Heinrich Heine
Universität
Düsseldorf

INSTITUT FÜR INFORMATIK
Softwaretechnik und Programmiersprachen
Universitätsstr. 1 D–40225 Düsseldorf

A Planning Tool for Room Booking Management

Bachelorarbeit

im Studiengang Informatik
zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

vorgelegt von
Vladimir Cabacov

Beginn der Arbeit:	14. April 2023
Abgabe der Arbeit:	14. July 2023
Erstgutachter:	Prof. Dr. Michael Leuschel
Zweitgutachter:	Dr. Jens Bendisposto

Selbstständigkeitserklärung

Hiermit versichere ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, den 14. July 2023

[illegible]

Vladimir Cabacov

Abstract

In this thesis, a web application was developed to manage room bookings in building 25.xx. The application consists of a backend and a frontend that work together to provide a user-friendly and efficient experience.

The backend of the application is responsible for database management and communication with the frontend via an api. It provides various functions including the management of semesters, courses, rooms and conflicts. The backend also reads a configuration file to load the necessary settings such as the database location and server port.

The front end of the web application consists of different views that allow users to interact with the data. The main view is the semester dashboard, where users can access the relevant data. In the courses view, courses can be displayed in an organized table, with search, sort, and filter functions for better organization. The rooms view provides a similar table for managing rooms. The individual course and room views provides a detailed view of events, with drag-and-drop functions for moving events and export options to various file formats. Collisions view displays found conflicts and allows their management.

The developed web application makes booking management of events in building 25.xx easier. Thanks to the intuitive user interface and the extensive functions, events can be managed quickly and efficiently.

Contents

1 Introduction

The efficient management of university rooms is an important task for educational institutions. Especially in the 25.xx building, where a large number of events take place, a structured booking management is essential. To meet this need, this bachelor thesis focuses on the development of a web application to manage bookings for the rooms and to avoid collisions between events in 25.xx building.

1.1 Challenge and goal of the work

The central challenge lies in the effective processing of bookings, the identification of conflicts and the presentation of data. The data should be converted from the external system - currently, the HIS-LSF, into the internal data structure so that it can be processed efficiently. Then, to identify criteria that define a conflict (for example, room collisions can occur where two events are booked in the same time slot) in bookings and then to develop a user-friendly and functional interface that provides the user with an intuitive interface to manipulate data.

The aim of this project is to develop a Planning Tool for managing bookings in the 25.xx area. The booking is done in three phases, where the first phase includes the booking of a 25.xx area. The second phase involves booking rooms in the 25.xx area that are still available after the first booking phase, and the third phase allows the Dean's Office to move existing bookings to another event. It represents a complex booking process and therefore requires the management of collisions/conflicts that may occur between bookings. Therefore, it is important to develop a system that detects and represents collisions in the bookings. The courses are only imported from the external system and cannot be created in the tool, it is important that the data can be imported several times, by inserting new data in the global system or changing existing data at different stages of the booking process. The changes made in the tool should be kept after the re-import. For this reason, the tool requires a simple import function that receives the data from an XML file, extracts required data and decides which data should be saved.

To achieve this goal, the tool should allow import, management and presentation of data and meet all specific requirements, which will be discussed in more detail in the next chapter.

In the context of this bachelor thesis, different technologies and approaches are investigated in order to fulfill the requirements for booking management. In particular, the representation and modification of data, the effective search for collisions, the implementation of the import function and the integration of the client/server components are considered.

1.2 Outline

The structure of this thesis is as follows:

Chapter 2 begins by listing the requirements for the tool. In chapter 3, the basic concepts, technologies and frameworks are explained. After that, in chapter 4, the backend is described and the implemented functions of the backend are presented. Chapter 5 focuses on the frontend and its functions. A summary the resulting software is presented in chapter 6. Finally, in chapter 7, the evaluation and, in chapter 8, an outlook on possible future extensions are given.

With this bachelor thesis a contribution is made to the optimization of the room administration at the university and a system for efficient booking management is developed. The presented web application allows to import bookings, identify collisions and make optimal use of resources in building 25.xx.

2 Requirements

The requirements for HERLSF - Planning Tool include a number of functions that are intended to ensure, that event management in Building 25.xx can be performed effectively and efficiently. The main requirements are as follows:

- Import of data: Since the courses can only be imported from the external system, the tool should be able to import data from xml files and convert them into an internal data structure. It should also be possible to perform multiple xml imports, updating only unchanged data so as not to overwrite the changes made by user in the tool.

- Conflict identification and presentation: The tool should be able to identify conflicts in the booking data and present a visual representation of these conflicts to the user. This allows the user to manually review the conflicts and decide how to resolve them.

- Booking management: the tool should provide management of existing bookings, including the possibility to add new bookings.

- Management of semesters: The tool should allow the separate management of semesters. This facilitates the organization and access to the booking data for specific semesters.

- Documentation: As part of the work, it is important to provide comprehensive documentation for the developed tool. The documentation should cover the functionality, the implementation details, the usage of the tool as well as possible extensions and optimizations.

The work includes deep planning and implementation of the tool, including extensive testing, optimization, and detailed documentation.

Possible extensions to the requirements for the tool that could be considered. Among them:

- Extension of the data structure: to include additional information in the data structure (such as workgroups).
- Additional representation of collisions: In addition to the visual representation of conflicts, the tool can be extended to provide more detailed information about the nature and extent of collisions. This could be achieved, for example, by highlighting the affected dates or by providing additional filtering and sorting options.
- Export of data: It can be useful to export the booking data stored in the tool in different formats. An extension could provide the possibility to export data as pdf, csv or excel files. This allows the data to be easily shared or further processed with other systems.

Some of these extensions are already implemented.

Implementing these enhancements would allow the tool to better meet the needs of managing booking appointments in Building 25.xx.

3 Fundamentals

This section explains the basic concepts, terms and methods required to understand this work. To meet all requirements a client/server architecture is used. On the server side, Clojure [Hic23a] is used to implement a robust backend with a corresponding database. The frontend is developed using ClojureScript [Hic23b] to provide a user-friendly and interactive user interface. It will cover both the theoretical foundations and the applied techniques used in the development of the web application with Clojure backend and ClojureScript frontend.

3.1 Clojure backend

The Clojure backend is the foundation of the web application and uses a number of libraries and tools to provide the needed functionality. In the context of this work, the datahike [lU21] database was used. Datahike is a powerful database for Clojure based on the Datalog model. By using this database system, the tool can store data efficiently and formulate complex queries in a simple and elegant way. Datahike also provides functions for transaction security and consistency of data.

Another powerful tool used in the Clojure backend is the reitit [Oy23] routing library. Reitit not only provides a flexible and robust way to manage routes, but also allows the use of Spec Coercion. Spec Coercion is a mechanism that allows validation and transformation of data to the route level. This allows incoming data to be automatically validated and transformed into the desired format, further simplifying the handling of requests and responses. With this combination, a clear structuring of the application is achieved, as well as ensuring the security and consistency of the data.

3.2 ClojureScript frontend

The ClojureScript frontend provides the user interface of the web application and also uses different libraries and frameworks to enable a reactive and interactive user experience. A core component here is the reitit router, which is used in both the backend and the frontend. This allows for consistent and easy route management on both sides of the application. The frontend uses the Bootstrap framework [Boo23] for styling.

Bootstrap is a popular and comprehensive CSS library that provides a large number of pre-built components, layouts, and styles. Using Bootstrap can give the frontend a responsive and consistent look.

