

## PROJEKT Z BAZ DANYCH

### Baza danych firmy badającej warunki środowiskowe

---

*Skład grupy:*

Krzysztof KURNIK, 237603

Jonatan BURY, 235471

Maciej MARUSZAK, 235437

*Termin:* śr 13:15-15:15

*Prowadzący:*

dr inż. Roman PTAK W4/K9

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>2</b>
1.1	Cel projektu . . . . .	2
1.2	Zakres projektu . . . . .	2
<b>2</b>	<b>Analiza wymagań</b>	<b>2</b>
2.1	Opis działania i schemat logiczny systemu . . . . .	2
2.2	Wymagania funkcjonalne . . . . .	2
2.3	Wymagania niefunkcjonalne . . . . .	3
2.3.1	Wykorzystywane technologie i narzędzia . . . . .	3
2.3.2	Wymagania dotyczące rozmiaru bazy danych . . . . .	3
2.3.3	Wymagania dotyczące bezpieczeństwa systemu . . . . .	3
2.4	Przyjęte założenia projektowe . . . . .	3
<b>3</b>	<b>Projekt systemu</b>	<b>4</b>
3.1	Projekt bazy danych . . . . .	4
3.1.1	Model fizyczny i ograniczenia integralności danych . . . . .	4
3.1.2	Inne elementy schematu – mechanizmy przetwarzania danych . . . . .	6
3.1.3	Widoki . . . . .	6
3.1.4	Procedury składowe . . . . .	6
3.1.5	Trigery . . . . .	6
3.1.6	Projekt mechanizmów bezpieczeństwa na poziomie bazy danych . . . . .	7
3.2	Projekt aplikacji użytkownika . . . . .	7
3.2.1	Architektura aplikacji i diagramy projektowe . . . . .	7
3.2.2	Interfejs graficzny i struktura menu . . . . .	10
3.2.3	Metoda podłączania do bazy danych – integracja z bazą danych . . . . .	12
3.2.4	Projekt zabezpieczeń na poziomie aplikacji . . . . .	12
<b>4</b>	<b>Implementacja systemu baz danych</b>	<b>13</b>
4.1	Tworzenie tabel i definiowanie ograniczeń . . . . .	13
4.2	Implementacja mechanizmów przetwarzania danych . . . . .	13
4.2.1	Indeksy . . . . .	13
4.2.2	Procedury składowe . . . . .	14
4.2.3	Trigger . . . . .	15
4.2.4	Widoki . . . . .	15
4.3	Implementacja uprawnień i innych zabezpieczeń . . . . .	16
4.4	Testowanie bazy danych na przykładowych danych . . . . .	17
<b>5</b>	<b>Implementacja i testy aplikacji</b>	<b>20</b>
5.1	Instalacja i konfigurowanie systemu . . . . .	20
5.2	Instrukcja użytkownika aplikacji . . . . .	21
5.2.1	Uruchamianie aplikacji . . . . .	21
5.2.2	Rejestracja . . . . .	21
5.2.3	Logowanie . . . . .	21
5.2.4	Konto użytkownika . . . . .	22
5.2.5	Wykresy pomiarów . . . . .	23
5.3	Testowanie opracowanych funkcji systemu . . . . .	23
5.3.1	• . . . . .	23
5.3.2	Próba logowania bez wprowadzania loginu i hasła . . . . .	24
5.3.3	Próba logowania z niepoprawnym hasłem . . . . .	24
5.4	Omówienie wybranych rozwiązań programistycznych . . . . .	24
5.4.1	Implementacja interfejsu dostępu do bazy danych . . . . .	24
5.4.2	Implementacja wybranych funkcjonalności systemu . . . . .	26
5.4.3	Implementacja mechanizmów bezpieczeństwa . . . . .	28
<b>6</b>	<b>Podsumowanie i wnioski</b>	<b>29</b>

# 1 Wstęp

## 1.1 Cel projektu

Stworzenie bazy danych firmy udostępniającej usługi w zakresie badania parametrów środowiskowych w miejscach pracy lub zamieszkania.

## 1.2 Zakres projektu

Firma oferująca usługi w postaci monitorowania miejsca pracy bądź miejscu zamieszkania. Firma posiada urządzenia wyposażone w czujniki. Zależnie od potrzeb klienta zostanie dobrane urządzenie, posiadające odpowiadającą mu ilość narzędzi pomiarowych. W momencie zgłoszenia zapotrzebowania, zespół monterów jest oddelegowany w wyznaczone przez zleceniodawcę miejsce, aby zamontować tam urządzenia pomiarowe. Cała procedura montażu jest po stronie przedsiębiorstwa, klient jedynie wskazuje miejsca w których chce badać warunki. Po umieszczeniu, skonfigurowaniu i podłączeniu pod sieć, takie urządzenie zaczyna działać i wysyłać okresowo pakiety danych pomiarowych do siedziby firmy. Dane są następnie analizowane i przetwarzane w ramach pełnionej przez firmę usługi. Klient otrzymuje informacje zwrotną o jakości warunków panujących w pomieszczeniu oraz sugestie, co należy przedsięwziąć, żeby te warunki poprawić. Usługi prowadzone przez firmę, mogą być długo lub krótkofalowe. Szczegóły dotyczące długości trwania umowy uzgadniane są z klientem podczas podpisywania umowy. Firma nie prowadzi sprzedaży urządzeń na własny użytek. Firma obsługuje klientów prywatnych i biznesowych. Podczas zawierania umowy klient jest zobligowany do podania danych osobowych, w tym miejsca zameldowania, bądź aktualnego zamieszkania.

# 2 Analiza wymagań

## 2.1 Opis działania i schemat logiczny systemu

Każdemu użytkownikowi zostanie udostępniona aplikacja webowa, w której będzie mógł sprawdzić aktualne wskazania zamontowanych u niego zestawu czujników. Przy każdym zleceniu zawarcia umowy, będzie określony czas świadczenia usług, który również będzie widoczny w aplikacji. Do aplikacji klient będzie potrzebował loginu i hasła. Login zostanie wygenerowany przez firmę, natomiast dobrane hasła pozostaje po stronie klienta, lecz będzie wymagało odpowiedniej składni. Posiadanie hasła jest obowiązkowe.

## 2.2 Wymagania funkcjonalne

- Klient
  - przeglądanie i zamawianie usług
  - przegląd danych pomiarowych z czujników
  - podgląd terminu upłynięcia umowy
- Firma
  - Właściciel:
    - Operacje „CRUD” klientów, pracowników oraz ich danych
    - Operacje „CRUD” urządzeń
    - Operacje „CRUD” pomiarów
  - Kierownik działu technicznego:
    - Operacje „CRUD” urządzeń
    - Przegląd pracowników działu technicznego
    - Operacje „CRUD” pomiarów
  - Administrator baz danych
    - Przegląd klientów oraz ich danych
    - Przegląd pomiarów

- Monter/technik
  - Przegląd dostępnych urządzeń
- Pracownik obsługi klienta
  - Przegląd danych klientów
  - Przegląd dostępnych urządzeń

## **2.3 Wymagania нефункционалне**

### **2.3.1 Wykorzystywane technologie i narzędzia**

- Baza danych MySQL
- Czujnik temperatury – SHT31
- Czujnik wilgotności – SHT31
- Czujnik natężenia światła – BH1750FV1
- Komunikacja TCP/IP

### **2.3.2 Wymagania dotyczące rozmiaru bazy danych**

- 1500 czujniki pomiarowe (około 1440 pomiarów na dobę)
- 350 klientów
- 20 pracowników
- Właściciel
- Kierownik działu technicznego
- Administrator – 6
- Monter/Technik – 9
- Obsługa klienta – 3

### **2.3.3 Wymagania dotyczące bezpieczeństwa systemu**

- Login i hasło – każdy pracownik, oraz klienci będą mieć utworzone prywatne konta, które umożliwią dostęp do udostępnionych im danych
- Poziomy dostępowe
  - Właściciel – dostęp do wszystkich zasobów firmy
  - Kierownik działu technicznego – dostęp do danych osobowych klientów i pracowników swojego działu
  - Administrator baz danych – dostęp do danych osobowych klientów, ale nie do danych pracowników
  - Monterzy – brak bezpośredniego dostępu do danych osobowych klientów i innych pracowników, fizyczny dostęp do urządzeń, ale nie do pomiarów
  - Klient – brak dostępu do danych firmy, dostęp do pomiarów z czujników

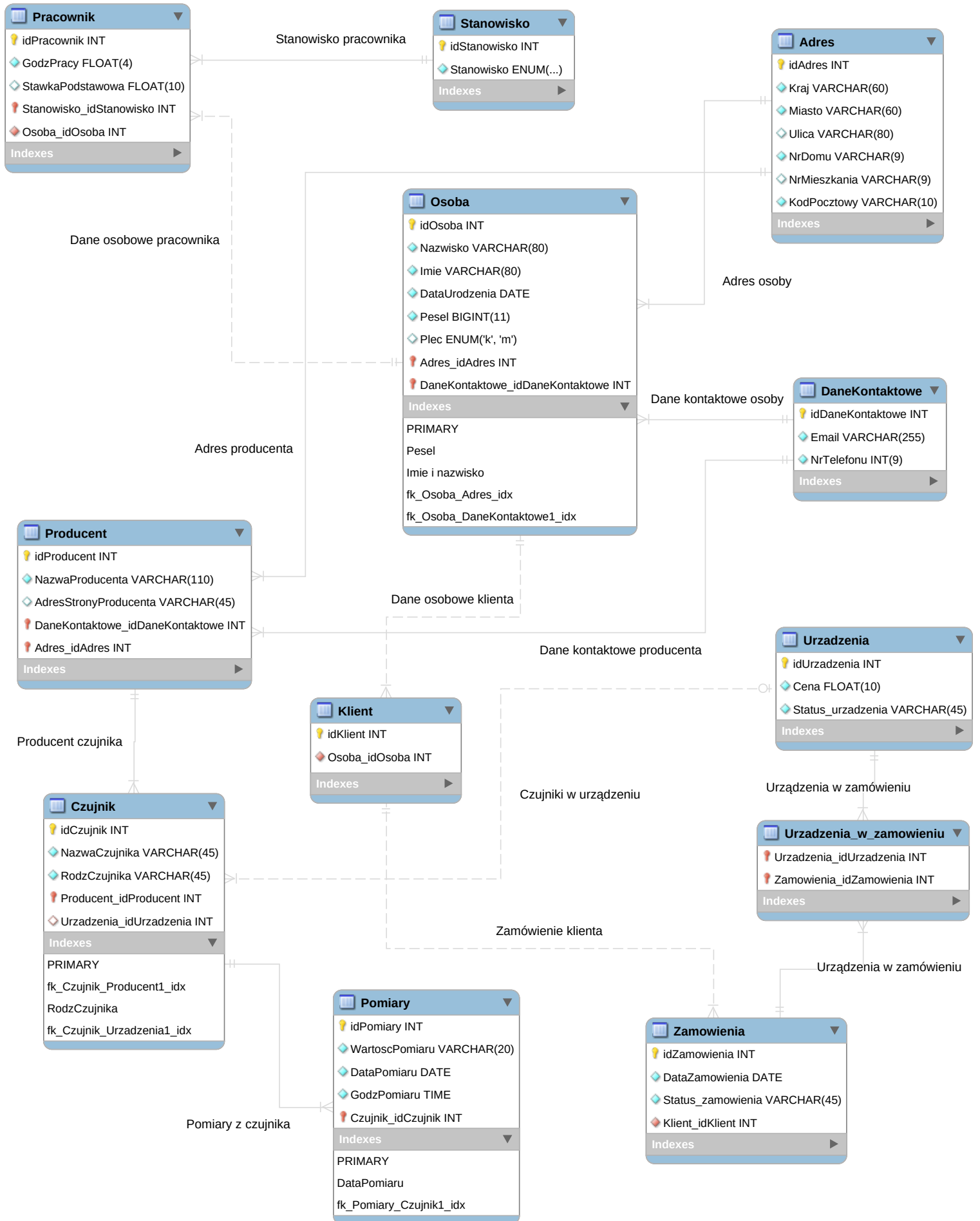
## **2.4 Przyjęte założenia projektowe**

