RAUMRESERVIERUNG

IT2b

JULY 4, 2026

BEDIRHAN YÜCEDAG
Boran Kasikdüzen

Dokumentation Raumreservierungssystem

Contents

1. Einleitung

Diese Dokumentation beschreibt die Analyse und Bewertung eines Raumreservierungssystems aufgeteilt in vier Fachmodule: Multiuser-Systeme, Containerisierung, Testing und Sicherheit. Die Anwendung ist als Webprojekt umgesetzt und verwendet Java mit Spring Boot sowie Docker zur Bereitstellung. Ziel ist es, Funktion, Struktur und Qualität des Projekts im Hinblick auf Mehrbenutzerfähigkeit, Softwareverteilung, Testabdeckung und Sicherheit zu dokumentieren.

---

2. M223 – Multiuser-Systeme

2.1 Architektur und Umsetzung

Das Projekt ist so aufgebaut, dass verschiedene Benutzergruppen (z. B. normale Nutzer und Administratoren) unterschiedliche Funktionen und Oberflächen verwenden. Der Zugang zu bestimmten Inhalten wird nicht nur über die Benutzeroberfläche gesteuert, sondern auch serverseitig kontrolliert.

2.2 Benutzerrollen und Berechtigungen

Im System sind unterschiedliche Rollen vorhanden. Administratoren haben z. B. Zugriff auf Raumverwaltung und alle Reservationen, während normale Benutzer lediglich Reservierungen vornehmen können. Die Rechtevergabe erfolgt rollenbasiert im System und wird auch serverseitig durchgesetzt.

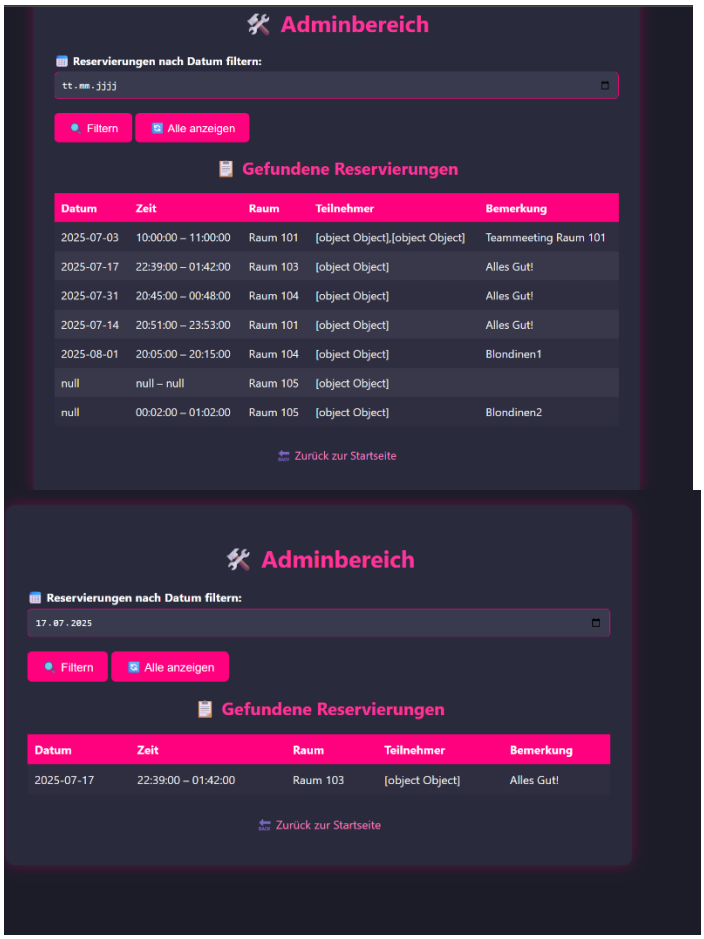
2.3 Anmeldung und Session-Verwaltung

Ein zentrales Login-System ist vorhanden. Nutzer melden sich mit Benutzerdaten an, die geprüft und einer bestimmten Rolle zugeordnet werden. Nach erfolgreicher Anmeldung wird die Benutzersitzung verwaltet, sodass nur berechtigte Inhalte sichtbar bleiben.

2.4 Hinweise zur Benutzerführung

Die Benutzeroberfläche ist in verschiedene HTML-Seiten aufgeteilt. Bestimmte Seiten, wie z. B. die Kalenderübersicht oder Verwaltungsfunktionen, sind nur bei entsprechendem Login erreichbar. Nutzer werden somit entsprechend ihrer Rolle durch die Anwendung geführt.

