VIA University College

Project Report

Surfing-couch

Rosalie Peree (258377) Grégoire Cartier (258378)

Supervisor & Academic advisor: Kasper & Jakob

2017 (Autumn semester)

Contents

1 Abstract **1**

2 Introduction **2**

2.1 Background Description 2

2.2 Problem Description 2

2.3 Limitations and Delimitations 2

3 Methods **3**

4 Requirements **5**

4.1 List of Requirements 5

4.1.1 Functional Requirements 5

4.1.2 Non-Functional Requirements 8

5 Analysis **9**

6 Design **10**

6.1 Architecture 10

6.2 Technologies 11

6.3 Design Patterns, Class Diagram, Sequence Diagrams 12

7 Implementation **14**

8 Test **15**

8.1 Test Specifications 15

8.2 White Box Testing 15

9 Results & Discussion **16**

10 Conclusion **17**

1 Abstract

2 Introduction

2.1 Background Description

2.2 Problem Description

2.3 Limitations and Delimitations

3 Methods

The following table presents our choice of model and methods for the problems that we listed in our project description (see Appendix #1):

What *Problem*	**Why** *Why study this problem*	**Which** *Which models/theories did you use to solve the problem?*
How can we develop a social application which is fun to use and rewards the user for helping travelers in their area?	How to make the application interesting for our potential users?	We have to pay attention to all the following problems to be able to develop an application that will address all of them as good as possible.
What is important for travelers?	As our application is mainly targeted to travelers, it is interesting to know what is important for them, so that we cant try to implement those things in the final application.	We conducted a survey via Google Forms to gather people's opinion on that subject.
What is important for hosts?	As we want hosts to offer services to travelers, we have to meet with what they want to encourage them to do so.	We conducted a survey via Google Forms to gather potential user's opinion on that subject.
How can we allow users to create and find trips in their area and offer a service to travelers?	This problem is the main problem for a couch-surfing-like application. We have to localize the user to be able to provide a service for them in a specific area.	We have to set up a way for the users to enter their city/location and look for hosts in cities they want to visit.
How can we make the service secure for users?	Users don't want their personal data to be forwarded to strangers.	No personal data is displayed on the application for users to see. We implemented an internal messaging system so people can communicate directly on the application.
Can we add reviews and comments about travelers and hosts?	If you have to host someone or sleep/shower at a stranger's place, safety is very important. Users need a way to see if someone behaved correctly or not before their stay.	We implemented a system of user reviews - with grades - so people can report how they experience was with that specific user - hosts and travelers.

What Problem	**Why** Why study this problem	**Which** Which models/theories did you use to solve the problem?
How can we define the point-value of the provided services?	As the system is based on a reward, all services must have a fixed value.	We defined a value for each service - sleep, shower and laundry. The points are given to the host once the booking is completed.
How can we exchange the points for gifts?	The system is based on rewarding hosts with gifts for providing a service.	We set up an online shop where users can exchange their points for goods.
How will the user interface be implemented?	The user interface has to be easy to get and use.	The interface will be an Android application.
What kind of tool will be used for this implementation?	The tools used to develop can make the process of developing the application more or less easy.	We decided to develop the application using Android Studio, as it is the platform we are both most familiar with.
How will the interface be user-friendly?	The interface has to be easy to use and understand for any user.	The interface is simple, with buttons and labels that are easy to understand.
How is the social interaction going to take place?	The point of the assignment is to enable a social interaction.	In our application, the social interaction will take place in the form of direct messaging between two users, a common chat for all the users and the possibility of reviewing experience with other users.
What kind of database will be implemented?	We need a database that is free and easy to integrate in Android.	The chosen database is Firebase for our project.

4 Requirements

The users of the application will be any person interested in traveling on a budget or having a different traveling experience, as well of any person that will be willing to host or provide a service -for now, shower and laundry - to a traveler visiting their city. They could be from everywhere in the world and have about any age, as long as they have access to a mobile phone running Android and a working Internet connection. They should not have an extensive knowledge of IT to be able to use the application.