- Zbieranie danych z czujników i wysyłanie ich na serwer
- Wyszukiwanie klientów w bazie danych poprzez podanie danych osobowych
- Zarządzanie klientami, zrywanie umowy w przypadku upłynięcia czasu wynajęcia urządzenia

## **3 Projekt systemu**

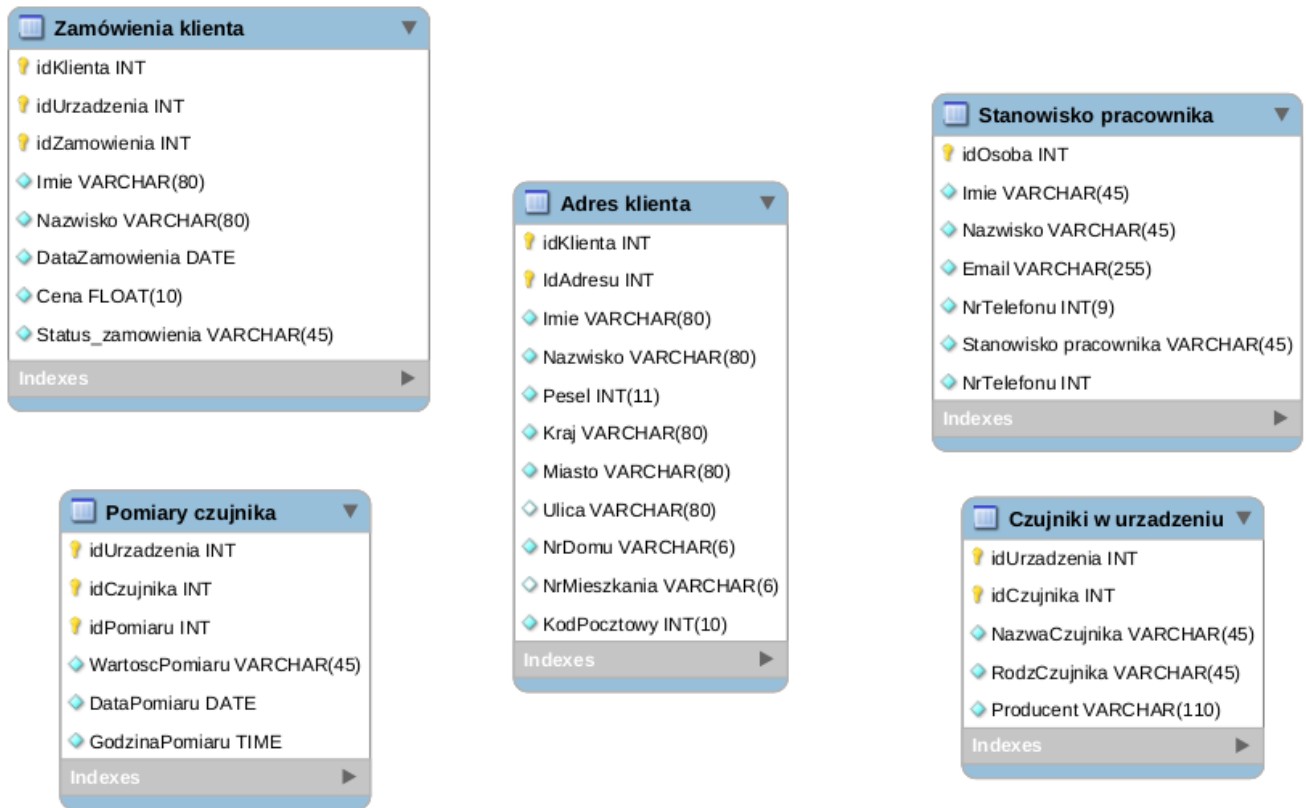
### **3.1 Projekt bazy danych**

#### **3.1.1 Model fizyczny i ograniczenia integralności danych**



### 3.1.2 Inne elementy schematu – mechanizmy przetwarzania danych

#### 3.1.3 Widoki



Rysunek 1: Widoki

#### 3.1.4 Procedury składowe

1. Format wpisywania danych personalnych klientów imię,nazwisko,pesel,data urodzenia,płeć Procedura będzie realizować wpisywanie poszczególnych wartości do odpowiednich tabel. Pierwsze 3 pola są wymagane, nie trzeba podawać daty urodzenia i płci. Jeżeli jednak zostaną wpisane, to procedura również te wartości umieści w odpowiednich tabelach.

#### 3.1.5 Trigery

1. Producent\_BEFORE\_INSERT

Listing 1: Producent\_BEFORE\_INSERT

```
1 CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`Producent_BEFORE_INSERT` BEFORE
  INSERT ON `Producent` FOR EACH ROW
2 BEGIN
3   SET NEW.NazwaProducenta = TRIM(NEW.NazwaProducenta);
4   SET NEW.AdresStronyProducenta = TRIM(NEW.AdresStronyProducenta);
5 END
```

2. Osoba\_AFTER\_UPDATE

Listing 2: Osoba\_AFTER\_UPDATE

```
1 CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`Osoba_AFTER_UPDATE` AFTER
  UPDATE ON `Osoba` FOR EACH ROW
2 BEGIN
3
4 END
```

Trigger używany będzie w przypadku zmiany danych osoby, czyli klienta albo pracownika. Przykładowo gdy pracownik zmieni miejsce zamieszkania.

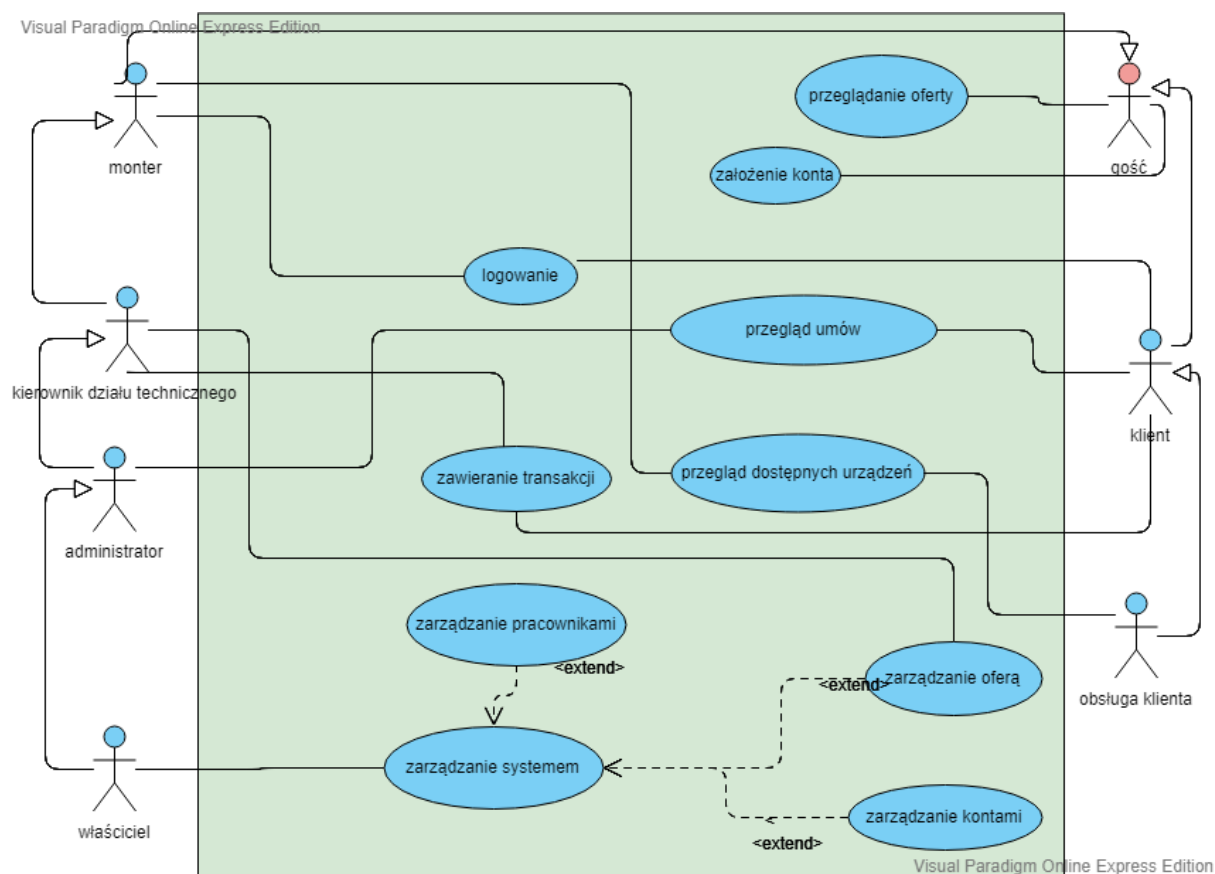
### 3.1.6 Projekt mechanizmów bezpieczeństwa na poziomie bazy danych

	Umowa z klientem	Swoje dane osobowe	Zakup urządzeń	Dane klientów	Oferowane urządzenia	Dane pracowników	Dostępne urządzenia	Dane pomiarów
Gość	-	-	-	-	R	-	-	-
Klient	A	W	R	-	R	-	-	R
Obsługa klienta	W	W	-	R	R	-	R	-
Monter	R	W	-	R	R	-	U	-
Kierownik działu technicznego	R	W	-	R	A	A	U	U
Administrator	U	W	-	U	A	-	A	R
Właściciel	U	W	-	U	A	A	A	A

