

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, W4/K9

**OCENA PRACY:**

# Spis treści

<b>Spis rysunków</b>	<b>2</b>
<b>Spis tablic</b>	<b>3</b>
<b>Spis listingów</b>	<b>5</b>
<b>1 Wstęp</b>	<b>7</b>
1.1 Wprowadzenie . . . . .	7
1.2 Cel i zakres pracy . . . . .	7
1.3 Układ pracy . . . . .	7
<b>2 Istniejące rozwiązania</b>	<b>9</b>
2.1 Fleetly . . . . .	9
2.2 Vinitysoft Fleet Manager . . . . .	10
<b>3 Wymagania funkcjonalne i нефункционалне</b>	<b>11</b>
3.1 Wymagania funkcjonalne . . . . .	11
3.2 Wymagania нефункционалне . . . . .	11
3.2.1 Interfejs użytkownika . . . . .	11
3.2.2 Interfejs programistyczny . . . . .	12
3.2.3 Bezpieczeństwo . . . . .	12
<b>4 Zastosowane technologie i narzędzia</b>	<b>13</b>
4.1 Zastosowane technologie . . . . .	13
4.1.1 Angular . . . . .	13
4.1.2 Bootstrap . . . . .	14
4.1.3 ASP.NET Core i EF Core . . . . .	14
4.2 Wykorzystane narzędzia . . . . .	14
4.2.1 Git . . . . .	14
4.2.2 Visual Studio . . . . .	14
4.2.3 Visual Studio Code . . . . .	14
4.2.4 SQL Server Management Studio . . . . .	15
4.2.5 Postman . . . . .	15
<b>5 Projekt i implementacja</b>	<b>16</b>
5.1 Przypadki użycia . . . . .	16
5.1.1 Szczegółowy opis przypadków użycia . . . . .	17
5.2 Architektura . . . . .	21
5.3 Standardy . . . . .	22
5.4 Interfejs programistyczny . . . . .	23

5.4.1	Logika biznesowa . . . . .	23
5.4.2	Baza danych . . . . .	26
5.4.3	Struktura solucji . . . . .	28
5.4.4	Wstrzykiwanie zależności . . . . .	28
5.4.5	Bezpieczeństwo . . . . .	29
5.4.6	Kontrolery . . . . .	30
5.4.7	Eksport statystyk floty . . . . .	39
5.5	Interfejs użytkownika . . . . .	40
5.5.1	Układ interfejsu użytkownika . . . . .	40
5.6	Walidacja danych . . . . .	40
5.7	Stopka audytowa . . . . .	42
5.8	Dialogi . . . . .	42
<b>6</b>	<b>Testy</b>	<b>43</b>
6.1	Testy jednostkowe . . . . .	43
6.2	Testy systemowe . . . . .	45
6.3	Testy dymne . . . . .	45
<b>7</b>	<b>Podsumowanie</b>	<b>46</b>
7.1	Wnioski . . . . .	46
7.2	Możliwości rozwoju . . . . .	46
	<b>Literatura</b>	<b>46</b>
	<b>Dodatki</b>	<b>48</b>
<b>A</b>	<b>Instrukcja użytkownika</b>	<b>48</b>
A.1	Ekran logowania . . . . .	48
A.2	Ekran wylogowywania . . . . .	49
A.3	Dostępne pojazdy . . . . .	50
A.4	Historia rezerwacji . . . . .	51
A.5	Wszystkie rezerwacje . . . . .	52
A.6	Szczegóły rezerwacji . . . . .	53
A.6.1	Szczegóły rezerwacji w trybie kierownika . . . . .	54
A.7	Wszystkie pojazdy . . . . .	55
A.8	Szczegóły pojazdu . . . . .	56
A.9	Szczegóły ubezpieczenia . . . . .	57
A.10	Szczegóły naprawy . . . . .	57
A.11	Wszystkie modele pojazdów . . . . .	58
A.12	Szczegóły modelu pojazdu . . . . .	59

# Spis rysunków

2.1	Interfejs użytkownika programu Fleetly . . . . .	9
2.2	Interfejs użytkownika programu Vinitysoft Fleet Management Software 4.0 . . . . .	10
4.1	Przykład kompilacji kodu TypeScript do JavaScript . . . . .	13
5.1	Diagram przypadków użycia . . . . .	16
5.2	Uproszczony schemat architektury z wyodrębnionymi najważniejszymi elementami składowymi . . . . .	21
5.3	Diagram klas . . . . .	25
5.4	Diagram bazy danych . . . . .	27
5.5	Przykładowy token <i>JWT</i> . . . . .	29
5.6	Widoki interfejsu użytkownika . . . . .	40
5.7	Widok z polami zawierającymi błędne wartości . . . . .	41
5.8	Stopka audytowa . . . . .	42
5.9	Przykładowy dialog . . . . .	42
6.1	Interfejs programu Postman . . . . .	45
A.1	Widok logowania (dashboard-login) . . . . .	48
A.2	Widok zalogowanego użytkownika (dashboard-user-details) . . . . .	49
A.3	Widok listy dostępnych pojazdów (vehicle-list-booking) . . . . .	50
A.4	Widok listy historii rezerwacji (booking-personal) . . . . .	51
A.5	Widok listy historii rezerwacji (booking-list) . . . . .	52
A.6	Widok szczegółowy rezerwacji (booking-details) . . . . .	53
A.7	Widok szczegółowy rezerwacji w trybie kierownika (booking-details) . . . . .	54
A.8	Widok listy pojazdów (vehicle-list) . . . . .	55
A.9	Widok szczegółowy pojazdu (vehicle-details) . . . . .	56
A.10	Widok szczegółowy ubezpieczenia (insurance-details) . . . . .	57
A.11	Widok szczegółowy naprawy (maintenance-details) . . . . .	57
A.12	Widok listy modeli pojazdów (vehicle-model-list) . . . . .	58
A.13	Widok szczegółowy modelu pojazdu (vehicle-model-details) . . . . .	59

# Spis tablic

5.1	Przypadek użycia: Przeglądanie dostępnych pojazdów . . . . .	17
5.2	Przypadek użycia: Tworzenie rezerwacji . . . . .	17
5.3	Przypadek użycia: Przeglądanie swoich rezerwacji . . . . .	18
5.4	Przypadek użycia: Edycja rezerwacji . . . . .	18
5.5	Przypadek użycia: Przeglądanie wszystkich rezerwacji . . . . .	18
5.6	Przypadek użycia: Przeglądanie wszystkich pojazdów . . . . .	19
5.7	Przypadek użycia: Edycja pojazdów . . . . .	19
5.8	Przypadek użycia: Tworzenie/edycja ubezpieczenia. . . . .	19
5.9	Przypadek użycia: Tworzenie/edycja naprawy . . . . .	19
5.10	Przypadek użycia: Przeglądanie modeli pojazdów . . . . .	20
5.11	Przypadek użycia: Edycja modelu pojazdu . . . . .	20
5.12	Przypadek użycia: Tworzenie pojazdu . . . . .	20
5.13	Przypadek użycia: Tworzenie modelu pojazdu . . . . .	20
5.14	Przypadek użycia: Generowanie raportów . . . . .	21
5.15	Najważniejsze konwencje nazewnicze . . . . .	22
5.16	Klasy obiektów biznesowych . . . . .	23
5.17	Domyślna konfiguracja relacji ról do poziomu uprawnień . . . . .	30
5.18	Endpoint <i>api/vehicle-models GET</i> . . . . .	30
5.19	Endpoint <i>api/vehicle-models/manufacturers GET</i> . . . . .	30
5.20	Endpoint <i>api/vehicle-models/id GET</i> . . . . .	31
5.21	Endpoint <i>api/vehicle-models POST</i> . . . . .	31
5.22	Endpoint <i>api/vehicle-models/id PUT</i> . . . . .	31
5.23	Endpoint <i>api/vehicle-models/id DELETE</i> . . . . .	32
5.24	Endpoint <i>api/vehicles GET</i> . . . . .	32
5.25	Endpoint <i>api/vehicles/id GET</i> . . . . .	32
5.26	Endpoint <i>api/vehicles POST</i> . . . . .	33
5.27	Endpoint <i>api/vehicles/id PUT</i> . . . . .	33
5.28	Endpoint <i>api/vehicles/id DELETE</i> . . . . .	33
5.29	Endpoint <i>api/insurances/vehicle/id GET</i> . . . . .	34
5.30	Endpoint <i>api/insurances/id GET</i> . . . . .	34
5.31	Endpoint <i>api/insurances POST</i> . . . . .	34
5.32	Endpoint <i>api/insurances/id PUT</i> . . . . .	34
5.33	Endpoint <i>api/insurances/id DELETE</i> . . . . .	35
5.34	Endpoint <i>api/maintenances/vehicle/id GET</i> . . . . .	35
5.35	Endpoint <i>api/maintenances/id GET</i> . . . . .	35
5.36	Endpoint <i>api/maintenances POST</i> . . . . .	35
5.37	Endpoint <i>api/maintenances/id PUT</i> . . . . .	36
5.38	Endpoint <i>api/maintenances/id DELETE</i> . . . . .	36
5.39	Endpoint <i>api/bookings GET</i> . . . . .	36

5.40	Endpoint <i>api/bookings/id GET</i> . . . . .	37
5.41	Endpoint <i>api/bookings POST</i> . . . . .	37
5.42	Endpoint <i>api/bookings/id PUT</i> . . . . .	37
5.43	Endpoint <i>api/bookings/id DELETE</i> . . . . .	38

# Spis listingów

5.1	Klasa abstrakcyjna <code>AuditableEntity</code> . . . . .	23
5.2	Klasa abstrakcyjna <code>CostGeneratingEntity</code> . . . . .	24
5.3	Przykład definiowania relacji między klasami w code-first . . . . .	26
5.4	Ładowanie powiązanych obiektów na przykładzie relacji rezerwacji do po- jazdu . . . . .	26
5.5	Przykładowy raport ze statystykami pojazdów . . . . .	39
5.6	Przykładowy walidator używający wyrażenia regularnego . . . . .	40
6.1	Test jednostkowy metody rozszerzeń . . . . .	43
6.2	Test kontrolera korzystający z <i>Moq</i> . . . . .	44

# Skróty

- **API** (ang. Application Programming Interface)
- **DB** (ang. Database)
- **GC** (ang. Garbage Collector)
- **UI** (ang. User Interface)
- **JSON** (ang. JavaScript Object Notation)
- **JWT** (ang. JSON Web Token)
- **ORM** (ang. Object-Relational Mapping)
- **PK** (ang. Primary Key)
- **VS** (ang. Visual Studio)



# Rozdział 1

## Wstęp

### 1.1 Wprowadzenie

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 wypożyczalnię umożliwiającą jak największe wykorzystanie dostępnej floty pojazdów przez pracowników, którzy nie mają potrzeby posiadania firmowego samochodu na wyłączność.

Pierwszym etapem projektu jest zebranie wymagań funkcjonalnych i нефункциональных oraz określenie zakresu pracy. Drugi etap projektu to wybór technologii i projekt architektury. Ostatnim, trzecim etapem jest implementacja systemu (wraz z testami).

W realizacji projektu została zwrócona szczególna uwaga na użycie dobrych praktyk programowania oraz nowoczesnych technologii.

### 1.2 Cel i zakres pracy

Celem niniejszej pracy dyplomowej jest opracowanie oraz implementacja projektu umożliwiającego zarządzanie flotą samochodów. Aplikacja jest skierowana do firm, które nie mają potrzeby lub wystarczających środków, by zapewnić pracownikom samochody na wyłączność. Przykładowym użyciem 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 rezerwacji — 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.

### 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 нефункциональные. W kolejnym, czwartym rozdziale, znajduje

się opis wybranych technologii oraz narzędzi, wraz z uzasadnieniem. Rozdział piąty skupia się na opisie technicznym projektu oraz jego implementacji. Szósty rozdział zawiera opis sposobu testowania systemu. Ostatni, siódmy rozdział, zawiera podsumowanie projektu.

