System design document for Studentbostäder Göteborg

Version: 1.0 - Final version
Date: 28/5-17
Author: Jonathan Gildevall, Peter Gärdenäs, Erik Magnusson, John Segerstedt

This version overrides all previous versions.

1 Introduction

The purpose of the application, Studentbostäder Göteborg, is to simplify and add a bit of extra convenience features to the tedious process of searching for student accommodations in Gothenburg. The application collects the accommodations from the two biggest suppliers of student accommodations, SGS and Chalmers. This combined with time saving features like getting notifications when an accommodation that suit your preferences become available.

The design of the system which Studentbostäder Göteborg is built on is described in this document.

1.1 Design Goals

The application has to be able to run on android as well as android emulators on desktop since not all developers have access to android devices for debugging.

It must also follow MVC, meaning the design pattern Model-View-Controller, in order to get a loosely coupled model that can be reused for other implementations. The service package that provides non application specific services must be loosely coupled with the rest of the application.

All code in the model must be testable.

1.2 Definitions, acronyms and abbreviation

Android: Mobile operating system.

Android activity: An Android activity contains one screen of the GUI of an Android app, similar to a window in a desktop app. It's also a java class which possibly contains both functions of a view class and a controller class.

Android drawable: A drawable is a general concept for a graphic that can be drawn on a Android screen.

Android intent:An intent is an abstract description of an operation to be performed. Mostly used in navigation.

Chalmers: Chalmers Studentbostäder, a student housing host.

Java: Programming language.

PMD: Source code analyzer, finds common programming flaws.

PNG: Portable Network Graphics, format for images.

Post/Get Request: Network request.

Push notification: Notifications sent to the user even though the application that the notification originates from isn't turned on or activated.

Request header: An attribute for a post/get request.

Search watcher: An autonomous service that observes an object feed until certain requirements are met.

SGS: Stiftelsen Göteborgs Studentbostäder, a student housing host.

STAN: A structure analysis tool for Java.

Studentbostäder Göteborg: The name of the application this document concerns.

Thread: A thread is an independent path of execution within a program. Many threads can run concurrently within a program.

TXT: Text file format.

XML: Extensible Markup Language, used in android for structuring design.

2 System architecture

The application is built on three different machines, an Android device and two servers provided by SGS and Chalmers respectively.

2.1 Android device

The main component, the Android device uses an Android application, Studentbostäder Göteborg, which is divided in two parts, background tasks and application. The parts utilize the same software, Android at API level 25. The background tasks are responsible for keeping the applications data up to date as well as notifying the user when new student accommodations that matches the criteria in a search watcher are available. The application is responsible for allowing the user to apply, browser and favorite accommodations as well as creating new search watchers.

The Android device communicate with the servers by sending post and get requests from the background tasks. The two parts of the application uses Android's broadcast receivers to speak with each other.

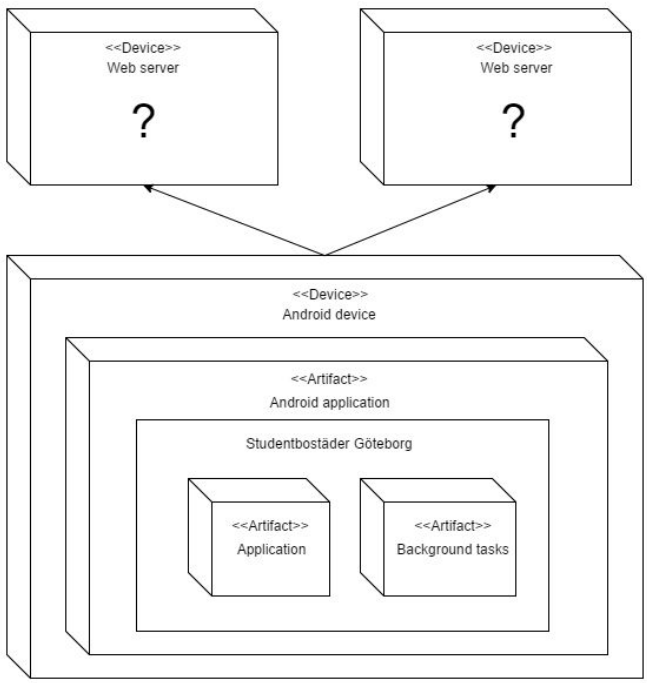

Figure 1: Deployment diagram for the application

2.2 Servers

Both the servers are utilized by the background tasks to fetch information about available student accommodations from respective company. Since both the servers are developed and owned by their respective companies no information about them is known.

