Developer details for Bookstore project

Table of contents

Version: 1.3
Date: 2012-07-28

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

This version overrides all previous versions.

Contact details:

Emil Nyström emil.nystrom@gmail.com 0703-200491
Rikard Andersson rikard.lars@gmail.com
Filip Askviken filip@askviken.se
Oscar Brodefors oscar.brodefors@gmail.com

1. Purpose of application

A simple application for smartphones where students can gather to sell and buy used books needed at courses given at Chalmers (and possibly other Universities). Users can register books for sale and other users can search and find these books. The unique selling point compared to existing sites like blocket or tradera is the nische market. When buying from our application you know that the seller is close by and that you can set up a meeting to get the book the same day.

1.2 General characteristics of application

No transactions are handled by the application. The application simply matches buyer with seller and provides the opportunity for market mechanisms to function in the interaction between these two parties.

Functionality to add a plethora of attributes about the book and for users to search for books on these criterias. Also the ability to describe a book in general and to upload a photo of the book taken with the mobile camera. A book could be uploaded by filling in a ISBN-number and then obtaining book title, author and edition.

The ability for other users to comment on the book.

Possible functions to be added in future versions:
- Auctions
- Connection to Cremona to compare price and to buy new books
- A website connected to the application

2 Test environment and installation base

This android-project is preferably tested and run from Eclipse. First step for a new tester or developer should be to install EGit from Eclipse Marketplace and the ADT plugin for Eclipse (http://developer.android.com/sdk/eclipse-adt.html.). Please visit and read the installation guide (http://developer.android.com/sdk/installing.html).

After installation of Android ADT and connection with Github in your Eclipse, import the project in Eclipse: file → import → Git → Projects from Git → next → URL → insert https://emil-nystrom@github.com/oscarbr/Brainiacs-startup.git as URL → next → next → next → finish. Now you are ready to build or run the

application.

The test report document to new developers and testers is avaliable on: https://docs.google.com/document/d/1oXC8qSHJzy3TRIEg0nVonFR0eNtPC01yBdP7oxgWOlc/edit.
The test report table is also avaliable at: https://docs.google.com/spreadsheet/ccc?key=0Ao75Xu-pWZuDdGxoc1ZROF9BZXlEUThuUnJ5UUl1OHc.

3. Software, hardware and working platforms

For the tests, three MacBook's and one Dell XPS M1530 have been used. The three MacBooks have the operating software Mac OSX-Lion and the Dell have Windows 8. An android emulator in Eclipse (Target Android 2.1, skin WVGA800) and a real mobile phone (Samsung Galaxy S plus) have been used for the tests.

For every test, the following software and working platforms have been used:

- Eclipse SDK version 3.7.2
- Android Development Toolkit version 18.0.0
- Eclipse Mylyn project version 3.6.5
- Eclipse EGit 1.3.0
- Github
- Google docs
- Jenkins
- Emma
- STAN4j

Compilation are made through the Eclipse environment and automatically via ant and Jenkins.

4. Project syntax and standard

This project is using SCRUM as project methodology.

This project uses the following standard for commit messages:
RID: TID: m:
Where RID is Requirement ID, TID is Task ID which is optionally, and then m for message. The message should include what you have done in your commit and in which file.

The project uses Java-doc for commenting code (http://en.wikipedia.org/wiki/Javadoc.

Traceability in this project have also been conducted by building new GitHub-branches for bugs and new functionality and an integration with GitHub-issues in Eclipse.

Text in this project is written in Google docs with the following syntax:
Normal text: Courier new, size 12
Headings: Use google docs standard headings 2,3 and 4 (font: arial)

Text documents in this project should be uploaded and distributed in pdf-format.

Communication in this project is made by email and WhatsApp. Please email emil.nystrom@gmail.com for access to the WhatsApp group.

4.1 Release routines

Every release is connected to a sprint. A release should bring new functionality to the application. Release 1.0 is a complete application ready for public use. A release with an odd number is an internal release, and a release with an even number is a public release. A release by the second digit, as: 0.1, 0.2, 0.3 et cetera indicate a change in a part of the application. Releases with a digit-length of 0.1.1, 0.1.2 et cetera are bug-fixes.

Every release includes the following documents:

- Changelog
- Test report
- Installation and new user instructions
- Apk-package

The test report should contain the following information:
purpose of the application, general characteristics of the application, test environment details (hardware and software), known bugs and limitations and test specifications.

The change-log should include the following information: added features, removed features, changes, bug fixes, known bugs, planned changes.

When readying for a release the latest commit is merged to the master-branch of the repository and tagged using the following syntax:
RELEASE 0.2.0 m: [message].

4.2 Code commenting syntax

The project uses Java-doc for commenting code (http://en.wikipedia.org/wiki/Javadoc. The Java-docs contain information about the classes/methods such as general usage, @params to define what parameters the method request, @req to specify requirements that must be fulfilled for the method to run properly. @return is used to specify what the method will return when called upon.

