Amadeusz Nowak

Sebastian Mikoś

wypożyczalnia Samochodów

Wypożyczalnia

Spis treści

[Rozpatrywany problem 2](#_Toc187771836)

[Schemat ERD 4](#_Toc187771837)

[Funkcjonalności bazy danych: 4](#_Toc187771838)

[Struktura bazy danych 5](#_Toc187771839)

[Tabela egzemplarze: 5](#_Toc187771840)

[Tabela klienci 5](#_Toc187771841)

[Tabela modele 6](#_Toc187771842)

[Tabela pracownicy: 6](#_Toc187771843)

[Tabela rezerwacje: 6](#_Toc187771844)

[Tabela wypożyczone: 6](#_Toc187771845)

[Tabele dodatkowe 6](#_Toc187771846)

[Relacje i węzły integralności 7](#_Toc187771847)

[Funkcje i procedury w bazie danych 8](#_Toc187771848)

[Dodawanie rekordów (CREATE). Pakiet „dodawanie” 8](#_Toc187771849)

[Przykład dla konkretnej tabeli 8](#_Toc187771850)

[Modyfikacja rekordów (UPDATE). Pakiet „modyfikowanie” 8](#_Toc187771851)

[Przykład dla konkretnej tabeli: 9](#_Toc187771852)

[Usuwanie rekordów(DELETE). Pakiet „usuwanie” 10](#_Toc187771853)

[Przykład dla konkretnej tabeli: 10](#_Toc187771854)

[Wyświetlanie tabel(READ). Pakiet „wyświetlanie” 11](#_Toc187771855)

[Przykład dla konkretnej tabeli: 11](#_Toc187771856)

[Funkcje i procedury klienta. Pakiet „public” 11](#_Toc187771857)

[Opis procedury: wypozycz() 11](#_Toc187771858)

[Opis procedury: zarezerwuj() 13](#_Toc187771859)

[Opis funkcji: do\_kiedy\_wolne() 15](#_Toc187771860)

[Opis funkcji: czy\_klient\_istnieje() 16](#_Toc187771861)

[Opis funkcji: sprawdz\_telefon() 17](#_Toc187771862)

[Opis funkcji: historia\_uzytkownika() 17](#_Toc187771863)

[Opis funkcji: info\_auto() 19](#_Toc187771864)

[Opis funkcji: koszt() 20](#_Toc187771865)

[Opis funkcji: wszystkierejestracje() 21](#_Toc187771866)

[Funkcje i procedury administratora 21](#_Toc187771867)

[Opis funkcji: liczenie\_pensji() 21](#_Toc187771868)

[Opis funkcji: sprawdz\_rejestracje() 22](#_Toc187771869)

[Opis funkcji wyliczających przychody z egzemplarzy 22](#_Toc187771870)

[Opis funkcji wyliczających przychody od klientów 23](#_Toc187771871)

[Inne procedury. Pakiet „public” 24](#_Toc187771872)

[Koncepcja dostępu zdalnego: 26](#_Toc187771873)

[Zrealizowany dostęp zdalny: 27](#_Toc187771874)

[Wykorzystywane technologie: 27](#_Toc187771875)

[Pliki połączenia z bazą danych 27](#_Toc187771876)

[Rejestracja klienta 28](#_Toc187771877)

[Logowanie klienta 29](#_Toc187771878)

[Okno opcji klienta 30](#_Toc187771879)

[Okno historii wypożyczeń użytkownika 31](#_Toc187771880)

[Wypożyczanie samochodów 32](#_Toc187771881)

[Rezerwowanie samochodów 33](#_Toc187771882)

[Logowanie administratora 35](#_Toc187771883)

[Okno administratora 36](#_Toc187771884)

[CRUD – operacje na bazie 37](#_Toc187771885)

[Dodawanie danych(CREATE) 37](#_Toc187771886)

[Wyświetlanie tabel (READ): 38](#_Toc187771887)

[Modyfikowanie danych(UPDATE) 38](#_Toc187771888)

[Usuwanie danych(DELETE) 38](#_Toc187771889)

[Statystyka 39](#_Toc187771890)

[Tabela klienci 39](#_Toc187771891)

[Tabela „egzemplarze” 42](#_Toc187771892)

[Tabela „pracownicy” 44](#_Toc187771893)

[Harmonogram zadań w pgAgent 45](#_Toc187771894)

# 

# Rozpatrywany problem

Rozpatrzmy bazę danych stworzoną dla wypożyczalni samochodów, która przechowuje: markę, model, pojemność silnika, moc silnika, rejestracja, kolor, rok produkcji, cenę za dzień wypożyczenia. O klientach: imię, nazwisko, numer PESEL, telefon, email. O pracownikach: imię, nazwisko, wynagrodzenie miesięczne, data zatrudnienia. Każdy pracownik ma przydzielony pojazd, którym się opiekuje, pilnując daty zwrotu. Baza powinna przechowywać aktualnie wypożyczone pojazdy i całą historię wypożyczeń. W momencie wypożyczenia znana jest data zwrotu. Baza powinna uwzględniać możliwość wypożyczenia wielu samochodów przez jednego klienta. Firma również może rezerwować wybrane pojazdy przez klientów.

1.Modele – spis wszystkich modeli posiadanych przez wypożyczalnię.

Atrybuty: ID\_modelu, marka, model, pojemność silnika, moc\_silnika, opiekun (relacja do pracownika)

2.Egzemplarze – spis wszystkich aut.

Atrybuty: Rejestracja, kolor, ID\_modelu, rok\_produkcji, cena, przebieg

3. Klienci – informacje o klientach banku.

Atrybuty: imię, nazwisko, numer\_PESEL, telefon, email

4. Pracownicy – informacje o pracownikach.

Atrybuty: ID\_pracownika, imię, nazwisko, wynagrodzenie\_bazowe, data\_zatrudnienia

5. Wypożyczone (auta) – stan magazynu.

Atrybuty: ID\_wypożyczenia, data wypożyczenia, ID\_klienta, data zwrotu

6. Rezerwacje – Przyszłościowe zamówienia.

Atrybuty: ID\_rezerwacji, ID\_klienta, Data\_rozpoczęcia, Data\_zakończenia

# Schemat ERD

Obraz zawierający tekst, diagram, zrzut ekranu, linia

Opis wygenerowany automatycznie

# Funkcjonalności bazy danych:

1. **Operacje CRUD na wszystkich tabelach:**
   1. Dodawanie rekordów
   2. Odczytywanie:
      1. Wszystkich rekordów z tabeli
      2. Pojedynczego rekordu z tabeli
   3. Modyfikowanie rekordów
   4. Usuwanie rekordów.
2. **Sprawdzanie istnienia rekordów w tabeli:**
   1. Numeru PESEL użytkownika
   2. Numeru rejestracyjnego samochodu
   3. Numeru telefonu klienta
3. **Liczenie pensji końcowej pracowników**
   1. Obliczanie miesięcznego wynagrodzenia pracowników na podstawie wynagrodzenia bazowego oraz ilości egzemplarzy którymi się opiekuje
4. **Sprawdzanie czy samochód jest dostępny do rezerwacji i wypożyczenia**
   1. Określanie, czy dany pojazd jest obecnie dostępny poprzez sprawdzenie daty wypożyczenia i daty zwrotu
5. **Obliczanie maksymalnej ilości dni wypożyczenia auta**
   1. Ustalanie maksymalnego okresu wynajmu pojazdu
6. **Wyświetlanie historii wypożyczeń użytkownika**
   1. Przedstawienie listy wszystkich wypożyczeń danego klienta
7. **Obliczanie kwoty do zapłaty przez klienta za wypożyczone samochody**
   1. Obliczenie całkowitego kosztu wypożyczenia, uwzględniając liczbę dni wynajmu i cenę za dzień.
8. **Obliczanie kwoty wygenerowanej przez każdy samochód**
   1. Obliczenie sumy, którą dany pojazd wygenerował na podstawie wypożyczeń.
9. **Obliczanie całkowitej kwoty, którą każdy klient wydał na wypożyczone samochody**
   1. Sumowanie wydatków danego klienta na wszystkie wypożyczenia
10. **Wypożyczenie danego egzemplarza**
    1. Zarejestrowanie wypożyczenia samochodu przez klienta, uwzględniając daty wypożyczenia i planowany termin zwrotu.
11. **Zarezerwowanie danego egzemplarza**
    1. Rejestracja rezerwacji pojazdu przez klienta, z określeniem daty rozpoczęcia i zakończenia rezerwacji.
12. **Realizacje rezerwacji**
    1. Potwierdzenie rezerwacji i rozpoczęcie procesu wypożyczenia pojazdu na podstawie wcześniejszej rezerwacji.

