POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Informatyka (INF)
SPECJALNOŚĆ: Inżynieria systemów informatycznych (INS)

PRACA DYPLOMOWA INŻYNIERSKA

Aplikacja webowa wspomagająca zarządzanie flotą samochodów

A web application supporting cars fleet management

AUTOR:
Jan Pajdak

PROWADZĄCY PRACĘ:
dr inż. Jarosław Mierzwa, K-9

OPIEKUN:
dr hab. inż. Olgierd Unold Prof. nadzw. PWr, K-9

OCENA PRACY:

WROCŁAW 2018

Spis treści

Rozdział 1

Wprowadzenie

1.1 Wstęp

Celem niniejszej pracy dyplomowej jest opracowanie projektu, implementacja oraz wdrożenie systemu umożliwiającego zarządzanie flotą samochodów. Pierwszym etapem projektu jest zebrane wymagań funkcjonalnych i niefunkcjonalnych oraz określenie zakresu pracy. Drugi etap projektu to wybór technologii i projekt architektury. Ostatnim celem jest implementacja systemu.

Temat projektu został wybrany ze względu na chęć wykorzystania wiedzy z dziedziny motoryzacji w celu stworzenia aplikacji ułatwiającej zarządzanie pojazdami. Z uwagi na rosnącą popularność rozwiązań związanych z wypożyczaniem samochodów celem projektu jest system, który można opisać jako wewnątrzfirmową wypożyczalnie umożliwiająca jak największe wykorzystanie dostępnej floty pojazdów przez pracowników, którzy nie mają potrzeby posiadania firmowego samochodu na wyłączność.

1.2 Cel i zakres pracy

Celem projektu jest stworzenie aplikacji umożliwiającej wypożyczanie oraz zarządzanie flotą samochodów. Aplikacja jest skierowana do firm które nie mają potrzeby lub wystarczających środków by zapewnić pracownikom samochody na wyłączność. Przykładowym przypadkiem użycia systemu może być jednorazowa potrzeba odwiedzenia klienta lub wyjazd na szkolenie. Typowe rozwiązania dla firm obecne na rynku skierowane są do firm świadczących usługi spedycyjne — aplikacje posiadają warstwę śledzenia ładunków oraz tworzenia zadań przewozowych dla kierowców; programy służące do obsługi komercyjnych wypożyczalni pomijają proces autoryzacji wypożyczenia — zwykle sprawdzana jest zdolność wypożyczającego do zapłaty.

Projekt utworzony w ramach tej pracy łączy mechanikę z komercyjnych wypożyczalni z dodatkową warstwą biznesową pozwalającą kontrolować sposób używania pojazdów.

Zakres pracy obejmuje utworzenie systemu spełniającego wymagania postawione w rozdziale 3 oraz przygotowanie projektu do wdrożenia, poprzez np. konteneryzacje.

1.3 Układ pracy

W rozdziale pierwszym zawarto wstęp oraz krótki opis celu projektu. Drugi rozdział porównuje istniejące rozwiązania do aplikacji będącej celem projektu. Rozdział trzeci zawiera

wymagania funkcjonalne oraz niefunkcjonalne. W kolejnym, czwartym rozdziale znajduje się opis wybranych technologii oraz narzędzi, wraz z uzasadnieniem.

Rozdział 2

Istniejące rozwiązania

Rozdział 3

Wymagania funkcjonalne i niefunkcjonalne

3.1 Wymagania funkcjonalne

Wymagania zostały opisane według poniższego wzorca:

Numer	Numer wymagania
Nazwa	Krótka nazwa
Opis	Dokładny opis
Aktor	Grupa użytkowników
Kryterium spełnienia	Funkcjonalność która musi zostać zaimplementowana by wymaganie można było uznać jako spełnione
Ograniczenia	Ograniczenia funkcjonalności, jeżeli takie istnieją

Rozróżniane są dwa rodzaje aktorów:

- Kierowca - użytkownik korzystający z funkcjonalności tworzenia i przeglądania historii wypożyczeń
- Kierownik - użytkownik z pełnym dostępem do systemu

Kierownik posiada wszelkie prawa i możliwości Kierowcy.

Dodatkowe pojęcia związane z modelami świata biznesowego:

- **Model Pojazdu** - model opisujący specyfikacje techniczną wspólną dla pewnego zbioru pojazdów
- **Pojazd** - model opisujący informacje unikatowe dla pewnego przedstawiciela zbioru Modeli Pojazdów.

Numer	1
Nazwa	Zarządzanie modelami pojazdów
Opis	System powinien pozwalać na dodawanie i edycję modeli pojazdów; specyfikacji technicznej dla danego modelu.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może dodawać nowe modele samochodów. Informacje mogą zostać w późniejszym czasie zmodyfikowane lub usunięte.
Ograniczenia	Model pojazdu może zostać usunięty wyłącznie gdy nie ma żadnych pojazdów

Numer	2
Nazwa	Zarządzanie pojazdami
Opis	System powinien pozwalać na dodawanie i edycję pojazdów będących egzemplarzami modeli z wymagania #2; pojazd zawiera informacje unikalne dla danego egzemplarza, takie jak numer rejestracyjny.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może dodawać nowe pojazdy dla wybranego modelu. Informacje mogą zostać w późniejszym czasie zmodyfikowane lub usunięte.
Ograniczenia	Pojazd nie może być modyfikowany gdy jest obecnie wypożyczony. Pojazd który był wypożyczany nie może zostać usunięty — może zostać oznaczony jako wycofany z użycia.

Numer	3
Nazwa	Zarządzanie ubezpieczeniami pojazdu
Opis	System powinien umożliwiać wprowadzanie informacji związanych z ubezpieczeniami danego pojazdu.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może przeglądać historię ubezpieczeń danego pojazdu oraz wprowadzać nowe dane. System bierze pod uwagę obecny stan pojazdu podczas tworzenia wypożyczenia; pojazd nie może zostać wypożyczony w okresie gdy nie ma aktywnego ubezpieczenia.
Ograniczenia	

Numer	4
Nazwa	Zarządzanie serwisami pojazdu
Opis	System powinien umożliwiać wprowadzanie informacji związanych z serwisami danego pojazdu.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może przeglądać historię napraw danego pojazdu oraz wprowadzać nowe dane. System rozróżnia różne rodzaje serwisowania takie jak regularny przegląd, zdarzenie wyjątkowe czy naprawa powypadkowa. System bierze pod uwagę obecny stan pojazdu podczas tworzenia wypożyczenia; pojazd nie może zostać wypożyczony gdy jest obecnie naprawiany.
Ograniczenia	

Numer	5
Nazwa	Tworzenie wypożyczeń
Opis	System powinien umożliwiać przeglądanie dostępnych pojazdów (dostępność określana jest na podstawie informacji z wymagania #2) i tworzenie wypożyczeń wraz z niezbędnymi danymi takimi jak okres czasu i potrzeba stojąca za wypożyczeniem
Aktor	Kierowca
Kryterium spełnienia	Kierowca może utworzyć wypożyczenie
Ograniczenia	Kierowca nie może utworzyć wypożyczenia dla innego użytkownika

Numer	6
Nazwa	Kontrola wypożyczeń
Opis	System umożliwia kontrolowanie stanu wypożyczenia. Wypożyczenie uznane jest za obowiązujące dopiero po akceptacji przez uprawnioną do tego osobę.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może przeglądać wypożyczenia utworzone przez użytkowników systemu oraz zmieniać ich obecny stan po ocenie zasadności wypożyczenia
Ograniczenia	Kierownik nie może akceptować własnych wypożyczeń

Numer	7
Nazwa	Zbieranie informacji o kosztach wypożyczeń
Opis	System umożliwia śledzenie kosztów utrzymania floty na podstawie raportów wprowadzanych przez wypożyczających.
Aktor	Kierowca
Kryterium spełnienia	Kierowca może wprowadzić informację związane z wypożyczeniem (zużyte litry paliwa, przejechane kilometry, całkowity koszt) po oddaniu samochodu.
Ograniczenia	

Numer	8
Nazwa	Zbieranie informacji o kosztach utrzymania
Opis	System umożliwia śledzenie kosztów utrzymania floty związanych z ubezpieczeniami oraz naprawami.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może wprowadzić koszty związane z ubezpieczeniem/serwisem pojazdu.
Ograniczenia	

Numer	9
Nazwa	Wyświetlanie informacji o kosztach utrzymania
Opis	System jest w stanie wygenerować plik kompatybilny z programem *Excel* zawierający dane na temat kosztów floty.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może wywołać utworzenie pliku ze statystykami
Ograniczenia	

Numer	10
Nazwa	Przechowywanie informacji audytowych
Opis	System zapisuje informacje o dacie i użytkowniku dokonującym wprowadzenia nowych danych lub modyfikacji istniejących.
Aktor	Kierownik
Kryterium spełnienia	Informacje o dacie i użytkowniku modyfikowane są w trakcie zapisu do bazy danych. Kierownik może przeglądać dane audytowe.
Ograniczenia	

3.2 Wymagania niefunkcjonalne

3.2.1 Interfejs użytkownika

- Wygląd powinien być prosty i nowoczesny
- Elementy strony powinny być rozmieszczone w intuicyjny sposób
- Struktura widoków powinna być ułożona zgodnie z zależnościami między wyświetlanymi danymi
- Aplikacja być wygodna w użyciu na ekranach komputerów o rozdzielczości HD (1366x768 pikseli) lub większej

3.2.2 Interfejs programistyczny

- System powinien wymagać niewielkich modyfikacji w przypadku integracji z istniejącymi zasobami firmy (np. baza danych pracowników)
- Komunikacja powinna opierać się na otwartych i uniwersalnych standardach, np. dane w postaci *JSON* lub *XML* przesyłane protokołem *HTTP*

- Interfejs programistyczny powinien być niezależny od platformy tak by w przyszłości mógł zostać wykorzystany przez inne aplikacje

3.2.3 Bezpieczeństwo

System powinien być zabezpieczony zarówno po stronie interfejsu użytkownika (np. blokada przed przejściem do podstrony) oraz po stronie interfejsu programistycznego (ignorowanie zapytań od nieupoważnionych aplikacji). Zabezpieczenie powinno obsługiwać różne poziomy autoryzacji w zależności od roli użytkownika.