Rysunek 2: Poziomy dostępowe

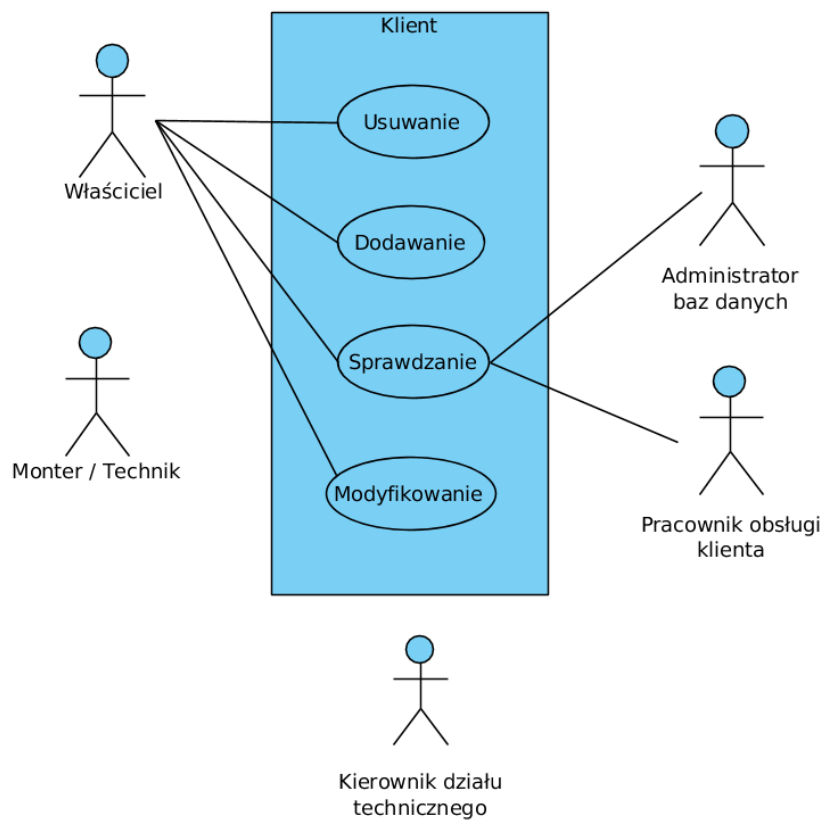
## 3.2 Projekt aplikacji użytkownika

### 3.2.1 Architektura aplikacji i diagramy projektowe

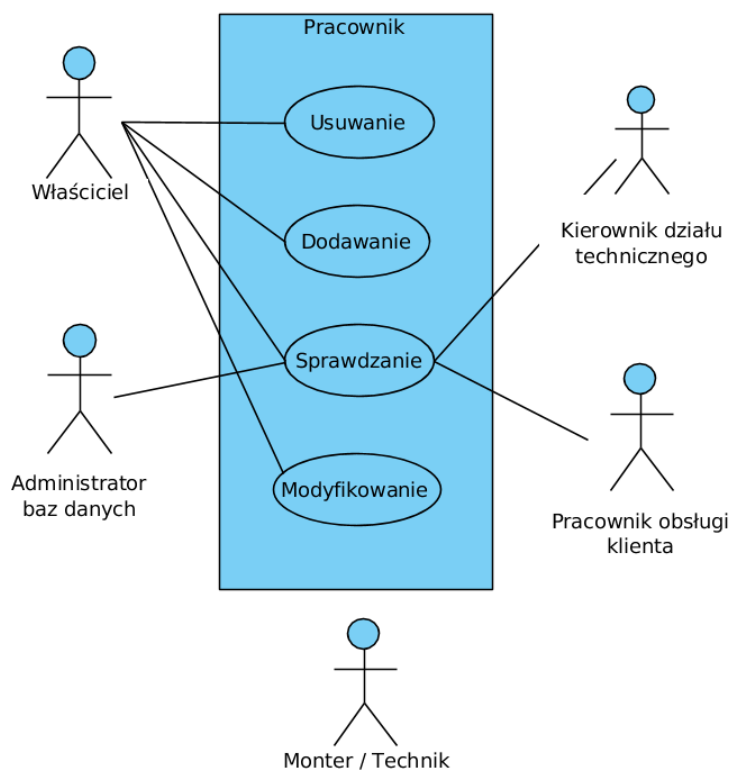


Rysunek 3: Ogólny diagram projektowy

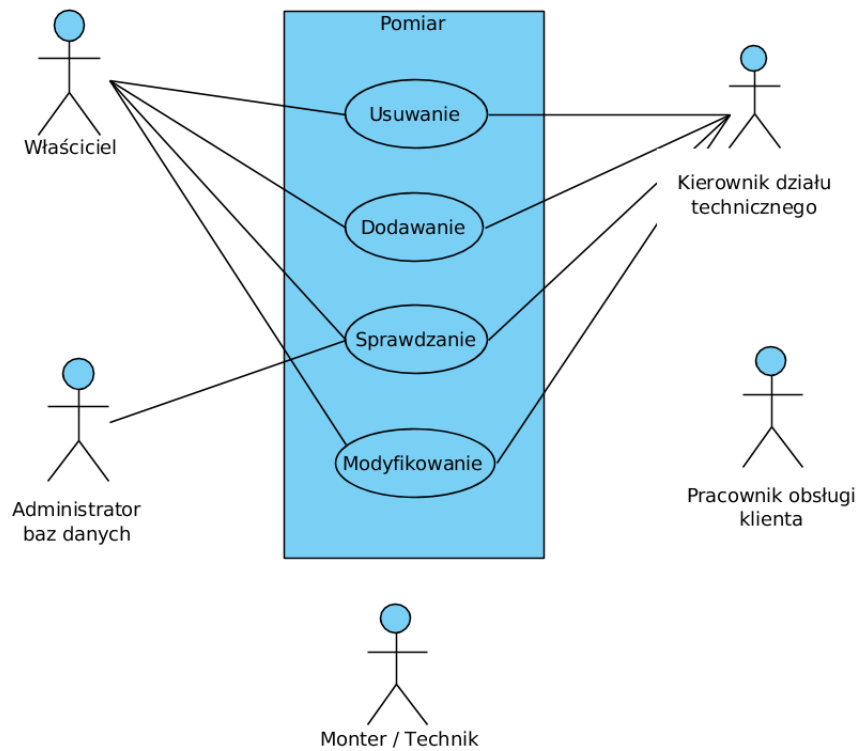




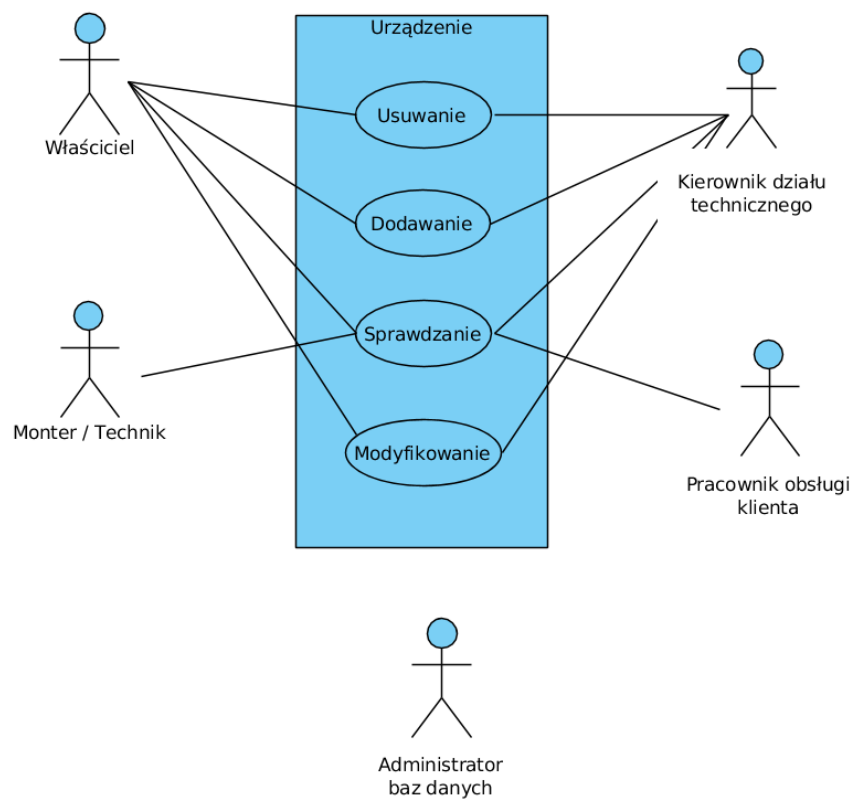
Rysunek 4: Klient



Rysunek 5: Pracownik



Rysunek 6: Pomiar



Rysunek 7: Urządzenie

### 3.2.2 Interfejs graficzny i struktura menu

strona  
główna

oferta

**nazwa produktu**  
krótki opis produktu

**cena**

strona  
logowania

logowanie

login

hasło

ewentualny komunikat błędu przy logowaniu

strona z  
danymi

konto

dane osobowe

podanie podczas rejestracji

strona z  
czujnikami  
klienta

Moje czynniki

lista posiadanych czujników

Rysunek 8: Interfejs graficzny aplikacji 1

strona  
główna po  
zalogowaniu

szukaj

nazwa urzytkownika

kontakt

wyloguj

oferta

nazwa produktu

krótki opis produktu

cena

strona z danymi o  
wybranym  
produkcie

szukaj

Utwórz konto

zaloguj się

Nazwa Produktu

cena

szczegółowy opis

zamów

strona  
rejestracji

szukaj

Utwórz konto

zaloguj się

kontakt

Rejestracja

email

imie

nawisko

data urodzenia

dzien

miesiąc

rok

płeć\*

☐ kobieta

☐ mężczyzna

kraj

mięscowosc

kod pocztowy

ulica\*

nr budynku

nr mieszkania\*

nr kontaktowy

haslo

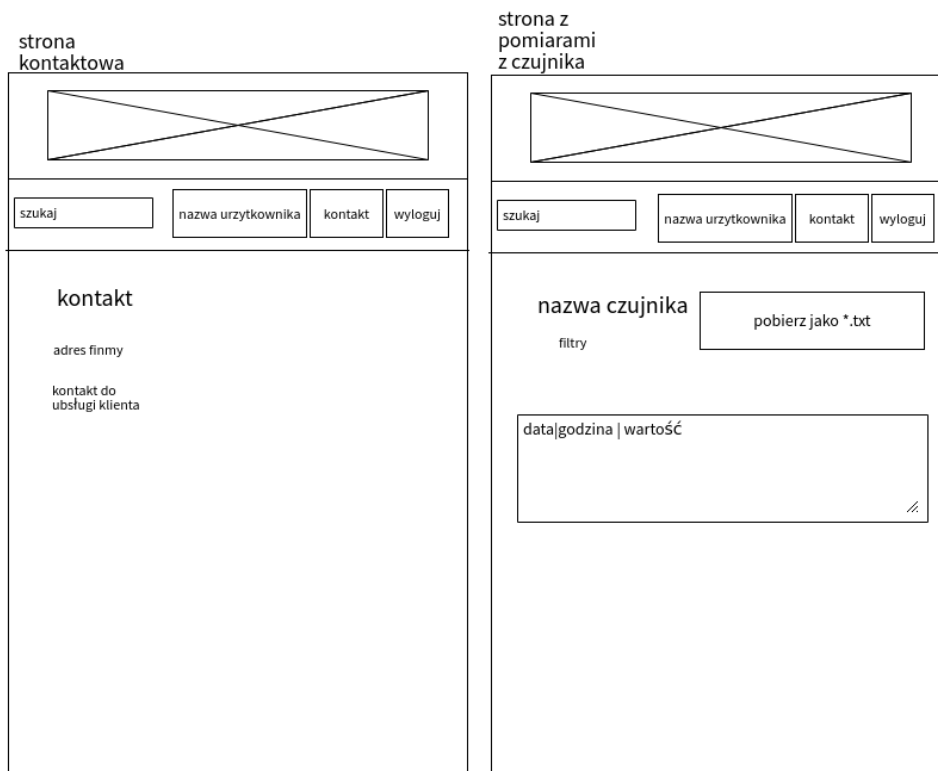
powtorz haslo

\* dane nieobowiazkowe

zarejestruj

ewentualny komunikat bledu przy logowaniu

Rysunek 9: Interfejs graficzny aplikacji 2



Rysunek 10: Interfejs graficzny aplikacji 3

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

Połączenie z aplikacji użytkownika zostanie zrealizowane przy pomocy języka PHP. Zostanie użyta biblioteka PDO(PHP Data Objects), które jest specjalnie stworzona do komunikacji z bazami danych. PDO zostało ustandaryzowane pod względem różnych systemów bazodanowych. Oznacza to, że pisząc kod łączący z bazą danych jednego systemu, w każdej chwili można zmienić na inny system, bez konieczności zmieniania kodu w PHP.

### 3.2.4 Projekt zabezpieczeń na poziomie aplikacji

- Hasło – wymagania:
  - Minimum 8 znaków, słownik składa się z dużych, małych liter, cyfr i znaków specjalnych
  - Co najmniej jedna duża i mała litera
  - Co najmniej jedna cyfra
  - Co najmniej jeden znak specjalny
- Wymuszenie logowania przez połączenie szyfrowane (HTTPS)
- Blokada konta po 3 nieudanych próbach, odblokowanie konta po konsultacji z działem obsługi klienta
- Wymagana zmiana hasła co 6 miesięcy
- Logowanie na konto administratora z wybranego adresu IP
- Logowanie na konto administratora wymaga podwójnej weryfikacji, kod z sms potwierdzającego wysłanego przez system

## 4 Implementacja systemu baz danych

### 4.1 Tworzenie tabel i definiowanie ograniczeń

CREATE SCHEMA rozpoczyna utworzenie bazy danych i podanej później nazwie. Sprawdzone zostaje również, czy nie ma bazy o takiej samej nazwie, jeżeli jest to zostanie zwrócony błąd o możliwości utraty danych i całej bazy o takiej nazwie.

Listing 3: Utworzenie bazy

```
1 CREATE SCHEMA IF NOT EXISTS `bazydanych` ;
2 SHOW WARNINGS;
3 USE `bazydanych` ;
```

Po utworzeniu bazy zostają dodawane kolejno poszczególne tabele poprzez komendę CREATE TABLE i tak jak w przypadku bazy jest sprawdzane, czy taka tabela nie została już stworzona. Następnie podawane są rekordy które mają znaleźć się w bazie, wraz z ich typem danych oraz właściwościami, przykładowo takimi jak automatyczna inkrementacja, czy wymuszenie, że musi zostać wpisana jakaś wartość. Na końcu tej sekcji tworzony jest klucz główny poprzez PRIMARY KEY.

Listing 4: Utworzenie tabeli Adres

```
1 CREATE TABLE IF NOT EXISTS `bazydanych`.`Adres` (
2   `idAdres` INT NOT NULL AUTO_INCREMENT,
3   `Kraj` VARCHAR(60) NOT NULL,
4   `Miasto` VARCHAR(60) NOT NULL,
5   `Ulica` VARCHAR(80) NULL,
