



WYDZIAŁ  
**ELEKTROTECHNIKI  
I INFORMATYKI**  
POLITECHNIKI RZESZOWSKIEJ

**Dawid Florian**

Portal internetowy dla firmy

**Projekt inżynierski**

Opiekun pracy:  
dr inż. Mariusz Gamracki

Rzeszów, 2023



# Spis Treści

1.	Wstęp.....	5
1.1.	Cel projektu.....	5
1.2.	Opis problemu.....	6
1.3.	Zakres pracy.....	7
2.	Analiza problemu .....	8
2.1.	Przegląd istniejących rozwiązań .....	8
2.2.	Analiza wymagań.....	10
2.2.1.	Wymagania funkcjonalne.....	10
2.2.2.	Wymagania niefunkcjonalne .....	12
2.3.	Analiza dostępnych technologii.....	12
2.3.1.	Analiza dostępnych rozwiązań serwerowych (backend) .....	14
2.3.2.	Analiza dostępnych rozwiązań UI (frontend).....	14
2.3.3.	Analiza dostępnych baz danych .....	15
2.4.	Analiza ryzyka.....	16
3.	Projektowanie systemu .....	17
3.1.	Przypadki użycia .....	17
3.1.1.	Diagram przypadków użycia.....	17
3.1.2.	Scenariusze przypadków użycia .....	18
3.2.	Architektura systemu.....	22
3.2.1.	Czysta architektura (architektura wielowarstwowa).....	22
3.2.2.	Wzorzec projektowy CQRS.....	23
3.2.3.	Wzorzec projektowy Dependency Inversion .....	23
3.3.	Baza danych .....	24
4.	Implementacja systemu.....	29
4.1.	Wybór technologii oraz narzędzi.....	29
4.1.1.	Narzędzia programistyczne .....	29
4.1.2.	Frameworki.....	31
4.1.3.	Główne biblioteki .....	33
4.1.3.	Platformy programistyczne .....	34
4.2.	Architektura systemu.....	36
4.3.	Przykładowe implementacje w kodzie źródłowym .....	37
4.4.	Testowanie systemu .....	42
4.4.1.	Testowanie Api.....	42
4.4.2.	Testowanie interfejsu graficznego .....	44
5.	Prezentacja systemu.....	45
6.	Podsumowanie i wnioski.....	52

Literatura.....	53
-----------------	----

# **1. Wstęp**

W dzisiejszych czasach, rozwój technologii internetowych coraz bardziej rewolucjonizuje sposoby docierania firm do klientów na całym świecie. Dzięki temu firmy poszerzają swoje możliwości oraz zasięgi. Jednym z sektorów, który odnosi dzięki temu wymierne korzyści, jest branża wypożyczalni samochodowych. Wypożyczalnie samochodowe coraz częściej wykorzystują portale internetowe jako narzędzia, które umożliwiają klientom łatwy dostęp do ich usług oraz wygodne i szybkie dokonywanie rezerwacji online. Dzięki wykorzystaniu Internetu, te firmy mogą dotrzeć do szerokiego grona potencjalnych klientów na całym świecie, niezależnie od ich geograficznego położenia. Rezerwacje online zapewniają wygodę i oszczędność czasu dla klientów, eliminując konieczność osobistej wizyty w siedzibie wypożyczalni. Ten rozwój technologii internetowych pozwala skutecznie konkurować na globalnym rynku, jednocześnie zapewniając klientom łatwiejszy i bardziej dostępny proces wypożyczenia pojazdu.

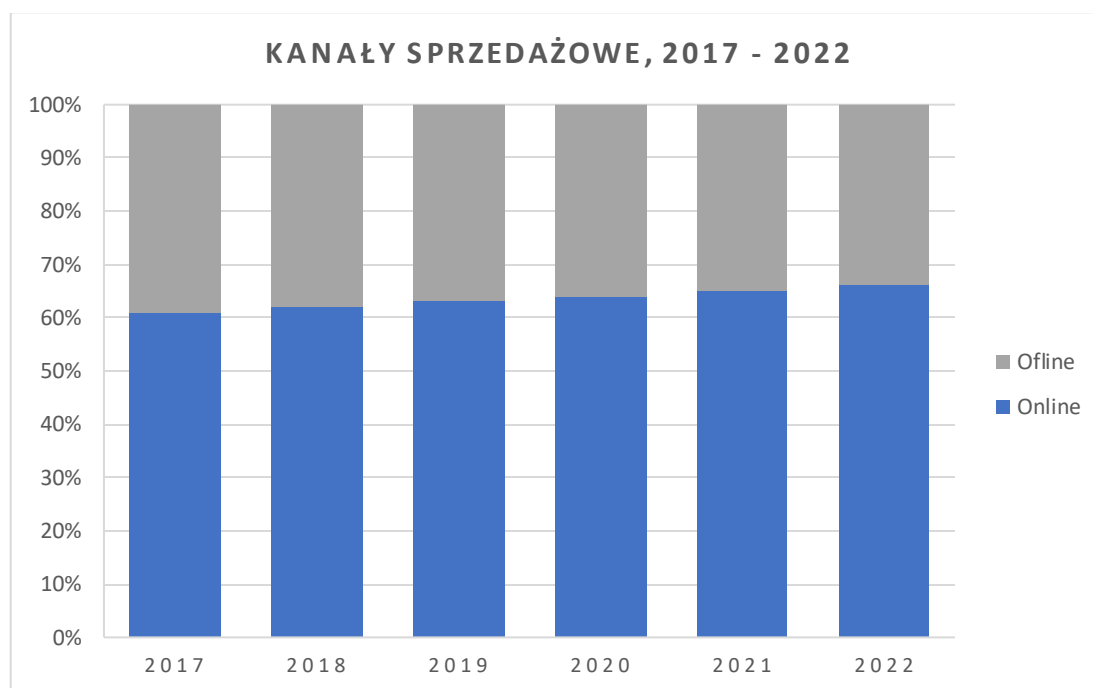
## **1.1. Cel projektu**

Celem projektu inżynierskiego było opracowanie oraz stworzenie wszechstronnego portalu internetowego, który może pozwolić małej firmie na skuteczne wykorzystywanie nowoczesnych technologii. Głównymi celami projektu było stworzenie platformy internetowej, umożliwiającej firmie reklamowanie swoich usług, efektywną sprzedaż oraz prowadzenie ewidencji usług. Projekt zakładał stworzenie platformy internetowej dla firmy zajmującej się wynajmem samochodów, pozwalającej na przyciągnięcie nowych oraz zadowolenie stałych klientów poprzez atrakcyjne prezentowanie oferty oraz intuicyjne zamawianie usług. Portal powinien umożliwiać klientom dokonywanie rezerwacji online, sprawdzanie dostępności samochodów, wycenę usług, dokonywanie płatności oraz uzyskiwanie informacji o firmie. Aby umożliwić użytkownikom korzystanie z portalu internetowego niezależnie od platformy, na której działają, projektant mógł wykorzystać dowolną platformę programistyczną, która gwarantuje stabilność systemu. Aplikacja została napisana zgodnie z dobrymi praktykami tak aby zachować jej skalowalność.

Jest to nowoczesne narzędzie, które znacząco ułatwi klientom jak i firmie obsługę procesu wynajmu samochodów, zapewni dostęp do informacji oraz umożliwi zdalne zarządzanie firmą.

## 1.2. Opis problemu

Kluczowym problemem podczas realizowania aplikacji webowej dla firmy jest zapewnienie klientom szybkiego oraz łatwego algorytmu rezerwowania samochodów. Strona powinna zachęcać użytkowników do korzystania z usług poprzez atrakcyjnie prezentowanie oferty. W tradycyjnej formie firmy to pracownik jest odpowiedzialny za zainteresowanie klienta ofertą, obsługę klienta jak i finalizację zamówienia. W internetowej wersji serwis musi przejąć wszystkie funkcje konsultanta, dlatego tak ważne jest aby system był łatwy, intuicyjny oraz przyjaźnie nastawiony do użytkownika, aby tworzyć jak najlepszą atmosferę. Jednym z plusów tego nowoczesnego rozwiązania może być dostępność, forma wirtualnej wypożyczalni pozwala dokonywać rezerwacji 24 godziny na dobę, 7 dni w tygodniu z dowolnego miejsca z dostępem do Internetu. Rys. 1.1 przedstawia zmiany udziałów poszczególnych kanałów sprzedażowych w wynajmie samochodów.



Rys. 1.1. Udział kanałów sprzedażowych wypożyczalni samochodowych [1]

Kilka powodów może przyczyniać się do wzrostu zainteresowania wynajmowaniem samochodów online. Głównym czynnikiem jest dostępność aplikacji internetowych, które są łatwe w użyciu. Dzięki temu, że każdy ma dostęp do Internetu, łatwo można korzystać z takich witryn. Kolejnym czynnikiem jest łatwość prezentowania ofert w korzystny sposób, klienci nie muszą oglądać katalogów czy makiet, zamiast tego dostają profesjonalnie przygotowane zdjęcia, dostępne bezpośrednio w serwisie.

### **1.3. Zakres pracy**

Rezultatem wykonanej pracy było opracowanie i stworzenie specjalnego oprogramowania dla firmy zajmującej się wynajmem samochodów, które będzie dostosowane do jej specyficznych potrzeb i wymagań. System powinien pozwalać na zarządzanie procesami zachodzącymi w firmie w sposób efektywny. Przed rozpoczęciem projektu konieczna była analiza konkurencji, w celu zrozumienia potrzeb klientów oraz określenia celów projektu. Pierwszym krokiem był dobór odpowiednich narzędzi oraz technologii, które zostaną użyte podczas realizacji pracy. Istotnym aspektem projektu było stworzenie bazy danych, która umożliwi przechowywanie informacji dotyczących klientów, ich rezerwacji oraz samochodów dostępnych do wynajęcia. Baza danych musi być zaprojektowana w sposób wydajny i bezpieczny, aby zagwarantować ochronę prywatności danych klientów. Następnie zaplanowano stronę internetową oraz stworzono interaktywną platformę, która pozwoli użytkownikom wypożyczyć pojazd online. Projekt musi zawierać łatwy w obsłudze i intuicyjny interfejs użytkownika oraz niezbędne funkcje, takie jak rezerwacja samochodu, płatności i możliwość uzyskania informacji kontaktowych.

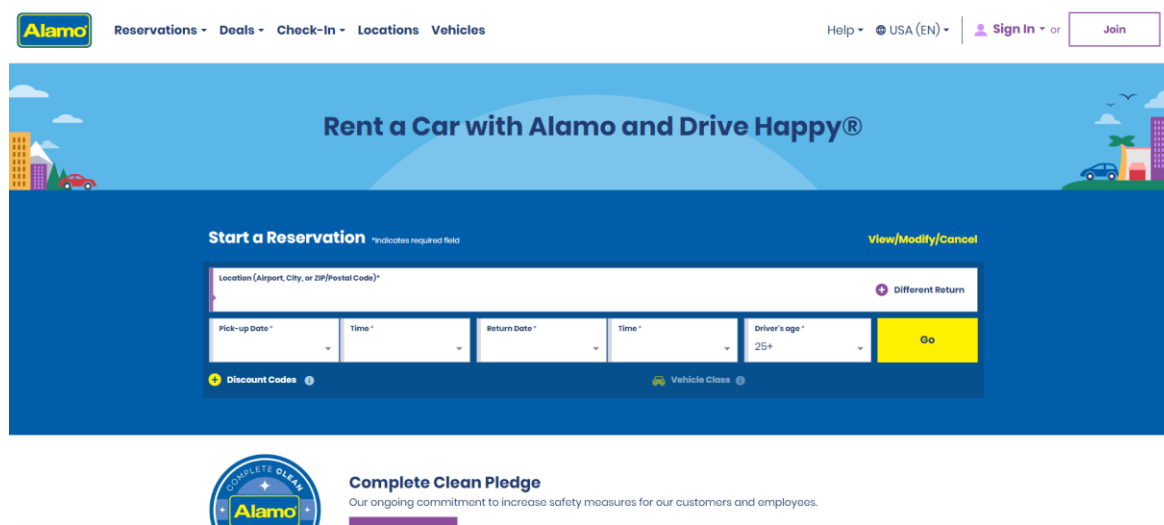
Podsumowując, wykonanie projektu inżynierskiego pod tytułem "Portal internetowy dla firmy" wymagało przeprowadzenia szeregu działań, w tym badań rynkowych, projektowania i wdrożenia strony internetowej, zaprojektowania bazy danych oraz przetestowania. Wszystkie te elementy muszą być ze sobą harmonijnie zintegrowane, aby zapewnić użytkownikom wygodną i skuteczną platformę do wynajmu samochodów.

## 2. Analiza problemu

Ten dział skupia się na analizie problemu związanego z tematem projektu oraz omówieniu aspektów związanych z tworzeniem aplikacji poprzez przegląd istniejących, podobnych rozwiązań. Zostały również określone wymagania, które powinna spełniać tworzona aplikacja oraz oceniono dostępne technologie umożliwiające realizację zadania.

### 2.1. Przegląd istniejących rozwiązań

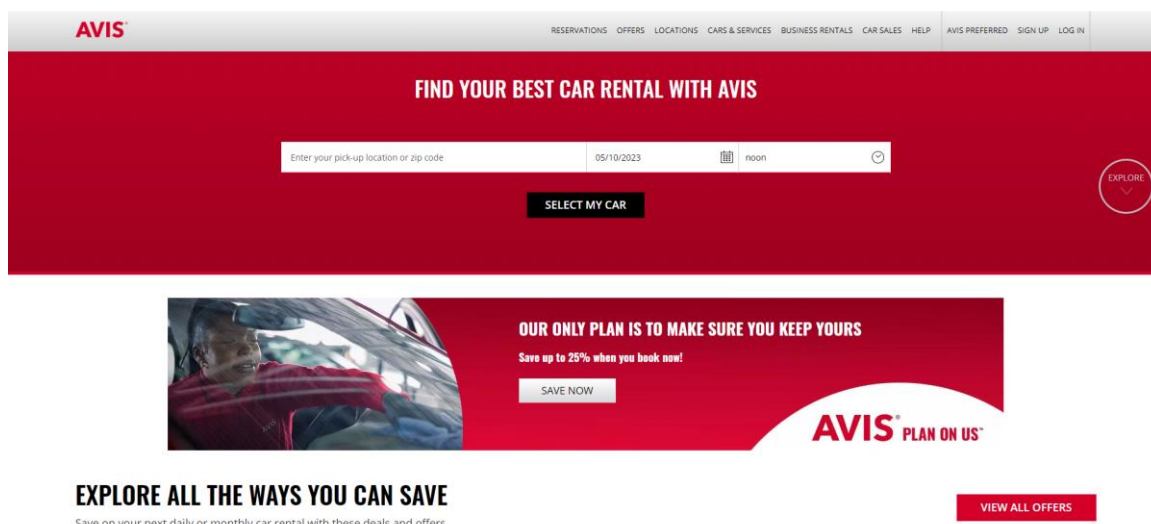
Wynikiem analizy rynku było pozyskanie informacji o podobnych, istniejących serwisach, świadczących usługi wynajmu samochodów. Kluczowymi cechami tych serwisów są: rozbudowane sekcje informacji o firmie pozwalające na zbudowanie poczucia zaufania klienta, możliwość przeglądania atrakcyjnie ukazanych samochodów oraz intuicyjny sposób wyszukiwania dostępnych terminów oraz ofert. Aplikacje największych firm stawiają na prostotę, nie przytłaczają użytkowników mnogością funkcjonalności czy efektów interfejsu graficznego. Na rys. 2.1 – 2.4 przedstawiono widoki wyszukiwania ofert największych firm tej branży.

The image shows the Alamo website's reservation interface. At the top, there's a navigation bar with the Alamo logo, links for Reservations, Deals, Check-In, Locations, and Vehicles, and a user section with 'Help', 'USA (EN)', 'Sign In', and 'Join'. The main header features the slogan 'Rent a Car with Alamo and Drive Happy®'. Below this is a 'Start a Reservation' form with a 'View/Modify/Cancel' link. The form includes a 'Location' field, a 'Different Return' option, and a row of five fields: 'Pick-up Date', 'Time', 'Return Date', 'Time', and 'Driver's age' (set to '25+'). A yellow 'Go' button is to the right. Below the form are links for 'Discount Codes' and 'Vehicle Class'. At the bottom, there's a 'Complete Clean Pledge' badge and text.