Rozdział 4

Zastosowane technologie i narzędzia

4.1 Zastosowane technologie

Interfejs użytkownika wykorzystuje platformę *Angular 7*. Podstawowymi elementami w *Angular* są komponenty [1], każdy z nich złożony z: pliku klasy *TypeScript* zawierającej logikę, wzorca *htm* opisującego wygląd widoku oraz opcjonalnego stylu *css*; w przypadku jego braku styl brany jest z komponentu-rodzica. Warto zwrócić uwagę na język programowania wykorzystywany przez platformę *Angular* — *TypeScript* [5], będący rozszerzeniem języka *JavaScript*. *TypeScript* dodaje silniejsze typowanie i kładzie większy nacisk na programowanie obiektowe, jednocześnie pozostając w pełni kompatybilnym z *JavaScript*, do którego jest kompilowany i następnie uruchamiany jest w przeglądarce. Proces kompilacji pozwala na usunięcie wielu błędów, które w przypadku *JavaScript* zostałyby zauważone dopiero po uruchomieniu aplikacji.

Jednym z ważniejszych komponentów aplikacji jest *Bootstrap* - framework interfejsu użytkownika pozwalający w prosty sposób tworzyć estetyczne strony internetowe. Dodatkowo, w projekcie wykorzystano motywy *Bootswatch*.

Interfejs programistyczny oparty został na technologii *ASP.NET Core 2.1* — jest to nowoczesna platforma oferująca działanie na wielu systemach operacyjnych oraz większa wydajność względem starszych rozwiązań firmy Microsoft. Wykorzystany język programowania to obiektowy, kompilowany i statycznie typowany *C# 7.3*. Bardzo ważnym elementem tej części projektu jest *EF (Entity Framework) Core 2.1* [6], framework ORM (Object-Relational Mapping) pozwalający na konwersję miedzy tabelami bazy danych a klasami *C#*. Jedną z najważniejszych funkcjonalności *EF Core* jest wykorzystana w niniejszym projekcie możliwość utworzenia bazy danych przy użyciu konwencji *Code First*; baza danych jest automatycznie generowana na podstawie klas C# znajdujących się w projekcie. *EF Core* współpracuje z większością popularnych baz danych; na potrzeby tego projektu wykorzystano *MS SQL Server*.

4.2 Wykorzystane narzędzia

W trakcie realizacji projektu wykorzystane zostały narzędzia najczęściej używane przy wybranych technologiach.

Do zarządzania kodem został wykorzystany system kontroli wersji *Git*. Lokalna kopia projektu była synchronizowana ze zdalnym, prywatnym repozytorium znajdującym się na serwisie GitHub. Wykorzystane rozwiązanie pozwala na łatwy dostęp do wcześniejszych wersji projektu oraz zmniejsza ryzyko utraty kodu, gdyż nie jest on przechowywany tylko

w jednym miejscu.

Ze względu na wykorzystane technologie, kod był rozwijany z pomocą narzędzi Microsoft, oferujących najlepsze wsparcie dla *TypeScript* oraz *C#*. Aplikacja klienta była rozwijana przy użyciu *Visual Studio Code 1.28*, nowoczesnego edytora który sprawdza się znakomicie przy tworzeniu interfejsów użytkownika ze względu na zintegrowaną konsolę pozwalającą na łatwe zarządzanie paczkami oraz łatwość dostosowywania do potrzeb użytkownika. W trakcie pracy wykorzystano wiele rozszerzeń, najważniejsze z nich to *TSLint*, linter wykrywający błędy w kodzie *TypeScript* oraz *GitLens* — rozszerzenie wspomagające zarządzanie repozytorium *Git*. Do rozwoju serwisów wykorzystano *Visual Studio 2017* pozwalające na łatwe debugowanie kodu oraz analizę aspektów takich jak wykorzystanie zasobów przez program. *Visual Studio* zostało wzbogacone o narzędzie *JetBrains ReSharper* automatycznie formatujące pliki projektu według zadanego wzorca, zapewniając spójność i przejrzystość kodu.

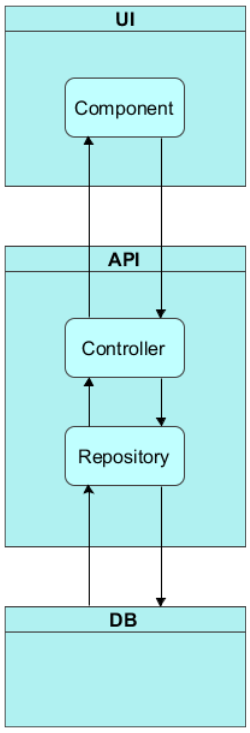
Interfejs programistyczny testowany był przy pomocy *Postman 6.5.2*, aplikacji pozwalającej na wysyłanie oraz zarządzanie zapytaniami HTTP.

Rozdział 5

Projekt i implementacja

5.1 Architektura

System został stworzony przy użyciu klasycznej architektury w której można wyodrębić trzy moduły - interfejs użytkownika (*UI*), interfejs programistyczny (*API*) oraz bazę danych (*DB*).



Rysunek 5.1 Uproszczony schemat architektury z wyodrębnionymi najważniejszymi elementami składowymi

System został zaprojektowany tak, by mógł zostać zintegrowany z istniejącymi zasobami firmy — jedyne dane, jakie przechowuje, dotyczą logiki biznesowej, związanej z wymaganiami funkcjonalnymi; wynika to z faktu, że większość firm ma już własne bazy danych przechowujące informacje o pracownikach więc duplikacja danych jest niepożądana ze względu na zużycie zasobów oraz możliwe problemy z synchronizacją. Dane związane z użytkownikami (np. imię, nazwisko, e-mail i numer telefonu) czy lokacjami firmy (np. adres) mogą zostać pobrane z innej bazy danych; ponadto interfejs użytkownika nie

umożliwia wprowadzania lub edycji takich danych. Implementacja opisana w dalszej części niniejszej pracy przechowuje przykładowe dane użytkowników do celów testowych w tej samej bazie danych, jednakże konfiguracja systemu tak by korzystał z innej, nie stanowi większego problemu.

W architekturze można rozróżnić trzy najważniejsze składowe, dwie pierwsze w interfejsie programistycznym i trzecią w interfejsie użytkownika:

- Kontroler (*Controller*) to klasa odpowiadająca za obsługę żądań *HTTP* [9].
- Repozytorium (*Repository*) zawiera logikę pośredniczącą w komunikacji między *API* a bazą danych.
- Komponent (*Component*) to podstawowy element definiujący działanie widoku w *Angular* [1].

5.2 Standardy

Projekt był tworzony zgodnie z dobrymi praktykami programowania, z naciskiem na poprawną implementację obiektowego paradygmatu programowania. Interfejs programistyczny był tworzony z użyciem sztandarowych możliwości języka C# takimi jak typy ogólne [4] (*Generics*) pozwalające na tworzenie pojedynczych metod i klas zdolnych do operacji na wielu typach, zachowując wszystkie zalety silnego, statycznego typowania i wysoką wydajność.

W celu zapewnienia przejrzystości kodu, nazewnictwo wszystkich elementów oraz dokumentacja kodu są zgodne ze standardową konwencją danego języka. Kod jest napisany w całości w języku angielskim.

Język	Typy	Pliki	Zmienne prywatne	Inne zmienne
C# [8]	PascalCase	PascalCase.cs	camelCase	PascalCase
TypeScript [2]	PascalCase	snake-case.typ.ts	camelCase	camelCase

Tablica 5.1 Najważniejsze konwencje nazewnicze

5.3 Bezpieczeństwo

Dostęp do systemu został zabezpieczony przy użyciu standardu *JSON Web Token (JWT)* [3]. Autoryzacja *JWT* bazuje na generowaniu podpisanych (przez co odpornych na sfałszowanie) tokenów po stronie interfejsu programistycznego, a następnie wysyłaniu ich do aplikacji klienta. *API* wcześniej wygenerowanego wymaga tokena w nagłówku *HTTP* dla każdego żądania wysłanego przez interfejs użytkownika; żądania z niepoprawnym tokenem zostają odrzucone.

