CsCompany

Project documentation

Student(s):

* Csatlos-Koncz Andrei

**Group: 30432**

**Contents**

[I Project specification 3](#_Toc104198264)

[1.1 Domain Model Diagram 3](#_Toc104198265)

[II Use-Case model 4](#_Toc104198266)

[2.1 Use-Case identification 4](#_Toc104198267)

[2.2 UML Use-Case diagram 4](#_Toc104198268)

[III Architectural design 5](#_Toc104198269)

[3.1 Conceptual architecture 6](#_Toc104198270)

[3.2 Package diagram 7](#_Toc104198271)

[3.3 Class diagram 8](#_Toc104198272)

[3.4 Database (E-R/Data model) diagram 9](#_Toc104198273)

[IV Supplementary specifications 9](#_Toc104198274)

[4.1 Non-functional requirements 9](#_Toc104198275)

[4.2 Design constraints 10](#_Toc104198276)

[V Future improvements 10](#_Toc104198277)

[VI Bibliography 10](#_Toc104198278)

# I Project specification

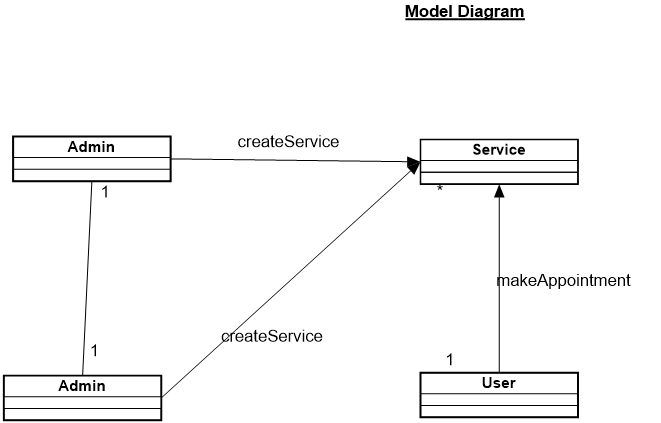
My project is a web application that consists in a software/hardware company with multiple employees to which one can make an appointment and choose his preferred employee for a software or hardware consultation. Customers could make accounts for longer consulting or for just a one-time appointment, they can simply input their e-mail and phone number.

Users can also input and search their past or future appointments by entering their e-mail. Everything will be send to e-mail for privacy reasons. If they have an account, they could see their appointments in their accounts.

This is a needed application for my software company for which I have a website but it is not so complex.

## 1.1 Domain Model Diagram

**Employee**



# II Use-Case model

The Use Case Diagram has 3 actors: admin, user and employee.

The admin can manage employees and add services.

The employee can add services, modify availability and manage appointments. He is created only by the admin.

The user can create an account, make appointments, contact an employee and see all of his appointments.

## 2.1 Use-Case identification

Use case name: Choose employee and department

Level: User-Goal

Main Actor: User

Main success scenario: The user will be redirected to make an appointment in that department for the chosen employee

