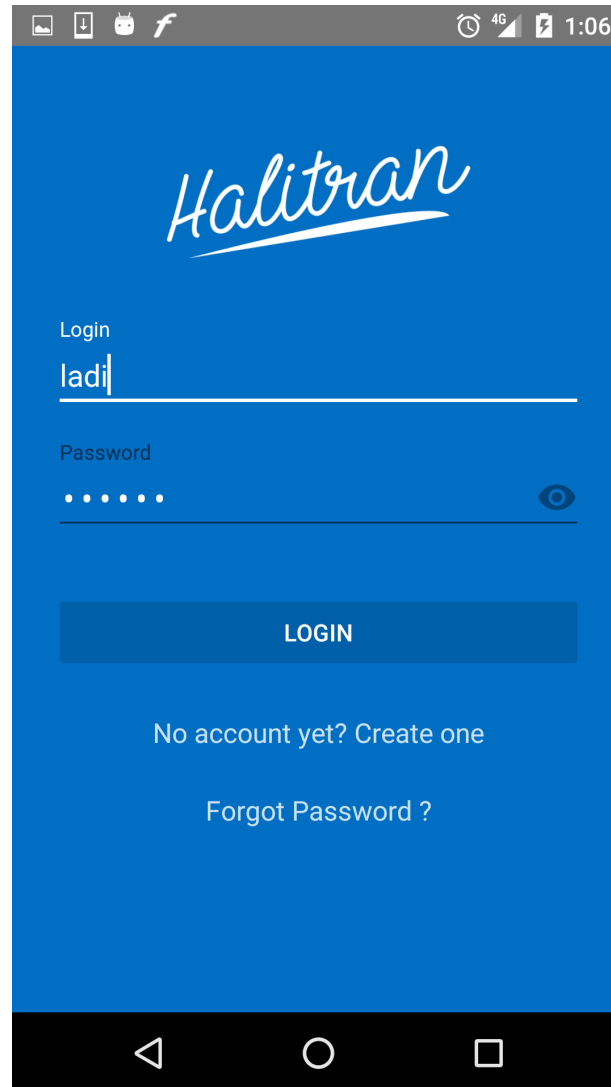


FIGURE 1.2 – Vue d'accueil de l'application en mode déconnecté

L'application démarre sur cette vue qui permet à l'utilisateur de se connecter à l'application à l'aide de son nom d'utilisateur et de son mot de passe. Il se connecte en cliquant sur le bouton LOGIN. Cette vue permet également d'atteindre la vue d'inscription, ainsi que la vue de récupération de mot de passe.

- **Récupération du mot de passe** : l'utilisateur doit pouvoir récupérer son mot de passe en cas d'oubli,

