RoomMate Dokumentation

Einführung und Ziele

Übersicht über die Anforderungen

RoomMate soll Mitarbeiter:innen der Universität - insbesondere jenen im Fachbereich Informatik - die Möglichkeit bieten, Ressourcen (wie zum Beispiel Büroplätze) mit einer bestimmten Hardwareausstattung für Zeiträume zu reservieren.

Neben regulären Benutzer:innen (i.d.R. wissenschaftliche Angestellte) gibt es Personen, welche die Plätze administrieren können. Eine Person kann sowohl normale:r Nutzer:in, als auch Administrator:in sein.

Für jeden Arbeitsplatz in RoomMate werden Informationen darüber gespeichert, in welchem Raum sich der Platz befindet und welche Ausstattung vorhanden ist.

Administrator:innen können die Ausstattung eines Platzes ändern, Plätze für bestimmte Zeiten sperren und Reservierungen stornieren oder überschreiben.

Regulären Benutzer:innen bietet RoomMate Möglichkeit, einen passenden Platz zu finden und Plätze für einen Zeitraum (oder auch wiederkehrende Zeiträume) zu reservieren. Die Suche nach einem Platz bietet Filtermöglichkeiten für Ausstattung und Zeiträume.

RoomMate ist in der Lage dazu, mit dem Berechtigungssystem KeyMaster zu interagieren, mit welchem die elektronischen Schließungen der Universität gesteuert werden kann.

Die offizielle Übersicht über die Anforderungen kann hier aufgerufen werden:

https://github.com/hhu-propra2-ws23/Organisation/blob/main/roommate.adoc

Qualitätsziele nach der ISO Norm 25010

RoomMate orientiert sich an der internationalen Norm für Qualitätskriterien von Software, IT-Systemen und Software-Engineering - der ISO Norm 25010.

Ziel ist ein System, welches funktional, verlässlich, intuitiv und sicher ist.

Übersicht kann hier gefunden werden: https://inztitut.de/blog/glossar/iso-25010/

Ziel	**Beschreibung**	**Konsequenz für RoomMate**
Funktionale Software	• Vollständig hinsichtlich Softwarefunktionen • Funktional korrekt • Angemessene Funktionalität	• RoomMate muss die oben beschriebenen Anforderungen erfüllen • Suche, Reservierung und Verwaltung von Ressourcen muss fehlerfrei möglich sein
Verlässliche Software	• Ausgereifte Softwarequalität • Verfügbarkeit • Fehlertoleranz • Wiederherstellbarkeit	• Es darf keine doppelten Reservierungen geben • Wenn Nutzer:innen eine Buchungsbestätigung erhalten, darf die Buchung nicht verloren gehen
Usability	• Barrierefreiheit • Leicht erlernbar • Gute Bedienbarkeit • Schutz vor Fehlbedienung durch Nutzer • Ästhetisches User-Interface • Leichter Zugang	• RoomMate soll für jegliche Nutzer:innen einfach zu bedienen sein • Bedienung muss sinnvoll ohne Maus und mit Screenreader möglich sein • Neue Nutzer:innen sollten ohne Anleitung innerhalb weniger Minuten lernen können, RoomMate zu nutzen
Höchste Sicherheit	• Datenschutz • Integrität • Nicht manipulierbar • Sichere Administration • *Authentifikation*	• Authentifizierung über OAuth2 • Zugriff auf Ressourcen darf nicht ohne Berechtigung erfolgen • Überprüfung aller Nutzereingaben auf Serverseite

Stakeholder

Stakeholder	**Beschreibung**
Universität	RoomMate wird von der Universität genutzt werden und muss daher ihren Ansprüchen gerecht werden
Propra-Team	Bewerten das Projekt, stehen zur Beratung zur Verfügung

