

Czech Technical University in Prague
Faculty of Electrical Engineering



Information system for managing student club members

Bachelor thesis

Varvara Muzovatova

Study program: Software Engineering and Technology

Supervisor: Ing. Jiří Šebek

Prague, January 2022

Thesis Supervisor:

Ing. Jiří Šebek
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2
160 00 Prague 6
Czech Republic

Copyright © January 2022 Varvara Muzovatova

In Prague, January 2022

.....
Varvara Muzovatova

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Muzovatova** Jméno: **Varvara** Osobní číslo: **371311**
Fakulta/ústav: **Fakulta elektrotechnická**
Zadávající katedra/ústav: **Katedra počítačů**
Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Informační systém pro správu členů studentského klubu

Název bakalářské práce anglicky:

Information system for managing student club members

Pokyny pro vypracování:

Provedte analýzu řešení dané problematiky. Navrhněte a implementujte aplikaci pro správu členů studentského klubu Masařka. Řešení implementujte jako webovou aplikaci primárně s použitím technologií Java (Spring Boot), JavaScript, Docker a případně dalších vhodných dodatečných technologií.

Aplikace by měla umožňovat a realizovat následující funkcionality:

- 1) evidence členů klubu a spravování jejich osobních informací
- 2) autentizace uživatelů a autorizace k oprávněným funkcím na základě uživatelských rolí
- 3) spravování zařízení členů klubu pro přístup k internetu
- 4) funkcionality rezervačního systému pro přístup členů klubu do společenských a sportovních místností Masarykovy koleje
- 5) periodické stahování transakcí z bankovního účtu přes API banky
- 6) párování transakcí s účty členů klubu a evidenci plateb členských příspěvků

Provedte nasazení webové aplikace na produkční server. Zhodnoťte přínos aplikace z pohledu koncových uživatelů informačního systému.

Seznam doporučené literatury:

- [1] BASTANI, Kenny. LONG, Josh. Cloud Native Java: Designing Resilient Systems with Spring Boot, Spring Cloud, and Cloud Foundry. Sebastopol: O'Reilly Media, 2017. ISBN 978-1449374648
- [2] CARNELL, John. Spring microservices in action. Shelter Island, NY: Manning, 2017. ISBN 978-1-61729-398-6
- [3] HANCHETT, Erik. Vue.js in action. Shelter Island: Manning, 2018. ISBN 978-1-61729-524-9
- [4] WALLS, Craig. Spring Boot in action. Shelter Island, NY: Manning, 2016. ISBN 978-1-61729-398-6
- [5] PECINOVSKÝ, Rudolf. Návrhové vzory. Brno: Computer Press, 2013. ISBN 978-80-251-1582-4
- [6] HEROUT, Pavel. Testování pro programátory. České Budějovice: Kopp, 2016. ISBN 978-80-7232-481-1

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jiří Šebek, kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **21.02.2021**

Termín odevzdání bakalářské práce: **04.01.2022**

Platnost zadání bakalářské práce: **19.02.2023**

Ing. Jiří Šebek
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Mgr. Petr Páta, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání

Podpis studentky

Abstract

The aim of this bachelor's thesis is to create a new information system for the Masarka student club, which will eliminate the shortcomings of the existing solution and simplify the registration of club members and the management of access to the club services. This thesis describes the entire process of development starting with an analysis of the former information system, the result of which was a summary of all available functionality and its shortcomings. Based on the performed analysis and specified requirements a new information system has been designed and implemented. After the implementation was completed, the new system was deployed to the production server and tested by the functional testing. At the end, the evaluation of the new system from the perspective of end users was performed as well as the proposal for future extensions was given.

Keywords: Information system, Java, Spring Boot, Vue.js, JWT

Abstrakt

Cílem této bakalářské práce je vytvořit nový informační systém pro studentský klub Masařka, který odstraní nedostatky stávajícího řešení a zjednoduší evidenci členů klubu a správu přístupu ke klubovým službám. Tato práce popisuje celý proces vývoje počínaje analýzou bývalého informačního systému, jejímž výsledkem bylo shrnutí veškeré dostupné funkcionality a jejích nedostatků. Na základě provedené analýzy a zadaných požadavků byl navržen a implementován nový informační systém. Po dokončení implementace byl nový systém nasazen na produkční server a otestován funkčním testováním. Na závěr bylo provedeno zhodnocení nového systému z pohledu koncových uživatelů a byl podán návrh budoucích rozšíření.

Klíčová slova: Informační systém, Java, Spring Boot, Vue.js, JWT

Acknowledgements

I would like to thank my supervisor Ing. Jiří Šebek for his help, useful advice and friendly attitude throughout the writing of my thesis.

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.

List of Tables

2.1	React advantages and disadvantages comparison [3] [6]	11
2.2	Angular advantages and disadvantages comparison [3] [6]	12
2.3	VueJS advantages and disadvantages comparison [3] [6]	12

List of Figures

3.1	Project architecture diagram	15
3.2	Class diagram	18
3.3	Session-based authentication	19
3.4	Token-based authentication	20
4.1	Spring Security objects interactions	26
4.2	Example of REST API using the Swagger UI	28
4.3	Login view	32
4.4	Registration view	33
4.5	Profile view	34
4.6	Reservations view	35
4.7	Banned reservations view	35
4.8	Admin panel - Users view	36
4.9	Admin panel - User roles view	37
5.1	Graph for compiling possible testing scenarios	43

Contents

Abstract	v
Abstrakt	vi
Acknowledgements	vii
Declaration	viii
List of Tables	ix
List of Figures	x
1 Introduction	1
1.1 Motivation	1
1.2 Aims of project	2
2 Analysis	3
2.1 Analysis of the former system	3
2.1.1 Technologies and security	3
2.1.2 Functionality and usability	3
2.1.3 Perspectives of repetitive exploitation	4
2.2 Analysis of user requirements	4
2.2.1 Functional requirements	4
2.2.2 Non-functional requirements	7
2.3 Research of existing solutions	8
2.3.1 Information system of Silicon Hill club	8
2.3.2 Information system of Pod-o-lee club	8
2.3.3 Information system on the Hlavkova dormitory	9
2.3.4 Conclusion	9
2.4 Research of front-end technologies	9
2.4.1 React	10
2.4.2 Angular	10
2.4.3 Vue.js	10
2.4.4 Comparison of advantages and disadvantages	11
2.5 Back-end technologies	13
2.5.1 Spring Boot	13

3	Design of new information system	14
3.1	Architecture and used technologies	14
3.1.1	Additional front-end components	15
3.2	Data model	17
3.3	Authentication	18
3.4	Authorization	20
4	Implementation	22
4.1	Back-end application	22
4.1.1	Spring Boot project initialization	22
4.1.2	Spring datasource configuration	23
4.1.3	Spring cache configuration	23
4.1.4	Spring Java Mail Sender configuration	23
4.1.5	Fetching of payments via bank API	24
4.1.6	Spring security configuration	25
4.1.7	Spring Boot project directory structure	27
4.2	Front-end application	28
4.2.1	Prerequisites	28
4.2.2	Vue.js project initialization	28
4.2.3	Front-end application directory structure	29
4.2.4	Axios configuration	30
4.2.5	Vue Router configuration	31
4.2.6	Vuex store	31
4.2.7	UI development	32
4.3	Deployment	37
5	Testing	39
5.1	Functional testing	39
5.1.1	Back-end API testing	39
5.1.2	Selenium tests	40
5.1.3	User testing	44
6	Conclusion	45
6.1	Evaluation of the system from the perspective of end users	46
6.2	Ideas for further improvements	46
6.2.1	Integration of payment gateway	47
6.2.2	Users activity logging	47
6.2.3	Language localisation	47
	Bibliography	49

Chapter 1

Introduction

Around 1997, a group of students created an experimental computer network at Strahov dormitories, which in mid-1998 outgrew the dimensions of a private local network. Therefore, the network had been dealing with numerous legislative and financial problems. The solution turned out to be relatively simple, to integrate the functioning of the network into the newly established [CTU Student Union](#) in the form of an autonomous club. This is how the Silicon Hill club was founded, which became the first club of [CTU Student Union](#). Gradually, students have built all the infrastructure, not only the network, but also application systems. All this has been performed within their own resources and independently, often as various semester projects or bachelor's and master's theses. As soon as this model proved successful, other dormitory clubs began to emerge and over time became part of the [CTU Student Union](#). One of them is a [Masarka Student Club \(MAS\)](#), which was created by merging the Student Self-government of Masaryk dormitory and the [MAS](#) club, which has so far been the club of administrators of the Masaryk dormitory network and their network users. [1]

Over time, the list of services offered by clubs grew as did the number of active members, and today clubs on most dormitories offer their members other services, such as running fitness centers and entertainment rooms, organizing leisure activities and much more, however the main mission of all dormitory clubs remains ensuring of a high-speed internet connection and access to online study materials for students accommodated in [Czech Technical University in Prague \(CTU\)](#) dormitories.

1.1 Motivation

Today Masarka student club offers its members, who are mainly students accommodated in Masaryk dormitory, two main categories of services. The first one is access to high-speed internet connection via Wi-Fi and via cable connection, the second is an access to sport and entertainment rooms available in Masaryk dormitory.

To ensure the effective functioning of the club, especially for the purposes of registration of club members, management of their personal data, management of access to individual services and an overview of paid membership fees, Masarka student club requires a proper information system.

For many years, in [MAS](#) club there had been used the DIS information system for these purposes, implemented in pure PHP far in 2008, also known as the “Disinformation System”, what doesn’t sound very optimistic. The former information system is very confusing, outdated and no longer meets today’s common standards. In the recent period, there have also been some changes in the configuration of network elements and the network topology, as a result of which the existing system became to be not usable anymore. It’s further exploitation would require a fundamental rework of the system. Due to these reasons, it seems much more appropriate to implement a completely new system.

As a member of the board of the [MAS](#) club and also a long-term active member, I have sufficient experience with the shortcomings of the previous information system and I also have a concrete idea of what features and functionalities the new information system should offer.

1.2 Aims of project

The aim of this bachelor’s thesis is to create a new information system for the [MAS](#) club, which will eliminate the shortcomings of the existing solution and simplify the registration of club members and the management of their services.

In the first steps the research and analysis will be performed. Based on the information obtained from analysis, comprehensive requirements for the new information system of the [MAS](#) club will be written. Subsequently, this system will be implemented and its functionality tested. Finally, an evaluation of the entire system will be made and proposals for its possible further development will be presented.

As some of the requirements of the new information system are closely related to the computer network, some chapters may contain basic information about network layer, however, network topology, mounting or configuration of network elements, and other network issues are not part of this project. Network layer and network elements are already configured by network administrators and this project is focused on creation of the information system.

Chapter 2

Analysis

This chapter deals with the analysis of user requirements, research of the former system and other already existing solutions as well as choosing the relevant technologies for application development. Thanks to the analysis, it is possible to correctly determine the requirements and then design the system, which will best meet user requirements.

2.1 Analysis of the former system

As it was already mentioned, since the year 2008 the MAS club has been using the DIS system, also known as the "Disinformation System". The system is very confusing, outdated and after recent changes in network configuration, it no longer meets today's requirements.

2.1.1 Technologies and security

The former information system has been implemented using pure HTML, CSS and PHP without using any libraries or frameworks. It also contained a set of support scripts written in Bash, Python and PHP, which helped with automatization of some tasks. The data storage function is performed by PostgreSQL RDBMS.

The security of the former system was very vulnerable and couldn't be guaranteed. Therefore an access to the information system has been regulated by firewall configuration, allowing access only from IP addresses owned by network administrators.

2.1.2 Functionality and usability

The system successfully deals with its primary function to store information about club members. It allows to store records of new pending registrations, records of already registered members and the records of their devices and membership payments. The system also offers the functionality of facilities management system, which actually wasn't fre-

quently used because of its impracticality.

The main shortcoming of the former information system is that it does not distinguish any user roles. The user simply can login the system, if its IP address has been added into the access list. And if the user has access, then it already has access to everything.

Another shortcoming is that the reservation system, which allows members to make reservations on sport and entertainment rooms, is not fully integrated into the common information system and some simple operations, like adding a new room or banning a user, could only be performed manually by direct database manipulation.

2.1.3 Perspectives of repetitive exploitation

The former information system successfully deals with its primary task, the club members evidence. But it has a lot of disadvantages and if it should be used again, then it definitely will be necessary to solve all its shortcomings, first of all the security issues at least.

The implementation of a new information system in the MAS club has been under consideration for a long period. In my opinion, it is much more reasonable to implement a new system than to rework the former one.