An example of our documentation in the SettingsActivity is shown below:

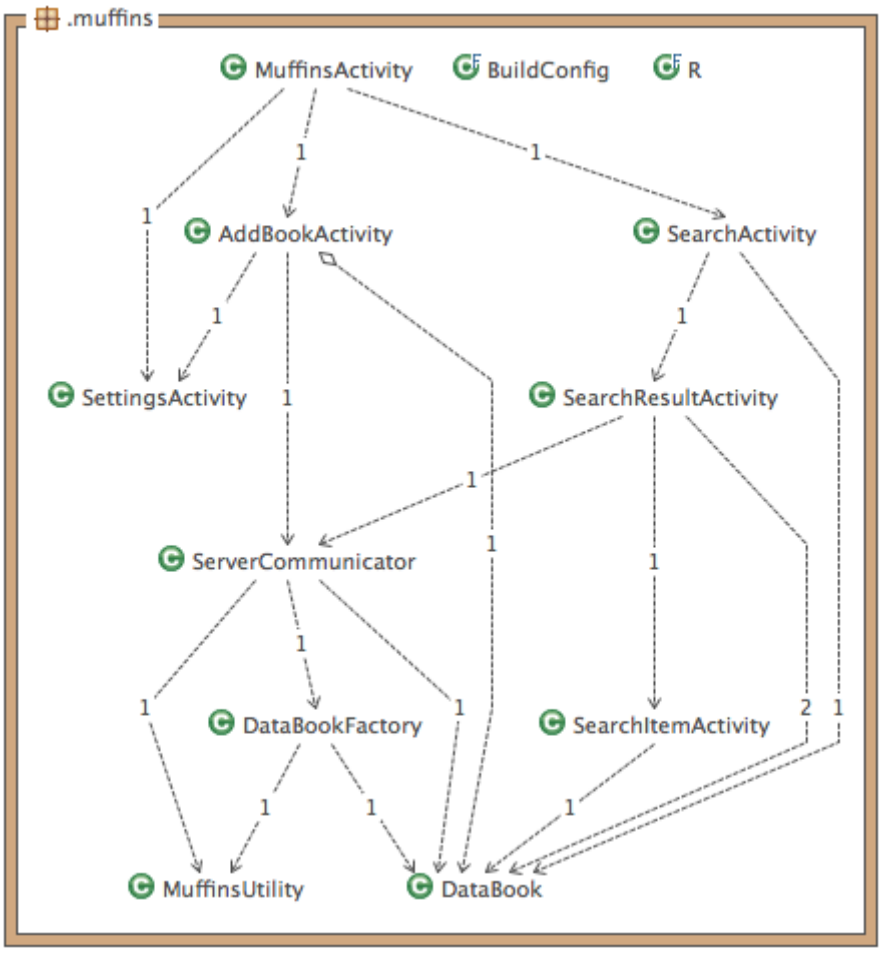
```
/**
 * Writes user details to private file in the internal storage of the
 * device.
 *
 * @param name
 *          Name of the application user
 * @param email
 *          the email
 * @param phone
 *          the phone
 * @param password
 *          the password
 * @req No null values. If no information is entered empty Strings should be
 *      provided.
 */
```

5. Classes and their functionality

ServerCommunicator: Class containg statics methods that makes calls to the server and saves books on the server or returns lists of books from a search. Uses DataBooks and DataBookFactory
DataBook: Class that contains all attributes a book up for sale can have including setters and getters. If an attribute that the book does not contained is called for the class will return null.
DataBookFactory: Class with static methods that converts DataBooks to JSONObjects and vice versa. Also makes calls to libris-database based on isbn-number and returns a book based on the results.

MuffinsUtility: Class containing all utility methods that does not logically fit anywhere else.
MuffinsActivity: A tab layout is used in the application and the source for this layout is contained within MuffinsActivity (extends TabActivity). Each tab is connected to another activity (SearchActivity, AddBookActivity and SettingsActivity).
SearchActivity: In this activity information is gathered from the user and sent to SearchResultActivity. Upon receiving search results SearchResultActivity is created and placed on top of SearchActivity.
SearchResultActivity: Receives a DataBook from SearchActivity and makes a call to ServerCommunicator in order to receive a list with the search results in form of DataBooks. This list populates SearchResultActivity.
SearchItemActivity: An activity presenting a book with all its information and is started when books are clicked in the list in SearchResultActivity. SearchItemActivity is placed over SearchResultActivity.
AddBookActivity: Collects data from the user and sends a DataBook to the ServerCommunicator with the information to be added to the database.
SettingsActivity: Collects, saves (persistent storage) and presents information about the user. This information is to be used when the user access the database to identify the user and authorize for example changes or adds.

This is an image showing how the dependency between classes look like.



6. Not implemented functionality to be done

This application can be used in various ways. For example, all universities in Sweden could have usage, and benefit, from this application. The aim is to make a quite generic application which easily can be customized for another university. The application could also be used in other areas, for example selling other products than second-hand books.

6.1 Functional requirements for add-on functionality

Requirement ID 6

A book could be sold in an auction

Requirement ID 6.1

Use case: **The seller can specify if he/she wants to have an auction, a buyout-price or both**

Requirement ID 6.2
Use case: **A book out for auction can have a reservation price**

Requirement ID 6.3
Use case: **A book out for auction have an ending date**

Requirement ID 6.4
Use case: **Anyone can make a bid for a book out for auction except for the owner of the auction ad/book**

Requirement ID 6.5
Use case: **A bid have to exceed the previous bid with at least 10 SEK and in even 10 SEK intervals**

Requirement ID 6.6
Use case: **If the time for an auction runs out, the book is automatically taken off the market.**

Requirement ID 7
Cremona Bookstore integration

Requirement ID 7.1
Use case: If a buyer search for a book without any sellers, the buyer should get information regarding if the book is available at Cremona Bookstore at Chalmers.

Requirement ID 8
Reservations of books

Requirement ID 8.1
Use case: When a reservation is made, the user making the reservation gets access to the buyer's contact information.

Requirement ID 8.2
Use case: A buyer can only make one book reservation at the time.

Requirement ID 9
Saving statistics

Requirement ID 9.1
Use case: The application saves statistics for book

pricing and number of sold books. This enables users to see interesting information regarding pricing.

Requirement ID 10, stakeholder SNI

Renting

Requirement ID 10.1

Use case: A book can either be sold or rented, or both.

Requirement ID 10.2

Use case: When a book is rented, the user should be able to take the book of market.

APPENDIX - Java doc summary