Praxis der Softwareentwicklung

Solidarische Raumnutzung

Praxis der Softwareentwicklung (PSE)
Wintersemester 2024/25

Entwurfsheft

Karlsruher Institut für Technologie (KIT)
Institut für Anthropomatik und Robotik (IAR)
Forschungsgruppe Mensch-Maschine-Interaktion und Barrierefreiheit (MBI)
Prof. Dr. Kathrin Gerling
Adenauerring 10, Gebäude 50.28
76131 Karlsruhe

Betreuer: Sabrina Burtscher, Dmitry Alexandrovsky

Projektteilnehmer:

Name	E-Mail-Adresse
Antonia Ammon	uipkm@student.kit.edu
Ben Steinle	ufikl@student.kit.edu
Johannes Frohnmeyer	uczkf@student.kit.edu
Alexander Klee	ukvpq@student.kit.edu
Jannik Hönlinger	uouyo@student.kit.edu

Karlsruhe, 26. März 2025

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Wie schon im zugehörigen Pflichtenheft erklärt, ist das Ziel dieses Projekts, eine Webanwendung *Soli* zur solidarischen Raumnutzung zu entwickeln. Um die im Pflichtenheft spezifizierten Anforderungen zu erfüllen, werden zwei Bestandteile entworfen: die entsprechenden HTML-Seiten im Frontend und die dazugehörige Logik im Backend. Gegenstand dieses Entwurfshefts ist die genaue Beschreibung der Systemarchitektur und der *Views* im Frontend. Dabei schließen wir ohne Änderungen an das Pflichtenheft an. Darüber hinaus werden wir einige Datenstrukturen definieren, die die Speicherung der Daten im Backend beschreiben.

2 Frameworks und Libaries

Für das Produkt *Soli* werden unterschiedliche Frameworks und Libraries verwendet, um die Entwicklung zu erleichtern und die Qualität des Codes zu erhöhen.

Im Backend wird das Spring Boot Framework verwendet. Zur Datenpersistenz wird PostgreSQL über HikariCP mittels Spring Data JPA verwendet. Datenbank Migrationen werden mit Flyway durchgeführt. Zum Verschicken von E-Mails wird Spring Boot Mail verwendet. Die Authentifizierung und Sicherheit wird mit Spring Boot Security realisiert, der KIT-Login wird mit dem OAuth2-Client von Spring Boot realisiert. Außerdem werden statische HTML-Seiten mit der Java Template Engine (JTE) generiert.

Im Frontend wird das CSS-Framework DaisyUI in Kombination mit TailwindCSS zur Gestaltung der Oberfläche verwendet. Zur Umsetzung eines gut integrierten und geprüften Kalenders wird die JavaScript-Bibliothek FullCalendar verwendet.

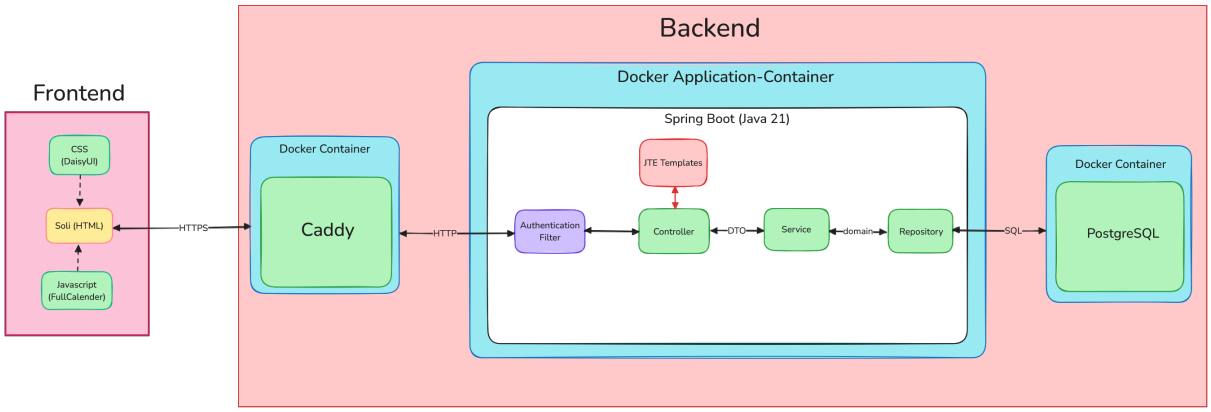
Zum Testen der Anwendung wird JUnit eingesetzt. Zum Mocken von Objekten wird Mockito verwendet.

Als Werkzeug zur Build-Automatisierung wird Gradle verwendet.

Zudem wird Docker verwendet, um die Anwendung und ihre Abhängigkeiten in Containern zu betreiben.

3 Aufbau

3.1 Architektur

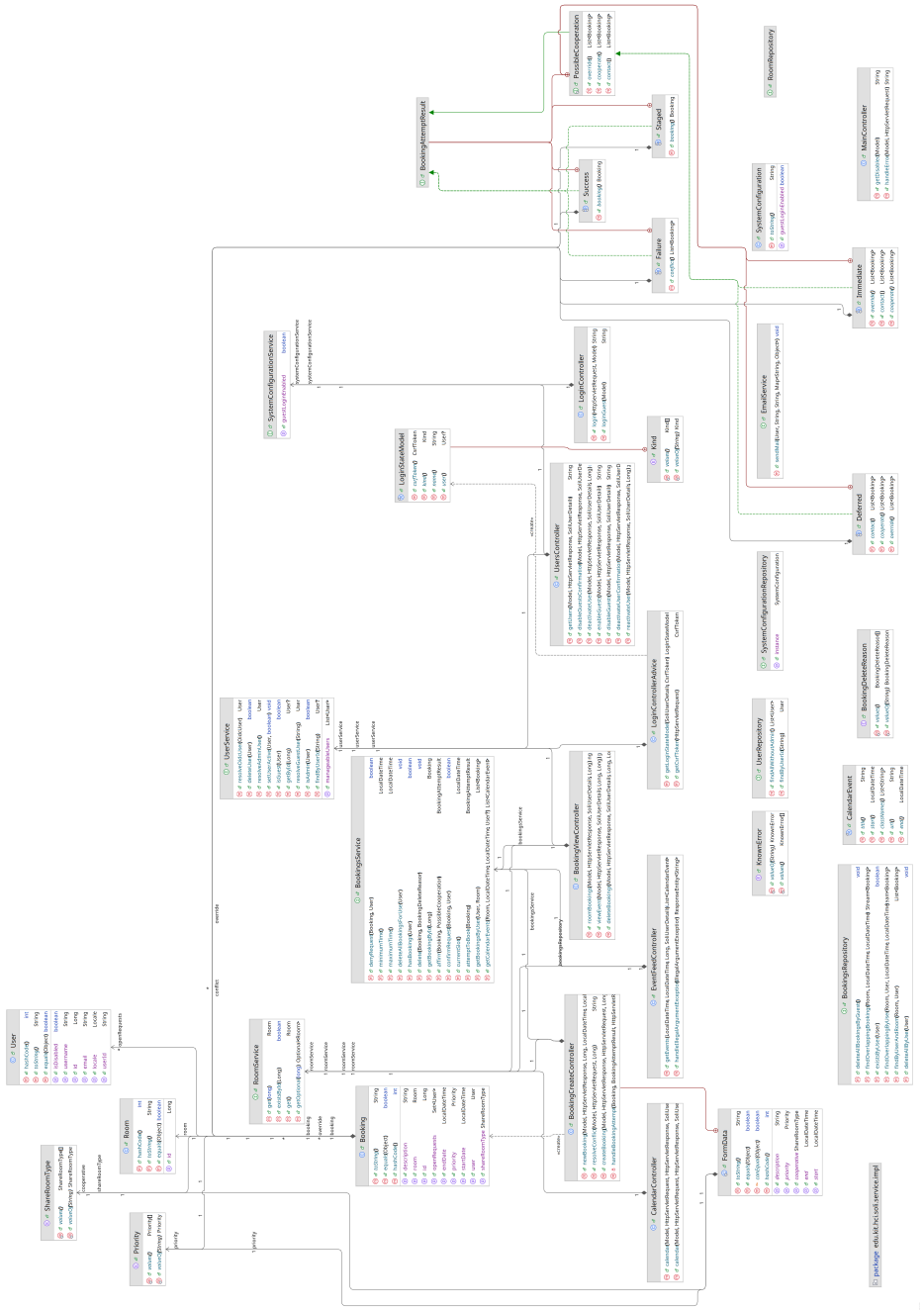


Die Architektur ist in *Frontend* und *Backend* unterteilt und folgt der MVC-Struktur zur klaren Trennung der Verantwortlichkeitsbereiche. Das *Frontend* nutzt HTML, DaisyUI für das Design und FullCalendar in JavaScript für die Kalenderfunktionalität. Die Kommunikation mit dem Backend erfolgt über HTTPS.

Ein Caddy-Server, welcher in einem eigenen Container läuft, dient als HTTPS-Reverse-Proxy und leitet Anfragen an das Backend weiter. Das *Backend* basiert auf Spring Boot (Java 21) und läuft in einem weiteren Container. Dort prüft ein *Authentication Filter* die Anfragen, die danach von einem Controller verarbeitet werden. Mithilfe von JTE-Templates werden dynamische HTML-Antworten generiert. Die *Businesslogik* liegt in den Services, die Anfragen koordinieren und weiterleiten. Die *Repositories* kapseln den SQL-Zugriff und ermöglichen eine saubere, abstrahierte Kommunikation mit der PostgreSQL-Datenbank in einem separaten Container.

Der Einsatz von Docker sorgt für eine einfache Bereitstellung und Wartbarkeit der Anwendung.

3.2 Klassendiagramm



4 View

4.1 Haupseite und Buchung

Die Startseite der Anwendung ist in 4.1 dargestellt. Auf dieser haben Nutzende die Möglichkeit, den Raum zu buchen oder auf andere Ansichten zu wechseln. Das Banner in allen Ansichten zeigt den aktuellen Raumstatus an. Insbesondere wird dadurch die Priorität des aktuellen Termins mithilfe der Farbe des Banners angezeigt.

Rechts in dieser Ansicht befindet sich ein Kalender, der die aktuelle Woche und die Termine in diesem Zeitraum anzeigt. Die Termine sind farblich nach Priorität gekennzeichnet, weitere Informationen können durch Klicken auf den Termin eingesehen werden. Neben dieser Möglichkeit, den Raum zu buchen, gibt es je nach Anmeldungsstatus einen Anmelde- oder Abmeldebutton und einen separaten Buchungsbutton, für den die Verwendung des Kalenders Probleme bereitet. Falls zur Zeit der Verwendung ein Termin des Nutzenden stattfindet, wird ein Quick-Checkout-Button angezeigt, der es ermöglicht, den Raum schnell freizugeben. Siehe dazu auch 4.4.

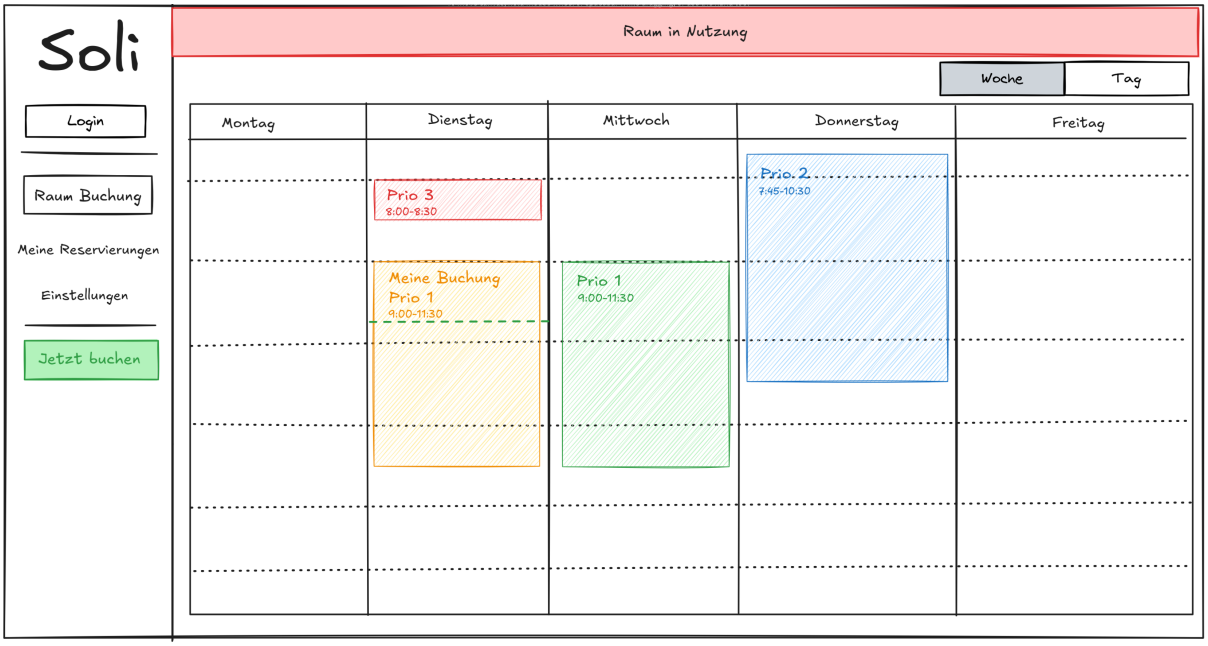


Abbildung 4.1: Startseite der Anwendung

Sollten Nutzende eine Buchung vornehmen wollen, so klicken diese in den gewünschten Zeitraum und es wird der Dialog in 4.2 dargestellt.

Der Dialog bietet Nutzenden die Möglichkeit, den genauen Start- und Endzeitpunkt des Termins festzulegen.

Außerdem können Nutzende die Priorität des Termins in Form einer Zahl zwischen 1 und 3 festlegen. Nutzende können auch angeben, ob sie bereit sind, den Raum mit anderen Nutzenden zu teilen. Für diesen Zweck werden Ihnen drei Optionen bereitgestellt: *Ja*, *Nein* und *Auf Anfrage*. Siehe dazu auch 4.2.

Letztlich können Nutzende eine Beschreibung für den Termin hinterlegen, die anderen angemeldeten Nutzenden angezeigt wird.

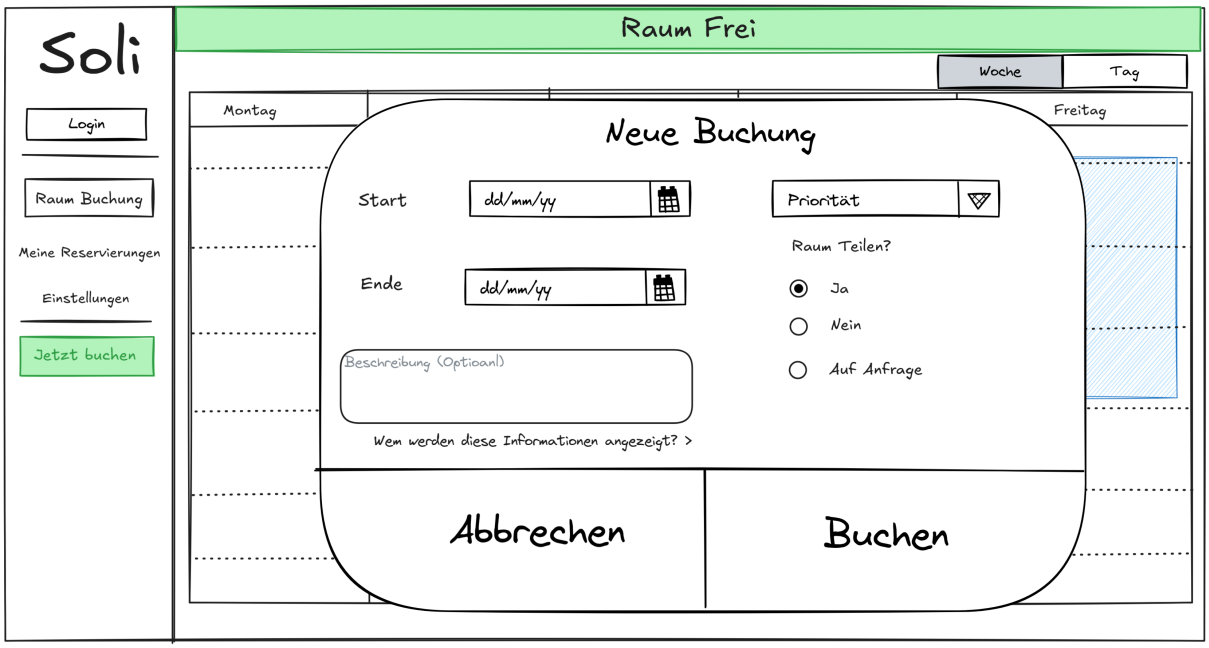


Abbildung 4.2: Termin-erstellen

Tätigen Nutzende eine Buchung, so werden diese aufgefordert, sich anzumelden. Der hierzu gehörige Dialog ist in 4.3 dargestellt. Alternativ ist dieser Dialog auch über den Anmeldungsbutton, der in Abbildung 4.1 zu sehen ist, erreichbar.

In diesem Dialog wird Nutzenden die Möglichkeit gegeben, sich mit ihrem KIT-Konto, mit einem lokalen Gastkonto oder als Admin anzumelden.

Falls Nutzende die Anmeldung per KIT-Konto wählen, werden sie auf die KIT-Login-Seite weitergeleitet. Von dort aus können sie sich mit ihren KIT-Zugangsdaten anmelden.

