### Architecture

Due to the nature of our project and the client’s demands we have recognized our solution has to be accessible through a web browser on a laptop/desktop PC. We are also planning to implement a mobile version in further stages of development.

The application will use information stored in a relational database and allow the user to browse it in a form of a graph loosely resembling a tube map. Therefore we need to create a well-designed, scalable frontend layer. This will be achieved mainly with the use of JavaScript.

Our application will be running on a Spring Boot server. The Spring framework seems to be a good fit for our project for a number of reasons, one of them being its JDBC abstraction framework, which takes care of all the low-level details when working with a database.

Spring Boot allows us to use Java as our main language. We will be implementing a directed graph, with the courses as vertices. Java provides us with extensive, well documented libraries for implementing different data structures, including graphs. Our choice is the JGraphT library, largely due to its flexibility. Any object can be used for vertex and edge types, it supports directed graphs and live subgraph views on other graphs.

Our editor of choice is IntelliJ, as it provides support to create and run Spring Boot based applications. It also offers great support and integration for common developer tools such as Maven, which we happen to be using too. What is more, we find it incredibly convenient to use, thanks to features such as smart completion.

Our application will be connected to a MariaDB database. While we do acknowledge that MySQL would work well too in our case, we are familiar with MariaDB from last year’s Databases and Cloud Concepts module thus making it easier for us to use. Furthermore MariaDB has several optimizations that tend to improve the performance as compared to MySQL. An example of that would be the fact that when a view is queried only those tables are queried that are required by the query as opposed to all of the tables that are connected to the view like it is in MySQL. Even though the difference in performance is unlikely to be significant for our project, it is definitely not a detriment to its quality either.

Users

Website

Spring-Boot server

Database

Displaying data

Allowing the user to browse it and providing them with access to the courses

Application logic

Sending processed data to display

Organizing the data into a graph

Retrieving data from the database

Storing and managing the data

Java

JavaScript

MariaDB