Rys. 2.1. System rezerwacji firmy „Alamo” [2]

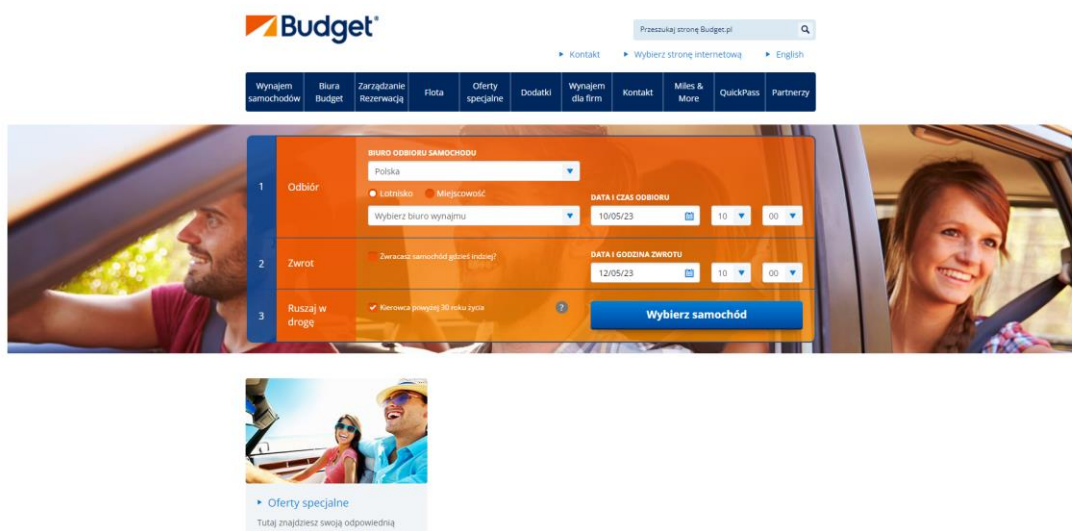
Alamo to amerykańska firma wypożyczalni samochodów założona w 1974 roku. Jest jednym z wiodących graczy na rynku, oferując szeroki wybór pojazdów dla klientów indywidualnych i korporacyjnych. Firma słynie z profesjonalnej obsługi klienta i konkurencyjnych cen [2].





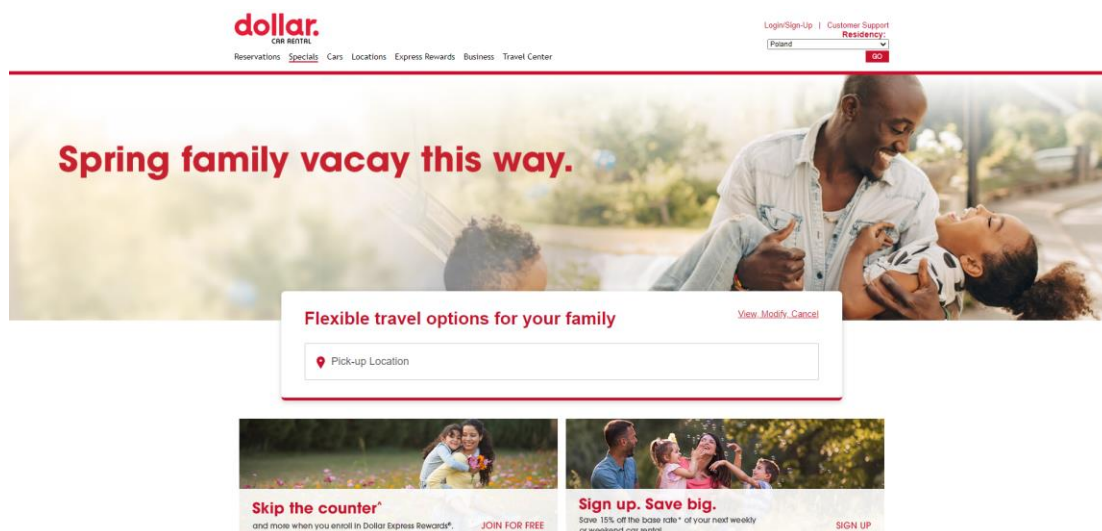
Rys. 2.2. System rezerwacji firmy „Avis” [3]

Avis to międzynarodowa firma wypożyczalni samochodów założona w 1946 roku. Zajmuje czołową pozycję na rynku, oferując różnorodny wybór pojazdów dostosowanych do potrzeb klientów. Avis jest znany z innowacyjnych usług, takich jak Avis Preferred, program lojalnościowy, oraz możliwości rezerwacji online [3].



Rys. 2.3. System rezerwacji firmy „Budget” [4]

Budget to międzynarodowa firma wypożyczalni samochodów, założona w 1958 roku. Jest znana jako dostawca przystępnych cenowo usług wynajmu pojazdów. Budget oferuje szeroki wybór samochodów w różnych kategoriach, zapewniając prostotę rezerwacji i elastyczne warunki wynajmu [4].



Rys. 2.4. System rezerwacji firmy „Dollar” [5]

Dollar to międzynarodowa firma wypożyczalni samochodów, założona w 1965 roku. Specjalizuje się w oferowaniu przystępnych cenowo usług wynajmu pojazdów. Firma zapewnia różnorodny wybór samochodów dostosowanych do potrzeb klientów indywidualnych i biznesowych, dążąc do prostoty i wygody w procesie rezerwacji [5].

Aplikacja tworzona na potrzeby pracy, projektowana była tak aby zawierała pozytywne cechy wymienionych wyżej serwisów w celu zapewnienia konkurencyjności.

## 2.2. Analiza wymagań

Analiza wymagań polega na ustaleniu wszystkich potrzeb oraz oczekiwań klienta wobec zaprojektowanej aplikacji. Jest to ważny punkt procesu tworzenia oprogramowania, od niego zależy czy produkt będzie spełniał oczekiwania użytkowników.

### 2.2.1. Wymagania funkcjonalne

Wymagania funkcjonalne dotyczą funkcji, zachowania oraz zdolności systemu, które należy spełnić w celu zrealizowania określonych celów biznesowych. W niniejszym rozdziale przedstawiono wymagania funkcjonalne dotyczące portalu internetowego dla wypożyczalni samochodów:

- 1) Dostęp do informacji bez logowania:
  - przeglądanie informacji o firmie,
  - przeglądanie danych kontaktowych,
  - przeglądanie samochodów.
- 2) Autoryzacja oraz autentykacja użytkowników:
  - możliwość samodzielnej rejestracji w serwisie,

- logowanie do serwisu,
- przeglądanie oraz edycja danych osobowych,
- zmiana hasła.

3) Dostęp do panelu klienta:

- przeglądanie ofert,
- dokonywanie rezerwacji,
- anulowanie rezerwacji,
- wgląd w historię rezerwacji,
- wykonywanie płatności,
- wgląd w historię płatności.

4) Dostęp do panelu pracownika:

- wgląd w istniejące samochody,
- dodawanie samochodów,
- edycja samochodów,
- usuwanie samochodów,
- wgląd w oferty,
- dodawanie ofert,
- usuwanie ofert,
- wgląd do wszystkich płatności,
- wgląd do wszystkich rezerwacji,
- zatwierdzanie rezerwacji,
- odrzucanie rezerwacji,
- realizowanie rezerwacji.

5) Dostęp do panelu administratora firmy:

- wgląd w oddziały firmy,
- dodawanie oddziałów,
- edycja oddziałów,
- usuwanie oddziałów,
- wgląd w pracowników,
- dodawanie pracowników,
- edycja pracowników,
- usuwanie pracowników,

- przypisywanie pracowników do oddziałów,
- usuwanie pracowników z oddziałów.

### **2.2.2. Wymagania niefunkcjonalne**

Wymagania niefunkcjonalne to szczegółowe opisy cech systemu lub oprogramowania, które mają wpływ na jakość i ogólne działanie systemu, jednak nie są związane bezpośrednio z jego funkcjonalnościami. W niniejszym rozdziale przedstawiono wymagania niefunkcjonalne dotyczące portalu internetowego dla wypożyczalni samochodów:

- 1) Wydajność systemu - działanie szybko i sprawnie, nawet przy dużym obciążeniu.
- 2) Bezpieczeństwo - zapewnienie wysokiego poziomu bezpieczeństwa danych, w tym zabezpieczenie danych logowania i informacji o użytkownikach.
- 3) Intuicyjność interfejsu – bycie łatwym w obsłudze, nawet dla osób nie mających dużego doświadczenia w korzystaniu z Internetu.
- 4) Skalowalność - bycie otwartym na zmieniające się warunki, umożliwiać łatwe rozwijanie i przystosowywanie do potrzeb.
- 5) Dostępność – bycie dostępnym dla użytkowników przez całą dobę, 7 dni w tygodniu,
- 6) Kompatybilność – wyświetlanie i działanie w sposób poprawny, zgodnie z założeniami, na różnych przeglądarkach oraz urządzeniach.
- 7) Wsparcie techniczne - zgłaszane błędy powinny być weryfikowane oraz naprawiane. Zaplecze specjalistów powinno dbać o jak najszybsze przywrócenie usług w przypadku katastrofy.

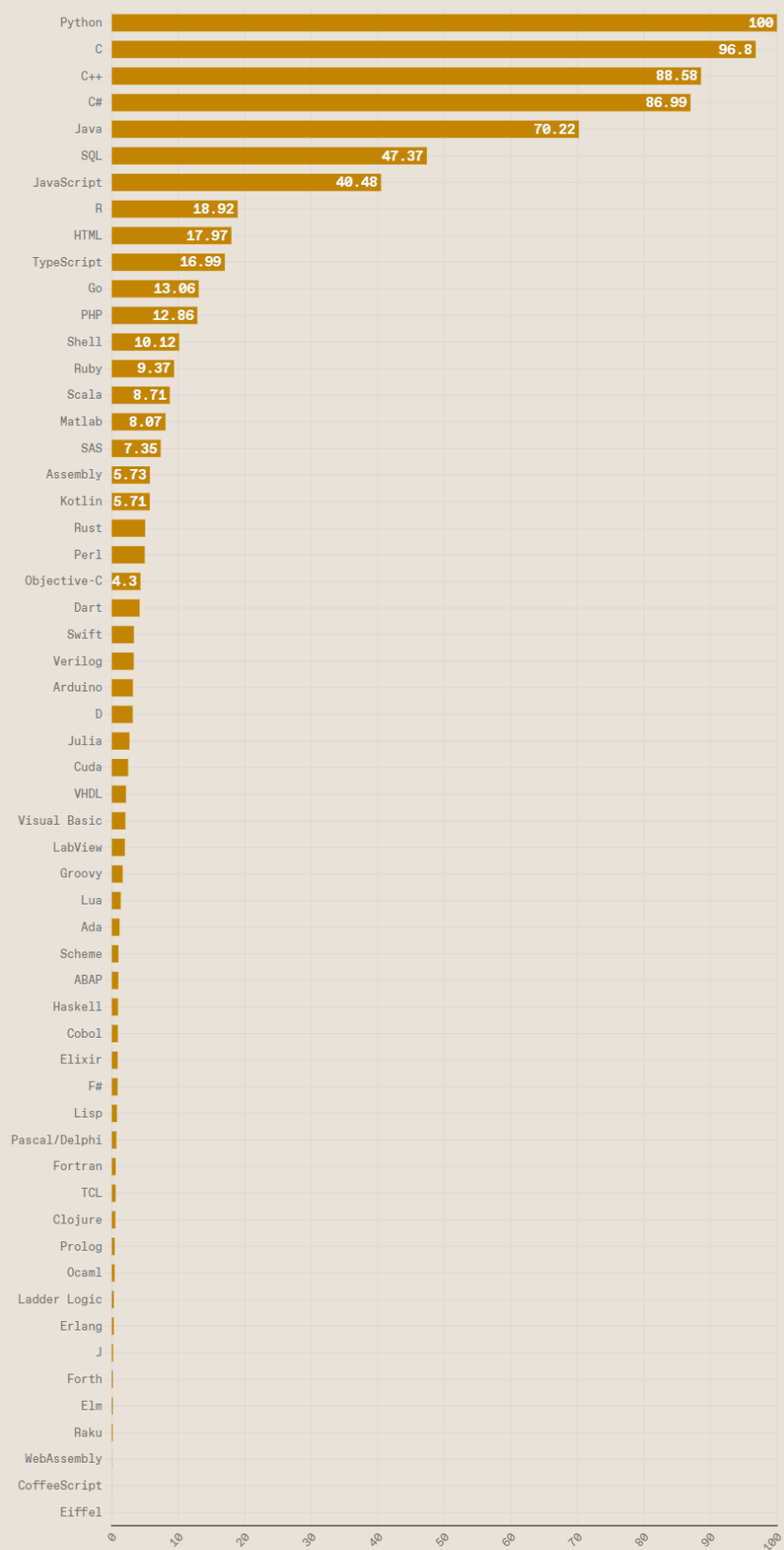
### **2.3. Analiza dostępnych technologii**

Przy realizacji projektu portalu internetowego dla firmy, ważne jest wybranie odpowiednich technologii i narzędzi, które umożliwią stworzenie funkcjonalnego, łatwego w obsłudze i skalowalnego systemu. Wymagane jest dobranie technologii dla serwera aplikacji, interfejsu użytkownika oraz bazy danych. Na rys. 2.5 pokazano zestawienie najpopularniejszych języków programowania, przygotowane przez instytut inżynierów elektryków i elektroników na rok 2022 [6].

# Top Programming Languages 2022

Click a button to see a differently weighted ranking

Spectrum Jobs Trending



Rys. 2.5. Ranking języków programowania [6]

### **2.3.1. Analiza dostępnych rozwiązań serwerowych (backend)**

Backend to część aplikacji odpowiedzialna za przetwarzanie danych i logikę biznesową, zwykle niewidoczna dla użytkownika ale niezwykle istotna dla prawidłowego działania systemu. Poniżej przedstawiono popularne języki programowania aplikacji backendowych.

- 1) Python - popularny język backendowy o czytelnej składni i bogatym ekosystemie. Wykorzystuje się go do przetwarzania danych, logiki biznesowej, interakcji z bazami danych i tworzenia API. Elastyczny i skalowalny, jest często wyborem dla tworzenia efektywnych aplikacji backendowych. Posiada wiele frameworków, takich jak Django czy Flask, które ułatwiają rozwijanie zaawansowanych aplikacji backendowych [7].
- 2) C# - wszechstronny język backendowy z silną typizacją i obiektowością. Wykorzystuje się go do tworzenia skalowalnych aplikacji backendowych. Zapewnia obsługę wielowątkowości, integrację z bazami danych i tworzenie usług sieciowych. C# jest często wybierany dla tworzenia wydajnych aplikacji backendowych opartych na platformie .NET. Platforma .NET zapewnia również silne wsparcie dla tworzenia usług sieciowych, takich jak usługi RESTful lub usługi sieciowe SOAP [8].
- 3) Java - popularny język backendowy, znany ze swojej niezawodności i przenośności. Wykorzystuje się go do tworzenia skalowalnych aplikacji backendowych. Zapewnia obsługę wątków, integrację z bazami danych i rozbudowany ekosystem narzędzi. Java jest często stosowana w dużej skali projektów enterprise [9].
- 4) PHP - powszechnie stosowany język backendowy. Jest szczególnie popularny w tworzeniu stron internetowych i aplikacji internetowych. Oferuje wsparcie dla interakcji z bazami danych, obsługę żądań HTTP i dynamiczną generację treści. PHP jest często wybierany ze względu na swoją prostotę i dostępność [10].

Analizując wady i zalety powyższych rozwiązań, wybrany został język C# a dokładniej framework .Net Core RESTful API.

### **2.3.2. Analiza dostępnych rozwiązań UI (frontend)**

Frontend to interaktywna część aplikacji, widoczna dla użytkownika, która obsługuje prezentację danych i umożliwia interakcję z użytkownikiem. Jest to jedyna warstwa aplikacji z którą użytkownik ma bezpośredni kontakt. Poniżej przedstawiono popularne frameworki programowania frontendowego.

