

Wypożyczalnia Samochodów

Autorzy: Daniel Boiński, Magdalena Sułkowska.

Krótki opis działania projektu:

Aplikacja internetowa "Wypożyczalnia samochodów" pozwala użytkownikom na przeglądanie dostępnych samochodów oraz cenników wynajmu dla poszczególnych pojazdów. Po zarejestrowaniu lub zalogowaniu się, użytkownik zyskuje możliwość przeglądania dostępnych terminów wynajmu samochodów, dokonywania rezerwacji na wybrany okres oraz przeglądania historii swoich rezerwacji. Aplikacja oferuje także panel administratora, w którym możliwe jest dodawanie nowych samochodów do floty, wyłączanie pojazdów z użytkowania oraz przeglądanie rezerwacji dokonanych przez klientów.

1. Specyfikacja wykorzystanych technologii

1.1 Wersja .Net

Projekt został zrealizowany z wykorzystaniem **.NET 8**, najnowszej stabilnej wersji platformy, zapewniającej wsparcie dla najnowszych funkcji języka C# oraz ulepszoną wydajność w środowiskach backendowych.

1.2 Baza danych

SQLite została wykorzystana jako baza danych. Jest to lekka, relacyjna baza danych w formie pliku, która nie wymaga instalacji serwera bazy danych, co czyni ją idealnym rozwiązaniem dla małych projektów oraz środowisk testowych.

1.3 Frameworki i biblioteki

Projekt korzysta z poniższych frameworków i bibliotek:

- **Entity Framework Core**
 - Wersja: najnowsza kompatybilna z .NET 8
 - Służy do mapowania obiektowo-relacyjnego (ORM), umożliwiającego zarządzanie bazą danych za pomocą obiektów C#.
 - Wykorzystano funkcje takie jak migracje bazy danych oraz linq do realizacji zapytań.

- **Stripe API**
 - Służy do obsługi płatności online w systemie rezerwacyjnym.
 - Zaimplementowano tworzenie sesji płatności oraz przekierowywanie użytkownika do interfejsu Stripe w celu opłacenia rezerwacji.
 - Przykładowe karty testowe zostały wykorzystane podczas rozwoju systemu.
- **SendGrid API**
 - Biblioteka umożliwia wysyłanie e-maili transakcyjnych, takich jak potwierdzenia rezerwacji lub powiadomienia o zakończeniu wynajmu.
 - Wysyłanie wiadomości zostało skonfigurowane w oparciu o klucze API w pliku appsettings.json.
- **Leaflet.js**
 - Wersja: 1.9.4
 - Otwarta biblioteka JavaScript służąca do obsługi interaktywnych map.
 - Wykorzystano do wyświetlania lokalizacji dostarczenia auta, a także wyboru miejsca na mapie podczas tworzenia rezerwacji.

1.4. Narzędzia pomocnicze

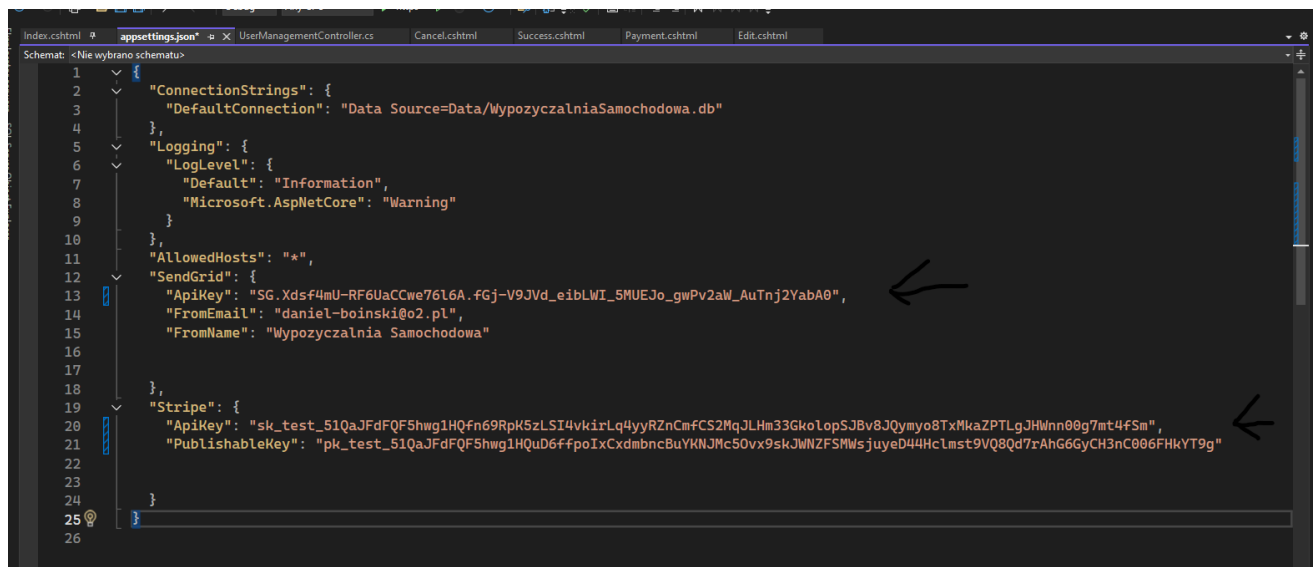
- **Visual Studio 2022**
 - Środowisko programistyczne używane do tworzenia, debugowania i uruchamiania projektu.
- **Git/GitHub**
 - System kontroli wersji używany do zarządzania kodem oraz współpracy w zespole. Projekt został opublikowany na GitHub w celu umożliwienia sprawdzenia kodu przez wykładowcę.

