<WasteLess A1>
Analysis and Design Document

Student: Andrei Rusu
Group: 30431

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

The purpouse of this assignment was to create an application using any language that would fullfill a few requirements, detailed below.

Functional Requirements

The app needs to have a login capability, where a user can authenticate and add grocery lists with the items they want to consume. The system should, based on this lists, calculate how much food is being wasted by the user, based on their calorie burn rate, notify them about expiring items and offer options for the excess food to be donated to charities where possible.

The application needs to work with a database, and have an ORM (Object Relational Mapper) for interacting with the database. The choice I made was to use the Hibernate ORM with Java and a MySql database.
Also, a Dependency Injection container was required, and I managed to use PicoContainer for this task.

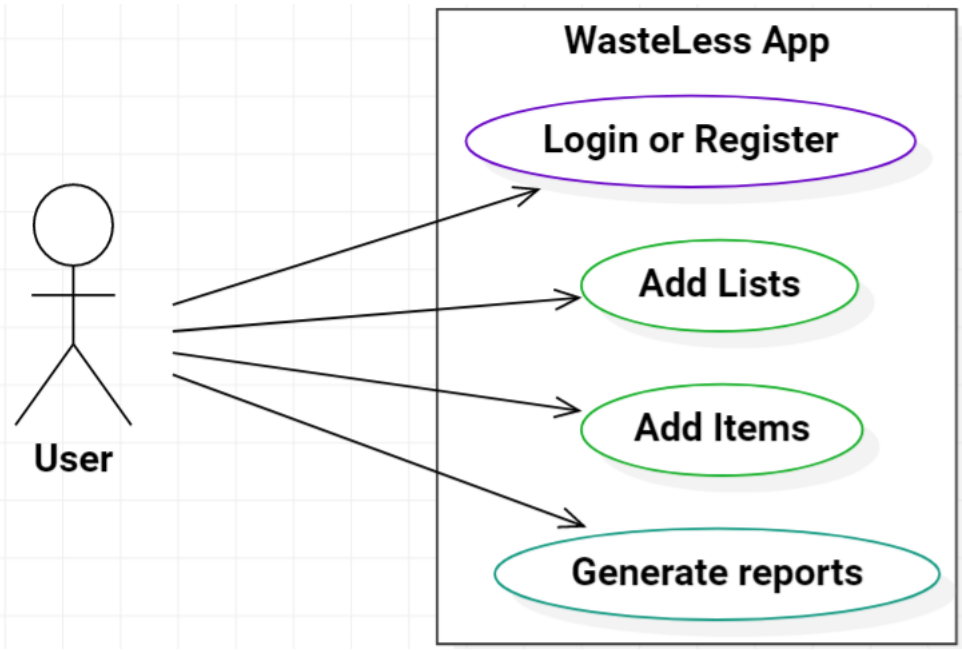
1.2 Non-functional Requirements

The application was required (if not explicitly, then by common sense) to be maintainable (hence the modularity), testable (point 7), have good usability, and also, from a further development standpoint, to be scalable and extensible further on.
Many of these requirements are fullfiled by using a good architecture, logic and coding style.

2. Use-Case Model

Use case: <Add an item to a list>
Level: <use-goal level>
Primary actor: <the user of the application>
Main success scenario: <Start the application, login or register, add a new list or select an existing one, fill out the new item details and add it to the current list>
Extensions: <failure if wrong data is input>
]

The use-case diagram is presented below:



3. System Architectural Design

3.1 Architectural Pattern Description

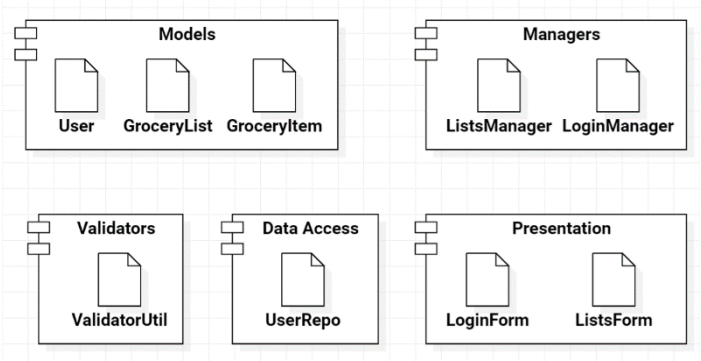
The system is based on a layered architecture, consisting of 4 layers:

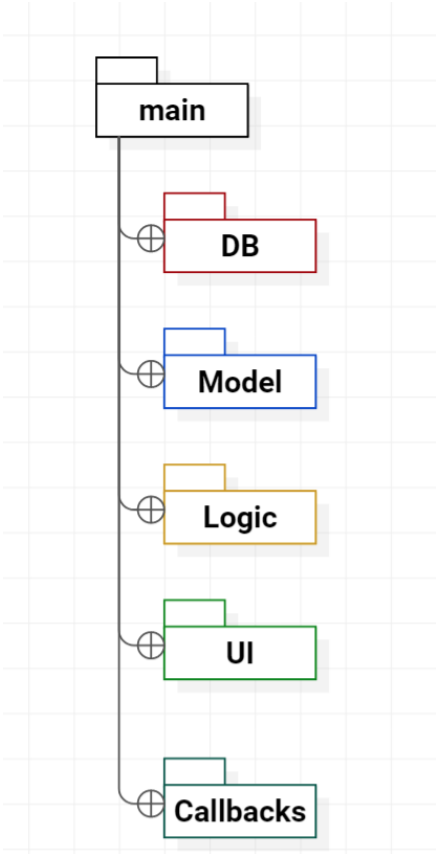
- The **presentation** layer, where the UI of the application is found;
- The **application/business logic** layer, where operations are performed using other components;
- The **persistence** layer, where the enitites of the application are found, like the User class;
- The **database** layer, consisting of the components that deal with the data source (DB) and persistence, like the ORM

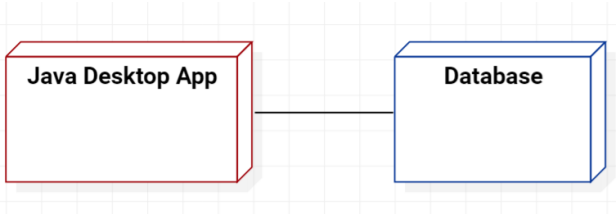
The dependencies go in a downward, meaning, for example, that the UI does not do any processing itself, it only calls methods in the business logic layer, which then goes on to read/write to the database thorugh dedicated methods in the database layer.

3.2 Diagrams

The component, package and deployment diagrams are shown below:







As described in the above section, the 4 layers interact with each other sequentially, usually starting with the UI, where the user presses or does something that triggers a listener to send a command to the manager, which then performs some taks and processes data, and then it sends a command to the data access components in order to read from or update the database.

In reality, the data access classes also have access to the system entities, due to the fact that the ORM software uses them, so there is a slight misaligment from the standard layered architecture pattern, but this is common in real applications.

4. UML Sequence Diagrams

[Create a sequence diagram for a relevant scenario.]

5. Class Design

5.1 Design Patterns Description

The abstract factory design pattern was used to generate the reports of wasted food, as per the requirements.

5.2 UML Class Diagram

[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]

6. Data Model

[Present the data models used in the system's implementation.]

7. System Testing

[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]

8. Bibliography