Politechnika Warszawska

WYDZIAŁ MATEMATYKI
I NAUK INFORMACYJNYCH

Praca dyplomowa licencjacka

na kierunku Matematyka

NAJPIERW WYGENERUJ STRONĘ TYTUŁOWĄ

KTÓRA ZNAJDUJE SIĘ

Numer albumu 000000

promotor

W KATALOGU title_page

konsultacje

I OTRZYMANEGO PDF-A NAZWIJ titlepage.pdf I WSTAW DO KATALOGU Z SZABLONEM!

WARSZAWA 2018

....

supervisor's signature author's signature

Abstract

A System for Resources Management in a Small Chemical Laboratory

For the subject of our engeeniring thesis, we have resolved a problem of management of supplies in a small chemical laboratory. We have designed and implemented a system supporting management of resources (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 graphical user-friendly interface which facilitates displaying and modifying the gathered data. Multiple functionalities in the system allow the user to:

- Classify the resource into groups
- Assign descriptions with multimedia content to resources
- Define and generate reports and notifications. reports are displaying current state of resources as well as plot of activity in time, plot of demand for some resource in time, bar chart, pie chart, whereas notifications are alerts about low level of some chemical reagent, etc.
- For each resource, store suppliers' contact data and order new resources directly from the application
- Predict future demand for resources based on available historical data

The implemented system is a Web Application. It consists of a client application – developed in AngularJS – and server – developed in SpringBoot Java. Elements of application exchange information using RESTful API. As per development tools, we used IntelliJ IDEA (a Java integrated development environment (IDE)).

Keywords: Resource Management, Database, Time Series Forecast, Web Application, Java, AngularJS, SpringBoot, RESTful API, IntelliJ IDEA

Streszczenie

System zarządzania zasobami małego laboratorium chemicznego

Jako temat pracy inżynierskiej, rozwiązalimy problem zarządzania zasobami w małym laboratorium chemicznym. Zaprojektowaliśmy i zaiplementowaliśmy system wspomagający rozporządzanie zasobami (reagentami chemicznymi, itd.) System zbiera informacje o stanie zasobów i przechowuje je w bazie danych. Aplikacja posada przyjazny dla użytkownika interfejs graficzny, co ułatwia wyświetlanie i modyfikacje zebranych danych. Pozostałe funkcjonalności systemu pozwalają na:

- Klasyfikację zasobów w grupy
- Przypisanie zasobom opisów i multimediów (np. zdjęć)
- Definiowanie i generowanie raportów i notyfikacji. Raporty przedstawiają obecny stan zasobów, jak również wykresy aktywnoci (produkcji) w czasie, wykresy zapotrzebowania na zasoby w czasie, wykresy słupkowe, wykresy kołowe. Notyfikacje to alerty o niskim poziomie zasobów.
- Dla każdego zasobu, przechowaywane są informacje o danych kontaktowych do dostawcy oraz umożliwione jest zamówienie nowych bezporednio z aplikacji.
- Przewewidywanie przyszłego zapotrzebowania na zasoby na podstawie danych historycznych

Zaimplementowany system jest aplikacją internetową. Składa się z aplikacji klieanta - zaimplementowanej w AngularJS – i serwera – zaimplementowanego przy użyciu SpringBoot Java. Alementy aplikacji wymieniając się danymi za pomocą RESTful API. jako narzędzie deweloperskie, korzystaliśmy z IntelliJ IDEA.

Słowa kluczowe: Zarządzanie zasobami, Baza danych, Prognozowania na podstawie szeregu czasowego, Aplikacja internetowa, Java, AngularJS, SpringBoot, RESTful API, IntelliJ IDEA

Warsaw, ...

Declaration

I hereby declare that the thesis entitled „A System for Resources Management in a Small Chemical Laboratory", submitted for the Engineer degree, supervised by dr inż. Agnieszka Jastrzębska, is entirely my original work apart from the recognized reference.

.....

Contents

Introduction

The subject of the presented paper is the implementation of the system for management of supplies in a small chemical laboratory. The idea for such solution came from a real life situation, where an existing chemical laboratory needed an application to faciliate their work. Throught all phases of design and implementation, the real needs of these existing chemical laboratory were taken into consideration and many elements of the application originated from there. For instance, when thinking of a database, we matched the design to what we knew would be stored in this database. For example there are tables to store different products and resources, but the relationship between those two table reflects the laboratories' formulas, that is, which products are made of which resources and in which proportion. The following chapters will describe in depth:

- The set of initial requirements including business analysis and the background of the problem
- The detailed design of the application
- The technical information of how we realised this project, including development model and work division
- Some final conclusions along with post implementation valuation of our resulting application

1. Work Division Plan

The project was designed so as it could be completed by the group of three people. The tasks were divided so that each memeber of the group was assigned a part, and these parts were thought to be equally time consuming. That being said, all three participants of the project contributed towards the building of the system, its design and the frame application. What is more, great emphasis has been put on collaboration and team work, resulting in members of the group often performing tasks outside their divised part, which contributed toward the project's final success.

1.1. Work Division

Table 1.1: Work Division

Name	Responsibility
Klaudia Jarosz	Implementation of the frame application
	Creating a database containg data about users, resources, suppliers and user's activity
	Design and implementation of a user-friendly interface
Maciej Głowala	Implementation of the frame application
	Handling users with different roles
	Saving and restoring system state
	Virtual server setup
Aleksandra Bułka	Implementation of the frame application
	Implementation of reports and notifications
	Forecasting module
	Ordering module

2. Background of the Problem

The idea of the L.I.M.E. project comes from the well-known problems of managing the warehouses and laboratories. Small companies tackles with many perplexities connected with governance of production and the resources.

2.1. Other Known Solutions

The known solutions for the problem:

1. World-Class Warehousing and Material Handling by Edward Frazelle
2. Essentials of Inventory Management by Max Muller
3. Warehouse Management: A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse by Gwynne Richards
4. Inventory Accuracy: People, Processes, & Technology by David J. Piasecki
5. Introduction to Materials Management by Steve Chapman

Those problems may be hard to resolve. And there comes the L.I.M.E. which allows to register products and resources, managers and workers. Application delivers job records, predictions, and auto resource orders. It's ideal for small companies, who wants to improve control of staff, products and resources.

3. Requirement Specification

3.1. Functional Requirements

The functional requirements cases of the applications are different for different users of the application. The table contained in this chapter provide functional requirements for different groups of application users. The three groups of users of the application are:

1. Administrator (manager of the whole system and its users)
2. Manager (a person with rights for laboratory resources management)
3. Registered user

Table 3.1: Functional Requirements

Actor	Description
Administrator	Log in to the system, change and recover his password
	Create, modify and remove an account in the system, modify roles
	View, create, modify and remove resources and products and their groups
	Define, generate and send a report or prediction
	Define notifications, turn notifications on/off
	Order resources, turn on/off automatic ordering
	Save the current state of the system, schedule a system backup or restore it from backup
Manager	Log in to the system, change and recover his password
	View, create, modify and remove resources and products and their groups
	Define, generate and send a report or prediction
	Define notifications, turn notifications on/off
	Order resources, turn on/off automatic ordering
User	Log in to the system, change and recover his password
	View, create, modify and remove resources and products

	Define, generate and send a report or prediction
	Order resources

3.2. Non-functional Requirements

Table 3.2: Non-functional requirements

Area	Number	Details
`Usability`	1	Application must be responsive. It must be working on PC, tablets and phones with resolution at least 720p.
`Reliability`	2	Application must be of type High Availability. It should be available 24h/7d between 08:00 and 23:00. There could be service breaks during the week between 24:00 and 8:00.
	3	Application must have quick restart in case of app machine failures.
`Recovery`	4	Application must have daily database recovery performed between 24:00 and 08:00.
`Performance`	5	Application should respond no longer than 3 seconds while strain being on level 100 queries per minute.
`Supportability`	6	Documentation should contain instruction for recovery data from database backup.
	7	Application should keep backward compatibility between the released versions.
`Security`	8	Application must have user levels security. It shall not pass a user who has inappropriate privileges.