Falls Nutzende die Anmeldung per Gastkonto wählen, werden sie aufgefordert, eine E-Mail-Adresse anzugeben. Mit der Bestätigung dieser E-Mail-Adresse wird ein temporäres Gastkonto erstellt und die Anmeldung per Cookie gespeichert.

Falls Nutzende die Anmeldung als Admin wählen, werden sie aufgefordert, das Passwort des Adminkontos einzugeben.

Nach einer erfolgreichen Anmeldung werden Nutzende auf die nächste Seite weitergeleitet.

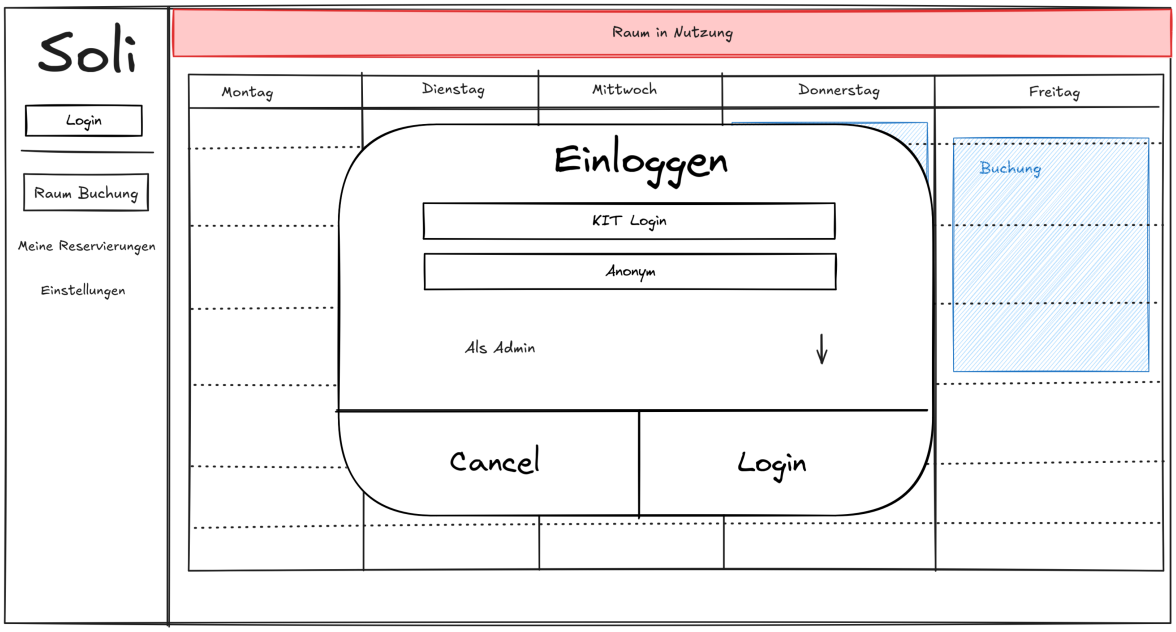


Abbildung 4.3: Anmeldungsseite

Sind Nutzende eingeloggt und belegen den Raum, so wird ihnen die in Abbildung 4.4 dargestellte Ansicht angezeigt. Hier können Nutzende den Raum wieder über den Quick-Checkout-Button freigeben.

Ziel dieser Ansicht ist es, Nutzenden das frühe Freigeben des Raumes ohne unnötigen Mehraufwand zu ermöglichen.

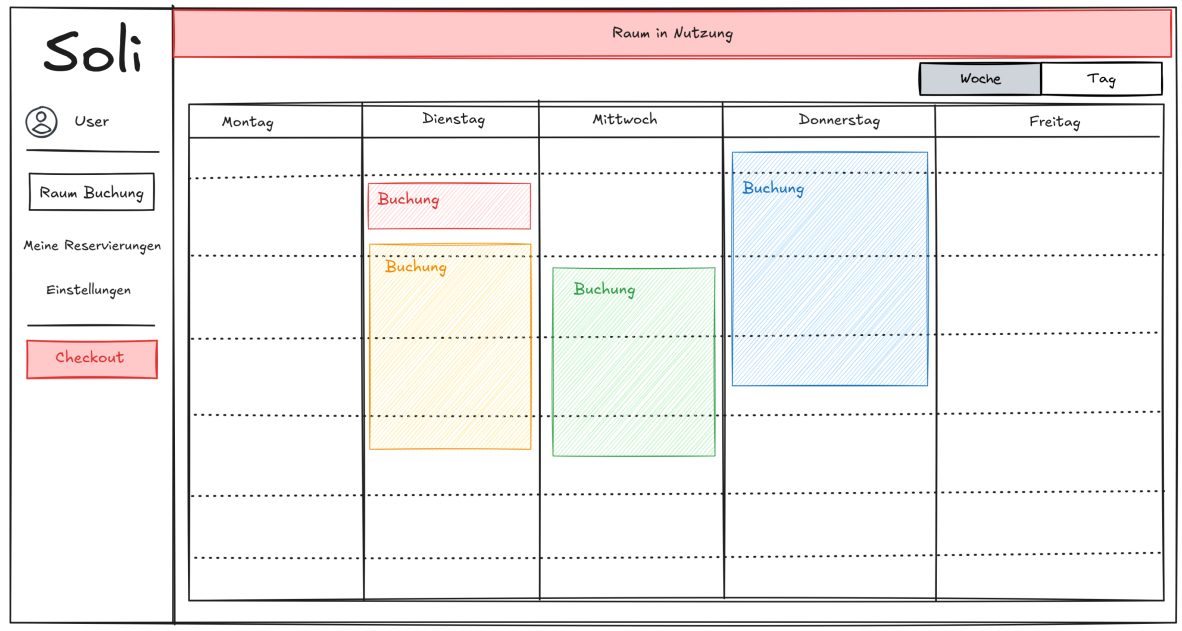


Abbildung 4.4: Quick Checkout

4.2 Terminübersicht

Nutzende, die eine Buchung vorgenommen haben, können diese in der Terminübersicht, die in Abbildung 4.5 dargestellt ist, einsehen und verwalten.

Die Termine in dieser Ansicht werden, anders als in der Ansicht *Kalender*, in Form einer sortierten Liste dargestellt. Dabei werden die Start- und Endzeitpunkte der Termine, wie auch die Priorität und die Beschreibung angezeigt.

Nutzende haben die Möglichkeit, eine Buchung zu stornieren, indem sie auf den Stornieren-Button klicken.

Außerdem können Nutzende die Ansicht *Termin* (4.6) zu einem Termin öffnen, indem sie auf den Termin klicken.

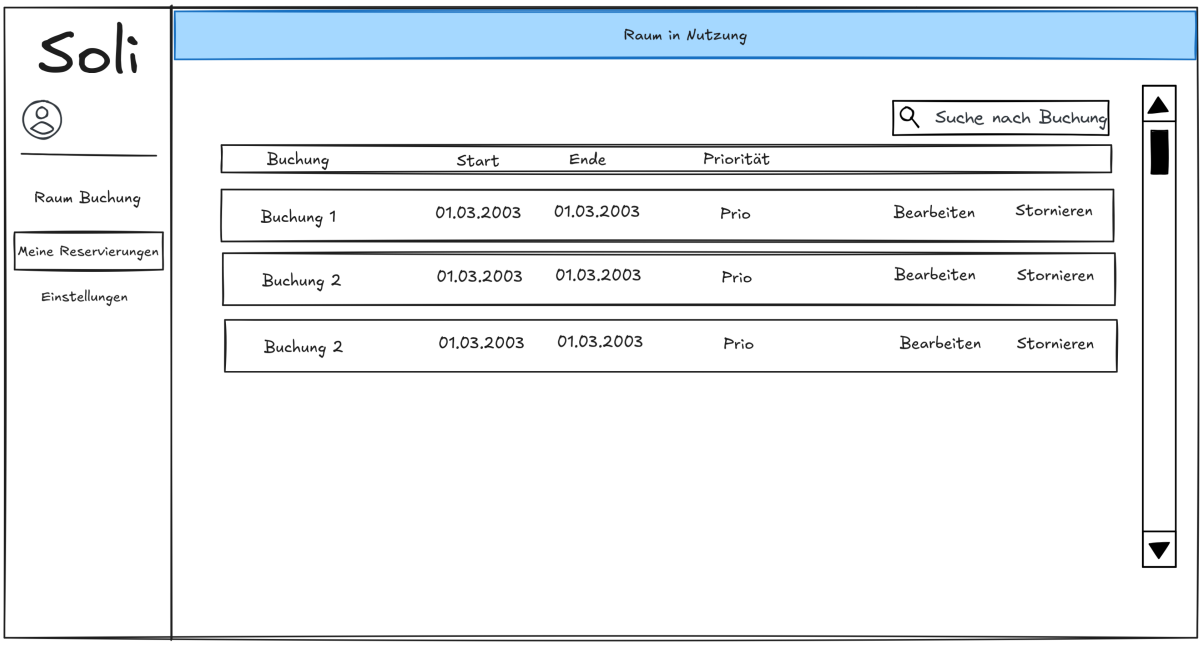


Abbildung 4.5: Reservierungsübersicht

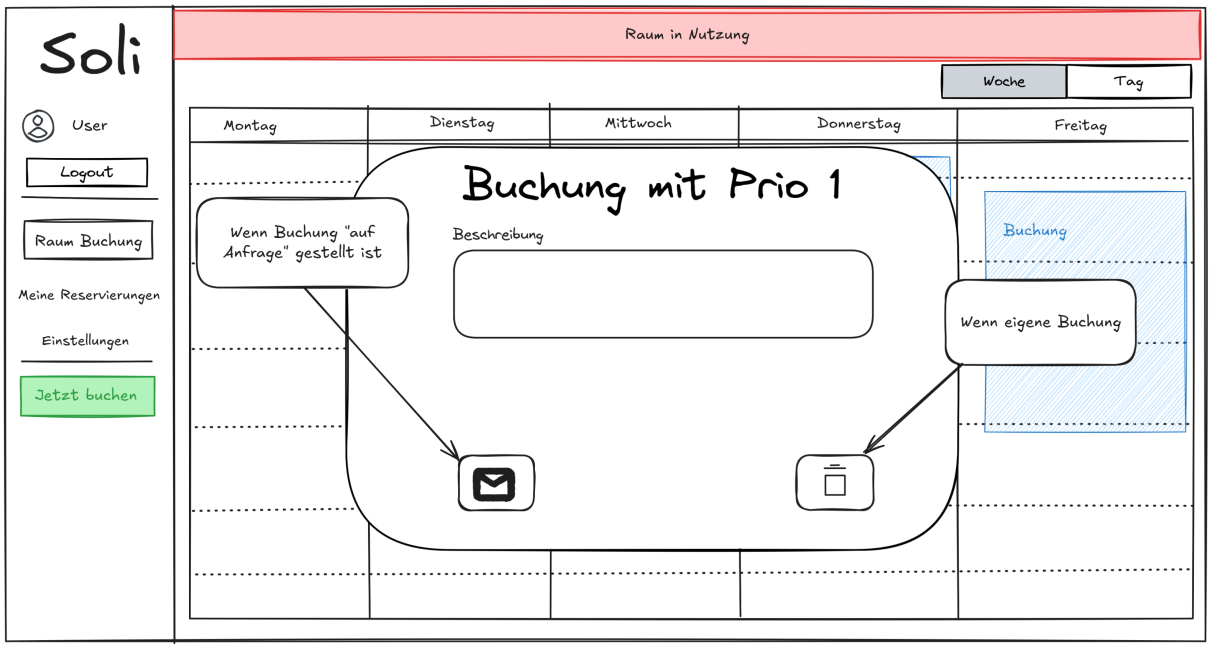


Abbildung 4.6: Reserverierung im Kalender

4.3 Adminstration

Ein/e Administrator*in hat die Möglichkeit, über die Benutzeradministrationsoberfläche, die in Abbildung 4.7 dargestellt ist, Nutzende einzusehen und zu verwalten.

In dieser Ansicht werden alle Nutzenden angezeigt, die sich in der Anwendung registriert haben. Ein Admin kann die Accounts von Nutzenden deaktivieren und somit ihre Anmeldung verhindern.

Um die Suche nach einem bestimmten Konto zu erleichtern, kann ein Admin nach Kontonamen filtern.

Zudem kann ein Admin die Anmeldung per Gastkonto deaktivieren. Zweck dieser Funktion ist es, den Missbrauch von Gastkonten z.B. durch Bots in vorübergehenden Zeiten zu verhindern.

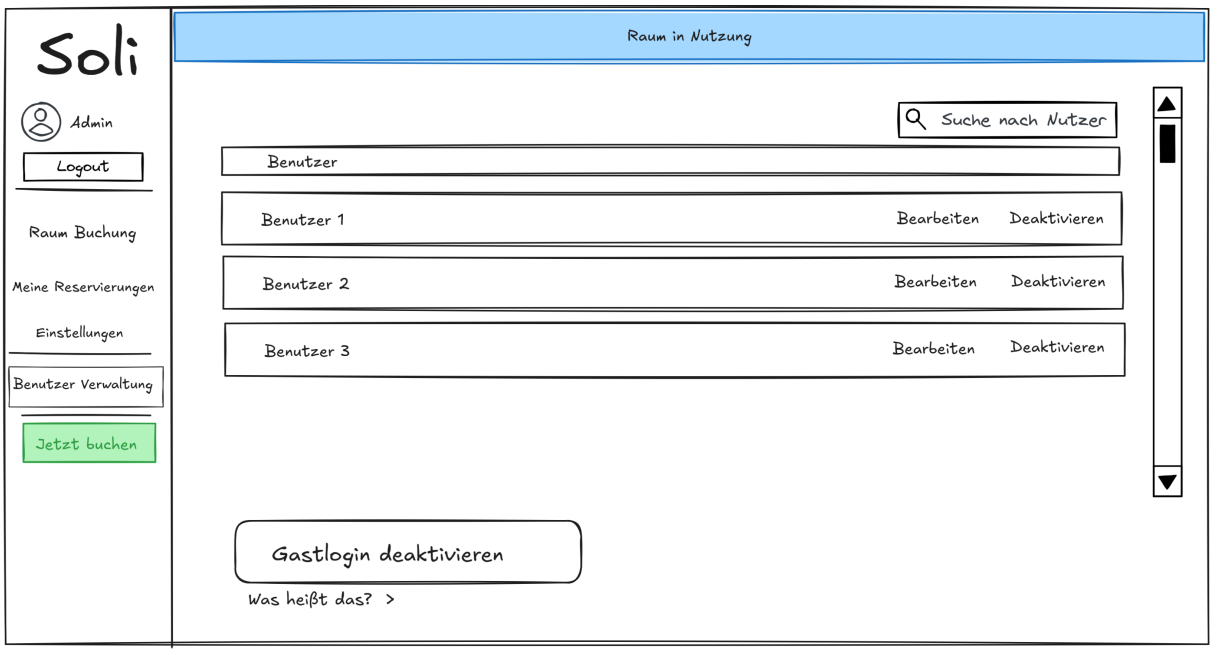


Abbildung 4.7: Benutzeradminstrationsoberfläche

5 Controller

Das Controller-Layer der Anwendung ist für die Verarbeitung von Anfragen und die Erzeugung von Antworten zuständig. Es dekodiert und prüft dabei in HTTP-Anfragen enthaltene Eingaben und generiert Parameter für die JTE-Templates der View. Zur Bearbeitung der überprüften Anfragen werden die Services des Service-Layers genutzt, welche die Datenverarbeitung kapseln. Da einige Endpunkte eine Anmeldung benötigen, werden unangemeldete Nutzende, welche versuchen, mit einem solchen Endpunkt zu interagieren, automatisch auf die Anmeldeseite weitergeleitet. Das Controller-Layer bietet zur Interaktion durch Nutzende die folgenden Endpunkte, welche die HTTP-Oberfläche der Anwendung kapseln und die im Pflichtenheft beschriebenen Ansichten modellieren.

Für die Ansicht *Kalender* (siehe `CalendarController`) werden folgende Endpunkte bereitgestellt:

- `GET /` gibt alle Buchungen für den Standardraum mit der fixierten ID `1` zurück.
- `GET /{roomId}` gibt die Kalenderansicht für den Raum mit der ID `roomId` zurück. Als Implementation für den Kalender wird dabei FullCalendar genutzt.
- `GET /api/{roomId}/events` gibt alle Buchungen für einen Raum im FullCalendar-Format zurück. Dieser Endpunkt wird von FullCalendar genutzt, um die Buchungen im Kalender anzuzeigen. Links zur Ansicht `Termin` werden für die Termine eingebettet. Da es sich bei diesem Endpunkt um einen REST-Endpunkt handelt, ist er separat in `EventFeedController` implementiert.

Für die Ansichten *Termin* und *Terminübersicht* (siehe `BookingViewController`) werden folgende Endpunkte bereitgestellt:

- `GET /{roomId}/bookings` gibt die Terminübersicht des angemeldeten Nutzers für den Raum mit der ID `roomId` zurück.
- `GET /{roomId}/bookings/{eventId}` gibt die Ansicht *Termin* für den Termin `eventId` zurück. Falls ein Administrator oder die Person, welche den Termin erstellte, angemeldet ist, wird eine Option, den Termin zu löschen, als Link eingebettet. Falls der Termin als kollaborativ markiert ist, wird ein Link zur Ansicht *Termin-Erstellen* mit den Start- und Endzeiten des angewählten Termins eingebettet.

- `DELETE /{roomId}/bookings/{eventId}` löscht den Termin `eventId` und leitet Nutzende auf die Terminübersicht weiter.

