Car Renting Application

Analysis and Design Document

Student: Voicu Sara-Ioana

**Group:30431**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 11.04.2023 | 1.0 | Domain model and architectural design | Voicu Sara-Ioana |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

# The car rental application is a web-based platform that allows customers to rent cars from a specific car rental company. The application will be developed using the Java Spring framework.

# The objectives of the project are:

# To create a web-based car rental application that allows users to search for and book rental cars from a specific car rental company.

# To provide an easy-to-use and intuitive interface for users to search for available cars based on their preferences, such as location, car type, rental period, etc.

# To implement a secure and reliable system that allows users to make online payments, manage their reservations, and communicate with the car rental company.

# To ensure high performance and scalability of the application, so that it can handle a large number of users and transactions.

# Elaboration – Iteration 1.1

# Domain Model

*Diagram

Description automatically generated*

This is an idea of how the class diagram may look.

The model contains three users: Administrator, Client, Employee. The administrator can have many cars, so we have a one-to-many relationship. The employee can edit the availability of the cars. The client can rent how many cars as he wants, and a car can be rented many times, so in this case we have a many-to-many relationship. The classes include the attributes, but not the methods yet. All of them must have getters, setters, and constructors.

# Architectural Design

## Conceptual Architecture

## Layered architecture is a widely used software design pattern that divides an application into distinct layers, each with a specific set of responsibilities.

## Persistence Layer: This layer is responsible for handling the data storage and retrieval of the application. It includes components such as data access objects (DAOs), database connectors, and ORM frameworks that handle the low-level details of interacting with the database. The persistence layer is typically responsible for implementing CRUD (Create, Read, Update, Delete) operations and ensuring the consistency of data. Here I will have the repositories, for each model.

## Model Layer: This layer represents the data models and domain objects of the application. It includes components such as entity classes, value objects, and data transfer objects (DTOs) that encapsulate the data and behavior of the system. The model layer is typically responsible for defining the structure and relationships of the data models.

## Controller Layer: This layer handles the user input and output of the application. It includes components such as controllers, views, and templates that manage the interactions between the user and the system. The controller layer is typically responsible for validating user input, processing requests, and rendering responses.

## Business Layer: This layer contains the business logic and rules of the application. It includes components such as services, workflows, and validators that handle the complex logic and interactions of the system. The business layer is typically responsible for implementing the core functionality of the application, such as handling transactions, enforcing security, and performing calculations.

## Overall, this architecture can help to separate the concerns of your application into distinct layers, making it easier to maintain, test, and scale the package design.

## Package design

The Controller package depends on the Business package since the controllers typically invoke the services and workflows provided by the business layer to implement the logic of the application.

The Business package depends on the Model package since the services and workflows typically operate on the data models and domain objects defined in the model layer.

The Business package also depends on the Persistence package, since the services and workflows typically interact with the data access objects (DAOs) provided by the persistence layer to store and retrieve data from the database.

The Persistence package depends on the Model package, since the DAOs typically work with the data models and domain objects to read and write data from the database.

Diagram

Description automatically generated

## Component and Deployment Diagrams

Component diagram:

Diagram

Description automatically generated

Deployment diagram:

Diagram

Description automatically generated

# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

**Sequence diagram:**

Diagram

Description automatically generated with low confidence

**Communication diagram:**

**Diagram

Description automatically generated**

## Class Design

*Diagram

Description automatically generated*

# Data Model

The project contains the following entities:

* **Cars:** represents the model for the car, containing details about the car like the brand, the model, the price, the kilometers, as well as the id of the administrator.
* **Rental:** represents the model for the renting part, when one client decides that he wants to rent the car. So, it will contain the following attributes: an id, a car, and a client id.
* **Contract:** represents the model for the contract that needs to be signed by the client, so it will contain the signature of the client.
* The model also contains three classes for three types of users: **Client**, **Administrator** and **Employee**. They contain details about the users, like username, password, email, full name, age, etc. They also have many relations with the other entities from the model package.

# Unit Testing

For making sure that the application works correctly, JUnit testing was used, where JUnit is a popular unit-testing framework in the Java ecosystem.

For this I created a new class called Test Service, where I have the annotation @Test, which tells the Junit that the public void method in which it is used can run as a test case.

I have 1 test: the test tests the method for creating an employee.

A screenshot of a computer

Description automatically generated with medium confidence

# Elaboration – Iteration 2

# Architectural Design Refinement

Deployment diagram:

Diagram

Description automatically generated

Layered architectures are said to be the most common and widely used architectural framework in software development. It is also known as an n-tier architecture and describes an architectural pattern composed of several separate horizontal [layers](https://www.baeldung.com/cs/layers-vs-tiers) that function together as a single unit of software. A layer is a logical separation of components or code.

Now, the number of layers in a layered architecture is not set to a specific number and is usually dependent on the developer or software architect. It is important to note that this framework will usually always have a user interaction layer, a layer for processing, and a layer that deals with data processing.

Spring Boot is a module of the Spring Framework. It is used to create stand-alone, production-grade Spring Based Applications with minimum effort. It is developed on top of the core Spring Framework. Spring Boot follows a layered architecture in which each layer communicates with the layer directly below or above (hierarchical structure) it.

Before understanding the **Spring Boot Architecture**, we must know the different layers and classes present in it. There are **four** layers in Spring Boot are as follows:

* **Presentation Layer:**

The presentation layer handles the HTTP requests, translates the JSON parameter to object, and authenticates the request and transfers it to the business layer. In short, it consists of **views** i.e., front-end part.

* **Business Layer:**

The business layer handles all the business logic. It consists of service classes and uses services provided by data access layers. It also performs authorization and validation.

* **Persistence Layer:**

The persistence layer contains all the **storage logic** and translates business objects from and to database rows.

* **Database Layer:**

In the database layer, CRUD (create, retrieve, update, delete) operations are performed.

Diagram

Description automatically generated

# Design Model Refinement

*A screenshot of a computer program

Description automatically generated with low confidence*

# Construction and Transition

# System Testing

For testing the operations that each user can make I used requests in Postman API.

Postman is one of the most popular software testing tools which is used for API testing. With the help of this tool, developers can easily create test, share, and document APIs. I used requests like: POST, DELETE, PUT and GET.

* POST: to create something and send it
* GET: to get something
* DELTE: to delete something
* PUT: to update some information

# Future improvements

For future improvements I would like to have a fronted part for the application, in React, so that the application can be friendlier for the user. I could also add some more entities and methods, such that a client can add his own car on webpage, to be rented by other clients. In this case, not only the administrators can be the owners of the cars.

# Bibliography

<https://www.baeldung.com/java-observer-pattern>

<https://www.tutorialspoint.com/design_pattern/iterator_pattern.htm>

<https://www.baeldung.com/spring-email>

<https://www.youtube.com/watch?v=KxqlJblhzfI>

<https://www.youtube.com/watch?v=9SGDpanrc8U&t=3144s>

<https://www.youtube.com/watch?v=bxy2JgqqKDU&t=5364s>

<https://stackoverflow.com/>

<https://www.baeldung.com/spring-boot-data-sql-and-schema-sql>

<https://www.geeksforgeeks.org/spring-boot-crud-operations/>

<https://springjava.com/spring-boot/export-data-into-csv-file-in-spring-boot>