3.3. Use Cases

The uses cases of the applications, similarily to functional requirements are different for different users of the application. The tables contained in the following chapters provide descriptions of use cases for different groups of application users.

3.3.1. Administrator

Table 3.3: Uses Cases for Administrator

Actor	Name	Description
Administrator	Login	Log in to the system
	Password Management	Recover his password
		Change his password
	User Account Management	Create an account in the system, assign the account to a role (user, manager)
		Modify an account in the system – change either personal data or assignment to a role (user, manager)
		Remove an account from the system
	Resource View	Display nicely current availability of resources and their categorization - multiple viewing perspectives, sorting and filtering are available
	Resource Management	Define a new type of resource, describe it with description card, add multimedia content to this resource (for example a photograph) and assign the resource with a supplier
		Modify a resource, change description card, multimedia content associated with this resource (for example a photograph) and its assignment to a supplier
		Delete a resource from database
	Resource Group Management	Create a group of laboratory resources, define which resources will belong to this group
		Modify a group of laboratory resources, redefine which resources will belong to this group
		Delete a group of laboratory resources
	Product View	Display nicely products produced by laboratory and their categorization
	Product Management	Define a new type of product, describe it with description card, add multimedia content to this product (for example a photograph)
		Modify a product, change description card, multimedia content associated with this product (for example a photograph)

		Delete a product from database
	Product Group Management	Create a group of laboratory product, define which products will belong to this group
		Modify a group of laboratory products, redefine which products will belong to this group
		Delete a group of laboratory products
	Report Definition	Define what the report will contain, for example a plot of production in time, plot of demand for some resource in time
		Define how the data will be presented, for example bar chart, pie chart, table with adjustable columns/rows
		Generate the desired report
		Define the recipients and send the report
	Prediction Report Definition	Define what the report will contain, this can be either for example production in time or demand for some resource in time
		Define for which resources, products or groups of resources or products the prediction should be made
		Generate the desired prediction report
		Define the recipients and send the prediction report
	Notification Definiton	Define whether notifications (an alert about low level of some chemical reagent) will it be sent
		Define when the notifications will it be sent (set the value which is critical for each reasource)
	Order Management	Define how many and which resources are to be ordered
		Send an order
		Turn on automatic ordering of resources or turn it off
	System State Management	Save the current state of the system
		Shedule an automatic back-up of a system state
		Restore system state based on an archived backup

3.3.2. Manager

Table 3.4: Uses Cases for Manager

Actor	Name	Description
Manager	Login	Log in to the system
	Password Management	Recover his password
		Change his password
	Resource View	Display nicely current availability of resources and their categorization - multiple viewing perspectives, sorting and filtering are available
	Resource Management	Define a new type of resource, describe it with description card, add multimedia content to this resource (for example a photograph) and assign the resource with a supplier
		Modify a resource, change description card, multimedia content associated with this resource (for example a photograph) and its assignment to a supplier
		Delete a resource from database
	Resource Group Management	Create a group of laboratory resources, define which resources will belong to this group
		Modify a group of laboratory resources, redefine which resources will belong to this group
		Delete a group of laboratory resources
	Product View	Display nicely products produced by laboratory and their categorization
	Product Management	Define a new type of product, describe it with description card, add multimedia content to this product (for example a photograph)
		Modify a product, change description card, multimedia content associated with this product (for example a photograph)
		Delete a product from database
	Product Group Management	Create a group of laboratory product, define which products will belong to this group
		Modify a group of laboratory products, redefine which products will belong to this group

		Delete a group of laboratory products
	Report Definition	Define what the report will contain, for example a plot of production in time, plot of demand for some resource in time
		Define how the data will be presented, for example bar chart, pie chart, table with adjustable columns/rows
		Generate the desired report
		Define the recipients and send the report
	Prediction Report Definition	Define what the report will contain, this can be either for example production in time or demand for some resource in time
		Define for which resources, products or groups of resources or products the prediction should be made
		Generate the desired prediction report
		Define the recipients and send the prediction report
	Notification Definiton	Define whether notifications (an alert about low level of some chemical reagent) will it be sent
		Define when the notifications will it be sent (set the value which is critical for each reasource)
	Order Management	Define how many and which resources are to be ordered
		Send an order
		Turn on automatic ordering of resources or turn it off

3.3.3. User

Table 3.5: Uses Cases for User

Actor	Name	Description
User	Login	Log in to the system
	Password Management	Recover his password
		Change his password
	Resource View	Display nicely current availability of resources and their categorization - multiple viewing perspectives, sorting and filtering are available

	Resource Management	Define a new type of resource, describe it with description card, add multimedia content to this resource (for example a photograph) and assign the resource with a supplier
		Modify a resource, change description card, multimedia content associated with this resource (for example a photograph) and its assignment to a supplier
		Delete a resource from database
	Product View	Display nicely products produced by laboratory and their categorization
	Product Management	Define a new type of product, describe it with description card, add multimedia content to this product (for example a photograph)
		Modify a product, change description card, multimedia content associated with this product (for example a photograph)
		Delete a product from database
	Report Definition	Define what the report will contain, for example a plot of production in time, plot of demand for some resource in time
		Define how the data will be presented, for example bar chart, pie chart, table with adjustable columns/rows
		Generate the desired report
		Define the recipients and send the report
	Prediction Report Definition	Define what the report will contain, this can be either for example production in time or demand for some resource in time
		Define for which resources, products or groups of resources or products the prediction should be made
		Generate the desired prediction report
		Define the recipients and send the prediction report
	Order Management	Define how many and which resources are to be ordered
		Send an order

4. Development Methodology

4.1. Methodology

For our project, the development methodology of choice was the Waterfall Development Model. This methodology was highly recommended to us.

