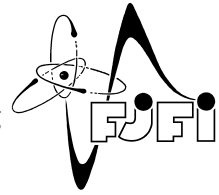




CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Nuclear Sciences and Physical Engineering



Web application for team work organization

Webová aplikace pro organizaci týmové práce

Bachelor's Degree Project

Author: **Kirill Borodinskiy**
Supervisor: **doc. Ing. Miroslav Virius, CSc.**
Academic year: 2024/2025

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Borodinskiy** Jméno: **Kirill** Osobní číslo: **518594**
Fakulta/ústav: **Fakulta jaderná a fyzikálně inženýrská**
Zadávající katedra/ústav: **Katedra matematiky**
Studijní program: **Aplikovaná informatika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Webová aplikace pro organizaci týmové práce

Název bakalářské práce anglicky:

Web application for team work organization

Pokyny pro vypracování:

1. Seznamte se s knihovnamy pro tvorbu webových aplikací založenými na programovacím jazyce Java.
2. Sestavte seznam požadavků na aplikaci pro organizaci týmové práce.
3. Na základě analýzy těchto požadavků navrhnete aplikaci.
4. Navrženou aplikaci implementujte a otestujte.

Seznam doporučené literatury:

- [1] Nick Williams: Professional Java for Web Applications. John Wiley & Sons 2014. ISBN 9781118656464
- [2] David A. Chappell, Tyler Jewell, Michael Wooten: Java Web Services. O'Reilly Media, 2002.
- [3] Rod Johnson, Juergen Hoeller, Alef Arendsen, Thomas Risberg, Colin Sampaleanu: Professional Java Development with the Spring Framework. Wrox, 2005.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

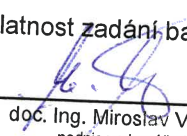
doc. Ing. Miroslav Virius, CSc. katedra softwarového inženýrství FJFI

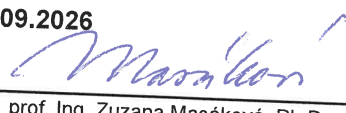
Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:


Datum zadání bakalářské práce: **25.10.2024**

Termín odevzdání bakalářské práce: **04.08.2025**

Platnost zadání bakalářské práce: **30.09.2026**


doc. Ing. Miroslav Virius, CSc.
podpis vedoucí(ho) práce


prof. Ing. Zuzana Masáková, Ph.D.
podpis vedoucí(ho) ústavu/katedry


doc. Ing. Václav Čuba, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen 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.

31. 10. 2024

Datum převzetí zadání

Podpis studenta

Acknowledgment:

I would like to thank my mother Elena and my girlfriend Anastasiia for their moral support. I would like to thank my supervisor, Miroslav Virius, for their help in organization of my bachelor's thesis project.

Author's declaration:

I declare that this Bachelor's Degree Project is entirely my own work and I have listed all the used sources in the bibliography. AI tools were used in full accordance with the guidelines established by CTU in Prague.

Prague, August 4, 2025

Kirill Borodinskiy

Název práce:

Webová aplikace pro organizaci týmové práce

Autor: Kirill Borodinskiy

Studijní program: Celý název studijního programu (nikoliv zkratka)

Specializace: Celý název specializace (Pokud se studijní program nedílí na specializace, tuto ádku odstranit.)

Druh práce: Bakalářská práce

Vedoucí práce: doc. Ing. Miroslav Virius, CSc., DrSc., pracoviť kolitele (název instituce, fakulty, katedry...)

Abstrakt: Abstrakt max. na 10 ádk.

Klíová slova: klíová slova (nebo výrazy) seazená podle abecedy a oddlená árkou

Title:

Title of the Work

Author: Kirill Borodinskiy

Abstract: Max. 10 lines of English abstract text.

Key words: keywords in alphabetical order separated by commas

Contents

1	Introduction	6
	Motivation	6
1.1	The current solution	6
2	Methods	8
2.1	Defining the toolbelt	8
2.2	Introduction to Spring Boot	9
2.3	Schema of the database	9
2.4	The implementation of a Spring Boot application	10
2.4.1	Useful tools	10
2.4.2	The architecture of the application	11
	Conclusion	12

Chapter 1

Introduction

Motivation

Efficient team schedule planning is a complex challenge, particularly in organizations that require real-time coordination and resource management. Existing scheduling services often have significant limitations, such as proprietary nature, lack of customization, and dependence on third-party infrastructure. This project aims to develop a self-hosted open-source booking system designed for organizations that need a private, adaptable scheduling solution. The system will provide a web-based interface where users can:

- Make and manage reservations
- Check real-time room availability
- Filter bookings by person or room
- View all reservations on a centralized calendar

The backend will be built using the Spring Framework, ensuring scalability, security, and ease of integration with existing infrastructure. Unlike cloud-based alternatives, this system will store all data locally, giving organizations complete privacy and control over scheduling information. By combining flexibility, transparency, and data privacy, this project can provide a practical alternative to commercial scheduling tools, empowering organizations with greater autonomy and customization options.