3.3 Re-frame and Reagent

For the implementation of the user interface, Reagent [rp23] is used. Reagent is a React [met23] binding for ClojureScript and allows easy creation of component-based user interfaces. Reagent can be used to create declarative views that reactively respond to changes in application state. Reagent uses a special syntax called hiccup [Ree23] that allows user interfaces to be described as data structures. These data structures represent the components and their properties. When the application state changes, Reagent automatically updates the appropriate components to keep the user interface in sync.

In addition, the re-frame [Tho22] framework is also used. Re-frame is a framework for state management in ClojureScript applications. It is based on the Flux architectural pattern and provides a clear structure for managing application state. With re-frame, data flows can be easily organized and state changes can be efficiently handled. Re-frame uses a unidirectional data flow pattern where actions are triggered to update state. These actions are handled by event handlers, which update the application's state according to the action performed. The state is stored in an atomic data structure managed by re-frame.

The combination of re-frame and Reagent enables efficient development of interactive user interfaces. The decoupling of state management from the user interface and the reactive updating of views, contribute to better maintainability and usability of the web application. In addition, the use of React as a substructure enables high performance and optimal use of user device resources.

Thanks to re-frame and Reagent, responsive user interfaces could be created in the web application that dynamically adapt to changing application data and provide a smooth user experience.

3.4 Related work

In the context of this bachelor thesis, the focus is on the representation and processing of booking data that comes from the central booking system (LSF). The imported booking data is carefully processed, stored and represented in a user-friendly web application. This allows users to create local booking entries, identify collisions between bookings and add new events that are not captured in the global system.

In researching related work, it was found that most existing room booking management tools and systems, such as classroombookings [Rod22], work with their own internal data and generally offer more flexibility. These systems can usually customize their data structure and meet various requirements of Room Planning. However, in the case of this work, there are specific requirements because the booking data comes from an external system and therefore requires special handling.

The development of this specific solution makes it possible to efficiently use existing resources and consider individual events that are not captured in the central booking system. This allows the user to have a more comprehensive and precise booking management that corresponds to specific requirements.

3.5 Experiments

This section presents and discusses the experiments that were conducted as part of the thesis. The aim was to try out different approaches and technologies in order to find the best possible solution for the given challenge.

3.5.1 Architecture

In the development of the tool, cljfx [clj22] was initially experimented with. cljfx is a powerful framework for developing desktop applications in Clojure. It offers a wide range of features and components to create user-friendly and responsive interfaces. During the experiments with cljfx, the possibility of developing the tool as a desktop application was explored.

However, as the process progresses, it was found that the use of Clojure and ClojureScript provides the possibility to make the tool accessible not only on desktop platforms, but also on mobile devices. ClojureScript enables the development of web applications that can be used on multiple platforms, including mobile devices.

This opened the perspective of implementing the tool as a web application accessible by user on both PCs and mobile devices through a web browser.

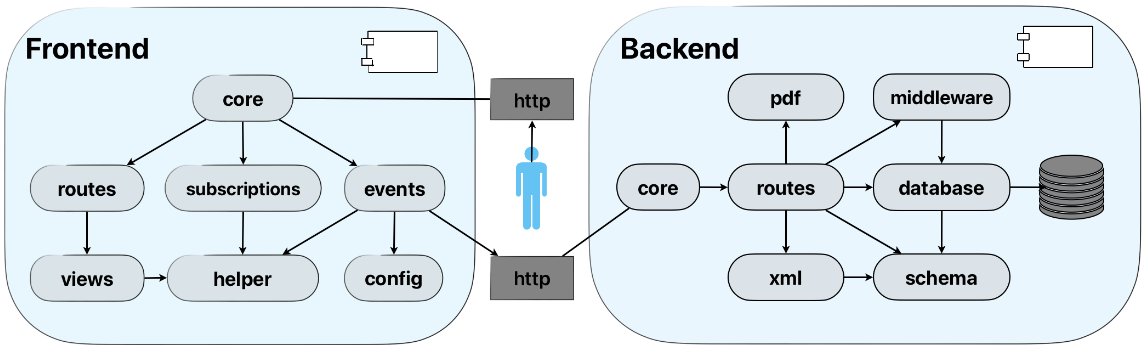


Figure 1: Application architecture

This decision made it possible to extend the scope of the tool and offer a flexible and platform-independent solution. And this has resulted in the architecture represented in Figure 1.

3.5.2 Router library

During the development of the web application, it was experimented with different routers to find the optimal frameworks and libraries for routing in the backend and frontend. Initially, the backend was implemented with Compojure [Ree22] and the frontend with bidi [LTD23]. However, it turned out that both bidi and Compojure are considered deprecated libraries.

To ensure uniformity and up-to-dateness of the libraries used, it was finally decided to switch to reitit [Oy23]. Reitit is a modern Clojure routing framework that provides a clean and flexible routing solution. This switch enabled a consistent routing implementation in both the backend and frontend.

The experiment showed that it is important to use up-to-date and well-maintained libraries to enable smooth development. The decision to switch to reitit helped improve the overall uniformity of the application and made it easier to maintain and extend the system in the future.

With this basic understanding of the concepts and techniques used, the reader is now ready to explore the subsequent chapters detailing the actual implementation and results of the work.

4 Backend implementation

This chapter provides a detailed insight into the backend implementation of the web application. Here, the different components of the backend are explained step by step, starting with the definition of the schema for the data structure via xml data processing for the import of booking data to the database integration with specific queries and the implementation of the routes with middleware functions for smooth context switching.

The backend contains a total of 8 components, which are shown in Figure 2.

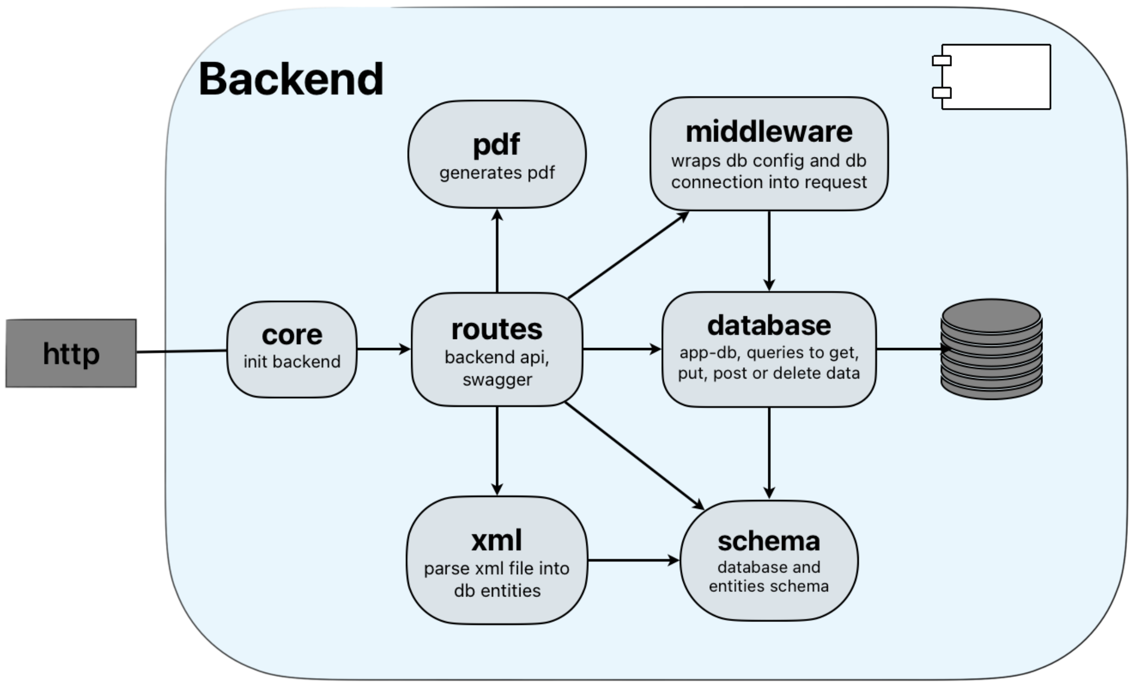

Backend
http
core
init backend
routes
backend api, swagger
pdf
generates pdf
middleware
wraps db config and db connection into request
database
app-db, queries to get, put, post or delete data
xml
parse xml file into db entities
schema
database and entities schema

Figure 2: Backend architecture

4.1 Schema

Through a careful analysis of a representative xml example, a customized database schema was developed that best met the specific requirements of the tool.

