BURGUBURU Côme
JORNET Benjamin

Dev App Mobile
MyChat
Architecture technique

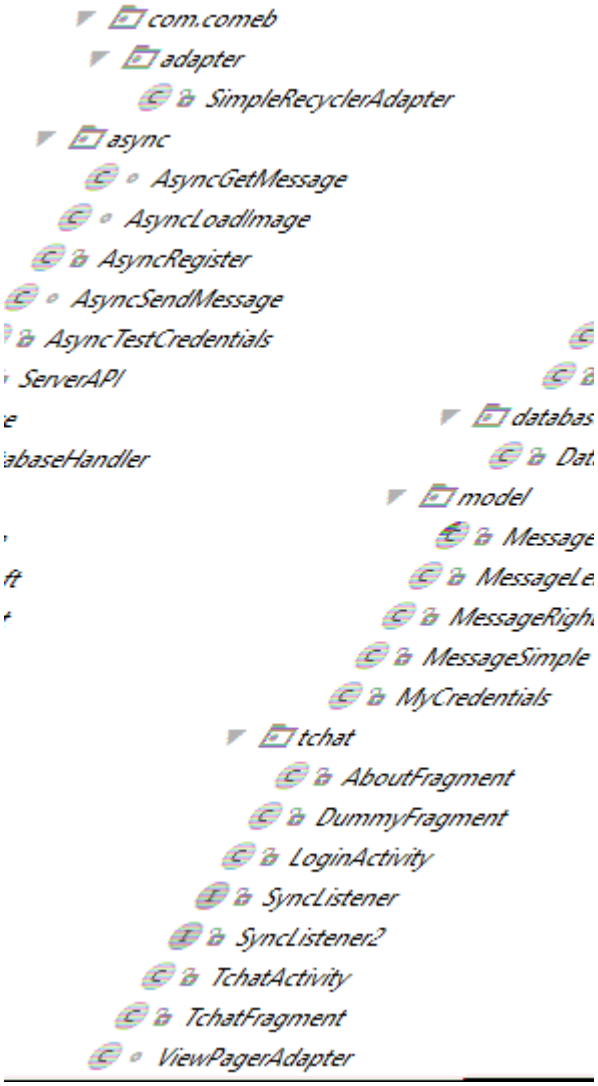
Introduction

MyChat est une application Android qui permet de communiquer entre différents utilisateurs grâce à du texte et des images.

Packages

Le code s'organise sous forme de cinq packages :

- adapter : adapter de cardview
- async : AsyncTask et API de communication
- database : databaseHandler
- model : classe Messages et ses dérivées
- tchat : Activités et fragments