Here we will talk about why the calendar is needed

Firstly, let's take a look at the current solution used by my university, CTU. rozvrh.fjfi.cvut.cz is a website where students can see their schedule.

1.1 The current solution

It can be seen on 1.1 that while the website completes its main purpose, it is not customizable, which makes it hard to use. For example, if a student has a class that is from another year or/and program, they have to look at another picture and manually compare them. From my experience, many students have screenshots on their phones and they cross-out the classes that they are not registered to. They may have a few screenshots, for different programs or years. It is not a good solution, as it allows for misunderstandings and mistakes.

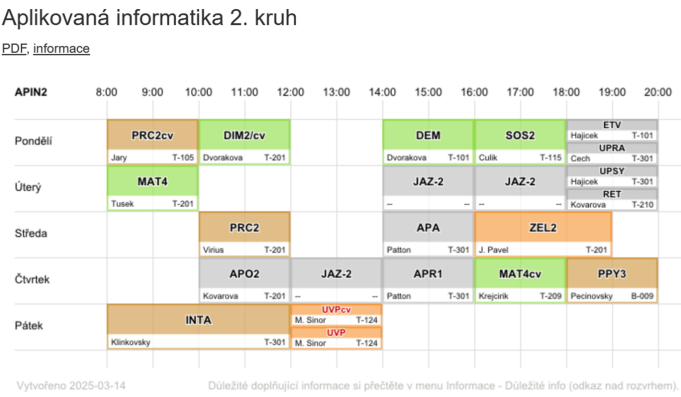


Figure 1.1: Screenshot of the current solution

Chapter 2

Methods

This section will focus on the tools and techniques used to create a web application to manage time and space resources of an organization.

2.1 Defining the toolbelt

For the programming language, the options were as follows: Java Spring Boot framework, C#.NET, and Python's Django and Flask. C# was first removed, as learning about it would take a considerable amount of time. Python's Flask is easy to use, but not customizable enough. Subsequent analysis, as presented in the study [ShreyashVardhan2024], indicates that Spring Boot exhibits superior performance. Moreover, my familiarity with Java might reduce the time required for development. Consequently, Spring Boot is selected as the back-end framework. Spring Boot, which is constructed on the Spring Framework, is recognized as a leading framework within the Java ecosystem due to its widespread popularity. It streamlines the original Spring Framework, thereby facilitating more straightforward maintenance and expediting deployment procedures. Henceforth, to maintain clarity, the term Spring Boot shall be used exclusively in reference to both the Spring Framework and Spring Boot.

An often-utilized integration of back-end and front-end frameworks with Spring Boot is accomplished through Thymeleaf, a template rendering engine which processes page rendering on the server side, thereby reducing computational demand on the client-side systems. However, alternative solutions are available, including the currently prevalent React framework along with other JavaScript frameworks. Opting for these alternatives requires considerable investment in research. Given my proficiency in HTML, the Thymeleaf template system presents a straightforward learning curve. Nevertheless, a certain degree of JavaScript is essential for contemporary websites, thereby necessitating its use for handling specific tasks such as requests.

For our database, PostgreSQL was chosen, as it is a popular ACID-compliant database. ACID stands for:

- **Atomicity:** Transaction is either fully completed, or not, with no in-betweens.
- **Consistency:** Guarantees that a transaction brings the database from a valid state to a valid state.
- **Isolation:** Concurrent transactions do not interfere with each other.
- **Durability:** Once a transaction is committed, it stays committed.

2.2 Introduction to Spring Boot

Spring Boot is a tool that allows the programmer to create a web server that uses the Model-View-Controller pattern, MVC for short.

The model is a part responsible for the data logic. The connection to the database, the processing of the requested data and other back-end transactions are what this part consists of.

The view is a part that displays the data to a user or gathers them from them. Whether HTML, plain text, or any other format such as our Thymeleaf.

The controller is a connector between the previous two, where the data is additionally processed before being sent into either the database or a client of a user.

