System design document for AWME bokbytarapp

Ali Alladin, Magnus Andersson, William Hugo, Elias Johansson

2020-10-02

0.1

1 Introduction

Give an introduction to the document and your application.

This document is intended to present our system design for our application “bokbytarapp”. The program functions as an online marketplace where users can sell used items that relate to their education in some form. No transactions are handled in the app as it only functions as a mediator.

1.1 Definitions, acronyms, and abbreviations

Definitions etc. probably same as in RAD

2 System architecture

The most overall, top level description of your application. If your application uses multiple components (such as servers, databases, etc.), describe their responsibilities here and show how they are dependent on each other and how they communicate (which protocols etc.)

You will describe the ‘flow’ of the application at a high level. What happens if the application is started (and later stopped) and what the normal flow of operation is. Relate this to the different components (if any) in your application.

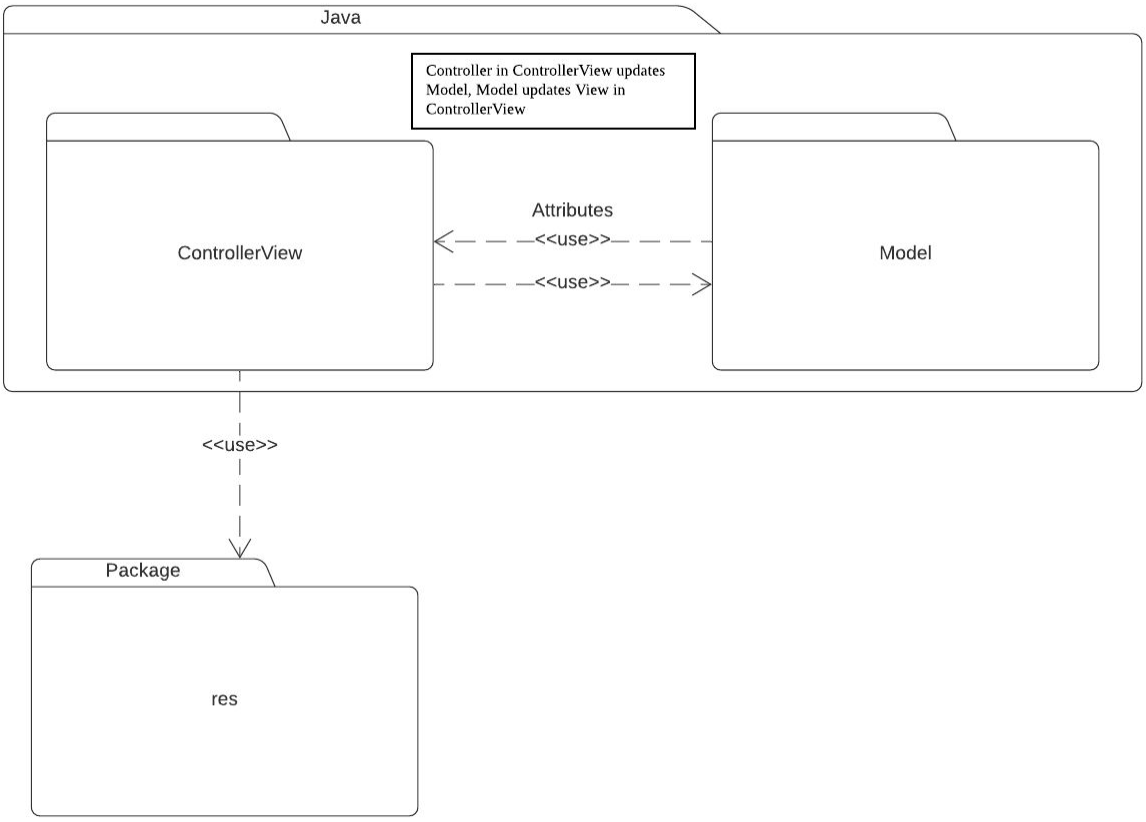
When you start the application you have to login. This starts a websocket session with our database, which will persist throughout the whole run of the application and allow for real time updates to the items shown in the application.
After login you come to searchPage, where all listings available in the database are displayed, should another sale be added it will update on this screen.
A listing can be interacted with, which will take you to listingPageActivity, here you will see the full information about the listing, as well as who posted it. Taking a step back to searchPage, you have another option as well. Going to accountPage.