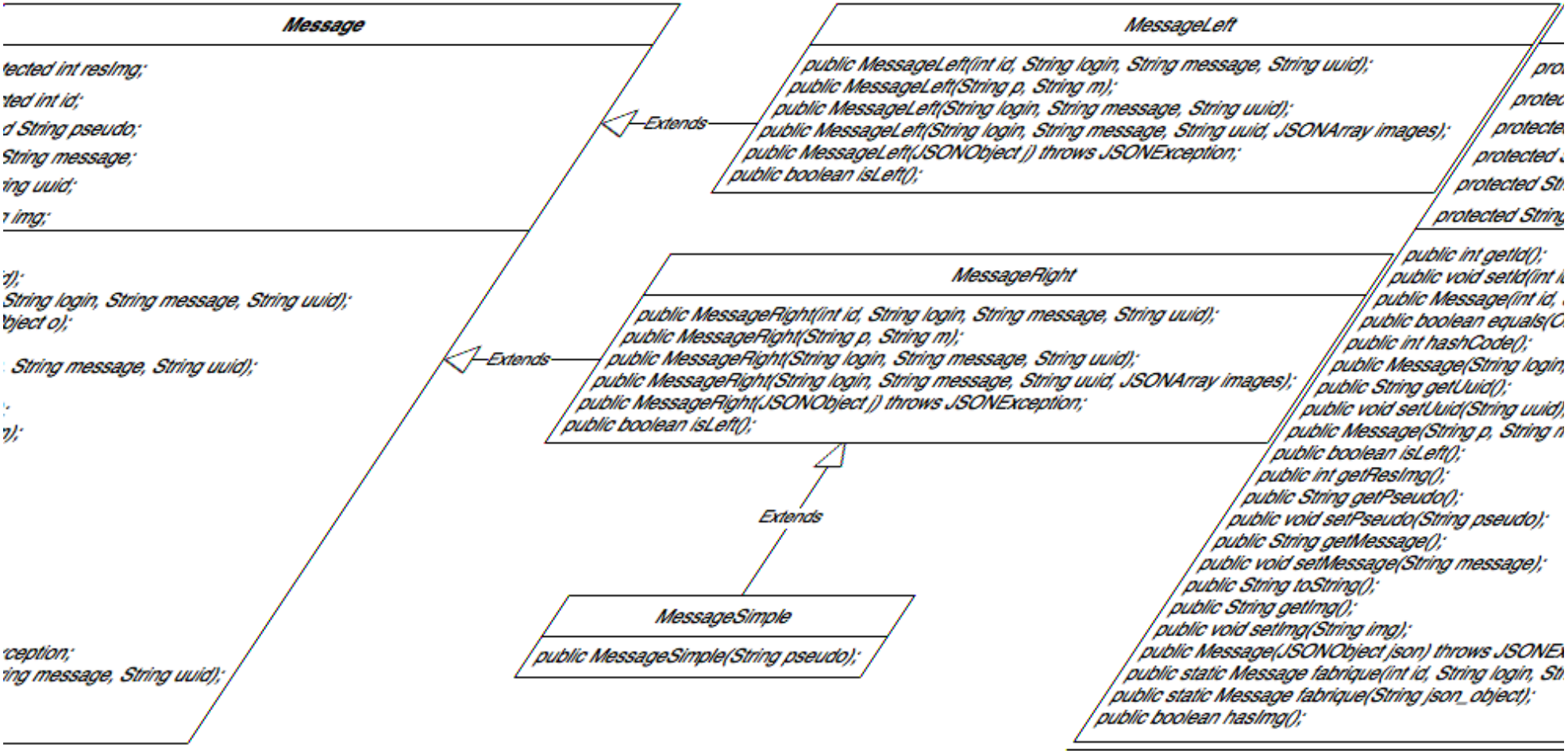
Model

Le modèle est composé d'une classe abstraite Message qui a pour attributs :

- int resImg
- int id
- String pseudo
- String message
- String uuid
- String img

On définit trois types de message :

- « MessageLeft » avec un avatar bleu situé à gauche pour les messages reçus
- « MessageRight » avec un avatar rouge situé à droite pour les messages envoyés par l'utilisateur
- « MessageSimple » un avatar rouge a gauche sans message pour les contacts