Schemat działania autoryzacji *JWT* w opisywanym projekcie wygląda następująco:

1. Użytkownik loguje się przez interfejs użytkownika, podając nazwę użytkownika oraz hasło
2. Interfejs programistyczny weryfikuje dane logowania
3. W przypadku prawidłowego hasła utworzony zostaje token *JWT* zawierający:

   Informacje o wydającym token

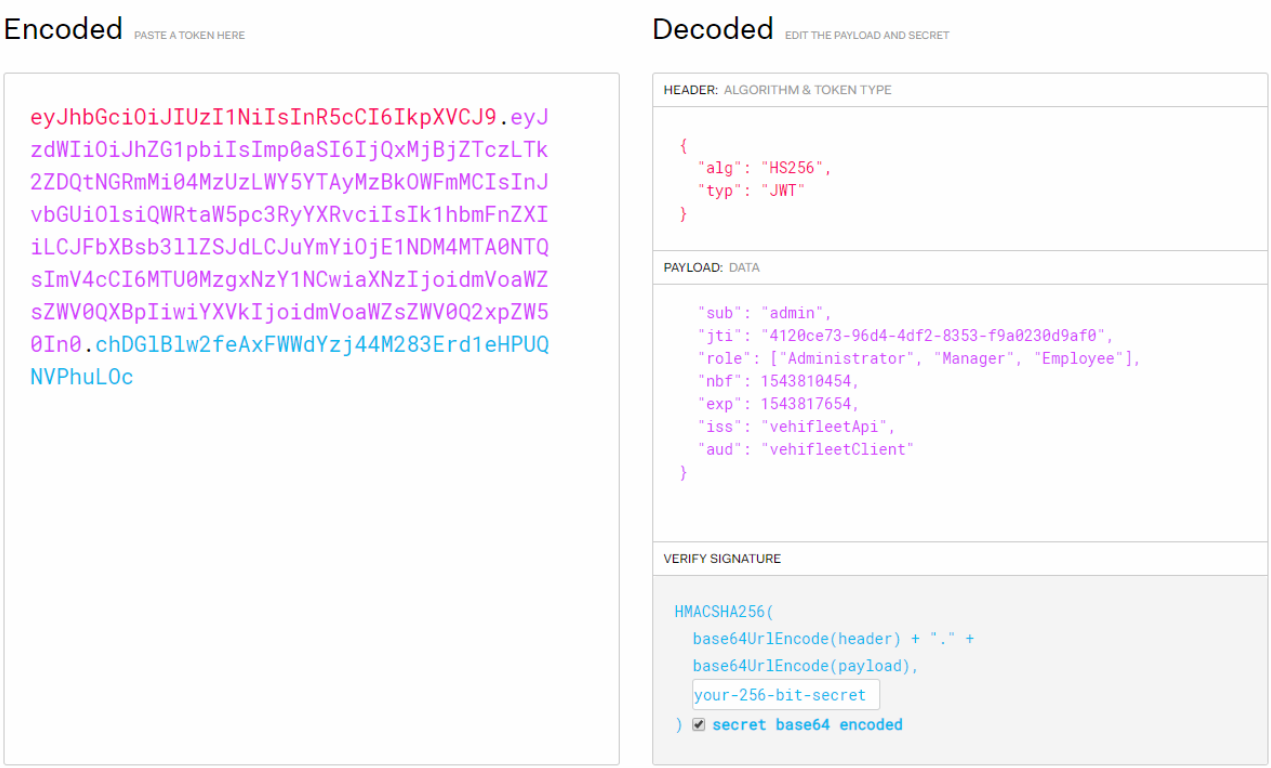
Informacje o użytkowniku: jego identyfikator (nazwa użytkownika) oraz role

4. Utworzony token zostaje zaszyfrowany (uniemożliwiając jego sfałszowanie) i zwrócony

5. Odebrany token zostaje umieszczony w pamięci przeglądarki internetowej użytkownika

Interfejs programistyczny weryfikuje poprawność tokena dla każdego żądania *HTTP* z wyjątkiem tych związanych z procesem autoryzacji użytkownika; jeżeli token jest niepoprawny lub zbyt stary (wydany więcej niż 2 godziny przed weryfikacją), żądanie jest odrzucone.

Przechowywanie ról w tokenie *JWT* pozwala na autoryzację z uwzględnieniem uprawnień użytkownika, przykładowo, ograniczając dostęp do poufnych informacji lub modyfikacji przechowywanych danych przez osoby nieuprawnione.



Rysunek 5.2 Przykładowy token *JWT*
Token został wygenerowany przy użyciu narzędzia ze strony *https://jwt.io/*.

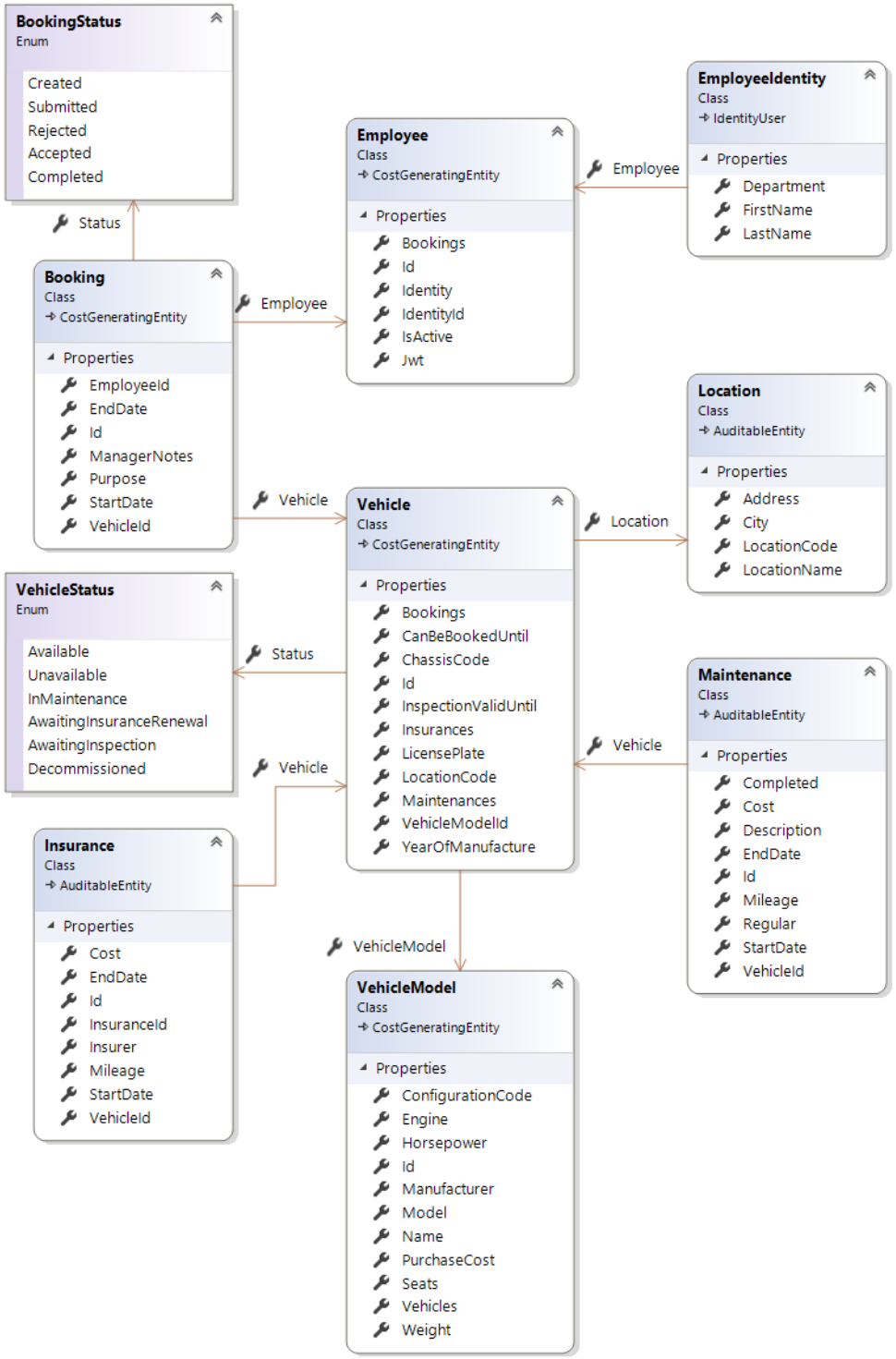
Tablica 5.2 Domyślna konfiguracja relacji ról do poziomu uprawnień

Aktor	Poziom dostępu	Wymagane role
Kierowca	Podstawowy	Employee
Kierownik	Pełny	Manager lub Administrator

5.4 Interfejs programistyczny

5.4.1 Logika biznesowa

Baza danych została automatycznie wygenerowana na podstawie klas opisujących świat biznesowy, przy użyciu *EF Core*.



Rysunek 5.3 Diagram klas
Diagram klas został wygenerowany przy użyciu *Visual Studio*.

Listing 5.1 Przykład definiowania relacji między klasami w code-first

```
public class Booking
{
    [Key]
    public int Id { get; set; }

    [Required]
    public int VehicleId { get; set; }

    public virtual Vehicle Vehicle { get; set; }
}

public class Vehicle
{
    [Key]
    public int Id { get; set; }

    public virtual ICollection<Booking> Bookings { get; set; }

}
```

Definiowanie właściwości kolumn wygenerowanych w bazie odbywa się poprzez umieszczenie odpowiednich adnotacji przy polach:

- [Key]: klucz główny (*PK*)
- [Required]: pole jest wymagane, nie może być puste (null)

Do zdefiniowania relacji między tabelami należy użyć pól typu takiego samego jak *PK* docelowej klasy[7]. Używanie adnotacji nie jest wymagane, o ile pole zostało nazwane według standardowej konwencji *EF Core - NazwaKlasyId* (np. *VehicleId*). Dodatkowo klasę można uzupełnić o pola nawigacyjne (*navigation properties*), pozwalające na odnoszenie się do powiązanej klasy w łatwy sposób w kodzie programu, należy jednak pamiętać że domyślnie *EF Core 2.1* nie wczytuje informacji o powiązanych obiektach; podczas komunikacji z bazą daną należy jawnie wywołać ładowanie powiązanych obiektów za pomocą metody *LINQ* Include().