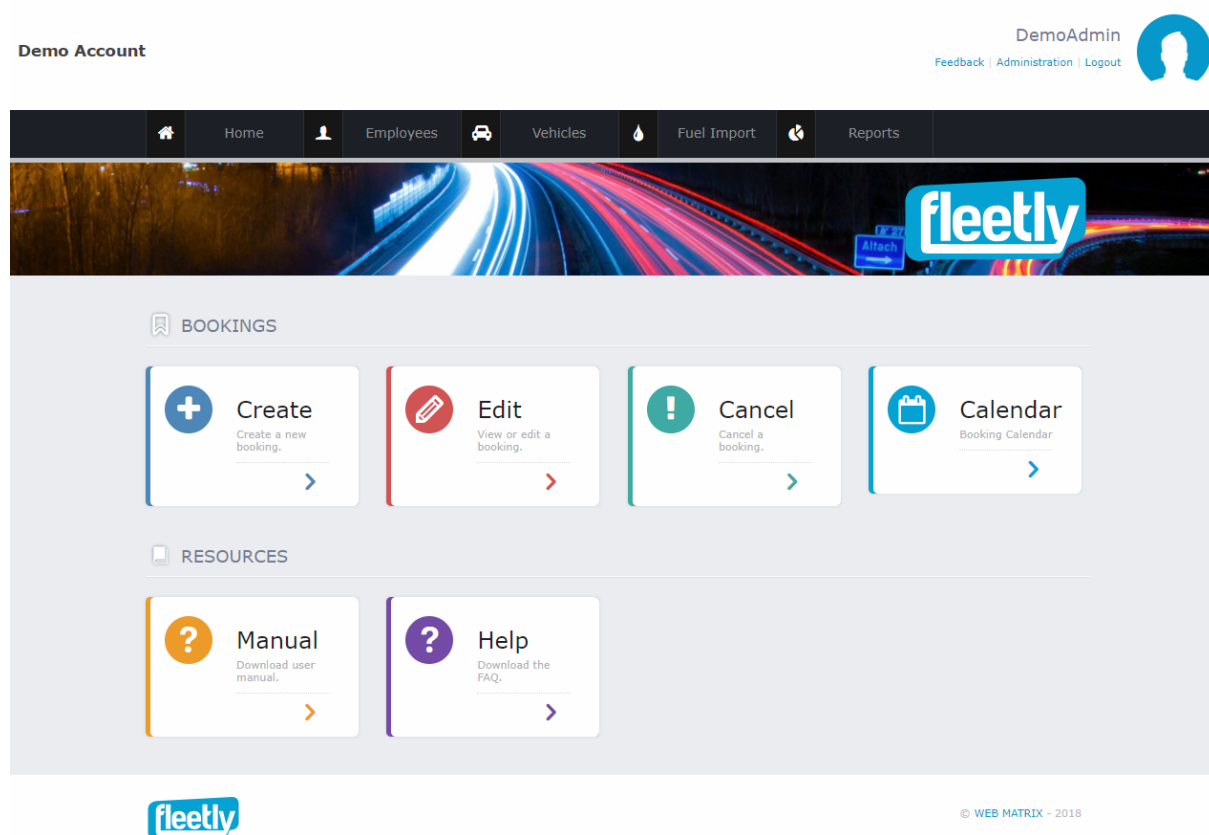
Dodatkowo, dołączona została instrukcja użytkownika.

# Rozdział 2

## Istniejące rozwiązania

### 2.1 Fleetly

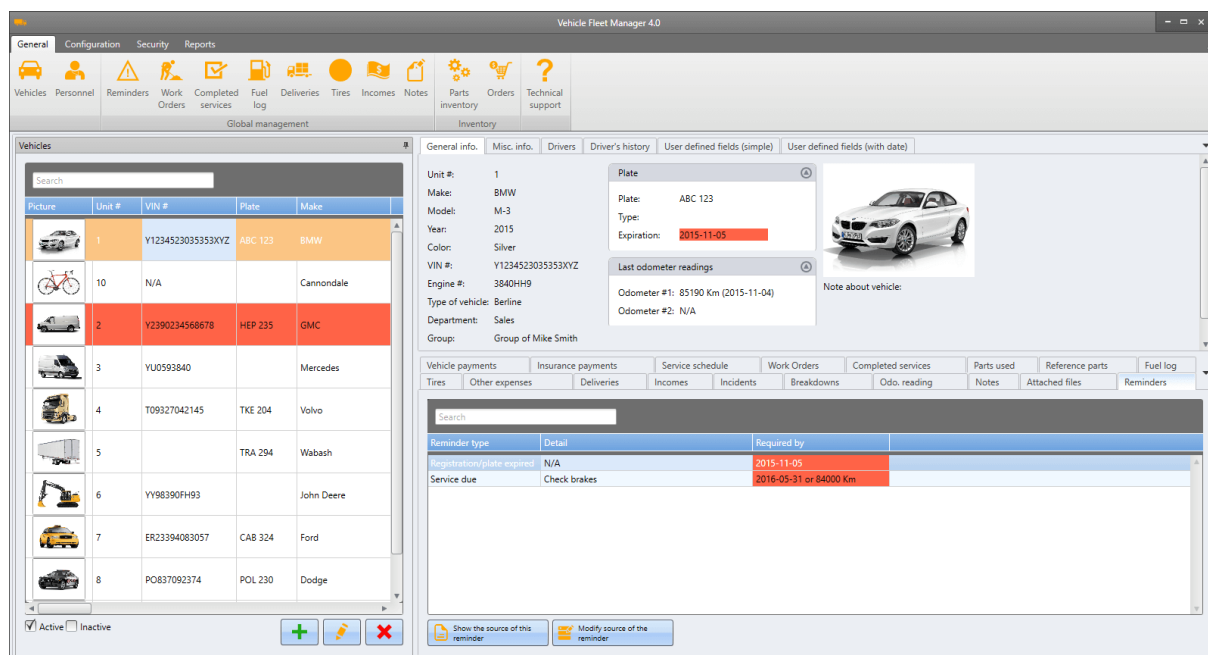
Jednym z popularniejszych rozwiązań obecnych na rynku jest *Fleetly* (<https://www.fleetly.co/>). Sposób działania *Fleetly* jest bliski działaniu systemu, który został stworzony w ramach projektu. Skupia się on jednak zaledwie na aspekcie wypożyczalni i pomija funkcjonalności przydatne w prowadzeniu firmy niezwiązanej z wypożyczaniem samochodów. Dodatkowym problemem *Fleetly* jest przechowywanie danych w chmurze — wiele firm preferuje posiadanie własnych systemów ze względów bezpieczeństwa danych. Jednym z większych problemów *Fleetly* jest wysoki koszt licencji. System nie oferuje również integracji z istniejącymi zasobami firmy.



Rysunek 2.1 Interfejs użytkownika programu Fleetly

## 2.2 Vinitysoft Fleet Manager

Kolejną popularną aplikacją jest *Vinitysoft Fleet Manager* (<https://www.vinitysoft.com/>). *Vinitysoft Fleet Manager* kładzie mały nacisk na kontrolę dostępu do pojazdów i posiada system śledzenia towarów — jest to system przeznaczony dla firm których procesy główne opierają się na wykorzystywaniu samochodów. Interfejs użytkownika jest przestarzały i mało intuicyjny, ponadto często nie pozwala na wycofanie wprowadzonych zmian, przez co wymaga czujności od użytkownika. Tak jak *Fleetly*, *Vinitysoft Fleet Manager* nie może zostać zintegrowany z zasobami firmy.



Rysunek 2.2 Interfejs użytkownika programu Vinitysoft Fleet Management Software 4.0

# Rozdział 3

## Wymagania funkcjonalne i niefunkcjonalne

### 3.1 Wymagania funkcjonalne

System powinien pozwalać na:

- tworzenie nowych rezerwacji,
- kontrolę dostępu do pojazdów — utworzone rezerwacje muszą być zaakceptowane przez kierowników,
- dodawanie, modyfikowanie oraz usuwanie informacji związanych z modelami pojazdów, pojazdami, ubezpieczeniami i naprawami,
- przechowywanie danych związanych z kosztami generowanymi przez flotę,
- monitorowanie akcji użytkowników (przechowywanie informacji o użytkownikach dokonujących modyfikacji danych).

Ponadto, system powinien korzystać z istniejących zasobów firmy (np. baz danych z danymi pracowników) by zredukować duplikację danych.

### 3.2 Wymagania niefunkcjonalne

#### 3.2.1 Interfejs użytkownika

Wymagania dotyczące wyglądu aplikacji są następujące:

- 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 powinna być wygodna w użyciu na ekranach komputerów o rozdzielczości HD (1366x768 pikseli) lub większej.

### 3.2.2 Interfejs programistyczny

Wymagania dotyczące interfejsu programistycznego są następujące:

- 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

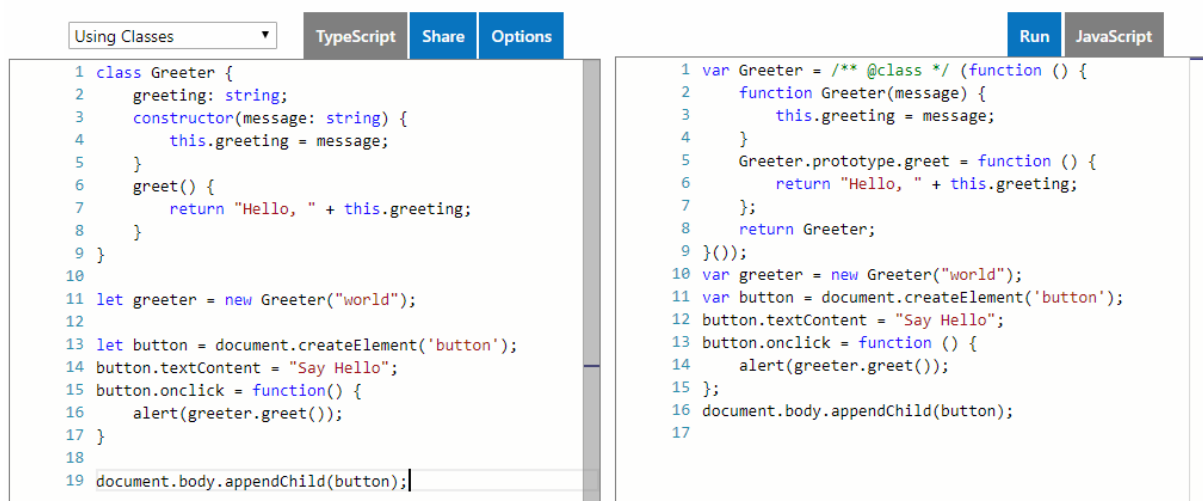
## Zastosowane technologie i narzędzia

### 4.1 Zastosowane technologie

Technologie wykorzystane w projekcie na chwilę obecną należą do czołówki platform dla aplikacji webowych.

#### 4.1.1 Angular

Interfejs użytkownika wykorzystuje platformę *Angular 7*. Podstawowymi elementami w *Angular* są komponenty [1], każdy z nich złożony z: pliku klasy *TypeScript* zawierającej logikę, wzorca *html* 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* [6], 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. Proces kompilacji pozwala na usunięcie wielu błędów, które w przypadku *JavaScript* zostałyby zauważone dopiero po uruchomieniu aplikacji.



The image shows a web-based TypeScript compiler interface. On the left, the 'TypeScript' tab is active, displaying the following code:

```
1 class Greeter {
2   greeting: string;
3   constructor(message: string) {
4     this.greeting = message;
5   }
6   greet() {
7     return "Hello, " + this.greeting;
8   }
9 }
10
11 let greeter = new Greeter("world");
12
13 let button = document.createElement('button');
14 button.textContent = "Say Hello";
15 button.onclick = function() {
16   alert(greeter.greet());
17 }
18
19 document.body.appendChild(button);
```

On the right, the 'JavaScript' tab is active, showing the compiled output:

```
1 var Greeter = /** @class */ (function () {
2   function Greeter(message) {
3     this.greeting = message;
4   }
5   Greeter.prototype.greet = function () {
6     return "Hello, " + this.greeting;
7   };
8   return Greeter;
9 }());
10 var greeter = new Greeter("world");
11 var button = document.createElement('button');
12 button.textContent = "Say Hello";
13 button.onclick = function () {
14   alert(greeter.greet());
15 };
16 document.body.appendChild(button);
17
```

Rysunek 4.1 Przykład kompilacji kodu TypeScript do JavaScript  
(<http://www.typescriptlang.org>)

### 4.1.2 Bootstrap

Jednym z ważniejszych komponentów aplikacji jest *Bootstrap* — biblioteka interfejsu użytkownika pozwalająca w prosty sposób tworzyć estetyczne strony internetowe. Poza wyglądem, *Bootstrap* oferuje również wiele elementów *UI* o rozszerzonej funkcjonalności w stosunku do zwykłych odpowiedników.

### 4.1.3 ASP.NET Core i EF Core

Interfejs programistyczny oparty został na technologii *ASP.NET Core 2.1* — jest to nowoczesna platforma oferująca działanie na wielu systemach operacyjnych oraz większą 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* [7], framework *ORM* pozwalający na konwersję między 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.

### 4.2.1 Git

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.

### 4.2.2 Visual Studio

Do rozwoju interfejsu programistycznego wykorzystano *Visual Studio 2017*, flagowy produkt dla programistów od firmy *Microsoft*. *VS* pozwala na łatwe debugowanie kodu oraz analizę aspektów takich jak wykorzystanie zasobów przez program. Zaawansowany mechanizm podpowiedzi umożliwia sprawną pracę bez dokumentacji. *Visual Studio* zostało wzbogacone o narzędzie *JetBrains ReSharper* automatycznie formatujące pliki projektu według zadanego wzorca, zapewniając spójność i przejrzystość kodu. *ReSharper* pozwala również na łatwiejsze uruchamianie i analizę testów.

