Diagrams description

1. Conceptual model

According to the key concepts defined in the previously constructed SRS document we can obtain 10 structural components in the conceptual architecture design:

- GUI - homepage/user interface for all users
- Admin Panel - administrator-only view of the systems
- Admin Service - service specifically for system administrators
- Location Service - used for managing all the functionalities
- Location Finder - used for importing data from an external source
- Search - used for searching the stored data
- User Manager - used for authenticating and authorizing users
- Amenity Finder - managing additional information about the institutions
- Geoinformation Manager - used for managing the locations on the map
- Database - used for persistent storage

Data	**Function**	**Stakeholder**	**System**	**Abstract concept**
Institutions	Log in	User	GUI	Institution
Users	Log out	Administrator	Admin Panel	Map
Locations	Register		Admin Service	Comment
User Credentials	Add		Location Service	Rating
Working Hours	Edit		Location Finder	Nearest institutions
Address	Remove		User Manager	
Category	Store		Amenity Finder	
Title	Show		Geoinformation Manager	
Favorite places	Search		Database	

2. Execution model

- AppUI component - user-initiated component, performs action because of user input. Sends a HTTP request to the Web Server to get the requested data.
- Web Server component - service component. Waits for requests of other components (AppUI and Search) and generates responses for such requests.
- Database component - service component. Waits for requests of other components (Web Server) and generates responses for such requests.
- Searching infrastructure by name or category component - synchronous communication is used. The calling component (Searching infrastructure by name or category component) waits for a response from the called component (Web Server).
- Generating a map with locations from the database component - callback is used. The calling component (Web Server) receives a response asynchronously by setting a means by which the called component (generating map component) can send response later.
- Showing details about the infrastructure component - synchronous communication is used. The calling component (Database) waits for a response of the called component (showing details about the infrastructure component).

Binding execution and conceptual models

The main purpose of this diagram is to address the exact location of the Conceptual Components within the Execution Model. The best decision in our case was to choose a loosely-coupled object-oriented design of the architecture which will provide us with the benefits of scalability, configurability, maintainability and testability.

Execution behavior

The main purpose of this diagram is to verify that the execution architecture supports the desired behavior and to find errors in the architecture by using use-case maps to model the behavior.

We modeled the execution behavior for use case “Search amenity”.

3.Implementation model

Having already discussed the benefits of a loosely-coupled object-oriented architecture and the potential benefits we stand to gain by using it for our application design, we decided to use the Java programming language and its Spring Framework. The team decided it's most comfortable using Spring MVC because the framework is one of the most well-documented for this design pattern.

The implementation model is described by a 3-layer architecture consisting of the Client Entity, Web Server and Database Server. Additionally, the Web Server also communicates with an external entity, the OpenStreetMap API.

The Client Entity is dedicated to take care of the appearance of the application using HTML, CSS and JavaScript. Clients will be able to access the application with any web browser.

The Web Server communicates with both the client's Web Browser and the Database Server. It is consisted of 3 layers: Presentation Layer, Service Layer and Integration Layer:

- The Presentation Layer uses the Spring Framework to implement the application's interactive nature
- The Service Layer contains the Student App Service which implements our business logic and communicates with the OpenStreetMap API to obtain the functionalities of the map
- The Integration Layer is responsible for communicating with the database for persistent storage of data

The Database Server is an implementation of the PostgreSQL relational database.