2.2 Analysis of user requirements

Based on the experiences obtained during my work in MAS club, as well as based on the analysis of the former information system and its shortcomings, that have been described in the previous part, I will try to formulate basic requirements of the new information system, in the following section.

2.2.1 Functional requirements

Functional requirements are product features, which focus on user requirements and define the basic system behaviour. Essentially, implementation of these features allows the system to function as it was intended.

This section will first describe user authorization roles that the system should implement, including the detailed description of their access rights. Afterwards, there will be described specific features of the information system related to the evidence of club members, evidence of received membership payments and features related to the providing of services to club members.

User authorization roles

The MAS club is managed by a group of active members, that includes chairman of the club, club board members, network administrators, registrars and facility managers, each of them performs some specific function in the club.

This distribution of functions should be reflected in the new information system by providing the specific access rights to each group of users. The system definitely should have at least the following user roles:

- Administrator
- Club chairman
- Club board member
- Registrar
- User

Administrator

The user with administrator role has an access to all system functionalities, except the ability to manage user roles of other members, this feature is available only for the club chairman. The list of administrator access rights:

- to manage and edit all system records, except the modification of user roles
- to register new club members
- to manage all system settings
- to ban (deactivate) the users for violation of club rules
- to activate the user without valid membership payment
- to assign a payment that cannot be automatically paired with a specific user

Club chairman

The club chairman has the same access rights like the administrator, but additionally he is allowed to assign and change the user roles of any club member. Essentially, the club chairman has absolutely all access rights.

Club board member

Club board members have access to absolutely all information in the system, but they only can read it and can not modify any data.

Registrar

The registrars are those active members who help the MAS club with registration of new members. They can confirm pending registrations and manage already registered users.

The list of registrar access rights:

- to register new club members
- to manage and edit users personal information, except the modification of user roles
- to reset users password
- to review users payments, but without ability to modify
- to print paper registration form

Facility managers

Facility managers are those active members who help the MAS club with managing sports or entertainment rooms.

The list of facility manager access rights:

- To review users personal information
- To review users reservations
- To manage settings related with reservation system

User

This role is implicitly assigned to each new member of the club.

The list of user access rights:

- To review its own profile and personal information
- To review its own membership payments
- To edit contact email address
- To edit password

Management of club members records

The application should allow to store and manage the records of club members. According to the CTU Student Union regulations, for each member, it is necessary to register the name, surname, date and place of birth, address of permanent residence, e-mail address and registration date. Optionally the system can store some additional information like phone number or dormitory room number.

Registration of new members

The application should enable any not registered user to apply for a new membership. Pending registrations are stored in a separate table and only after the identity of the person is verified and paper registration form is signed, then the registration can be completed and a new user record is created. The new members can be registered only by users authorized by the club chairman, usually registrars and administrators.

Management of payments records

The new information system should be connected to the club bank account via its API and information about new transactions should be regularly downloaded to the information system database. Incoming payments should be automatically assigned to users based on the matching of variable symbol and user ID. In the case that incoming payment can not be assigned automatically, the administrators or club chairman should be able to assign it manually.

Reservations of sport and entertainment rooms

The new information system should integrate the reservation system for sport and entertainment rooms managed by MAS club. Access to these rooms is one of the services available to club members with paid membership fee. Users should be able to create reservations on the desired room. Facility managers should have an ability to ban (deactivate) access to the reservation system to users for violation of MAS club rules.

2.2.2 Non-functional requirements**Security**

The new information system must ensure the security of user sensitive data and only authorized user should be able to access users personal data.

Usability

The new information system should have clear and responsive design. It should work fine on mobile and desktop devices. Working with this system should be easy and intuitive without any necessity to learn users how to use it. All significant deletion operations must be confirmed by pop-up dialogues to prevent accidental deletion of data.

Maintainability

The implemented solution should be easy to maintain. Administrators should be able to easily monitor the status of the application and be able to restart it at any time if necessary for some reason.

Extensibility

The implemented system should take into account possible further extensions, so it should be appropriately designed with this in mind.

2.3 Research of existing solutions

Research of the existing technologies was focused mainly on the information systems of other clubs of the CTU Student Union, as their systems represent the closest alternative to the system that satisfy requirements of the Masarka club. In the following section will be presented only the systems that represent the highest perspective.

2.3.1 Information system of Silicon Hill club

The Information system of Silicon Hill club is a professional fully horizontally scalable application. Its front-end was implemented using the Twitter Bootstrap framework. The back-end application was implemented using the framework Ruby on Rails and PostgreSQL as a datastore.

The system allows to distinguish different membership types and provides a pretty versatile management of club members. But the system is primarily adapted to manage the access to the internet connection and it doesn't provide many options to manage other types of services. Additionally, the system provides management of the club's assets and an overview of drawing funds from the club's budget.

Information system of the Silicon Hill is definitely a professional and very complex solution that proved itself on Strahov dormitories, where it has been operated to manage thousands of users and their devices. Deployment of this system for the Masarka club purposes would require an extensive adaptation of computer network configurations. Additionally it will be necessary to extend the system and implement own reservation system. Therefore, exploitation of this system for the Masarka club purposes is considered as too complex.

2.3.2 Information system of Pod-o-lee club

The information system of Pod-o-lee club is built from individual micro-services that are implemented in Python. The system contains the following micro-services:

- Interface for an overview and management of club members.
- Micro-service for administration of whole system.
- Micro-service for communication with bank API.

- Micro-service for communication with card chip system

All services are controlled by the front-end application that was implemented using the Twitter Bootstrap framework. Services are implemented mainly in Python language using the Django framework. Communication between individual services is performed by RabbitMQ message broker.

2.3.3 Information system on the Hlavkova dormitory

The information system on the Hlavkova dormitory was implemented using the Twitter Bootstrap technology for front-end and PHP language with Nette Framework for the back-end application. The information system provides the following functionalities:

- Management of club members.
- Management of devices and their access to the internet connection.
- Management of user payments and their automatic downloading via bank API.
- Integration with e-mail server and possibility to send emails.
- Integration with LDAP database.
- Generating of reports.
- Generating and printing of user registration form.

2.3.4 Conclusion

All three information systems, presented in this section, could theoretically be used for the needs of the Masarka club. They all provide a sufficient functionalities to manage club members, their devices and payments. But none of them has an implemented functionality of the reservation system and all related issues. Another disadvantage is that each of these systems requires its own specific computer network configuration and topology. This mean that eventual exploitation of any of these systems would require an extensive adaptation to make them suitable for the club needs.

2.4 Research of front-end technologies

This section will focus on research and comparison of JavaScript front-end technologies. As I had almost no serious experiences with front-end development before, this analysis will be based mainly on numerous articles available on internet. According to many resources, today the top three most popular JavaScript front-end technologies are React, Vue.js and Angular. At first these three technologies will be briefly introduced, afterwards some pros and cons will be presented.

2.4.1 React

React is an open-source JavaScript library for building user interfaces which has been developed by Facebook engineers. It provides a powerful model for work and helps to create declarative and component-oriented users interfaces featuring JSX syntax across multiple platforms. [2] The key feature of React is a virtual Document Object Model (DOM) with one-way data binding. Since React is a library, unlike other frameworks, it does not maintain some important features. That's why React is meant to work together with other libraries, such as for state management, routing, and interaction with API. [3]

Key features:

- Open-source JavaScript library for building user interfaces, originally created by engineers at Facebook.
- React provides simple and flexible component-based API.
- Components are a fundamental unit in React are extensively used in React applications.
- React implements a virtual DOM layer between the program and the browser DOM.
- Virtual DOM allows efficiently update browser DOM using a fast diffing algorithm.
- The virtual DOM allows for excellent performance. [2]

2.4.2 Angular

Angular or also referred as Angular 2+ is a modern TypeScript based open-source JavaScript library that is sponsored and maintained by Google and has been used in some of their large and complex web applications. Angular released in 2016 is an improved edition of AngularJS, with a boosted performance and a bunch of powerful features added. Angular taps into some of the best aspects of server-side development and uses them to enhance HTML in the browser, creating a foundation that makes building rich applications simpler and easier. Angular ensures two-way data binding for immediate synchronization between the model and the view, so any change in the view will instantly reflect in the model and in reverse. Angular features Directives that allow developers to program special behaviors of the DOM, making it possible to create rich and dynamic HTML content. Angular applications are built around a clear design pattern that emphasizes creating applications that are easily extendable, maintainable, testable and standardized. [4] [3]

2.4.3 Vue.js

Vue.js is a progressive open-source JavaScript library that allows to add interactive behavior and functionality into context wherever JavaScript is running. It can be used to

create separate websites, or be the basis for an entire enterprise application. [5]

Vue.js is one of the most popular front-end frameworks nowadays that similarly to React features virtual DOM and component-based architecture but offers a two-way data binding, which underlies its high-speed performance. All that makes it easier to update related components and track data changes, what is desired for any application in which real-time updates are a must. [3]

2.4.4 Comparison of advantages and disadvantages


	
Advantages	Disadvantages
<ul style="list-style-type: none"> • Reusability of components • Virtual DOM - consistent and seamless performance • Can be combined with many other JS libraries • Frequently updated • Backed by Facebook 	<ul style="list-style-type: none"> • Lack of a well-elaborated documentation • Complexities related to JSX syntax

Table 2.1: React advantages and disadvantages comparison [3] [6]


	
Advantages	Disadvantages
<ul style="list-style-type: none"> • Component-based architecture • Two-way data binding • Directives and dependency injection features • Highly testable / manageable applications • Strong community, good training materials, etc. • Backed by Google 	<ul style="list-style-type: none"> • Difficult for beginners and overwhelming for smaller teams • Limited SEO capabilities • Bloated code and large in size

Table 2.2: Angular advantages and disadvantages comparison [3] [6]


	
Advantages	Disadvantages
<ul style="list-style-type: none"> • Fast and tiny in size • Excellent documentation • Component-based architecture • Beginners friendly • Simple syntax • Typescript support • Two-way data binding • Has a positive effect on SEO 	<ul style="list-style-type: none"> • Pretty new and developed by private individuals • Relatively small community • Has no powerful business behind it

Table 2.3: VueJS advantages and disadvantages comparison [3] [6]

2.5 Back-end technologies

Despite the fact that back-end technologies is a quite broad topic, this section will be limited and focused mainly on a Java and Spring Boot framework. Therefore, it will be more of a brief introduction to specific technology rather than a full research of a given topic. It is my personal choice and it is reasoned by the fact that Java and Spring Boot are the prime back-end technologies experienced and taught within my study program at the CTU.

2.5.1 Spring Boot

Spring is an open-source framework that provides comprehensive infrastructure support for developing Java applications. Spring Boot is basically an extension of the Spring framework, which eliminates the boilerplate configurations required for setting up a Spring application. Its main goal is to make Spring applications development faster and more efficient.[7]

Main Spring Boot features:

- Ability to create stand-alone Spring applications
- Embedded web server Tomcat, Jetty or Undertow
- Provides set of 'starter' dependencies to simplify build configuration
- Avoidance of complexities with XML configurations
- Provides production-ready features such as metrics, health checks, and externalized configuration

Additionally, at the Spring official web-site there is available a Spring Initializr tool that allows to prepare pre-configured Maven or Gradle projects with selected dependencies and makes the development of Spring Boot applications even easier.

Chapter 3

Design of new information system

An analysis performed in the previous chapter revealed that the most of existing solutions as well as the club's former information system are inappropriate for the purposes of MAS club. Their exploitation would require extensive modifications to satisfy all the requirements of our club, therefore it seems to be much more reasonable to implement new information system from scratch.

In this chapter the new information system will be designed. It will be designed primarily with regard to today's commonly used trends, with the main emphasis on the simplicity of the system and especially on its easy extensibility.

3.1 Architecture and used technologies

Based on the performed analysis and user requirements it has been chosen, that new information system should be designed to be a web application of client-server model. As it is expected that the functionality of the new information system may be extended in the future, or it may be integrated with some other applications, the use of client-server architecture will allow an easier integration of possible extensions.

The new information system will be developed as two separate applications, one for the server side and one for the client side. Therefore in all further sections this will be taken into account and distinguished respectively. The figure 3.1 demonstrates a simplified project architecture diagram.

The selection of technologies was impacted mainly by today's current trends and my personal experiences from studying on [CTU](#). The development of this application is performed for exploitation in the [MAS](#) club, what is a non-profit organization, therefore the selection of open-source libraries and frameworks is highly preferable.