4.1 List of Requirements

4.1.1 Functional Requirements

Use cases:

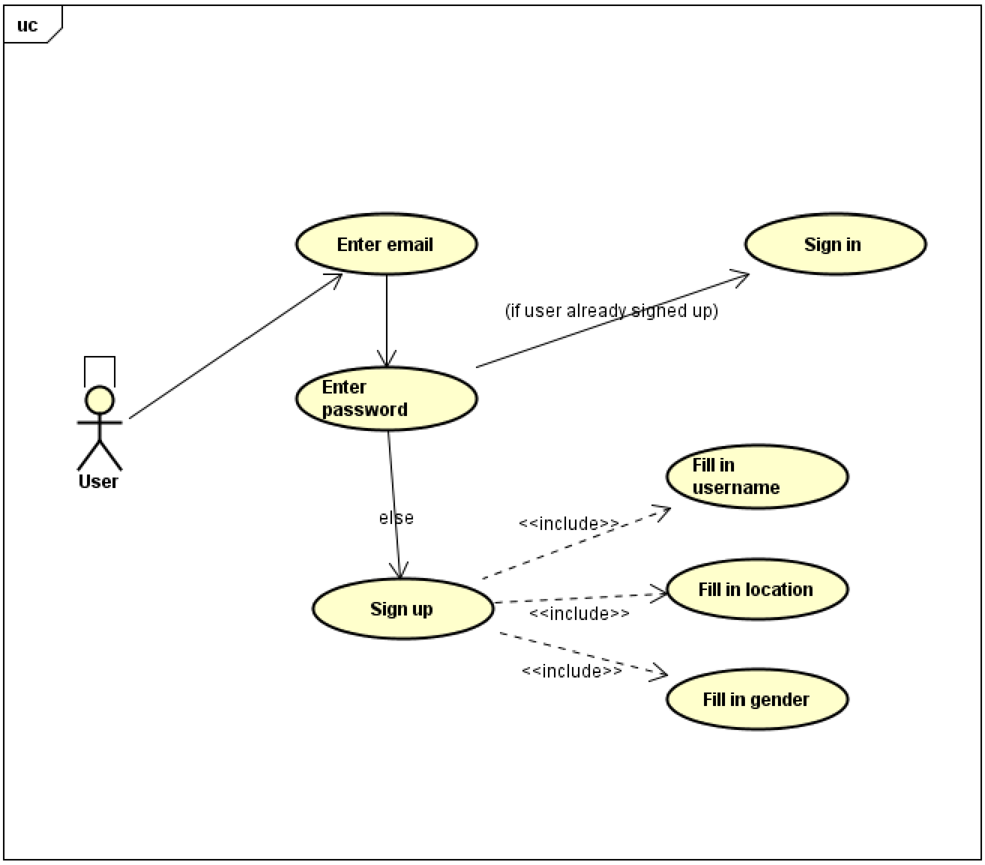

Figure 1: Use Case 1: how to access to application

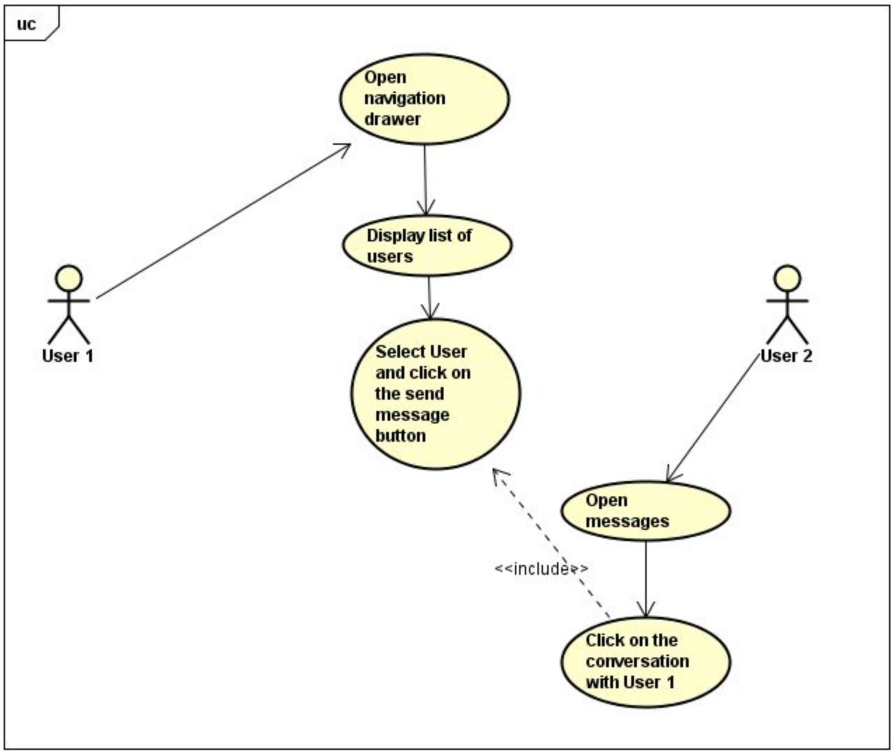
uc
Open navigation drawer
Display list of users
Select User and click on the send message button
User 1
User 2
Open messages
<<include>>
Click on the conversation with User 1