Listing 5.2 Ładowanie powiązanych obiektów na przykładzie relacji wypożyczenia do pojazdu

```
public override Task<Booking> GetById(int id)
{
    return Set
        .Include(b => b.Vehicle)
        .AsNoTracking()
        .SingleOrDefaultAsync(b => b.Id == id);
}
```

5.4.2 Opis punktów końcowych

Modele pojazdów

Tablica 5.3 Punkt końcowy *api/vehicle-models GET*

<table>
<tr><td colspan="3">Opis</td></tr>
<tr><td>URL</td><td colspan="2">api/vehicle-models</td></tr>
<tr><td>Wymagane role</td><td colspan="2">Employee</td></tr>
<tr><td>Metoda</td><td colspan="2">GET</td></tr>
<tr><td colspan="3">Przykładowe odpowiedzi</td></tr>
<tr><td rowspan="2">200 (OK)</td><td>Odpowiedź</td><td><pre>
[
  {
    "id": 3,
    "manufacturer": "Ford",
    "model": "Focus",
    "configurationCode": "FF2018184",
    "engine": "1.8 TDCi",
    "horsepower": 115,
    "seats": 4,
    "weight": 1200,
    "purchaseCost": 70000,
    "hasVehicles": false,
    "mileage": 7806,
    "fuelConsumed": 0,
    "cost": 778,
    "addedOn": "2018-11-19T23
      :01:58.9837098",
    "addedBy": "admin",
    "modifiedOn": null,
    "modifiedBy": null
  },
  {
    "id": 4,
    "manufacturer": "Ford",
    "model": "Focus",
    "configurationCode": "FF2018154",
    "engine": "1.6 TDi",
    "horsepower": 85,
    "seats": 4,
    "weight": 1150,
    "purchaseCost": 60000,
    "hasVehicles": false,
    "mileage": 23845,
    "fuelConsumed": 631,
    "cost": 6521,
    "addedOn": "2018-11-19T23
      :01:58.9845431",
    "addedBy": "admin",
    "modifiedOn": null,
    "modifiedBy": null
  }
]
</pre></td></tr>
<tr><td>Opis</td><td>Lista modeli pojazdów</td></tr>
</table>

Tablica 5.4 Punkt końcowy *api/vehicle-models/manufacturers GET*

Opis		
URL	api/vehicle-models	
Wymagane role	Employee	
Metoda	GET	
Przykładowe odpowiedzi		
200 (OK)	Odpowiedź	`["Ford", "Skoda", "Toyota"]`
	Opis	Lista marek pojazdów

Tablica 5.5 Punkt końcowy *api/vehicle-models/id GET*

Opis		
URL	api/vehicle-models/id	
Wymagane role	Employee	
Metoda	GET	
Przykładowe odpowiedzi		
200 (OK)	Odpowiedź	`{ "id": 3, "manufacturer": "Ford", "model": "Focus", "configurationCode": "FF2018184", "engine": "1.8 TDCi", "horsepower": 115, "seats": 4, "weight": 1200, "purchaseCost": 70000, "hasVehicles": true, "mileage": 7806, "fuelConsumed": 0, "cost": 778, "addedOn": "2018-11-19T23 :01:58.9837098", "addedBy": "admin", "modifiedOn": null, "modifiedBy": null }`
	Opis	Model pojazdu o żądanym *id*
404 (Not Found)	Odpowiedź	`"No such vehicle model"`
	Opis	Model pojazdu o żądanym *id* nie istnieje

Tablica 5.6 Punkt końcowy *api/vehicle-models POST*

<table>
<tr><td colspan="3">Opis</td></tr>
<tr><td>URL</td><td colspan="2">api/vehicle-models</td></tr>
<tr><td>Wymagane role</td><td colspan="2">Manager, Administrator</td></tr>
<tr><td>Metoda</td><td colspan="2">POST</td></tr>
<tr><td colspan="3">Przykładowe ciało</td></tr>
<tr><td colspan="3"><pre>{
  "manufacturer": "Ford",
  "model": "Mondeo",
  "configurationCode": "FM2018184",
  "engine": "1.8 EcoBoost",
  "horsepower": 155,
  "seats": 4,
  "weight": 1300,
  "purchaseCost": 120000,
  "hasVehicles": true,
  "mileage": 49089,
  "fuelConsumed": 2041,
  "cost": 21809
}</pre></td></tr>
<tr><td colspan="3">Przykładowe odpowiedzi</td></tr>
<tr><td rowspan="2">200 (OK)</td><td>Odpowiedź</td><td>Id utworzonego modelu pojazdu.</td></tr>
<tr><td>Opis</td><td>Utworzenie nowego modelu pojazdu</td></tr>
</table>

Tablica 5.7 Punkt końcowy *api/vehicle-models/id PUT*

<table>
<tr><td colspan="3">Opis</td></tr>
<tr><td>URL</td><td colspan="2">api/vehicle-models/id</td></tr>
<tr><td>Wymagane role</td><td colspan="2">Manager, Administrator</td></tr>
<tr><td>Metoda</td><td colspan="2">GET</td></tr>
<tr><td colspan="3">Przykładowe ciało</td></tr>
<tr><td colspan="3"><pre>{
  "id": 5,
  "manufacturer": "Ford",
  "model": "Mondeo",
  "configurationCode": "FM2018184",
  "engine": "1.8 EcoBoost",
  "horsepower": 155,
  "seats": 4,
  "weight": 1300,
  "purchaseCost": 120000,
  "hasVehicles": true,
  "mileage": 49089,
  "fuelConsumed": 2041,
  "cost": 21809,
  "addedOn": "2018-11-19T23:01:58.9845434",
  "addedBy": "admin",
  "modifiedOn": null,
  "modifiedBy": null
}</pre></td></tr>
<tr><td colspan="3">Przykładowe odpowiedzi</td></tr>
<tr><td>200 (OK)</td><td>Opis</td><td>Model pojazdu został zaktualizowany</td></tr>
<tr><td rowspan="2">404 (Not Found)</td><td>Odpowiedź</td><td>"No such vehicle model."</td></tr>
<tr><td>Opis</td><td>Model pojazdu o żądanym id nie istnieje</td></tr>
</table>

Tablica 5.8 Punkt końcowy *api/vehicle-models/id DELETE*

Opis		
URL	api/vehicle-models/id	
Wymagane role	Manager, Administrator	
Metoda	DELETE	
Przykładowe odpowiedzi		
200 (OK)	Opis	Model pojazdu został usunięty
404 (Not Found)	Odpowiedź	"No such vehicle model."
	Opis	Model pojazdu o żądanym *id* nie istnieje
400 (Bad Request)	Odpowiedź	"Vehicle model has vehicles."
	Opis	System posiada egzemplarze modelu pojazdu o żądanym *id*, nie może on zostać usunięty

Pojazdy

Tablica 5.9 Punkt końcowy *api/vehicles GET*

<table>
<tr><td colspan="3">Opis</td></tr>
<tr><td>URL</td><td colspan="2">api/vehicles</td></tr>
<tr><td>Wymagane role</td><td colspan="2">Employee</td></tr>
<tr><td>Metoda</td><td colspan="2">GET</td></tr>
<tr><td colspan="3">Przykładowe odpowiedzi</td></tr>
<tr><td rowspan="2">200 (OK)</td><td>Odpowiedź</td><td><pre>
[
  {
    "id": 31,
    "manufacturer": "Ford",
    "model": "Focus",
    "horsepower": 115,
    "seats": 4,
    "yearOfManufacture": 2016,
    "locationCode": "KRK-1",
    "canBeBookedUntil": "2019-07-07T00
      :00:00",
    "status": "Available",
    "chassisCode": "I1Y9HNIMESHU"
  },
  {
    "id": 32,
    "manufacturer": "Ford",
    "model": "C-Max",
    "horsepower": 105,
    "seats": 5,
    "yearOfManufacture": 2016,
    "locationCode": "WRO-1",
    "canBeBookedUntil": "2019-03-10T00
      :00:00",
    "status": "Available",
    "chassisCode": "571VIXS34I71"
  }
]
</pre></td></tr>
<tr><td>Opis</td><td>Lista pojazdów</td></tr>
</table>

Tablica 5.10 Punkt końcowy *api/vehicles/id GET*

<table>
<tr><td colspan="3">Opis</td></tr>
<tr><td>URL</td><td colspan="2">api/vehicles/id</td></tr>
<tr><td>Wymagane role</td><td colspan="2">Employee</td></tr>
<tr><td>Metoda</td><td colspan="2">GET</td></tr>
<tr><td colspan="3">Przykładowe odpowiedzi</td></tr>
<tr><td rowspan="2">200 (OK)</td><td>Odpowiedź</td><td><pre>{
  "id": 51,
  "vehicleModelId": 9,
  "manufacturer": "Ford",
  "model": "S-Max",
  "seats": 6,
  "horsepower": 105,
  "engine": "1.6 TDDi",
  "status": "Available",
  "licensePlate": "WRO CFKK",
  "yearOfManufacture": 2016,
  "chassisCode": "1X6HF531O8QH",
  "locationCode": "WRO-3",
  "inspectionValidUntil": "2019-03-03T00
    :00:00",
  "canBeBookedUntil": "2019-02-23T00
    :00:00",
  "hasBookings": true,
  "mileage": 9934,
  "fuelConsumed": 668,
  "cost": 7050,
  "addedOn": "2018-11-21T20
    :46:16.0330605",
  "addedBy": "admin",
  "modifiedOn": null,
  "modifiedBy": null
}</pre></td></tr>
<tr><td>Opis</td><td>Pojazd o żądanym id</td></tr>
<tr><td rowspan="2">404 (Not Found)</td><td>Odpowiedź</td><td><pre>"No such vehicle."</pre></td></tr>
<tr><td>Opis</td><td>Pojazd o żądanym id nie istnieje</td></tr>
</table>