- 1) React - popularny framework JavaScript, używany do tworzenia interaktywnych interfejsów użytkownika. Opiera się na komponentach, umożliwiając efektywne zarządzanie stanem aplikacji. Dzięki wirtualnemu DOM i jednokierunkowemu przepływowi danych, React zapewnia szybkie i efektywne renderowanie UI. Jest wykorzystywany zarówno w aplikacjach webowych, jak i mobilnych [11].
- 2) Blazor - innowacyjny framework Microsoftu, który umożliwia tworzenie aplikacji interaktywnych w C# i .NET, bezpośrednio w przeglądarce. Wykorzystuje technologię WebAssembly, co pozwala uruchamiać kod backendowy wewnątrz przeglądarki, zapewniając płynne i responsywne interfejsy użytkownika [12].
- 3) Angular - kompleksowy framework JavaScript, rozwijany przez Google, służący do tworzenia skalowalnych aplikacji webowych. Oferuje modułowość, dwukierunkowe wiązanie danych, reaktywne funkcje i zaawansowane narzędzia do testowania. Angular wspiera rozbudowane struktury projektowe, takie jak komponenty i usługi, zapewniając efektywne zarządzanie kodem aplikacji [13].
- 4) Vue - lekki i elastyczny framework JavaScript, który umożliwia tworzenie interaktywnych interfejsów użytkownika. Zapewnia jednokierunkowe wiązanie danych, komponenty, reaktywność i prostotę integracji. Vue jest łatwy w nauce i ma rozbudowaną społeczność, co przyczynia się do jego popularności w tworzeniu aplikacji webowych [14].

Jako interfejs graficzny został wybrany framework Blazor, pozwala on tworzyć aplikacje WebAssembly oraz również korzysta z technologii .Net Core.

### **2.3.3. Analiza dostępnych baz danych**

Relacyjna baza danych to rodzaj bazy danych, w której dane są przechowywane w postaci tabel zdefiniowanych zgodnie z relacyjnym modelem danych. Tabele te są powiązane ze sobą za pomocą kluczy i umożliwiają kompleksowe zarządzanie danymi, włączając w to zapytania, aktualizacje i tworzenie relacji między danymi. Poniżej przedstawiono popularne relacyjne bazy danych.

- 1) MySQL - popularna relacyjna baza danych, używana do przechowywania danych w aplikacjach webowych. Zapewnia wysoką wydajność, łatwość konfiguracji i skalowalność. Obsługuje wiele języków programowania, oferuje zaawansowane funkcje, takie jak transakcje, indeksowanie i replikację danych [15].
- 2) Microsoft SQL Server - potężna relacyjna baza danych stworzona przez Microsoft. Zapewnia niezawodność, skalowalność i bezpieczeństwo danych. Posiada

zaawansowane funkcje, takie jak obsługa transakcji, analiza danych, replikacja i narzędzia biznesowe. Często wykorzystywana w dużych i średnich przedsiębiorstwach [16].

- 3) PostgreSQL - zaawansowana, open-source'owa relacyjna baza danych. Oferuje skalowalność, niezawodność i obsługę transakcji. Posiada bogaty zestaw funkcji, w tym indeksowanie pełnotekstowe, procedury składowane i replikację. PostgreSQL jest często wybierany dla projektów wymagających zaawansowanych możliwości i większej kontroli nad bazą danych [17].

Podczas realizacji projektu została zaimplementowana baza danych Microsoft SQL Server dzięki czemu cała aplikacja opiera się na rozwiązaniach firmy Microsoft.

## **2.4. Analiza ryzyka**

Przed rozpoczęciem projektu wdrożenia portalu internetowego dla wypożyczalni samochodów, istnieje potrzeba przeprowadzenia analizy ryzyka, aby zidentyfikować potencjalne zagrożenia i podjąć odpowiednie działania zapobiegawcze. Poniżej znajdują się kluczowe obszary ryzyka, które należy uwzględnić w procesie analizy:

- 1) Bezpieczeństwo danych.
- 2) Awaria systemu.
- 3) Negatywne opinie i oceny klientów.
- 4) Konkurencja na rynku.
- 5) Brak zaakceptowania klientów.

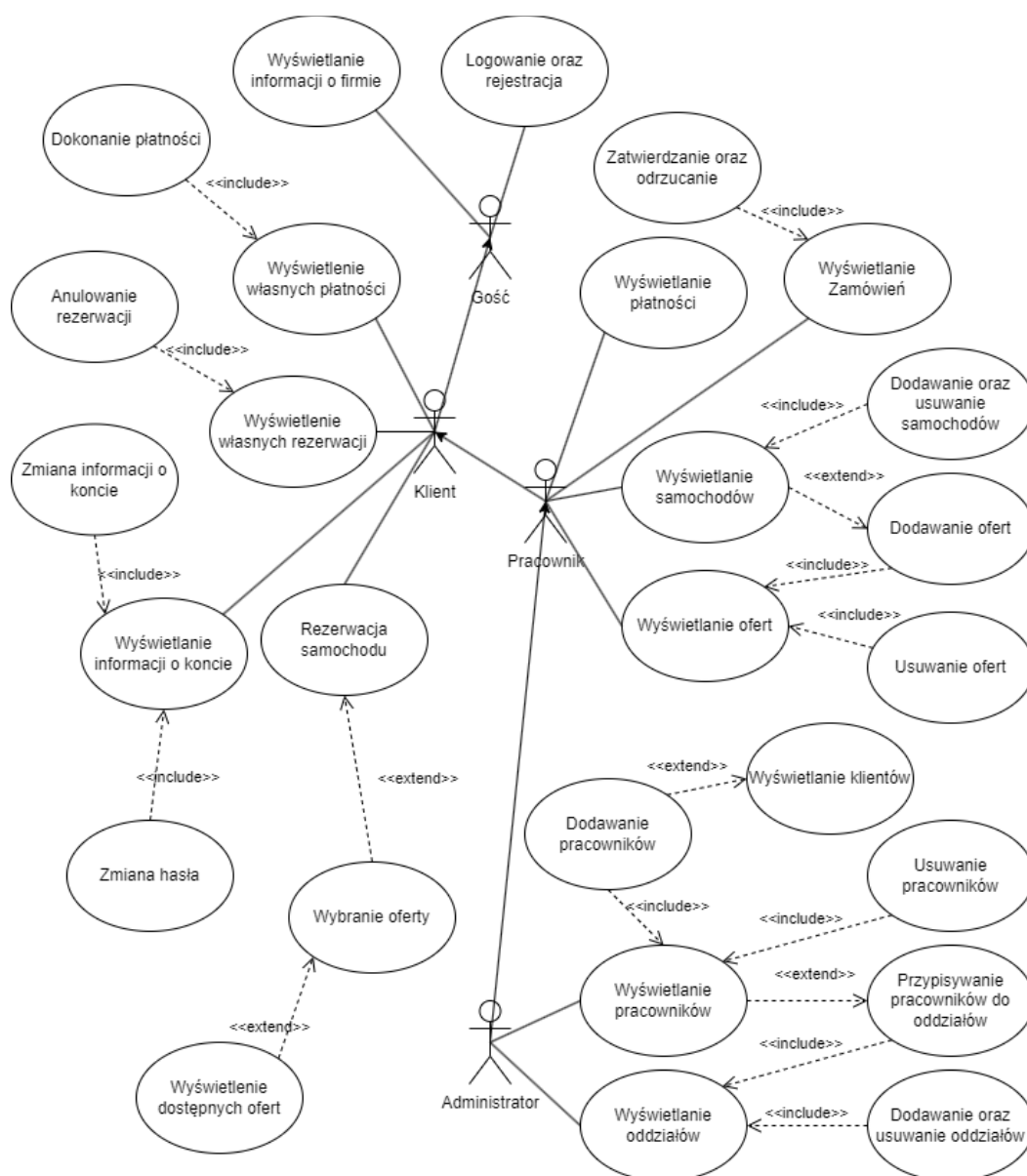


### 3. Projektowanie systemu

#### 3.1. Przypadki użycia

##### 3.1.1. Diagram przypadków użycia

W fazie projektowania systemu portalu internetowego dla wypożyczalni samochodów, należy skoncentrować się na zaplanowaniu struktury, funkcjonalności i interfejsu użytkownika. Serwis wypożyczający samochody oferuje wiele różnych przypadków użycia, które mogą być przydatne dla klientów. Podstawowymi oraz najważniejszymi funkcjonalnościami są: dokonywanie rezerwacji, wykonywanie płatności, zarządzanie ofertami oraz personelem. Diagram na rys. 3.1 przedstawia interakcje użytkownika z systemem rezerwacji samochodów podczas korzystania z portalu.



Rys. 3.1. Diagram przypadków użycia

### 3.1.2. Scenariusze przypadków użycia

Poniżej przedstawiono scenariusze przypadków użycia dla najważniejszych funkcjonalności portalu.

#### 1) Rejestracja

**Aktor:** gość.

**Dane wejściowe:** email, imię, nazwisko, data urodzenia, numer telefonu, pesel, hasło, potwierdzenie hasła.

**Wymagania początkowe:** niezalogowany użytkownik wprowadza poprawne dane do utworzenia konta, nie posiada konta w serwisie.

**Scenariusz:**

- przejście na stronę rejestracji,
- wprowadzenie poprawnych danych dla nowego konta,
- rejestrowanie za pomocą przycisku „Zarejestruj się”,
- walidacja poprawności wprowadzonych danych,
- sprawdzenie unikalności emaila w bazie danych,
- przekierowanie na widok logowania.

**Rezultat końcowy:** utworzenie nowego konta użytkownika, przekierowanie na ekran logowania.

#### 2) Logowanie

**Aktor:** gość.

**Dane wejściowe:** email, hasło.

**Wymagania początkowe:** niezalogowany użytkownik posiada konto w serwisie, wprowadza poprawne dane swojego konta.

**Scenariusz:**

- przejście na stronę logowania,
- wprowadzenie poprawnych danych przypisanych do konta użytkownika,
- logowanie za pomocą przycisku „Zaloguj się”,
- walidowanie poprawności wprowadzonych danych,
- porównanie wprowadzonych danych z danymi bazy danych,
- przekierowanie na widok panelu klienta.

**Rezultat końcowy:** autentykacja konta w systemie, przekierowanie na ekran panelu klienta.

### 3) Rezerwacja samochodu

**Aktor:** klient.

**Dane wejściowe:** data od, data do, oddał, samochód, oferta.

**Wymagania początkowe:** zalogowany użytkownik wprowadza poprawne dane, istnieje w systemie oferta zgodna z wprowadzonymi danymi.

**Scenariusz:**

- przejście na stronę listy ofert wybierając z menu „Zarezerwuj”,
- wprowadzenie nazwy samochodu, oddziału, daty od oraz do,
- wyszukanie samochodów za pomocą przycisku „Szukaj”,
- wyświetlenie ofert dla danego samochodu za pomocą przycisku „Dalej”,
- wybranie oferty klikając „Wybierz” na wierszu danej oferty,
- przekierowanie na widok rezerwacji klienta.

**Rezultat końcowy:** stworzenie nowej rezerwacji w statusie „Nowa”, stworzenie nowej płatności w statusie „Nowa”, przekierowanie klienta na widok listy jego rezerwacji.

### 4) Anulowanie rezerwacji

**Aktor:** klient.

**Dane wejściowe:** brak.

**Wymagania początkowe:** zalogowany użytkownik posiada rezerwacje samochodu w statusie „Nowa”.

**Scenariusz:**

- wybranie z menu „Moje zamówienia”,
- anulowanie rezerwacji za pomocą przycisku „Anuluj”,
- aktualizowanie listy zamówień.

**Rezultat końcowy:** zmiana statusu rezerwacji na „Anulowana”, zmiana statusu płatności na „Anulowana”, aktualizacja listy rezerwacji.

**Rezultat końcowy:** wyświetlenie listy płatności przypisanych do użytkownika.

### 5) Dokonanie płatności

**Aktor:** klient.

**Dane wejściowe:** brak.

**Wymagania początkowe:** zalogowany użytkownik posiada przynajmniej jedną płatność w statusie „Oczekująca”.

**Scenariusz:**

- wybranie z menu „Moje płatności”,
- dokonanie płatności za pomocą przycisku „Opłać”,
- aktualizowanie listy płatności.

**Rezultat końcowy:** zmiana statusu płatności na „Opłacona”, aktualizacja listy płatności.

#### 6) Potwierdzanie rezerwacji

**Aktor:** pracownik.

**Dane wejściowe:** brak.

**Wymagania początkowe:** zalogowany użytkownik, istnieje przynajmniej jedna rezerwacja w statusie „Nowa”.

**Scenariusz:**

- wybranie z menu „Zamówienia”,
- potwierdzenie zamówienia za pomocą przycisku „Potwierdź”,
- aktualizowanie listy wszystkich rezerwacji.

**Rezultat końcowy:** status rezerwacji zmieniono na „Zaakceptowane”, status płatności zmieniono na „Oczekująca”.

#### 7) Wykonanie rezerwacji

**Aktor:** pracownik.

**Dane wejściowe:** brak.

**Wymagania początkowe:** zalogowany użytkownik, istnieje przynajmniej jedna rezerwacja w statusie „Zaakceptowane”.

**Scenariusz:**

- wybranie z menu „Zamówienia”,
- odrzucenie zamówienia za pomocą przycisku „Wykonaj”,
- aktualizowanie listy wszystkich rezerwacji.

**Rezultat końcowy:** status rezerwacji zmieniono na „Zrealizowane”.

#### 8) Dodawanie oddziałów

**Aktor:** administrator.

**Dane wejściowe:** nazwa, telefon, kraj, ulica, kod pocztowy, miasto.

**Wymagania początkowe:** zalogowany użytkownik, wprowadza poprawne dane nowego oddziału.

**Scenariusz:**

- wybranie z menu „Oddziały”,
- wyświetlenie formularza za pomocą przycisku „Dodaj”,
- wprowadzenie poprawnych danych nowego oddziału,
- dodanie nowego oddziału za pomocą przycisku „Dodaj”,
- walidowanie wprowadzonych danych nowego oddziału,
- wyświetlenie komunikatu świadczącego o poprawnym dodaniu oddziału.

**Rezultat końcowy:** zapisanie nowego oddziału w bazie danych.

#### 9) Dodawanie Pracowników

**Aktor:** administrator.

**Dane wejściowe:** użytkownik.

**Wymagania początkowe:** zalogowany użytkownik, system zawiera przynajmniej jednego użytkownika, który nie jest pracownikiem.

**Scenariusz:**

- wybranie z menu „Pracownicy”,
- wyświetlenie listę możliwych pracowników do dodania za pomocą przycisku „Dodaj”,
- dodanie wybranego użytkownika za pomocą przycisku „+”,
- wyświetlenie listy wszystkich pracowników.

**Rezultat końcowy:** do bazy danych dodano nowego pracownika bez praw administracyjnych.

#### 10) Przypisywanie pracowników do oddziałów

**Aktor:** pracownik.

**Dane wejściowe:** oddział, pracownik.

**Wymagania początkowe:** zalogowany użytkownik, istnieje przynajmniej jeden pracownik oraz oddział.

**Scenariusz:**

- wybranie z menu „Oddziały”,
- wyświetlanie pracowników w oddziale za pomocą przycisku, zawierającego ikonę ludzi,
- wyświetlenie możliwych do dodania pracowników za pomocą przycisku „Dodaj”,
- przypisanie wybranego pracownika do oddziału za pomocą przycisku „+”,
- przekierowano na widok listy pracowników w danym oddziale.