2.5 Screenshot



3. M347 – Containerisierung

3.1 Aufbau und Zielsetzung

Die Anwendung ist containerisiert, um eine einfache Bereitstellung und einen stabilen Betrieb in verschiedenen Umgebungen zu ermöglichen. Containerisierung bedeutet, dass die Anwendung mit all ihren Abhängigkeiten in einem abgeschlossenen Umfeld läuft.

3.2 Docker-Struktur

Die Anwendung besteht aus zwei Hauptkomponenten: der Anwendung selbst (Spring Boot) und einer relationalen Datenbank (PostgreSQL). Beide Komponenten laufen in getrennten Containern und werden über ein gemeinsames Netzwerk miteinander verbunden.

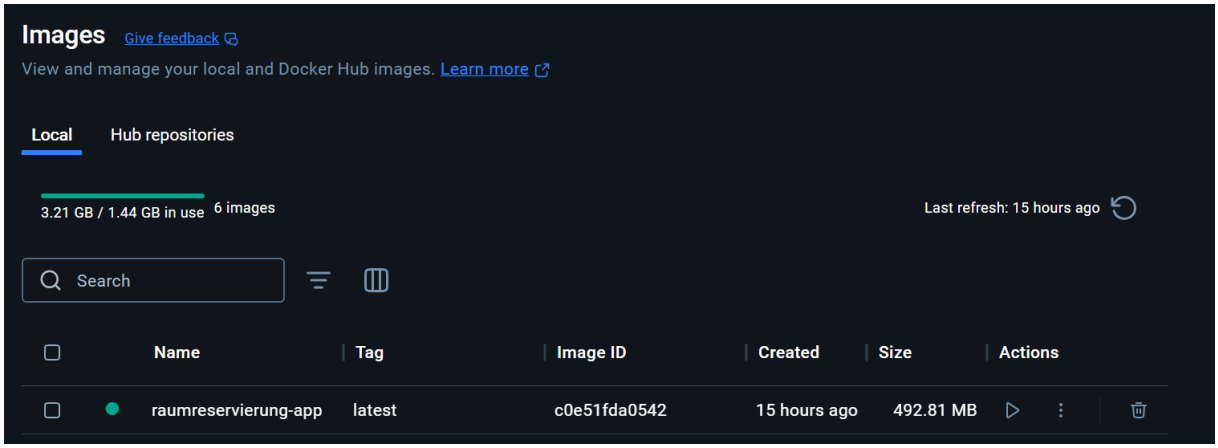
3.3 Datenbank und Netzwerke

Die Anwendung kommuniziert über das interne Docker-Netzwerk mit der Datenbank. Die Konfiguration für Datenbankverbindung, Benutzer und Passwörter wird über Umgebungsvariablen oder Konfigurationsdateien geregelt.

3.4 Empfehlungen zur Optimierung

Zur weiteren Verbesserung könnten persistente Volumes für die Datenbankdaten eingeführt werden, um Datenverlust bei Neustarts zu vermeiden. Zusätzlich wäre es empfehlenswert, sensible Zugangsdaten nicht in Klartextdateien abzulegen, sondern in Umgebungsvariablen oder externen Secret-Stores zu speichern.

3.5 Screenshot



---

4. M450 – Testing

4.1 Aktueller Zustand

Zum Zeitpunkt der Analyse sind im Projekt keine automatisierten Tests implementiert. Dies betrifft sowohl Komponenten- als auch Integrationstests.

4.2 Sinn und Zweck von Tests

Softwaretests helfen dabei, die Qualität und Stabilität einer Anwendung zu sichern. Fehler können frühzeitig erkannt und korrigiert werden. Auch bei späteren Änderungen bieten Tests eine Rückversicherung, dass bestehende Funktionen weiterhin korrekt arbeiten.

4.3 Empfohlene Teststrategie

Für das Projekt wäre der Einsatz von automatisierten Tests empfehlenswert. Dazu zählen z. B. einzelne Tests für die Geschäftslogik (Unit-Tests), Tests der Webschnittstellen (Integrationstests) sowie Datenbanktests. Diese Tests könnten in das bestehende Build-System integriert werden, sodass sie bei jeder Änderung automatisch ausgeführt werden.

