# Strona Główna

I. Imię Nazwisko autorów: Miłosz Gierus, Mateusz Florian

II. Numer albumu: 127758, 125260

III. Temat projektu: CarDealer

IV. Nazwa przedmiotu: Bazy Danych

V. Grupa laboratoryjna: Grupa 1

VI. Data oddania projektu: 11.06.2024

# Spis treści



# Wprowadzenie

## Cel aplikacji:

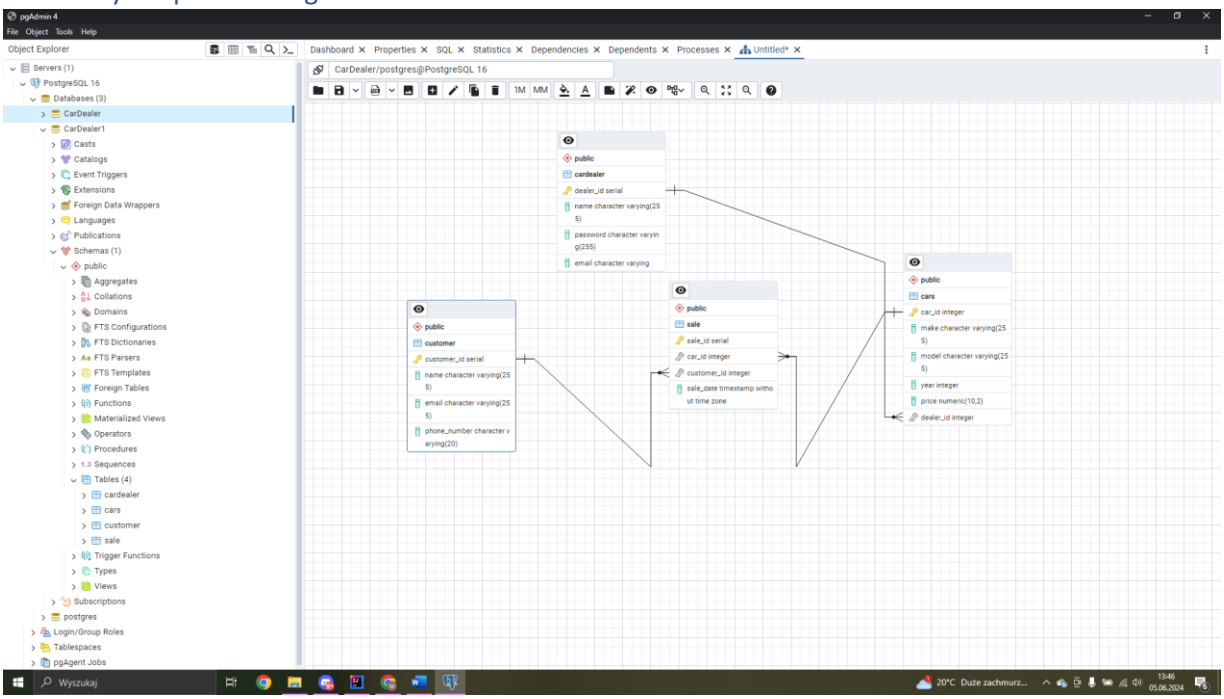
Celem aplikacji jest zapewnienie użytkownikom możliwości zarządzania informacjami dotyczącymi samochodów. Aplikacja umożliwia dodawanie nowych samochodów do bazy danych, edycję istniejących wpisów samochodów oraz usuwanie niepotrzebnych rekordów. Dodatkowo, aplikacja pozwala na wyświetlanie listy samochodów oraz szczegółowych informacji o każdym samochodzie.
Użytkownik ma możliwość rejestracji, ponieważ aby korzystać z aplikacji trzeba być zalogowanym użytkownikiem.

## Wymaganie systemowe:

Aby uruchomić aplikacje potrzebny jest komputer/laptop z dostępnym internetem oraz zainstalowanymi programami takimi jak IntelliJ najnowszej wersji biblioteki JavaFx oraz środowisko PostgreSQL do z importowania bazy danych lub stworzenia nowej z takimi samymi polami i typami danych

- JavaFX version 21: Do tworzenia interfejsu użytkownika.
- Java SDK version 21.0.2: Do pisania funkcjonalności
- PostgreSQL version 16: Do przechowywania danych o czołgach.
- pgAdmin4 version 7.6: Do zarządzania bazą

## Baza danych opis ERD diagram:



*Rysunek 1- Diagram ERD*

Baza zawiera tabele:

Tabela: `cars`

- Klucz główny: `car_id` (typ danych: INT, autoinkrementacja)
- Pola:
    - `car_id`: Unikalny identyfikator samochodu (typ danych: INT)
    - `make`: Marka samochodu (typ danych: VARCHAR)
    - `model`: Model samochodu (typ danych: VARCHAR)
    - `year`: Rok produkcji samochodu (typ danych: INT)
    - `price`: Cena samochodu (typ danych: DECIMAL lub FLOAT)
    - `dealer_id`: Klucz obcy do tabeli `dealers` (typ danych: INT)

2. Tabela: `cardealer`

- Klucz główny: `dealer_id` (typ danych: INT, autoinkrementacja)
- Pola:
    - `dealer_id`: Unikalny identyfikator dealera (typ danych: INT)
    - `name`: Nazwa dealera (typ danych: VARCHAR)
    - password: Hasło dealera (typ danych: VARCHAR)
    - email: email dealera (typ danych: VARCHAR)

2. Tabela: `customer`

- Klucz główny: `customer_id` (typ danych: INT, autoinkrementacja)
- Pola:
    - `dealer_id`: Unikalny identyfikator dealera (typ danych: INT)
    - `name`: Nazwa klienta (typ danych: VARCHAR)

- email: email klienta (typ danych: VARCHAR)
- phone_number: numer klienta (typ danych: VARCHAR)

2. Tabela: `sale`

- Klucz główny: `sale_id` (typ danych: INT, autoinkrementacja)
- Pola:
  - `sale_id`: Unikalny identyfikator dealera (typ danych: INT)
  - `car_id`: id auta (typ danych: INTEGER)
  - customer_id: id klienta (typ danych: INTEGER)
  - sale_date: data sprzedarzy (typ danych: DATE)

3. Relacje:

- Tabela `cars` zawiera klucz obcy `dealer_id`, który odnosi się do `dealer_id` w tabeli `dealers`. To oznacza, że każdy samochód jest przypisany do konkretnego dealera.

# Instalacja

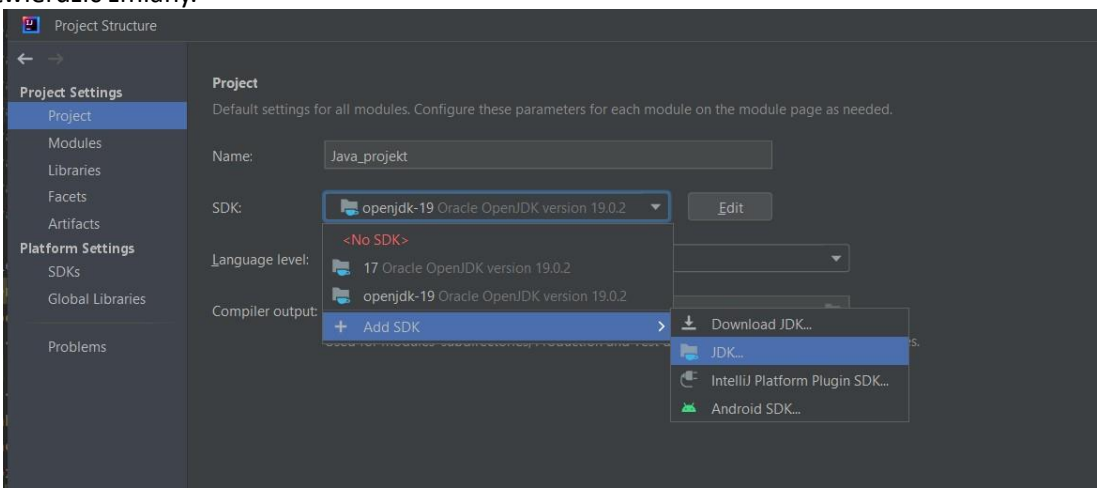
## Instalacja:

Aby korzystać z aplikacji trzeba podjąć następujące kroki:

✞ Projekt należy otworzyć w środowisku intelliJ w najnowszej wersji, można pobrać z strony: https://www.jetbrains.com/idea/download/?section=windows

✞ Zainstalować Java Development Kit (JDK)

Pobrać z strony: https://www.oracle.com/pl/java/technologies/downloads/#jdk21-windows w zależności od systemu operacyjnego z którego korzystamy. Rekomendowana wersja JDK to Java 8 lub nowsza. Aby dodać JDK do projektu należy kliknąć prawym przyciskiem myszy na projekt wybrać Open Module Settings następnie wybrać Projekt i w SDK dać ścieżkę do wcześniej pobranego pliku i zatwierdzić zmiany.

