

POLITECHNIKA WROCŁAWSKA  
WYDZIAŁ ELEKTRONIKI

## PROJEKT Z BAZ DANYCH

### **System obsługi hotelu**

Termin zajęć: Środa, 13:15–15:00

AUTOR/AUTORZY:

Tomek Kubat 218551

PROWADZĄCY ZAJĘCIA:

dr inż. Roman Ptak, W4/K9

Wrocław, 2019 r.

## Spis treści:

<u>1. Wstęp</u>	4
<u>1.1. Cel projektu</u>	4
<u>1.2. Zakres projektu</u>	4
<u>2. Analiza wymagań</u>	4
<u>2.1. Opis działania i schemat logiczny systemu</u>	4
<u>2.2. Wymagania funkcjonalne</u>	4
<u>2.3. Wymagania niefunkcjonalne</u>	4
<u>2.3.1. Wykorzystywane technologie i narzędzia</u>	4
<u>2.3.2. Wymagania dotyczące rozmiaru bazy danych</u>	4
<u>2.3.3. Wymagania dotyczące bezpieczeństwa systemu</u>	4
<u>2.4. Przyjęte założenia projektowe</u>	4
<u>3. Projekt systemu</u>	4
<u>3.1. Projekt bazy danych</u>	4
<u>3.1.1. Analiza rzeczywistości i uproszczony model conceptualny</u>	4
<u>3.1.2. Model logiczny i normalizacja</u>	4
<u>3.1.3. Model fizyczny i ograniczenia integralności danych</u>	4
<u>3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych</u>	4
<u>3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych</u>	4
<u>3.2. Projekt aplikacji użytkownika</u>	4
<u>3.2.1. Architektura aplikacji i diagramy projektowe</u>	4
<u>3.2.2. Interfejs graficzny i struktura menu</u>	4
<u>3.2.3. Projekt wybranych funkcji systemu</u>	4
<u>3.2.4. Metoda podłączania do bazy danych – integracja z bazą danych</u>	4
<u>3.2.5. Projekt zabezpieczeń na poziomie aplikacji</u>	4
<u>4. Implementacja systemu baz danych</u>	4
<u>4.1. Tworzenie tabel i definiowanie ograniczeń</u>	4

<a href="#"><u>4.2. Implementacja mechanizmów przetwarzania danych</u></a>	5
<a href="#"><u>4.3. Implementacja uprawnień i innych zabezpieczeń</u></a>	5
<a href="#"><u>4.4. Testowanie bazy danych na przykładowych danych</u></a>	5
<a href="#"><u>5. Implementacja i testy aplikacji</u></a>	5
<a href="#"><u>5.1. Instalacja i konfigurowanie systemu</u></a>	5
<a href="#"><u>5.2. Instrukcja użytkowania aplikacji</u></a>	5
<a href="#"><u>5.3. Testowanie opracowanych funkcji systemu</u></a>	5
<a href="#"><u>5.4. Omówienie wybranych rozwiązań programistycznych</u></a>	5
<a href="#"><u>5.4.1. Implementacja interfejsu dostępu do bazy danych</u></a>	5
<a href="#"><u>5.4.2. Implementacja wybranych funkcjonalności systemu</u></a>	5
<a href="#"><u>5.4.3. Implementacja mechanizmów bezpieczeństwa</u></a>	5
<a href="#"><u>6. Podsumowanie i wnioski</u></a>	5
<a href="#"><u>Literatura</u></a>	5
<a href="#"><u>Spis rysunków</u></a>	5
<a href="#"><u>Spis tabel</u></a>	5

# **1. Wstęp**

## **1.1. Cel projektu**

Stworzenie bazy danych i aplikacji webowej do obsługi rezerwacji hotelu.

## **1.2. Zakres projektu**

Projekt będzie polegał na stworzeniu bazy danych która będzie przechowywać dane dotyczące hotelu, klientów, pracowników i rezerwacji. Stworzenie aplikacji webowej która pozwoli w szybki i prosty sposób na utworzenie rezerwacji i sprawdzenie konkretnych danych.