Figure 2: Use Case 2: how users can communicate together by messages

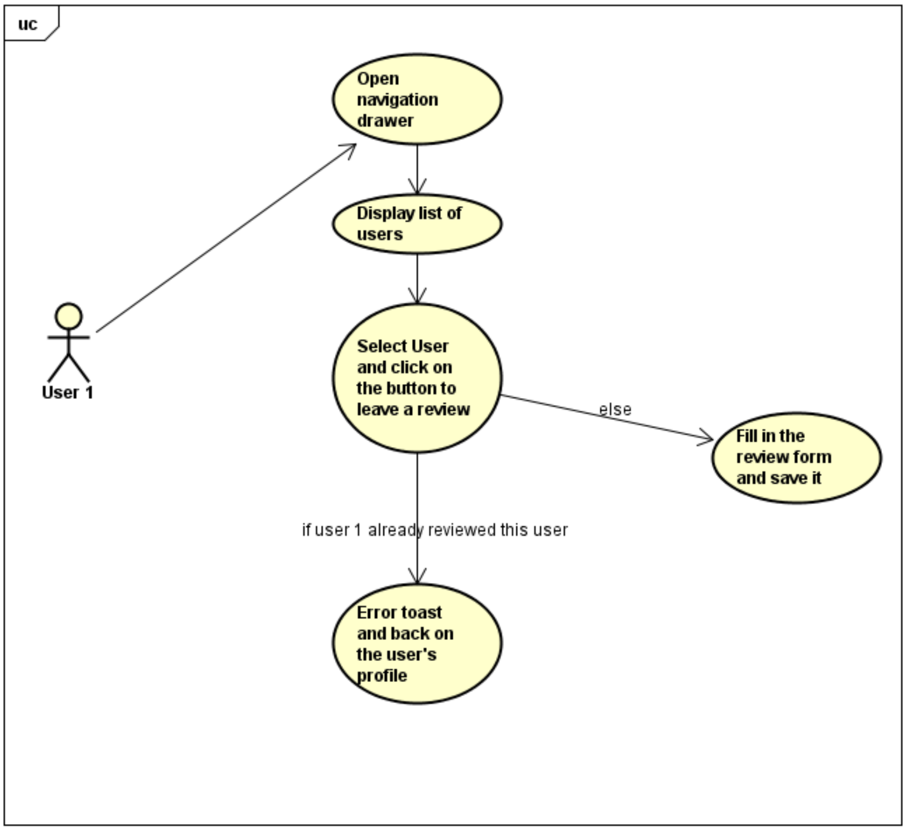
uc
Open navigation drawer
Display list of users
User 1
Select User and click on the button to leave a review
else
Fill in the review form and save it
if user 1 already reviewed this user
Error toast and back on the user's profile