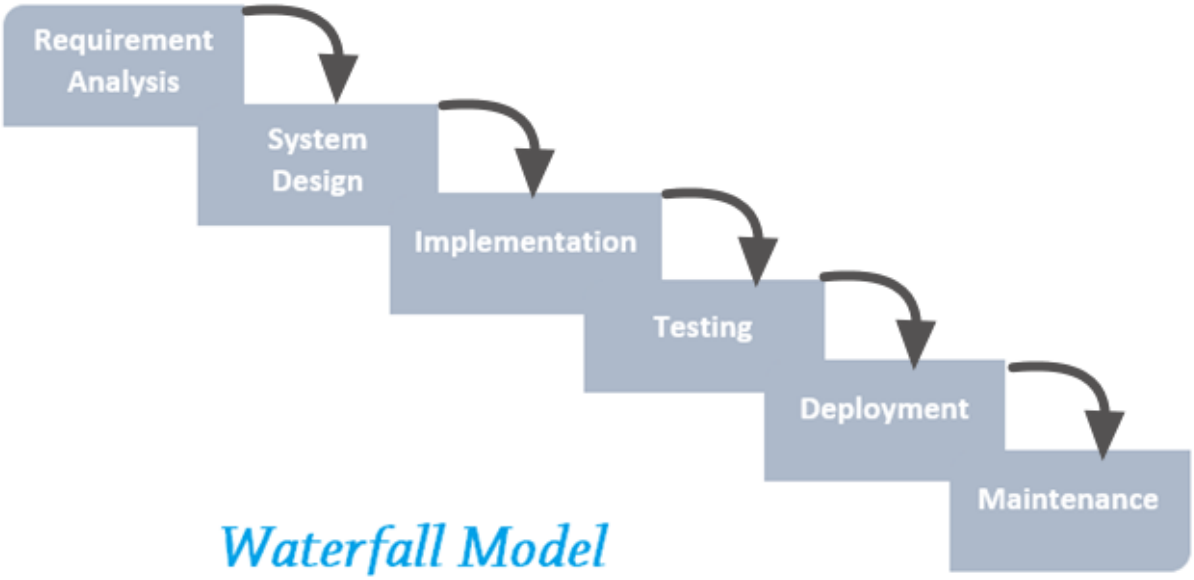

Figure 4.1: Waterfall Model

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

4.1.1. 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
- Fixed set of requirements, as the initial requirements waere submitted by team to the Faculty and could not be changed later
- 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

5. System Architecture

5.1. 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.

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.

5.2. Backend Architectural Pattern

L.I.M.E. project base on many architectural patterns. Precisely chosen patterns make overall architecture strong and well built.

5.2.1. Data Access Object Pattern

Data access object (DAO) is an object that provides an abstract interface to some type of database or other persistence mechanism. By mapping application calls to the persistence

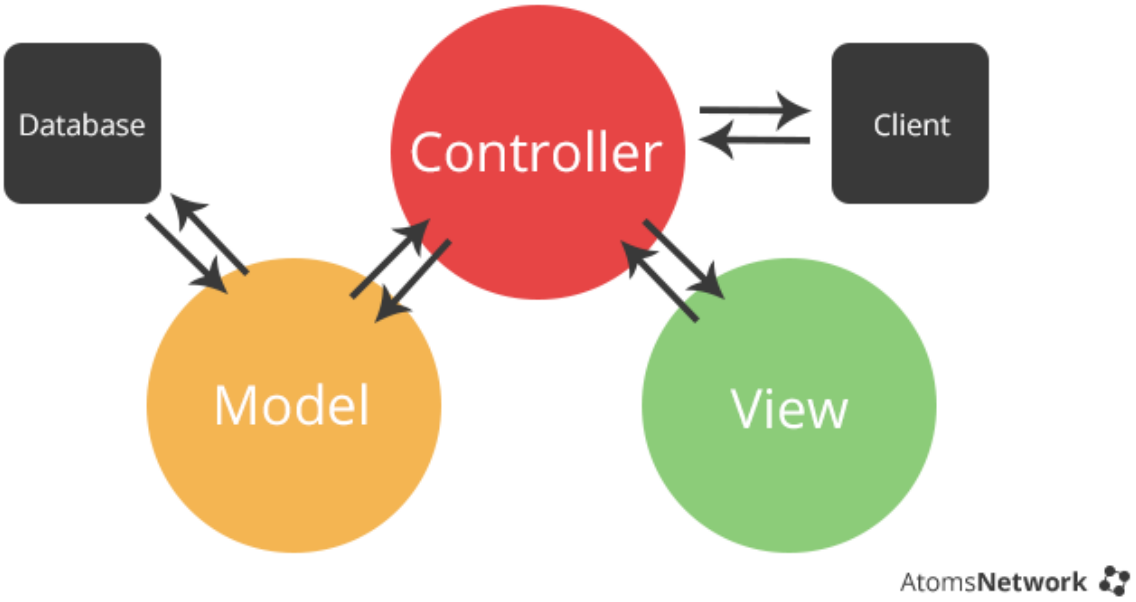

Figure 5.1: MVC Pattern

layer, the DAO provides some specific data operations without exposing details of the database. This isolation supports the Single responsibility principle. It separates what data access the application needs, in terms of domain-specific objects and data types (the public interface of the DAO), from how these needs can be satisfied with a specific DBMS, database schema, etc. (the implementation of the DAO). The most important fact and advantage is the relatively simple and rigorous separation between two important parts of an application that can but should not know anything of each other, and which can be expected to evolve frequently and independently. Changing business logic can rely on the same DAO interface, while changes to persistence logic do not affect DAO clients as long as the interface remains correctly implemented. All details of storage are hidden from the rest of the application (see information hiding). Thus, possible changes to the persistence mechanism can be implemented by just modifying one DAO implementation while the rest of the application isn't affected. L.I.M.E. implements one template parent interface with methods which are used by all DAO classes. IBasicCrudRepository extends hibernate CrudRepository and inherits from it all CRUD database operations. Singular DAO classes implement service-specific methods which are defined in DAO interfaces.

5.2.2. Plain Old Java Object

Plain old Java object (POJO) is an ordinary Java object, not bound by any special restriction and not requiring any class path. The main aim of the POJO classes is to differentiate business logic from database entity. It makes code cleaner and easier to read and understand.

