

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

**OCENA PRACY:**

# Spis treści

<b>1</b>	<b>Wprowadzenie</b>	<b>5</b>
1.1	Wstęp . . . . .	5
1.2	Cel i zakres pracy . . . . .	5
1.3	Układ pracy . . . . .	5
<b>2</b>	<b>Istniejące rozwiązania</b>	<b>7</b>
<b>3</b>	<b>Wymagania funkcjonalne i niefunkcjonalne</b>	<b>8</b>
3.1	Wymagania funkcjonalne . . . . .	8
3.2	Wymagania niefunkcjonalne . . . . .	11
3.2.1	Interfejs użytkownika . . . . .	11
3.2.2	Interfejs programistyczny . . . . .	12
3.2.3	Bezpieczeństwo . . . . .	12
<b>4</b>	<b>Zastosowane technologie i narzędzia</b>	<b>13</b>
4.1	Zastosowane technologie . . . . .	13
4.1.1	Interfejs użytkownika . . . . .	13
4.1.2	Interfejs programistyczny . . . . .	14
4.2	Wykorzystane narzędzia . . . . .	14
4.2.1	Repozytorium . . . . .	14
4.2.2	Edytory i środowiska programistyczne . . . . .	14
4.2.3	Dodatkowe narzędzia . . . . .	14
<b>5</b>	<b>Projekt i implementacja</b>	<b>15</b>
5.1	Architektura . . . . .	15
5.2	Standardy . . . . .	16
5.3	Przypadki użycia . . . . .	17
5.4	Bezpieczeństwo . . . . .	18
5.5	Interfejs programistyczny . . . . .	20
5.5.1	Logika biznesowa . . . . .	20
5.5.2	Struktura solucji . . . . .	23
5.5.3	Wstrzykiwanie zależności . . . . .	23
5.5.4	Kontrolery . . . . .	23
5.5.5	Eksport statystyk floty . . . . .	32
5.6	Interfejs użytkownika . . . . .	33
5.6.1	Układ interfejsu użytkownika . . . . .	33
5.7	Walidacja danych . . . . .	33
5.8	Stopka audytowa . . . . .	35
5.9	Dialogi . . . . .	35

<b>SPIS TREŚCI</b>	<b>1</b>
<b>6 Testy</b>	<b>36</b>
6.1 Testy jednostkowe . . . . .	36
6.2 Testy systemowe . . . . .	38
6.3 Testy dymne . . . . .	38
<b>7 Podsumowanie</b>	<b>39</b>
7.1 Wnioski . . . . .	39
7.2 Możliwości rozwoju . . . . .	39
<b>Literatura</b>	<b>39</b>
<b>Appendices</b>	<b>41</b>
<b>A Instrukcja użytkownika</b>	<b>41</b>
A.1 Ekran logowania . . . . .	41
A.2 Ekran wylogowywania . . . . .	41
A.3 Dostępne pojazdy . . . . .	42
A.4 Historia rezerwacji . . . . .	43
A.5 Szczegóły rezerwacji . . . . .	44
A.6 Szczegóły rezerwacji w trybie kierownika . . . . .	45
A.7 Wszystkie pojazdy . . . . .	46
A.8 Szczegóły pojazdu . . . . .	47
A.9 Szczegóły ubezpieczenia . . . . .	48
A.10 Szczegóły naprawy . . . . .	48
A.11 Wszystkie modele pojazdów . . . . .	49
A.12 Szczegóły modelu pojazdu . . . . .	50

# Spis rysunków

2.1	Vinitysoft Fleet Management Software 4.0. . . . .	7
4.1	Przykład kompilacji kodu TypeScript do JavaScript. ( <a href="http://www.typescriptlang.org">http://www.typescriptlang.org</a> )	13
5.1	Uproszczony schemat architektury z wyodrębnionymi najważniejszymi elementami składowymi. . . . .	15
5.2	Diagram przypadków użycia. . . . .	17
5.3	Przykładowy token <i>JWT</i> . . . . .	18
5.4	Diagram klas. . . . .	20
5.5	Widoki interfejsu użytkownika. . . . .	33
5.6	Widok z polami zawierającymi błędne wartości. . . . .	34
5.7	Stopka audytowa. . . . .	35
5.8	Przykładowy dialog. . . . .	35
6.1	Interfejs programu Postman. . . . .	38
A.1	Widok logowania (dashboard-login). . . . .	41
A.2	Widok zalogowanego użytkownika (dashboard-user-details). . . . .	41
A.3	Widok listy dostępnych pojazdów (vehicle-list-booking). . . . .	42
A.4	Widok listy historii rezerwacji (booking-personal). . . . .	43
A.5	Widok szczegółowy rezerwacji (booking-details). . . . .	44
A.6	Widok szczegółowy rezerwacji w trybie kierownika (booking-details). . . . .	45
A.7	Widok listy pojazdów (vehicle-list). . . . .	46
A.8	Widok szczegółowy pojazdu (vehicle-details). . . . .	47
A.9	Widok szczegółowy ubezpieczenia (insurance-details). . . . .	48
A.10	Widok szczegółowy naprawy (maintenance-details). . . . .	48
A.11	Widok listy modeli pojazdów (vehicle-model-list). . . . .	49
A.12	Widok szczegółowy modelu pojazdu (vehicle-model-details). . . . .	50