### 4.2.3 Visual Studio Code

Aplikacja klienta była rozwijana przy użyciu *Visual Studio Code 1.28*, nowoczesnego edytora firmy *Microsoft*, 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*.

#### 4.2.4 SQL Server Management Studio

*SQL Server Management Studio*, kolejny produkt firmy *Microsoft*, to program który wspomagał bezpośrednią pracę z bazą danych. *Management Studio* to bardzo potężne narzędzie z bogatą funkcjonalnością przydatną przy rozwoju i zarządzaniu bazami danych. Program był wykorzystywany głównie do sprawdzania poprawności encji, tabel oraz relacji.

#### 4.2.5 Postman

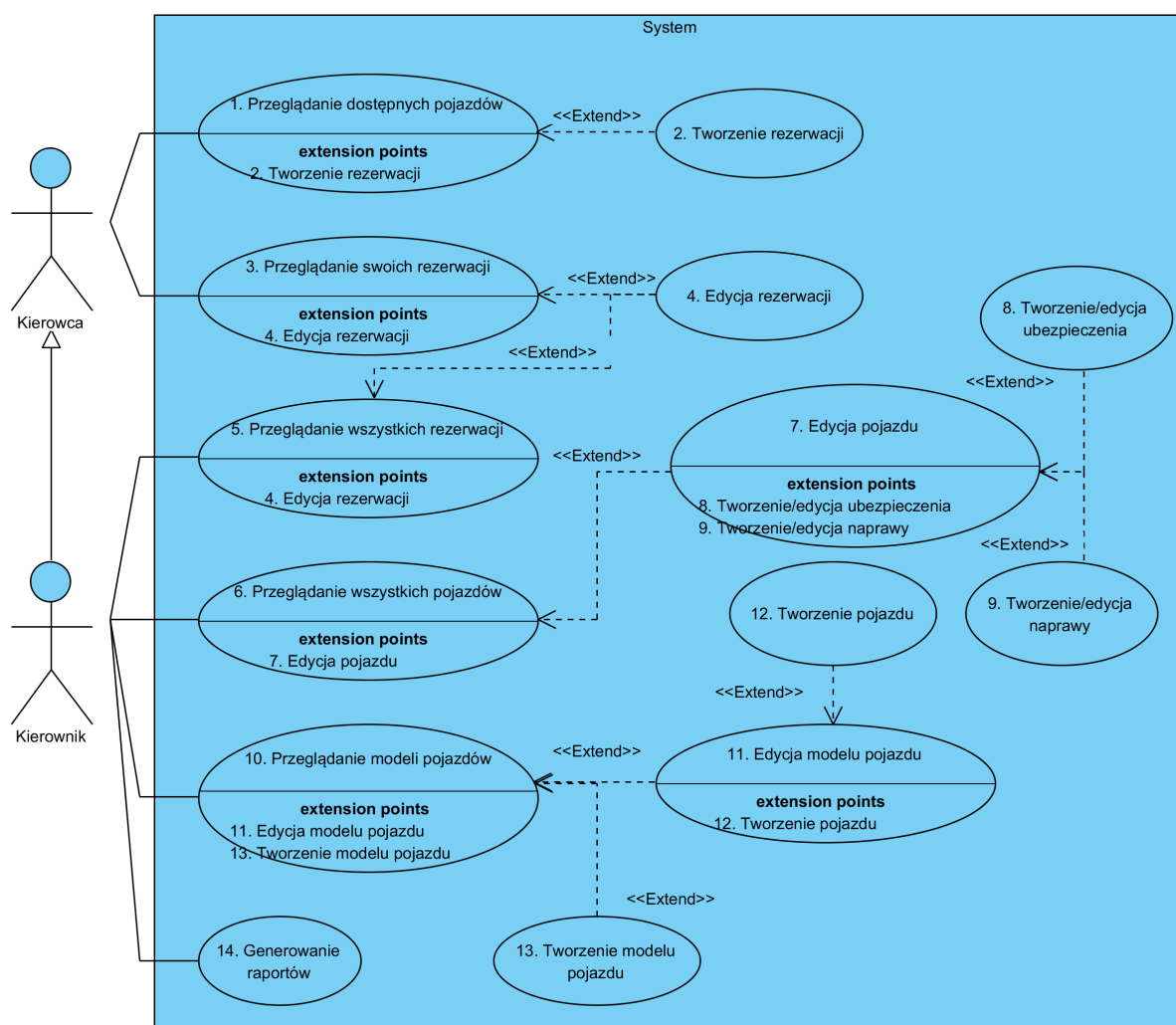
Interfejs programistyczny testowany był przy pomocy *Postman 6.5.2*, aplikacji pozwalającej na zaawansowane testowanie projektów *REST API*. *Postman* umożliwia wysyłanie oraz zarządzanie zapytaniami *HTTP*. Zaawansowany edytor żądań pozwala bez problemu umieścić w ciele tekst *JSON* lub przekazać informacje w formie parametrów *URL*. Utworzone żądania mogą być organizowane w kolekcje oraz przechowywane co znacznie ułatwia wielokrotne testowanie tej samej funkcjonalności. Informacje zwrócone przez *API* są automatycznie formatowane, co znacznie zwiększa ich czytelność.

# Rozdział 5

## Projekt i implementacja

### 5.1 Przypadki użycia

Na podstawie wymagań funkcjonalnych opracowany został diagram przypadków użycia oraz szczegółowy opis wymaganej funkcjonalności.



Rysunek 5.1 Diagram przypadków użycia

### 5.1.1 Szczegółowy opis przypadków użycia

Przypadki użycia zostały opisane według poniższego wzorca:

Numer	Numer PU
Nazwa	Krótką nazwa (widoczna na diagramie)
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 rezerwacji.
- 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 specyfikację techniczną wspólną dla pewnego zbioru pojazdów.
- **Pojazd** — model opisujący informacje unikatowe dla pewnego przedstawiciela zbioru Modeli Pojazdów.

Tablica 5.1 Przypadek użycia: Przeglądanie dostępnych pojazdów

Numer	1
Nazwa	Przeglądanie dostępnych pojazdów
Opis	System powinien pozwalać przeglądać listę dostępnych (możliwych do zarezerwowania) pojazdów.
Aktor	Kierowca
Kryterium spełnienia	Kierowca może wyświetlić listę wszystkich dostępnych pojazdów oraz filtrować wyniki.
Ograniczenia	Brak

Tablica 5.2 Przypadek użycia: Tworzenie rezerwacji

Numer	2
Nazwa	Tworzenie rezerwacji
Opis	System powinien umożliwiać rezerwowanie dostępnych pojazdów, wyświetlonych w ramach <b>PU #2</b> , wymagając wprowadzenia niezbędnych informacji jak okres i potrzeba stojąca za rezerwacją.
Aktor	Kierowca
Kryterium spełnienia	Kierowca może utworzyć rezerwację.
Ograniczenia	Kierowca nie może utworzyć rezerwacji dla innego użytkownika.

Tablica 5.3 Przypadek użycia: Przeglądanie swoich rezerwacji

Numer	3
Nazwa	Przeglądanie swoich rezerwacji
Opis	System powinien pozwalać przeglądać listę rezerwacji utworzonych przez obecnie zalogowanego użytkownika.
Aktor	Kierowca
Kryterium spełnienia	Kierowca może wyświetlić listę rezerwacji które utworzył.
Ograniczenia	Brak

Tablica 5.4 Przypadek użycia: Edycja rezerwacji

Numer	4
Nazwa	Edycja rezerwacji
Opis	System powinien pozwalać edytować rezerwacje. Edycja pozwala na zmianę określonych pól w zależności od obecnego stanu i poziomu uprawnień zalogowanego użytkownika.
Aktor	Kierowca, Kierownik
Kryterium spełnienia	Kierowca może wprowadzić podstawowe informacje dotyczące rezerwacji oraz przesłać ją do oceny kierownika. Po oddaniu samochodu kierowca może wpisać przejechane kilometry, zużyte paliwo oraz całkowitego koszt — po uzupełnieniu tych informacji rezerwacja może zostać oznaczona jako zakończona. Kierownik może zaakceptować lub odrzucić rezerwacje przesłane przez Kierowców. Kierownik ma również możliwość edycji większości pól, by mógł naprawić ew. błędy.
Ograniczenia	Jeżeli osobą tworzącą rezerwację jest użytkownik z uprawnieniami Kierownika, w ramach tego wypożyczenia może on podejmować wyłącznie działania wchodzące w rolę Kierowcy (np. nie może zaakceptować swojej rezerwacji).

Tablica 5.5 Przypadek użycia: Przeglądanie wszystkich rezerwacji

Numer	5
Nazwa	Przeglądanie wszystkich rezerwacji
Opis	System powinien pozwalać przeglądać listę wszystkich rezerwacji.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może wyświetlić listę wszystkich rezerwacji (niezależnie od stanu oraz użytkownika który rezerwację utworzył) oraz filtrować wyniki.
Ograniczenia	Brak

Tablica 5.6 Przypadek użycia: Przeglądanie wszystkich pojazdów

Numer	6
Nazwa	Przeglądanie wszystkich pojazdów
Opis	System powinien pozwalać przeglądać listę pojazdów.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może wyświetlić listę pojazdów.
Ograniczenia	Brak

Tablica 5.7 Przypadek użycia: Edycja pojazdów

Numer	7
Nazwa	Edycja pojazdów
Opis	System powinien umożliwiać edycję pojazdów oraz wyświetlać powiązane informacje (ubezpieczenia i naprawy).
Aktor	Kierownik
Kryterium spełnienia	Kierownik może wyświetlić szczegółowe informacje na temat pojazdu oraz dokonać ich zmian. Kierownik może również przeglądać ubezpieczenia i naprawy powiązane z pojazdem.
Ograniczenia	Edycja pojazdu jest niemożliwa jeżeli pojazd jest zarezerwowany. Pojazd nie może zostać usunięty jeżeli był kiedykolwiek rezerwowany.

Tablica 5.8 Przypadek użycia: Tworzenie/edycja ubezpieczenia.

Numer	8
Nazwa	Tworzenie/edycja ubezpieczenia
Opis	System powinien umożliwiać tworzenie oraz późniejszą edycję ubezpieczeń.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może wprowadzić informacje związane z ubezpieczeniem pojazdu oraz edytować wcześniej utworzone ubezpieczenia.
Ograniczenia	Tworzenie/edycja ubezpieczeń nie jest możliwa dla pojazdu który jest zarezerwowany.

Tablica 5.9 Przypadek użycia: Tworzenie/edycja naprawy

Numer	9
Nazwa	Tworzenie/edycja naprawy
Opis	System powinien umożliwiać tworzenie oraz późniejszą edycję napraw (zdarzeń serwisowych).
Aktor	Kierownik
Kryterium spełnienia	Kierownik może wprowadzić informacje związane z naprawą pojazdu oraz edytować wcześniej utworzone naprawy.
Ograniczenia	Tworzenie/edycja napraw nie jest możliwa dla pojazdu który jest zarezerwowany.

Tablica 5.10 Przypadek użycia: Przeglądanie modeli pojazdów

Numer	10
Nazwa	Przeglądanie modeli pojazdów
Opis	System powinien pozwalać przeglądać listę modeli pojazdów.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może przeglądać modele pojazdów.
Ograniczenia	Brak

Tablica 5.11 Przypadek użycia: Edycja modelu pojazdu

Numer	11
Nazwa	Edycja modelu pojazdu
Opis	System powinien pozwalać przeglądać listę modeli pojazdów.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może dodawać modele pojazdów oraz edytować wcześniej utworzone modele.
Ograniczenia	Model pojazdu nie może zostać usunięty jeżeli system posiada egzemplarze danego modelu.

