Developer documentation

How does the build process work

First you need the Secrets file containing the applications passwords for databases and APIs, to get this please contact: vilddjur@chalmers.it.
You build the program using gradle, to run the program as a developer you need to authenticate vs. Facebook using a key hash. More information about facebooks keyhashes here: https://developers.facebook.com/docs/android/getting-started, however there is an option in this version that allows you to acess the app via the “developer login” button, this will log you in to the application without the need for a keyhash from facebook. This is only available for developers and will not be a part of the final application.

What major parts / components are there in the application

There are 3 major parts to the application;
The first one is the MainActivity, this keeps track of all the fragments used to display the application data, these are; the matchlist-fragment (for displaying people you’ve matched with), the chat fragment(for chatting with another user) and the matchscreen fragment (for finding new people to talk to). The MainActivity also has references to the data handling classes, such as the DataHandler class.

The second large part of the application is the messaging feature. This program uses Sinch for the instant messaging, as well as MySQL for saving the conversation history. When the user presses the send button the message is added to the adapter (and list). This sends, using a listener, to the MessageService which in return adds to the history in the database and sends to Sinch. When the user receives a message this is simply added to the adapter.

The third big part of the program is the bridge between the backend and the frontend, as well as the bridge between the application and the facebook SDK, this part (the databasehandler and facebooklogin) translates MySQL code into java objects, retrieves data from the backend and connects the app to facebook via the facebook integrated login that is packaged with the facebook sdk.

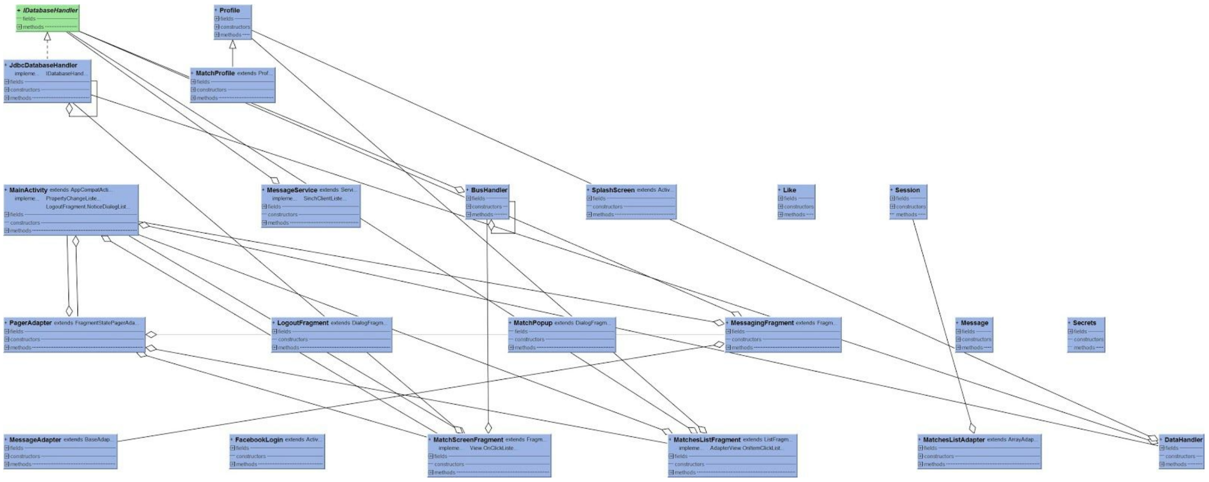
Design decisions (such as API level, etc.)

We chose to use android version 21 (5.1) because we needed some of the functions that were added in this revision, we could therefore not choose an earlier release. And since this is not the latest version of android we had a larger device coverage than we would have had if we chose the latest release.