# Struktura bazy danych

## Tabela egzemplarze:

1. Przechowuje informacje o dostępnych egzemplarzach pojazdów.
2. Kolumny:
   1. (PK) Rejestracja (text, NOT NULL): numer rejestracyjny pojazdu.
   2. kolor (text, NOT NULL): kolor pojazdu.
   3. (FK) ID\_modelu (integer, NOT NULL): identyfikator modelu pojazdu.
   4. rok\_produkcji (date, NOT NULL): rok produkcji pojazdu.
   5. cena (integer, NOT NULL): cena pojazdu (ograniczenie CHECK wymaga, aby wartość była większa niż 0).
   6. przebieg (integer, NOT NULL): przebieg pojazdu w kilometrach.

## Tabela klienci

1. Przechowuje informacje o klientach.
2. Kolumny:
   1. (PK) pesel (bigint, NOT NULL): numer PESEL klienta.
   2. imie (text, NOT NULL): imię klienta.
   3. nazwisko (text, NOT NULL): nazwisko klienta.
   4. email (text): adres e-mail klienta.
   5. telefon (bigint, NOT NULL): numer telefonu klienta (ograniczenie CHECK wymaga, aby numer znajdował się w zakresie <0;999999999>).

## Tabela modele

1. Przechowuje informacje o modelach pojazdów.
2. Kolumny:
   1. (PK) ID\_modelu (integer): identyfikator modelu. (auto increment)
   2. marka (text, NOT NULL): marka pojazdu.
   3. model (text, NOT NULL): model pojazdu.
   4. pojemność\_silinika (integer, NOT NULL): pojemność silnika pojazdu (ograniczenie CHECK wymaga, aby wartość była większa niż 0)
   5. moc\_silnika (integer, NOT NULL): moc silnika pojazdu (ograniczenie CHECK wymaga, aby wartość była większa niż 0)
   6. (FK) opiekun (integer)

## Tabela pracownicy:

1. Przechowuje informację o pracownikach
2. Kolumny:
   1. (PK) ID\_pracownika (integer): identyfikator pracownika (auto increment)
   2. Imie (text, NOT NULL): imię pracownika
   3. Nazwisko (text, NOT NULL): nazwisko pracownika
   4. Wynagrodzenie\_bazowe (integer, NOT NULL): wynagrodzenie pracownika, jeśli nie ma pod swoją opieką żadnych modeli (ograniczenie CHECK wymaga, aby wartość była większa niż 0)
   5. Data\_zatrudnienia (date, NOT NULL): data zatrudnienia pracownika

## Tabela rezerwacje:

1. Przechowuje informacje o rezerwacjach egzemplarzy
2. Kolumny:
   1. (PK/FK) ID\_rezerwacji (integer): identyfikator rezerwacji (auto increment)
   2. Pesel (bigint, NOT NULL): pesel klienta, który rezerwuje
   3. Data\_rozpoczecia (date, NOT NULL): początek rezerwacji
   4. Data\_zakończenia (date, NOT NULL): koniec rezerwacji

## Tabela wypożyczone:

1. Przechowuje informacje o wypożyczeniach egzemplarzy
2. Kolumny:
   1. (PK/FK)ID\_wypozyczenia (integer): identywikator wypożyczenia (auto increment)
   2. Data\_wypozyczenia (date, NOT NULL): data wypożyczenia egzemplarzu
   3. Data\_zwrotu (date, NOT NULL): data zwrotu egzemplarza
   4. Pesel (bigint) pesel osoby, która wypozycza

## Tabele dodatkowe

W bazie znajdują się 2 relacje wiele do wielu między tabelami:

1. Egzemplarze-wypożyczone
2. Egzemplarze-rezerwacje

Obydwie te relacje zawierają tabelę łączącą:

Tabela wypozyczone\_egzemplarze:

Kolumny:

1. wypozyczone\_ID\_wypozyczeina (integer, NOT NULL)
2. egzemplarze\_Rejestracja (text, NOT NULL)

Tabela rezerwacje\_egzemplarze:

Kolumny:

1. rezerwacje\_ID\_rezerwacji (integer, NOT NULL)
2. egzemplarze\_Rejestracje (text, NOT NULL)

# Relacje i węzły integralności

**Legenda:**

> - relacja wiele do jednego

< - relacja jeden do wielu

>< - relacja wiele do wielu

W bazie jest 6 relacji:

1. egzemplarze>modele – wiele egzemplarzy może należeć do 1 modelu:
   1. aktualizacja: kaskadowa
   2. usuwanie: kaskadowe
2. modele>pracownicy – opiekunowie mogą opiekować się kilkoma modelami na raz:
   1. aktualizacja: kaskadowa
   2. usuwanie: kaskadowe
3. rezerwacje>klienci – klienci mogą dokonywać wiele rezerwacji:
   1. aktualizacja: kaskadowa
   2. usuwanie: kaskadowe
4. wypożyczone>klienci – klienci mogą dokonywać wiele wypożyczeń:
   1. aktualizacja: kaskadowa
   2. usuwanie: wstaw NULL
5. egzemplarze><wypożyczone – w 1 wypożyczeniu może być wiele egzemplarzy a każdy egzemplarz może być wypożyczany wiele razy:
   1. aktualizacja: kaskadowa
   2. usuwanie: kaskadowe
6. egzemplarze><rezerwacje – w 1 rezerwacji może być wiele egzemplarzy a każdy egzemplarz może być rezerwowany wiele razy:
   1. aktualizacja: kaskadowa
   2. usuwanie: kaskadowe

# Funkcje i procedury w bazie danych

W bazie znajduje się podział na schematy względem CRUD. Występują 4 pakiety:

1. **dodawanie** – procedury dodające nowe wpisy do bazy danych
2. **modyfikowanie** – procedury umożliwiające edytowanie istniejących rekordów
3. **usuwanie** – procedury usuwające dane z bazy
4. **wyswietlanie** – funkcja pobierające informacje z bazy.

## Dodawanie rekordów (CREATE). Pakiet „dodawanie”

1. Procedury dodawania rekordów są odpowiedzialne za wprowadzanie nowych danych do tabel bazy danych. Każda procedura przyjmuje jako parametry wartości odpowiadające kolumnom w tabeli, a następnie wstawia je do odpowiedniej tabeli przy użyciu instrukcji INSERT.
2. Struktura ogólna procedur dodawania:
   * 1. Nazwa procedury**:** Zwykle w formacie dodaj\_<nazwa\_tabeli>.
     2. Parametry wejściowe**:** Każdy parametr odpowiada jednej kolumnie w tabeli.
     3. Działanie: Wstawienie nowego rekordu do tabeli.
3. Działanie w GUI: rysunek 9

Przykład dla konkretnej tabeli:

1. Procedura: dodajegzemplarz
2. Parametry:
   1. rejestracja text
   2. kolor text
   3. id\_modelu integer
   4. rok\_produkcji date
   5. cena integer
   6. przebieg integer
3. Działanie:
   1. Dodaje nowy egzemplarz do tabeli egzemplarze
4. Przykładowe ciało funkcji:  
   Obraz zawierający tekst, Czcionka, zrzut ekranu

   Opis wygenerowany automatycznie

## Modyfikacja rekordów (UPDATE). Pakiet „modyfikowanie”

1. Procedury modyfikacji rekordów służą do aktualizacji istniejących danych w tabelach. Każda procedura przyjmuje jako parametry identyfikator rekordu, który ma być zmodyfikowany, oraz nowe wartości dla wybranych kolumn. Aktualizacja odbywa się za pomocą instrukcji UPDATE.
2. Struktura ogólna procedur modyfikacji:
   1. Nazwa procedury: Zwykle w formacie modyfikuj\_<nazwa\_tabeli w licznie pojedynczej >.
   2. Parametry wejściowe:
      1. Klucz identyfikujący rekord do zmiany (np. klucz główny tabeli, taki jak id lub pesel).
      2. Parametry dla kolumn, które mogą być aktualizowane.
3. Działanie: Modyfikacja wartości w wybranych kolumnach dla rekordu wskazanego przez identyfikator.
4. Działanie w GUI: **rysunek 9**