Tablica 5.12 Przypadek użycia: Tworzenie pojazdu

Numer	12
Nazwa	Tworzenie pojazdu
Opis	System powinien pozwalać na wprowadzanie informacji o pojazdach będących egzemplarzami wcześniej dodanych modeli.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może dodać nowy egzemplarz pojazdu
Ograniczenia	Brak

Tablica 5.13 Przypadek użycia: Tworzenie modelu pojazdu

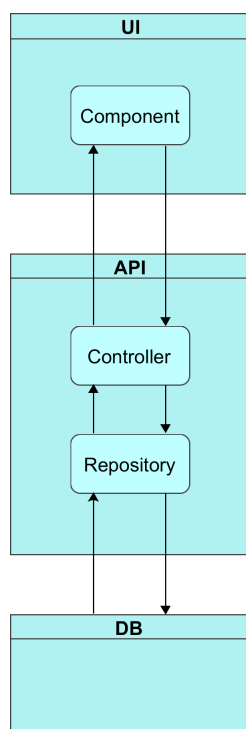
Numer	13
Nazwa	Tworzenie modelu pojazdu
Opis	System powinien pozwalać na tworzenie modeli pojazdów.
Aktor	Kierownik
Kryterium spełnienia	Kierownik może wprowadzać nowe modele pojazdów do systemu.
Ograniczenia	Brak

Tablica 5.14 Przypadek użycia: Generowanie raportów

Numer	14
Nazwa	Generowanie raportów
Opis	System powinien umożliwiać eksportowanie kosztów generowanych przez flotę. Format raportu powinien być kompatybilny z programem <i>Microsoft Excel</i> .
Aktor	Kierownik
Kryterium spełnienia	System generuje raporty, z podziałem na rodzaj (raport dotyczący pojazdów, wypożyczeń itd.), Raporty mogą zostać zaimportowane do programu <i>Microsoft Excel</i> ; dane nie mogą wymagać skomplikowanych akcji ze strony użytkownika w celu utworzenia tabeli.
Ograniczenia	Brak

## 5.2 Architektura

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



Rysunek 5.2 Uproszczony schemat architektury z wyodrębnionymi najważniejszymi elementami składowymi

System został zaprojektowany tak, by mógł zostać zintegrowany z istniejącymi zasobami firmy. Baza danych z której korzysta system przechowuje wyłącznie informacje związane z logiką biznesową — 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 zwią-

zane 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* [10].
- Repozytorium (*Repository*) zawiera logikę pośredniczącą w komunikacji między *API* a bazą danych.
- Komponent (*Component*) to podstawowy element definiujący działanie widoku w *Angular* [1].

### 5.3 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 [5] (*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.

Tablica 5.15 Najważniejsze konwencje nazewnnicze

Język	Typy	Pliki	Zmienne prywatne	Inne zmienne
C# [9]	PascalCase	PascalCase.cs	camelCase	PascalCase
TypeScript [2]	PascalCase	snake-case.typ.ts	camelCase	camelCase



## 5.4 Interfejs programistyczny

### 5.4.1 Logika biznesowa

Implementacja projektu rozpoczęła się od utworzenia klas opisujących świat biznesowy.

Tablica 5.16 Klasy obiektów biznesowych

Klasa	Opis
VehicleModel	Specyfikacja techniczna wspólna dla wielu pojazdów
Vehicle	Informacje unikatowe dla pewnego pojazdu
Insurance	Ubezpieczenie
Maintenance	Naprawa, serwis pojazdu
Employee	Klasa używana do powiązania logiki biznesowej z informacjami o pracowniku
EmployeeIdentity	Dane personalne użytkownika; mogą być pobierane z innej bazy danych
Booking	Rezerwacja pojazdu
Location	Informacje o budynkach należących do firmy korzystającej z systemu; mogą być pobierane z innej bazy danych

Wszystkie klasy związane z logiką biznesową dziedziczą po klasie *AuditableEntity* (listing 5.1) posiadającej pola przechowujące informacje (data i nazwa użytkownika) o utworzeniu i ostatniej edycji encji. Klasy związane z elementami generującymi koszty (pojazdami, modelami pojazdów, rezerwacjami i użytkownikami) dziedziczą po klasie *CostGeneratingEntity* (listing 5.2) przechowującej informacje o koszcie, zużytych paliwie i przejechanych kilometrach.

Listing 5.1 Klasa abstrakcyjna AuditableEntity

```
public abstract class AuditableEntity
{
    [Required]
    public DateTime AddedOn { get; set; }

    [Required]
    public string AddedBy { get; set; }

    public DateTime? ModifiedOn { get; set; }

    public string ModifiedBy { get; set; }
}
```

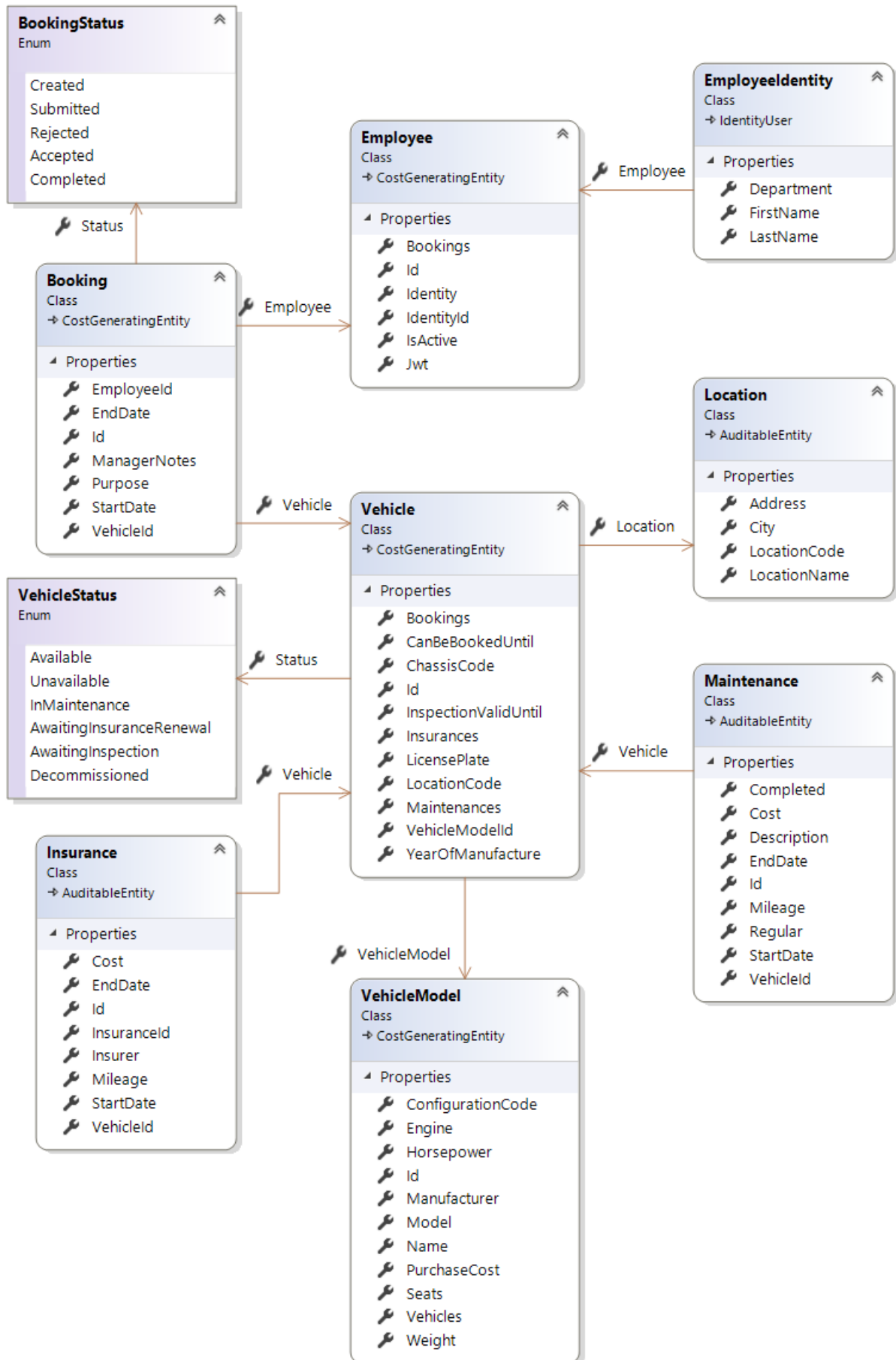
Listing 5.2 Klasa abstrakcyjna CostGeneratingEntity

```
public abstract class CostGeneratingEntity : AuditableEntity
{
    [Required]
    public int Mileage { get; set; }

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

    [Required]
    [Column(TableName = "decimal(16, 2)")]
    public decimal Cost { get; set; }
}
```

Klasy były projektowane z myślą o późniejszym użyciu konwencji *code-first* przy tworzeniu bazy danych (proces migracji został opisany w rozdziale 5.4.2). Diagram klas (wygenerowany w *Visual Studio*) przedstawia rysunek 5.3.



Rysunek 5.3 Diagram klas

### 5.4.2 Baza danych

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

Do zdefiniowania relacji między tabelami należało użyć pól typu takiego samego jak klucz główny docelowej klasy[8]. 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 (*ang. 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 5.3 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; }
}

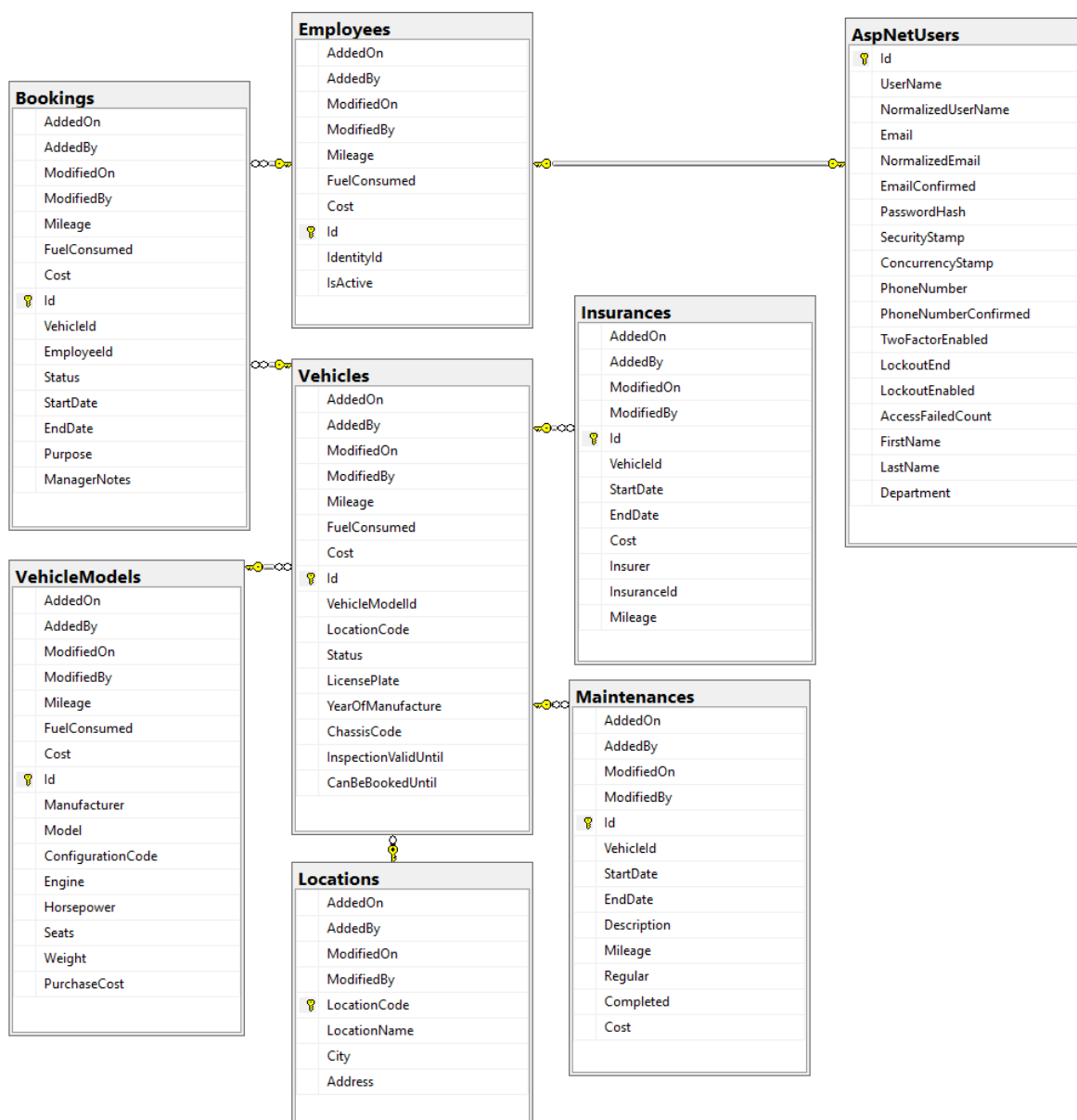
public class Vehicle
{
    [Key]
    public int Id { get; set; }

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

Listing 5.4 Ładowanie powiązanych obiektów na przykładzie relacji rezerwacji do pojazdu

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

Wybrana strategia dziedziczenia to *TPC* — *Table per Concrete Type*. W strategii *TPC* tabele utworzone w bazie danych zawierają wszystkie kolumny odpowiadające polom wszystkich klas w hierarchii dziedziczenia.



Rysunek 5.4 Diagram bazy danych

Jak widać na rysunku 5.4 (wygenerowanym przy użyciu *SQL Server Management Studio*) właściwości oraz relacje tabel odzwierciedlają zależności między klasami (diagram klas, rysunek 5.3).

Tabela *AspNetUsers* zawiera informacje związane z pracownikiem firmy. *AspNetUsers* odpowiada klasie *EmployeeIdentity*, dziedziczącej po klasie należącej do standardowych klas wchodzących w platformę autoryzacji *ASP.NET Core* — *IdentityUser*.

