Faculty of Mathematics and Information Science Warsaw University of Technology



**A System for Resources Management**

**in a Small Chemical Laboratory**

Technical Project

Choice of the development model

Aleksandra Bułka

Maciej Głowala

Klaudia Jarosz

Version no. 1.0

17.11.2017.

Table of Contents

Table of Contents 2

1 History of Document Changes 3

2 Technical Project 4

2.1 Executive summary 4

2.2 System architecture 4

2.3 Database Design 6

2.4 Class Diagrams 8

2.4.1 Software Architectural Pattern 8

2.4.2 Class Diagram 9

3 Choice of Development Model 10

3.1 Description of Chosen Development Model 10

3.2 Argumentation 10

4 Bibliography 11

# History of Document Changes

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Description** | **Version** |
| 15.11.2017 | Aleksandra Bułka | Initial version | 1.0.0 |
| 16.11.2017 | Aleksandra Bułka | Added Class Diagrams,  Database Design and  Choice of Development Model | 1.0.1 |
| 17.11.2017 | Klaudia Jarosz | System architecture | 1.0.2 |
| 20.11.2017. | Aleksandra Bułka | Some Corrections | 1.0.3 |

# Technical Project

## Executive summary

The students will design and implement a system supporting management of supplies in a small chemical laboratory (chemical reagents, instruments, etc.). The system keeps track of the state of resources in the laboratory, and stores the data in a database. The system has a graphical user-friendly interface which facilitates displaying and modifying the gathered data.

The technical project of the system is described in the following chapters.

## System architecture

InLIME application there will be used REST architecture style and MVC design pattern. At the very high level, the architecture looks as shown in Figure 1.

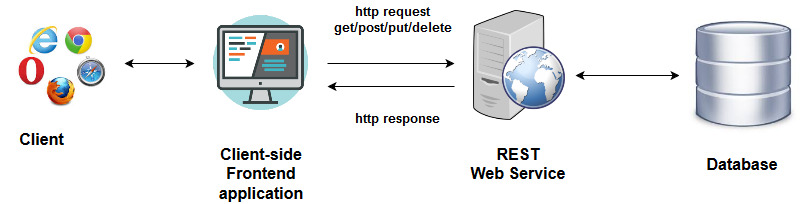


Figure 1. System architecture

It is constructed from the following three main components:

**Client-side application with user interface** - user access LIME application by using a Web browser. UI components collect data from the user and pass it to the Backend application It communicates only through HTTP requests and cannot read files off of a server directly.

**REST Web Service** – coordinates the application, manages business logic and data access.

**Database** – stores data about business objects. Information in form of entities can be inserted, updated and retrieved from the database and eventually passed back to the logic layer for processing.

The request flow in our application is the following:

1) The user hits the client-side frontend application

2) Client sends HTTP request to a specific URL to REST API

3) Dispatcher Servlet after receiving request, passes it to a specific controller method based on the relative part of the URL.

4) Rest API Controller performs business logic and calls Service method

5) Service communicates with DAO to get (insert/update/delete) data from database and then returns this data back to the Service

6) Received data will be processed according to business requirement and returned to the Controller

7) Controller returns response to Dispatcher Servlet. Servlet consults View Resolver to find the correct data representation format.

8) Servlet sends response back to the client

REST architecture was chosen from few reasons. Firstly, separation of an application into these components allows for any part of application to be modified without having to change the other parts. Moreover, due to this kind of separation, we can also use tools that specialize in the development of specific component, rather than making use of general purpose tools, which would be sufficient to build an entire application, but would be missing some powerful features.

## Targeted Technologies

The LIME application will be a Web Application for small laboratory business. It will consist of client – developed in AngularJS – and server – developed in SpringBoot java. Elements of application will exchange information by RESTful API. Client will be responsive and will be suited for phones, tablets and PCs.

As far as developer tools are concerned, we will be using IntelliJ IDEA (a Java integrated development environment (IDE)).

## Database Design

The database will be a very important part of the system, and the database objects will be represented by classes, therefore the design of the database is presented here.

The database consists of 6 tables:

1. **Resource** – Represents a basic laboratory resource, its properties, image in form of byte array (stored as string), category etc.
2. **Formula –** Deals with the many-to-many relation between resources and product, represents the proportions of resources used to make a product
3. **Product** – Stores products (complex structures of resources) with their properties
4. **Job** – Logs the jobs performed by the user (insertion/ deletion of a resource)
5. **User** – This table stores properties of every user of the system, role etc.
6. **Supplier** – Stores data about suppliers assigned to each resources

Relations between tables are presented on the Figure 2.

