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

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

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 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 navrhněte 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**

[illegible] — doc. Ing. Miroslav Virius, CSc., podpis vedoucí(ho) práce

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

[illegible] — 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 nedlí na specializace, tuto ádku odstranit.)

Druh práce: Bakaláská práce

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

Abstrakt: Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. Abstrakt max. na 10 ádk. 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. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text.

Key words: keywords in alphabetical order separated by commas

Contents

Chapter 1

Introduction

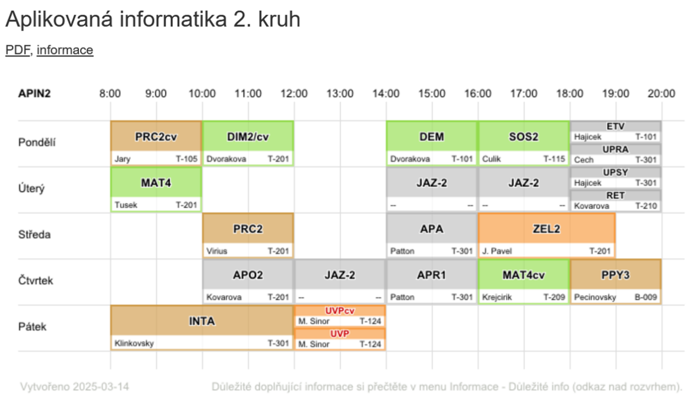
Motivation

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



Aplikovaná informatika 2. kruh

PDF, informace

APIN2	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00
Pondělí	PRC2cv Jary T-105		DIM2/cv Dvorakova T-201				DEM Dvorakova T-101		SOS2 Culik T-115		ETV Hajicek T-101 / UPRA Cech T-301		
Úterý	MAT4 Tusek T-201						JAZ-2 -- --		JAZ-2 -- --		UPSY Hajicek T-301 / RET Kovarova T-210		
Středa			PRC2 Virius T-201				APA Patton T-301		ZEL2 J. Pavel T-201				
Čtvrtek			APO2 Kovarova T-201		JAZ-2 -- --		APR1 Patton T-301		MAT4cv Krejcirik T-209		PPY3 Pecinovsky B-009		
Pátek	INTA Klinkovsky T-301				UVPcv M. Sinor T-124 / UVP M. Sinor T-124								

Vytvořeno 2025-03-14 Důležité doplňující informace si přečtěte v menu Informace - Důležité info (odkaz nad rozvrhem).

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*. While all 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 for building 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:

- 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 frontend technology selection required careful consideration of integration capabilities with Spring Boot. **Thymeleaf**, a server-side template rendering engine, emerged as the primary choice for several reasons. It processes page rendering on the server side, reducing computational demands on client systems. While 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 modern web application security requirements. 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** for compiling the application and running it in a separate stage of dockerfile. The system includes integrated health checks for monitoring container status, ensuring reliable operation and quick detection of potential issues.

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 streamlined, 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

This section will focus on the tools and techniques used to create a web application tShreyash-Vardhan2024o manage time and space resources of an organization.

3.1 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.

