2021

Dokumentacja projektu

CAR RENTAL – FRAMEWORKI BIZNESOWE

CHOJECKI MATEUSZ, MARCZUK MICHAŁ

Opis projektu

Głównym założeniem projektu było udostępnienie REST API do wynajmowania samochodów. Cena wynajmu samochodu obejmuje jego wynajem na okres 24 godzin. Użytkownik przed przystąpieniem do realizacji wynajmu, zobowiązany jest do rejestracji i zalogowania. Klient posiada uprawnienia do przeglądania listy aut, wykonania i edycji swojego zamówienia, a także do zaktualizowania swoich danych profilowych.

Diagram przypadków użycia

Funkcjonalności

Użytkownik:

- Logowanie
- Rejestracja
- Przeglądanie listy samochodów
- Aktualizacja danych profilowych
- Wypożyczanie samochodu
- Przegląd własnych wypożyczeń

Administrator:

- Dodawanie/Usuwanie/Przeglądanie/Edytowanie Samochodów
- Dodawanie/Usuwanie/Przeglądanie/Edytowanie Wypożyczeń
- Dodawanie/Usuwanie/Przeglądanie/Edytowanie statusów samochodów/ wypożyczeń
- Anulowanie wypożyczenia
- Zarządzanie użytkownikami
- Zarządzanie statusem wypożyczenia

Wykorzystane technologie

-Spring Framework

-Spring Boot (z wykorzystaniem Maven'a)

-Spring Security

-Spring Data(JPA)

-Lombok

-Swagger

-JSON Web Token

-Baza danych MySQL

Wersjonowanie

Istnieje wiele rozwiązań na wersjonowanie api, np. za pomocą ścieżki URI, parametru URI lub przez nagłówek Accept.

W naszym przypadku chcielibyśmy zaproponować wersjonowanie za pomocą ścieżki URI, czyli mając do dyspozycji link

http://ourwebsite.com/api/cars

dodalibyśmy do ścieżki {**wersja_api**}

http://ourwebsite.com/api/{**wersja_api**}/cars

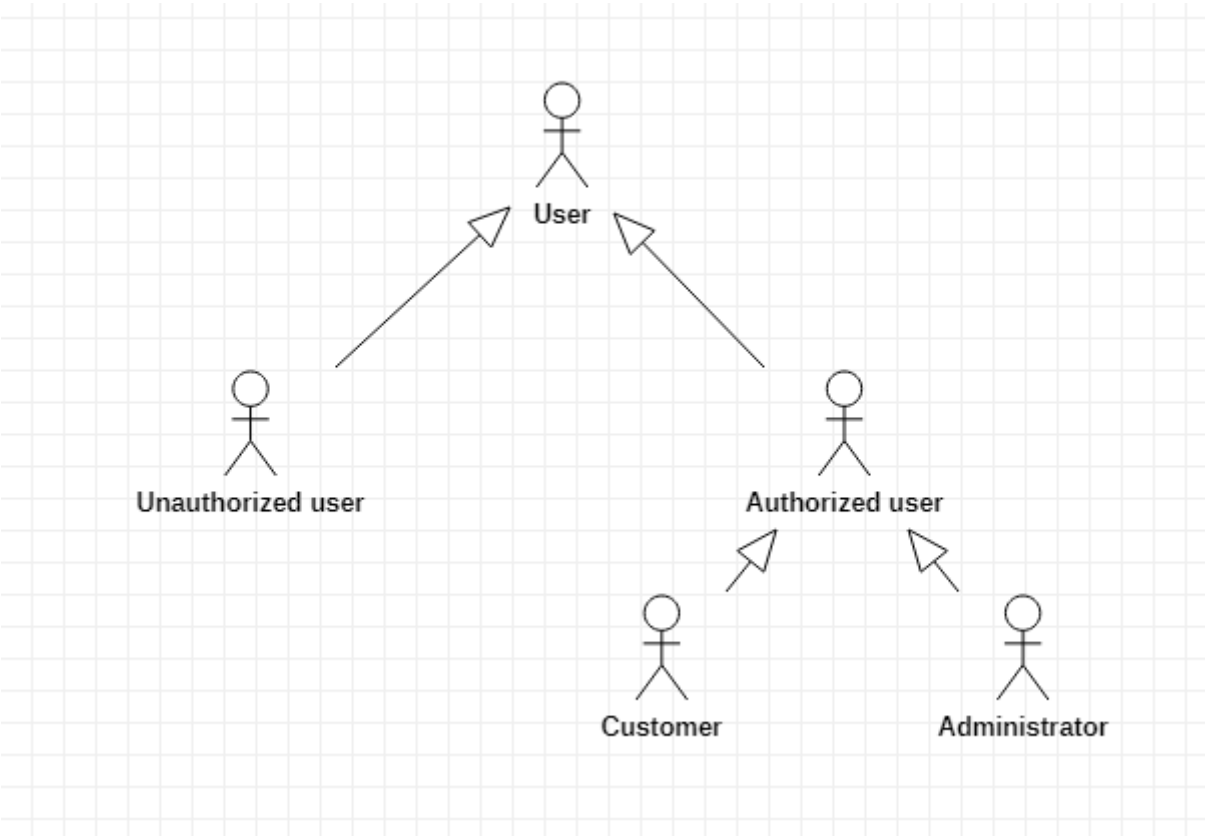
Oczywiście należałoby wykorzystać także wersjonowanie semantyczny **MAJOR MINOR PATCH** , gdzie:

MAJOR{v1,v2,v3,itd.) – oznaczałoby wprowadzenie zmian niekompatybilnych z wersjami API

MINOR(1,2,3, itd.) – w przypadku wprowadzania funkcjonalności, która jest kompatybilna wstecz(z poprzednimi wersjami API).Niewidoczne dla użytkownika.

PATCH(01,02,03 itd.) – w sytuacji naprawy małych błędów i bugów niewprowadzających ingerencji w kompatybilność wsteczną. Niewidoczne dla użytkownika.

Role użytkowników

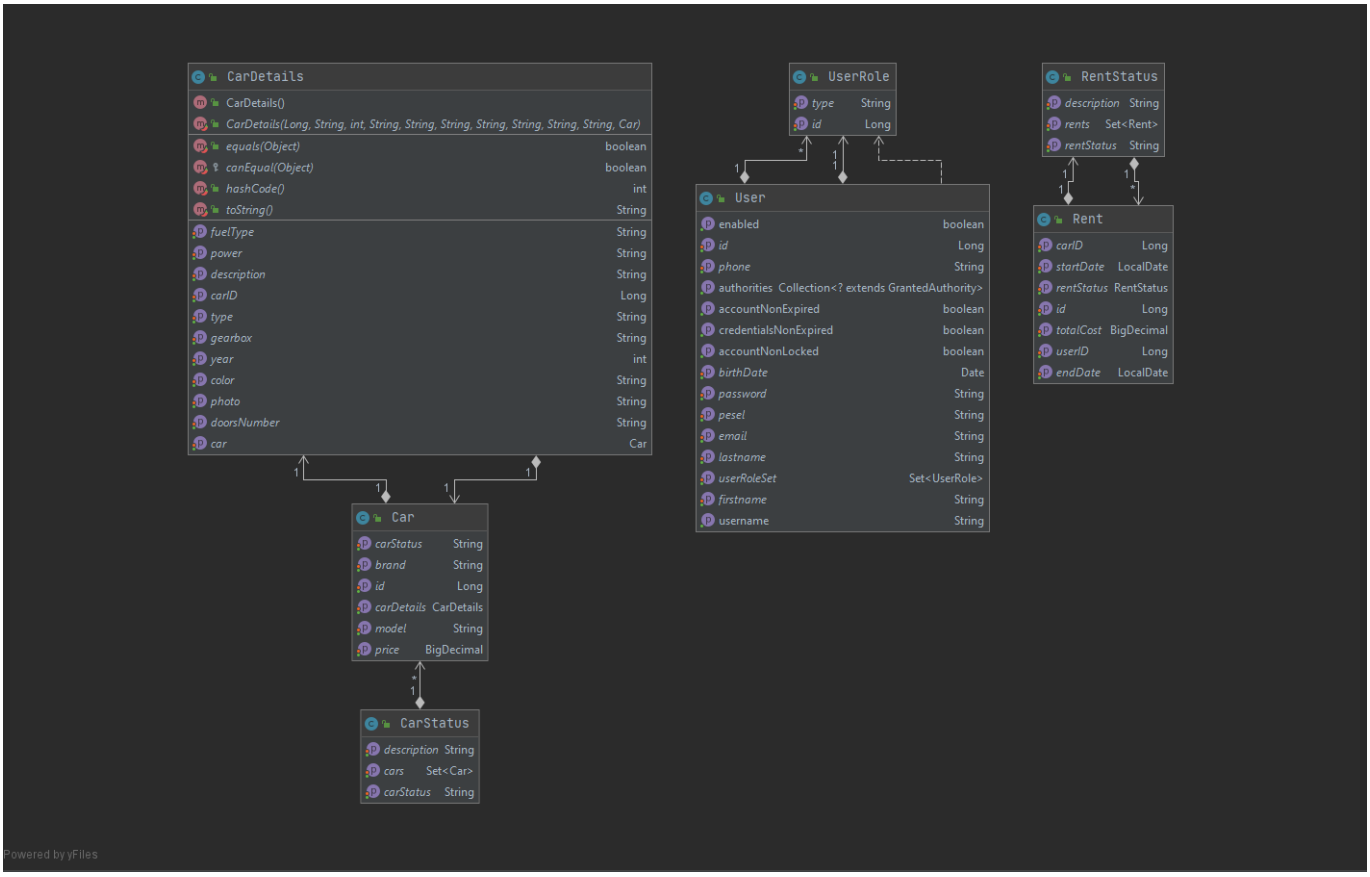


Unauthorized user – użytkownik niezalogowany posiadający uprawnienia do przeglądania samochodów

Customer – użytkownik zalogowany posiadający uprawnienia do przeglądania i wypożyczania samochodów i aktualizacji swojego konta

Administrator – główny użytkownik zarządzający serwisem i posiadający najwięcej uprawnień.

Diagram klas(encji)



Modele:

Car – model przedstawiający pojedynczy samochód.

CarDetails – każdy samochód posiada unikalne szczegóły, którymi się wyróżnia i są one przechowywane w encji.

CarStatus - Samochód może być np. gotowy, zepsuty bądź zarezerwowany, dlatego jego aktualny stan przechowywany jest w encji CarStatus.

