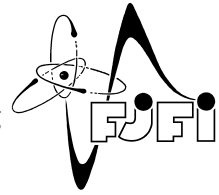




CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Nuclear Sciences and Physical Engineering



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**
Zadávací 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 knihovnami 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:

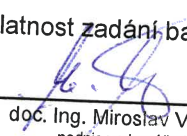
doc. Ing. Miroslav Virius, CSc. katedra softwarového inženýrství FJFI

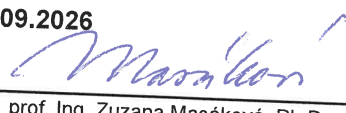
Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:


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	7
	Motivation	7
1.1	The current solution	8
1.2	Thesis Aims and Objectives	9
1.2.1	Primary Objectives	9
1.2.2	Expected Outcomes	10
2	Tools	11
2.1	Tool Selection and Technology Stack	11
2.1.1	Framework Selection	11
2.1.2	Frontend Technology Selection	12
2.1.3	Database Selection	12
2.1.4	Security Architecture Decisions	12
2.1.5	Development Tool Selection	12
2.1.6	Containerization Strategy	13
2.1.7	Technology Stack Integration	13
3	Methods	14
3.1	Requirements Specification	14
3.1.1	Functional Requirements	14
3.1.2	Non-Functional Requirements	14
3.2	System Architecture	15
3.2.1	Overall Architecture	15
3.2.2	Key Components	15
3.2.3	API Design	15
3.3	Database Design	16
3.3.1	Database Schema	16
3.3.2	Key Tables and Relationships	16
3.3.3	Spring Data JPA Implementation	17
3.4	User Interface Design	17
3.4.1	Design Philosophy	17
3.4.2	Key Interface Components	17
3.5	Implementation Details	18
3.5.1	Backend Implementation	18
3.5.2	Frontend Implementation	18
3.6	Development Tools and Environment	18
3.6.1	Development Environment	18

3.6.2	Testing Framework	18
3.6.3	Development Workflow	18
Conclusion		19

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 highlights the need for tailored solutions that can address specific organizational requirements more effectively. 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.

1.1 The current solution

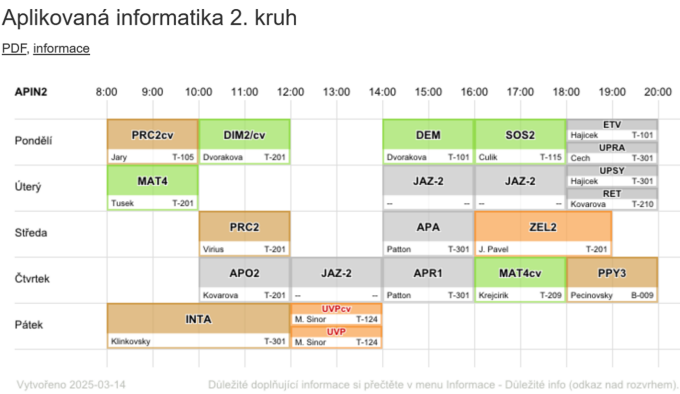


Figure 1.1: Screenshot of the current solution

Initially, it is pertinent to examine the existing solution employed by my institution, CTU. The website Rozvrh.fjfi.cvut.cz serves as a platform where students can view their academic schedules.

It can be seen in 1.1 that while the website completes its main purpose, it is not customizable, making it difficult to use. The current implementation presents several significant limitations that impact the user experience and efficiency.

Limited Customization: The system lacks personalization features, forcing all users to view the same static schedule format regardless of their specific needs or preferences.

Cross-Program Scheduling Complexity: Students enrolled in classes over different years or programs face particular challenges. They must manually navigate between multiple schedule views and compare information from different sources. This process is not only time-consuming, but is also prone to errors.

Manual Workarounds: The limitations of the current system have led to widespread adoption of inefficient workarounds. Many students resort to taking screenshots of their schedules and manually marking or crossing out irrelevant classes. This practice, while common, introduces several problems: Increased risk of scheduling conflicts due to manual tracking, difficulty in keeping track of schedule updates, time wasted on manual schedule management, potential for missed classes due to human error.