6   `NrDomu` VARCHAR(9) NOT NULL,
7   `NrMieszkania` VARCHAR(9) NULL,
8   `KodPocztowy` VARCHAR(10) NOT NULL,
9   PRIMARY KEY (`idAdres`))
```

Dodawanie klucza obcego podczas tworzenia tabeli możliwe jest poprzez użycie komendy CONSTRAINT, w przeciwnym wypadku klucz obcy musiałby być dodany po pełnym utworzeniu tabeli. Poniżej podawane jest z którą tabelą ma być połączony oraz jego własności, czyli reakcje na określone czynności usuwania bądź aktualizowania.

Listing 5: Utworzenie klucza obcego

```
1 CONSTRAINT `fk_Osoba_Adres`
2   FOREIGN KEY (`Adres_idAdres`)
3   REFERENCES `bazydanych`.`Adres` (`idAdres`)
4   ON DELETE NO ACTION
5   ON UPDATE NO ACTION
```

Utworzenie tabeli zakończone jest określeniem mechanizmu składowania danych, w tym przypadku InnoDB, po czym określone zostają typy konwersji znaków.

Listing 6: Własności tworzonej tabeli

```
1 ENGINE = InnoDB
2 DEFAULT CHARACTER SET = utf8
3 COLLATE = utf8_polish_ci;
```

### 4.2 Implementacja mechanizmów przetwarzania danych

#### 4.2.1 Indeksy

Utworzenie unikatowego indeksu odbywa się używając polecenie UNIQUE INDEX, w tym przypadku dla komórki Pesel w tabeli Osoba. Oznacza to, że nie mogą istnieć dwa rekordy z takim samym peselem. Indeksy zwykle tworzone są jedynie poprzez słowo INDEX.

Listing 7: Definiowanie indeksów

```
1 UNIQUE INDEX `Pesel` (`Pesel` ASC)
2 INDEX `Imie i nazwisko` (`Imie` ASC, `Nazwisko` ASC, `idOsoba` ASC, `DataUrodzenia`
   ASC, `Pesel` ASC)
```

### 4.2.2 Procedury składowe

Procedura umożliwia dodanie klienta ze wszystkimi danymi, czyli adres zamieszkania, kontakt i dane osobowe. Komendą tworzącą procedurę jest wyrażenie CREATE PROCEDURE, po którym następuje podanie nazwy procedury. W nawiasach okrągłych zostały wpisane argumenty jakie będą przekazywane i wykorzystywane wewnątrz. Zmienne podawane są po przecinkach z poprzedzającym je znacznikiem IN a po nazwie wypisany jest typ zmiennej i ewentualna, maksymalna ilość znaków. Początkowo poniższa procedura dodaje kolejne wartości do tabeli Adres, po czym pobiera maksymalną wartość indeksu z tej tabeli, wykorzystując składnię: SELECT MAX(idAdres) FROM Adres. Wpisanie wartości do zmiennej ustawianej przy pomocy słowa SET zostanie wykorzystane podczas tworzenia kolejnej tabeli Osoba, gdzie jest wymagane podanie klucza obcego dla tabeli Adres.

Listing 8: Procedura dodaj\_klienta

```
1 DELIMITER //
2 CREATE PROCEDURE dodaj_klienta
3 (IN `Kraj` VARCHAR(60), IN `Miasto` VARCHAR(60), IN `Ulica` varchar(80), IN `
  NrDomu` varchar(9), IN `NrMieszkania` varchar(9), in `KodPocztowy` varchar(10),
  in `Email` varchar(255), in `NrTelefonu` integer(9), in `Nazwisko` varchar(80),
  in `Imie` varchar(80), in `DataUrodzenia` DATE, in `Pesel` BIGINT(11), in `
  Plec` enum('k', 'm'))
4 BEGIN
5   INSERT INTO `Adres` (`Kraj`, `Miasto`, `Ulica`, `NrDomu`, `NrMieszkania`, `
    KodPocztowy`) VALUES (`Kraj`, `Miasto`, `Ulica`, `NrDomu`, `NrMieszkania`, `
    KodPocztowy`);
6   SET @id_adres = (SELECT MAX(idAdres) FROM Adres);
7
8   INSERT INTO `DaneKontaktowe` (`Email`, `NrTelefonu`) VALUES (`Email`, `NrTelefonu
    `);
9
10  SET @id_dane = (SELECT MAX(idDaneKontaktowe) FROM DaneKontaktowe);
11
12  INSERT INTO `Osoba` (`Nazwisko`, `Imie`, `DataUrodzenia`, `Pesel`, `Plec`, `
    Adres_idAdres`, `DaneKontaktowe_idDaneKontaktowe`) VALUES (`Nazwisko`, `Imie
    `, `DataUrodzenia`, `Pesel`, `Plec`, @id_adres, @id_dane);
13
14  SET @id_osoba = (SELECT MAX(idOsoba) FROM Osoba);
15  INSERT INTO `Klient` (`Osoba_idOsoba`) VALUES (@id_osoba);
16 END //
17 DELIMITER ;
```

Przykładowe użycie procedury dodaj\_klienta:

Listing 9: Użycie dodaj\_klienta

```
1 CALL dodaj_klienta('Polska', 'Warszawa', 'Klonowa', '1', '3', '50-230', 'testujemy@o2.pl
  ', '111222333', 'Kowalski', 'Jan', '1990-12-11', '90121112345', 'm');
```

Procedura wprowadzająca dane nowego pracownika.