2.3 Flow

General

The background tasks starts when the device is booted, since the they are designed to always be running. Other parts such as the application is started and closed as normal.

The search screen is the entry point of the application. When started the first time the model is initialized. Always when navigating to a page a new instance of said page is created rather than reusing an old one, the same goes in general for the controller and views.

Use case

The use case simple search exemplifies the standard sequence of the application well. The controller initialize the operation by calling the model. When the model is done, the view, which in this case is an adapter to the view, is called by the controller. If the view requires additional information it calls the model which responses with the required information and then displays the changes.

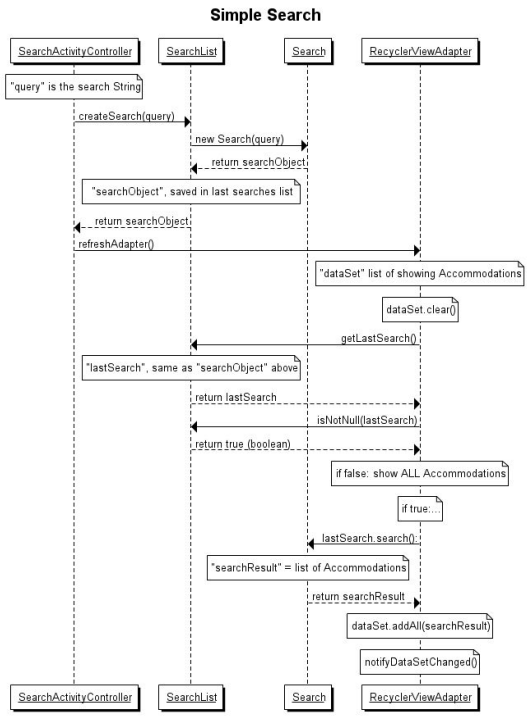

Figure 2: Sequence diagram of a Simple Search.

3. Subsystem decomposition

Only the Android application will be mentioned since the two other main systems were not developed by us.

3.1 Studentbostäder Göteborg

The application is divided in two subsystems, the activity part and background tasks part. Both of the subsystems use the model and service packages. By broadcasting and receiving Android intents to each other they can communicate and still avoid direct dependencies to each other.

3.1.1 Activity

The activity part is what the user would call the application, it contains everything which is visible and interactive. The activity section provides an environment for the user to properly search for a student accommodation by providing the tools necessary to search, browse, and organize the accommodations. Additionally it also allows user to create and edit search watchers and apply for student accommodations.

The activity section is composed of three unique packages, activity, view, and controller. More about them in 3.1.4 Package responsibilities.

3.1.2 Background tasks

The background tasks are autonomous tasks that run in the background providing new data for the application. The tasks are either initialized when the phone is booted or the first time the application is started, there is no way to manually stop them except for force stopping the application or turning of the device.

The processes fetches new data from the servers every six hours. The fetched data contains information about student accommodations which is converted to Java objects and saved into the D4bo database (more about this in 4. Persistent data management), images are also saved. All this enables the application to start without slow network processes making the task of starting the application significantly faster. Additionally while the new data is processed, the search watchers look for matches and if a match is found a push notification is sent to the Android device.

3.1.3 Model-View-Controller

The MVC architecture is implemented throughout the project. We have three distinct packages; model, view, and controller of whom the model is not aware of the controller or the view, and only the controller is aware of the view. However both the view and controller is aware of the model.

The views are divided in two different groups. One group consisting of XML files and the other of Java files. The XML files are responsible for visualising the data while the Java files are responsible for initializing and dynamically setting values for the XML files to display.

