Pierre Augustin TOTARO
Kaoutar BENNOUNA
Sandra VILLAFUERTE
Mikaël PHILIP

Rapport de conception - HouseHunter

ISTIC - 2016/2017 Université de Rennes 1

1)Présentation

Actuellement, de plus en plus de personnes déménagent, en particulier les étudiants. Une des principales difficultés est de trouver des logements de manière simple et rapide. Le principal problème est qu'une fois arrivé dans une nouvelle ville, il n'est pas évident de savoir ou sont les logements disponibles (dans un quartier précis, près du lieu de travail ou de l'université par exemple).

C'est pour cela que nous avons créé une application Android "HouseHunter" qui permet de répondre à ce besoin. En effet, avec notre application, l'utilisateur va pouvoir localiser en temps réel les logements disponibles autour de lui ou à une adresse précise.

L'application répondra à trois objectifs:

- Permettre à des propriétaires de poster et gérer des annonces.
- Permettre aux utilisateurs de visualiser les appartements disponibles sur une carte.
- Permettre aux utilisateurs d'accéder aux informations importantes par rapport à un logement précis.

2)Fonctions et types utilisateurs

L'application est composé de 9 fonctions qui vont nous permettre de répondre au besoin.

L'application aura deux types d'utilisateurs:

- L'utilisateur classique (non connecté) qui utilisera la carte pour sélectionner puis lire des annonces.
- Le propriétaire (qui a un compte) qui pourra, en plus, créer et gérer ses propres annonces.

Les fonctions sont:

- Affichage d'une carte et des marqueurs: L'utilisateur pourra afficher les annonces sur une carte en fonction de sa position et sélectionner une annonce pour en voir le détail.
- Recherche d'une adresse d'un lieu: L'utilisateur pourra entrer une adresse et afficher sur la carte les annonces autour de cette adresse.
- Sélection d'un marqueur : affichage liste annonces: L'utilisateur, quand il cliquera sur un marker, affichera une liste avec les annonces situées à cette adresse.
- Affichage des infos d'une annonce: En sélectionnant une des annonces de la liste, l'utilisateur pourra voir toutes les infos de l'annonce.
- Propriétaire: se connecter à un compte: Le propriétaire doit se connecter à son compte pour gérer ses annonces
- Propriétaire: créer un compte: Le propriétaire peut passer par l'application pour créer son compte.
- Propriétaire: gérer ses annonces: Le propriétaire peut afficher et gérer ses annonces.
- Propriétaire: ajouter une annonce: Le propriétaire peut créer son annonce.
- Propriétaire: supprimer une annonce: Le propriétaire peut supprimer une annonce.

3)Technologies utilisées

- Base de données (BDD): Pour la base de données, l'application exploite une BDD No-Sql en ligne, sur un serveur distant: Il s'agit de Firebase. En effet ce choix est judicieux car l'API de Firebase est très complète et s'occupe de beaucoup de choses: la sauvegarde en local sur les smartphones, la synchronisation avec la BDD distante, l'ensemble des fonctions pour manipuler la BDD.Mais surtout c'est une API dédiée, entre autres, à Android.

- Authentification: Pour l'authentification nous utilisons une API faite pour fonctionner avec Firebase: FirebaseAuth. En effet, cela nous permet d'avoir une sauvegarde et une gestion des comptes sûres, complètes et de nombreuses possibilitées prédéfinies: Authentification en deux étapes, connections avec des comptes FB, Google ou encore la possibilité de changer de mot de passe.

- Carte et recherche d'adresses: Pour la carte nous utilisons les services Google: Ces services fournissent un fond de carte complet et une interface parfaitement adaptée aux applications Android. Ces services permettent aussi de calculer les coordonnées d'une adresse fournie par l'utilisateur ce qui se révèle utile pour la recherche de lieu où pour l'enregistrement de la position géographique d'une annonce.

- Géolocalisation: On utilisera LocationManager présent de base dans Android.

- Détection et affichage des annonces: Pour afficher et détecter des annonces nous avons besoin d'une API qui permet de gérer des coordonnées mais surtout de détecter des objets (avec des coordonnées) autour d'un lieu: Pour cela nous utiliserons une autre API de Firebase: GeoFire. GeoFire fonctionne donc parfaitement avec notre BDD mais surtout, dans notre cas, il nous permet de détecter en temps réel la position des annonces dans un rayon de N km autour de l'utilisateur ou d'un lieu précis.