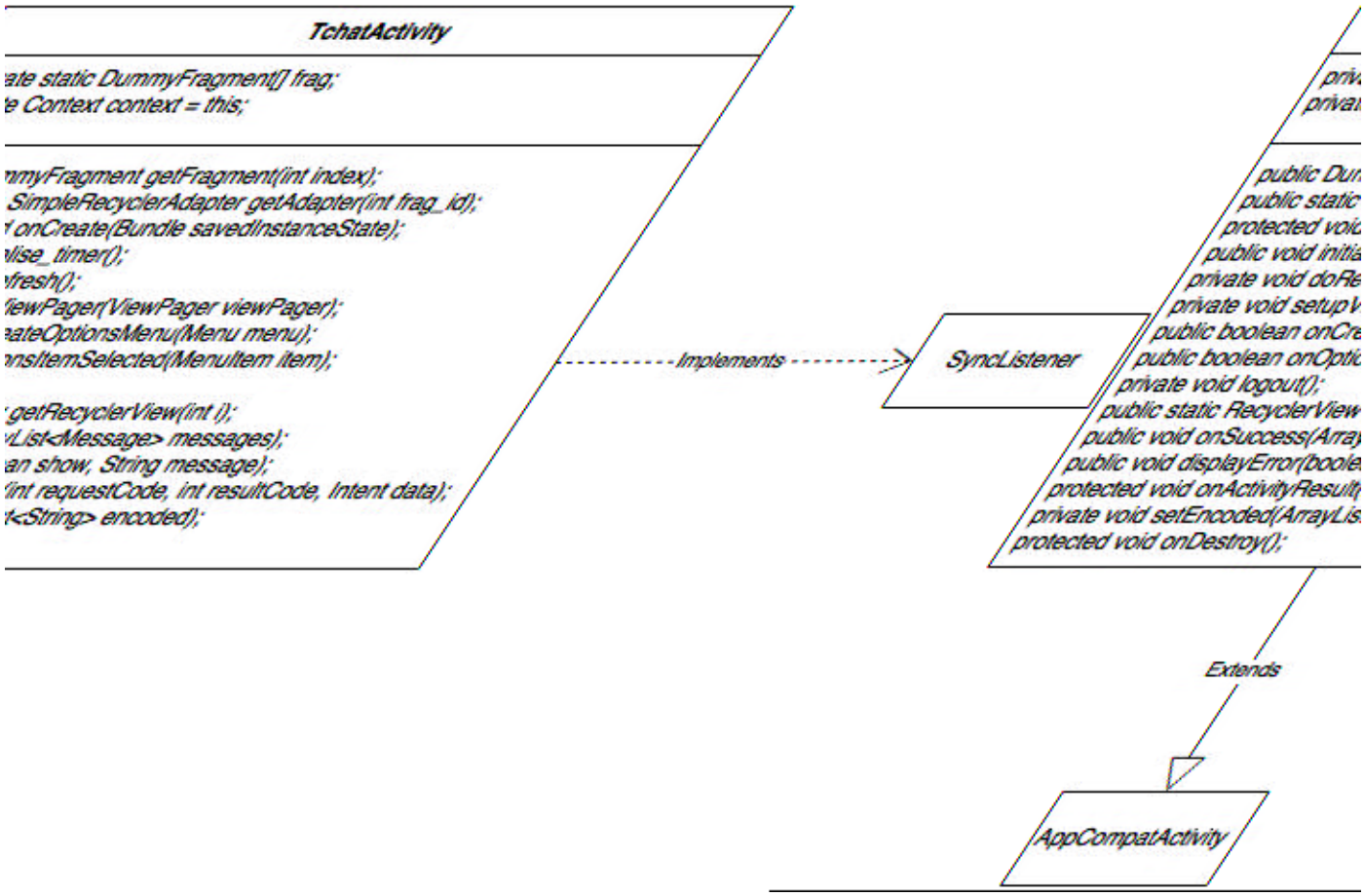

Message
tected int resImg;
ted int id;
d String pseudo;
String message;
ing uuid;
img;
);
String login, String message, String uuid);
bject o);
String message, String uuid);
);
ception;
ing message, String uuid);
Extends
MessageLeft
public MessageLeft(int id, String login, String message, String uuid);
public MessageLeft(String p, String m);
public MessageLeft(String login, String message, String uuid);
public MessageLeft(String login, String message, String uuid, JSONArray images);
public MessageLeft(JSONObject j) throws JSONException;
public boolean isLeft();
Extends
MessageRight
public MessageRight(int id, String login, String message, String uuid);
public MessageRight(String p, String m);
public MessageRight(String login, String message, String uuid);
public MessageRight(String login, String message, String uuid, JSONArray images);
public MessageRight(JSONObject j) throws JSONException;
public boolean isLeft();
Extends
MessageSimple
public MessageSimple(String pseudo);
public int getId();
public void setId(int
public Message(int id,
public boolean equals(O
public int hashCode();
public Message(String login,
public String getUuid();
public void setUuid(String uuid)
public Message(String p, String
public boolean isLeft();
public int getResImg();
public String getPseudo();
public void setPseudo(String pseudo);
public String getMessage();
public void setMessage(String message);
public String toString();
public String getImg();
public void setImg(String img);
public Message(JSONObject json) throws JSONE
public static Message fabrique(int id, String login, St
public static Message fabrique(String json_object);
public boolean hasImg();

Activity

Nous avons 2 types d'activité : LoginActivity et TchatActivity.

