2025

Projekt Technologie Obiektowe

KONSOLOWA WYPOŻYCZALNIA SAMOCHODÓW

MICHAŁ BOHUN

Spis treści

1. Cel projektu

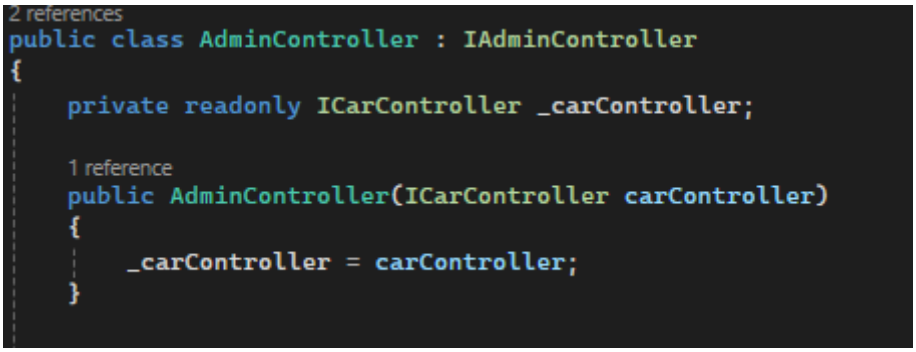
Celem projektu jest stworzenie konsolowej aplikacji do zarządzania wypożyczalnią samochodów. System umożliwia użytkownikom logowanie, przeglądanie dostępnych pojazdów, wynajem aut, a administratorom – zarządzanie flotą pojazdów.

Wzorce architektoniczne: MVC (Model-View-Controller)

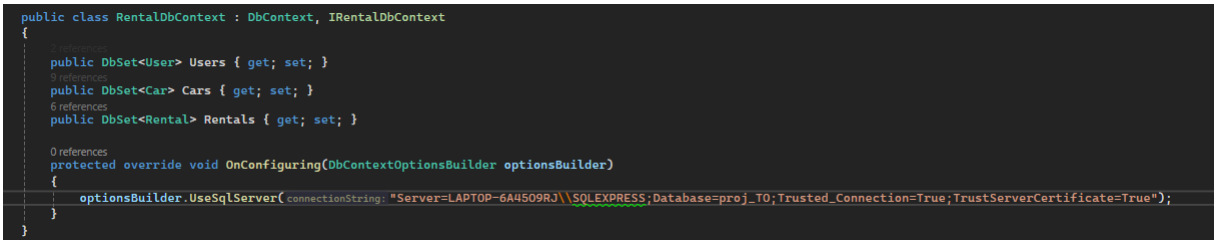
- Model – odpowiada on za przechowywanie danych i ich reprezentację
- View – jak sama nazwa wskazuje odpowiada on za przechowywanie widoków, czyli wszystko to co wyświetla się nam na pulpicie (w moim przypadku czysto w konsoli)
- Controller – przetwarza dane wejściowe, obsługuje główną logikę aplikacji i komunikuje się z modelem i widokami.

Wzorce projektowe:

Dependency Injection – kontrolery nie tworzą samodzielnie instancji klas, lecz otrzymują je przez konstruktor



Unit of Work i Repository - RentalDbContext (dziedziczący po DbContext z Entity Framework) pełni rolę Unit of Work, zarządzając transakcjami i śledzeniem zmian.



Fasada - AdminPanel() w AdminController oraz UserPanel() w UserController upraszczają dostęp do złożonych operacji (np. zarządzanie samochodami).

```
2 references
public void AdminPanel(IRentalDbContext db, IUserView view)
{
    while (true)
    {
        Console.WriteLine("1. Wyświetl dostępne auta\n2. Dodaj pojazd\n3. Usuń pojazd\n4. Edytuj pojazd\n5. Wyjście");
        string choice = Console.ReadLine();
        switch (choice)
        {
            case "1":
                Console.Clear();
                _carController.ShowAvailableCars(db, view);
                break;
            case "2":
                Console.Clear();
                _carController.AddCar(db, view);
                break;
            case "3":
                Console.Clear();
                _carController.RemoveCar(db, view);
                break;
            case "4":
                Console.Clear();
                _carController.EditCar(db, view);
                break;
            case "5":
                return;
        }
    }
}
```

```
public void UserPanel(IRentalDbContext db, User user, IUserController userController, ICarController carController, IUserView userView)
{
    while (true)
    {
        Console.WriteLine("1. Wyświetl dostępne auta\n2. Wynajmij auto\n3. Moje wynajmy\n4. Wyjście");
        string choice = Console.ReadLine();
        switch (choice)
        {
            case "1":
                Console.Clear();
                _carController.ShowAvailableCars(db, userView);
                break;
            case "2":
                Console.Clear();
                userController.RentCar(db, user, userView, carController);
                break;
            case "3":
                Console.Clear();
                ShowUserRentals(db, user, userView);
                break;
            case "4":
                return;
        }
    }
}
```