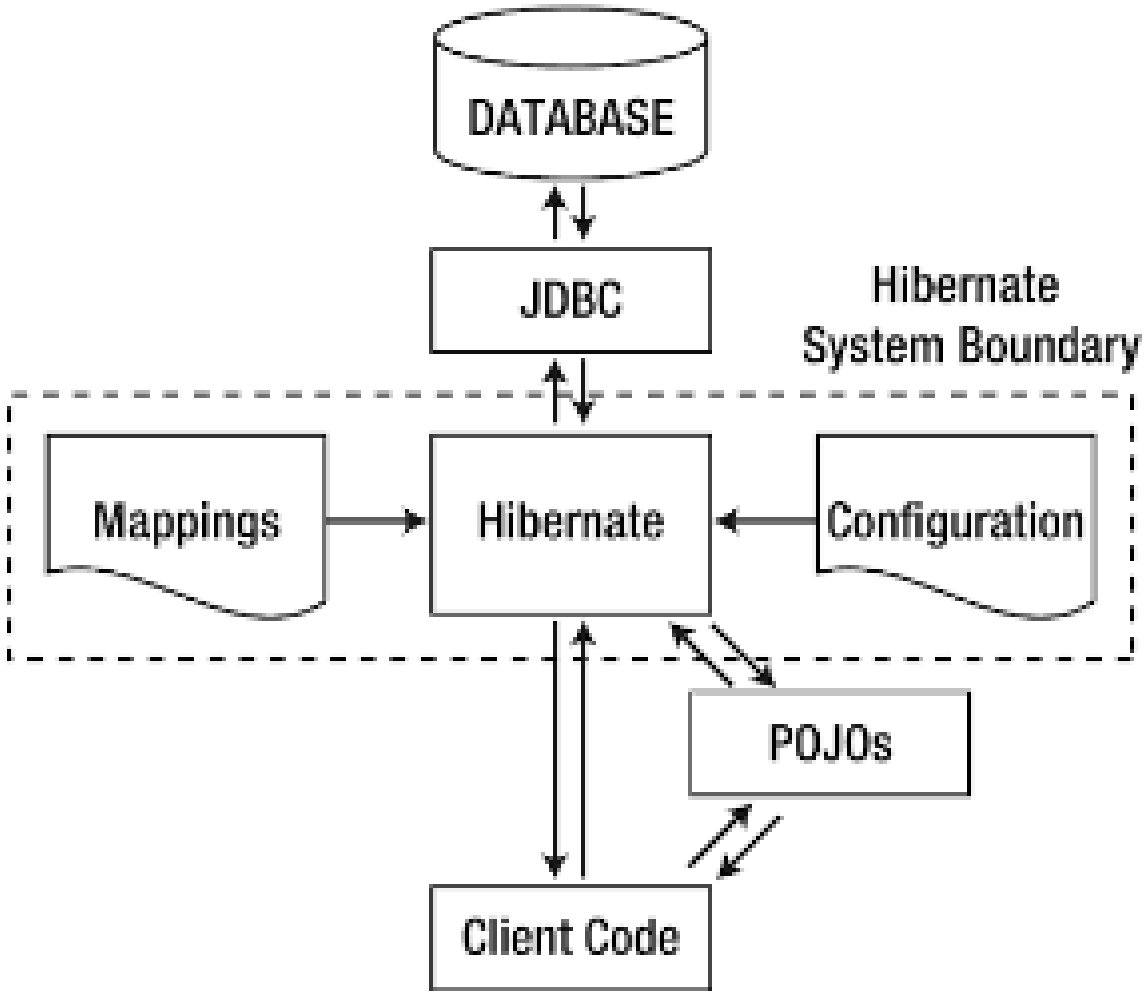

Figure 5.2: Plain Old Java Object in Backend Architecture

The POJO phenomenon has most likely gained widespread acceptance because of the need for a common and easily understood term that contrasts with complicated object frameworks.

L.I.M.E implemens POJO classes for all existing entity class. Those classes are free from frameworks and complicated annotations. There is object mapper implemented, which maps directly Entity to POJO.

5.2.3. Dependency Injection

Dependency Injection is a kind of broader version of "inversion of control" (IoC) principle. It relates to the way in which an object obtains references to its dependencies - the object is passed its dependencies through constructor arguments or after construction through setter methods or interface methods. It is called dependency injection since the dependencies of an

object are 'injected' into it, the term dependency is a little misleading here, since it is not a new 'dependency' which is injected but rather a 'provider' of that particular capability. For example, passing a database connection as an argument to a constructor instead of creating one internal would be categorized as dependency injection. The pattern seeks to establish a level of abstraction via a public interface and to remove dependencies on components by supplying a 'plugin' architecture. This means that the individual components are tied together by the architecture rather than being linked together themselves. The responsibility for object creation and linking is removed from the objects themselves and moved to a factory.

The main advantage of Dependency Injection is fact that there is only one instance of each object shared through multiple controllers. What is more, it allows a client the flexibility of being configurable. Only the client's behavior is fixed. The client may act on anything that supports the intrinsic interface the client expects. Dependency injection can be used to externalize a system's configuration details into configuration files, allowing the system to be reconfigured without recompilation. Separate configurations can be written for different situations that require different implementations of components. This includes, but is not limited to, testing. L.I.M.E. uses spring dependency injection. There are bean services, which implements the domain logic. Each service implements interface defining the methods. Interfaces and services corresponds to hibernate entities and fulfill the needs of application logic. Interface beans are autowired into the REST controllers.

5.2.4. RESTful API

REpresentational State Transfer (REST) is an architectural style that defines a set of constraints and properties based on HTTP. REST-compliant web services allow the requesting systems to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations. Other kinds of web services, such as WSDL and SOAP, expose their own arbitrary sets of operations. In a RESTful web service, requests made to a resource's URI will elicit a response that may be in XML, HTML, JSON, or some other format. The response may confirm that some alteration has been made to the stored resource, and the response may provide hypertext links to other related resources or collections of resources. When HTTP is used, as is most common, the operations available are GET, POST, PUT, DELETE, and other predefined CRUD HTTP methods.

L.I.M.E. implements RESTful controllers which are used to communicate with a client part of application. It communicates with JSON body type. There are CRUD used methods like PUT, DELETE, POST, GET to make action more readable and indicate what exactly it does. There

