Politechnika Wrocławska

# Projekt bazy danych - warsztat samochodowy

*Piotr Domagała* *Krzysztof Chmielewski*

wtorek TP 15
zima 2019

# Spis treści

# 1 Wstęp

## 1.1 Cel projektu

Zapoznanie się z zagadnieniem baz danych, w rzeczywistym kontekście - małej firmy prowadzącej warsztat samochodowy. Poznanie środowiska i języka SQL, następnie zaimplementowanie kodu na potrzeby projektu bazy danych, obsługującej warsztat. Analiza i przedstawieni rozwiązania problemów wydajności i opłacalności zastosowanej formy bazy.

## 1.2 Zakres projektu

Projekt obejmuje przeprowadzenie analizy zapotrzebowania małego warsztatu samochodowego, zależnie od kadr, spektrum wykonywanych usług, jak i ilości operacji online. Pomijana jest dokładna implementacja aplikacji webowej i strony internetowej. Implementacja tyczy się głównie zagadnienia bazy, jej zawartości i mechanizmów przy niej wykorzystanych.

# 2 Analiza wymagań

## 2.1 Opis działania i schemat logiczny

Plan działania dla projektu można podzielić na kilka etapów:

1. Wykonanie modelu konceptualnego i analiza wymagań
2. Wykonanie modelu logicznego
3. Wykonanie modelu fizycznego
4. Implementacja bazy danych
5. Wykonanie strony internetowej i konfiguracja jej z bazą danych
6. Testowanie projektu
7. Wykonanie dokumentacji i prezentacji projektu

## 2.2 Wymagania funkcjonalne

Poniżej znajdują się wymagania wraz z krótkim opisem warunków użycia.

- Rejestracja wizyty w terminarzu

  Warunek początkowy:
  Nowe zlecenie i istnienie wolnych terminów.

  Warunek końcowy:
  Potwierdzenie wizyty przez Mechanika i utworzenie nowego rekordu w bazie odpowiadającego nowej wizycie.

- Sprawdzenie historii naprawy

  Warunek początkowy:
  Klient miał już przeprowadzone wcześniej naprawy.

  Warunek końcowy:
  Uzyskanie przez Klienta/Mechanika pożądanych informacji.

- Sprawdzenie statusu i stanu naprawy

  Warunek początkowy:
  a) Istnienie naprawy,
  b) Klient posiada założoną wizytę w firmie.

  Warunek końcowy:
  a) Sprawdzenie stanu,
  b) Zaakceptowanie wprowadzonej zmiany przez Mechanika.

- Generowanie prostych statystyk

  Warunek początkowy:
  Baza danych zawiera dane o wizytach.

  Warunek końcowy:
  Zakończenie przeglądania poprzez naciśnięcie przycisku.

- Przechowywanie danych o wizytach i klientach

  Warunek początkowy:
  Prowadzenie działalności i sukcesywne wprowadzanie informacji.

  Warunek końcowy:
  Posiadanie skompletowanych danych.

- Modyfikacja klienta

  Warunek początkowy:
  a) Klient musi pojawić się w warsztacie lub uruchomić stronę internetową,
  b) Posiadanie konta w firmie konta przez klienta,
  c) Istnieje w bazie danych konto klienta.

  Warunek końcowy
  a) Kompletne wypełnienie formularza rejestracyjnego,
  b) W bazie nie ma już danych klienta (z wyjątkiem niezbędnych do prowadzenia księgowości),
  c) Dane w bazie po aktualizacji odpowiadają stanowi rzeczywistemu.

- Modyfikacja mechanika

  Warunek początkowy:
  Aby dokonać zmiany danych pracownik musi istnieć, a rekord być do wglądu.

  Warunek końcowy:
  Dodanie/usunięcie/modyfikacja wszystkich niezbędnych danych pracowników w bazie.

- Tworzenie kopi zapasowych (niezaimplementowane)

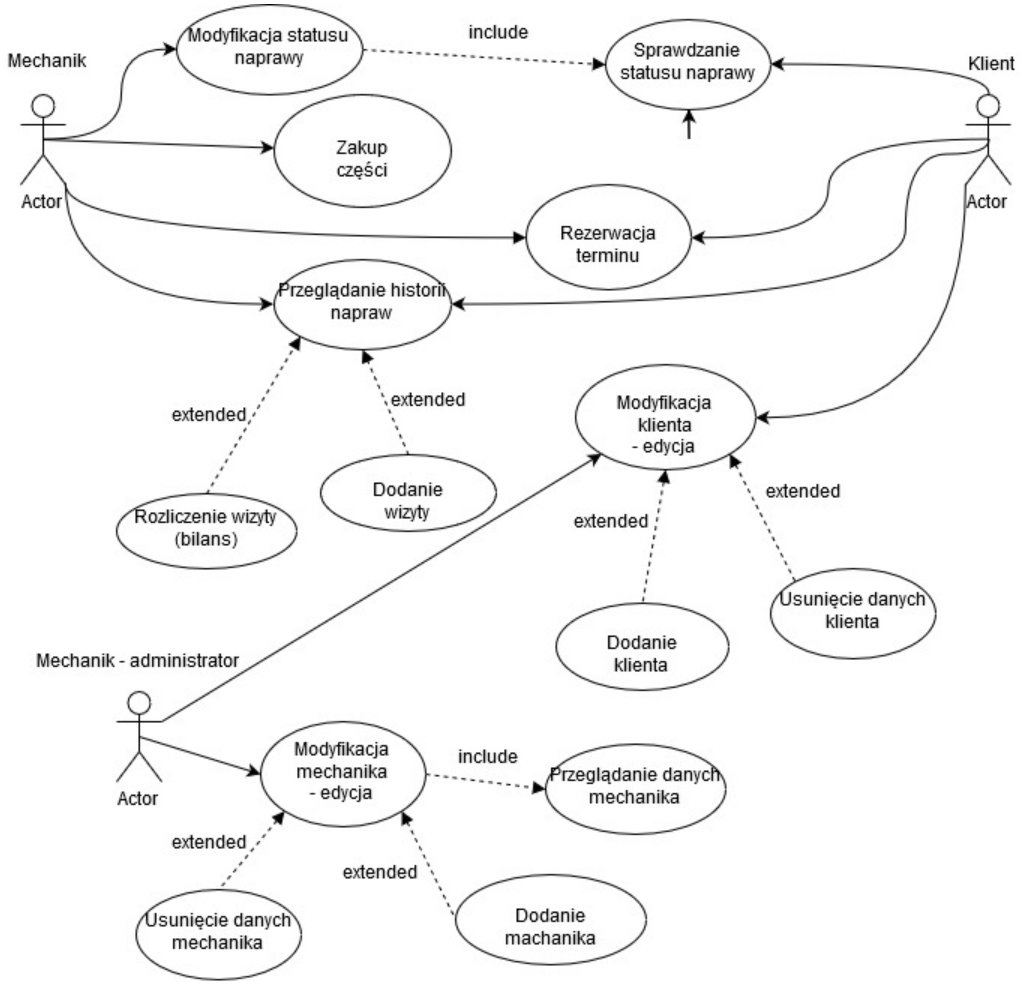
  Warunek początkowy:
  Baza danych zawiera jakiekolwiek dane.

  Warunek końcowy:
  Zakończenie archiwizacji poprzez utworzenie pliku z którego możliwe jest przywróceniestanu z chwili obecnej.

- Przypomnienie o najbliższych wizytach (niezaimplementowane)

  Warunek początkowy:
  Istnieje wizyta w terminarzu.

  Warunek końcowy:
  Automatyczne przypomnieni dla klienta o wizycie dzień wcześniej.




Rysunek 1: Diagram przypadków użycia

## 2.3 Wymagania niefunkcjonalne

1. Odporność na utratę danych - tworzenie kopi zapasowych i archiwizacja danych, według wcześniej ustalonego harmonogramu

2. Poufność danych – ograniczony dostęp do przechowywanych informacji. Ograniczenie uprawnień pracowników warsztatu. Inne uprawnienia dla osoby obsługującej klienta, inne dla mechaników, w zakresie operacji i dostępu do informacji. Realizacja za pomocą kont w systeme.