# **2. Analiza wymagań**

Wybór i opracowanie wstępnych założeń dotyczących wybranych tematów projektów.

## **2.1. Opis działania i schemat logiczny systemu**

Hotel usytuowany we Wrocławiu o wysokości 3 pięter. Na pierwszych dwóch znajdują się pokoje dwuosobowe na trzecim natomiast apartamentowce które mieszczą po cztery osoby. Liczba pokoi dwuosobowych 20 (10 na piętro) i 4 apartamentowce. Hotel oferuje możliwość rezerwacji mailowej i telefonicznej. Każda potwierdzona rezerwacja jest zapisywana i widoczna w bazie danych (dane wynajmującego, termin, opłata). Aplikacja webowa pozwala łatwo śledzić i sprawdzać informacje o gościach, wolnych pokojach i opłatach.

## **2.2. Wymagania funkcjonalne**

1.Pracownik :

- wprowadzenie rezerwacji
- modyfikacja rezerwacji
- usuwanie rezerwacji
- potwierdzenie zameldowania
- modyfikacja meldunku
- sprawdzenie czy są wolne pokoje
- dodawanie danych gości
- sprawdzenie danych gości
- modyfikacja danych gości
- przeglądanie historii rezerwacji
- wyszukiwanie rezerwacji w danym przedziale czasu
- obliczanie kosztu pobytu

- generowanie rachunku

## 2.Administrator:

- posiada wszystkie funkcjonalności pracownika
- usuwanie oraz modyfikacja rachunku
- zarządzanie danymi pracowników (dodawanie, usuwanie itp.)

## 2.3. Wymagania niefunkcjonalne

### 2.3.1. Wykorzystywane technologie i narzędzia

Baza danych powstanie w mySQL natomiast aplikacja webowa powstanie w ASP.net.

### 2.3.2. Wymagania dotyczące rozmiaru bazy danych

Baza danych będzie niewielka ze względu na małą liczbę pokoi w hotelu. Wraz z wzrostem popularności hotelu baza będzie posiadać dane kilku tysięcy klientów oraz podobną ilość rezerwacji. Dane klientów będą przechowywane przez 5 lat. Przewidujemy w dobrych warunkach około czterech tysięcy gości rocznie i około dwóch tysięcy pobyków. Liczba pracowników będzie oscylować w okolicy pięciu osób. Dane dotyczące samego hotelu, pokoi nie będzie się zmieniać.

### 2.3.3. Wymagania dotyczące bezpieczeństwa systemu

Ze względu na możliwy dostęp do bazy danych przez internet, zastosowano szyfrowanie pomiędzy użytkownikiem oraz serwerem, potwierdzone certyfikatem SSL. Zastosowano system archiwizacji bazy danych jak i całego systemu. Napisano skrypt który codziennie w godzinach nocnych wykonuje backup wszystkich danych oraz wysyła je na dysk twardy. W przypadku niepowodzenia/powodzenia backupu administrator zostaje poinformowany. W przypadku błędu systemu lub użytkownika będzie możliwe przywrócenie bazy danych oraz całego systemu.