Integration Challenges: The system operates in isolation, making it difficult to: sync with personal calendars, share schedules with other students, and export schedule data for other purposes.

These limitations highlight the urgent need for a more sophisticated scheduling solution. A modern calendar system should address these issues by providing:

Personalization: Allow users to customize their view based on their specific needs and preferences.

Cross-Program Integration: Enable seamless viewing of classes across different programs and years.

Digital-First Approach: Eliminate the need for manual workarounds by providing a comprehensive digital solution

Integration Capabilities: Allow for easy sharing and synchronization with other tools and platforms.

This project aims to develop such a solution, focusing on creating a calendar system that is not only functional, but also user-friendly and customizable.

1.2 Thesis Aims and Objectives

The primary objective of this thesis is to explore and evaluate Java-based web development libraries while designing and implementing a practical team-work organization application. This project aims to demonstrate the application of modern Java web technologies in creating a functional and user-friendly system for team collaboration.

1.2.1 Primary Objectives

The development process is guided by several key objectives that form the foundation of this project. These objectives are designed to ensure a systematic and comprehensive approach to the development of the team-work organization system.

The first objective focuses on the *exploration and evaluation of technology*. In this phase, we will conduct comprehensive research into available Java web development libraries and frameworks. This exploration will encompass various aspects including web application development, database integration, user interface implementation, and security measures. Based on this research, we will carefully select and document the most appropriate technologies for the project's specific requirements.

The second objective involves a thorough *requirement analysis*. This phase will identify and document all the functional requirements necessary for effective team-work organization. Additionally, we will establish non-functional requirements covering performance expectations, security needs, usability standards, scalability considerations, and maintainability requirements. This comprehensive analysis will ensure that the system can meet both immediate and long-term organizational needs.

The system design is the third objective, where we will develop a complete system architecture incorporating the selected technologies. This includes designing an efficient database schema for the team work organization, creating detailed user interface mockups, defining system components and their interactions, and establishing a robust security architecture. The design phase will focus on creating a scalable and maintainable system structure.

Implementation forms the fourth objective, where we will develop core functionalities essential for the system's operation. This includes comprehensive user management capabilities, room creation and management features, event assignment and tracking systems, and advanced event displaying and filtering options. The implementation will also incorporate robust security measures, a responsive user interface, and ensure complete data integrity throughout the system.

The final objective focuses on *testing and validation*. This phase will involve developing and executing comprehensive test cases at multiple levels, including unit testing, integration testing, and user acceptance testing. We will verify all system functionality against the established requirements, document test results and any issues discovered, and implement necessary improvements based on testing outcomes. This thorough testing approach will ensure the reliability and effectiveness of the system in real-world applications.

1.2.2 Expected Outcomes

The successful completion of these objectives will result in:

- **A comprehensive understanding** of Java web development libraries and their practical applications
- **A functional application** for team work organization that: Facilitates efficient team collaboration, Provides intuitive event management, Ensures secure interactions with the system, Offers responsive and user-friendly interface.
- **Detailed documentation** covering: technology evaluation and selection process, system architecture and design decisions, implementation details, testing methodology, and results.
- **Practical experience** in: Java web application development, database design and implementation, User interface development, system testing and validation.

Chapter 2

Tools

2.1 Tool Selection and Technology Stack

This chapter examines the decision-making process behind selecting the technologies and tools for the team-work organization system. The selection criteria were based on several key factors: project requirements, scalability needs, security considerations, and long-term maintainability.

2.1.1 Framework Selection

Given the project's requirement to develop a Java application, the framework selection was focused on Java-based solutions. The primary options considered were *The Spring Framework*, *Spring Boot*, *Jakarta EE* (formerly Java EE), and *Micronaut*. Although all of these frameworks are capable of building robust web applications, they differ significantly in their approach and complexity.

