**Dokumentacja techniczna projektu**

**Projekt: System zarządzania sklepem samochodowym online**

Krótki opis działań projektu

Projekt jest aplikacją internetową dla sklepu internetowego, w którym użytkownicy mogą oglądać samochody, składać zamówienia i nimi zarządzać. System posiada sekcje służące do zarządzania kategoriami, pojazdami, zamówieniami i użytkownikami. Administrator może dodawać, edytować i usuwać samochody, kategorie i zamówienia. Użytkownicy mogą przeglądać pojazdy, dodawać je do koszyka i składać zamówienia.

Autor: Maksym Krytskyi

**Specyfikacja zastosowanych technologii**

* .NET: C# .NET 8
* Baza danych: MS SQL Server (lub SQLite)
* ORM: rdzeń Entity Framework
* Interfejs: Razor Pages
* Swagger: Do dokumentacji API
* UI: Używane są standardowe znaczniki i style CSS

**Instrukcje dotyczące pierwszego uruchomienia projektu**

* Sklonuj repozytorium projektu.
* Otwórz projekt w programie Visual Studio (lub innym edytorze obsługującym platformę .NET 8).
* Upewnij się, że masz zainstalowany program MS SQL Server lub SQLite. Jeśli używasz programu SQL Server, sprawdź ciąg połączenia w pliku appsettings.json.

Uruchom polecenie, aby dodać migracje:

*dotnet ef migrations add InitialCreate*

Zastosuj migrację:

*dotnet ef database update*

Uruchom projekt za pomocą polecenia:

*dotnet run*

Aby pracować z aplikacją, przejdź do https://localhost:5001.

**Opis struktury projektu**

Projekt składa się z następujących głównych komponentów:

* Modele: modele danych (np. samochód, kategoria, zamówienie) i ich relacje.
* Kontrolery: Kontrolery obsługujące żądania użytkowników.
* Widoki: strony Razor służące do wyświetlania danych użytkownikom.
* Dane: Kontekst bazy danych i migracje.

**Modele**

Category

* Opis: Model przedstawiający kategorię samochodów.
* Pola:
* Id: Unikalny identyfikator kategorii.
* Nazwa: Nazwa kategorii (wymagana).
* Opis: Opis kategorii.
* Car: Zbiór samochodów należących do kategorii .

Samochód

* Opis: Model przedstawiający samochód.
* Pola:
* Id: Unikalny identyfikator pojazdu.
* Nazwa: Nazwa pojazdu (wymagane).
* Opis: Opis pojazdu.
* Cena: Cena samochodu (wymagane).
* Ilość: Liczba dostępnych samochodów (wymagane).
* CategoryId: ID kategorii (wymagane).
* Category: Związek z kategorią pojazdu.

Order

* Opis: Model służący do reprezentacji zamówienia.
* Pola:
* Id: Unikalny identyfikator zamówienia.
* UserId: Identyfikator użytkownika, który złożył zamówienie.
* Status: Status zamówienia (np. „W trakcie realizacji”, „Wysłane”).
* OrderCars: Kolekcja samochodów na zamówienie.

OrderCar

* Opis: Model umożliwiający połączenie pojazdu z zamówieniem.
* Pola:
* Id: Unikalny identyfikator rekordu.
* OrderId: Identyfikator zamówienia.
* CarId: Identyfikator pojazdu.
* Ilość: Ilość samochodów w zamówieniu.
* Order: Link do zamówienia.
* Car: Komunikacja z samochodem.

**System użytkownika**

Role w systemie:

1. Administrator: może dodawać, edytować i usuwać kategorie, pojazdy i zamówienia.
2. Użytkownik: Może przeglądać, zamawiać i zarządzać pojazdami.

Rejestracja i autoryzacja użytkownika:

1. Do pracy z użytkownikami wykorzystywany jest standardowy system tożsamości.
2. Po rejestracji użytkownicy mogą się zalogować i dokonać transakcji zamówień.

**Charakterystyka funkcjonalności**

1. Wyświetlanie i zarządzanie kategoriami pojazdów: Użytkownicy i administratorzy mogą wyświetlać listę kategorii, a także je dodawać, edytować i usuwać.
2. Wyświetlanie i zarządzanie pojazdami: Administratorzy mogą dodawać, edytować i usuwać pojazdy, a także przeglądać je według kategorii.
3. Składanie i zarządzanie zamówieniami: Użytkownicy mogą dodawać pojazdy do koszyka, a następnie składać zamówienia. Administratorzy mogą zarządzać zamówieniami (np. zmieniać status).
4. Autoryzacja i uwierzytelnianie użytkownika: Wszyscy użytkownicy muszą być zarejestrowani i upoważnieni do wykonywania czynności związanych z zamówieniami.

**Podsumowanie**

Podczas pracy nad projektem zdobyłem cenne doświadczenie w tworzeniu aplikacji internetowych przy użyciu technologii C# i .NET 8, a także pracy z bazami danych, takimi jak MS SQL Server. Udało mi się zaimplementować pełną funkcjonalność CRUD do zarządzania danymi, w tym kategoriami, samochodami i zamówieniami.

W trakcie pracy nauczyłem się:

* Projektować i tworzyć modele danych z relacjami między nimi (jeden do wielu, wiele do wielu), co pozwoliło na efektywne operowanie na bazie danych.
* Korzystać z Entity Framework Core do interakcji z bazą danych, co znacznie ułatwiło operacje dodawania, aktualizowania i usuwania danych.
* Opanować pracę z uwierzytelnianiem i autoryzacją użytkowników w .NET, w tym konfigurację Identity do zarządzania użytkownikami i rolami.
* Tworzyć API obsługujące różne metody HTTP (GET, POST, PUT, DELETE) oraz integrować je z frontendem.
* Praktycznie stosować technologie do budowy usług RESTful oraz wykorzystywać dokumentację Swagger do testowania API.

Projekt pozwolił mi pogłębić wiedzę na temat tworzenia aplikacji internetowych oraz logiki serwerowej, poprawić umiejętności pracy z bazami danych, a także opanować ważne aspekty takie jak uwierzytelnianie, autoryzacja i ochrona danych użytkowników.

Projekt stanowił doskonały punkt wyjścia do dalszego rozwoju w dziedzinie backendu, a zdobyte doświadczenie będzie przydatne przy pracy nad realnymi projektami.