The Java language and the Spring Boot framework were chosen as a core technology for

the back-end application. Beside my personal preferences, the reason for choosing Java and Spring Boot framework is also the fact that Java and Spring Boot framework are frequently taught on CTU and many administrators of the MAS club have also experienced it during their studies. This also increases the chance that other programmers will take over the further development of the system. The back-end application will be designed to implement a RESTful API that will provide endpoints for manipulation with data.

The front-end part will be developed as a single-page application using JavaScript programming language and Vue.js framework, which belongs to one of the most popular open-source JavaScript frameworks today. Since I had no experience with any JavaScript framework before, my personal choice for Vue.js can be reasoned mainly because of its detailed documentation, excellent performance as well as its rising popularity and rich variety of additional components.

In the following sections there will be briefly described all additional components and technologies selected to be used within this project.

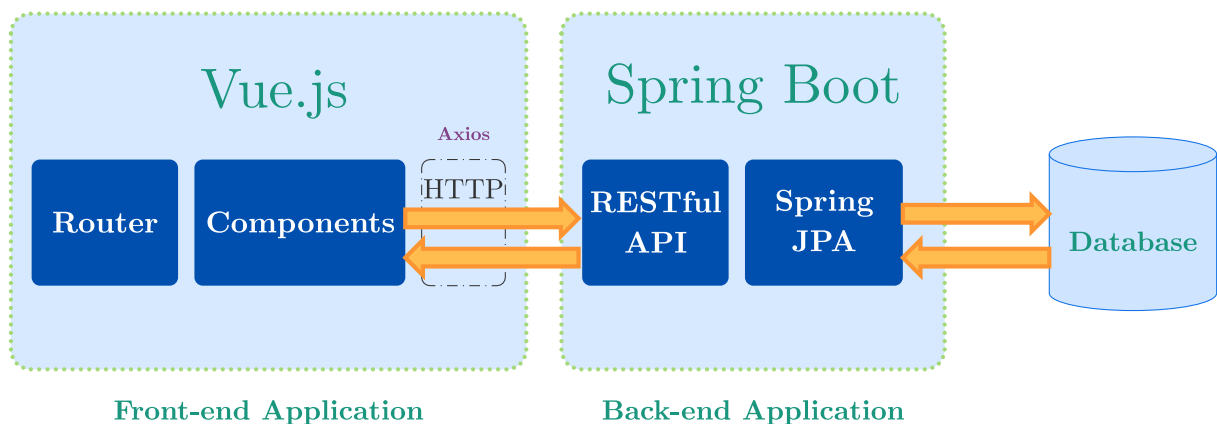


Figure 3.1: Project architecture diagram

3.1.1 Additional front-end components

Vue.js

Vue.js is a progressive and one of the most popular open-source JavaScript frameworks nowadays. More detailed information about this framework and its features has been presented in the previous chapter.

Vuetify

Vuetify is a complete UI framework built on top of Vue.js. It is developed exactly according to Material Design specification with every component crafted to be modular, responsive, and performant. The main goal of the project is to provide developers with the plenty of tools and components they need to build rich and responsive user interfaces. Vuetify puts a great emphasis on the responsive design and ensures that its applications just works out of the box whether it's on a phone, tablet, or desktop computer. [8]

Vuetify has a very active development cycle and is patched weekly. In addition, every major release is accompanied with 18 months of long-term support for the previous minor version. [8]

In this project I have decided to use Vuetify for the purposes of UI development as it seems to be a good choice for it, as well as it seems to fully satisfy the responsive design user requirement.

Vue Router

Vue Router is the official library for page navigation in Vue.js applications. It deeply integrates with Vue.js core library and allows building of the SPA (Single Page Applications) by associating certain URL routes to the specific components that should be rendered. [9]

In this project Vue Router component will be used for implementation of SPA page navigation.

Vuex

Vuex is a the official state management pattern and library for Vue.js applications. It serves as a centralized store for all the components in an application, with rules ensuring that the state can only be mutated in a predictable fashion. [10]

Within this project, Vuex will be used to store the information of logged-in user and its JWT token.

Axios

Axios is a promise-based HTTP client library. It will be used within the client application to make the asynchronous requests to the RESTful API of the server application.

VeeValidate

VeeValidate is very useful form validation library for Vue.js applications. It enables not only to ensure correct values are submitted, but also provides a pleasant UX (User Experience) for users and UI related features. VeeValidate allows significantly to save the time spent working on custom form validation solutions. [11] VeeValidate main features:

- Tracking form state
- UI and UX
- Synchronous and asynchronous validation
- Handling submissions

In this project I have decided to use VeeValidate for the form validation tasks.

3.2 Data model

The figure 3.2 contains a class diagram that describes project entities and relationships between them.

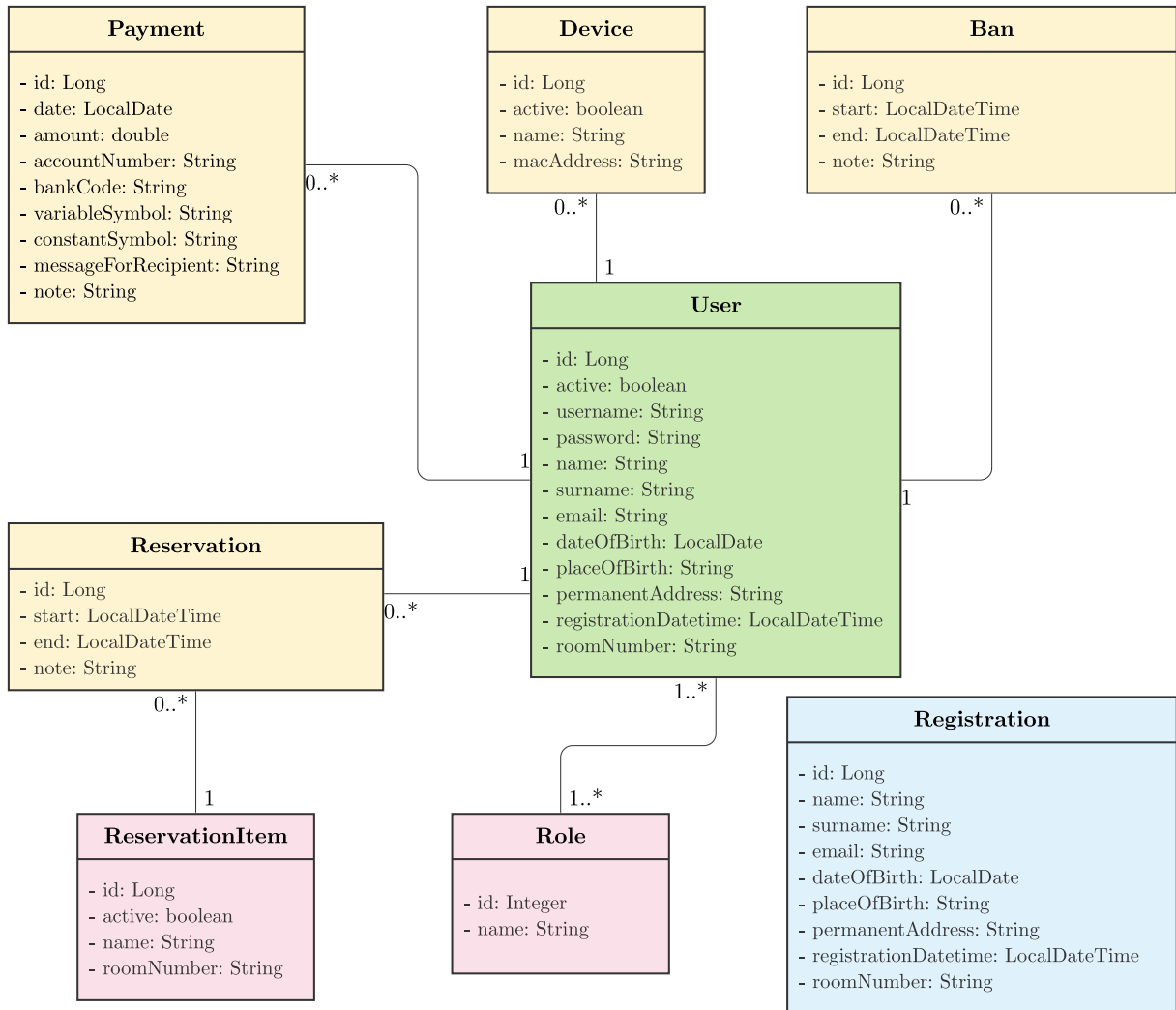


Figure 3.2: Class diagram

3.3 Authentication

When designing this application, two authentication options were taken into account. The first option is a session-based authentication, quite simple and popular authentication method, that is frequently used on many websites.

The figure 3.3 demonstrates a process of session-based authentication, when a user logs into a website, the server will generate a session for that user and store it in memory or database. Server then returns a session id for the client to save it in browser cookies. Each session on the server has its expiration time. After that time, the session expires and the user must re-login to create another session. If the user has logged in and the session has not expired yet, the cookie, including session id, is always sent with all HTTP requests to the server. Server compares this session id with the stored session to authenticate and

then returns a corresponding response. [12]

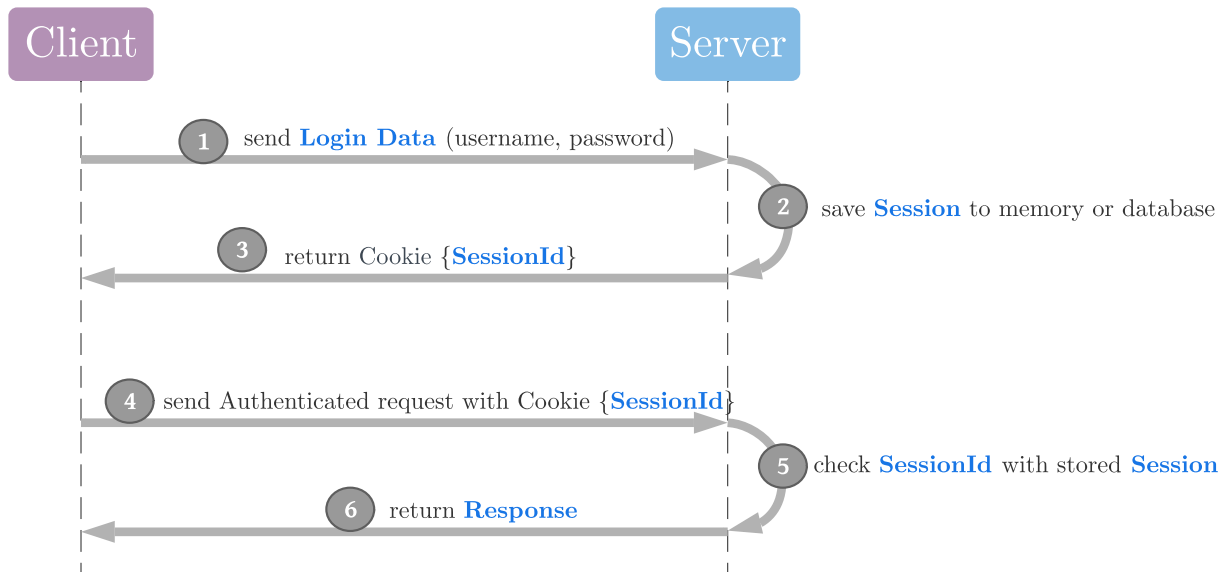


Figure 3.3: Session-based authentication

The session-based authentication has proved itself and works well with many websites. But the problem may occur, if the server application should serve not only for browser clients, but also some others, as not all clients must support the cookies. For example if one day a mobile client-side application will be developed and should be also served by the same server application, then the session-based authentication may be not the best solution. For these cases there exists another option, a token-based authentication, which works in a slightly different way and may be more suitable. [12]