- Gestion des listes: Pour gérer les listes nous utiliserons une ListView. En effet les ListView sont conçues pour afficher une liste d'objet à partir de données récupérées depuis une BDD.

- Vues: Pour les vues nous utiliserons les fragments et une seule activité. Tout simplement pour faciliter les transitions mais aussi pour permettre de faire facilement évoluer l'application au niveau de l'interface.

4)Implémentation

a) Authentification et gestion des comptes

Pour l'authentification, le projet utilise la librairie d'authentification de Firebase. Un visiteur peut s'inscrire en fournissant adresse email et mot de passe. Les interactions entre l'application et la base de données Firebase permettent d'effectuer les requêtes suivantes:

- Création d'un compte
    - OK si nouvel email
    - erreur si l'email existe déjà en base de données
- Système d'envoi d'email en cas de mot de passe oublié (prérequis: l'adresse email existe déjà en base de données)
- Connexion à l'application (prérequis: l'adresse email existe déjà en base)
- Déconnexion du compte actif (prérequis: l'utilisateur est connecté)
- Changement de mot de passe (prérequis: l'utilisateur est connecté)
- Suppression du compte (prérequis: l'utilisateur est connecté)

En cas de connexion réussie, la connexion est crédentiale et permet ainsi de garder une connexion permanente même en cas de réouverture de l'application.

b) Cartes et markers

La carte contient un certain nombre d'éléments à implémenter.
Les principaux éléments sont donc:

- L'objet qui représente la carte.
- Le marker de l'utilisateur.
- Les markers des annonces (dans une hashmap).
- Un objet LocationManager pour la géolocalisation.
- Un objet Geofire pour utiliser l'API pour afficher les annonces.
- Un booléen ("search") pour indiquer si on se sert de la position réelle de l'utilisateur où d'une recherche fait par celui-ci pour chercher les annonces.

<u>Affichage d'une carte:</u>

Comme nous l'avons expliqué avant nous utilisons les services Google pour récupérer une carte. On va donc créer et initialiser un objet "gmap" qui représente notre carte.

On va aussi décrire un listener, que l'on va rattacher à notre carte, pour indiquer que si l'on clique sur le marker d'une annonce, on doit récupérer l'ensemble des ids des annonces (qui sont les titres de chaque marker d'annonce) situés à l'endroit du marker, puis passer cette liste d'id au fragment qui affiche une liste d'annonce pour les utilisateurs.

Géolocalisation de l'utilisateur:

On utilisera LocationManager pour chercher la position de l'utilisateur et la mettre à jour en temps réel. Dans les listeners de LocationManager, on indiquera, lors de la détection ou de la mise à jour des coordonnées d'un utilisateur, de créer et de placer sur la carte le marker de l'utilisateur (si le booléen search est à false). Aussi on n'oubliera pas d'appeler la méthode pour rechercher les annonces autour du marker.

Recherche d'une adresse:

Pour la recherche d'une adresse, dès que l'utilisateur appuiera sur le bouton rechercher, on utilisera une fonction de GoogleService pour trouver les coordonnées de cette adresse et positionner le marker utilisateur dessus. On appellera la méthode pour rechercher les annonces autour du marker et on passera le booléen "search" à true (ce qui évitera de recentrer le marker sur la position de l'utilisateur si sa géolocalisation change).

On dispose d'un bouton à côté qui met fin à cette recherche et repositionne le marker sur la position réelle de l'utilisateur.

Affichage d'une annonce:

Pour afficher une annonce on va lancer , grâce à GeoFire, une geoquery qui va donc chercher dans un rayon autour d'un point (ici les coordonnées de notre marker utilisateur) les annonces, on va donc implémenter le listener de cette GeoQuery:

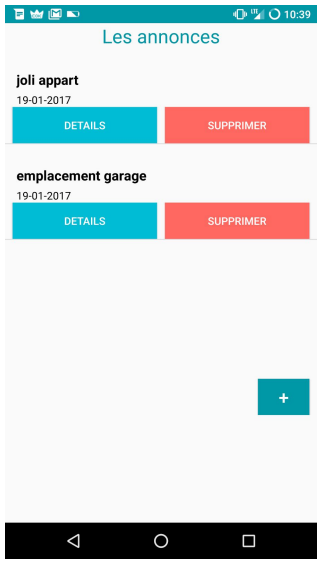
- En cas de détection d'un nouvel élément: on crée et ajoute ce marker dans la liste des markers des annonces et on l'affiche sur la carte
- En cas de mise-à-jour d'un élément: on supprime l'ancien marker et on en refait un nouveau avec les nouvelles coordonnées
- En cas de disparition d'un élément: on supprime le marker sur la carte et dans la liste