### Przykład dla konkretnej tabeli:

Tabela: egzemplarze

1. Procedura**:** modyfikujegzemplarz
2. Parametry obowiązkowe:
   1. rejestracja text. Identyfikator egzemplarza, który ma zostać zmodyfikowany.
3. Parametry nieobowiązkowe:
   1. kolor text. Nowy kolor egzemplarza.
   2. id\_modelu integer. Identyfikator nowego modelu.
   3. rok\_produkcji date. Nowy rok produkcji.
   4. cena integer. Nowa cena egzemplarza.
   5. przebieg integer. Nowy przebieg egzemplarza.
4. Procedura wyszukuje rekord w tabeli egzemplarze na podstawie wartości parametru rejestracja. Następnie modyfikuje wartości pól zgodnie z podanymi parametrami. Jeśli wartość dla dowolnego parametru nieobowiązkowego jest NULL, odpowiednia kolumna pozostaje bez zmian.
5. Uwagi:
   1. W przypadku braku podania wartości dla parametru rejestracja, procedura zgłasza wyjątek.
   2. Jeśli rekord o podanej rejestracji nie istnieje, procedura zgłasza wyjątek informujący o braku rekordu.
6. Ciało funkcji:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

## Usuwanie rekordów(DELETE). Pakiet „usuwanie”

1. Procedury usuwania rekordów służą do usuwania istniejących danych w tabelach. Każda procedura przyjmuje jako parametry identyfikator rekordu, który ma być usunięty.
2. Struktura ogólna procedur usuwania:
   1. Nazwa procedury: Zwykle w formacie usun\_<nazwa\_tabeli w licznie pojedynczej>.
   2. Parametry wejściowe:
      1. Klucz identyfikujący rekord do usunięcia (np. klucz główny tabeli, taki jak id lub pesel).
   3. Działanie: Usunięcie rekordu w wybranej tabeli.
3. Działanie w GUI: **rysunek 9**

### Przykład dla konkretnej tabeli:

Tabela: egzemplarze

1. Procedura: usunegzemplarz
2. Parametry:
   1. rejestracja text. Identyfikator egzemplarza, który ma zostać usunięty.
3. Procedura wyszukuje rekord w tabeli egzemplarze na podstawie wartości parametru rejestracja i go usuwa.
4. Uwagi:
   1. W przypadku braku podania wartości dla parametru rejestracja, procedura zgłasza wyjątek.
   2. Jeśli rekord o podanej rejestracji nie istnieje, procedura zgłasza wyjątek informujący o braku rekordu.
5. Ciało funkcji:

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

## Wyświetlanie tabel(READ). Pakiet „wyświetlanie”

1. Funkcje wyświetlania dzielą się na 2 rodzaje. Do wyświetlania całych tabeli i do wyświetlania pojedynczych rekordów. W zależności, która funkcja zostanie wywołana, taka tabela zostanie obsłużona.
2. Ogólna struktura:
   1. Nazwa funkcji: W zależności czy chcemy wyświetlić pojedynczy rekord czy całą tabelę funkcje mają odpowiednią nazwę: <nazwa\_tabeli>all-do wyświetlania całych tabeli i wyświetl<nazwa\_tabeli>-do wyświetlania 1 rekordu.
   2. Parametry wejściowe:Funkcje do wyświetlania całych tabel nie przyjmują żadnych argumentów, natomiast te drugie, przyjmują klucz główny z danej tabeli.
   3. Działanie: Wypisanie potrzebnej informacji.
3. Działanie w GUI: **rysunek 9**

### Przykład dla konkretnej tabeli:

Tabela: egzemplarze

1. Funkcja: egzemplarzeall()
2. Parametry: brak
3. Funkcja zwraca zawartość całej tabeli egzemplarze.
4. Ciało funkcj:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

# Funkcje i procedury klienta. Pakiet „public”

## Opis procedury: wypozycz()

Procedura odpowiedzialna za realizację procesu wypożyczania samochodów przez klienta. Sprawdza dostępność samochodów w określonym terminie, weryfikuje istnienie klienta w bazie danych, a następnie zapisuje szczegóły wypożyczenia w odpowiednich tabelach.

Parametry:

1. **pesel\_** *(TEXT)* – Numer PESEL klienta dokonującego wypożyczenia.
2. **data\_zwrotu\_** *(DATE)* – Data planowanego zwrotu samochodów
3. **rejestracje\_** *(ARRAY[TEXT])* – Tablica zawierająca numery rejestracyjne samochodów do wypożyczenia.

Działanie:

1. Sprawdzenie dostępności samochodów:
   1. Procedura iteruje przez listę numerów rejestracyjnych samochodów (rejestracje\_).
   2. Dla każdego samochodu wywołuje funkcję czy\_wolny, która sprawdza
      1. Czy samochód istnieje w bazie.
      2. Czy nie jest aktualnie wypożyczany.
      3. Czy nie jest już zarezerwowany w podanym terminie
   3. W przypadku problemów, procedura przerywa działanie i generuje odpowiedni komunikat błędu:
      1. Auto z rejestracją % nie istnieje! – jeśli samochód o podanym numerze rejestracyjnym nie znajduje się w bazie.
      2. Auto z rejestracją % jest teraz wypożyczane! – jeśli samochód jest obecnie niedostępny z powodu trwającego wypożyczenia.
      3. Auto z rejestracją % jest już zarezerwowane w tym terminie! – jeśli samochód jest zarezerwowany w podanym przedziale czasowym
2. Weryfikacja klienta:
   1. Procedura sprawdza, czy klient o podanym numerze PESEL (pesel\_) istnieje w tabeli klienci.
   2. Jeśli klient nie istnieje, procedura przerywa działanie z komunikatem błędu: Ten klient nie istnieje.
3. Zapis wypożyczenia:
   1. Procedura dodaje nowe wypożyczenie do tabeli wypozyczone, rejestrując datę wypożyczenia (CURRENT\_DATE), datę zwrotu (data\_zwrotu\_) oraz numer PESEL klienta (pesel\_).
   2. Następnie iteruje przez listę rejestracji samochodów i zapisuje powiązane dane w tabeli wypozyczone\_egzemplarze, łącząc wypożyczenie z wypożyczanymi egzemplarzami pojazdów.

Wyjątki:

Procedura zgłasza następujące wyjątki w przypadku błędów:

1. Auto z rejestracją % nie istnieje! – gdy podany numer rejestracyjny nie znajduje się w bazie.
2. Auto z rejestracją % jest teraz wypożyczane! – gdy samochód jest aktualnie wypożyczany.
3. Auto z rejestracją % jest już zarezerwowane w tym terminie! – gdy samochód jest zarezerwowany w podanym terminie.
4. Ten klient nie istnieje – gdy klient o podanym numerze PESEL nie figuruje w bazie.

Tabele wykorzystywane w procedurze:

1. **klienci** – Weryfikacja istnienia klienta w bazie.
2. **wypozyczone** – Sprawdzanie czy podane egzemplarze są wolne w podanym terminie.
3. **wypozyczone\_egzemplarze** – Tabela łącząca z relacji wypozyczone - egzemplarze.
4. **egzemplarze** – Walidacji podanych rejestracji pojazdów.
5. **rezerwacje** – Sprawdzenie czy egzemplarze nie jest rezerwowany w tym terminie.
6. **rezerwacje\_egzemplarze** – Tabela łącząca z relacji rezerwacje – egzemplarze.

Działanie w GUI**: rysunek 5**

Ciało procedury:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

## Opis procedury: zarezerwuj()

Procedura odpowiedzialna za obsługę procesu dodawania rezerwacji.

Funkcjonalność:

Procedura umożliwia utworzenie nowej rezerwacji pojazdów przez klienta, przy uwzględnieniu kilku warunków weryfikacyjnych. Proces obejmuje sprawdzenie dostępności pojazdów, walidację klienta oraz zapis rezerwacji w odpowiednich tabelach.

Parametry wejściowe:

1. pesel\_ BIGINT – Numer PESEL klienta dokonującego rezerwacji
2. data\_rozpoczecia\_ DATE – Data rozpoczęcia rezerwacji.
3. data\_zakonczenia\_ DATE – Data zakończenia rezerwacji.
4. rejestracje\_ TEXT[] – Tablica zawierająca numery rejestracyjne pojazdów, które mają być zarezerwowane.