## 2.4.Przyjęte założenia projektowe

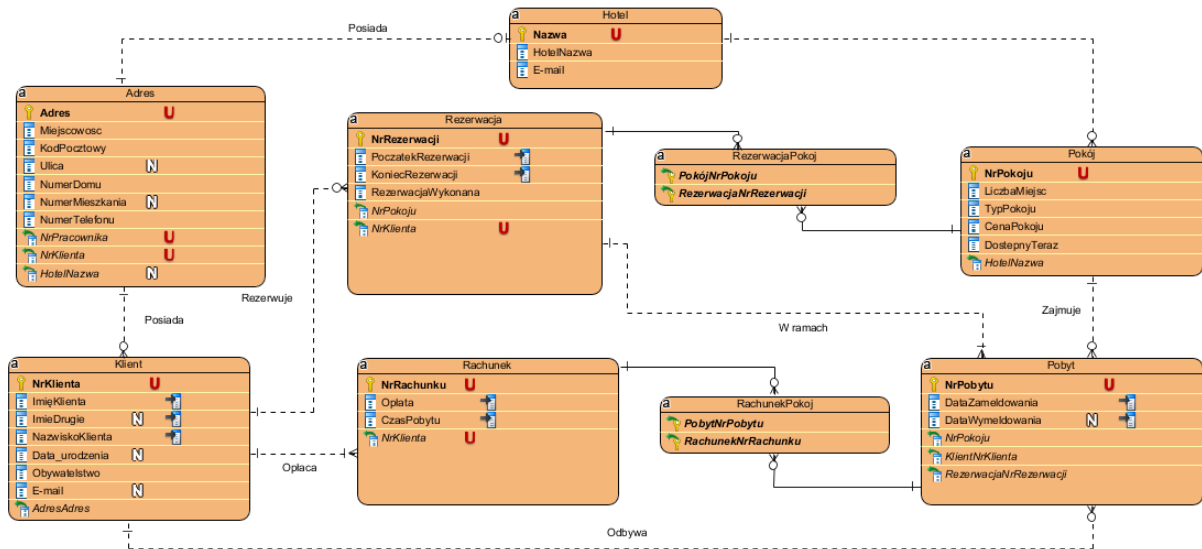
Ze względu na przyjęte dane, baza nie będzie zbyt duża. Pozwoli w prosty sposób sprawdzić informacje i rezerwację miejsc hotelowych z poziomu aplikacji webowej. Administrator hotelu będzie miał dodatkowe możliwości dodawania nowych kont pracowników, zmieniania ich danych osobowych oraz dostęp do usunięcia bądź edycji rachunku przy losowych problemach. Prosty system zabezpieczający logowania pracowników i archiwizacji daje bezpieczeństwo przed utratą danych.

## 3. Projekt systemu

Projekt i struktury bazy danych, mechanizmów zapewniania poprawności przechowywanych informacji, oraz kontroli dostępu do danych.

