Badr Herress
Guillaume Velay
Master 1 AIGLE, Université de Montpellier

Projet programmation mobile

EasyChat

Année universitaire 2015 - 2016

Table des matières

1 Présentation de l'application

L'application que nous avons choisi de développer, dans le cadre du projet de l'UE programmation mobile, est une application mettant en place un salon de chat permettant à ses utilisateurs de communiquer entre eux.
Elle permet dans un premier temps à l'utilisateur de s'inscrire en fournissant un nom d'utilisateur et un mot de passe, ce qui lui permettra de s'authentifier par la suite et avoir un pseudo associé pour pouvoir être reconnue par les autres utilisateurs. Chaque utilisateur possède un profil contenant quelques informations sur lui.
Après que l'utilisateur se soit inscris et authentifié, il accède au salon de chat ou il pourra échanger avec les autres utilisateurs. Ces échanges peuvent avoir la forme de messages texte, ou de photos. En effet l'utilisateur peut choisir entre écrire un message au clavier, prendre une photo avec l'appareil photo du téléphone, récupérer une photo de la galerie, et les envoyés aux autres utilisateurs.
L'utilisateur a aussi la possibilité de connaitre l'état de connection des autres utilisateurs à l'application ainsi qu'accéder à leur profil.

2 Les fonctionnalités développées pour l'application

Afin de rendre l'application la plus agréable possible, nous avons ajouter quelques fonctionnalités à celle-ci. Voici les plus importantes d'entres elles :

- **Service d'authentification :** L'application permet aux utilisateur de s'inscrire une fois et de se connecter par la suite avec les mêmes nom d'utilisateur et mot de passe.
- **Profil utilisateur :** Chaque utilisateur possède un profil. Dans ce dernier on trouve une photo, le pseudo, ainsi qu'une description. Les utilisateurs peuvent accéder aux profils des autres utilisateurs.
- **Edition du profil :** Les utilisateurs peuvent éditer leur profil. Ils ont la possibilité de modifier leur photo, pseudo, mot de passe et description.
- **Gestion des états de connexions :** L'application gère différents états de connexion. Connécté et en première tâche associé à un voyant vert dans la liste des utilisateurs, c'est l'état ou l'utilisateur est actif sur l'application. Connécté et en fond de tâche associé à un voyant vert pomme, représente l'état où l'utilisateur était sur l'application et que le téléphone se met en veille. Le dernier état représenté par un voyant rouge correspond à déconnécté et il correspond à l'état quand l'utilisateur n'est plus sur l'application.
- **Désinscription :** L'application permet aux utilisateurs de se désinscrire.
- **Récupération des messages antérieurs :** Lors de la connexion d'un utilisateur, celui-ci récupère tous les derniers messages échangés, même lors de son absence, ce qui lui permet de suivre les conversations.
- **Echanges de photos :** Les utilisateurs peuvent échanger des photos entres eux, soit en prenant la photo sur place à l'aide de l'appareil photo de l'appareil, ou bien en récupérant une photo de la galerie du téléphone.
- **Sauvegarde de photos :** Un utilisateur à la possibilité d'enregistrer une photo reçu.
- **L'envoie de wizz :** En secouant son téléphone, un utilisateur peut envoyer un wizz aux autres utilisateurs.
- **Un système de notification :** A chaque message reçu, sous toutes les formes (textes, photos, wizz), l'application envoie une notification à l'utilisateur. Si l'appareil de l'utilisateur est en mode vibreur alors ça sera une vibration, que nous avons personnalisé, si silencieux alors juste une notification sans son ni vibration, et sinon, quand le téléphone est en mode son, une notification sonore se déclenche correspondant au son associé aux

notifications de l'appareil.
— **Gestion des dates de messages :** L'application enregistre les dates des messages échangés, et permet aux utilisateurs de savoir quand un message a été envoyé en cliquant dessus.
— **Langues :** L'application est disponible en deux langues, le français et l'anglais.

3 Les composants de l'application

3.1 Les vues

Nous avons une vue correspondante à l'activité qui se charge de la connection et l'inscription au chat. Dès que l'utilisateur est connecté, il est amené sur l'activité principale qui se compose de quatre fragments où chacun est associé à une vue. La première correspond au profil utilisateur, la seconde au chat, la troisième aux utilisateurs et la dernière aux paramètres de l'application.

3.2 Les composants

Tout d'abord, nous utilisons un *Service* qui communique avec le reste de l'application et le serveur.

Nous utilisons deux *Activity* l'une pour la connection au chat, l'autre pour l'utilisation du chat décrit précédemment. La seconde activité contient quatre $Fragment$ contenu dans un objet $ViewPager$.

Nous utilisons des threads : $AsyncTask$ à l'intérieur du service pour plus de rapidité au traitement des requêtes de l'utilisateur.

Pour la réception des messages provenant du serveur au reste de l'application, l'activité principale et certain fragments possède des $BroadcastReceiver$ qui sont enregistré ou désenregistré selon que l'activité est en pause ou active.

Lorsqu'un utilisateur envoit un message dans le chat, les autres sont automatiquement notifié par un son, une vibration et une barre de notification qui s'affiche en haut de l'écran, l'objet utilisé est un $NotificationCompat.Builder$.

Quand un utilisateur souhaite modifier ses informations de profil, il doit passer des $DialogFragment$, nous les utilisons aussi pour afficher la page de profil des autres utilisateur et pour enregistrer une image.

Afin de créer une liste qui affiche les messages du chat, les utilisateurs de l'application ou les paramètres nous utilisons des $ListView$ auquel sont attaché des adaptateurs $ArrayAdapter$ pour faire le lien entre les données et la vue.

