**CA - 2 - JPA mapping and REST**

Group members

Nikolaj Desting

Boyko Surlev

Peter Tomascik

Index

1. Design description of the chosen design
2. Section explaining your test strategy, including test results
3. Section stating who did what
4. Description of what you have added, apart from step 1-7 in this document
5. Description of the strategy used to implement inheritance and why this strategy was chosen

**Design description of the chosen design**

We could divide our project into three parts. First would be database part, second façade and third server and html part.

Database:

Based on Business Domain which was given to us, we created entity classes which mimic real tables for our relational database. From the beginning we were using our local database. After some time we connect to oracle database and work with the *real* one. After all testing and making sure that database works perfectly we connect to other user database and put all valid data there. That database is going to be the one we just for deploying on Azur. Entity manager takes care for inserting data into the database. It is located in façade.

Façade

Façade holds entire logic of the program. It contains logic for running database and also all methods which are required for the server’s proper work.

Server and html:

Contains code for running our server and also has logic how to handle various situations which could occur on the web page. Html part contains actual JavaScript, css and html file which are used for both browser and server logic handlers.

**Section explaining your test strategy, including test results**

**tests**

**Section stating who did what**

It was very hard for us establish who was working on which section. We wrote a lot of comments into the code so evaluator can get better feeling of who did what. The process or work flow in general was following.

We created GIT repository and 3 branches for each of us. We all work separately on Business domain and we help each other a lot. It turns out as a great way of work because everybody could touch the code and also had had great understanding of it.

Working on façade was a bit harder. Boyko did most of the work in this part and by the end we realize that his code is most robust so we decided to use it. However we all had been working and helping each other during this process.

Rest of the work was done together although REST API was mostly done by Nick and Peter with Boyko’s help.

IMPORTANT

During the entire time we were using GIT-hub and we tried to use and implement it best we could. You can see commits of each member of our group in separated branches. By the end we built up one perfectly working project and push it to main. This may not seem as the best way of using the *main branch* but the way we decided to work on the project (close to pair-programming) did not let us use it in the proper way. We think that this is all a process of learning so we use this tool in the way that it serves us, not vice versa.

**Description of what you have added, apart from step 1-7 in this project**

We did all work which was required for steps 1-7. However we fix some of the bugs we found and based on teacher’s html file we changed it and make it more friendly. We customize JavaScript code so it works with our „logic“, create an input fields for adding role for the person but mostly we add error handling to the input fields so if input data are wrong or any other wrong or unexpected command is made we catch it and do not mess up with server or databse.

**Description of the strategy used to implement inheritance and why this strategy was chosen**

We start our project with creating DB on localhost. Following teachers advices we accomplished to have working DB using entity classes. In one part we had to make a decision which generation strategy we want to use. The options were “Single table strategy” and “Joined”. To make a decision was pretty easy. Single table strategy stores all data from all instances of inheritance hierarchy in one single table. That makes sometimes impossible to reach specific required data. On the other hand Joined strategy stores data in separated tables and keeps them in reachable way. It also follow normalization rules.