2. Instrukcja pierwszego uruchomienia projektu

- Zainstaluj środowisko .NET 8 oraz Visual Studio 2022.
- Pobierz projekt z GitHuba i otwórz go w Visual Studio.
- W terminalu Visual Studio uruchom polecenie **update-database**, aby przygotować bazę danych.
- W pliku appsettings.json w miejsce x wpisz klucze api:

SendGrid
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

Stripe
sk_test_51QaJFdFQF5hwg1HQfn69RpK5zLSI4vkiRlQ4yyRZnCmFCs2MqJLHm33GkolopSJBv8JQymyo8TxMkaZPTLgJHwnn00g7mt4fSm
pk_test_51QaJFdFQF5hwg1HQDuD6ffpoIxCxdmbncBuYKNJMc50vx9skJWNZFSMwsjuyeD44Hclmst9VQ8Qd7rAhG6GyCH3nC006FHkYT9g



- Możemy uruchomić projekt
- Dodatkowo, żeby móc otrzymać mail potwierdzający przykładowy wynajem zalecamy wykorzystać mail tymczasowy, lub jakiś do którego mamy realny dostęp.
- Dane administratora to: admin@admin.com hasło: Zaq123!
- Przy wybraniu płatności online podajemy testową kartę płatniczą stripe 4242 4242 4242 4242, reszta danych bez znaczenia.

3. Opis struktury projektu

Projekt "Wypożyczalnia Samochodowa" został zorganizowany w strukturę katalogów i plików zgodnie z najlepszymi praktykami ASP.NET Core MVC. Poniżej przedstawiono szczegółowy opis struktury projektu:

3.1 Root projektu

- **Program.cs**
Plik konfiguracji i uruchamiania aplikacji, w którym definiowane są usługi i middleware używane w projekcie.
- **appsettings.json**
Plik konfiguracyjny zawierający dane związane z połączeniem z bazą danych, kluczami API (SendGrid, Stripe) oraz innymi ustawieniami.

3.2 Folder Controllers

Folder zawiera kontrolery obsługujące logikę aplikacji oraz odpowiadające na żądania HTTP:

1. **AutoController.cs**
Odpowiada za zarządzanie pojazdami (CRUD operacje i blokowanie pojazdów).
2. **HomeController.cs**
Obsługuje podstawowe strony, takie jak strona główna i polityka prywatności.
3. **OfertaController.cs**
Obsługuje logikę dotyczącą ofert wynajmu samochodów.
4. **StripeController.cs**
Obsługuje logikę związaną z integracją płatności Stripe.
5. **UserManagementController.cs**
Obsługuje zarządzanie użytkownikami, w tym ich usuwanie i edycję.
6. **WynajecieController.cs**
Obsługuje logikę dotyczącą wynajmów, w tym CRUD operacje oraz filtrowanie.

3.3 Folder Data

1. **ApplicationDbContext.cs**
Główna klasa kontekstu Entity Framework, zarządzająca połączeniem z bazą danych oraz definiująca tabele (DbSety).
2. **Baza danych (*.db)**
Pliki SQLite przechowujące dane aplikacji.

