Driving School Management System

Analysis and Design Document

Student:Fazekas Vlad

Group:30234

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Table of Contents

I. Project Specification

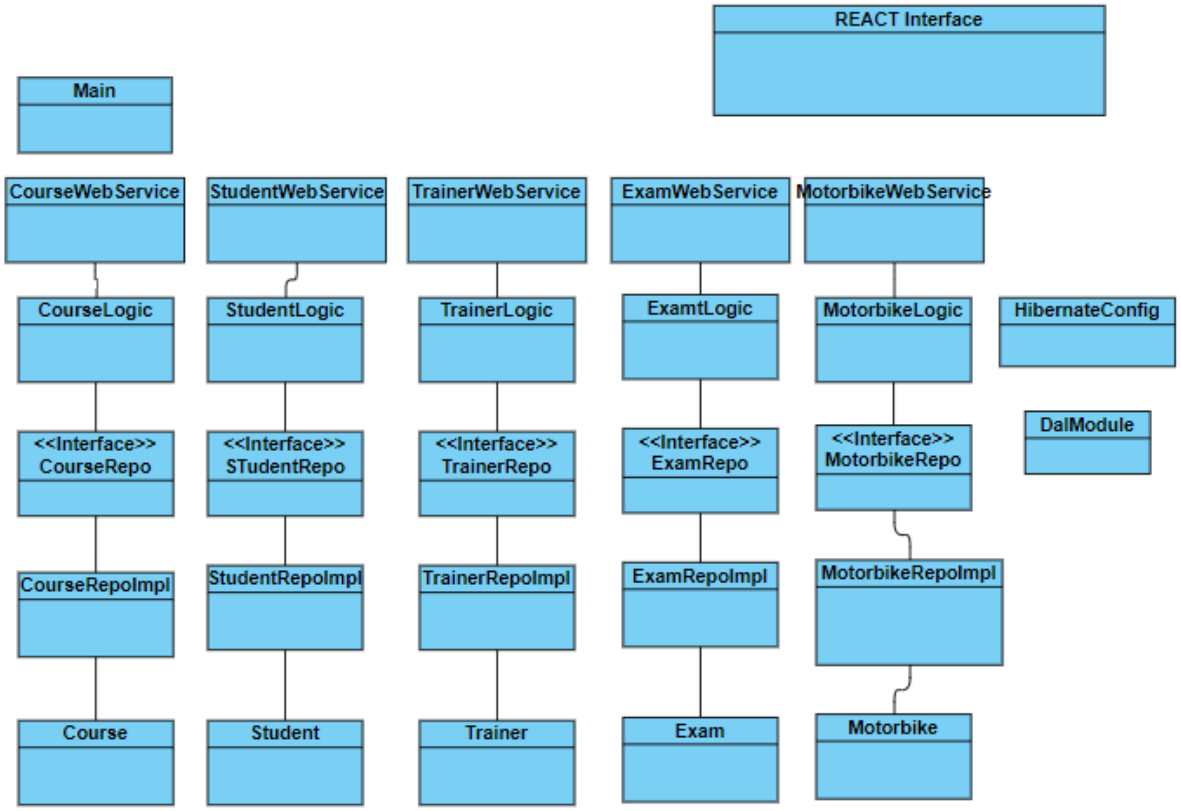
A web application for the management of a driving school, that includes the interaction from the user/students and from the administrators/trainers of the driving school.

There are two types of users:

- Trainers – will be able to see their students, see raports, see who is going to have the exams next week, add exam questions, keep track of the motorcycles
- Students – will be able to take exams, change their data, chose the motorcycles

II. Elaboration – Iteration 1.1

1. Domain Model



2. Architectural Design

2.1 Conceptual Architecture

The systems architecture will be based on the Layers architectural pattern and model-view-controller pattern.

Dividing an application into separate layers that have distinct roles and functionalities helps you to maximize maintainability of the code, optimize the way that the application works when deployed in different ways, and provides a clear delimitation between locations where certain technology or design decisions must be made.