# Spis tablic

5.1	Najważniejsze konwencje nazewnicze. . . . .	16
5.2	Domyślna konfiguracja relacji ról do poziomu uprawnień. . . . .	19
5.3	Klasy obiektów biznesowych. . . . .	21
5.4	Endpoint <i>api/vehicle-models GET</i> . . . . .	23
5.5	Endpoint <i>api/vehicle-models/manufacturers GET</i> . . . . .	24
5.6	Endpoint <i>api/vehicle-models/id GET</i> . . . . .	24
5.7	Endpoint <i>api/vehicle-models POST</i> . . . . .	24
5.8	Endpoint <i>api/vehicle-models/id PUT</i> . . . . .	24
5.9	Endpoint <i>api/vehicle-models/id DELETE</i> . . . . .	25
5.10	Endpoint <i>api/vehicles GET</i> . . . . .	25
5.11	Endpoint <i>api/vehicles/id GET</i> . . . . .	25
5.12	Endpoint <i>api/vehicles POST</i> . . . . .	26
5.13	Endpoint <i>api/vehicles/id PUT</i> . . . . .	26
5.14	Endpoint <i>api/vehicles/id DELETE</i> . . . . .	26
5.15	Endpoint <i>api/insurances/vehicle/id GET</i> . . . . .	27
5.16	Endpoint <i>api/insurances/id GET</i> . . . . .	27
5.17	Endpoint <i>api/insurances POST</i> . . . . .	27
5.18	Endpoint <i>api/insurances/id PUT</i> . . . . .	27
5.19	Endpoint <i>api/insurances/id DELETE</i> . . . . .	28
5.20	Endpoint <i>api/maintenances/vehicle/id GET</i> . . . . .	28
5.21	Endpoint <i>api/maintenances/id GET</i> . . . . .	28
5.22	Endpoint <i>api/maintenances POST</i> . . . . .	28
5.23	Endpoint <i>api/maintenances/id PUT</i> . . . . .	29
5.24	Endpoint <i>api/maintenances/id DELETE</i> . . . . .	29
5.25	Endpoint <i>api/bookings GET</i> . . . . .	29
5.26	Endpoint <i>api/bookings/id GET</i> . . . . .	30
5.27	Endpoint <i>api/bookings POST</i> . . . . .	30
5.28	Endpoint <i>api/bookings/id PUT</i> . . . . .	30
5.29	Endpoint <i>api/bookings/id DELETE</i> . . . . .	31

# Listingi

5.1	Przykład definiowania relacji między klasami w code-first. . . . .	21
5.2	Ładowanie powiązanych obiektów na przykładzie relacji rezerwacji do po- jazdu. . . . .	22
5.3	Klasa abstrakcyjna <code>AuditableEntity</code> . . . . .	22
5.4	Klasa abstrakcyjna <code>CostGeneratingEntity</code> . . . . .	22
5.5	Przykładowy raport ze statystykami pojazdów. . . . .	32
5.6	Przykładowy walidator używający wyrażenia regularnego. . . . .	33
6.1	Test jednostkowy metody rozszerzeń. . . . .	36
6.2	Test kontrolera korzystający z <i>Moq</i> . . . . .	37

# 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 нефункциональных 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 wypożyczalnię umożliwiającą 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 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 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.

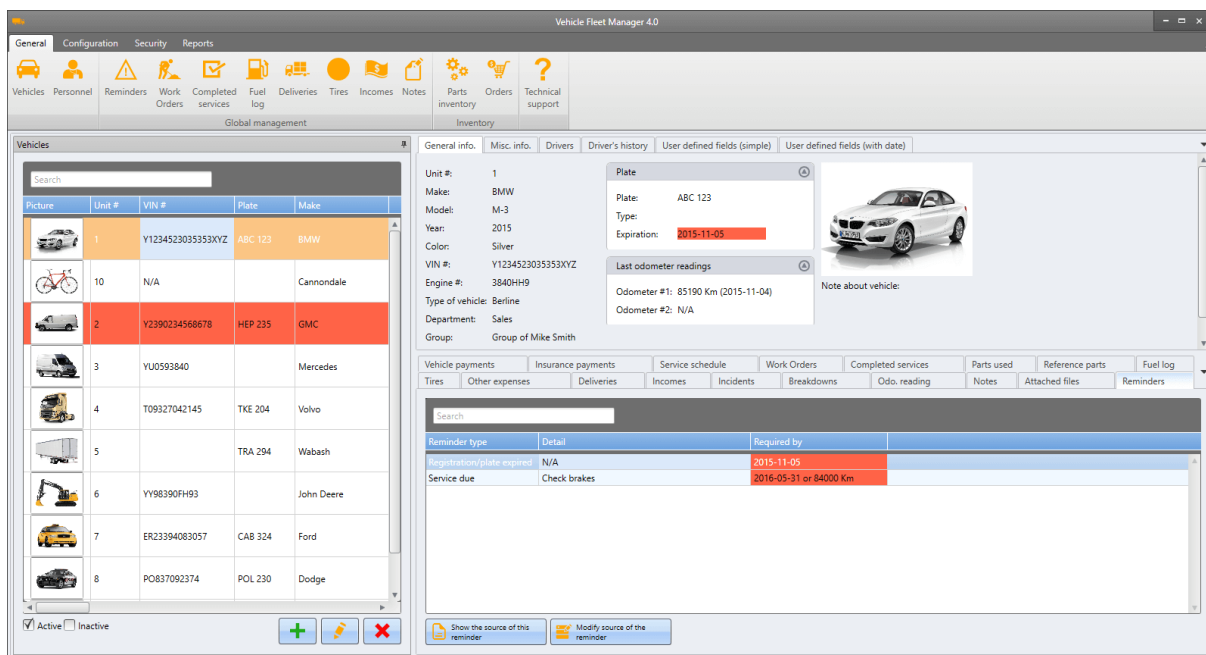
Dodatkowo, jako dodatek dołączona została instrukcja użytkownika.



