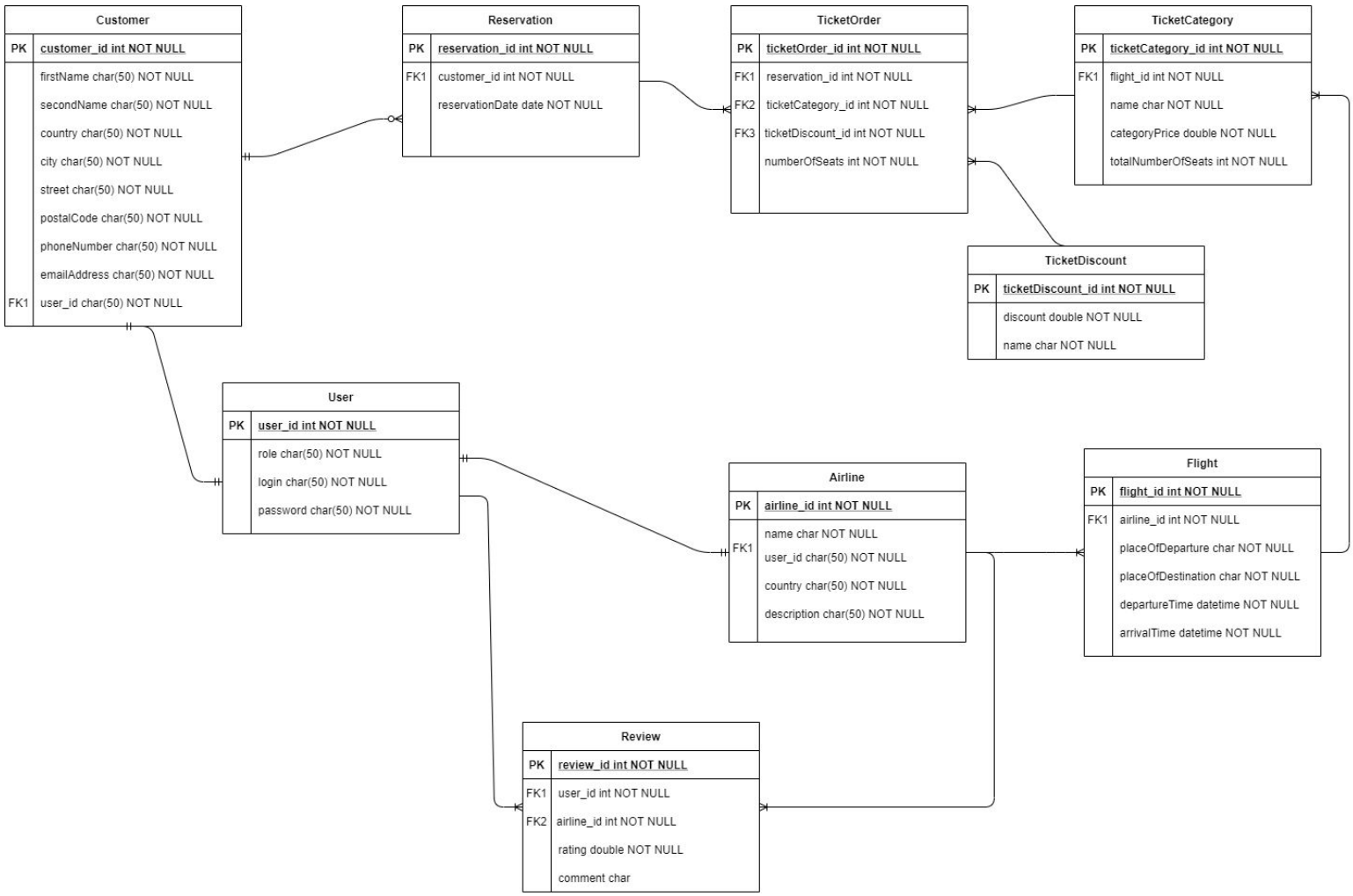
Technologie obiektowe 2020

Projekt - grupa czwartek 16:15

Bartosz Kaszuba, Tomasz Kozyra, Kacper Rosiak, Amadeusz Szabała

1. Model danych

1.1. Schemat modelu danych



1.2. Opis struktur modelu

Airline

Tabela przechowuje informacje o liniach lotniczych. Pole description może pozostać puste. Kluczem głównym jest airline_id. Klucz obcy to user_id.