Einschränkungen

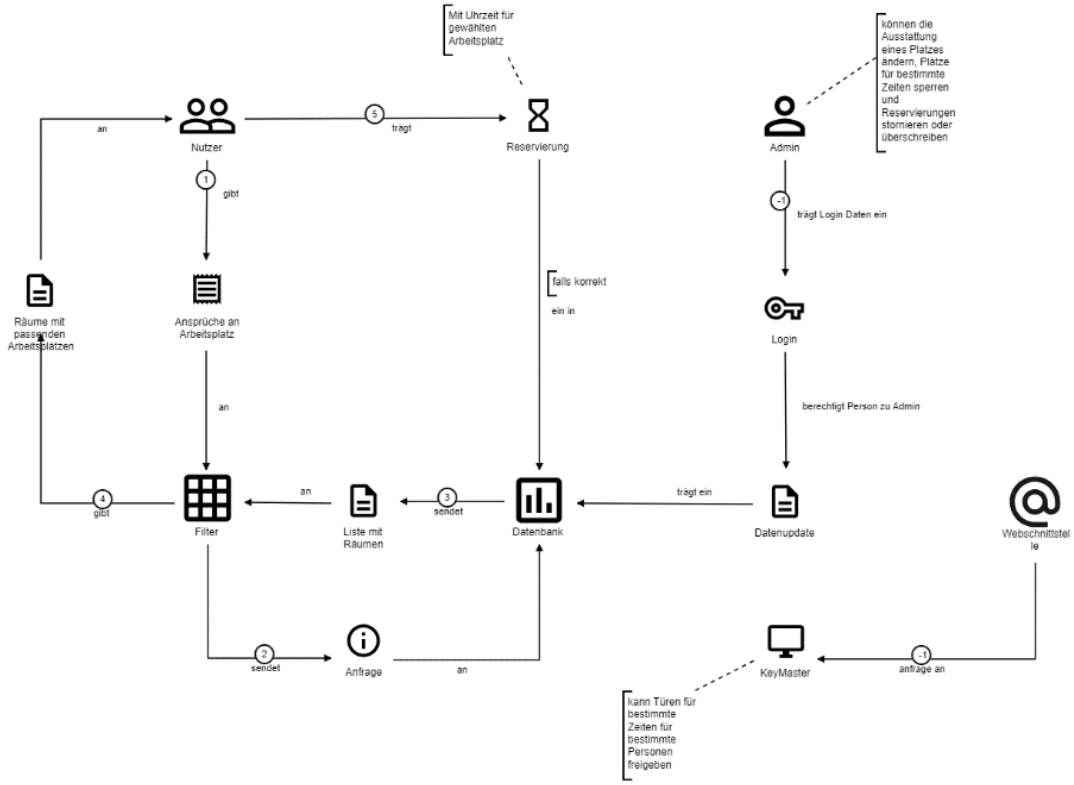
Einschränkung	Erklärung
Zeit	Das System muss während des Semesters und der Klausurphase entwickelt werden. Die Entwicklung muss neben anderen Modulen und der Vorbereitung auf Klausuren erfolgen.
Erfahrung	Die Entwicklung von RoomMate ist das erste größere Softwareentwicklungsprojekt im Team für alle Gruppenmitglieder. Die Gruppenmitglieder haben wenig praktische Erfahrung, haben bisher hauptsächlich mit Java gearbeitet, und müssen in diesem Projekt lernen, ihr theoretisches Wissen aus den vergangenen Semestern anzuwenden.
Personalausstattung	Die gesamte Entwicklung von RoomMate muss durch ein kleines Team bestehend aus drei Gruppenmitgliedern erreicht werden
Plattformunabhängig	RoomMate muss sowohl auf PC-Browsern als auch mobil funktionieren
Barrierefreiheit	Nutzung mit Screenreader und ohne Maus muss ermöglicht sein
Architektur	RoomMate soll ein Self-Contained-System sein, das die Onionarchitektur umsetzt

Weiteres:

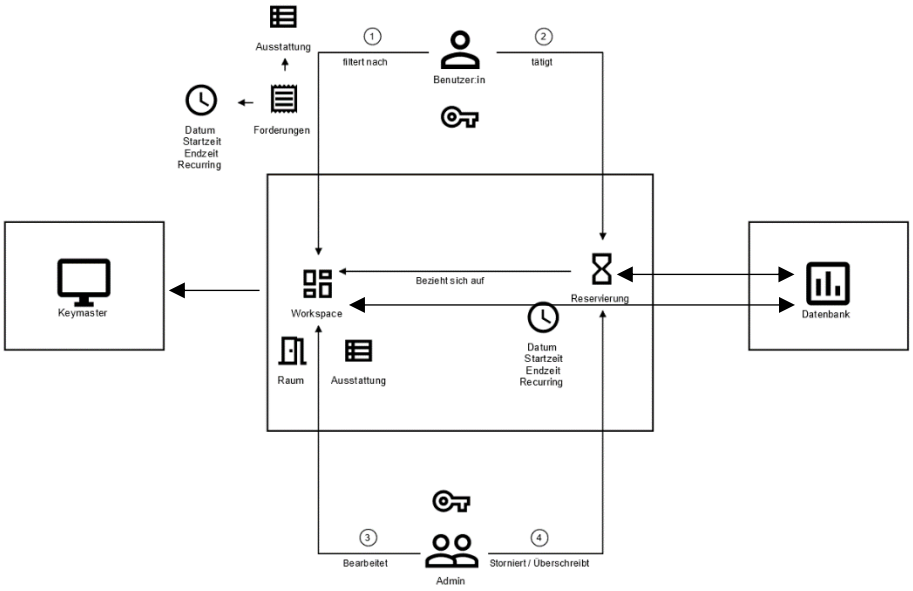
- Integriert mit Gradle build tool
- Ausführbar von der Kommandozeile

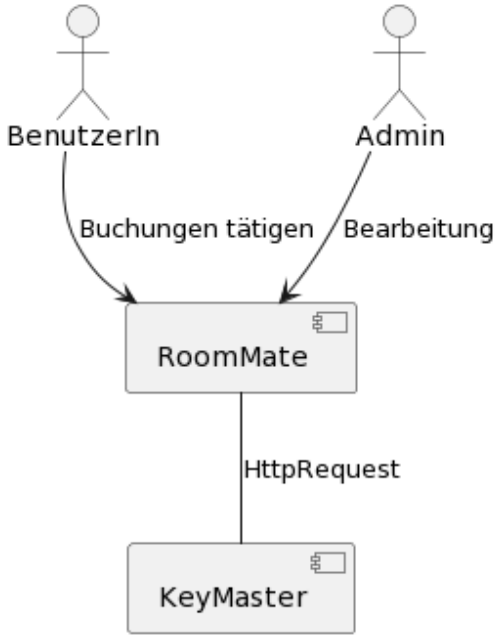
Context und Scope

Ursprüngliches Design, vor Start der Entwicklung:



Endgültiges Design:





Spring (Framework)
Als grundlegendes Framework wird Spring Boot in der Buildversion 3.2.1 verwendet. Entscheidend für die aufbauenden Systembausteine wie Spring Web MVC oder Spring Security.

Spring Security (Fremdsystem - Authentifizierung)
Spring Security wird verwendet, um die Zugriffsverwaltung der Software zu gewährleisten. Als Authentifizierungsclient wird OAuth2 verwendet.

Spring Web MVC (Fremdsystem - Webschnittstelle)
Mithilfe von Spring Web MVC wird eine grafische Webschnittstelle erzeugt, welche den Benutzer:innen ermöglicht, über entsprechende grafische Benutzeroberflächen nach Arbeitsplätzen zu filtern und Reservierungen zu tätigen.

