

# Sprawozdanie

---

**Kurs:** Tworzenie aplikacji bazodanowych

**Temat:** System informatyczny wspomagający obsługę floty przedsiębiorstwa

**Data:** 24.06.2025

**Autorzy:** Krzysztof Klecha (lider sekcji)  
Przemysław Dąbrowski  
Paweł Dziergas  
Piotr Mastalerz

## Spis treści

### Zawartość

1. Treść zadania .....	3
2. Analityka .....	3
2.1. Wymagania funkcjonalne .....	3
2.2. Wymagania niefunkcjonalne .....	3
2.3. Wybrane technologie .....	4
2.3.1. Technologia bazy danych .....	4
2.3.2. Technologia Backendu .....	4
2.3.3. Technologia Frontendu .....	4
3. Architektura systemu .....	5
3.1. Baza danych .....	5
3.2. Backend .....	6
3.3. Frontend .....	8
3.4. Komunikacja .....	10
4. Instrukcja dla użytkownika - (Use Cases) .....	11
4.1. Dodawanie pojazdu .....	11
4.2. Dodawanie serwisu pojazdu .....	13
4.3. Wyświetlanie historii serwisowej pojazdu .....	14
4.4. Sortowanie chronologiczne .....	16
5. Wnioski .....	17

## 1. Treść zadania

Celem projektu było stworzenie aplikacji wspomagającej zarządzanie flotą pojazdów w firmie.

W skład systemu miały wchodzić:

- relacyjna baza danych przechowująca informacje obecne w systemie
- backend, pracujący nieustannie na serwerze, realizujący logikę aplikacji oraz udostępniający REST API
- aplikacja kliencka umożliwiająca interakcję z systemem

## 2. Analityka

### 2.1. Wymagania funkcjonalne

Aplikacja powinna umożliwiać wykonywanie następujących czynności:

- Przechowywanie informacji o pracownikach firmy,
- Przechowywanie informacji o posiadanych samochodach,
- Zapisywanie informacji o kosztach utrzymania pojazdów takich jak tankowanie czy serwis,
- Dokumentacja przeglądów technicznych pojazdów,
- Rezerwowanie pojazdów dla pracowników na użytek służbowy lub prywatny,
- Przeglądanie historii zdarzeń takich jak rezerwacje, przeglądy czy serwisy.

### 2.2. Wymagania niefunkcjonalne

Poza wyżej zdefiniowanymi wymaganiami funkcjonalnymi, system powinien również spełniać wymagania niefunkcjonalne:

- Posiadać responsywny interfejs użytkownika,
- Zapewniać bezpieczeństwo dostępu do danych (autoryzacja użytkowników),
- Przeglądalna struktura kodu i dokumentacja API,
- Modułowość komponentów aplikacji.

## 2.3. Wybrane technologie

Na podstawie analizy wymagań funkcjonalnych i нефункциональных podjęto decyzję o wyborze nowoczesnych, popularnych i dobrze wspieranych technologii, za pomocą których zamierzaliśmy realizować projekt.

### 2.3.1. Technologia bazy danych

#### **MariaDB**

Powody:

- Dobra skalowalność,
- MariaDB jest open-source i posiada duże wsparcie społeczności,
- pełne wsparcie standardu SQL,
- dobra integracja z entity framework.

### 2.3.2. Technologia Backendu

#### **.NET, Swagger, Entity Framework Core**

Powody:

- entity framework core, jako ORM umożliwia dostęp do bazy danych
- Swagger umożliwia automatyczne generowanie dokumentacji API oraz ułatwia testowanie