The analysis of the xml examples made it possible to understand the structure, elements, and relationships of the data and to translate them into a suitable database schema. This involved translating the various elements of the xml example into suitable data schemas, defining the fields, data types and relationships in a way that preserved the integrity of the data and enabled efficient queries.

By developing the database schema in a customized way based on the analysis of the xml example, it was possible to create an optimal schema (see Figure 3) that best meets the specific requirements of the tool and the data to be processed. It allows efficient storage, retrieval and manipulation of the analyzed data and ensures high data integrity and consistency.

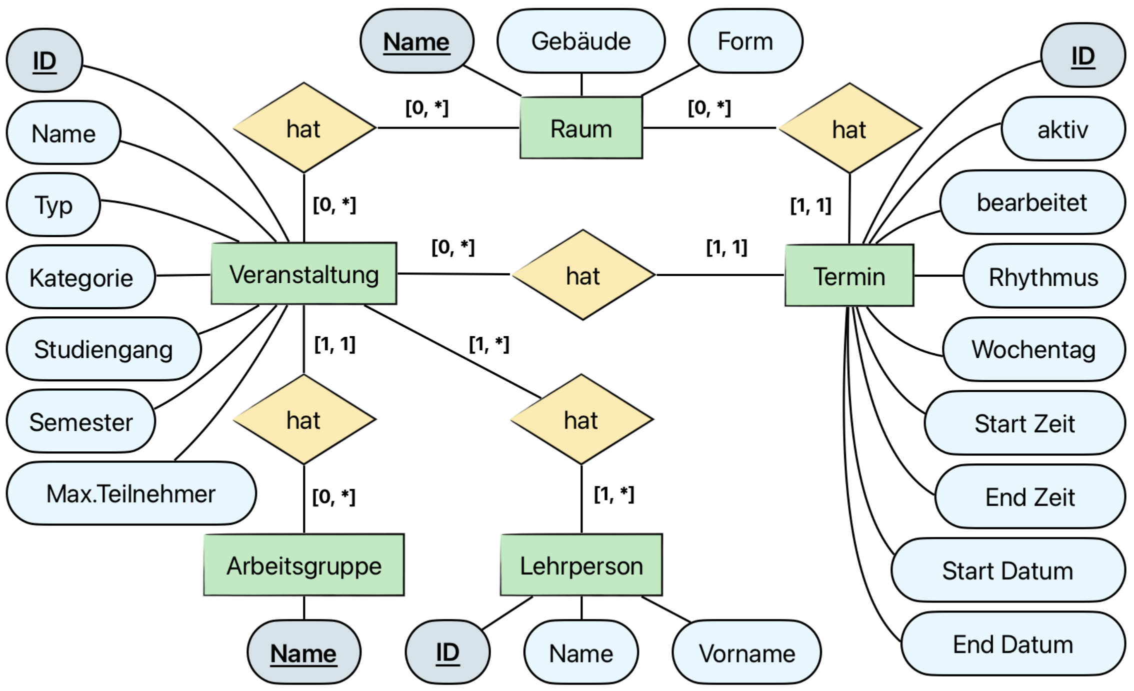


Figure 3: Entities schema

In this work during the development of Schema, some critical decisions were made, which are explained in the following:

- Not only positive numbers are used for the assignment of event IDs. When creating local entries, it is assumed that no appointments are exported from the central system that contain negative IDs. This ensures that when a local entry with a negative ID is created, no collision occurs when the xml file is re-imported.
- To allow re-importing the xml file, a field called "edited" has been added to the event schema. This allows all edited events to be deleted from the import list, while the changes made in the app are preserved.
- Events that have not been created in the app cannot be deleted, as they will be added again when they are imported again. For this reason, the schema provides a field called "active" in the event schema that can be used to disable and hide an event. This also removes the event from the conflict set.

- The app is designed to identify and represent any conflicts. However, there are situations where not all conflicts are resolvable. To provide a way to suppress a conflict, a conflict suppression scheme (see Figure 4) has been developed. This schema contains information about suppressed collisions and allows to identify them. It assigns an ID and a conflict type (room or lecturer conflict) to the identified conflict and stores as data the course ID with the corresponding event ID of all affected events, where “Raum oder Lehrperson” stores room name or lecturer id according to the conflict type. The schema allows the frontend to differentiate the suppressed conflicts and hide them if necessary.

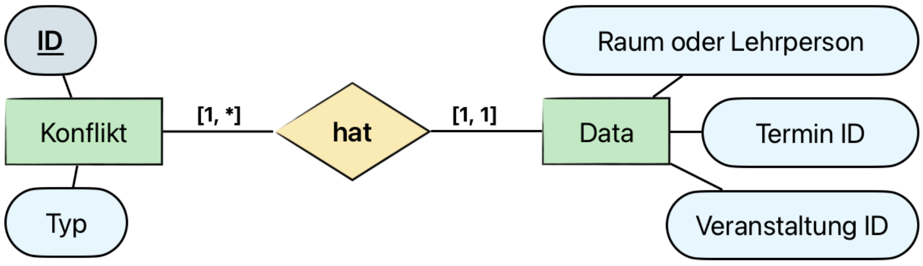


Figure 4: Suppression schema

4.2 Xml

The processing of xml data has an important role in the correct parsing of existing bookings. The tool allows importing xml files that can be updated multiple times. This approach was achieved by developing a suitable xml processing mechanism and a corresponding database schema.

When importing an xml file, the tool analyzes the contained data and extracts relevant information using a specter library [Lab22]. During this process, different elements and attributes of the xml structure are taken under consideration in order to ensure an accurate mapping of the data in the database.

4.3 Database

The database plays a central role in the implementation of the developed tool. The datahike database has been used for the implementation, which allows efficient and flexible storage and querying of the analyzed data. During the development, comprehensive experiments with different query methods were conducted to find the fastest and most robust approach to optimize the queries.

As part of the implementation of the tool, it was decided to separate databases for different semesters. This allowed a better organization and management of the data. To support this functionality, special middlewares have been developed, which will be discussed in the next step.

An important aspect of the implementation was the optimization of the queries to ensure fast response times and smooth interaction with the database. Several techniques and approaches were investigated to improve the performance of the queries.

Through the experiments and optimizations, an efficient query execution was achieved that can handle both large data sets and complex query requests. This enabled the tool to quickly and accurately retrieve the analyzed data.

The developed schema was designed in such a way that when the xml file is imported again, only the data that has not yet been changed in the tool is updated. This flexibility and updating capability of the xml import allows the tool to respond to changes in the xml data and make the appropriate updates in the database. By analyzing and processing the xml files, the imported data can be effectively integrated into the tool.

Some decisions were made, which are explained below:

- For a stable use of the database, it was decided to set a constant scope "localhost" when writing the schema. Otherwise the database cannot be accessed with another ip address.