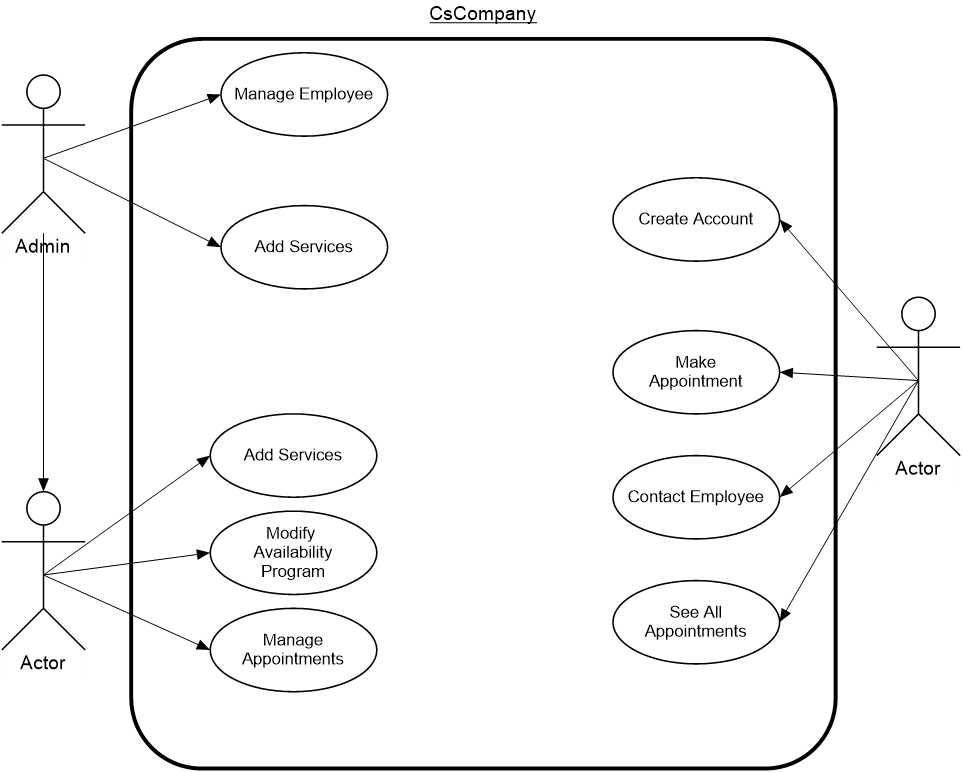
Use case name: Make appointment

Level: User-Goal

Main Actor: User

Main success scenario: An appointment is created in the database with the user id and employee’s id. The appointment is later displayed on the employee’s page.

## 2.2 UML Use-Case diagram



# III Architectural design

The application will use a layered architecture in order to organize the components in

a better way and group together related classes. The layers are the following:

Presentation Layer, Business Layer, Persistence Layer and Database Layer,

each having a specific purpose.

Keeping this in mind, the project will have 4 packages: Controller – with all the

presentation classes, Service – with all the bussiness layer classes which contain

the functional requirements of the application, Repository – with the classes which

directly access the database and Model – containing the model classes which

directly map the database tables.

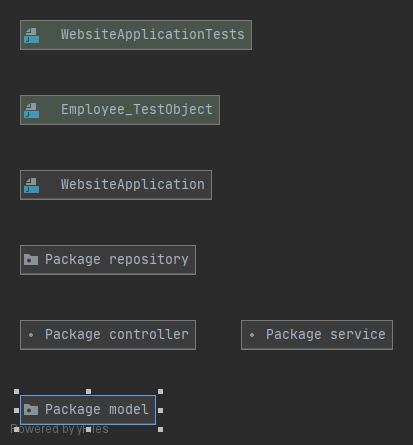
## 3.1 Conceptual architecture

My application will be a web application made with Java and Spring framework and the front-end part will be made using React. The app will respect the MVC architecture (Model View Controller). MVC is known as an architectural pattern, which embodies three parts Model, View and Controller, or to be more exact it divides the application into three logical parts: the model part, the view and the controller. Model: Handles data logic. View: It displays the information from the model to the user. Controller: It controls the data flow into a model object and updates the view whenever data changes.