### 2.3.3. Technologia Frontendu

#### **React, Typescript**

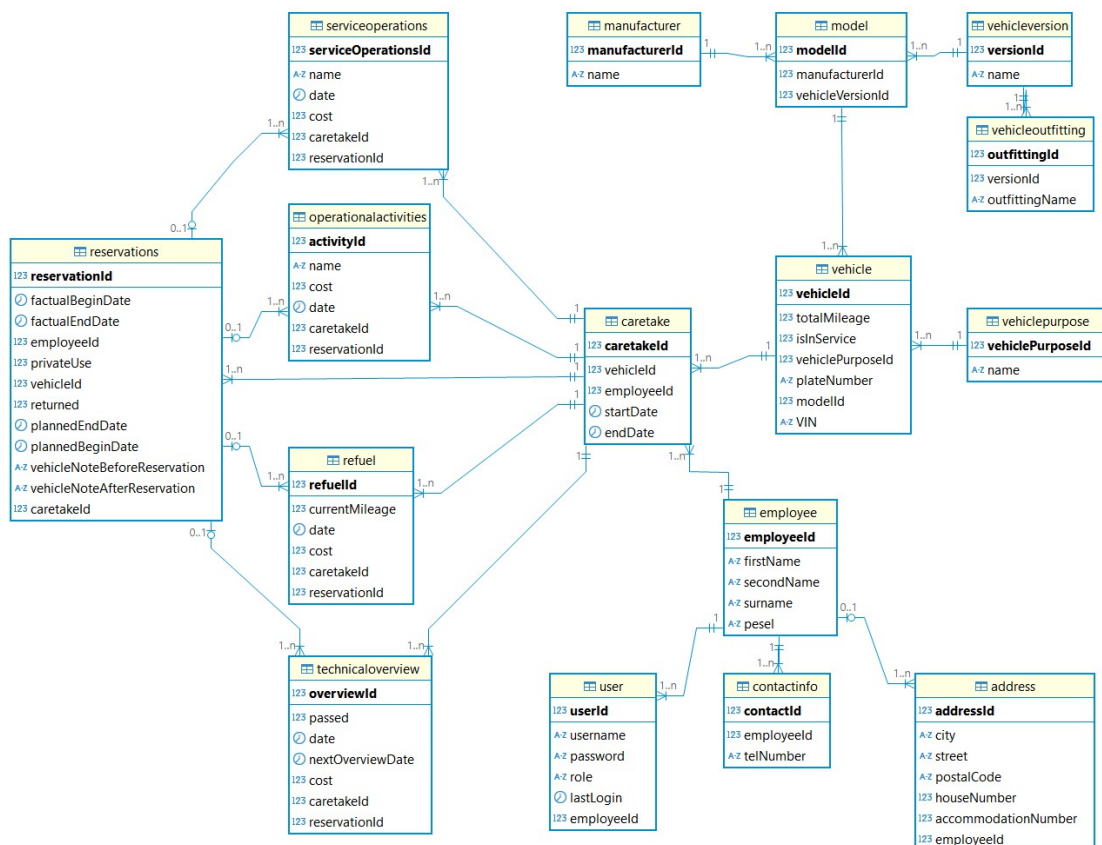
Powody:

- doskonała współpraca z REST API,
- popularność, skutkująca dobrą dostępnością materiałów i wzorów,
- możliwość dostępu z przeglądarki.

### 3. Architektura systemu

#### 3.1. Baza danych

System wykorzystuje relacyjną bazę danych SQL (Maria DB).



Rys. 1 – schemat bazy danych

Taka struktura bazy danych umożliwia:

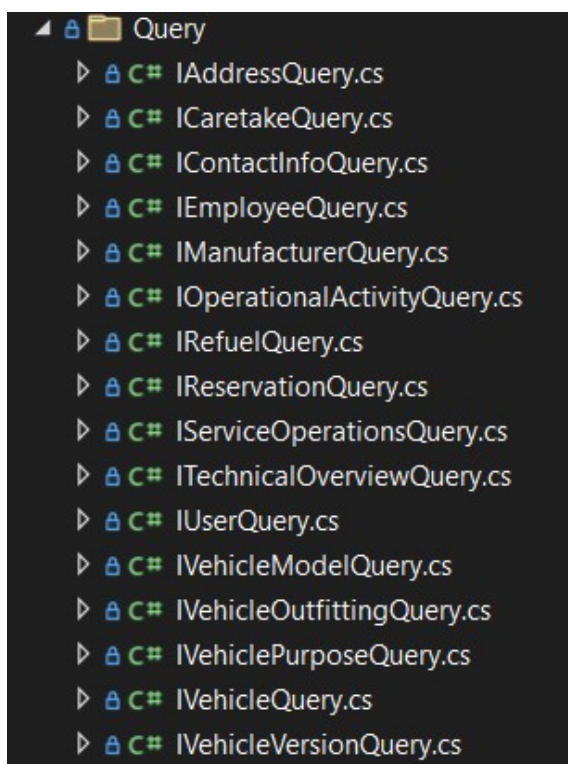
- śledzenie, który pracownik jest odpowiedzialny za którą czynność,
- przeglądanie historii operacji,
- minimalizację rozmiaru danych na dysku.