3. Wydajność – maksymalnym czasem odpowiedzi programu na zapytanie - 5 sekund. Wyjątkiem są operacje wykonywane relatywnie rzadko,

jak np. generowanie rocznych raportów. Założenia projektowe obejmują bazę dla maksymalnie 1000 klientów i 6 mechaników.

4. Skalowalność - możliwość późniejszej rozbudowy aplikacji i bazy danych o kolejne funkcje bazująca na warstwowym modelu systemu. Baza danych oraz aplikacja powinny być zbudowane tak, aby możliwe było dołączenie modułów odpowiedzialnych za nowe funkcjonalnościtakie jak np. współpraca z innymi systemami zewnętrznymi, generowanie innych typów raportów.

### 2.3.1 Wykorzystywane technologie i narzędzia

Na potrzeby projektu wykorzystane zostanie narzędzie do zarządzania relacyjną bazą danych - MySQL. Strona internetowa i jej działanie będzie testowane na systemie operacyjnym Windows 10. Strona internetowa będzie napisana w Flask Python - mikro framework.

### 2.3.2 Wymagania dotyczące rozmiaru bazy danych

Analiza wymagań zakłada że dane są zbierane przez cały rok, następnie są archiwizowane. Tabela samochód i użytkownicy są archiwizowani po dwóch latach.

| Tabela | Maksymalna liczba rekordów | Średnia liczba rekordów | Ilość kolumn | Maksymalny rozmiar tabeli | Średni rozmiar tabeli |
|---|---|---|---|---|---|
| Uzytkownik | 1000 | 500 | 7 | 7000 | 3500 |
| Samochód | 1500 | 750 | 10 | 15000 | 7500 |
| Adres | 1200 | 500 | 5 | 6000 | 2500 |
| WizytaCzesci | 18000 | 15000 | 2 | 36000 | 30000 |
| Wizyta | 7000 | 4000 | 6 | 42000 | 24000 |
| Czesci | 14000 | 10000 | 5 | 70000 | 50000 |
| WizytaMechanik | 21000 | 15000 | 2 | 42000 | 30000 |
| Mechanik | 10 | 5 | 4 | 40 | 20 |

Rysunek 2: Wymagania liczbowe

### 2.3.3 Wymagania dotyczące bezpieczeństwa systemu

Wymagania dotyczące uprawnień urzytkowników.
Klient może:
- dodać, usunąć, przeglądać swoje dane
- dodać, usunąć rezerwację wizyty
- przeglądać status wizyty
Mechanik:
- zamawiać części
- obsługa klienta (dodanie, usunięcie)
- edycja wizyty i jej statusu

Administrator: (jak wyżej)
- modyfikacja wszystkich rekordów
- wykonywanie statystyk
- robienie backupu

Dodatkowo system ma zapewniać poufność i integralność przechowywanych danych w odpowiadających im zakresach.

## 2.4 Przyjęte założenia projektowe

Na potrzeby projektu zostały poczynione pewne założenia dotyczące rzeczywistości, bazy danych i aplikacji. Nie wszystkie opisywane mechnizmy i funkcjonalności będą w pełni zaimplementowane w aplikacji i połączone z bazą, gdyż nacisk jest kładziony na strukturę bazy danych. Poza tym założenia dotyczące prowadzonej działalności są następujące:
- zatrudnionych jest 4 mechaników
- jeden z mechaników jest administratorem systemu
- w warsztacie są dwa stanowiska
- praca w trybie ośmiogodzinnym
- wykonywane są podstawowe naprawy

# 3 Projekt systemu

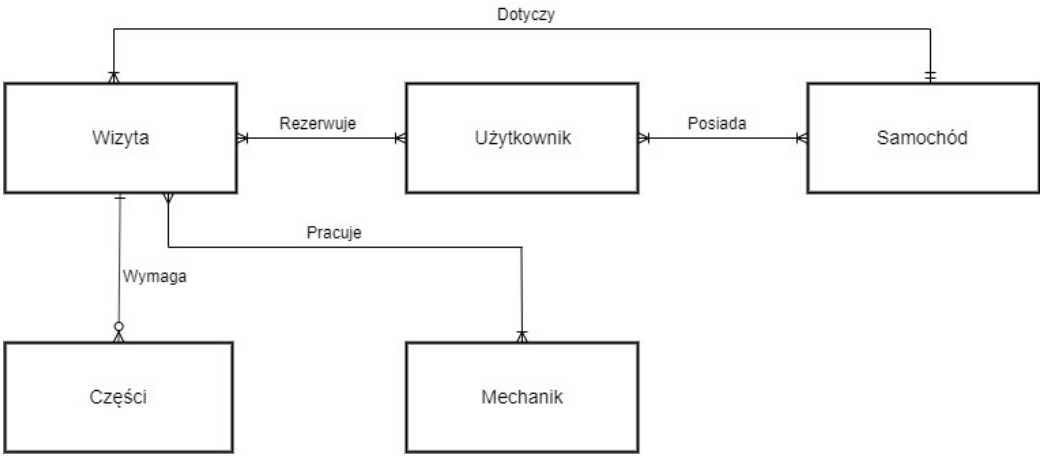
## 3.1 Projekt bazy danych

### 3.1.1 Analiza rzeczywistości i uproszczony model konceptualny

Warsztat samochodowy znajduje w dużym mieście (500 tys. ludzi). Klient może oczekiwać wykonania przeglądu, naprawy pojazdów, także jednośladów. Zakres obsługiwanych uszkodzeń to mechanika, hydraulika oraz elektryka. Dodatkowo istnieje możliwość wymiany oraz naprawy opon. W warsztacie pracuje jedna osoba do obsługi klienta i 4 mechanicy, którzy zazwyczaj pracują parami obsługując 2 boxy równocześnie.

Działalność w celu usprawnienia terminarza wizyt i obsługi klienta zostaje zmianieona do formy bazy danych wraz z obsługującą ją stroną internetową lub aplikacją. Zmiana ta przynosi korzyści zarówno dla warsztatu, ponieważ umożliwia wygodne i przejrzyste prowadzenie terminarza w formie elektronicznej, jak i dodaje nowe funkcje dla klienta. Do dokumentacji na-

leżą m.in. dane pojedynczej naprawy, zamówione części, dane pracowników, klientów i samochodów.(raporty miesięczne).




Rysunek 3: Diagram encji

**Klient**

Dla usprawnienia pracy warsztatu, klient powinien mieć dostęp do panelu klienta, w obrębie którego powinny zostać zawarte takie funkcje:

- Umówienie się na termin wizyty z samochodem
- Sprawdzenie stanu naprawy
- Wyświetlenie historii napraw samochodu (względem samochodu i klienta)
- Obsługa swojego konta (dane osobowe, hasło)

Aby klient mógł korzystać z funkcji wymienionych wyżej, musi spełnić pewne wymagania. Najważniejsze, klient musi mieć założone konto oraz przypisany samochód. Warunek ten zostanie spełniony w momencie pierwszej wizyty w warsztacie lub rejestracji online. Kolejny warunek, w przypadku rezerwacji terminu wizyty, dany termin musi być wolny. Należy również przewidzieć obsługę sytuacji, w których pracownik lub pracownicy są na zwolnieniu (nieobecność nieplanowana) lub urlopie (nieobecność planowana). W pierwszym przypadku należy zaznaczyć taki przypadek w systemie oraz, w razie potrzeby, wysłać do klienta powiadomienie o zmianie terminu lub statusu wizyty z powodu choroby pracownika i propozycji nowego terminu. Podobny proces postępowania powinien zajść w przypadku awarii sprzętu

niezbędnego do przeprowadzenia naprawy. W drugiej sytuacji (nieobecność planowana) system powinien zapobiec możliwości rezerwacji większej ilości wizyt w danym terminie, niż pozwalają na to zasoby warsztatu.