Tablica 5.11 Punkt końcowy *api/vehicles POST*

<table>
<tr><td colspan="3">Opis</td></tr>
<tr><td>URL</td><td colspan="2">api/vehicles</td></tr>
<tr><td>Wymagane role</td><td colspan="2">Manager, Administrator</td></tr>
<tr><td>Metoda</td><td colspan="2">POST</td></tr>
<tr><td colspan="3">Przykładowe ciało</td></tr>
<tr><td colspan="3"><pre>
{
  "vehicleModelId": 9,
  "manufacturer": "Ford",
  "model": "S-Max",
  "seats": 6,
  "horsepower": 105,
  "engine": "1.6 TDDi",
  "status": "Available",
  "licensePlate": "WRO CFKK",
  "yearOfManufacture": 2016,
  "chassisCode": "1X6HF531O8QH",
  "locationCode": "WRO-3",
  "inspectionValidUntil": "2019-03-03T00:00:00",
  "canBeBookedUntil": "2019-02-23T00:00:00",
  "hasBookings": true,
  "mileage": 9934,
  "fuelConsumed": 668,
  "cost": 7050
}
</pre></td></tr>
<tr><td colspan="3">Przykładowe odpowiedzi</td></tr>
<tr><td rowspan="2">200 (OK)</td><td>Odpowiedź</td><td>Id utworzonego pojazdu</td></tr>
<tr><td>Opis</td><td>Utworzenie nowego pojazdu</td></tr>
</table>

Tablica 5.12 Punkt końcowy *api/vehicles/id PUT*

<table>
<tr><td colspan="3">Opis</td></tr>
<tr><td>URL</td><td colspan="2">api/vehicles/id</td></tr>
<tr><td>Wymagane role</td><td colspan="2">Manager, Administrator</td></tr>
<tr><td>Metoda</td><td colspan="2">GET</td></tr>
<tr><td colspan="3">Przykładowe ciało</td></tr>
<tr><td colspan="3"><pre>
{
  "id": 51,
  "vehicleModelId": 9,
  "manufacturer": "Ford",
  "model": "S-Max",
  "seats": 6,
  "horsepower": 105,
  "engine": "1.6 TDDi",
  "status": "Available",
  "licensePlate": "WRO CFKK",
  "yearOfManufacture": 2016,
  "chassisCode": "1X6HF531O8QH",
  "locationCode": "WRO-3",
  "inspectionValidUntil": "2019-03-03T00:00:00",
  "canBeBookedUntil": "2019-02-23T00:00:00",
  "hasBookings": true,
  "mileage": 9934,
  "fuelConsumed": 668,
  "cost": 7050,
  "addedOn": "2018-11-21T20:46:16.0330605",
  "addedBy": "admin",
  "modifiedOn": null,
  "modifiedBy": null
}
</pre></td></tr>
<tr><td colspan="3">Przykładowe odpowiedzi</td></tr>
<tr><td>200 (OK)</td><td>Opis</td><td>Pojazd został zaktualizowany</td></tr>
<tr><td rowspan="2">404 (Not Found)</td><td>Odpowiedź</td><td>"No such vehicle."</td></tr>
<tr><td>Opis</td><td>Pojazd o żądanym id nie istnieje</td></tr>
</table>

Tablica 5.13 Punkt końcowy *api/vehicles/id DELETE*

Opis		
URL	api/vehicles/id	
Wymagane role	Manager, Administrator	
Metoda	DELETE	
Przykładowe odpowiedzi		
200 (OK)	Opis	Pojazd został usunięty
404 (Not Found)	Odpowiedź	"No such vehicle."
	Opis	Pojazd o żądanym *id* nie istnieje
400 (Bad Request)	Odpowiedź	"Vehicle has bookings."
	Opis	System posiada rezerwacje przypisane do pojazdu o żądanym *id*, nie może on zostać usunięty

Ubezpieczenia

Tablica 5.14 Punkt końcowy *api/insurances/vehicle/id GET*

<table>
<tr><td colspan="3">Opis</td></tr>
<tr><td>URL</td><td colspan="2">api/insurances</td></tr>
<tr><td>Wymagane role</td><td colspan="2">Manager, Administrator</td></tr>
<tr><td>Metoda</td><td colspan="2">GET</td></tr>
<tr><td colspan="3">Przykładowe odpowiedzi</td></tr>
<tr><td rowspan="2">200 (OK)</td><td>Odpowiedź</td><td><pre>
[
  {
    "id": 105,
    "vehicleId": 31,
    "startDate": "2017-09-03T00:00:00",
    "endDate": "2018-09-03T00:00:00",
    "cost": 683,
    "insurer": "Protecto",
    "insuranceId": "INS-2017-9-3-1004",
    "mileage": 4739950,
    "addedOn": "2018-11-21T20
      :49:13.6555747",
    "addedBy": "admin",
    "modifiedOn": null,
    "modifiedBy": null
  },
  {
    "id": 106,
    "vehicleId": 31,
    "startDate": "2016-09-03T00:00:00",
    "endDate": "2017-09-03T00:00:00",
    "cost": 682,
    "insurer": "Goldberg Insurance",
    "insuranceId": "INS-2016-9-3-1140",
    "mileage": 4737600,
    "addedOn": "2018-11-21T20
      :49:13.6549581",
    "addedBy": "admin",
    "modifiedOn": null,
    "modifiedBy": null
  }
]
</pre></td></tr>
<tr><td>Opis</td><td>Lista ubezpieczeń dla danego pojazdu</td></tr>
</table>

Tablica 5.15 Punkt końcowy *api/insurances/id GET*

Opis		
URL	api/insurances/id	
Wymagane role	Manager, Administrator	
Metoda	GET	
Przykładowe odpowiedzi		

200 (OK)	Odpowiedź	{
		"id": 105,
		"vehicleId": 31,
		"startDate": "2017-09-03T00:00:00",
		"endDate": "2018-09-03T00:00:00",
		"cost": 683,
		"insurer": "Protecto",
		"insuranceId": "INS-2017-9-3-1004",
		"mileage": 4739950,
		"addedOn": "2018-11-21T20
		:49:13.6555747",
		"addedBy": "admin",
		"modifiedOn": null,
		"modifiedBy": null
		}

	Opis	Ubezpieczenie o żądanym *id*
404 (Not Found)	Odpowiedź	"No such insurance."
	Opis	Ubezpieczenie o żądanym *id* nie istnieje

Tablica 5.16 Punkt końcowy *api/insurances POST*

Opis		
URL	api/insurances	
Wymagane role	Manager, Administrator	
Metoda	POST	
Przykładowe ciało		

		{
		"vehicleId": 31,
		"startDate": "2017-09-03T00:00:00",
		"endDate": "2018-09-03T00:00:00",
		"cost": 683,
		"insurer": "Protecto",
		"insuranceId": "INS-2017-9-3-1004",
		"mileage": 4739950
		}

Przykładowe odpowiedzi		
200 (OK)	Odpowiedź	*Id* utworzonego ubezpieczenia
	Opis	Utworzenie nowego ubezpieczenia

Tablica 5.17 Punkt końcowy *api/insurances/id PUT*

Opis		
URL	api/insurances/id	
Wymagane role	Manager, Administrator	
Metoda	PUT	
Przykładowe ciało		

{
"id": 105,
"vehicleId": 31,
"startDate": "2017-09-03T00:00:00",
"endDate": "2018-09-03T00:00:00",
"cost": 683,
"insurer": "Protecto",
"insuranceId": "INS-2017-9-3-1004",
"mileage": 4739950,
"addedOn": "2018-11-21T20:49:13.6555747",
"addedBy": "admin",
"modifiedOn": null,
"modifiedBy": null
}		

Przykładowe odpowiedzi		
200 (OK)	Opis	Ubezpieczenie zostało zaktualizowane
404 (Not Found)	Odpowiedź	"No such insurance."
	Opis	Ubezpieczenie o żądanym *id* nie istnieje

Tablica 5.18 Punkt końcowy *api/insurances/id DELETE*

Opis		
URL	api/insurances/id	
Wymagane role	Manager, Administrator	
Metoda	DELETE	
Przykładowe odpowiedzi		
200 (OK)	Opis	Ubezpieczenie zostało usunięte
404 (Not Found)	Odpowiedź	"No such insurance."
	Opis	Ubezpieczenie o żądanym *id* nie istnieje