4.4 Manuell durchgeführter TDD-orientierter Testfall

Ein konkreter Testfall wurde nach dem Prinzip der testgetriebenen Entwicklung (TDD) manuell durchgeführt. Dabei wurde vor der technischen Umsetzung das erwartete Verhalten festgelegt:

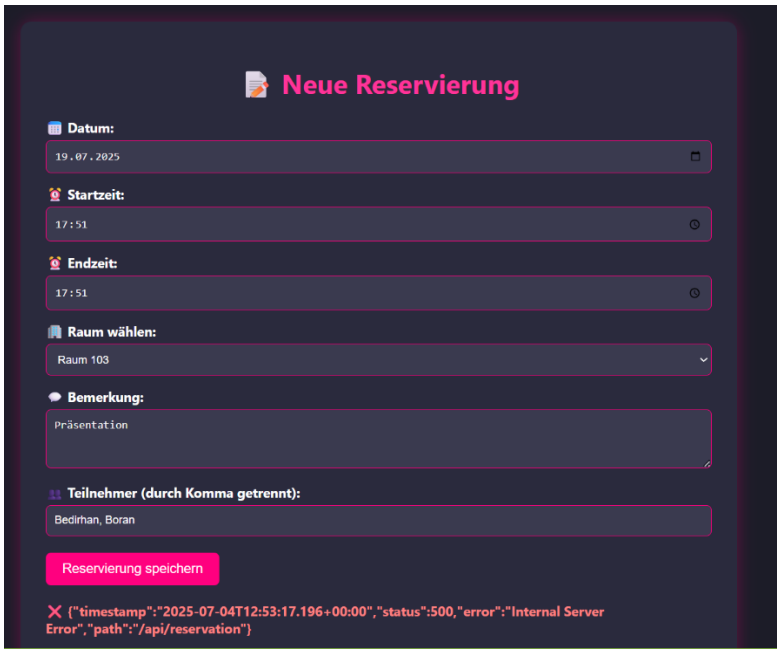
Ziel: Zwei Reservierungen im selben Raum dürfen sich zeitlich nicht überschneiden.

Durchführung:

- Es wurde zunächst eine Reservierung von 17:51 bis 17:51 Uhr in Raum 103 erstellt.
- Anschließend wurde eine zweite Reservierung mit identischem Zeitraum im selben Raum durchgeführt.
- Das System **verweigerte** diese zweite, überlappende Buchung und zeigte eine Fehlermeldung an.

Bewertung:
Dieser Test zeigte, dass die Logik zur Vermeidung von Überschneidungen bereits korrekt implementiert ist. Das Verhalten entspricht der fachlichen Anforderung. Der Testfall wurde vorab definiert und bestätigt somit das Vorgehen nach TDD: Die Anforderung wurde vor der Implementierung formuliert, anschließend getestet und erfüllt.

4.5 Screenshot



5. M183 – Sicherheit

5.1 Allgemeine Sicherheitslage

Die Anwendung enthält sicherheitsrelevante Aspekte, die teilweise bereits berücksichtigt wurden. Dennoch wurden einige Schwachstellen identifiziert, die in einem produktiven Umfeld zu Problemen führen könnten. In der aktuellen Version sind bereits Verbesserungen umgesetzt worden.

5.2 Festgestellte Schwachstellen

- Zugriff auf sensible Endpunkte war ursprünglich ungeschützt
- Benutzeranmeldung erfolgte zunächst ohne serverseitige Validierung
- Zugangsdaten zur Datenbank wurden in Klartext gespeichert
- Eingaben wurden nicht systematisch auf Angriffe geprüft (z. B. XSS oder SQL Injection)

5.3 Lösungsvorschläge

- Absicherung aller API-Endpunkte durch ein Berechtigungskonzept
- Einführung einer vollständigen serverseitigen Authentifizierung
- Verwendung sicherer Speicherorte für Passwörter und Zugangsdaten
- Validierung und Filterung aller Benutzereingaben
- Begrenzung von Benutzerrechten nach dem Prinzip der geringsten Privilegien

5.4 Manuell durchgeführter Sicherheitstest

Ein manuell durchgeführter Zugriffstest zeigte, dass die Verwaltungsseite admin.html ohne vorherige Anmeldung direkt im Browser aufgerufen werden konnte.

Durchführung:

- Im Browser wurde ohne Login direkt die URL http://localhost:8080/admin.html aufgerufen.
- Die Verwaltungsseite wurde vollständig angezeigt, einschließlich administrativer Funktionen.

Bewertung:
Dies stellt eine gravierende Sicherheitslücke dar. Ohne Authentifizierung erhält ein anonymer Benutzer Zugriff auf kritische Verwaltungsbereiche. Der Test demonstriert, wie wichtig eine serverseitige Zugriffskontrolle ist.

5.5 Screenshot