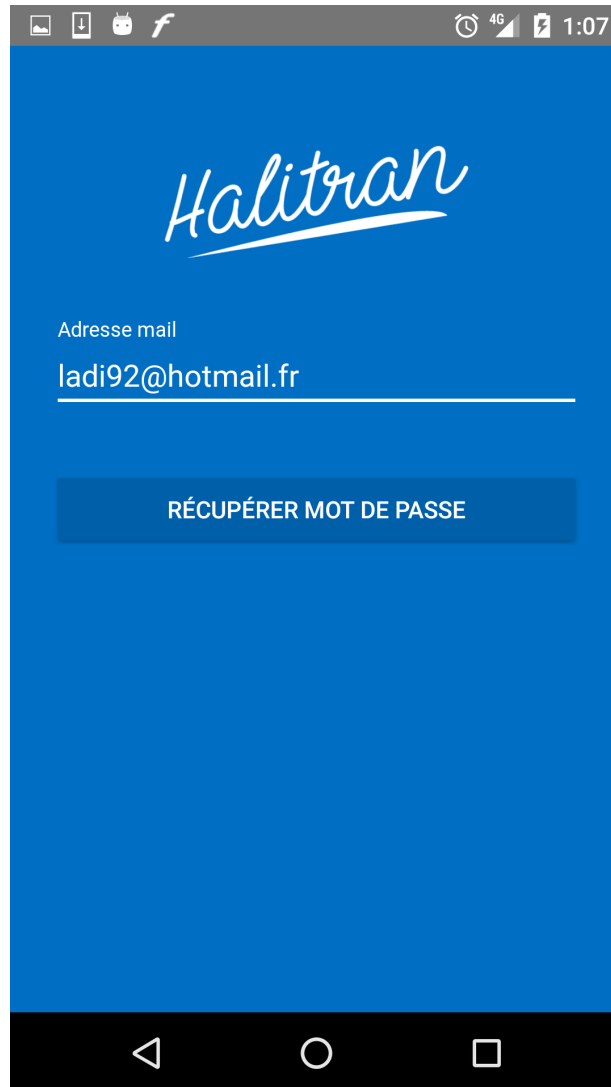


FIGURE 1.3 – Vue de récupération du mot de passe

Cette vue permet à l'utilisateur de récupérer son mot de passe en saisissant l'adresse mail saisie lors de son inscription au site. Un mail contenant son mot de passe lui sera envoyé après un clic sur le bouton.

- **Gestion du compte utilisateur** : l'utilisateur doit pouvoir modifier les informations de son compte (édition du mot de passe, édition de l'adresse mail),

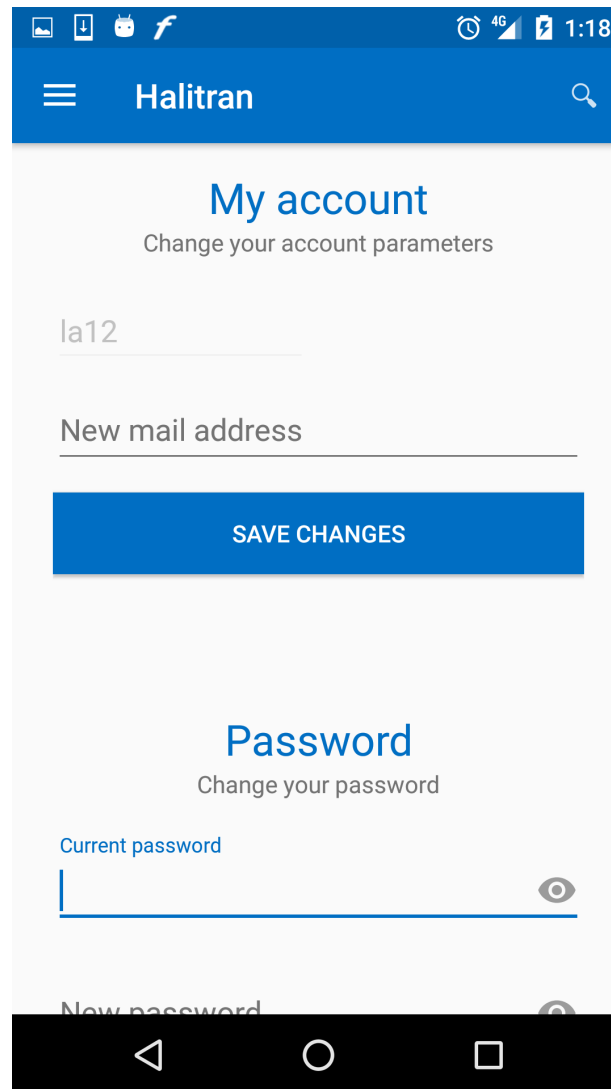


FIGURE 1.4 – Page de Gestion du compte utilisateur

Cette vue permet à l'utilisateur de changer l'adresse mail de son compte utilisateur, et de modifier son mot de passe. Les boutons "SAVE CHANGES" permettent de prendre en compte les changements. Cette vue n'est accessible que si l'utilisateur s'est précédemment connecté à l'application.

- **Gestion des messages publics** : l'utilisateur doit pouvoir afficher tout les messages disponibles sur le site