### 5.4.3 Struktura rozwiązania

Kod interfejsu programistycznego znajduje się w jednej rozwiązaniu podzielonej na poniższe projekty:

- **Vehifleet.API** — konfiguracja systemu oraz kontrolery; projekt startowy,
- **Vehifleet.API.QueryFilters** — filtry używane w żądaniach GET,
- **Vehifleet.Data.DbAccess** — konfiguracja połączenia z bazą danych,
- **Vehifleet.Data.Dtos** — modele transportowe (*ang. Data Transfer Objects*) używane w komunikacji z interfejsem użytkownika,
- **Vehifleet.Data.Models** — modele używane wewnątrz interfejsu programistycznego oraz przy tworzeniu bazy danych,
- **Vehifleet.Helper** — pomocnicze metody rozszerzające [4],
- **Vehifleet.Repositories** — repozytoria odpowiedzialne za interakcje z bazą danych,
- **Vehifleet.Services.UserService** — obsługa logowania użytkowników,
- **Vehifleet.Services.CsvService** — serwis generujący raporty ze statystykami.

### 5.4.4 Wstrzykiwanie zależności

Obiekty klas z logiką są tworzone przy użyciu wstrzykiwania zależności (*ang. dependency injection*). Większość obiektów jest tworzona dla konkretnego żądania odebranego przez kontroler — po wykonaniu operacji i zwróceniu odpowiedzi obiekt trafia do puli oczekującej na *GC*.

### 5.4.5 Bezpieczeństwo

Dostęp do systemu został zabezpieczony przy użyciu standardu *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* wymaga wcześniej wygenerowanego 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; kod *HTTP: 401 Unauthorized*.

[illegible]Rysunek 5.5 Przykładowy token *JWT*

Token został wygenerowany przy użyciu narzędzia ze strony <https://jwt.io/>.

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.

Tablica 5.17 Domyślna konfiguracja relacji ról do poziomu uprawnień

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

Zarówno interfejs użytkownika jak i programistyczny utworzone w ramach projektu posiadają mechanizmy obrony przed nieautoryzowanym dostępem — *API* zwraca błąd *401 (Unauthorized)* dla żądań bez poprawnego tokena, a aplikacja użytkownika blokuje dostęp do widoków przy użyciu mechanizmu *RouteGuard*.

### 5.4.6 Kontrolery

Jednymi z najważniejszych klas są kontrolery, komponenty obsługujące żądania *HTTP*. Logika z różnych projektów jest łączona i używana w klasach kontrolerów. Poniżej zamieszczony został opis wszystkich punktów końcowych (*ang. endpoint*) w interfejsie programistycznym.

Tablica 5.18 Endpoint *api/vehicle-models GET*

Opis		
URL	api/vehicle-models	
Wymagane role	Employee	
Metoda	GET	
Odpowiedzi		
200 (OK)	Zawartość	Tablica <i>JSON</i>
	Opis	Lista modeli pojazdów

Tablica 5.19 Endpoint *api/vehicle-models/manufacturers GET*

Opis		
URL	api/vehicle-models/manufacturers	
Wymagane role	Employee	
Metoda	GET	
Odpowiedzi		
200 (OK)	Zawartość	Tablica <i>JSON</i>
	Opis	Lista marek pojazdów



Tablica 5.20 Endpoint *api/vehicle-models/id GET*

Opis		
URL	api/vehicle-models/id	
Wymagane role	Employee	
Metoda	GET	
Odpowiedzi		
200 (OK)	Zawartość	JSON
	Opis	Model pojazdu o żądanym <i>id</i>
404 (Not Found)	Zawartość	"No_such_vehicle_model"
	Opis	Model pojazdu o żądanym <i>id</i> nie istnieje

Tablica 5.21 Endpoint *api/vehicle-models POST*

Opis		
URL	api/vehicle-models	
Wymagane role	Employee	
Metoda	POST	
Odpowiedzi		
200 (OK)	Zawartość	Id utworzonego modelu pojazdu
	Opis	Model pojazdu został utworzony

Tablica 5.22 Endpoint *api/vehicle-models/id PUT*

Opis		
URL	api/vehicle-models/id	
Wymagane role	Manager, Administrator	
Metoda	PUT	
Odpowiedzi		
200 (OK)	Opis	Model pojazdu został zaktualizowany
404 (Not Found)	Zawartość	"No_such_vehicle_model."
	Opis	Model pojazdu o żądanym <i>id</i> nie istnieje

Tablica 5.23 Endpoint *api/vehicle-models/id DELETE*

Opis		
URL	api/vehicle-models/id	
Wymagane role	Manager, Administrator	
Metoda	DELETE	
Odpowiedzi		
200 (OK)	Opis	Model pojazdu został usunięty
404 (Not Found)	Zawartość	"No_such_vehicle_model."
	Opis	Model pojazdu o żądanym <i>id</i> nie istnieje
400 (Bad Request)	Zawartość	"Vehicle_model_has_vehicles."
	Opis	System posiada egzemplarze modelu pojazdu o żądanym <i>id</i> , nie może on zostać usunięty

Tablica 5.24 Endpoint *api/vehicles GET*

Opis		
URL	api/vehicles	
Wymagane role	Employee	
Metoda	GET	
Odpowiedzi		
200 (OK)	Zawartość	Tablica <i>JSON</i>
	Opis	Lista pojazdów

Tablica 5.25 Endpoint *api/vehicles/id GET*

Opis		
URL	api/vehicles/id	
Wymagane role	Employee	
Metoda	GET	
Odpowiedzi		
200 (OK)	Zawartość	JSON
	Opis	Pojazd o żądanym <i>id</i>
404 (Not Found)	Zawartość	"No_such_vehicle."
	Opis	Pojazd o żądanym <i>id</i> nie istnieje

Tablica 5.26 Endpoint *api/vehicles POST*

Opis		
URL	api/vehicles	
Wymagane role	Manager, Administrator	
Metoda	POST	
Odpowiedzi		
200 (OK)	Zawartość	Id utworzonego pojazdu
	Opis	Pojazd został utworzony

Tablica 5.27 Endpoint *api/vehicles/id PUT*

Opis		
URL	api/vehicles/id	
Wymagane role	Manager, Administrator	
Metoda	PUT	
Odpowiedzi		
200 (OK)	Opis	Pojazd zostało zaktualizowane
404 (Not Found)	Zawartość	"No_such_vehicle."
	Opis	Pojazd o żądanym <i>id</i> nie istnieje

Tablica 5.28 Endpoint *api/vehicles/id DELETE*

Opis		
URL	api/vehicles/id	
Wymagane role	Manager, Administrator	
Metoda	DELETE	
Odpowiedzi		
200 (OK)	Opis	Pojazd został usunięty
404 (Not Found)	Zawartość	"No_such_vehicle."
	Opis	Pojazd o żądanym <i>id</i> nie istnieje
400 (Bad Request)	Zawartość	"Vehicle_has_bookings."
	Opis	System posiada rezerwacje przypisane do pojazdu o żądanym <i>id</i> , nie może on zostać usunięty

Tablica 5.29 Endpoint *api/insurances/vehicle/id GET*

Opis		
URL	api/insurances/vehicle/id	
Wymagane role	Employee	
Metoda	GET	
Odpowiedzi		
200 (OK)	Zawartość	Tablica <i>JSON</i>
	Opis	Lista ubezpieczeń przypisanych do danego pojazdu

Tablica 5.30 Endpoint *api/insurances/id GET*

Opis		
URL	api/insurances/id	
Wymagane role	Employee	
Metoda	GET	
Odpowiedzi		
200 (OK)	Zawartość	JSON
	Opis	Ubezpieczenie o żądanym <i>id</i>
404 (Not Found)	Zawartość	"No_such_insurance."
	Opis	Ubezpieczenie o żądanym <i>id</i> nie istnieje

Tablica 5.31 Endpoint *api/insurances POST*

Opis		
URL	api/insurances	
Wymagane role	Manager, Administrator	
Metoda	POST	
Odpowiedzi		
200 (OK)	Zawartość	Id utworzonego ubezpieczenia
	Opis	Ubezpieczenie został utworzony

Tablica 5.32 Endpoint *api/insurances/id PUT*

Opis		
URL	api/insurances/id	
Wymagane role	Manager, Administrator	
Metoda	PUT	
Odpowiedzi		
200 (OK)	Opis	Ubezpieczenie zostało zaktualizowane
404 (Not Found)	Zawartość	"No_such_insurance."
	Opis	Ubezpieczenie o żądanym <i>id</i> nie istnieje

Tablica 5.33 Endpoint *api/insurances/id DELETE*

Opis		
URL	api/insurances/id	
Wymagane role	Manager, Administrator	
Metoda	DELETE	
Odpowiedzi		
200 (OK)	Opis	Ubezpieczenie został usunięty
404 (Not Found)	Zawartość	"No_such_insurance."
	Opis	Ubezpieczenie o żądanym <i>id</i> nie istnieje

Tablica 5.34 Endpoint *api/maintenances/vehicle/id GET*

Opis		
URL	api/maintenances/vehicle/id	
Wymagane role	Employee	
Metoda	GET	
Odpowiedzi		
200 (OK)	Zawartość	Tablica <i>JSON</i>
	Opis	Lista napraw przypisanych do danego pojazdu

Tablica 5.35 Endpoint *api/maintenances/id GET*

Opis		
URL	api/maintenances/id	
Wymagane role	Employee	
Metoda	GET	
Odpowiedzi		
200 (OK)	Zawartość	JSON
	Opis	Naprawa o żądanym <i>id</i>
404 (Not Found)	Zawartość	"No_such_maintenance."
	Opis	Naprawa o żądanym <i>id</i> nie istnieje

Tablica 5.36 Endpoint *api/maintenances POST*

Opis		
URL	api/maintenances	
Wymagane role	Manager, Administrator	
Metoda	POST	
Odpowiedzi		
200 (OK)	Zawartość	<i>Id</i> utworzonego ubezpieczenia
	Opis	Naprawa została utworzona

Tablica 5.37 Endpoint *api/maintenances/id PUT*

Opis		
URL	api/maintenances/id	
Wymagane role	Manager, Administrator	
Metoda	PUT	
Odpowiedzi		
200 (OK)	Opis	Naprawa została zauktualizowana
404 (Not Found)	Zawartość	"No_such_maintenance."
	Opis	Naprawa o żądanym <i>id</i> nie istnieje

Tablica 5.38 Endpoint *api/maintenances/id DELETE*

Opis		
URL	api/maintenances/id	
Wymagane role	Manager, Administrator	
Metoda	DELETE	
Odpowiedzi		
200 (OK)	Opis	Naprawa została usunięta
404 (Not Found)	Zawartość	"No_such_maintenance."
	Opis	Naprawa o żądanym <i>id</i> nie istnieje

Tablica 5.39 Endpoint *api/bookings GET*

Opis		
URL	api/bookings	
Wymagane role	Employee	
Metoda	GET	
Odpowiedzi		
200 (OK)	Zawartość	Tablica <i>JSON</i>
	Opis	Lista rezerwacji

Tablica 5.40 Endpoint *api/bookings/id GET*

Opis		
URL	api/bookings/id	
Wymagane role	Employee	
Metoda	GET	
Odpowiedzi		
200 (OK)	Zawartość	JSON
	Opis	Rezerwacja o żądanym <i>id</i>
404 (Not Found)	Zawartość	"No_such_booking."
	Opis	Rezerwacja o żądanym <i>id</i> nie istnieje

Tablica 5.41 Endpoint *api/bookings POST*

Opis		
URL	api/bookings	
Wymagane role	Employee	
Metoda	POST	
Odpowiedzi		
200 (OK)	Zawartość	Id utworzonej rezerwacji
	Opis	Rezerwacja została utworzona
400 (Bad Request)	Zawartość	"No_such_employee."
	Opis	Pracownik powiązany z rezerwacją nie istnieje
400 (Bad Request)	Zawartość	"No_such_vehicle."
	Opis	Pojazd powiązany z rezerwacją nie istnieje

Tablica 5.42 Endpoint *api/bookings/id PUT*

Opis		
URL	api/bookings/id	
Wymagane role	Employee	
Metoda	PUT	
Odpowiedzi		
200 (OK)	Opis	Rezerwacja została zauktualizowana
404 (Not Found)	Zawartość	"No_such_booking."
	Opis	Rezerwacja o żądanym <i>id</i> nie istnieje

Tablica 5.43 Endpoint *api/bookings/id DELETE*

Opis		
URL	api/bookings/id	
Wymagane role	Employee	
Metoda	DELETE	
Odpowiedzi		
200 (OK)	Opis	Rezerwacja została usunięta
404 (Not Found)	Zawartość	"No_such_booking."
	Opis	Rezerwacja o żądanym <i>id</i> nie istnieje



