# Dokumentacja Projektu Wypożyczalni Samochodów

**Tytuł projektu**

**Wypożyczalnia Samochodowa**

**Opis działania**

**Projekt "Wypożyczalnia Samochodów" to aplikacja internetowa umożliwiająca zarządzanie wynajmem samochodów. System pozwala użytkownikom na przeglądanie oferty dostępnych pojazdów, dokonywanie rezerwacji, oraz zarządzanie swoimi rezerwacjami. Administratorzy mają możliwość pełnego zarządzania flotą samochodów, typami, producentami, oraz rezerwacjami. Aplikacja wykorzystuje technologię .NET 8.0, framework ASP.NET Core MVC oraz bazę danych Microsoft SQL Server. System obsługuje również autoryzację i uwierzytelnianie użytkowników poprzez Identity Framework, co zapewnia odpowiednie poziomy dostępu do różnych funkcji aplikacji.**

**Autorzy**

**Marcin Karaś**

**Miłosz Przybył**

## 1. Specyfikacja wykorzystanych technologii

* **Platforma:** .NET 8.0
* **Framework:** ASP.NET Core MVC
* **Baza danych:** Microsoft SQL Server
* **ORM:** Entity Framework Core 8.0.10
* **Język programowania:** C#
* **Narzędzia:** Visual Studio 2022
* **Inne technologie:** Bootstrap (frontend), Identity Framework (autoryzacja i uwierzytelnianie)

## 2. Instrukcje pierwszego uruchomienia projektu

1. **Klonowanie repozytorium:** Pobierz kod z repozytorium za pomocą komendy:
2. git clone <https://github.com/Whiperr/ProjektWypozyczalniaFinal.git>
3. lub pobierz I wypakuj projekt w twoim folderze repos i uruchom plik .sln
4. **Przygotowanie bazy danych:**
   * Otwórz Konsolę Menedżera Pakietów (Package Manager Console).
   * Wykonaj migracje za pomocą polecenia:
   * Update-Database
5. **Uruchomienie projektu:**
   * Uruchom aplikację klawiszem F5 lub Ctrl + F5.

## 3. Opis struktury projektu

* **Controllers:** Zawiera kontrolery obsługujące logikę aplikacji oraz akcje CRUD dla modeli.
* **Models:** Definicje modeli danych oraz ich właściwości.
* **Views:** Pliki widoków (Razor) odpowiedzialne za renderowanie interfejsu użytkownika.
* **Data:** Konfiguracja bazy danych i kontekst aplikacji (ApplicationDbContext).
* **Areas:** Sekcje aplikacji podzielone na moduły, np. Admin,User.
* **wwwroot:** Zasoby statyczne (CSS, JS, obrazy).

## 4. Modele danych

**Car**

Reprezentuje samochód w systemie.

| **Pole** | **Typ** | **Opis** |
| --- | --- | --- |
| Id | int | Unikalny identyfikator samochodu. |
| RegistrationNumber | string | Numer rejestracyjny samochodu. |
| ModelId | int | Identyfikator modelu samochodu. |
| Model | Model | Relacja do modelu samochodu. |
| TypeId | int | Identyfikator typu samochodu. |
| Type | CarType | Relacja do typu samochodu. |
| Reservations | ICollection <Reservation> | Lista rezerwacji związanych z samochodem. |

**CarType**

Reprezentuje typ samochodu (np. SUV, sedan).

| **Pole** | **Typ** | **Opis** |
| --- | --- | --- |
| Id | int | Unikalny identyfikator typu. |
| Code | string | Kod typu samochodu. |
| Description | string | Opis typu samochodu. |

**Manufacturer**

Reprezentuje producenta samochodów.

| **Pole** | **Typ** | **Opis** |
| --- | --- | --- |
| Id | int | Unikalny identyfikator producenta. |
| Name | string | Nazwa producenta. |
| Details | string | Szczegóły dotyczące producenta. |
| Models | ICollection <Model> | Lista modeli powiązanych z producentem. |

**Model**

Reprezentuje model samochodu.

| **Pole** | **Typ** | **Opis** |
| --- | --- | --- |
| Id | int | Unikalny identyfikator modelu. |
| Name | string | Nazwa modelu. |
| PricePerDay | decimal | Cena wynajmu za dzień. |
| ManufacturerId | int | Identyfikator producenta. |
| Manufacturer | Manufacturer | Relacja do producenta. |
| Cars | ICollection <Car> | Lista samochodów tego modelu. |

**Reservation**

Reprezentuje rezerwację w systemie.

| **Pole** | **Typ** | **Opis** |
| --- | --- | --- |
| Id | int | Unikalny identyfikator rezerwacji. |
| StartDate | DateTime | Data rozpoczęcia rezerwacji. |
| EndDate | DateTime | Data zakończenia rezerwacji. |
| StatusId | int | Identyfikator statusu. |
| Status | Status | Relacja do statusu. |
| CarId | int | Identyfikator samochodu. |
| Car | Car | Relacja do samochodu. |
| UserId | string | Identyfikator użytkownika. |

**Status**

Reprezentuje status dostępności auta (np. Dostępny, Nie dostępny).

| **Pole** | **Typ** | **Opis** |
| --- | --- | --- |
| Id | int | Unikalny identyfikator statusu. |
| Code | string | Kod statusu. |
| Description | string | Opis statusu. |
| Reservations | ICollection<Reservation> | Lista rezerwacji. |

## 5. Kontrolery

**CarsController**

* Obsługuje standardowe operacje CRUD dla modelu Car.
* **Metody:**
  + Index() - lista samochodów.
  + Details(int id) - szczegóły samochodu.
  + Create() - formularz tworzenia.
  + Create(Car car) - zapisuje nowy samochód.
  + Edit(int id) - formularz edycji.
  + Edit(Car car) - zapisuje zmiany.
  + Delete(int id) - potwierdzenie usunięcia.
  + DeleteConfirmed(int id) - usuwa samochód.

