Sprawozdanie z realizacji .NET APP

1. Wprowadzenie

Celem zadania było stworzenie aplikacji webowej w technologii .NET umożliwiającej zarządzanie danymi dotyczącymi samochodów, modeli, marek oraz kontraktów. Aplikacja pozwala na dodawanie, usuwanie oraz pobieranie informacji o dostępnych pojazdach i ich powiązaniach poprzez API.

2. Technologia i narzędzia

- ASP.NET Core 9.0 (Web API + MVC)
- Entity Framework Core (EF Core)
- InMemory Database (do testów)
- JSON jako format wymiany danych
- Swagger do testowania API
- CSS dla stylizacji

3. Struktura projektu

Projekt został podzielony na następujące warstwy:

- **Models** – Definicje encji: Car, Model, Marka, Kontrakt
- **Controllers** – Kontrolery obsługujące żądania HTTP
- **Data** – Konfiguracja DbContext
- **ModelsDTO** – Modele do transferu danych
- **Views** – Widoki oparte na Razor Pages

4. Przykład modelu danych Car

```
public class Car
{
    5 references
    public int Id { get; set; }
    4 references
    public string LicensePlate { get; set; } = string.Empty;
    5 references
    public int ModelId { get; set; }
    2 references
    public Model Model { get; set; } = null!;
    4 references
    public int Year { get; set; }
}
```

Opis pól:

- **Id** *(int)* **-** Unikalny identyfikator samochodu.
- **LicensePlate** *(string)* **-** Numer rejestracyjny pojazdu.
- **ModelId** *(int)* **-** Identyfikator modelu pojazdu (klucz obcy do tabeli Model).
- **Model** *(Model)* **-** Obiekt reprezentujący powiązany model (referencja do encji Model).
- **Year** *(int)* **-** Rok produkcji pojazdu.

5. Komunikacja z api

```
[HttpGet]
2 references
public async Task<IActionResult> Index()
{
    List<Car> carsList = new();
    List<Model> modelsList = new();

    HttpResponseMessage response = await _client.GetAsync("Car");
    if (response.IsSuccessStatusCode)
    {
        string data = await response.Content.ReadAsStringAsync();
        carsList = JsonConvert.DeserializeObject<List<Car>>(data) ?? new List<Car>();
    }
    HttpResponseMessage response1 = await _client.GetAsync("Model");
    if (response1.IsSuccessStatusCode)
    {
        string data = await response1.Content.ReadAsStringAsync();
        modelsList = JsonConvert.DeserializeObject<List<Model>>(data) ?? new List<Model>();
    }

    ViewBag.CarsList = carsList;
    ViewBag.ModelsList = modelsList;

    return View();
}
```

Kod przedstawia metodę kontrolera w ASP.NET Core, która obsługuje żądanie **GET** ([HttpGet]). Pobiera ona asynchronicznie dane o **samochodach** i **modelach** z dwóch endpointów API (Car i Model) za pomocą HttpClient. Otrzymane dane są deserializowane z formatu JSON do list obiektów (List<Car> i List<Model>) i przechowywane w ViewBag, aby można było je wykorzystać w widoku.

```
// Akcja do dodawania nowego samochodu
[HttpPost]
0 references
public async Task<IActionResult> AddCar(CarDto car)
{
    if (ModelState.IsValid)
    {
        var response = await _client.PostAsJsonAsync("Car", c
        if (response.IsSuccessStatusCode)
        {
            return RedirectToAction(nameof(Index));
        }
    }
    return View("Index");
}

// Akcja do usuwania samochodu
[HttpPost]
0 references
public async Task<IActionResult> DeleteCar(int id)
{
    var response = await _client.DeleteAsync($"Car/{id}");
    if (response.IsSuccessStatusCode)
    {
        return RedirectToAction(nameof(Index));
    }
    return View("Index");
}
```

Kod przedstawia dwie metody kontrolera w ASP.NET Core, obsługujące dodawanie i usuwanie samochodów za pomocą żądań **POST**.

6. Interfejs

Interfejs aplikacji został zbudowany w oparciu o Razor Pages i HTML z wykorzystaniem CSS do stylizacji. Składa się z kilku kluczowych elementów:

1. Formularz dodawania samochodu

Formularz pozwala użytkownikowi na wprowadzenie danych nowego samochodu i przesłanie ich do API.

```
<!-- Formularz dodawania nowego samochodu -->
<h3>Add a New Car</h3>
<form asp-action="AddCar" method="post">
    <div>
        <label for="LicensePlate">License Plate:</label>
        <input type="text" id="LicensePlate" name="LicensePlate" required />
    </div>
    <div>
        <label for="ModelId">Model:</label>
    <select name="ModelId" id="ModelId" required>
        @if (modelsList != null && modelsList.Any())
        {
            foreach (var model1 in modelsList)
            {
                <option value="@model1.Id">@model1.Name</option>
            }
        }
        else
        {
            <option value="">No models available</option>
        }
    </select>
    </div>
    <div>
        <label for="Year">Year:</label>
        <input type="number" id="Year" name="Year" required />
    </div>
    <button type="submit">Add Car</button>
</form>
```

Opis:

- Formularz ma **trzy pola**: LicensePlate (numer rejestracyjny), Name (nazwa modelu), Year (rok produkcji).
- asp-action="AddCar" – wskazuje, że formularz zostanie przesłany do kontrolera CarsController do metody AddCar.
- **Wymagane pola** (required) zapobiegają wysyłaniu pustych danych.

2. Tabela wyświetlająca listę samochodów

Tabela prezentuje pobrane dane z API, wyświetlając numer rejestracyjny, model i rok produkcji każdego samochodu.

```
<tbody>
    @if (carsList != null && carsList.Any())
    {
        foreach (var car in carsList)
        {
            <tr>
                <td>@car.Id</td>
                <td>@car.LicensePlate</td>
                <td>@car.Model.Mark.Name</td>
                <td>@car.Model.Name</td>
                <td>@car.Year</td>
                <td>
                    <form asp-action="DeleteCar" method="post" style="display:inline;">
                        <input type="hidden" name="id" value="@car.Id" />
                        <button type="submit" onclick="return confirm('Are you sure you want to delete this car?');">Delete</button>
                    </form>
                </td>
            </tr>
        }
    }
    else
    {
        <tr>No cars</tr>
    }
```

Opis:

- Każdy samochód jest prezentowany w osobnym wierszu.
- Kolumny **"Numer rejestracyjny"**, **"Marka"**, **"Model"** i **"Rok"** wyświetlają dane z listy samochodów.
- W ostatniej kolumnie znajduje się **przycisk "Usuń"**, który umożliwia usunięcie danego auta z bazy.