DAT255

Prototype Design document

Group: Zweigbergk

Product: SpeedSwede Android chat app

Team members: Andreas Carlsson, Kevin Chen Trieu, Petros Debesay, Felix Engelbrektsson, Jonathan Krän, Robin Lilius-Lundmark

1. Design rationale

1.1 Choice of API Level

We chose to make our application run on API level 18 and above. This decision was made as a result of balancing the drawbacks of not having newer development tools available as readily against the advantage of supporting more devices. By targeting a minimum API level of 18, we were able to cover near 85 % of all live devices running Android. Since going below API 18 would result in a marginal increase of device coverage but incur a significant reduction in the tools available to us, we decided to stay at this level. Had we not, we anticipate that the end product would have suffered in terms of features as we would have had an increased workload due to having to use support libraries to work around the limitations of earlier API levels.

1.2 External Dependencies

1.2.1 Support frameworks

Three support frameworks (support, appcompat and design) developed by the Android team were used to provide a consistent experience for each supported API level.

1.2.2 Testing frameworks

The JUnit testing framework was used to simplify the writing of unit tests.

1.2.3 APIs

Three APIs are included in the project. Two of these are Facebook and Firebase, which are both used for authentication purposes. The user first logs in through Facebook, and the login information is then passed on to Firebase which completes the login process.

Furthermore, Firebase also acts as the database for the application. Both chats and users are stored in Firebase.

Lastly, Google Translate is used to handle translations of messages.

1.2.4 Graphical

Several different libraries were used to improve the graphical look and feel of the application. For instance, the “pull to refresh”-animation in the list of chats and the path menu in the top-right corner of the chat view are both accomplished through the usage of external libraries.

2. Overview

2.1 Behavioral and Structural

The goal was to divide the application into three major components in accordance to the MVP (Model-View-Presenter) design pattern. Activities and fragments (found in the *activity* & *fragment* packages) act as views with the sole task of registering events and presenting data, while the Presenter classes (found in the *presenter* package) handle the application logic. Finally, the Model (found in the *core* package) consists of classes that relate to the business domain, like User, Message and Chat.

The remaining major component is the database (found in the *database* package), which handles all communication with the external database (Firebase).

The views are the main entry point of the application: they receive the events from the user, and forward these to the presenter. From here, the presenter utilizes the model as well as the database to determine what will happen, and then tells the view to update itself if needed.

However, this flow of information ended up not being followed very strictly. In several cases the activities and fragments ended up acting as both views and presenters, since refactoring was left out in favor of continuing with implementing successive features.

2.2 Protocol (client/server)

Firebase is used as the database solution. It is a JSON-based database (No SQL). In an attempt to simplify working with the database functionality, a framework emulating the Promise functionality of javascript was developed. The reasoning for this was to allow us to escape the “callback hell” that is easily encountered when handling asynchronous code, and enable us write code that is easier to understand and maintain.

All of the server calls are handled through the database package. Its entry point is the DatabaseHandler class, which is the class the rest of the application uses to gain access to features of the database. Within the database package, there is one specialized class for managing users, and one for managing chats. Listeners are attached to the database system internally in the database package, and other parts of the application subscribe to these listeners using the (Name)Reference (eg. ChatReference) when they want to receive updates from or write information to the database.

3. User stories

The weekly velocity was set to 48. Effort per user story is noted within parenthesis. (20) denotes an effort of 20 for that particular user story.

3.1 First Sprint

- As a refugee / mentor I want to be able to open an application to be able use the service *(5)*
- As a refugee / mentor I want to be able to login using Facebook so that I can have an identity in the app *(20)*
- As a refugee / mentor I want to be able to send messages while in offline mode and have them be delivered once my internet connection returns. *(2)*
- As a refugee / mentor I want to be able to send messages to a global chat *(20)*

3.2 Second Sprint

- As a refugee / mentor I want to be able to be matched with a stranger *(8)*
- As a refugee / mentor I want to be able to block another user *(5)*
- As a refugee / mentor I want to be able to use the application in offline mode so that I am still able to view messages even there is no connection available *(13)*
- As a refugee / mentor I want to be able to chat with one other user so that I can learn/help *(20)*
- As a refugee / mentor I want to be able to open a private chat *(13)*
- As a product owner, I want a rad colour scheme and super awesome design *(13)*

3.3 Third Sprint

- As a refugee / mentor I want to be able to change my settings in this particular application *(13)*
- As a developer, I want to have a structured way of storing Chats in the database so that I can continue working on user stories that generate value *(13)*
- As a refugee / mentor I want to be able to change language of all app text *(13)*
- As a refugee / mentor I want to be able to start a chat when I have been matched with a stranger *(8)*

3.4 Fourth Sprint

- As a User I want to be able to have multiple conversations so that I can talk with several people at once *(13)*
- As a refugee / mentor I want to be able to change my preferred chat partner's Swedish skill level *(8)*
- As a refugee / mentor I want to be able to terminate a chat with a stranger *(5)*
- As a refugee / mentor I want to be able to click on a message to see a translation of it so that I can learn the language *(13)*

- As a refugee / mentor I want to be able to change the randomized chat name to something more personal, in order to keep track of who is who. *(5)*
- As a developer I want to be able to send queries to the database. *(20)*

3.5 Leftovers / Stretch Goals

- As a refugee / mentor I want to be able to have a menu in chat to reach more choices
- As a refugee / mentor I want to be able to get a random topic so that I can talk about it
- As a refugee / mentor I want to be able to change preferred language in tap-translation
- As a refugee / mentor I want to be able to login without facebook so that I don't need Facebook to use this app
- As a refugee / mentor I want to be able to have a separate choose language page
- As a refugee / mentor I want to be able to see in the chat list (as well as in the actual chat) that a person is blocked
- As a refugee / mentor I want to be able to unblock a person
- As a refugee I want to be able to request a new conversation topic (if both agrees) so that I do not get bored
- As a refugee / mentor I want to be able to choose if the app should not automatically login for me
- As a refugee I want to be able to take a quiz (or game) on the given topic so that I can learn more
- As a refugee / mentor I want to be able to choose if the app should not automatically login for me
- As a refugee I want to be able to take a quiz (or game) on the given topic so that I can learn more
- As a owner of the application I do not want to pay too much money for Google Translate
- As a refugee I want to be able to get words spoken to me so that I can hear how they are pronounced

3.6 Discussion

The team unanimously agrees on that the project ended up too big in scope. There were recurring issues with time estimations, where user stories were normally underestimated in terms of effort. This was brought up during basically every sprint retrospective yet during the planning of the next sprint, this issue would manifest itself again.

Due to this, many features were completed outside of the actual sprint, during nights and weekends. Instead of accepting that there had been too many stories in a sprint, certain team members would put in extra hours to complete these unfinished stories. This was beneficial to the product in the short-term, but caused issues where other team members fell behind in terms of understanding of the codebase as the project progressed. Additionally, it caused team members a lot of stress and as such was likely a net negative for the project.