**CarTypesController**

* Obsługuje operacje CRUD dla modelu CarType.

**ManufacturersController**

* Obsługuje operacje CRUD dla modelu Manufacturer.

**ModelsController**

* Obsługuje operacje CRUD dla modelu Model.

**StatusesController**

* Obsługuje operacje CRUD dla modelu Status.

**ReservationsController**

* Obsługuje operacje CRUD oraz dodatkowe operacje dla modelu Reservation:
  + Index(string? registrationNumber) - lista rezerwacji z filtrowaniem.
  + Details(int id) - szczegóły rezerwacji.
  + Create() - formularz tworzenia rezerwacji.
  + Create(Reservation reservation) - zapisuje nową rezerwację.
  + Edit(int id) - formularz edycji rezerwacji.
  + Edit(Reservation reservation) - zapisuje zmiany.
  + Delete(int id) - potwierdzenie usunięcia.
  + DeleteConfirmed(int id) - usuwa rezerwację.

ReservationsController dla Usera

1. **Widok rezerwacji (Index)**:
   * Pokazuje listę rezerwacji przypisanych do aktualnie zalogowanego użytkownika.
   * Używa filtracji po identyfikatorze użytkownika, zapewniając, że użytkownik zobaczy tylko swoje rezerwacje.
2. **Szczegóły rezerwacji (Details)**:
   * Pozwala na podgląd szczegółów pojedynczej rezerwacji.
   * Użytkownik ma dostęp tylko do swoich rezerwacji; próba wyświetlenia rezerwacji należącej do innego użytkownika skutkuje odpowiedzią 403 (Forbidden).
3. **Tworzenie rezerwacji (Create)**:
   * Umożliwia użytkownikowi stworzenie nowej rezerwacji dla wybranego samochodu.
   * Formularz rezerwacji zawiera dane dotyczące pojazdu, statusu oraz daty rozpoczęcia i zakończenia rezerwacji.
   * Przed zapisaniem rezerwacji, identyfikator użytkownika jest automatycznie przypisywany do rezerwacji.
4. **Edycja rezerwacji (Edit)**:
   * Pozwala na edytowanie istniejącej rezerwacji przez użytkownika.
   * Użytkownik może edytować tylko swoje rezerwacje. Próba edycji rezerwacji innego użytkownika skutkuje odpowiedzią 403 (Forbidden).
5. **Usuwanie rezerwacji (Delete)**:
   * Umożliwia usunięcie rezerwacji.
   * Podobnie jak w przypadku edycji, użytkownik może usunąć tylko swoje rezerwacje.
6. **Ochrona przed nieautoryzowanym dostępem**:
   * W kontrolerze zastosowano mechanizm sprawdzania, czy użytkownik ma odpowiedni dostęp do edytowania, przeglądania lub usuwania rezerwacji. W przypadku próby dostępu do rezerwacji innego użytkownika, zostaje zwrócone zabezpieczenie 403 (Forbidden).

**Funkcje zabezpieczeń i walidacji:**

* Każda akcja kontrolera, która umożliwia modyfikowanie danych (np. tworzenie, edytowanie, usuwanie rezerwacji), zawiera odpowiednią walidację, aby zapobiec zmianom danych przez użytkowników, którzy nie mają do nich dostępu.
* Walidacja jest przeprowadzana na poziomie użytkownika oraz danych rezerwacji (np. daty).

## 6. System użytkowników

**Role w systemie:**

* **Administrator:** Dostęp do panelu admina (obsługa CRUD wszystkich modeli).
* **Zalogowany użytkownik:** Może rezerwować samochody.
* **Gość:** Może przeglądać ofertę, ale bez możliwości rezerwacji.

**Przypisywanie ról:**

* Role przypisywane są za pomocą Identity Framework.

**Powiązania:**

* **Użytkownicy:** Każda rezerwacja jest powiązana z zalogowanym użytkownikiem.
* **Globalne dane:** Statusy, typy samochodów i producenci są wspólne dla wszystkich użytkowników.

## 7. Najciekawsze funkcjonalności

1. **Filtrowanie rezerwacji po numerze rejestracyjnym dla Admina:**
   * Metoda Index w kontrolerze ReservationsController umożliwia wyszukiwanie rezerwacji po numerze rejestracyjnym samochodu.
2. **Walidacja dat rezerwacji:**
   * W metodzie Create kontrolera ReservationsController sprawdzane jest, czy data początkowa nie jest późniejsza niż końcowa.
3. **Przypisywanie rezerwacji do użytkownika:**
   * W metodzie Create automatycznie przypisywany jest identyfikator zalogowanego użytkownika (User.Identity.Name).
   * Inni użytkownicy nie mogą sprawdzić czyiś rezerwacji.
4. **Automatycznie przypisanie roli po zarejestrowaniu**
   * Użytkownik dostaje rolę User po rejestracji, co odblokwywuje mu dostęp do możliwości rezerwacji