Jakarta EE, the enterprise edition of Java, provides a comprehensive set of specifications to build enterprise applications. However, it requires significant boilerplate code and configuration, making it less suitable for this project. **Micronaut**, while promising with its ahead-of-time compilation and low memory footprint, is relatively new and has a smaller community compared to Spring Boot.

The Spring Framework, the foundation of Spring Boot, is a comprehensive programming and configuration model for modern Java-based enterprise applications. It provides a wide range of features including dependency injection, aspect-oriented programming, and transaction management. However, the Spring Framework requires extensive configuration and setup, which can be time consuming and complex.

Spring Boot, built on top of the Spring Framework, was selected as it addresses these configuration challenges while maintaining all the benefits of the Spring Framework. It provides the following:

- Auto-configuration of Spring Framework components
- Embedded servers for simplified deployment
- Production-ready features like metrics and health checks
- Reduced boilerplate code and configuration

This combination of Spring Framework's robust features and Spring Boot's simplified development approach makes it ideal for this project. The framework's extensive documentation, large community support, and proven track record in enterprise applications further reinforce this choice.

2.1.2 Frontend Technology Selection

The selection of front-end technology required careful consideration of integration capabilities with Spring Boot. **Thymeleaf**, as a server-side template rendering engine, was identified as the preferred option due to several factors. It utilizes server-side page rendering to decrease the computational load on client systems. Its integration of Java with HTML makes it relatively straightforward to use. Furthermore, it is frequently used in conjunction with the Spring Boot. Although modern alternatives like **React** and other JavaScript frameworks are prevalent, they would require significant additional research and development time.

The decision to use Thymeleaf was influenced by the simplicity of combining HTML and Java code in a single file. However, modern web development requirements necessitated the incorporation of some JavaScript for specific client-side tasks, particularly for handling asynchronous requests and dynamic content updates.

2.1.3 Database Selection

PostgreSQL was selected as the primary database system after evaluating various database options, including NoSQL alternatives. This choice was driven by PostgreSQL's prevalence in the market and its robust ACID compliance, which ensures data integrity and reliability. The ACID properties provide essential guarantees for data management:

- **Atomicity:** Ensures transactions are completed entirely or not at all.
- **Consistency:** Maintains database validity through all transactions.
- **Isolation:** Prevents concurrent transactions from interfering with each other.
- **Durability:** Guarantees committed transactions remain permanent.

These properties are crucial for maintaining data integrity in a multi-user environment where concurrent access and modifications are common.

2.1.4 Security Architecture Decisions

The security implementation strategy was developed after careful analysis of the security requirements of modern web applications. The decision to use **JWT-based authentication** was made after considering several factors, including the benefits of a stateless architecture that eliminates server-side session storage and improves scalability. The approach also offers excellent cross-platform compatibility, enabling seamless integration with various clients while maintaining compliance with industry best practices for web security. Alternative approaches like **session-based authentication** were evaluated but rejected due to their increased complexity with server-side session storage, increased server resource requirements, and complex session management.

The main security framework used is **Spring Security**, which provides a comprehensive security solution for Spring-based applications. Although it is a powerful framework, it is not without its own challenges, such as the need to configure multiple security components and the potential for increased complexity in the codebase. Still, the benefits of using a well-established security framework benefits the project in the long run.

2.1.5 Development Tool Selection

The choice of development tools was guided by the need to improve code quality and development efficiency. **Lombok** was selected because it effectively reduces boilerplate code without runtime overhead, integrates seamlessly with existing IDEs, and maintains code readability while reducing verbosity. The IDE used is **IntelliJ IDEA**, which has excellent support for Java.

2.1.6 Containerization Strategy

Containerization was implemented using **Docker** to ensure consistent deployment across different environments and simplify the development workflow. The containerization strategy employs a multi-stage build process that optimizes both build time and final image size. The build process utilizes **Amazon Corretto JDK 21** to compile the application and run it in a separate dockerfile stage. The system includes integrated health checks to monitor container status, ensuring reliable operation and quick detection of potential problems.