Logika działania:

1. Walidacja dostępności pojazdów:
2. Procedura iteruje przez tablicę numerów rejestracyjnych (rejestracje\_) i dla każdego z nich wywołuje funkcję czy\_wolny, która sprawdza:
   1. Czy pojazd istnieje w bazie danych (wynik = 1).
   2. Czy pojazd jest aktualnie wypożyczony (wynik = 2).
   3. Czy pojazd jest już zarezerwowany w podanym terminie (wynik = 3).
   4. W przypadku niespełnienia któregokolwiek warunku procedura przerywa działanie, podając odpowiedni komunikat błędu.
3. Weryfikacja klienta:
   1. Procedura sprawdza, czy klient o podanym numerze PESEL istnieje w tabeli klienci. Jeśli nie, generowany jest wyjątek: Ten klient nie istnieje.
4. Sprawdzenie daty rezerwacji:
   1. Rezerwacja musi dotyczyć przyszłości, tj. data\_rozpoczecia\_ musi być większa niż bieżąca data (CURRENT\_DATE). W przeciwnym razie procedura przerywa działanie z wyjątkiem: Rezerwować można tylko na przyszłość!.
5. Zapis rezerwacji:
   1. Po pozytywnej weryfikacji danych rezerwacja zostaje zapisana w tabeli rezerwacje, zawierającej szczegóły takie jak data rozpoczęcia i zakończenia rezerwacji oraz PESEL klienta
   2. Następnie dla każdego pojazdu z tablicy rejestracje\_ dodawany jest rekord w tabeli rezerwacje\_egzemplarze, łączącej konkretną rezerwację z rejestracjami egzemplarzy

Używane tabele:

1. **klienci** – Weryfikacja istnienia klienta w bazie.
2. **wypozyczone** – Sprawdzanie czy podane egzemplarze są wolne w podanym terminie.
3. **wypozyczone\_egzemplarze** – Tabela łącząca z relacji wypozyczone - egzemplarze.
4. **egzemplarze** – Walidacji podanych rejestracji pojazdów.
5. **rezerwacje** – Sprawdzenie czy egzemplarze nie jest rezerwowany w tym terminie.
6. **rezerwacje\_egzemplarze** – Tabela łącząca z relacji rezerwacje – egzemplarze.

Wyjątki obsługiwane w procedurze:

1. Auto z rejestracją % nie istnieje! – Pojazd o podanym numerze rejestracyjnym nie znajduje się w bazie.
2. Auto z rejestracją % jest teraz wypożyczane! – Pojazd jest aktualnie wypożyczany i niedostępny do rezerwacji.
3. Auto z rejestracją % jest już zarezerwowane w tym terminie! – Pojazd jest już zarezerwowany na wskazany termin.
4. Ten klient nie istnieje – Brak klienta o podanym numerze PESEL w bazie.
5. Rezerwować można tylko na przyszłość! – Próba rezerwacji na datę wcześniejszą lub równą bieżącej.

Działanie w GUI**: rysunek 6**

Ciało procedury:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

## Opis funkcji: do\_kiedy\_wolne()

Funkcja odpowiedzialna za określenie najpóźniejszej daty zwrotu pojazdów.

Funkcjonalność:

Funkcja umożliwia określenie klientowi, na jak długo może wypożyczyć wybrane egzemplarze oraz zachowuje integralność bazy danych.

Parametry wejściowe:

1. rejestracje\_ text[] – tablica zawierająca wszystkie egzemplarze, dla których należy określić najpóźniejszą datę.

Logika działania:

1. Deklaracja 2 zmiennych
   1. do\_kiedy DATE- potrzebna to przechowania najpóźniejszej daty dla 1 egzemplarza.
   2. najwcześniej DATE – używana przy algorytmie znalezienia najwcześniejszej daty z podanych egzemplarzy.
2. Określenie „z góry” najpóźniejszej daty (pozwalamy wypożyczać egzemplarze maksymalnie na 30 dni)
3. Przejście po wszystkich egzemplarzy i dla każdego egzemplarza:
   1. Sprawdzić czy aktualny egzemplarz ma rezerwację w przyszłości.
   2. Jeśli tak to pobrać daty rozpoczęcia wszystkich rezerwacji dla aktualnego egzemplarza, wybranie najwcześniejszej daty i przypisanie jej do zmiennej „do\_kiedy”.
   3. Sprawdzenie czy data „do\_kiedy” jest wcześniejsza niż data w zmiennej „najwcześniej”, jeśli tak to zaktualizuj zmienną „najwcześniej”.
4. Po przejściu przez wszystkie egzemplarze zwrócić datę „najwcześniej”

Ważne uwagi:

Funkcja ta powinna być wywoływana tylko dla **aktualnie wolnych egzemplarzach**, czyli takich dla których funkcja czy\_wolny() (opisana przy omawianiu procedur wypożycz() i zarezerwuj) zwraca 0 dla aktualnej daty.

Używane tabele:

1. **rezerwacje** – używana do porównywania dat
2. **egzemplarze** – odczyt podanych egzemplarzy
3. **rezerwacje\_egzemplarze** – tabela łącząca z relacji rezerwacje- egzemplarze

Działanie w GUI**: rysunek 6**

Ciało funkcji:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

## Opis funkcji: czy\_klient\_istnieje()

Funkcja używana podczas logowania klienta.

Funkcjonalność:

Sprawdza czy podany pesel istnieje w bazie danych.

Parametry wejściowe:

1. pesel\_ bigint – pesel rzekomego klienta do sprawdzenia.

Logika działania:

1. Wysyła zapytanie do bazy danych o pobranie podanego peselu, jeśli baza zwróciła wynik funkcja zwraca wartość logiczną **true** w przeciwnym razie **false**.

Używane tabele:

1. **klienci** – używana do sprawdzenia warunku.

Działanie w GUI**: rysunek 1,2**

Ciało funkcji:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

## Opis funkcji: sprawdz\_telefon()

Funkcja używana podczas rejestracji nowego klienta.

Funkcjonalność:

Sprawdza czy podany telefon istnieje w bazie danych.

Parametry wejściowe:

1. telefon\_\_ bigint – telefon rzekomego klienta do sprawdzenia.

Logika działania:

1. Wysyła zapytanie do bazy danych o pobranie podanego telefonu, jeśli baza zwróciła wynik funkcja zwraca wartość logiczną **true** w przeciwnym razie **false**.

Używane tabele:

1. **klienci** – używana do sprawdzenia warunku.

Działanie w GUI**: rysunek 1**

Ciało funkcji:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

## Opis funkcji: historia\_uzytkownika()

Funkcja przedstawia użytkownikowi jego historię wypożyczeń.

Funkcjonalność:

Pokazuję użytkownikowi jego wszystkie stare wypożyczenia oraz aktualnie wypożyczenia.

Parametry wejściowe:

1. pesel\_ bigint – pesel aktualnie zalogowanego użytkownika (klienta).

Logika działania:

1. Deklaracja 3 zmiennych:
   1. Historia\_tab historia– to specjalny typ rekordowy, który przechowuje datę wypozyczenia, datę zwrotu, kolor egzemplarza, rok\_produkcji, markę, model oraz informację czy jest to aktualne wypożyczenie.
   2. Historia – zmienna tego samego typu co powyższa, używana do przechowania wyniku zapytania.
   3. I int- deklaracja licznika pętli.
2. Pobranie od pazy danych wszystkich wypożyczeń klienta i iterowanie po nich.
   1. Dla każdej pozycji w zwróconym zapytaniu przepisanie pól z zmiennej historia do historia\_tab[] i dodanie informacji czy jest to aktualne wypożyczenie.
3. Zwrócenie tablicy w formie tabeli historia\_tab[];

Dane wyjściowe:

Funkcja zwraca tabelę, gdzie każdy rekord jest wypożyczeniem na konkretnym egzemplarzu, o którym wyświetlane są następująca dane:

1. Datę początek wypożyczenia
2. Datę końca wypożyczenia
3. Kolor wypożyczanego pojazdu
4. Rok produkcji pojazdu
5. Marka pojazdu
6. Model pojazdu
7. Informację czy wypożyczenie jest aktualne (czy klient powinien w przyszłości zwrócić jakiś pojazd)

Używane tabele:

1. **klienci** – dla wybrania aktualnie zalogowanego klienta
2. **wypozyczone** – dla odczytania informacji o datach wypożyczenia
3. **egzemplarze** – dla odczytania informacji o egzemplarzu
4. **modele** -dla uzyskaniu informacji o modelu konkretnego egzemplarza
5. **wypozyczone\_egzemplarze** – tabela łącząca w relacji wypożyczone – egzemplarze

Działanie w GUI**: rysunek 13,4**

Ciało funkcja:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

## Opis funkcji: info\_auto()

Funkcja wywoływana podczas wyświetlania informacji o aktualnie wybranym egzemplarzu.

Funkcjonalność:

Funkcja wydobywa z bazy danych informacje o podanym egzemplarzu.

Parametry wejściowe:

1. rejestracja\_ text – rejestracja egzemplarzu, dla którego będą zwracane informację.

Logika działania:

1. Deklaracja zmiennej informacje, która przechowuje wynik zapytania funkcji.
2. Pobranie informacji z bazy danych i przypisanie wyniku do zmiennej.
3. Sprawdzenie czy zmienna jest pusta, jeśli tak wyrzucenie wyjątku a jeśli nie to zwrócenie zmiennej.

Dane wyjściowe:

Funkcja zwraca 1 rekord w którym są zawarte informacje o egzemplarzu tj.