- Collisions: Two types of collisions have been defined:

  - Lecturer collision: This occurs when a lecturer is scheduled in two courses at the same time and on the same day.
  - Room collision: This occurs when two events take place at the same time, in the same day and in the same room.

- It is important to note that an event must be active and the fields for time, date and day of the week must be filled in.

- To identify potential conflicts, appropriate database queries were developed to provide an overview of which events are affected.

- An extension of the scheme was carried out to integrate workgroups. This allows for better management and planning. This extension allows assignment the corresponding courses to a workgroup.

4.4 Middleware

The middleware plays a crucial role in handling the database connections and configurations for each semester. When a request is sent to the server, the middlewares extracts the name of the associated semester from the request. Based on this name, the middleware adjusts the database configuration accordingly and creates a connection to the specific database for the context.

This specialized middleware allows the tool to seamlessly switch between different databases depending on which context is being used. It ensures that the data is isolated and properly organized, and that the correct database connection is made for each request.

After the middleware establishes the database connection for the current context, it passes the connection directly to the corresponding handler, which processes the request further. This ensures that the handler can access the specific database relevant to the current context and manipulate or retrieve the data accordingly.

The development of this middleware allows efficient and flexible handling of databases for different contexts within the tool. It ensures clear separation of data and allows users to access specific data for a particular context without conflicting with other data.

4.5 Pdf

The integration of pdf generation into the developed tool opens other possibilities for the output and presentation of the analyzed and modified data. By using a suitable clj-pdf [cp23] library, the tool can generate pdf files that present the relevant information in a user-friendly format.

This enables the quick generation of PDF files that contain information about a course or a room along with their respective events. Additionally, clear and concise tables representing the events can also be generated as PDF files.

The pdf features extend the scope of the tool and allow the results of the analysis to be viewed and shared in a formatted manner.

4.6 Routes

The implementation of routes ensures that the developed tool can process incoming requests efficiently and in a target-oriented way. By defining routes, incoming traffic is directed to specific endpoints. The implementation of routes enables the tool to handle requests according to their nature and parameters and to perform the right actions. This helps to clearly structure and organize the code and enables communication between the tool and frontend/user.

Additionally, the routes component provides a swagger api. When the tool is running, the swagger ui can be accessed via the path "http://<backend-url>:<backend-port>/swagger". There, detailed information about each route is provided with the corresponding input and output parameters.

Using the swagger api makes it easier to document and interact with the backend. Developers can easily identify the available routes and their functionalities and understand the required input parameters as well as the expected output parameters. This fosters smooth communication and integration between the frontend and the backend.

4.7 Core

First, the core reads the configuration file located in the "resources" directory. This configuration file contains information such as the "db-location", which specifies the relative path to the databases, and the "server-port", which defines the port on which to execute the backend.

After reading the configuration file, the core initializes the application database by loading all the required configurations for the databases located under the specified path. This step ensures that the required databases are correctly configured and ready for further operation of the application.

After that, the core starts the server, using the defined routes to process the requests. The routes serve as an interface between the frontend and the backend, and the core uses them to process the received requests and generate the appropriate responses.

4.8 Testing

Backend testing took a number of different approaches to ensure comprehensive test coverage and guarantee the quality of the developed functions.

For database entity checking, generators were implemented using the test.check [Hic21] library for Clojure. These generators allow the creation of stub data that can be used to create test data. By using these generators, both the database entities and the schema could be successfully tested by generating a very large amount of random data with the particular schema, applying the functions and then checking if the functions produce a correct output.

To perform tests related to xml file import, a special generator was developed that generates an xml file with LSF identical export structure with random data. This generator made it possible to achieve good test coverage when parsing data from xml files. By using this generator, different scenarios could be simulated and the robustness of the parsing process could be verified.

The database queries were also extensively tested. For these tests, an in-memory database was used in some cases to simulate the database queries and ensure that they functioned correctly. These tests ensured that the database queries worked without errors and efficiently.

Middleware components were also thoroughly tested to ensure that the database configuration and database connection were established securely under various scenarios.

The routes tests represent the integration tests, as here the entire flow chain with coercion, middlewares and database is run through. These tests make it possible to check the correct functioning of the backend under real conditions and ensure that all components smoothly work together.

These comprehensive testing strategies ensured that the backend works reliably, error-free and meets the requirements.

4.9 Backend summary

The backend was carefully planned, developed and tested to create a reliable and powerful infrastructure. From the implementation of the database and the processing of xml data to the implementation of the various middleware components and routes, all the necessary steps were taken to ensure the functionality and stability of the backend.

Altogether, the backend of the web web application provides a solid basis for the efficient management of appointment bookings in the 25.xx building. It ensures effective data processing, supports the identification of conflicts and provides a reliable communication interface.

5 Frontend implementation

This chapter provides a detailed insight into the implementation of the web application. Here, the frontend implementation is explained in more detail. The frontend of the application consists of multiple components. These include the core that initializes the frontend, the routes that allow navigation inside the application, events and subscriptions for user interactions, the config for configuring the frontend, the helper for utility functions, translations and initializations, and views that represent different views of the application.

In total, there are 7 components (Figure 5), each of them performing specific tasks. Every one of these components plays an important role in the presentation, navigation and interaction with the web application.

The following sections take a detailed look at the different components and explain their functions as well as how they work together within the frontend.

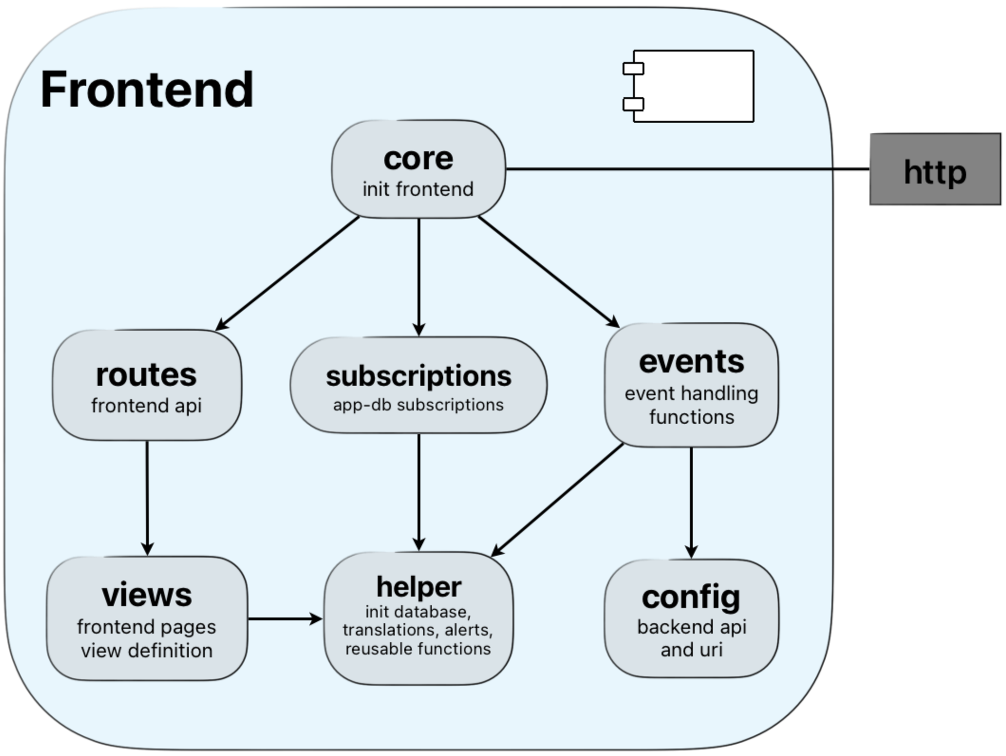


Figure 5: Frontend architecture

5.1 Config

In the configuration file, the url address of the server on which the backend code is executed is specified. This url enables the communication between frontend and backend.