**Mechanik**

Kolejno po rezerwacji oraz potwierdzeniu terminu następuje etap naprawy. Przed rozpoczęciem naprawy mechanik powinien zweryfikować czy klient umówił się na wizytę poprzez panel kliencki (online), gdzie jego dane oraz dane samochodu automatycznie są przypisane do naprawy, czy też klient zarejestrował wizytę w inny sposób (np. telefonicznie). Mechanik musi podjąć działanie jedynie w drugim przypadku i musi ręcznie utworzyć wpis wizyty oraz przypisać do niej klienta oraz samochód. Zmiana etapów naprawy również jest w gestii mechanika zajmującego się daną naprawą, a jej składowymi są:

1. Rezerwacja,
2. Potwierdzenie,
3. W naprawie,
3.1 Opcjonalne: oczekiwanie na części,
4. Do odbioru,
5. Zakończone.

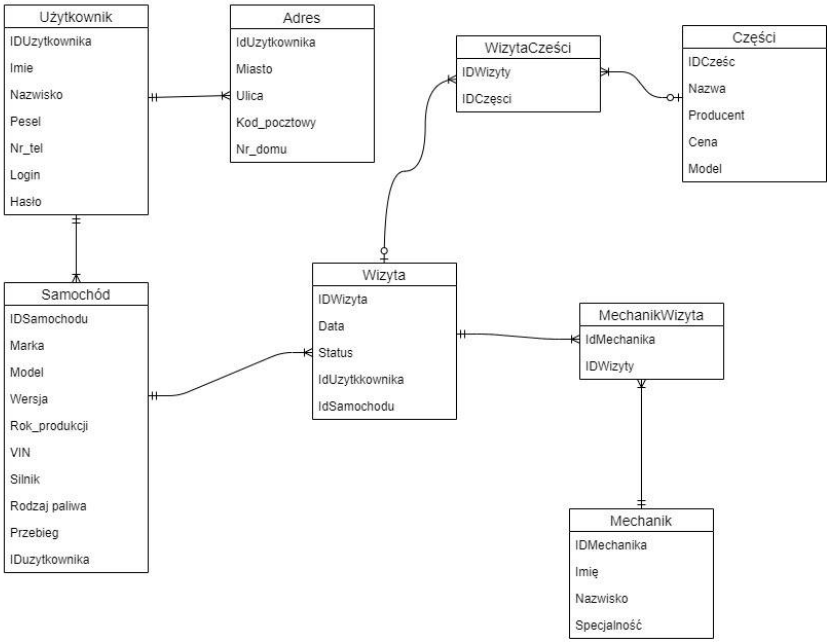
Aby można było zidentyfikować mechaników, którzy zajmowalisię samochodem przy danej naprawie, musi znaleźć się odpowiednie pole, gdzie można przypisać jednego lub kilku mechaników do naprawy. Do każdej odbytej wizyty należy dodać krótki opis elementów, które zostały poddane naprawie, bądź wymianie, np. „wymiana filtru oleju + wymiana klocków hamulcowych", „regulacja świateł".

**Administrator**

W celu ograniczenia funkcji dostępu dla mechaników, wprowadzony jest wyższy poziom uprawnień *mechanik-administrator*. Opisywana osoba może spełnia funkcje mechanika rozszerzone o dodatkowe uprawnienia jak modyfikacje danych mechaników i klienta. Do nich także należy m.in. robienie kopii zapasowych bazy, dodawanie kont mechaników, zmiana parametrów bazy danych oraz naprawianie błędów systemu.

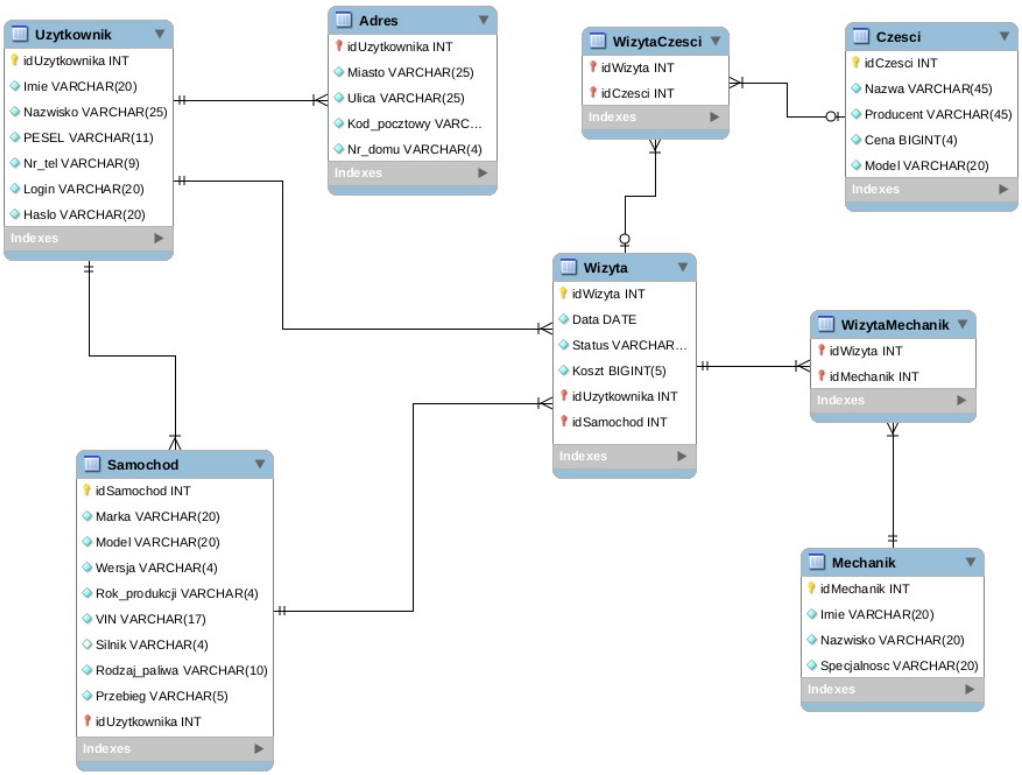
### 3.1.2 Model logiczny

Text




Rysunek 4: Model logiczny

### 3.1.3 Model fizyczny




Rysunek 5: Model fizyczny

## 3.2 Projekt aplikacji użytkownika

### 3.2.1 Architektura aplikacji i diagramy projektowe

Realizacja naszego projektu zakładała utworzenie strony internetowej warsztatu, która będzie służyła klientom oraz pracownikom warsztatu. Głównymi założeniami od strony klienta co do strony internetowej, było możliwość przeglądania historii napraw, rezerwacja terminu następnej wizyty, możliwość dodania do swojego profilu samochodu, oraz edycja danych osobowych.

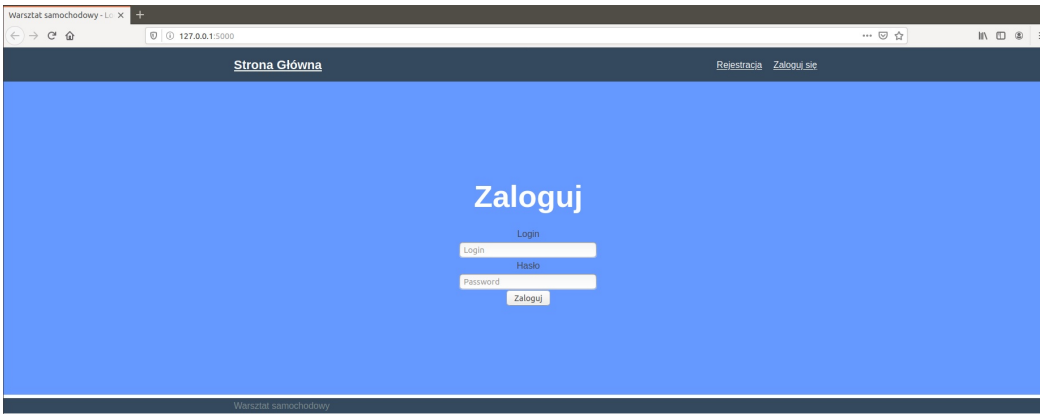
Panel pracownika warsztatu w założeniu dzieli się na dwie grupy, mechanik oraz administrator. Możliwości mechanika z poziomy strony internetowej to przeglądanie aktualnych napraw którymi zajmuję się mechanik, dodanie części do naprawy, modyfikowanie statusu naprawy.