- Airline_id - Jest to pole przechowujące ID unikalne dla każdej linii lotniczej w bazie.
- Name - Nazwa linii lotniczej
- Country - kraj
- Description - Opcjonalny opis linii lotniczej.

Flight

Tabela przechowuje informacje o lotach. Klucz główny to Fligh_ID. Klucz obcy to airline_id. Każda linia lotnicza może posiadać wiele lotów. Wszystkie pola nie mogą być puste.

- Flight_id - Pole przechowujące ID lotu. Dla każdego lotu ID jest unikalne.
- Airline_id - ID linii lotniczej obsługującej dany lot.
- Departure - Miejsce wylotu
- Destination - Miejsce przylotu
- departureDate - Dzień wylotu
- departureTime - Czas wylotu
- arrivalDate - Dzień wylotu
- arrivalTime - Czas przylotu

Review

Tabela przechowująca oceny oraz komentarze do ocen tworzone przez użytkowników. Kluczem głównym jest review_id, a kluczami obcymi airline_id oraz user_id. Pole comment można zostawić puste.

- Review_id - Pole przechowujące ID oceny.
- User_id - ID użytkownika, który napisał daną ocenę.
- Airline_id - ID linii lotniczej, dla której dana ocena została napisana.
- Rating - Wartość liczbowa oceny.
- Comment - Komentarz do oceny. Nie jest on obowiązkowy.

User

Tabela przechowującą użytkowników zarejestrowanych w naszym systemie. Kluczem głównym jest user_id. Wszystkie pola nie mogą być puste.

- User_id - Unikalny klucz przypisany każdemu użytkownikowi
- Role - uprawnienia użytkownika w systemie
- Login - login użytkownika
- Password - hasło użytkownika. Powinno być zaszyfrowane.

Customer

Tabela przechowująca dane prywatne użytkowników. Kluczem głównym jest customer_id. Wszystkie pola nie mogą być puste.

- customer_ID - Unikalny klucz przypisany każdemu prywatnemu użytkownikowi, który może dokonywać rezerwacji lotów. Różnica między customer_ID, a user_ID jest taka, że każdy Customer ma user_ID, ale nie każdy User ma customer_ID. Może być User, który jest odpowiedzialny za linie lotniczą i ma airline_ID. Customer może mieć kilka rezerwacji.
- user_ID - jest to ID użytkownika wskazujące na jego konto oraz uprawnienia.
- Username - nazwa użytkownika przypisana danemu klientowi
- Pozostałe pola to są dane użytkownika, jego adres oraz mail i numer telefonu.

Reservation

Tabela przechowująca informacje o rezerwacjach klienta. Klucz główny to reservation_id, a klucz obcy customer_id. Wszystkie pola nie mogą być puste.

- Reservation_id - Unikalny klucz rozróżniający rezerwacje w systemie.
- Customer_id - Pole przechowujące ID klienta, który dokonał rezerwacji.
- reservationDate - Pole przechowujące datę dokonanej rezerwacji.

TicketOrder

Tabela przechowuje informacje o naszej rezerwacji. Dla każdej rezerwacji wybieramy sobie klasę biletów np biznes i ilość rezerwowanych biletów. Jeżeli np chcemy zarezerwować 2 bilety klasy biznes oraz 3 klasy ekonomicznej to dostaniemy dwa wpisy w TicketOrder. Kluczem głównym jest ticketOrder_ID, a kluczami obcymi reservcation_id, ticketCategory_id oraz ticketDiscount_id. Wszystkie pola nie mogą być puste.

- ticketOrder_id - Unikalny klucz dla każdej rezerwacji.
- reservation_ID - ID rezerwacji. Do jednego ID może być przypisane kilka TicketOrder.
- ticketCategory_ID - ID kategorii, dla której rezerwujemy bilety.
- ticketDiscount ID - ID zniżki, którą wybraliśmy.

- numberOfSeats - ilość miejsc, które rezerwujemy.

TicketDiscount

Tabela przechowująca informacje o wszystkich zniżkach. Wszystkie pola nie mogą być puste.

- ticketDiscount_ID - Unikalny klucz dla każdej zniżki.
- Discount - Wartość zniżki.
- Name - nazwa zniżki

TicketCategory

Tabela przechowuje informacje o klasach biletów dostępnych dla danego lotu. Każdy lot może mieć przypisane kilka wpisów w TicketCategory w zależności od ilości dostępnych klas. Tabela ta przechowuje również bazową cenę biletów dla danej kategorii oraz ilość miejsc przeznaczonych na daną klasę w danym locie. Wszystkie pola nie mogą być puste. Kluczem głównym jest ticketCategory_Id, a kluczem obcym flight_ID.