### 3.2. Backend

Część serwerowa systemu została zaimplementowana w technologii .NET. Backend odpowiada za logikę biznesową, komunikację z bazą danych oraz udostępnienie interfejsu API dla aplikacji frontendowej.

Centralny plik konfiguracyjny Program.cs zawiera definicje usług oraz konfigurację połączenia z bazą danych przy pomocy Entity Framework Core i providera Pomelo.EntityFrameworkCore.MySql (dla MariaDB). Aplikacja obsługuje także Swaggera, który automatycznie generuje dokumentację dostępnych endpointów REST API.

Wszystkie zapytania do bazy danych realizowane są przez odpowiednio wydzielone klasy implementujące interfejsy IQuery. Przykładowe interfejsy to IRefuelQuery, IVehicleQuery, IServiceOperationsQuery i inne.

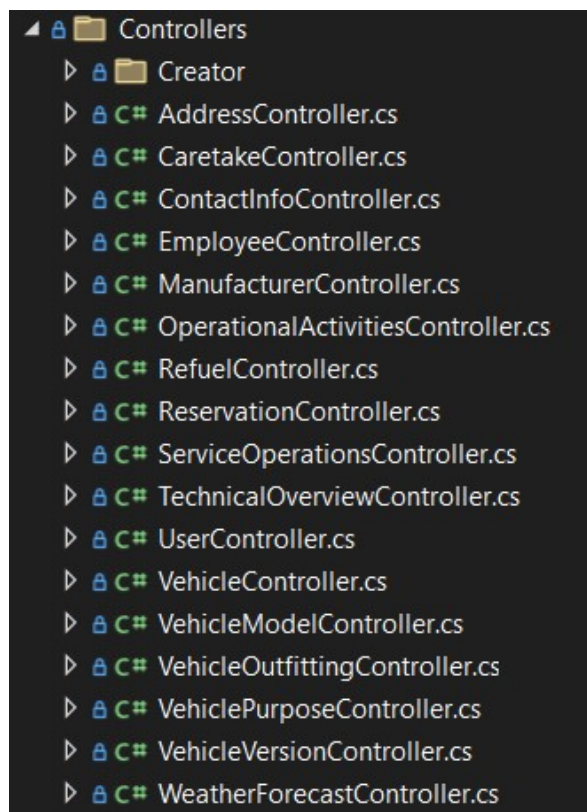


Rys. 2 - interfejsy

Backend obsługuje różne typy operacji:

- GET – do pobierania danych (np. historii serwisu),
- PUT – do zapisu nowych danych (np. zapis tankowania),
- POST – do aktualizacji już istniejących danych (np. zapis daty końca rezerwacji),
- DELETE – do usuwania wpisów.

Struktura kontrolerów (np. RefuelController.cs) zapewnia przejrzystość kodu oraz umożliwia łatwą rozbudowę systemu. Dzięki podzieleniu na fragmenty, komponenty są osobno testowalne i łatwe do utrzymania.



Rys. 3 - kontrolery

### 3.3. Frontend

Aplikacja frontendu została utworzona przy użyciu biblioteki React z wykorzystaniem TypeScript dla zwiększenia wygody oraz czytelności kodu. Za przekierowywanie na strony zawierające konkretne komponenty aplikacji odpowiada „router” z biblioteki react.

Plik App.tsx definiuje strukturę aplikacji. Interfejs jako pierwszą część zawiera zawsze pasek nawigacyjny „Navbar” na górze strony. Pozwala on na przenoszenie się między poszczególnymi kartami. Jako drugą część zawiera jedną ze stron wybranych przez użytkownika.

## Fleet Manager

Vehicles Employees Service Reservation Costs

Rys. 4 – „Navbar”

Użytkownik, poprzez widoczny powyżej pasek nawigacyjny może wybrać jedną z pięciu stron:

- Vehicles
- Employees
- Service
- Reservation
- Costs



Dwie przykładowe strony:

## Fleet Manager

Vehicles Employees Service Reservation Costs

### Home page

This component demonstrates fetching data from the server.