In addition to this, the backend api is also defined in the configuration. This includes the defined endpoints that the frontend can use to access the backend services. The backend api acts as an interface through which the frontend retrieves or sends data.

Managing this information centrally in the config component improves the flexibility and maintainability of the application. When changes are made to the backend URL or the backend api, only the configuration file needs to be modified instead of changing the code in the other parts of the frontend. This also makes it easier to deploy and manage the application in different environments.

5.2 Helper

The helper component plays an important role in managing the default application database, which initializes on every reload.

At the beginning of the frontend development, it was decided to implement the application in two languages to ensure better accessibility. For this reason, translation data is also defined in the helper component. This data makes it possible to provide texts and content in different languages to make the application more user-friendly.

In connection with language support, the helper component also provides a function that gets the currently selected language from the application database and returns the corresponding translation. This allows texts and user interface elements to be dynamically adapted based on the selected language.

In addition, the error messages that are displayed when the communication with the backend was not successful are also defined here. These error messages allow the user to be informed about possible problems during communication or data transfer.

There are still a number of helper functions that are used to abstract and centralize certain recurring tasks or functionalities. This simplifies the code in the individual components and improves reusability. The helper functions can include, for example, mathematical calculations, data formatting, date conversions, or other general utility functions.

Another advantage of the helper component is that it enables a consistent and standardized approach to certain tasks. By centralizing these functions, code maintainability is improved and inconsistencies are avoided.

5.3 Events

The events component plays a central role in triggering various actions in the application. It not only enables initializing the application database and retrieving/sending data from/to the backend, but also storing data in the local application database and generating temporary links for downloading files such as csv or pdf. Retrieving/sending data from/to the backend is one of the most common tasks in an application. By defining appropriate events, data can be exchanged with the backend to retrieve current information or send updates.

Another important aspect of the events component is the implementation of drag-and-drop functionality. When an “on-drag” event is triggered, information such as the day of the week, start time, and end time of the event is required. When the “on-drop” event is triggered, the data is processed and the new time information with the updated start and end times is assigned. The new times are calculated based on the newly assigned time block, with the difference added.

For user input, text fields are available. Normally, each time an input is made, the data is processed immediately, resulting in a re-rendering of the page and a dynamic update of the data. To reduce the load, a debouncer is also integrated into the events component to improve the performance of input into the application. The debouncer provides a small delay and collects the entered letters before they are stored in the application database. As a result, the letters are not processed individually, but in bundles.

The events component keeps these functions centralized and organized for efficient and consistent data processing and interaction with the backend. The initialization of the application database ensures that the required data is available and can be provided when the application is started.

Altogether, the events component plays an essential role in defining and managing functions required for interacting with the backend, saving data, and downloading files.

5.4 Subscriptions

The subscriptions component acts as an intermediary between the application database and the other components in the frontend. It ensures that the data is in a consistent format suitable for use in the various parts of the application. This includes formatting, filtering, sorting, or aggregating the data to ensure that it meets the requirements of the specific components.

By providing the data in this standardized way, the subscriptions component improves the efficiency and reusability of the code. The other components can access the provided data and use it directly without having to perform additional data processing steps. This simplifies the implementation of new functions and makes testing easier, since the data is available in a predefined and familiar format.

The subscriptions component helps to achieve a clear separation of data access and manipulation from the other parts of the application.

5.5 Views

The views (Figure 6) component describes the views that the frontend provides. Views consists of seven subcomponents, six of which are views and a helper for the reusable templates.

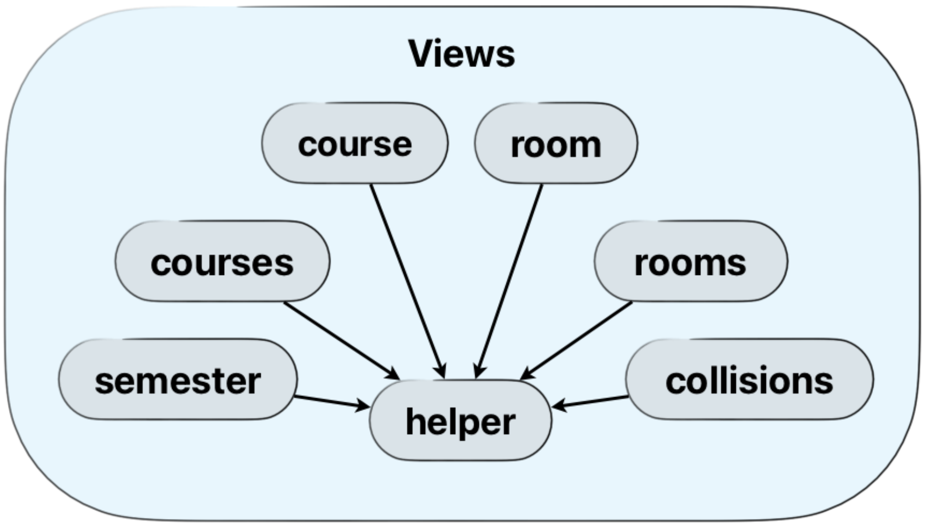


Figure 6: Frontend views

5.5.1 Views helper

This part includes reusable view functions that are used in different views of the application. For example, templates are implemented here to create certain elements such as tables, modals, navigation bar, dropdown menus and accordions.

In addition, some utility functions are developed here to make repetitive tasks easier. To improve the user experience, modal dialogs for each action have been implemented to focus on the essentials.

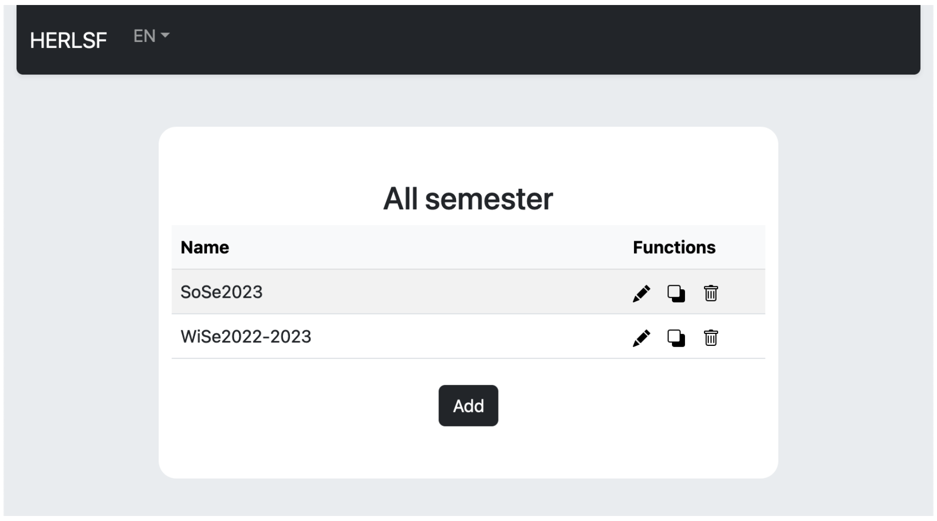
5.5.2 Semester view

This view (Figure 7) serves as the main page of the app and enables the management of semesters. It allows the user to create new semesters, rename, duplicate and delete existing semesters. From the semester view, the user can access the courses overview.

5.5.3 Courses view

In this view 8, all courses are clearly displayed in a responsive table. It contains important functions that are defined as the foundation for the tool:

- Here is the possibility to import data from an xml file.