This containerization approach delivers significant benefits to the project. It ensures a consistent runtime environment across development, testing, and production stages, eliminating environment-specific issues. The deployment process is simplified, reducing the complexity of moving the application between different environments. Container isolation provides an additional layer of security, while the standardized container format enables easy scalability and orchestration capabilities. Furthermore, this approach effectively eliminates the common “works on my machine” problem, as all environments run the same containerized application.

2.1.7 Technology Stack Integration

The selected technologies were tested for their ability to work together. The integration strategy focused on ensuring compatibility between all components, maintaining optimal system performance across all layers, and creating a system that can be easily updated and modified. This careful selection and integration of technologies has resulted in a robust, scalable, and maintainable system that meets project requirements while providing a solid foundation for future enhancements.

Chapter 3

Methods

3.1 Requirements Specification

3.1.1 Functional Requirements

The system provides comprehensive functionality in three main areas: **Basic Calendar Features**, **Additional Calendar Features**, **Secure Interactions**.

Under the **Basic Calendar Features** category, the system offers weekly and daily views, with filtering capabilities by room and user. The system allows managers to create, edit, and delete events, while users can only view the calendar and search for resource availability.

In the **Additional Calendar Features** category, the system offers a day view, which displays events for a specific day organized by room. Furthermore, the system offers a **FindAvailable** feature, which allows users to search for available resources (rooms, users, or events) based on various criteria.

Under the **Secure Interactions** category, the system offers a login system, which allows users to log into the system using a username and password. The system stores all passwords in a hashed format using the BCrypt algorithm. Instead of traditional session-based authentication, the system uses JWT tokens to authenticate users stateless-way.

3.1.2 Non-Functional Requirements

The application prioritizes function over design by providing a simple and intuitive user interface. Navigation and interaction patterns remain consistent throughout the system, with clear error messages and feedback mechanisms to guide users.

Security is implemented through JWT-based authentication, with all communication restricted to HTTPS. The system incorporates protection against common web vulnerabilities and maintains strict access controls.

Reliability is ensured through robust error handling and data consistency mechanisms. The system maintains data integrity and implements graceful degradation under load conditions.

Maintainability is supported by comprehensive documentation, a modular architecture, and extensive test coverage. This ensures that the system can be easily updated and maintained over time.

3.2 System Architecture

3.2.1 Overall Architecture

The application follows a four-tier architecture pattern, separating concerns across Models, Views, Controllers, and the Database. This 3.1 separation enables better maintainability and scalability while keeping the system modular and efficient.

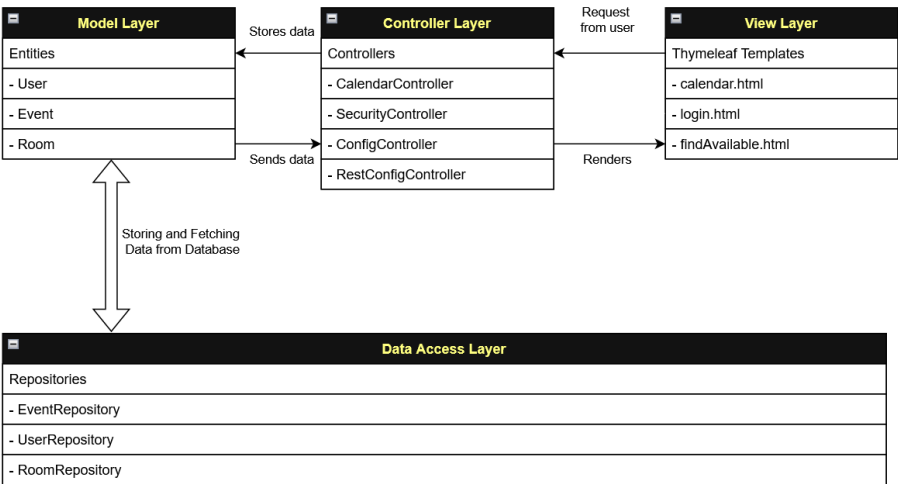


Figure 3.1: System Architecture Diagram

3.2.2 Key Components

The front-end components utilize Thymeleaf templates for server-side rendering, enhanced with JavaScript for client-side interactivity and CSS for design. This combination provides a modern, dynamic user experience while maintaining performance, as the Thymeleaf templates are rendered on the server side.