Figure 3: Use Case 3: how a user can leave a review to another user

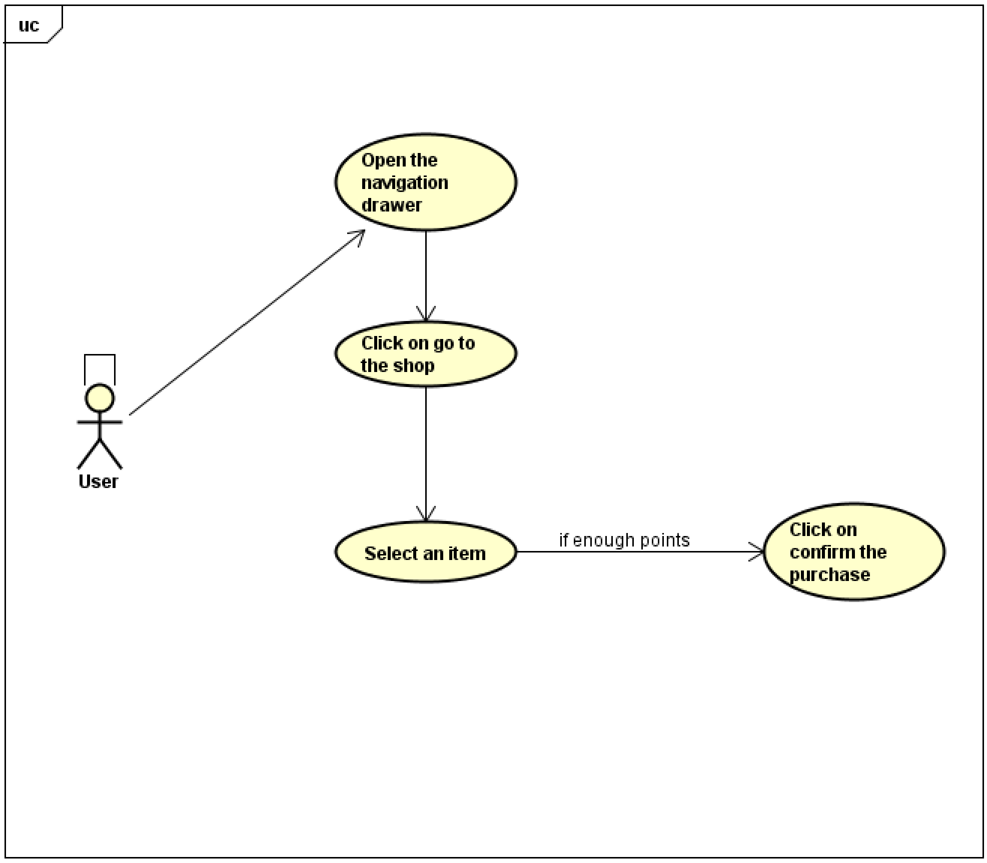

Figure 4: Use Case 4: how a user can cash his points for a reward

Activity Diagrams:

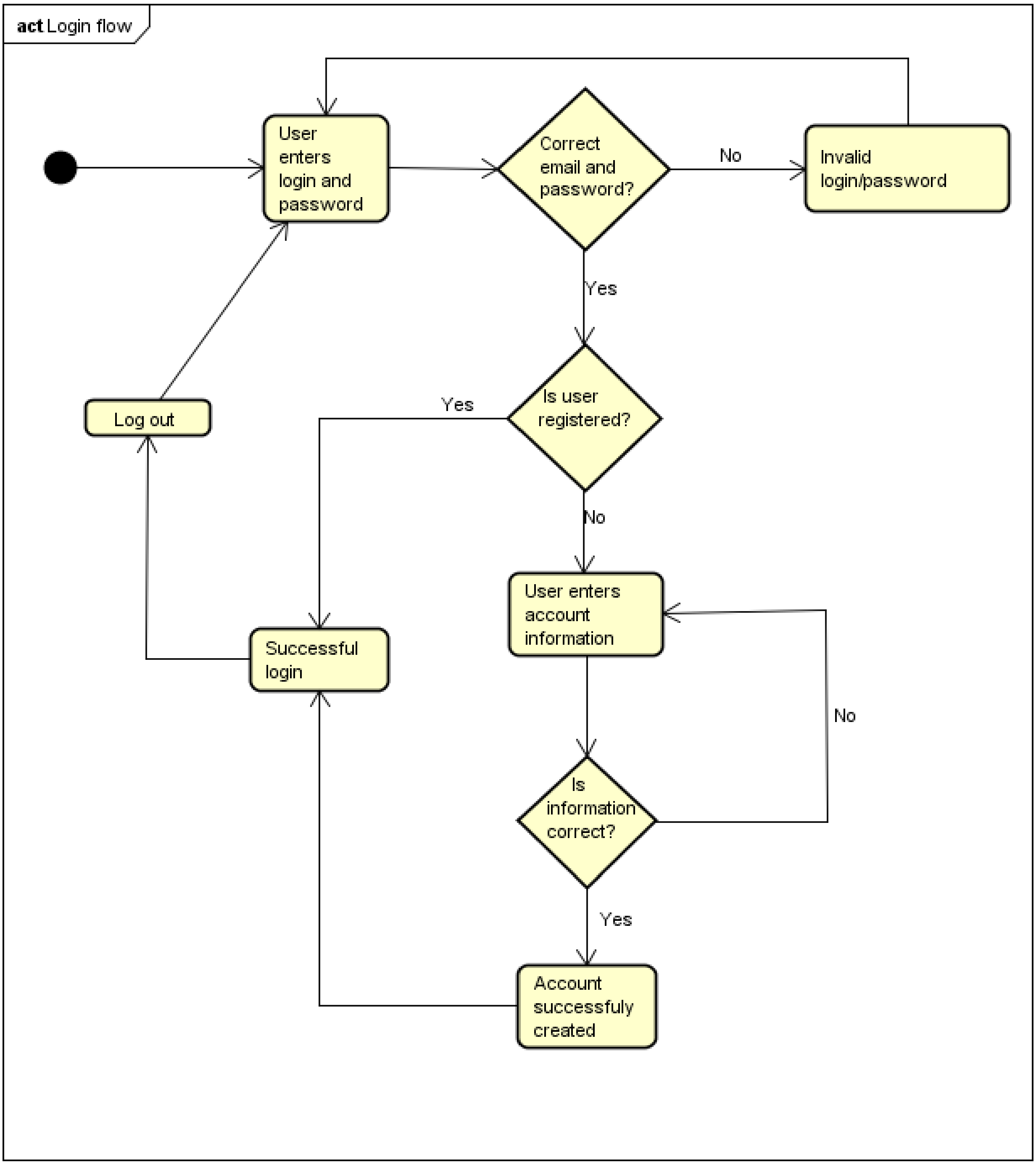

Figure 5: Activity Diagram 1: sign in on the application

4.1.2 Non-Functional Requirements

5 Analysis

6 Design

6.1 Architecture

We used the Adapter in Android which is close to a Pattern Adapter.

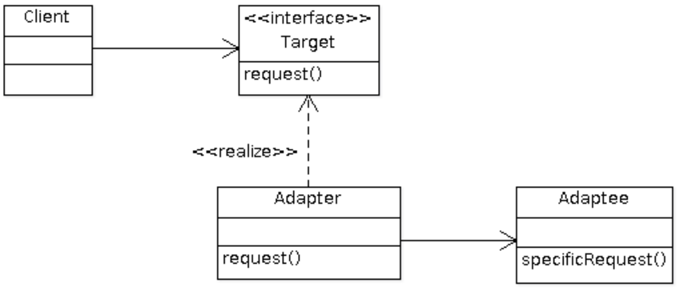

Figure 6: Adapter Pattern Template (*softwarepassion.com*)

We used those Adapters to populate the listViews with ArrayList and make clickable. Those Adapters are customizable, and help to display the content of the Objects that are kept in the database

We also wanted to use a MVC (Model View Controller), which is a MVP (Model View Presenter) but the scope of the project was too short to implement it correctly, and we didn't studied it during the Android course.

6.2 Technologies

Surfing Couch uses different technologies :
To do deal with all of the data manipulated by the users, the applications uses **Firebase** as database :

- Pros :
  - Firebase provides an easy implementation on an Android application, there is not that much of a setup to do, the read and write functions are really easy to implement
  - Firebase use JSON which permits an implementation of data in a fairly effortless way. That create a database that is flexible, well-structured and easy to read to some extent.
  - Firebase gives useful tools, including a user authentication that deals with the security of database thanks to its data hashing. Firebase is also loaded with email confirmation, data analytics, and stability tools, which helps to make the user experience better.
  - Made by Google
- Cons :
  - When the data structure starts to be a bit complex, it becomes quite hard to navigate through all of the data. Especially if you have a lot of data.
  - The querying and indexing is very limited due to the use of the JSON. It makes the search of data is very redundant.
  - Firebase is a bit high level, so you don't feel you have that much of control over it
  - Made by Google

One of the alternative to Firebase could be to use a more "classical" database management system like **SQLite** :

- Pros :
  - Classical way to store data, you can do SQL queries on it to retrieve data with more efficiency. Include also a good indexing of the data.
  - Is better than Firebase at storing huge load of data.
- Cons :
  - You don't have the tools proposed by Firebase, so you can really analyze your data usage and you must make all of the part of the security by yourself
  - Harder to use and implement than Firebase.