TchatActivity

L'activité TchatActivity correspond à l'activité principale de l'application, c'est à dire celle des dialogues entre les différents utilisateurs.



L'activité TchatActivity est composé d'un viewPager qui est composé de trois fragments.

Le fragment Contacts liste de façon unique les utilisateurs.

Le fragment Tchat affiche les messages et permet de les composés.

Le fragment About affiche les informations relative à l'application.

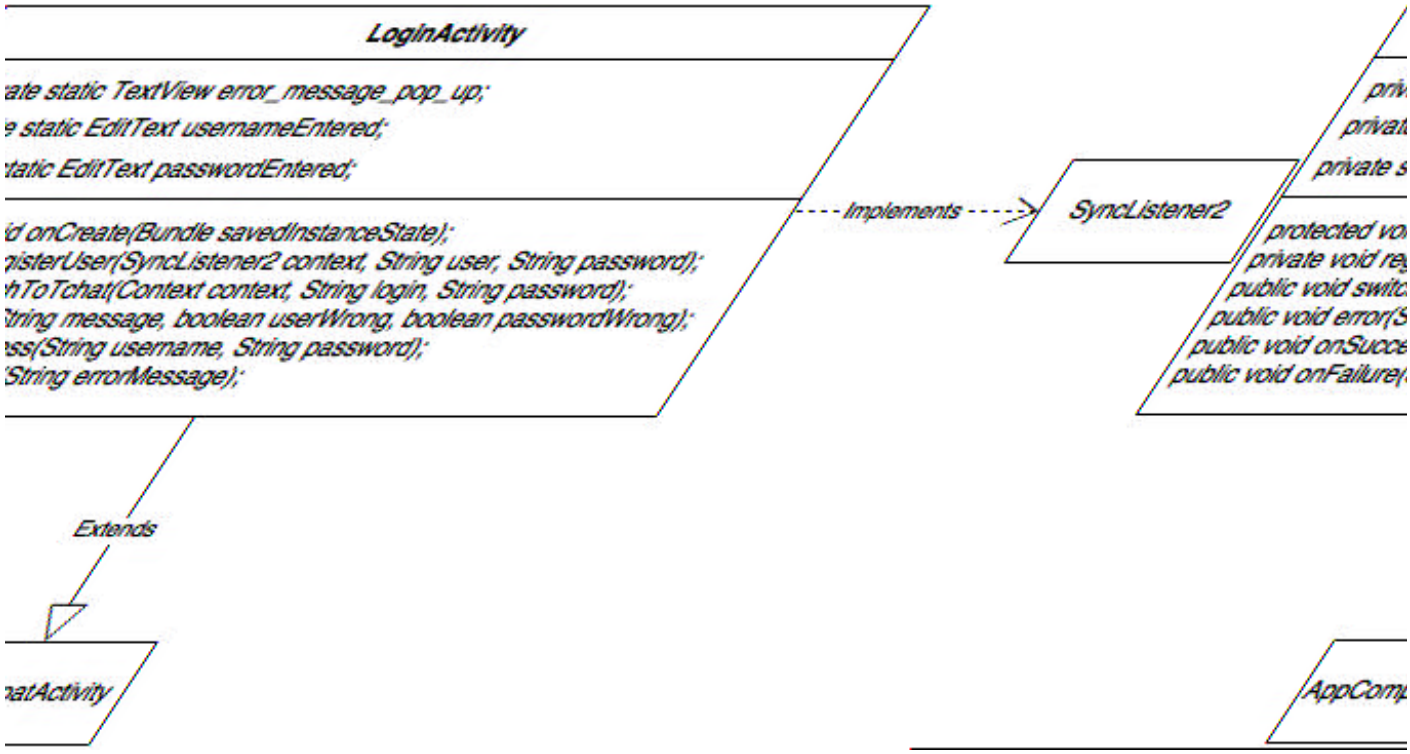
Cette activité est composée d'un viewPager qui est composé de trois fragments :

- Le fragment Contacts liste de façon unique les utilisateurs.
- Le fragment Tchat affiche les messages et permet de les composés.
- Le fragment About affiche les informations relative à l'application.