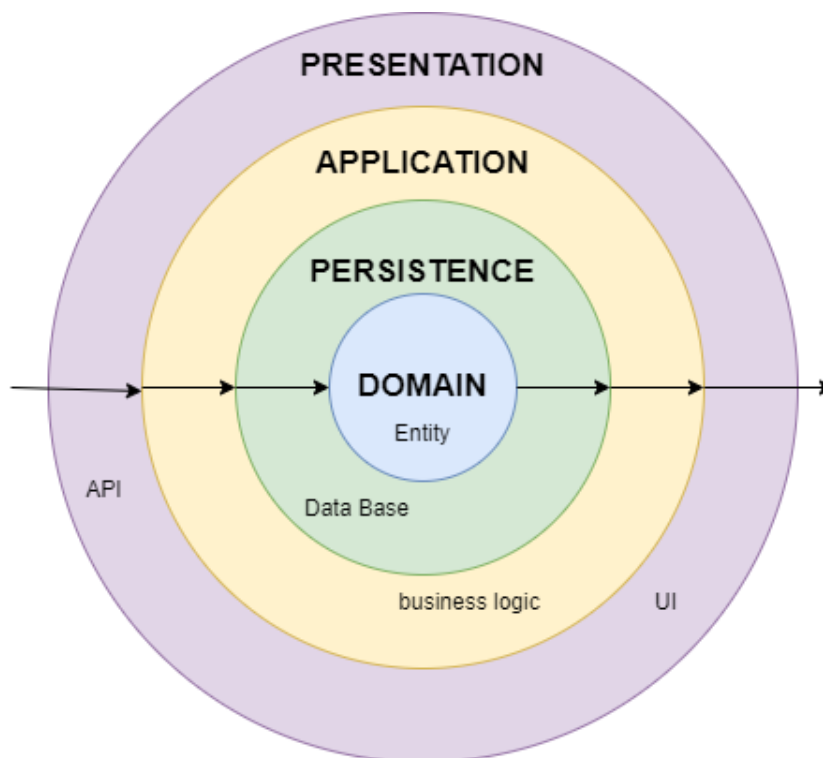
**Rezultat końcowy:** Przypisano wybranego użytkownika do wybranego oddziału.

## 3.2. Architektura systemu

W procesie projektowania portalu internetowego dla wypożyczalni samochodów, ważnym aspektem jest określenie odpowiedniej architektury systemu. Architektura systemu odzwierciedla strukturę i organizację komponentów, które współpracują ze sobą w celu dostarczenia funkcjonalności portalu.

### 3.2.1. Czysta architektura (architektura wielowarstwowa)

Czysta architektura to strategia projektowania oprogramowania, która stawia na rozdzielanie zależności między różnymi elementami systemu. Jej głównym celem jest tworzenie kodu, który jest elastyczny, łatwy w skalowaniu i zrozumieniu. W ramach czystej architektury wyróżniamy kilka warstw: interfejs użytkownika, aplikacji, domeny i dostępu do bazy danych. Każda z tych warstw pełni określone funkcje i jest niezależna od innych. Czysta architektura opiera się na zasadzie jednokierunkowego przepływu danych, gdzie informacje są przekazywane od interfejsu użytkownika przez kolejne warstwy. Takie podejście umożliwia modyfikację jednej warstwy bez wpływu na pozostałe, co znacznie ułatwia testowanie, utrzymanie i modyfikowanie kodu. Czysta architektura zachęca również do wykorzystywania wzorców projektowych, takich jak CQRS (Command Query Responsibility Segregation) czy Dependency Inversion, w celu zwiększenia elastyczności i odseparowania różnych części systemu. Schemat pokazany na rys. 3.2 przedstawia strukturę projektu oraz kierunek przepływu danych.



Rys. 3.2. Schemat czystej architektury

Opis poszczególnych warstw:

- 1) Warstwa „Domain” - Odpowiada za reprezentację struktury bazy danych a co za tym idzie struktury firmy. W tej warstwie definiowane są klasy na podstawie których są tworzone tabele bazy danych.
- 2) Warstwa „Persistence” - Zajmuje się przechowywaniem i pobieraniem danych z bazy danych lub innych magazynów danych. W tej warstwie definiowane są klasy i narzędzia, które umożliwiają warstwie aplikacji dostęp do danych.
- 3) Warstwa „Application” – Zawiera logikę biznesową aplikacji, czyli wszystko, co związane jest z przetwarzaniem danych i realizacją funkcjonalności. W tej warstwie znajdują się modele danych, funkcje przetwarzające dane i zasoby, takie jak bazy danych, pliki i usługi sieciowe.
- 4) Warstwa „Presentation” w realizowanym systemie składa się z dwóch projektów:
  - API (Interfejs programowania aplikacji) - Umożliwia komunikację między aplikacją a innymi systemami lub aplikacjami poprzez API. W tej warstwie definiowane są klasy i narzędzia, które umożliwiają dostęp do zasobów i funkcjonalności aplikacji poprzez API.
  - UI (Interfejs użytkownika) - odpowiada za prezentowanie informacji użytkownikowi i umożliwia interakcję z systemem. W tej warstwie definiowane są wszelkie elementy interfejsu użytkownika, takie jak przyciski, pola tekstowe itp.

### **3.2.2. Wzorzec projektowy CQRS**

Wzorzec CQRS (Command Query Responsibility Segregation) to podejście architektoniczne, które rozdziela operacje zapisu (komendy) od operacji odczytu (zapytania) w systemie informatycznym. Wzorzec ten wprowadza podział między modelami zapisu (command model) a modelami odczytu (query model), aby lepiej odzwierciedlić specyficzne wymagania i zachowania aplikacji [18].

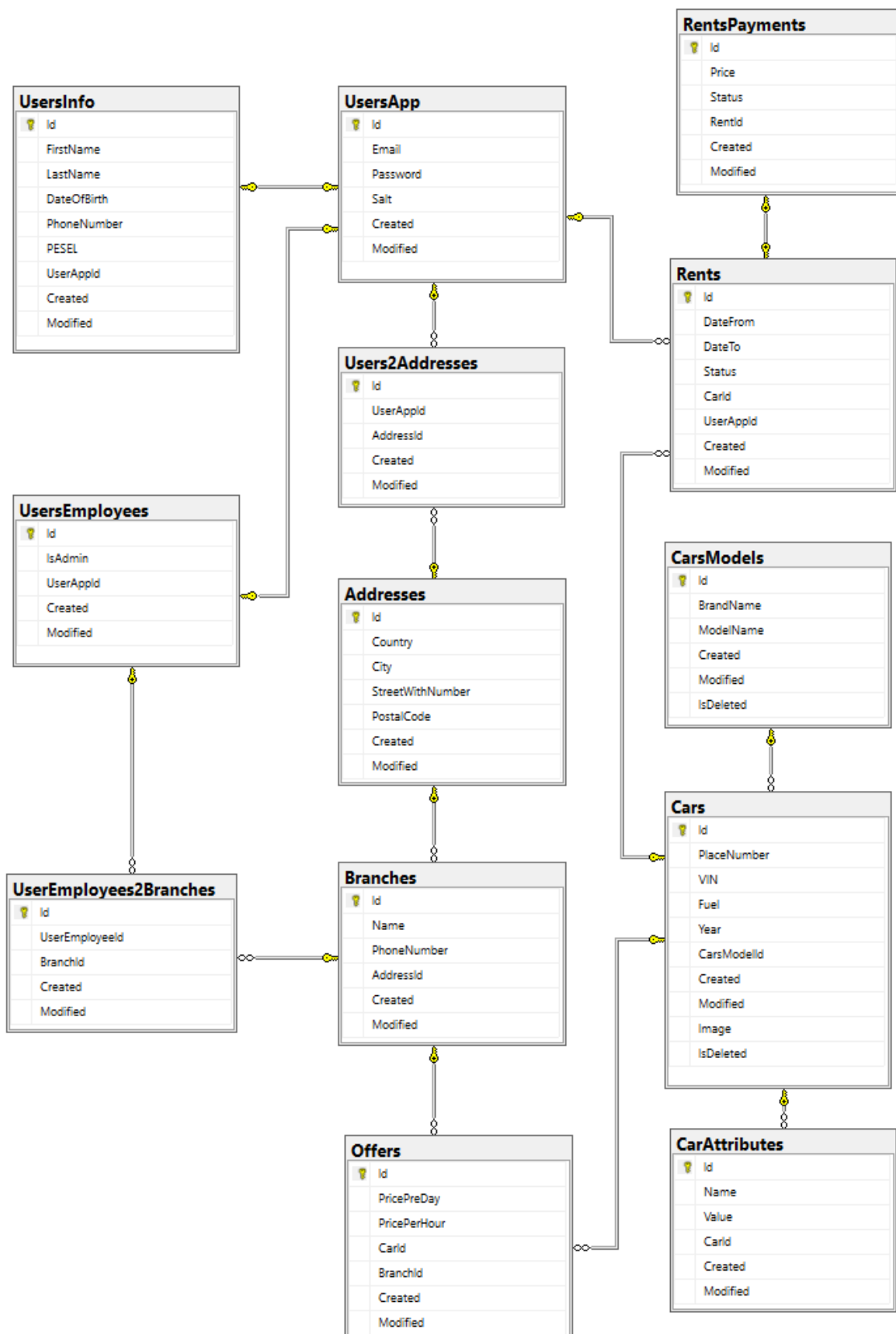
### **3.2.3. Wzorzec projektowy Dependency Inversion**

Dependency inversion odnosi się bezpośrednio do sposobu, w jaki klasy w aplikacji są ze sobą powiązane. W tradycyjnym podejściu klasy wyższego poziomu zależą od klas niższego poziomu. Na przykład, w aplikacji związanej z przetwarzaniem danych klasa odpowiedzialna za logikę biznesową może zależeć od klasy odpowiedzialnej za dostęp do bazy danych [19].

### 3.3. Baza danych

W ramach projektu zastosowano Microsoft SQL Server, jako narzędzie do zarządzania bazą danych. Baza danych odgrywa istotną rolę w przechowywaniu i zarządzaniu różnymi informacjami dotyczącymi samochodów, klientów, rezerwacji, płatności i innych aspektów tego systemu. W bazie danych zostały stworzone i zaimplementowane liczne tabele, które reprezentują różne elementy wypożyczalni samochodów. Przykładowe tabele obejmują: samochody, użytkowników, rezerwacje oraz płatności. Każda tabela ma odpowiednie kolumny, które przechowują konkretne dane, takie jak marka, model, rok produkcji samochodu, dane klienta, daty rezerwacji, kwoty płatności i tym podobne. Rys. 3.3 przedstawia diagram ERD bazy stworzonej podczas realizacji projektu.





Rys. 3.3. Diagram ERD portalu

W celu odwzorowania struktury bazy danych w kodzie, zastosowano Entity Framework Core, popularny framework ORM (Object-Relational Mapping) dla platformy .NET. EF umożliwia mapowanie obiektów zdefiniowanych w kodzie na odpowiednie tabele i relacje w bazie danych. Dzięki EF, operacje CRUD (Create, Read, Update, Delete) na danych można wygodnie wykonywać za pomocą języka LINQ (Language-Integrated Query). Poniżej został przedstawiony proces powstawania tabeli reprezentującej informacje o użytkowniku [20].

Tabela 3.1. Klasa UserInfo

```
public class UserInfo : AuditableEntity
{
    public int Id { get; set; }
    public string FirstName { get; set; } = string.Empty;
    public string LastName { get; set; } = string.Empty;
    public DateTime DateOfBirth { get; set; }
    public string PhoneNumber { get; set; } = string.Empty;
    public string PESEL { get; set; } = string.Empty;

    public int? UserAppId { get; set; }
    public UserApp? UserApp { get; set; }
}
```

Klasa UserInfo (Tabela 3.1) dziedziczy po klasie AuditableEntity i reprezentuje informacje dotyczące użytkownika. Oto opis poszczególnych właściwości tej klasy:

- 1) Id - identyfikator użytkownika, reprezentowany przez wartość całkowitą (int).
- 2) FirstName - imię użytkownika, reprezentowane przez wartość tekstową (string).
- 3) LastName - nazwisko użytkownika, reprezentowane przez wartość tekstową (string).
- 4) DateOfBirth - data urodzenia, reprezentowana wartością daty i czasu (DateTime).
- 5) PhoneNumber - telefon, reprezentowany przez wartość tekstową (string).
- 6) PESEL - numer PESEL, reprezentowany przez wartość tekstową (string).

Dodatkowo, klasa UserInfo zawiera dwie dodatkowe właściwości:

- 1) UserAppId - identyfikator użytkownika, reprezentowany przez wartość całkowitą (int). Stanowi klucz obcy, relacji jeden do jednego z tabelą użytkowników aplikacji.
- 2) UserApp - obiekt reprezentujący wiersz z tabeli użytkowników, na którym EF wykonuje operacje.

Tabela 3.2. Klasa AuditableEntity

```
public class AuditableEntity
{
    public DateTime? Created { get; set; }
    public DateTime? Modified { get; set; }
}
```

Klasa AuditableEntity (Tabela 3.2) jest klasą bazową, która zawiera właściwości typu DateTime służące do audytowania rekordów w bazie danych. Oto opis tych właściwości:

- 1) Created - reprezentuje datę i czas utworzenia obiektu.
- 2) Modified - reprezentuje datę i czas ostatniej modyfikacji obiektu.

Aby framework EF wiedział, że jest to klasa reprezentująca tabelę w pliku kontekstu bazy danych została stworzona kolekcja obiektów UserInfo.

Tabela 3.3. Konfiguracja tabeli UserInfo

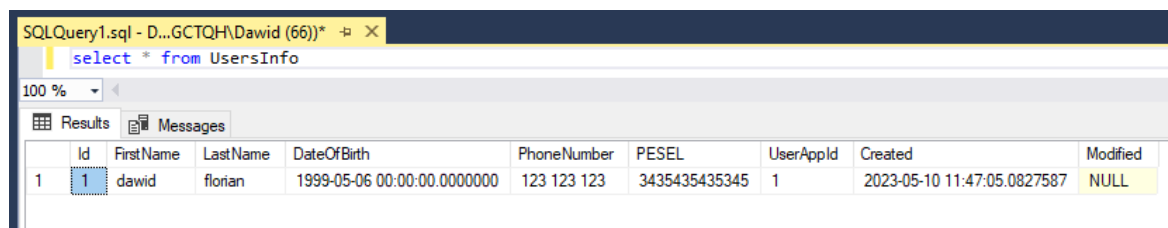
```
public class RentCarsDbContext : DbContext
{
    public DbSet<UserInfo> UserInfo { get; set; }
    ...
}
```

Na podstawie takiej konfiguracji (Tabela 3.3) EF tworzy migracje (Tabela 3.4), która zawiera operacje jakie należy wykonać na bazie danych aby stworzyć tabelę.

Tabela 3.4. Migracja bazy danych

```
migrationBuilder.CreateTable(
    name: "UserInfo",
    columns: table => new
    {
        Id = table.Column<int>(type: "int", nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        ...
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_UserInfo", x => x.Id);
        table.ForeignKey(
            name: "FK_UserInfo_UsersApp_UserAppId",
            column: x => x.UserAppId,
            principalTable: "UsersApp",
            principalColumn: "Id");
    });
```

Na podstawie tej migracji tworzona jest finalna wersja tabeli bezpośrednio w bazie danych. Na rys. 3.4 przedstawiono powstałą tabelę za pomocą polecenia select wykonanego narzędziem SQL Server Management Studio.



SQLQuery1.sql - D:\GCTQH\Nawid (66))*									
select * from UserInfo									
100 %									
Results Messages									
	Id	First Name	Last Name	Date Of Birth	Phone Number	PESEL	User App Id	Created	Modified
1	1	dawid	florian	1999-05-06 00:00:00.0000000	123 123 123	3435435435345	1	2023-05-10 11:47:05.0827587	NULL

Rys. 3.4. Tabela UserInfo wyświetlona za pomocą SSMS

Wszystkie pozostałe tabele oraz relacje zostały stworzone w analogiczny sposób, z wykorzystaniem Entity Framework Core.

## 4. Implementacja systemu

Po ukończeniu fazy projektowania portalu internetowego dla wypożyczalni samochodów, kolejnym etapem było jego implementacja, czyli praktyczne wprowadzenie zaprojektowanych rozwiązań w życie. Po zakończeniu procesu projektowania, przystąpiono do realizacji planów, które zostały opracowane na wcześniejszych etapach. Wdrożenie to kluczowy moment, w którym projekt staje się rzeczywistością i użytkownicy mogą zacząć korzystać z nowego systemu. Podczas implementacji pracowano nad dostosowaniem zaprojektowanych funkcji i interfejsu do rzeczywistych wymagań i ograniczeń. Mogą być wprowadzane poprawki, dostosowania i optymalizacje, aby zapewnić płynne działanie portalu internetowego.