1. Rejestracja
2. Kolor
3. Rok produkcji
4. Cena za 1 dzień wypożyczenia
5. Stan licznika
6. Marka
7. Model
8. Pojemność silnika
9. Moc silnika
10. Imię opiekuna egzemplarza
11. Nazwisko opiekuna

Używane tabele:

1. **egzemplarze** – dla poprania informacji.
2. **modele** – dla pobrania informacji.
3. **pracownicy** – dla pobrania informacji o opiekunie egzemplarza.

Działanie w GUI**: rysunek 5,6**

Ciało funkcji: Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

## Opis funkcji: koszt()

Wywoływana podczas wybrania ilości dni wypożyczenia przez użytkownika.

Funkcjonalność:

Funkcja oblicza koszt wypożyczenia/rezerwacji egzemplarzy.

Parametry wejściowe:

1. początek date- początek naliczania opłat
2. koniec date- koniec naliczania opłat
3. rejestracje\_ text[] – tablica zamówionych egzemplarzy

Logika działania:

1. Deklaracja zmiennej suma, sumująca koszt wypożyczenia każdego wybranego egzemplarza.
2. Zsumowanie kosztu wypożyczenia każdego egzemplarza na dzeń.
3. Zwrócenie i pomnożenie sumy przez ilość dni między rozpoczęciem a zakończeniem naliczania kosztów.

Używane tabele:

1. **egzemplarze** – pobranie cen wypożyczeń.

Działanie w GUI**: rysunek 5**

Ciało funkcji:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

## Opis funkcji: wszystkierejestracje()

Wywoływana przy wczytywaniu gui klienta.

Funkcjonalność:

Zwraca rejestracje wszystkich egzemplarzy.

Dane wejściowe: brak

Logika działania:

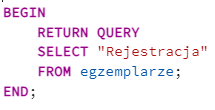
1. Pobiera wszystkie rejestracje z bazy i wypisuje je.

Używane tabele:

1. **egzemplarze** – pobranie informacji.

Działanie w GUI**: rysunek 5,6**

Ciało funkcji:



# Funkcje i procedury administratora

## Opis funkcji: liczenie\_pensji()

Funkcjonalność:

Liczy należną pensję pracownikom w zależności od wynagrodzenia bazowego i ilości egzemplarzy pod swoją opieką.

Parametry wejściowe: brak

Logika działania:

1. Pobiera i oblicza pensję dla każdego pracownika według wzoru: wynagrodzenie\_bazowe+300\*egzemplarze\_pod\_opieką.

Dane wyjściowe:

1. Imie pracownika
2. Nazwisko pracownika
3. Należna pensja
4. Data zatrudnienia pracownika.

Używane tabele:

1. **pracownicy** – pobranie imienia nazwiska i datę zatrudnienia.
2. **modele** – połączenie relacji pracownicy – egzemplarze
3. **egzemplarze** – liczenie ilości egzemplarzy dla 1 pracownika

Działanie w GUI**: rysunek 19**

## Opis funkcji: sprawdz\_rejestracje()

Wywoływana gdy administrator wpisuje rejestrację do sprawdzenia przychodów z niego.

Funkcjonalność:

Sprawdza czy egzemplarz z podaną rejestracja istnieje w bazie.

Parametry wejściowe:

1. rejestracja\_ text – rejestracja podana przez administratora.

Logika działania:

1. Wysyła zapytanie do bazy danych o pobranie podanej rejestracji, jeśli baza zwróciła wynik funkcja zwraca wartość logiczną **true** w przeciwnym razie **false**.

Używane tabele:

1. **egzemplarze** – sprawdzenie poprawności wpisanej rejestracji.

Działanie w GUI**: rysunek 15**

Ciało funkcji:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

## Opis funkcji wyliczających przychody z egzemplarzy

W bazie występują 2 funkcje odpowiedzialne za liczenie przychodów jakie wygenerowały egzemplarze (przychody\_z\_egzemplarza() i przychody\_z\_egzemplarzy()). Różnią się tylko tym że jedna pobiera parametr rejestracje danego egzemplarza i liczy przychody tylko dla 1 egzemplarza, a druga oblicza przychody dla wszystkich egzemplarzy, dlatego będą one opisywane wspólnie

Logika działania:

Funkcja używa zagnieżdżonego zapytania sql w którym:

1. Wewnętrzne zapytanie pobiera:
   1. Rejestrację
   2. markę
   3. model
   4. i koszt konkretnego wypożyczenia używając wcześniej omawianej funkcji koszt()
2. Wewnętrzne zapytanie jest grupowane względem rejestracji, marki i modelu a następnie funkcjami agregującymi liczona jest dla nich ilość wypożyczeń i sumowany koszt każdego wypożyczenia.
3. Zwrócenie całego zapytania.

Dane wyjściowe:

1. Rejestracja egzemplarzu
2. Marka
3. Model
4. Ilość wypożyczeń
5. Przychód jakie dany egzemplarz wygenerował

Używane tabele:

1. **egzemplarze** – do grupowania względem rejestracji
2. **modele** – do pobrania informacji o egzemplarzu
3. **wypozyczone** – do zliczenia kosztów wypożyczenia
4. **wypożyczone\_egzemplarze** – tabela łącząca w relacji wypożyczone – egzemplarze

Działanie w GUI**:**

* 1. przychody\_z\_egzemplarza() – **Rysunek 17**
  2. przychody\_z\_egzemplarzy() – **Rysunek 16,18**

Ciało przykładowej funkcji:

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

## Opis funkcji wyliczających przychody od klientów

Podobnie jak wyżej, w bazie występują 2 funkcje różniące się tylko parametrami wyjściowymi, dlatego będą one opisywane wspólnie,

Logika działania:

Funkcja używa zagnieżdżonego zapytania sql w którym:

1. Wewnętrzne zapytanie pobiera:
   1. Pesel
   2. Imię
   3. Nazwisko
   4. I koszt konkretnego wypożyczenia, jakie dokonał badany klient używając funkcji koszt()
2. Wewnętrzne zapytanie jest grupowane po peselu, imieniu i nazwisku a następnie funkcją agregującą liczona suma kosztów każdego wypożyczenia.
3. Zwrócenie całego zapytania.

Dane wyjściowe:

1. Pesel
2. Imię
3. Nazwisko
4. Koszt wszystkich wypożyczeń

Używane tabele:

1. **klienci** – grupowanie po peselu
2. **wypozyczone** – naliczanie kosztów wypożyczeń
3. **egzemplarze** – pobrane informacji jakie egzemplarze były wypożyczane
4. **wypozyczone\_egzemplarze** – tabela łącząca w relacji wypożyczone – egzemplarze

Działanie w GUI**:**

1. przychody\_z\_klienta() – **Rysunek 12**
2. przychody\_z\_klientow() – **Rysunek 11,14**

Ciało przykładowej funkcji:

# Inne procedury. Pakiet „public”

Opis procedury: **zrealizuj\_rezerwacje\_dla\_wszystkich()**

**Procedura odpowiedzialna za obsługę procesu realizacji rezerwacji.**  
Funkcjonalność: Procedura umożliwia przetworzenie i realizację rezerwacji pojazdów, które mają rozpoczęcie w przeszłości, a następnie usuwa przetworzone rezerwacje i ich powiązania.

1. Parametry wejściowe:
   1. Brak. Procedura działa na wszystkich rezerwacjach, które spełniają warunki weryfikacyjne.
2. Logika działania:
   1. **Wybór rezerwacji do realizacji:**
      1. Procedura iteruje przez wszystkie rekordy w tabeli rezerwacje, które mają datę rozpoczęcia wcześniejszą niż bieżąca (CURRENT\_DATE).
   2. Dodanie danych do tabeli wypożyczone:
      1. Dla każdej rezerwacji tworzony jest nowy rekord w tabeli wypozyczone, zawierający informacje o:
         1. Dacie wypożyczenia.
         2. Numerze PESEL klienta.
         3. Dacie zwrotu pojazdu (zgodnej z datą zakończenia rezerwacji).
   3. Przetwarzanie egzemplarzy związanych z rezerwacją:
      1. Procedura iteruje przez wszystkie egzemplarze związane z aktualną rezerwacją (w tabeli rezerwacje\_egzemplarze) i:
         1. Dodaje odpowiedni rekord do tabeli wypozyczone\_egzemplarze.
         2. Wykorzystuje identyfikator wypożyczenia (currval('wypozyczone\_seq')) w celu poprawnego powiązania.
   4. Usunięcie danych rezerwacyjnych:
      1. Usunięcie rekordów z tabeli rezerwacje\_egzemplarze dla przetwarzanej rezerwacji.
      2. Usunięcie samej rezerwacji z tabeli rezerwacje.
3. Używane tabele:
   1. **rezerwacje:** Źródło danych dla rezerwacji do realizacji.
   2. **rezerwacje\_egzemplarze:** Powiązania między rezerwacjami a egzemplarzami pojazdów.
   3. **wypozyczone:** Docelowa tabela zawierająca dane o zrealizowanych wypożyczeniach.
   4. **wypozyczone\_egzemplarze:** Tabela łącząca wypożyczenia z egzemplarzami pojazdów.
4. Procedura wykonywana jest codziennie o północy za pomocą pgAgent.