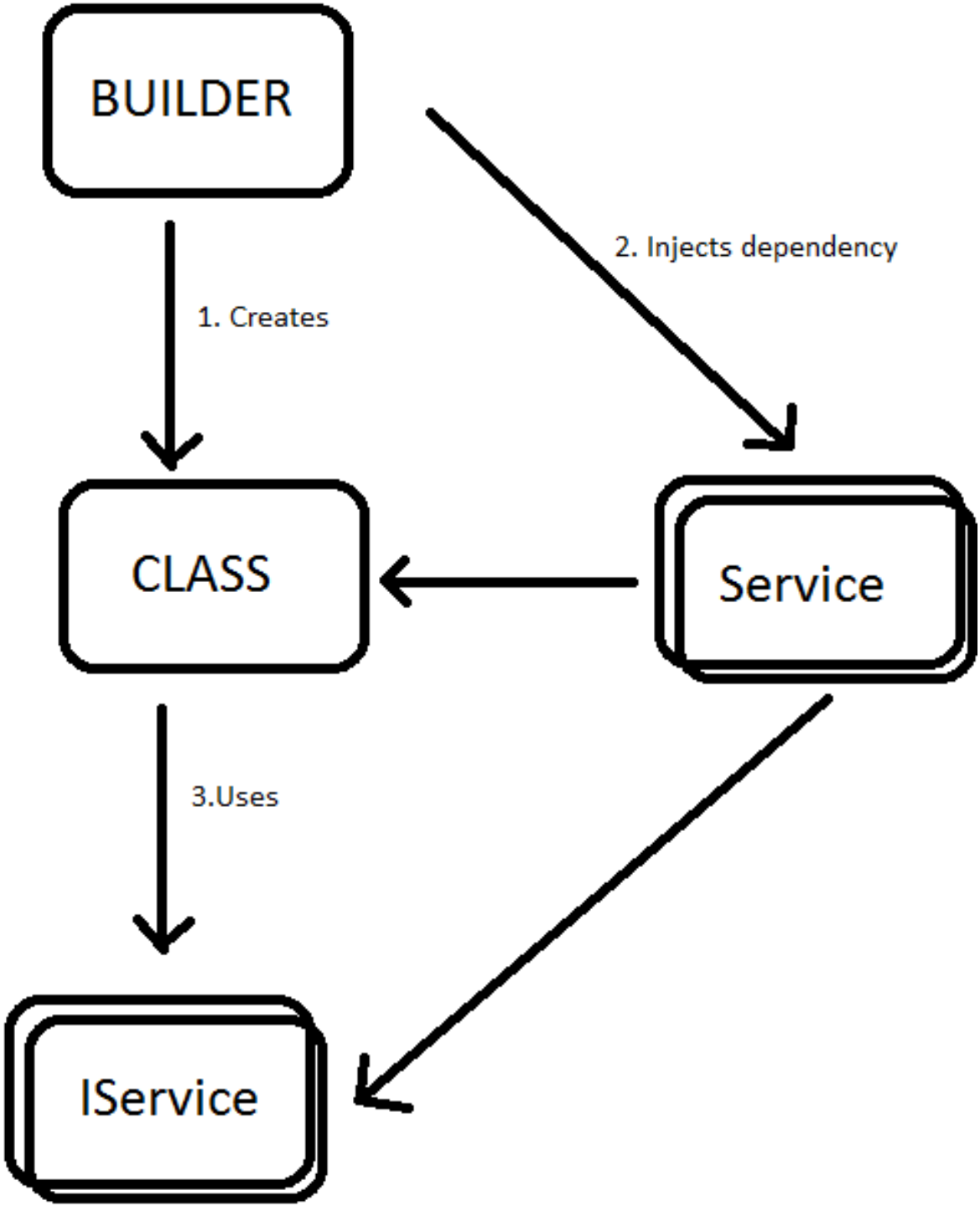

Figure 5.3: Dependency Injection

are Controllers corresponding to view in a client part. .

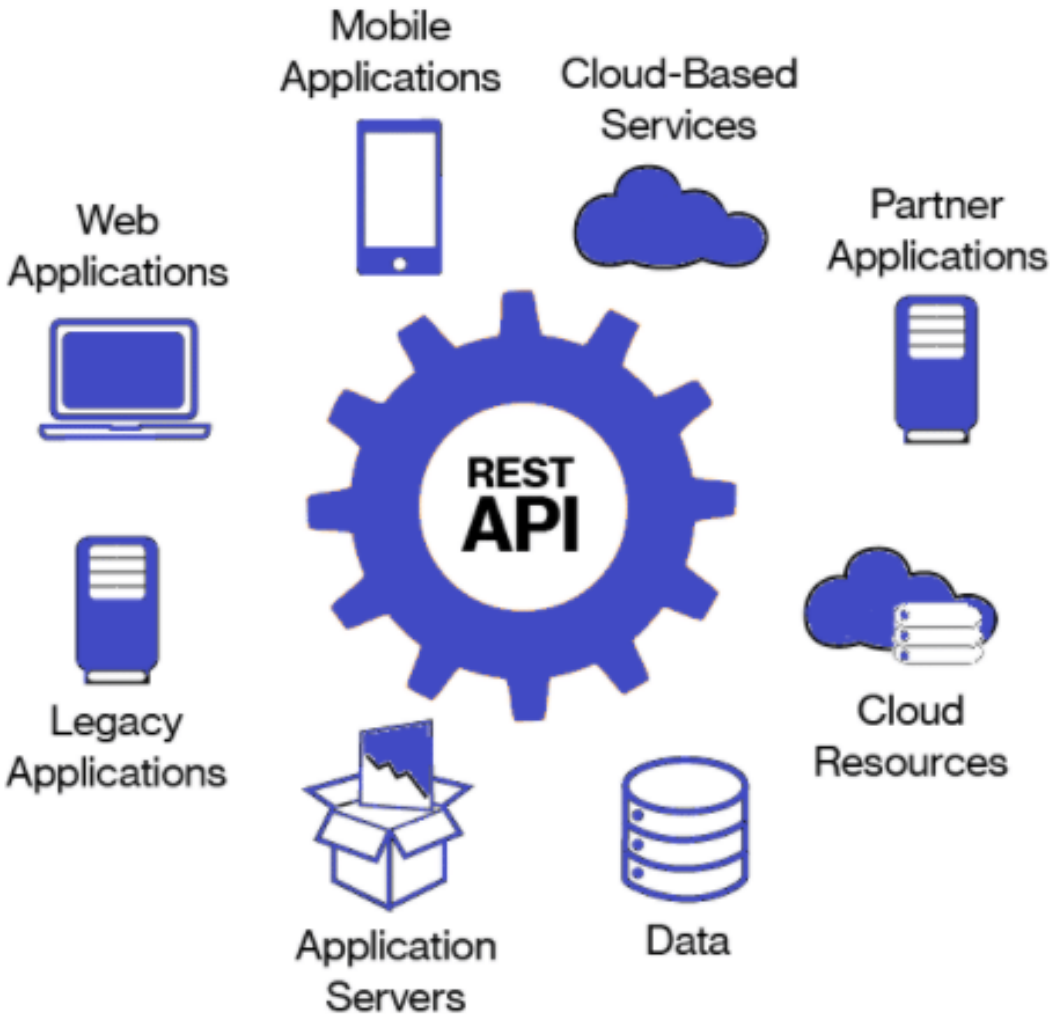

Figure 5.4: RESTful API

6. Technical Analysis

6.1. Client-Server Architecture

Client/server architecture is a producer/consumer computing architecture where the server acts as the producer and the client as a consumer. The server houses and provides high-end, computing-intensive services to the client on demand. L.I.M.E. implements client-service architecture. There are 2 applications in the project:

1. Server implemented in JAVA,

2. GUI implemented with AngularJS

Communication between client and server is done with RESTful API. Server accepts request from authenticated users sent from GUI. Then process it, saves data in database and sends answer.

6.2. Login

L.I.M.E. application have got system to recognize users and its roles. It is implemented using Spring Security. When the user logs in GUI application sends login request to server and the server application checks if the user exist and it's roles. In the response, it passes the authorization data which is valid for 12 hours (session timeout). Each request from GUI to Server contains a header with the authorization data, so server can recognize which user sends request and if has appropriate roles.

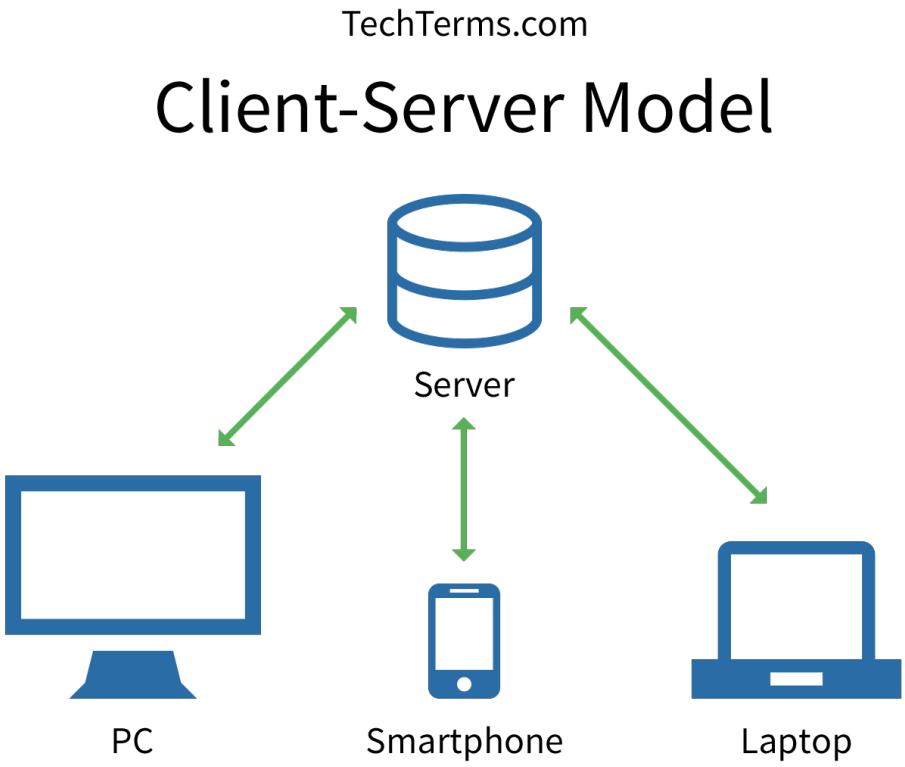

Figure 6.1: Client-Server Architecture

6.3. Program Interface

6.4. System Classes

6.5. Database Design

The database consists of the following tables:

1. **Product** Stores information about products (complex structures of resources) with their properties:
   (a) **Product ID** ID of a product
   (b) **Added At** Date of adding a product to a database
   (c) **Describtion** Describtion of a product
   (d) **Expected Value** Expected value - how much of a product shall be obtained after

production

(e) **Image** Image for this product

(f) **Name** Name of a product

(g) **Unit** Unit of a measurement

(h) **Category ID** Assigns a product to a category

2. **Product Category** Stores information about product categories: its name and ID

3. **Job** Logs the jobs performed by the user and the following details:

(a) **Job ID** ID of a job

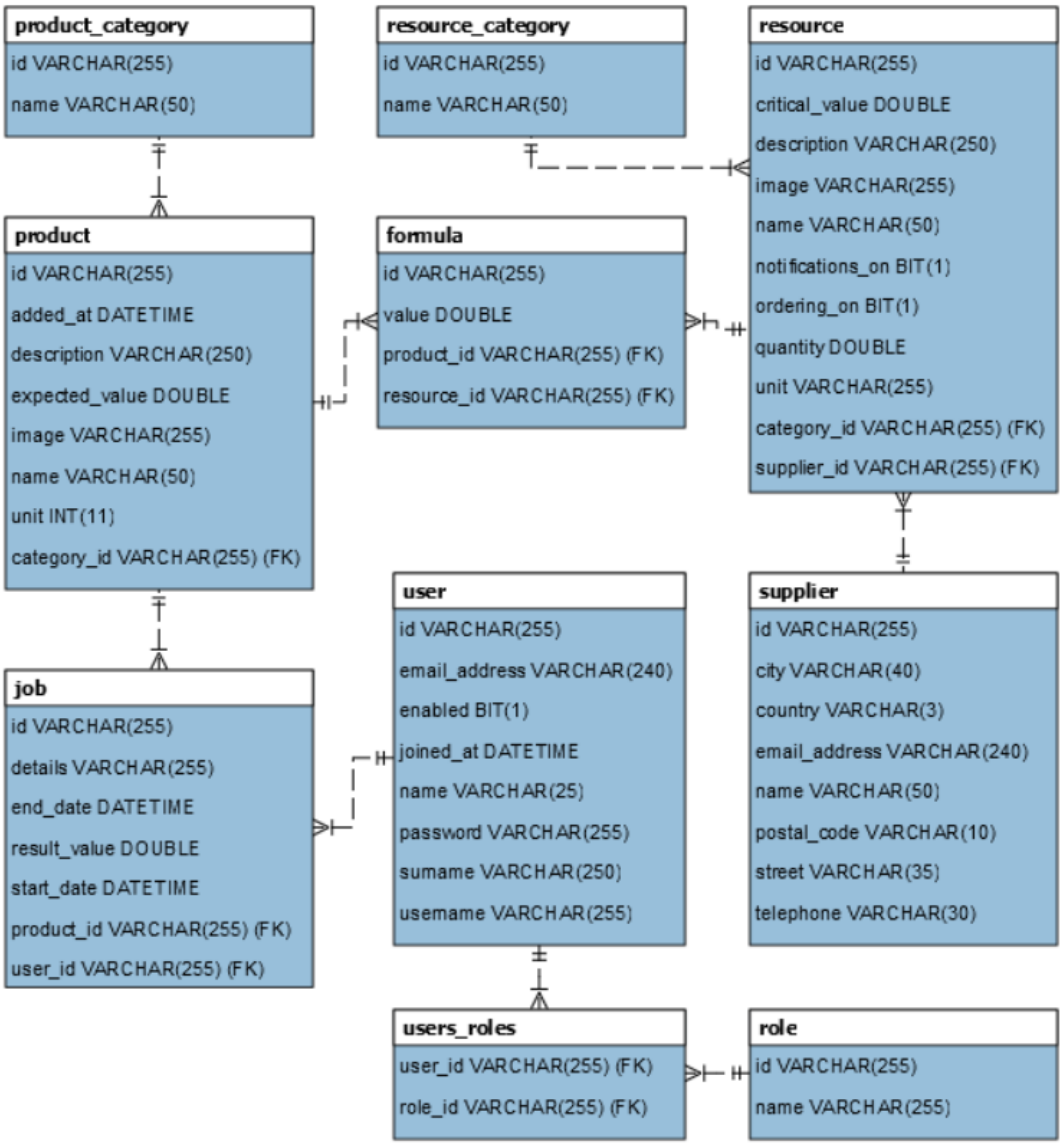

Figure 6.2: Database Design

(b) **Details**More details about the job

(c) **Result Value** Result value - how much of a product was actually obtained after production

(d) **Start Date** Start date of a job

(e) **End Date** End date of a job

(f) **Product ID** ID of a product that resulted from this job

(g) **User ID** Assigns a job to a user which have performed it

4. **User** This table stores properties of every user of the system, such as:

(a) **User ID** ID of a user

(b) **Email address** Email address of this user

(c) **Enabled** Is this an active user of the system

(d) **Joined At** Date of joining the system

(e) **Name** Given Name

(f) **Surname** Surname

(g) **Username** Username chosen by the user

(h) **Password** User's password in encrypted form

5. **User Roles** Assigns users to their roles (resolves the many-to-many relationship)

6. **Role** Stores each role with its name and ID

7. **Formula** Resolves the many-to-many relation between resources and product, represents the quantity of resources used to make a product

(a) **Formula ID** ID of a formula

(b) **Product ID** ID of a product

(c) **Resource ID** ID of a resource

(d) **Value** How much of a given resource is neede to make the given product

8. **Resource** Stores information about a basic laboratory resource and its properties, such as:

(a) **Product ID** ID of a resource

(b) **Describtion** Describtion of a resource

(c) **Image** Image for the resource

(d) **Name** Name of a resource

(e) **Critical Value** critical value - how much of a resource shall be left for the notifications of the low level of resource to be triggered

(f) **Notifiactions On** Are the notifications about the low level of the resource turned on

(g) **Ordering On** Is automatic ordering of the resource turned on

(h) **Quantity** How much of a resource at the storage at the moment

(i) **Unit** Unit of a measurement

(j) **Category ID** Assigns a resource to a category

(k) **Supplier ID** Assigns a resource to a supplier

9. **Resource Category** Stores information about resource categories: its name and ID

10. **Supplier** Stores data about suppliers assigned to each resources

(a) **Supplier ID** ID of a supplier

(b) **Email address** Email address of this supplier

(c) **Name** Name of the supplier

(d) **Street** Address of the supplier: street name and number

(e) **City** Address of the supplier: City

(f) **Postal Code** Address of the supplier: Postal Code

(g) **Country** Address of the supplier: Country

(h) **Telephone** Telephone number of this supplier

6.6. Prediction Model

7. Post Execution Documentation

In this section the final application was tested against the set of initial requirements. for every requirement given in Chapter 3, discussed is how the particular requirement is implemented in the application.