Für die Ansicht *Termin-Erstellen* (siehe `BookingCreateController`) werden folgende Endpunkte bereitgestellt:

- `GET /{roomId}/bookings/new` gibt das Formular (implementiert als HTML-Form) zur Erstellung eines neuen Termins für den Raum mit der ID `roomId` zurück.
- `POST /{roomId}/bookings/new` erstellt einen neuen Termin für den Raum mit der ID `roomId` und leitet Nutzende auf die Ansicht *Termin* weiter. Die Form-Eingaben des Formulars werden dabei dekodiert und geprüft. Falls ein Konflikt mit einem bereits bestehenden Termin besteht, werden Nutzende auf die Konfliktseite weitergeleitet. In diesem Fall wird der dekodierte Termin als `Booking` in der Session gespeichert.
- `POST /{roomId}/bookings/new/conflict` bietet Nutzenden die Möglichkeit, einen Konflikt entsprechend der Spezifikation im Pflichtenheft zu lösen. Dabei wird eine mögliche Lösung vorgeschlagen, welche Nutzende bestätigen oder ablehnen können. Im Falle der Bestätigung wird der in der Session gespeicherte Termin erstellt und Nutzende auf die Ansicht *Termin* weitergeleitet. War unter den Konflikten ein Termin, welcher Zustimmung zur Kollaboration benötigt, wird entsprechend eine E-Mail an die betroffenen Nutzenden versendet. War unter den Konflikten ein Termin, welcher durch die Erstellung gelöscht wurde, wird entsprechend eine E-Mail an die betroffenen Nutzenden versendet.

Zum akzeptieren von Kollaborationen (siehe `CollaborationRequestController`) werden folgende Endpunkte bereitgestellt:

- `GET /{roomId}/bookings/{eventId}/collaboration` gibt das Formular zur Bestätigung einer Kollaboration für den Termin `eventId` zurück.
- `PUT /{roomId}/bookings/{eventId}/collaboration` akzeptiert die Kollaboration für den Termin `eventId` und leitet Nutzende auf die Ansicht *Termin* weiter.
- `DELETE /{roomId}/bookings/{eventId}/collaboration` lehnt die Kollaboration für den Termin `eventId` ab und leitet Nutzende auf die Ansicht *Terminübersicht* weiter.

Für die Ansicht *Login* (siehe `LoginController`) werden folgende Endpunkte bereitgestellt:

- `GET /login` gibt das Formular zur Anmeldung zurück. Die Anmeldung kann mit einem eingebetteten Formular als Administrator oder mit einem Link als Gast oder über OIDC erfolgen. Für die Anmeldung als Administrator wird eine HTML-Form genutzt, welche die Eingabe eines Passworts fordert.

- `POST /login` erlaubt die Anmeldung mit lokalen Anmeldedaten (also als Gast oder Administrator) und leitet Nutzende auf die Ansicht, welche die Anmeldung verursachte, oder die Ansicht *Kalender* weiter. Dieser Endpunkt wird bei Bestätigung einer Gast- oder Administratoren-Anmeldung aufgerufen.
- `GET /login/guest` gibt das Formular zur Anmeldung als Gast zurück.
- `GET /oauth2/authorization/kit` verarbeitet die Anmeldung über OIDC und leitet Nutzende auf die Ansicht, welche die Anmeldung verursachte, oder die Ansicht *Kalender* weiter. Auf diesen Endpunkt werden Nutzende bei Bestätigung einer OIDC-Anmeldung durch den OIDC-Server weitergeleitet.

Für die Ansicht *Kontoliste* (siehe `UsersController`) werden folgende Endpunkte bereitgestellt:

- `GET /admin/users` gibt die Kontoliste aller Nutzenden zurück. Das Administrationskonto wird dabei nicht angezeigt. Für jedes Konto wird ein Link zur Deaktivierung als HTML-Form eingebettet (dies erlaubt die Nutzung von `PUT`-Anfragen).
- `GET /admin/users/userId/deactivate` gibt das Formular zur Deaktivierung des Kontos mit der ID `userId` zurück.
- `PUT /admin/users/userId/deactivate` deaktiviert das Konto mit der ID `userId` und leitet Nutzende auf die Kontoliste weiter.
- `PUT /admin/users/userId/reactivate` reaktiviert das Konto mit der ID `userId` und leitet Nutzende auf die Kontoliste weiter.
- `GET /admin/users/disable-guests` gibt das Formular zur Deaktivierung der Anmeldung als Gast zurück.
- `PUT /admin/users/disable-guests` deaktiviert den Login durch Gastkonten und leitet Nutzende auf die Kontoliste weiter. Zudem werden alle Buchungen von Gastkonten gelöscht.
- `PUT /admin/users/enable-guests` reaktiviert den Login durch Gastkonten und leitet Nutzende auf die Kontoliste weiter.

Außerdem werden folgende allgemeine Endpunkte (siehe `MainController`) bereitgestellt:

- `GET /error` gibt eine Fehlerseite zurück. Dieser Endpunkt wird von Spring automatisch aufgerufen, wenn ein Fehler auftritt.
- `GET /disabled` gibt eine Seite zurück, die Nutzende darüber informiert, dass ihr Konto deaktiviert wurde.

5.1 Package edu.kit.hci.soli.controller

5.1.1 Class BookingCreateController

Signature	`public class BookingCreateController`
Behaviour	Controller for handling booking creation requests.

Constructor

Signature	`public BookingCreateController(TimeService timeService, BookingsService bookingsService, RoomService roomService)`	
Behaviour	Constructs a BookingCreateController with the specified services.	
Parameters	`bookingsService`:	the service for managing bookings
	`roomService`:	the service for managing rooms

Method resolveConflict

Signature	`public String resolveConflict(Model model, HttpServletRequest request, Long roomId, LayoutParams layout)`	
Behaviour	Resolves a booking conflict.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`request`:	the HTTP request
	`roomId`:	the ID of the room
	`layout`:	state of the site layout

Method createBooking

Signature	`public String createBooking(Model model, HttpServletResponse response, HttpServletRequest request, Long roomId, LayoutParams layout, SoliUserDetails principal, CreateEventForm formData)`	
Behaviour	Creates a new booking.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`request`:	the HTTP request
	`roomId`:	the ID of the room
	`principal`:	the authenticated user details
	`formData`:	the form data for the booking

Method newBooking

Signature	`public String newBooking(Model model, HttpServletResponse response, Long roomId, LayoutParams layout, LocalDateTime start, LocalTime end, Boolean cooperative)`	
Behaviour	Displays the form for creating a new booking.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`roomId`:	the ID of the room
	`start`:	the start date and time of the booking
	`end`:	the end date and time of the booking
	`cooperative`:	whether the booking is cooperative

5.1.2 Class BookingViewController

Signature	`public class BookingViewController`
Behaviour	Controller for handling booking-related requests.

Constructor

Signature	`public BookingViewController(BookingsService bookingsService, RoomService roomService, UserService userService, SoliConfiguration soliConfiguration, TimeService timeService)`	
Behaviour	Constructs a BookingViewController with the specified services.	
Parameters	`bookingsService`:	the service for managing bookings
	`roomService`:	the service for managing rooms
	`userService`:	the service for managing users
	`soliConfiguration`:	the configuration of the application

Method editBooking

Signature	`public String editBooking(Model model, HttpServletResponse response, SoliUserDetails principal, Long roomId, Long eventId, LayoutParams layout, EditBookingDescriptionForm formData)`	
Behaviour	Edit the Description for a booking.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`principal`:	the authenticated user details
	`roomId`:	the ID of the room
	`eventId`:	the ID of the event
	`layout`:	state of the site layout

Method bookingICalendar

Signature	`public String bookingICalendar(Model model, HttpServletResponse response, SoliUserDetails principal, Long roomId, Long eventId)`	
Behaviour	Download the iCalendar file for a booking.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`principal`:	the authenticated user details
	`roomId`:	the ID of the room
	`eventId`:	the ID of the event

Method viewEvent

Signature	`public String viewEvent(Model model, HttpServletResponse response, SoliUserDetails principal, Long roomId, Long eventId, LayoutParams layout)`	
Behaviour	Displays the details of a specific event.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`principal`:	the authenticated user details
	`roomId`:	the ID of the room
	`eventId`:	the ID of the event
	`layout`:	state of the site layout

Method roomBookings

Signature	`public String roomBookings(int page, int size, Model model, HttpServletResponse response, SoliUserDetails principal, Long roomId, LayoutParams layout)`	
Behaviour	Displays the bookings for a specific room.	
Returns	the view name	
Parameters	`page`:	the page number
	`size`:	the number of items per page
	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`principal`:	the authenticated user details
	`roomId`:	the ID of the room

Method deleteBookings

Signature	`public String deleteBookings(Model model, HttpServletResponse response, SoliUserDetails principal, Long roomId, Long eventId, LayoutParams layout)`	
Behaviour	Deletes a booking for a specific room and event.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`principal`:	the authenticated user details
	`roomId`:	the ID of the room
	`eventId`:	the ID of the event

5.1.3 Class CalendarController

Signature	`public class CalendarController`
Behaviour	Controller for handling calendar-related requests.

Constructor

Signature	`public CalendarController(RoomService roomService)`	
Behaviour	Constructs a CalendarController with the specified `RoomService`.	
Parameters	`roomService`:	the service for managing rooms

Method calendar

Signature	`public String calendar(Model model, HttpServletRequest request, HttpServletResponse response, SoliUserDetails principal, long roomId, LayoutParams layout)`	
Behaviour	Displays the calendar for a specific room.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`request`:	the HTTP request
	`response`:	the HTTP response
	`principal`:	the authenticated user details
	`roomId`:	the ID of the room
	`layout`:	state of the site layout

5.1.4 Class CollaborationRequestController

Signature	`public class CollaborationRequestController`
Behaviour	Controller for handling collaboration requests.

Constructor

Signature	`public CollaborationRequestController(BookingsService bookingsService)`
Behaviour	Constructor for CollaborationRequestController.
Parameters	`bookingsService`: the service to manage bookings

Method rejectCollaborationRequest

Signature	`public String rejectCollaborationRequest(Model model, HttpServletResponse response, SoliUserDetails principal, LayoutParams layout, Long roomId, Long eventId)`
Behaviour	Handles HTTP DELETE requests to reject a collaboration request for a specific booking.
Returns	the view name to be rendered
Parameters	`model`: the model to add attributes to `response`: the HTTP response to set status codes `principal`: the authenticated user details `roomId`: the ID of the room `eventId`: the ID of the event (booking)

Method acceptCollaborationRequest

Signature	`public String acceptCollaborationRequest(Model model, HttpServletResponse response, SoliUserDetails principal, LayoutParams layout, Long roomId, Long eventId)`
Behaviour	Handles HTTP PUT requests to accept a collaboration request for a specific booking.
Returns	the view name to be rendered
Parameters	`model`: the model to add attributes to `response`: the HTTP response to set status codes `principal`: the authenticated user details `roomId`: the ID of the room `eventId`: the ID of the event (booking)

Method viewCollaborationRequest

Signature	`public String viewCollaborationRequest(Model model, HttpServletResponse response, SoliUserDetails principal, LayoutParams layout, Long roomId, Long eventId)`
Behaviour	Handles HTTP GET requests to view a collaboration request for a specific booking.
Returns	the view name to be rendered
Parameters	`model`: the model to add attributes to `response`: the HTTP response to set status codes `principal`: the authenticated user details `roomId`: the ID of the room `eventId`: the ID of the event (booking)

5.1.5 Class EventFeedController

Signature	`public class EventFeedController`
Behaviour	REST controller for generating the FullCalendar event feed.

Constructor

Signature	`public EventFeedController(BookingsService bookingsRepository, RoomService roomService, SoliConfiguration soliConfiguration)`
Behaviour	Constructs an EventFeedController with the specified `BookingsService`.
Parameters	`bookingsRepository`: the service for managing bookings `roomService`: the service for managing rooms `soliConfiguration`: the system configuration

Method handleIllegalArgumentException

Signature	`public ResponseEntity handleIllegalArgumentException(IllegalArgumentException e)`
Behaviour	Handles `IllegalArgumentException` and returns a bad request response with the exception message.
Returns	the response entity with the exception message
Parameters	`e`: the IllegalArgumentException

Method getHolidays

Signature	`public String getHolidays()`
Behaviour	Retrieves the holidays.
Returns	the holidays in iCalendar format to be used by FullCalendar
Throws	`Exception`: if it could not be retrieved

Method getEvents

Signature	`public List getEvents(LocalDateTime start, LocalDateTime end, Long roomId, SoliUserDetails principal)`
Behaviour	Retrieves calendar events within the specified time range.
Returns	the list of calendar events
Parameters	`start`: the start date and time `end`: the end date and time `roomId`: the id of the room to show `principal`: the authenticated user details

5.1.6 Class LayoutParamsAdvice

Signature	`public class LayoutParamsAdvice`
Behaviour	Controller advice for injecting the login state.

Constructor

Signature	`public LayoutParamsAdvice(UserService userService, BookingsService bookingsService, TimeService timeService, RoomService roomService)`
Behaviour	Constructs a LoginControllerAdvice with the specified UserService.
Parameters	`userService`: the service for managing users `bookingsService`: the service for managing bookings `timeService`: the service for managing time `roomService`: the service for managing rooms

Method getLayoutParams

Signature	`public LayoutParams getLayoutParams(LoginStateModel login, HttpServletRequest request)`
Behaviour	Retrieves the state of the site layout
Returns	state of the site layout
Parameters	`login`: the state of the users login `request`: the HTTP request

Method getLoginStateModel

Signature	`public LoginStateModel getLoginStateModel(SoliUserDetails principal, CsrfToken csrf)`
Behaviour	Computes a login state model based on the authenticated user details.
Returns	the login state model
Parameters	`principal`: the authenticated user details `csrf`: the CSRF token

Method getCsrfToken

Signature	`public CsrfToken getCsrfToken(HttpServletRequest request)`
Behaviour	Retrieves the CSRF token from the request.
Returns	the CSRF token
Parameters	`request`: the HTTP request

5.1.7 Class LoginController

Signature	`public class LoginController`
Behaviour	Controller for handling login-related requests.

Constructor

Signature	`public LoginController(SystemConfigurationService systemConfigurationService, SoliConfiguration soliConfiguration)`	
Behaviour	Constructs a LoginController with the specified `SystemConfigurationService`.	
Parameters	`systemConfigurationService`:	the service for retrieving the system configuration
	`soliConfiguration`:	the configuration of the application

Method loginGuest

Signature	`public String loginGuest(Model model)`
Behaviour	Displays the login UI for guests.
Returns	the view name
Parameters	`model`: the model to be used in the view

Method login

Signature	`public String login(HttpServletRequest request, Model model)`	
Behaviour	Displays the login UI.	
Returns	the view name	
Parameters	`request`:	the HTTP request
	`model`:	the model to be used in the view

5.1.8 Class MainController

Signature	`public class MainController extends AbstractErrorController`
Behaviour	Main controller for miscellaneous status requests.

Constructor

Signature	`public MainController(DefaultErrorAttributes errorAttributes, SoliConfiguration soliConfiguration, RoomService roomService)`
Behaviour	Constructs a MainController with the specified DefaultErrorAttributes .
Parameters	`errorAttributes`: the default error attributes `soliConfiguration`: the configuration of the application `roomService`: the service for managing rooms

Method getPublisher

Signature	`public String getPublisher(Model model)`
Behaviour	Returns the view for the publisher information (impressum).
Returns	the view name
Parameters	`model`: the model to be used in the view

Method index

Signature	`public String index(Model model, LayoutParams layout)`
Behaviour	Handles GET requests to the root endpoint. If there is only one room, redirect to that room.
Returns	the name of the view to be rendered
Parameters	`model`: the model to which attributes are added `layout`: the layout parameters

Method securityTxt

Signature	`public String securityTxt(HttpServletResponse response)`
Behaviour	Returns the security.txt file content.
Returns	the security.txt content
Parameters	`response`: the Http response

Method getDisabled

Signature	`public String getDisabled(Model model)`
Behaviour	Returns the view for disabled users.
Returns	the view name
Parameters	`model`: the model to be used in the view

Method handleError

Signature	`public String handleError(Model model, HttpServletRequest request)`
Behaviour	Handles errors and returns the appropriate error view.
Returns	the view name
Parameters	`model`: the model to be used in the view `request`: the HTTP request

5.1.9 Class OpeningHoursController

Signature	`public class OpeningHoursController`
Behaviour	Controller for managing the opening hours of rooms.

Constructor

Signature	`public OpeningHoursController(RoomService roomService)`
Behaviour	Constructs a new OpeningHoursController with the specified RoomService.
Parameters	`roomService`: the service for managing rooms

Method showOpeningHoursForm

Signature	`public String showOpeningHoursForm(Long roomId, Model model, HttpServletResponse response, LayoutParams layout)`
Behaviour	Displays the form for editing the opening hours of a specific room.
Returns	the view name
Parameters	`roomId`: the ID of the room `model`: the model to add attributes to `response`: the HTTP response `layout`: the layout parameters

Method saveOpeningHours

Signature	`public String saveOpeningHours(Long roomId, Model model, LayoutParams layout, HttpServletResponse response, SaveOpeningHoursForm form)`
Behaviour	Saves the opening hours for a specific room.
Returns	the view name
Parameters	`roomId`: the ID of the room `model`: the model to add attributes to `layout`: the layout parameters `response`: the HTTP response `form`: the form containing the opening hours

5.1.10 Class RoomsController

Signature	`public class RoomsController`
Behaviour	Controller class for viewing and managing rooms.