3.4 Folder Models

Zawiera modele reprezentujące dane w aplikacji:

1. **Auto.cs**
Model reprezentujący pojazdy w systemie.
2. **ErrorViewModel.cs**
Model używany do obsługi błędów.
3. **Oferta.cs**
Model reprezentujący oferty wynajmu.
4. **PaymentViewModel.cs**
Model używany w procesie płatności Stripe.
5. **StripeOptions.cs**
Klasa pomocnicza do konfiguracji kluczy API Stripe.
6. **Uzytkownicy.cs**
Model rozszerzający tożsamość użytkownika w systemie.
7. **Wynajecie.cs**
Model reprezentujący wynajem samochodu.

3.5 Folder Services

1. **EmailService.cs**
Klasa obsługująca wysyłanie wiadomości e-mail za pomocą SendGrid.

3.6 Folder Views

Folder zawiera widoki Razor odpowiadające za prezentację danych. Podzielone są na subfoldery odpowiadające kontrolerom:

1. **Auto**
Widoki CRUD dla zarządzania samochodami (Index.cshtml, Create.cshtml, Edit.cshtml, Delete.cshtml, Block.cshtml).
2. **Home**
Widoki dla strony głównej i polityki prywatności (Index.cshtml, Privacy.cshtml).
3. **Oferta**
Widoki obsługujące zarządzanie ofertami.

4. **Stripe**

Widoki związane z płatnościami Stripe (Payment.cshtml, Success.cshtml, Cancel.cshtml).

5. **UserManagement**

Widoki umożliwiające zarządzanie użytkownikami (Index.cshtml, Details.cshtml, Delete.cshtml).

6. **Wynajęcie**

Widoki CRUD dla wynajmu samochodów (Index.cshtml, Create.cshtml, Edit.cshtml, Delete.cshtml, Details.cshtml).

7. **Shared**

Wspólne widoki, takie jak układ strony (_Layout.cshtml), obsługa błędów i skrypty walidacyjne.

3.7 Folder wwwroot

Zawiera pliki statyczne, takie jak CSS, JS i inne zasoby statyczne wymagane przez aplikację.

4 Modele danych

4.1 Model Auto

Opis modelu: Model reprezentuje samochody w systemie, które są dostępne do wynajęcia.

Rola modelu w projekcie: Model Auto przechowuje dane o samochodach, takie jak marka, model, silnik, cena, status dostępności oraz zdjęcie. Służy do zarządzania flotą pojazdów.

Lista pól:

- Autoid (*int*) - Identyfikator samochodu.
- Marka (*string*) - Marka samochodu.
- Model (*string*) - Model samochodu.
- Silnik (*string*) - Typ silnika samochodu.
- RokProdukcji (*int*) - Rok produkcji samochodu.
- Opis (*string*) - Opis samochodu.
- Cena (*int*) - Cena wynajmu za dobę.
- Status (*bool*) - Status dostępności samochodu (domyślnie true).
- Zdjecie (*string*) - Ścieżka do zdjęcia samochodu.
- Oferta (*virtual Oferta?*) - Powiązana oferta.

4.2 Model Wynajecie

Opis modelu: Model reprezentuje rezerwacje wynajmu samochodów przez użytkowników.

Rola modelu w projekcie: Przechowuje dane o wynajmie samochodu, w tym daty, użytkownika, cenę oraz metodę płatności.

Lista pól:

- Wynajecid (*int*) - Identyfikator wynajmu.
- Autoid (*int*) - Identyfikator wynajętego samochodu (relacja z modelem Auto).
- DataRozpoczenia (*DateTime*) - Data rozpoczęcia wynajmu.
- DataZakonczenia (*DateTime*) - Data zakończenia wynajmu.
- CenaCalkowita (*int*) - Całkowita cena wynajmu.
- UserId (*string*) - Identyfikator użytkownika (relacja z systemem ASP.NET Identity).
- User (*IdentityUser?*) - Powiązany użytkownik.
- Latitude (*double*) - Szerokość geograficzna dostawy.
- Longitude (*double*) - Długość geograficzna dostawy.
- Address (*string*) - Adres dostarczenia samochodu.
- PaymentMethod (*string*) - Metoda płatności.
-

Walidacje i ograniczenia:

- Latitude - Musi być między -90 a 90.
- Longitude - Musi być między -180 a 180.
- DataRozpoczenia i DataZakonczenia muszą być poprawne (walidowane w kontrolerze).