Date	Temp. (C)	Temp. (F)	Summary
2025-06-24	-7	20	Freezing
2025-06-25	45	112	Cool
2025-06-26	29	84	Freezing
2025-06-27	30	85	Mild
2025-06-28	22	71	Sweltering
2025-06-29	52	125	Freezing
2025-06-30	-9	16	Hot
2025-07-01	47	116	Cool
2025-07-02	0	32	Chilly
2025-07-03	20	67	Warm
2025-07-04	-10	15	Chilly

Rys. 5 – strona domyślna „Home”

## Fleet Manager

Vehicles Employees Service Reservation Costs

### Not in service

Vehicle ID	Plate Number	Model ID
2	9CGO5724	6
3	DXZ 545	10
4	WGO 256	2

Send to Service  
Return from Service  
Refuel  
Service history

### In service

Vehicle ID	Plate Number	Model ID
1	OCM 6100	1

Rys. 6 – strona obsługująca serwisy pojazdów „Service”

### 3.4. Komunikacja

Frontend i backend komunikują się poprzez protokół HTTP/HTTPS, wykorzystując REST API. Przykładowo, komponent Service.tsx umożliwia użytkownikowi zarejestrowanie tankowania pojazdu. W tym celu użytkownik wybiera pojazd z listy i wprowadza dane tankowania w formularzu. Dane są następnie przesyłane do backendu za pomocą zapytania PUT na endpoint /api/refuel/create.

Rysunki 7 i 8 pokazują przykład komunikacji frontendu z backendem.

```
37 [HttpPut("create")]
38 public async Task<IActionResult> CreateRefuelAsync(Refuel model)
39 {
40     try
41     {
42         await query.CreateRefuelAsync(model);
43         return Ok();
44     }
45     catch (Exception ex)
46     {
47         var msg = exCreator.ConstructErrorMessage("put", model, ex);
48         return StatusCode(500, msg);
49         throw;
50     }
51 }
```

Rys 7. – fragment pliku „RefuelController.cs”

```
147 await fetch(`/api/refuel/create`, {
148     method: 'PUT',
149     headers: { 'Content-Type': 'application/json' },
150     body: JSON.stringify(payload)
151 });
```

Rys. 8 – fragment pliku „Service.tsx”

## 4. Instrukcja dla użytkownika - (Use Cases)

### 4.1. Dodawanie pojazdu

#### 1. Użytkownik przechodzi do zakładki „Vehicles”

**Fleet Manager** Vehicles Employees Service Reservation Costs

#### My Vehicles

Vehicle identification	Plate Number	VIN
1	OCM 6100	210IP2
2	9CGO5724	5NPEB4AC1DH579041
3	DXZ 545	1FMCU9G96DUA85544
4	WGO 256	1HGFA1684L083662
10	SLU 3451TR	5J21OP22
12	STA 420	OECD325W

Plate number:

Vin:

Purpose:

Overseer:

Total mileage:

Serviced: ☐

#### Vehicle history

#### Reservations

Description:

Name and Surname:

Purpose:

Planned start - end date:

Actual start - end date:

#### 2. Użytkownik klika „Add Vehicle”

**Fleet Manager** Vehicles Employees Service Reservation Costs

#### My Vehicles

Vehicle identification	Plate Number	VIN
1	OCM 6100	210IP2
2	9CGO5724	5NPEB4AC1DH579041
3	DXZ 545	1FMCU9G96DUA85544
4	WGO 256	1HGFA1684L083662
10	SLU 3451TR	5J21OP22
12	STA 420	OECD325W
18	string	string

Plate number:

Vin:

Purpose:

Overseer:

Total mileage:

Serviced: ☐

#### Vehicle history

#### Reservations

Description:

Name and Surname:

Purpose:

Planned start - end date:

Actual start - end date:

3. Użytkownik edytuje potrzebne pola, takie jak rejestracja i numer VIN, po czym klika „save changes”

## Fleet Manager

Vehicles

Employees

Service

Reservation

Costs

### My Vehicles

Vehicle identification	Plate Number	VIN
1	OCM 6100	210JP2
2	9CGO5724	5NPEB4AC1DH579041
3	DXZ 545	1FMCU9G96DUA85544
4	WGO 256	1HGFA16846L083662
10	SLU 3451TR	SI21OPZ2
12	STA 420	OECD325W
18	string	string