Naprawy

Tablica 5.19 Punkt końcowy *api/maintenances/vehicle/id GET*

Opis		
URL	api/maintenances	
Wymagane role	Manager, Administrator	
Metoda	GET	
Przykładowe odpowiedzi		
200 (OK)	Odpowiedź	[{ "id": 149, "vehicleId": 55, "startDate": "2017-04-09T00:00:00", "endDate": "2017-04-23T00:00:00", "description": "Maintenance example generated during seeding.", "mileage": 6006626, "regular": true, "completed": true, "cost": 1154, "addedOn": "2018-11-21T20:49:14.513674", "addedBy": "admin", "modifiedOn": null, "modifiedBy": null }, { "id": 150, "vehicleId": 55, "startDate": "2017-09-06T00:00:00", "endDate": "2017-09-16T00:00:00", "description": "Maintenance example generated during seeding.", "mileage": 6006626, "regular": false, "completed": true, "cost": 457, "addedOn": "2018-11-21T20:49:14.5136756", "addedBy": "admin", "modifiedOn": null, "modifiedBy": null }]
	Opis	Lista napraw dla danego pojazdu

Tablica 5.20 Punkt końcowy *api/maintenances/id GET*

<table>
<tr><td colspan="3">Opis</td></tr>
<tr><td>URL</td><td colspan="2">api/maintenances/id</td></tr>
<tr><td>Wymagane role</td><td colspan="2">Manager, Administrator</td></tr>
<tr><td>Metoda</td><td colspan="2">GET</td></tr>
<tr><td colspan="3">Przykładowe odpowiedzi</td></tr>
<tr><td rowspan="2">200 (OK)</td><td>Odpowiedź</td><td><pre>{
  "id": 149,
  "vehicleId": 55,
  "startDate": "2017-04-09T00:00:00",
  "endDate": "2017-04-23T00:00:00",
  "description": "Maintenance example
     generated during seeding.",
  "mileage": 6006626,
  "regular": true,
  "completed": true,
  "cost": 1154,
  "addedOn": "2018-11-21T20
     :49:14.513674",
  "addedBy": "admin",
  "modifiedOn": null,
  "modifiedBy": null
}</pre></td></tr>
<tr><td>Opis</td><td>Naprawa o żądanym id</td></tr>
<tr><td rowspan="2">404 (Not Found)</td><td>Odpowiedź</td><td><pre>"No such maintenance."</pre></td></tr>
<tr><td>Opis</td><td>Naprawa o żądanym id nie istnieje</td></tr>
</table>

Tablica 5.21 Punkt końcowy *api/maintenances POST*

Opis		
URL	api/maintenances	
Wymagane role	Manager, Administrator	
Metoda	POST	
Przykładowe ciało		

{
"id": 149,
"vehicleId": 55,
"startDate": "2017-04-09T00:00:00",
"endDate": "2017-04-23T00:00:00",
"description": "Maintenance example generated during seeding.",
"mileage": 6006626,
"regular": true,
"completed": true,
"cost": 1154
}		

Przykładowe odpowiedzi		
200 (OK)	Odpowiedź	*Id* utworzonej naprawy
	Opis	Utworzenie nowej naprawy

Tablica 5.22 Punkt końcowy *api/maintenances/id PUT*

Opis		
URL	api/maintenances/id	
Wymagane role	Manager, Administrator	
Metoda	PUT	
Przykładowe ciało		

{
"id": 149,
"vehicleId": 55,
"startDate": "2017-04-09T00:00:00",
"endDate": "2017-04-23T00:00:00",
"description": "Maintenance example generated during seeding.",
"mileage": 6006626,
"regular": true,
"completed": true,
"cost": 1154,
"addedOn": "2018-11-21T20:49:14.513674",
"addedBy": "admin",
"modifiedOn": null,
"modifiedBy": null
}		

Przykładowe odpowiedzi		
200 (OK)	Opis	Naprawa została zaktualizowana
404 (Not Found)	Odpowiedź	"No such maintenance."
	Opis	Naprawa o żądanym *id* nie istnieje

Tablica 5.23 Punkt końcowy *api/maintenances/id DELETE*

Opis		
URL	api/maintenances/id	
Wymagane role	Manager, Administrator	
Metoda	DELETE	
Przykładowe odpowiedzi		
200 (OK)	Opis	Naprawa została usunięta
404 (Not Found)	Odpowiedź	"No such maintenance."
	Opis	Naprawa o żądanym *id* nie istnieje

Wypożyczenia

Tablica 5.24 Punkt końcowy *api/bookings/vehicle/id GET*

Opis		
URL	api/booking	
Wymagane role	Employee	
Metoda	GET	
Przykładowe odpowiedzi		
200 (OK)	Odpowiedź	[{ "id": 140, "employeeId": 5, "employeeUserName": "kkrzysztofowski", "vehicle": "Ford Focus", "status": "Completed", "startDate": "2017-01-24T00:00:00", "endDate": "2017-02-06T00:00:00" }, { "id": 141, "employeeId": 3, "employeeUserName": "mmarkowski", "vehicle": "Skoda Octavia", "status": "Completed", "startDate": "2017-02-07T00:00:00", "endDate": "2017-02-20T00:00:00" }]
	Opis	Lista napraw dla danego pojazdu

Tablica 5.25 Punkt końcowy *api/bookings/id GET*

<table>
<tr><td colspan="3">Opis</td></tr>
<tr><td>URL</td><td colspan="2">api/bookings/id</td></tr>
<tr><td>Wymagane role</td><td colspan="2">Employee</td></tr>
<tr><td>Metoda</td><td colspan="2">GET</td></tr>
<tr><td colspan="3">Przykładowe odpowiedzi</td></tr>
<tr><td rowspan="2">200 (OK)</td><td>Odpowiedź</td><td><pre>{
  "id": 141,
  "vehicleId": 48,
  "employeeId": 3,
  "status": "Completed",
  "startDate": "2017-02-07T00:00:00",
  "endDate": "2017-02-20T00:00:00",
  "purpose": "Example purpose generated
     during seeding.",
  "notes": null,
  "mileage": 158,
  "fuelConsumed": 0,
  "cost": 0,
  "addedOn": "2018-11-21T20
     :49:14.2964683",
  "addedBy": "admin",
  "modifiedOn": null,
  "modifiedBy": null
}</pre></td></tr>
<tr><td>Opis</td><td>Wypożyczenie o żądanym id</td></tr>
<tr><td rowspan="2">404 (Not Found)</td><td>Odpowiedź</td><td><pre>"No such booking."</pre></td></tr>
<tr><td>Opis</td><td>Wypożyczenie o żądanym id nie istnieje</td></tr>
</table>

Tablica 5.26 Punkt końcowy *api/booking POST*

<table>
<tr><td colspan="3">Opis</td></tr>
<tr><td>URL</td><td colspan="2">api/booking</td></tr>
<tr><td>Wymagane role</td><td colspan="2">Employee</td></tr>
<tr><td>Metoda</td><td colspan="2">POST</td></tr>
<tr><td colspan="3">Przykładowe ciało</td></tr>
<tr><td colspan="3"><pre>
{
  "vehicleId": 48,
  "employeeId": 3,
  "status": "Completed",
  "startDate": "2017-02-07T00:00:00",
  "endDate": "2017-02-20T00:00:00",
  "purpose": "Example purpose generated during seeding.",
  "notes": null,
  "mileage": 158,
  "fuelConsumed": 15,
  "cost": 60
}
</pre></td></tr>
<tr><td colspan="3">Przykładowe odpowiedzi</td></tr>
<tr><td rowspan="2">200 (OK)</td><td>Odpowiedź</td><td>Id utworzonego wypożyczenia</td></tr>
<tr><td>Opis</td><td>Utworzenie nowego wypożyczenia</td></tr>
<tr><td rowspan="2">400 (Bad Request)</td><td>Odpowiedź</td><td>"No such employee."</td></tr>
<tr><td>Opis</td><td>Pracownik powiązany z wypożyczeniem nie istnieje</td></tr>
<tr><td rowspan="2">400 (Bad Request)</td><td>Odpowiedź</td><td>"No such vehicle."</td></tr>
<tr><td>Opis</td><td>Pojazd powiązany z wypożyczeniem nie istnieje</td></tr>
</table>

Tablica 5.27 Punkt końcowy *api/bookings/id PUT*

Opis		
URL	api/bookings/id	
Wymagane role	Employee	
Metoda	PUT	
Przykładowe ciało		

{
"id": 141,
"vehicleId": 48,
"employeeId": 3,
"status": "Completed",
"startDate": "2017-02-07T00:00:00",
"endDate": "2017-02-20T00:00:00",
"purpose": "Example purpose generated during seeding.",
"notes": null,
"mileage": 158,
"fuelConsumed": 15,
"cost": 60,
"addedOn": "2018-11-21T20:49:14.2964683",
"addedBy": "admin",
"modifiedOn": null,
"modifiedBy": null
}		

Przykładowe odpowiedzi		
200 (OK)	Opis	Wypożyczenie zostało zaktualizowane
404 (Not Found)	Odpowiedź	"No such booking."
	Opis	Wypożyczenie o żądanym *id* nie istnieje

Tablica 5.28 Punkt końcowy *api/bookings/id DELETE*

Opis		
URL	api/bookings/id	
Wymagane role	Employee	
Metoda	DELETE	
Przykładowe odpowiedzi		
200 (OK)	Opis	Wypożyczenie zostało usunięte
404 (Not Found)	Odpowiedź	"No such booking."
	Opis	Wypożyczenie o żądanym *id* nie istnieje
400 (Bad request)	Odpowiedź	"Cannot delete bookings that have been submitted."
	Opis	Wypożyczenie o żądanym *id* nie istnieje

5.4.3 Filtrowanie wyników żądań GET

Interfejs programistyczny umożliwia filtrowanie wyników żądań *GET* poprzez warunki przesyłane w *URL*.