Ciało procedury:

Obraz zawierający tekst, zrzut ekranu, Czcionka, dokument

Opis wygenerowany automatycznie

# Koncepcja dostępu zdalnego:

1. Możliwości dostępu zdalnego:
   1. Klient – użytkownik końcowy systemu z ograniczonym dostępem.
   2. Administrator – użytkownik z pełnym dostępem do funkcji systemu.
2. Funkcjonalności administratora dla wszystkich tabel:
   1. Dodawanie nowych rekordów do bazy
   2. Usuwanie dowolnych rekordów.
   3. Modyfikowanie dowolnych rekordów
   4. Wyświetlanie zawartości tabel
3. Funkcjonalności administratora dla tabeli „klienci”
   1. Podgląd łącznych przychodów wygenerowanych przez wszystkich klientów.
   2. Podgląd przychodów jaki wygenerował pojedynczy klient
   3. Generowanie wykresu przedstawiającego przychody wygenerowane przez klientów.
   4. Wyświetlanie historii wypożyczeń każdego klienta
4. Funkcjonalności administratora dla tabeli „egzemplarze”
   1. Podgląd łącznych przychodów wygenerowanych przez wszystkie samochody.
   2. Podgląd przychodów wygenerowanych przez pojedynczy samochód.
   3. Generowanie wykresu przedstawiającego przychody wygenerowane przez wszystkie samochody.
5. Funkcjonalności klienta:
   1. Tworzenie nowego użytkownika (konta) – umożliwienie rejestracji nowego użytkownika w systemie.
   2. Wyświetlanie swoich danych wpisanych podczas rejestracji
   3. Podglądanie historii wypożyczeń – wgląd w historię wypożyczonych samochodów.
   4. Wypożyczanie dowolnej liczby samochodów – możliwość jednoczesnego wypożyczenia wielu pojazdów
   5. Rezerwowanie dowolnej liczby samochodów – możliwość dokonywania rezerwacji na wiele pojazdów.
6. Funkcjonalności ogólne
   1. Sprawdzanie istnienia klienta – weryfikacja, czy klient o podanym numerze PESEL znajduje się w bazie danych.
   2. Sprawdzanie istnienia samochodu – weryfikacja, czy samochód o podanym numerze rejestracyjnym znajduje się w bazie danych.
   3. Ustawianie maksymalnej liczby dni na wypożyczenie samochodu – definiowanie limitu czasu, na jaki można wypożyczyć pojazd.
   4. Wyświetlanie dostępnych samochodów do wypożyczenia – pokazywanie wyłącznie pojazdów, które nie są obecnie wypożyczone.
   5. Wyświetlanie dostępnych samochodów do rezerwacji – pokazywanie wyłącznie pojazdów, które nie są obecnie wypożyczone.

# Zrealizowany dostęp zdalny:

## Wykorzystywane technologie:

* 1. Język programowania C#.
  2. WPF Application– aplikacja wykorzystująca framework WPF do tworzenia interfejsu użytkownika.
  3. Visual Studio –środowisko programistyczne używane do tworzenia aplikacji.
  4. .NET – platforma programistyczna do tworzenia aplikacji.
  5. ADO.NET – technologia umożliwiająca dostęp do bazy danych.
  6. Npgsql – biblioteka umożliwiająca połączenie z bazą danych PostgreSQL.
  7. OxyPlot – biblioteka służąca do generowania wykresów.

## Pliki połączenia z bazą danych

1. Projekt w Visual Studio
   1. Plik konfiguracyjny: **App.config**, dane do uzupełnienia:
      1. Host – domyślnie „localhost”
      2. Port – domyślnie „5433”
      3. Username – domyślnie „postgres”
      4. Password – domyślnie „admin”
      5. Database – domyślnie „Wypozyczalnia”
2. Zainstalowany program:
   1. Plik konfiguracyjny: **GUI.dll.config**
   2. Dane do uzupełnienia: identyczne jak w punkcie (a)

## Rejestracja klienta

1. Funkcjonalność rejestracji:
   1. Nowy klient może utworzyć konto poprzez wprowadzenie danych w formularzu rejestracyjnym. Formularz zawiera następujące pola:
   2. PESEL – pole obowiązkowe oznaczone (\*)
      1. Maksymalna długość: 11 znaków
      2. Służy jako klucz główny do identyfikacji klienta w bazie danych
   3. Imię – pole obowiązkowe oznaczone (\*)
      1. Służy do personalizacji i identyfikacji klienta.
   4. Nazwisko - pole obowiązkowe oznaczone (\*)
      1. Wymagane do identyfikacji.
   5. Email – pole nieobowiązkowe
   6. Numer telefonu - pole obowiązkowe oznaczone (\*)
      1. Maksymalna długość: 9 znaków
      2. Unikalny numer w systemie
2. Obsługa przycisków:
   1. Przycisk „Zarejestruj”
      1. Funkcja: Sprawdza poprawność danych wejściowych.
      2. W przypadku poprawnych danych: Tworzy nowy rekord w tabeli klienci w bazie danych PostgreSQL przy użyciu ADO.NET i biblioteki Npgsql.
      3. Obsługiwane komunikaty sukcesu/błędu są wyświetlane w aplikacji.
   2. Przycisk „Zaloguj
      1. Funkcja: Zamknięcie okna rejestracji i powrót użytkownika do okna logowania.
3. Obsługa błędów:
   1. System walidacji danych zapewnia:
      1. Sprawdzenie, czy wszystkie obowiązkowe pola zostały wypełnione.
      2. Weryfikację długości numeru PESEL.
      3. Sprawdzenie, czy PESEL i numer telefonu są unikalne (nie istnieją już w bazie danych).
4. W przypadku błędnych danych, komunikaty o błędach są wyświetlane w zależności od pola, którego dotyczą.
   1. "Pesel jest pusty lub jest za krótki"
   2. "Musisz podać imię"
   3. "Musisz podać nazwisko"
   4. "Musisz podać numer telefonu lub sprawdzić jego długość"
   5. "Użytkownik o podanym peselu już istnieje"
   6. "Numer telefonu jest już zajęty"
5. Podgląd okna:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Rysunek 1**

## Logowanie klienta

1. Funkcjonalność logowania:
   1. Klient, który posiada już konto, może zalogować się do aplikacji poprzez wypełnienie formularza logowania. Formularz logowania zawiera następujące pola
   2. PESEL
      1. Maksymalna długość: 11 znaków
      2. Służy jako klucz główny do identyfikacji klienta w bazie danych
2. Obsługa przycisków:
   1. Przycisk „Zaloguj”
      1. Funkcja: Sprawdza poprawność wpisanego numeru PESEL.
      2. W przypadku poprawnych danych: zamyka okno logowania i przechodzi do okna opcji użytkownika.
      3. Obsługiwane komunikaty błędu są wyświetlane w aplikacji.
   2. Przycisk „Zarejestruj się”:
      1. Funkcja: Zamknięcie okna logowania i przeniesienia użytkownika do okna rejestracji.
   3. Przycisk „Zaloguj się” – administratora
      1. Zamyka okno logowania użytkownika oraz otwiera okno logowania administratora.
3. Obsługa błędów:
   1. System walidacji danych zapewnia:
      1. Sprawdzenie, czy pole PESEL zostało wypełnione.
      2. Weryfikację długości numeru PESEL
4. W przypadku błędnych danych, komunikat o błędzie jest wyświetlany bezpośrednio pod przyciskiem "Zaloguj”.
   1. „Użytkownik o podanym peselu nie istnieje”
5. Podgląd okna:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Rysunek 2**

## Okno opcji klienta

1. Funkcjonalność okna:
   1. Klient, który zalogował się przy użyciu numeru PESEL, ma dostęp do trzech głównych opcji.
   2. Każda opcja jest uruchamiana po kliknięciu odpowiedniego przycisku w interfejsie.
2. Obsługa przycisków:
   1. Działanie wspólne dla wszystkich przycisków:
      1. Zamknięcie aktualnie otwartego okna
   2. Przycisk „Informacje o koncie”:
      1. Otwiera okno z historią wypożyczeń użytkownika oraz informacjami o koncie.
   3. Przycisk „Wypożycz samochód”:
      1. Otwiera okno aplikacji umożliwiające klientowi wypożyczanie dostępnych samochodów.
   4. Przycisk „Zarezerwuj samochód”:
      1. Otwiera okno aplikacji umożliwiające klientowi rezerwowanie dostępnych samochodów.
   5. Przycisk „Wyloguj”
      1. Otwiera okno logowania klienta.
3. Podgląd okna:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Rysunek 3**