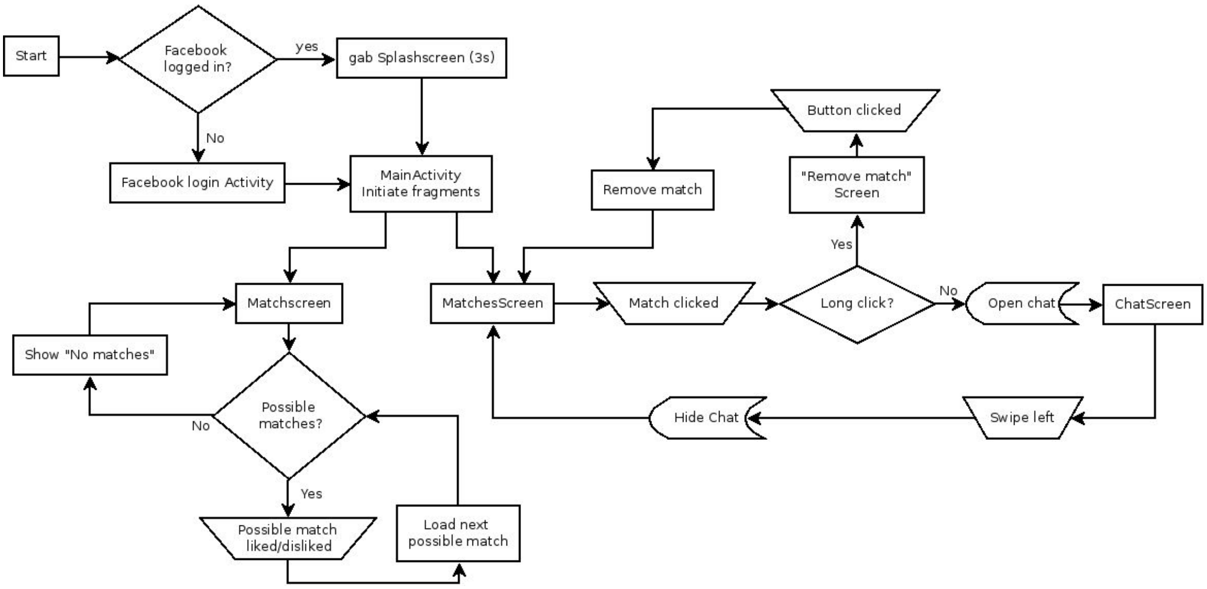
We also chose to separate all the raw data files (models) to accommodate the MVC pattern we also chose to separate the handlers from the activities, this is to separate all the data handling from the actual application and to try to split the application into reusable modules.

The navigation structure is a tabbed structure from the android design library, we chose this because it met our requirements and was recommended by the android developer team.

UML


+ IDatabaseHandler
+ Profile
+ JdbcDatabaseHandler impleme... IDatabaseHand...
+ MatchProfile extends Prof...
+ MainActivity extends AppCompatActi... impleme... PropertyChangeListe... LogoutFragment.NoticeDialogList...
+ MessageService extends Servi... impleme... SinchClientListe...
+ BusHandler
+ SplashScreen extends Activ...
+ Like
+ Session
+ PagerAdapter extends FragmentStatePagerAda...
+ LogoutFragment extends DialogFragm...
+ MatchPopup extends DialogFragm...
+ MessagingFragment extends Fragm...
+ Message
+ Secrets
+ MessageAdapter extends BaseAdap...
+ FacebookLogin extends Activ...
+ MatchScreenFragment extends Fragm... impleme... View.OnClickListe...
+ MatchesListFragment extends ListFragm... impleme... AdapterView.OnItemClickList...
+ MatchesListAdapter extends ArrayAdap...
+ DataHandler
fields
constructors
methods

Flowchart



Protocol (client/server)

The application uses TCP to talk to the external MySQL server through the JdbcDatabaseHandler for saving and fetching users, likes, sessions, and conversations. Sinch is an external service which we use for sending and receiving messages instantly which are also stored on the MySQL database.

Database

The following tables and columns can be found
t_users
user_id - Auto increment row id
name - Name of the user

fb_id - facebook id
interests - A String object with the users facebook likes

t_likes
id - Auto increment row id
origin_id - the user id of the user clicking like
like_id - the user id of the liked person
like_name - the name of the liked person

t_dislikes
id - Auto increment row id
origin_id - the user id of the user clicking dislike
like_id - the user id of the disliked person
like_name - the name of the disliked person

t_messages
message_id - row id
sender_id - user id of the sender
reciever_id - user id of the receiver
message - message payload
sinch_id - id for sinch service

t_sessions
session_id - row id
user_id - user id of a person logged in on a bus network.
dgw - the bus dgw
timestamp - timestamp when connecting to bus network. This can later be used to clear old sessions with server scripts.

t_busses
buss_id - row id
system_id - the bus system ID
dgw - the bus dgw

v_matches
This is a view of all users who have liked another user that likes them back. This view is used to get all users you are able to chat with.

External dependencies

We have used three external dependencies (not counting the android design library that was used to design the framework and gradle). These three are the facebook SDK, Flipboard bottomsheet and Sinch.

The facebook SDK is used to connect the app to facebook, so that users can log in via their facebook account, from there the app retrieves the user's name, and likes. These are then stored in the backend, to be used to generate profiles corresponding to each user (these are the classes that will represent the person).

The flipboard bottomsheet package is used when you long-click on a profile you’ve matched with, when you do this a “sheet” is expanded from the bottom of the app,dimming the rest of the app, this sheet contains a delete button to remove the specific person.

Sinch is used for the instant messaging of the application, Sinch is free while we have less than 1000 monthly users. With more resources the application could simply use the database and some PHP to listen to changes in the database.