7.1. Evaluating Functional Requirements

The following sections will discuss how the functional requirements were implemented, separately for every group of users (Administrator, Mangaer and User)

7.1.1. Administrator

Table 7.1: Evaluating Functional Requirements for Administrator

Functional Requirement	Implementation
Log in to the system and change his	L.I.M.E. application recognize users and its roles. It is implemented using spring security. Password can be changed by administrator of the system.
Create, modify and remove an account in the system, modify its roles	Administrator can create account via manage users view. It is possible to chose within 3 roles: administrator, manager, staff. It is also possible to edit or remove an account in the same subpage
View, create, modify and remove resources and products	Administrator can create products and recources in products view and resources view respectively. It is also possible to modify and remove objects in the same subpage.
View, create, modify and remove groups of resources and products	Administrator can create groups of products and recources choosing products or resources in manage groups respectively. It is also possible to modify and remove objects in the same subpage.

View, create, modify formula of the product	Administrator can create formula for the given product choosing resources and putting the needed value. It is possible creade and modify formula in products view.
Denfine, generate and send a report	Administrator can define and generate raport depending on date range and resources or products. It can be done in reports view, product and resource subpage respectively.
Denfine, generate and send a prediction	Administrator can define and generate prediction depending on date range and respoures or products. It can be done in prediction view, product and resource subpage respectively.
Define notifications, turn notifications on	Administrator can define critical value of the resource in notifications view. It will notify with email that resource exceed critical value. It can be set on/off separately for each object or for all with one button.
Create job, declare time range of the job	Administrator can create job of an product and declare time range within the job was done. It can be done in job view
Order resources, turn on/off automatic ordering	Administrator can order the resources - supplier will be notified with a mail. It is possible to group resources and send one order. It is possible to turn on/off automatic orders - order will perform when the resource exceed critical value. It can be done in manage orders view.

7.1.2. Manager

Table 7.2: Evaluating Functional Requirements for Manager

`Functional Requirement`	`Implementation`
Log in to the system	L.I.M.E. application recognize users and its roles. It is implemented using spring security. Password can be changed by administrator of the system.
View, create, modify and remove resources and products	Manager can create products and recources in products view and resources view respectively. It is also possible to modify and remove objects in the same subpage.
Define, generate and send a report	Manager can define and generate raport depending on date range and resoures or products. It can be done in reports view, product and resource subpage respectively.

Define, generate and send a prediction	Manager can define and generate prediction depending on date range and resoures or products. It can be done in prediction view, product and resource subpage respectively.
Define notifications, turn notifications on	Manager can define critical value of the resource in notifications view. It will notify with email that resource exceed critical value. It can be set on/off separately for each object or for all with one button.
Order resources, turn on/off automatic ordering	Manager can order the resources - supplier will be notified with a mail. It is possible to group resources and send one order. It is possible to turn on/off automatic orders - order will perform when the resource exceed critical value. It can be done in manage orders view.
Create job, declare time range of the job	Manager can create job of an product and declare time range within the job was done. It can be done in job view.

7.1.3. User

Table 7.3: Evaluating Functional Requirements for User

`Functional Requirement`	`Implementation`
Log in to the system	L.I.M.E. application recognize users and its roles. It is implemented using spring security. Password can be changed by administrator of the system.
Create job, declare time range of the job	User can create job of an product and declare time range within the job was done. It can be done in job view.

7.2. Evaluating Non-Functional Requirements

Similarly, this section will discuss how the non-functional requirements were implemented

Table 7.4: Evaluating Non-Functional Requirements

`Area`	`Non-Functional Requirement`	`Implementation`
Usability	Application must be responsive. It must be working on PC, tablets and phones with resolution at least 720p.	GUI is implemented with responsive frameworks: Angular and bootstrap

Reliability	Application must be of type High Availability. It should be available 24h/7d between 08:00 and 23:00. There could be service breaks during the week between 24:00 and 8:00.	Application is deployed on heroku. It uses database mysql addon. It runs 24/7
	Application must have quick restart in case of app machine failures.	It is possible to restart app with command: heroku restart
Recovery	Application must have daily database backup performed 24:00 and 08:00.	Database has default scheduled database dumps. It can be resheduled in heroku db configuration
Performance	Application should respond no longer than 3 seconds while strain being on level 100 queries per minute.	Application uses client server architecture. RESTful api deliver much higher capacity.
Supportability	Documentation should contain instruction for recovery data from database backup.	There is delivered step by step manual with database backup manual and sheduled and recovery.
	Application should keep backward compatibility between the released versions.	With the future releases version, there will be database script provided, which will fill previous entries.
Security	Application must have user levels security. It shall not pass a user who has inappropriate privileges.	Application has implemented user service implemented with spring security, which validates if the user exists and check for each REST endpoint if the user has appropriate privileges.

Conclusions

One of the most important goal of our thesis was to deliver a software that will be used by the consumer of the market. Work began with market research. We received several proposals and chose the one that we considered the most appropriate. We decided to develop the ultimate system to manage the warehouse and human resources of the laboratory. Our project – L.I.M.E (Laboratory Internal Management Entity) was developed in line with market needs, consulted with major consumers.

At first, we had to decide how the database will look like - what entities, relations, type of the database. That is the fundamental part of the application. We can say with pride, that we achieved excellent result. There were only few minor changes, most of the primal database project fulfilled needs of the application.

Secondly, we had to develop basic back-end parts of the application. Connection with database, user authentication and roles. Many helpers like database populator for test purposes.

Subsequently we had to start develop the client application. This was the hardest part of the work. It was hard to write simultaneously front-end and back-end. There was many disagreements, and we had to discuss for a long time often to come to the best solutions.

Afterwards, when we had already most of the functionalities, we started to develop features that uses most of the functionalities – reports, predictions, notifications. It required knowledge from the area of algorithms and mathematics.

To develop our application, we used the top software technologies. Server side is built with Java, Spring, SpringBoot, SpringSecurity, MySQL and the client with AngularJS, Bootstrap. The communication between units is made with RESTful API. The we well-known and wide used solutions helped us to achieve success.

During all the time we were documenting each progress we made. The documentation is essential for our project. We need to deliver all manuals and support for the consumers, to help them understand how to properly use and take full advantage of L.I.M.E.

Despite the many difficulties we encountered and the fact that it took about 40% more time than we expected, we are happy, because there are some people, who will use. We have meet our goals and we can say with pride that we achieved great success!

Bibliography

[1] A. Author, *Title of a book*, Publisher, year, page–page.