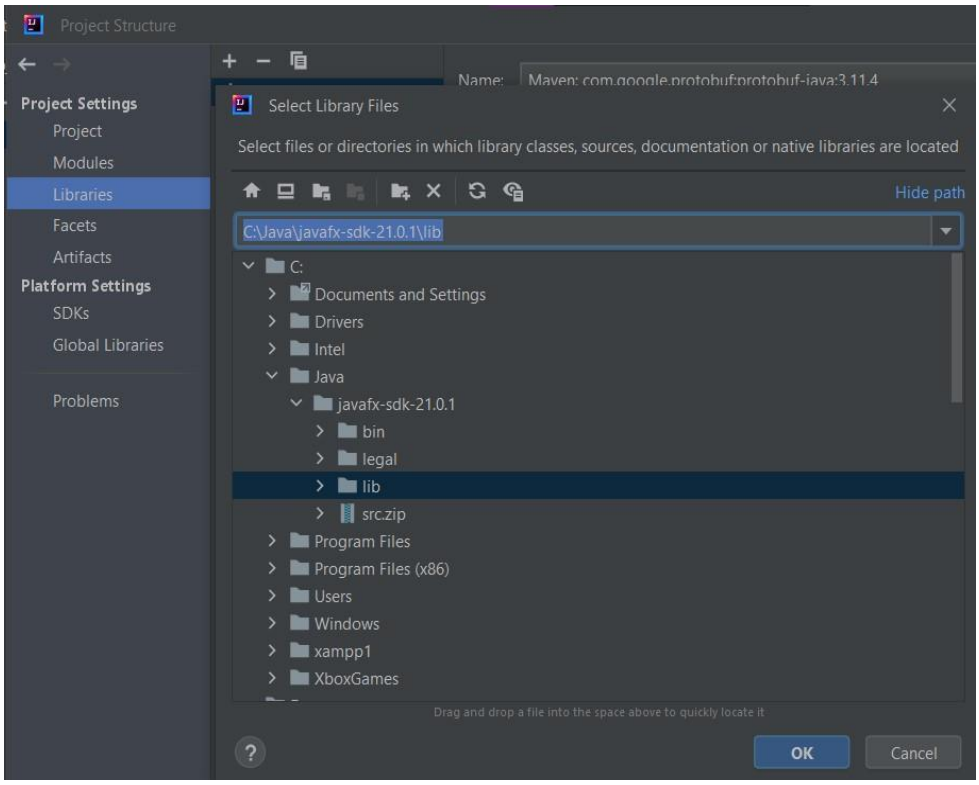



*Rysunek 2 -instalacja*

✞ Zainstalować JavaFx:

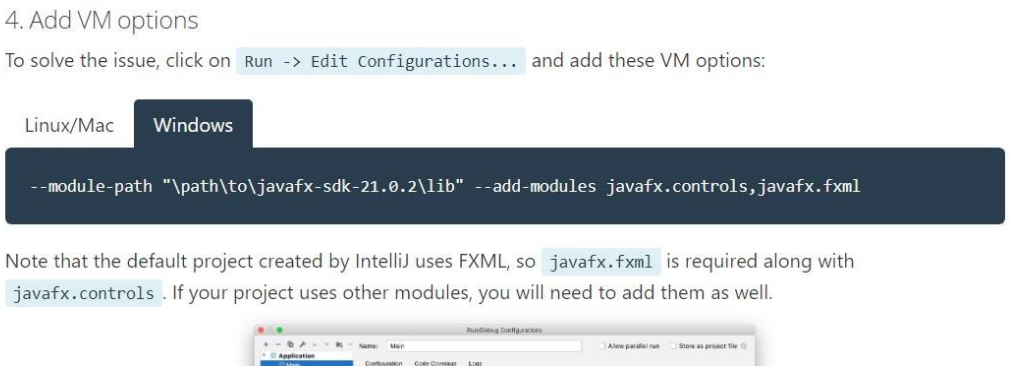
Należy pobrać javaFx z strony: https://gluonhq.com/products/javafx/ dla swojego systemu operacyjnego typu SDK w wersji 21.0.2


Aby projekt działał trzeba dodać odpowiednie biblioteki dla JavyFx z wcześniej pobranego pliku można wybrać taki folder lib gdzie znajdują się odpowiednie biblioteki.

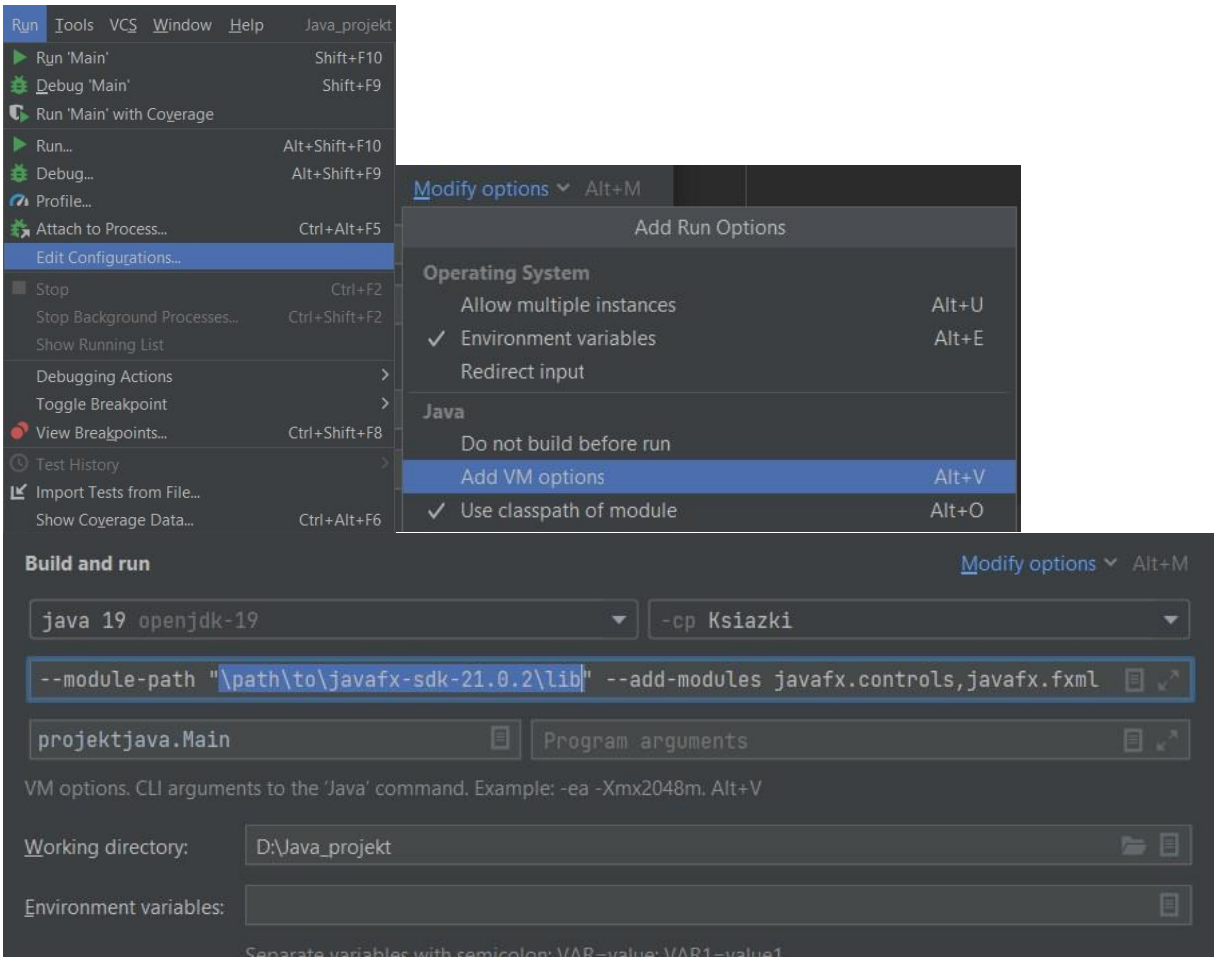



*Rysunek 3- instalacja*

Kopiujemy z strony https://openjfx.io/openjfx-docs/#install-javafx ścieżkę dla swojego systemu operacyjnego i dodajemy ją do IntelliJ




*Rysunek 4 - instalacja*




*Rysunek 5- instalacja*

W zaznaczonym miejscu dodać ścieżkę do biblioteki JavaFx z wcześniej popranego pliku podobnie jak krok wyżej.

✞ Zainstalować postgres i dodać bazę danych

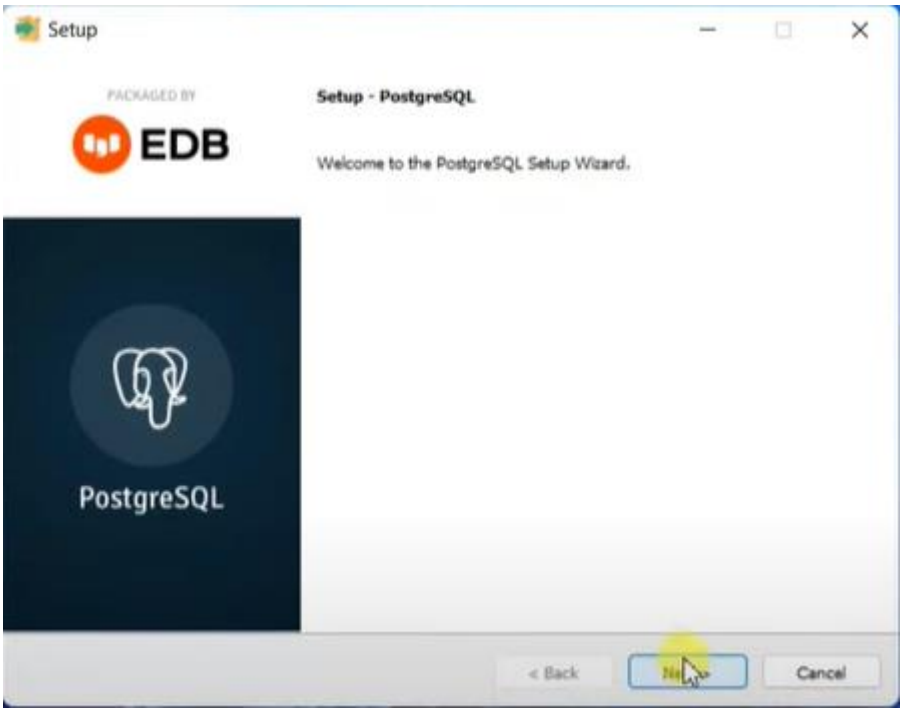
Poprać najnowszą wersje postgreSQL dla swojego systemu operacyjnego z strony