Il est aussi possible de se déconnecter, et donc de retourner à LoginActivity via un bouton.

LoginActivity

LoginActivity est la première activité que l'utilisateur va rencontrer en utilisant l'application. Cette dernière lui permettra de se connecter ou de créer un compte afin de pouvoir communiquer avec ses contacts. Une fois que l'utilisateur s'est connecté une première fois, il ne reviendra pas sur cette page par défaut, à moins qu'il clique volontairement sur le bouton en haut à droite dans la TchatActivity.



LoginActivity implémente SyncListener2, que nous détaillerons plus bas.

Cette activité est composée de 3 attributs :

- TextView error_message_pop_up, est un message qui s'affiche en cas d'impossibilité de connexion,

- EditText usernameEntered, correspond au formulaire où l'utilisateur rentre son identifiant,

- EditText passwordEntered, correspond au formulaire où l'utilisateur rentre son mot de passe.

Deux action possible se connecter ou créer le compte.

Async

Les requêtes vers le serveur web sont encapsulé dans la classe ServerAPI qui a pour rôle de générer les URL, exécuter et arrêter les taches asynchrones.

- AsyncTestConnexion envoie au serveur un JSON avec le login et le mot de passe. Si la réponse est valide l'activité TchatActivity est lancé sinon un message d'erreur s'affiche.
- AsyncRegister enregistre un nouvel utilisateur et gère les erreurs si le login n'est pas disponible
- AsyncGetMessage charge la liste des messages et rafraîchit la recycleView
- AsyncLoadImage charge de façon asynchrone les images et les affiches dans les imageView
- AsyncSendMessage récupère le message et l'image et les envoie au serveur l'erreur 413 causé par une image trop lourde est prise en compte et affiche un avertissement.

La tache AsyncGetMessage est appelé toutes les 5 secondes,

Database

Ce package contient la classe DatabaseHandler.

Nous utiliserons pour cela des requêtes SQL via SQLite.

Le pattern Singleton sera utilisé pour être sûr d’instancier une

seule fois un objet du type DatabaseHandler.

Cette classe aura pour attributs :

