Airport Management System

June 19, 2025

Contents

1 Einleitung

Dieses Dokument beschreibt die Planung und Umsetzung eines Flughafen-Verwaltungssystems, bestehend aus einem Backend mit Spring Boot und einem Frontend mit Vue.js. Ziel ist es, Buchungen, Passagiere, Flüge und Benutzerrollen zu verwalten.

2 Anforderungen

2.1 Funktionale Anforderungen

- Benutzer können sich registrieren und einloggen.
- Kunden können verfügbare Flüge suchen und anzeigen.
- Kunden können Flüge buchen und stornieren.
- Kunden können Passagierdaten erfassen.
- Mitarbeiter können Flüge erstellen, bearbeiten und löschen.
- Mitarbeiter können Passagiere einchecken.
- Für jede Buchung kann ein Check-in mit Gepäck durchgeführt werden.
- Zu jedem Flug sind ein Flugzeug und eine Fluggesellschaft zugeordnet.

2.2 Nicht-funktionale Anforderungen

- Das System verwendet REST-Architektur.
- Backend: Spring Boot, Frontend: Vue.js.
- Passwörter werden verschlüsselt gespeichert.
- Zugriffskontrolle durch Rollen (Customer, Employee, Admin).
- Das System ist responsiv.
- Daten werden in einer relationalen Datenbank gespeichert.
- Kommunikation erfolgt über HTTPS.
- Validierungen verhindern fehlerhafte Eingaben.