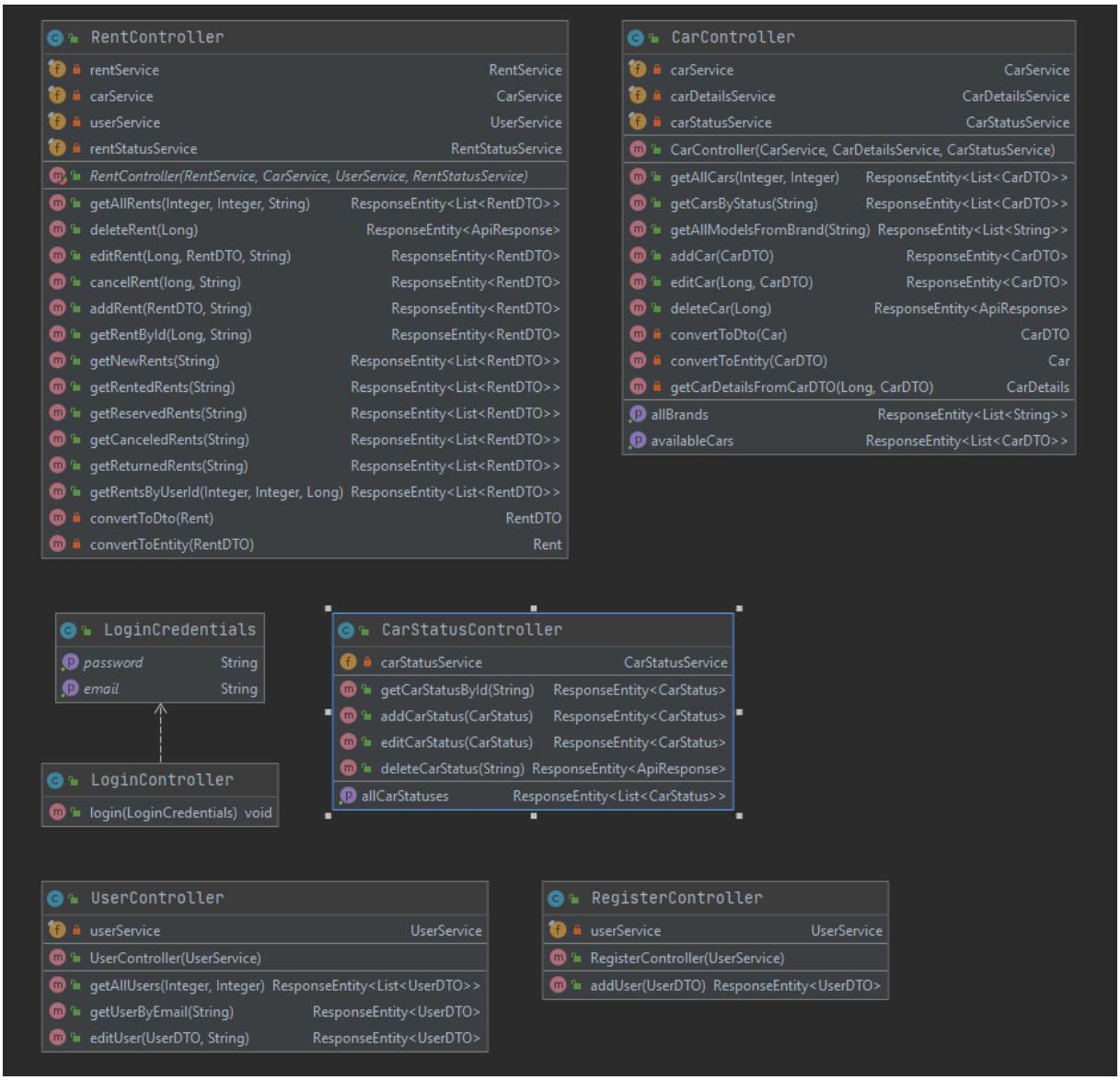
User - klasa przechowująca informacje o użytkownikach