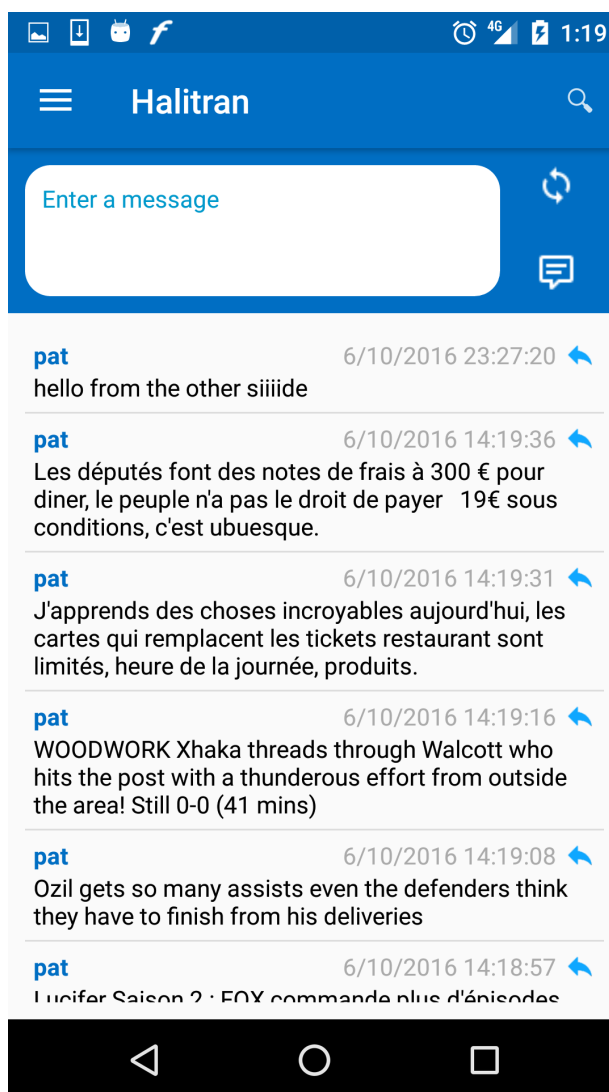


FIGURE 1.5 – Page d'accueil de l'application en mode connecté

Cette vue s'affiche lorsque l'utilisateur se connecte. Elle permet à l'utilisateur de consulter les messages présents sur le site, de rafraichir manuellement la liste des messages, de poster un nouveau message sur le site, de supprimer un message dont il est propriétaire en cliquant sur l'icone en forme de poubelle. La vue permet aussi d'afficher la vue de recherche et d'ouvrir le menu déroulant. Elle permet également de consulter le profil d'un autre utilisateur en cliquant sur son nom.



- **Gestion des messages de l'utilisateur** : l'utilisateur doit pouvoir ajouter un nouveau message, supprimer les messages enregistrés sur son compte
- **Gestion des profils** : l'utilisateur doit pouvoir consulter son profil, ainsi que les profils d'autres utilisateurs

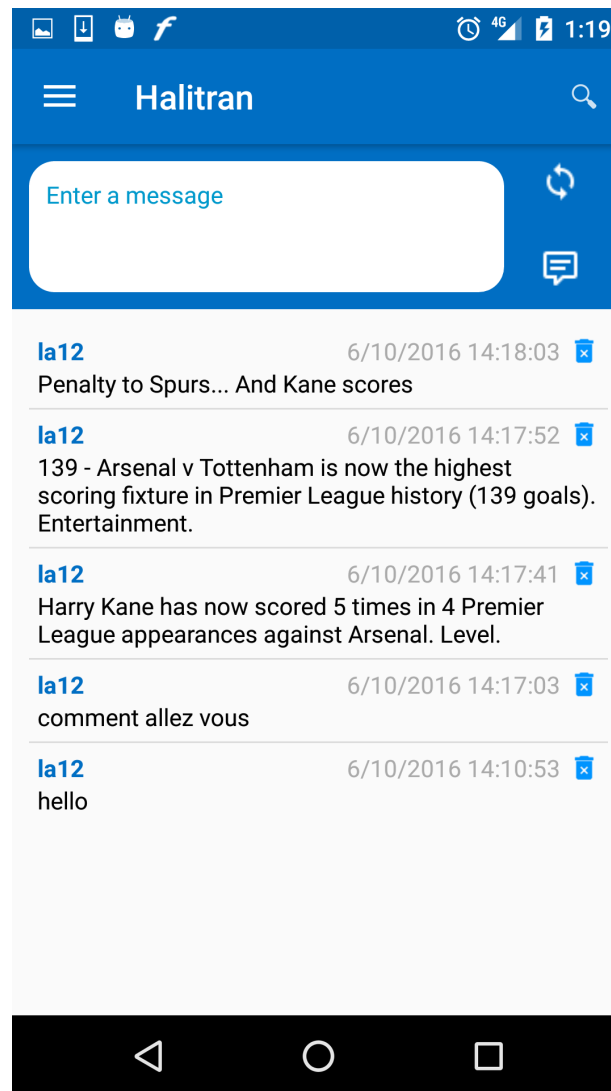


FIGURE 1.6 – Vue du profil de l'utilisateur

Cette vue est la page de profil de l'utilisateur connecté, elle n'affiche que les messages de l'utilisateur. L'utilisateur peut aussi ajouter des nouveaux messages ou en supprimer depuis cette vue.