c) Listes et affichage/gestion des annonces

Lister les annonces

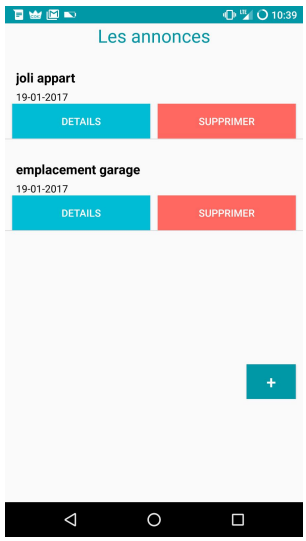
Lorsque l'utilisateur est connecté, il peut lister les annonces qu'il a enregistré dans la section de son "profil", il peut ainsi afficher les détails de l'annonce ou la supprimer (expliqué plus bas).

Lorsqu'un utilisateur n'est pas forcément connecté, il peut sélectionner une annonce (représentée par un marqueur sur la carte) pour accéder au détail.



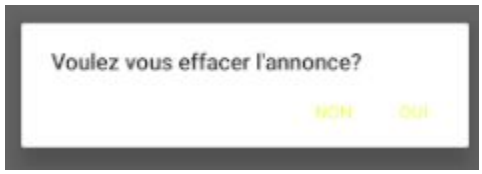
Ajouter une annonce

Lorsque l'utilisateur est connecté, au même emplacement que la liste de ses annonces, il peut créer une nouvelle annonce en fournissant divers détails et une adresse postale valide. Au moment de sa validation et grâce à Geofire, la carte se verra dotée d'un nouveau marqueur visible à l'emplacement correspondant à l'adresse postale fournie. Un utilisateur distant visionnant la carte en direct verra apparaître automatiquement ce marqueur.



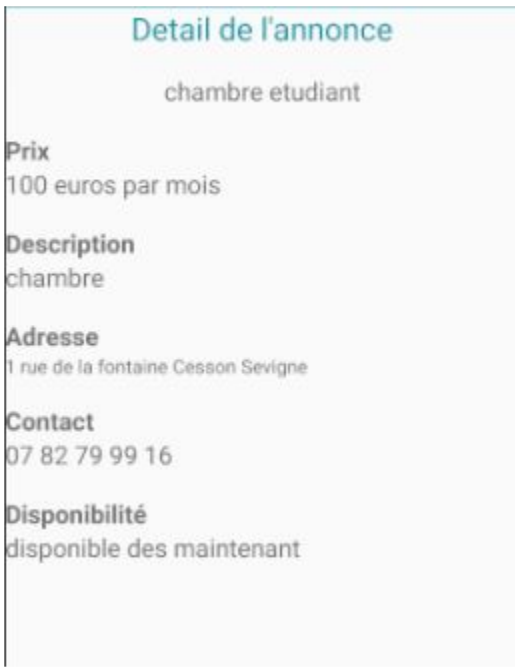
Supprimer une annonce

Quand l'utilisateur clique sur le bouton de suppression, un message de confirmation est affiché pour s'assurer que le propriétaire veut vraiment effacer son annonce. L'annonce est supprimée de la base de données et la liste des annonces est mise à jour.