3.1.4 Package responsibilities

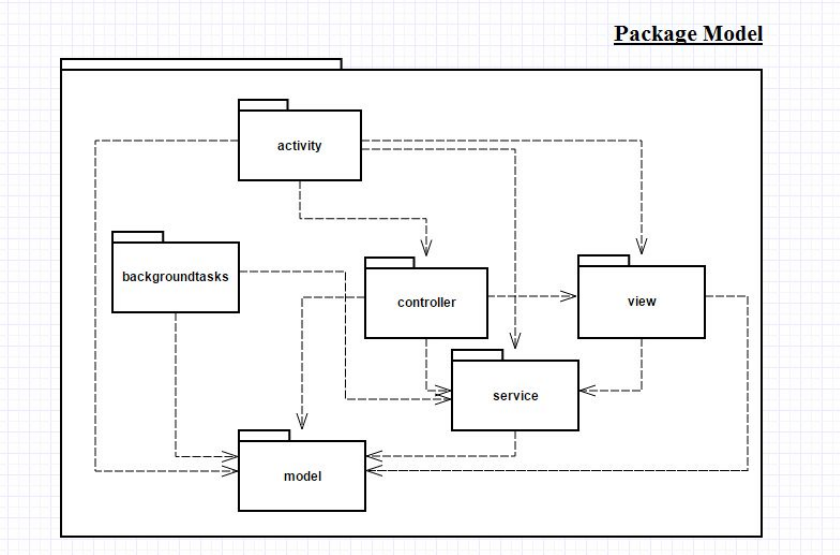

Figure 3: UML package diagram.

activity: Activities are responsible for initializing each page in the application. They create the controller, views, and processes, and sets the necessary connections between them. The activities contains close to no functionality.
backgroundtasks; The background tasks package is responsible for its own subsystem, maintaining data autonomously.
controller: Handles user interactions by notifying the relevant components such as the views or model.
model: The core of the application, contains the data and the logic concerning said data.
service: The service package is responsible for handling the application logic not covered by the views or model. This consists primarily of all network activities and file management.
view: Dynamically sets the content displayed on the application.

3.1.5 Concurrency issues

The application actively uses threads mainly because Android by default requires network activities to run on another thread than the main thread [1]. Other times threads are required to make the application feel fast and responsive.

Studentbostäder Göteborg solves possible concurrency issues in two different ways, either by forcing the application to wait until the asynchronous method is done or by making the program able to run independently of the thread.

The first way is used when either when the asynchronous process is critical or the process is in fact contains several asynchronous processes. One example of this would be when loading images for the

accommodations, having the screen changing content several times within a second will only irritate the user.

The other way is more suitable when it is only one process and it simply would be to slow to wait for the entire activity. How student accommodations are loaded from server and then displayed would be an example of that, during the process the user is free to browse the application.