- **Moteur de recherche** : l'utilisateur doit pouvoir effectuer une recherche sur les messages disponibles sur le site

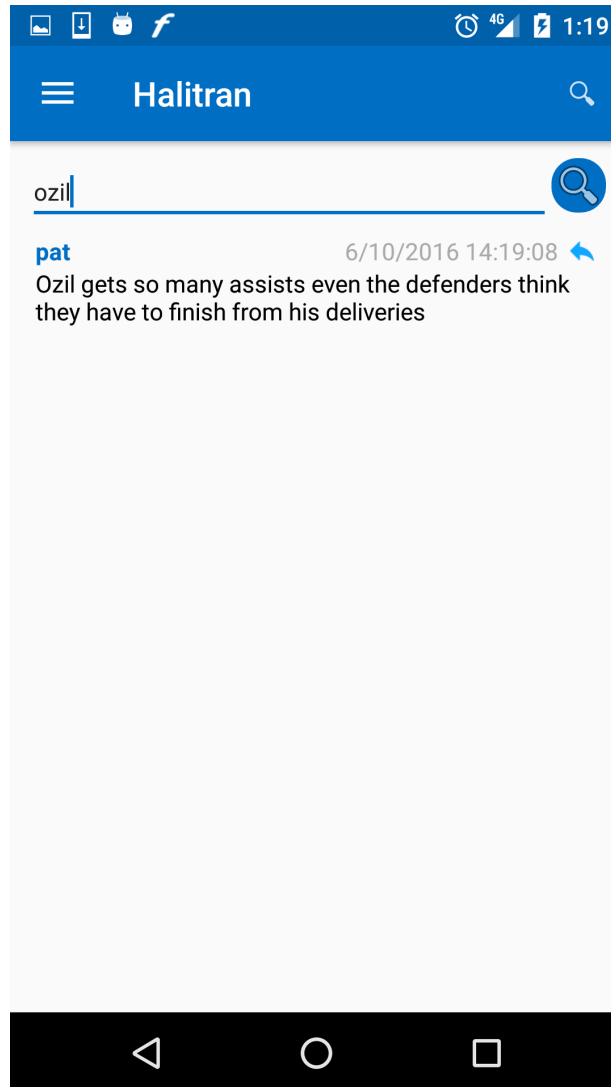


FIGURE 1.7 – Vue du moteur de recherche

Cette vue permet à l'utilisateur d'effectuer une recherche, les messages correspondants à sa recherche s'affichent après avoir cliqué sur la loupe bleue. Cette vue n'est accessible que si l'utilisateur s'est précédemment connecté à l'application.