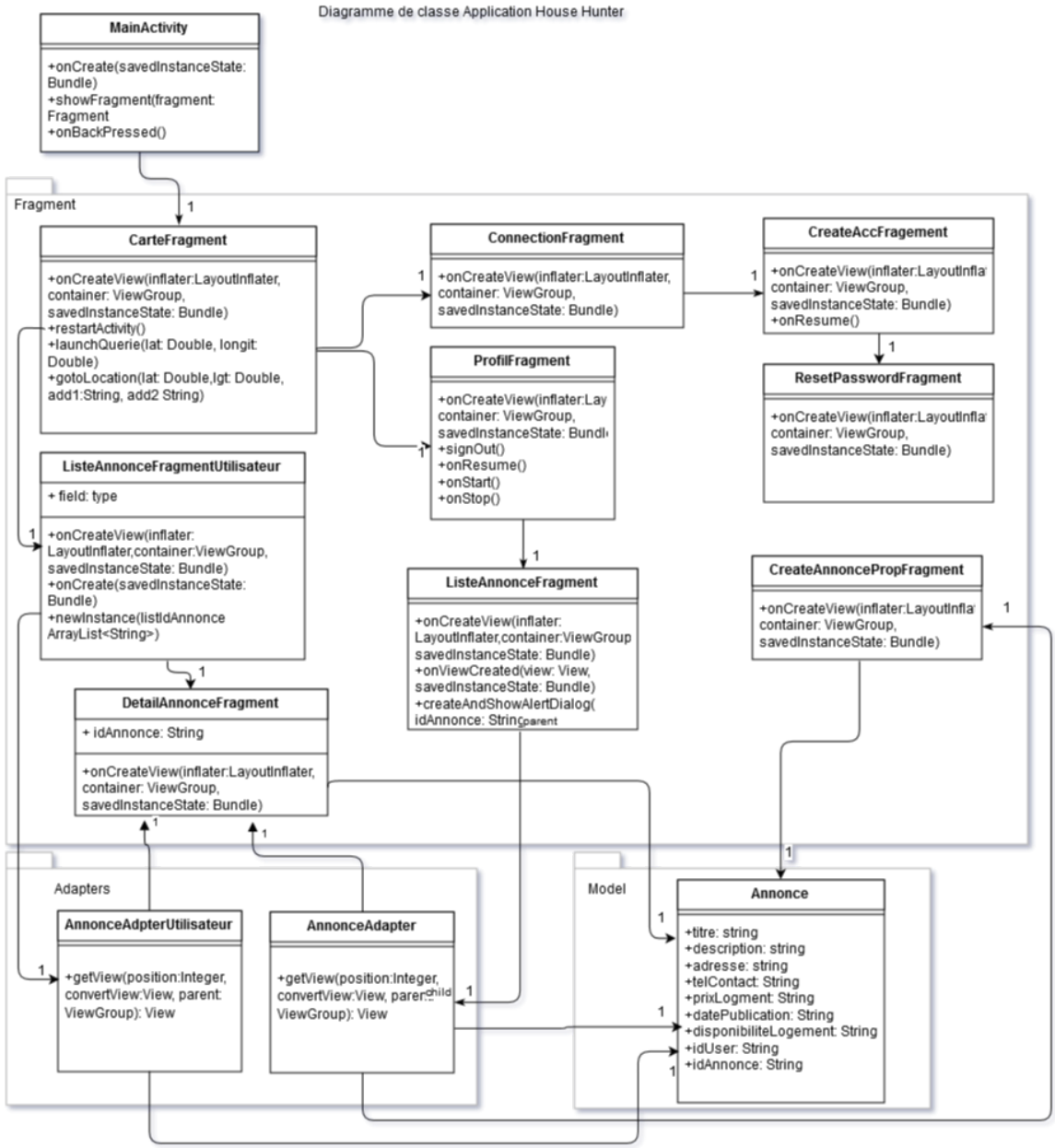
Afficher les détails de l'annonce

L’utilisateur peut afficher les détails de l'annonce en cliquant sur le bouton “Details” qui est sur l'élément de la liste qu’il veut consulter. Pour chaque élément de la liste des annonces, il y a un champ caché qui contient l’id de l’annonce, ce champ caché permet de passer en paramètre l’id de l’annonce pour qu’on puisse récupérer de la base de données les informations correspondantes de l’annonce sélectionnée.



5)Architecture

Pour l'architecture de la solution nous avons fait au plus simple: Nous avons créé une seule et unique activité qui va servir de base pour charger tous les différents fragments. A chaque fragment correspond une vue de l'application. Nous aurons aussi deux classes Adapters pour décrire le contenu de chaque ligne pour les deux listView. Enfin nous avons un modèle pour décrire les informations, éléments qui composent une annonce.
Voici le diagramme de classe de notre application:
On a simplifié en évitant de tout lister (Certaines variables et dépendances externe pour éviter un schéma trop lourd). Les associations représentent plus ici une création d'instance dans une méthode, plutôt que l'utilisation d'une variable qui représente la classe



6)Futures évolutions

L’application permet déjà de répondre au besoin principal mais il y a de nombreuses améliorations possible:

- La possibilité de régler le rayon de recherche des annonces sur la carte.
- La possibilité de filtrer les annonces à afficher.
- La possibilité de contacter un propriétaire d’une annonce directement par le biais de l’application.
- La possibilité de créer un compte pour l’utilisateur pour par exemple indiquer ses annonces favorites.
- La possibilité d’ajouter une photo à l'annonce (ex. photo d’une chambre).
- La possibilité de valider son inscription par mail ou par SMS.