All semester

Name	Functions
SoSe2023	
WiSe2022-2023	

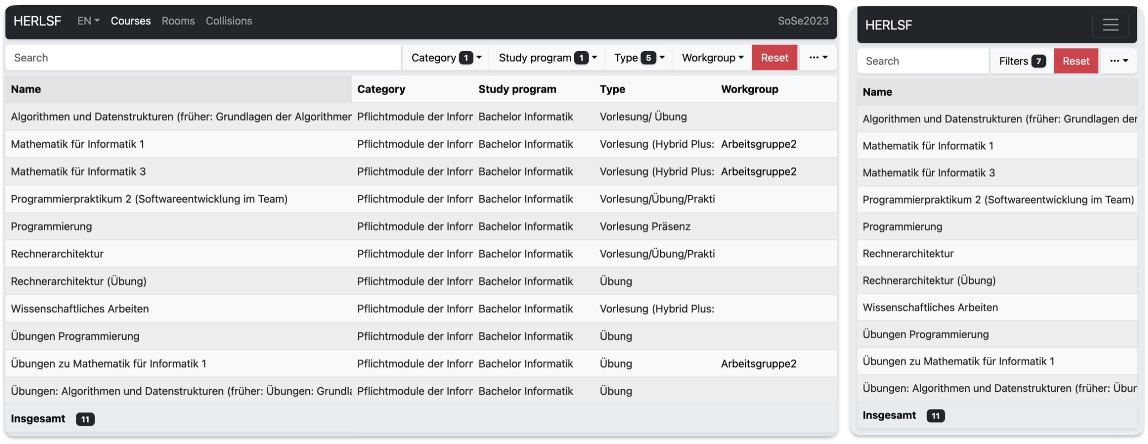
Figure 7: Semester view

- Sorting and filtering functions are available to organize the courses according to specific criteria.
- The search function allows to search for course names and lecturers (last name and first name).
- In addition to the course overview, the management of workgroups is also implemented here. New workgroups can be created and existing ones can be deleted.
- The integration of search and filter functions allows users to search for specific courses and assign them to a workgroup.
- Another usability enhancement is a checkbox that makes it possible to select all displayed events and assign them to a workgroup with just one click. This simplifies the mass assignment of events and optimizes the management process. To avoid accidentally assigning events to a workgroup that already has an assignment, the events are processed accordingly.

5.5.4 Course view

This view displays a single course with all associated events. Here can be found a number of useful functions and data for the management of bookings:

- The attributes of the course that represent filters are clickable to filter the courses accordingly in the courses overview.



Name	Category	Study program	Type	Workgroup
Algorithmen und Datenstrukturen (früher: Grundlagen der Algorithmen	Pflichtmodule der Inforr	Bachelor Informatik	Vorlesung/ Übung	
Mathematik für Informatik 1	Pflichtmodule der Inforr	Bachelor Informatik	Vorlesung (Hybrid Plus:	Arbeitsgruppe2
Mathematik für Informatik 3	Pflichtmodule der Inforr	Bachelor Informatik	Vorlesung (Hybrid Plus:	Arbeitsgruppe2
Programmierpraktikum 2 (Softwareentwicklung im Team)	Pflichtmodule der Inforr	Bachelor Informatik	Vorlesung/Übung/Prakti	
Programmierung	Pflichtmodule der Inforr	Bachelor Informatik	Vorlesung Präsenz	
Rechnerarchitektur	Pflichtmodule der Inforr	Bachelor Informatik	Vorlesung/Übung/Prakti	
Rechnerarchitektur (Übung)	Pflichtmodule der Inforr	Bachelor Informatik	Übung	
Wissenschaftliches Arbeiten	Pflichtmodule der Inforr	Bachelor Informatik	Vorlesung (Hybrid Plus:	
Übungen Programmierung	Pflichtmodule der Inforr	Bachelor Informatik	Übung	
Übungen zu Mathematik für Informatik 1	Pflichtmodule der Inforr	Bachelor Informatik	Übung	Arbeitsgruppe2
Übungen: Algorithmen und Datenstrukturen (früher: Übungen: Grundla	Pflichtmodule der Inforr	Bachelor Informatik	Übung	
Insgesamt 11				

Name
Algorithmen und Datenstrukturen (früher: Grundlagen der
Mathematik für Informatik 1
Mathematik für Informatik 3
Programmierpraktikum 2 (Softwareentwicklung im Team)
Programmierung
Rechnerarchitektur
Rechnerarchitektur (Übung)
Wissenschaftliches Arbeiten
Übungen Programmierung
Übungen zu Mathematik für Informatik 1
Übungen: Algorithmen und Datenstrukturen (früher: Übur
Insgesamt 11

Figure 8: Courses view with mobile version

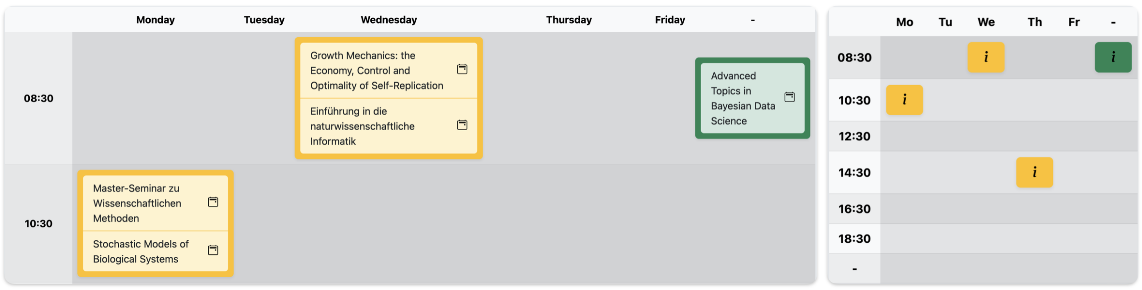
- In addition, the participating lecturers are displayed, as well as information on their other courses.
- In this view, there is also a possibility to define or remove very quickly the rooms tags for the course.
- Each individual event has a direct link to the corresponding room.
- There are two different representations of events: a table with a drag-and-drop function (similar to Figure 9) to quickly move the events, and a more detailed representation (similar to Figure 10) that shows the events grouped by days of the week.
- The events of the course can be deactivated, new events can be added and the course can be assigned to a workgroup.
- It is possible to download information about the course as a pdf or csv file.

5.5.5 Rooms view

Similar to the courses view (Figure 8), this view displays a table with all rooms and the corresponding information. Search, sort and filter functions are also available here to organize the rooms according to different criteria. In addition, a new room can be added. From the rooms view, the user can access the room overview.

5.5.6 Room view

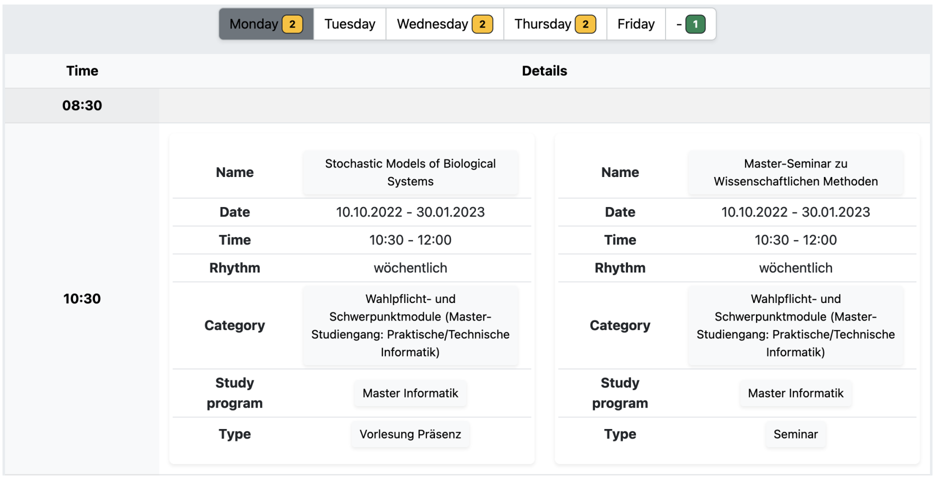
This view displays individual room with information about form and building of the room with all related events.