[2] J. Bobkowski, S. Dobkowski, Title of an article, *Magazine X, No. 7*, year, PAGE–PAGE.

[3] C. Brink, Power structures, *Algebra Universalis 30(2)*, 1993, 177–216.

[4] F. Burris, H. P. Sankappanavar, *A Course of Universal Algebra*, Springer-Verlag, New York, 1981.

Glossary

Administrator	A person who is responsible for the upkeep, configuration, and reliable operation of a system.
Algorithm	A procedure or formula for solving a problem, based on conductiong a sequence of specified actions. A computer program can be viewed as an elaborate algorithm.
Angularjs	A structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly.
Application	A program designed to perform a specific function directly for the user.
Application Service	A services that are made available from a business's Web server for Web users or other Web-connected programs.
Asynchronous	An adjective describing objects or events that are not coordinated in time.
Authorization	The process of giving someone permission to do or have something. In multi-user computer systems, a system administrator defines for the system which users are allowed access to the system and what privileges of use (such as access to which file directories, hours of access, amount of allocated storage space, and so forth).

Back-End	Application serves indirectly in support of the front-end services.
Backup	Copying of physical or virtual files or databases to a secondary site for preservation in case of equipment failure or other catastrophe.
Boilerplate	A unit of writing that can be reused over and over without change. By extension, the idea is sometimes applied to reusable programming as in "boilerplate code."
Bootstrap	A free and open-source front-end library for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.
Browser	An application program that provides a way to look at and interact with all the information on the World Wide Web. The word "browser" seems to have originated prior to the Web as a generic term for user interfaces that let you browse (navigate through and read) text files online.
Cloud	A host on virtual server.
Cloud-Based	A software program where cloud-based and local components work together.
Commit	The final step in the successful completion of a previously started transaction in a computing system.
Continuous Deployment	A strategy for software releases wherein any code commit that passes the automated testing phase is automatically released into the production environment, making changes that are visible to the software's users.

Dao (Data Access Objects)	An API that lets a programmer request access to an database.
Data	The information that has been translated into a form that is more convenient to move or process.
Data Availability	An assurance that data continues to be available at a required level of performance in any situation.
Database	A collection of data that is organized so that its contents can easily be accessed, managed, and updated.
Deploy	To spread out or arrange strategically.
Deprecated	Something is acknowledged but discouraged.
Development	The process of conceiving, specifying, designing, programming, documenting, testing, and bug fixing involved in creating and maintaining applications, frameworks, or other software components.
Distribution	The phase that follows packaging.
Endpoint	Defines the address for a resource , an endpoint is any user device connected to a network.
Event	Any identifiable occurrence that has significance for system hardware or software. User-generated events include keystrokes and mouse clicks, among a wide variety of other possibilities.
Feature	A distinguishing characteristic of a software item (e.g., performance, portability, or functionality)
Field	A location for a single piece of data in a database.

Foreign Key	A key that targets a primary key in another table.
Framework	A real or conceptual structure intended to serve as a support or guide for the building of something that expands the structure into something useful.
Front-End	An application that interacti with users directly.
Gui	A graphical user interface to a computer.
Header	Something that goes in front of something else and is usually repeated as a standard part of the units of something else
Heroku	A cloud-based development platform as a service (PaaS) provider.
Hosting	The business of housing, serving, and maintaining files for one or more Web sites. More important than the computer space that is provided for Web site files is the fast connection to the Internet.
Html (Hypertext Markup Language)	A standard programming language for describing the contents and appearance of Web pages.
Insatnce	In object-oriented programming (OOP), is a specific realization of any object. An object may be varied in a number of ways. Each realized variation of that object is an instance. The creation of a realized instance is called instantiation.
Integration	The act of bringing together smaller components into a single system that functions as one.

Interface	A group of related methods with empty bodies. Implementing an interface allows a class to become more formal about the behavior it promises to provide. Interfaces form a contract between the class and the outside world, and this contract is enforced at build time by the compiler.
Iterative	Used to describe a situation in which a sequence of instructions can be executed multiple times. One pass through the sequence is called an iteration. If the sequence of instructions is executed repeatedly, it is called a loop, and we say that the computer iterates through the loop.
Jar (Java Archive)	A package file format typically used to aggregate many Java class files and associated metadata and resources (text, images, etc.) into one file for distribution.
Java	Aidely used programming language expressly designed for use in the distributed environment of the internet.
Javascript	An interpreted programming or script language from Netscape.
Json (Javascript Object Notation)	Text-based, human-readable data interchange format used for representing simple data structures and objects in Web browser-based code.
Junit	An open source framework designed for the purpose of writing and running tests in the Java programming language.
LIME	Laboratory Internal Management Entity - the name of the application discussed in this document. LIME is the system for resources management in a small chemical laboratory.
Linux	An open-source operating system modelled on UNIX.

Login	A string used to differentiate between users.
Manager	A person who is responsible for the managing resources, products and reports
Mysql	An open source relational database management system that uses SQL.
Password	A string that authorize access for a gine user login.
Plugin	A program that can easily be installed and used as part of another program.
Primary Key	A key in a relational database that is unique for each record, used to identify a particular record.
Query	A request to a database for information, or to update, modify, or delete information.
Relational Database	A collection of data items organized as a set of formally-described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables.
Rest (Representational State Transfer)	Simple stateless architecture that generally runs over HTTP. It is used to communicate between applications.
Rollback	The undoing of partly completed database changes when a database transaction is determined to have failed.
Row	A group of fields in a database table organized to contain all the information relevant to a specific entity.

Server	A computer program that provides a service to another computer programs (and its user). In the client/server programming model, a server program awaits and fulfills requests from client programs, which may be running in the same or other computers.
Software	General term for the various kinds of programs used to operate computers and related devices.
Spring Framework	a Injection dependency framework at first (it's still as it is today) targeting managing life-cycle of Java components (what so-called beans).
Springboot	A suite, pre-configured, pre-sugared set of frameworks/technologies to reduce boiler plate configuration providing you the shortest way to have a Spring web application up and running with smallest line of code/configuration out-of-the-box.
Spring security	A Java/Java EE framework that provides authentication, authorization and other security features for enterprise applications.
Sql (Structured Query Language)	A standard interactive and programming language for getting information from and updating a database.
Staff	A person who is responsible for endpoint work and jobs.
Synchronous	An adjective describing objects or events that are coordinated in time.
Table	In a relational database, a data structure that organizes the information about a single topic into rows and columns.
Upgrade	A new version of or addition to a hardware or, more often, software product that is already installed or in use.

Validation	A quality assurance used to check if the typed data is correct.
Virtual Server	A server (computer and various server programs) at someone else's location that is shared by multiple Web site owners so that each owner can use and administer it as though they had complete control of the server.
War	A file used to distribute a collection of JAR-files, JavaServer Pages, Java Servlets, Java classes, XML files, tag libraries, static web pages (HTML and related files) and other resources that together constitute a web application.
Web Application (Web App)	An application program that is stored on a remote server and delivered over the Internet through a browser interface.

List of Figures

If you don't need it, delete it.

List of tables

If you don't need it, delete it.

List of appendices

A. User Manual

There we will have manual

B. Test Report

Testi Report