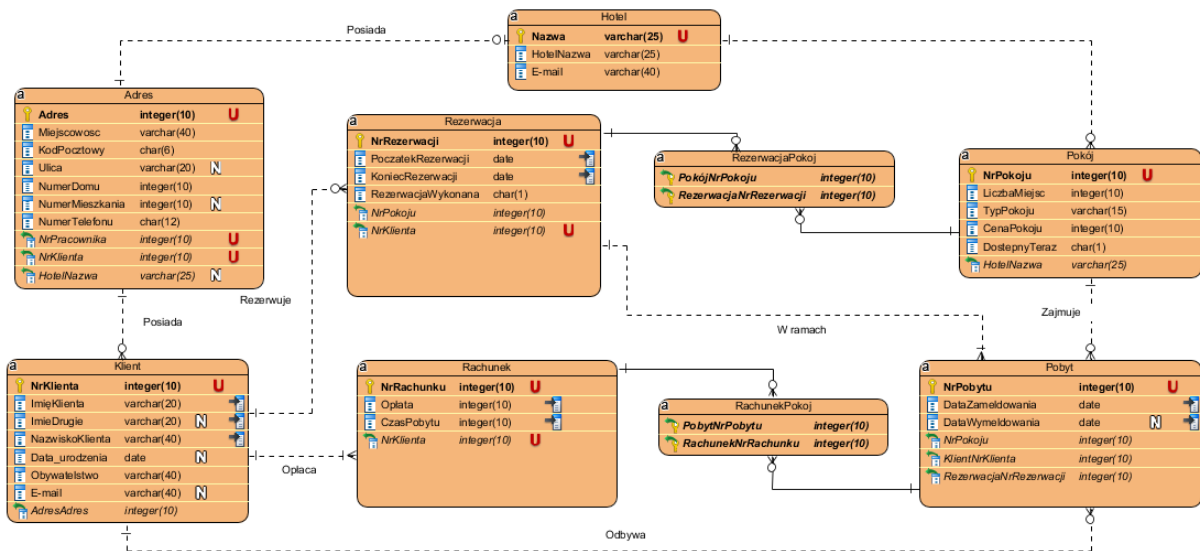
## 3.1. Projekt bazy danych

### 3.1.1. Analiza rzeczywistości i uproszczony model konceptualny



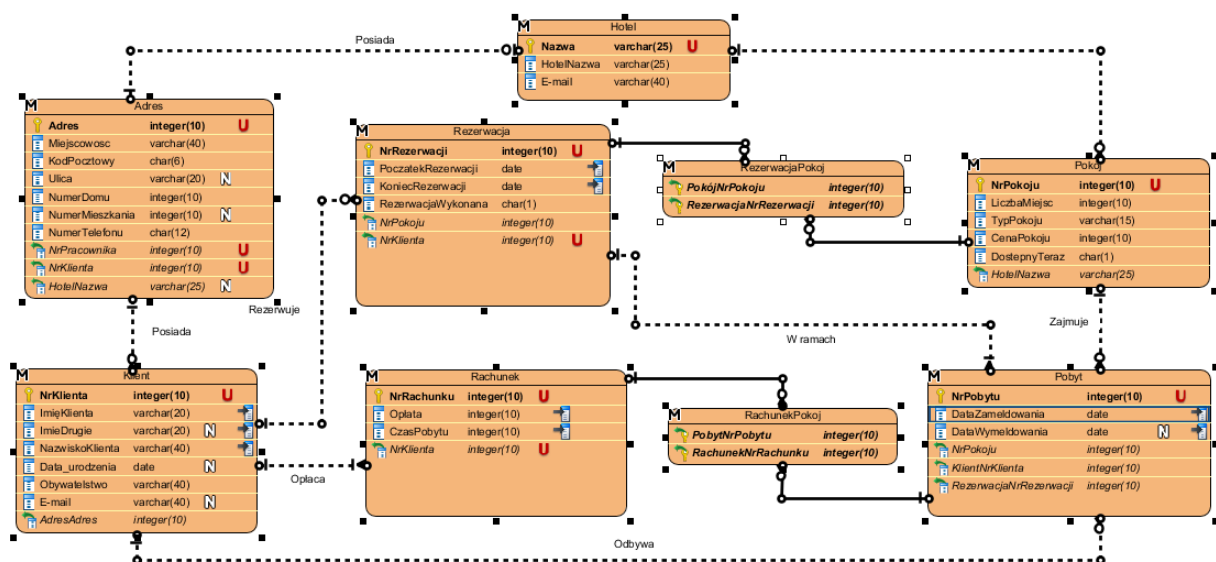
3.1. Model konceptualny stworzony w programie Visual Paradigm dla systemu bazodanowego MySQL.

### 3.1.2. Model logiczny i normalizacja



3.2. Model logiczny stworzony w programie Visual Paradigm dla systemu bazodanowego MySQL.

### 3.1.3. Model fizyczny i ograniczenia integralności danych



3.3. Model fizyczny stworzony w programie Visual Paradigm dla systemu bazodanowego MySQL.

### 3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych

Sekwencje w przy tworzeniu kluczy będą tworzone przy pomocy funkcji „autoincrement” systemu MySQL.