### 4.1. Wybór technologii oraz narzędzi

Podczas implementacji witryny internetowej dla wypożyczalni samochodów, istotne jest dokonanie odpowiedniego wyboru technologii i narzędzi, które będą używane do tworzenia systemu. Poniżej przedstawiamy ważne czynniki, które należy uwzględnić w trakcie tego procesu.

#### 4.1.1. Narzędzia programistyczne

C# to język programowania, rozwinięty przez Microsoft, który służy do tworzenia różnorodnych aplikacji na platformie .NET. Język ten, wprowadzony w 2000 roku jako część .NET Framework, cieszy się obecnie dużą popularnością i jest jednym z głównych języków używanych w tej technologii. Jest to język programowania, który opiera się na programowaniu obiektowym (POO) i oferuje wsparcie dla tworzenia klas, obiektów, interfejsów, dziedziczenia, polimorfizmu i innych koncepcji związanych z POO. Jego cechy są zainspirowane językami C i C++, co sprawia, że jest relatywnie łatwy do nauki dla programistów z doświadczeniem w tych językach.

Główne cechy C# obejmują:

- 1) Silne typowanie: jest językiem, w którym wszystkie operacje muszą być wykonane na odpowiednich typach danych, co zapewnia bezpieczeństwo typów. Dzięki temu ryzyko błędów związanych z niezgodnością typów jest zmniejszone.
- 2) Zarządzanie pamięcią: wykorzystuje mechanizm zarządzania pamięcią znanym jako "garbage collection" (zbieranie nieużywanej pamięci). Programista nie musi ręcznie zarządzać pamięcią, co ułatwia unikanie wycieków pamięci i błędów związanych z niepoprawnym zarządzaniem nią.

- 3) Wielo platformowość: może być używany do tworzenia aplikacji na różnych platformach, takich jak Windows, Linux i macOS. Dzięki platformom takim jak .NET Core (wieloplatformowa wersja .NET Framework) oraz .NET 5, programiści mogą tworzyć aplikacje C# na różnych systemach operacyjnych.
- 4) Bogate biblioteki: udostępnia szeroki zakres bibliotek klas, które zawierają gotowe komponenty do różnych zadań, takich jak obsługa sieci, operacje na plikach, tworzenie interfejsów graficznych użytkownika i wiele innych. Biblioteki takie jak .NET Framework i .NET Core zapewniają bogactwo funkcji, które mogą być wykorzystane w aplikacjach.
- 5) Rozszerzalność: Język ten umożliwia tworzenie rozszerzeń w postaci bibliotek klas, które mogą być wykorzystane w innych projektach. Dzięki temu programiści mają możliwość tworzenia własnych narzędzi i komponentów, które można wielokrotnie używać.

C# znajduje zastosowanie w różnych dziedzinach, takich jak tworzenie aplikacji desktopowych, aplikacji internetowych, gier komputerowych, usług sieciowych (np. API) i wiele innych.

**JavaScript** to popularny język programowania, który jest powszechnie używany do tworzenia dynamicznych stron internetowych. Powstał w połowie lat 90. i nadal pozostaje jednym z najpopularniejszych języków na świecie.

Główne cechy JavaScript obejmują:

- 1) Skryptowy charakter: jest językiem skryptowym, co oznacza, że jego kod źródłowy jest wykonywany linia po linii w czasie rzeczywistym. Nie wymaga on kompilacji, co umożliwia szybką iterację i testowanie kodu.
- 2) Dynamiczne typowanie: wykorzystuje dynamiczne typowanie, co oznacza, że zmienne nie są związane z konkretnym typem danych. Zmienna może przechowywać wartości różnych typów, a typ zmiennej może być zmieniany w trakcie działania programu.
- 3) Programowanie obiektowe: jest językiem opartym na obiektach, gdzie wszystkie elementy, takie jak funkcje, tablice czy nawet liczby, są obiektami. Programiści mogą tworzyć własne obiekty i korzystać z dziedziczenia, polimorfizmu i innych koncepcji programowania obiektowego.

**HTML** to język znaczników używany do tworzenia stron internetowych. Pozwala na określanie wyglądu i funkcjonalności różnych elementów w ramach projektowanej aplikacji. Jest to powszechnie stosowana technologia w każdej witrynie internetowej. Struktura HTML opiera się na znacznikach, które przeglądarka interpretuje w celu wyświetlania poszczególnych elementów.

**CSS** jest językiem, który służy do opisywania wyglądu i formatowania elementów na stronie internetowej. Pozwala na precyzyjne określanie wyglądu poszczególnych elementów, umożliwiając łatwą i elastyczną zmianę wyglądu witryny zgodnie z wymaganiami projektu. CSS może być umieszczony bezpośrednio w pliku HTML lub w osobnym pliku, który jest dołączany do strony. Taka separacja umożliwia wygodne oddzielenie aspektów związanych z wyglądem od struktury i zawartości witryny, co ułatwia zarządzanie stylem i utrzymanie spójności projektu.

**SQL** to język programowania, który służy do zarządzania relacyjnymi bazami danych. SQL pozwala na tworzenie, modyfikowanie oraz usuwanie danych w bazie danych. Jest on wykorzystywany przez większość systemów zarządzania bazami danych (DBMS) takich jak Oracle, MySQL, Microsoft SQL Server, PostgreSQL, itp. Składa się z szeregu poleceń, które umożliwiają na manipulowanie danymi w bazie danych. Polecenia te pozwalają na tworzenie tabel, dodawanie, usuwanie lub modyfikowanie rekordów w tabelach, a także na wykonywanie różnego rodzaju zapytań, które umożliwiają na pobieranie i analizowanie danych. Jest językiem deklaratywnym, co oznacza, że nie opisuje on krok po kroku jak ma działać program, ale raczej co ma zostać osiągnięte. Programista określa co chce zrobić, a silnik bazy danych tłumaczy te instrukcje na konkretne kroki, które muszą zostać wykonane, aby osiągnąć żądane wyniki.

#### **4.1.2. Frameworki**

**.NET Core** to framework programistyczny stworzony przez Microsoft, który umożliwia tworzenie wysoko wydajnych aplikacji na różne platformy, w tym Windows, macOS i Linux. Jest to następca .NET Framework, który został zaprojektowany z myślą o nowoczesnym środowisku programistycznym, cechującym się skalowalnością, wydajnością i elastycznością.

Główne cechy .Net Core obejmują:

- 1) Wielo platformowość: został zaprojektowany tak, aby działać na różnych platformach, umożliwiając tworzenie aplikacji, które są przenośne między systemami Windows, macOS i Linux. To daje programistom większą elastyczność i możliwość dostarczania aplikacji na różne platformy.

- 2) **Wydajność:** .NET Core oferuje doskonałą wydajność dzięki zoptymalizowanemu kompilatorowi Just-in-Time (JIT) oraz możliwości kompilacji AOT (Ahead-of-Time). Dzięki temu aplikacje uruchamiają się szybko i osiągają wysoką wydajność nawet w przypadku złożonych scenariuszy.
- 3) **Modułowość:** opiera się na zasadzie modułowości, co oznacza, że można używać tylko tych elementów frameworka, które są potrzebne do tworzenia danej aplikacji. Dodatkowo, można dodawać niestandardowe komponenty i biblioteki, aby dostosować framework do specyficznych wymagań projektu.
- 4) **Bezpieczeństwo:** zapewnia wbudowane funkcje bezpieczeństwa, takie jak zarządzanie pamięcią, kontrola dostępu do plików i wiele innych. Framework obsługuje również najnowsze protokoły szyfrowania i podpisu cyfrowego, co pomaga w zabezpieczaniu aplikacji przed zagrożeniami.
- 5) **Obsługa chmury:** .NET Core jest ściśle zintegrowany z platformami chmurowymi, takimi jak Microsoft Azure. Pozwala to programistom na łatwe tworzenie skalowalnych i elastycznych aplikacji, które mogą być łatwo wdrażane i zarządzane w chmurze.

**Blazor WebAssembly** jest frameworkiem programistycznym stworzonym przez Microsoft, który umożliwia tworzenie zaawansowanych aplikacji internetowych przy użyciu języka C# i platformy .NET. To część rodziny frameworków Blazor, która obejmuje również Blazor Server.

Główne cechy Blazor obejmują:

- 1) **Aplikacje klienckie:** Blazor WebAssembly umożliwia uruchamianie aplikacji bezpośrednio w przeglądarce internetowej, bez potrzeby ciągłego łączenia się z serwerem. Kod napisany w C# jest kompilowany do kodu WebAssembly, który jest wykonywany bezpośrednio w przeglądarce. Dzięki temu aplikacje Blazor WebAssembly są w stanie działać offline i zapewniają szybką i interaktywną obsługę użytkownika.
- 2) **Współdzielenie kodu:** Blazor WebAssembly korzysta z tego samego kodu .NET, co Blazor Server, co oznacza, że można używać tych samych bibliotek i komponentów po obu stronach (serwerowej i klienta). To umożliwia programistom ponowne wykorzystanie kodu, przyspieszając rozwój aplikacji i ułatwiając jej utrzymanie.



### 4.1.3. Główne biblioteki

**Entity Framework Core** to narzędzie opracowane przez Microsoft, które jest otwarto źródłowym systemem mapowania obiektowo-relacyjnego (ORM). Dzięki EF Core programiści mogą łatwo przekształcać obiekty napisane w języku C# na struktury danych przechowywane w relacyjnych bazach danych. Framework ten działa z różnymi bazami danych, takimi jak SQL Server, MySQL, SQLite, PostgreSQL i inne. Umożliwia tworzenie, zarządzanie i odpytywanie baz danych, eliminując konieczność bezpośredniego korzystania z złożonych zapytań SQL. Core oferuje szereg funkcji, takich jak migracje kodu, obsługa relacji, obsługa transakcji, optymalizacja wydajności oraz wsparcie dla wielu dostawców baz danych. Jest również niezależny od konkretnej platformy, co oznacza, że może być używany na różnych systemach operacyjnych.

**AutoMapper** to darmowa biblioteka open-source dla języka C#, która ułatwia automatyczne mapowanie danych między różnymi obiektami. Jest szczególnie przydatna w przypadku konieczności przekształcenia obiektów pomiędzy różnymi warstwami aplikacji lub modelami danych. Dzięki AutoMapperowi programiści mogą zdefiniować reguły mapowania między dwoma obiektami, określając, które właściwości powinny zostać skopiowane. Biblioteka obsługuje również bardziej zaawansowane scenariusze, takie jak mapowanie kolekcji, ignorowanie określonych właściwości czy dostosowywanie zachowania mapowania. Korzystanie z AutoMappera przyspiesza proces mapowania obiektów, poprawia czytelność kodu i zmniejsza ilość powtarzającego się kodu w aplikacji. Dzięki temu programiści mogą skupić się na logice biznesowej zamiast tracić czas na ręczne mapowanie danych między obiektami.

**MediatR** to biblioteka open-source dla języka C#, która implementuje wzorzec mediatora, umożliwiającą łatwą komunikację między komponentami aplikacji. Biblioteka została zaprojektowana z myślą o tworzeniu aplikacji zgodnych z zasadami CQRS (Command Query Responsibility Segregation) i rozdzielenia odpowiedzialności. P programistom tworzyć komunikację opartą na zapytaniach (queries) i poleceniach (commands) między różnymi częściami aplikacji. Komponenty aplikacji, takie jak kontrolery, obsługują żądania od użytkownika i przekazują je do mediatora. Mediator następnie przekazuje zapytania lub polecenia do odpowiednich obsługujących je handlerów. Korzystanie z MediatR ułatwia implementację jednolitej i skalowalnej architektury aplikacji. Można łatwo dodawać nowe zapytania i polecenia oraz odpowiednie handlersy bez wpływu na istniejący kod. Biblioteka zapewnia również mechanizmy obsługi błędów, walidacji danych i rozwiązywania konfliktów nazw.

#### 4.1.3. Platformy programistyczne

**Visual Studio Enterprise** jest rozbudowanym środowiskiem programistycznym (IDE), stworzonym przez Microsoft, które dostarcza pełen zestaw funkcji i narzędzi. Jest to jedna z edycji Visual Studio, zapewniająca kompleksowe rozwiązania, mające na celu ułatwienie pracy programistów, zwiększenie ich wydajności oraz umożliwienie tworzenia oprogramowania najwyższej jakości.

Oto kilka kluczowych cech i funkcji Visual Studio Enterprise:

- 1) Kompleksowe środowisko programistyczne (Integrated Development Environment, IDE): Visual Studio Enterprise to zaawansowane i zintegrowane środowisko, które obsługuje różnorodne języki programowania, takie jak C#, C++, Visual Basic, F#, Python, JavaScript i inne. Zapewnia narzędzia do edycji kodu, debugowania, zarządzania projektem, refaktoryzacji, kontroli wersji oraz testowania.
- 2) Profilowanie kodu: Visual Studio Enterprise umożliwia programistom analizę wydajności ich kodu poprzez profilowanie. Dzięki temu mogą zidentyfikować obszary, w których występują opóźnienia lub obciążenia. Profilowanie jest przydatne w optymalizacji aplikacji i poprawie jej wydajności.
- 3) IntelliSense: Visual Studio Enterprise wykorzystuje funkcję IntelliSense, która zapewnia automatyczne uzupełnianie kodu, podpowiedzi składniowe, dokumentację i sugestie, usprawniając proces pisanie kodu. IntelliSense wspomaga programistów w szybkim i precyzyjnym tworzeniu poprawnego kodu.
- 4) Zaawansowane narzędzia debugowania: Visual Studio Enterprise oferuje rozbudowane narzędzia do debugowania, umożliwiające analizę i rozwiązywanie problemów w aplikacjach. Dostępne są funkcje takie jak punkty kontrolne, śledzenie zmiennych, narzędzia diagnostyczne i inne, które pomagają odnaleźć i naprawić błędy w kodzie.
- 5) Zarządzanie cyklem życia aplikacji: Visual Studio Enterprise udostępnia narzędzia służące do zarządzania cyklem życia aplikacji, wspomagające planowanie, śledzenie i raportowanie postępów projektu. Programiści mogą zarządzać zadaniami, śledzić błędy, analizować metryki oraz współpracować w zespole.

Warto zaznaczyć, że Visual Studio Enterprise jest przeznaczone głównie dla dużych zespołów programistycznych i przedsiębiorstw, które potrzebują zaawansowanych narzędzi i funkcji do tworzenia, testowania i zarządzania rozbudowanymi projektami oprogramowania.

**Visual Studio Code** to lekki i rozszerzalny edytor kodu opracowany przez Microsoft. Jest to narzędzie darmowe i dostępne na wielu platformach, które zdobyło popularność wśród programistów.

Podstawowe cechy:

- 1) Dostarcza intuicyjny interfejs użytkownika oraz funkcje edycji kodu.
- 2) Posiada rozbudowaną bazę rozszerzeń.
- 3) Łatwa konfiguracja oraz personalizacja.
- 4) Integracja z systemami kontroli wersji.

VS Code jest cenionym narzędziem w świecie programowania ze względu na swoją elastyczność, rozszerzalność i wsparcie dla szerokiej gamy języków i technologii. Jest często wybierany przez programistów zarówno do małych projektów jak i dużych projektów rozwojowych.

**SQL Server Management Studio** to stworzone przez Microsoft narzędzie służące do zarządzania bazami danych w systemie Microsoft SQL Server. Oferuje kompleksowe funkcje administracyjne i programistyczne, które pozwalają programistom i administratorom efektywnie zarządzać bazami danych SQL Server.

Podstawowe cechy:

- 1) Dostarcza intuicyjny interfejs graficzny.
- 2) Pozwala na wykonywanie i debugowanie zapytań SQL.
- 3) Pozwala na projektowanie baz danych.
- 4) Pozwala na monitorowanie i optymalizację.
- 5) Dostarcza funkcje administracyjne, takie jak np. zarządzanie użytkownikami.

SSMS jest niezastąpionym narzędziem dla osób pracujących z bazami danych SQL Server. Dzięki swoim funkcjom i możliwościom, SSMS ułatwia zarządzanie bazami danych, projektowanie schematów, pisanie zapytań SQL oraz wykonywanie zadań administracyjnych związanych z SQL Server.