3.1.1 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 setters 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.

```java
@Data
@Entity
@Table(name = "rooms")
@EqualsAndHashCode(callSuper = true)
public class Room extends Auditable {
    @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

3.2 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.

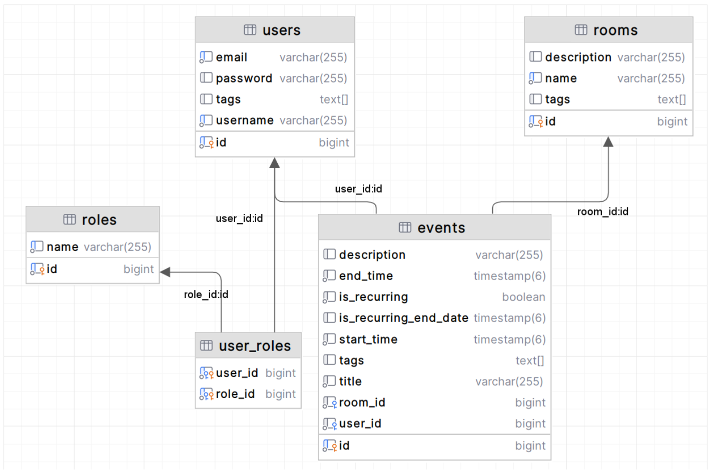


Figure 3.1: Schema of the database for the application

3.3 Schema of the database

Database 3.1 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.

3.4 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.

3.4.1 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.

Then, the `EventRepository` is used to find all the events that are happening in the week of the date sent. The `EventRepository` is a Spring Data JPA repository that allows for easy access to the database. They are split into a list of events that are happening on a specific day, where the days are filtered by comparing the `userIds` and `roomIds` with the events that are in the database. the data inside each event is such:

- **eventId**
- **eventTitle**
- **eventDescription**
- **eventStartTime** - the time when the event starts in LocalDateTime format.
- **eventEndTime**
- **eventRoomId** - the room that is assigned to this event.
- **eventUserId** - the user that is assigned to this event.
- **eventTags** - the tags that are assigned to this event.
- **eventIsRecurring** - if the event is recurring or not.

The `/calendar` then sends such 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**

3.4.2 Additional Calendar Endpoints

The CalendarController also provides a day view endpoint at `/calendar/day`, which displays events for a specific day organized by room. This endpoint accepts the following parameters:

- **date** (required): The specific date to display events for.
- **userIds** (optional): Filter events by specific users.
- **roomIds** (optional): Filter events by specific rooms.

The day view provides a more detailed perspective of events occurring on a single day, organized by room. This is particularly useful when there are many events scheduled on a specific day, making the weekly view potentially cluttered.

Another important endpoint is `/calendar/findAvailable`, which allows users to search for available resources (rooms, users, or events) based on various criteria:

- **searchType**: Specifies whether to search for “rooms”, “users”, or “events”.
- **tags**: Allows filtering by specific tags associated with the resources.
- **date**: The date to search for availability (defaults to current date if not provided).
- **startTime** & **endTime**: The time range to check for availability.

This functionality is particularly valuable for quickly identifying available resources during a specific time slot, facilitating efficient scheduling and resource allocation.

3.4.3 The findAvailable Algorithm

The findAvailable algorithm is a sophisticated process that identifies available time slots for rooms, users, or events within a specified time range. Here’s how it works:

1. **Input Processing**: The algorithm accepts several parameters:
   - `searchType`: Determines whether to search for available rooms, users, or events (By default, “rooms”)
   - `tags`: Optional tags to filter resources by (e.g., “projector”, “whiteboard”, etc.)
   - `date`, `startTime`, `endTime`: Define the time range to check (defaulting to the current date if not provided)
   - `durationMinutes`: Minimum duration required for an available time slot (By default, 30 minutes)
2. **Data Collection**: The system retrieves all events that overlap with the specified time range.
3. **Entity Filtering**: Based on the searchType, the algorithm filters entities (rooms, users, or events) by the provided tags.
4. **Availability Calculation**: For each entity, the algorithm:
   - Identifies all events associated with the entity

- Creates a list of occupied time slots from these events
- Sorts occupied slots by start time and merges any overlapping slots
- Calculates unoccupied time slots by finding gaps between occupied slots
- Filters unoccupied slots by the minimum duration requirement

5. **Result Generation**: The algorithm creates a list of available entities along with their unoccupied time slots.

3.4.3.1 Implementation Details

The key component of this algorithm is the `calculateUnoccupiedTimesFromOccupied` method, which efficiently identifies available time slots by:

1. Sorting occupied time slots by start time
2. Merging overlapping occupied slots
3. Iterating through occupied slots to find gaps (unoccupied times)
4. Filtering unoccupied slots by minimum duration
5. Sorting the resulting unoccupied slots

3.4.3.2 Handling Overlapping Time Slots

A critical aspect of the algorithm is the `removeOverlaps` method, which merges overlapping time intervals:

1. The method requires that time slots are already sorted by start time
2. It iterates through adjacent pairs of time slots