In accountPage you can see your own listings, add a new listing, or logout. If you add a new listing you come to a new page where you can fill in a form to submit a new listing in your name to the database, which will then appear to all users in searchPage, and for you on accountPage.

If you are to close the application at any stage, you would break the connection to the database, and be greeted with the login screen upon the next launch.

3 System design

Draw an UML package diagram for the top level for all components that you have identified above (which can be just one if you develop a standalone application). Describe the interfaces and dependencies between the packages. Describe how you have implemented the MVC design pattern.



The MVC pattern is used by dividing our program into three packages: Model, ControllerView and res. Model represent all the model classes in our program. ControllerView are classes that function to some extent as classes that create views and ordinary controllers. Res contains the xml files and images in our project.

Create an UML class diagram for every package. One of the packages will contain the model of your application. This will be the design model of your application, describe in detail the relation between your domain model and your design model. There should be a clear and logical relation between the two. Make sure that these models stay in ‘sync’ during the development of your application.

Describe which (if any) design patterns you have used.

Module design pattern in form of MVC. Template method in our item hierarchy, an observer-like pattern where the class listingAdapter observes the Database class for database communication.

The above describes the static design of your application. It may sometimes be necessary to describe the dynamic design of your application as well. You can use an UML sequence diagram to show the different parts of your application communicating in what order.

4 Persistent data management

If your application makes use of persistent data (for example stores user profiles etc.), then explain how you store data (and other resources such as icons, images, audio, etc.).

We use a database to store all items and users in our program. Each item is stored as a dictionary of strings and numbers.
Books isbn numbers have to be stored as long integers because they are 13 digits long.
We store the price as doubles, to retain the decimals.
Images are stored as encrypted Base64 strings.

Users are stored as a unique value generated by our database upon creation of the user, with attached name, and two lists of items, one that user is selling, and one that the user favours.

Some icons are stored in our res folder.

5 Quality

- Describe how you test your application and where to find these tests. If applicable, give a link to your continuous integration.
    - We write jUnit tests, which gradle will run every time you build the application, and travis, the continuous integration service, will run every time someone

submits code to our repo. These tests can be found at \AMWE\app\src\androidTest\java\com\example\amwe and \AMWE\app\src\test\java\com\example\amwe
Here is a link to our CI, https://travis-ci.com/github/magnusGU/AMWE

- List all known issues.
    - The camera has crashed the application, it happens when the program is run for the first time. It works as intended otherwise. Reason is unknown but has probably to do with files being stored in folders not yet created by the program.
- Run analytical tools on your software and show the results. Use for example:
    - Dependencies: STAN or similar.
    - Quality tool reports, like PMD.
    - Gradle handles both quality and dependency checks and reports. The following is a link to a gradle generated report, containing results of building, checking and generating a dependency tree. For the most in detail report check under the console log tab, where the whole “gradlew clean build app:dependencies” command can be followed step-for-step. https://gradle.com/s/pjrzgmr36d3k4

NOTE: Each Java, XML, etc. file should have a header comment: Author, responsibility, used by ..., uses ..., etc.

5.1 Access control and security

If you applications has some kind of access control, for example a login, of has different user roles (admin, standard, etc.), then explain how you application manages this.

To use our application you have to have a user account, or create one with a “semi-valid” email, (no verification mail is sent to the email does not have to be real). But there has to be an at-symbol ‘@’ followed by some service dot some domain
I.e name@mail.com
These accounts, and their password security is handled by our database, firebase.

6 References

List all references to external tools, platforms, libraries, papers, etc. The purpose is that the reader can find additional information quickly and use this to understand how your application works.

Android.
Firebase
Gradle
Travis

JUnit 4