Listing 10: Procedura dodaj\_pracownika

```
1 DELIMITER //
2 CREATE PROCEDURE dodaj_pracownika
3 (IN `Kraj` VARCHAR(60), IN `Miasto` VARCHAR(60), IN `Ulica` varchar(80), IN `
  NrDomu` varchar(9), IN `NrMieszkania` varchar(9), in `KodPocztowy` varchar(10),
  in `Email` varchar(255), in `NrTelefonu` integer(9), in `Nazwisko` varchar(80),
  in `Imie` varchar(80), in `DataUrodzenia` DATE, in `Pesel` BIGINT(11), in `
  Plec` enum('k', 'm'), IN `Stanowisko` enum('Monter', 'Właściciel', 'Kierownik
  działu technicznego', 'Administrator', 'Obsługa klienta'), IN `GodzPracy` float
  (4), IN `StawkaPodstawowa` float(10))
4 BEGIN
5   INSERT INTO `Adres` (`Kraj`, `Miasto`, `Ulica`, `NrDomu`, `NrMieszkania`, `
    KodPocztowy`) VALUES (`Kraj`, `Miasto`, `Ulica`, `NrDomu`, `NrMieszkania`, `
    KodPocztowy`);
6   SET @id_adres = (SELECT MAX(idAdres) FROM Adres);
7
```

```

8  INSERT INTO `DaneKontaktowe` (`Email`,`NrTelefonu`) VALUES (`Email`,`NrTelefonu`
9  `);
10 SET @id_dane = (SELECT MAX(idDaneKontaktowe) FROM DaneKontaktowe);
11
12 INSERT INTO `Osoba` (`Nazwisko`,`Imie`,`DataUrodzenia`,`Pesel`,`Plec`,`
13   Adres_idAdres`,`DaneKontaktowe_idDaneKontaktowe`) VALUES (`Nazwisko`,`Imie`
14   `,`,`DataUrodzenia`,`Pesel`,`Plec`,`@id_adres,@id_dane);
15
16 SET @id_osoba = (SELECT MAX(idOsoba) FROM Osoba);
17
18 INSERT INTO `Stanowisko` (`Stanowisko`) VALUES (`Stanowisko`);
19
20 SET @id_stanowisko = (SELECT MAX(idStanowisko) FROM Stanowisko);
21
22 INSERT INTO `Pracownik` (`Osoba_idOsoba`,`GodzPracy`,`StawkaPodstawowa`,`
23   Stanowisko_idStanowisko`) VALUES (@id_osoba`,`GodzPracy`,`StawkaPodstawowa`,`
24   @id_stanowisko);
25 END //
26 DELIMITER ;

```

Przykładowe użycie procedury dodaj\_pracownika:

Listing 11: Użycie dodaj\_pracownika

```

1 CALL dodaj_pracownika('Polska','Warszawa','Karmelkowa','21','13','59-750','pwr@o2.
2   pl','999888000','Jagoda','Renata','1987-09-11','87091189765','k','Monter','8',
3   '1234');

```

### 4.2.3 Trigger

Triger usuwa wszelkie białe znaki podczas wprowadzania danych. Czyli jeżeli zostanie wprowadzona nazwa zawierająca kilka spacji przed i po " Polska "to triger je usunie i pozostawi jedynie "Polska".

Listing 12: Adres\_BEFORE\_INSERT

```

1 DELIMITER //
2 CREATE TRIGGER `Adres_BEFORE_INSERT` BEFORE
3 INSERT ON `Adres` FOR EACH ROW
4 BEGIN
5 SET NEW.Kraj = TRIM(NEW.Kraj);
6 SET NEW.Miasto = TRIM(NEW.Miasto);
7 SET NEW.Ulica = TRIM(NEW.Ulica);
8 SET NEW.NrDomu = TRIM(NEW.NrDomu);
9 SET NEW.NrMieszkania = TRIM(NEW.NrMieszkania);
10 SET NEW.KodPocztowy = TRIM(NEW.KodPocztowy);
11 END //
12 DELIMITER ;

```

### 4.2.4 Widoki

CREATE VIEW jest to polecenie tworzące widok, po którym podawana jest jego nazwa. W sekcji pomiędzy SELECT a FROM wypisane są komórki tabel, które mają znaleźć się w widoku. Nazwy poprzedzają skrótowe nazwy każdej z używanej tabeli, które zainicjalizowane zostały po słowie FROM. Do połączenia tabel użyto połączenia wewnętrznego INNER JOIN, w celu wyszczególnienia jedynie elementów zgodnych z podanymi warunkami w sekcji ON.

Listing 13: Widok adres\_klienta

```

1 CREATE VIEW `adres_klienta` AS
2 SELECT
3   kl.`idKlient`,
4   os.`Imie`,
5   os.`Nazwisko`,

```



```

6      os.`Pesel`,
7      adr.`Kraj`,
8      adr.`Miasto`,
9      adr.`Ulica`,
10     adr.`NrDomu`,
11     adr.`NrMieszkania`,
12     adr.`KodPocztowy`
13 FROM
14     `Osoba` AS os
15 INNER JOIN
16     `Adres` AS adr
17 ON
18     os.`Adres_idAdres` = adr.`idAdres`
19 INNER JOIN
20     `Klient` AS kl
21 ON
22     kl.`Osoba_idOsoba` = os.`idOsoba`;

```

Jako, że widok jest wirtualną tabelą, wywołanie i wyświetlenie odbywa się tak samo jak w przypadku zwykłych tabel.

Listing 14: Użycie widoku adres\_klienta

```

1 SELECT * FROM adres_klienta

```

### 4.3 Implementacja uprawnień i innych zabezpieczeń

Nadanie uprawnień klienta

Listing 15: Nadanie uprawnień klienta

```

1 GRANT SELECT, ON Osoba TO klient
2 GRANT SELECT, ON Adres TO klient
3 GRANT SELECT, ON DaneKontaktowe TO klient
4 GRANT SELECT ON Czujniki TO klient
5 GRANT SELECT, UPDATE, CREATE ON Zamoweienie TO klient
6 GRANT SELECT ON Czujnk TO klient
7 GRANT SELECT ON Producent TO klient
8 GRANT SELECT ON Urzadzenia TO klient
9 GRANT SELECT, UPDATE ON Urzadzenia_w_zamowieniu TO klient
10 GRANT SELECT ON Pomiary TO klient

```

Nadanie uprawnień montera

Listing 16: Nadanie uprawnień montera

```

1 GRANT SELECT ON Osoba TO monter
2 GRANT SELECT ON Adres TO monter
3 GRANT SELECT ON DaneKontaktowe TO monter
4 GRANT SELECT ON Czujnik TO monter
5 GRANT SELECT ON Producent TO monter
6 GRANT SELECT ON Urzadzenia TO monter
7 GRANT SELECT ON Urzadzenia_w_zamowieniu TO monter

```

Nadanie uprawnień administratora

Listing 17: Nadanie uprawnień administratora

```

1 GRANT SELECT ON Osoba TO administrator
2 GRANT SELECT ON Adres TO administrator
3 GRANT SELECT ON DaneKontaktowe TO administrator
4 GRANT SELECT ON Zamowienia TO administrator
5 GRANT ALL PRIVILEGES ON Czujnik TO administrator
6 GRANT ALL PRIVILEGES ON Producent TO administrator
7 GRANT ALL PRIVILEGES ON Urzadzenia TO administrator
8 GRANT SELECT, UPDATE, DROP ON Urzadzenia_w_zamowieniu TO administrator
9 GRANT ALL PRIVILEGES ON Pomiary TO administrator

```

Nadanie uprawnień kierownika działu technicznego

Listing 18: Nadanie uprawnień kierownika działu technicznego

```
1 GRANT SELECT DROP ON Osoba TO KierownikDzialuTechnicznego
2 GRANT SELECT ON Adres TO KierownikDzialuTechnicznego
3 GRANT SELECT ON DaneKontaktowe TO KierownikDzialuTechnicznego
4 GRANT ALL PRIVILEGES ON Czujnik TO KierownikDzialuTechnicznego
5 GRANT SELECT ON Zamowienia TO KierownikDzialuTechnicznego
6 GRANT ALL PRIVILEGES ON Producent TO KierownikDzialuTechnicznego
7 GRANT ALL PRIVILEGES ON Urzadzenia TO KierownikDzialuTechnicznego
8 GRANT SELECT, UPDATE, DROP ON Urzadzenia_w_zamowieniu TO
  KierownikDzialuTechnicznego
9 GRANT ALL PRIVILEGES ON Pomiary TO KierownikDzialuTechnicznego
10 GRANT SELECT ON Pracownik TO KierownikDzialuTechnicznego
11 GRANT SELECT ON Stanowisko TO KierownikDzialuTechnicznego
12 GRANT SELECT ON Klient TO KierownikDzialuTechnicznego
```

Nadanie uprawnień właściciela

Listing 19: Nadanie uprawnień właściciela

```
1 GRANT ALL PRIVILEGES ON Osoba TO Wlasciciel
2 GRANT ALL PRIVILEGES ON Adres TO Wlasciciel
3 GRANT ALL PRIVILEGES ON DaneKontaktowe TO Wlasciciel
4 GRANT ALL PRIVILEGES ON Czujniki TO Wlasciciel
5 GRANT ALL PRIVILEGES ON Zamowienia TO Wlasciciel
6 GRANT ALL PRIVILEGES ON Czujnik TO Wlasciciel
7 GRANT ALL PRIVILEGES ON Producent TO Wlasciciel
8 GRANT ALL PRIVILEGES ON Urzadzenia TO Wlasciciel
9 GRANT ALL PRIVILEGES ON Urzadzenia_w_zamowieniu TO Wlasciciel
10 GRANT ALL PRIVILEGES ON Pomiary TO Wlasciciel
11 GRANT ALL PRIVILEGES ON Pracownik TO Wlasciciel
12 GRANT ALL PRIVILEGES ON Stanowisko TO Wlasciciel
13 GRANT ALL PRIVILEGES ON Klient TO Wlasciciel
```

Nadanie uprawnień obsługi klienta

Listing 20: Nadanie uprawnień obsługi klienta

```
1 GRANT SELECT ON Klient TO Obsluga
2 GRANT SELECT ON Osoba TO Obsluga
3 GRANT SELECT ON Adres TO Obsluga
4 GRANT SELECT ON DaneKontaktowe TO Obsluga
5 GRANT SELECT ON Zamowienia TO Obsluga
6 GRANT SELECT ON Producent TO Obsluga
7 GRANT SELECT ON Urzadzenia_w_zamowieniu TO Obsluga
```

## 4.4 Testowanie bazy danych na przykładowych danych

Podczas próby wprowadzenia wartości, która przekracza maksymalną dopuszczalną ilość znaków otrzymamy komunikat z Rys. 11.



NrDomu    varchar(9)    123456789A    Please enter no more than 9 characters

Rysunek 11: Ograniczenie ilości znaków

Podczas dodawania nowej osoby do bazy danych, może się okazać, że osobę z podanym peselem już posiadamy w bazie danych. W takim wypadku zadziała zabezpieczenie dbające o to, aby wartości w polu "Pesel" była unikalna. Podczas próby dodania osoby o takim samym peselu wyświetli się komunikat z Rys. 12. oraz informację nie zostaną dodane do bazy danych.