4.3 Model PaymentViewModel

Opis modelu: Model pomocniczy do przechowywania informacji o płatności.

Rola modelu w projekcie: Używany do obsługi płatności Stripe, przechowuje dane potrzebne do sesji płatności.

Lista pól:

- Amount (*int*) - Kwota do zapłaty.
- Autold (*int*) - Identyfikator samochodu.
- StartDate (*string*) - Data rozpoczęcia wynajmu.
- EndDate (*string*) - Data zakończenia wynajmu.

Walidacje i ograniczenia:

- Walidacje wykonywane są na etapie przetwarzania płatności w kontrolerze Stripe.

4.4 Model StripeOptions

Opis modelu: Model konfiguracyjny dla kluczy API Stripe.

Rola modelu w projekcie: Służy do przechowywania kluczy API Stripe w celu konfiguracji środowiska płatności.

Lista pól:

- ApiKey (*string*) - Klucz prywatny Stripe.
- PublishableKey (*string*) - Klucz publiczny Stripe.

Walidacje i ograniczenia:

- Klucze muszą być poprawnie ustawione w pliku appsettings.json.

4.5 Model Oferta

Opis modelu: Model pomocniczy dla obsługi ofert wynajmu.

Rola modelu w projekcie: Reprezentuje relację między ofertą a samochodem w bazie danych.

Lista pól:

- OfertId (*int*) - Identyfikator oferty.
- AutoId (*int*) - Identyfikator samochodu.
- Auto (*virtual Auto?*) - Powiązany samochód.

Walidacje i ograniczenia:

- Relacja Auto musi być zdefiniowana.

5. Kontrolery i ich metody

5.1 Kontroler Auto

1. Nazwa metody: Index

- **Dopuszczalne metody HTTP:** GET
- **Parametry:** Brak
- **Opis działania:** Wyświetla listę wszystkich samochodów dostępnych w systemie.
- **Zwracane dane:** Widok z listą samochodów.

2. Nazwa metody: Details

- **Dopuszczalne metody HTTP:** GET
- **Parametry:** id: int?
- **Opis działania:** Wyświetla szczegóły wybranego samochodu na podstawie jego ID.
- **Zwracane dane:** Widok z informacjami o samochodzie.

3. Nazwa metody: Create

- **Dopuszczalne metody HTTP:** GET, POST
- **Parametry:** Auto, IFormFile Zdjecie
- **Opis działania:** Dodaje nowy samochód do systemu, opcjonalnie ze zdjęciem.
- **Zwracane dane:** Po pomyślnym dodaniu samochodu, przekierowanie do *Index*. W przypadku błędów, zwracany jest widok formularza.

4. Nazwa metody: Edit

- **Dopuszczalne metody HTTP:** GET, POST
- **Parametry:** id: int, Auto, IFormFile? noweZdjecie
- **Opis działania:** Aktualizuje dane wybranego samochodu. Może także zmienić zdjęcie przypisane do samochodu.
- **Zwracane dane:** Po pomyślnym zapisaniu zmian, przekierowanie do *Index*. W przypadku błędów, widok formularza edycji.

5. Nazwa metody: Delete

- **Dopuszczalne metody HTTP:** GET, POST
- **Parametry:** id: int
- **Opis działania:** Usuwa samochód oraz jego zdjęcie z systemu.
- **Zwracane dane:** Po usunięciu, przekierowanie do *Index*.

6. Nazwa metody: Block

- **Dopuszczalne metody HTTP:** GET, POST
- **Parametry:** autold: int, dataRozpoczenia: DateTime, dataZakonczenia: DateTime
- **Opis działania:** Blokuje wybrany samochód na podany okres, uniemożliwiając jego wynajem w tym czasie.
- **Zwracane dane:** Po pomyślnym zapisaniu blokady, przekierowanie do *Index*.

5.2 Kontroler Oferta

1. Nazwa metody: Index

- **Dopuszczalne metody HTTP:** GET
- **Parametry:** marka: string, cenaOd: int?, cenaDo: int?, dataOd: DateTime?, dataDo: DateTime?, sortowanie: string
- **Opis działania:** Wyświetla listę ofert z możliwością filtrowania i sortowania.
- **Zwracane dane:** Widok z listą ofert wynajmu.