#### Widoki:

wolne\_ten\_miesiac – jest to widok, który wyświetla informacje o nr pokoi oraz terminach dostępności w danym miesiącu. Wykorzystuje informacje z tabel rezerwacja oraz pokój.

platnosc\_pokoj – jest to widok, który wyświetla dane o nr rezerwacji, nr pobytu na określony pokój przez danego klienta(nr klienta) oraz cenie pobytu. Wykorzystuje do tego złączenie tabel: pobyt, klient, rachunek.

historia\_rezerwacji – jest to widok, który wyświetla informacje na temat wszystkich gości przebywających kiedykolwiek w hotelu oraz okresie rezerwacji na podstawie tabeli rezerwacja.

goscie\_teraz – jest to widok, który wyświetla informacje na temat wszystkich gości obecnie znajdujących się hotelu. Korzysta z tabeli pobyt.

dane\_klient- jest to widok, który wyświetla informacje na temat określonego klienta. Korzysta z tabeli klient.

### **Wyzwalacze:**

mozliwosc\_rezerwacji – jest to wyzwalacz przypięty do tabeli rezerwacja, uruchamia się on przed dodaniem nowej krotki do tabeli rezerwacja i sprawdza czy można zarezerwować określony pokój w określonym przedziale czasowym, w tym celu sprawdza zawartość tabeli rezerwacja oraz pobyt.

duze\_litery\_klient – wyzwalacz przypięty do tabeli klient, który w momencie wprowadzania nowego klienta będzie sprawdzał czy pola Imię, DrugieImię, Nazwisko zaczynają się od dużej litery w przeciwnym wypadku podmienia je.

duze\_litery\_adres- wyzwalacz przypięty do tabeli adres, który w momencie podawania nowego adresu będzie sprawdzał czy pola Miejscowosc oraz Ulica zaczynają się od dużej litery, w przeciwnym wypadku podmienia je.

poprawny\_nr\_tel – wyzwalacz przypięty do tabeli adres, który w momencie wpisywania wartości do kolumny NumerTelefonu sprawdza czy spełniają one założenia wskazujące na poprawność wprowadzanego numeru.

### **Procedury składowane:**

PokazRachunekKlienta – procedura przyjmująca jako argument id klienta i zwracająca jego sumę zobowiązań oraz czas pobytu.

SprawdzDostepnePokoje – procedura przyjmująca jako argument pożądaną przedział czasowy i zwracająca numery pokoi dostępnych w danym przedziale.

WykonajRezerwacje – procedura przyjmująca jako argument początek i koniec rezerwacji, nrKlienta, nr pokoju i dokonuje wpisu rezerwacji do bazy danych.

ZweryfikujKlienta – procedura przyjmująca jako argument nazwisko Klienta oraz zwracająca informacje czy osoba o danym nazwisku posiada aktywny pobyt.