Tablica 5.29 Filtr modeli pojazdów (*api/vehicle-models GET*)

Filtr modeli pojazdów	
URL	api/vehicle-models
Warunki	
Nazwa	**Opis**
Manufacturer	Producent
Przykładowy filtr	
URL	api/vehicle-models?manufacturer=Ford
API zwróci wyłącznie modele samochodów wyprodukowane przez Forda	

Tablica 5.30 Filtr pojazdów (*api/vehicles GET*)

Filtr modeli pojazdów	
URL	api/vehicle-models
Warunki	
Nazwa	**Opis**
Manufacturer	Producent
VehicleModelId	Id modelu
LocationCode	Id obecnej lokalizacji
ChassisCode	Kod karoserii
MinBookingDays	Minimalna ilość dni w których pojazd jest dostępny
Status	Obecny status pojazdu
Przykładowy filtr	
URL	api/vehicles?Manufacturer=Ford&LocationCode=KRK-1&MinBookingDays=30
API zwróci wyłącznie samochodowy wyprodukowane przez Forda, przebywające w lokalizacji KRK-1 oraz dostępne na co najmniej 30 dni	

Tablica 5.31 Filtr wypożyczeń (*api/bookings GET*)

Filtr wypożyczeń	
URL	api/bookings
Warunki	
Nazwa	**Opis**
EmployeeId	Id pracownika który utworzył wypożyczenie
EmployeeUserName	Nazwa użytkownika który utworzył wypożyczenie
VehicleId	Id wypożyczanego pojazdu
Statuses	Dozwolone statusy wypożyczenia
FromDate	Data po której rozpoczęły się wypożyczenia
ToDate	Data przed którą zakończyły się wypożyczenia
Przykładowy filtr	
URL	api/bookings?Statuses=Completed&Statuses=Rejected&EmployeeUserName=jkowalski
API zwróci zakończone (*Completed*) lub odrzucone (*Rejected*) wypożyczenia utworzone przez użytkownika jkowalski	

Mechanizm filtrowania jest wydajny nawet dla skomplikowanych żądań - implementacja korzysta z interfejsu *IQueryable*, pozwalającego na dodawanie wielu warunków które zostaną wykonane wyłącznie raz. W trakcie wywołania funkcji *ToListAsync()*, warunki dodane do *IQueryable* zostaną przetłumaczone do pojedynczego zapytania *SQL* które zostanie wykonane w bazie danych po czym zwróci listę obiektów.

Listing 5.3 Logika filtrującą dla modeli pojazdów

```
public class VehicleModelFilter : IQueryFilter<VehicleModel>
{
    public string Manufacturer { get; set; }

    public IQueryable<VehicleModel> Filter(IQueryable<VehicleModel>
       query)
    {
        if (Manufacturer.NotNullOrEmpty())
        {
            query = query.Where(vm =>
                        vm.Manufacturer == Manufacturer);
        }

        return query;
    }
}
```

5.4.4 Eksport statystyk floty

System przechowuje informacje na temat bieżących kosztów generowanych przez flotę; raporty zawierające te informacje mogą być wyeksportowane do pliku *.csv* by następnie zostać zaimportowane do narzędzia kalkulacyjnego, takiego jak *Microsoft Excel*.

W utworzonym systemie eksport do pliku wywoływany jest przez żądanie *HTTP POST* na adres *api/reports/generate/days*, gdzie *days* to liczba określająca z jak wielu dni wstecz powinny być brane dane dotyczące wypożyczeń.

Listing 5.4 Przykładowy raport ze statystykami pojazdów

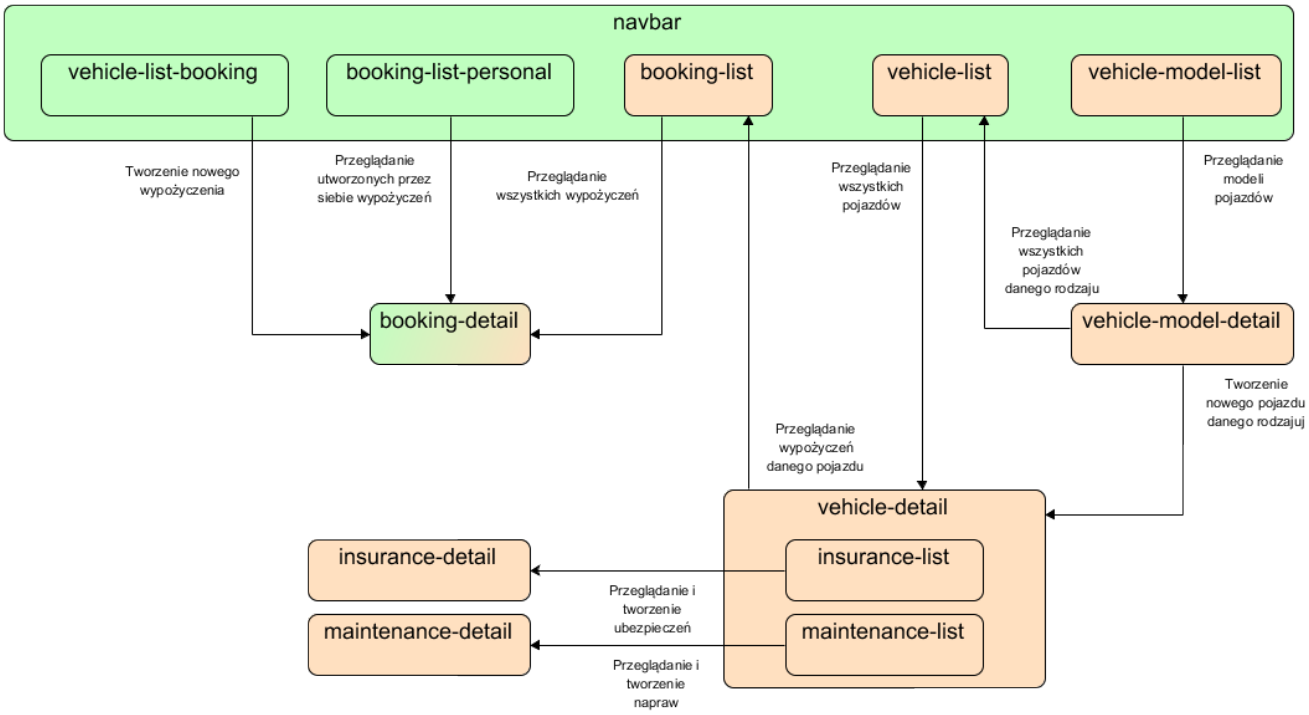
```
ChassisCode,Manufacturer,Model,YearOfManufacture,Cost,Mileage,
    FuelConsumed
I1Y9HNIMESHU,Ford,Focus,2016,"9497,00",7135,373
571VIXS34I71,Ford,C-Max,2016,"4812,00",9823,532
UTQYXSB44JGE,Toyota,Corolla,2018,"2006,00",659,0
6T7HDCTIV0BZ,Toyota,Auris,2015,"13743,00",20026,815
336J4Q7ZFI4E,Skoda,Superb,2017,"3166,00",6474,385
5SX28LAN2LPK,Ford,Focus,2015,"5858,00",13492,647
MHLI99XWS3OL,Skoda,Octavia,2018,"1023,00",1465,0
TN2VWMC54JP1,Skoda,Superb,2018,"1194,00",446,0
TZ1UXM08X7ZU,Ford,Focus,2015,"6823,00",10776,667
0UYVWETOC5C7,Toyota,Corolla,2017,"4389,00",5138,224
4VZLW2GQ7JUC,Ford,C-Max,2017,"3802,00",5781,216
NAJIA1OE0C2B,Ford,Mondeo,2016,"8742,00",9059,498
HS11MIV1AR8W,Ford,Focus,2017,"4277,00",8118,362
TIZOCIKJINBR,Skoda,Superb,2018,"3704,00",47,0
E5RQAGK8NWGE,Skoda,Superb,2017,"5618,00",4630,229
```