https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

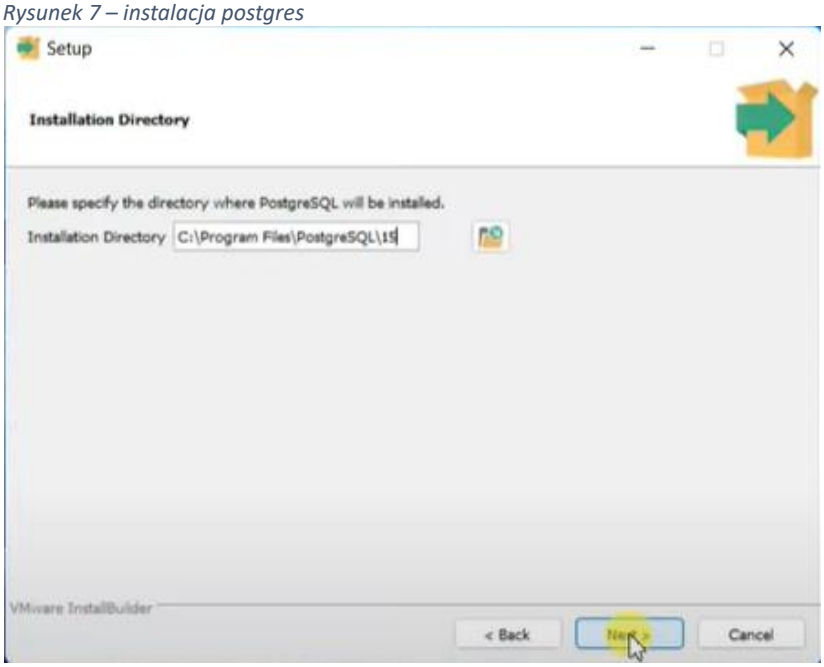
Instalacja krok po kroku:

| PostgreSQL Version | Linux x86-64 | Linux x86-32 | Mac OS X | Windows x86-64 | Windows x86-32 |
|---|---|---|---|---|---|
| 16.3 | postgresql.org | postgresql.org | | | Not supported |