### 5.4.7 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 rezerwacji.

Listing 5.5 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
```

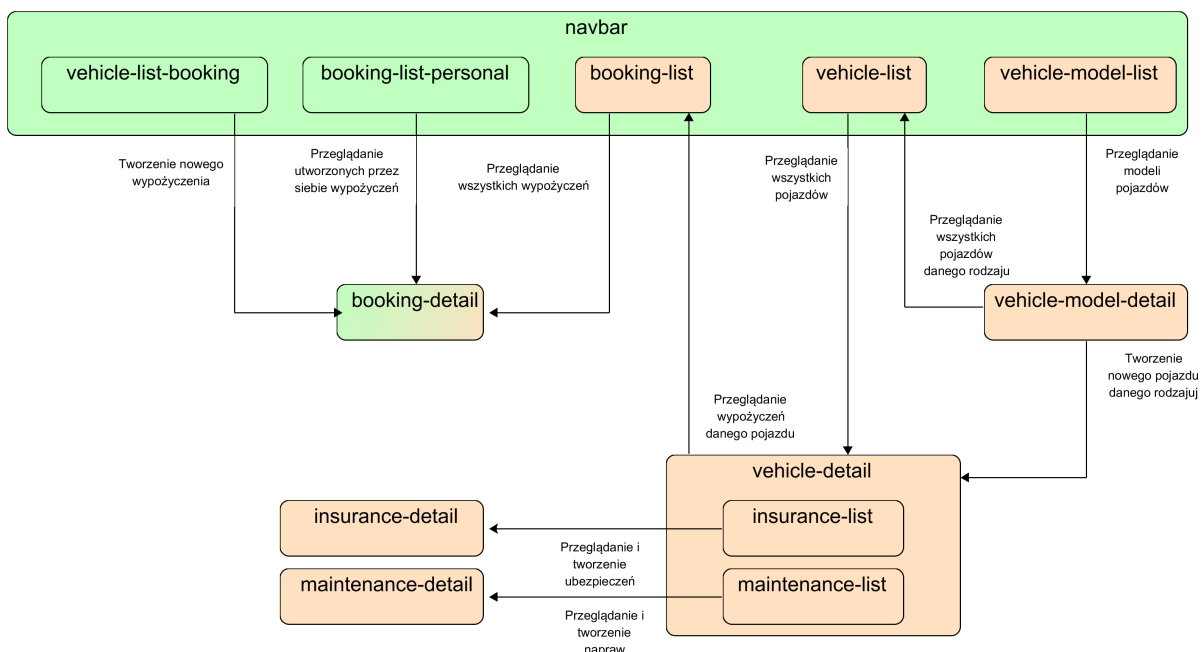
## 5.5 Interfejs użytkownika

### 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*).

Układ oraz przejścia między widokami zostały przedstawione na rysunku 5.6. 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 5.6 Widoki interfejsu użytkownika

## 5.6 Walidacja danych

Dane wprowadzane przez użytkownika są sprawdzane pod kątem poprawności. W przypadku pól tekstowych walidacja najczęściej weryfikuje obecność jakiegokolwiek tekstu — pola, które powinny zawierać liczby, są weryfikowane przy użyciu wyrażeń regularnych.

Listing 5.6 Przykładowy walidator używający wyrażenia regularnego

```
mileage: new FormControl('', [
    Validators.required,
    Validators.pattern('[0-9]*$')
])
```

W przypadku wykrycia błędnych danych, pod polem zawierającym niepoprawne dane pojawia się opis błędu oraz przycisk zatwierdzający zmiany zostaje zablokowany.

**Insurance INS-2018-4-10-1021**

**Insurer:**  
  
Insurer name is required.

**Insurance ID:**  
  
Insurance ID is required.

**Start date:**

**End date:**

**Cost (PLN):**  
  
Cost must be a valid number.

**Mileage (km):**  
  
Mileage must be a valid number.

**Save**

**Delete**

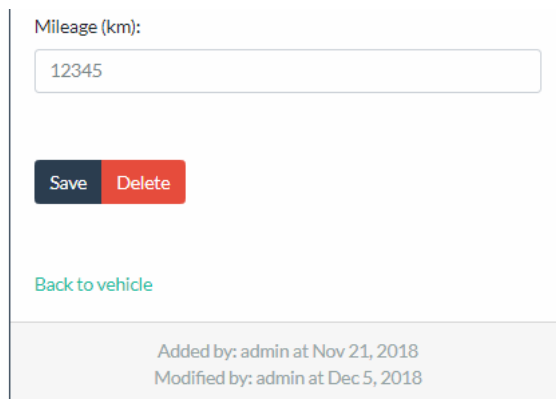
[Back to vehicle](#)

Added by: admin at Nov 21, 2018

Rysunek 5.7 Widok z polami zawierającymi błędne wartości

## 5.7 Stopka audytowa

Każdy widok szczegółowy korzysta z komponentu *AuditFooter* wyświetlającego nazwę użytkownika, który utworzył/modyfikował obiekt oraz datę kiedy akcja została wykonana.



Mileage (km):

12345

Save Delete

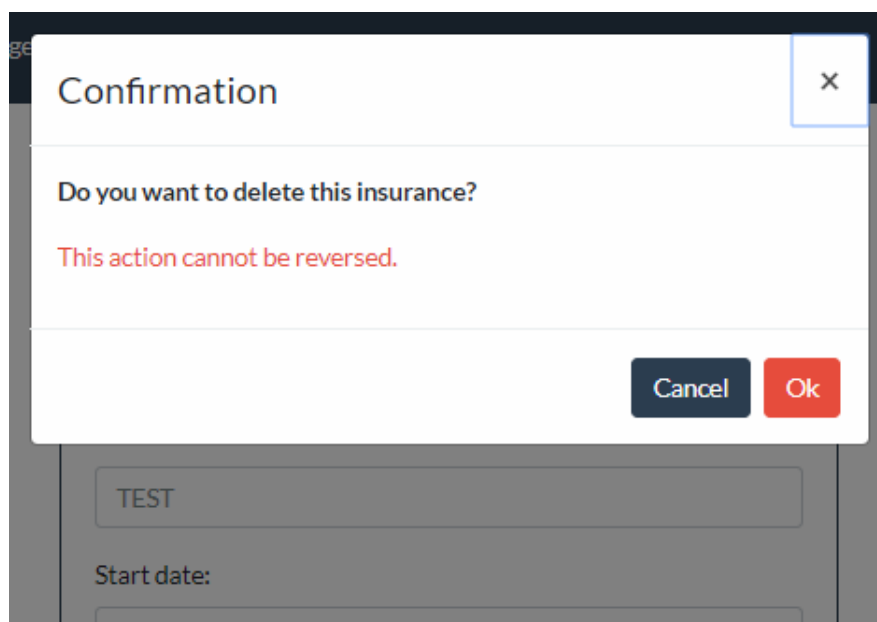
[Back to vehicle](#)

Added by: admin at Nov 21, 2018  
Modified by: admin at Dec 5, 2018

Rysunek 5.8 Stopka audytowa

## 5.8 Dialogi

Każda akcja która niesie za sobą znaczne (np. akceptacja rezerwacji) lub nieodwracalne (np. usunięcie obiektu) konsekwencje, musi zostać zatwierdzona przez użytkownika w dialogu który pojawia się po naciśnięciu przycisku zapisu/usuwania.



Rysunek 5.9 Przykładowy dialog

# 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

W testach klas wymagających wstrzykiwania zależności wykorzystano bibliotekę *Moq* do tworzenia atrap obiektów.

Przetestowane zostały m. in. kluczowe funkcjonalności kontrolerów oraz metody rozszerzeń.

Listing 6.1 Test jednostkowy metody rozszerzeń

```
[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);
}
```

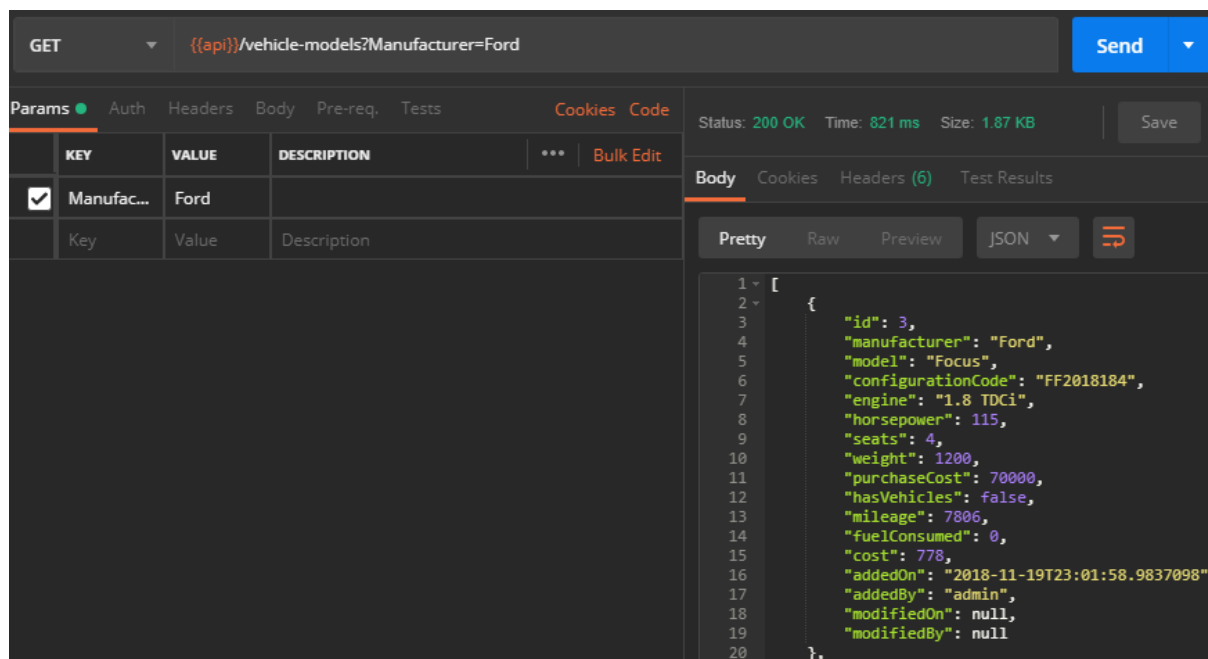
Listing 6.2 Test kontrolera korzystający z *Moq*

```
[Fact]
public async void Should_Return_Filtered_Vehicles()
{
    // Arrange
    var vehicles = DataProvider.GetVehicles();
    var vehiclesMock = vehicles.AsQueryable().BuildMock();
    var filter = new VehicleFilter
    {
        Manufacturer = "Ford"
    };
    var vehicleRepositoryMock
        = new Mock<IGenericRepository<Vehicle, int>>();
    vehicleRepositoryMock.Setup(m => m.Get())
        .Returns(vehiclesMock.Object);
    var vehicleController
        = new VehicleController(
            vehicleRepositoryMock.Object,
            Mock.Of<IGenericRepository<VehicleModel, int>>(),
            Mock.Of<IGenericRepository<Booking, int>>(),
            MapperProvider.GetMapper());
    // Act
    var request = await vehicleController.Get(filter);
    var result = request as OkObjectResult;