2.3 Schema of the database

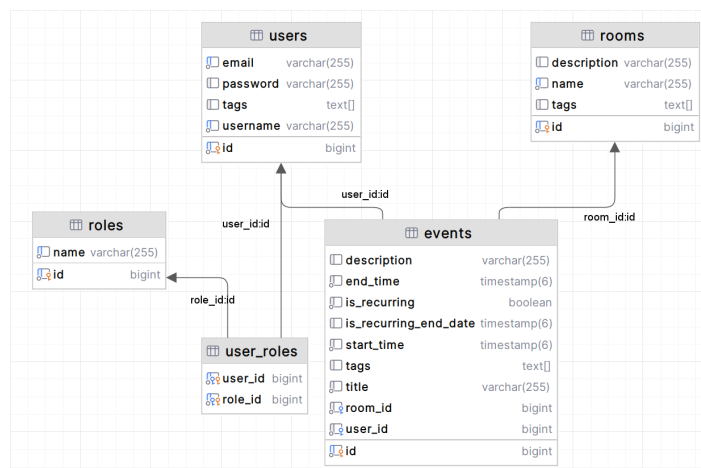


Figure 2.1: Schema of the database for the application

Database 2.1 Architecture

- **users**: Stores user credentials and personal data. Each user has a unique ID.
- **roles** & **user_roles**: Implements role-based access control via a many-to-many relation between users and roles.
- **rooms**: Contains information on event locations. Each room has a `name`, `description` and a unique ID.
- **events**: Represents scheduled activities. Includes data such as `is_recurring`, `title`, `start_time` and `end_time`. Each event references a `user` and a `room` that were assigned to this event.
- **users**, **events** and **rooms** all contain `tags` to help categorize them.

2.4 The implementation of a Spring Boot application

In our application, the main controller is `CalendarController`. It renders the main page `/calendar`. The non-required parameters of a request are: “`date`”, “`roomIds`” and “`userIds`”.

Parameter “`date`” is simply the day of the week the calendar renders. If not provided, we use the default value of a current day.

Parameters “`roomIds`” and “`userIds`” are used to filter out which events the user wants to see in their calendar. If not provided, events from every room and every user are displayed.

In this controller a week around the current day is generated, all the events that are happening in that week are found and are put into an array that is then sent with the model.

2.4.1 Useful tools

In this application, *Lombok* was used to reduce the amount of boilerplate code. It allows for the usage of annotations such as `@Getter`, `@Setter` to automatically generate getters and setters for all fields that need them. In addition, annotations `@AllArgsConstructor`, `@NoArgsConstructor` can automatically create the correct construction function for the class. Finally, `@Data` combines `@Getter`, `@Setter` and some more functions in one annotation, so the classes remain clean and functional.

The use of the tool is demonstrated in the list 1. It can be seen that no setter or getter functions are needed, no construction function is needed, and the class looks clean and complete.

```
@Data
@Entity
@Table(name = "rooms")
public class Room {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, unique = true)
    private String name;

    @Column()
    private String description;

    @Type(io.hypersistence.utils.hibernate.type.array.ListArrayType.class)
    @Column(name = "tags", columnDefinition = "text[]")
    private Set<String> tags = new HashSet<>();
}
```

Listing 1: Lombok-annotated JPA entity

2.4.2 The architecture of the application

The main file responsible for most tasks is `CalendarController`, where the `/calendar` endpoint is located.

The `/calendar` takes 3 non-required parameters.

- **date**: Show the events that happen on the week of the day sent.
- **userIds**: Filter the users shown on the central calendar.
- **roomIds**: Filter rooms displayed in the central calendar.

The `/calendar` then sends these data to `calendar.html`:

- **userIds** & **roomIds** (if supplied from request).
- **selectedDate** (if not provided, current date is used).
- **nextWeek** & **previousWeek** to enable navigation within weeks.
- **currentWeekStart**.
- **weekDays** - a list of events that take place on a specific day.
- **eventRepository**, **userRepository** and **roomRepository**

Conclusion

Text of the conclusion...

Bibliography

- [1] S. Allen, J. W. Cahn: *A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening*. Acta Metall., 27:1084-1095, 1979.
- [2] G. Ballabio et al.: *High Performance Systems User Guide*. High Performance Systems Department, CINECA, Bologna, 2005. www.cineca.it
- [3] J. Becker, T. Preusser, M. Rumpf: *PDE methods in flow simulation post processing*. Computing and Visualization in Science, 3(3):159-167, 2000.