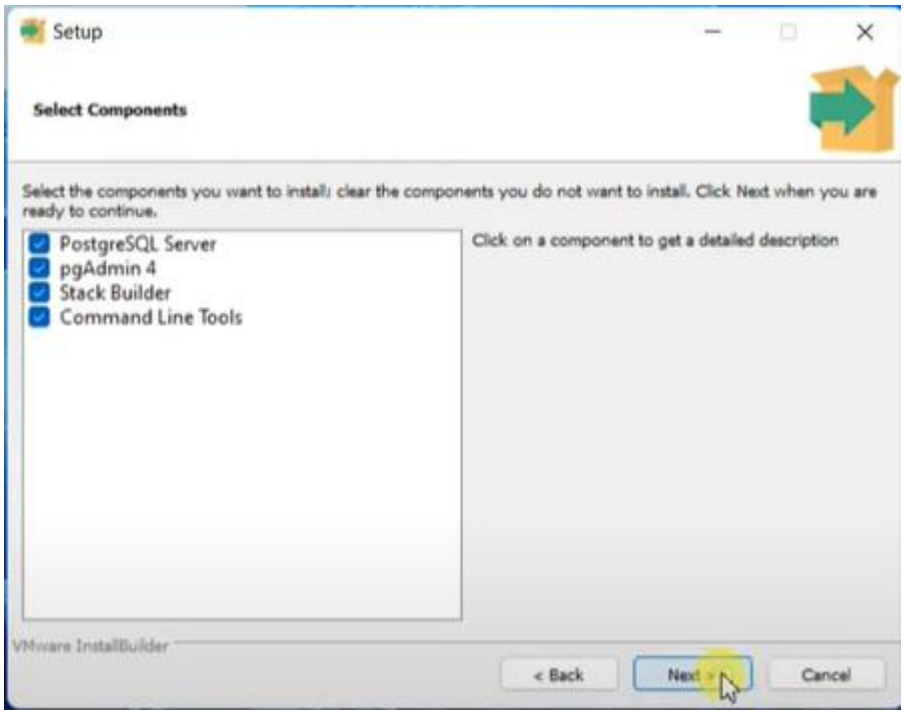
*Rysunek 6 - instalacja postgres*



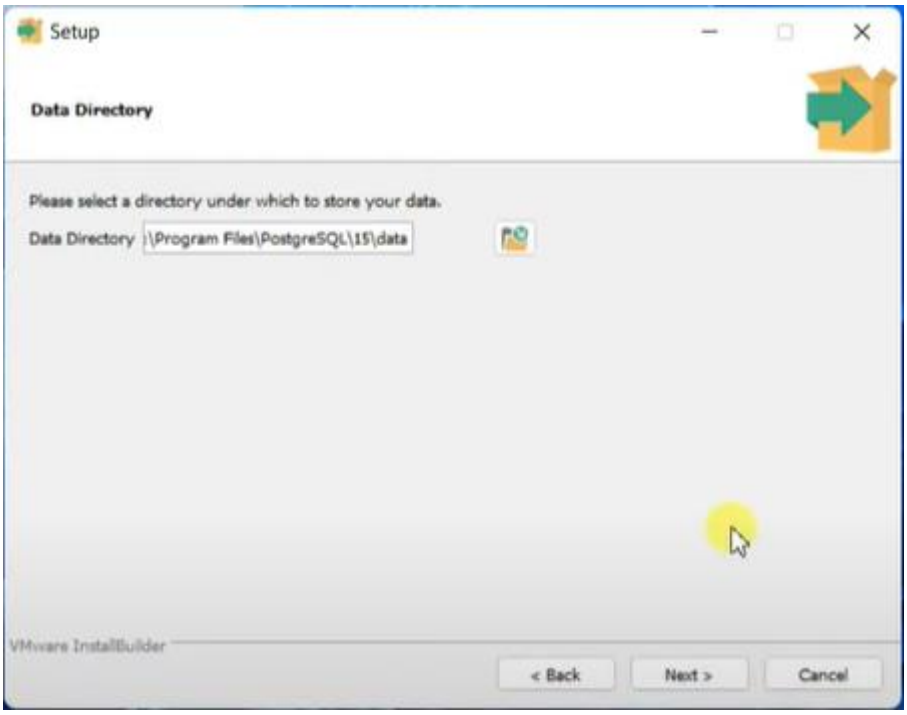

*Rysunek 7 – instalacja postgres*




*Rysunek 8 - instalacja postgres*

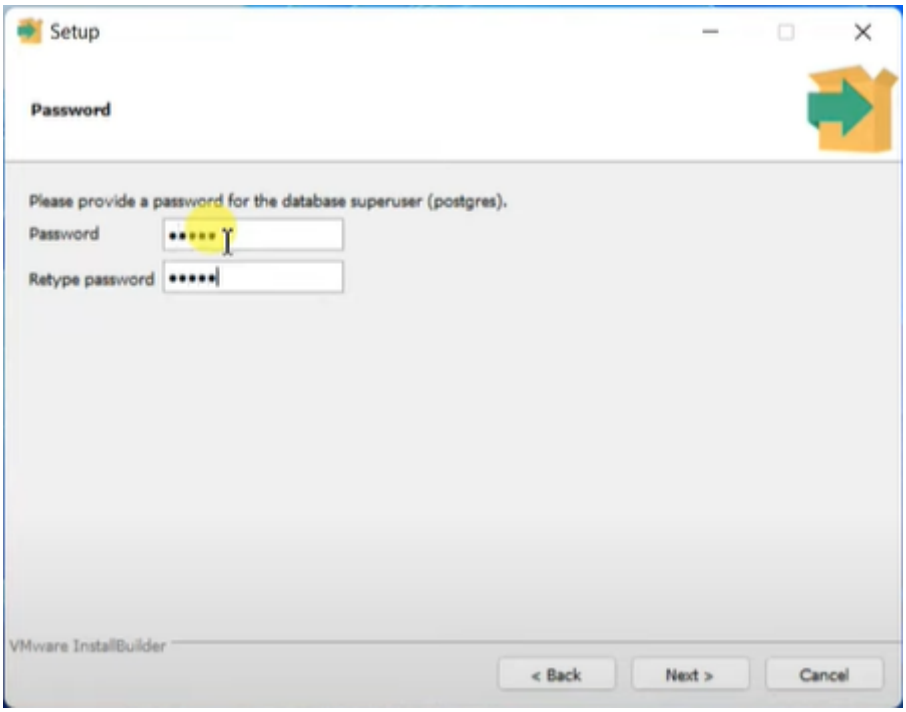



*Rysunek 9 - instalacja postgres*

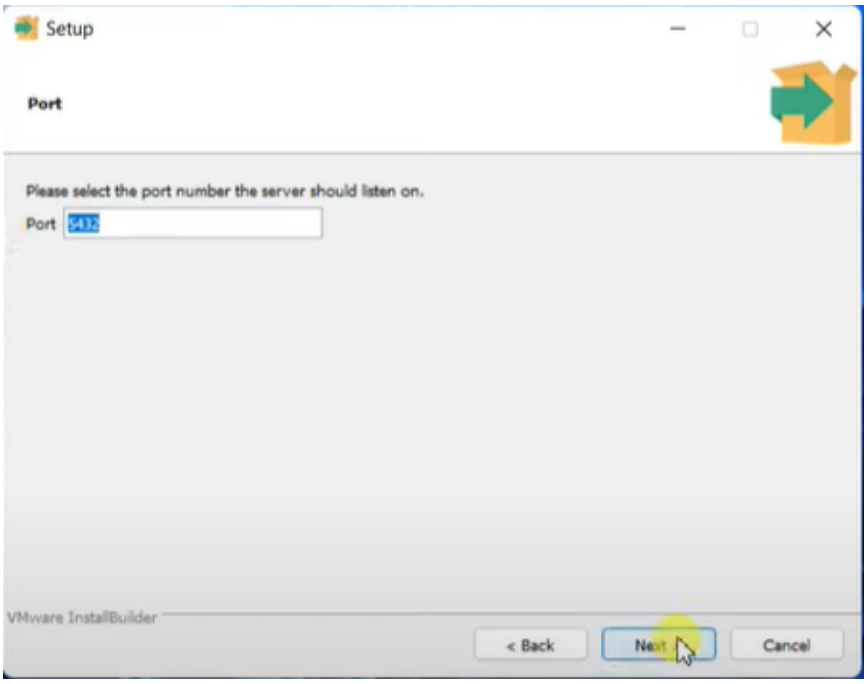



*Rysunek 10 - instalacja postgres*

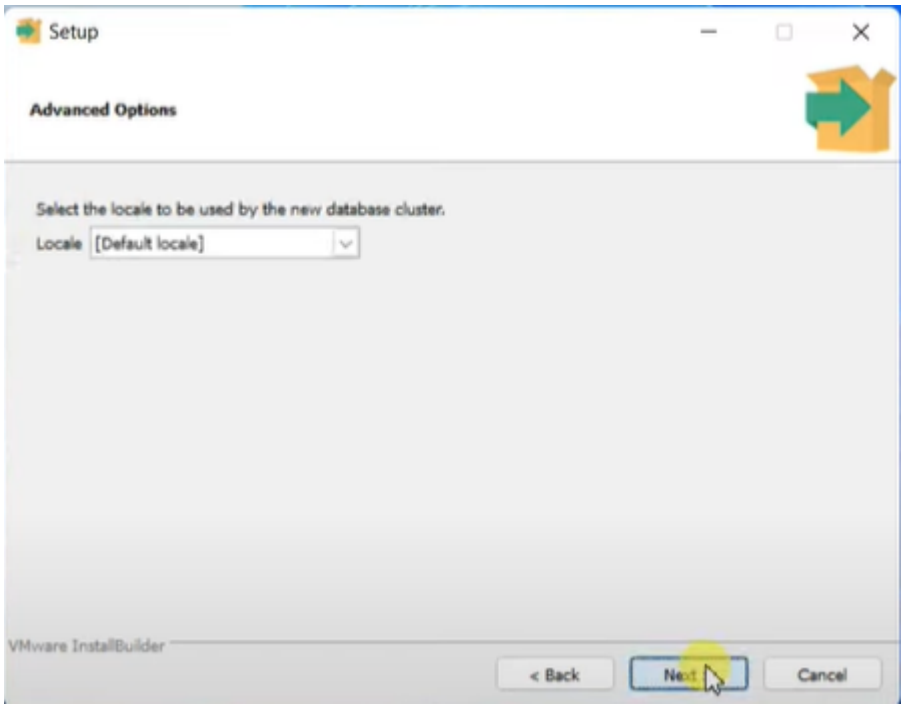
Tutaj wpisujemy hasło w tym przypadku aby aplikacja działała poprawnie należy wpisać hasło ‘admin’ ponieważ jesto ono potrzebane do połączenia z bazą