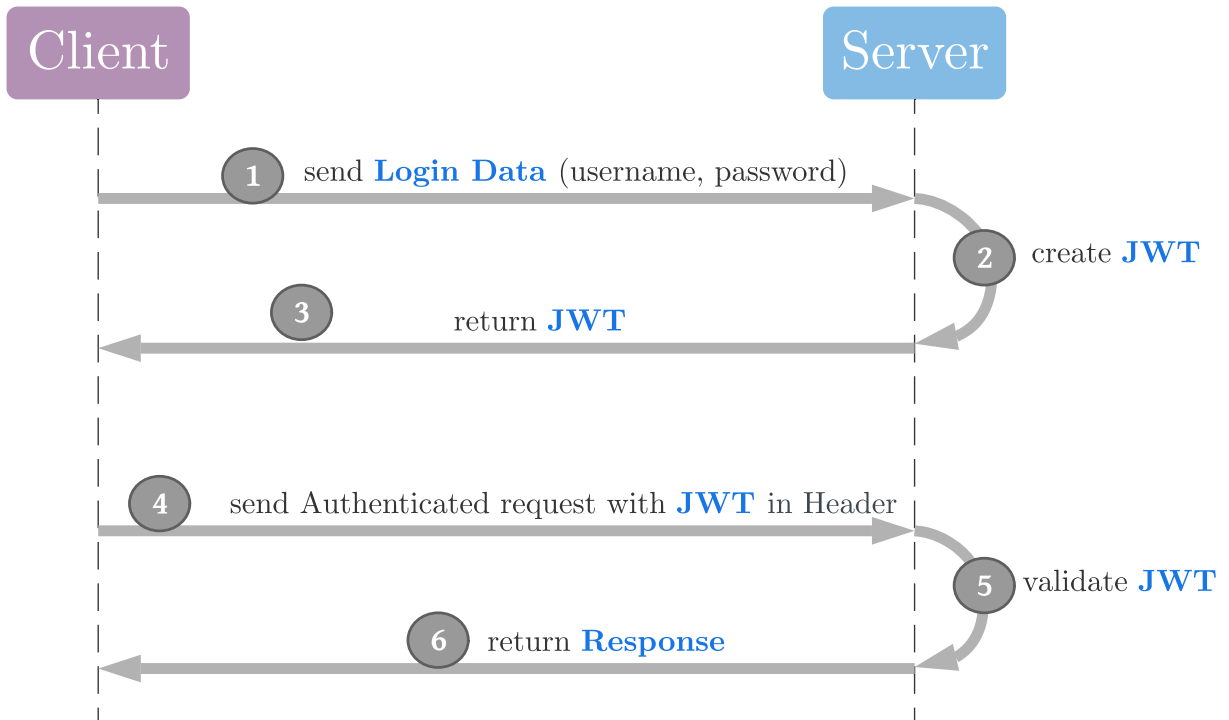


Figure 3.4: Token-based authentication

The figure 3.4 demonstrates a process flow of token-based authentication using **JSON Web Token (JWT)**. Instead of creating a session, the server application generates a **JWT** from user login data and sends it to the client. The client saves the **JWT** and attaches it to every request that requires authentication. The server will then validate the **JWT** and return the corresponding response. In comparison with session-based authentication, that needs to store all sessions on the server side, the big advantage of token-based authentication using **JWT** is that the tokens are stored in a local storage on the client side.

For the purposes of the information system from this project a token-based authentication using **JWT** has been selected. As it was already mentioned, there is a perspective that the given information system may be later integrated with other applications, therefore the token-based authentication seems to be more suitable for this project. [13]

3.4 Authorization

In this project the authorization process will be performed on the both sides, one on the side of back-end application and another control mechanism will be made on the side of front-end application. In the back-end application all authorization and authentication settings will be configured together in Spring Security configuration classes. In the client application authorization will be managed by Vue Router component, according to the predefined rules, which will determine whether the requested view may be accessed by

currently logged-in user or not. Both applications will implement all authorization roles defined in the analysis chapter, with the corresponding access rules.

Chapter 4

Implementation

In this chapter there will be summarized practical steps of implementation of new information system of the MAS club. Development of the information system has been separated into two projects, one for the back-end application and another for the front-end application, therefore all steps will be respectively distinguished for both applications.

4.1 Back-end application

4.1.1 Spring Boot project initialization

The Spring Boot application has been initialized as a Maven project using the Spring Initializr tool, with the following components added:

- **Spring Web** - Component for building Spring web applications, including RESTful applications. It contains embedded Apache Tomcat web server by default.
- **Spring Security** - Component allowing to customize authentication and authorization of the Spring application.
- **Spring Data JPA** - Component allowing to interact with data stores using the Java Persistence API.
- **PostgreSQL Driver** - JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database.
- **Spring Data Redis (Access+Driver)** - Java Redis client that allows Spring applications to interact with Redis cache.
- **Spring Java Mail Sender** - Java Mail client that allows to send e-mails in Spring applications.
- **Thymeleaf** - Java HTML templating engine.
- **Lombok** - Java annotation library which helps to reduce boilerplate code.

- **Apache PDFBox** - An open-source Java library for working with PDF documents.
- **(Optional) Spring Boot DevTools** - Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

As a result, pre-configured project already contains prepared `pom.xml` file with all necessary build configurations and dependencies.

4.1.2 Spring datasource configuration

PostgreSQL server application is running on the MAS server. Specific user account and database for the given project have been created. The following configurations have been added to `application.properties` file:

```
1 spring.datasource.driverClassName=org.postgresql.Driver
2 spring.datasource.url=jdbc:postgresql://mk.cvut.cz:5432/masarka_is
3 spring.datasource.username=masarka_is
4 spring.datasource.password=<password>
```

This is the only place in project, where the database specific credentials are provided. Therefore, theoretically, if it will be needed, database configuration can be easily switched to some other database only by modifying this credentials.

4.1.3 Spring cache configuration

The cache configuration in Spring Boot project is performed similarly like datasource configuration in `application.properties` file. For the purposes of this project a Redis cache has been used. Redis server application has been launched in Docker on the MAS server. The following configurations have been added to `application.properties` file:

```
1 spring.cache.type=redis
2 spring.redis.host=mk.cvut.cz
3 spring.redis.port=6379
4 spring.redis.password=<password>
```

To enable caching, additionally the `@EnableCaching` annotation must be added to main Spring Boot application class. Afterwards cache-specific annotations `@Cacheable`, `@CachePut` and `@CacheEvict` can be used inside the project classes to interact with the cache storage.

4.1.4 Spring Java Mail Sender configuration

The configuration of Java SMTP client in Spring Boot project was performed in application properties file. Masarka club uses Google G-Suite services for hosting an e-mail server. Therefore, the configuration of the e-mail client is similar like configuration of any regular G-Mail account. The following configurations have been added to `application.properties` file:

```
1 spring.mail.host=smtp.gmail.com
2 spring.mail.port=587
3 spring.mail.username=no-reply@mk.cvut.cz
4 spring.mail.password=<password>
5 spring.mail.properties.mail.smtp.auth=true
6 spring.mail.properties.mail.smtp.starttls.enable=true
```

After providing proper configurations to `application.properties` file, sending of e-mails can be easily performed just by importing of an autowired `JavaMailSender` object and then calling of an appropriate functions on it.

```
1 @Autowired
2 JavaMailSender javaMailSender;
3
4 public void sendRegistrationConfirmationEmail(User user) throws
5     MessagingException {
6     // create thymeleaf context object
7     Context context = new Context();
8     context.setVariable("user", user);
9
10    // process email template
11    String emailText = templateEngine.process("emails/registration-
12        confirmation", context);
13
14    // create new email message object
15    javax.mail.internet.MimeMessage mimeMessage = javaMailSender.
16        createMimeMessage();
17    MimeMessageHelper helper = new MimeMessageHelper(mimeMessage);
18    helper.setSubject("Masarka Club Registration");
19    helper.setText(emailText, true);
20    helper.setTo(user.getEmail());
21
22    // send email message
23    javaMailSender.send(mimeMessage);
24 }
```

Listing 4.1: Example of method that sends a registration confirmation email

4.1.5 Fetching of payments via bank API

Regular payments fetching via Fio bank API was solved by implementation of Spring's scheduling tasks. Spring allows to create periodical tasks by adding the `@Scheduled` annotation to the methods. A specific execution time is then defined by a classic cron expression in the annotation of the given function. Additionally, to allow scheduling, the `@EnableScheduling` annotation must be added to main Spring Boot application class.[14]

The process of fetching the payments and a subsequent users activation or deactivation is performed by two scheduled methods. One scheduled method just downloads the

payments and assigns them to the corresponding users. The second scheduled method then iterates all users and activates only those, who have assigned payment for the given semester with a sufficient amount.

```

1 @Scheduled(cron = "0 0 * * * ?")
2 public void fetchPaymentsFromBankAPI() {
3     try {
4         paymentService.fetchPaymentsFromBankAPI();
5     } catch (JsonProcessingException e) {
6         System.out.println("ERROR Payments fetching failed");
7     }
8 }

```

Listing 4.2: Example of Spring scheduled method that runs payments fetching

Fetching of the transactions itself is then performed by sending appropriate HTTP request to the URL with bank API token inside. Bank API token with read-only rights was generated using the Fio bank internet banking.

```

1 public void fetchPaymentsFromBankAPI() throws JsonProcessingException {
2     RestTemplate restTemplate = new RestTemplate();
3     String url = "https://www.fio.cz/ib_api/rest/periods/" +
4         BANK_API_TOKEN + "/2021-07-01/2121-01-01/transactions.json";
5     ResponseEntity<String> response = restTemplate.getForEntity(url,
6         String.class);
7
8     ObjectMapper mapper = new ObjectMapper();
9     JsonNode root = mapper.readTree(response.getBody());
10    JsonNode transactions = root.path("accountStatement").path("
11        transactionList").path("transaction");
12
13    for (JsonNode transaction : transactions) {
14        Payment payment = parseTransactionNode(transaction);
15
16        if (!paymentRepository.existsByTransactionId(payment.
17            getTransactionId())) {
18            paymentRepository.save(payment);
19        }
20    }
21 }

```

Listing 4.3: Example of method that fetches transactions via Fio bank API

4.1.6 Spring security configuration

As it was described earlier, this project is based on a token-based authentication using JWT. All classes related to authentication are implemented in the `security` package. The core of Spring Security configuration is a `WebSecurityConfigurerAdapter` class that has been extended and customized in `WebSecurityConfig` class. It provides `HttpSecurity`

configurations to configure cors, csrf and rules for protected resources.

`UserDetailsService` interface has been implemented in `UserDetailsServiceImpl` class. It overrides a method which loads user by username and returns a `UserDetails` object that Spring Security can use for authentication and validation. `UserDetails` object implemented in `UserDetailsImpl` class contains all necessary information, such as username, password and authorities, to build an `Authentication` object.

`UsernamePasswordAuthenticationToken` gets username and password parameters from login request and `AuthenticationManager` will use it to authenticate a login account. `AuthenticationManager` object validates the `UsernamePasswordAuthenticationToken` object and if successfully, then it returns a fully populated `Authentication` object, including granted authorities. Based on this `Authentication` object, a new JWT token is generated by a specific builder method of `jjwt` library. The JWT token is generated using a HS512 signature algorithm with a preset secret and an expiration set to 86400000 ms. After JWT is generated, it is attached to the request response.

`OncePerRequestFilter` object provides a `doFilterInternal()` method that is executed every time a new request to API is performed. The method performs parsing and validation of JWT, loading of user details and validation of authorities. After all authorization constraints are met, then the appropriate request response is returned. All these Spring Security objects interactions have been visualized in the figure 3.2. [13], [15], [16]

