Lot-zadanie rekrutacyjne – Java Developer Dokumentacja

Autor: Paweł Martyniuk

1. Treść zadania

Celem jest opracowanie prostego systemu do zarządzania bazą danych lotów i pasażerów. Napisz program w języku Java (8+) , który będzie umożliwiał operacje CRUD na danych lotów i pasażerów. Program powinien umożliwiać dodawanie, odczytywanie, aktualizowanie i usuwanie informacji o lotach oraz pasażerach oraz pozwalać na dodanie/usunięcie pasażera do konkretnego lotu. Klasa `Flight` powinna zawierać podstawowe informacje o locie, takie jak numer lotu, trasę, datę i godzinę wylotu, ilość dostępnych miejsc. Klasa `Passenger` powinna zawierać podstawowe informacje o pasażerze, takie jak imię, nazwisko, numer telefonu.

2. Środowisko

Aplikacja została napisana z wykorzystaniem Java w wersji 21. Oprócz tego wykorzystano bazę danych PostgreSQL w wersji 16. Baza danych została skonteneryzowana przy użyciu Dockera. Wykorzystano środowisko Intellij Idea.

Kod źródłowy dostępny jest na moim githubie:

https://github.com/wshknmt/Lot-management-system

3. Instrukcja uruchomienia

3.1. Uruchomienie bazy danych

W katalogu z plikiem docker-compose.yaml należy uruchomić komendę:

> docker compose up

Za pomocą powyższej komendy zostanie uruchomiona lokalnie baza danych postgreSQL na porcie 5432. Oprócz tego skonfigurowano narzędzie PGAdmin4 za pomocą którego można przeglądać i edytować bazę danych z poziomu przeglądarki. Uruchamiam to na porcie 8888. Logowanie zgodnie z danymi w pliku docker-compose.yml, czyli email: admin@admin.com, hasło: root

3.2. Uruchamianie aplikacji

> mvn install

> java -jar target/Lot-zadanie-1.0-SNAPSHOT.jar

4. Sposób rozwiązania

Po uruchomieniu aplikacji należy wpisać jedną z poniższych komend w konsoli w celu interakcji z systemem. Następnie poszczególne parametry należy wpisywać pojedynczo zgodnie z informacjami wyświetlanymi na ekranie.

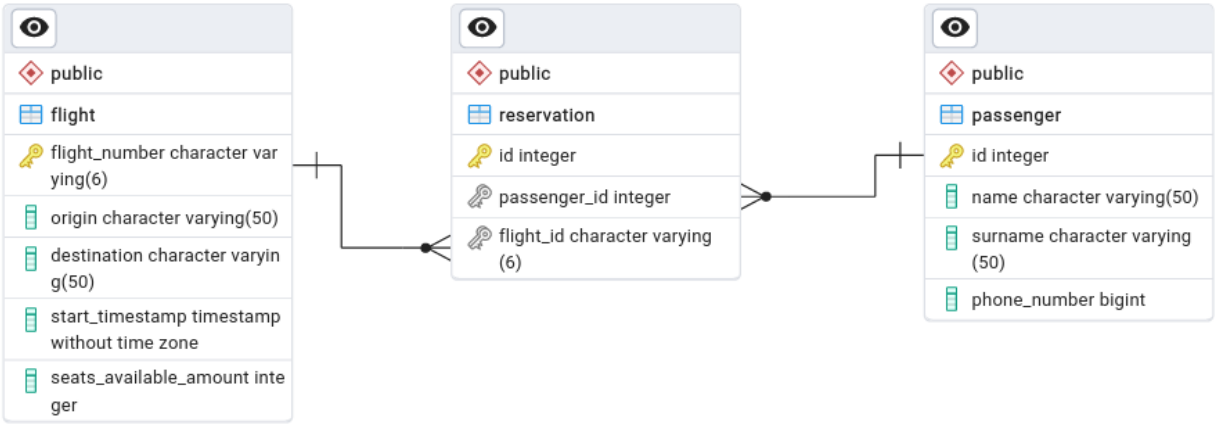
4.1. Dostępne funkcjonalności:

- **add_flight** – dodaje nowy lot do bazy. Obowiązkowo należy podać numer lotu, miejsce wylotu, miejsce docelowe, data wylotu, godzina wylotu oraz liczbę dostępnych miejsc siedzących. Walidacja numeru lotu polega na sprawdzeniu czy wprowadzona długość jest mniejsza lub równa 6, oraz czy znaki od 3 pozycji są numeryczne. Data i godzina wylotu są walidowane czy zostały podane w prawidłowym formacie (RRRR:MM:DD HH:MM). Natomiast walidacja liczby miejsc siedzących polega na sprawdzeniu czy wartość jest numeryczna i czy nie jest mniejsza od zera.
- **delete_flight** – usunięcie lotu o podanym numerze lotu. Obowiązkowo należy podać numer lotu. Walidacja numeru jak w poprzednim punkcie. Oprócz tego jest sprawdzane czy lot o takim numerze faktycznie istnieje i jeżeli nie, to użytkownik jest powiadamiany specjalnym komunikatem. Oprócz tego usunięcie lotu również nie będzie możliwe gdy została już dokonana jakaś rezerwacja na ten lot. W celu usunięcia takiego lotu najpierw należy anulować rezerwacje.
- **add_passenger** – dodanie pasażera do bazy. Obowiązkowo należy podać imię, nazwisko oraz numer telefonu pasażera. Walidacja numeru sprawdza czy podana wartość jest numeryczna.
- **delete_passenger** – usunięcie pasażera o podanym Id. Obowiązkowo należy podać Id pasażera. Oprócz tego jest sprawdzane czy pasażer o takim Id faktycznie istnieje i jeżeli nie to użytkownik jest powiadamiany specjalnym komunikatem. Oprócz tego usunięcie pasażera również nie będzie możliwe gdy została już dokonana jakaś rezerwacja dla tego pasażera. W celu usunięcia takiego pasażera najpierw należy anulować rezerwacje.
- **search_flights** – wyszukanie lotów o zadanych kryteriach. Podanie wartości poszczególnych kryteriów jest opcjonalne. Można wpisać wartości następujących kryteriów: Numer lotu, miejsce wylotu, miejsce docelowe, data i godzina wylotu oraz liczba dostępnych miejsc. Walidowane są w tym przypadku tylko data i godzina oraz liczba dostępnych miejsc i w przypadku niepoprawnych wartości te kryteria zostaną pominięte. Do budowania zapytania został wykorzystany dynamiczny SQL to znaczy, że w zapytaniu znajdują się tylko te kryteria które zostały podane przez użytkownika. W przypadku gdy użytkownik nie wypełni żadnego kryterium wtedy zostaną zwrócone wszystkie rekordy z tabeli.
- **search_passengers** - wyszukanie pasażerów o zadanych kryteriach. Podanie wartości poszczególnych kryteriów jest opcjonalne. Można wpisać wartości następujących kryteriów: id pasażera, imię, nazwisko oraz numer telefonu. Walidowane są w tym przypadku tylko id oraz numer telefonu i w przypadku niepoprawnych wartości te kryteria zostaną pominięte. Do budowania zapytania został wykorzystany dynamiczny SQL to znaczy, że w zapytaniu znajdują się tylko te kryteria które zostały podane przez użytkownika. W przypadku gdy użytkownik nie wypełni żadnego kryterium wtedy zostaną zwrócone wszystkie rekordy z tabeli.
- **search_reservations** - wyszukanie rezerwacji o zadanych kryteriach. Podanie wartości poszczególnych kryteriów jest opcjonalne. Można wpisać wartości następujących kryteriów: id rezerwacji, id pasażera oraz numer lotu. Walidowane są w tym przypadku są wszystkie pola i w przypadku niepoprawnych wartości te kryteria zostaną pominięte. Do

budowania zapytania został wykorzystany dynamiczny SQL to znaczy, że w zapytaniu znajdują się tylko te kryteria które zostały podane przez użytkownika. W przypadku gdy użytkownik nie wypełni żadnego kryterium wtedy zostaną zwrócone wszystkie rekordy z tabeli. Oprócz tego w przypadku tego zapytania zastosowałem złączenie wszystkich tabel za pomocą klauzuli JOIN, aby dane wyświetlane na ekranie były bardziej zrozumiałe dla użytkownika.