3. When two slots overlap (the end time of one is after the start time of the next), they are merged
4. The merged slot spans from the start time of the first slot to the end time of the second slot
5. The second slot is removed from the list, and the iteration index is decremented to recheck the newly merged slot
6. This process continues until no more overlaps exist

3.4.3.3 Calculating Unoccupied Time Slots

The algorithm identifies unoccupied time slots through the following process:

1. Start with the requested start time as the current point
2. For each occupied time slot (after sorting and merging):
   - If the current point is before the start of the occupied slot, add an unoccupied slot from the current point to the start of the occupied slot
   - Update the current point to the end of the occupied slot

3. After processing all occupied slots, if the current point is before the requested end time, add a final unoccupied slot from the current point to the requested end time
4. Filter the resulting unoccupied slots to include only those that meet or exceed the minimum duration requirement

3.4.3.4 Edge Case Handling

The algorithm handles several edge cases:

- **Empty tag sets**: If no tags are specified, all entities of the requested type are included
- **No events**: If there are no events in the specified time range, the entire range is considered unoccupied
- **Completely occupied time range**: If the entire time range is occupied, no unoccupied slots are returned
- **Invalid entity type**: If an invalid entity type is provided, an IllegalArgumentException is thrown

3.4.3.5 Performance Considerations

Several optimizations enhance the algorithm's performance:

- **Early filtering**: Events are filtered by time range at the database level before further processing
- **Efficient tag filtering**: Uses `Collections.disjoint()` to quickly check for tag matches
- **Sorting before processing**: Ensures that time slot operations can be performed in a single pass
- **Stream operations**: Leverages Java streams for concise and efficient filtering operations

This approach ensures that users can quickly find available resources that meet their specific time and duration requirements, optimizing resource allocation and scheduling efficiency. The algorithm's modular design also allows for easy extension to support additional entity types in the future.

3.5 Authentication and Security

The application implements a robust security system using JSON Web Tokens (JWT) for authentication. The SecurityController handles the authentication process through several REST endpoints:

- `/auth/signin` (POST): Authenticates a user with username and password, generating a JWT token stored in a secure HTTP-only cookie.
- `/auth/signup` (POST): Registers a new user with username, email, and password. The password is securely hashed before storage.
- `/auth/signout` (POST): Logs out a user by invalidating their JWT cookie.

The JWT implementation enhances security by:

- Using HTTP-only cookies to prevent JavaScript access to the token
- Setting the Secure flag to ensure transmission only over HTTPS
- Implementing token expiration to limit the window of vulnerability

The MainController provides view endpoints for authentication-related pages:

- `/`, `/login`, `/signin`: All route to the sign-in page
- `/register`, `/signup`: Route to the registration page
- `/signout`: Routes to the sign-out page

3.6 Configuration and Administration

The application provides both view-based and REST API endpoints for configuration and administration.

The ConfigController offers view endpoints for managing system entities:

- `/config/rooms`: Lists all rooms in the system
- `/config/events`: Lists all events in the system
- `/config/users`: Lists all users in the system

For programmatic interaction, the RestConfigController provides REST API endpoints:

- `/api/v1/addrooms` (POST): Creates a new room
- `/api/v1/addevents` (POST): Creates a new event, with conflict checking
- `/api/v1/checkavailability` (POST): Verifies if a room is available during a specific time period
- `/api/v1/deleterooms/{id}` (DELETE): Removes a room by ID
- `/api/v1/deleteevents/{id}` (DELETE): Removes an event by ID
- `/api/v1/validateJWT` (POST): Validates the current JWT token

3.7 Error Handling

The application implements a CustomErrorController that provides a unified approach to error handling. When an error occurs, the controller captures details such as:

- Error status code
- Exception type
- Error message
- Request path that caused the error

- Stack trace (in development environments)

This information is then rendered in a user-friendly error page, enhancing the debugging process while maintaining a consistent user experience even when errors occur. This could be further improved by adding a system that would notify the administrator about the error, ensuring that it is addressed promptly.

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.