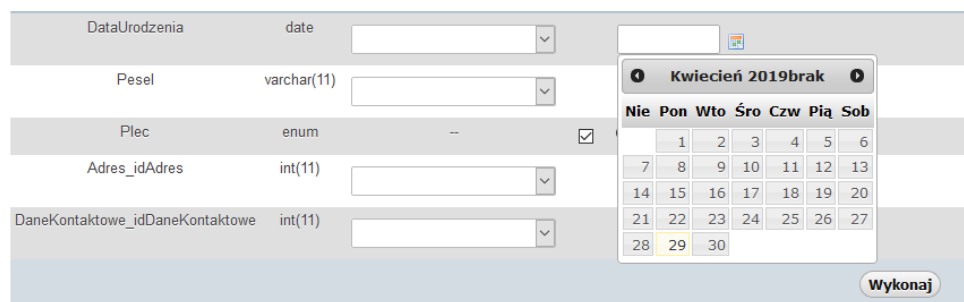
Rysunek 12: Unikalny pesel

Komórka "Płeć" jest typu "enum" i możliwe są trzy opcje do wyboru. Pierwsza opcja oznacza, że pole jest puste, drugi przypadek to literka "k", która oznacza kobietę, oraz trzeci przypadek to literka "m", która analogicznie oznacza mężczyznę.



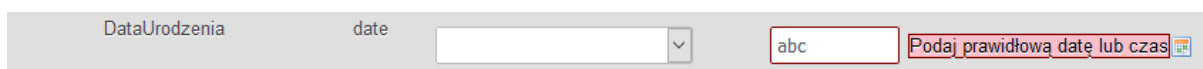
Rysunek 13: Dwie możliwe płcie do wyboru

W przypadku pola zawierającego datę urodzenia osoby, ze względu na wybrany typ "date", podczas dodawania nowego rekordu możliwe jest wybranie konkretnego dnia za pomocą wyświetlającego się kalendarza (Rys. 14).



Rysunek 14: Typ "DATE"

Wpisanie w komórkę formatu innego niż "rrrr\_mm\_dd" wyskakuje błąd informujący o podaniu nie poprawnej wartości w komórce z datą urodzenia.



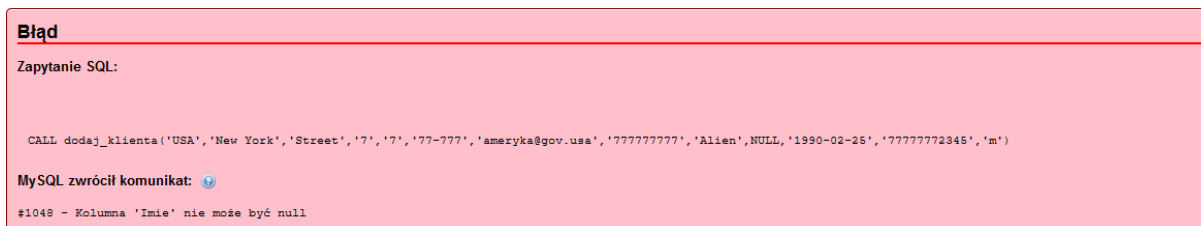
Rysunek 15: Niepoprawny format daty

W przypadku komórki zawierającej informacje dotyczącą stanowiska, na którym pracuje dany pracownik, zaimplementowane zostały trzy domyślne wartości (Rys. 15): Programista, Technik oraz Elektronik.



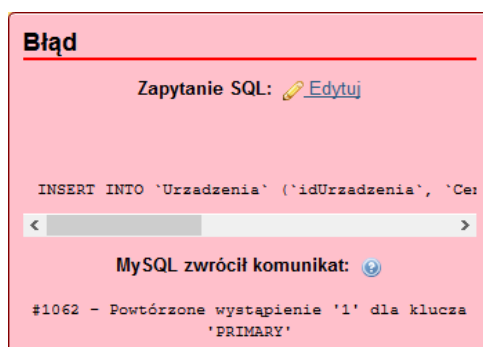
Rysunek 16: Domyślne wartości komórki (Programista, Technik, Elektryk)

W przypadku wywołania procedury "dodaj\_klienta" SZBD sprawdza czy wszystkie komórki z flagą "NN" czy są puste. W momencie wykrycia nieprawidłowości operacja "INSERT" jest przerywana i wyskakuje błąd informujący, która kolumna nie może być "NULL".

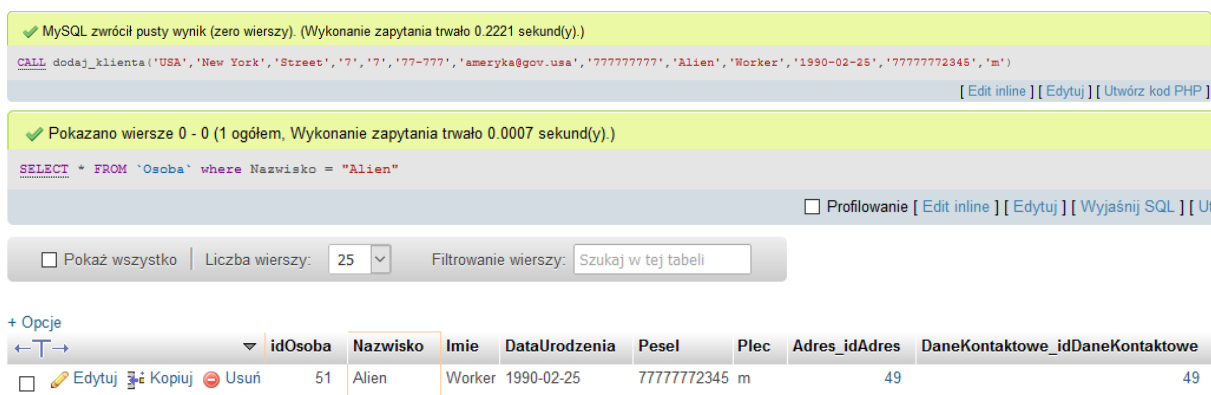


Rysunek 17: Błąd wartości NULL

Każdy z głównych kluczy dla danej tabeli powinien być unikalny, dlatego SZBD nie pozwala dodać rekordu z istniejącym już kluczem. Podczas próby wstawienia wystąpi błąd pokazany na Rys. 18.

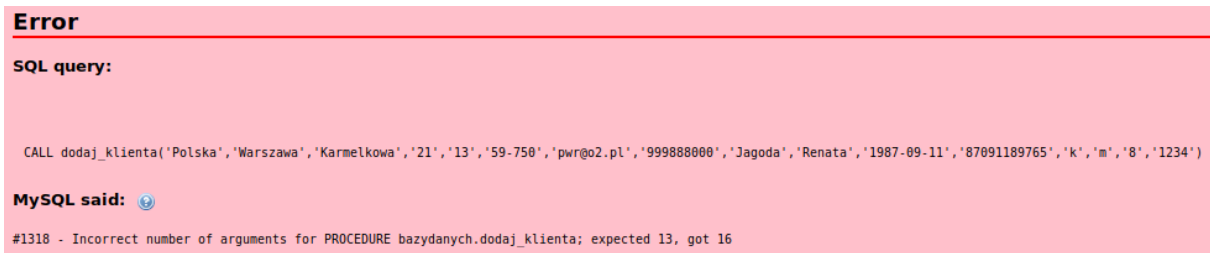


Rysunek 18: Powtórzone wystąpienie klucza



Rysunek 19: Wywołanie procedury "dodaj\_klienta" wraz z pokazanym poniżej nowo dodaną osobą

Podanie niewłaściwej ilości argumentów podczas wywoływania procedury skutkuje błędem informującym o ilości podanych parametrów oraz ile powinno być wprowadzonych.



Rysunek 20: Wywołanie procedury "dodaj\_klienta" niewłaściwa liczba argumentów

Przy próbie wywołania dozwolonej akcji dla danego użytkownika, np. edycja danych z tabeli "Pracownik" używając polecenia UPDATE 'Pracownik' SET 'StawkaPodstawowa' = '2000' WHERE 'Pracownik'. 'idPracownik' = 2 AND 'Pracownik'. 'Stanowisko\_idStanowisko' = 2; Akcja zostaje wykonana.

Przy próbie wywołania akcji niedozwolonej dla danego użytkownika, np. odczyt danych z tabeli 'Pracownik' używając polecenia SELECT \* FROM 'Pracownik' wyświetla się komunikat o braku możliwości wykonania danej akcji.



Rysunek 21: Komunikat błędu

## 5 Implementacja i testy aplikacji

### 5.1 Instalacja i konfigurowanie systemu

Całym fundamentem strony z interfejsem dla użytkownika jest język Python w wersji 3.6 wraz ze szkieletem, który został oparty na bibliotece "Flask". Biblioteka ta umożliwia uruchomienie strony internetowej, zarządzanie plikami z rozszerzeniem "html" oraz stosowanie wstawek w języku JavaScript.

Listing 21: Wykorzystane moduły w Python

```
1 from flask import Flask, render_template, flash, redirect, url_for, session,
   request, logging, jsonify, app
2 import mysql.connector
3 from wtforms import Form, StringField, TextAreaField, PasswordField, validators
4 from passlib.hash import sha256_crypt
5 from functools import wraps
6 from datetime import timedelta
```

- render\_template – Umożliwia wyświetlanie stron za pomocą szablonów w języku HTML
- flash – Umożliwia wyświetlanie wydarzeń w postaci popout
- redirect – przekierowuje użytkownika do innej lokalizacji docelowej z określonym kodem stanu.
- url\_for – Przekierowuje sesję na inną stronę
- session – Dodaje wsparcie po stronie serwera dla sesji użytkowników

- request – Umożliwia przesyłanie danych między Pythonem a HTMLem
- datetime – Zliczanie czasu, użyty został do wylogowania użytkownika po 5 minutach bezczynności
- passlib.hash – Umożliwia zaszyfrowanie hasła w standardzie SHA256
- mysql.connector – Moduł, który umożliwia nawiązanie połączenia z bazą danych

Listing 22: Przykładowa instalacja wybranych modułów

```
1 #!/bin/bash
2
3 apt-get install python3-flask
4 pip3 install wtforms
5 pip3 install mysql-connector
6 pip3 install Datetime
```

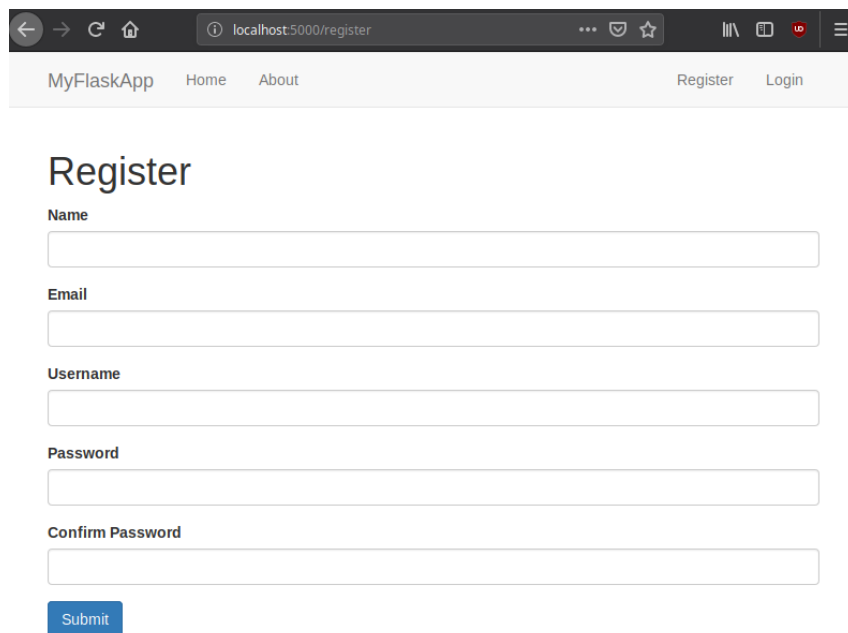
## 5.2 Instrukcja użytkownika aplikacji

### 5.2.1 Uruchamianie aplikacji

Aplikację uruchamia się przy pomocy Python 3.6 z bibliotekami. Po uruchomieniu należy w przeglądarce wpisać adres localhost:5000

### 5.2.2 Rejestracja

Rejestracja odbywa się w zakładce ”/register”.