	Monday	Tuesday	Wednesday	Thursday	Friday	-
08:30			Growth Mechanics: the Economy, Control and Optimality of Self-Replication; Einführung in die naturwissenschaftliche Informatik			Advanced Topics in Bayesian Data Science
10:30	Master-Seminar zu Wissenschaftlichen Methoden; Stochastic Models of Biological Systems					

	Mo	Tu	We	Th	Fr	-
08:30			i			i
10:30	i					
12:30						
14:30				i		
16:30						
18:30						
-						

Figure 9: Week view responsive table with mobile version



Monday 2 | Tuesday | Wednesday 2 | Thursday 2 | Friday | - 1

Time	Details	
08:30		
10:30		

	Stochastic Models of Biological Systems	Master-Seminar zu Wissenschaftlichen Methoden
Name	Stochastic Models of Biological Systems	Master-Seminar zu Wissenschaftlichen Methoden
Date	10.10.2022 - 30.01.2023	10.10.2022 - 30.01.2023
Time	10:30 - 12:00	10:30 - 12:00
Rhythm	wöchentlich	wöchentlich
Category	Wahlpflicht- und Schwerpunktmodule (Master-Studiengang: Praktische/Technische Informatik)	Wahlpflicht- und Schwerpunktmodule (Master-Studiengang: Praktische/Technische Informatik)
Study program	Master Informatik	Master Informatik
Type	Vorlesung Präsenz	Seminar

Figure 10: Weekday view responsive table

- The attributes of the room that represent filters are clickable to filter the rooms accordingly in the rooms overview.
- Information with links to the courses where the room is tagged is also available here.
- Each individual event has a direct link to the corresponding course.
- There are also, as in the course view, two representations of events: a table (Figur 9) with a drag-and-drop function, and a more detailed representation (Figur 10) of events grouped by days of the week.
- It is possible to download information about the room as a pdf or csv file.

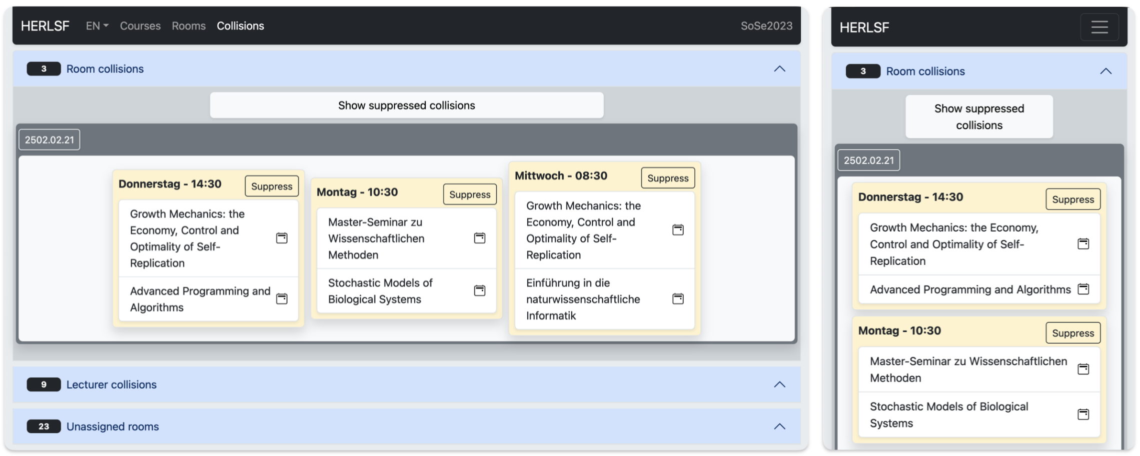

HERLSF
EN
Courses
Rooms
Collisions
SoSe2023
3 Room collisions
Show suppressed collisions
2502.02.21
Donnerstag - 14:30
Suppress
Growth Mechanics: the Economy, Control and Optimality of Self-Replication
Advanced Programming and Algorithms
Montag - 10:30
Suppress
Master-Seminar zu Wissenschaftlichen Methoden
Stochastic Models of Biological Systems
Mittwoch - 08:30
Suppress
Growth Mechanics: the Economy, Control and Optimality of Self-Replication
Einführung in die naturwissenschaftliche Informatik
9 Lecturer collisions
23 Unassigned rooms
HERLSF
3 Room collisions
Show suppressed collisions
2502.02.21
Donnerstag - 14:30
Suppress
Growth Mechanics: the Economy, Control and Optimality of Self-Replication
Advanced Programming and Algorithms
Montag - 10:30
Suppress
Master-Seminar zu Wissenschaftlichen Methoden
Stochastic Models of Biological Systems

Figure 11: Collisions view with mobile version

5.5.7 Collisions view

This view (Figur 11) shows conflicts found in the bookings. Different types of conflicts are distinguished, such as room conflicts, lecturer conflicts and events with undefined fields that are not included in the conflict set. Additionally, an overview with disabled events is provided. Room and lecturer conflicts can be suppressed so that they are no longer displayed. However, a button is available to display suppressed conflicts again. It provides direct links to rooms, courses, events and information about the lecturers.

5.6 Routes

The routes component is an integral part of the frontend and plays an important role in navigation within the application. It is responsible for triggering the right events to get all the relevant data from the backend that is needed for the specific page or view. Routes define which components and views should be loaded for specific path.

In the frontend, a total of six different views are provided to cover various aspects of booking management. The views include the management of semesters, courses, a single course, rooms, a single room as well as an overview page showing existing collisions in the bookings.

By effectively designing the routes and triggering the corresponding views, seamless navigation and interaction within the application is ensured.

5.7 Core

The core component is a central part of the frontend that performs several important tasks. The core initializes the initial application database and reads cookies stored in the browser for language and events representation by calling corresponding events. By reading cookies, the frontend can access previously stored information and use it in the application, this allows a better user experience. In addition, Core also initializes Routes to control navigation and data flow in the frontend. Finally, the core is also responsible for rendering the index.html page. The index.html is the main file with a "<div>" container with an ID that acts as a base to inject scripts into the html. The core takes care of loading this file.

5.8 Testing

Frontend testing was performed with the aim of achieving comprehensive test coverage and ensuring the correct functioning of the developed functions.

The frontend has been designed to use subscriptions and events to process data. These features have been carefully tested to ensure that they work properly and deliver the expected results. For subscriptions, the logic was extracted into the functions, allowing to test all the data that is processed before landing on the web page and ensure that everything is passed in the correct format. The views on the frontend receive the processed data in the correct format and there is no specific logic within the views that needs to be tested. Instead, the views fire events to trigger specific actions on the frontend.

For full test coverage and functionality verification of the frontend it was decided to use the Cypress [Cyp23] library. Using this library, tests were written to test the frontend for all implemented functionality. The Cypress library provides the ability to intercept and respond to backend requests without the need to start an actual backend server or use a separate mock server. The tests check that all elements of a view are loaded, the data coming from the backend response is processed and displayed correctly, and all links and buttons work. It also tests all events that can be retrieved from the views. Among them are also the storage and retrieval of stored cookies as well as the request bodies that are sent to the backend. This allows for efficient and reliable testing in the frontend.