Constructor

Signature	`public RoomsController(RoomService roomService)`
Behaviour	Constructs a new RoomsController instance.
Parameters	`roomService`: the room service to be used by this controller

Method editOrCreateRoom

Signature	`public String editOrCreateRoom(Model model, HttpServletResponse response, EditOrCreateRoomForm formData)`
Behaviour	Handles applying changes to a room from modals created by this controller.
Returns	the name of the view to be rendered
Parameters	`model`: the model to which attributes are added `response`: the HTTP response `formData`: the form data for the room

Method deleteRoom

Signature	`public String deleteRoom(Model model, HttpServletResponse response, long roomId, LayoutParams layout)`
Behaviour	Deletes a room.
Returns	the name of the view to be rendered
Parameters	`model`: the model to which attributes are added `response`: the HTTP response `roomId`: the ID of the room to be deleted

Method showDeletionDialog

Signature	`public String showDeletionDialog(Model model, HttpServletResponse response, long roomId)`
Behaviour	Shows the deletion dialog for a room.
Returns	the name of the view to be rendered
Parameters	`model`: the model to which attributes are added `response`: the HTTP response `roomId`: the ID of the room to be deleted

Method editRoom

Signature	`public String editRoom(Model model, HttpServletResponse response, long roomId)`
Behaviour	Shows the form for editing a room.
Returns	the name of the view to be rendered
Parameters	`model`: the model to which attributes are added `response`: the HTTP response `roomId`: the ID of the room to be edited

Method newRoom

Signature	`public String newRoom(Model model, LayoutParams layout)`
Behaviour	Shows the form for creating a new room.
Returns	the name of the view to be rendered
Parameters	`model`: the model to which attributes are added `layout`: the layout parameters

Method adminRoomList

Signature	`public String adminRoomList(Model model, LayoutParams layout)`
Behaviour	Shows the room list in edit mode.
Returns	the name of the view to be rendered
Parameters	`model`: the model to which attributes are added `layout`: the layout parameters

5.1.11 Class StatisticsController

Signature	`public class StatisticsController`
Behaviour	Controller class for showing statistics.

Constructor

Signature	`public StatisticsController(StatisticsService statisticsService)`
Behaviour	Constructs a new StatisticsController instance.
Parameters	`statisticsService`: the statistics service to be used by this controller

Method showStatistics

Signature	`public String showStatistics(Model model)`
Behaviour	Handles GET requests to the /admin/statistics endpoint. Adds various booking statistics to the model.
Returns	the name of the view to be rendered
Parameters	`model`: the model to which attributes are added

5.1.12 Class UsersController

Signature	`public class UsersController`
Behaviour	Controller class for managing user-related operations.

Constructor

Signature	`public UsersController(UserService userService, SystemConfigurationService systemConfigurationService, SoliConfiguration soliConfiguration)`	
Behaviour	Constructs a UsersController with the specified `UserService`.	
Parameters	`userService`:	the service for managing users
	`systemConfigurationService`:	the service for managing the system configuration
	`soliConfiguration`:	the configuration of the application

Method enableGuests

Signature	`public String enableGuests(Model model, HttpServletResponse response, SoliUserDetails principal)`	
Behaviour	Enables guest login.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`principal`:	the authenticated user details

Method disableGuests

Signature	`public String disableGuests(Model model, HttpServletResponse response, SoliUserDetails principal)`	
Behaviour	Disables guest login.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`principal`:	the authenticated user details

Method disableGuestsConfirmation

Signature	`public String disableGuestsConfirmation(Model model, HttpServletResponse response, SoliUserDetails principal)`	
Behaviour	Displays the confirmation dialog for disabling guest login.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`principal`:	the authenticated user details

Method getUsers

Signature	`public String getUsers(int page, int size, Model model, HttpServletResponse response, SoliUserDetails principal)`	
Behaviour	Retrieves all manageable users and returns the view for the users page.	
Returns	the view name	
Parameters	`page`:	the page number (0-based, i.e., the first page is page 0)
	`size`:	the number of users per page
	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`principal`:	the authenticated user details

Method reactivateUser

Signature	`public String reactivateUser(Model model, HttpServletResponse response, SoliUserDetails principal, Long userId)`	
Behaviour	Reactivates a user by their ID.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`principal`:	the authenticated user details
	`userId`:	the ID of the user to be deactivated

Method deactivateUser

Signature	`public String deactivateUser(Model model, HttpServletResponse response, SoliUserDetails principal, Long userId)`	
Behaviour	Deactivates a user by their ID.	
Returns	the view name	
Parameters	`model`:	the model to be used in the view
	`response`:	the HTTP response
	`principal`:	the authenticated user details
	`userId`:	the ID of the user to be deactivated

Method deactivateUserConfirmation

Signature	`public String deactivateUserConfirmation(Model model, HttpServletResponse response, SoliUserDetails principal, Long userId)`

6 Services

Der Service-Layer koordiniert die Logik der Applikation und stellt diese den Endpunkten, beispielsweise den Controllern bereit. Somit wird auch die Datenverarbeitung von der verarbeitung der HTTP-Anfragen und antworten abgekapselt. Zur bearbeitung dieser Aufgaben nutzen die Services den Repository-Layer, welcher die Interaktion mit der Datenbank abstrahiert. Vor allem komplexere aufgaben, wie das Buchen eines Termines, auflösen eines Terminkonfliktes oder das Versenden von E-Mails wird von den Services behandelt.

Der *Bookings Service* beinhaltet alle Logik für das Verwalten von Terminen: Löschen, Buchen, Koordination der Raumteilung, sowie das Vorbereiten der Termine, welche in der Kalenderansicht zu beobachten sind. Dabei wird auch ein potenziell beim Buchen entstehender Terminkonflikt identifiziert und eine mögliche Auflösung dieses Konfliktes vorbereitet. Nachdem ein/e Nutzende*r die Auflösung des Konfliktes bestätigt hat, wird diese angewandt und die womöglich nötigen anpassungen von anderen Terminen vollzogen.

Der *E-Mail Service* ermöglicht das Senden von E-Mails an die in den Konten hinterlegte E-Mail-Adresse. Diese wurde je nach Kontotyp von den Nutzenden selber oder vom OIDC Provider bereitgestellt.

Der *Room Service* dient zur Auswahl des Raumes um welchen sich z.B. eine Buchung dreht. Diese Klasse ermöglicht insbesondere eine einfache erweiterung der Applikation auf mehrere Räume.

Der *System Configuration Service* dient zur Einstellung der Applikation bereitgestellten funktionen, so wird hier eingestellt ob Gastkonten aktiviert sind. Wird von einem Admin die Gastkonten funktion deaktiviert, so werden von diesem Service auch die Termine aller betroffenen Konten gelöscht.

Der *User Service* hilft mit dem umgang der Konten: Das Deaktivieren eines Kontos, überprüfen, ob ein Konto Admin oder Gast status besitzt und erstellen sowie Löschen von Konten.

6.1 Package edu.kit.hci.soli.service

6.1.1 Interface BookingsService

Signature	`abstract public interface BookingsService`
Behaviour	Service class for managing `Booking` entities. Provides methods to register and retreive booking details.

Method getCurrentBookingOfUser

Signature	`abstract public Optional getCurrentBookingOfUser(Room room, LocalDateTime time, User user)`
Behaviour	Gets booking of logged-in user, which is at the given time if there is one.
Returns	the current booking of user if there is one
Parameters	`room`: room that is currently selected `user`: logged-in user `time`: current time

Method getCurrentHighestBooking

Signature	`abstract public Optional getCurrentHighestBooking(Room room, LocalDateTime time)`
Behaviour	Gets the booking of the highest priority, which is right now if there is one.
Returns	the current booking
Parameters	`room`: room that is currently selected `time`: current time

Method updateDescription

Signature	`abstract public void updateDescription(Booking booking, String description)`
Behaviour	Update the description for a booking.
Parameters	`booking`: the booking `description`: the new description

Method getICalendar

Signature	`abstract public String getICalendar(Booking booking, Locale locale)`
Behaviour	Exports a booking in the iCalendar format.
Returns	the content for the .ics file
Parameters	`booking`: the booking `locale`: the locale for wich to localize

Method getCalendarEvents

Signature	`abstract public List getCalendarEvents(Room room, LocalDateTime start, LocalDateTime end, User user)`
Behaviour	Retrieves calendar events within a specified time range for a user.
Returns	a list of calendar events within the specified time range
Parameters	`start`: the start of the time range `end`: the end of the time range `user`: the user associated with the events (nullable)

Method hasBookings

Signature	`abstract public boolean hasBookings(User user)`
Behaviour	Checks if a user has any bookings.
Returns	true if the user has any bookings, false otherwise
Parameters	`user`: the user to be checked

Method getBookingsByUser

Signature	`abstract public Page getBookingsByUser(User user, Room room, int page, int size)`
Behaviour	Retrieves bookings for a specified user and room.
Returns	a list of bookings for the specified user and room
Parameters	`user`: the user associated with the bookings `room`: the room associated with the bookings `page`: the page number `size`: the number of items per page

Method denyRequest

Signature	`abstract public boolean denyRequest(Booking stagedBooking, User user)`
Behaviour	Denies a request created as the result of an affirmation of a deferred booking. This is the action taken by a user that has an existing booking marked with `ShareRoomType.ON_REQUEST` and denies that a different user may book the room.
Returns	true if the user was in the list of outstanding requests, and the request was denied
Parameters	`stagedBooking`: the booking to be denied `user`: the user who denies the booking

Method confirmRequest

Signature	`abstract public boolean confirmRequest(Booking stagedBooking, User user)`
Behaviour	Confirms a request created as the result of an affirmation of a deferred booking. This is the action taken by a user that has an existing booking marked with `ShareRoomType.ON_REQUEST` and confirms that a different user may book the room.
Returns	true if the user was in the list of outstanding requests and was removed
Parameters	`stagedBooking`: the booking to be confirmed `user`: the user who confirms the booking

Method delete

Signature	`abstract public void delete(Booking booking, BookingDeleteReason reason)`
Behaviour	Deletes a booking for a specified reason.
Parameters	`booking`: the booking to be deleted `reason`: the reason for deletion

Method getBookingById

Signature	`abstract public Booking getBookingById(Long id)`
Behaviour	Retrieves a booking by its ID.
Returns	the booking with the specified ID, or null if not found
Parameters	`id`: the ID of the booking

Method affirm

Signature	`abstract public BookingAttemptResult affirm(Booking booking, PossibleCooperation result)`
Behaviour	Affirms that a possible booking is to be made. This is equivalent to the user confirming the results of an attempt to book. If something has changed in the meantime, a new affirmation may be requested via a returned PossibleCooperation. If some conflicts require contact, a Staged result is returned.
Returns	the result of the affirmation
Parameters	`booking`: the booking to be affirmed `result`: the result of the booking attempt

Method deleteAllBookingsForUser

Signature	`abstract public void deleteAllBookingsForUser(User user)`
Behaviour	Deletes all bookings for a specified user.
Parameters	`user`: the user whose bookings are to be deleted

Method attemptToBook

Signature	`abstract public BookingAttemptResult attemptToBook(Booking booking)`
Behaviour	Attempts to book a room. If successful, the booking is saved and Success is returned. If there are unresolvable conflicts, Failure is returned. If the booking is possible but has effects the user should be informed about, PossibleCooperation is returned.
Returns	the result of the booking attempt
Parameters	`booking`: the booking to be attempted

6.1.2 Interface EmailService

Signature	`abstract public interface EmailService`
Behaviour	Service for sending emails

Method sendMail

Signature	`abstract public void sendMail(User to, String subject, String template, Map model)`
Behaviour	Sends an email to the specified user.
Parameters	`to`: the user to send the email to `subject`: the translation key of the subject `template`: the template to use `model`: the model to use for the template

6.1.3 Interface RoomService

Signature	`abstract public interface RoomService`
Behaviour	Service class for managing `Room` entities. Provides methods to change and retrieve room details.

Method delete

Signature	`abstract public void delete(Room room)`
Behaviour	Deletes a room.
Parameters	`room`: the room to delete

Method save

Signature	`abstract public Room save(Room room)`
Behaviour	Creates a new room.
Returns	the created room
Parameters	`room`: the room to create

Method count

Signature	`abstract public long count()`
Behaviour	Retrieves the number of rooms.
Returns	the number of rooms

Method getAll

Signature	`abstract public List getAll()`
Behaviour	Retrieves all rooms.
Returns	a list of all rooms

Method getOptional

Signature	`abstract public Optional getOptional(long id)`
Behaviour	Retrieves a room by its ID.
Returns	the room with the specified ID or `Optional.empty()`
Parameters	`id`: the ID of the room

Method existsById

Signature	`abstract public boolean existsById(Long id)`
Behaviour	Checks if a room exists by its ID.
Returns	true if the room exists, false otherwise
Parameters	`id`: the ID of the room

6.1.4 Interface StatisticsService

Signature	`abstract public interface StatisticsService`
Behaviour	Service interface for generating booking statistics.

Method countBookingsPerMonthRecent

Signature	`abstract public Map countBookingsPerMonthRecent(TemporalAmount frame)`
Behaviour	Counts the number of bookings for each month of the year within a recent time frame.
Returns	a map where the key is the month and the value is the count of bookings
Parameters	`frame`: the temporal amount representing the recent time frame

Method countBookingsPerMonthAllTime

Signature	`abstract public Map countBookingsPerMonthAllTime()`
Behaviour	Counts the number of bookings for each month of the year for all time.
Returns	a map where the key is the month and the value is the count of bookings

Method countBookingsPerHourRecent

Signature	`abstract public Map countBookingsPerHourRecent(TemporalAmount frame)`
Behaviour	Counts the number of bookings for each hour of the day within a recent time frame.
Returns	a map where the key is the hour of the day and the value is the count of bookings
Parameters	`frame`: the temporal amount representing the recent time frame

Method countBookingsPerHourAllTime

Signature	`abstract public Map countBookingsPerHourAllTime()`
Behaviour	Counts the number of bookings for each hour of the day for all time.
Returns	a map where the key is the hour of the day and the value is the count of bookings

Method countBookingsPerWeekdayRecent

Signature	`abstract public Map countBookingsPerWeekdayRecent(TemporalAmount frame)`
Behaviour	Counts the number of bookings for each day of the week within a recent time frame.
Returns	a map where the key is the day of the week and the value is the count of bookings
Parameters	`frame`: the temporal amount representing the recent time frame

Method countBookingsPerWeekdayAllTime

Signature	`abstract public Map countBookingsPerWeekdayAllTime()`
Behaviour	Counts the number of bookings for each day of the week for all time.
Returns	a map where the key is the day of the week and the value is the count of bookings

6.1.5 Interface SystemConfigurationService

Signature	`abstract public interface SystemConfigurationService`
Behaviour	Service interface for managing the shared system configuration.

Method setGuestLoginEnabled

Signature	`abstract public void setGuestLoginEnabled(boolean enabled)`
Behaviour	Sets the guest login status.
Parameters	`enabled`: true to enable guest login, false to disable it.

Method isGuestLoginEnabled

Signature	`abstract public boolean isGuestLoginEnabled()`
Behaviour	Checks if guest login is enabled.
Returns	true if guest login is enabled, false otherwise.

6.1.6 Interface TimeService

Signature	`abstract public interface TimeService`

Method normalize

Signature	`abstract public LocalDateTime normalize(LocalDateTime time)`
Behaviour	Normalizes a given `LocalDateTime` to the nearest 15-minute interval. The seconds and nanoseconds are set to zero.
Returns	the normalized `LocalDateTime`
Parameters	`time`: the time to be normalized

Method now

Signature	`abstract public LocalDateTime now()`
Behaviour	Retrieves the current time for the configured time zone.
Returns	the current time

Method maximumTime

Signature	`abstract public LocalDateTime maximumTime(Room room)`
Behaviour	Retrieves the maximum ending time for a booking, which is 14 days from now.
Returns	the maximum time for a booking
Parameters	`room`: the room for which the maximum time is calculated

Method minimumTime

Signature	`abstract public LocalDateTime minimumTime(Room room)`
Behaviour	Retrieves the minimum starting time for a booking.
Returns	the minimum time for a booking
Parameters	`room`: the room for which the minimum time is calculated

Method currentSlot

Signature	`abstract public LocalDateTime currentSlot()`
Behaviour	Retrieves the current time slot.
Returns	the current time slot

6.1.7 Interface UserService

Signature	`abstract public interface UserService`
Behaviour	Service class for managing `User` entities. Provides methods for user creation, retrieval, and management.