# Rozdział 2

## Istniejące rozwiązania

Jednymi z popularniejszych rozwiązań obecnych na rynku są *Fleetly* (<https://www.fleetly.co/>) oraz *Vinitysoft Fleet Manager* (<https://www.vinitysoft.com/>). Sposób działania *Fleetly* jest bliski działaniu systemu, który został stworzony w ramach projektu, skupia się on jednak zanadto 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. *Vinitysoft Fleet Manager* kładzie mały nacisk na kontrolę dostępu do pojazdów i system śledzenia towarów; jest to system przeznaczony dla firm których procesy główne opierają się na wykorzystywaniu samochodów. Kolejnym problemem tego systemu jest przestarzały i mało intuicyjny interfejs użytkownika.



Rysunek 2.1 Vinitysoft Fleet Management Software 4.0.

Oba wymienione systemy nie oferują integracji z istniejącymi zasobami firmy oraz są drogie w utrzymaniu ze względu na koszt licencji.

# Rozdział 3

## Wymagania funkcjonalne i niefunkcjonalne

### 3.1 Wymagania funkcjonalne

Wymagania zostały opisane według poniższego wzorca:

Numer	Numer wymagania
Nazwa	Krótką 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 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.

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 zarezerwowany. Pojazd który był rezerwowany, 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 rezerwacji; pojazd nie może zostać zarezerwowany 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 rezerwacji; pojazd nie może zostać zarezerwowany, gdy jest obecnie naprawiany.
Ograniczenia	

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

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

Numer	7
Nazwa	Zbieranie informacji o kosztach rezerwacji
Opis	System umożliwia śledzenie kosztów utrzymania floty na podstawie raportów wprowadzanych przez kierowców.
Aktor	Kierowca
Kryterium spełnienia	Kierowca może wprowadzić informację związane z rezerwacją (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 <i>Excel</i> 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

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.

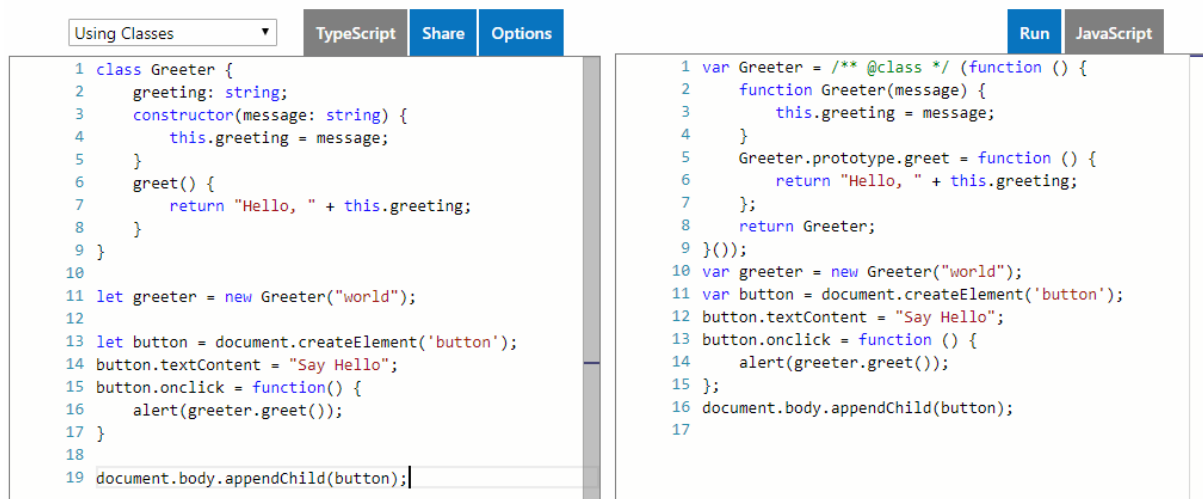
# Rozdział 4

## Zastosowane technologie i narzędzia

### 4.1 Zastosowane technologie

#### 4.1.1 Interfejs użytkownika

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* [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.



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

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

### 4.1.2 Interfejs programistyczny

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* [7], framework ORM (Object-Relational Mapping) 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 Repozytorium

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 Edytory i środowiska programistyczne

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 interfejsu programistycznego wykorzystano *Visual Studio 2017* pozwalającą 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.

### 4.2.3 Dodatkowe narzędzia

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

Do tworzenia diagramów wykorzystano program *Visual Paradigm*.

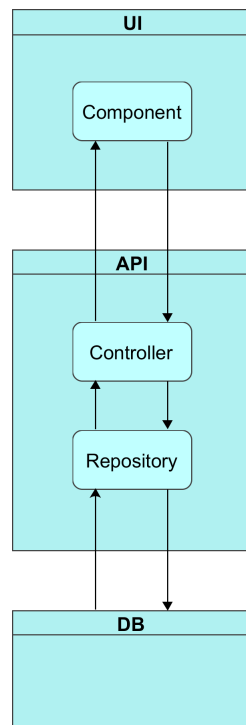


# Rozdział 5

## Projekt i implementacja

### 5.1 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 5.1 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 lokalizacjami 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].