- ticketCategory_ID - Unikalny klucz dla każdej kategorii.
- flight_ID - Klucz lotu, dla którego opisywane są klasy.
- Name - Nazwa kategorii
- categoryPrice - Bazowa cena biletu w danej klasie. Na jej podstawie obliczane są zniżki.
- totalNumberOfSeats - Ilość miejsca przeznaczonych na daną klasę w danym locie.

2. Struktura projektu

2.1. Model

W katalogu *model* znajdują się klasy definiujące model danych w aplikacji. Klasy zostały przygotowane do podłączenia do bazy danych (pola mają potrzebne adnotacje, klucze obce, definicje relacji itd.), lecz połączenie z bazą danych nie zostało jeszcze zrealizowane.

2.2. Repository

W katalogu *repository* znajdują się klasy odpowiedzialne za pobieranie danych z bazy danych. Klasy te pozwalają na wykonywanie na podłączonej bazie danych operacji CRUDowych za pomocą klas modelu. Ponadto, repozytoria pozwalają na definiowanie metod służących do

pobierania danych spełniających jakieś wymagania (np. metoda *Airline findByName(String name)* w klasie AirlineRepository, pozwalająca na pobranie linii lotniczej o podanej nazwie).

2.3. Service

W katalogu *service* znajdują się klasy, które są pośrednikami pomiędzy modelem a repozytoriami.

- AirlineService,
- CustomerService,
- FlightService,
- ReservationService,
- TicketCategoryService,
- TicketOrderService,
- MockDataService - posiada metody pozwalające na zapełnianie bazy danych przy starcie aplikacji. Potrzebne ze względu na to, że korzystamy z wbudowanej bazy danych która resetuje się przy każdym uruchomieniu aplikacji.

2.4. Controller

W katalogu *controller* znajdują się klasy odpowiedzialne za pośrednictwo pomiędzy widokami a modelem. Klasy *controller* obsługuję logikę wyświetlanego widoku, służą do walidacji pól formularza, routingu między poszczególnymi widokami, a także do interakcji z użytkownikiem (obsługiwanie naciśnięcia przycisków).

2.5. View

W katalogu *view* znajdują się widoki *.fxml*. Widoki zostały podzielone na 3 kategorie.

1. Widoki dla użytkownika niezalogowanego (anonymous)
    - Widok główny (AnonymousMainView)
    - Widok listy przewoźników (AnonymousAirlinesView)
        - Użytkownik niezalogowany może jedynie przeglądać przewoźników
    - Widok listy lotów (AnonymousFlightView)
        - Użytkownik niezalogowany może jedynie przeglądać loty - nie może dokonywać na nie rezerwacji
2. Widoki dla użytkownika zalogowanego (user)
    - Widok główny (MainView)
    - Widok listy przewoźników (UserAirlinesView)
    - Widok listy lotów (UserFlightView)
    - Widok listy rezerwacji (UserReservationView)
        - Użytkownik zalogowany może przeglądać jedynie **swoje** rezerwacje
    - Widok formularza do dodawania/edycji rezerwacji (AddReservationView)
        - Użytkownik zalogowany może składać rezerwacje na loty

3. Widoki dla administratora (admin)
    - Widok główny (MainView)
    - Widok listy przewoźników (AirlinesView)
        - Administrator może dodawać, edytować i usuwać przewoźników
        - Usunięcie przewoźnika skutkuje usunięciem wszystkich lotów tego przewoźnika, i w konsekwencji wszystkich rezerwacji na loty tego przewoźnika
    - Widok formularza do dodawania i edycji przewoźników (addAirlineView)
    - Widok listy lotów (FlightView)
        - Administrator może dodawać, edytować i usuwać loty
        - Usunięcie lotu przez administratora oznacza usunięcie wszystkich rezerwacji na ten lot
    - Widok formularza do dodawania i edycji lotów (addFlightView)
    - Widok listy rezerwacji (ReservationListView)
        - Administrator może usuwać i edytować rezerwacje
    - Widok listy klientów (CustomersView)
        - Administrator może dodawać, edytować i usuwać klientów
    - Widok formularza do dodawania klientów (addCustomers)

Za zarządzanie widokami odpowiedzialne są klasy znajdujące się w katalogu *controller*

2.6. Utils

W katalogu *utils* znajdują się klasy pomocnicze służące do obsługiwania pewnych powtarzalnych funkcjonalności w różnych fragmentach kodu (zgodnie z regułą dry). Na chwilę obecną znajduje się tam:

- Enumerator UserRole
- Klasa Validator - która zawiera różne metody walidacji pól formularza, wykorzystywana w formularzach dodawania użytkownika i linii lotniczej
- GenericFilter - klasa zawiera metody pozwalające na realizację filtrowania danych w widokach

3. Role w aplikacji - logowanie

W systemie zdefiniowane są 3 role. Zakres ich możliwości został opisany w punkcie 2.5. dokumentacji.