2. Nazwa metody: Details

- **Dopuszczalne metody HTTP:** GET
- **Parametry:** id: int?, autold: int?, cena: int?
- **Opis działania:** Wyświetla szczegóły wybranej oferty wynajmu na podstawie jej ID.
- **Zwracane dane:** Widok z informacjami o ofercie.

3. Nazwa metody: Create

- **Dopuszczalne metody HTTP:** GET, POST
- **Parametry:** Oferta
- **Opis działania:** Dodaje nową ofertę wynajmu samochodu.
- **Zwracane dane:** Po zapisaniu oferty, przekierowanie do *Index*.

4. Nazwa metody: Edit

- **Dopuszczalne metody HTTP:** GET, POST
- **Parametry:** id: int, Oferta
- **Opis działania:** Aktualizuje szczegóły istniejącej oferty wynajmu.
- **Zwracane dane:** Po zapisaniu zmian, przekierowanie do *Index*.

5. Nazwa metody: Delete

- **Dopuszczalne metody HTTP:** GET, POST
- **Parametry:** id: int

- **Opis działania:** Usuwa ofertę wynajmu samochodu.
- **Zwracane dane:** Po usunięciu, przekierowanie do *Index*.

5.3 Kontroler StripeController

1. Nazwa metody: Payment

- **Dopuszczalne metody HTTP:** GET
- **Parametry:** amount: int, autold: int, startDate: string, endDate: string
- **Opis działania:** Wyświetla formularz płatności online za wynajem samochodu.
- **Zwracane dane:** Widok formularza płatności.

2. Nazwa metody: CreateSession

- **Dopuszczalne metody HTTP:** POST
- **Parametry:** PaymentViewModel
- **Opis działania:** Tworzy sesję Stripe, inicjując proces płatności.
- **Zwracane dane:** JSON z informacjami o sesji Stripe.

3. Nazwa metody: Success

- **Dopuszczalne metody HTTP:** GET
- **Parametry:** autold: int, startDate: string, endDate: string, amount: int
- **Opis działania:** Potwierdza płatność i zapisuje szczegóły wynajmu.
- **Zwracane dane:** Widok potwierdzenia płatności.

4. Nazwa metody: Cancel

- **Dopuszczalne metody HTTP:** GET
- **Parametry:** Brak
- **Opis działania:** Informuje użytkownika o anulowaniu płatności.
- **Zwracane dane:** Widok anulowania płatności.

5.4 Kontroler UserManagementController

1. Nazwa metody: Index

- **Dopuszczalne metody HTTP:** GET
- **Parametry:** Brak
- **Opis działania:** Wyświetla listę wszystkich użytkowników systemu.
- **Zwracane dane:** Widok z listą użytkowników.

2. Nazwa metody: Details

- **Dopuszczalne metody HTTP:** GET
- **Parametry:** id: string
- **Opis działania:** Wyświetla szczegóły wybranego użytkownika.
- **Zwracane dane:** Widok z informacjami o użytkowniku.

3. Nazwa metody: Delete

- **Dopuszczalne metody HTTP:** GET, POST
- **Parametry:** id: string
- **Opis działania:** Usuwa użytkownika z systemu.
- **Zwracane dane:** Po usunięciu, przekierowanie do *Index*.

5.5 Kontroler Wynajecie

1. Nazwa metody: Index

- **Dopuszczalne metody HTTP:** GET
- **Parametry:** userEmail: string?
- **Opis działania:** Wyświetla historię wynajmów. Administratorzy widzą wszystkie wynajmy, użytkownicy tylko swoje.
- **Zwracane dane:** Widok z historią wynajmów.

2. Nazwa metody: Details

- **Dopuszczalne metody HTTP:** GET
- **Parametry:** id: int?
- **Opis działania:** Wyświetla szczegóły wybranego wynajmu.
- **Zwracane dane:** Widok z informacjami o wynajmie.

3. Nazwa metody: Create

- **Dopuszczalne metody HTTP:** GET, POST
- **Parametry:** autoId, dataRozpoczecia, dataZakonczenia, paymentMethod
- **Opis działania:** Tworzy nowy wynajem samochodu.
- **Zwracane dane:** Przekierowanie do *Index* lub formularz z błędami.

4. Nazwa metody: Edit

- **Dopuszczalne metody HTTP:** GET, POST
- **Parametry:** id: int, Wynajecie
- **Opis działania:** Edytuje szczegóły istniejącego wynajmu.
- **Zwracane dane:** Po zapisaniu zmian, przekierowanie do *Index*.