Administrator w naszym przypadku jest równoważne z właścicielem warsztatu. Z poziomu strony, właściciel ma możliwość przeglądania wszystkich aktualnych oraz zakończonych napraw. Ma również możliwość przegląda-

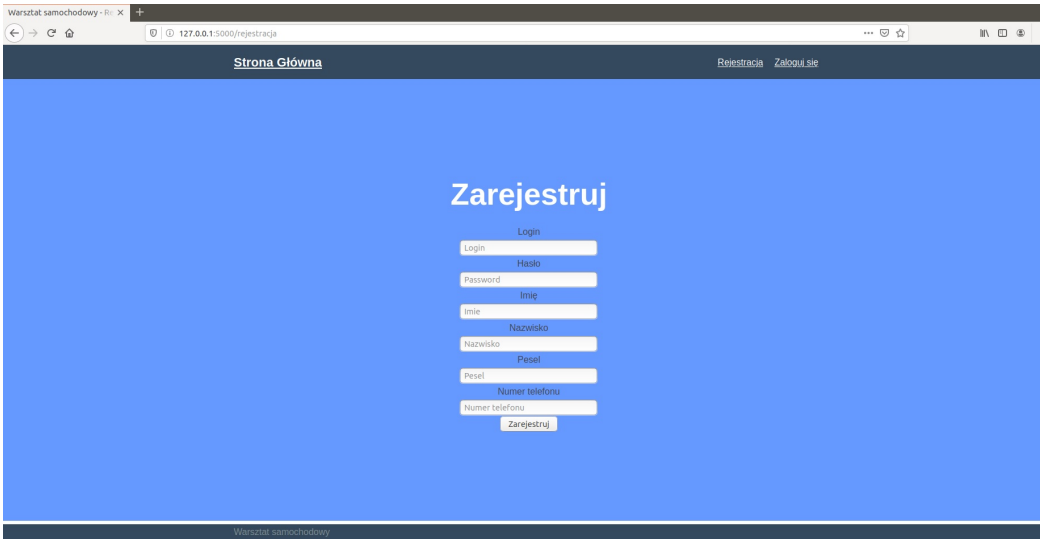
nia listy pracowników, zarejestrowanych klientów oraz przypisanych do nich samochodów. Panel administratora w założeniu miał usprawniać działanie warsztatu poprzez możliwość szybkiego dostępu do historii napraw.

### 3.2.2 Interfejs graficzny i struktura menu

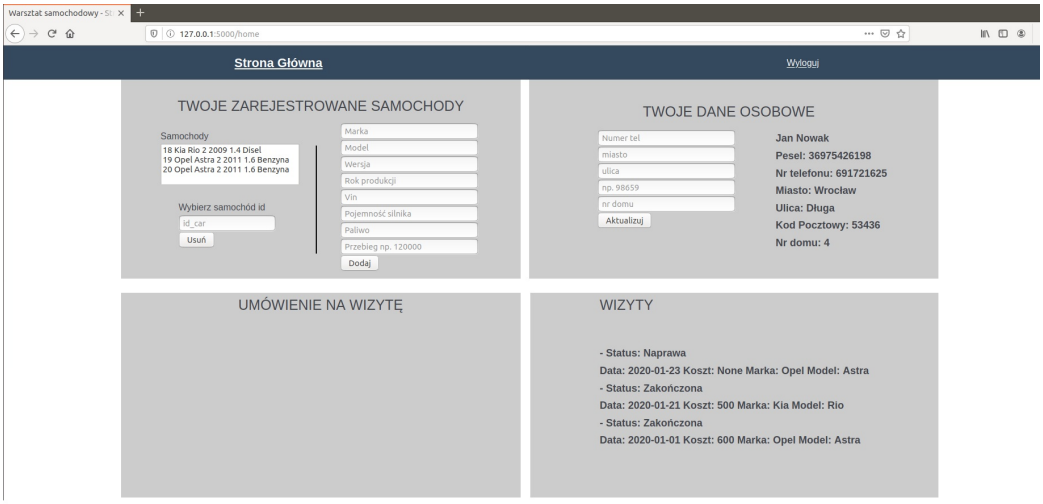
Stronę internetową podzieliliśmy na kilka sekcji: logowanie, rejestracja, panel klienta, panel administratora oraz panel mechanika. Na tym etapie realizacji projektu udało nam ukończyć sekcję logowania oraz rejestracji. Panel klienta jest mocno rozbudowany jednak nie zaimplementowaliśmy mechanizmu umówienia na wizytę co planujemy zrealizować w przyszłości. Panel administratora jest ukończony na poziomie pozwalającym zalogować się do niego oraz przejrzeć listę pracowników oraz klientów. Nie udało nam się na tym etapie zrealizować panelu mechanika.



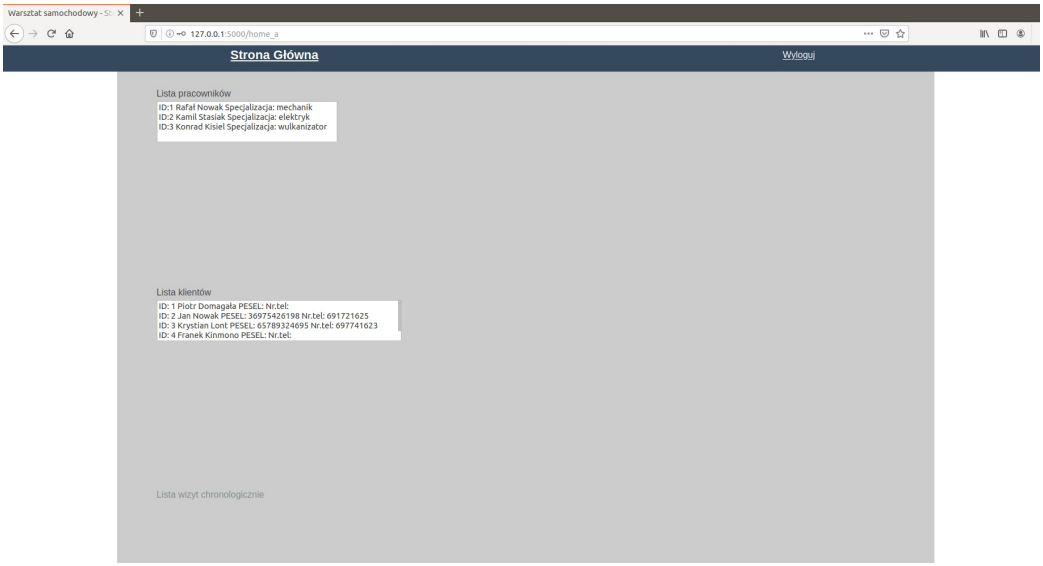

Rysunek 6: Strona logowania




Rysunek 7: Strona rejestracji




Rysunek 8: Panel klienta




Rysunek 9: Panel administratora (Niepełna implementacja)

### 3.2.3 Metoda podłączania do bazy danych – integracja z bazą danych

Tak jak już było wcześniej wspomniane w naszym projekcie skorzystaliśmy z bazy MySql. Bazę danych przechowujemy na serwerze lokalnym. Do połączenia naszej bazy z strona internetową wykorzystaliśmy framework Flask który opiera się na języku programowania Python. Integracja z baza odbyła się poprzez wykorzystanie wbudowanych modułów co sprawiło że połączenie naszej aplikacji z baza danych odbyło się z wykorzystaniem kilku linijek kodu. Od tego momentu mogliśmy tworzyć obiektu kursora który był odpowiedzialny za komunikację z baza. To za jego pomocą były wykonywane zapytania MySql w naszym kodzie.