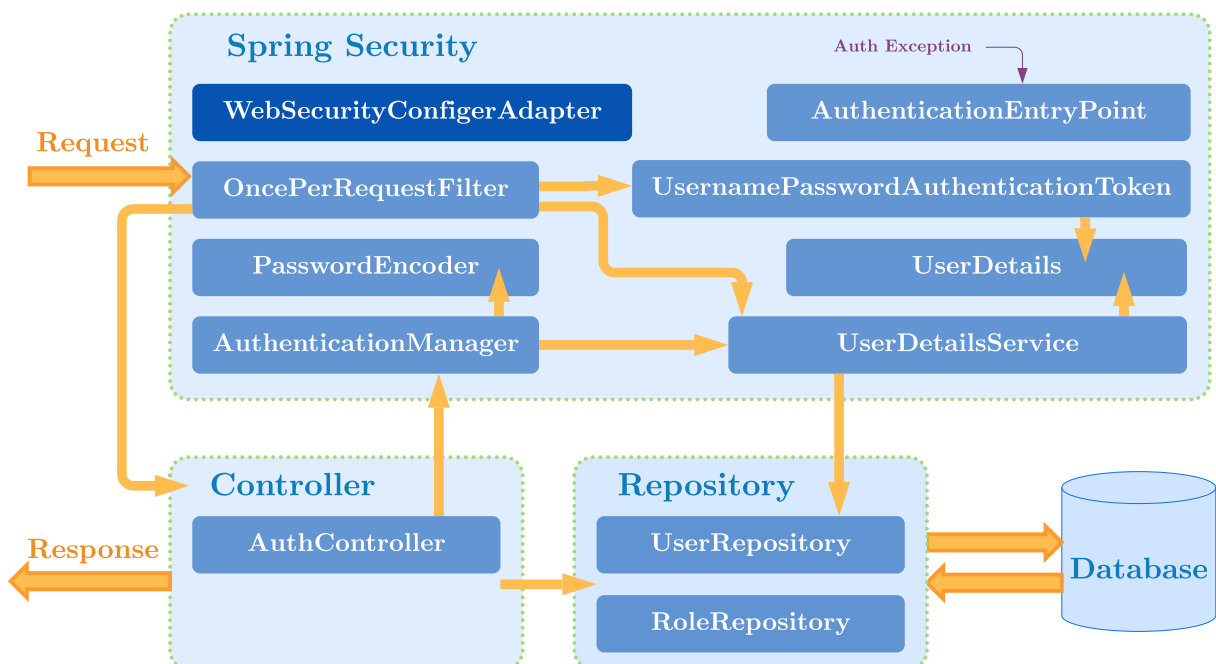
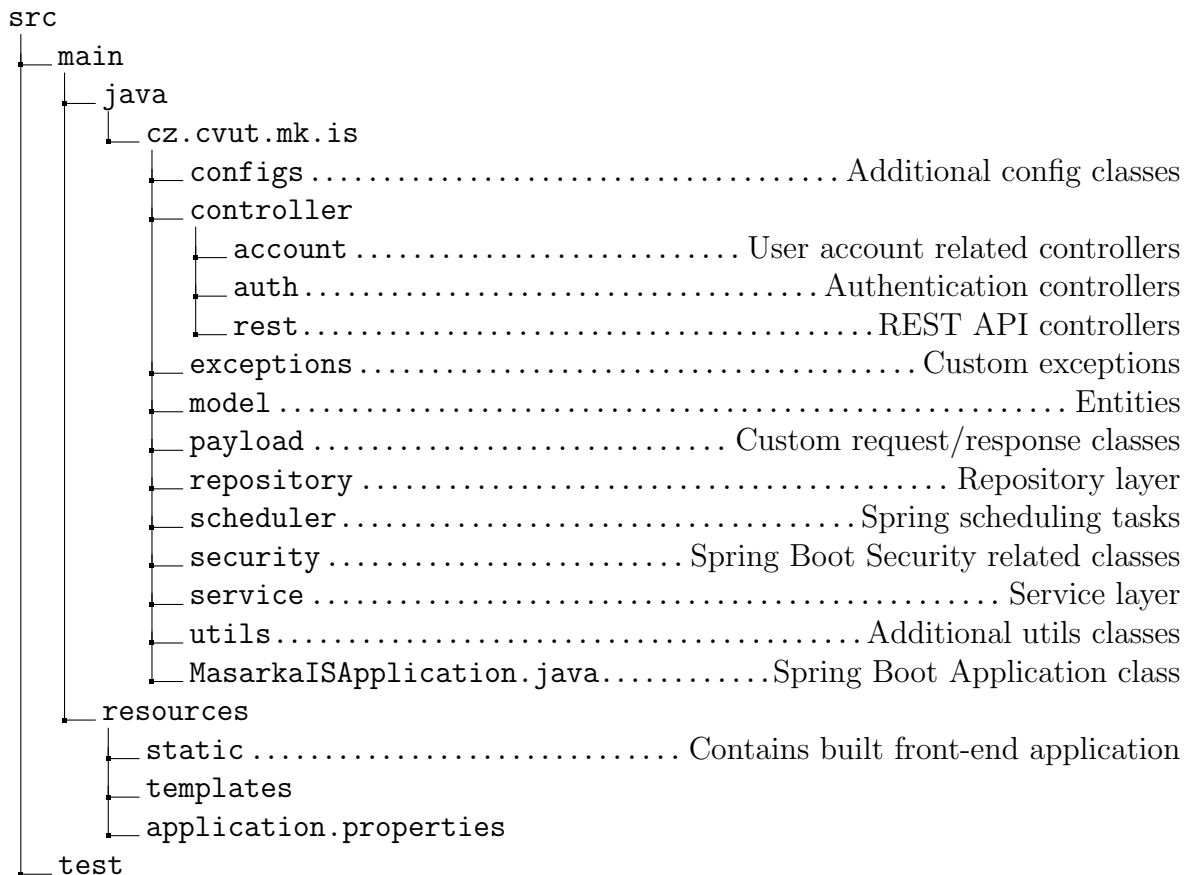


Figure 4.1: Spring Security objects interactions

4.1.7 Spring Boot project directory structure

All project classes are structured into the packages according to Spring Boot naming conventions from official documentation. The project contains four typical packages `model`, `repository`, `service` and `controller`, where the most of the server application is implemented. The following graph demonstrates a source code directory structure of the server application.



Application `controller` package contains three other packages, where the controllers are divided according to the provided functionality. Controllers of the `account` package provide only the functionality related to currently logged-in user. Controller from the `auth` package provides only the login endpoint. Controllers from the `rest` package provide RESTful API for all entities. The figure 4.2 demonstrates the back-end application API using the Swagger UI tool.

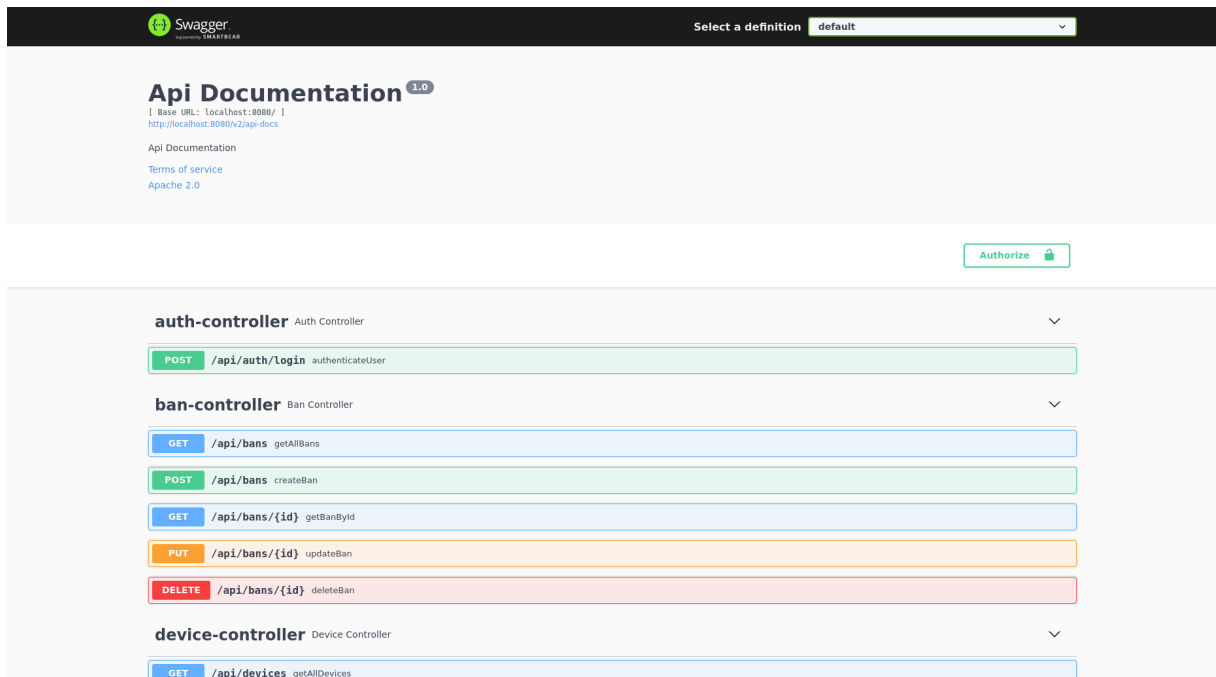


Figure 4.2: Example of REST API using the Swagger UI

4.2 Front-end application

4.2.1 Prerequisites

Development of JavaScript applications usually requires two prerequisites to be installed. The first is `Node.js` - an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside a web browser. The second is `npm` (Node Package Manager), which allows to install additional packages. In my case `Node.js v14.16.0` and `npm 7.10.0` versions have been used.

Additionally, `Vue CLI` component is required to initialize the `Vue.js` project. It can be easily installed by `npm` using the following command `npm install -g @vue/cli`. After installation is complete, there will become available new executable command `vue` in the terminal. In this project `Vue CLI` version `4.5.11` has been used.

4.2.2 `Vue.js` project initialization

At this moment it is assumed, that all prerequisites from the previous section are already installed. Now a new `Vue.js` project can be initialized using the following command `vue create project-name`, where the "project-name" should be replaced by actual name of the project. Alternatively, the `Vue.js` projects can be initialized and then managed by a very handy graphical tool, that can be launched by command `vue ui`.

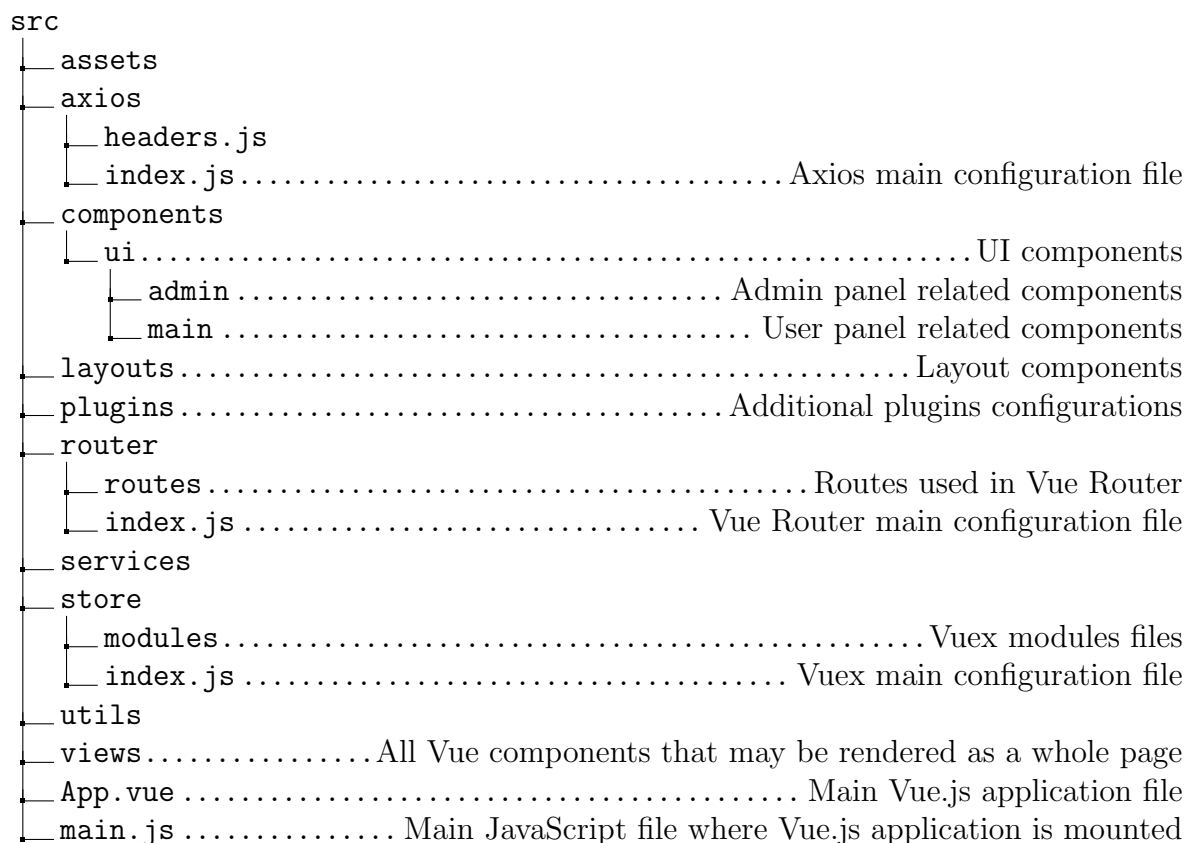
After running the `vue create` command, the Vue CLI tool will start an interactive command line dialog, where the default preset or manually configured settings can be selected. The manual mode is preferred, as in the next step it will allow to specify which settings and additional components should be used in the project. For example in my case additional components Vue Router and Vuex have been selected. Another important option to be selected is a version of Vue.js library to be used. In this project Vue.js version 2 has been used.

After the project initialization is complete, the rest of additional components should be installed. Additional components like Vuetify, Axios, VeeValidate can be easily installed using the npm tool.

After the all installations are complete, information about all dependencies will be reflected in the `package.json` file.