## 5.2 Standardy

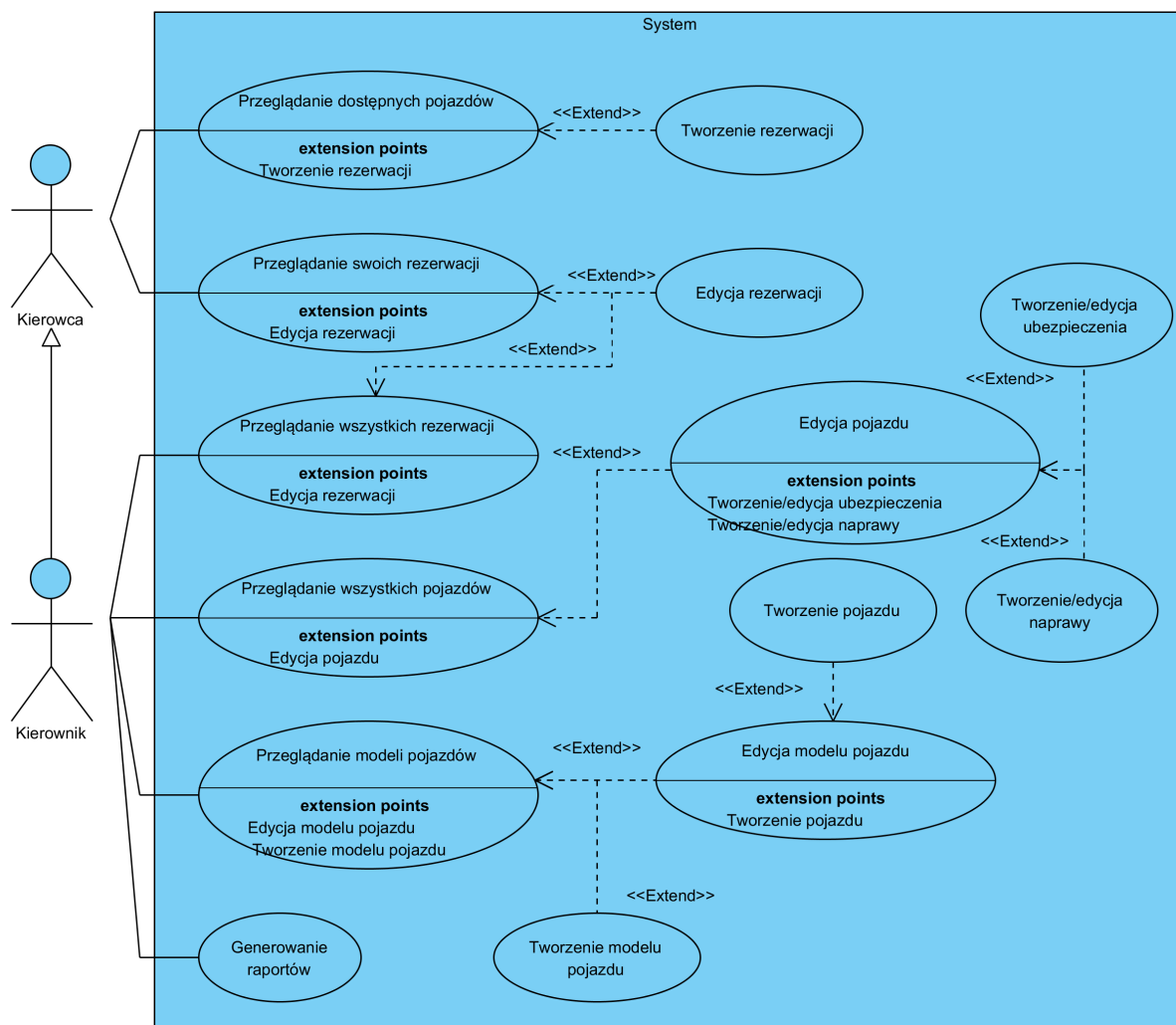
Projekt był tworzony zgodnie z dobrymi praktykami programowania, z naciskiem na poprawną implementację obiektowego paradygmatu programowania. Interfejs programistyczny był tworzony z użyciem sztandarowych możliwości języka C# takimi jak typy ogólne [5] (*Generics*) pozwalające na tworzenie pojedynczych 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.

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

Tablica 5.1 Najważniejsze konwencje nazewnicze.

## 5.3 Przypadki użycia



Rysunek 5.2 Diagram przypadków użycia.



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.

