System design document

Table of contents

Version: 1.4
Date: 2012-07-28

Authors: Oscar Brodefors, Filip Askviken, Rikard Andersson, Emil Nyström

This version overrides all previous versions.

1 Introduction

The system architecture affects the performance, robustness, distributability and maintainability of a system. Therefore, the architecture is important and critical in making a well functioning system (Sommerville, 2007).

1.1 Design goals

The aim for this document regarding the system architecture is to be a helpful tool in the process of implementing the system design. The document is also intended to be useful in the process of introducing a new member in the project. The design of the application should be suitable, logical, easy to understand, and easy to implement.

1.2 Definitions, acronyms and abbreviations

GUI = Graphical user interface

1.3 References

Sommerville, Ian (2007). Software engineering. Eight edition, Pearson Education Limited, Edinburgh Gate, England.

2 Proposed system architecture

In this section a high level architecture is proposed.

2.1 Overview

The user of the application is to interact with a GUI on his/her handheld device.

2.2 Software decomposition

2.2.1 General

We aim to have a five-layer structure. With a database and the GUI at opposite ends. The GUI connects to a functional layer which in turn communicates with a server communication-layer, all this still on the client. The server communication-layer handles all the remote communication with the server. This to have a more loose connection between the functional layer and the server, in case we choose to change this model in the future. Communication between the server communication-layer and the server goes through php-scripts on the server side which makes inquiries into the final layer, the database.

2.2.2 Tiers

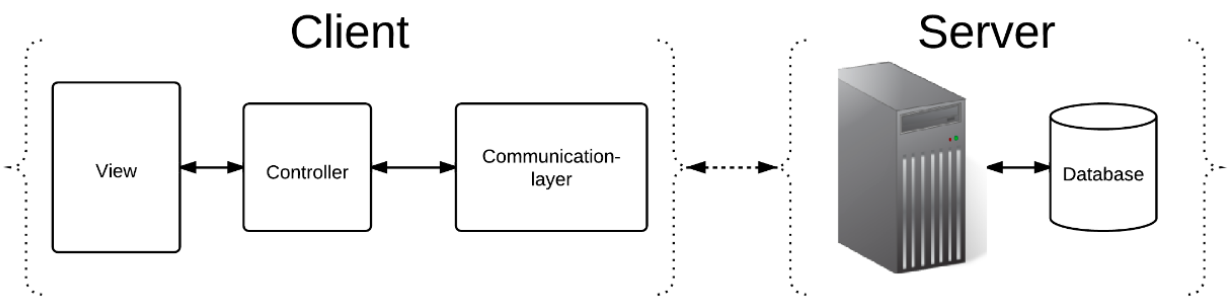
The system will be set in two tiers in a server/client-sided manner. The server will only house the database itself while the GUI, functional layer and the server communication-layer all reside client-side - the handheld device.

2.2.3 Communication

We aim to have as little communication as possible between the layers and especially between server and client. This is due to the fact that the application is dependent on a internet connection which can be of limited capacity. Instead old information will be accessible when there is no internet connection (i.e. information from the last time there were a connection). There will be no communication between the server and client which the client did not initiate.

The server communication-layer will be made up of a static class with static methods, hence this layer will not be aware of of the functional layer. The GUI and the functional layer will be aware of each other.