Method deleteUser

Signature	`abstract public boolean deleteUser(User user)`
Behaviour	Deletes a user. If the user has any bookings, this will not delete the user.
Returns	true if the user was deleted, false otherwise
Parameters	`user`: the user to be deleted

Method updateLastLogin

Signature	`abstract public void updateLastLogin(User user)`
Behaviour	Updates the user's last login timestamp. This method is called whenever a user logs in.
Parameters	`user`: the user whose last login timestamp is to be updated

Method resolveAnonUser

Signature	`abstract public User resolveAnonUser()`
Behaviour	Resolves the anonymous user. This user is used for bookings that have been anonymized.
Returns	the resolved anonymous user

Method isGuest

Signature	`abstract public boolean isGuest(User user)`
Behaviour	Checks if a user is a guest.
Returns	true if the user is a guest, false otherwise
Parameters	`user`: the user to be checked

Method createGuestUser

Signature	`abstract public User createGuestUser(String email)`
Behaviour	Creates a guest user with the given email. The user ID will be generated.
Returns	the created guest user
Parameters	`email`: the email of the guest user

Method resolveGuestUser

Signature	`abstract public User resolveGuestUser(String userId)`
Behaviour	Resolves a guest user by their userId. If the guest user does not exist, creates a new one.
Returns	the resolved guest user
Parameters	`userId`: the email of the guest user

Method isAdmin

Signature	`abstract public boolean isAdmin(User user)`
Behaviour	Checks if a user is an admin.
Returns	true if the user is an admin, false otherwise
Parameters	`user`: the user to be checked

Method resolveAdminUser

Signature	`abstract public User resolveAdminUser()`
Behaviour	Resolves the admin user. If the admin user does not exist, create it.
Returns	the resolved admin user

Method resolveOidcUser

Signature	`abstract public User resolveOidcUser(OidcUser oidcUser)`
Behaviour	Resolves an OidcUser to a `User` entity. If the user does not exist, creates a new one.
Returns	the resolved user
Parameters	`oidcUser`: the OIDC user to be resolved

Method getById

Signature	`abstract public User getById(Long userId)`
Behaviour	Retrieves a user by their ID.
Returns	the user with the specified ID, or null if not found
Parameters	`userId`: the ID of the user

Method getManageableUsers

Signature	`abstract public Page getManageableUsers(int page, int size)`
Behaviour	Retrieves all users that can be managed.
Returns	a list of manageable users
Parameters	`page`: the page number (0-based, i.e., the first page is page 0) `size`: the number of users per page

Method setUserActive

Signature	`abstract public void setUserActive(User user, boolean active)`
Behaviour	Deactivates or Reactivates a user. If deactivating, deletes all bookings for the user.
Parameters	`user`: the user to be reactivated `active`: the new active state of the user

Method findByUserId

Signature	`abstract public User findByUserId(String userId)`
Behaviour	Finds a user by their (external) user ID.
Returns	the user with the specified user ID, or null if not found
Parameters	`userId`: the user ID of the user

7 Daten

Der Daten-Layer befindet sich im Paket `repository` dieses enthält Schnittstellen, die für den Datenzugriff und die Verwaltung von Entitäten in der Datenbank verantwortlich sind. Hierfür wird das Spring Data JPA Framework verwendet, um CRUD-Operationen und benutzerdefinierte Abfragen auf den Entitäten durchzuführen. Die Hauptaufgaben des Daten-Layers sind:

- Verwaltung von Entitäten: Es definiert Schnittstellen, die von JPA-Repository erben, um CRUD-Operationen auf den Entitäten durchzuführen.
- Benutzerdefinierte Abfragen: Es definiert benutzerdefinierte Abfragen, um spezifische Daten aus der Datenbank zu extrahieren.

Welche Operationen auf welcher Entität durchgeführt werden können, ist im Folgenden dargestellt.

7.1 Package edu.kit.hci.soli.repository

7.1.1 Interface BookingsRepository

Signature	`abstract public interface BookingsRepository implements JpaRepository`
Behaviour	Repository for managing `Booking` entities. Extends JpaRepository to provide CRUD operations.

Method deleteByRoom

Signature	`abstract public void deleteByRoom(Room room)`
Behaviour	Deletes all bookings for the specified room.
Parameters	`room`: the room whose bookings are to be deleted

Method getCurrentBookingOfUser

Signature	`abstract public Optional getCurrentBookingOfUser(Room room, LocalDateTime time, User currentUser)`
Behaviour	Gets the booking of the logged-in user that overlaps with the given time if there is one.
Returns	the highest priority of all bookings that overlap with the specified time
Parameters	`room`: the currently selected room `time`: the time to check for booking `currentUser`: user that is logged in

Method getHighestPriority

Signature	`abstract public Optional getHighestPriority(Room room, LocalDateTime time)`
Behaviour	Gets the highest priority of all bookings that overlap with the specified time.
Returns	the highest priority of all bookings that overlap with the specified time
Parameters	`room`: the currently selected room `time`: the time to check for overlapping bookings

Method deleteOutdatedRequests

Signature	`abstract public void deleteOutdatedRequests(LocalDateTime date)`
Behaviour	Deletes all bookings that are outdated and have open requests.
Parameters	`date`: the date to compare the start date to

Method anonymizeOlderThan

Signature	`abstract public void anonymizeOlderThan(LocalDateTime date, User anonymousUser)`
Behaviour	Anonymizes bookings older than the specified date.
Parameters	`date`: the date to compare the end date to `anonymousUser`: the user to anonymize the bookings to

Method countBookingsPerMonthRecent

Signature	`abstract public Stream countBookingsPerMonthRecent(Duration frame)`
Behaviour	Returns the number of bookings per month (recent).
Returns	the number of bookings per month
Parameters	`frame`: the amount of time to look back

Method countBookingsPerMonthAllTime

Signature	`abstract public Stream countBookingsPerMonthAllTime()`
Behaviour	Returns the number of bookings per month (all time).
Returns	the number of bookings per month

Method countBookingsPerHourRecent

Signature	`abstract public Stream countBookingsPerHourRecent(Duration frame)`
Behaviour	Returns the number of bookings per hour of day (recent).
Returns	the number of bookings per hour of day
Parameters	`frame`: the amount of time to look back

Method countBookingsPerHourAllTime

Signature	`abstract public Stream countBookingsPerHourAllTime()`
Behaviour	Returns the number of bookings per hour of day (all time).
Returns	the number of bookings per hour of day

Method countBookingsPerWeekdayRecent

Signature	`abstract public Stream countBookingsPerWeekdayRecent(Duration frame)`
Behaviour	Returns the number of bookings per weekday (recent).
Returns	the number of bookings per weekday
Parameters	`frame`: the amount of time to look back

Method countBookingsPerWeekdayAllTime

Signature	`abstract public Stream countBookingsPerWeekdayAllTime()`
Behaviour	Returns the number of bookings per weekday (all time).
Returns	the number of bookings per weekday

Method findFutureBookingsByGuests

Signature	`abstract public Stream findFutureBookingsByGuests(LocalDateTime time)`
Behaviour	Finds all bookings created by a guest.

Method findWithOutstandingRequests

Signature	`abstract public Stream findWithOutstandingRequests(User user)`
Behaviour	Finds bookings with outstanding requests for the specified user.
Returns	a stream of bookings with outstanding requests for the specified user
Parameters	`user`: the user to check for

Method deleteAllByUser

Signature	`abstract public void deleteAllByUser(User user)`
Behaviour	Deletes all bookings for the specified user.
Parameters	`user`: the user whose bookings are to be deleted

Method existsByUser

Signature	`abstract public boolean existsByUser(User user)`
Behaviour	Checks if a booking exists for the specified user.
Returns	true if a booking exists for the user, false otherwise
Parameters	`user`: the user to check for

Method findByUserAndRoom

Signature	`abstract public Page findByUserAndRoom(User user, Room room, Pageable pageable)`
Behaviour	Finds bookings by user and room.
Returns	the page of bookings
Parameters	`room`: the room associated with the bookings `user`: the user associated with the bookings `pageable`: the pagination information

Method findOverlappingWithOutstandingRequests

Signature	`abstract public Stream findOverlappingWithOutstandingRequests(Room room, LocalDateTime start, LocalDateTime end)`
Behaviour	Finds bookings that overlap with the specified time range and have open requests.
Returns	a stream of bookings that overlap with the specified time range and have open requests
Parameters	`room`: the room in which to search `start`: the start of the time range `end`: the end of the time range

Method findOverlappingBookings

Signature	`abstract public Stream findOverlappingBookings(Room room, LocalDateTime start, LocalDateTime end, User user)`
Behaviour	Finds bookings that overlap with the specified time range.
Returns	a stream of bookings that overlap with the specified time range
Parameters	`room`: the room in which to search `start`: the start of the time range `end`: the end of the time range `user`: the user to filter by

Method findOverlappingBookings

Signature	`abstract public Stream findOverlappingBookings(Room room, LocalDateTime start, LocalDateTime end)`
Behaviour	Finds bookings that overlap with the specified time range.
Returns	a stream of bookings that overlap with the specified time range
Parameters	`room`: the room in which to search `start`: the start of the time range `end`: the end of the time range

7.1.2 Interface RoomRepository

Signature	`abstract public interface RoomRepository implements JpaRepository`
Behaviour	Repository for managing `Room` entities. Extends JpaRepository to provide CRUD operations.

7.1.3 Interface SystemConfigurationRepository

Signature	`abstract public interface SystemConfigurationRepository implements Repository`
Behaviour	Repository interface for managing the system configuration. This repository provides a method to retrieve the single instance of `SystemConfiguration`.

Method getInstance

Signature	`abstract public SystemConfiguration getInstance()`
Behaviour	Retrieves the single instance of `SystemConfiguration`.
Returns	the `SystemConfiguration` instance.

7.1.4 Interface UserRepository

Signature	`abstract public interface UserRepository implements JpaRepository`
Behaviour	Repository interface for managing `User` entities. Extends JpaRepository to provide CRUD operations.

Method deleteUnusedOlderThan

Signature	`abstract public void deleteUnusedOlderThan(LocalDateTime date)`
Behaviour	Deletes all users that have not logged in since the specified date and have no bookings.
Parameters	`date`: the date to compare the last login timestamp to

Method existsByUserId

Signature	`abstract public boolean existsByUserId(String userId)`
Behaviour	Checks if a user with the given userId exists.
Returns	true if a user with the given userId exists, false otherwise
Parameters	`userId`: the user ID to check for

Method findByUserId

Signature	`abstract public User findByUserId(String userId)`
Behaviour	Finds a user by their (external) user ID.
Returns	the user with the specified user ID
Parameters	`userId`: the user ID of the user

Method findAllWithoutAdminOrAnon

Signature	`abstract public Page findAllWithoutAdminOrAnon(Pageable pageable)`
Behaviour	Finds all users except the admin user.
Returns	a list of users excluding the admin user

8 Data Model

Das Data Model von *Soli* ist in die Pakete *dto* und *domain* aufgeteilt. Die *dto*-Klassen sind die Datenübertragungsobjekte, die die Daten zwischen den Schichten des Systems übertragen. Die *domain*-Klassen sind die Datenobjekte, die in der Datenbank gespeichert werden, und bilden somit das interne Datenmodell des Systems.

Dieses Datenmodell wird mithilfe von Spring Data in der PostgreSQL-Datenbank (welche sich zur Isolation in einem separaten Container befindet) abgebildet. In dieser wird das Datenmodell in Form von Tabellen und Beziehungen zwischen diesen gespeichert. In der Anwendung werden die Datenobjekte nur im Kontext aktiver Anfragen im Arbeitsspeicher gehalten und bei Bedarf in die Datenbank geschrieben. Als Konsequenz können die ACID-Eigenschaften der Datenbank genutzt werden, um die Daten vor Verlust oder Inkonsistenzen zu schützen. Damit die Daten auch über zukünftige Aktualisierungen des Systems hinweg erhalten bleiben, wird das Datenbankmigrationssystem Flyway genutzt, das die Datenbankstruktur anpasst, wenn sich das Datenmodell ändert.

Den Kern des Buchungsmodells bildet die Klasse `Booking`, die eine Buchung eines Raumes repräsentiert. Jede Buchung ist einem Raum und einem Nutzer zugeordnet und hat einen Start- und Endzeitpunkt. Außerdem kann eine Buchung optional einen Kommentar enthalten, der von dem Nutzer hinterlassen wurde. Zur Umsetzung von Buchungen des Kooperationstyps *ON_REQUEST* existiert zudem eine automatisch verwaltete Tabelle von noch offenen Anfragen für eine Buchung.

Da die Authentifikation von Nutzenden nicht durch das System selbst, sondern durch das OIDC-System des KIT erfolgt, werden statt vollen Anmeldedaten nur die OIDC-IDs der Nutzenden gespeichert. Diese sind im `User`-Objekt unter dem Namen `userId` mit dem Präfix `kit/` abgelegt. Gäste erhalten eine interne ID, welche mit dem Präfix `guest/` beginnt, und somit von den IDs der KIT-Nutzenden unterschieden werden kann. Der Administrator des Systems hat die fixe interne ID `admin`. Die Tabelle der Nutzenden ergänzt diese Information um die E-Mail-Adresse, den Namen und die Sprache der Nutzenden, welche für die Kommunikation mit diesen (z.B. per E-Mail) genutzt werden. Außerdem wird eine Flagge gespeichert, die es Administratoren ermöglicht, einzelne Nutzende zu sperren.

Eine Raumtabelle wird genutzt, um die Erweiterbarkeit des Systems zu gewährleisten. In dieser Tabelle wird im minimalen System entsprechend den Musskriterien nur ein Raum gespeichert, welcher die ID `1` trägt.

Eine Übersicht des Datenbankmodells ist in Abbildung 8.1 dargestellt.

8.1 Package edu.kit.hci.soli.dto

8.1.1 Interface BookingAttemptResult

Signature	`abstract public sealed interface BookingAttemptResult`
Behaviour	Algebraic data type representing the result of a booking attempt.

Interface PossibleCooperation

Signature	`static abstract public sealed interface PossibleCooperation implements BookingAttemptResult`
Behaviour	Algebraic data type representing a failed booking attempt that may be resolved by cooperation.

Method contact

Signature	`abstract public List contact()`
Behaviour	Gets the list of bookings that require the permission for this cooperation. If this is not empty, the successful result of affirming this cooperation would be of type `Staged`.
Returns	the list of bookings to contact

Method cooperate

Signature	`abstract public List cooperate()`
Behaviour	Gets the list of bookings that would happen concurrently to this booking.
Returns	the list of bookings to cooperate with

Method override

Signature	`abstract public List override()`
Behaviour	Gets the list of bookings that would have to be overwritten for this cooperation.
Returns	the list of bookings to override

Record Deferred

Signature	`static final public record Deferred implements PossibleCooperation`
Behaviour	Represents a deferred possible cooperation in a booking attempt. The successful result of affirming this cooperation is of type `Staged` (and, as such, will require confirmation by others).

Constructor

Signature	`public Deferred(List override, List contact, List cooperate)`

Method override

Signature	`public List override()`
Behaviour	Gets the list of bookings that would have to be overwritten for this cooperation.
Returns	the list of bookings to override
Overrides	`PossibleCooperation.override()`

Method contact

Signature	`public List contact()`
Behaviour	Gets the list of bookings that require the permission for this cooperation. If this is not empty, the successful result of affirming this cooperation would be of type `Staged`.
Returns	the list of bookings to contact
Overrides	`PossibleCooperation.contact()`

Method cooperate

Signature	`public List cooperate()`
Behaviour	Gets the list of bookings that would happen concurrently to this booking.
Returns	the list of bookings to cooperate with
Overrides	`PossibleCooperation.cooperate()`

Record Immediate

Signature	`static final public record Immediate implements PossibleCooperation`
Behaviour	Represents an immediate possible cooperation in a booking attempt. The successful result of affirming this cooperation is an immediate `Success`.

Constructor

Signature	`public Immediate(List override, List cooperate)`

Method contact

Signature	`public List contact()`
Behaviour	Gets the list of bookings that require the permission for this cooperation. If this is not empty, the successful result of affirming this cooperation would be of type `Staged`.
Returns	the list of bookings to contact
Overrides	`PossibleCooperation.contact()`

Method override

Signature	`public List override()`
Behaviour	Gets the list of bookings that would have to be overwritten for this cooperation.
Returns	the list of bookings to override
Overrides	`PossibleCooperation.override()`

Method cooperate

Signature	`public List cooperate()`
Behaviour	Gets the list of bookings that would happen concurrently to this booking.
Returns	the list of bookings to cooperate with
Overrides	`PossibleCooperation.cooperate()`

Record Failure

Signature	`static final public record Failure implements BookingAttemptResult`
Behaviour	Represents a booking attempt that has failed because of an unresolveable conflict with a booking of higher or equal priority.

Constructor

Signature	`public Failure(List conflict)`

Method conflict

Signature	`public List conflict()`

Record Staged

Signature	`static final public record Staged implements BookingAttemptResult`
Behaviour	Represents a booking attempt that was successful but requires confirmation by one or more users.

Constructor

Signature	`public Staged(Booking booking)`

Method booking

Signature	`public Booking booking()`

Record Success

Signature	`static final public record Success implements BookingAttemptResult`
Behaviour	Represents a successful booking attempt.

Constructor

Signature	`public Success(Booking booking)`

Method booking

Signature	`public Booking booking()`

8.1.2 Class BookingByDay

Signature	`public class BookingByDay`
Behaviour	Data Transfer Object (DTO) representing the number of bookings for a specific day of the week.

Constructor

Signature	`public BookingByDay(int dayOfWeek, long count)`	
Behaviour	Constructs a new BookingByDay instance.	
Parameters	`dayOfWeek`:	the day of the week (1 for Monday, 7 for Sunday)
	`count`:	the number of bookings for the specified day of the week

Method hashCode

Signature	`public int hashCode()`

Method equals

Signature	`public boolean equals(Object o)`

Method getCount

Signature	`public long getCount()`
Behaviour	Gets the number of bookings for the specified day of the week.
Returns	the number of bookings

Method getDayOfWeek

Signature	`public int getDayOfWeek()`
Behaviour	Gets the day of the week as Sunday (0) to Saturday (6).
Returns	the day of the week

8.1.3 Class BookingByHour

Signature	`public class BookingByHour`
Behaviour	Data Transfer Object (DTO) representing the number of bookings for a specific hour of the day.

Constructor

Signature	`public BookingByHour(int hour, long count)`
Behaviour	Constructs a new BookingByHour instance.
Parameters	`hour`: the hour of the day (0-23) `count`: the number of bookings for the specified hour

Method hashCode

Signature	`public int hashCode()`

Method equals

Signature	`public boolean equals(Object o)`

Method getCount

Signature	`public long getCount()`
Behaviour	Gets the number of bookings for the specified hour.
Returns	the number of bookings

Method getHour

Signature	`public int getHour()`
Behaviour	Gets the hour of the day.
Returns	the hour of the day

8.1.4 Class BookingByMonth

Signature	`public class BookingByMonth`
Behaviour	Data Transfer Object (DTO) representing the number of bookings for a specific month.