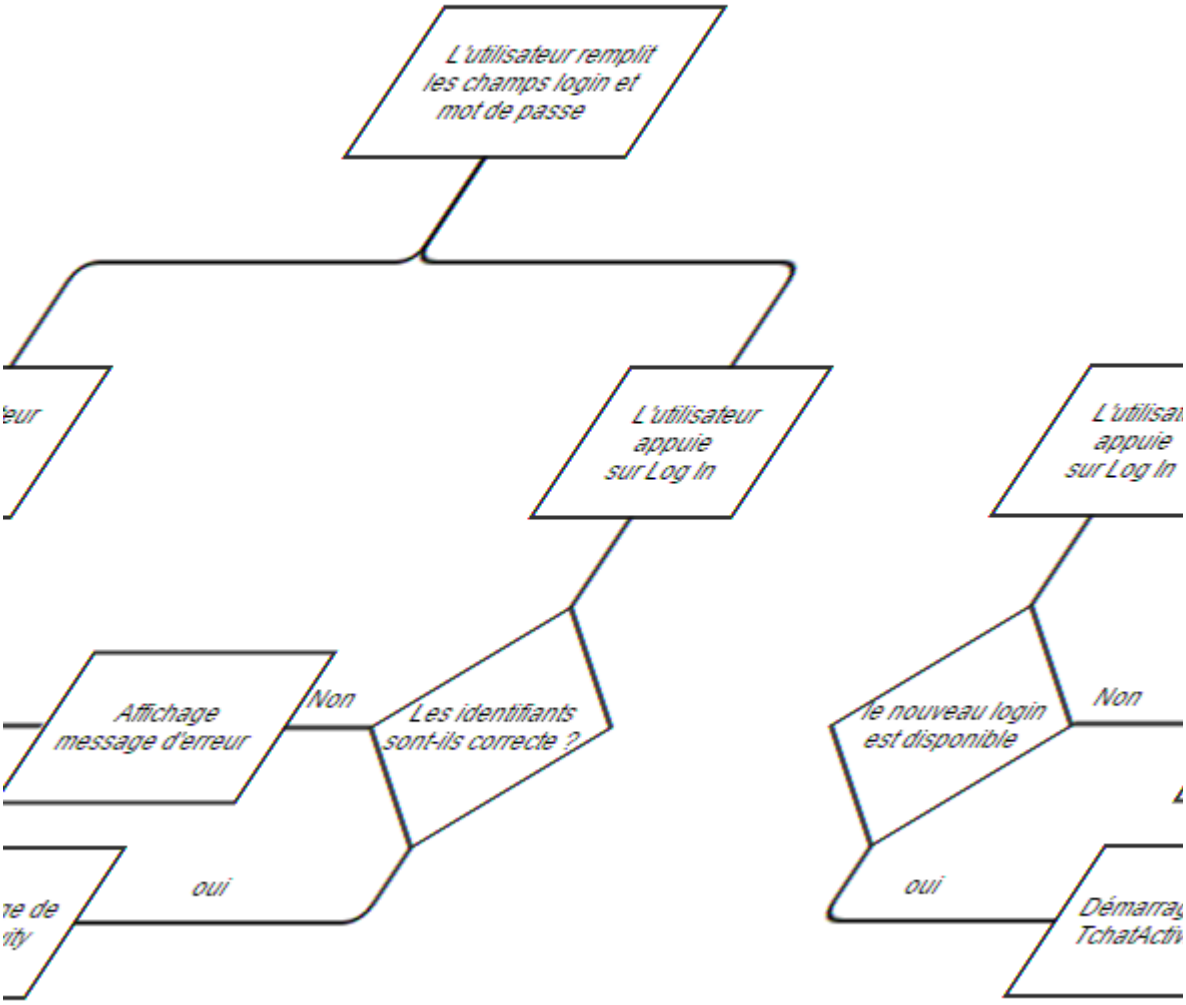
- private static DatabaseHandler singleton;
- private static final int DATABASE_VERSION = 2;
- private static final String DATABASE_NAME = "messagesManager";
- private static final String TABLE_MESSAGES = "messages";
- private static final String KEY_IMG = "img";
- private static final String KEY_ID = "id";
- private static final String KEY_NAME = "name";
- private static final String KEY_MESSAGE = "message";

Dans cette classe, nous avons à disposition plusieurs méthodes :

