Projekt Technologie Obiektowe

Konsolowa wypożyczalnia samochodów

Michał Bohun

2025

Spis treści

[1. Cel projektu 2](#_Toc193719555)

[2. Baza danych i struktury danych 3](#_Toc193719556)

[3. Działanie aplikacji 4](#_Toc193719557)

[4. UML 5](#_Toc193719558)

# Cel projektu

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

Wzorce architektoniczne: MVC (Model-View-Controller)

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

Wzorce projektowe:

Dependency Injection – kontrolery nie tworzą samodzielnie instancji klas, lecz otrzymują je przez konstruktor   
  
Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Factory Method – tworzenie obiektów encji (np. Rental) i zarządznie ich cyklem życia.

Singleton – tworzenie klasy odpowiadającej za komunikację z bazą. Jest ona dostępna w całym projekcie a jej instancja jest jednorazowo tworzona.

Obraz zawierający tekst, zrzut ekranu

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.  


# Baza danych i struktury danych

**a) Baza danych i model danych**

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

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

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

**b) Kontrolery**

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

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

**Interakcja z użytkownikiem (widoki)**

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

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

# Działanie aplikacji

1. **Logowanie**
   * Użytkownik podaje login i hasło.
   * System sprawdza poprawność danych w bazie.
   * Jeśli dane są błędne, prosi o ponowne wpisanie.
2. **Funkcje użytkownika**
   * Może przeglądać dostępne samochody.
   * Może wynająć auto, podając jego ID i czas wynajmu.
   * Może sprawdzić swoje aktualne wynajmy.
3. **Funkcje administratora**
   * Może dodawać, edytować i usuwać pojazdy.
   * Może przeglądać dostępne pojazdy.
4. **Zarządzanie wynajmami**
   * Po upływie okresu wynajmu system automatycznie zwalnia pojazd (zmienia jego status na dostępny).
   * Przy wynajmie sprawdzana jest dostępność pojazdu.

# UML