Constructor

Signature	`public BookingByMonth(int month, long count)`
Behaviour	Constructs a new BookingByMonth instance.
Parameters	`month`: the month of the year (1-12) `count`: the number of bookings for the specified month

Method hashCode

Signature	`public int hashCode()`

Method equals

Signature	`public boolean equals(Object o)`

Method getCount

Signature	`public long getCount()`
Behaviour	Gets the number of bookings for the specified month.
Returns	the number of bookings

Method getMonth

Signature	`public int getMonth()`
Behaviour	Gets the month of the year.
Returns	the month of the year

8.1.5 Enum BookingDeleteReason

Signature	`final public enum BookingDeleteReason`
Behaviour	Enumeration representing the reasons for deleting a booking.

8.1.6 Record CalendarEvent

Signature	`final public record CalendarEvent`	
Behaviour	Record representing an event as specified by FullCalendar.	
Parameters	`url`:	the URL associated with the event
	`title`:	the title of the event
	`start`:	the start date and time of the event
	`end`:	the end date and time of the event
	`shareRoomType`:	the type of room sharing
	`favorite`:	whether the event is owned by the current user
	`backgroundColor`:	the background color of the event
	`textColor`:	the text color of the event
	`priority`:	the priority level of the event

Constructor

Signature `public CalendarEvent(String url, String title, LocalDateTime start, LocalDateTime end, ShareRoomType shareRoomType, boolean favorite, String backgroundColor, String textColor, Priority priority)`

Method url

Signature `public String url()`

Method title

Signature `public String title()`

Method start

Signature `public LocalDateTime start()`

Method end

Signature `public LocalDateTime end()`

Method shareRoomType

Signature	`public ShareRoomType shareRoomType()`

Method favorite

Signature	`public boolean favorite()`

Method backgroundColor

Signature	`public String backgroundColor()`

Method textColor

Signature	`public String textColor()`

Method priority

Signature	`public Priority priority()`

8.1.7 Enum KnownError

Signature	`final public enum KnownError`
Behaviour	Enumeration representing known errors with their corresponding titles and messages.

Field message

Signature	`final public KnownError message`
Behaviour	The detailed error message.

Field title

Signature	`final public KnownError title`
Behaviour	The title of the error message.

8.1.8 Class LayoutParams

Signature `public class LayoutParams`

Constructor

Signature `public LayoutParams(LoginStateModel login, Room room, RoomChangeListener onRoomChange, boolean hasMultipleRooms)`

Method isMultipleRooms

Signature `public boolean isMultipleRooms()`

Method getCurrentBookingOfUser

Signature `public Booking getCurrentBookingOfUser()`

Method getCurrentHighestBooking

Signature `public Booking getCurrentHighestBooking()`

Method setRoom

Signature `public void setRoom(Room room)`

Method getRoom

Signature `public Room getRoom()`

Method getLogin

Signature `public LoginStateModel getLogin()`

Record ParamsUpdate

Signature	`static final public record ParamsUpdate`	
Behaviour	Encapsulates the component of `LayoutParams` that have to be updated when the current room changes.	
Parameters	`currentHighestBooking`:	the booking, which has the highest priority at the current time if there is one
	`currentBookingOfUser`:	the booking of the logged-in user at the current time if there is one

Constructor

Signature	`public ParamsUpdate(Booking currentHighestBooking, Booking currentBookingOfUser)`

Method currentHighestBooking

Signature	`public Booking currentHighestBooking()`

Method currentBookingOfUser

Signature	`public Booking currentBookingOfUser()`

Interface RoomChangeListener

Signature	`static abstract public interface RoomChangeListener`
Behaviour	Functional interface for listening to room changes on a `LayoutParams` and modifying relevant fields as needed.

Method onRoomChange

Signature	`abstract public ParamsUpdate onRoomChange(Room room)`	
Behaviour	Invoked initially when constructing `LayoutParams` and whenever the current room changes. Use this to recompute necessary components by returning the appropriate `ParamsUpdate`.	
Returns	the new current highest booking	
Parameters	`room`:	the new room

8.1.9 Record LoginStateModel

Signature	`final public record LoginStateModel`	
Behaviour	Record representing the login state model.	
Parameters	`name`:	the name of the user
	`kind`:	the kind of login
	`csrfToken`:	the CSRF token associated with the login
	`user`:	the user associated with the login, can be null

Constructor

Signature	`public LoginStateModel(String name, Kind kind, CsrfToken csrfToken, User user)`

Method name

Signature	`public String name()`

Method kind

Signature	`public Kind kind()`

Method csrfToken

Signature	`public CsrfToken csrfToken()`

Method user

Signature	`public User user()`

Enum Kind

Signature	`static final public enum Kind`
Behaviour	Enumeration representing the kind of login.

8.2 Package edu.kit.hci.soli.domain

8.2.1 Class Booking

Signature	`public class Booking`
Behaviour	The datamodel for a Booking as it is stored in the database.

Constructor

Signature	`public Booking()`
Behaviour	Default constructor for JPA.

Constructor

Signature	`public Booking(Long id, String description, LocalDateTime startDate, LocalDateTime endDate, ShareRoomType shareRoomType, Room room, User user, Priority priority, Set openRequests)`	
Behaviour	Constructs a new Booking with the specified details.	
Parameters	`id`:	the unique identifier for the booking
	`description`:	a description of the booking
	`startDate`:	the start date and time of the booking
	`endDate`:	the end date and time of the booking
	`shareRoomType`:	the kind of room sharing for the booking
	`room`:	the room associated with the booking
	`user`:	the user who made the booking
	`priority`:	the priority level of the booking
	`openRequests`:	the set of share requests that must still be resolved

Method hashCode

Signature	`public int hashCode()`

Method equals

Signature	`public boolean equals(Object o)`

Method setOpenRequests

Signature	`public void setOpenRequests(Set openRequests)`
Behaviour	Sets the set of share requests that must still be resolved for this booking to be confirmed.
Parameters	`openRequests`: the set of share requests that must still be resolved

Method setPriority

Signature	`public void setPriority(Priority priority)`
Behaviour	Sets the priority level of the booking.
Parameters	`priority`: the priority level of the booking

Method setUser

Signature	`public void setUser(User user)`
Behaviour	Sets the user who made the booking.
Parameters	`user`: the user who made the booking

Method setRoom

Signature	`public void setRoom(Room room)`
Behaviour	Sets the room associated with the booking.
Parameters	`room`: the room associated with the booking

Method setShareRoomType

Signature	`public void setShareRoomType(ShareRoomType shareRoomType)`
Behaviour	Sets the kind of room sharing for the booking.
Parameters	`shareRoomType`: the kind of room sharing for the booking

Method setEndDate

Signature	`public void setEndDate(LocalDateTime endDate)`
Behaviour	Sets the end date and time of the booking.
Parameters	`endDate`: the end date and time of the booking

Method setStartDate

Signature	`public void setStartDate(LocalDateTime startDate)`
Behaviour	Sets the start date and time of the booking.
Parameters	`startDate`: the start date and time of the booking

Method setDescription

Signature	`public void setDescription(String description)`
Behaviour	Sets the description of the booking.
Parameters	`description`: the description of the booking

Method setId

Signature	`public void setId(Long id)`
Behaviour	Sets the unique identifier for the booking.
Parameters	`id`: the unique identifier for the booking

Method getOpenRequests

Signature	`public Set getOpenRequests()`
Behaviour	Gets the set of share requests that must still be resolved for this booking to be confirmed.
Returns	the set of share requests that must still be resolved

Method getPriority

Signature	`public Priority getPriority()`
Behaviour	Gets the priority level of the booking.
Returns	the priority level of the booking

Method getUser

Signature	`public User getUser()`
Behaviour	Gets the user who made the booking.
Returns	the user who made the booking

Method getRoom

Signature	`public Room getRoom()`
Behaviour	Gets the room associated with the booking.
Returns	the room associated with the booking

Method getShareRoomType

Signature	`public ShareRoomType getShareRoomType()`
Behaviour	Gets the kind of room sharing for the booking.
Returns	the kind of room sharing for the booking

Method getEndDate

Signature	`public LocalDateTime getEndDate()`
Behaviour	Gets the end date and time of the booking.
Returns	the end date and time of the booking

Method getStartDate

Signature	`public LocalDateTime getStartDate()`
Behaviour	Gets the start date and time of the booking.
Returns	the start date and time of the booking

Method getDescription

Signature	`public String getDescription()`
Behaviour	Gets the description of the booking.
Returns	the description of the booking

Method getId

Signature	`public Long getId()`
Behaviour	Gets the unique identifier for the booking.
Returns	the unique identifier for the booking

8.2.2 Enum Priority

Signature	`final public enum Priority`
Behaviour	Enumeration representing the priority levels of a booking. Higher levels indicate higher priority.

8.2.3 Class Room

Signature	`public class Room`
Behaviour	The datamodel for a room as it is stored in the database.

Constructor

Signature	`public Room()`
Behaviour	Default constructor for JPA.

Constructor

Signature	`public Room(Long id, String name, String description, String location)`	
Behaviour	Constructs a new room with the specified details.	
Parameters	`id`:	the unique identifier for the room
	`name`:	the name of the room
	`description`:	the description of the room
	`location`:	the location of the room

Method hashCode

Signature	`public int hashCode()`

Method equals

Signature	`public boolean equals(Object o)`

Method setOpeningHours

Signature	`public void setOpeningHours(Map openingHours)`
Behaviour	Sets the opening hours of the room.
Parameters	`openingHours`: the opening hours of the room

Method setLocation

Signature	`public void setLocation(String location)`
Behaviour	Sets the description for the location of the room.
Parameters	`location`: the new location for the room

Method setDescription

Signature	`public void setDescription(String description)`
Behaviour	Sets the description for the room.
Parameters	`description`: the description of the room

Method setName

Signature	`public void setName(String name)`
Behaviour	Sets the name of the room.
Parameters	`name`: the name of the room

Method setId

Signature	`public void setId(Long id)`
Behaviour	Sets the unique identifier for the room.
Parameters	`id`: the unique identifier for the room

Method getOpeningHours

Signature	`public Map getOpeningHours()`
Behaviour	Gets the opening hours of the room. <p> The resulting value is a copy of the map with missing values for week days added
Returns	the opening hours of the room

Method getLocation

Signature	`public String getLocation()`
Behaviour	Gets the description for the location of the room.
Returns	the location description

Method getDescription

Signature	`public String getDescription()`
Behaviour	Gets the description of the room.
Returns	the description of the room

Method getName

Signature	`public String getName()`
Behaviour	Gets the name of the room.
Returns	the name of the room

Method getId

Signature	`public Long getId()`
Behaviour	Gets the unique identifier for the room.
Returns	the unique identifier for the room

8.2.4 Enum ShareRoomType

Signature	`final public enum ShareRoomType`
Behaviour	Enumeration representing the types of room sharing options.

8.2.5 Class SystemConfiguration

Signature	`public class SystemConfiguration`
Behaviour	Entity class representing the system configuration. This table is designed to contain only a single row.

Constructor

Signature	`public SystemConfiguration()`

Method setGuestLoginEnabled

Signature	`public void setGuestLoginEnabled(boolean guestLoginEnabled)`
Behaviour	Sets the value of the guestLoginEnabled flag.
Parameters	`guestLoginEnabled`: true to enable guest login, false to disable it.

Method isGuestLoginEnabled

Signature	`public boolean isGuestLoginEnabled()`
Behaviour	Gets the value of the guestLoginEnabled flag.
Returns	true if guest login is enabled, false otherwise.

8.2.6 Class TimeTuple

Signature	`public class TimeTuple`
Behaviour	A tuple representing the start and end times.

Constructor

Signature	`public TimeTuple(LocalTime start, LocalTime end)`

Constructor

Signature	`public TimeTuple()`
Behaviour	Default constructor.

Method toString

Signature	`public String toString()`

Method hashCode

Signature	`public int hashCode()`

Method equals

Signature	`public boolean equals(Object o)`

Method setEnd

Signature	`public void setEnd(LocalTime end)`
Behaviour	Sets the end time.
Parameters	`end`: the end time

Method setStart

Signature	`public void setStart(LocalTime start)`
Behaviour	Sets the start time.
Parameters	`start`: the start time

Method getEnd

Signature	`public LocalTime getEnd()`
Behaviour	Gets the end time.
Returns	the end time

Method getStart

Signature	`public LocalTime getStart()`
Behaviour	Gets the start time.
Returns	the start time

8.2.7 Class User

Signature	`public class User`
Behaviour	The datamodel for a User as it is stored in the database.

Constructor

Signature	`public User()`
Behaviour	Default constructor for User.

Constructor

Signature	`public User(Long id, String username, String email, String userId, boolean isDisabled, Locale locale, LocalDateTime lastLogin)`
Behaviour	Constructs a new User with the specified details.
Parameters	`id`: the unique identifier for the user `username`: the username of the user `email`: the email address of the user `userId`: the unique user ID `isDisabled`: whether the user is disabled `locale`: the locale of the user `lastLogin`: time of the last login of the user

Method setLastLogin

Signature	`public void setLastLogin(LocalDateTime lastLogin)`
Behaviour	Sets the date and time of the last login of the user.
Parameters	`lastLogin`: the date and time of the last login of the user

Method getLastLogin

Signature	`public LocalDateTime getLastLogin()`
Behaviour	Gets the date and time of the last login of the user.
Returns	the date and time of the last login of the user

Method hashCode

Signature	`public int hashCode()`

Method equals

Signature	`public boolean equals(Object o)`

Method setLocale

Signature	`public void setLocale(Locale locale)`
Behaviour	Sets the locale of the user. This is used to send mails in the user's preferred language.
Parameters	`locale`: the locale of the user

Method setDisabled

Signature	`public void setDisabled(boolean isDisabled)`
Behaviour	Sets whether the user is disabled.
Parameters	`isDisabled`: true if the user is disabled, false otherwise

Method setUserId

Signature	`public void setUserId(String userId)`
Behaviour	Sets the unique user ID.
Parameters	`userId`: the unique user ID

Method setEmail

Signature	`public void setEmail(String email)`
Behaviour	Sets the email address of the user.
Parameters	`email`: the email address of the user

Method setUsername

Signature	`public void setUsername(String username)`
Behaviour	Sets the username of the user.
Parameters	`username`: the username of the user

Method setId

Signature	`public void setId(Long id)`
Behaviour	Sets the unique identifier for the user.
Parameters	`id`: the unique identifier for the user

Method getLocale

Signature	`public Locale getLocale()`
Behaviour	Gets the locale of the user.
Returns	the locale of the user

Method isDisabled

Signature	`public boolean isDisabled()`
Behaviour	Checks if the user is disabled.
Returns	true if the user is disabled, false otherwise

Method getUserId

Signature	`public String getUserId()`
Behaviour	Gets the unique user ID.
Returns	the unique user ID

Method getEmail

Signature	`public String getEmail()`
Behaviour	Gets the email address of the user.
Returns	the email address of the user

Method getUsername

Signature	`public String getUsername()`
Behaviour	Gets the username of the user.
Returns	the username of the user

Method getId

Signature	`public Long getId()`
Behaviour	Gets the unique identifier for the user.
Returns	the unique identifier for the user

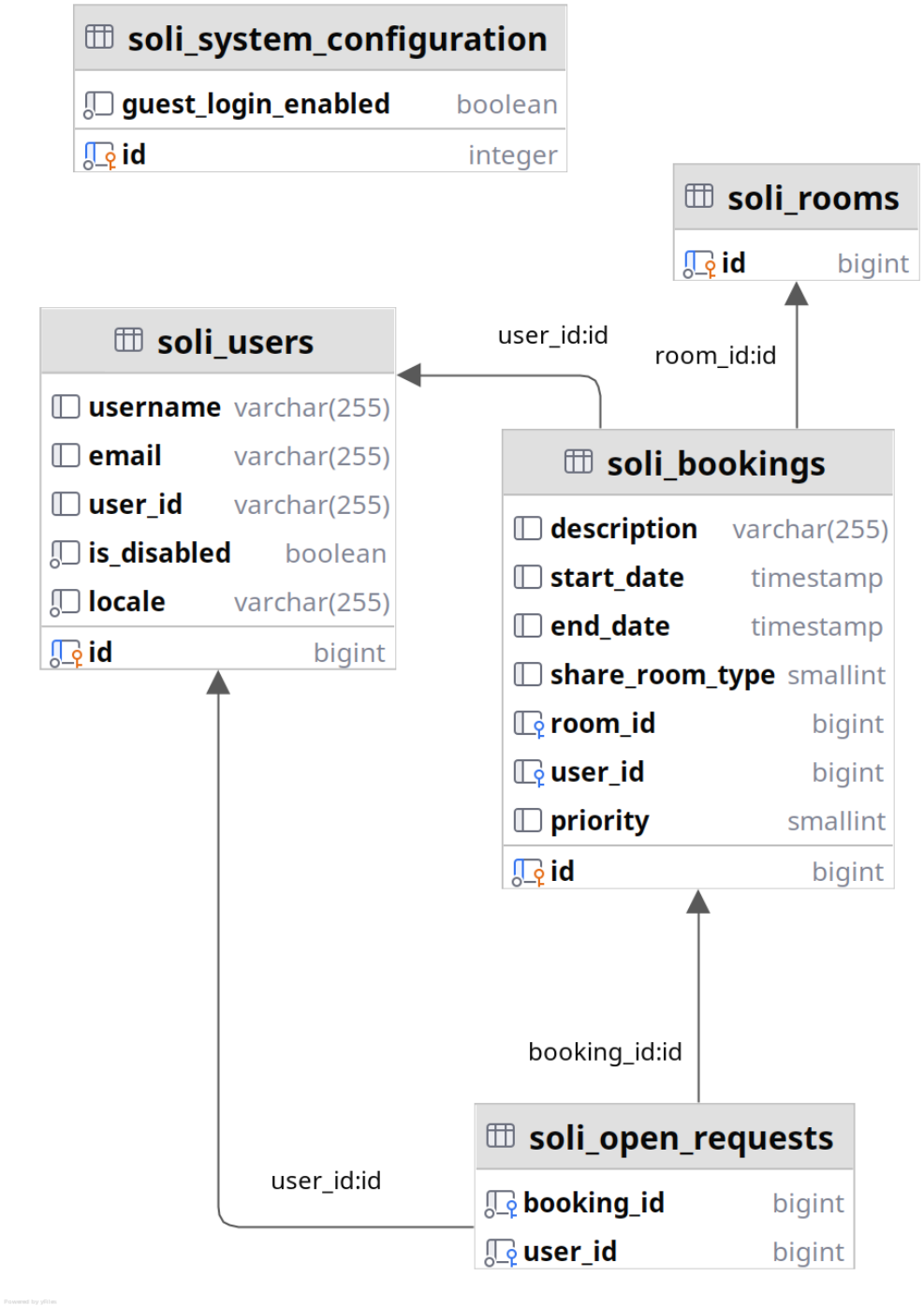


Abbildung 8.1: Datenbankmodell von *Soli*

9 Abläufe

Im folgenden Kapitel werden die wichtigsten Abläufe des Systems beschrieben und zur Veranschaulichung in Diagrammen dargestellt.