5. Nazwa metody: Delete

- **Dopuszczalne metody HTTP:** GET, POST
- **Parametry:** id: int

- **Opis działania:** Usuwa wybrany wynajem.
- **Zwracane dane:** Po usunięciu, przekierowanie do *Index*.

6. System użytkowników

Role w systemie:

1. **Administrator:** ma dostęp do wszystkich funkcji systemu, takich jak zarządzanie samochodami, ofertami, użytkownikami i wynajmami.
2. **Użytkownik:** ma dostęp do podstawowych funkcji systemu, takich jak przeglądanie ofert samochodów i dokonywanie wynajmu.

Jak nadać rolę użytkownikowi:

1. **Przy rejestracji:**
Domyślnie użytkownicy są przypisywani do roli "Użytkownik".
2. **Przez administratora:**
Administrator może edytować użytkownika i przypisać mu rolę w panelu zarządzania użytkownikami (*UserManagementController*).

Uprawnienia:

Zalogowani użytkownicy mogą:

- Przeglądać dostępne samochody oraz ich szczegóły.
- Rezerwować samochody na określony okres.
- Dokonywać płatności za wynajem samochodów.
- Przeglądać historię swoich wynajmów.
- Wybierać miejsce dostarczenia samochodu na mapie.

Administratorzy dodatkowo mogą:

- Zarządzać flotą samochodów (dodawać, edytować, usuwać, blokować).
- Przeglądać i edytować wszystkie wynajmy w systemie.
- Zarządzać użytkownikami (przeglądać listę, szczegóły i usuwać użytkowników).
- Tworzyć i edytować oferty wynajmu.
- Blokować samochody na określony czas.

Dane powiązane z użytkownikiem:

1. **Wynajęcia:**
Model *Wynajecie* zawiera pole *UserId*, które odnosi się do identyfikatora użytkownika w systemie.
 - Każdy użytkownik może mieć wiele wynajmów.

- Wynajęcia zawierają szczegóły, takie jak daty rozpoczęcia i zakończenia, cena całkowita, metoda płatności oraz lokalizacja dostarczenia samochodu.

2. Dane kontaktowe:

System wykorzystuje ASP.NET Identity, które domyślnie obsługuje dane takie jak:

- E-mail użytkownika.
 - Hasło (zaszyfrowane).
- Rozszerzenie modelu użytkownika (jeśli istnieje) może przechowywać dodatkowe dane, takie jak imię, nazwisko czy numer telefonu.

3. Historia rezerwacji:

Przechowywana w tabeli Wynajecie.

- Użytkownik może przeglądać swoją historię wynajmów w widoku *Wynajecie/Index*.

7. Charakterystyka najciekawszych funkcjonalności

1. Płatności za pomocą Stripe

Opis funkcjonalności:

System umożliwia realizację płatności online za wynajem samochodów przy użyciu Stripe. Użytkownicy mogą dokonywać płatności za pomocą karty kredytowej/debetowej, a integracja obsługuje przekierowanie użytkownika do bezpiecznego interfejsu Stripe.

Implementacja:

Stripe API zostało zintegrowane z aplikacją za pomocą biblioteki Stripe SDK. Klucz API Stripe jest przechowywany w pliku *appsettings.json* i ładowany w kontrolerze *StripeController*. Proces płatności rozpoczyna się w metodzie *CreateSession*, która tworzy sesję Stripe z parametrami płatności, takimi jak kwota, opis transakcji i adresy URL sukcesu/anulowania. Po zakończeniu płatności Stripe przekierowuje użytkownika na stronę sukcesu (*Success*) lub anulowania (*Cancel*), gdzie odpowiednie informacje są wyświetlane, a dane płatności zapisane w bazie.

Działanie:

Użytkownik wybiera opcję płatności online podczas rezerwacji. System generuje sesję Stripe, a użytkownik jest przekierowywany do bezpiecznej bramki płatności. Po zakończeniu płatności dane są zapisane w tabeli *Wynajecie*, a użytkownik otrzymuje potwierdzenie.

← wypożyczalnia TEST MODE

Wynajem samochodu

1600,00 zł

Zapłać za pomocą ➤ link

Lub zapłać kartą

E-mail

a@a.com

Informacje o karcie

4242 4242 4242 4242

VISA

12 / 35

123



Imię i nazwisko posiadacza karty

adam kowalski

Kraj lub region

Polska



☐ Bezpiecznie zapisuj moje dane w celu finalizacji jednym kliknięciem

Plać szybciej na wypożyczalnia i wszędzie tam, gdzie akceptowana jest usługa Link.