    // Assert
    Assert.NotNull(result);
    Assert.True(result.StatusCode == 200);
    var dtos = result.Value as List<VehicleListItemDto>;
    Assert.NotNull(dtos);
    Assert.True(dtos.Count == 2);
}
```

## 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 przyspiesza proces testowania.



Rysunek 6.1 Interfejs programu Postman

## 6.3 Testy dymne

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

Najważniejsze testy które zostały wykonane:

- Utworzenie rezerwacji i przejście wszystkich stanów (przy użyciu dwóch kont testowych o różnych uprawnieniach),
- Tworzenie oraz edycja modeli pojazdów, pojazdów, ubezpieczeń i napraw,
- Wprowadzanie niepoprawnych danych (we wszystkich widokach) w celu przetestowania walidacji.

# Rozdział 7

## Podsumowanie

### 7.1 Wnioski

Celem pracy było utworzenie systemu pozwalającego na kontrolę dostępu do pojazdów oraz śledzeniu ich stanu — cel ten został spełniony. System został napisany w sposób zgodny ze standardami, co pozwala na łatwiejsze utrzymanie (w tym dalszą rozbudowę). Zastosowanie nowoczesnych technologii takich jak framework *ASP.NET Core 2.1* pozwala na łatwe rozwijanie aplikacji przy użyciu języka *C#*, dodatkowo oferując wiele zalet takich jak multiplatformowość i zwiększoną wydajność względem tradycyjnego *ASP.NET Framework*. Użycie aplikacji webowej jako interfejsu użytkownika pozwala na dostęp do systemu bez potrzeby instalacji aplikacji klienckiej. Aplikacja webowa eliminuje również typowe problemy związane z utrzymywaniem systemów, takie jak aktualizacje do nowych wersji i zmniejsza koszt utrzymania całego systemu.

### 7.2 Możliwości rozwoju

System może być rozwinięty na wiele sposobów; kilka z nich:

- automatyczna generacja raportów o kosztach (np. co miesiąc) oraz wprowadzenie serwisu wysyłającego raport e-mailem do użytkowników,
- wyświetlanie statystyk w formie graficznej w interfejsie użytkownika,
- przystosowanie aplikacji to użycia na urządzeniach mobilnych,
- udostępnienie interfejsu programistycznego innym systemom — przykładowo system zbierający koszty generowane przez dany dział w firmie mógłby sprawdzać wydatki pracownika wiążące się z rezerwowaniem pojazdów,
- konteneryzacja aplikacji,
- przechowywanie pełnej historii edycji oraz generowanie raportów audytowych w formacie zgodnym z *Microsoft Excel*.



# Literatura

- [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. *Extension methods*, 2015. URL <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/extension-methods>. Dostęp 05.12.2018.
- [5] MSDN. *Generics*, 2015. URL <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/>. Dostęp 03.12.2018.
- [6] MSDN. *TypeScript - Understanding TypeScript*, 2015. URL <https://msdn.microsoft.com/en-us/magazine/dn890374.aspx>. Dostęp 03.12.2018.
- [7] MSDN. *Entity Framework Core*, 2016. URL <https://docs.microsoft.com/en-us/ef/core>. Dostęp 03.12.2018.
- [8] MSDN. *Entity Framework Core: Relationships*, 2016. URL <https://docs.microsoft.com/en-us/ef/core/modeling/relationships>. Dostęp 03.12.2018.
- [9] MSDN. *General Naming Conventions*, 2017. URL <https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/general-naming-conventions>. Dostęp 03.12.2018.
- [10] 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.

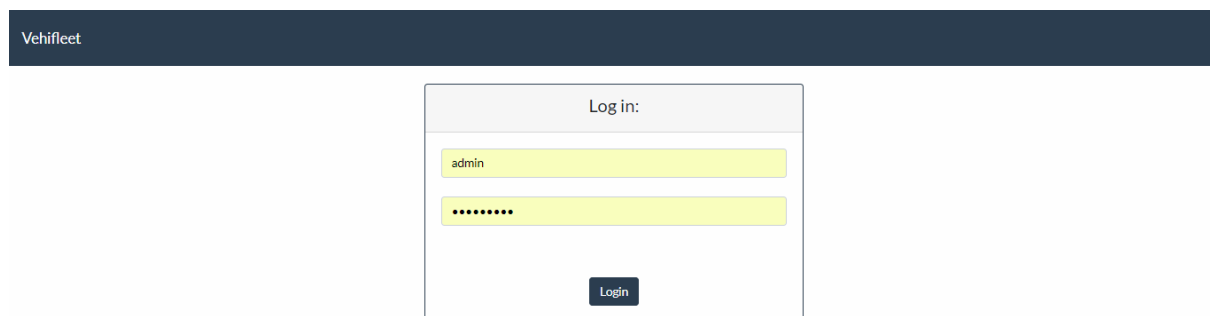
# Dodatek A

## Instrukcja użytkownika

Instrukcja użytkownika opisująca pokrótce każdy widok została zamieszczona poniżej. Instrukcja była tworzona przy użyciu konta z pełnymi uprawnieniami — użytkownik posiadający wyłącznie uprawnienia kierowcy ma do dyspozycji tylko widoki opisane w punktach 1-4 oraz 6.

### A.1 Ekran logowania

Jedyny widok dostępny dla niezalogowanego użytkownika. Aplikacja uniemożliwia dostęp do wszystkich innych widoków osobom niezalogowanym. Przy ręcznej zmianie adresu w przeglądarce nieupoważniony użytkownik zostanie przekierowany do tego widoku.



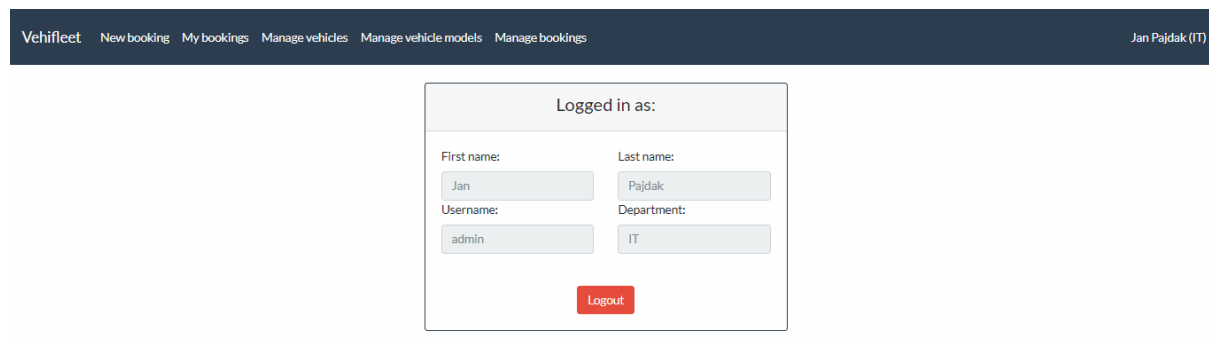
The screenshot shows a web application interface. At the top, there is a dark blue header bar with the text "Vehifleet" in white. Below the header, centered on the page, is a white login form with a thin grey border. The form has a title "Log in:" at the top. It contains two input fields: the first is labeled "admin" and the second is filled with dots, representing a password. Below these fields is a dark blue button with the text "Login" in white.

Rysunek A.1 Widok logowania (dashboard-login)

Sesja użytkownika jest przechowywana w pamięci lokalnej przeglądarki do czasu wygaśnięcia tokenu (120 minut).

## A.2 Ekran wylogowywania

Podstawowy widok zalogowanego użytkownika z informacjami o użytkowniku i możliwością wylogowania się.



The screenshot displays a web application interface. At the top, a dark blue navigation bar contains the following links: 'Vehifleet', 'New booking', 'My bookings', 'Manage vehicles', 'Manage vehicle models', and 'Manage bookings'. On the right side of this bar, the user's identity is shown as 'Jan Pajdak (IT)'. Below the navigation bar, the main content area features a light gray box with a white background. This box is titled 'Logged in as:' and contains four input fields arranged in a 2x2 grid. The first row shows 'First name:' with the value 'Jan' and 'Last name:' with the value 'Pajdak'. The second row shows 'Username:' with the value 'admin' and 'Department:' with the value 'IT'. At the bottom center of this box is a red button labeled 'Logout'.

Rysunek A.2 Widok zalogowanego użytkownika (dashboard-user-details)

## A.3 Dostępne pojazdy

Widok pozwalający na przeglądanie oraz rezerwowanie dostępnych pojazdów, otwierana z poziomu paska nawigacyjnego, za pomocą przycisku (*New booking*). Lista pojazdów może być dodatkowo filtrowana. Po kliknięciu na jakikolwiek pojazd, jego dokładne informacje zostają wczytane i wyświetlone w oknie po prawej stronie; użytkownik może utworzyć nową rezerwację używając przycisku *Book*.

VehifleetNew bookingMy bookingsManage vehiclesManage vehicle modelsManage bookingsJan Pajdak (IT)

ManufacturerLocation codeMin. booking daysFilterReset

Manufacturer	Model	YOM	Seats	Horsepower	Location	Available until
Ford	Focus	2016	4	115	KRK-1	Jul 7, 2019
Ford	C-Max	2016	5	105	WRO-1	Mar 10, 2019
Toyota	Corolla	2018	4	90	WRO-3	Oct 16, 2019
Skoda	Superb	2017	4	185	WRO-2	Mar 11, 2019
Ford	Focus	2015	4	85	KRK-1	Mar 11, 2019
Skoda	Octavia	2018	4	140	OPO-1	May 7, 2019
Skoda	Superb	2018	4	185	WRO-2	Apr 10, 2019
Ford	Focus	2015	4	85	WRO-2	Jun 10, 2019
Toyota	Corolla	2017	4	90	OPO-1	May 2, 2019
Ford	C-Max	2017	5	105	OPO-1	Jan 8, 2019
Ford	Mondeo	2016	4	155	WRO-2	Mar 22, 2019
Ford	Focus	2017	4	85	WRO-3	Feb 12, 2019
Skoda	Superb	2018	4	185	WRO-1	Jan 1, 2019
Skoda	Superb	2017	4	185	OPO-1	Jul 8, 2019
Ford	Focus	2018	4	115	KRK-1	Jan 3, 2019
Ford	Focus	2015	4	85	WRO-3	Sep 13, 2019
Skoda	Octavia	2017	4	140	OPO-1	Jan 10, 2019
Skoda	Octavia	2017	4	140	WRO-1	Sep 9, 2019
Ford	C-Max	2016	5	105	WRO-1	Mar 16, 2019
Ford	S-Max	2016	6	105	WRO-3	Feb 23, 2019
Toyota	Corolla	2017	4	90	WRO-3	Jan 19, 2019
Skoda	Superb	2015	4	185	WRO-2	Jan 10, 2019
Toyota	Corolla	2015	4	90	WRO-1	Feb 5, 2019
Ford	Focus	2017	4	115	WRO-3	Feb 6, 2019
Ford	C-Max	2015	5	105	WRO-1	Sep 20, 2019
Ford	C-Max	2015	5	105	WRO-2	Mar 19, 2019
Toyota	Auris	2016	4	115	OPO-1	Sep 1, 2019
Skoda	Octavia	2015	4	140	WRO-1	Apr 18, 2019

Toyota Corolla (2017)

Status:Available

Available until:May 2, 2019

License plate:DWR 1C4Q

Current location:OPO-1

Mileage:5138 km

Engine:1.0 4A-GAE

Horsepower:90

Seats:4

Book

Rysunek A.3 Widok listy dostępnych pojazdów (vehicle-list-booking)

## A.4 Historia rezerwacji

Widok (pozycja *My bookings* na pasku nawigacyjnym) zawiera rezerwacje utworzone tylko i wyłącznie przez zalogowanego użytkownika; użytkownik bez praw kierownika nie jest w stanie przeglądać cudzych rezerwacji. Kliknięcie w rekord otwiera szczegółowy widok rezerwacji.

VehifleetNew bookingMy bookingsManage vehiclesManage vehicle modelsManage bookingsJan Pajdak (IT)

My bookings:

Vehicle	Status	Start date	End date
Skoda Octavia	Completed	Mar 16, 2017	Mar 23, 2017
Skoda Octavia	Completed	Jul 18, 2017	Jul 30, 2017
Ford Focus	Completed	Jul 8, 2015	Jul 22, 2015
Ford Focus	Completed	Aug 10, 2018	Aug 14, 2018
Ford Focus	Completed	Mar 20, 2018	Mar 23, 2018
Ford Focus	Completed	Jun 8, 2018	Jun 21, 2018
Ford C-Max	Completed	Mar 23, 2018	Apr 5, 2018
Ford C-Max	Completed	Mar 6, 2016	Mar 18, 2016
Ford Focus	Completed	Aug 1, 2018	Aug 9, 2018
Ford C-Max	Completed	Sep 24, 2017	Oct 1, 2017
Ford C-Max	Completed	Jun 10, 2017	Jun 15, 2017
Ford Mondeo	Completed	Jun 4, 2018	Jun 9, 2018
Ford Mondeo	Completed	Aug 19, 2018	Sep 2, 2018
Skoda Superb	Completed	Apr 21, 2017	Apr 24, 2017
Skoda Superb	Completed	Jan 19, 2018	Jan 31, 2018
Ford Focus	Completed	Jul 21, 2017	Jul 31, 2017
Ford Focus	Completed	Apr 22, 2018	May 4, 2018
Ford Focus	Completed	Jul 17, 2017	Jul 31, 2017
Ford Mondeo	Completed	Jan 17, 2017	Jan 24, 2017
Ford Mondeo	Completed	Jun 1, 2016	Jun 10, 2016
Toyota Corolla	Completed	Jan 5, 2017	Jan 9, 2017
Ford S-Max	Completed	Jan 15, 2018	Jan 27, 2018
Skoda Octavia	Completed	Jul 2, 2016	Jul 12, 2016
Skoda Octavia	Completed	Apr 17, 2017	Apr 26, 2017
Ford Focus	Completed	Apr 3, 2018	Apr 12, 2018
Ford C-Max	Completed	Mar 16, 2018	Mar 21, 2018
Ford C-Max	Completed	May 23, 2018	May 27, 2018
Ford C-Max	Completed	Jan 17, 2018	Jan 24, 2018

Rysunek A.4 Widok listy historii rezerwacji (booking-personal)

## A.5 Wszystkie rezerwacje

Widok zawierający wszystkie rezerwacje, dostępny z poziomu paska nawigacyjnego (*Manage bookings*).

VehifleetNew bookingMy bookingsManage vehiclesManage vehicle modelsManage bookings

Jan Pajdak (IT)

Employee	From date	To date	Statuses	Filter	Reset
Vehicle	Status	Start date	End date		