**NSwag** to narzędzie stworzone do generowania kodu klienta oraz dokumentacji API na podstawie specyfikacji OpenAPI (dawniej znanej jako Swagger). Dzięki NSwag programiści mogą automatycznie tworzyć klientów HTTP w różnych językach programowania, takich jak C#, TypeScript, JavaScript, Java i inne, bazując na opisie API zawartym w pliku OpenAPI.

## 4.2. Architektura systemu

W projekcie została użyta architektura trójwarstwowa [Rys. 4.1]. W tej architekturze można wyodrębnić trzy główne warstwy: warstwę danych, warstwę logiki biznesowej i warstwę prezentacji.

### 1) Warstwa danych

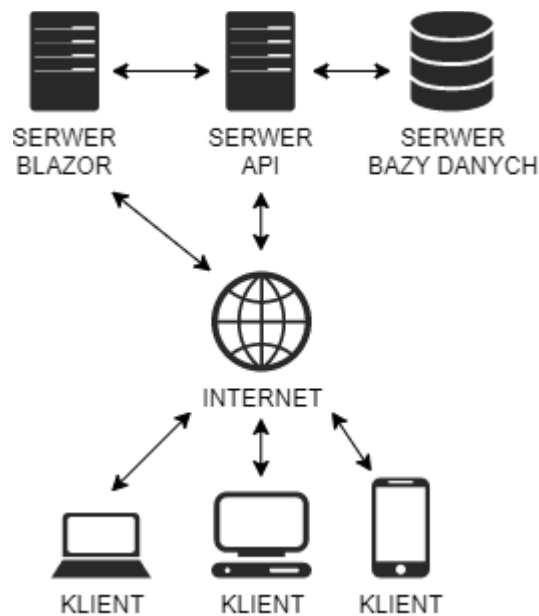
- Serwer bazy danych znajduje się w tej warstwie i jest odpowiedzialny za przechowywanie danych i wykonywanie operacji bazodanowych.
- Serwer bazy danych udostępnia interfejs, który umożliwia dostęp do danych z innych warstw aplikacji, takich jak warstwa logiki biznesowej i warstwa prezentacji.

### 2) Warstwa logiki biznesowej

- Serwer API to główny komponent tej warstwy, jest odpowiedzialny za implementację logiki biznesowej aplikacji. Przyjmuje zapytania od http klienta i przetwarza je, wykonując odpowiednie operacje na danych.
- Serwer API może komunikować się z serwerem bazy danych, pobierając i zapisując dane, które są potrzebne do obsługi zapytań klienta.
- Ta warstwa również zawiera inne komponenty, takie jak serwisy, które wykonują specyficzne zadania biznesowe, walidatory danych, moduły autoryzacji i uwierzytelniania, oraz wszelkie inne elementy wymagane do obsługi logiki biznesowej aplikacji.

### 3) Warstwa prezentacji

- Blazor WebAssembly uruchamia kod w przeglądarce internetowej klienta. Serwer Blazor WebAssembly obsługuje prezentację danych i interakcję użytkownika. Komunikuje się z serwerem API, aby pobierać dane i wykonywać operacje, a następnie renderuje interfejs użytkownika na podstawie tych danych.
- Ta warstwa może również zawierać inne elementy interfejsu użytkownika, takie jak style CSS, skrypty JavaScript, pliki obrazów itp.



Rys. 4.1. Architektura systemu

W tej architekturze (Rys. 4.1) komunikacja odbywa się zazwyczaj w taki sposób, że klient wysyła żądania do serwera Blazor WebAssembly, a ten przekazuje je do serwera API. Serwer API przetwarza żądania, wykonuje odpowiednie operacje na danych w serwerze bazy danych i zwraca odpowiedź do serwera Blazor WebAssembly. Serwer Blazor WebAssembly otrzymuje odpowiedź i renderuje interfejs użytkownika, prezentując dane klientowi. Podsumowując, architektura oparta na serwerze bazy danych, serwerze API i serwerze Blazor WebAssembly stosuje model trójwarstwowy, gdzie każda warstwa ma swoje konkretne zadania i odpowiedzialności. Ta architektura zapewnia skalowalność, bezpieczeństwo i łatwość w utrzymaniu i rozwoju aplikacji.

### 4.3. Przykładowe implementacje w kodzie źródłowym

Poniższa klasa (Tabela 4.1) zawiera metody generujące hash oraz salt hasła. Dzięki tym metodą hasło nie jest zapisywane w bazie danych czystym tekstem.

Tabela 4.1. Kod klasy AuthPasswordHashHandler

```

public static class AuthPasswordHashHandler
{
    public static string HashPassword(string password, string salt)
    {
        SHA256 = SHA256.Create();
        var passwordBytes =
            Encoding.Default.GetBytes($"{password}{salt}");
        var passwordHash = SHA256.ComputeHash(passwordBytes);
        return Convert.ToString(passwordHash);
    }

    public static string SaltGenerator()
    {

```

```

        var rng = RandomNumberGenerator.Create();
        byte[] salt = new byte[32];
        rng.GetNonZeroBytes(salt);
        return Convert.ToHexString(salt);
    }
}

```

Poniższy fragment (Tabela 4.2) implementuje logikę logowania użytkownika do systemu, przy użyciu metody `Generate` zwracającej token.

Tabela 4.2. Kod klasy `SignInCommand`

```

public async Task<BaseResponse<string>>
    Handle(SignInCommand request,
           CancellationToken cancellationToken)
{
    var validator = new SignInCommandValidator();
    var validationResult =
        await validator.ValidateAsync(request);

    if (!validationResult.IsValid)
        return new BaseResponse<string>(validationResult);

    var user =
        await _userRepository
            .GetUserByEmail(request.Email.ToLower());

    if (user == null)
        return new BaseResponse<string>
            ("Nie ma takiego użytkownika", false);

    if (AuthPasswordHashHandler
        .HashPassword(request.Password, user.Salt)
        != user.Password)
        return new BaseResponse<string>("Złe hasło", false);

    var token = Generate(user);

    if (token == null)
        return new BaseResponse<string>
            ("Problem z kontem użytkownika", false);

    return new BaseResponse<string>
        (token, "Poprawnie zalogowano");
}

```

Poniższy fragment (Tabela 4.3) kodu służy do wyliczania kosztu wypożyczenia samochodu na podstawie stawki godzinowej, dniowej oraz dat rozpoczęcia i zakończenia.

Tabela 4.3. Kod klasy `RentPriceCalcHandler`

```

public static class RentPriceCalcHandler
{
    public static decimal CalcPrice(
        DateTime fromDate, DateTime toDate, decimal pricePerHour,
        decimal pricePreDay)
    {
        decimal price = 0;
    }
}

```

```

        var timeRent = toDate - fromDate;

        price += timeRent.Days * pricePreDay;

        price +=
            Math.Min((decimal) (Math.Ceiling(timeRent.TotalHours)
                - (timeRent.Days * 24))
                * pricePerHour, pricePreDay);

        return price;
    }
}

```

Dzięki poniższej nadpisanej klasie `SaveChangesAsync` (Tabela 4.4) jesteśmy w stanie audytować, kiedy dany rekord został zapisany oraz edytowany w bazie danych.

Tabela 4.4. Kod nadpisanej metody `SaveChangesAsync`

```

public override Task<int> SaveChangesAsync(
    CancellationToken cancellationToken = new CancellationToken())
{
    foreach (var entry in ChangeTracker
        .Entries<AuditableEntity>())
    {
        switch (entry.State)
        {
            case EntityState.Added:
                entry.Entity.Created = DateTime.Now;
                break;
            case EntityState.Modified:
                entry.Entity.Modified = DateTime.Now;
                break;
        }
    }
    return base.SaveChangesAsync(cancellationToken);
}

```

Poniższy przykład użycia EF (Tabela 4.5) służy do pobrania ofert na podstawie daty wypożyczenia. Zwracane dane są posortowane oraz podzielone na strony co poprawia wydajność systemu.

Tabela 4.5. Kod metody `GetOffers`

```

public async Task<List<Offer>> GetOffers(
    int page, int pageSize, string search)
{
    return await _context.Offers
        .Include(e => e.Car).ThenInclude(ee => ee.CarsModel)
        .Include(e => e.Branch)
        .Where(e => e.Car.IsDeleted == false)
        .AsNoTracking()
        .Where(e => search.IsNullOrEmpty()
            || (e.Car.CarsModel.BrandName
                + " "
                + e.Car.CarsModel.ModelName
                + " "
                + e.Car.VIN

```

```

        + " "
        + e.Car.PlaceNumber).ToLower()
        .Contains(search.ToLower()))
    .Skip(pageSize * (page - 1))
    .Take(pageSize)
    .ToListAsync();
}

```

Poniżej przedstawiono zawartość pliku App.razor (Tabela 4.6), plik ten jest wykonywany na samym początku po przejściu na dany adres witryny.

Tabela 4.6. Kod pliku App.razor

```

<CascadingAuthenticationState>
  <Router AppAssembly="@typeof(App).Assembly">
    <Found Context="routeData">
      <AuthorizeRouteView RouteData="@routeData">
        <NotAuthorized>
          <LayoutView Layout="@typeof(EmptyLayout)">
            <Error H1="Nie masz tu dostępu :c" P="401">
            </Error>
          </LayoutView>
        </NotAuthorized>
      </AuthorizeRouteView>
    </Found>
    <NotFound>
      <LayoutView Layout="@typeof(EmptyLayout)">
        <Error H1="Nie ma takiej strony :c" P="404">
        </Error>
      </LayoutView>
    </NotFound>
  </Router>
</CascadingAuthenticationState>

```

Poniższy komponent (Tabela 4.7) wyświetla banner na stronie głównej.

Tabela 4.7. Kod komponentu banneru

```

<div id="mainbanner">
  <div id="mbup"></div>
  <div id="mbdown">
    <div id="mbdt"></div>
    <div id="mbdhero" class="p-4">
      <div id="hero" class="pe-3">
        <h1>@H1</h1>
        <h4 class="text-end tx-c3">@H4</h4>
      </div>
    </div>
  </div>
</div>

@code {
  [Parameter]
  public string H1 { get; set; } = string.Empty;

  [Parameter]
  public string H4 { get; set; } = string.Empty;
}

```



Poniższy kontroler (Tabela 4.8) obsługuje endpointy zarządzania pracownikami.

Tabela 4.8. Kod kontrolera EmployeesController

```
[Authorize(Roles = "Admin")]
[Route("api/[controller]/[action]")]
[ApiController]
public class EmployeesController : ControllerBase
{
    private readonly IMediator _mediator;

    public EmployeesController( IMediator mediator)
    {
        _mediator = mediator;
    }

    [HttpPost]
    public async Task<BaseResponse<List<GetEmployeesQueryVM>>>
        GetEmployees([FromBody] GetEmployeesQuery request)
    {
        return await _mediator.Send(request);
    }

    [HttpPost]
    public async Task<BaseResponse> ChangeAdminStatus(
        [FromBody] ChangeAdminStatusCommand request)
    {
        return await _mediator.Send(request);
    }

    [HttpPost]
    public async
        Task<BaseResponse<List<GetUsersNoEmployeeQueryVM>>>
        GetUsersNoEmployee(
            [FromBody] GetUsersNoEmployeeQuery request)
    {
        return await _mediator.Send(request);
    }

    [HttpPut]
    public async Task<BaseResponse> AddEmployee(
        [FromBody] AddEmployeeCommand request)
    {
        return await _mediator.Send(request);
    }

    [HttpDelete]
    public async Task<BaseResponse> DeleteEmployee(
        [FromBody] DeleteEmployeeCommand request)
    {
        return await _mediator.Send(request);
    }
}
```

Poniższy kod (Tabela 4.9) definiuje, sposób walidacji pól formularza logowania.

Tabela 4.9. Walidator klasy SignInCommand

```
public class SignInCommandValidator
    : AbstractValidator<SignInCommand>
{
}
```

```

public SignInCommandValidator()
{
    RuleFor(x => x.Email)
        .EmailAddress()
        .WithMessage("Podaj poprawny email")
        .NotEmpty()
        .WithMessage("Musisz podać email");

    RuleFor(x => x.Password)
        .NotEmpty()
        .WithMessage("Musisz podać hasło");
}
}

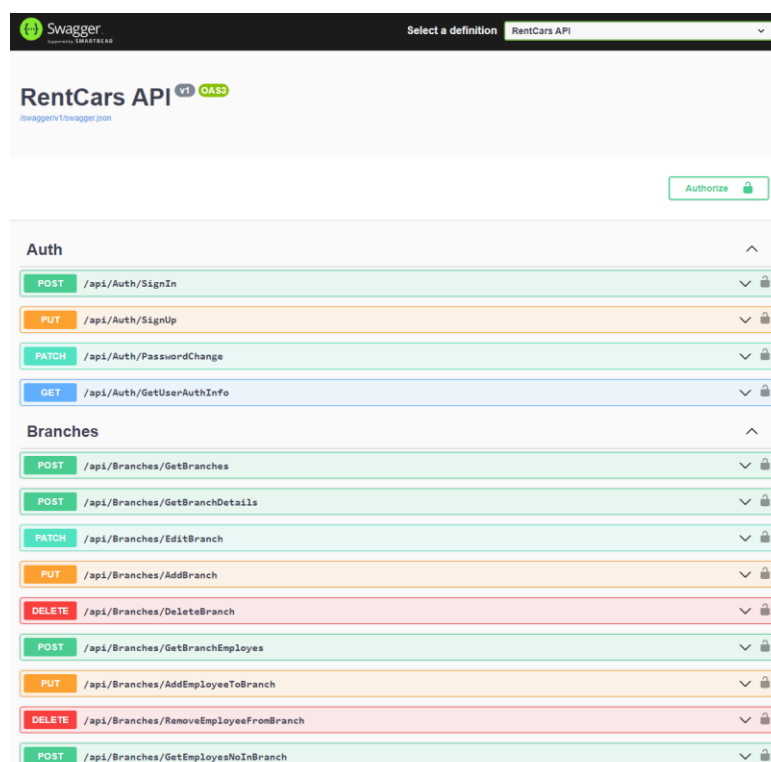
```

## 4.4. Testowanie systemu

Podczas implementacji portalu internetowego, przeprowadzanie testów odgrywa istotną rolę w gwarantowaniu wysokiej jakości systemu. Testowanie jest niezbędne do sprawdzenia, czy poszczególne funkcjonalności systemu działają poprawnie i wydajnie, wykrycia ewentualnych błędów i ich naprawa wpływa na końcową satysfakcję użytkowników korzystających z systemu.