The application must be developed natively on Android, so **Android Studio** was a default choice to develop the application. But it's interesting to point out that there is alternatives to Android Studio such as **Xamarin** or **Cordova**.

6.3 Design Patterns, Class Diagram, Sequence Diagrams

Like said earlier, we are not using a specific design pattern.

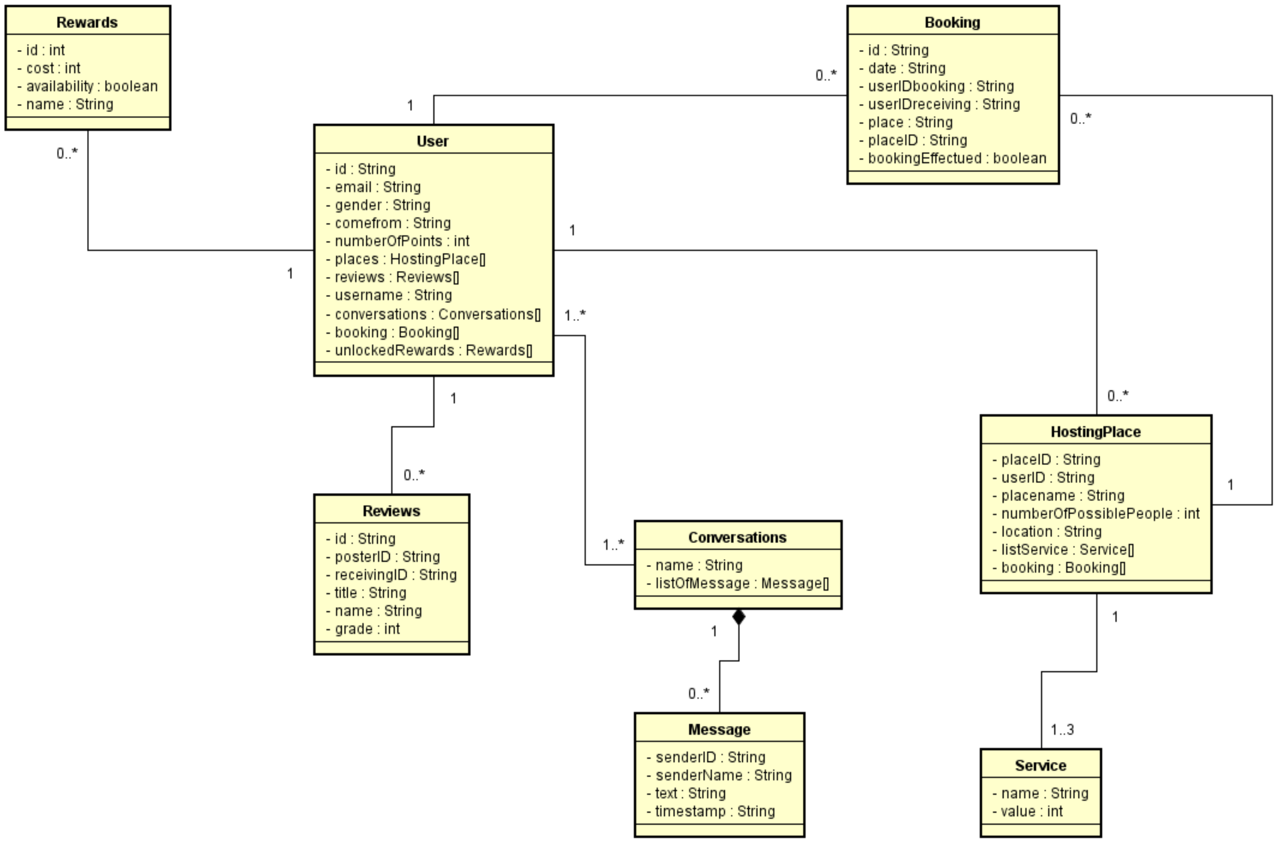

Figure 7: Class Diagram of the database

The database[1] is pretty straightforward, everything is based around the User. However, we must highlight some details. By default, when a user is created, the bookings are null, so not present in the database JSON. The list is created with the first booking he makes. Also, a default chat is assigned to each user which is the “General Chat” so can talk all together. The other List are defined as “undefined” and taken as HashMaps inside of the app.