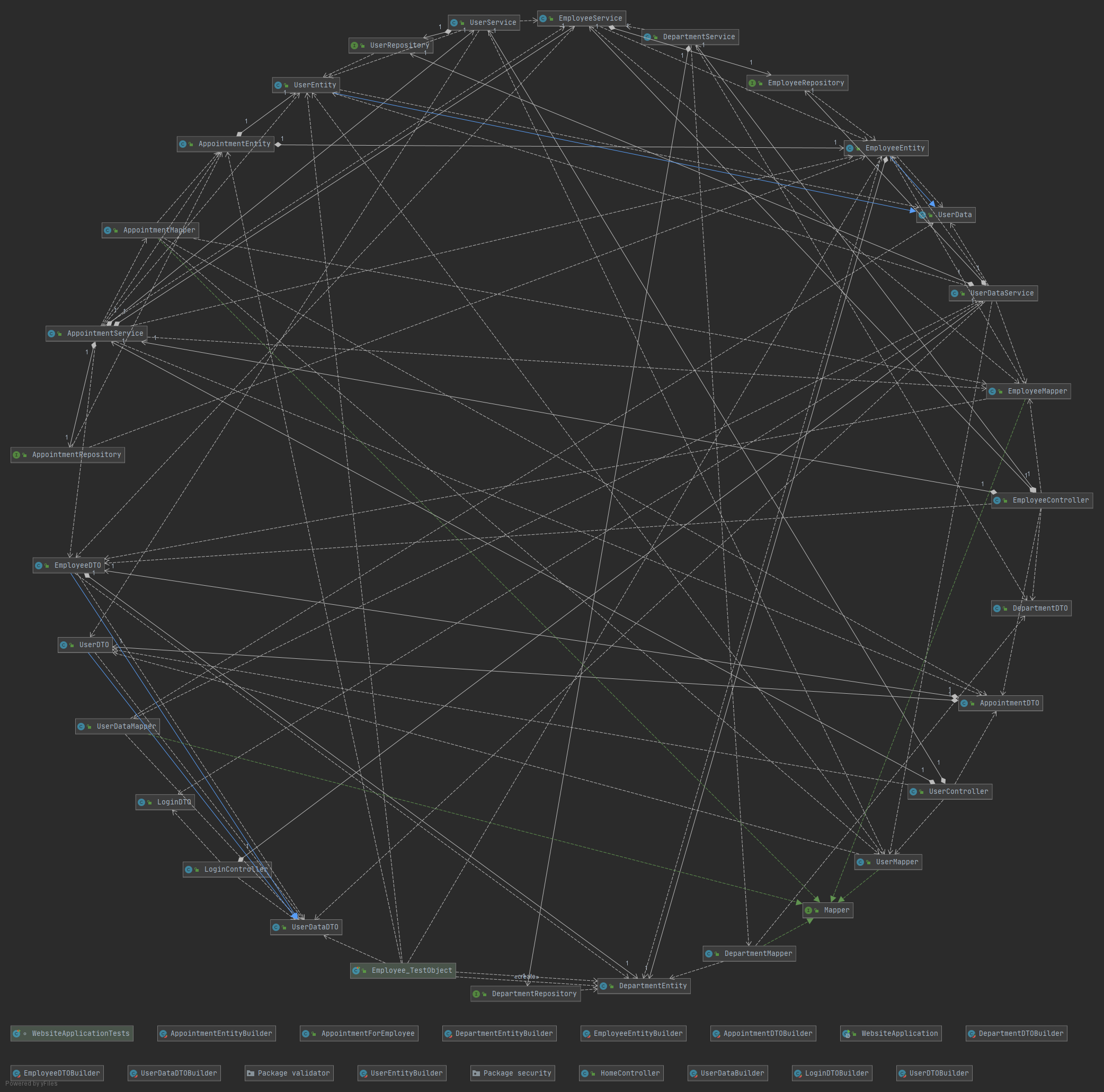
The database from the app will be a MySQL Database and I will use MySQL Workbench in order to manage it. The database will have 5 tables: one table for the user’s data, one table employee, one table for the user, one table for the appointments and one for the departments.

The MVC architecture fits my app because in the model package I will have all the logic, in the view I will have all the front-end files and all the classes related to the connection between the front-end and the actual back-end and in the controller package I will have all the database related logic.

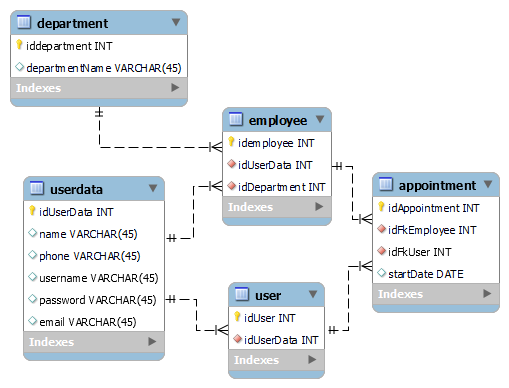
## 3.2 Package diagram



## 3.3 Class diagram



## 3.4 Database (E-R/Data model) diagram



# IV Supplementary specifications

## 4.1 Non-functional requirements

* Performance: My app would be as fast as possible. The waiting time would be maximum 2 seconds.
* Accessibility across devices: The app is running both on mobile and desktop platforms.
* Scalability: The app can support many users at any time, given the actions a user can make is not time consuming.
* Security: This requirement is implemented with data validation.
* Manageability: This is implemented with functions available to the admins, and they will be able to access and control everything.

## 4.2 Design constraints

The App is made using Java and Spring framework and the front-end part will be made using React.

For the database part and data manipulation, I will use MySQL.

For the security part, I will use data validation and maybe some hashing techniques for the passwords.

In order to make the app I will use IntelliJ and MySQL Workbench; for testing, I will use PostMan.

# V Future improvements

For further development I want to add more functionalities to the app such as creating employees in the app.

# VI Bibliography

* <https://stackoverflow.com/>
* <https://reactjs.org/>
* <https://react-bootstrap.github.io/>
* <https://getbootstrap.com/>
* <https://www.baeldung.com/>