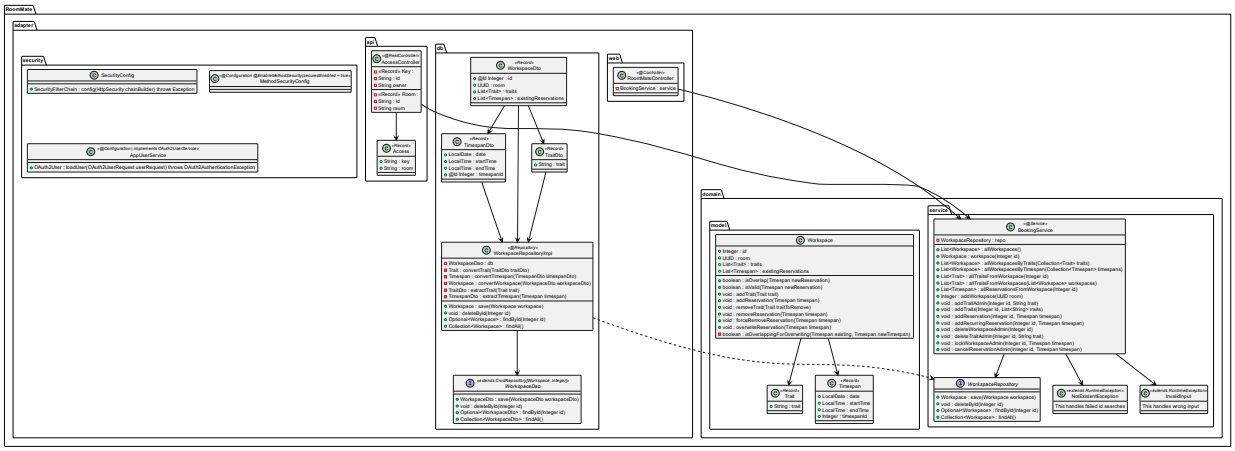
OAuth 2 (Fremdsystem - Authentifizierungsclient)
OAuth2 wird in dieser Anwendung als Authentifizierungssystem genutzt. Hierbei werden Benutzer:innen beim Aufruf der Startseite automatisch auf die Authentifizierungsseite von GitHub weitergeleitet. Wird die Authentifizierung erfolgreich abgeschlossen, so kann der/die Benutzer:in auf alle für ihn/sie bestimmten Webseiten zugreifen.

Thymeleaf (Fremdsystem - dynamische Webseitengenerierung)
Thymeleaf ermöglicht durch Platzhalterwerte und Iterationsmöglichkeiten eine dynamische Webseitengenerierung, welche die für Benutzer:innen relevanten Inhalte auf der Seite rendert.

Jackarta Bean Validation (Fremdsystem - Benutzerinput Validierung)
Wird genutzt um die durch die Webschnittstelle und REST API erhaltenen Benutzereingaben auf Korrektheit zu überprüfen.

REST API (Fremdsystem – Schnittstelle für andere Systeme)
Die Schnittstelle für andere Systeme wird über die REST API gewährleistet. Hierbei werden spezielle Handlermethoden in einem separaten Webcontroller bereitgestellt, welcher die Raum- und Key Abfrage über entsprechende URL-Aufrufe ermöglicht.

Building Block View



Schicht Adapter

Package api

> Access: DTO für die Übertragung von Daten über JSON

> Access Controller: Beeinhaltet die für die REST API notwendigen Handler-methoden

Package web

> Webcontroller: Hauptcontroller, welcher alle Handlermethoden für die Webschnittstelle beinhaltet. Ruft BookingService auf.

Package Security

> AppUserService: Implementiert OAuth2UserService und ist für die SecurityConfig relevant.

> SecurityConfig: Konfiguration für Spring Security, sichert die Webseiten gegen unautorisierten Zugriff ab.

Schicht ApplicationService

> BookingService: Zentraler Service zum Verwalten der Workspace-Aggregate, beinhaltet Geschäftslogik

Schicht Domain

Package model

> Timespan: Modelliert eine Reservierung, indem es ein Datum, eine Startzeit und eine Endzeit festhält.

> Trait: Modelliert ein Hardwareausstattungsdetail bzw. eine Eigenschaft eines Arbeitsplatzes

> Workspace: Aggregate Root.
Modelliert einen Arbeitsplatz; hält eine Raumnummer, eine Liste von Eigenschaften und eine Liste von Reservierungen fest