### Opel Basic

Plate number

STA 6210A

Vin

LU24AC51OP1

Purpose

Dostawczy

Overseer

Not available

Total mileage

0

Serviced

☐

Add Vehicle

Delete Vehicle

Save changes

### Vehicle history

### Reservations

Description

Here is your description

Name and Surname

Purpose

Planned start - end date

factual start - end date

Reserve

Refuse

4. Pojazd jest dodany i zapisany w odpowiedniej tabeli

My Vehicles		
Vehicle identification	Plate Number	VIN
1	OCM 6100	210JP2
2	9CGO5724	5NPEB4AC1DH579041
3	DXZ 545	1FMCU9G96DUA85544
4	WGO 256	1HGFA16846L083662
10	SLU 3451TR	SI21OPZ2
12	STA 420	OECD325W
18	STA 6210A	LU24AC51OP1

## 4.2. Dodawanie serwisu pojazdu

1. Użytkownik przechodzi do zakładki „Service”

Vehicles   Employees   **Service**   Reservation   Costs

2. Zaznacza pojazd, który chce oddać do serwisu

Not in service		
Vehicle ID	Plate Number	Model ID
2	9CGO5724	6
3	DXZ 545	10
4	WGO 256	2

3. Klika przycisk „send to service”

Not in service		
Vehicle ID	Plate Number	Model ID
2	9CGO5724	6
3	DXZ 545	10
4	WGO 256	2

Send to Service

Return from Service

Refuel

Service history

4. Wypełnia dane w wyświetlonym okienku i naciska przycisk „submit”

The image shows two states of the 'Add Service Operation' dialog box. In the first state, the form is empty. In the second state, the form is filled with the following data:

Field	Value
Service Name	wymiana przedniej szyby
Date (dd.mm.yyyy)	11.06.2025
Cost	800

The 'Submit' button is highlighted with a red box in the second state.

5. Pojazd pojawia się w tabeli „In service”

In service		
Vehicle ID	Plate Number	Model ID
1	OCM 6100	1
2	9CGO5724	6

#### 4.3. Wyświetlanie historii serwisowej pojazdu

1. Użytkownik przechodzi do zakładki „Service”

Vehicles Employees **Service** Reservation Costs

2. Zaznacza pojazd, którego historię serwisu chce zobaczyć

Not in service		
Vehicle ID	Plate Number	Model ID
2	9CGO5724	6
3	DXZ 545	10
4	WGO 256	2

3. Klika przycisk „service history”



4. Wyświetla się historia czynności serwisowych wykonanych dla wybranego pojazdu

Service history of vehicle 2				
service ID	name of service	date	cost	caretakeld
7	malowanie	2025-05-14	9050	2
9	wymiana przedniej szyby	2025-06-11	800	2

#### 4.4. Sortowanie chronologiczne

Warto wspomnieć, że w wielu miejscach stosowana jest tabela „sortable table”. Umożliwia to sortowanie po dowolnej kolumnie. Najważniejszą z kolumn, po których można sortować jest data – kliknięcie „date” w dowolnej tabeli powoduje posortowanie chronologicznym.

Service history of vehicle 1				
service ID	name of service	date ▲	cost	caretakeld
1	wymiana rozrządu na zamiennik	2025-01-01	1000	1
8	regeneracja rozrządu	2025-02-05	1500	1
2	naprawa koła przedniego	2025-06-06	2000	1
5	wymiana rozrządu na oryginalnych częściach	2025-06-20	5680	1



## 5. Wnioski

Projekt pozwolił na praktyczne wykorzystanie wiedzy z zakresu tworzenia aplikacji bazodanowych. Stworzenie kompletnej aplikacji umożliwiło zapoznanie się z:

- pełnym cyklem projektowania aplikacji (od analizy do wdrożenia),
- integracją różnych technologii (np. .NET, SQL, React),
- organizacją warstwowej architektury,
- obsługą API i dokumentowaniem go z użyciem Swaggera,
- dobrymi praktykami projektowania baz danych i interfejsów użytkownika.

Projekt realizuje założony cel - stworzenie aplikacji wspierającej zarządzanie flotą pojazdów w firmie. Dzięki wykorzystaniu ostrożnie wybranych technologii system jest:

- Modułowy – łatwo rozwijać o nowe funkcjonalności,
- Przenośny – działa zarówno lokalnie, jak i na serwerach produkcyjnych,
- Czytelny zarówno pod względem kodu, jak i struktury API,
- przyjazny dla użytkownika i intuicyjny

Zaprojektowana architektura umożliwia dalszą rozbudowę, na przykład dodanie możliwości zarządzania całą siecią placówek, a nie tylko jedną.