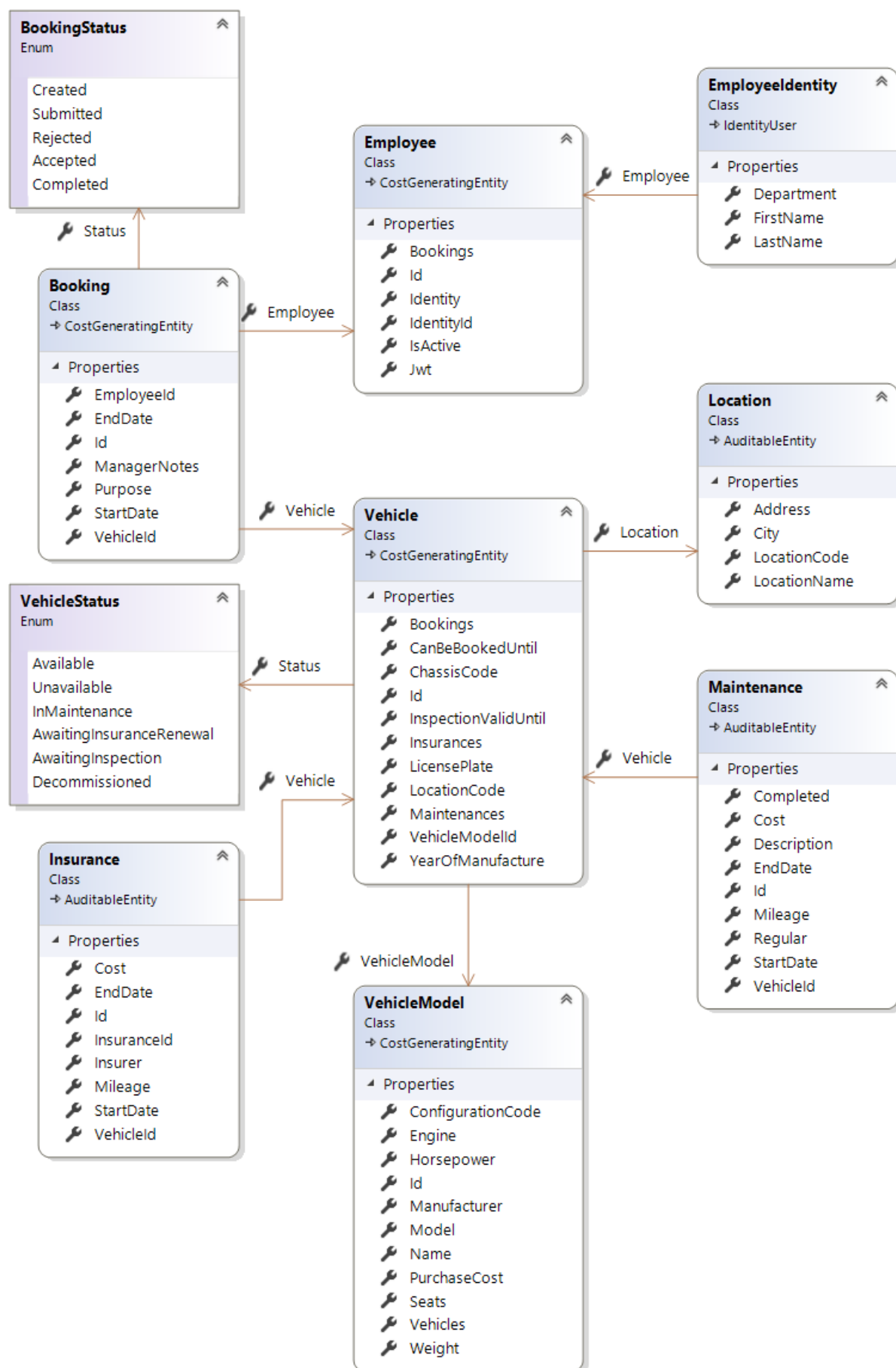
Tablica 5.2 Domyślna konfiguracja relacji ról do poziomu uprawnień.

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; Aplikacja użytkownika blokuje dostęp do widoków przy użyciu mechanizmu *RouteGuard*.

## 5.5 Interfejs programistyczny

### 5.5.1 Logika biznesowa



Rysunek 5.4 Diagram klas.

Diagram klas został wygenerowany przy użyciu *Visual Studio*.

Tablica 5.3 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

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

Do zdefiniowania relacji między tabelami należy użyć pól typu takiego samego jak *PK* 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 (*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()*.

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).

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; }
}
```

Listing 5.2 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);
}
```

Wszystkie klasy związane z logiką biznesową dziedziczą po klasie *AuditableEntity* 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) dodatkowo dziedziczą po klasie *CostGeneratingEntity* przechowującej informacje o koszcie, zużytych paliwie i przejechanych kilometrach.

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.

Listing 5.3 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 5.4 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; }