3.1. Użytkownik niezalogowany (anonymous)

3.2. Użytkownik zalogowany (user)

Logowanie:

- Login: user
- Hasło: user

3.3. Administrator (admin)

Logowanie:

- Login: admin
- Hasło: admin

4. Podział pracy

4.1. Etap m1

4.1.1. Praca wspólna

- Skonstruowanie modelu - diagram

4.1.2. Bartosz Kaszuba

- Dodanie klas TicketDiscount, User, Reservation do modelu aplikacji
- Stworzenie relacji pomiędzy elementami modelu (klucze obce itd.)
- Stworzenie widoku formularza do dodawania przewoźników
- Zaprogramowanie dodawania przewoźników

4.1.3. Tomasz Kozyra

- Inicjalizacja projektu
- Dodanie klas Airline i Flight do modelu aplikacji
- Stworzenie widoku przewoźników
- Zaprogramowanie widoku przewoźników tak aby wyświetlał listę przewoźników

4.1.4. Kacper Rosiak

- Dodanie klas TicketCategory, TicketOrder do modelu aplikacji
- Stworzenie widoku klientów
- Zaprogramowanie widoku klientów tak aby wyświetlał listę klientów
- Stworzenie opisu struktury projektu (punkt 2 dokumentacji)

4.1.5. Amadeusz Szabała

- Dodanie klas Customer, Review do modelu aplikacji
- Stworzenie widoku formularza do dodawania klientów
- Zaprogramowanie dodawania klientów
- Stworzenie opisu tabel w dokumentacji (punkt 1.2 dokumentacji)

4.2. Etap m2

4.2.1. Bartosz Kaszuba

- Filtrowanie przewoźników, lotów, rezerwacji
- Stworzenie widoku formularza do dodawania rezerwacji
- CRUD dla rezerwacji + persystencja
- Rozwiązywanie konfliktów przy tworzeniu rezerwacji

4.2.2. Tomasz Kozyra

- Aktualizacja wyglądu aplikacji
- CRUD dla przewoźników i klientów + persystencja
- Dodanie zapełniania bazy danych przykładowymi danymi przy starcie aplikacji
- Naprawa pomniejszych błędów
- Dokumentacja

4.2.3. Kacper Rosiak

- Autentykacja i autoryzacja
- Stworzenie dodatkowych widoków dla zalogowanego użytkownika
- Podział widoków na klasy dla użytkownika niezalogowanego/zalogowanego/administratora
- Naprawa pomniejszych błędów

4.2.4. Amadeusz Szabała

- Stworzenie widoku dla lotów oraz dodawania lotów
- CRUD dla lotów + persystencja
- Rozwiązywanie konfliktów przy tworzeniu lotów
- Dokumentacja

5. Wykorzystane wzorce projektowe

- Inversion of Control - realizowane przez wstrzykiwanie zależności - wykorzystaliśmy framework Spring. Wykorzystywane m.in. do wstrzykiwania właściwych implementacji kontrolerów.
- MVC - podział strony graficznej aplikacji na kontrolery, widoki i model danych (pozwala na niezależne rozwijanie poszczególnych elementów aplikacji, enkapsulacja logiki aplikacji w obszarze kontrolerów)
- Repository - dostęp do danych - dodatkowa warstwa abstrakcji dodana nad klasami modelu

6. Uruchamianie projektu

6.1. Sposób 1 - Gradle

1. Przechodzimy do katalogu projektu
2. Uruchamiamy komendę “gradle run”

6.2. Sposób 2 - jar

Projekt uruchamiamy przez uruchomienie pliku **yourflights-2.0.0-m2.jar** znajdującego się w katalogu głównym projektu.

1. Przechodzimy do katalogu głównego projektu
2. Wywołujemy komendę “java -jar yourflights-2.0.0-m2.jar”