example sdd: http://www.cse.chalmers.se/edu/course/TDA367/case/SDDMonopoly.pdf

System design document for Gasqueue

Version: 0.2

Date 10-05-2016

Author :Kevin,Petros,Long,Eric

This version overrides all previous versions.

1 Introduction

1.1 Design goals

Creating loosely coupled components to be reused in other systems with different views. E.g model should be able to be reused, controller should be able to handle different views. The code should also be testable.

The model should be loosely coupled to the rest of the program to allow it to be used in other systems and with different views as well. It must also be designed to that it is testable so that errors can be established.

1.2 Definitions, acronyms and abbreviations

2 System design

2.1 Overview

The application uses an modified MVC model suited for the Android design principle. Instead of using the Activity class as the main controller, other controller class are made that connects the activity with the model and database. By doing this the application is not dependent on the android Activity and most of the components can be reused with another View, e.g. Swing.

2.1.1 Unique Identifiers

Every user will have an clientID based on their android phone. The clientID will be used to track tthe user. E.g tracking the order made by the customer throught his/her clientID.

2.1.2 Event paths

User inputs will be handled by the following chain: View->Activity->Controller class->Model

- All updates in model will be through setters
- Sending data to the database will be handled by the controllers
- Retrieving data from the database will be handled by event listeners

2.2 Software decomposition

2.2.1 General

Activity, contains all GUI related classes
Model, containts all the OO-models
Controller,contains all the controller class
Service, contains all service related content. E.g api to access database

2.2.2 Decomposition into subsystems

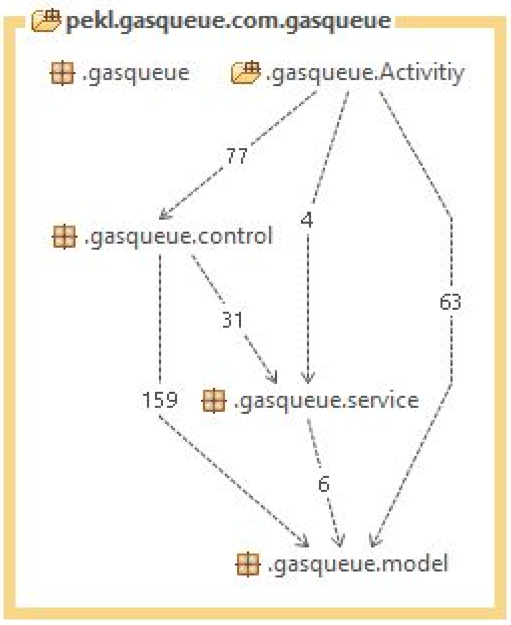
Database, responsible for storing data.

//Database interface

```
public interface IDatabaseManager {

   public void saveMap(String address, Map<String, List<Product>> map);

   public void addToMap(String address, String clientID, List<Product> list);

   public void saveStringList(String address, List<String> list);

   public void sendObject(String address, Object object);

   public Firebase createChildReference(String childReference);

   //public void addUpdateListener(String address);

   public boolean checkCode(String barCode);

}
```

2.2.3 Layering

2.2.4 Dependency analysis
Dependencies in the figure below.



//todo

2.3 Concurrency issues
//todo

2.4 Persistent data management
All persistent data will be stored on Firebase. All data can be accessed and altered through the database API.

2.5 Access control and security

2.6 Boundary conditions

3 References

APPENDIX