Rysunek 22: Rejestracja użytkownika

W pola należy wpisać kolejno imię, email, nazwę użytkownika, hasło.

### 5.2.3 Logowanie

Logowanie odbywa się w zakładce ”/login”.

← → ↻ 🏠 localhost:5000/login ... 📄 📱 🔒 ☰

MyFlaskApp Home About Register Login

## Login

Username

Password

Submit

Rysunek 23: Rejestracja użytkownika

Logowanie następuje poprzez podanie nazwy użytkownika i hasła podanego w rejestracji. Po zalogowaniu pojawia się powiadomienie o poprawnym zalogowaniu i następuje przekierowanie na stronę użytkownika `"/dashboard"`, gdzie wyświetlone są przyciski dostępnych czujników.

#### 5.2.4 Konto użytkownika

Po zalogowaniu się wyświetla się strona startowa (`/dashboard`).

← → ↻ 🏠 localhost:5000/dashboard ... 📄 📱 🔒 ☰

MyFlaskApp Home About Dashboard Logout

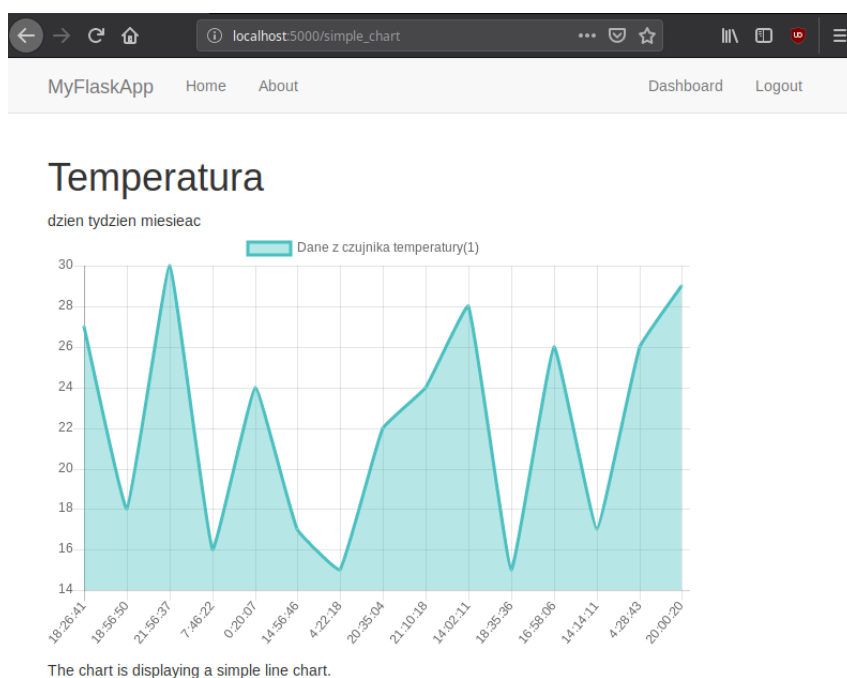
You are now logged in

Welcome bogdan with id 1

Temperatura Wilgotność

Rysunek 24: Rejestracja użytkownika

## 5.2.5 Wykresy pomiarów



Rysunek 25: Rejestracja użytkownika

Powrót do strony startowej można wykonać poprzez przycisk "wstecz" w przeglądarce.

## 5.3 Testowanie opracowanych funkcji systemu

### 5.3.1 •

Unauthorized, Please login

**Login**

Username

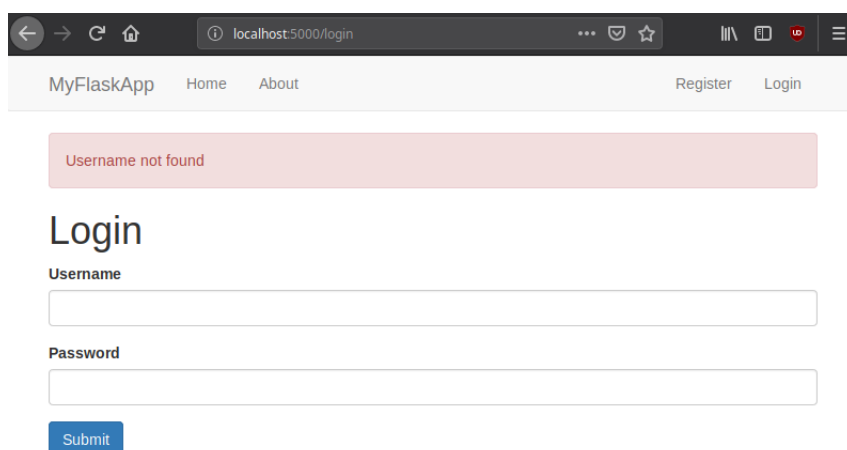
Password

Submit

Rysunek 26: Niepoprawna próba zalogowania



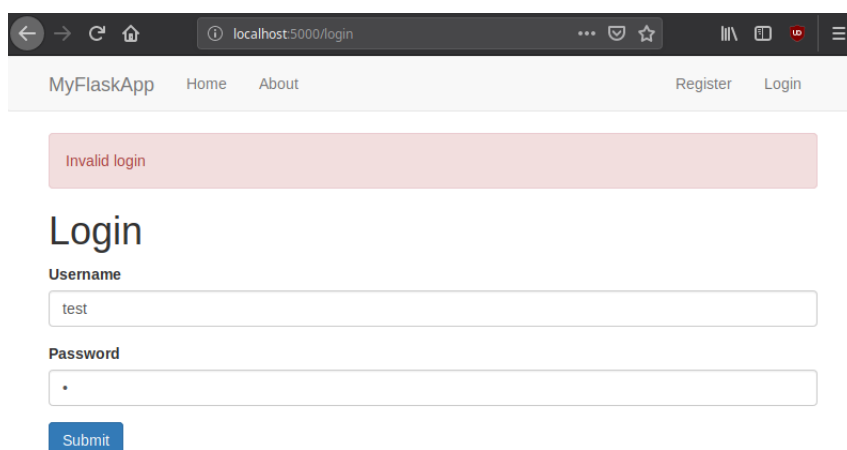
### 5.3.2 Próba logowania bez wprowadzania loginu i hasła



The screenshot shows a web browser at localhost:5000/login. The page has a header with 'MyFlaskApp', 'Home', 'About', 'Register', and 'Login' links. A red error message 'Username not found' is displayed at the top. Below it is the 'Login' section with 'Username' and 'Password' input fields, both of which are empty. A blue 'Submit' button is at the bottom.

Rysunek 27: Wprowadzenie niepoprawnej lub nie podanie nazwy użytkownika

### 5.3.3 Próba logowania z niepoprawnym hasłem



The screenshot shows the same web browser at localhost:5000/login. The 'Username' field now contains the text 'test'. The 'Password' field contains a single dot, indicating it is masked. A red error message 'Invalid login' is displayed at the top. The 'Submit' button remains at the bottom.

Rysunek 28: Wprowadzenie niepoprawnego hasła

## 5.4 Omówienie wybranych rozwiązań programistycznych

### 5.4.1 Implementacja interfejsu dostępu do bazy danych

Listing 23: Nawiązywanie połączenia z bazą danych

```
1 mySQL = mysql.connector.connect(host='adres_hostu',  
2                                database='nazwa_bazy_danych',  
3                                user='login',  
4                                password='hasło')
```

Listing 24: Pobieranie danych z bazy danych

```

1 #Przypisanie kursora bazy danych do zmiennej
2 cursor = mySQL_conn.cursor()
3 #Wywołanie zapytania w języku SQL. Zmienna [session['id']] jest indywidualną
   zmienną z sesji każdego zalogowanego użytkownika.
4 cursor.execute("SELECT WartoscPomiaru,GodzPomiaru FROM Pomiary WHERE
   Czujnik_idCzujnik=%s ORDER BY idPomiary DESC LIMIT 15",[session['id']])
5
6 #Zapisanie wyniku funkcji "cursor.execute" do zmiennej "rows"
7 rows = cursor.fetchall()
8 #Wyciągnięcie poszczególnych kolumn do zmiennych typu lista "value" i "hour"
9 for row in rows:
10     value.append(row[0])
11     hour.append(str(row[1]))
12 value.reverse()
13 hour.reverse()
14 #Zakończenie połączenia z bazą danych
15 mySQL_conn.close();

```

Listing 25: Wprowadzanie nowych użytkowników do bazy danych

```

1 #Wykorzystanie modułu request do pobrania danych od użytkownika ze strony
2 if request.method == 'POST' and form.validate():
3     name = form.name.data
4     email = form.email.data
5     username = form.username.data
6     password = sha256_crypt.encrypt(str(form.password.data))
7
8     # Stworzenie kursora
9     cur = mySQL.cursor(buffered=True)
10
11     # Wywołanie zapytanie w języku SQL
12     cur.execute("INSERT INTO user(name, email, username, password) VALUES(%s, %s,
   %s, %s)", (name, email, username, password))
13
14     # Wprowadzenie zmian w bazie danych
15     mySQL.commit()
16
17     # Zamknięcie połączenia z bazą danych
18     mySQL.close()
19
20 #Wyświetlenie komunikatu o poprawnej rejestracji
21 flash('You are now registered and can log in', 'success')

```

Listing 26: Odbieranie danych z czujników po UDP przez serwer

```

1 import mysql.connector
2 from mysql.connector import Error
3 from mysql.connector import errorcode
4 import socket
5
6 UDP_IP = "Nasłuchiwany_adres"
7 UDP_PORT = "Nasłuchiwany_port"
8
9 sock = socket.socket(socket.AF_INET, # Internet
10                      socket.SOCK_DGRAM) # UDP
11 sock.bind((UDP_IP, UDP_PORT))
12
13
14 mySQL_conn = mysql.connector.connect(host='Adres_hosta',