That's why we will divide our application in three big layers:

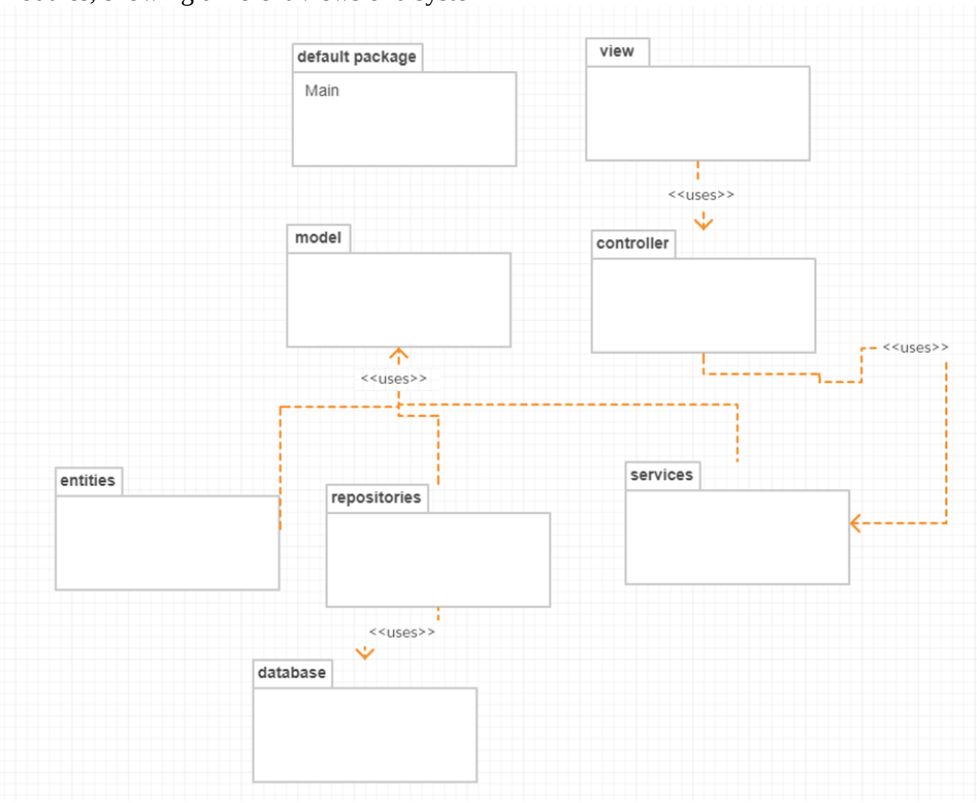
- Presentation layer (where all the interaction of the user with the system will reside)
- Business layer (where the core functionality and relevant logic of the system will be implemented)
- Data layer (where access to data is provided(generally through generic interfaces that the components in the business layer can consume)

Model–View–Controller (usually known as MVC) is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. It also provides a much clearer overall view of the system and a more structured code that is easier to expand with new functionalities. That's why we will divide our application in the following structure:

- Controller
- View
- Model
  - Entities
  - Repositories
  - Services
- Database