5.5 Interfejs użytkownika

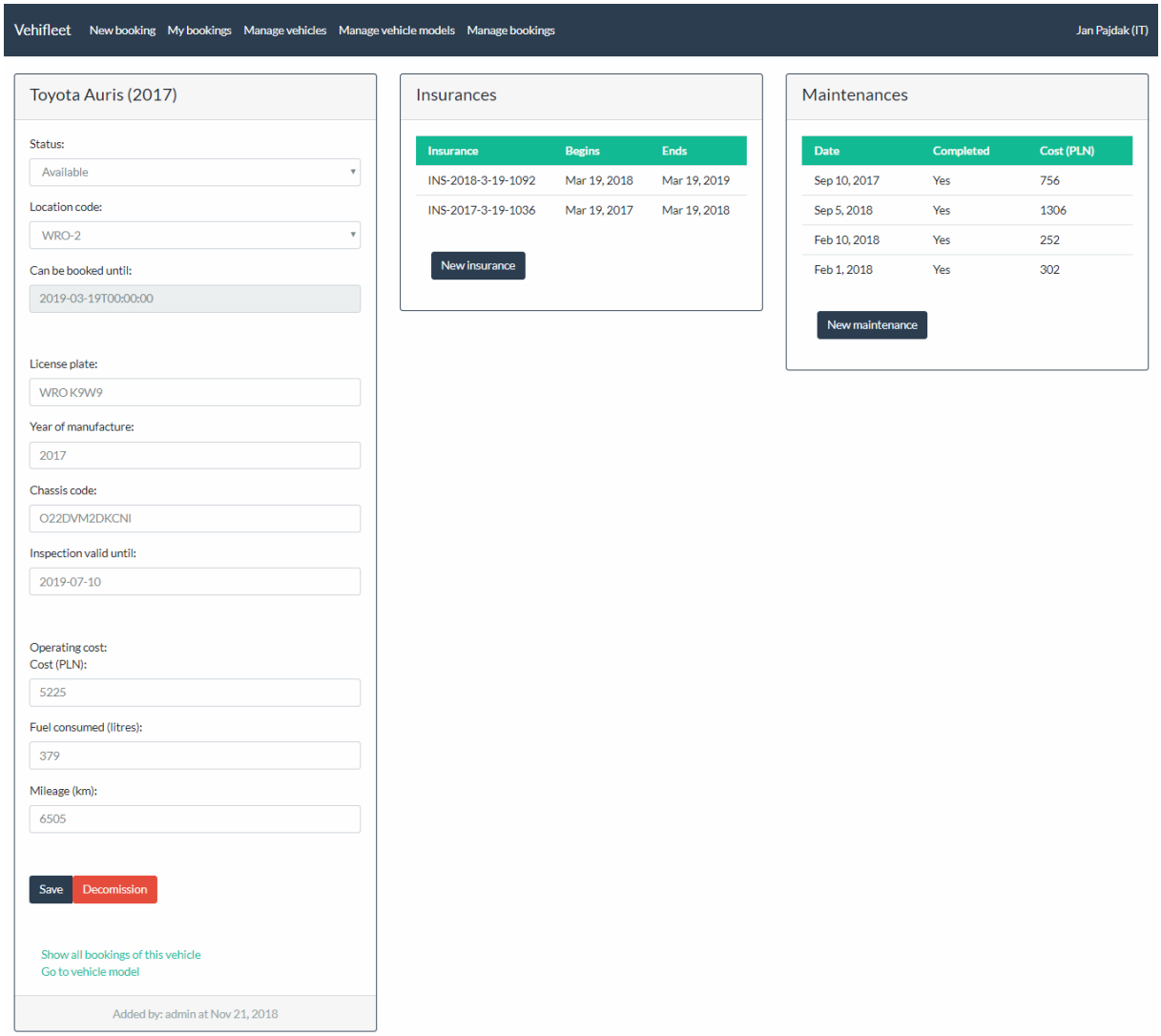
5.5.1 Układ interfejsu użytkownika

Komponenty (widoki) wchodzące w skład interfejsu użytkownika można podzielić na dwa główne rodzaje:

- **Widok szczegółowy** (*detail*) który zawiera komplet informacji o danym obiekcie i umożliwia jego edycję.
- **Widok listy** (*list*) zawierający małą ilość informacji wymaganych do identyfikacji danego obiektu oraz możliwość przejścia do **widoku szczegółowego**. Widoki tego typu są punktem wejściowym do bardziej zaawansowanej logiki interfejsu, dostępnym bezpośrednio za pomocą paska nawigacji (*navbar*).



Rysunek 5.4 Widoki interfejsu użytkownika
Widoki zielone dostępne są dla każdego użytkownika; widoki pomarańczowe wyłącznie dla użytkownika o odpowiednich uprawnieniach. Widok *booking-detail* jest specjalnym przypadkiem oferującym różne możliwości w zależności od uprawnień użytkownika.



Rysunek 5.5 Widoki interfejsu użytkownika
Widoki zielone dostępne są dla każdego użytkownika; widoki pomarańczowe wyłącznie dla użytkownika o odpowiednich uprawnieniach. Widok *booking-detail* jest specjalnym przypadkiem oferującym różne możliwości w zależności od uprawnień użytkownika.

Rozdział 6

Testy

6.1 Testy jednostkowe

System był testowany przy użyciu testów jednostkowych korzystających z biblioteki *XUnit* oraz napisanych w schludny, zgodny z często stosowaną w języku C# konwencją *AAA* sposób, bazujący na podziale testu na trzy sekcje:

1. Przygotuj (*Arrange*): przygotowanie niezbędnych zmiennych

2. Działaj (*Act*): wywołanie metod które mają być testowane

3. Sprawdź (*Assert*): sprawdzenie wyniku

Listing 6.1 Przykład testu jednostkowego zgodnego z konwencją *AAA*

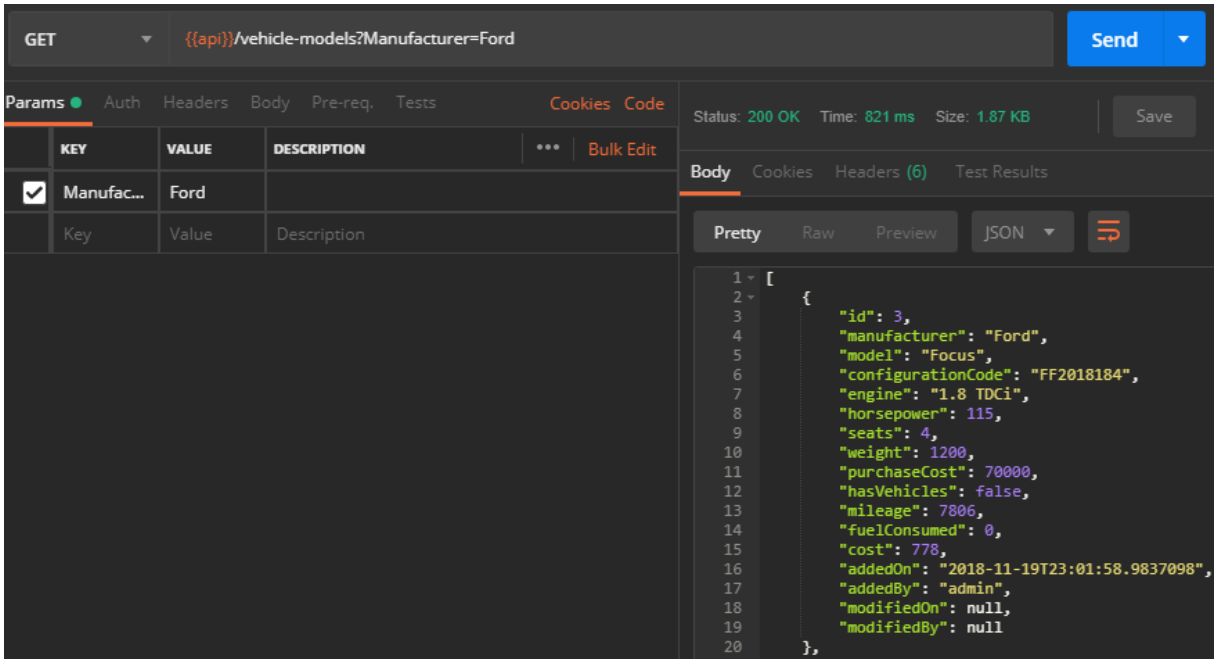
```
[Fact]
public void Should_Add_Spaces_1()
{
    // Arrange
    var text = "SomeTestText";
    var expected = "Some Test Text";

    // Act
    var actual = text.AddSpaces();

    // Assert
    Assert.Equal(expected, actual);
}
```

6.2 Testy systemowe

Najczęściej stosowanym rodzajem testów były testy systemowe przy użyciu narzędzia *Postman* pozwalającego na wygodne tworzenie i wysyłanie skomplikowanych żądań *HTTP*. *Postman* pozwala również na zapisywanie oraz organizowanie żądań co znacznie przyśpiesza proces testowania.



Rysunek 6.1 Interfejs programu Postman

6.3 Testy dymne

Testy dymne (*smoke test*) były używane w końcowej fazie projektu; polegały na wcieleniu się w rolę użytkownika i przechodzeniu najczęściej używanych ścieżek (np. tworzeniu wypożyczenia) . Testy tego typu pozwalają szybko zweryfikować czy kluczowa funkcjonalność systemu działa bezproblemowo.

Rozdział 7

Podsumowanie

7.1 Wnioski

7.2 Możliwości rozwoju

Bibliografia

[1] Angular Docs. *Introduction to components*, 2018. URL `https://angular.io/guide/architecture-components`. Dostęp 03.12.2018.

[2] Angular Docs. *Style Guide*, 2018. URL `https://angular.io/guide/styleguide`. Dostęp 03.12.2018.

[3] JWT. *JSON Web Tokens*, 2018. URL `https://jwt.io`. Dostęp 03.12.2018.

[4] MSDN. *Generics*, 2015. URL `https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/`. Dostęp 03.12.2018.

[5] MSDN. *TypeScript - Understanding TypeScript*, 2015. URL `https://msdn.microsoft.com/en-us/magazine/dn890374.aspx`. Dostęp 03.12.2018.

[6] MSDN. *Entity Framework Core*, 2016. URL `https://docs.microsoft.com/en-us/ef/core`. Dostęp 03.12.2018.

[7] MSDN. *Entity Framework Core: Relationships*, 2016. URL `https://docs.microsoft.com/en-us/ef/core/modeling/relationships`. Dostęp 03.12.2018.

[8] MSDN. *General Naming Conventions*, 2017. URL `https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/general-naming-conventions`. Dostęp 03.12.2018.

[9] MSDN. *Build web APIs with ASP.NET Core*, 2018. URL `https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-2.1`. Dostęp 03.12.2018.

Spis rysunków

Spis tablic

Listingi