Zapłać



2. System rezerwacji aut z uwzględnieniem blokad

Opis funkcjonalności:

System umożliwia użytkownikom rezerwowanie samochodów na wybrane okresy, jednocześnie uwzględniając istniejące rezerwacje i blokady (np. samochód w serwisie).

Implementacja:

Rezerwacje są obsługiwane w kontrolerze *WynajecieController*. Podczas tworzenia rezerwacji aplikacja sprawdza dostępność samochodu w podanym okresie, porównując go z istniejącymi rezerwacjami i blokadami zapisanymi w tabeli *Wynajecie*. Blokad są tworzone przez administratora w kontrolerze *AutoController* metodą *Block*, która zapisuje w tabeli *Wynajecie* blokadę z zerową ceną i użytkownikiem o roli administratora.

Działanie:

Użytkownik wybiera daty wynajmu, a system sprawdza, czy samochód jest dostępny. Jeśli samochód jest zablokowany w danym okresie, rezerwacja nie jest możliwa. Administrator może zarządzać blokadami, np. wyłączając samochód z użytkowania na czas serwisu.

3. Automatyczne wysyłanie e-maili (SendGrid)

Opis funkcjonalności:

System automatycznie wysyła e-maile z potwierdzeniem rezerwacji i innymi powiadomieniami, korzystając z usługi SendGrid.

Implementacja:

Klucz API SendGrid jest przechowywany w pliku *appsettings.json* i ładowany w klasie *EmailService*. Metoda *SendEmailAsync* w *EmailService* obsługuje wysyłanie wiadomości, w tym treści w formatach HTML i tekstowym. Po dokonaniu rezerwacji w kontrolerze *WynajecieController* system wywołuje *EmailService*, wysyłając e-mail z podsumowaniem rezerwacji.

Działanie:

Użytkownik dokonuje rezerwacji lub zmienia dane rezerwacji. System automatycznie wysyła e-mail z potwierdzeniem szczegółów, takich jak daty wynajmu, cena i adres dostarczenia samochodu. E-maile są dostosowane do treści transakcji.



Wypożyczalnia Samochodowa daniel-boinski@o2.pl [przez](#) sendgrid.net
do mnie ▾

Dziękujemy za wynajęcie auta **Rolls Royce Cullican**.

Twoje wynajęcie trwa od **23.12.2024** do **24.12.2024**.

Adres dostarczenia: Nie podano adresu



Wypożyczalnia Samochodowa daniel-boinski@o2.pl [przez](#) sendgrid.net
do mnie ▾

Dziękujemy za wynajęcie auta **Rolls Royce Cullican**.

Twoje wynajęcie trwa od **26.12.2024** do **28.12.2024**.

Adres dostarczenia: Saski Point, 111, Marszałkowska, Za Żelazną Bramą, Śródmieście Północne, Śródmieście, Warszawa, województwo mazowieckie, 00-102, Polska

4. Integracja z Leaflet.js do obsługi map

Opis funkcjonalności:

Użytkownicy mogą wybierać lokalizację dostarczenia samochodu na interaktywnej mapie. Aplikacja wykorzystuje bibliotekę Leaflet.js do obsługi map.

Implementacja:

Leaflet.js jest załadowany w widokach Razor za pomocą plików JavaScript i CSS z CDN. Formularze rezerwacji w widoku *Wynajecie/Create* zawierają mapę, na której użytkownik może wskazać lokalizację dostarczenia samochodu. Współrzędne geograficzne (latitude i longitude) są automatycznie zapisywane w bazie danych na podstawie wyboru użytkownika.

Działanie:

Użytkownik otwiera formularz rezerwacji i widzi mapę. Klikając na mapę, ustawia marker w miejscu dostarczenia samochodu. Po zapisaniu rezerwacji wybrane współrzędne są przechowywane w tabeli *Wynajecie* jako *Latitude* i *Longitude*.


☒ Czy dostarczyć auto na wybrany adres?

Adres dostarczenia

50, Grzybowska, Mirów, Wola, Warszawa, województwo mazowieckie, 00-863, Polska

Pokaż na mapie

Wybierz lokalizację podstawienia na mapie



Metoda płatności