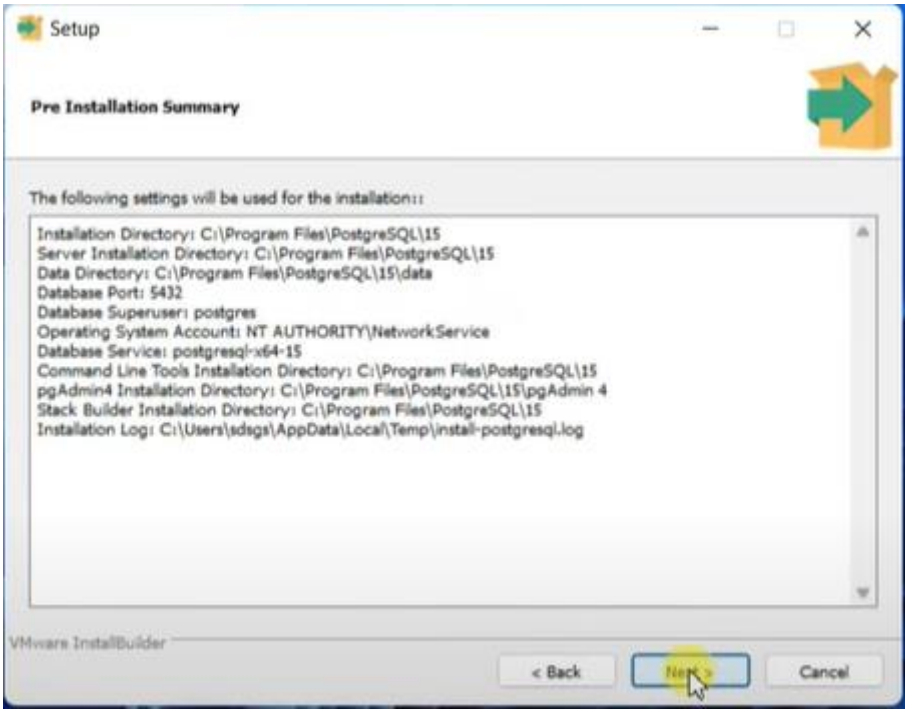

*Rysunek 11 - instalacja postgres*



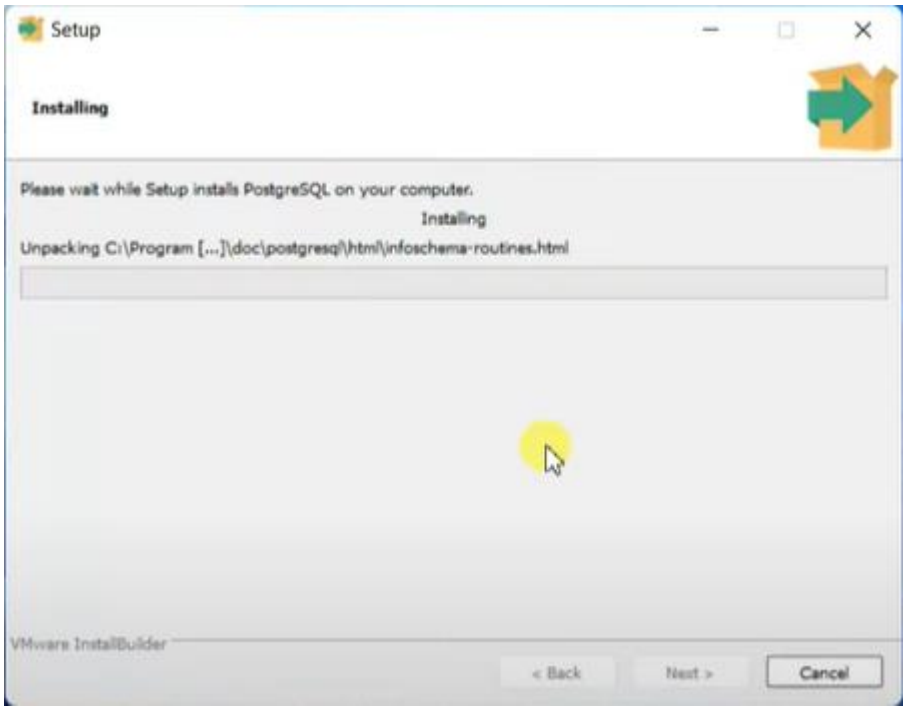

*Rysunek 12 - instalacja postgres*




*Rysunek 13 - instalacja postgres*



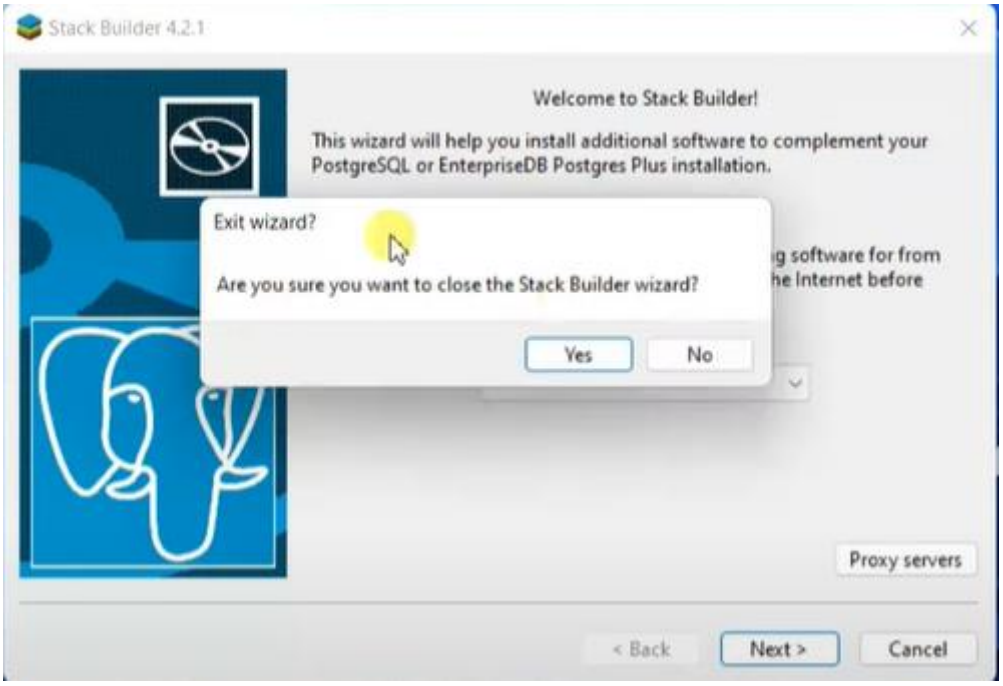

*Rysunek 14 - instalacja postgres*




*Rysunek 15 - instalacja postgres*


*Rysunek 16 - instalacja postgres*


*Rysunek 17 - instalacja postgres*

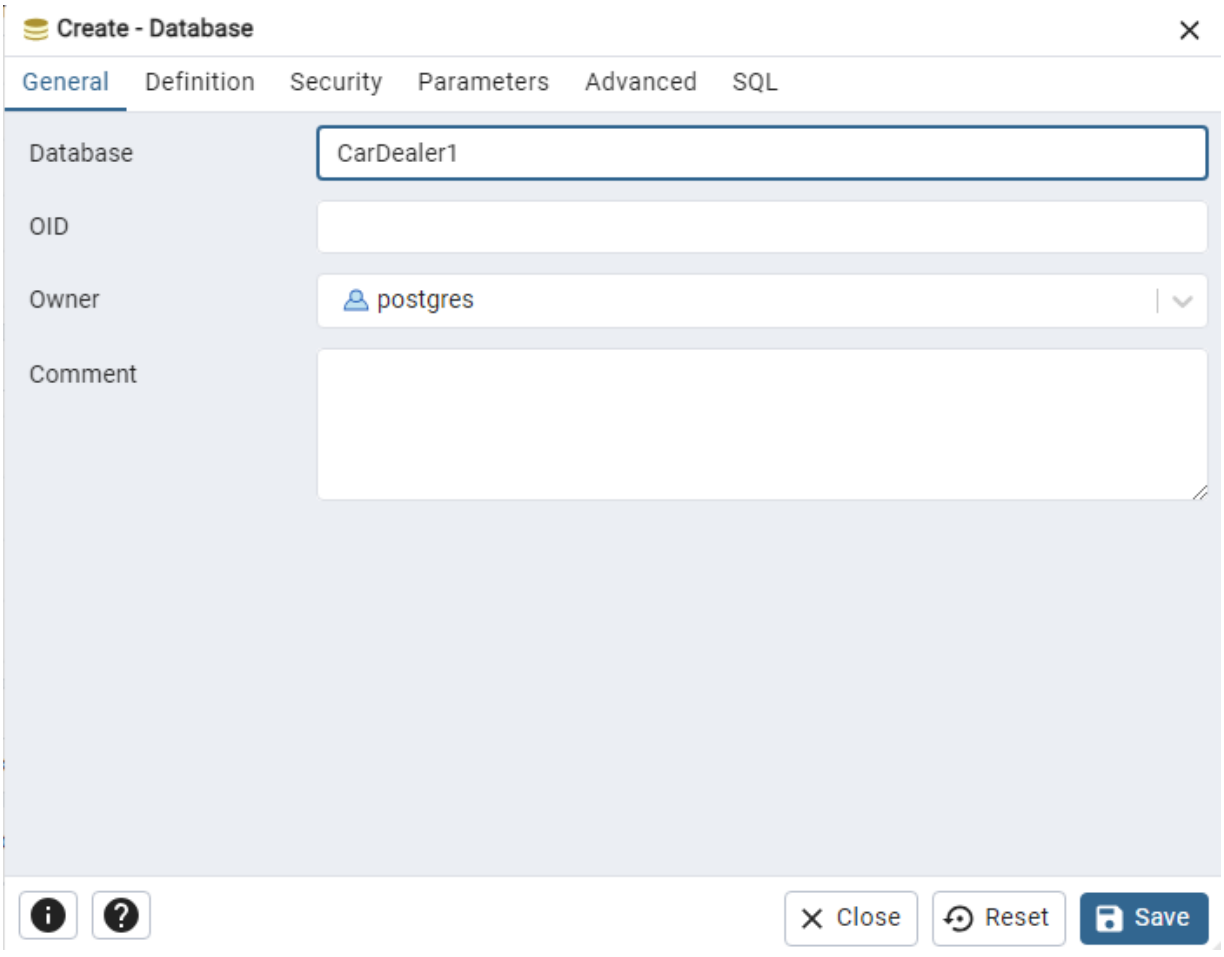



*Rysunek 18 - instalacja postgres*

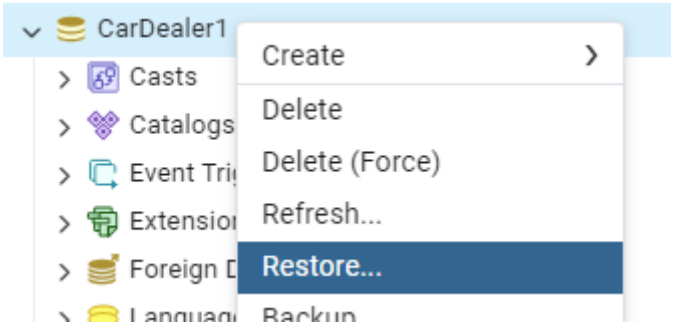

*Rysunek 19 - instalacja postgres*


*Rysunek 20 - tworzenie bazy*

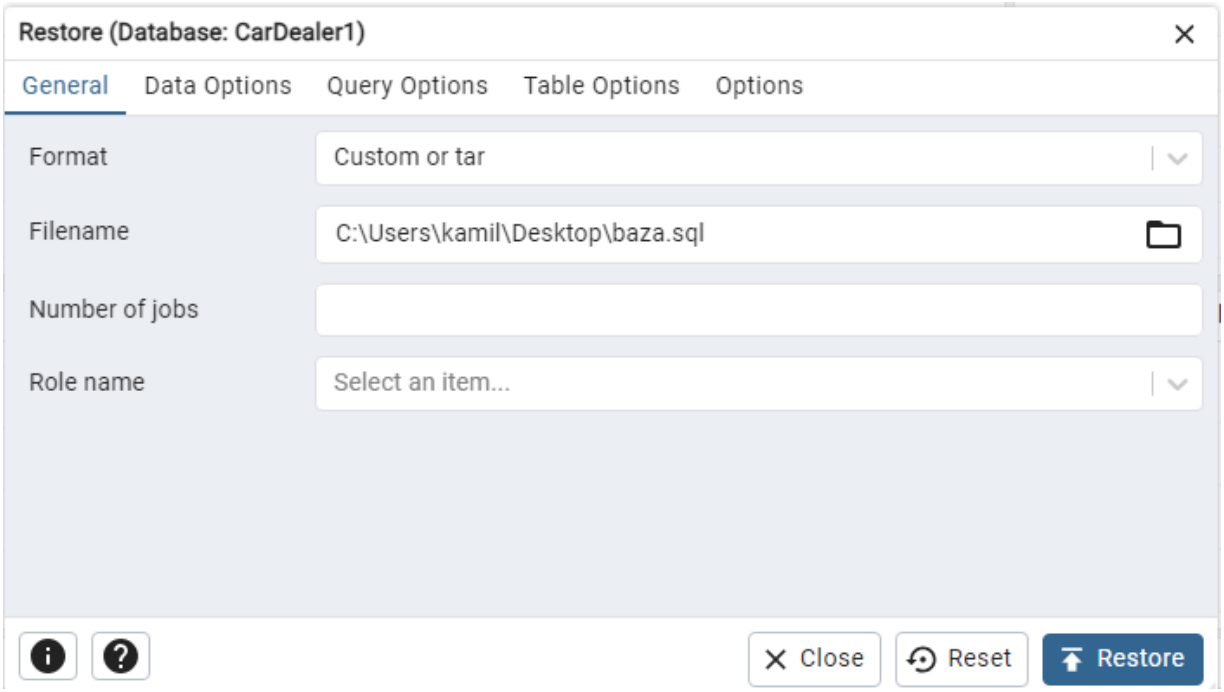
Tworzymy tabele z nazwą ‘Car Dealer’




*Rysunek 21- tworzenie bazy*




*Rysunek 22 - tworzenie tabel*




*Rysunek 23 - tworzenie tabel*

wybieramy plik ‘baza.sql’ z tabelami bazy danych aby wszysko działało poprawnie

## Konfiguracja systemu/aplikacji/serwerów:

**Java:** Upewnij się że na systemie zainstalowana jest Java zgodnie z wymaganiami
**PostgreSQL Database:** Aplikacja wykorzystuje bazę danych postgreSQl.

```
private static final String URL = "jdbc:postgresql://localhost:5432/CarDealer";  1 usage
private static final String USER = "postgres";  1 usage
private static final String PASSWORD = "admin";  1 usage
```

*Rysunek 24 - połączenie z baza w kodzie*

**Uruchamianie aplikacji:** Po skonfigurowaniu środowiska, uruchom aplikację z klasy zawierającej metodę `main`. Upewnij się, że aplikacja ma dostęp do bazy danych.

### Zarządzanie Zależnościami:

Jeśli projekt korzysta z zależności (np. bibliotek JavaFX), upewnij się, że są one prawidłowo skonfigurowane i dostępne dla projektu. Powyżej instrukcja instalacji bibliotek do projektu.

## Opis działań które osoba dana osoba wykonała

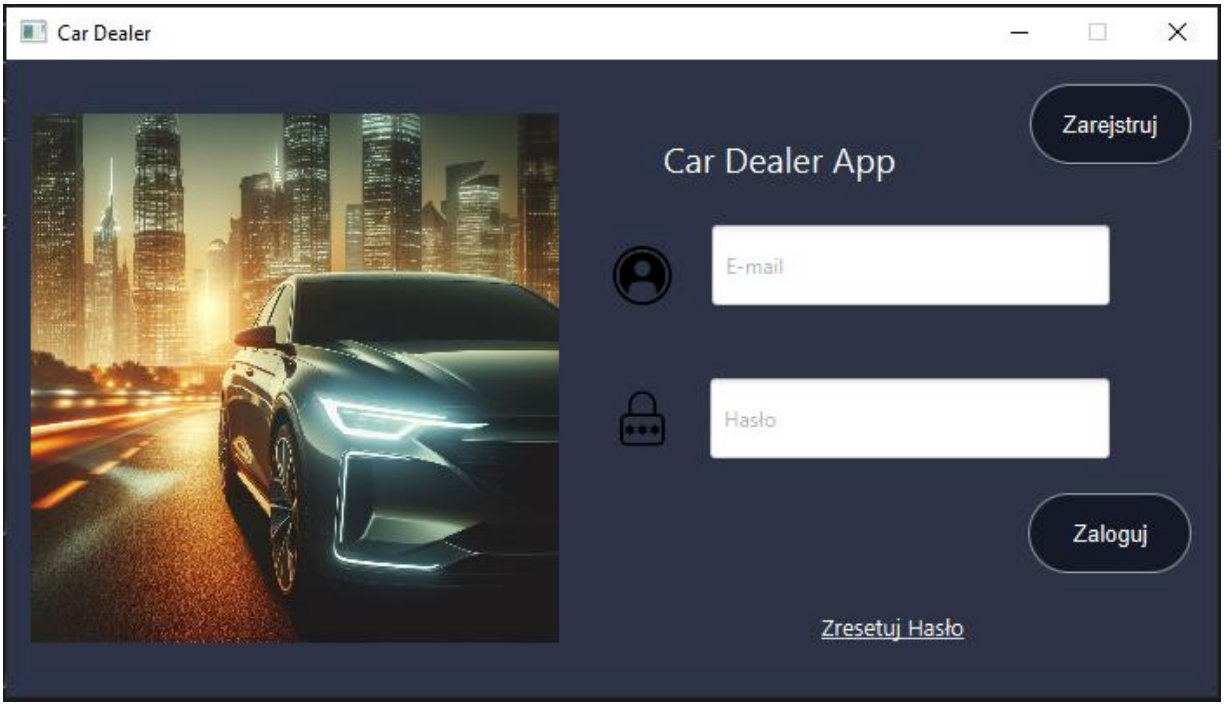
Miłosz Gierus – zrobił panel logowania, rejestracji, zmiany hasła oraz implementacja w kodzie oraz połowę dokumentacji