Strategia - Interfejs IUserView definiuje metody komunikacji z użytkownikiem (np. DisplayAvailableCars()). ConsoleUserView to konkretna strategia realizująca te metody.

```
1 reference
public class ConsoleUserView : IUserView
{
    2 references
    public string GetUsername()
    {
        Console.Write("Podaj login: ");
        return Console.ReadLine();
    }

    2 references
    public string GetPassword()
    {
        Console.Write("Podaj hasło: ");
        return Console.ReadLine();
    }

    2 references
    public void DisplayLoginFailed()
    {
        Console.WriteLine("Niepoprawne dane logowania!");
        Console.Clear();
        Thread.Sleep(millisecondsTimeout: 1000);
        Console.Clear();
    }
```

2. Baza danych i struktury danych

a) Baza danych i model danych

Projekt wykorzystuje **Entity Framework Core** do obsługi bazy danych SQL Server. Definiuje trzy główne tabele:

- **User** – przechowuje dane użytkowników (login, hasło, uprawnienia administratora).
- **Car** – reprezentuje pojazdy dostępne w wypożyczalni (marka, model, moc, cena wynajmu, status wynajmu).
- **Rental** – przechowuje informacje o wynajmach (klient, auto, data rozpoczęcia, długość wynajmu).

RentalDbContext to klasa, która zarządza połączeniem z bazą danych i operacjami na powyższych tabelach.

b) Kontrolery

System został podzielony na **moduły kontrolerów**, które zarządzają różnymi aspektami aplikacji:

1. **UserController**
    - Obsługuje logowanie użytkowników.
    - Pozwala użytkownikowi zobaczyć jego wynajmy.
    - Umożliwia wynajem pojazdów.
    - Udostępnia menu użytkownika z odpowiednimi opcjami.
2. **CarController**
    - Wyświetla dostępne pojazdy.
    - Pozwala administratorowi dodawać, usuwać i edytować samochody.
    - Aktualizuje status wynajmowanych samochodów (np. zwalnia auto po zakończeniu wynajmu).
3. **AdminController**
    - Umożliwia zarządzanie flotą pojazdów (dodawanie, usuwanie, edycja).
    - Zapewnia dostęp do panelu administracyjnego.

Interakcja z użytkownikiem (widoki)

Aplikacja działa w trybie konsolowym. Do interakcji z użytkownikiem służy klasa ConsoleUserView, która:

- Pobiera dane logowania.
- Wyświetla listę dostępnych aut.
- Informuje użytkownika o powodzeniu lub niepowodzeniu operacji.

3. Działanie aplikacji

1. **Logowanie**

    - Użytkownik podaje login i hasło.
    - System sprawdza poprawność danych w bazie.
    - Jeśli dane są błędne, prosi o ponowne wpisanie.

2. **Funkcje użytkownika**

    - Może przeglądać dostępne samochody.
    - Może wynająć auto, podając jego ID i czas wynajmu.
    - Może sprawdzić swoje aktualne wynajmy.

3. **Funkcje administratora**

    - Może dodawać, edytować i usuwać pojazdy.
    - Może przeglądać dostępne pojazdy.

4. **Zarządzanie wynajmami**

    - Po upływie okresu wynajmu system automatycznie zwalnia pojazd (zmienia jego status na dostępny).
    - Przy wynajmie sprawdzana jest dostępność pojazdu.