## Okno historii wypożyczeń użytkownika

1. Funkcjonalności okna:
   1. Okno umożliwia użytkownikowi przeglądanie historii wypożyczeń. Wyświetlane informacje o każdym wypożyczeniu obejmują:
      1. Datę wypożyczenia.
      2. Datę zwrotu
      3. Kolor samochodu
      4. Roku produkcji samochodu
      5. Markę samochodu
      6. Model samochodu
      7. Informację, czy samochód jest aktualnie wypożyczony
   2. Dodatkowo, okno wyświetla dane osobowe użytkownika podane podczas rejestracji:
      1. Imię
      2. Nazwisko
      3. Numer PESEL
      4. Email (jeśli podano).
      5. Numer telefonu.
2. Obsługa przycisków:
   1. Przycisk „Powrót”:
      1. Zamyka obecnie otwarte okno
      2. Przenosi użytkownika do wyboru opcji klienta
3. Podgląd okna dla użytkownika o numerze PESEL (11111111111)

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Rysunek 4**

## Wypożyczanie samochodów

1. Funkcjonalności:
   1. Wybieranie dostępnych samochodów z listy
   2. Po wybraniu samochodu wyświetlane są szczegółowe informacje:
      1. Marka
      2. Model
      3. Pojemność silnika
      4. Przebieg
      5. Data Produkcji
      6. Moc silnika
      7. Numer Rejestracyjny
      8. Kolor samochodu
      9. Imię i nazwisko opiekuna
      10. Cena za dzień wypożyczenia
   3. Dodawanie samochodu do koszyka:
      1. Samochód można dodać do koszyka poprzez zaznaczenie checkboxa w podglądzie egzemplarzu.
   4. Wyświetlanie koszyka:
      1. Lista wszystkich wybranych przez użytkownika samochodów jest wyświetlana w koszyku.
   5. Usuwanie samochodu z koszyka:
      1. Poprzez podwójne kliknięcie na nazwę samochodu w koszyku
      2. Kliknięcie zaznaczonego check box’a
   6. Wybór liczby dni wypożyczenia:
      1. Użytkownik wybiera liczbę dni, na które chce wypożyczyć samochody.
      2. Liczba dni jest ograniczona do najkrótszego czasu dostępności spośród wszystkich samochodów w koszyku.
      3. Dostępność jest obliczana za pomocą funkcji **pg/plsql do\_kiedy\_wolne()**.
   7. Wyświetlanie ceny końcowej w zależności od ilości dni i egzemplarzy samochodów wybranych przez użytkownika
      1. Cena końcowa = (Cena za dzień każdego egzemplarzu) × (Liczba dni)
2. Obsługa przycisków:
   1. Przycisk „Zamów”:
      1. Resetuje obecne ustawienia okna do ustawień domyślnych:
         1. Check box
         2. Koszyk
         3. Listę dostępnych samochodów
      2. Dodaje wszystkie samochody, wybrane przez użytkownika do tabeli „wypozyczone”
      3. Wyświetla komunikat o pomyślnym wypożyczeniu
   2. Przycisk „Wyloguj”:
      1. Zamyka obecnie otwarte okno
      2. Przenosi użytkownika do wyboru opcji klienta
3. Podgląd okna:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Rysunek 5**

## Rezerwowanie samochodów

1. Funkcjonalności:
   1. Wybieranie dostępnych samochodów z listy
   2. Po wybraniu samochodu wyświetlane są szczegółowe informacje:
      1. Marka
      2. Model
      3. Pojemność silnika
      4. Przebieg
      5. Data Produkcji
      6. Moc silnika
      7. Numer Rejestracyjny
      8. Kolor samochodu
      9. Imię i nazwisko opiekuna
      10. Cena za dzień wypożyczenia
   3. Dodawanie samochodu do koszyka:
      1. Samochód można dodać do koszyka poprzez zaznaczenie checkboxa w podglądzie egzemplarzu.
   4. Wyświetlanie koszyka:
      1. Lista wszystkich wybranych przez użytkownika samochodów jest wyświetlana w koszyku.
   5. Usuwanie samochodu z koszyka:
      1. Poprzez podwójne kliknięcie na nazwę samochodu w koszyku
      2. Kliknięcie zaznaczonego check box’a
   6. Ustawianie daty początkowej rezerwacji pojazdów:
      1. Data ustawiana jest według wzoru: Dziś + jeden dzień
   7. Ustawianie daty końcowej rezerwacji:
      1. Domyślna data końcowa to: **Wybrana data początkowa + 1 dzień**.
      2. Użytkownik może zmieniać datę końcową, jednak:
         1. Data końcowa nie może być wcześniejsza niż data początkowa.
         2. Maksymalny zakres wyboru daty końcowej to **30 dni do przodu** od daty początkowej.
2. Obsługa przycisków:
   1. Przycisk „Zarezerwuj”:
      1. Resetuje obecne ustawienia okna do ustawień domyślnych:
         1. Check box
         2. Koszyk
         3. Listę dostępnych samochodów
      2. Dodaje wszystkie samochody, wybrane przez użytkownika do tabeli „rezerwacje”
      3. Wyświetla komunikat o pomyślnej rezerwacji
   2. Przycisk „Wyloguj”:
      1. Zamyka obecnie otwarte okno
      2. Przenosi użytkownika do wyboru opcji klienta
3. Obsługa błędów:
   1. W przypadku, gdy użytkownik ustawi zakres rezerwacji, w którym co najmniej jeden z wybranych samochodów jest już zajęty:
      1. Wyświetli się komunikat o tym , że co najmniej jeden z samochodów jest zarezerwowany w tym terminie
      2. Następuje przywrócenie ustawień domyślnych okna, takich jak opisane w punkcie **2a**
4. Podgląd okna:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Rysunek 6**

## Logowanie administratora

1. Funkcjonalność logowania:
   1. Administrator loguje się, wpisując stałe hasło w formularzu logowania, który zawiera następujące pole:
      1. Hasło
      2. Domyślne hasło to: **"123"**.
      3. Służy do identyfikacji administratora zarządzającego bazą danych.
2. Obsługa przycisków:
   1. Przycisk „Zaloguj”
      1. Funkcja: Sprawdza poprawność wpisanego hasła.
      2. W przypadku poprawnych danych: zamyka okno logowania i przechodzi do okna administratora.
   2. Przycisk „Zaloguj się” – klienta
      1. Zamyka okno logowania administratora oraz otwiera okno logowania klienta.
3. Obsługa błędów jest całkowicie zrobione w C#:
   1. System walidacji danych zapewnia:
      1. Sprawdzenie, czy hasło jest poprawne.
      2. Niepoprawne hasło nie pozwala przejść do okna administratora
4. Podgląd okna:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

**Rysunek 7**

## Okno administratora

1. Funkcjonalność okna:
   1. Administrator, który się zalogował, ma dostęp do sześciu głównych opcji.
      1. Każda opcja jest uruchamiana po kliknięciu odpowiedniego przycisku w interfejsie.
      2. Każdy przycisk odpowiada tabeli w bazie danych, którą administrator może zarządzać
2. Obsługa przycisków:
   1. Działanie wspólne dla wszystkich przycisków:
      1. Zamknięcie aktualnie otwartego okna
   2. Przycisk „Egzemplarze”:
      1. Otwiera okno z danymi z tabeli „egzemplarze”.
   3. Przycisk „Klienci”:
      1. Otwiera okno z danymi z tabeli „klienci”.
   4. Przycisk „Modele”:
      1. Otwiera okno z danymi z tabeli „modele”.
   5. Przycisk „Pracownicy”:
      1. Otwiera okno z danymi z tabeli „pracownicy”.
   6. Przycisk „Rezerwacje”:
      1. Otwiera okno z danymi z tabeli „rezerwacje”.
   7. Przycisk „Wypozyczone”:
      1. Otwiera okno z danymi z tabeli „wypozyczone”.
   8. Przycisk „Wyloguj”
      1. Otwiera okno logowania administratora.
3. Podgląd:

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

**Rysunek 8**

## CRUD – operacje na bazie

Wszystkie tabele pobierane są z bazy danych za pomocą funkcji. Oraz działają w identyczny sposób, różnią się tylko danymi, które zależą od tabeli, która została wybrana w oknie administratora.