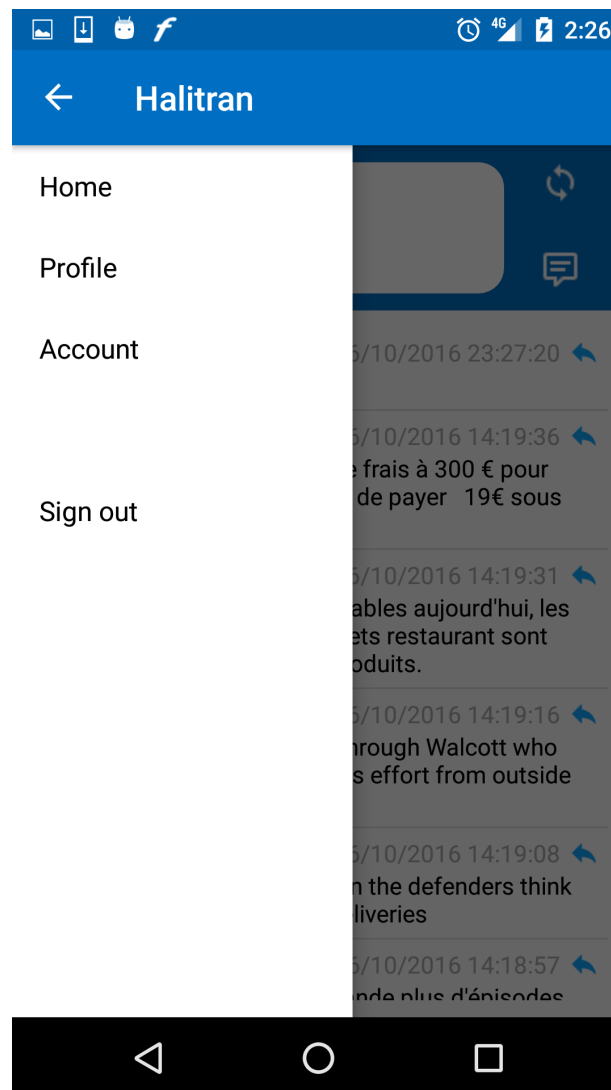


FIGURE 1.8 – Menu Déroulant

Le Menu déroulant est disponible sur toutes les vues s'affichant lorsque l'utilisateur est en mode connecté. Il est accessible soit en effectuant un swipe vers la droite depuis le bord gauche de l'écran, soit en cliquant sur le bouton en haut à gauche. Il permet à l'utilisateur de naviguer entre les différentes vues de l'application, et lui permet également de se déconnecter de l'application.

# Chapitre 2

## Implémentation

### 2.1 Requettes HTTP

Notre application utilisant l'API d'Halitran qui est une API Rest, il a fallu trouver un moyen de communiquer via le protocole HTTP. Le format de réponse de l'API étant des documents JSON, nous avons utilisé les `JsonObjectRequest` de la bibliothèque Volley, qui permet de construire et envoyer les requêtes ainsi que de traiter les réponses reçues. Un listener est passé au constructeur de cet objet, dans lequel un traitement est effectué en fonction de la réponse. Une fois le `JsonObjectRequest` paramétré, il est passé à une `RequestQueue` qui se charge d'envoyer les requêtes au serveur au fur et à mesure qu'elle les reçoit.

### 2.2 Gestion des sessions

L'application étant une application de type réseau social, il était nécessaire de faire en sorte que l'utilisateur n'ait pas à se reconnecter à chaque lancement de l'application (à l'image des sessions de la version web, qui permet de rester connecté un certain temps). Dans le cadre d'une application mobile, il est préférable d'avoir une session qui n'expire jamais tant que l'utilisateur ne se déconnecte pas de lui même. Pour cela, nous avons utilisé une classe `HalitranApplication` qui étend la classe `Application` et qui nous permettra de sauvegarder les informations propres à la session à savoir : le nom d'utilisateur, l'id de l'utilisateur, la clé session. Nous y avons aussi implémenté les fonctions permettant d'accéder aux "shared preferences" dans lesquels nous stockons le nom d'utilisateur et le mot de passe, ce qui permet de reconnecter l'utilisateur à chaque lancement d'application. Enfin, nous y avons aussi stocké les messages de la vue courante pour qu'ils ne soient pas effacés lors du changement d'orientation du mobile. Si jamais l'utilisateur voit sa session expirée,

et qu'il décide quand même d'utiliser un service de l'API, une fonction de reconnexion est lancée et le service est rappelé automatiquement avec les mêmes paramètres. Par exemple, s'il reste sur la vue "Account" pendant 30 minutes, et qu'il décide de changer son mot de passe, une reconnexion sera faite et le service "changer mot de passe" qui avait échoué la première fois à cause de la session expirée, sera une nouvelle fois appelé.

## 2.3 Activités et Fragments

Notre application compte en tout 5 activités et 5 fragments. Les activités sont les suivantes :

- LoginActivity qui est l'activité contenant le formulaire de connexion.
- RegisterActivity qui est l'activité contenant le formulaire d'inscription.
- ForgotMdpActivity qui est l'activité contenant le formulaire de récupération de mot de passe.
- MainActivity qui gère les vues quand l'utilisateur est connecté.
- SplashActivity qui fait office d'écran de connexion et qui appelle MainActivity ou LoginActivity en fonction de si un utilisateur est connecté ou non.

Les fragments qui s'affichent dans l'activité principale sont les suivants :

- HomeFragment qui affiche la page d'accueil de l'application dans laquelle tous les messages sont affichés.
- ProfilFragment qui affiche le profil de l'utilisateur (sa bio et ses messages).
- UserprofilFragment qui affiche le profil d'un utilisateur autre que l'utilisateur connecté sur l'application.
- AccountFragment qui affiche les paramètres du compte de l'utilisateur (les paramètres qu'il peut changer).
- SearchFragment dans laquelle l'utilisateur peut chercher un message via des mots clés.