9.1 Anmeldeprozess

Das Programm ist so aufgebaut, dass man sich für die Interaktion mit dem System zuerst anmelden muss. Man kann sich als Administrator*in, als Gast oder mit dem KIT-Account anmelden.

Hier wird der Ablauf des Anmeldens in einem Diagramm beschrieben.

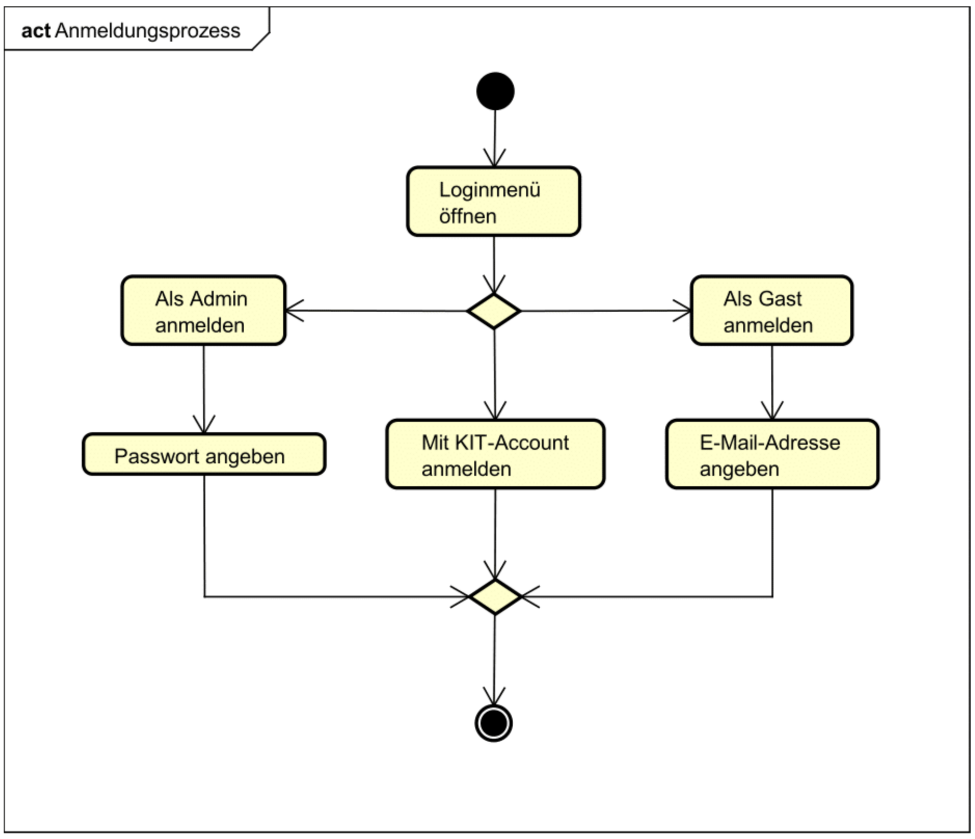


Abbildung 9.1: Diagramm zur Erklärung des Anmeldeprozesses

9.2 Buchung Erstellen

9.2.1 Nutzer*in erstellt Buchung

Weiterhin können Buchungen erstellt werden. Mit der Voraussetzung, dass die nutzende Person angemeldet ist, kann die Buchungsansicht geöffnet werden. Dann gibt muss eine Priorität angegeben werden und festgelegt werden, ob man den Raum teilen möchte. Hier sind die Auswahlmöglichkeiten "Ja", "Neinünd Äuf Anfrage". Bei den Äuf Anfrage"gestellten Buchungen können Nuter*innen über die Website Anfragen stellen, wenn sie den Termin mit nutzen wollen. Weiterhin kann der ausgewählte Zeitraum angepasst oder eine Beschreibung angegeben werden.

Das Ablaufdiagramm beschreibt den Ablauf einer Terminerstellung.

Dieses Sequenzdiagramm zeigt den Ablauf der Buchungserstellung auf der Programmebene.

9.2.2 Konfliktlösung bei Buchungserstellung

Bei der Erstellung eines Termins kann ein Terminkonflikt auftreten, falls im angegebenen Zeitraum bereits eine Buchung getätigt wurde. In diesem Fall wird nach der Priorität der Buchung entschieden und die Buchung mit höherer Priorität wird beibehalten. Im Fall, dass beide Buchungen die gleiche Priorität haben, wird die zuerst erstellte Buchung beibehalten. Wenn beide Buchungen bei der Kategorie "Raum teilenäuf "Ja"gesetzt sind, wird die Buchung mit der niedrigeren Priorität beibehalten. Falls die zuerst getätigte Buchunge auf Äuf Anfrageßteht, besteht die Möglichkeit eine Anfrage zu stellen.

Das Ablaufdiagramm stellt den oben beschrieben Konliktlösungsprozess dar.

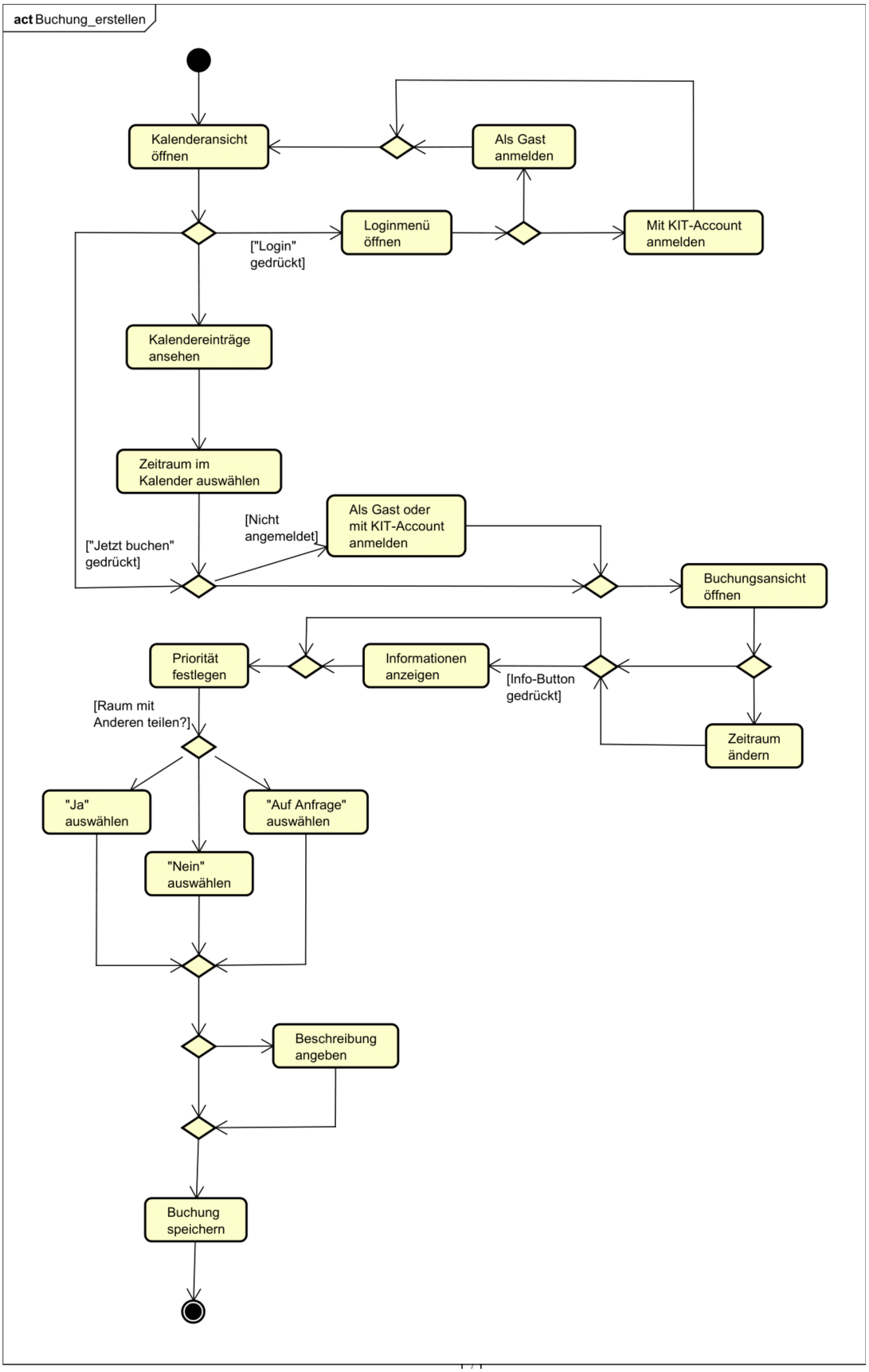


Abbildung 9.2: Diagramm zur Erklärung des Buchungserstellungsprozesses

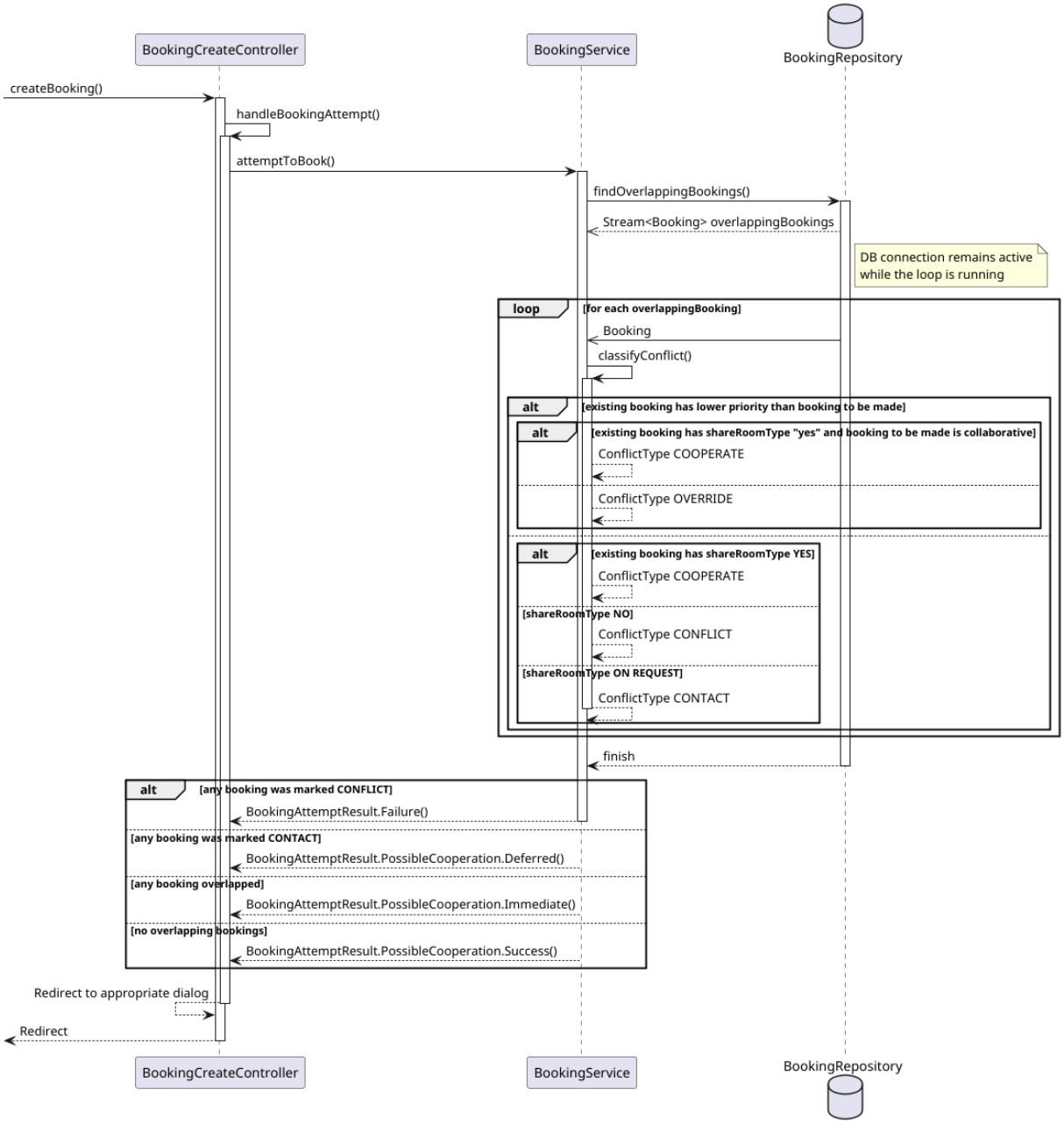


Abbildung 9.3: Sequenzdiagramm zur Erklärung des Buchungserstellungsprozesses

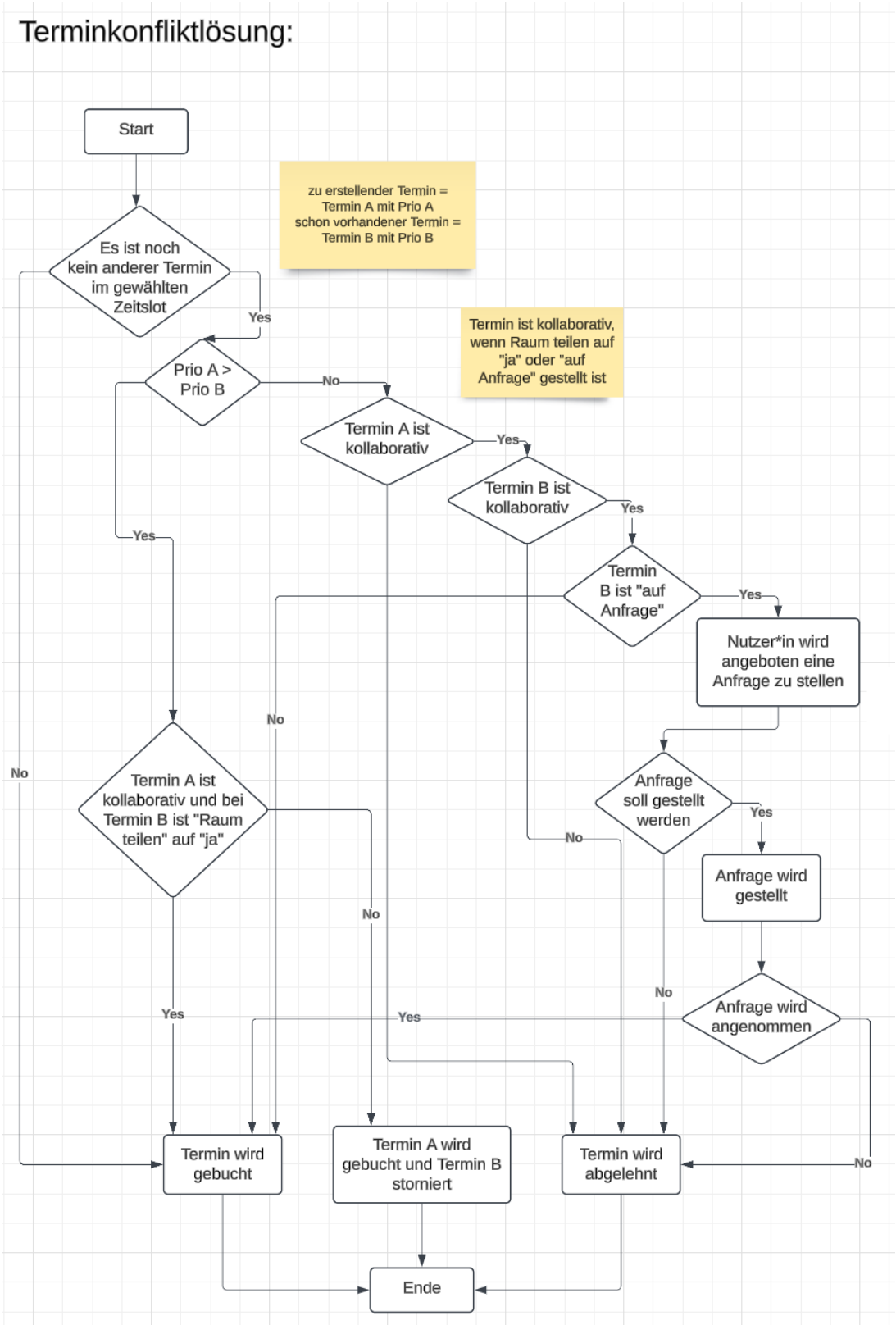


Abbildung 9.4: Diagramm zur Erklärung des Konfliktlösungsprozesses bei der Buchungserstellung

9.3 Buchungen Verwalten

Die Website bietet die Möglichkeit, die eigenen Buchungen zu verwalten. Mit der Voraussetzung, dass man angemeldet ist, kann man die Terminübersicht, welche eine Liste der eigenen Termine anzeigt, öffnen und hat dann die Möglichkeit Termine zu löschen.