    [NotMapped]
    public double AverageFuelConsumption => Mileage == 0
        ? 0
        : (FuelConsumed / Mileage);
}
```



### 5.5.2 Struktura solucji

Kod interfejsu programistycznego znajduje się w jednej solucji podzielonej na projekty.

- **Vehifleet** — solucja.

**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 (*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 statystkami.

### 5.5.3 Wstrzykiwanie zależności

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

### 5.5.4 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 (*endpoint*) w interfejsie programistycznym.

Tablica 5.4 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 5.5 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 5.6 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 5.7 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 5.8 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 5.9 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 5.10 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 5.11 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 5.12 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 5.13 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 5.14 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 5.15 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 5.16 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 5.17 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 5.18 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 5.19 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 5.20 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 5.21 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 5.22 Endpoint *api/maintenances POST*.

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

Tablica 5.23 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 5.24 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 5.25 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 5.26 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 5.27 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 5.28 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 5.29 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

### 5.5.5 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.

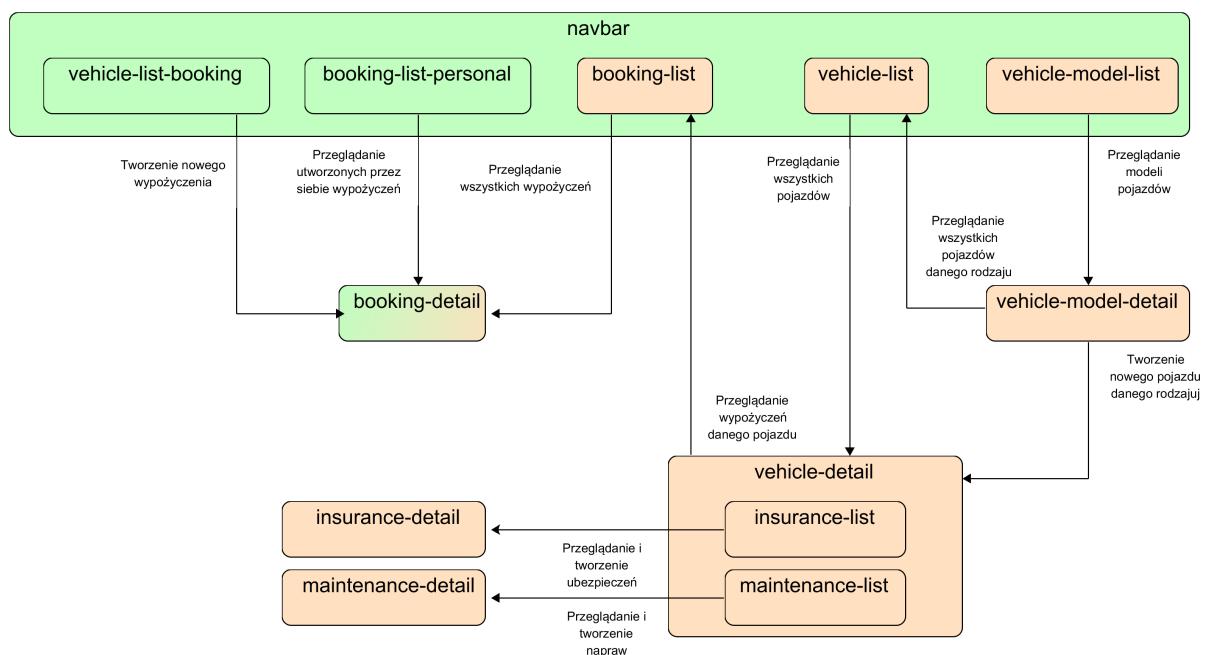
ChassisCode	Manufacturer	Model	YearOfManufacture	Cost	Mileage	FuelConsumed
I1Y9HNIMESHU	Ford	Focus	2016	"9497,00"	7135	373
571VIXS34I71	Ford	C-Max	2016	"4812,00"	9823	532
UTQYXSB44JGE	Toyota	Corolla	2018	"2006,00"	659	0
6T7HDCTIV0BZ	Toyota	Auris	2015	"13743,00"	20026	815
336J4Q7ZFI4E	Skoda	Superb	2017	"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

## 5.6 Interfejs użytkownika

### 5.6.1 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 5.5 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.

## 5.7 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 błędy 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:

2018-04-10

End date:

2019-04-10

Cost (PLN):

Cost must be a valid number.

Mileage (km):

abcdefghijklm

Mileage must be a valid number.

Save

Delete

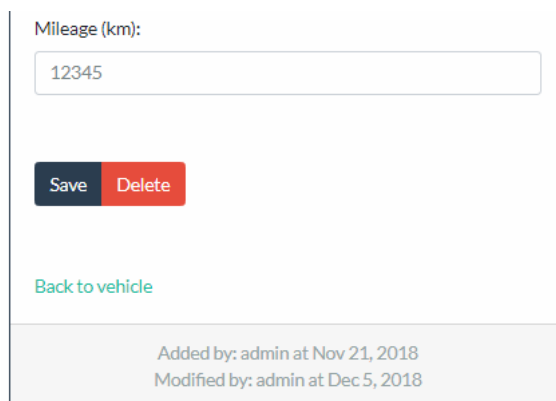
[Back to vehicle](#)

Added by: admin at Nov 21, 2018

Rysunek 5.6 Widok z polami zawierającymi błędne wartości.

## 5.8 Stopka audytowa

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

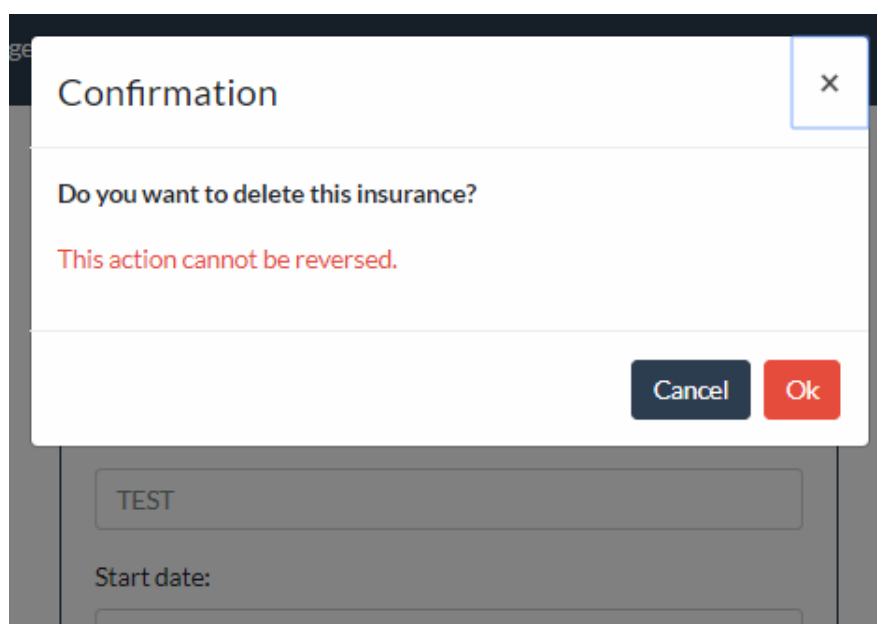


The screenshot shows a form with a label "Mileage (km):" above a text input field containing "12345". Below the input field are two buttons: "Save" (dark blue) and "Delete" (red). Underneath these buttons is a link "Back to vehicle" in green text. At the bottom of the form, there is a grey footer area containing the text "Added by: admin at Nov 21, 2018" and "Modified by: admin at Dec 5, 2018".

Rysunek 5.7 Stopka audytowa.

## 5.9 Dialogi

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



The screenshot shows a "Confirmation" dialog box with a close button (X) in the top right corner. The main text inside the dialog asks "Do you want to delete this insurance?" and includes a warning in red text: "This action cannot be reversed." At the bottom right of the dialog are two buttons: "Cancel" (dark blue) and "Ok" (red). The background shows a blurred form with a "TEST" button and a "Start date:" label.

Rysunek 5.8 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);
}
```