[1] We made a class diagram for the databse because the data is kept in a JSON

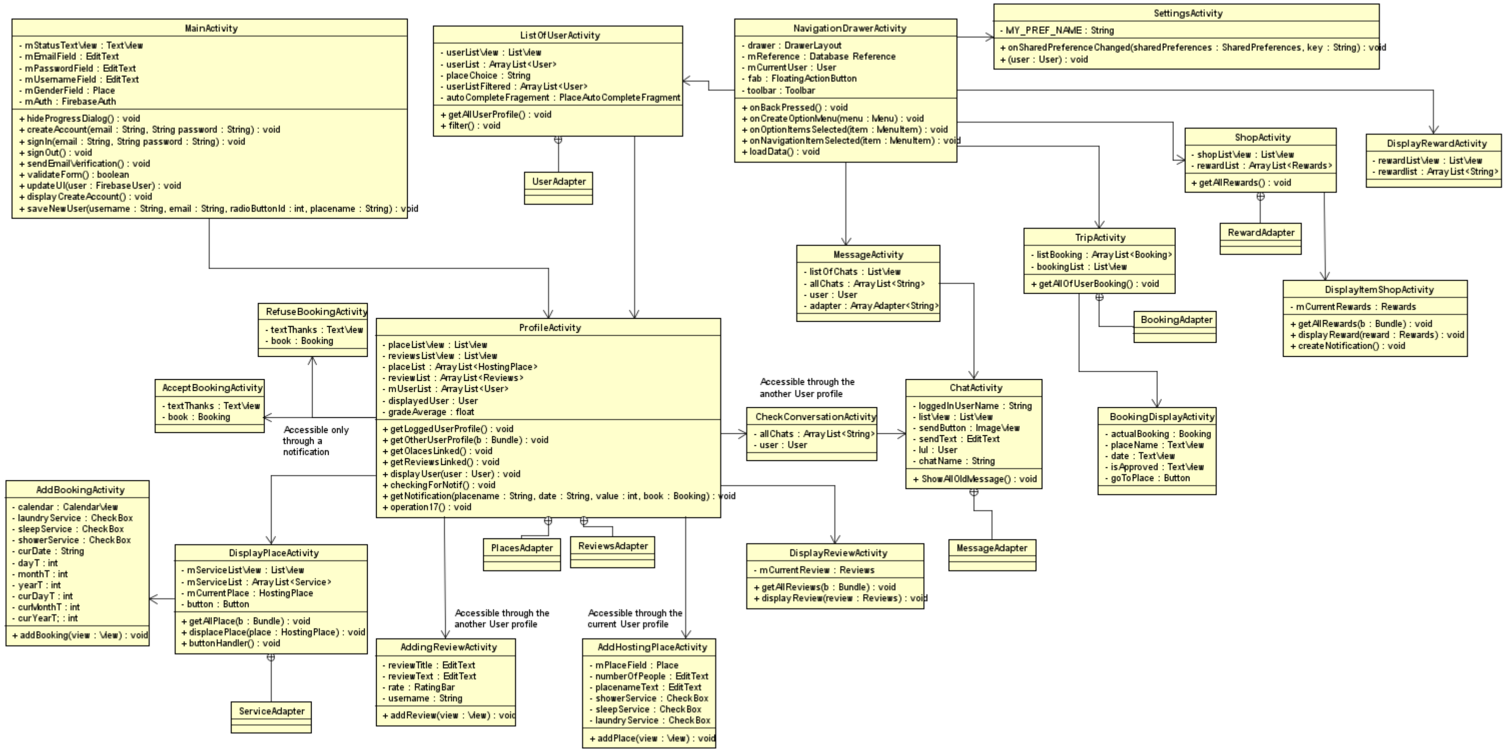
MainActivity
- mStatusTextView : TextView
- mEmailField : EditText
- mPasswordField : EditText
- mUsernameField : EditText
- mGenderField : Place
- mAuth : FirebaseAuth
+ hideProgressDialog() : void
+ createAccount(email : String, String password : String) : void
+ signIn(email : String, String password : String) : void
+ signOut() : void
+ sendEmailVerification() : void
+ validateForm() : boolean
+ updateUI(user : FirebaseUser) : void
+ displayCreateAccount() : void
+ saveNewUser(username : String, email : String, radioButtonId : int, placename : String) : void
ListOfUserActivity
- userListView : ListView
- userList : ArrayList<User>
- placeChoice : String
- userListFiltered : ArrayList<User>
- autoCompleteFragement : PlaceAutoCompleteFragment
+ getAllUserProfile() : void
+ filter() : void
UserAdapter
NavigationDrawerActivity
- drawer : DrawerLayout
- mReference : Database Reference
- mCurrentUser : User
- fab : FloatingActionButton
- toolbar : Toolbar
+ onBackPressed() : void
+ onCreateOptionMenu(menu : Menu) : void
+ onOptionItemsSelected(item : MenuItem) : void
+ onNavigationItemSelected(item : MenuItem) : void
+ loadData() : void
SettingsActivity
- MY_PREF_NAME : String
+ onSharedPreferenceChanged(sharedPreferences : SharedPreferences, key : String) : void
+ (user : User) : void
ShopActivity
- shopListView : ListView
- rewardList : ArrayList<Rewards>
+ getAllRewards() : void
DisplayRewardActivity
- rewardListView : ListView
- rewardlist : ArrayList<String>
RewardAdapter
MessageActivity
- listOfChats : ListView
- allChats : ArrayList<String>
- user : User
- adapter : ArrayAdapter<String>
TripActivity
- listBooking : ArrayList<Booking>
- bookingList : ListView
+ getAllOfUserBooking() : void
BookingAdapter
DisplayItemShopActivity
- mCurrentRewards : Rewards
+ getAllRewards(b : Bundle) : void
+ displayReward(reward : Rewards) : void
+ createNotification() : void
RefuseBookingActivity
- textThanks : TextView
- book : Booking
AcceptBookingActivity
- textThanks : TextView
- book : Booking
Accessible only through a notification
ProfileActivity
- placeListView : ListView
