1. Introduction

This project was set to create an Android application for newly arrived to Sweden. When we started, we sat down to discuss the different applications that already exist and what we ourselves wanted to create. We all wanted a new approach for the problems of moving to a new country. Language will always be a problem in the beginning, but we wanted to focus more on integration.

So our app “Mentorite”, comes from the concept of a new way of getting to know people. It works similarly to a contact book, with contact information for mentors that are interested in talking and befriending newly arrived. Also, we wanted to make the connection a relaxed one between the user and the mentors, and to get around language barriers. Mentorite was therefore decided to be as visual as possible, driven by icons rather than text.

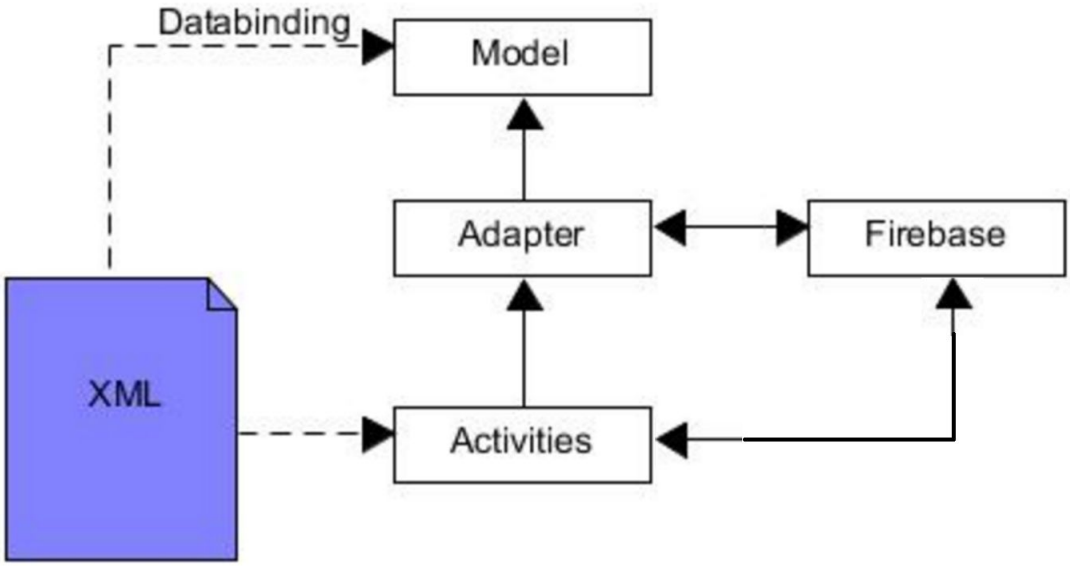
Another decision we made was that we did not want to have chat functionality. Our reasoning is that there already exist so many different ways of communication and we would like for our users to create connections outside our app. We think this would make the connection more genuine and make a growing relation more likely.

We decided to use Facebook and Google’s sign in features and in the userprofile, we linked up Facebook and e-mail to the mentor.

2. Program design

2.1 Design patterns

We didn’t follow a strict pattern but instead we used a combination of the MVC, MVVM and Adapter pattern. We laid our foundation through the MVC pattern because everyone in the group had worked with it before.The Activities acts as the Controller Layer. The XML layout is the View layer. The adapter and models represents the model layer. But through Databinding we get the MVVM pattern. We used Databinding because it saved us from large amounts of code and it separated the View. With a separated View it allowed us to test the other layers without a View even existing. More specifically we used two-way Databinding. When it goes two-ways, properties in the model gets updated and so does the View. When the View elements get updated, the changes get propagated back to the model. We really liked this concept, however later into the project we began to also work by the “old way” of using “findViewById”. But at that stage the View was mostly finished so a de-coupling of the View wasn’t as necessary anymore.



2.2 Firebase

Firebase was the perfect choice for us. We wanted to have a working prototype as soon as possible. By using firebase, our development time was greatly reduced. Firebase provides a server and data storage. We only needed to make our own client. The server could be accessed and manipulated by everyone in the group, through a web browser.

Firebase Database uses data synchronization instead of usual HTTP requests. So a connected client receives changes in no time. It stores all the data in a real time cloud-hosted database. An app will also work when it goes offline since the data is stored locally. When it goes back online again it will receive all the data it missed.

Firebase also gave us the many different authentication options and made it easy for us to implement google and facebook login.

2.3 API-level

Supported API-levels were level 15 to 23(the latest). We wanted to support to API-level 15 since you then support 97% of the worlds Android phones. Perhaps we should have used an even lower API-level as newly arrived likely have relatively older phone, but decided against it to have more functionality and as said statistics show that 97% use 15 or over.