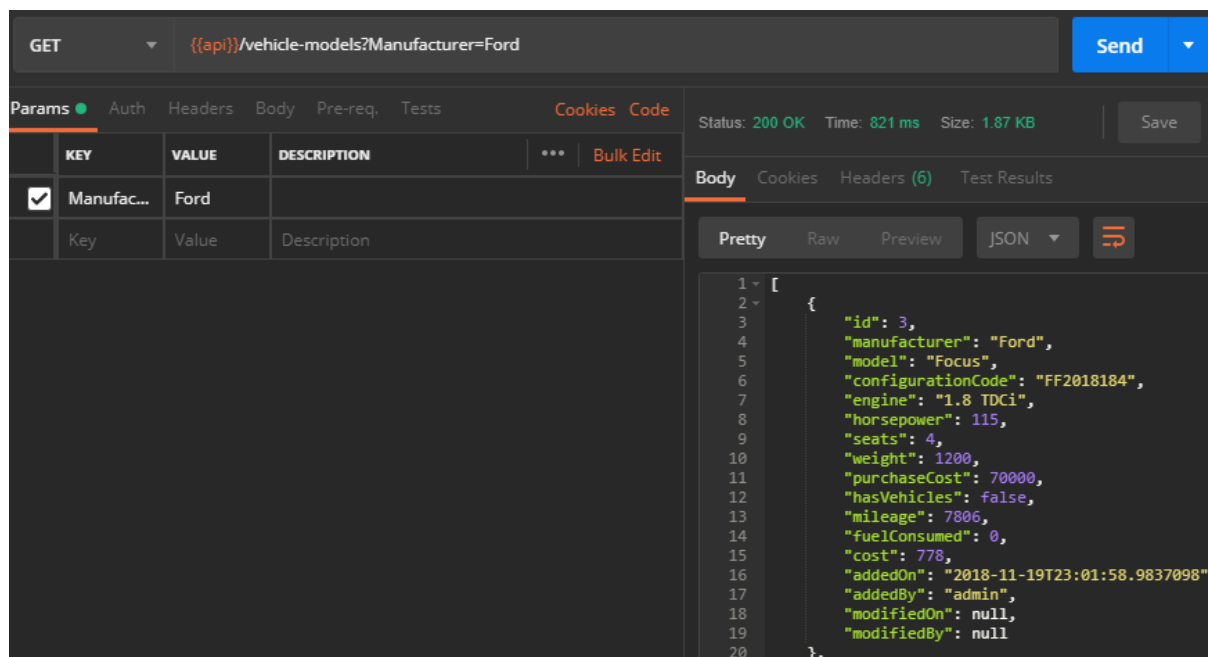
Listing 6.2 Test kontrolera korzystający z *Moq*.

```
[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 (*smoke test*) były użyte w końcowej fazie projektu; 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.



# 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 problemy 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 do 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

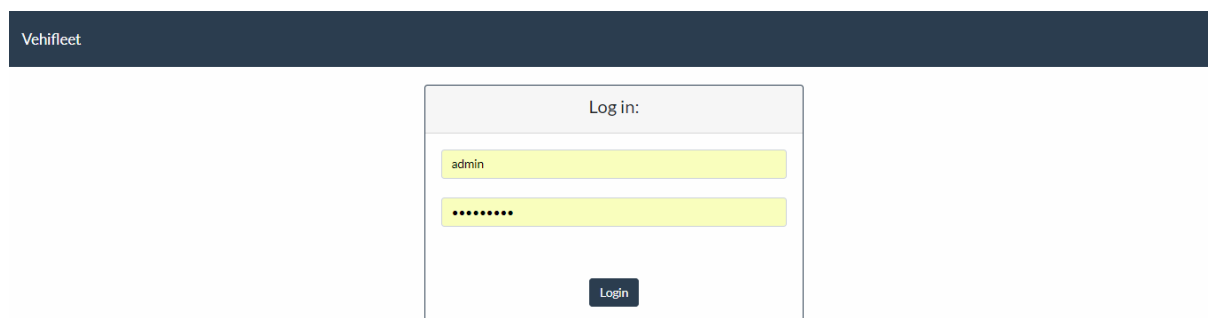
- [1] Angular Docs. *Introduction to components*, 2018. URL <https://angular.io/guide/architecture-components>. Dostęp 03.12.2018.
- [2] Angular Docs. *Style Guide*, 2018. URL <https://angular.io/guide/styleguide>. Dostęp 03.12.2018.
- [3] JWT. *JSON Web Tokens*, 2018. URL <https://jwt.io>. Dostęp 03.12.2018.
- [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

### A.1 Ekran logowania

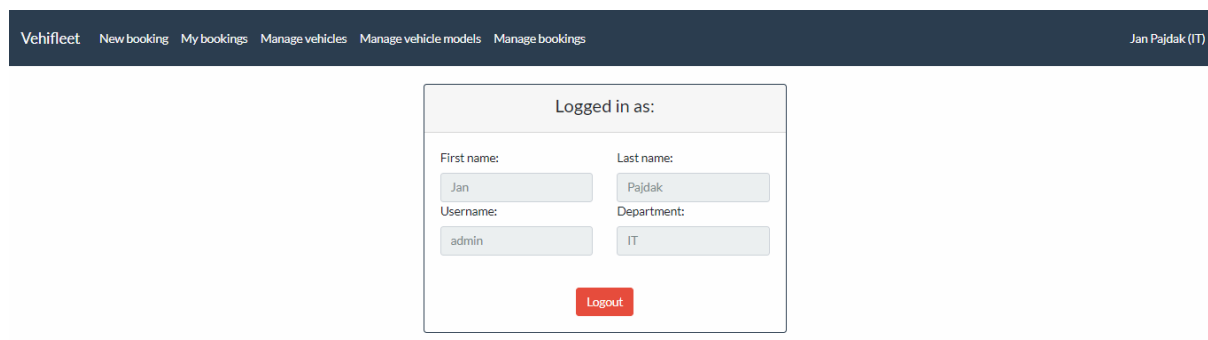
Jedyny widok dostępny dla niezalogowanego użytkownika. Aplikacja uniemożliwia dostęp do wszystkich innych widoków osobom niezalogowanym; przy ręcznej zmianie adresu w przeglądarce nieupoważniony użytkownik zostanie przekierowany do tego widoku.



Rysunek A.1 Widok logowania (dashboard-login).

### A.2 Ekran wylogowywania

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



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

## A.3 Dostępne pojazdy

Lista dostępnych pojazdów która 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*.

Vehifleet
New booking
My bookings
Manage vehicles
Manage vehicle models
Manage bookings
Jan Pajdak (IT)

FilterReset

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 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.

Vehifleet   New booking   My bookings   Manage vehicles   Manage vehicle models   Manage 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.4 Widok listy historii rezerwacji (booking-personal).

## A.5 Szczegóły rezerwacji

W tym widoku można zarówno utworzyć nowe rezerwację, jak i edytować już istniejące.

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.5 Widok szczegółowy rezerwacji (booking-details).

## A.6 Szczegóły rezerwacji w trybie kierownika

Jeżeli zalogowany użytkownik posiada pełne uprawnienia i przegląda cudze rezerwacji, po stronie prawej zostają wyświetlone dodatkowe informacje 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.6 Widok szczegółowy rezerwacji w trybie kierownika (booking-details).

## A.7 Wszystkie pojazdy

Widok pozwala na przeglądanie wszystkich pojazdów. Lista może być filtrowana po numerze karoserii, kodzie obecnej lokalizacji oraz statusie.

Vehifleet

New booking

My bookings

Manage vehicles

Manage vehicle models

Manage bookings

Jan Pajdak (IT)

Chassis code

Location code

Status

Filter

Reset

Chassis Code	Manufacturer	Model	YOM	Status	Location