L'utilisateur navigue à travers ces différentes vues grâce à une drawerlist situées à gauche de l'écran, qui contient aussi le bouton de déconnexion. En ce qui concerne le fragment search, il s'affiche lorsque l'on appuie sur le bouton de type "option item" dans l'action bar. Pour accéder au profil d'un autre utilisateur, il suffit de cliquer sur son nom dans un des messages qu'il a écrit.

Au niveau du design, la bibliothèque Android Support Design Library a été utilisée pour tous les formulaires à l'exception des formulaires d'envoi de message et de recherche. Cette bibliothèque permet d'avoir des inputs et des boutons à la fois modernes et sobres notamment grâce à l'utilisation de "TextInputLayout" qui encadrent les EditText et qui permettent l'affichage d'infobulles d'erreur, ou l'affichage d'une petite animation au focus sur l'input. La plupart des vues sont des Scrollview ou contiennent des Scrollview, permettant à l'utilisateur d'avoir accès à toutes les informations d'une vue à tout moment.

## 2.4 Liste de messages

Les messages sont affichés sous forme de liste. Ce sont des ListView qui ont été utilisés. Ces ListView sont accompagnés de MessageAdapter qui étendent ArrayAdapter<Message>. Ce sont dans ces Adapter que sont mis en place les fonctions de suppression de message, de réponse à un utilisateur, et de création de lien vers un profil utilisateur. Quant aux Message, ce sont des objets contenant toutes les informations liées aux messages à savoir id, date, texte, auteur. Ce sont ces informations qui seront affichés dans les éléments de la ListView, dont les vues seront créées par le MessageAdapter.

# Chapitre 3

## Difficultés

### 3.1 Problèmes rencontrés et solutions

Un des problèmes majeurs rencontrés est celui de la gestion de session. Comme expliqué précédemment, c'est en utilisant le singleton `HalitranApplication` que nous stockons les informations liés à la session qui sont partagées entre fragments et activités. Cette solution a été choisie par facilité, mais nous sommes conscients que d'autres solutions existent et sont peut-être plus sûres comme l'envoi de données via les Intents ou encore l'utilisation d'une base de donnée `Sql Lite`. Un autre problème lié aux sessions a été abordé précédemment : celui d'une session expirée. Si la session d'un utilisateur a expiré et qu'il essaie de faire appel à un service, il recevra une erreur de la part du serveur qu'il ne résoudra qu'en se déconnectant et en se reconnectant. L'idée a été de reconnecter automatiquement l'utilisateur lorsqu'une erreur de session expirée a été reçue et de relancer directement le service concerné. Tout cela s'effectue en arrière-plan et l'utilisateur ne se rend compte de rien.

C'est surtout le stockage du mot de passe de l'utilisateur connecté qui pose un véritable problème de sécurité. Nous ne voulions pas stocker ce mot de passe dans `HalitranApplication` car celui-ci serait effacé à chaque fois que l'application est fermée. Nous avons donc choisi de stocker les mots de passe dans `Shared Preferences`. Malheureusement, ceux-ci sont facilement accessibles à partir de téléphones rootés. Dans le cadre du projet, ce système est suffisant mais dans le cadre d'une application vendue ou publiée sur le Play Store, il aurait fallu revoir le système de clé-session, en le couplant avec un protocole de sécurisation des échanges (SSL par exemple).

Un problème mineur rencontré est celui du positionnement du bouton "Sign out" dans la `DrawerList`. Nous voulions qu'il soit séparé des autres boutons permettant d'accéder aux différents fragments mais il était assez difficile de le faire. Nous avons donc inséré un "bouton vide" entre le bouton "Sign out" et les autres.

## 3.2 Améliorations

Parmi les améliorations, se trouvent les fonctionnalités existantes dans l'API mais non traitées. On y trouve :

- Les amis, qui permettent un affichage personnalisé des messages (seuls les messages des amis s'affichent sur la vue Home) et qui permettent de faire une recherche de messages seulement parmi ceux des amis.
- La suppression de tous les messages.
- La suppression d'un compte.
- La modification de sa bio dans le profil.
- la prise en charge des hashtag
- La reconnaissance des liens http (<http://www.upmc.fr>), des hashtags (lol) et de la mention d'utilisateur (@pabloescobar),
- Le top 10 des hashtags utilisés.

Ces fonctionnalités n'ont pas été traitées par manque de temps. D'autres améliorations pour rendre l'application plus attractive sont : les photos de profil, les images, vidéos dans les messages et l'ajout des messages privés. Ces fonctionnalités n'ont malheureusement pas encore été implémentées côté serveur.