### Dodawanie danych(CREATE)

* 1. Administrator może dodać nowy rekord, korzystając z przycisku **„+”**:
     1. Po kliknięciu przycisku zostaje utworzony nowy wiersz w tabeli.
     2. Administrator musi uzupełnić wszystkie wymagane pola w odpowiednich tabelach:
        1. „Egzemplarze”
        2. „Klienci”
     3. W tabelach, w których pierwsza kolumna to identyfikator (**ID**) ustawiony jako **auto-increment**, pole to należy pozostawić puste – ID zostanie wygenerowane automatycznie.
  2. Po kliknięciu przycisku **„+”**, administrator zostaje poinformowany o trybie dodawania za pomocą komunikatu:
     1. „Jesteś w trybie dodawania. Zmodyfikowane rekordy nie zostaną zapisane!”
  3. Podczas trybu dodawania:
     1. Modyfikowanie istniejących rekordów w tabeli nie wpływa na ich rzeczywiste dane w bazie.
  4. Aby zapisać nowo dodany rekord w bazie danych, należy kliknąć przycisk **„Zapisz**”.
     1. Wpisane dane zostają przekazane jako parametry procedury.
     2. Procedura zostaje wywołana.
  5. Aby anulować proces dodawania, należy kliknąć przycisk **„Anuluj”**, co spowoduje:
     1. Wyjście z trybu dodawania.
     2. Usunięcie niezapisanego wiersza tabeli.
  6. Walidacja danych:
     1. Tabela, w której przechowywane są dane, automatycznie, sprawdza typ wpisywanych danych
        1. Zapobiega wprowadzeniu danych o niewłaściwym typie (np. liczby w polach tekstowych lub znaków w polach liczbowych).

### Wyświetlanie tabel (READ):

1. Tabela z danymi wyświetla się automatycznie po załadowaniu okna.
2. Administrator ma pełny podgląd wszystkich kolumn i wierszy w wybranej tabeli.
3. Data wyświetlana jest w systemie miesiąc/dzień/rok

### Modyfikowanie danych(UPDATE)

1. a. Modyfikacja danych w tabeli odbywa się poprzez **dwukrotne kliknięcie myszką** w komórkę, którą chcemy edytować.
2. Modyfikacji podlegają wszystkie kolumny **z wyjątkiem**:
   * 1. Pierwszej – klucz główny tabeli
     2. Ostatniej – przeznaczonej na przycisk usuwania rekordu.
3. Zatwierdzenie wprowadzonych zmian odbywa się za pomocą:
   * 1. **Naciśnięcia klawisza „Enter”** – zapisuje zmiany w bieżącej komórce.
     2. **Kliknięcia poza edytowaną komórkę tabeli** – automatycznie zatwierdza wprowadzone zmiany.
4. Zapis:
   * 1. Zmodyfikowany rekord zostaje przekazany jako parametr procedury.
     2. Procedura zostaje wywołana.
5. Walidacja danych:
   * 1. Tabela, w której przechowywane są dane, automatycznie, sprawdza typ wpisywanych danych
        1. Zapobiega wprowadzeniu danych o niewłaściwym typie (np. liczby w polach tekstowych lub znaków w polach liczbowych).

### Usuwanie danych(DELETE)

* 1. Każdy wiersz tabeli posiada w ostatniej kolumnie przycisk umożliwiający jego usunięcie.
  2. Po kliknięciu na przycisk:
     1. System automatycznie odczytuje wartość klucza głównego z pierwszej kolumny wybranego wiersza.
     2. Klucz główny pełni rolę unikalnego identyfikatora rekordu, który ma zostać usunięty.
     3. Funkcja usuwa dany rekord poprzez wywołanie odpowiedniej procedury

1. Podgląd okna dla tabeli „modele”:

Obraz zawierający tekst, zrzut ekranu, numer, oprogramowanie

Opis wygenerowany automatycznie

**Rysunek 9**

# Statystyka

## Tabela klienci

Możliwość dostępu uzyskuje się poprzez wypełnienie odpowiednich danych, lub/oraz wciśniecie odpowiednich przycisków.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

**Rysunek 10**

* 1. Administrator ma podgląd do:
     1. Wyświetlania przychodów ze wszystkich klientów oraz ich danych (przycisk – „przychód z klientów”):
        1. Numer PESEL
        2. Imię
        3. Nazwisko
        4. Przychód (koszt)

Podgląd okna:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

**Rysunek 11**

* 1. Wyświetlania przychodów z pojedynczego klienta poprzez wpisanie jego numeru PESEL (przycisk – przycisk „dochód z klienta”)
     1. Dane wyświetlane dla klienta są identyczne jak w punkcie (a)

Podgląd dla klienta o numerze PESEL: 11111111111

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Rysunek 12**

* 1. Wyświetlania historii wypożyczeń klienta wraz z danymi(przycisk – „historia wypożyczeń klienta”):
     1. Datę wypożyczenia.
     2. Datę zwrotu
     3. Kolor samochodu
     4. Roku produkcji samochodu
     5. Markę samochodu
     6. Model samochodu
     7. Informację, czy samochód jest aktualnie wypożyczony

Podgląd okna dla klienta o numerze PESEL: 11111111111

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

**Rysunek 13**

* 1. Administrator ma dostęp do wykresu, który jest bezpośrednim odzwierciedleniem danych z punktu (a).
     1. Wykres jest dostępny pod przyciskiem „Pokaż wykres przychodów”
     2. Wykres przedstawia:
        1. Imię i nazwisko
        2. Kwotę jaką wygenerował klient

Podgląd okna:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Opis wygenerowany automatycznie

**Rysunek 14**

## Tabela „egzemplarze”

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

**Rysunek 15**

1. Administrator ma podgląd do:
2. Wyświetlania przychodów jakie wygenerowały samochody oraz ich danych (przycisk – „Przychód z pojazdów”):
   * + 1. Numer rejestracyjny
       2. Marka
       3. Model
       4. Ilość wypożyczeń
       5. Przychód samochodu

Podgląd okna:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

**Rysunek 16**

1. Wyświetlania przychodów z pojedynczego samochodu poprzez wpisanie jego numeru rejestracyjnego (przycisk – „dochód z jednego pojazdu”)
2. Dane wyświetlane dla klienta są identyczne jak w punkcie (a)

Podgląd okna dla samochodu z numerem rejestracyjnym WW3344:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Rysunek 17**

1. Administrator ma dostęp do wykresu, który jest bezpośrednim odzwierciedleniem danych z punktu (a).
2. Wykres jest dostępny pod przyciskiem „Pokaż wykres przychodów”
3. Wykres przedstawia:
   * + 1. Marka
       2. Model
       3. Przychód jaki wygenerował dany samochód

Podgląd okna:

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, numer

Opis wygenerowany automatycznie

**Rysunek 18**

## Tabela „pracownicy”

1. Administrator ma podgląd do graficznej reprezentacji końcowej pensji pracowników, opisanej w funkcji: liczenie\_pensji():
   * 1. Prezentacja wyników jest wyświetlana w przed ostatniej kolumnie.

Obraz zawierający tekst, zrzut ekranu, numer, oprogramowanie

Opis wygenerowany automatycznie

**Rysunek 19**

# Harmonogram zadań w pgAgent

W systemie PostgreSQL, przy użyciu narzędzia **pgAgent**, utworzono zadanie automatyzujące proces realizacji rezerwacji w bazie danych.

Opis zadania:

Zadanie zostało skonfigurowane w sposób, który codziennie o godzinie **00:00** wywołuje procedurę **zrealizuj\_rezerwacje\_dla\_wszystkich()**. Procedura ta odpowiada za obsługę wszystkich rezerwacji w systemie, które powinny zostać zrealizowane tego dnia.

Kluczowe ustawienia zadania:

* + - * 1. **Nazwa zadania**: codziene\_sprawdzanie\_rezerwacji
        2. **Częstotliwość uruchamiania**: Codziennie o godzinie 00:00
        3. **Wywoływana procedura**: zrealizuj\_rezerwacje\_dla\_wszystkich()
        4. **Narzędzie do harmonogramowania**: pgAgent
        5. **Typ zadania**: Zapytanie SQL
        6. **Treść polecenia SQL**: CALL zrealizuj\_rezerwacje\_dla\_wszystkich();

Cel zadania:

Automatyczne uruchamianie procedury umożliwia:

* 1. Regularne przetwarzanie rezerwacji w sposób bezobsługowy.
  2. Zapewnienie aktualności danych w systemie.
  3. Optymalizację pracy systemu dzięki uniknięciu ręcznego uruchamiania procedur.

Zrzuty ekranu:

* 1. Ustawienia ogólne:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

* 1. Kroki:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

1. Plan wykonywania:
   1. Obraz zawierający tekst, zrzut ekranu, numer, oprogramowanie

      Opis wygenerowany automatycznie
   2. Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

      Opis wygenerowany automatycznie