I1Y9HNIMESHU	Ford	Focus	2016	Available	KRK-1
571VXS34I71	Ford	C-Max	2016	Available	WRO-1
UTQYXS844JGE	Toyota	Corolla	2018	Available	WRO-3
6T7HDCTIV0BZ	Toyota	Auris	2015	InMaintenance	OPO-1
336J4Q7ZF14E	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
4VZLW2GQ7JUC	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
ESRQAGK8NWGE	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
H4L68TM5K2P4	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.7 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. 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.

Vehifleet

New bookingMy bookingsManage vehiclesManage vehicle modelsManage bookings

Jan 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.8 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. The header contains a navigation menu with links: 'Vehifleet', 'New booking', 'My bookings', 'Manage vehicles', 'Manage vehicle models', and 'Manage bookings'. On the right side of the header, the user 'Jan Pajdak (IT)' is logged in. The main content area displays a form titled 'Insurance INS-2017-10-22-1144'. The form includes several input fields: 'Insurer:' with the value 'Protecto', 'Insurance ID:' with 'INS-2017-10-22-1144', 'Start date:' with '2017-10-22', 'End date:' with '2018-10-22', 'Cost (PLN):' with '718', and 'Mileage (km):' with '4879123'. Below these fields are two buttons: 'Save' (dark blue) and 'Delete' (red). A green link 'Back to vehicle' is positioned below the buttons. At the bottom of the form, a grey footer bar states 'Added by: admin at Nov 21, 2018'.

Rysunek A.9 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.9. The main content area displays a form titled 'Maintenance'. The form includes a 'Description:' field with the text 'Maintenance example generated during seeding.', 'Start date:' with '2018-02-06', 'End date:' with '2018-02-14', 'Cost (PLN):' with '1969', and 'Mileage (km):' with '5436492'. Below these fields are 'Save' (dark blue) and 'Delete' (red) buttons. A green link 'Back to vehicle' is located below the buttons. The bottom grey footer bar indicates 'Added by: admin at Nov 21, 2018'.

Rysunek A.10 Widok szczegółowy naprawy (maintenance-details).

## A.11 Wszystkie modele pojazdów

Widok zawiera listę modeli pojazdów; lista może być filtrowana po nazwie producenta.

Pod listą znajduje się przycisk umożliwiający wprowadzenie nowego pojazdu.

VehifleetNew bookingMy bookingsManage vehiclesManage vehicle modelsManage bookings

Jan Pajdak (IT)

Filter

Reset

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.11 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 bookingsJan 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 typeShow all vehicles of this type

Added by: admin at Nov 19, 2018

Rysunek A.12 Widok szczegółowy modelu pojazdu (vehicle-model-details).