UserRole – informacje o uprawnieniach danego użytkownika

Rent – dane przechowujące informacje o wypożyczeniu

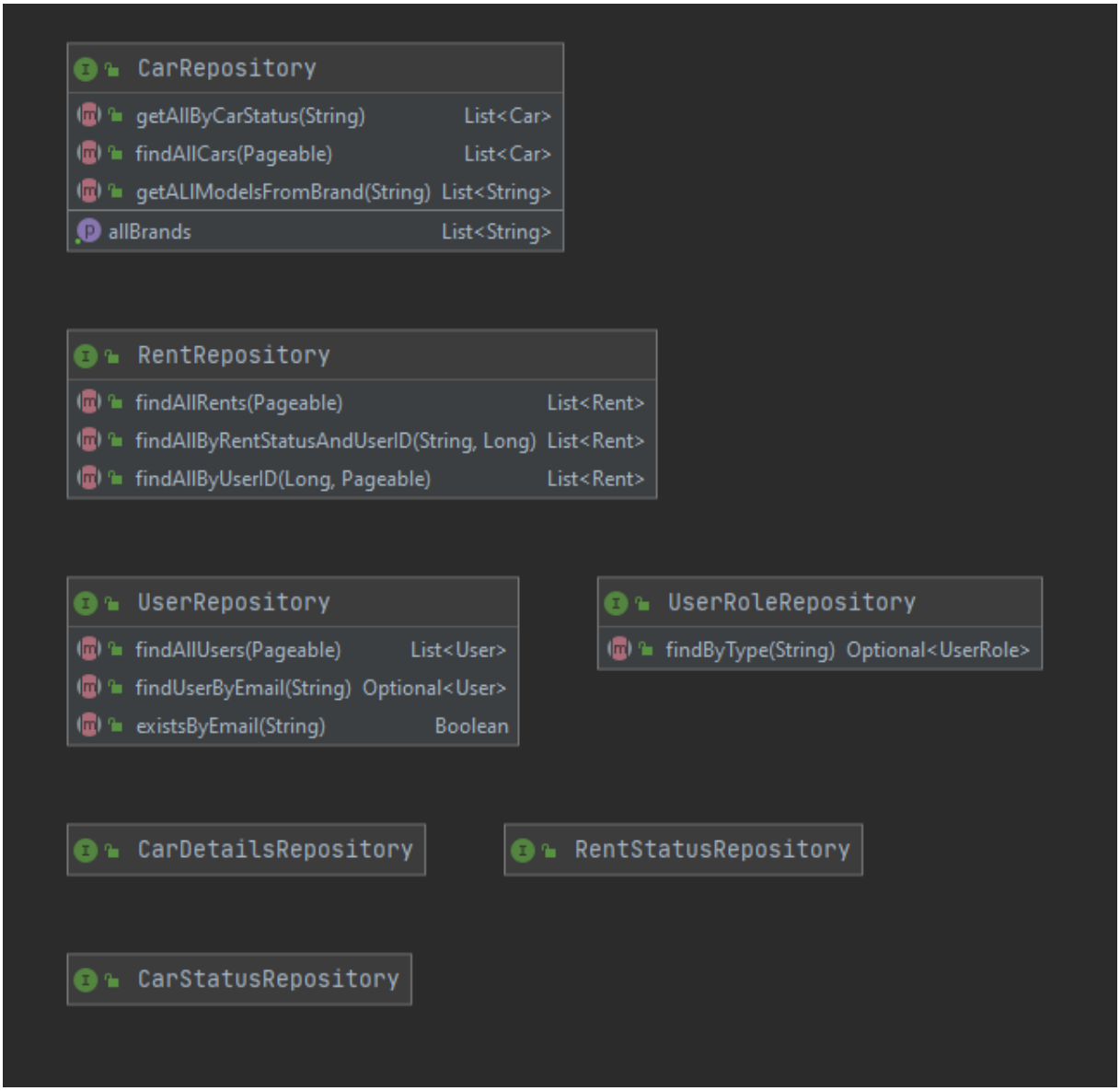
RentStatus – encja mówiąca o statusie danego zamówienia

Diagram klas(kontrolerów)