Un utilisateur peut envoyer un "wizz" dans le chat qui s'effectue par un agitement du téléphone, nous implémentons cela en utilisant le capteur d'accélération s'il existe, dans le code on passe donc par un objet $Sensor$ et $SensorManager$.

3.3 Les ressources

En ce qui concerne les layouts, nous utilisons des objets natifs android tel que $LinearLayout$, $TextView$, $ImageView$, $EditText$ avec leurs propriétés. Nous avons essayé de réutiliser au maximum dans les layouts d'autres ressources tel que les chaines de caractères, les couleurs, les dimensions. Nous avons défini des couleurs, des dimensions et des strings.

Afin de gérer la langue anglaise (par défaut) et française nous avons créé deux fichiers de ressource.

Nous avons également une ressource contenant deux animations qui sont utilisé pour faire apparaitre et disparaitre la date d'un message lorsque l'on clique dessus.

Pour les ressources de type drawable, nous avons codé directement en xml quelques fonds d'écran qui sont utilisé par exemple pour contenir le message textuel dans le chat, les messages à gauche sont oranges alors qu'à droite ils sont bleus. Le voyant qui indique l'état de connection d'un utilisateur est aussi une ressource xml que nous avons codé.

Android nous aide en ce qui concerne les icones, ce sont les $Material\ Icons$ de google, elles fournissent d'ailleurs les images pour tous les types d'écran.

Enfin, nous avons créer un style comme ressource pour l'esthétique de l'application.

3.4 Le Manifest

Le manifest de notre application comporte la définition de nos deux activités, nous avons défini que les activités pouvaient seulement s'afficher en mode portrait car nous avons manqué de temps pour définir leur layout en mode paysage. L'activité principale comporte des $intent-filter$ pour permettre aux $BroacastReceiver$ de filtrer les intents selon leur action.

Au niveau des permissions $uses-permission$, nous avons donné le droit d'acces à internet, le droit d'écrire et lire sur le stockage externe du téléphone et le droit de vibrer.

Pour les caractéristiques $uses-feature$, l'application peut ou pas nécessairement utiliser la caméra et idem pour l'accéléromètre.

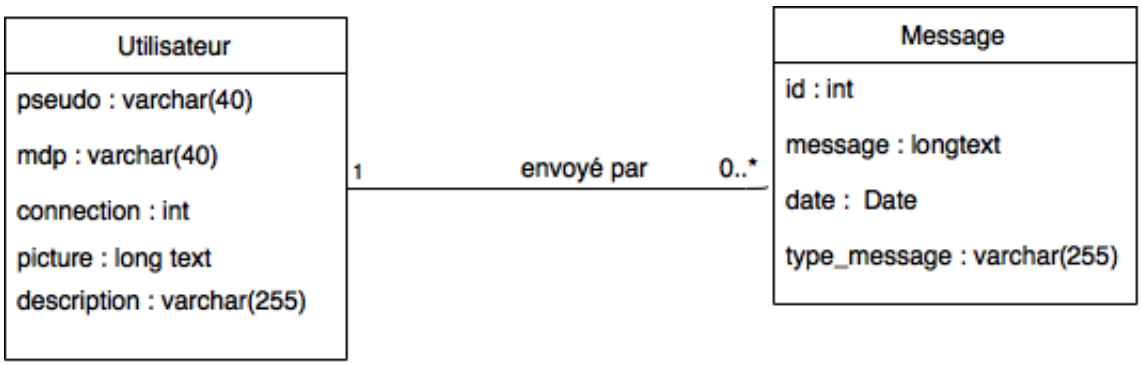
Le manifest déclare également notre service.

4 Communication client serveur

Pour réaliser toutes ces fonctionnalités, nous avions déjà besoin de concevoir les composants de notre application. Tout d'abord une base de données, nous permettant de stocker les données nécéssaire à l'utilisation de l'application, ensuite un serveur qui va nous permettre de récupérer les données et de les envoyer côté client. Ensuite l'interface utilisateur qui va permettre à l'utilisateur de naviguer dans l'application.

4.1 La base de données MySQL

La base de données, utilisée par l'application, a été réalisée avec MySQL. Elle contient 2 tables, la première "Utilisateur", qui contient les informations concernant de ce dernier, et la seconde "Message" qui elle contient les informations sur les messages échangés.



4.2 Le serveur

Nous avons créer un serveur qui établi la connection entre l'application et la base de données. Celui-ci a été implémenté à l'aide de $nodejs$. Le serveur permet de gérer les requêtes utilisateurs

comme l'envoie d'un message, la modification des informations de l'utilisateur, la connection, la déconnection, l'inscription, la désinscription. Le serveur et la base de données se trouve sur un de nos ordinateurs personnels.

4.3 Envoie - Réception des requêtes

Toutes les requêtes de l'utilisateur sont envoyées au serveur grâce à un service créé dans l'application que nous avons nommé *SocketService*. Ce service recoit également toutes les réponses du serveur et se charge de les distribuer dans l'application à l'aide de *broadcast*. Le service contient la socket de communication entre l'application et le serveur. Le serveur envoit et reçoit les messages à l'aide d'une socket et choisi d'envoyer une réponse soit à tous les utilisateurs sauf à l'utilisateur qui a effectué la requête, soit seulement à ce dernier.

5 Améliorations possibles

Nous avons pensé à quelques améliorations que nous pouvons apporter à notre application.

Tout d'abord une base de données interne qui permettrai de stocker les informations des utilisateurs, les photos, et des messages, dans le but d'alléger la communication entre le client et le serveur.

Ensuite, un système de gestion des messages antérieurs, qui permet de charger au fur et à mesure les anciens messages, au lieu de tout charger à l'ouverture de l'activité principale.

Une autre amélioration est la gestion de l'affichage de l'application lors de la rotation de l'écran.