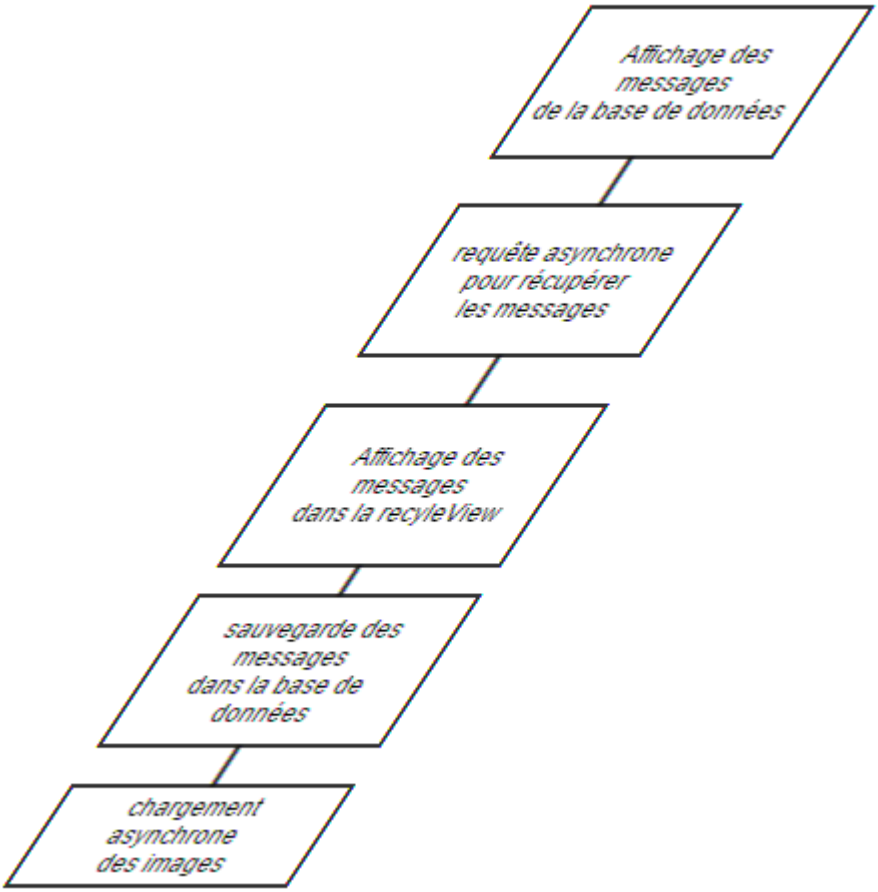
- onCreate(), pour créer une nouvelle discussion ainsi que de nouvelles tables,
- addMessage(), pour ajouter un message à une discussion,
- getMessage(), pour récupérer un message via l'id,
- updateMessage(), afin de modifier un message en cours,
- deleteMessage(), pour supprimer un message,
- getMessagesCount(), pour compter le nombre de messages,
- onUpgrade(), à appeler dans le cas d'un changement de version.

Cette base de données est appelée dans les cas de connexion, d'inscription, d'envoi et de consultations de messages.