3 Use Case Diagramm

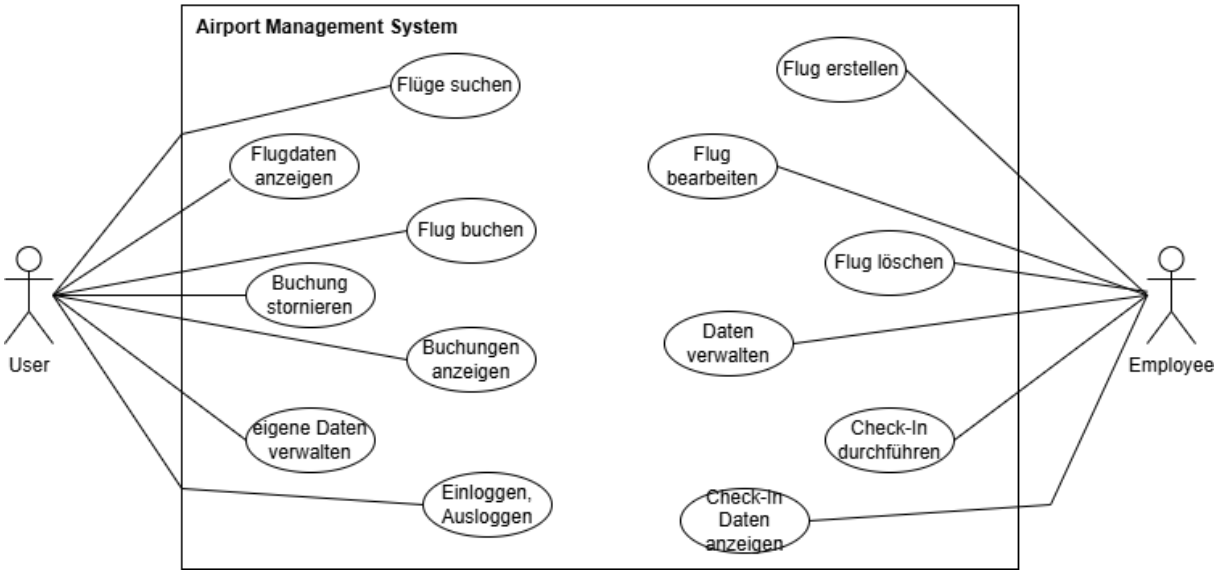

Figure 1: Use Case Diagram

4 Domänenklassen

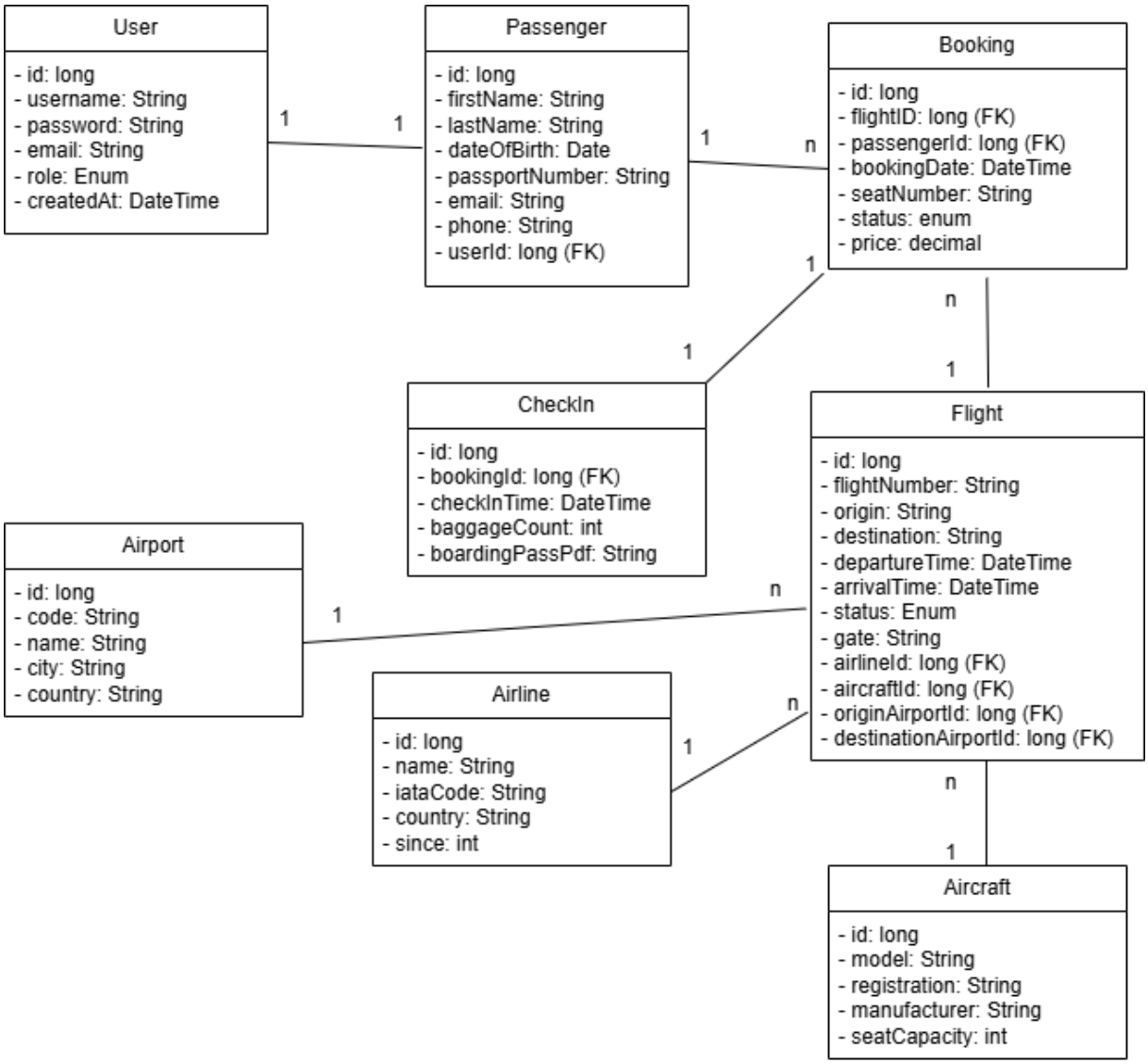

Figure 2: Domain Model

5 REST API Übersicht

5.1 Flight

- `GET /api/flights` – Liste aller Flüge
- `GET /api/flights/{id}` – Einzelnen Flug abrufen
- `POST /api/flights` – Flug erstellen
- `PUT /api/flights/{id}` – Flug bearbeiten
- `DELETE /api/flights/{id}` – Flug löschen