### 4.4.1. Testowanie Api

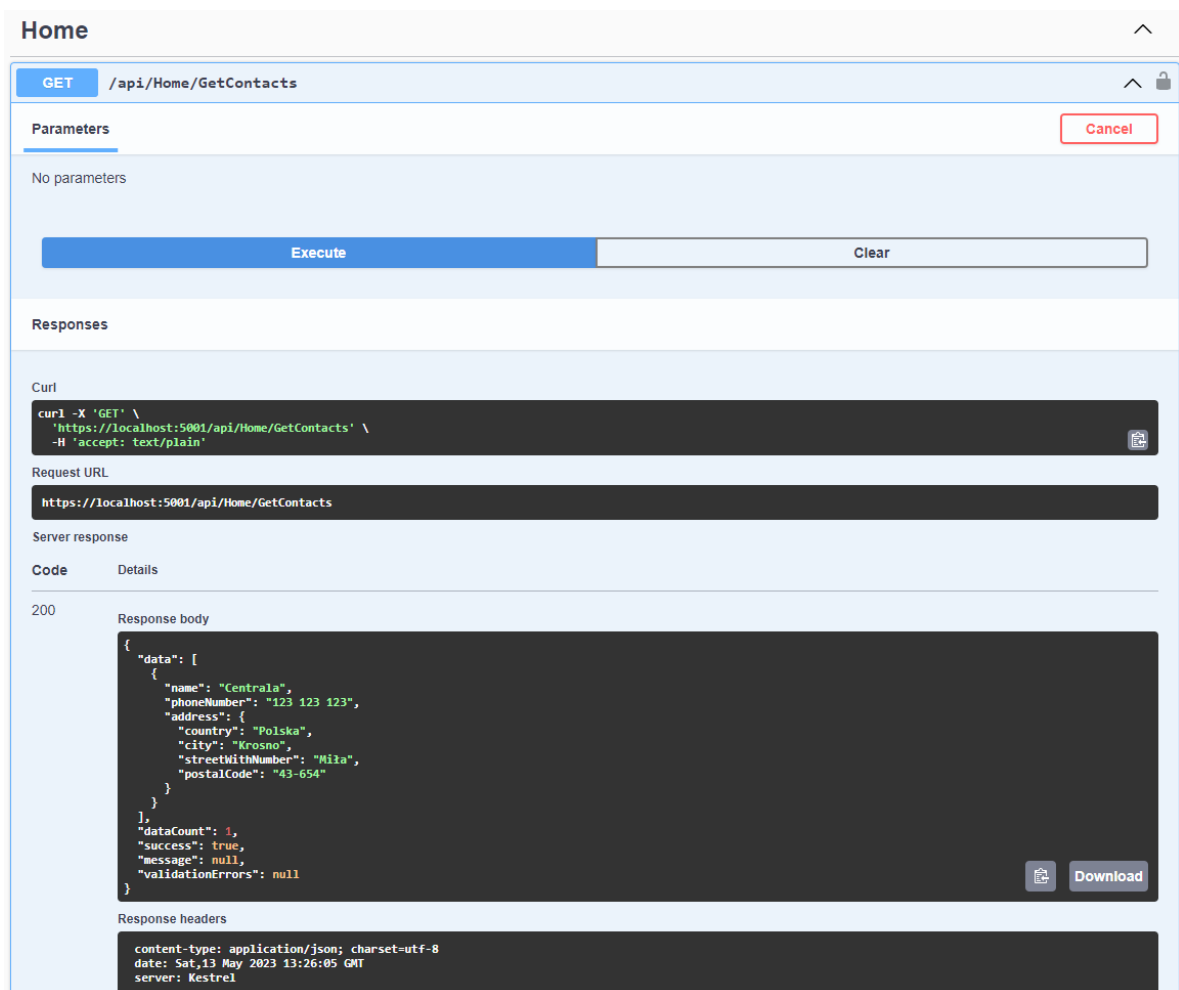
Swagger jest popularnym narzędziem, które jest szeroko wykorzystywane do tworzenia, dokumentowania i testowania interfejsów API. Jego intuicyjny interfejs graficzny (Rys. 4.2) umożliwia programistom oraz testerom łatwe zrozumienie i eksplorację dostępnych punktów końcowych API.



Rys. 4.2. Interfejs graficzny swaggera

Oto kilka istotnych aspektów związanych z testowaniem API przy użyciu Swaggera:

- 1) Dokumentacja API: Swagger umożliwia tworzenie szczegółowej dokumentacji, która opisuje wszystkie dostępne punkty końcowe, parametry, nagłówki, odpowiedzi oraz inne ważne informacje. Ta dokumentacja stanowi podstawę dla procesu testowania i stanowi wartościowe źródło informacji dla programistów oraz testerów.
- 2) Testowanie żądań i odpowiedzi: Swagger umożliwia tworzenie oraz wysyłanie żądań HTTP bezpośrednio z interfejsu (Rys. 4.3). Użytkownik ma możliwość określenia metod HTTP, parametrów, treści żądań oraz oczekiwanych odpowiedzi. Po wysłaniu żądania, Swagger prezentuje otrzymaną odpowiedź, co umożliwia weryfikację poprawności działania API.
- 3) Walidacja danych: Swagger może być również używany do wspomagania walidacji danych wejściowych i wyjściowych API. Poprzez zdefiniowanie schematów danych, Swagger jest w stanie sprawdzać, czy przesyłane dane są zgodne z określonymi regułami i wymaganiami.



Rys. 4.3. Testowanie endpointu przy pomocy swaggera

#### 4.4.2. Testowanie interfejsu graficznego

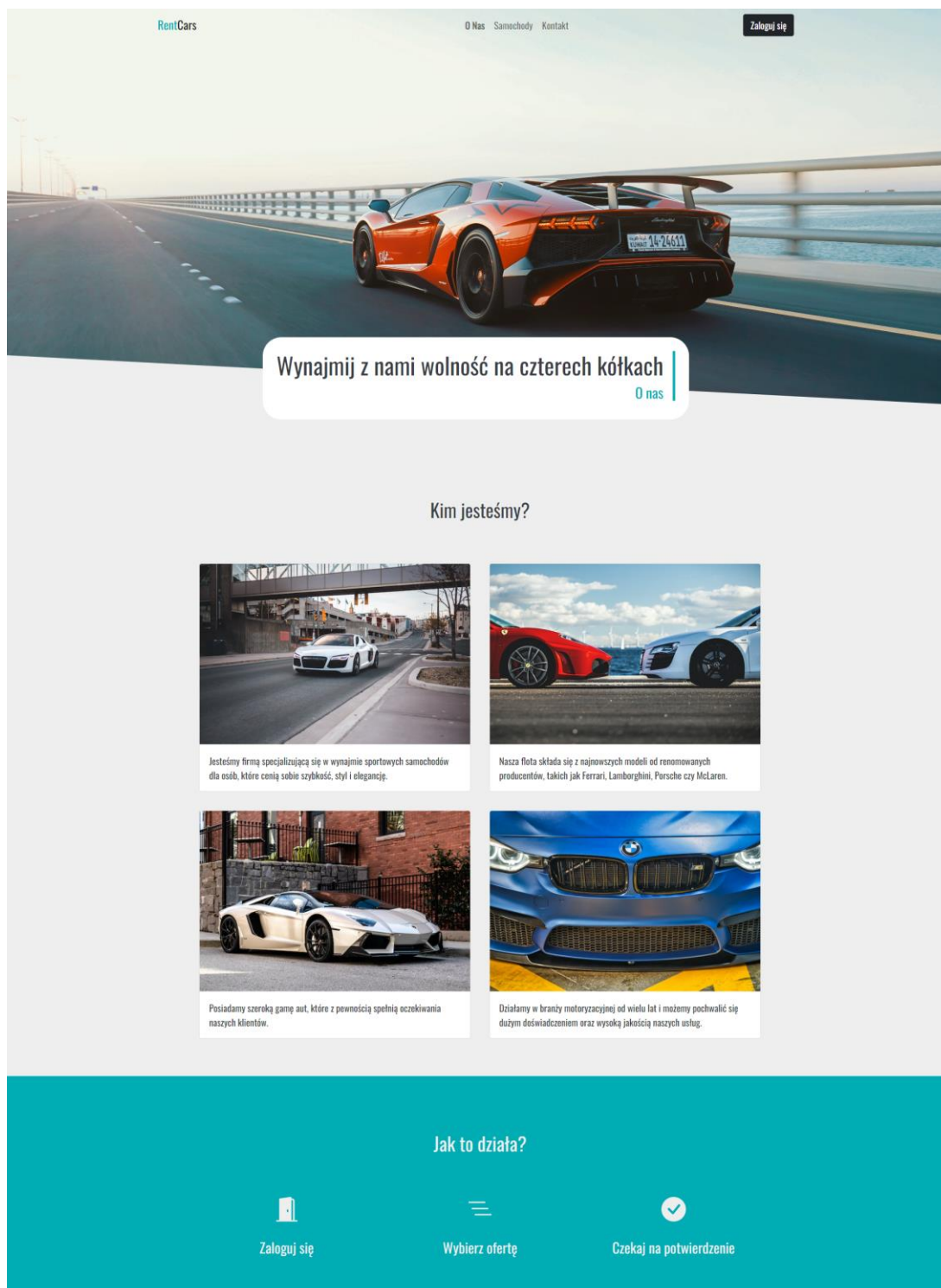
Testowanie interfejsu graficznego (UI) stanowi istotny element procesu tworzenia aplikacji lub systemu, skupiający się na ocenie intuicyjności, responsywności i spełniania oczekiwań użytkowników podczas interakcji z interfejsem. Poniżej przedstawiamy kilka aspektów, które są kluczowe podczas testowania interfejsu graficznego:

- 1) Weryfikacja funkcjonalności: Podczas testowania interfejsu graficznego, istotne jest sprawdzenie, czy wszystkie funkcjonalności są dostępne i działają zgodnie z oczekiwaniami. Przykładowe scenariusze testowe mogą obejmować klikanie przycisków, wprowadzanie danych do formularzy oraz nawigację między stronami. Weryfikacja funkcjonalności zapewnia, że interfejs poprawnie reaguje na działania użytkownika i umożliwia pełne korzystanie z aplikacji.
- 2) Responsywność i wydajność: Interfejs graficzny powinien cechować się responsywnością, czyli szybkim reagowaniem na akcje użytkownika. Testowanie responsywności polega na sprawdzeniu, czy interfejs reaguje bez opóźnień, a wczytywanie stron i elementów odbywa się sprawnie. Warto również przetestować interfejs na różnych urządzeniach i przeglądarkach, aby upewnić się, że wygląd i zachowanie są spójne na wszystkich platformach.
- 3) Walidacja danych i komunikaty: Testowanie interfejsu graficznego obejmuje również weryfikację poprawności walidacji wprowadzanych danych. Istotne jest, aby interfejs wyświetlał odpowiednie komunikaty błędów i ostrzeżeń w przypadku nieprawidłowych danych. Dzięki temu użytkownik otrzymuje informacje o sposobie poprawy błędów i unikania potencjalnych problemów.
- 4) Kompatybilność: Interfejs graficzny powinien być kompatybilny z różnymi przeglądarkami internetowymi, systemami operacyjnymi i urządzeniami. Testowanie kompatybilności polega na sprawdzeniu, czy interfejs wygląda i działa poprawnie na różnych platformach. Warto również upewnić się, że interfejs jest dostosowany do różnych rozdzielczości.

## 5. Prezentacja systemu

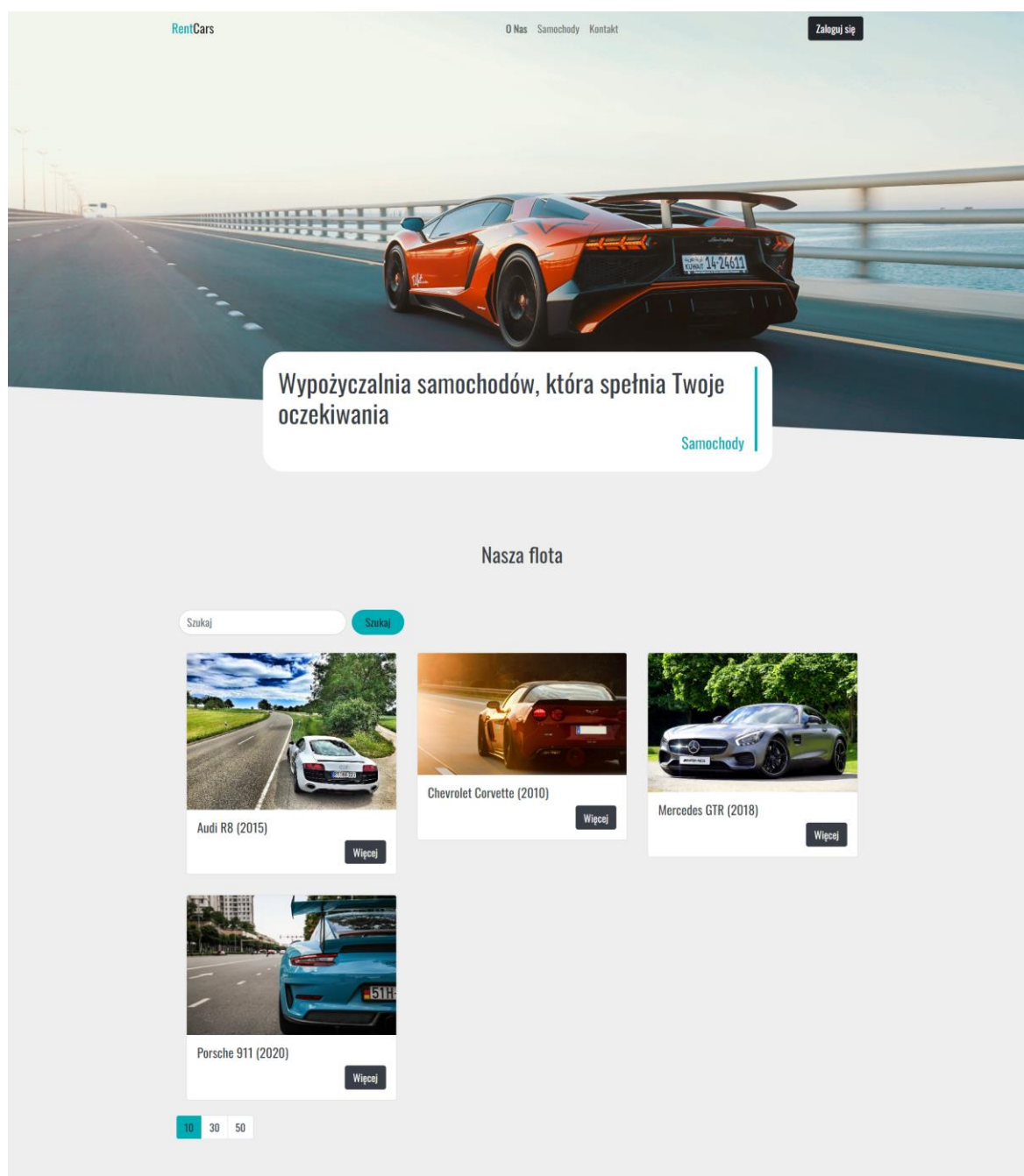
W tym rozdziale zaprezentowano gotową aplikację wraz z omówieniem jej działania. Przedstawione zostały poszczególne funkcjonalności, które umożliwiają użytkownikom łatwe korzystanie z aplikacji.

Przechodząc na adres witryny, użytkownik widzi stronę główną, przedstawioną na rys. 5.1 wraz z informacjami o firmie, ta podstrona pełni rolę reklamową oraz reprezentacyjną firmy.



Rys. 5.1. Podstrona „O Nas”

Rys. 5.2 przedstawia prezentację samochodów dostępnych do wypożyczenia, bez konieczności logowania do systemu.



Rys. 5.2. Podstrona „Samochody”

Klikając w przycisk „Więcej” umieszczony na karcie danego pojazdu, powoduje wyświetlenie szczegółowych informacji o pojeździe oraz ofert. Widok ten został zaprezentowany na rys. 5.3.

RentCars
O Nas Samochody Kontakt
Zaloguj się

Wypożyczalnia samochodów, która spełnia Twoje oczekiwania

Samochody

Mercedes GTR

**Dane techniczne**

ROK PRODUKCJI 2018  
RODZAJ PALIWA BENZYNNA  
NAPĘD RWD  
SKRZYŃNIA BIEGÓW AUTOMATYCZNA  
KOLOR SZARY  
NADWOZIE COUPE  
MOC 600KM

**Oferta**

Centrala

Cena za godzinę	200
Cena za dzień	1000

Wroc

Rys. 5.3. Szczegóły samochodu

Aby przejść na ekran logowania lub rejestracji wystarczy użyć przycisku „Zaloguj się”, wyświetlony zostanie formularz z pełną walidacją pól pokazany na rys. 5.4. W poniższym przykładzie użytkownik podczas tworzenia konta podał email już istniejący w systemie.



### Utwórz swoje nowe konto

dawidflorian99@gmail.com

Dawid

Florian

06.05.1999

123 123 123

25466744352

\*\*\*\*\*

\*\*\*\*\*

Użytkownik z takim emailem już istnieje

Zarejestruj się

### Posiadasz konto?

Nie trać czasu!

Zaloguj się

Rys. 5.4. Walidacja rejestracji

Po zalogowaniu użytkownik zostaje przeniesiony na widok swojego profilu, pokazano na rys. 5.5, widok ten pozwala na edycje informacji o koncie oraz zmianę hasła.