### 3.2.4 Projekt zabezpieczeń na poziomie aplikacji

Na poziomie aplikacji wprowadziliśmy kilka prostych zabezpieczeń. Przede wszystkim, sprawdzanie czy dane odnośnie logowania są poprawne i czy znajdują się w bazie danych, w przypadku jeśli dane są błędne zostanie zwrócony komunikat o tym błędzie.

Kolejną rzeczą jest przechowywanie informacji o sesji. Jeśli klient poprawnie wpisze dane logowania, zostanie to odnotowane co później pozwoli przekierować na stronę klienta. Jest to zabezpieczenie przed dostaniem się do panelu klienta lub administratora bez podania danych logowania. Do panelu administratora może dostać się tylko użytkownik z odpowiednim id.

W naszym przypadku podczas logowania jest sprawdzana zmienna id która została pobrana z bazy. Jeśli id tego użytkownika jest równe 1, zostaje przekierowany do panelu administratora.

Oprócz tego zaimplementowaliśmy w naszym projekcie zabezpieczenia przed zarejestrowaniem użytkownika z loginem który widnieje już w bazie danych. Inne z zabezpieczeń to sprawdzenie czy wszystkie wymagane dane w formularzu rejestracji zostały podane. Rozwiązania programistyczne tych zabezpieczeń zostaną po krótce opisane w późniejszej części dokumentacji.

# 4 Implementacja systemu baz danych

Na potrzeby projektu skorzystaliśmy z bazy danych MySql. W celu ułatwienia pracy korzystaliśmy z narzędzia Mysql Workbench, które bardzo usprawnił prace z baza. W łaty sposób stworzyliśmy w tym narzędziu model fizyczny bazy, a następnie automatycznie program wygenerował dla nas skrypt i utworzył bazę. W dalszej części dokumentacji przedstawimy dokładnie sposób tworzenia bazy.

## 4.1 Tworzenie tabel i definiowanie ograniczeń

```
DROP SCHEMA IF EXISTS ‘mydb‘ ;
CREATE SCHEMA IF NOT EXISTS ‘mydb‘ DEFAULT CHARACTER SET utf8 ;
USE ‘mydb‘ ;
DROP TABLE IF EXISTS ‘mydb‘.‘Uzytkownik‘ ;
CREATE TABLE IF NOT EXISTS ‘mydb‘.‘Uzytkownik‘ (
‘idUzytkownika‘ INT NOT NULL AUTO_INCREMENT,
‘Imie‘ VARCHAR(20) NOT NULL,
‘Nazwisko‘ VARCHAR(25) NOT NULL,
‘PESEL‘ VARCHAR(11) NOT NULL,
‘Nr_tel‘ VARCHAR(9) NOT NULL,
‘Login‘ VARCHAR(20) NOT NULL,
‘Haslo‘ VARCHAR(20) NOT NULL,
PRIMARY KEY (‘idUzytkownika‘),
UNIQUE INDEX ‘idUzytkownika_UNIQUE‘ (‘idUzytkownika‘ ASC))
ENGINE = InnoDB;

DROP TABLE IF EXISTS ‘mydb‘.‘Adres‘ ;
CREATE TABLE IF NOT EXISTS ‘mydb‘.‘Adres‘ (
‘idUzytkownika‘ INT NOT NULL,
```

```
‘Miasto‘ VARCHAR(25) NOT NULL,
‘Ulica‘ VARCHAR(25) NOT NULL,
‘Kod_pocztowy‘ VARCHAR(5) NOT NULL,
‘Nr_domu‘ VARCHAR(4) NOT NULL,
INDEX ‘fk_Adres_Użytkownik_idx‘ (‘idUzytkownika‘ ASC),
PRIMARY KEY (‘idUzytkownika‘),
CONSTRAINT ‘fk_Adres_Użytkownik‘
FOREIGN KEY (‘idUzytkownika‘)
REFERENCES ‘mydb‘.‘Uzytkownik‘ (‘idUzytkownika‘)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;

DROP TABLE IF EXISTS ‘mydb‘.‘Samochod‘ ;
CREATE TABLE IF NOT EXISTS ‘mydb‘.‘Samochod‘ (
‘idSamochod‘ INT NOT NULL AUTO_INCREMENT,
‘Marka‘ VARCHAR(20) NOT NULL,
‘Model‘ VARCHAR(20) NOT NULL,
‘Wersja‘ VARCHAR(4) NOT NULL,
‘Rok_produkcji‘ VARCHAR(4) NOT NULL,
‘VIN‘ VARCHAR(17) NOT NULL,
‘Silnik‘ VARCHAR(4) NULL,
‘Rodzaj_paliwa‘ VARCHAR(10) NOT NULL,
‘Przebieg‘ VARCHAR(5) NOT NULL,
‘idUzytkownika‘ INT NOT NULL,
PRIMARY KEY (‘idSamochod‘, ‘idUzytkownika‘),
UNIQUE INDEX ‘idSamochod_UNIQUE‘ (‘idSamochod‘ ASC),
INDEX ‘fk_Samochod_Uzytkownik1_idx‘ (‘idUzytkownika‘ ASC),
CONSTRAINT ‘fk_Samochod_Uzytkownik1‘
FOREIGN KEY (‘idUzytkownika‘)
REFERENCES ‘mydb‘.‘Uzytkownik‘ (‘idUzytkownika‘)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

DROP TABLE IF EXISTS ‘mydb‘.‘Wizyta‘ ;
CREATE TABLE IF NOT EXISTS ‘mydb‘.‘Wizyta‘ (
‘idWizyta‘ INT NOT NULL AUTO_INCREMENT,
‘Data‘ DATE NOT NULL,
‘Status‘ VARCHAR(20) NOT NULL,
‘Koszt‘ BIGINT(5) UNSIGNED NOT NULL,
```

```
‘idUzytkownika‘ INT NOT NULL,
‘idSamochod‘ INT NOT NULL,
PRIMARY KEY (‘idWizyta‘, ‘idUzytkownika‘, ‘idSamochod‘),
UNIQUE INDEX ‘idWizyta_UNIQUE‘ (‘idWizyta‘ ASC),
INDEX ‘fk_Wizyta_Uzytkownik1_idx‘ (‘idUzytkownika‘ ASC),
INDEX ‘fk_Wizyta_Samochod1_idx‘ (‘idSamochod‘ ASC),
CONSTRAINT ‘fk_Wizyta_Uzytkownik1‘
FOREIGN KEY (‘idUzytkownika‘)
REFERENCES ‘mydb‘.‘Uzytkownik‘ (‘idUzytkownika‘)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT ‘fk_Wizyta_Samochod1‘
FOREIGN KEY (‘idSamochod‘)
REFERENCES ‘mydb‘.‘Samochod‘ (‘idSamochod‘)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;

DROP TABLE IF EXISTS ‘mydb‘.‘Czesci‘ ;
CREATE TABLE IF NOT EXISTS ‘mydb‘.‘Czesci‘ (
‘idCzesci‘ INT NOT NULL AUTO_INCREMENT,
‘Nazwa‘ VARCHAR(45) NOT NULL,
‘Producent‘ VARCHAR(45) NOT NULL,
‘Cena‘ BIGINT(4) NOT NULL,
‘Model‘ VARCHAR(20) NOT NULL,
‘Ilosc‘ INT NOT NULL,
PRIMARY KEY (‘idCzesci‘),
UNIQUE INDEX ‘idCzesci_UNIQUE‘ (‘idCzesci‘ ASC))
ENGINE = InnoDB;

DROP TABLE IF EXISTS ‘mydb‘.‘WizytaCzesci‘ ;
CREATE TABLE IF NOT EXISTS ‘mydb‘.‘WizytaCzesci‘ (
‘idWizyta‘ INT NOT NULL,
‘idCzesci‘ INT NOT NULL,
PRIMARY KEY (‘idWizyta‘, ‘idCzesci‘),
CONSTRAINT ‘fk_WizytaCzesci_Wizyta1‘
FOREIGN KEY (‘idWizyta‘)
REFERENCES ‘mydb‘.‘Wizyta‘ (‘idWizyta‘)
ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT ‘fk_WizytaCzesci_Cześci1‘
```

```
FOREIGN KEY (‘idCzesci‘)
REFERENCES ‘mydb‘.‘Czesci‘ (‘idCzesci‘)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;

DROP TABLE IF EXISTS ‘mydb‘.‘Mechanik‘ ;
CREATE TABLE IF NOT EXISTS ‘mydb‘.‘Mechanik‘ (
‘idMechanik‘ INT NOT NULL AUTO_INCREMENT,
‘Imie‘ VARCHAR(20) NOT NULL,
‘Nazwisko‘ VARCHAR(20) NOT NULL,
‘Specjalnosc‘ VARCHAR(20) NOT NULL,
PRIMARY KEY (‘idMechanik‘),
UNIQUE INDEX ‘idMechanik_UNIQUE‘ (‘idMechanik‘ ASC))
ENGINE = InnoDB;

DROP TABLE IF EXISTS ‘mydb‘.‘WizytaMechanik‘ ;
CREATE TABLE IF NOT EXISTS ‘mydb‘.‘WizytaMechanik‘ (
‘idWizyta‘ INT NOT NULL,
‘idMechanik‘ INT NOT NULL,
INDEX ‘fk_WizytaMechanik_Mechanik1_idx‘ (‘idMechanik‘ ASC),
PRIMARY KEY (‘idWizyta‘, ‘idMechanik‘),
CONSTRAINT ‘fk_WizytaMechanik_Wizyta1‘
FOREIGN KEY (‘idWizyta‘)
REFERENCES ‘mydb‘.‘Wizyta‘ (‘idWizyta‘)
ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT ‘fk_WizytaMechanik_Mechanik1‘
FOREIGN KEY (‘idMechanik‘)
REFERENCES ‘mydb‘.‘Mechanik‘ (‘idMechanik‘)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Dzięki wykorzystaniu MySql Workbench zaoszczędziliśmy czas na tworzenie tabel ręcznie. Program wygenerował nam powyższy skrypt który został wykorzystany do utworzenia bazy. Rozwiązania zabezpieczeń i ograniczeń jakie nałożyliśmy na atrybutu w tabelach można łatwo wyczytać z załączonego skryptu. Do tych najważniejszych rozwiązań można zaliczyć unikalny klucz główny, auto inkrementacja wartości id klucza która usprawnia doda-

wanie np. nowych pojazdów użytkownika, ustawienie wymaganych trybutów na NOT NULL, co powoduje konieczność uzupełnienia tego pola podczas dodawania rekordu do bazy.

## 4.2 Implementacja mechanizmów przetwarzania danych

W naszym projekcie wykorzystaliśmy mechanizmy które usprawniają prace z baza oraz wspomagają przetwarzanie i przeszukiwanie w bazie. Na potrzeby wyświetlania rekordów z różnych tabel utworzyliśmy 2 widoki. Pierwszy z nich to widok pozwalający na łączenie danych z tabeli użytkownicy oraz adres w ten sposób możemy łatwo wyświetlić dane każdego użytkownika na stronie np. w sekcji dane osobiste w panelu klienta.