3.1.6 Design Model

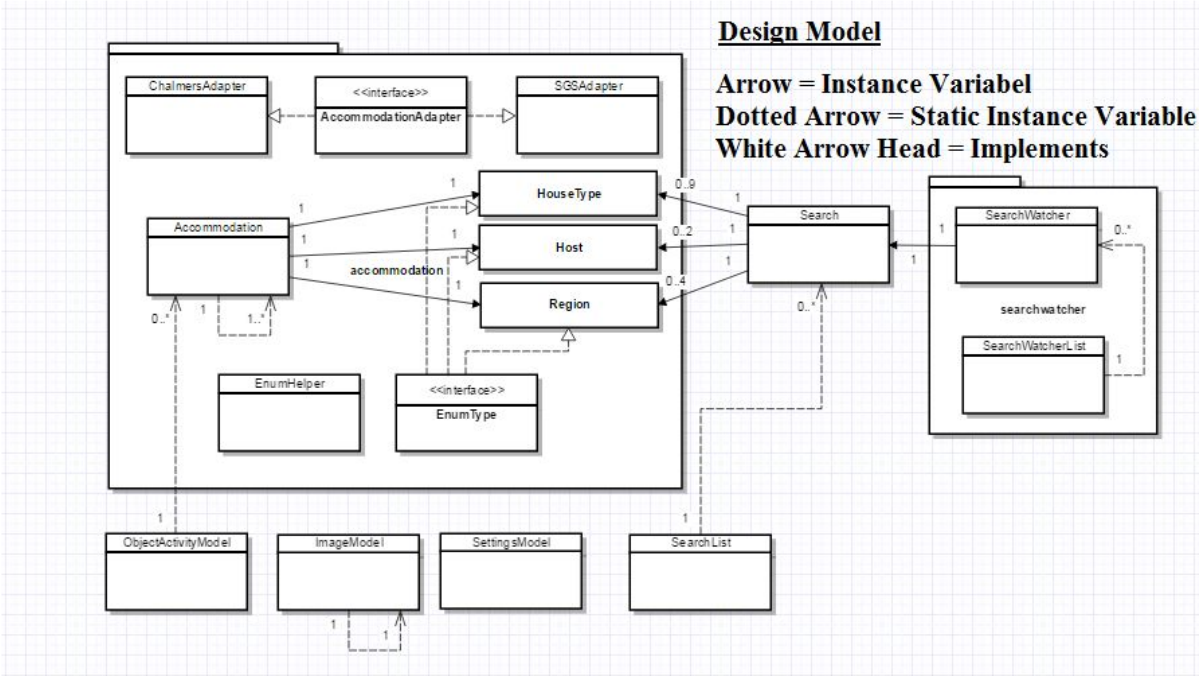

Figure 4: UML of the Design Model.

accommodation package:
Accommodation: The Java representation of a student accommodation.
ChalmersAdapter, AccommodationAdapter, SGSAdapter: These classes are responsible for converting text into useful Accommodation objects.
AccommodationHouseType, AccommodationHost, Region: Different possible values for respective class. Eg. AccommodationHost contains SGS and Chalmers as values.
EnumHelper: Provides useful methods for the classes implementing EnumTypes.

search watcher package:
SearchWatcherItem: Performs scans of newly added accommodations and notifies the user if any of them matches all the criterias defined in the search watcher.
SearchWactherList: Creates and stores search watcher objects

root package:
ImageModel: Contains all images related to the accommodations.

SettingsModel**:** Saves user settings and preferences.
ObjectActivityModel: Holds data for displaying a certain group of accommodations. In retrospect, this should not be part of the model, but rather of a view.
Search: Search through the accommodations after matches.
SearchList: Creates and stores search objects

3.1.7 Dependency analysis

Dependencies between our packages can be seen in figure 5 below, the diagram was rendered with the help of STAN. As can be seen there are no circular dependencies between packages. Additionally there are no dependencies between the two subsystems, background tasks and activity.

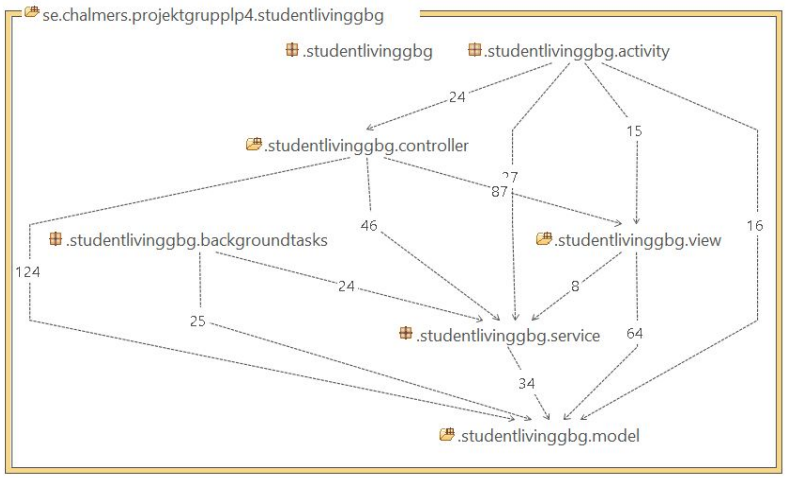

Figure 5: STAN diagram over dependencies between packages.

STAN reports five cases of circular dependencies within the packages, four of them includes the ActivityWithNavigation class, those dependencies are related to navigation and no good solutions to the circular dependencies was found.

3.1.8 Quality

Tests

Our goal of testing was to fully cover all methods in our model. A test class has been constructed for each normal model class which each cover 100 % of their respective model class. These include: Accommodation, ChalmersAdapter, EnumHelper, Host, HouseType, Region, Search, SearchList, SettingsModel, ImageModel, ObjectActivityModel, SGSAdapter, SearchWatcher, and SearchWatcherList. All of our test classes can be find in our project within the app/src/test directory.