RentCars MENU Mój profil

Mój Profil  
Zarezerwuj  
Moje Zamówienia  
Moje Płatności

Wyloguj się

Informacje podstawowe

Email	dawidflorian99@gmail.com	Telefon	123 123 123
Imię	DAWID	Nazwisko	FLORIAN
PESEL	3435435435345	Data urodzenia	06.05.1999

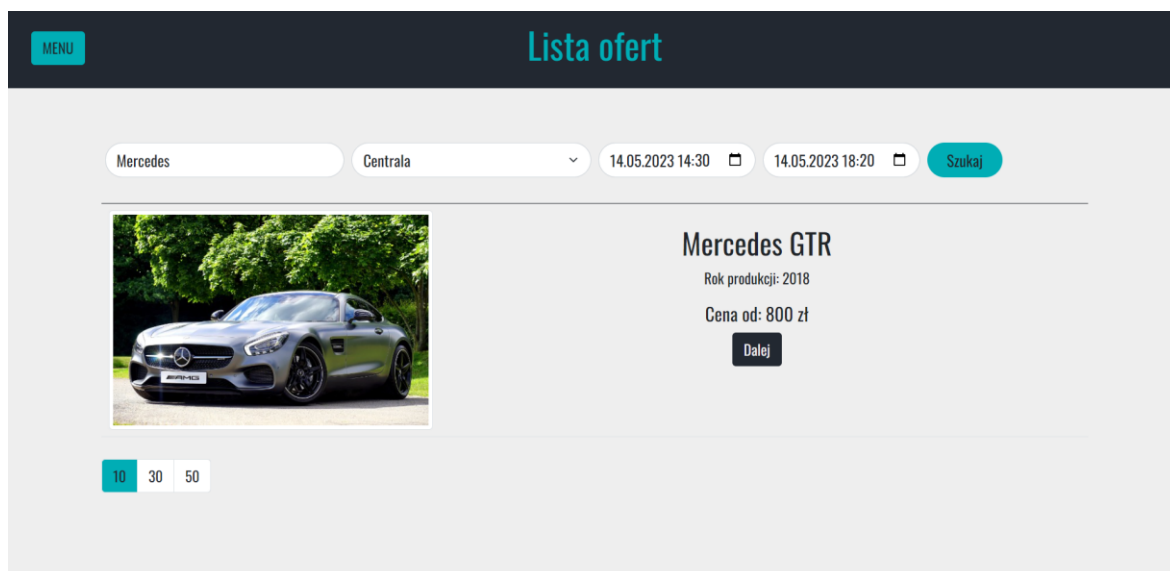
Zapisz

Zmiana hasła

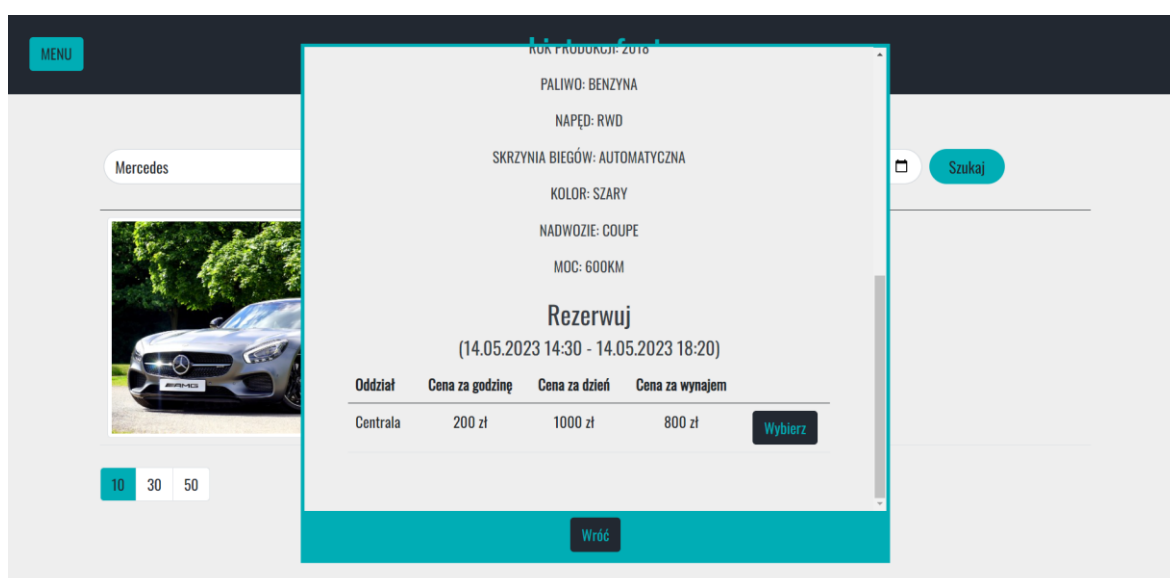
Rys. 5.5. Profil użytkownika

Aby rozpocząć proces rezerwacji samochodu należy wybrać z menu „Zarezerwuj”, wyświetlona zostanie strona wyszukiwania ofert przedstawiona na rys. 5.6, po wpisaniu zakresu dat oraz filtrów cena na ofercie zostanie automatycznie przeliczona. Po kliknięciu „Dalej” zostaną zaprezentowane informacje o samochodzie oraz dostępne oferty w poszczególnych oddziałach, widok przedstawiono na rys 5.7. Aby zarezerwować pojazd w wybranym terminie wystarczy użyć przycisku „Wybierz” przy wybranej ofercie. Rys. 5.8 przedstawia historie zamówień, wyświetlaną po dokonaniu rezerwacji.

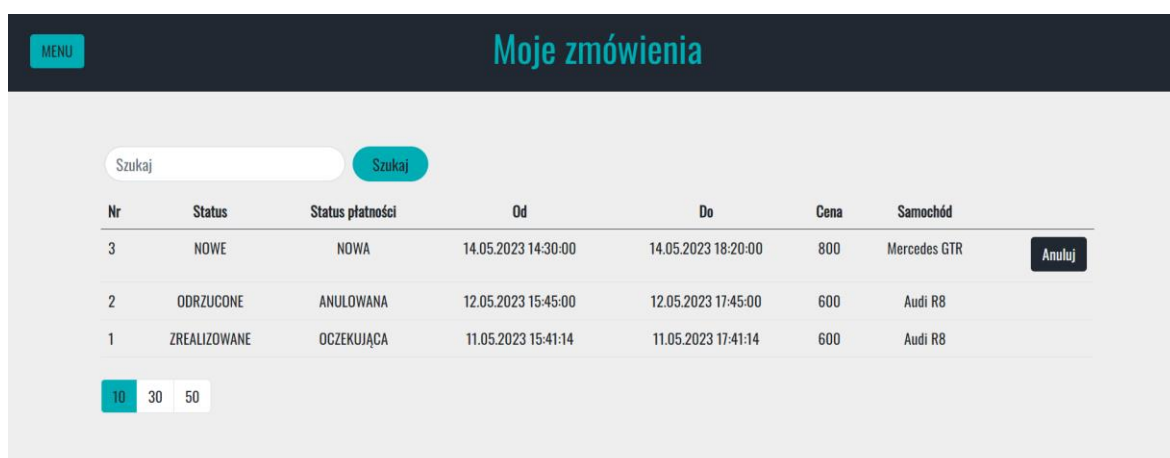




Rys. 5.6. Wyszukiwanie oferty



Rys. 5.7. Dostępne oferty dla samochodu



Rys. 5.8. Historia zamówień

Panel pracownika posiada w menu dodatkowe opcje takie jak: „Zamówienia”, „Samochody”, „Oferty”, „Płatności”, służące do wykonywania swojej pracy. Rys. 5.9 pokazuje elementy dostępne dla pracowników.

Panel pracownika posiada w menu dodatkowe opcje takie jak: „Zamówienia”, „Samochody”, „Oferty”, „Płatności”, służące do wykonywania swojej pracy. Rys. 5.9 pokazuje elementy dostępne dla pracowników.

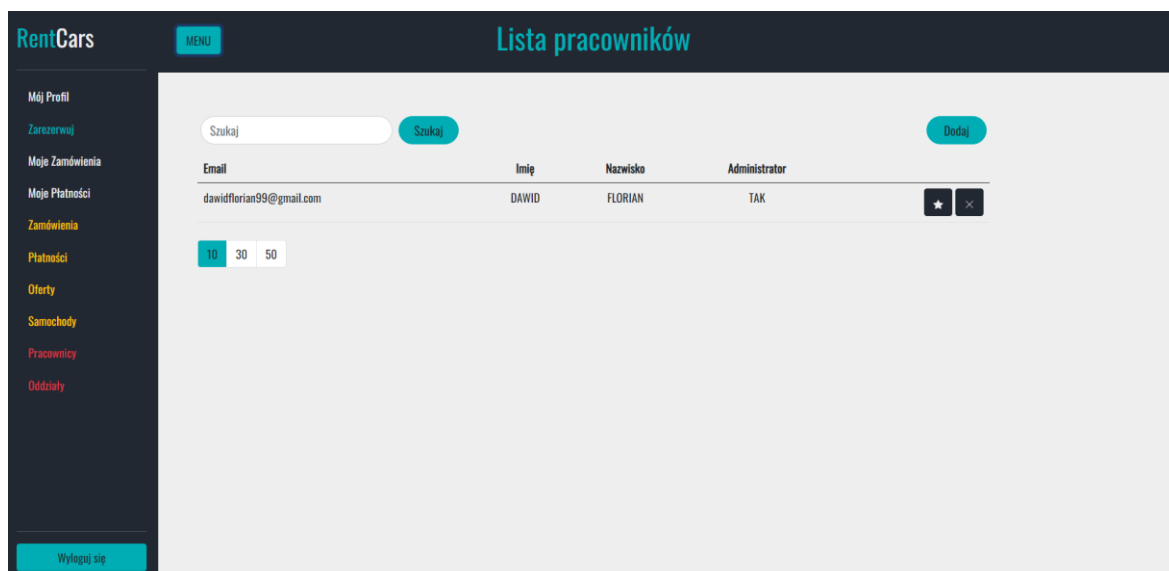
Rys. 5.9. Panel pracownika

Pracownik edytując model samochodu nie musi wchodzić na jego edycję, może to zrobić bezpośrednio z listy samochodów. Pomyślna akcja edycji zaznaczana jest kolorem zielonym co pokazano na zdjęciu poniżej.

Marka	Model	Samochody
Audi	R8	1
Chevrolet	Corvette	1
Mercedes	GTR	1
Porsche	911	1

Rys. 5.10. Edycja samochodu

Panel administratora firmy pozwala na zarządzanie oddziałami i pracownikami oraz ma możliwość przypisywać pracowników do poszczególnych oddziałów co pokazano na rys. 5.11 oraz rys. 5.12.



Rys. 5.11. Panel administratora firmy



Rys. 5.12. Przypisywanie pracownika do oddziału

## 6. Podsumowanie i wnioski

Celem projektu inżynierskiego, było zaprojektowanie oraz stworzenie portalu internetowego dla firmy, umożliwiającego rezerwacje samochodów online. Efektem wykonanych prac jest aplikacja, spełniająca założone wcześniej cele, takie jak: rejestracja użytkowników, logowanie do serwisu, dokonywanie rezerwacji, wykonywanie płatności, zarządzanie własnym kontem, panel pracownika pozwalający na obsługę samochodów, ofert, zamówień oraz płatności, panel administratora firmy pozwalający na zarządzanie oddziałami oraz pracownikami. Największym problemem podczas realizacji projektu była stosunkowo mała ilość dostępnych powszechnie materiałów, ponieważ korzystano z najnowszych technologii takich jak np. Blazor.

Autor za własny wkład pracy uważa następujące elementy:

- 1) Opracowanie projektu i koncepcji portalu: Autor zaproponował plan, wizję portalu internetowego dla firmy oraz zidentyfikował potrzeby i wymagania firmy.
- 2) Analiza rynku i konkurencji: Autor dokonał szczegółowej analizy rynku, usług wynajmu samochodów oraz badania konkurencji. Na podstawie zebranych danych i informacji, autor zidentyfikował mocne i słabe strony istniejących rozwiązań.
- 3) Projektowanie interfejsu użytkownika: Autor zajmował się projektowaniem intuicyjnego i atrakcyjnego interfejsu użytkownika, który zapewnia prostą, czytelną prezentację informacji. Skupił się na zapewnieniu łatwości obsługi, dostosowaniu do różnych urządzeń i systemów operacyjnych.
- 4) Implementacja funkcjonalności: Autor był odpowiedzialny za implementację kluczowych funkcji portalu, takich jak system rezerwacji samochodów, mechanizmy zarządzania flotą oraz możliwość obsługi zamówień. Wykorzystał najnowsze technologie i narzędzia programistyczne w celu zapewnienia wysokiej jakości.
- 5) Testowanie i optymalizacja: Autor przeprowadził liczne testy aby upewnić się, że portal działa poprawnie i spełnia wszystkie wymagania.

## Literatura

- [1] <https://www.enterpriseappstoday.com/stats/car-rental-statistics.html>, Dostęp 2023
- [2] <https://www.alamo.com/en/home.html>, Dostęp 2023
- [3] <https://www.avis.com/en/home>, Dostęp 2023
- [4] <https://www.budget.pl>, Dostęp 2023
- [5] <https://www.dollar.com>, Dostęp 2023
- [6] <https://spectrum.ieee.org/top-programming-languages-2022>, Dostęp 2023
- [7] <https://www.python.org>, Dostęp 2023
- [8] <https://learn.microsoft.com/dotnet/csharp>, Dostęp 2023
- [9] <https://docs.oracle.com/en/java>, Dostęp 2023
- [10] <https://www.php.net/docs.php>, Dostęp 2023
- [11] <https://pl.legacy.reactjs.org>, Dostęp 2023
- [12] <https://learn.microsoft.com/pl-pl/aspnet/core/blazor>, Dostęp 2023
- [13] <https://angular.io>, Dostęp 2023
- [14] <https://vuejs.org>, Dostęp 2023
- [15] <https://www.mysql.com>, Dostęp 2023
- [16] <https://www.microsoft.com/sql-server>, Dostęp 2023
- [17] <https://www.postgresql.org>, Dostęp 2023
- [18] <https://learn.microsoft.com/azure/architecture/patterns/cqrs>, Dostęp 2023
- [19] <https://learn.microsoft.com/dotnet/core/extensions/dependency-injection>, Dostęp 2023
- [20] <https://learn.microsoft.com/en-us/ef/core>, Dostęp 2023

Sygnatura:

POLITECHNIKA RZESZOWSKA im. I. Łukasiewicza  
Wydział Elektrotechniki i Informatyki

Rzeszów, 2023

## **STRESZCZENIE PROJEKTU INŻYNIERSKIEGO**

### **PORTAL INTERNETOWY DLA FIRMY**

Autor: Dawid Florian, nr albumu: EF-ZI-165075

Opiekun: dr inż. Mariusz Gamracki

Słowa kluczowe: Aplikacja internetowa, System rezerwacji samochodów, ASP .Net Core, Blazor WebAssembly, Microsoft SQL Server

Projekt inżynierski "Portal internetowy dla firmy" został zrealizowany w celu stworzenia kompleksowego i funkcjonalnego portalu dla firmy zajmującej się wynajmem samochodów. Portal ten został zaprojektowany w celu ułatwienia procesu rezerwacji samochodów, dokonywania płatności online, przeglądania oferty, zarządzania flotą, przeglądania oferty oraz zapewnienia wygodnej interakcji między firmą a klientami. W rezultacie realizacji projektu powstał produkt, mogący pozwolić firmie wynajmującej samochody, łatwiejsze, bardziej wydajne zarządzanie procesem rezerwacji samochodów. Aplikacja pozwala na elastyczne i wygodne korzystanie z usług firmy, co może wpłynąć na konkurencyjność i efektywność biznesową firmy.

RZESZOW UNIVERSITY OF TECHNOLOGY  
Faculty of Electrical and Computer Engineering

Rzeszow, 2023

## **DIPLOMA THESIS (BS) ABSTRACT**

### **ONLINE PORTAL FOR A COMPANY**

Author: Dawid Florian, code: EF-ZI -165075

Supervisor: Mariusz Gamracki, PhD, Eng.

Key words: Web Application, Car Rental System, ASP .Net Core, Blazor WebAssembly, Microsoft SQL Server

The engineering project "Internet Portal for a Car Rental Company" was implemented to create a comprehensive and functional portal for a car rental company. This portal was designed to facilitate the car reservation process, online payment, browsing the car selection, fleet management, and to provide convenient interaction between the company and its customers. As a result of the project implementation, a product was developed that enables the car rental company to manage the car reservation process more easily and efficiently. The application allows for flexible and convenient utilization of the company's services, which can impact the company's competitiveness and business efficiency.