The backend architecture consists of controllers handling HTTP requests and responses, services implementing business logic, repositories managing data access, and a comprehensive security layer handling authentication and authorization.

3.2.3 API Design

The application provides both REST API endpoints and view-based endpoints to support various client needs. The REST API follows standard conventions and provides endpoints for authentication, resource management, and system queries.

Authentication endpoints handle user sign-in, registration, and logout operations. Resource endpoints manage rooms and events, while query endpoints provide functionality to check resource availability and validate authentication tokens.

View endpoints serve the user interface, with dedicated routes for calendar views and system configuration. Calendar views support weekly and daily perspectives, while configuration views provide interfaces for managing rooms, events, and users.

3.3 Database Design

3.3.1 Database Schema

The database schema 3.2 is designed around three main entities: users, rooms, and events.

After that, the roles table is used to manage the user roles.

In addition, 3 tables of tags are used to manage the tags for rooms, events, and users to simplify the search for resources.

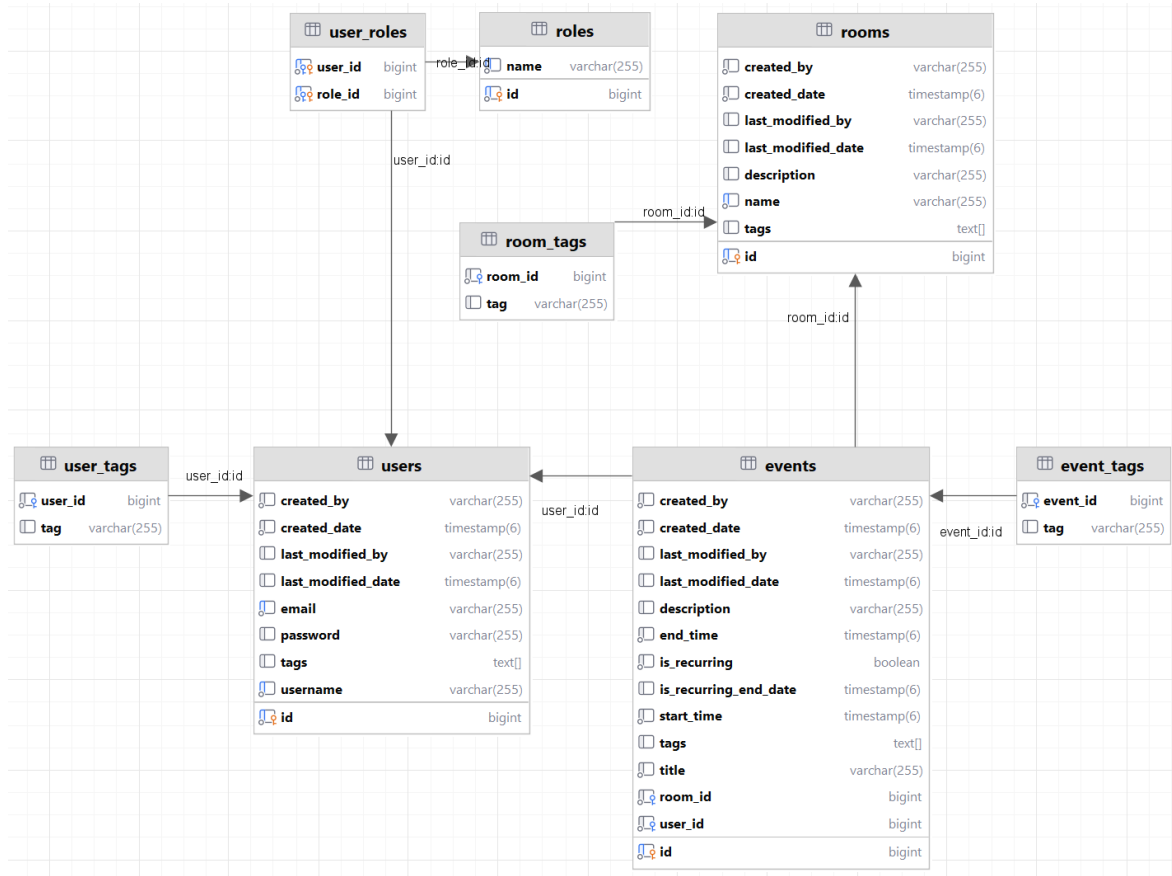


Figure 3.2: Database Schema Diagram

3.3.2 Key Tables and Relationships

The table “users” stores essential user information, including username, email, and securely hashed passwords. Maintains a many-to-many relationship with the role table, enabling flexible role assignment and management. This solution is not the most elegant one, as it requires additional queries to get the roles of a user. However, it was chosen to simplify the implementation of authentication and authorization.

The roles table defines the various user roles within the system, with a connection to the users table using another table called `user_roles`. This design allows for the easy addition of new roles and modification of existing role permissions.

The room table contains information about spaces, including names, descriptions, and categorization tags. Creating a one-to-many relationship with the events table, the rooms table can have multiple events, but not at the same time.

The event table serves as the central scheduling entity, storing event details such as title, description, timing information, and recurrence settings. Events maintain relationships with both users and rooms, tracking who is involved and where the event takes place. Recurrence is implemented following the RFC 5545 standard.

3.3.3 Spring Data JPA Implementation