```
CREATE
VIEW ‘uzytkonik_4‘ AS
SELECT
‘Uzytkownik‘.‘idUzytkownika‘ AS ‘idUzytkownika‘,
‘Uzytkownik‘.‘Imie‘ AS ‘Imie‘,
‘Uzytkownik‘.‘Nazwisko‘ AS ‘Nazwisko‘,
‘Uzytkownik‘.‘PESEL‘ AS ‘PESEL‘,
‘Uzytkownik‘.‘Nr_tel‘ AS ‘Nr_tel‘,
‘Adres‘.‘Miasto‘ AS ‘Miasto‘,
‘Adres‘.‘Ulica‘ AS ‘Ulica‘,
‘Adres‘.‘Kod_pocztowy‘ AS ‘Kod_pocztowy‘,
‘Adres‘.‘Nr_domu‘ AS ‘Nr_domu‘
FROM
(‘Uzytkownik‘
JOIN ‘Adres‘)
WHERE
(‘Uzytkownik‘.‘idUzytkownika‘ = ‘Adres‘.‘idUzytkownika‘)
```

Drugim widok jaki utworzyliśmy wykorzystujemy aby w łatwy sposób wyświetlać informacje na temat wizyty oraz samochodów przypisanych do tej wizyty.

```
CREATE
VIEW ‘widok_wiz‘ AS
SELECT
‘Wizyta‘.‘Status‘ AS ‘Status‘,
‘Wizyta‘.‘Data‘ AS ‘Data‘,
‘Wizyta‘.‘Koszt‘ AS ‘Koszt‘,
```

```
‘Samochod‘.‘Marka‘ AS ‘Marka‘,
‘Samochod‘.‘Model‘ AS ‘Model‘,
‘Wizyta‘.‘idUzytkownika‘ AS ‘idUzytkownika‘
FROM
((‘Uzytkownik‘
JOIN ‘Wizyta‘)
JOIN ‘Samochod‘)
WHERE
((‘Uzytkownik‘.‘idUzytkownika‘ = ‘Samochod‘.‘idUzytkownika‘)
AND (‘Samochod‘.‘idUzytkownika‘ = ‘Wizyta‘.‘idUzytkownika‘)
AND (‘Uzytkownik‘.‘idUzytkownika‘ = ‘Wizyta‘.‘idUzytkownika‘)
AND (‘Samochod‘.‘idSamochod‘ = ‘Wizyta‘.‘idSamochod‘))
```

Kolejnym elementem wykorzystanym do usprawnienia pracy bazy były indeksy. Indeksy automatycznie zostały utworzone w naszej bazie na pola należące do kluczy głównych, zastanawialiśmy się nad dodaniem własnych jednak ostatecznie uznaliśmy że indeksy na klucze główne nam wystarczą ponieważ każde zapytanie do bazy jakie wykonujemy na stronie internetowej jest za pośrednictwem klucza głównego czyli id. Przykład:

```
cursor.execute('SELECT * From Adres WHERE idUzytkownika = %s',(str(indeks)))s
```

Wykorzystaliśmy również sekwencje. Każdy z naszych kluczy głównych jest auto inkrementowany co usprawnia prowadzenie uporządkowanych indeksów.

## 4.3 Implementacja uprawnień i innych zabezpieczeń

Na poziomie bazy danych mamy jednego użytkownika który posiada wszystkie uprawnienia do wykonywania operacji na bazie. To właśnie przez tego użytkownika tworzymy połączenie bazy ze strona internetowa naszego warsztatu.

Jednak aby zabezpieczyć dane innych użytkowników strony internetowej znajdujące się w bazie danych, postanowiliśmy rozwiązać ten problem na poziomie kodu strony internetowej. Każde zapytanie wykonujemy w odniesieniu do id użytkownika, tzn. w każdym zapytaniu sql jest zawarta cześć gdzie wymuszamy odnalezienie lub dodanie jakiegoś rekordu według id np. id_użytownika, id_samochodu, id_wizyty. Dzięki temu mamy pewność ze dane które nie należą do zalogowanej osoby nie zostaną nielegalnie odczytane.

## 4.4 Testowanie bazy danych na przykładowych danych

W celu testowania działania bazy wygenerowaliśmy przykładowe dane wykorzystując darmowy generator na stronie www.generatedata.com. Oto krótki przykład otrzymanych wyników z generatora:

```
INSERT INTO ‘Uzytkownicy‘

(‘idUzytkownika‘,‘Imie‘,‘Nazwisko‘,‘PESEL‘,‘Nr_tel‘,‘Login‘,‘Haslo‘)
VALUES
(6,"Flynn","Cruz","1687092164099","830652592","placerat,","sit"),
(7,"Garrison","Simmons","1603070234999","772937700","scelerisque","Cras"),
(8,"May","Sharp","1629050229999","630265730","Curabitur","adipiscing"),
(9,"Olga","Ball","1678100235999","350986591","erat.","Suspendisse"),
(10,"Maggy","Cobb","1636101198799","426818942","elementum,","ullamcorper"),
(11,"Willa","Combs","1606070246599","103309597","magna","non,");
```

Po dodaniu rekordów testowaliśmy bazę pod kontem zwracanych wyników. Wykonaliśmy kilka zapytań Select w celu weryfikacji czy dostajemy prawidłowe wyniki. Podsumowując test, nie zauważyliśmy żadnych problemów związanych z działaniem bezy po wprowadzeniu wielu rekordów.

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| # | idUzytkownika | Imie | Nazwisko | PESEL | Nr_tel | Login | Haslo |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Piotr | Domagała | | | piterek2707 | Anglia2707 |
| 2 | 2 | Jan | Nowak | 36975426198 | 691721625 | alicja | alicja |
| 3 | 3 | Krystian | Lont | 65789324695 | 697741623 | krystian | krystian |
| 4 | 4 | Franek | Kinmono | | | franek | franek |
| 5 | 5 | Krzysztof | Nowak | 76591254098 | 675497356 | krzysztof | krzysztof |
| 6 | 6 | Flynn | Cruz | 1687092164099 | 830652592 | placerat, | sit |
| 7 | 7 | Garrison | Simmons | 1603070234999 | 772937700 | scelerisque | Cras |
| 8 | 8 | May | Sharp | 1629050229999 | 630265730 | Curabitur | adipiscing |
| 9 | 9 | Olga | Ball | 1678100235999 | 350986591 | erat. | Suspendisse |
| 10 | 10 | Maggy | Cobb | 1636101198799 | 426818942 | elementum, | ullamcorper |
| 11 | 11 | Willa | Combs | 1606070246599 | 103309597 | magna | non, |
| 12 | 12 | Maxwell | Roach | 1662041615599 | 762238277 | magna. | malesuada |
| 13 | 13 | Dylan | Nixon | 1677062239799 | 195475312 | sit | consectetuer, |
| 14 | 14 | Caleb | Greer | 1600100210199 | 692986429 | lacus. | ultricies |
| 15 | 15 | Charissa | William | 1631110501999 | 430648603 | Aliquam | non, |
| 16 | 16 | Lillian | Browning | 1621110961599 | 809679350 | laoreet | Cum |
| 17 | 17 | Kaden | Best | 1628051917199 | 796705174 | Fusce | mus. |
| 18 | 18 | Hedda | Riggs | 1670090827199 | 455318142 | Quisque | Sed |
| 19 | 19 | Patricia | Bullock | 1618051394499 | 366852568 | enim, | nulla. |
| 20 | 20 | Reed | Kerr | 1693120287299 | 372753595 | ante | non, |
| 21 | 21 | Grady | Gonzalez | 1657012308099 | 405724427 | dictum | morbi |
| 22 | 22 | Kelly | Bradford | 1662041473799 | 840514748 | porttitor | imperdiet |
| 23 | 23 | Rachel | Walsh | 1602121864299 | 702702663 | velit | vitae |

Rysunek 10: Fragment wyniku otrzymango po wydaniu polecenia (Select * From Uzytkownicy;)

# 5 Implementacja i testy aplikacji

## 5.1 Instalacja i konfigurowanie systemu

Do prawidłowego testowania oraz działania naszej strony potrzebny jest serwer który będzie nasłuchiwał zapytań skierowanych ze strony internetowej oraz będzie odpowiadał użytkownikowi. W naszym projekcie wykorzystaliśmy z frameworku Flask który jest micro frameworkiem pythona do tworzenia aplikacji webowych. Sama instalacja Flaska zamyka się do wydania polecenia pip install flask (w systemach Linux) jest to instalacja poprzez menadżera pakietów pythona. Od tej pory możemy tworzyć pliki programu i uruchamiać je. Pliki Flaska są centralną częścią naszej strony internetowej, to tutaj mamy zdefiniowany routing oraz tutaj przetwarzamy zapytania, sprawdzamy poprawność danych z formularzy itd. Oprócz plików pythonowych, strona została wykonana z wykorzystaniem HTML który odpowiada za prezentacje

strony na serwerze, oraz CSS który całość oprawia w przyjemniejszy do wyświetlenia sposób. Wszystkie dane przechowujemy natomiast w Mysql-owej bazie danych. Jest to skrócona prezentacja układ całego projektu tzn. baza danych, pliki Flaska, pliki HTML oraz CSS.

## 5.2 Instrukcja użytkowania aplikacji

Użytkownik chcący skorzystać z usług warsztatu samochodowego, powinien wykonać następujące kroki:

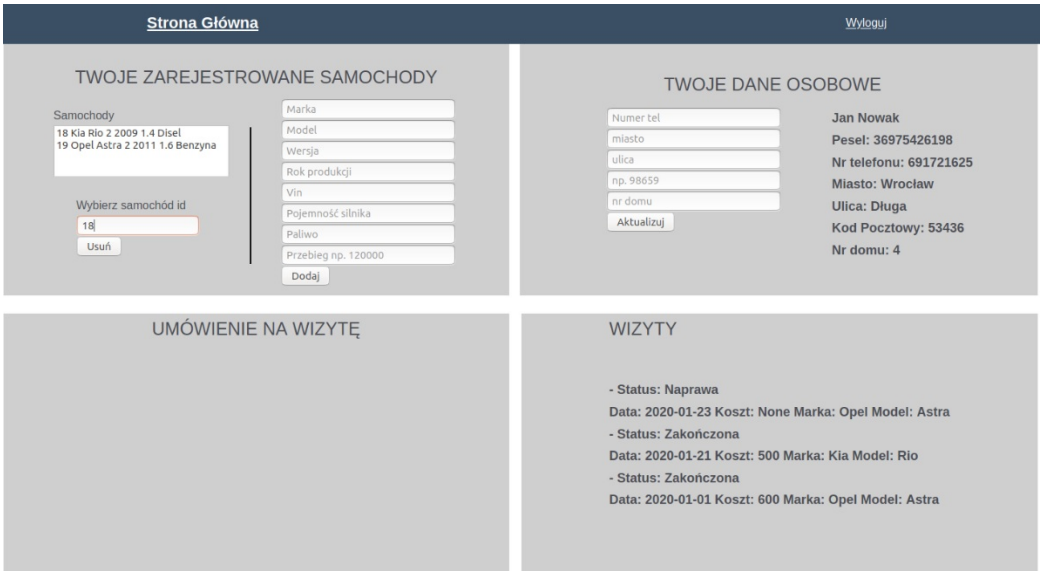
1. Zarejestrować konto na stronie internetowej warsztatu samochodowego, uzupełniając wszystkie niezbędne dane,
2. Zalogować się na swój profil i dodać do zakładki „Twoje samochody", pojazd który będzie podlegać naprawie,
3. W sekcji „Umawianie na wizytę", powinien umówić się na termin oddania samochodu do naprawy,
4. Użytkownik za pośrednictwem swojego konta na stronie internetowej może śledzić status naprawy samochodu w sekcji „Wizyty", w momencie gdy status będzie zakończony, można zgłosić się po odbiór samochodu.

## 5.3 Testowanie opracowanych funkcji systemu

Według założeń projektu z poziomu strony internetowej możliwe jest przeprowadzenia takich operacji jak:

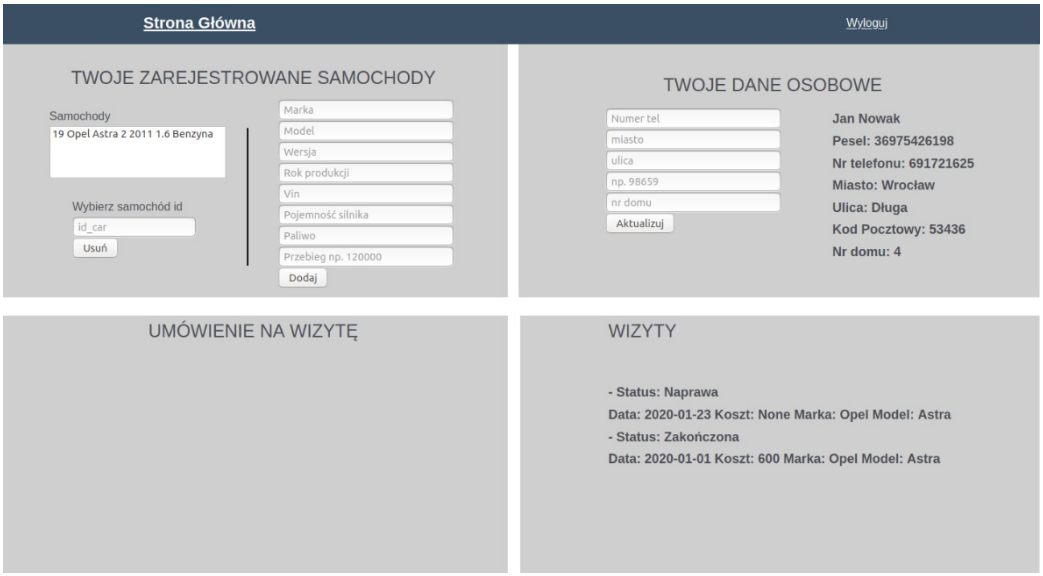
- Klient może sprawdzać status naprawy samochodu,
- Klient może przeglądać historię napraw,
- Klient może dodawać samochodu do swojego panelu na stronie,
- Klient może usuwać zarejestrowane samochody z swojego panelu.

Oraz kilka innych. W ramach testów czy zaimplementowane funkcjonalności działają poprawnie i czy wywołują pożądany efekt w rekordach bazy danych. W ramach prezentacji prześledzimy teraz proces usunięcia pojazdu z listy pojazdów.




Rysunek 11: Panel klienta

Użytkownik Jan Nowak posiada zarejestrowane 2 samochody. Każdy z nich jest reprezentowany poprzez id znajdujący się jako pierwsza wartość w sekcji samochody. W celu usunięcia samochodu z panelu należy wpisać id do pola „Wybierz samochód id", a następie „Usuń".




Rysunek 12: Panel klienta

W konsekwencji samochód został usunięty z bazy danych co jest

widoczne w skecji „Samochody” na stronie, która pobiera wartości z tabeli bazy danych.

## 5.4 Omówienie wybranych rozwiązań programistycznych

### 5.4.1 Implementacja interfejsu dostępu do bazy danych

W naszym projekcie wykorzystaliśmy dostęp do bazy za pośrednictwem mysql workbench co znacznie usprawniło tworzenie bazy oraz później praca nad nią. Integracja bazy z nasza stroną internetową odbywa się poprzez wprowadzenie przypisanie do odpowiednich zmiennych danych użytkownika bazy. U nas w kodzie odbywa się to w następujący sposób:

```
from flask import Flask , render_template , request , redirect , session, url_for
from flask_mysqldb import MySQL
import MySQLdb.cursors


app = Flask(__name__)


app.secret_key = 'projekt_baza'
mysql = MySQL(app)
app.config['MYSQL_USER'] = 'test'
app.config['MYSQL_PASSWORD'] = 'test'
app.config['MYSQL_DB'] = 'mydb'
app.config['MYSQL_HOST'] = 'localhost'
```

Rysunek 13: Fragment kody przedstawiający import modułu potrzebengo do połączenia z baza oraz komendy wymagane do poprawnego połączenie.

Jest to proste rozwiązanie, przykładowe dane logowania jakie tutaj wykorzystujemy oczywiście w dalszej części rozbudowy projektu zostaną zmienione np. hasło będzie odpowiednio zabezpieczone.

### 5.4.2 Implementacja mechanizmów bezpieczeństwa

Na poziomie aplikacji wykorzystaliśmy kilka prostych rozwiązań zapewniających bezpieczeństwo. Przede wszystkim zablokowaliśmy dostęp do panelu klienta poprzez ULR, w taki sposób że podczas przekierowania na stronę panelu klienta jest sprawdzana zmienna odnośnie sesji, jeśli zawiera ona wartość False, zostaniemy przekierowani na stronę logowania. Dopiero po poprawnym zalogowaniu zmienna przechowująca informacje na temat sesji

użytkownika zmienia wartość na True. Dzięki temu mamy pewność że nikt nie uzyska dostępu do panelu klienta bez poprawnego zalogowania.

Również sprawdzenie czy dany login i hasło są poprawne jest wykonywania na poziomie kodu strony internetowej.

```
    cursor.execute('SELECT idUzytkownika FROM Uzytkownik WHERE Login = %s AND Haslo = %s', (login,password))
    account = cursor.fetchone()

    if account[0]==1:
        session['loggedin'] = True
        session['id'] = account[0]
        return redirect(url_for('admin'))
    elif account:
        session['loggedin'] = True
        session['id'] = account[0]
        return redirect(url_for('home'))
    else:
        msg = 'Bledne dane, nie ma takiego konta'

return render_template('index.html' , msg=msg)
```

Rysunek 14: Fragment kodu z funkcji logowania który zawiera zmienną odnośnie sesji która zabezpiecza przed niedozwolonym wejsciem na strone bez logowania oraz sprawdza czy isnieje użytkownik z podanymi danymi w bazie.

Kolejnym mechanizmem bezpieczeństwa wykorzystanym w naszej stronie internetowej jest zabezpieczenie przed rejestracją dwa razy tego samego loginu użytkownika. Rozwiązaliśmy to w taki sposób że gdy użytkownik rejestruje się do naszej strony internetowej, w momencie gdy wpisze wszystkie dane i prześle formularz, pobieramy wartość z pola login i w pierwszej kolejności sprawdzamy czy w bazie istnieje już taki wpis z identycznym loginem. Jeżeli tak zostanie zwrócony odpowiedni komunikat błędy. Jeśli nie, proces rejestracji przeszedł pomyślnie oczywiście gdy wszystkie wymagane pola będą uzupełnione.

```
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute("SELECT * FROM Uzytkownik WHERE login = %s" , [login])
    account = cursor.fetchone()

    if account:
        msg = 'Login jest juz zajęty'
    elif not login or not password or not imie or not nazwisko or not pesel or not nr_tel:
        msg = 'Nie podano wszytskich danych'
    else:
        cursor.execute("INSERT INTO Uzytkownik (Imie,Nazwisko,PESEL,Nr_tel,Login,Haslo) VALUES (%s,%s,%s,%s,%s,%s)"
        mysql.connection.commit()
        msg = 'Rejestracja zakonczona powodzeniem'

elif request.method == 'POST':
    msg = "Prosze wprowadzic wszystkie dane!"

return render_template('rejestracja.html' ,msg=msg)
```

Rysunek 15: Fragment kodu z funkcji rejestracji który sprawdza czy login jest już zajęty oraz czy wszystkie wymagane dane są wpisane.

# 6 Podsumowanie i wnioski

W projekcie udało się stworzyć stronę testową, bazę dancyh ze wszystkimi działającymi mechanizmami i następnie połączyć ją. Część wymagań funkcjonalnych i niefunkcjonalnych udało się w drożyć, lecz implementacja nie była pełna np. brak implementacji tworzenia kopi zapasowych, czy mechanizmów bezpieczeństwa. Największe trudności sprawiało brak wiedzy o optymalizacji i wydajności bazy danych. Praca nad projektem była prowadzona we własnym zakresie.

# 7 Literatura

https://docs.python.org/3/
http://flask.palletsprojects.com/en/1.1.x/
MySQL. Darmowa baza danych. Ćwiczenia praktyczne. Marcin Lis

# Spis rysunków