4.2.3 Front-end application directory structure

The following graph demonstrates a source code directory structure of the client application with a brief description of most important folders and files.



4.2.4 Axios configuration

In this application Axios component will be used for HTTP communication with server application. Its main configuration has been placed to the following file `axios/index.js`. First important thing that is going to be configured here is a base URL of the server application, so later only the relative paths can be used in whole application. And in the case the base URL will be changed for some reason, it will be enough to change it only in one place.

Another important things that can be set here are interceptor functions, that should be executed before or after sending HTTP requests. In this application there will be set an interceptor for processing HTTP response status. In any case the response status code equals 401 (Unauthorized), 403 (Forbidden) or 500 (Server error), the corresponding alert message will be displayed to user.

```
1 ...
2 const requestAxios = axios.create({
3   baseURL: process.env.VUE_APP_SERVER_URL
4 })
5
6 requestAxios.interceptors.response.use(null, error => {
7   // console.log(error)
8   if (error.response.status === 401) {
9     router.push('/login?alert=login')
10  }
11  if (error.response.status === 403) {
12    store.dispatch('alert/setAlert', {
13      message: 'Unauthorized - You don\'t have permission to perform
14              this action.',
15      type: 'warning'
16    })
17  }
18  if (error.response.status === 500) {
19    store.dispatch('alert/setAlert',
20      {
21        message: error.response.data.message ? error.response.data.
22                message : 'Unknown error',
23        type: error.response.data.type ? error.response.data.type : '
24              error'
25      })
26  }
27  return Promise.reject(error)
28 })
```

Listing 4.4: Example of `axios/index.js` file

4.2.5 Vue Router configuration

In this application Vue Router component has been used to implement page navigation between individual views. Additionally, Vue Router plays very important role in managing access to views based on the predefined authorization roles. All main Vue Router configurations can be found in the `router/index.js` file. This file basically contains initialization of new router object and definition of routes array, which are imported and concatenated from individual routes files in `routes` folder. Routes parameter is an array of objects, that keep an information about URL path and its corresponding view component to be rendered. Additionally every object keeps an information about the `auth` parameter, which determines whether the authentication is required to access this view and `roles` parameter which defines user roles that may access this view.

Additionally, in `router/index.js` there is implemented a router `beforeEach()` function, which acts like an interceptor that is executed each time when some view is accessed. It checks whether all conditions for the given view are met. If yes, the requested view is rendered. If no, the proper alert message is displayed or redirect performed.

4.2.6 Vuex store

In this application Vuex store plays a fundamental role in implementation of JWT authentication. After the login request is successfully sent and JWT token is received from the server application, the JWT token is then stored into the centralized Vuex store, which is accessible from any component of Vue.js application. Afterwards, before each request to server, the JWT token is retrieved from the store and attached to request headers. The following code snippet demonstrates a login function implemented in Vuex authentication module (`store/modules/auth.module.js`):

```
1  actions: {
2    async login ({ commit, dispatch }, payload) {
3      try {
4        const { data } = await axios.post('/auth/login', { ...payload })
5        commit('setUser', data)
6        commit('setToken', data.accessToken)
7        await router.push('/')
8      } catch (e) {
9        await store.dispatch('alert/setAlert', {
10         message: 'Wrong credentials.',
11         type: 'error'
12       })
13     }
14   }
15 },
```

Listing 4.5: Example of login function from Vuex auth module

4.2.7 UI development

In this section there will be briefly described the development of UI components of the front-end application. All UI components can be categorized into three groups. The layout components, smaller building components like header, sidebar, footer, and the last group are view components, which combine everything together and present page as a whole. According to the access rules, all views can be further divided into the views with public access, views of authenticated user and views of the admin panel.

Login view

Any unauthenticated user is automatically redirected to Login view (4.3).

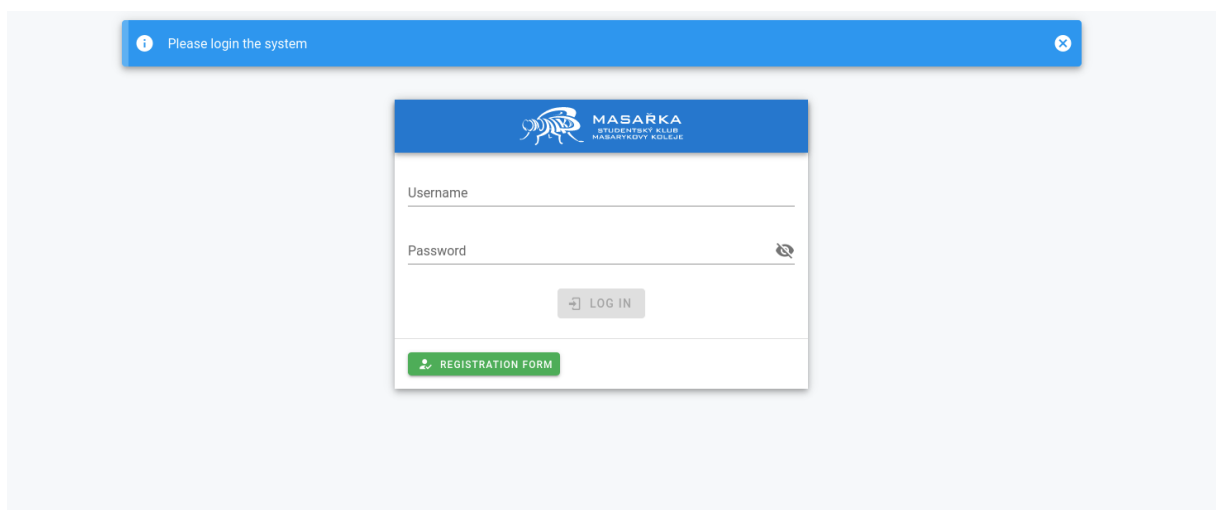
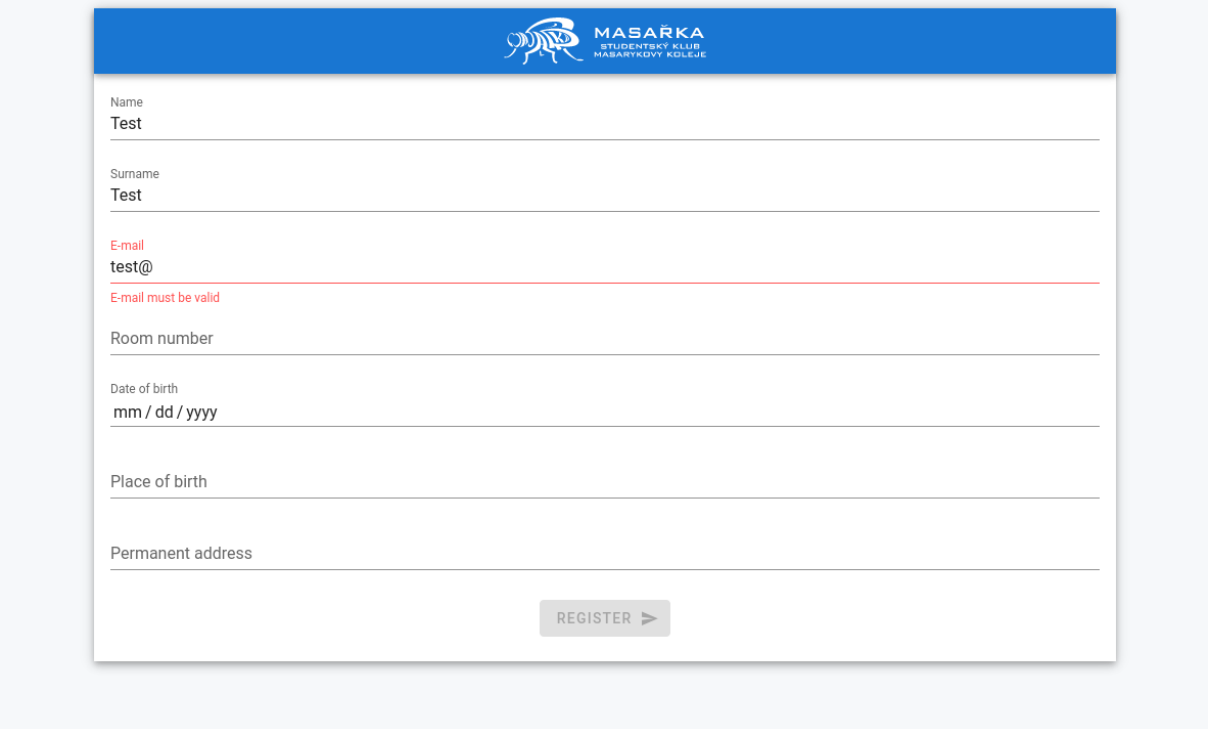


Figure 4.3: Login view

Registration view

Each field of the registration form is validated using VeeValidate component. Submit button is available only after all input fields are correctly filled (4.4).



The image shows a registration form with a blue header containing the MASARKA logo and text: MASARKA STUDENTSKÝ KLUB MASARYKOVY KOLEJE. The form fields are: Name (Test), Surname (Test), E-mail (test@), Room number, Date of birth (mm / dd / yyyy), Place of birth, and Permanent address. The E-mail field is highlighted in red with the error message "E-mail must be valid". A "REGISTER >" button is located at the bottom of the form.

Figure 4.4: Registration view

Profile view

After the successful login, user is automatically redirected to the Profile view (4.5), where can be observed all personal data filled during the registration. User can only change its email or update password. Other data can be changed only in admin panel by authorized users.

If the currently logged-in user is authorized to access the admin panel, then the corresponding "Admin panel" button in the top-right corner will be available (next to the Home button).

On the left side of the screen is a collapsible sidebar with menu navigation.

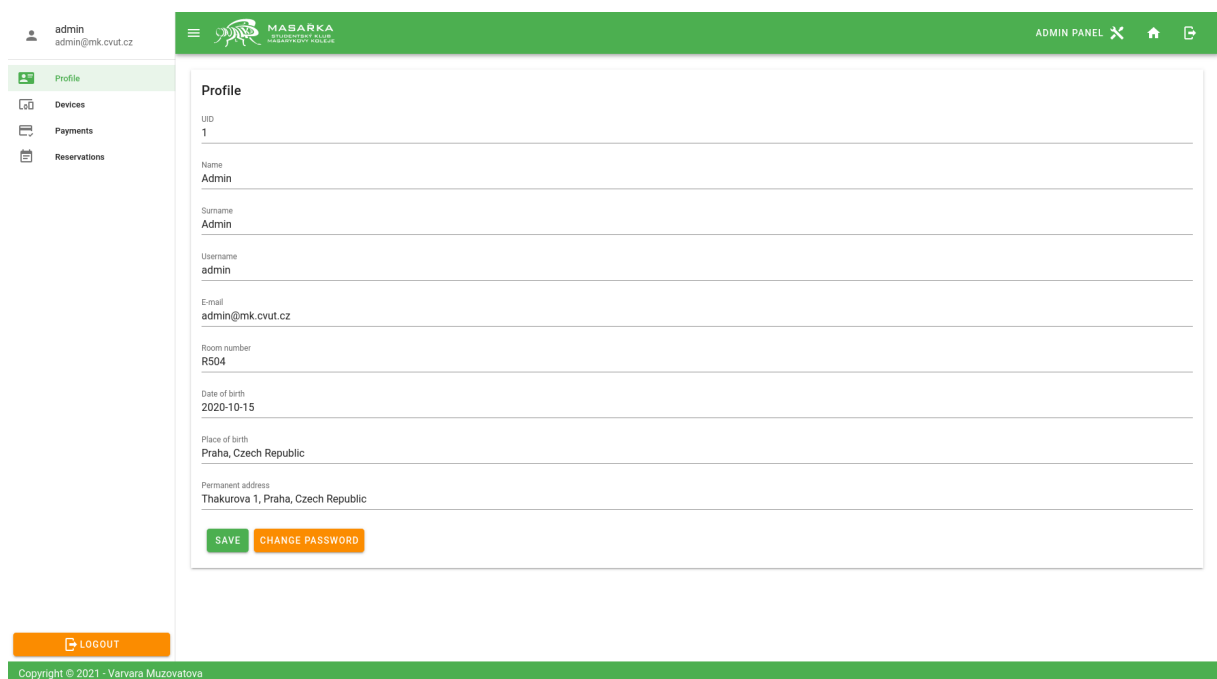


Figure 4.5: Profile view

Reservations view

The following figures demonstrate Reservation view, where the user can make a reservation for sport and entertainment rooms.