Das Ablaufdiagramm beschreibt den Ablauf des Buchungsverwaltungsprozesses.

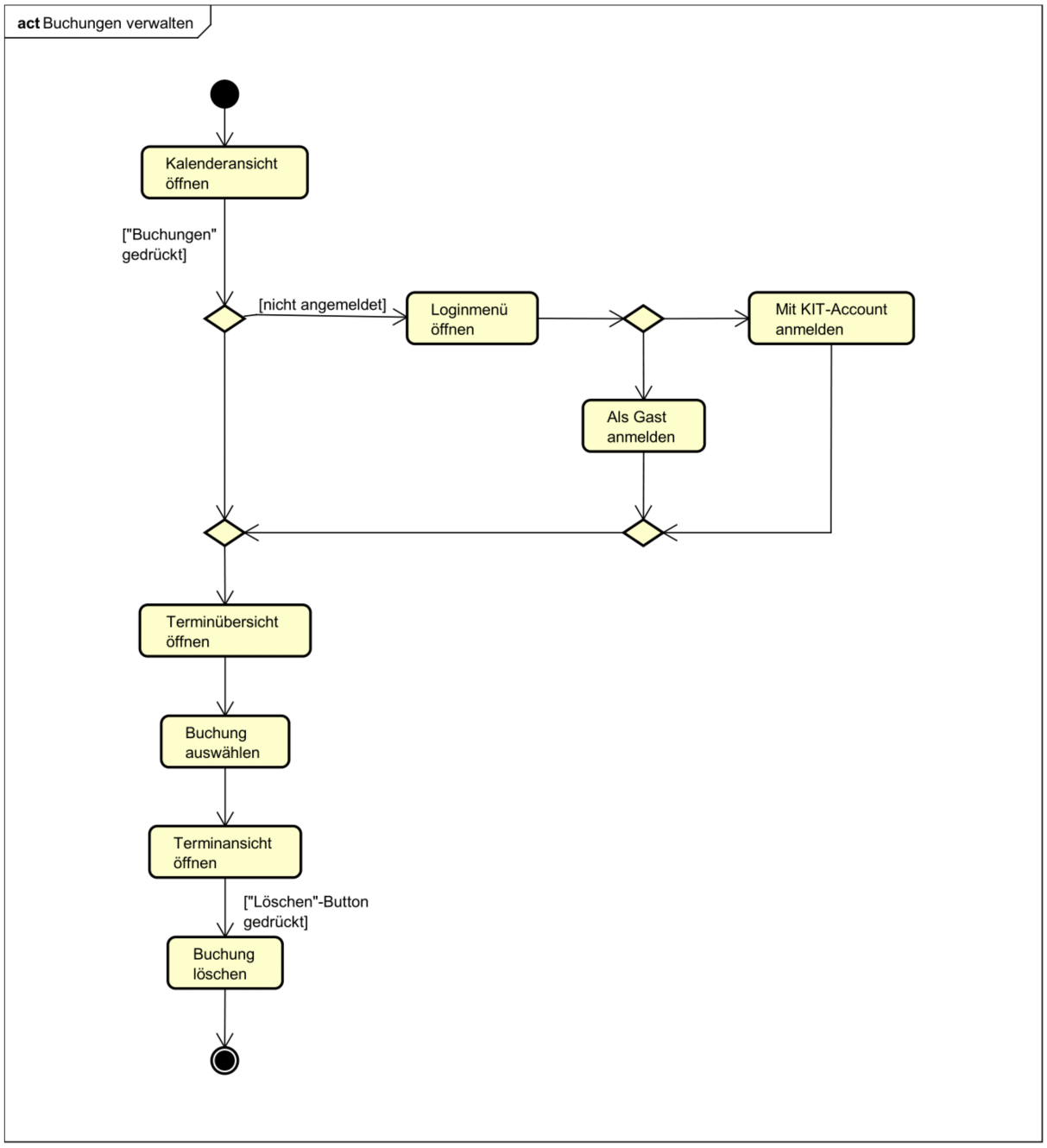


Abbildung 9.5: Diagramm zur Erklärung des Buchungsverwaltungsprozesses

9.4 Terminaktionen

Hier wird die Funktionalität der Terminansicht dargestellt. Wenn der angesehene Termin ein eigener Termin ist oder man als Admin eingeloggt ist, gibt es hier die Möglichkeit den gewählten Termin zu löschen. Ist der Termin nicht der eigene und er steht auf äuf Anfrage"teilen, dann gibt es einen Anfrage-Button über diesen man eine Anfrage schicken kann.

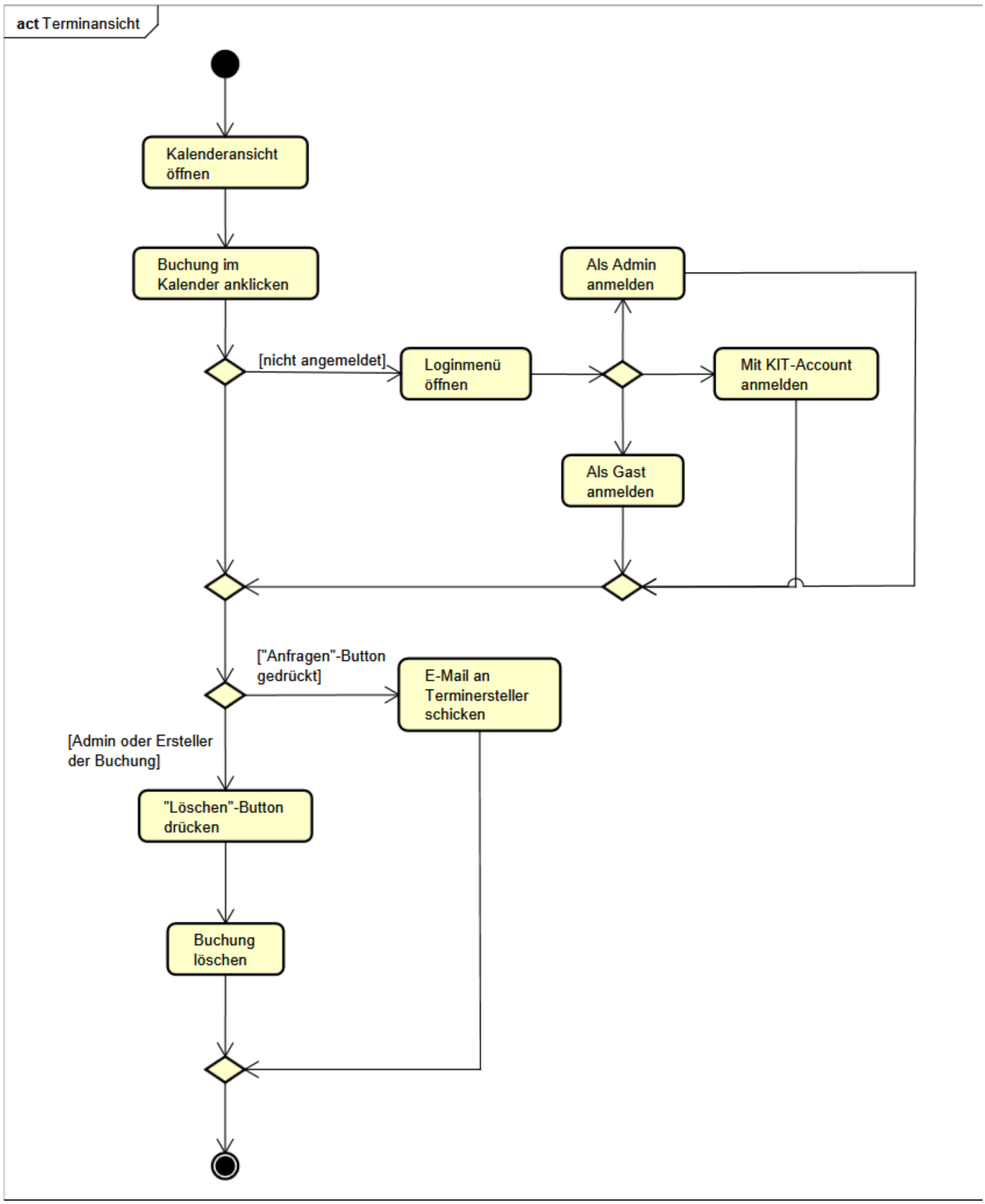


Abbildung 9.6: Diagramm zur Erklärung der Terminaktionen

9.5 Checkoutprozess

Der Checkout-Button bietet die Möglichkeit die eigene laufende Buchung vorzeitig zu beenden und den Raum für andere Nutzende freizugeben.

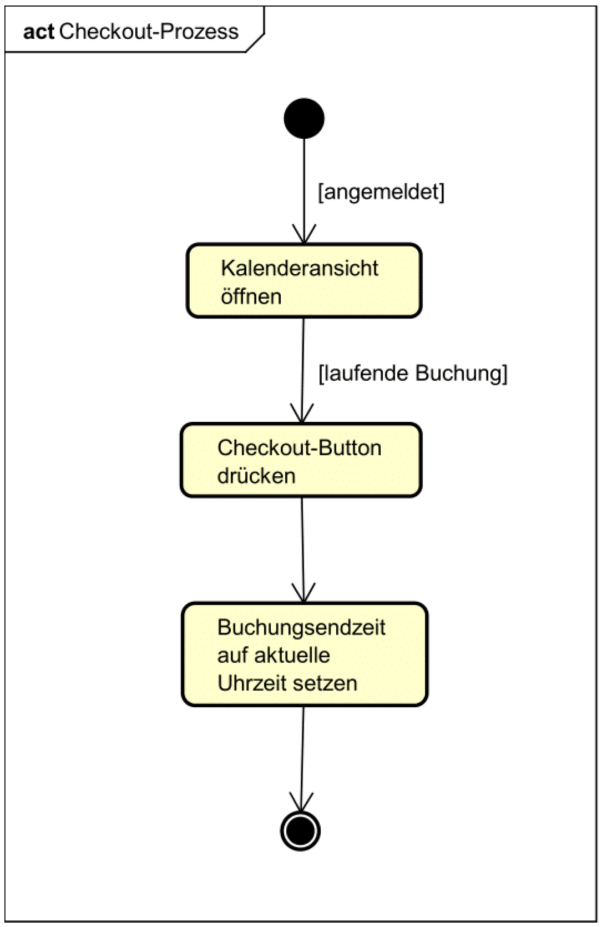


Abbildung 9.7: Diagramm zur Erklärung des Checkoutprozesses

9.6 Adminfunktionalität

Die Adminfunktionalität beinhaltet mehrere Möglichkeiten, die Website zu verwalten. Es kann ein Termin gelöscht, Kontos gebannt, die Öffnungszeiten geändert und die Gästefunktion ausgeschalten werden.

Dieses Ablaufdiagramm stellt die unterschiedlichen Aktionsmöglichkeiten des/r Admins dar.

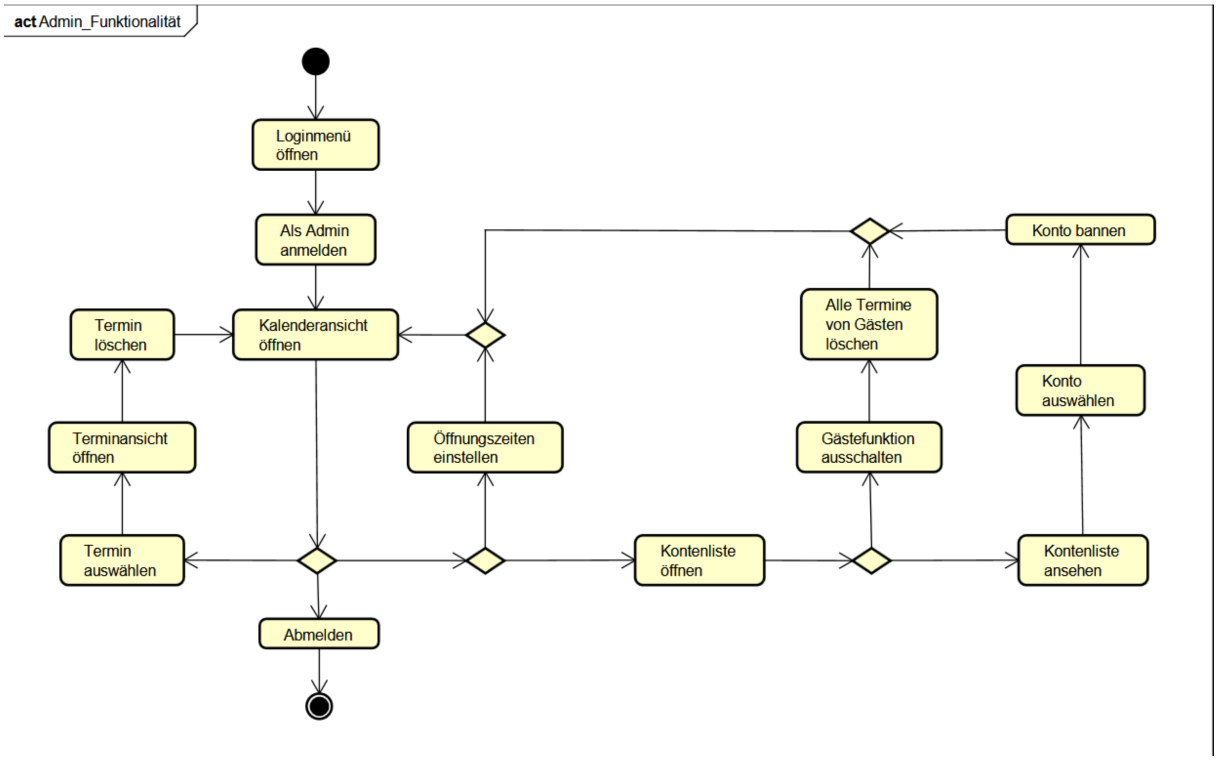


Abbildung 9.8: Diagramm zur Erklärung der Adminfunktionalität

Glossar

ACID Eigenschaften von Datenbanken: Atomar, Konsistent, Isoliert, Dauerhaft. 56

Caddy Webserver, welcher HTTPS-Reverse-Proxy Funktionalität bietet. 6

Container Isolierte Umgebung um Software unabhängig von der zugrunde liegenden Umgebung auszuführen. 5, 6, 56

Controller Klasse, welche Anfragen entgegennimmt und verarbeitet. 6

CRUD Create, Read, Update, Delete, Standardoperationen auf Datenbanken. 49

CSS Cascading Style Sheets, ist eine Sprache um das Visuelle aussehen einer Website zu definieren. 5

DaisyUI CSS-Framework für das Design von Webseiten. 5, 6

Docker Software zur Bereitstellung von Anwendungen innerhalb von Containern. 5, 6

Flyway Bibliothek zur Datenbankmigration. 5, 56

FullCalendar JavaScript-Bibliothek zur Darstellung von Kalendern. 5, 6, 15

Gradle Build-Management-Tool welches auf Java basiert. 5

HTML (Hypertext Markup Language) Eine Sprache, um die Struktur und den Inhalt einer Website zu definieren. 5, 6

HTML-Form Element in HTML, um Nutzereingaben zu sammeln. 16, 17

HTTP (Hypertext Transfer Protocol) Protokoll zur Übertragung von Daten im Internet. 15

HTTPS (Hypertext Transfer Protocol Secure) Verschlüsselte Variante von HTTP. 6

HTTPS-Reverse-Proxy Server, welcher Anfragen entgegennimmt und an andere Server weiterleitet, dabei HTTPS verwendet. 6

JavaScript Programmiersprache um das Logische verhalten von Webseiten zu steuern. 5, 6

JTE (Java Template Engine) Bibliothek zur Generierung von HTML aus Java. 6, 15

JUnit Framework zur Erstellung von Tests in Java. 5

Mockito Framework zur Erstellung von Mocks in Java. 5

MVC-Struktur Model-View-Controller, Architekturmuster zur Trennung von Datenmodell, Darstellung und Steuerung. 6

OIDC (OpenID Connect) Authorisierungsframework welches vom KIT genutzt wird um dritten Webseiten Logins basierend auf KIT-Konten bereitzustellen. 16, 17, 36, 56

PostgreSQL Objekt-Relationales Datenbankmanagementsystem, welches zum Speichern und Verwalten von Daten verwendet wird. 5, 6, 56

REST (Representational State Transfer) Architekturstil von APIs für das Internet. 15

Services Klassen, welche die Businesslogik enthalten und über das Data-Layer abstrahieren. 6

Spring Boot Erweiterung des Spring Frameworks, um Webanwendungen zu entwickeln. 5, 6

Spring Data Erweiterung des Spring Frameworks, um Datenbanken zu verwalten. 5, 56