Quality tool reports

Known issues include not using android's string resources for text on design elements such as buttons. These are hard coded which is bad if for future development because this increases the effort needed to add multiple languages. Another internationalization issue that was not addressed is the placement

of design elements, these have been placed among the right or left side etc. Which doesn't allow them to move to better adjust to languages where you read from the right to left.

PMD complains about catch blocks that just do a printStackTrace, these issues were ignored because it was deemed that the program could continue to run without handling the issue.

On the emulated desktop version of the application there exists many issues, for example surrounding navigation and updating data. Since this application has had its main focus as a mobile application, the bugs of the desktop version has been discarded.

4. Persistent data management

The application stores three kinds of things between sessions; Java objects, images, and text files. Java objects are stored by utilizing the db4o database library. The database allows storing Java objects in its entirety with no interfaces or attributes predefined. The database has clear limits, one being that it is no longer being developed [2], limiting the support.

The text files and images are stored differently than the Java objects, both are stored using Java's FileOutputStream. The text files contains the response from the post and get requests to SGS and Chalmers and are stored as txt files. While the images are displayed as Android drawable objects they are saved as PNGs in Android's internal storage.

Db4o is the prefered database since the file has no need to be converted to a Java object after load, the content can be used directly without extra methods. The main reason for using FileOutputStream as well is how the files are fetched. The files in db4o are always accessed in a group containing all files of the same Java class. All files of the same class is needed and equally important. While the files saved by FileOutputStream are found by name and it's likely that several files never gets loaded during a session. The db4o provides good support fetching all objects for each class but fetching a specific object requires extra implementation both in the database file and in the class, making it easier to just use FileOutputStream instead.

5. Access control and security

The application stores no sensitive information about the user or device. The only information leaving the application is the information sent through the post and get requests. The application does not have full control of the content leaving the application since some headers can't be modified [3]. The modifiable headers does not alter for user to user and contains nothing sensitive. The unmodifiable contains nothing sensitive as well, all request headers can be seen in appendix I and II.

The information received from the requests is not validated in any way before it's saved to disc. For loading, the only requirement is that the file's data follows a specific structure, the data which is later displayed, will not be checked in any other way.

6. References

[1] Google Inc., “NetworkOnMainThreadException”, [Online], Tillgänglig: https://developer.android.com/reference/android/os/NetworkOnMainThreadException.html. Hämtad 28 maj, 2017

[2] actian, “We are restructuring our Versant Community Website”, [Online], Tillgänglig:, http://supportservices.actian.com/versant/default.html. Hämtad 28 maj, 2017

[3] F. Scholz, C. Mills, A. Pfeiffer, K, Scarfone, “Forbidden Header Name”, *Mozilla Foundation*, 2016, [Online], Tillgänglig: https://developer.mozilla.org/en-US/docs/Glossary/Forbidden_header_name. Hämtad 28 maj, 2017

7. Appendix

I.

```
POST /API/Service/SearchServiceHandler.ashx HTTP/1.1
Content-EnumType: application/x-www-form-urlencoded
X-Momentum-API-KEY: u15fJ8yRMCIu////+aEYR7+XJwj1hiE9gIXfoo/eje4=
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36
Accept: application/json,text/*
Accept-Language: en-US,en;q=0.8,sv;q=0.6,und;q=0.4
Origin: https://marknad.sgsstudentbostader.se
Referer: https://marknad.sgsstudentbostader.se/
X-Requested-With: XMLHttpRequest
Content-Type: application/x-www-form-urlencoded
Host: marknad.sgsstudentbostader.se
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Length: 873
```

Header for post request to SGS

II.

```
GET /widgets/?actionId=&omraden=&objektTyper=&maxAntalDagarPublice
rad=&callback=jQuery172027556341802877515_1492691004676&widgets
%5B%5D=alert&widgets%5B%5D=objektfilter%40lagenheter&widgets%5B
%5D=objektsummering%40lagenheter&widgets%5B%5D=objektsortering&widgets
%5B%5D=objektlistabilder%40lagenheter&objektfilter
%40lagenheter.maxyta=130&objektfilter%40lagenheter.maxhyra=14000 HTTP/1.1
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; HUAWEI VNS-L31 Build/
HUAWEIVNS-L31)
Host: www.chalmersstudentbostader.se
Connection: Keep-Alive
Accept-Encoding: gzip
```

Header for get request to Chalmers