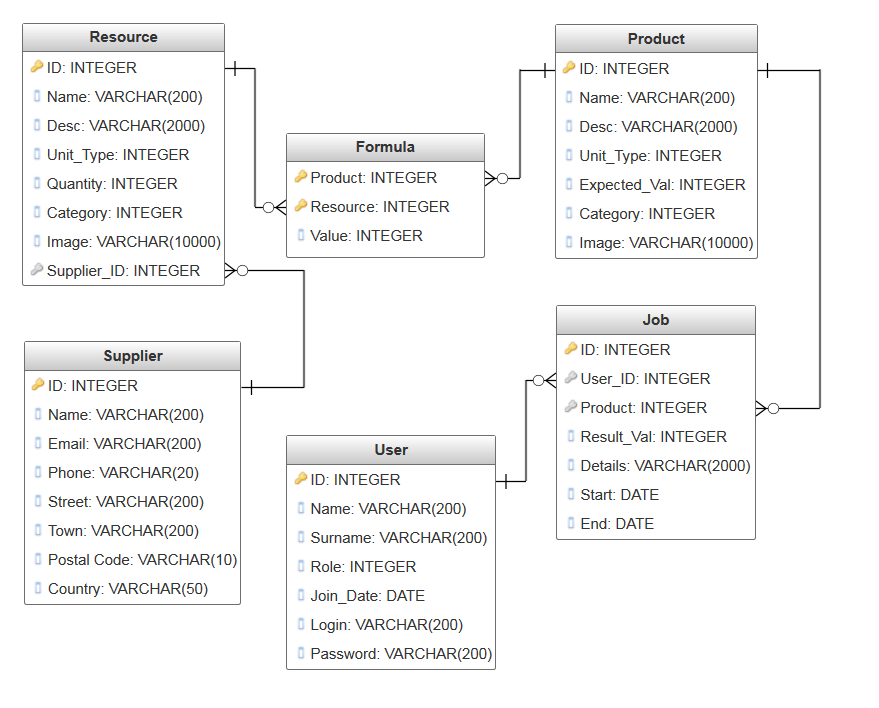


Figure 2. Database Design Diagram

## Class Diagrams

### Software Architectural Pattern

To facilitate the design of classes, we have decided to follow a software architectural pattern. A pattern of choice here was Model–view–controller (MVC) Pattern.

This pattern is used to separate a given 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.

* **Model** - Model represents an object carrying data. It can also have logic to update controller if its data changes.
* **View** - View represents the visualization of the data that model contains.
* **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

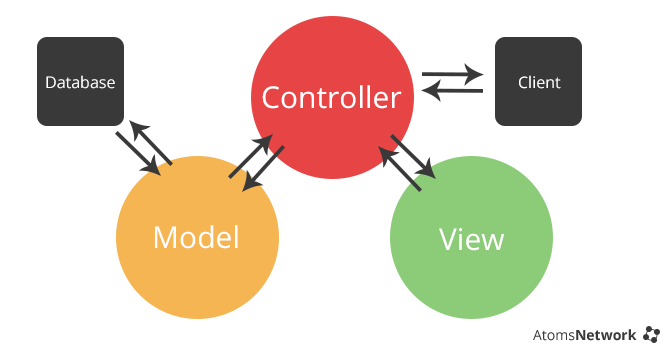


Figure 3. MVC Pattern

As mentioned before, it has helped us develop the division into classes, visible on the class diagram below.

**LIME** – is the program’s main class.

Then, the **Model**, **View** and **Controller** Classes were designed for each UI element.

The database objects are handled by **Servlet** (classes taking an article from http POST and passing it to JDBC), **DAO** (classes responsible for the communication with database) and **Query** (classes to parse SQL) **classes.**

### Class Diagram

Figure 4. illlustrates the class diagram for the application. An arrow pointing from class A to class B on the discussed diagram means that class A uses class B.

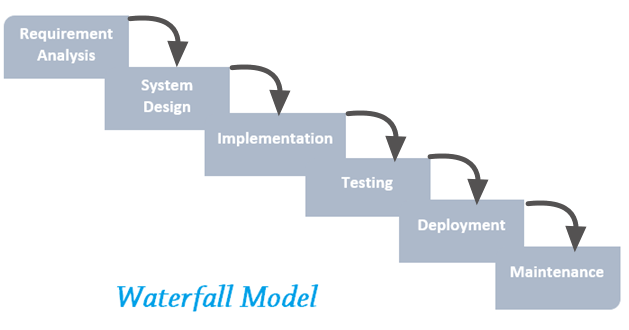


Figure 4. Class Diagram

# Choice of Development Model

## Description of Chosen Development Model

For our project, the development model of choice was the Waterfall Development Model



*Figure 5. Waterfall Model*

In Waterfall Model phases are executed sequentially, in linear way. We have a steady set of requirements, non changeable in time. The system is developed progressively and the user is involved only in the early phases

Advantages of this model include it being easy to manage. Disadvantages are that it contains a strict sequence of activities, it has high cost of errors in beginning stages and high importance and cost of documentation and also a contact with the customer is weak. Therefore it can be used when it is possible to precisely define the requirements

## Argumentation

The reasons we have chosen this particular development model are presented below:

* The sequential order of phases matched perfectly the organized schedule of the development of our Engineering Thesis
* There is no ‘user’ in this project as this is an engineering thesis, so weak contact with the user is not concern
* Easy management is facilitating our work as we work as a team and have no manager
* It was highly recommended by the coordinators of the Group Project

# Glossary

## LIME – stands for Laboratory Internal Management Entity. It is the name we have given to the application that is discussed in this document (application we are preparing for Engineering Thesis)

# Bibliography

1. tutorialspoint.com, (N/A). ASP.NET MVC Tutorial [online], Available at: http://www. tutorialspoint.com/design\_pattern/ [Accessed 15 Nov. 2017].
2. Wikipedia (N/A), *Model–view–controller* [online]. Available at <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller> [Accessed 15 Nov. 2017].
3. software-testing-tutorials-automation.com (2016), *Waterfall Model* [online]. Available at <http://www.software-testing-tutorials-automation.com/2016/06/waterfall-model.html> [Accessed 15 Nov. 2017].
4. Legierski J. (N/A). *Lecture Notes [*online] Available at <http://pages.mini.pw.edu.pl/~legierskij/eng/dydaktyka_eng.html> [Accessed 15 Nov. 2017].