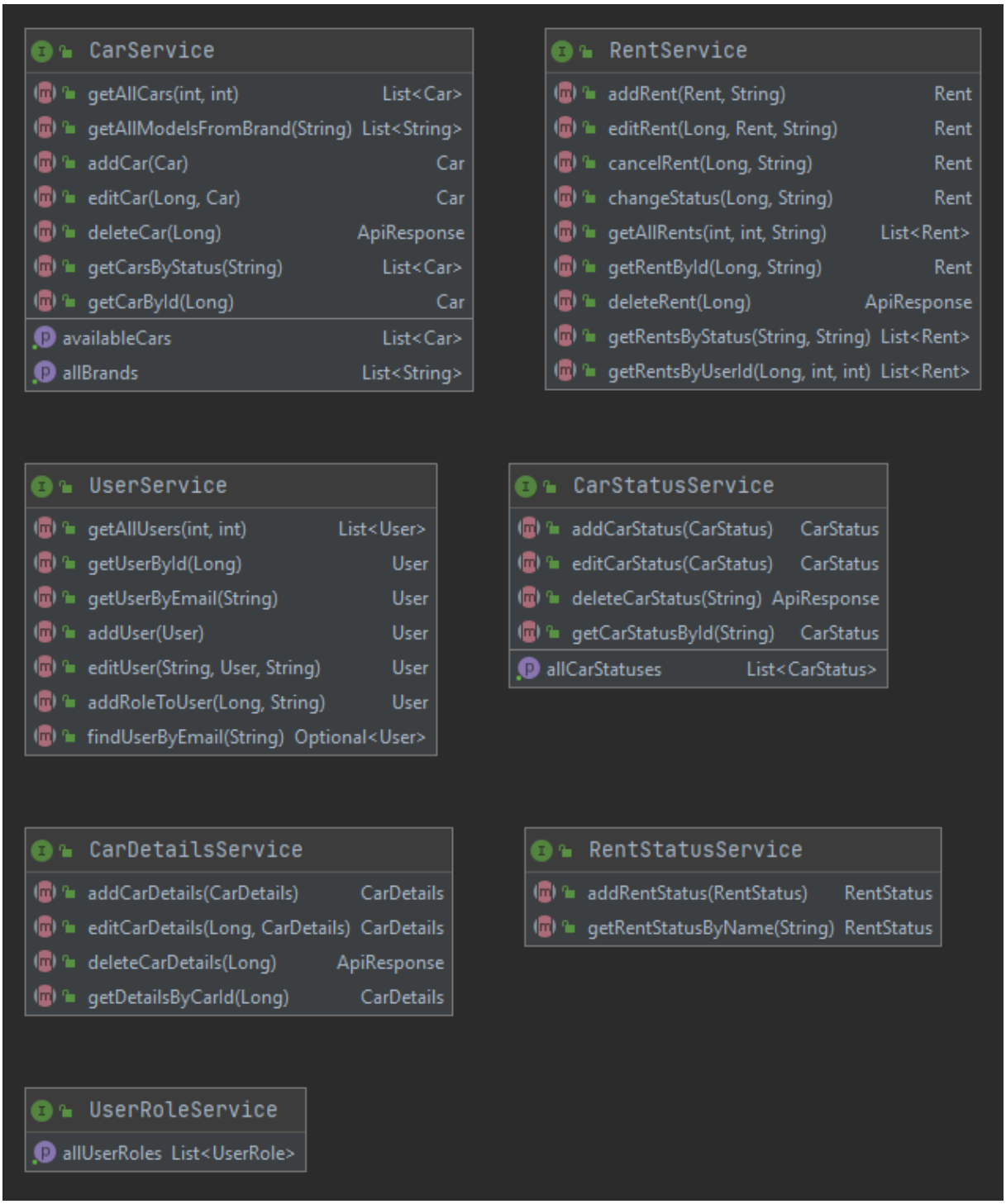
Kontrolery służą do obsługi end-pointów i „rozmowy” z serwerem przy użyciu zapytań HTTP.

Diagram klas(repozytoriów)



