**Dokumentacja Flight Booking System**

Autor: Mateusz Kantorski

**1. Wprowadzenie**

**Flight Booking System** to aplikacja umożliwiająca tworzenie, modyfikowanie, przeglądanie oraz usuwanie rezerwacji lotów, lotów oraz pasażerów. Aplikacja jest zaprojektowana w języku Java z wykorzystaniem frameworku Spring Boot. Użytkownicy mogą tworzyć nowe rezerwacje, edytować istniejące rezerwacje i pasażerów, a także anulować rezerwacje i zarządzać lotami.

Celem aplikacji jest automatyzacja procesu rezerwacji biletów lotniczych, umożliwiająca użytkownikom łatwe zarządzanie lotami i rezerwacjami.

**2. Zakres projektu**

**Podstawowe funkcjonalności:**

* **Tworzenie** nowych lotów, pasażerów, rezerwacji.
* **Modyfikacja** istniejących lotów, pasażerów, rezerwacji.
* **Usuwanie** lotów, pasażerów, rezerwacji.
* **Przeglądanie** danych pasażerów, lotów oraz rezerwacji.
* **Sprawdzanie dostępności miejsc** przed dokonaniem rezerwacji.
* **Wysyłanie e-maili** do pasażera po dokonaniu rezerwacji.
* **Walidacja** danych wejściowych.
* **Obsługa** błędów i wyjątków.

**3. Technologie**

* **Język programowania**: Java 21
* **Framework**: Spring Boot wersja 3.3.4
* **Baza danych**: Relacyjna baza danych H2
* **Testowanie**: JUnit, Mockito
* **Docker**: Konteneryzacja aplikacji za pomocą Dockerfile i Docker Compose
* **Swagger**: Dokumentacja API dostępna poprzez Swagger UI

**4. Struktura projektu**

**Foldery i pliki w projekcie:**

* **/src**: Kod źródłowy aplikacji
  + **/src/main/java**: Kody źródłowe aplikacji
  + **/src/main/resources**: konfiguracja (application.yml)
  + **/src/test**: Folder z testami jednostkowowymi
* **/pom.xml**: Plik konfiguracyjny Mavena z zależnościami
* **Pliki Dockerfile i docker-compose.yml:** pliki dokonteneryzacji aplikacji

**5. Instrukcja uruchamiania**

**Przygotowanie środowiska:**

Aby uruchomić aplikację (bez Dockera), należy mieć zainstalowane następujące oprogramowanie:

* Java 21
* Maven wersja 3.8.6 lub wyższa

Sklonuj repozytorium:

1. Sklonuj repozytorium:

git clone https://github.com/Qumpell/FlightBookingSystem.git

cd FlightBookingSystem

1. Skonfiguruj plik .env w katalogu głównym zawierający dane do konfiguracji e-maila i innych ustawień. Plik musi zawierać takie pola, gdzie APP\_MAIL\_USERNAME to adres maila (gmail) i APP\_MAIL\_PASSWORD to hasło aplikacji. Hasło aplikacji można wygenerować na koncie Google po włączeniu weryfikacji dwuetapowej.

APP\_MAIL\_USERNAME=your\_email@gmail.com

APP\_MAIL\_PASSWORD=your\_password

**Uruchamianie aplikacji bez Docker:**

Uruchom aplikację za pomocą Mavena za pomocą komendy:

mvn clean spring-boot:run

**Uruchamianie aplikacji z Dockerem:**

1. Za pomocą Docker Compose uruchom i zbuduj aplikację:

docker-compose up --build

1. Po uruchomieniu aplikacji dostęp do API uzyskasz pod adresem:

http://localhost:8080

**6. Instrukcja API**

**Endpointy:**

**1. Loty**

* **GET** /api/v1/flight – Pobierz wszystkie loty
* **POST** /api/v1/flight – Tworzenie nowego lotu
* **PUT** /api/v1/flight/{id} – Modyfikacja lotu o podanym ID
* **DELETE** /api/v1/flight/{id} – Usunięcie lotu (nie usuwa jeśli istnieje dla niego rezerwacja)

**2. Pasażerowie**

* **GET** /api/v1/passenger – Pobierz wszystkich pasażerów
* **POST** /api/v1/passenger – Tworzenie nowego pasażera
* **PUT** /api/v1/passenger/{id} – Modyfikacja pasażera o podanym ID
* **DELETE** /api/v1/passenger/{id} – Usunięcie pasażera (nie usuwa jeśli istnieje dla niego rezerwacja)

**3. Rezerwacje**

* **GET** /api/v1/reservation – Pobierz wszystkie rezerwacje
* **POST** /api/v1/reservation – Tworzenie nowej rezerwacji
* **PUT** /api/v1/reservation/{id} – Modyfikacja rezerwacji o podanym ID
* **DELETE** /api/v1/reservation/{id} – Usunięcie rezerwacji

**Przykłady odpowiedzi:**

**Tworzenie nowej rezerwacji (POST):**

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

**Błąd, gdy miejsce już zajęte (BadRequestException):**

Obraz zawierający tekst, zrzut ekranu, Czcionka, wyświetlacz

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

**7. Testy jednostkowe**

Projekt zawiera testy jednostkowe weryfikujące poprawność działania aplikacji.

**8. Obsługa błędów i wyjątków**

Aplikacja wykorzystuje niestandardowe wyjątki, które są rzucane w przypadku błędów, takich jak:

* **BadRequestException** – Gdy użytkownik próbuje zarezerwować już zajęte miejsce.
* **PassengerNotFoundException** – Gdy nie znaleziono pasażera.
* **FlightNotFoundException** – Gdy nie znaleziono lotu.
* **ReservationNotFoundException** – Gdy nie znaleziono rezerwacji.
* **MailException** – Gdy wystąpił błąd z wysyłaniem emaila.

Te wyjątki są obsługiwane i zwracają odpowiedni kod statusu HTTP oraz szczegóły błędu w odpowiedzi.

**9. Przykładowe dane**

Po uruchomieniu aplikacji, system automatycznie inicjalizuje przykładowe dane w bazie danych w pamięci (**H2**).

Dodanych zostaje:

* 5 przykładowych **lotów**
* 5 przykładowych **pasażerów**
* 5 przykładowych **rezerwacji**

Dzięki temu możliwe jest natychmiastowe testowanie funkcjonalności systemu (np. poprzez interfejs Swaggera) bez konieczności ręcznego wprowadzania danych.

**10. Dostęp do bazy danych (H2 Console)**

Aplikacja korzysta z bazy danych H2 w trybie pamięciowym, co pozwala na szybki podgląd danych bez potrzeby konfiguracji zewnętrznej bazy.

Po uruchomieniu aplikacji można uzyskać dostęp do konsoli H2 i podejrzeć dane w tabelach.

Aby to zrobić, wystarczy przejść do:

<http://localhost:8080/h2-console>

**Domyślne dane logowania:**

* **URL:** jdbc:h2:mem:mydb
* **Login:** sa
* **Hasło:** password

Konsola pozwala na przeglądanie i zarządzanie danymi w bazie H2 bezpośrednio z przeglądarki.