5.2 Passenger

- `GET /api/passengers` – Liste aller Passagiere
- `GET /api/passengers/{id}` – Einzelnen Passagier abrufen
- `POST /api/passengers` – Neuen Passagier erstellen
- `PUT /api/passengers/{id}` – Passagier bearbeiten
- `DELETE /api/passengers/{id}` – Passagier löschen

5.3 Booking

- `GET /api/bookings` – Liste aller Buchungen
- `GET /api/bookings/{id}` – Einzelne Buchung abrufen
- `POST /api/bookings` – Buchung erstellen
- `PUT /api/bookings/{id}` – Buchung ändern
- `DELETE /api/bookings/{id}` – Buchung stornieren

5.4 CheckIn

- `POST /api/checkins` – Check-in durchführen
- `GET /api/checkins/{bookingId}` – Check-in zu einer Buchung abrufen

5.5 Aircraft

- `GET /api/aircrafts` – Liste aller Flugzeuge
- `POST /api/aircrafts` – Neues Flugzeug hinzufügen

5.6 Airline

- `GET /api/airlines` – Liste aller Fluggesellschaften
- `POST /api/airlines` – Neue Fluggesellschaft anlegen

5.7 Airport

- `GET /api/airports` – Liste aller Flughäfen

6 Validierungsregeln

Im Folgenden sind die wichtigsten Validierungen für REST-Anfragen aufgeführt, um fehlerhafte Eingaben zu vermeiden. Bei Verstößen wird jeweils ein HTTP-Status 400 (Bad Request) zurückgegeben.

6.1 Passenger

- `firstName` und `lastName` müssen jeweils mindestens 2 Zeichen lang sein.
- `email` muss eine gültige E-Mail-Adresse sein (z. B. `max@mail.com`).
- `passportNumber` muss eindeutig und alphanumerisch sein (6–9 Zeichen).
- `dateOfBirth` darf nicht in der Zukunft liegen.

6.2 Flight

- `flightNumber` muss eindeutig sein ((z. B. `LH1234`).
- `departureTime` muss vor `arrivalTime` liegen.
- `originAirportId` und `destinationAirportId` müssen existieren und verschieden sein.
- `status` muss ein gültiger Wert sein: `SCHEDULED`, `DELAYED`, `CANCELLED`.

6.3 Booking

- `passengerId` und `flightId` müssen auf existierende Entitäten verweisen.
- `seatNumber` darf für denselben Flug nur einmal vergeben werden.
- Der Flug darf zum Buchungszeitpunkt nicht voll sein.
- `status` darf nur vordefinierte Werte enthalten: `CONFIRMED`, `CANCELLED`, `CHECKED_IN`.

6.4 CheckIn

- `bookingId` muss existieren und darf noch nicht eingecheckt sein.
- `baggageCount` darf maximal 3 betragen.
- `boardingPassPdf` darf nicht leer sein (Pfad oder URL).

6.5 Aircraft

- `model` und `registration` dürfen nicht leer sein.
- `registration` muss eindeutig sein.
- `seatCapacity` muss eine positive Ganzzahl sein (¿ 0).

6.6 Airline

- `name` und `iataCode` dürfen nicht leer sein.
- `iataCode` muss aus genau 2 oder 3 Großbuchstaben bestehen.
- `since` darf kein zukünftiges Jahr sein.

6.7 User

- `username` muss eindeutig und mindestens 4 Zeichen lang sein.
- `password` muss mindestens 8 Zeichen lang sein.
- `email` muss eine gültige E-Mail-Adresse sein.
- `role` darf nur Werte enthalten: `CUSTOMER`, `EMPLOYEE`, `ADMIN`.