Mateusz – zrobił główny panel aplikacji procedury CRUD oraz implementacja w kodzie oraz połowę dokumentacji

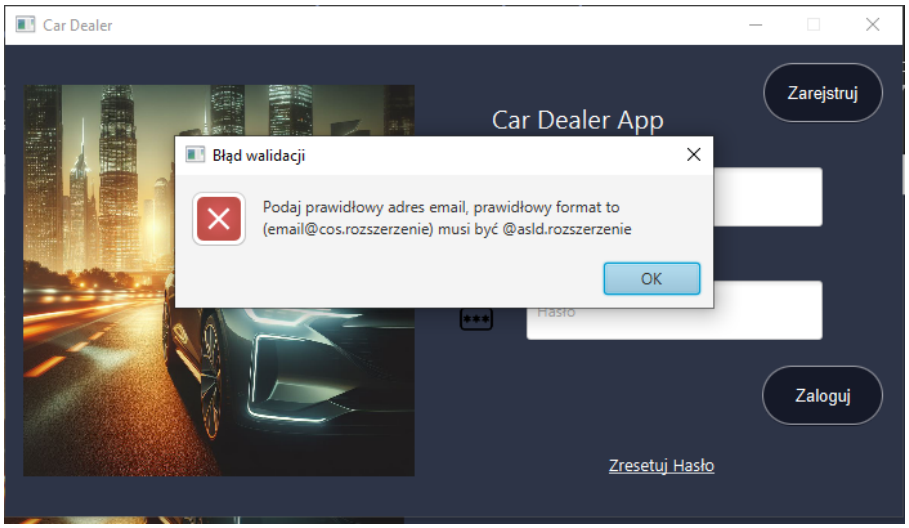
## Opis działania aplikacji

### Opis interfejsu:

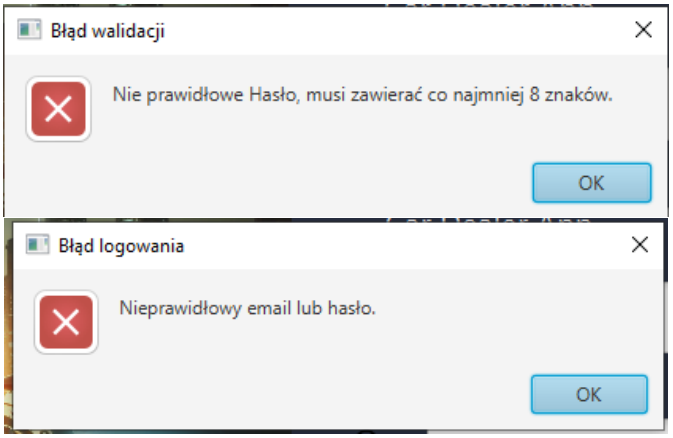
Aplikacja CarDealer posiada intuicyjny interfejs użytkownika, który umożliwia łatwe logowanie do aplikacji możliwe jest logowanie na administratora aplikacji który ma dodatkowe funkcje korzystając z aplikacji email: ‘jd@example.pl’ hasło: ‘12345678’. Posiada również walidacje jeżeli wpisze się nie poprawne dane.




*Rysunek 25- widok loginu*

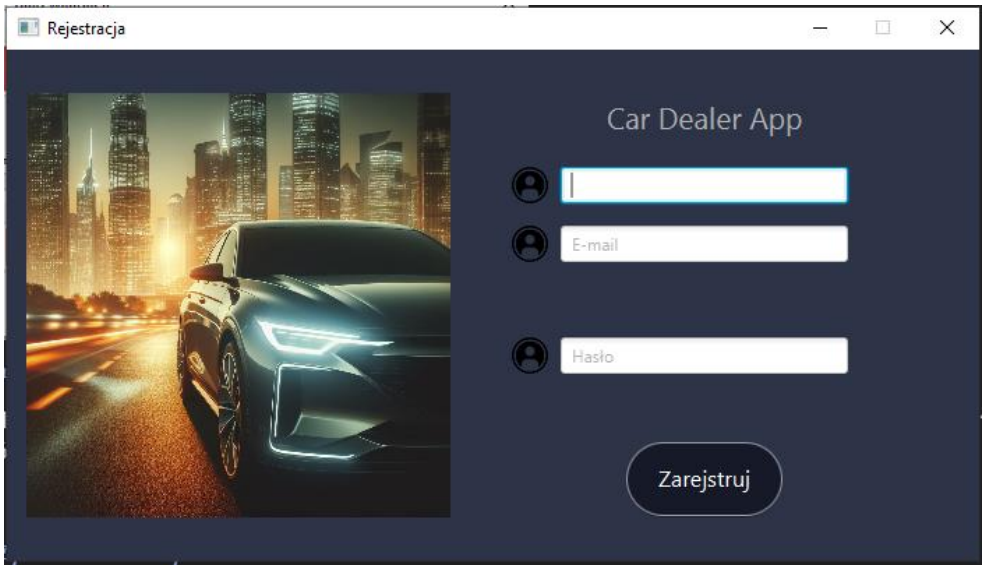



*Rysunek 26- walidacja login*



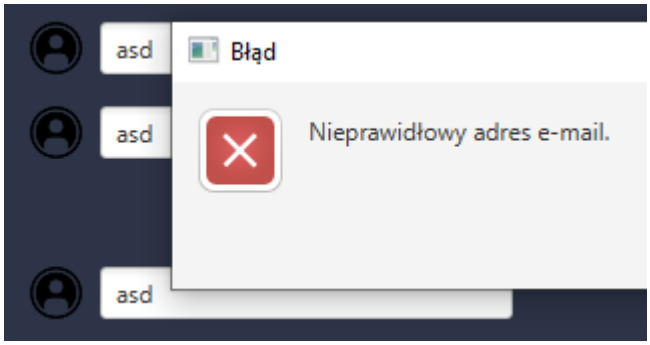

*Rysunek 27 - walidacja hasło*

Aplikacja pozwala na rejestracje użytkownika po kliknięciu w panelu logowania przycisku ‘Zarejestruj’ która otwiera nowy widok rejestracji. Po poprawnym zarejestrowaniu dodany jest nowy użytkownik aplikacji (który nie jest administratorem). Rejestracja również posiada walidacje:

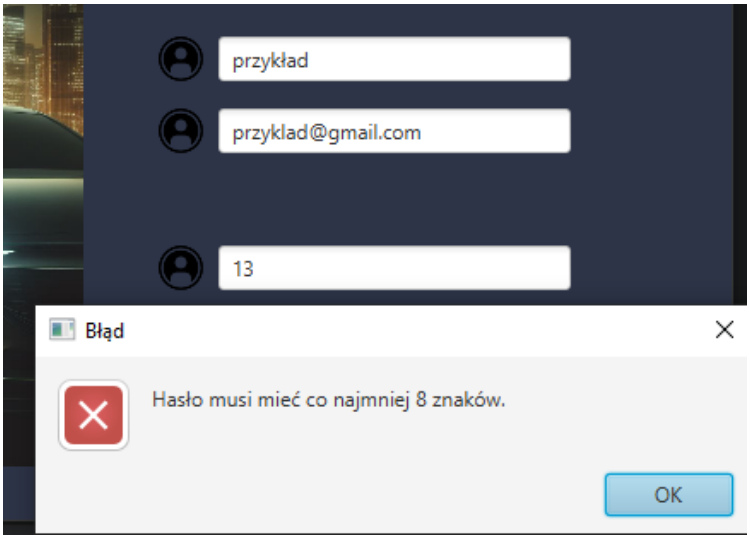



*Rysunek 28 – rejestracja*

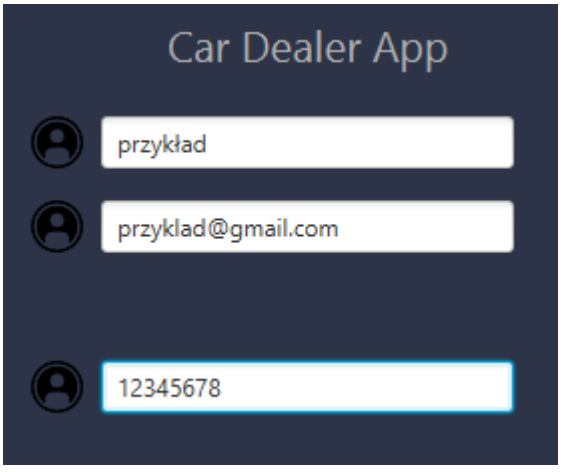

*Rysunek 29 - nie wypełnione pola*



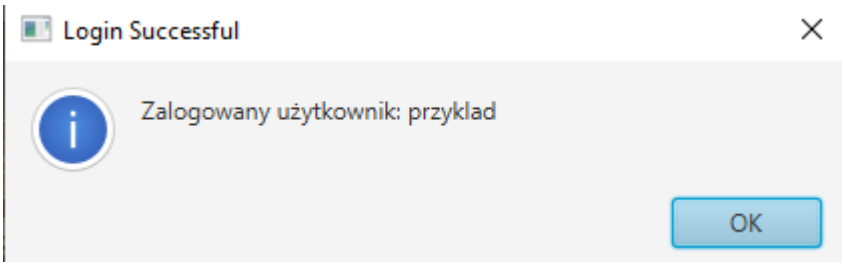

*Rysunek 30 - nieprawidłowy e-mail*




*Rysunek 31 - za mało znaków hasła*



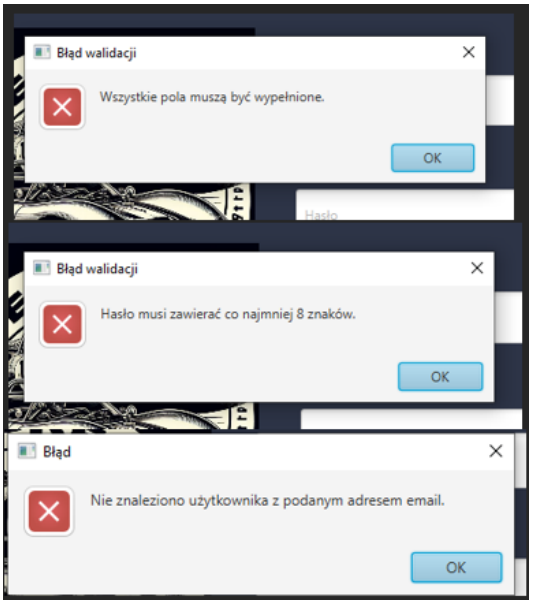

*Rysunek 32 - przykładowa rejestracja*



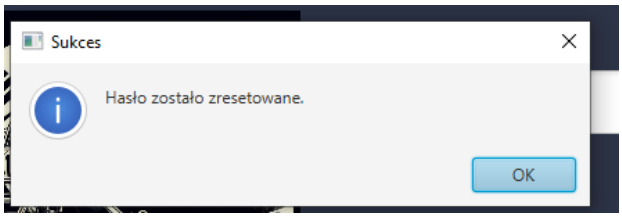

*Rysunek 33 - udany login*

W przypadku gdy użytkownik zapomni hasło ma możliwość jego zmienienia po kliknięciu w napis ‘Zresetuj hasło’, który otwiera nowy widok umożliwiający zmianę hasła. To okno również posiada walidacje


*Rysunek 34- zmiana hasła*

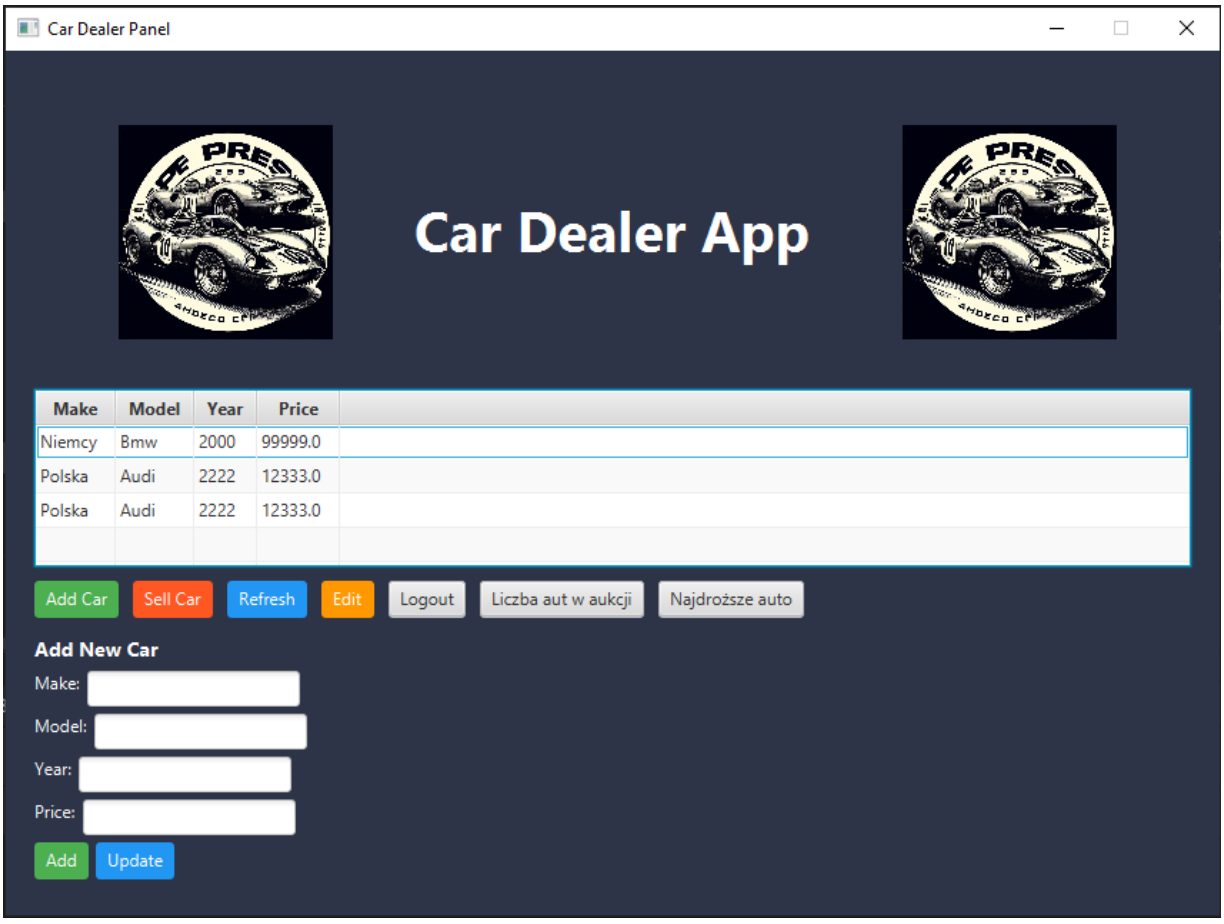



*Rysunek 35 - walidacja zmiana hasła*

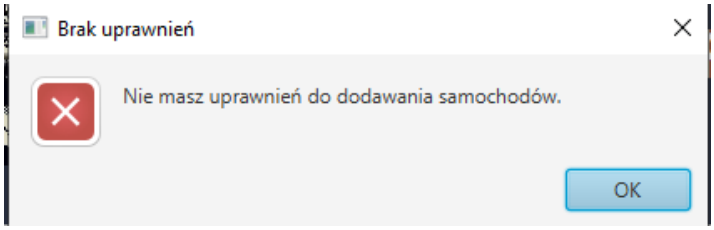



*Rysunek 36 - pomyślnie zmienione hasło*

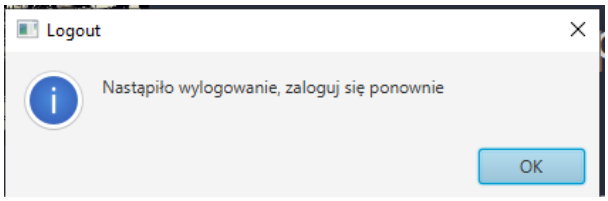
Możliwości zalogowanego użytkownika który nie jest administratorem. Zalogowany użytkownik może przeglądać auta dostępne do sprzedaży może się wylogować, zobaczyć ile aut jest w aukcji po kliknięciu przycisku ‘Liczba aut w aukcji’, zobaczyć jakie jest najdroższe auto w aukcji klikając przycisk ‘Najdoższe auto’. Ograniczenia są sygnalizowane odpowiednimi alertami