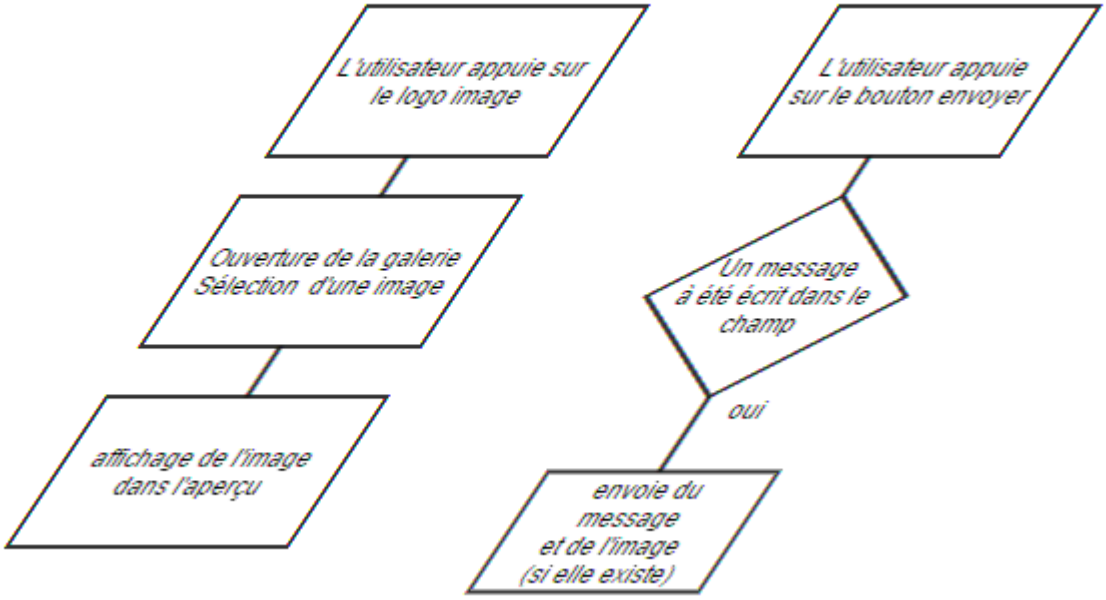
Schéma d'interaction



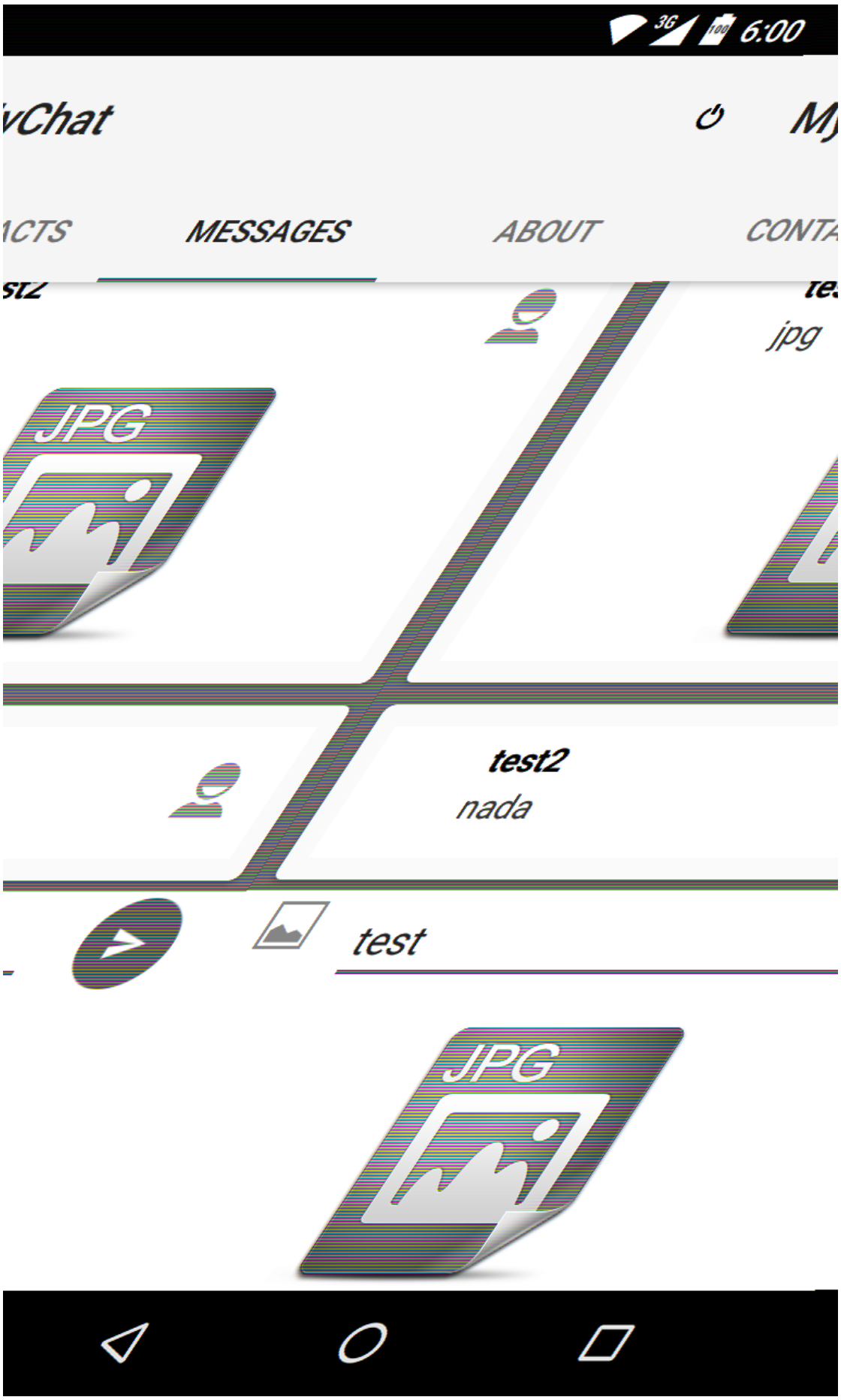
Connexion ou enregistrement de l'utilisateur



réception des messages (toutes les 5 secondes)



envoie d'un message


6:00
vChat
MESSAGES
ABOUT
jpg
JPG
test2
nada
test
JPG