Bazy danych – projekt

Tomasz Kajda
Przemysław Kociuba

System sprzedaży biletów kolejowych

Wymagania aplikacji

- Możliwość wyszukania najszybszego połączenia pomiędzy wybranymi miastami (z uwzględnieniem przesiadek)
- Możliwość zakupu biletu przy uwzględnieniu rodzaju wagonu (przedziałowy/bez przedziałów)
- Możliwość rejestracji użytkownika, jak i zakupu bez rejestracji
- W przypadku rejestracji, możliwość wglądu w historię swoich biletów
- Określa uprawnienia użytkowników (dodawanie/usuwanie przejazdów)
- Przechowuje historię logowań

Architektura i proponowane technologie

- Relacyjna baza danych przechowująca dane o użytkownikach (PostgreSQL)
- Dokumentowa baza danych przechowująca dane o trasach pociągów, miejscach i zakupionych biletach (MongoDB)
- Zamodelowany obiektowo backend aplikacji, pośredniczący pomiędzy frontendem i obiema bazami (Java i Hibernate do ułatwienia obsługi danych z relacyjnej bazy)
- Zamodelowany obiektowo frontend aplikacji (React)

Harmonogram

1. 29.03.2022 – przygotowanie środowiska, wstępny zarys
2. 05.04.2022 – określenie proponowanej podstawowej struktury bazy z informacjami o pociągach
3. 19.04.2022 – implementacja bazy i przygotowanie do połączenia z backendem
4. 30.05.2022 – gotowy frontend
5. 10.04.2022 – gotowy backend
6. 17.04.2022 – powiązanie wszystkich elementów architektury
7. 24.05.2022 – wykonanie ostatnich poprawek, prezentacja

Proponowana struktura bazy TrainsInfo (MongoDB)

Kolekcja *TrainRoute* – pola dokumentów (identyfikowanych trainRouteID)

- *TrainID : Int*
- *DepartureTime : Tablica map (Station : Str -> depart_time Date)*
- *ArrivalTime: Tablica map(Station : Str -> arriv_time Date)*

Kolekcja ticket – pola dokumentów (identyfikowanych ticketID)

- *Name : Str*
- *Surname : Str*
- *Email : Str*
- *trainRouteID : Int*
- *(opcjonalne) id_użytkownika : Int*

Kolekcja train – pola dokumentów

- *Places_in_cabin_cars : Int*
- *Places_in_non_cabin_cars : Int*

Proponowany front-end (React)

8 podstron – strona główna, formularz rejestracji, formularz logowania, formularz zakupu biletu, historia biletów, Strona wprowadzenia danych pasażera/logowania się, formularz dodawania i listowania przejazdów (dla niektórych użytkowników)

Strona główna zawiera odnośniki do formularza rejestracji, logowania, zakupu biletu

Formularz rejestracji pobiera wszystkie dane opisane w tabeli Users

Formularz logowania pobiera login i hasło

Formularz zakupu biletów pobiera:

- Stację początkową
- Stację końcową
- Opcje wyszukiwania z przesiadkami (zawsze będą to max. 3 przesiadki)
- Datę wyjazdu
- Wybór czy miejsce ma być w wagonie z przedziałami czy bez przedziałów

Po wypełnieniu formularza i otrzymaniu odpowiedzi z backendu z listą proponowanych połączeń, użytkownik wybiera jedno z nich (o ile istnieje). Po wybraniu wypełnia dane (imię, nazwisko, email) lub się loguje. Wypełnienie tych pól (lub zalogowanie się) potwierdza rezerwację biletu (lub biletów w przypadku podróży z przesiadkami).

Strona historii biletów (dostępna tylko dla zalogowanych użytkowników).

Po otrzymaniu odpowiedzi z backendu listuje wszystkie bilety:

- Id biletu
- Stację początkową
- Stację końcową
- Datę wyjazdu
- Informację o typie miejsca (w wagonie przedziałowym/bezprzedziałowym)

Strona dodawania przejazdu (dostępna dla użytkowników o uprawnieniach dodania przejazdu)

Jest formularzem w którym są pola:

- Data wyjazdu
- Pociąg
- Stacja
- Godzina przyjazdu na stacje
- Godzina odjazdu ze stacji
  (można przyciskiem dodać kolejną pustą trójkę ostatnich pól aby wprowadzić informację o kolejnym punkcie trasy)

Strona listowania przejazdu

Listuje wszystkie przyszłe przejazdy, przy każdym istnieje przycisk do ich usuwania

Proponowany back-end (Java)

Oferuje API do:

- *Rejestracji użytkownika (zwraca informację o powodzeniu)*
- *Zalogowania użytkownika (zwraca informację o powodzeniu)*
- *Zapytania dotyczącego trasy (zwraca listę proponowanych połączeń uwzględniając dostępność biletów)*
- *Akceptacji proponowanego po wcześniejszym zapytaniu biletu (zwraca informację o powodzeniu)*
- *W bazie obsługiwanej przez hibernate przechowuje informacje o sesjach i kluczach użytkowników do API*
- *Kontroluje czy dany użytkownik może dodać lub usunąć przejazd*