```

```

15                                     database='Nazwa_bazy_danych',
16                                     user='Login',
17                                     password='Haslo')
18 cursor = mySQL_conn.cursor()
19
20 while True:
21     # Oczekiwanie na pakiet danych
22     data, addr = sock.recvfrom(1024)
23
24     print("\n", type(data))
25     print("Otrzymane dane: ", data.decode(), "\n")
26
27     lista = data.decode().split(" ")
28     # Wyświetlenie zmiennej zawierające przetworzone dane
29     print(type(lista))
30     print(lista, "\n")
31     # Wywołanie procedury dodające dane do bazy danych
32     results_args = cursor.callproc('dodaj_klienta', lista)
33     print("Argumenty przekazane do funkcji callproc:\n", results_args)
34     # Zatwierdzenie zmian w bazie danych
35     mySQL_conn.commit()

```

Listing 27: Wysyłanie danych z czujników na serwer

```

1 import socket
2
3 UDP_IP = "Adress"
4 UDP_PORT = Port
5
6 MESSAGE = b"Rumunia Czarno Biala 01 01 00-109 NOWYMAIL@NA.pl 123321456 Jerzy
      ZWierzy 1990-01-01 98712312312 k"
7
8 print("UDP target IP:", UDP_IP)
9 print("UDP target port:", UDP_PORT)
10 print(MESSAGE.decode())
11
12 # Stworzenie obiektu typu socket i ustawienie parametrow
13 sock = socket.socket(socket.AF_INET, # Internet
14                      socket.SOCK_DGRAM) # UDP
15 # Wysłanie zmiennej MESSAGE na podany adres
16 sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))

```

#### 5.4.2 Implementacja wybranych funkcjonalności systemu

Listing 28: Timeout sesji

```

1 #Ustawienie timeout dla sesji (5min)
2 @app.before_request
3 def make_session_permanent():
4     session.permanent = True
5     app.permanent_session_lifetime = timedelta(minutes=5)

```

Listing 29: Rysowanie wykresu wraz ze skryptem aktualizującym dane co określony czas

```

1 </head>
2 <body>
3     <h1>Temperatura</h1>
4     /// Przyciski zmieniające zakres danych z tabelki /
5     <span class="d">dzien</span>
6     <span class="t">tydzien</span>
7     <span class="m">miesiac</span>
8     //////////////////////////////////////

```

```

9
10 // Wyświetlenie wykresu //////////////////////////////////////
11 <!-- bar chart canvas element -->
12 <canvas id="myChart" width="600" height="400"></canvas>
13 <p id="caption">The chart is displaying a simple line chart.</p>
14 //////////////////////////////////////
15
16 <script>
17     var url="/val";
18     var getvalues = function(results){
19 console.log(results);
20     // Globalne zmienne: //////////////////////////////////
21     Chart.defaults.global.responsive = false;
22     Chart.defaults.global.animation = false;
23     //////////////////////////////////
24     // define the chart data
25     var chartData = {
26
27 labels: results.hour,
28
29 datasets : [{
30     label: '{{ legend }}',
31     fill: true,
32     lineTension: 0.1,
33     backgroundColor: "rgba(75,192,192,0.4)",
34     borderColor: "rgba(75,192,192,1)",
35     borderCapStyle: 'butt',
36     borderDash: [],
37     borderDashOffset: 0.0,
38     borderJoinStyle: 'miter',
39     pointBorderColor: "rgba(75,192,192,1)",
40     pointBackgroundColor: "#fff",
41     pointBorderWidth: 1,
42     pointHoverRadius: 5,
43     pointHoverBackgroundColor: "rgba(75,192,192,1)",
44     pointHoverBorderColor: "rgba(220,220,220,1)",
45     pointHoverBorderWidth: 2,
46     pointRadius: 1,
47     pointHitRadius: 10,
48     //data : [{% for item in values %}
49         //     {{item}},
50         //     {% endfor %}],
51 data: results.value ,
52 spanGaps: false
53     }]
54 }
55
56 // get chart canvas
57 var ctx = document.getElementById("myChart").getContext("2d");
58
59 // create the chart using the chart canvas
60 var myChart = new Chart(ctx, {
61     type: 'line',
62     data: chartData,
63     });
64 };
65
66 var fun = function(){
67     var values = $.get(url)
68         .then(getvalues)};
69 setInterval(fun, 5000);
70
71 $('d').click(function(){setInter(15,2)});

```

```

72 $('t').click(function(){setInter(7,5)});
73 $('m').click(function(){setInter(30,3)});
74
75 function setInter(inter,sdf){
76     url="/val?inter="+inter+"&cos="+sdf;
77     fun();
78 }
79
80 </script>
81
82 </body>
83 </html>

```

Listing 30: Rejestrowanie użytkownika

```

1 @app.route('/register', methods=['GET', 'POST'])
2 def register():
3     form = RegisterForm(request.form)
4     if request.method == 'POST' and form.validate():
5         name = form.name.data
6         email = form.email.data
7         username = form.username.data
8         password = sha256_crypt.encrypt(str(form.password.data))
9
10        # Create cursor
11        cur = MySQL.cursor(buffered=True)
12
13        # Execute query
14        cur.execute("INSERT INTO user(name, email, username, password) VALUES(%s,
15            %s, %s, %s)", (name, email, username, password))
16
17        # Commit to DB
18        MySQL.commit()
19
20        # Close connection
21        MySQL.close()
22
23        flash('You are now registered and can log in', 'success')
24
25        return redirect(url_for('login'))
26    return render_template('register.html', form=form)

```

### 5.4.3 Implementacja mechanizmów bezpieczeństwa

Listing 31: Porównanie zaszyfrowanych haseł

```

1     # Pobranie hasła z bazy danych
2     password = data[4]
3
4     # Porównanie hasła
5     if sha256_crypt.verify(password_candidate, password):
6         # Passed
7         session['logged_in'] = True
8         session['username'] = username
9         session['id'] = data[0]
10        flash('You are now logged in', 'success')

```

Listing 32: Narzucenie długości znaków w poszczególnych polach

```

1 class RegisterForm(Form):
2     name = StringField('Name', [validators.Length(min=3, max=50)])

```

```

3     username = StringField('Username', [validators.Length(min=4, max=25)])
4     email = StringField('Email', [validators.Length(min=6, max=50)])
5     password = PasswordField('Password', [
6         validators.DataRequired(),
7         validators.EqualTo('confirm', message='Passwords do not match')
8     ])
9     confirm = PasswordField('Confirm Password')

```

Listing 33: Indywidualne dane pomiarowe w zakładce "/simple\_chart" dla każdego użytkownika

```

1 # Porównanie hasła
2 if sha256_crypt.verify(password_candidate, password):
3     # Poprawne hasło
4     session['logged_in'] = True
5     session['username'] = username
6     session['id'] = data[0]
7 (...)
8 # Użycie zmiennej session['id'] indywidualnej dla każdego użytkownika i wycią
   gnięcie danych pomiarowych należących do zalogowanego użytkownika
9     cursor.execute("SELECT WartoscPomiaru,GodzPomiaru FROM Pomiar WHERE
   Czujnik_idCzujnik=%s ORDER BY idPomiary DESC LIMIT 15",[session['id']
   ])

```

## 6 Podsumowanie i wnioski

### Spis rysunków

1	Widoki . . . . .	6
2	Poziomy dostępowe . . . . .	7
3	Ogólny diagram projektowy . . . . .	7
4	Klient . . . . .	8
5	Pracownik . . . . .	8
6	Pomiar . . . . .	9
7	Urządzenie . . . . .	9
8	Interfejs graficzny aplikacji 1 . . . . .	10
9	Interfejs graficzny aplikacji 2 . . . . .	11
10	Interfejs graficzny aplikacji 3 . . . . .	12
11	Ograniczenie ilości znaków . . . . .	17
12	Unikalny pesel . . . . .	18
13	Dwie możliwe płcie do wyboru . . . . .	18
14	Typ "DATE" . . . . .	18
15	Niepoprawny format daty . . . . .	18
16	Domyślne wartości komórki (Programista, Technik, Elektryk) . . . . .	18
17	Błąd wartości NULL . . . . .	19
18	Powtórzone wystąpienie klucza . . . . .	19
19	Wywołanie procedury "dodaj_klienta" wraz z pokazanym poniżej nowo dodaną osobą . .	19
20	Wywołanie procedury "dodaj_klienta" niewłaściwa liczba argumentów . . . . .	20
21	Komunikat błędu . . . . .	20
22	Rejestracja użytkownika . . . . .	21
23	Rejestracja użytkownika . . . . .	22
24	Rejestracja użytkownika . . . . .	22
25	Rejestracja użytkownika . . . . .	23
26	Niepoprawna próba zalogowania . . . . .	23
27	Wprowadzenie niepoprawnej lub nie podanie nazwy użytkownika . . . . .	24
28	Wprowadzenie niepoprawnego hasła . . . . .	24

## Listings

1	Producent_BEFORE_INSERT . . . . .	6
2	Osoba_AFTER_UPDATE . . . . .	6
3	Utworzenie bazy . . . . .	13
4	Utworzenie tabeli Adres . . . . .	13
5	Utworzenie klucza obcego . . . . .	13
6	Własności tworzonej tabeli . . . . .	13
7	Definiowanie indeksów . . . . .	13
8	Procedura dodaj_klienta . . . . .	14
9	Użycie dodaj_klienta . . . . .	14
10	Procedura dodaj_pracownika . . . . .	14
11	Użycie dodaj_pracownika . . . . .	15
12	Adres_BEFORE_INSERT . . . . .	15
13	Widok adres_klienta . . . . .	15
14	Użycie widoku adres_klienta . . . . .	16
15	Nadanie uprawnień klienta . . . . .	16
16	Nadanie uprawnień montera . . . . .	16
17	Nadanie uprawnień administratora . . . . .	16
18	Nadanie uprawnień kierownika działu technicznego . . . . .	17
19	Nadanie uprawnień właściciela . . . . .	17
20	Nadanie uprawnień obsługi klienta . . . . .	17
21	Wykorzystane moduły w Python . . . . .	20
22	Przykładowa instalacja wybranych modułów . . . . .	21
23	Nawiązywanie połączenia z bazą danych . . . . .	24
24	Pobieranie danych z bazy danych . . . . .	25
25	Wprowadzanie nowych użytkowników do bazy danych . . . . .	25
26	Odbieranie danych z czujników po UDP przez serwer . . . . .	25
27	Wysyłanie danych z czujników na serwer . . . . .	26
28	Timeout sesji . . . . .	26
29	Rysowanie wykresu wraz ze skryptem aktualizującym dane co określony czas . . . . .	26
30	Rejestrowanie użytkownika . . . . .	28
31	Porównanie zaszyfrowanych haseł . . . . .	28
32	Narzucenie długości znaków w poszczególnych polach . . . . .	28
33	Indywidualne dane pomiarowe w zakładce "/simple_chart" dla każdego użytkownika . . . . .	29