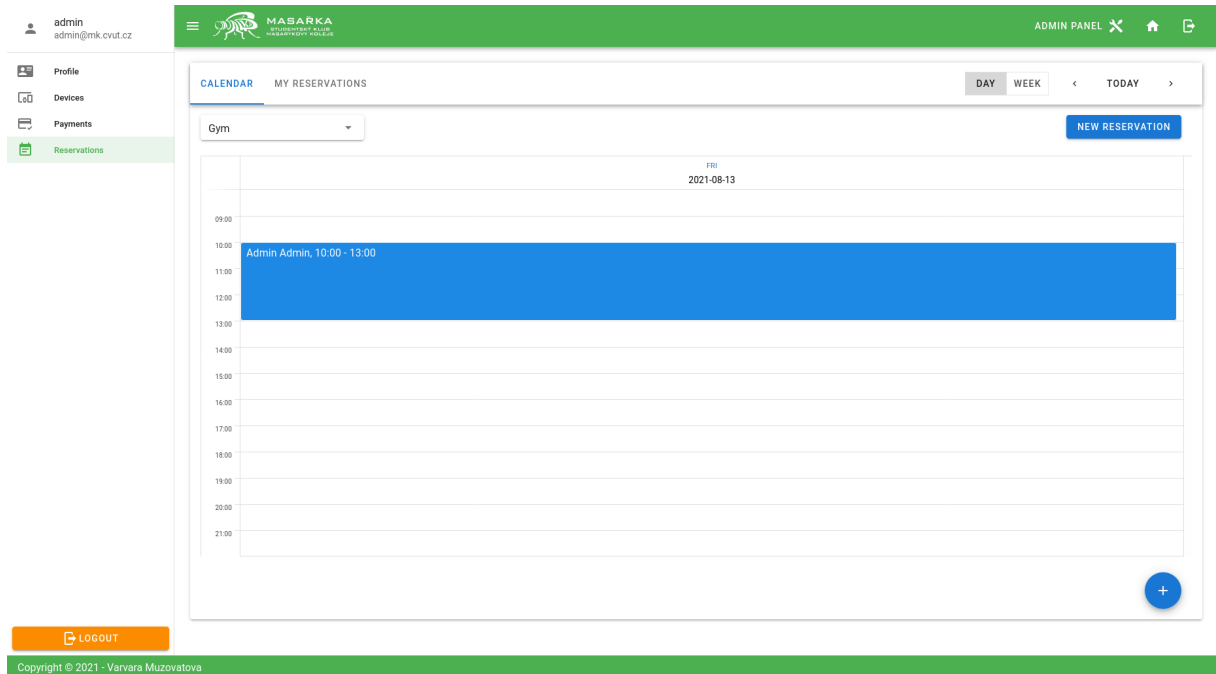


Figure 4.6: Reservations view

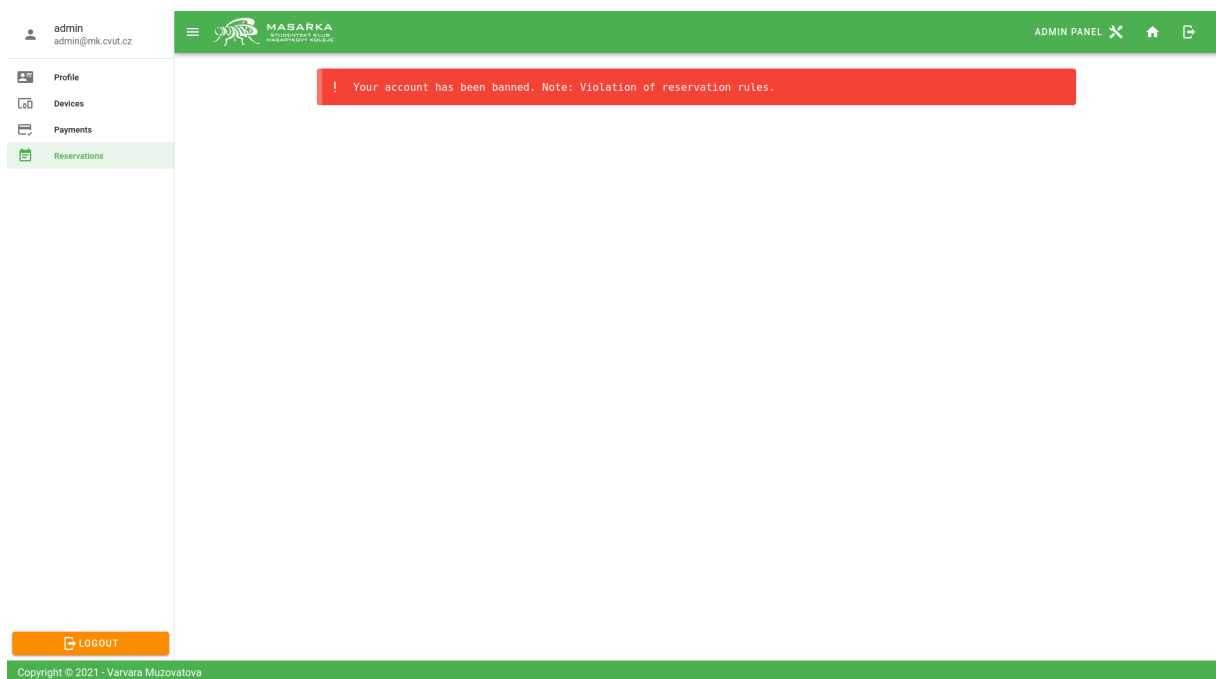


Figure 4.7: Banned reservations view

Admin panel - Users view

Figure 4.8 demonstrates admin panel for managing user records. It allows to add new user, edit or remove existing user, reset user's password. Data table is interactive and allows to search items by specified word.

UID	NAME	SURNAME	USERNAME	E-MAIL	DATE OF BIRTH	PLACE OF BIRTH	PERMANENT ADDRESS	REGISTRATION DATETIME	ROOM NUMBER	ACTIONS
1	Admin	Admin	admin	admin@mk.cvut.cz	2020-10-15	Praha, Czech Republic	Thakurova 1, Praha, Czech Republic	13.08.21 1:25	RS04	[Reset] [Edit] [Delete]
2	UserName	UserSurname	UserUsername	user@mk.cvut.cz	2020-10-15	Praha, Czech Republic	Thakurova 1, Praha, Czech Republic	13.08.21 1:25	RS05	[Reset] [Edit] [Delete]

Figure 4.8: Admin panel - Users view

Admin panel - User roles view

Figure 4.8 demonstrates admin panel for managing user roles. Currently, only the user with "chairman" role can assign or change users roles.

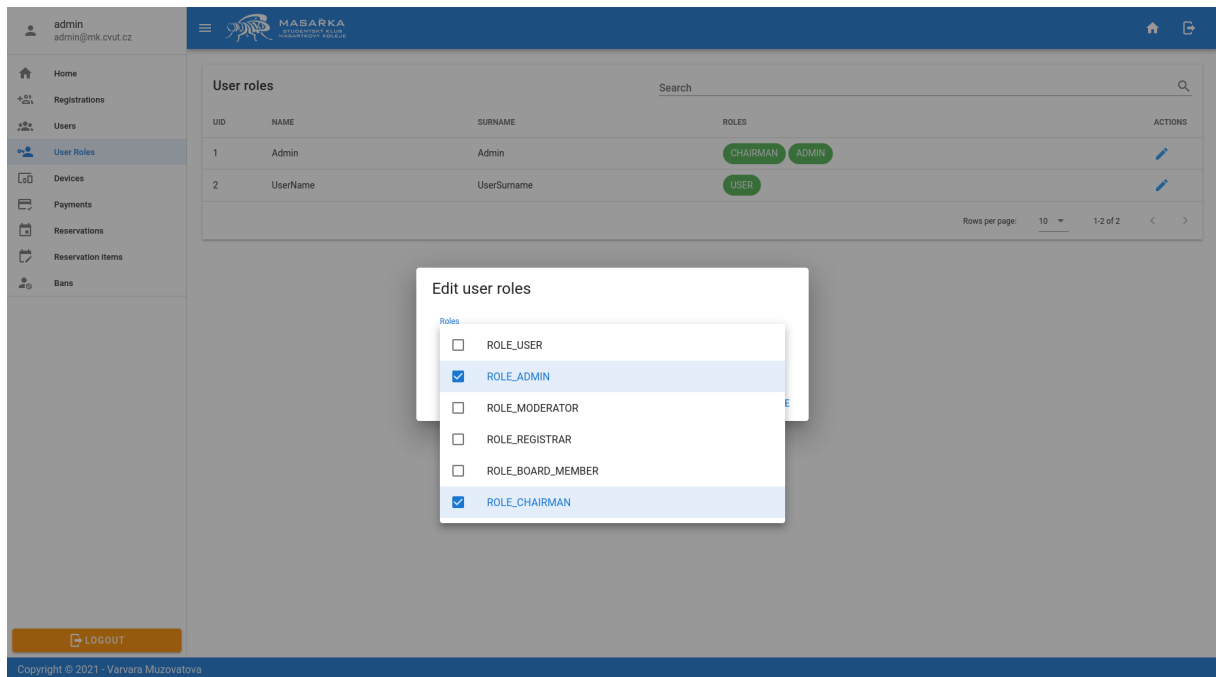


Figure 4.9: Admin panel - User roles view

4.3 Deployment

Both, the back-end and the front-end, applications were deployed as Docker containers on the production server of the Masarka club. All the communication with Docker containers is performed through the Traefik proxy server, which ensures three very important functionalities in this project:

- It assigns domain names to a corresponding containers
- It redirects all traffic from HTTP to HTTPS
- It ensures automatic generation of Let's Encrypt TLS certificates for HTTPS communication

The source codes of both, the back-end and front-end, applications are accessible in the following GitLab repositories:

- Back-end repository - <https://gitlab.fel.cvut.cz/muzovvar/masarka-is-backend-app>
- Front-end repository - <https://gitlab.fel.cvut.cz/muzovvar/masarka-is-frontend-app>

Building processes for both, back-end and front-end, applications are handled by CI/CD pipelines of GitLab. Used Dockerfile and .gitlab-ci.yml files can be observed in corresponding repositories.

The new information system was launched in October 2021 and since that moment the system has been successfully operated in the Masarka club for the entire period of the winter semester 2021/2022.

Production version of the information system can be accessed on the following URLs:

- Main application (front-end) - <https://is.mk.cvut.cz>
- API (back-end) - <https://api.is.mk.cvut.cz>

Chapter 5

Testing

Testing is an important part of any software development project. It allows to determine whether the application meets all function and non-functional user requirements. In this chapter there will be presented functional tests that were used to test the implemented system.

5.1 Functional testing

Functional tests are used to verify that the resulting application performs all the tasks for which it is meant to. In general, all functionality implemented in the application is tested and it is verified that this functionality works properly, meets user requirements and has all the required functionality specified in the customer's requirements list.

In this project application functionality has been tested by two sets of tests. One for testing the back-end application API and another set of integration tests for testing the functionality of the whole system.

5.1.1 Back-end API testing

During the development phase of the back-end application, a set of regression tests was created to help debug and verify the functionality of the back-end API. Postman application was used for these purposes. It is a quite popular tool for manual testing of the RESTful APIs, but it also allows to create collections of automatic tests, which then can be run at once. The set of tests may be run using the graphical user interface of Postman application as well as in the command line using the Newman tool.[17] The Postman application allows to save the whole collection of tests and export them to the one JSON configuration file.

The Postman application allows to test the request response status, data querying using GET method, data storing using POST method, data deletion using DELETE method.

Basically, it allows to cover all types of requests for RESTful API. For more complex solutions the Postman also allows to create environments and work with the variables across the various tests.[18] This functionality is very useful for handling the user authentication and authorization processes.

While developing RESTful API, I have created a set of regression tests using the Postman tool, what made the application development much easier. Created tests cover the functionality of CRUD methods for all project entities as well as the testing of other methods related to user account. All tests have been saved to one collection and exported to `MAS_IS_postman_tests.json` file. The file is available in the Gitlab repository of the back-end application. This set of tests can be very useful in case of any further extensions development.