The application relies on Spring Data JPA for efficient data persistence. Entity classes are annotated with `@Entity` and `@Table`, while repositories extend `JpaRepository` to provide standard CRUD operations. Relationships between entities are managed through JPA annotations and Spring handles transaction management automatically.

3.4 User Interface Design

3.4.1 Design Philosophy

The design of the user interface emphasizes simplicity, consistency, responsiveness, and accessibility.

The main colors are

- **–color-gold** (46.8%, 92.9%, 61.6%): Gold
- **–color-primary** (0%, 0%, 100%): White
- **–color-secondary** (336%, 25%, 60%): Dusty Rose
- **–color-tertiary** (314%, 53%, 21%): Deep Magenta
- **–color-accent** (339%, 27%, 17%): Plum
- **–color-additional** (345%, 18%, 4%): Charcoal

3.4.2 Key Interface Components

The calendar view presents a weekly grid layout with color-coded events.

Event management features include modal forms to create and edit events, quick filters for different event types, and clear conflict warnings. The interface guides users through the event creation process while preventing scheduling conflicts.

Navigation is streamlined through a consistent menu structure.

3.5 Implementation Details

3.5.1 Backend Implementation

The calendar module implements sophisticated event scheduling logic, availability checking, and recurring event handling. The security module manages JWT token generation and validation, role-based access control, and password encryption. The resource management system handles room allocation, user assignment, and conflict resolution.

Key algorithms include the findAvailable algorithm for time slot calculation and overlap detection, and the event conflict resolution system to verify resource availability and user schedules.

3.5.2 Frontend Implementation

The front-end structure is built around Thymeleaf templates, JavaScript for interactivity, and CSS for styling. The base layout template provides a consistent framework, while component fragments enable modular development. The error page ensures a better user experience even when problems occur.

JavaScript handles interactions with the controllers and the backend. It also handles dynamic content updates and form validation.

3.6 Development Tools and Environment

3.6.1 Development Environment

The development environment is IntelliJ IDEA as the primary IDE, with Gradle handling build management and Git providing version control. Dockerfile was utilized to build and containerize the application. Docker-Compose is configured to correctly set up the application and the database and link them together.

3.6.2 Testing

The testing strategy employs JUnit 5 for unit testing, Spring Test for integration testing.

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.