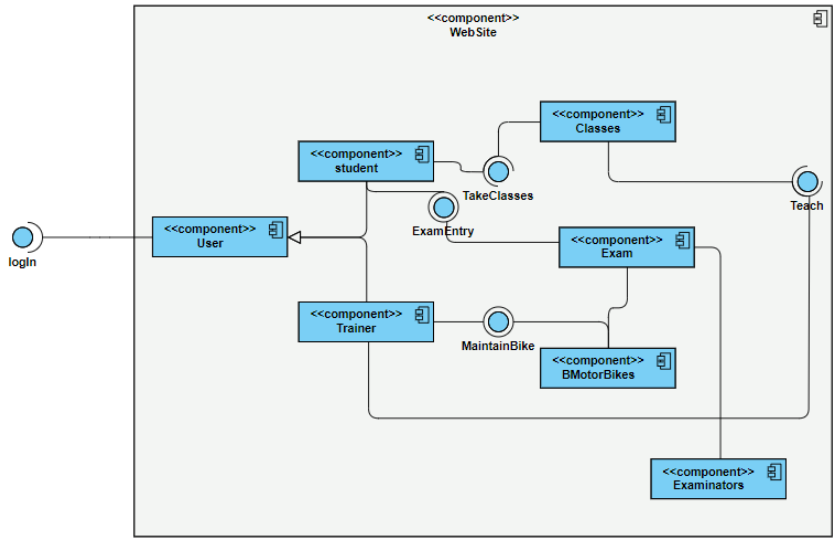
2.2 Package Design

Package diagram, a kind of structural diagram, shows the arrangement and organization of model elements in middle to large scale project. Package diagram can show both structure and dependencies between sub-systems or modules, showing different views of a system

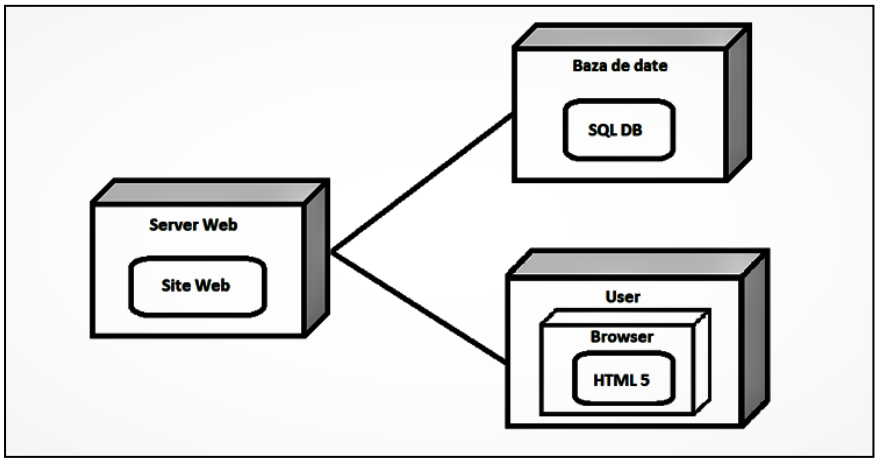


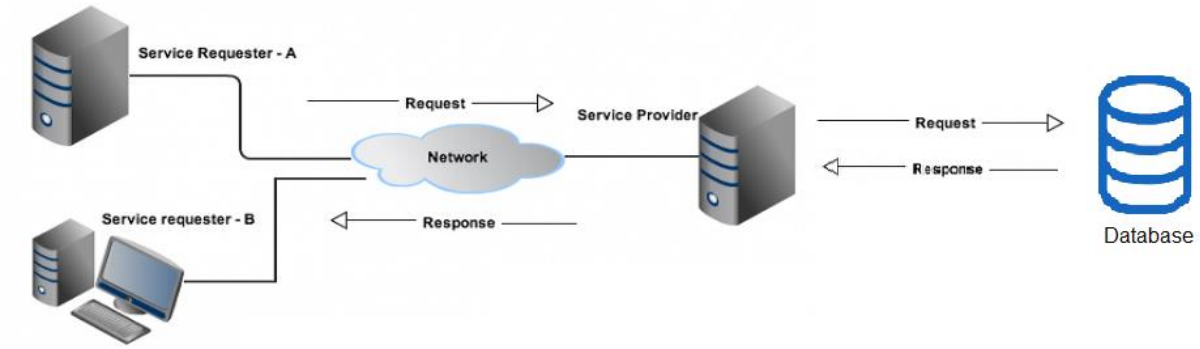
2.3 Component and Deployment Diagrams

Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.



A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).





III. Elaboration – Iteration 1.2

1. Design Model

1.1 Dynamic Behavior

[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]

1.2 Class Design

[Create the UML class diagram; apply GoF patterns and motivate your choice]

2. Data Model

[Create the data model for the system.]

3. Unit Testing

[Present the used testing methods and the associated test case scenarios.]

IV. Elaboration – Iteration 2

1. Architectural Design Refinement

[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]

2. Design Model Refinement

[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]

V. Construction and Transition

1. System Testing

[Describe how you applied integration testing and present the associated test case scenarios.]

2. Future improvements

[Present future improvements for the system]

VI. Bibliography