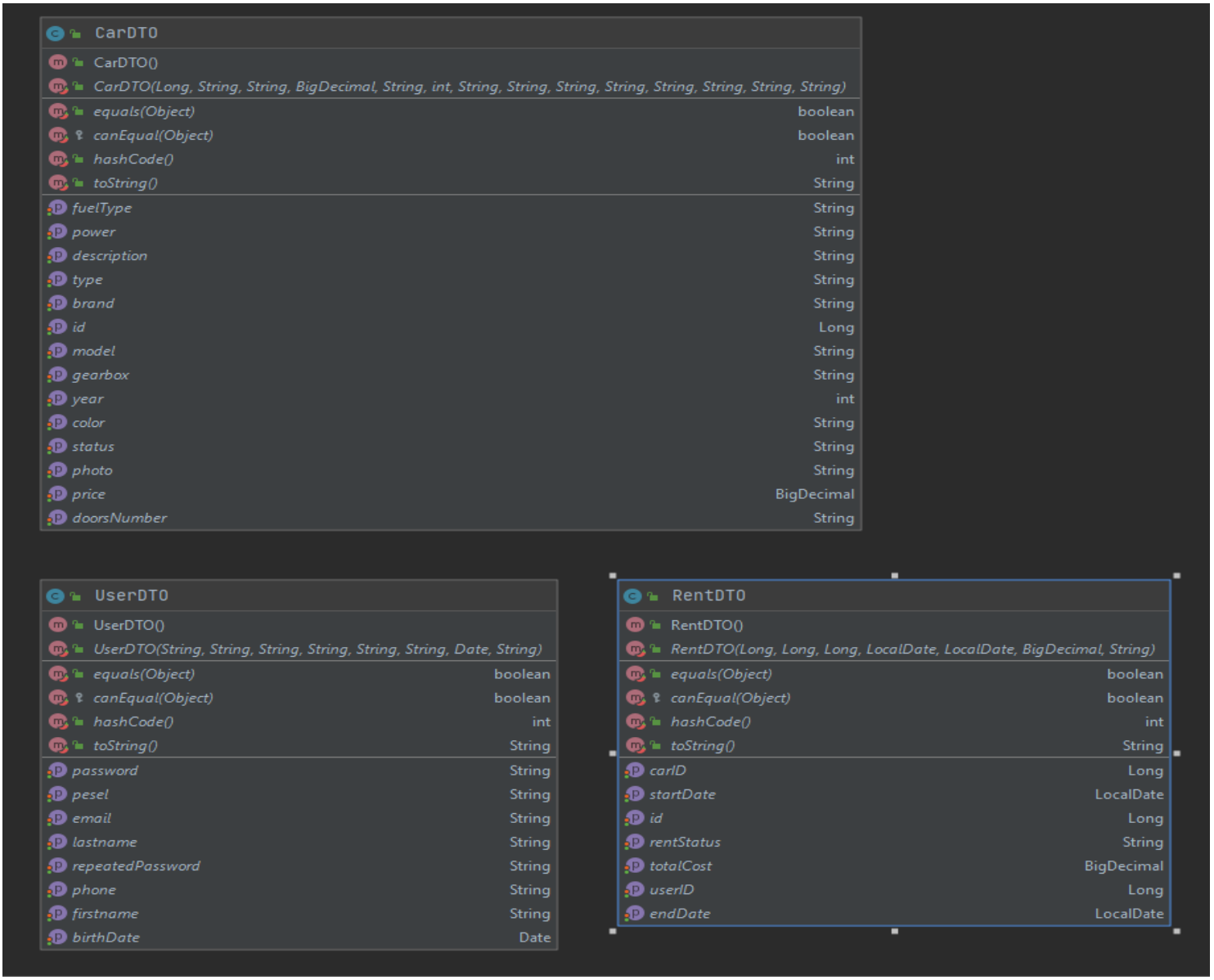
Repozytoria pozwalają na zarządzanie dostępem do danych i wykonywanie operacji na konkretnym typie danych.

Diagram klas(serwisów)



Serwisy wykorzystywane są jako „warstwa pośrednia”, czyli element pośredni pomiędzy bazą danych(naszymi repozytoriami) a widokami(naszymi kontrolerami).

Diagram klas(Data Transfer Object's)



Obiekty DTO służą do transferu danych pomiędzy klientem a serwerem. Dzięki takiemu zastosowaniu, nie operujemy na encjach z bazy danych, a na obiektach pośredniczących, dzięki temu możemy manipulować, jakie dane chcemy udostępnić, a jakie nie.

JSON Web Token

Do obsługi procesu uwierzytelnienia i autoryzacji wykorzystaliśmy JSON Web Token, który jest otwartym standardem umożliwiającym transfer danych lub wymianę informacji między dwiema frakcjami. Dane przekazywane są zakodowane jako obiekty JSON Web Signature i podpisane niejawnie cyfrowo za pomocą algorytmu HMAC lub jawnie za pomocą klucza prywatnego/publicznego RSA. Otrzymany token należy umieścić w nagłówki zapytania „Authorization”

Przykład wygenerowanego tokenu w aplikacji car rental:

Endpointy

Endpointy zostały umieszczone w plik endpoints-carrental.json wraz z przykładami użycia. Endpointy zostały zaimportowane ze swaggera.

Baza danych SQL

Skrypt bazy danych został umieszczony w pliku schema.sql