- **book_flight** – rezerwacja miejsca w locie dla konkretnego pasażera. Obowiązkowo należy podać id użytkownika oraz numer lotu, które również są walidowane. Dla id czy jest to wartość numeryczna, a dla numeru lotu zgodnie z zasadami opisanymi w poprzednich punktach. Następnie następuje sprawdzenie, czy faktycznie pasażer i lot o podanych identyfikatorach faktycznie istnieje. Jeżeli tak to sprawdzane jest, czy w wybranym locie są jeszcze dostępne miejsca i jeżeli tak to rezerwacja jest dodawana do tabeli z rezerwacjami, a liczba miejsc dla danego lotu jest aktualizowana w tabeli z lotami.
- **cancel_flight** – anulowanie rezerwacji w locie dla konkretnego pasażera. Obowiązkowo należy podać id rezerwacji, które jest walidowane czy jest wartością numeryczną. Następnie jest sprawdzane czy rezerwacja faktycznie istnieje i jeżeli tak to usuwany jest wpis na temat tej rezerwacji z tabeli z rezerwacjami oraz aktualizowana jest wartość dostępnych miejsc w locie z tej rezerwacji.
- **update_flight** – aktualizacja danych wybranego lotu. Podanie numeru lotu obowiązkowe, pozostałe wartości są opcjonalne w zależności jakie dane potrzeba zaktualizować. Do budowania zapytania został wykorzystany dynamiczny SQL to znaczy, że w zapytaniu znajdują się tylko te wartości które zostały podane przez użytkownika. Po sprawdzeniu czy istnieje taki lot w bazie dane są aktualizowane.
- **update_passenger**– aktualizacja danych wybranego pasażera. Podanie id pasażera obowiązkowe, pozostałe wartości są opcjonalne w zależności jakie dane potrzeba zaktualizować. Do budowania zapytania został wykorzystany dynamiczny SQL to znaczy, że w zapytaniu znajdują się tylko te wartości które zostały podane przez użytkownika. Po sprawdzeniu czy istnieje taki pasażer w bazie dane są aktualizowane.
- **help** – za pomocą tej komendy można wyświetlić wszystkie dostępne komendy
- **exit** – po wybraniu tej komendy następuje koniec działania programu.

4.2. Baza danych



Rysunek 1 Wygenerowany schemat bazy danych

Na rysunku 1 przedstawiłem schemat wykorzystanej w aplikacji bazy danych wygenerowany przez narzędzie PGAdmin4. Oprócz zdefiniowanych w poleceniu tabel „Flight” oraz „Passenger” dodałem klasę „Reservation” wewnątrz której przechowuje rezerwacje poszczególnych klientów na poszczególne loty.

5. Testowanie

Podczas uruchamiania bazy danych w docker-compose zdefiniowałem plik init.sql, który po utworzeniu bazy od razu stworzy wymagane struktury i każda z tabel zostanie zasilona 10 różnymi testowymi rekordami. Z uwagi na niedostateczną ilość czasu testowanie działania aplikacji ograniczyłem do wpisywania w konsoli różnych komend i sprawdzania czy operacja spowodowała zamierzony efekt weryfikując zawartość tabeli przy użyciu narzędzia PGAdmin4.

Wszystkie przedstawione funkcjonalności podczas testowania działają zgodnie z zamierzonym celem. W szczególności wyświetlane są specjalne komunikaty w przypadku wpisania niepopranych danych, braku danych w przypadku próby ich modyfikowania lub tworzenia rezerwacji. Oprócz tego w przypadku braku wolnych miejsc użytkownik jest informowany o tym specjalnym komunikatem. W danych testowych umieszczono dla lotu o numerze TP1038 tylko 3 wolne miejsca, dzięki czemu można łatwo sprawdzić że dla 3 pasażerów proces tworzenia nowej rezerwacji zakończy się pomyślnie, a gdy czwarta osoba będzie próbowała zarezerwować lot, wtedy operacja nie zostanie wykonana a użytkownik o tym fakcie powiadomiony specjalnym komunikatem. Oczywiście anulowanie rezerwacji odblokowuje zajęte wcześniej miejsca.