All Postman tests can be accessed at the back-end project GitLab repository:

<https://gitlab.fel.cvut.cz/muzovvar/masarka-is-backend-app>

5.1.2 Selenium tests

Selenium is an open-source project that brings together a number of tools and libraries aimed to automate web browsers. It is mainly used for automated testing of web applications, but certainly it is not limited to just that.[19] It provides web drivers for all popular web browsers and supports wide range of programming languages including Python, Java, Ruby, Perl, PHP, JavaScript and others.[20]

In this project, selenium tests were implemented to test the functionality of the front-end application and system as a whole, therefore they act in the role of some kind of acceptance tests. The tests itself were implemented in Python using the selenium package and the Firefox web driver.[21]

In following section are described possible passages through the application. For detailed analysis have been chosen all possible patterns of the user with the chairman role, which has all options of the user role and at the same time the possibilities of all other available roles.

Branch points:

- Start
- LP - Login panel
- PREG - Show pre-registration form
- FP - Send link to reset password

- SPP - Show profile panel
- SNER- Save new email or room
- CP - Change password
- SUP- Show user payments
- SUD - Show user devices
- SUR - Show user reservations
- ND - Add new device
- ED - Edit device
- DD - Delete device
- CHR - Choose room
- SMR - Show my reservations
- SRR- Show reservations of the room
- NR - Save new reservation
- DMR - Delete my reservations
- SAP - Show admin panel
- SREG - Show registrations panel
- SNU - Submit new user
- DREG - Delete registration
- SU - Show all users
- RES - Reset password
- SGP - Show generated pdf
- EDU - Edit user
- DELU - Delete user
- SURL - Show users roles
- CURL - Change user roles
- SD - Show all devices

- ND - Save new device
- EDD - Edit device
- DELD - Delete device
- SP - Show all payments
- EDP - Edit payment
- DELP - Delete payment
- SR - Show all reservations
- EDR - Edit reservation
- DELR - Delete reservation
- SRI - Show reservation items
- NRI - Save new reservation item
- ERI - Edit reservation item
- DRI - Delete reservation item
- BANS - Show all bans
- NB - Save new ban
- EDB - Edit ban
- DELB - Delete ban
- End

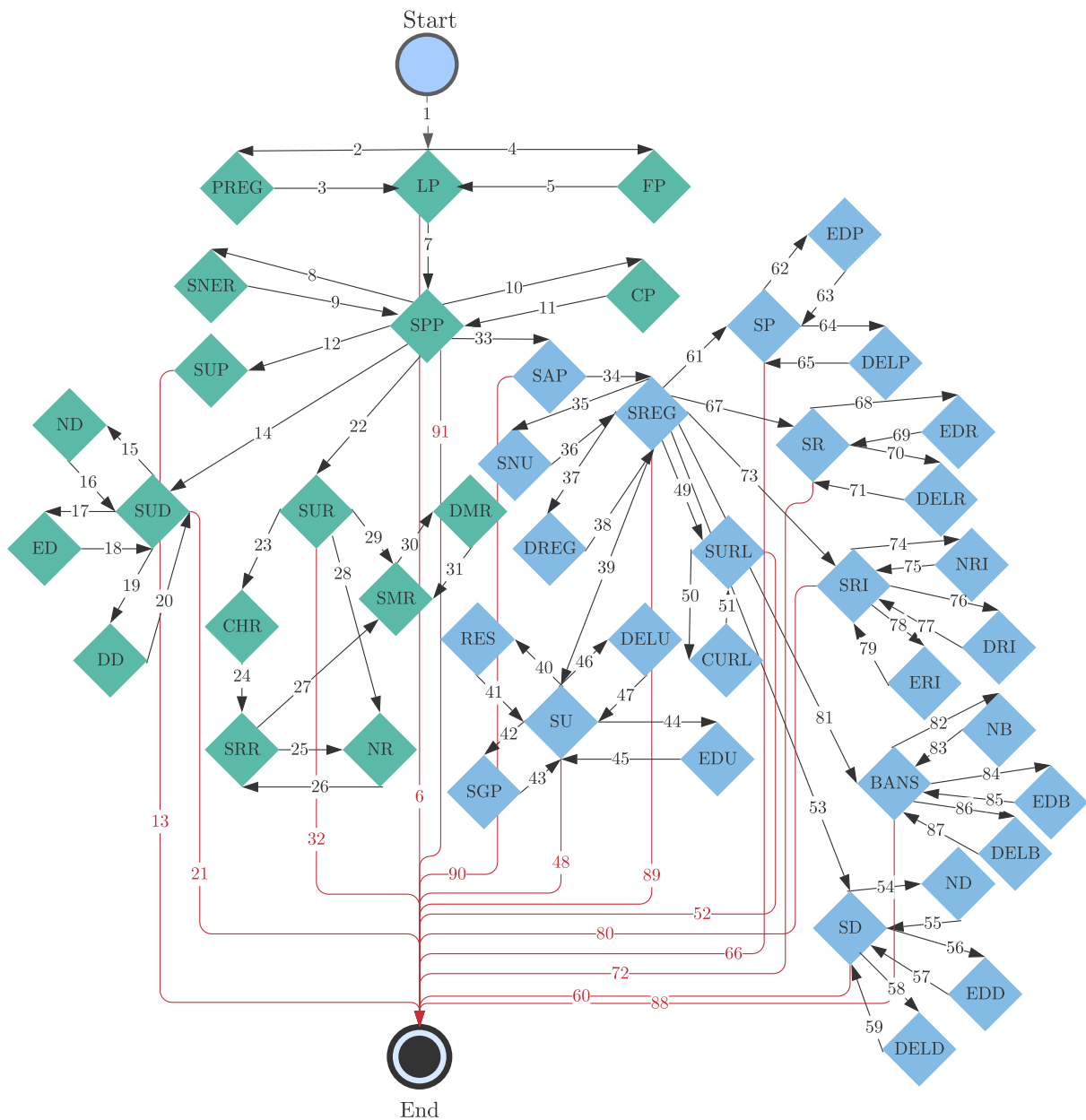


Figure 5.1: Graph for compiling possible testing scenarios

Implemented selenium tests cover all presented combinations.

All selenium tests can be accessed at the front-end project GitLab repository:

<https://gitlab.fel.cvut.cz/muzovvar/masarka-is-frontend-app>

Test results

Functional tests helped to reveal a series of bugs related mainly to the functionality of the reservation system. For example, users could make reservations with the wrong time ranges or make parallel reservations for the same item, which should not be available. All detected mistakes have been fixed.

5.1.3 User testing

In parallel with the automated functional testing, user testing was also being performed with the help of the club administrators team. User testing covered approximately the same testing scenarios that have been presented with selenium tests.

Chapter 6

Conclusion

The main aim of this project was to create a new information system for the administration of members and services of the Masarka student club, which would fully replace the former system and at the same time eliminate its shortcomings, as the former system was very outdated and no longer satisfied all the needs of the club.

In the first steps an analysis of the former information system solution was performed, the result of which was a summary of all available functionality and its shortcomings. During this analysis, major shortcomings were revealed, which mainly concerned the security issues, the absence of user authorization roles, inability of the user to register its devices by himself and the absence of one centralized system for managing all club services including user reservations.

In the next step, based on the results of the analysis of the former solution, the comprehensive functional and non-functional requirements for the new information system were designed and presented. With regard to these requirements, an analysis of available existing solutions was performed, oriented primarily on the open-source projects. The research of available technologies revealed that neither of the solutions covered all the requirements of the club, and their eventual exploitation would require some kind of further adaptation to make them suitable for the club needs. Based on the results of research, I decided that the implementation of a new information system is much more reasonable.

The choice of back-end technologies to be used in the implementation was mainly affected by my personal experiences and knowledge acquired during the study on CTU. Therefore, Java programming language and Spring Boot framework were preferred. But in the case of front-end technologies the choice was not so obvious, as I previously had almost no experience with front-end development. In the next step I performed a brief research of the most popular JavaScript front-end technologies and decided to choose Vue.js as the main framework for the further front-end development.

Based on the performed analysis and specified requirements a new information system has been designed and implemented, using the selected technologies. After the implementation was completed, the new information system was deployed on the production server of the Masarka club. Afterwards, it was necessary to test the functionality of the complete system before it could be launched for public exploitation by students. A set of automatic tests and user testing by a team of club administrators was performed. Few iterations of tests helped to reveal some bugs, which were then gradually fixed and the system was finally launched for public access by club members.

The new system was launched in October 2021 and since that moment the system has been successfully operated in the Masarka club. For the entire period of operation in the winter semester 2021/2022 approximately 250 new users have been registered, more than 650 user devices have been added and more than 1500 reservations have been performed by club members. Currently, since the moment the system was launched for public access, no significant errors or shortcomings were detected.

6.1 Evaluation of the system from the perspective of end users

From the point of view of the regular user as well as the privileged user (user with at least one additional user role), the main advantages of the new system can be summarized as:

- An intuitive and responsive design, that works fine on mobile and desktop devices.
- One common system for managing all club services at one place.
- Ability of regular users to add their own devices without necessity to contact administrators.
- Fast and regular synchronization of payments from bank account.
- Intuitive pre-registration form with fields validation, that makes the process of users registration faster.

Especially, the ability of regular users to add their own devices without the necessity to contact the club admins, was very positively evaluated by club members. Basically, it brings simplicity for regular users and less work for club administrators.

6.2 Ideas for further improvements

There are many ideas for further extensions, but the most significant are the following:

6.2.1 Integration of payment gateway

Currently, the information system can only fetch transactions from the bank account and pair them with the corresponding user, but it doesn't provide users any options on how to pay the membership fee online. It would be great to implement this feature, integrate the information system with some payment gateway and enable users to pay their membership fees directly in our system.

6.2.2 Users activity logging

Currently, the information system doesn't provide any sophisticated logging of users activity. It would be good to implement it, as it can be very useful while monitoring some suspicious activity. For example, it can be useful for monitoring the frequent switching of user devices, which can lead to a suspicion of an internet connection sharing.

6.2.3 Language localisation

Currently, the information system provides only the English language localisation, as the most students at Masaryk dormitory are international students. But in general, it would be good to adapt the application and implement an ability to simply add other language localisations.

Bibliography

- [1] *Historie: Studentská unie čvut.* [Online]. Available: <https://old.su.cvut.cz/cs/historie>.
- [2] M. T. Thomas, *React in action*. Manning Publications, 2018, ISBN: 978-1617293856.
- [3] D. Karczewski, *What are the best frontend frameworks to use in 2021?* [Online]. Available: <https://www.ideamotive.co/blog/best-frontend-frameworks>.
- [4] A. Freeman, *Pro Angular 9 build powerful and dynamic web apps*. Apress, 2020, ISBN: 978-1484259979.
- [5] E. Hanchett, *Vue. js in action*. Manning Publications Company, 2018, ISBN: 978-1617294624.
- [6] *Best frontend frameworks of 2021 for web development*, 2021. [Online]. Available: <https://www.simform.com/best-frontend-frameworks/>.
- [7] Baeldung, *A comparison between spring and spring boot*, 2021. [Online]. Available: <https://www.baeldung.com/spring-vs-spring-boot>.
- [8] *Why you should be using vuetify.* [Online]. Available: <https://vuetifyjs.com/en/introduction/why-vuetify/>.
- [9] *Vue router.* [Online]. Available: <https://router.vuejs.org/>.
- [10] *What is vuex?* [Online]. Available: <https://vuex.vuejs.org/>.
- [11] *Vee-validate overview.* [Online]. Available: <https://vee-validate.logaretm.com/v4/guide/overview>.
- [12] Y. Balaj, “Token-based vs session-based authentication: A survey”, Sep. 2017.
- [13] A. Anand, *Spring boot api security with jwt and role-based authorization*, 2020. [Online]. Available: <https://medium.com/@akhileshanand/spring-boot-api-security-with-jwt-and-role-based-authorization-fe1fd7c9e32>.
- [14] *Scheduling Tasks.* [Online]. Available: <https://spring.io/guides/gs/scheduling-tasks/>.
- [15] J. Carnell, *Spring microservices in action*. Manning Publications Company, 2017.
- [16] C. Walls, *Spring Boot in action*. Manning, 2016.
- [17] Postman, *Running collections on the command line with newman.* [Online]. Available: <https://learning.postman.com/docs/running-collections/using-newman-cli/command-line-integration-with-newman/>.
- [18] Postman2, *Postman writing tests.* [Online]. Available: <https://learning.postman.com/docs/writing-scripts/test-scripts/>.

- [19] *The Selenium Browser Automation Project*. [Online]. Available: <https://www.selenium.dev/documentation/>.
- [20] S. Raghavendra, *Python Testing with Selenium: Learn to Implement Different Testing Techniques Using the Selenium WebDriver*, 1st ed. Apress, 2020.
- [21] *Selenium with Python — Selenium Python Bindings 2 documentation*. [Online]. Available: <https://selenium-python.readthedocs.io/index.html>.