System design document for Pinjobs project (SDD)

Contents

Version: 1.0

26th May 2015

Authors: Filip Slottner Seholm, Filip Larsson, Isaac Gonzalez, Carl Albertsson

This version overrides all previous versions.

1. Introduction

1.1 Design goals

The design must be loosely coupled to make it possible to switch GUI as well as switch database and extend the program with new features. The design must be testable which means it should be possible to isolate models for test.

1.2 Definitions, acronyms and abbreviations

Parse - cloud database using NoSQL

NoSQL - database that can handle big data and multiple queries.

MVC - model-view-controller, design pattern used for…

Java - object oriented programming language.

Android Studios - integrated developer environment for develop android applications.

GUI - graphical user interface.

Xml - extensible markup language, documentation readable for both human and computer.

Activity - A android specific class which works as a controller in the system.

Package by feature - package structure where one package contains classes specific for one feature in the application.

Ad - A Class in the system which represents an advertisement that consists of different attributes.

Intent - An object used to start an activity, which contains different information about the activity the intent is sent from and can be used to send additional data to the new intent.

Serializable - An interface which, if implemented, make objects of custom created classes containing only types already defined in java available for sending through intents via the putExtra method.

Parcelable - An interface which, if implemented, make objects of custom created classes containing serializable objects available for sending through intents via the putExtra method.

Service - There are two classes that have all the communication with the database. These classes are called Services.

API - application programming interface

2. System design

2.1 Overview

The application uses a modified android MVC structure.It also uses a modified “package-by-feature” structure and services classes for communication with a database.

2.1.1 MVC-android structure

The MVC design we have used is to have models as plain java classes independent of any android specific classes. All the models functionality are represented by an individual interface.

The controllers are represented as subclasses of the android specific class Activity. The system consists of one controller for each view with a similar name as the view. ie. CreateAdView has the controller CreateAdActivity.

The view is represented by both .xml files and Java classes where data from the .xml file to the Java view file is given by sending the Activity to the view. The view itself then extracts the references of the desired components. The .xml file holds the code for the elements of the view and the java class is used to manipulate these elements.

2.1.2 Package-by-feature

We have decided to use a package by feature structure, which means that we have created packages for every small module that is independent. in every package we have a model and a view and in some packages some extra utility classes. To avoid circular dependencies we decided to put all the controllers in one package to make easy communication between the controllers possible.

2.1.3 Services for database

To communicate with the database without compromising the sustainability and exchangeability of the program we decided to put all the communication in two Service classes. These classes also have their own interfaces. To change storage from our database Parse to some other data storage service all one needs to do is change the Service class.

2.1.4 Communication between controllers

The communication between the controllers are a bit different compared to a normal Java application. Since the program is android based and uses Activities as controllers the communication between the activities is used by intents.

Intents works almost like a constructor in the way that when an Intent is called to an activity, the activity starts and calls the onCreate method. If you want to pass some kind of data through an intent this is done by a putExtra method which takes a keyword that the data is related to. To send own objects unknown by the android API the class that is sent though the extra method has to implement either the interface Serializable or Parcelable.

There is also a StartforForResult method for intents which is used pretty frequently. It starts an activity and when the newly started activity is done and the finish method in that activity is called the onActivityResult method in the first activity is called. This is very useful because the second activity can finish and send that data back to the creator activity without any knowledge of the creator activity. This is done to stand clear of circular dependencies between the controllers.

2.2 Software decomposition

2.2.1 General

The application consists of following modules, see figure 1.

advertisement - Contains all classes relevant for creating and modifying ads. Some classes for showing ads created by the user. Contains both model and view parts for MVC

main - Contains main view, background thread and usermodel class. View and model for MVC

controller - Contains all the activity-classes. Contains controller for MVC.

profile - Contains all classes relevant for creating, modifying and showing profiles. Contains both model and view parts for MVC.

handler - Contains classes for showing all advertisements in the system. View and model for MVC.

user - Contains classes for logging in to a user's profile. View and model for MVC.

services - Contains classes for store and retrieve data from the Parse database.

utils - Contains a class for getting the user's position and one class for checking if information is put in correctly when getting input from the user.

2.2.2 Decomposition into subsystems

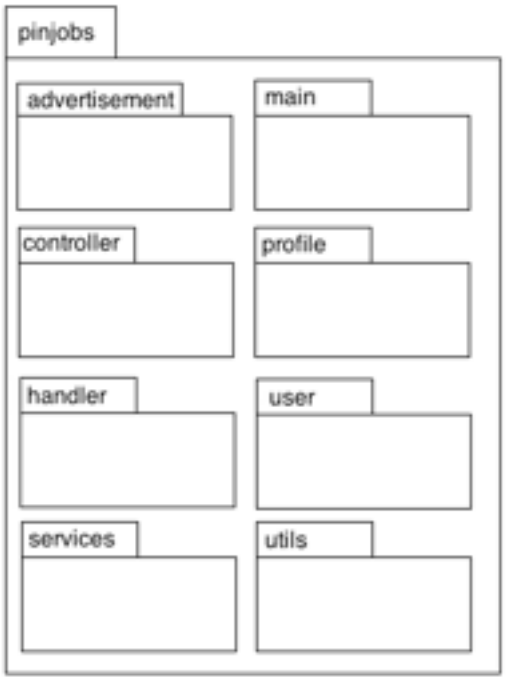
***** Fråga handledare!! *******

2.2.3 Layering

The Figure below shows the applications layering. The top of Figure 2 contains the higher layers.

2.2.4 Dependency analysis

Dependencies are as shown in Figure. There are no circular dependencies.



Figur 1.

(BILD PÅ STAN)

Figur 2.

2.3 Concurrency issues

The program uses a background thread to receive a list of advertisements but because Parse is a NoSQL database no concurrency issues is possible. NoSQL database can handle multiple queries.

2.4 Persistent data management

The application don’t contain much data itself but it handles data contributed from the user such as profiles and advertisements. To store and retrieve this data we use the Parse database.

2.5 Access control and security

Could be some issues since the application depend on Parse. If Parse would fail with something there is nothing the application can do about it.

2.6 Boundary conditions

NA. The application is launched and exited as a regular android application.

3. References