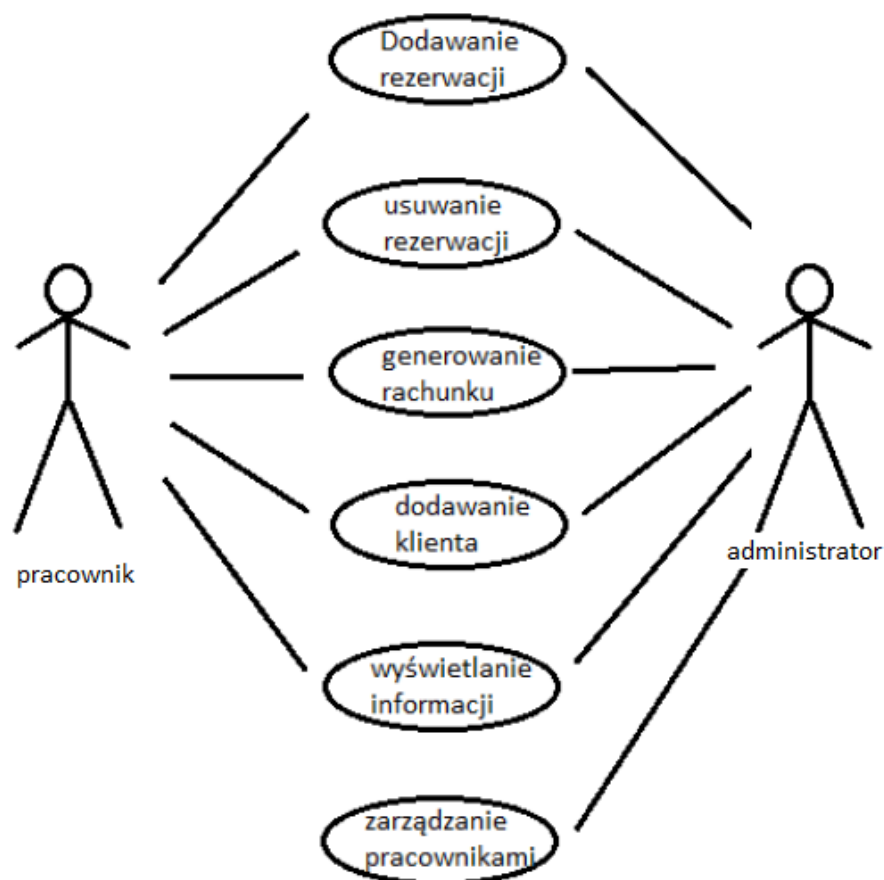


### 3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych

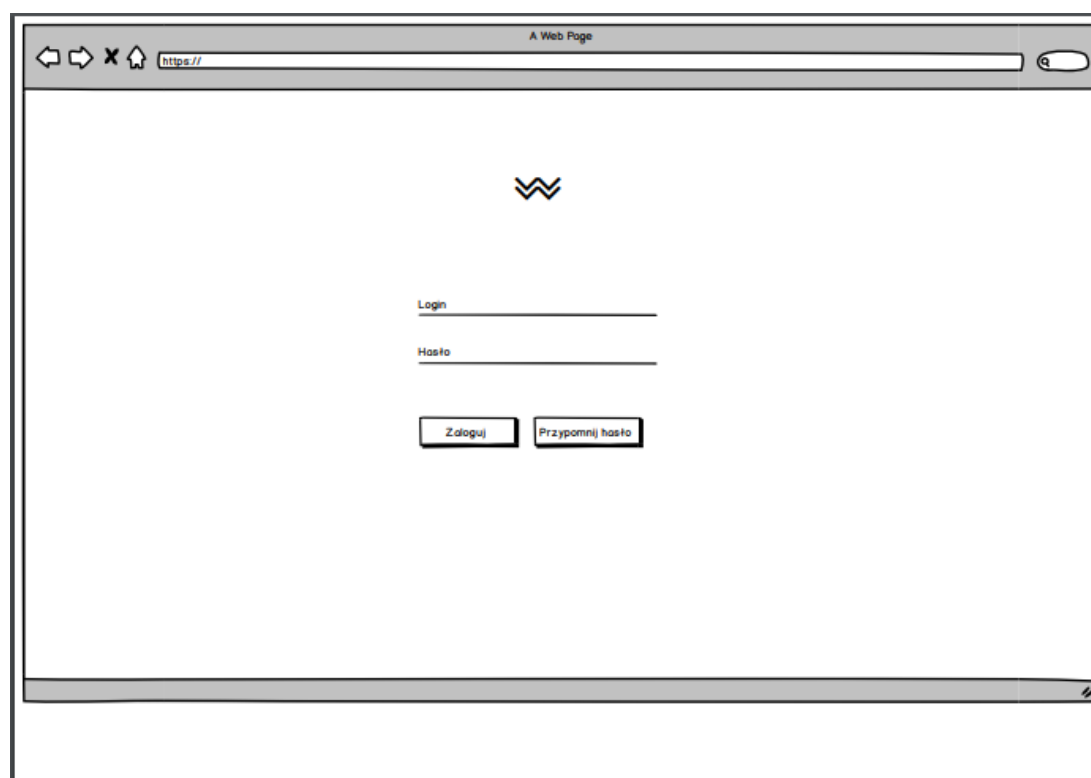
	Pracownik	Administrator
Hotel	Wyświetlanie	wyświetlanie modyfikacja usuwanie dodawanie
Klient	wyświetlanie modyfikacja usuwanie dodawanie	wyświetlanie modyfikacja usuwanie dodawanie
Adres	wyświetlanie modyfikacja usuwanie dodawanie	wyświetlanie modyfikacja usuwanie dodawanie
Rezerwacja	wyświetlanie modyfikacja usuwanie dodawanie	wyświetlanie modyfikacja usuwanie dodawanie
Pobyt	wyświetlanie modyfikacja usuwanie dodawanie	wyświetlanie modyfikacja usuwanie dodawanie
Rachunek	wyświetlanie dodawanie	wyświetlanie modyfikacja usuwanie dodawanie
Pokoj	Wyświetlanie	wyświetlanie modyfikacja usuwanie dodawanie