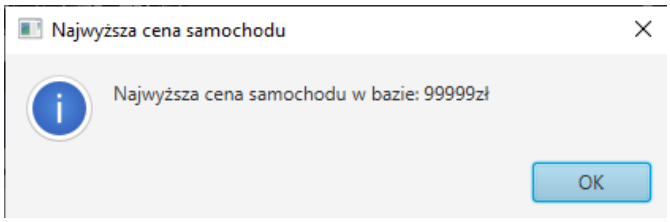

*Rysunek 37 - widok główny po zalogowaniu*



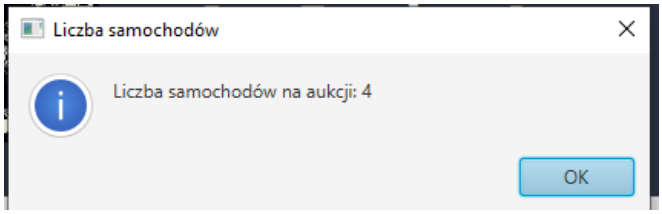

*Rysunek 38 - brak uprawnień do dodawania, usuwania i edycji*




*Rysunek 39 - po kliknięciu przycisku Logout*

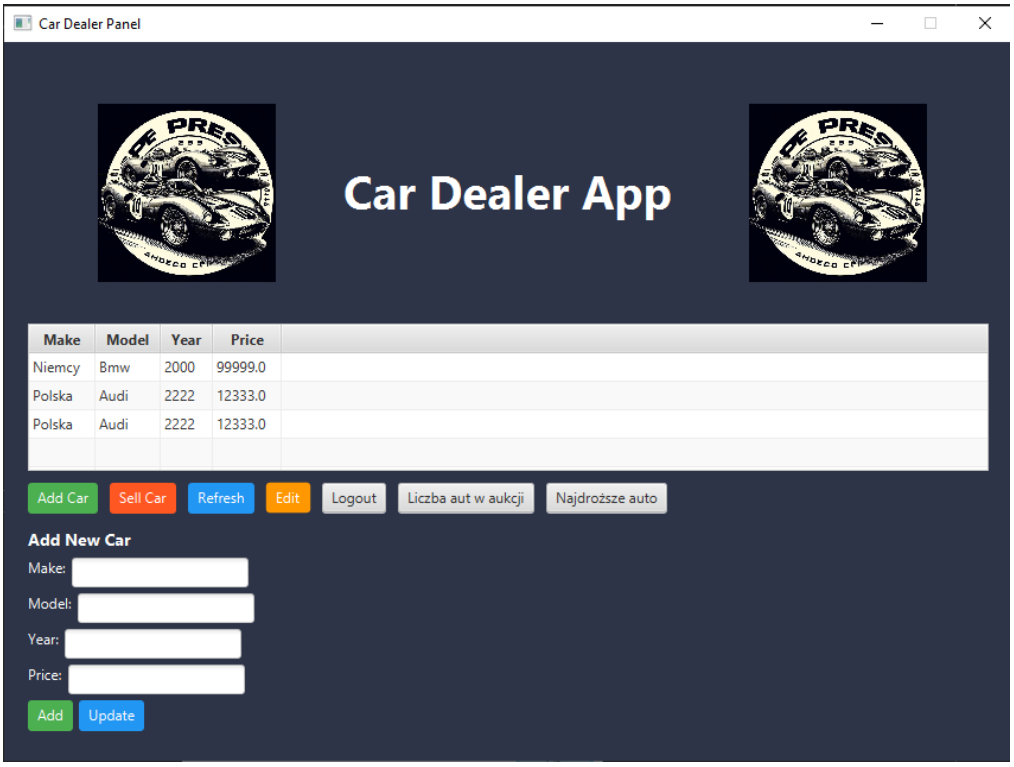



*Rysunek 40 - przycisk najwyższa cena*

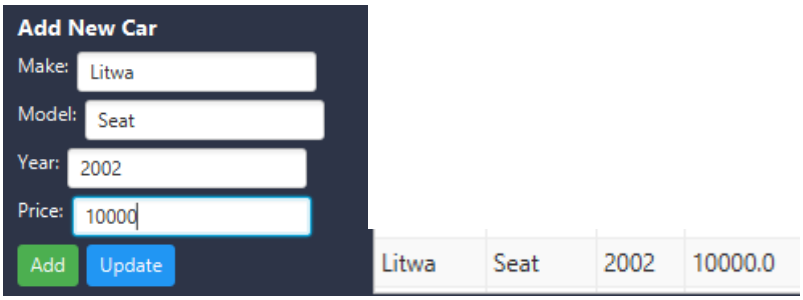



*Rysunek 41 - przycisk liczba samochodów*

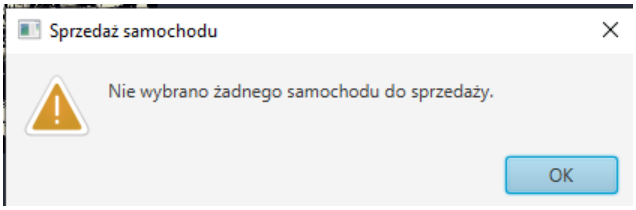
Możliwości zalogowanego administratora aplikacji. Administrator może dodawać nowe auta wypełniając pola zgodnie z walidacją, może usuwać auta oraz je edytować.



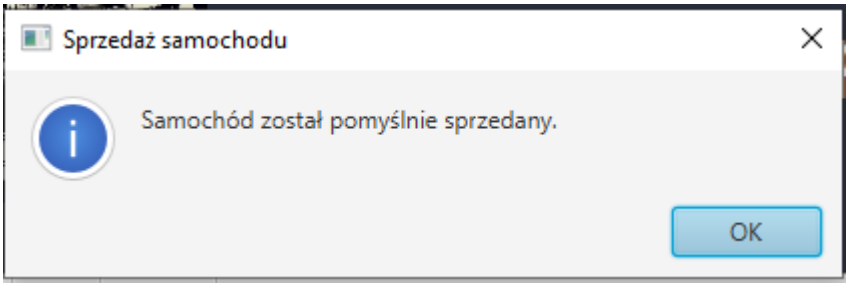

*Rysunek 42 - widok główny*



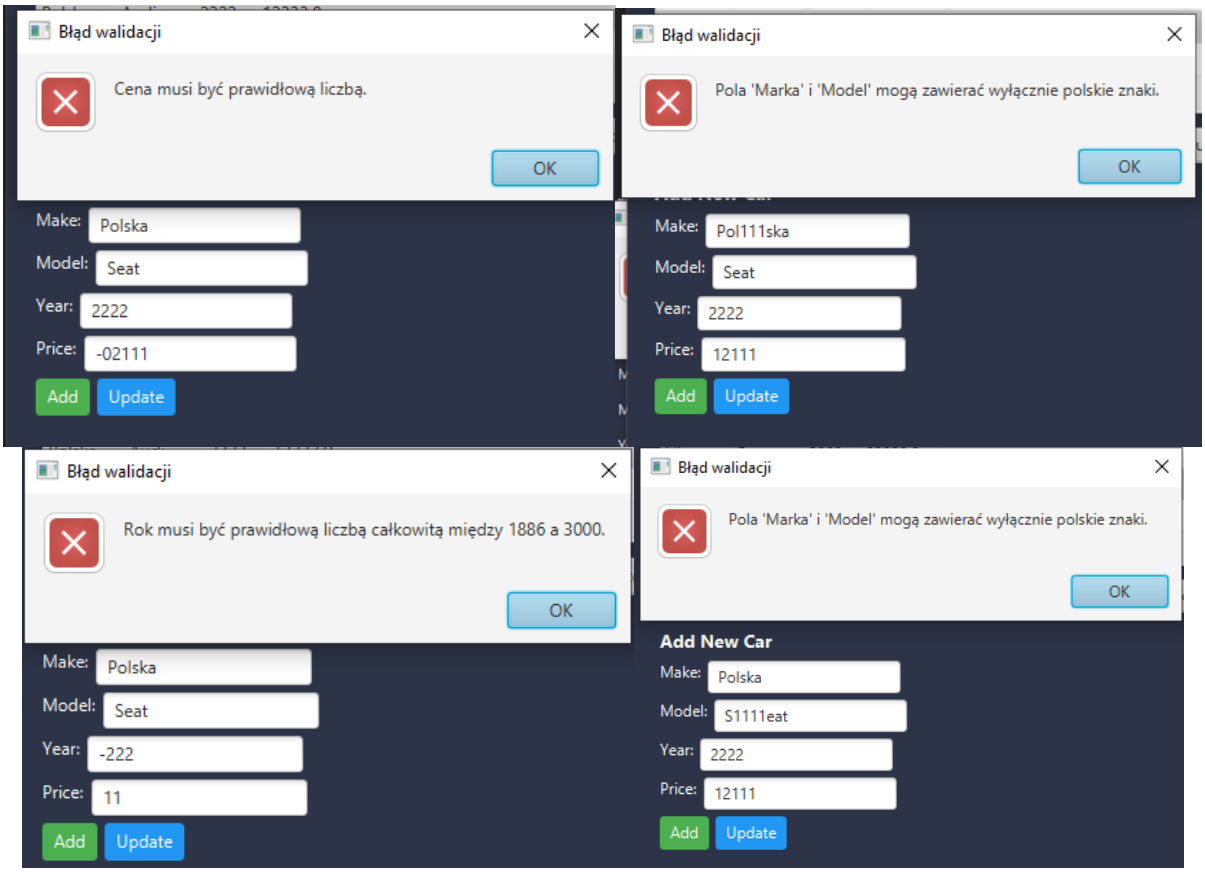

*Rysunek 43 - dodawanie auta*



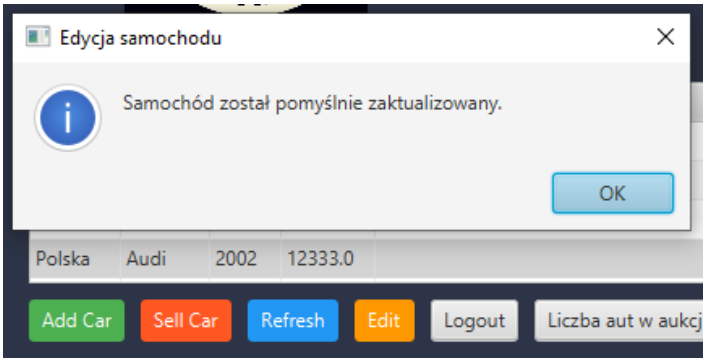

*Rysunek 44 - usuwanie*




*Rysunek 45 - po pomyślnym usunięciu*




*Rysunek 46 - walidacja dodawania*




*Rysunek 47 - edycja*

## Wykorzystane funkcje jeżyka proceduralnego

Query | Query History

```
1  CREATE OR REPLACE FUNCTION get_most_expensive_car_price()
2  RETURNS INTEGER AS
3  $$
4  DECLARE
5      max_price INTEGER;
6  BEGIN
7      SELECT MAX(price) INTO max_price FROM cars;
8      RETURN max_price;
9  END;
10 $$ LANGUAGE plpgsql;
```

*Rysunek 48 - funkcja najdoższe auto*

**Funkcja `get_most_expensive_car_price()`**

Ta funkcja `get_most_expensive_car_price()` jest napisana w języku PL/pgSQL, który jest proceduralnym rozszerzeniem języka SQL dla bazy danych PostgreSQL. Funkcja ta ma na celu zwrócenie najwyższej ceny samochodu zapisanej w tabeli `cars`. Szczegółowy opis:

- Funkcja get_most_expensive_car_price nie przyjmuje żadnych argumentów.
- Deklaruje zmienną max_price typu INTEGER.
- Wykonuje zapytanie SELECT MAX(price) FROM cars, aby znaleźć maksymalną wartość w kolumnie price w tabeli cars.
- Przypisuje tę wartość do zmiennej max_price.
- Zwraca wartość max_price jako wynik działania funkcji.

**Funkcja get_car_count()**

Ta funkcja `get_car_count()` jest napisana w języku PL/pgSQL dla bazy danych PostgreSQL. Funkcja ta ma na celu zwrócenie liczby samochodów zapisanych w tabeli `cars`. Szczegółowy opis:

- Funkcja get_car_count służy do zliczania wszystkich rekordów w tabeli cars.
- Używa zapytania SELECT COUNT(*) do obliczenia liczby wierszy.
- Wynik jest zwracany jako wartość typu INTEGER.

## Podsumowanie

Aplikacja CarDealer została stworzona przez Miłosza Gierusa i Mateusza Floriana jako projekt na przedmiot "Bazy Danych" w Grupie 1. Projekt, oddany 11 czerwca 2024 roku, ma na celu zarządzanie informacjami o samochodach.

Aplikacja pozwala użytkownikom na dodawanie, edytowanie, usuwanie oraz przeglądanie samochodów, a także wyświetlanie szczegółowych informacji o każdym pojeździe. Dostęp do aplikacji wymaga rejestracji i logowania.