- reviewsListView : ListView
- placeList : ArrayList<HostingPlace>
- reviewList : ArrayList<Reviews>
- mUserList : ArrayList<User>
- displayedUser : User
- gradeAverage : float
+ getLoggedUserProfile() : void
+ getOtherUserProfile(b : Bundle) : void
+ getOlacesLinked() : void
+ getReviewsLinked() : void
+ displayUser(user : User) : void
+ checkingForNotif() : void
+ getNotification(placename : String, date : String, value : int, book : Booking) : void
+ operation17() : void
Accessible through the another User profile
CheckConversationActivity
- allChats : ArrayList<String>
- user : User
ChatActivity
- loggedInUserName : String
- listView : ListView
- sendButton : ImageView
- sendText : EditText
- lul : User
- chatName : String
+ ShowAllOldMessage() : void
BookingDisplayActivity
- actualBooking : Booking
- placeName : TextView
- date : TextView
- isApproved : TextView
- goToPlace : Button
AddBookingActivity
- calendar : CalendarView
- laundryService : CheckBox
- sleepService : CheckBox
- showerService : CheckBox
- curDate : String
- dayT : int
- monthT : int
- yearT : int
- curDayT : int
- curMonthT : int
- curYearT; : int
+ addBooking(view : View) : void
DisplayPlaceActivity
- mServiceListView : ListView
- mServiceList : ArrayList<Service>
- mCurrentPlace : HostingPlace
- button : Button
+ getAllPlace(b : Bundle) : void
+ displacePlace(place : HostingPlace) : void
+ buttonHandler() : void
ServiceAdapter
PlacesAdapter
ReviewsAdapter
DisplayReviewActivity
- mCurrentReview : Reviews
+ getAllReviews(b : Bundle) : void
+ displayReview(review : Reviews) : void
MessageAdapter
Accessible through the another User profile
AddingReviewActivity
- reviewTitle : EditText
- reviewText : EditText
- rate : RatingBar
- username : String
+ addReview(view : View) : void
Accessible through the current User profile
AddHostingPlaceActivity
- mPlaceField : Place
- numberOfPeople : EditText
- placenameText : EditText
- showerService : CheckBox
- sleepService : CheckBox
- laundryService : CheckBox
+ addPlace(view : View) : void

Figure 8: Class Diagram of the Activities of the application

See the annex file for better version

7 Implementation

8 Test

8.1 Test Specifications

8.2 White Box Testing

9 Results & Discussion

10 Conclusion