2.2.4 Decomposition into subsystems



2.2.5 Layering

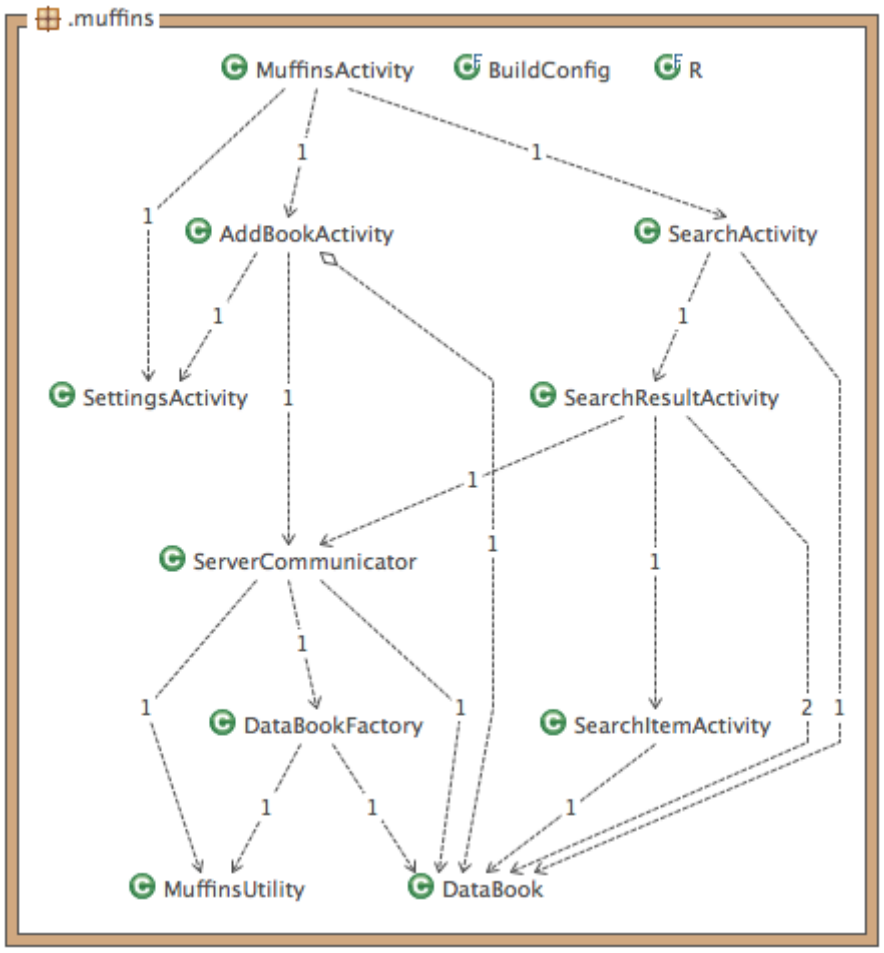
The base of our application is to rely on a database which contains all information regarding the books registered for sale. It will also contain information about users who have books registered for sale, or requested to buy. This will be required to allow the creator of an entry, and the creator alone, to edit and remove said entry. The user end of the application will be constructed according to Android-standards with a MVC-design.

2.2.6 Dependency analysis

The server communication-layer is hard coded to connect to and communicate with the certain server used so far in the project. However the server-side php-scripts and database is not dependent on the application but can be reused if for example an iOS application is made.

Client-side the view and the functionality is interlaced in activities. The server communication-layer is not aware of any layers locally on the client. The activities are however dependent on the server communication-layer through several references to this class in the activities.

This is an image generated by the Eclipse plugin stan4j showing how the actual dependency between classes look like.



2.3 Concurrency issues

The only person who can actually edit an entry in the database is the creator. Therefore the only concurrency case will be if the one user access the application from the different devices and try to edit the same entry. This issue will be handled in the database where only one session is allowed to alter an entry at the time. All other information is read-only (i.e. entries which the user did not create can not be edited).

2.4 Persistent data management

Concerning base functionality, we have not identified any data that would require to be persistently managed, as all data regarding the user and the books available are stored server-side.

2.5 Access control and security

Upon first launch of the application, the user can see and search the bookmarket instantly. When the user presses the button for adding a new book, the application checks if the user have filled in user details. The user details consists of: Name, email, phone number, and a password, and are stored in the settings view. If the user details are not filled in, the user needs to fill it in before uploading a book to the market. The password is used in order to enable the user editing the book advertisement. The password is stored in the database in a hashtable.
Accordingly, To modify data (i.e. a book for sale or personal information) in the database, you must be the owner of the information and have a password. Ownership is ruled by who created the data entry.
To retrieve a forgotten password in this application is not possible. The advertisement needs to remember the password in order to edit the advertisement. However, the retrieval of passwords will be fixed in the next release of the application.

To access information about a book for sale no authorisation is needed. All users can access all books for sale. Anyone with access to android market can access the application.

2.6 Boundary conditions

On launch the application will start up with the search-view as default. On close the application will save any unsaved persistent data.

If there is no internet connection the request (i.e. to add a book) will time out after a while and tell the user what happened. If there is old data from a similar request stored in the application this will be presented instead.

2.7 References

Sommerville, Ian (2007). Software engineering. Eight edition, Pearson Education Limited, Edinburgh Gate, England.