By using the Cypress library, different scenarios and user interactions could be simulated to ensure that the frontend responds properly to user input and delivers the expected results. This also confirms the robustness of the frontend, as almost no delays are used to wait until all calculations on the page are finished.

The tests include checking data processing, testing user actions, and ensuring smooth interaction between the different components of the frontend.

Thanks to this comprehensive testing strategy, it was possible to ensure that the frontend of the web application works properly and meets the requirements.

5.9 Frontend summary

The frontend of the application is made up of multiple components that work together to create an attractive and user-friendly interface.

With these components working closely together, the frontend can retrieve, process, and display data from the backend, handle user interactions, and make the application seamless and user-friendly. Each component performs a specific task and contributes to the overall functionality and aesthetics of the front end.

Altogether, the frontend of the application provides an intuitive user interface that allows the user to operate the application smoothly and perform its tasks effectively. The combination of thoughtful design, efficient functionality and high usability contributes to a positive and satisfying user experience.

6 Summary

This work gave me valuable insights. Especially the development of the frontend was a new and fascinating experience for me. Being able to dynamically design and understand the user interface enabled performance optimization and created a solid foundation for a user-friendly application. Furthermore, a deeper understanding of functional programming was achieved and the handling of immutable data was improved.

The development of the frontend views was a great experience. With hiccup notation it is possible to write html code in a very practical way. Using the flexible and intuitive Datomic database queries made it easier to find the relevant information. In addition, defining the interfaces for communication between the frontend and backend proved successful. Feedback and support were crucial to ensure that the developed solution met the requirements.

Setting up the development environment was somewhat difficult, as there were not many libraries for Clojure that could be integrated quickly and some available libraries were already outdated. The stability of the database while switching between semesters as well as the configuration of the database required special attention. Furthermore, the handling of Coercion on both sides posed an additional challenge.

In terms of trade-offs, some important decisions were made to ensure the functionality and efficiency of the tool. The use of separate databases for changing the semester context allowed for unique assignment of events. The decision to override the “scope” of the database configuration and always store “localhost” as the scope ensured stable database usage. Storing negative IDs for adding events in the tool allowed correct handling of existing events in the xml data.

The definition of conflicts and their suppression required careful consideration. Due to the specific requirements of the tool, an event can only be considered a conflict if all required information (time, date, day of the week, and room) is complete.

In addition, implementing a debouncer allowed the application to perform better by collecting input in input fields and only then processing it to avoid frequent re-renderings.

Overall, the developed solution presents a powerful tool for planning and managing bookings. It meets the requirements of efficient use of resources and enables the consideration of data created in the tool and not recorded in the central booking system. The lessons learned about frontend development, functional programming and handling immutable data represent valuable experience that will also be useful in future projects.

The complete code can be found in the following repository:

`https://gitlab.cs.uni-duesseldorf.de/stups/abschlussarbeiten/cabacov-bachelor`

7 Evaluation

A detailed evaluation of the developed solution was carried out to assess its performance, reliability and user-friendliness. Different aspects of the application were examined to ensure that all requirements were met.

In terms of performance and reliability, extensive testing was carried out to ensure that the application runs smoothly. Both backend and frontend functions were checked to ensure that they function without errors under different scenarios. Measures were also taken to ensure data integrity and minimize potential sources of errors. Database queries were optimized to ensure efficient processing of large data volumes.

The usability of the application was ensured by an intuitive user interface and clear interaction options. The design was carefully created to make the navigation and operation of the application simple and user-friendly.

Altogether, the developed solution has shown that it meets the requirements of booking management in the 25.xx building. The application's performance, reliability and usability provide an efficient and effective way to manage events, identify conflicts and make bookings.

By implementing a user-friendly interface and taking specific requirements into consideration, it was possible to develop a customized solution that optimally meets the needs of the application domain.

8 Outlook for future extensions

In terms of future developments, there are several ideas to further expand and improve the tool for planning and managing bookings.

One possible enhancement is to introduce a feature to highlight all newly added data after a new xml import. This allows users to quickly see what information has been updated or newly added.

Another interesting option is to display the free available blocks in the rooms while editing or creating a new event. This allows the user to more easily identify available time slots for their events and identify possible conflicts early on.

In addition, an alternative display of collisions can be considered to improve the visual representation of conflicts. This could provide a clearer and more intuitive display to help the user better understand conflicts and find appropriate solutions.

Another possibility for future enhancements is the implementation of automatic conflict resolution suggestions, which could be done by a constraint solver, as in the PlueS [DS18] project. Based on the data collected and the information available, the tool could make intelligent suggestions for resolving conflicts, making it easier for the user and enabling more efficient planning.

These mentioned ideas represent potential extensions to further develop the tool and provide even more functionality and comfort to the user. By implementing such extensions, the tool can be continuously improved and adapted to the needs of the users.

List of Figures

References

[Boo23] Bootstrap. Bootstrap. https://getbootstrap.com/docs/5.3/getting-started/introduction/, 2023. [Online; Accessed 2023-07-01].

[clj22] cljfx. cljfx. https://github.com/cljfx/cljfx, 2022. [Online; Accessed 2023-07-01].

[cp23] clj pdf. Dmitri sotnikov. https://github.com/clj-pdf/clj-pdf, 2023. [Online; Accessed 2023-07-01].

[Cyp23] Cypress. Cypress. https://www.cypress.io/, 2023. [Online; Accessed 2023-07-01].

[DS18] Tobias Witt David Schneider, Michael Leuschel. Model-based problem solving for university timetable validation and improvement. https://dl.acm.org/doi/10.1007/s00165-018-0461-7, 2018. [Online; Accessed 2023-07-01].

[Hic21] Rich Hickey. test.check. https://github.com/clojure/test.check, 2021. [Online; Accessed 2023-07-01].

[Hic23a] Rich Hickey. Clojure. https://clojure.org/, 2023. [Online; Accessed 2023-07-01].

[Hic23b] Rich Hickey. Clojurescript. https://clojurescript.org/, 2023. [Online; Accessed 2023-07-01].

[Lab22] Red Planet Labs. Specter. https://github.com/redplanetlabs/specter, 2022. [Online; Accessed 2023-07-01].

[LTD23] JUXT LTD. bidi. https://github.com/juxt/bidi, 2023. [Online; Accessed 2023-07-01].

[lU21] lambdaforge UG. datahike. https://datahike.io/, 2021. [Online; Accessed 2023-07-01].

[met23] meta. React. https://react.dev/, 2023. [Online; Accessed 2023-07-01].

[Oy23] Metosin Oy. reitit. https://cljdoc.org/d/metosin/reitit/0.6.0/doc/introduction, 2023. [Online; Accessed 2023-07-01].

[Ree22] James Reeves. Compojure. https://github.com/weavejester/compojure, 2022. [Online; Accessed 2023-07-01].

[Ree23] James Reeves. hiccup. https://github.com/weavejester/hiccup, 2023. [Online; Accessed 2023-07-01].

[Rod22] Craig Rodway. classroombookings. https://www.classroombookings.com/, 2022. [Online; Accessed 2023-07-01].

[rp23] reagent project. reagent. http://reagent-project.github.io/, 2023. [Online; Accessed 2023-07-01].

[Tho22] Michael Thompson. re-frame. https://day8.github.io/re-frame/re-frame/, 2022. [Online; Accessed 2023-07-01].