Toyota Corolla	Completed	Sep 23, 2018	Sep 28, 2018		
Ford C-Max	Completed	Sep 21, 2018	Oct 2, 2018		
Ford Focus	Completed	Sep 24, 2018	Oct 5, 2018		
Ford Focus	Completed	Sep 21, 2018	Sep 30, 2018		
Ford Focus	Completed	Sep 6, 2018	Sep 18, 2018		
Ford Focus	Completed	Sep 7, 2018	Sep 12, 2018		
Ford C-Max	Completed	Sep 21, 2018	Sep 28, 2018		
Toyota Auris	Completed	Sep 18, 2018	Sep 27, 2018		
Toyota Auris	Completed	Sep 11, 2018	Sep 16, 2018		
Toyota Auris	Completed	Sep 21, 2018	Sep 27, 2018		
Skoda Octavia	Completed	Sep 20, 2018	Oct 3, 2018		
Ford Focus	Completed	Sep 12, 2018	Sep 26, 2018		
Toyota Auris	Completed	Sep 12, 2018	Sep 23, 2018		
Skoda Octavia	Completed	Sep 13, 2018	Sep 18, 2018		
Ford Focus	Completed	Sep 17, 2018	Sep 21, 2018		
Ford C-Max	Completed	Sep 9, 2018	Sep 21, 2018		
Ford Mondeo	Completed	Sep 8, 2018	Sep 15, 2018		
Skoda Superb	Completed	Sep 18, 2018	Sep 27, 2018		
Skoda Superb	Completed	Sep 15, 2018	Sep 22, 2018		
Toyota Auris	Completed	Sep 10, 2018	Sep 13, 2018		
Toyota Auris	Completed	Sep 24, 2018	Oct 3, 2018		

Rysunek A.5 Widok listy historii rezerwacji (booking-list)

## A.6 Szczegóły rezerwacji

W tym widoku można zarówno utworzyć nową rezerwację, jak i edytować już istniejącą.

VehifleetNew bookingMy bookingsManage vehiclesManage vehicle modelsManage bookingsJan Pajdak (IT)

Booking

Status:

Completed

Start date:

2018-03-23

End date:

2018-04-05

Purpose:

Example purpose generated during seeding.

Cost (PLN):

40

Fuel consumed (litres):

8

Mileage (km):

86

Manager notes:

Save

Added by: admin at Nov 21, 2018

Rysunek A.6 Widok szczegółowy rezerwacji (booking-details)

### A.6.1 Szczegóły rezerwacji w trybie kierownika

Jeżeli zalogowanym użytkownikiem jest kierownik, wyświetlony zostaje dodatkowy panel z informacjami o użytkowniku który utworzył rezerwację.

VehifleetNew bookingMy bookingsManage vehiclesManage vehicle modelsManage bookingsJan Pajdak (IT)

Booking

Status:

Completed

Start date:

2018-09-07

End date:

2018-09-12

Purpose:

Example purpose generated during seeding.

Cost (PLN):

60

Fuel consumed (litres):

15

Mileage (km):

200

Manager notes:

Save

Added by: admin at Nov 21, 2018

Employee information:

First name:

Sebastian

Last name:

Sebastianowski

Username:

ssebastianowski

Department:

Sales

E-mail:

ssebastianowski@somedomain.com

Phone number:

123 456 789

Rysunek A.7 Widok szczegółowy rezerwacji w trybie kierownika (booking-details)



## A.7 Wszystkie pojazdy

Widok dostępny wyłącznie dla kierowników — jeżeli zalogowany użytkownik posiada odpowiednie uprawnienia, może przejść do tego widoku za pomocą przycisku *Manage vehicles* na pasku nawigacyjnym.

Widok pozwala na przeglądanie wszystkich pojazdów. Lista może być filtrowana po numerze karoserii, kodzie obecnej lokalizacji oraz statusie. Kliknięcie na pojazd otwiera jego widok szczegółowy.

VehifleetNew bookingMy bookingsManage vehiclesManage vehicle modelsManage bookingsJan Pajdak (IT)

Chassis code

Location code

Status

Filter

Reset

Chassis Code	Manufacturer	Model	YOM	Status	Location
I1Y9HNIMESHU	Ford	Focus	2016	Available	KRK-1
571VIXS34I71	Ford	C-Max	2016	Available	WRO-1
UTQYXSB4JGE	Toyota	Corolla	2018	Available	WRO-3
6T7HDCITV0BZ	Toyota	Auris	2015	InMaintenance	OPO-1
336J4Q7ZF4E	Skoda	Superb	2017	Available	WRO-2
5SX28LAN2LPK	Ford	Focus	2015	Available	KRK-1
MHLI99XWS3OL	Skoda	Octavia	2018	Available	OPO-1
TN2VWMC54JP1	Skoda	Superb	2018	Available	WRO-2
TZ1UXM08X7ZU	Ford	Focus	2015	Available	WRO-2
0UYVWETOC5C7	Toyota	Corolla	2017	Available	OPO-1
4VZLW2GG7JUC	Ford	C-Max	2017	Available	OPO-1
NAJIA1OE0C2B	Ford	Mondeo	2016	Available	WRO-2
HS11MIV1AR8W	Ford	Focus	2017	Available	WRO-3
TIZOCIKJINBR	Skoda	Superb	2018	Available	WRO-1
E5RQAGK8NWGE	Skoda	Superb	2017	Available	OPO-1
XNEUII4NQAJH	Ford	Focus	2018	Available	KRK-1
NIZV3W2GYK3W	Ford	Focus	2015	Available	WRO-3
QLC444Z0H6HA	Skoda	Octavia	2017	Available	OPO-1
OATYAWL7HMFQ	Skoda	Octavia	2017	Available	WRO-1
H4L68TMSK2P4	Ford	C-Max	2016	Available	WRO-1
1X6HF531O8QH	Ford	S-Max	2016	Available	WRO-3
56WJ6OTCSBEJ	Toyota	Corolla	2017	Available	WRO-3
HYCBBSUEEP3X	Skoda	Superb	2015	Available	WRO-2
V3FEOOPXDE3O	Toyota	Corolla	2015	Available	WRO-1
W5FNPI50DCEI	Ford	Focus	2017	Available	WRO-3
05POJNOYXMEM	Ford	C-Max	2015	Available	WRO-1
2VCRJ2KLF9U3	Ford	C-Max	2015	Available	WRO-2
6YYTZNSEGWGW	Toyota	Auris	2016	Available	OPO-1

Rysunek A.8 Widok listy pojazdów (vehicle-list)

## A.8 Szczegóły pojazdu

Widok zawierający wszystkie informacje na temat pojazdu wraz z listą ubezpieczeń i napraw, które mogą być dodawane lub edytowane. Przejście do widoków szczegółowych ubezpieczeń/napraw odbywa się przez kliknięcie na odpowiedni wiersz w liście. Wprowadzanie nowych informacji odbywa się odpowiednio przy użyciu przycisków *New Insurance* i *New Maintenance*. Użytkownik może wyświetlić wszystkie rezerwacje danego pojazdu lub jego dokładną specyfikację techniczną po kliknięciu na odpowiedni odnośnik na dole okna.

Pojazd nie może być edytowany jeżeli jest obecnie zarezerwowany.

VehifleetNew bookingMy bookingsManage vehiclesManage vehicle modelsManage bookingsJan Pajdak (IT)

Toyota Auris (2017)

Status:

Available

Location code:

WRO-2

Can be booked until:

2019-03-19T00:00:00

License plate:

WRO K9W9

Year of manufacture:

2017

Chassis code:

O22DVM2DKCNI

Inspection valid until:

2019-07-10

Operating cost:

Cost (PLN):

5225

Fuel consumed (litres):

379

Mileage (km):

6505

Save

Decomission

Show all bookings of this vehicle

Go to vehicle model

Added by: admin at Nov 21, 2018

Insurances

Insurance	Begins	Ends
INS-2018-3-19-1092	Mar 19, 2018	Mar 19, 2019
INS-2017-3-19-1036	Mar 19, 2017	Mar 19, 2018

New insurance

Maintenances

Date	Completed	Cost (PLN)
Sep 10, 2017	Yes	756
Sep 5, 2018	Yes	1306
Feb 10, 2018	Yes	252
Feb 1, 2018	Yes	302

New maintenance

Rysunek A.9 Widok szczegółowy pojazdu (vehicle-details)

## A.9 Szczegóły ubezpieczenia

Widok pozwalający na utworzenie lub modyfikację ubezpieczenia.

The screenshot shows a web application interface with a dark blue header bar containing navigation links: 'Vehifleet', 'New booking', 'My bookings', 'Manage vehicles', 'Manage vehicle models', and 'Manage bookings'. The user 'Jan Pajdak (IT)' is logged in. The main content area displays a form titled 'Insurance INS-2017-10-22-1144'. The form fields are: 'Insurer:' with a text input containing 'Protecto'; 'Insurance ID:' with a text input containing 'INS-2017-10-22-1144'; 'Start date:' with a date input containing '2017-10-22'; 'End date:' with a date input containing '2018-10-22'; 'Cost (PLN):' with a text input containing '718'; and 'Mileage (km):' with a text input containing '4879123'. At the bottom of the form are two buttons: 'Save' (dark blue) and 'Delete' (red). Below the buttons is a link 'Back to vehicle' in green. At the very bottom of the form, it says 'Added by: admin at Nov 21, 2018'.

Rysunek A.10 Widok szczegółowy ubezpieczenia (insurance-details)

## A.10 Szczegóły naprawy

Widok pozwalający na utworzenie lub modyfikację zdarzenia serwisowego.

The screenshot shows the same web application interface as Figure A.10. The main content area displays a form titled 'Maintenance'. The form fields are: 'Description:' with a text area containing 'Maintenance example generated during seeding.'; 'Start date:' with a date input containing '2018-02-06'; 'End date:' with a date input containing '2018-02-14'; 'Cost (PLN):' with a text input containing '1969'; and 'Mileage (km):' with a text input containing '5436492'. At the bottom of the form are two buttons: 'Save' (dark blue) and 'Delete' (red). Below the buttons is a link 'Back to vehicle' in green. At the very bottom of the form, it says 'Added by: admin at Nov 21, 2018'.

Rysunek A.11 Widok szczegółowy naprawy (maintenance-details)

## A.11 Wszystkie modele pojazdów

Widok dostępny wyłącznie dla kierowników. Jeżeli zalogowany użytkownik posiada odpowiednie uprawnienia, to może przejść do tego widoku za pomocą przycisku *Manage vehicle models* na pasku nawigacyjnym.

Widok zawiera listę modeli pojazdów; lista może być filtrowana po nazwie producenta. Kliknięcie na wiersz przechodzi do widoku szczegółowego danego pojazdu.

Pod listą znajduje się przycisk umożliwiający wprowadzenie nowego modelu.

Vehifleet
New booking
My bookings
Manage vehicles
Manage vehicle models
Manage bookings

Jan Pajdak (IT)

FilterReset

Manufacturer	Model	Horsepower	Engine	Seats	Weight	Configuration code
Ford	Focus	115	1.8 TDCi	4	1200	FF2018184
Ford	Focus	85	1.6 TDI	4	1150	FF2018154
Ford	Mondeo	155	1.8 EcoBoost	4	1300	FM2018184
Toyota	Auris	115	2.1 E7-GMD	4	1300	TA2018174
Toyota	Corolla	90	1.0 4A-GAE	4	1270	TC2018144
Ford	C-Max	105	1.6 TDDi	5	1300	FC2014184
Ford	S-Max	105	1.6 TDDi	6	1400	FS2012184
Skoda	Superb	185	1.8 SKT-D3TT	4	1200	SS185184
Skoda	Octavia	140	1.8 SKT-D2	4	1250	SO14184
Skoda	Rapid	90	1.4 SKT-D1	4	1100	SR14184

New vehicle model

Rysunek A.12 Widok listy modeli pojazdów (vehicle-model-list)

## A.12 Szczegóły modelu pojazdu

Widok pozwala na dodawanie oraz edycje modeli pojazdów. Za pomocą odnośników na dole strony można wyszukać wszystkie pojazdy danego rodzaju oraz wprowadzić nowy egzemplarz.

VehifleetNew bookingMy bookingsManage vehiclesManage vehicle modelsManage bookings

Jan Pajdak (IT)

Skoda Rapid

Manufacturer:

Skoda

Model:

Rapid

Configuration code:

SR14184

Purchase cost (PLN):

67000

Engine:

1.4 SKT-D1

Horsepower:

90

Seats:

4

Weight:

1100

SaveDelete

Vehicle models that have vehicles cannot be deleted.

Create new vehicle of this type

Show all vehicles of this type

Added by: admin at Nov 19, 2018

Rysunek A.13 Widok szczegółowy modelu pojazdu (vehicle-model-details)