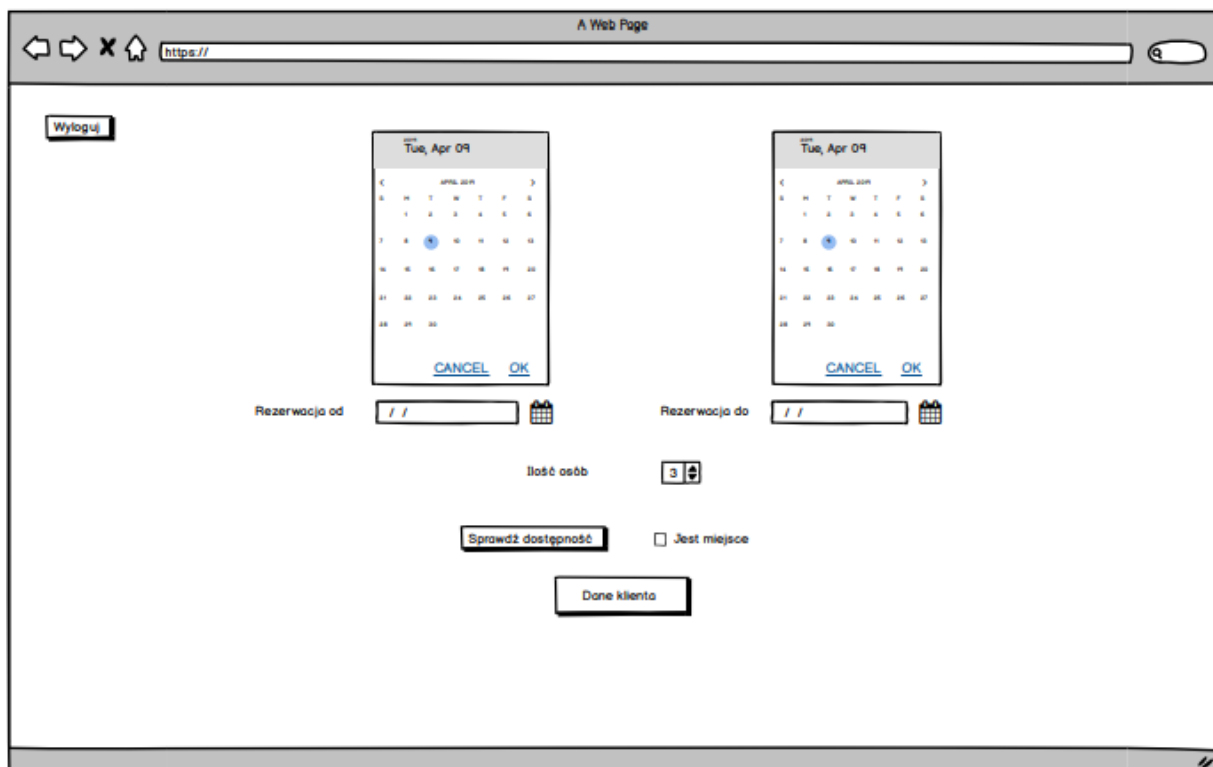
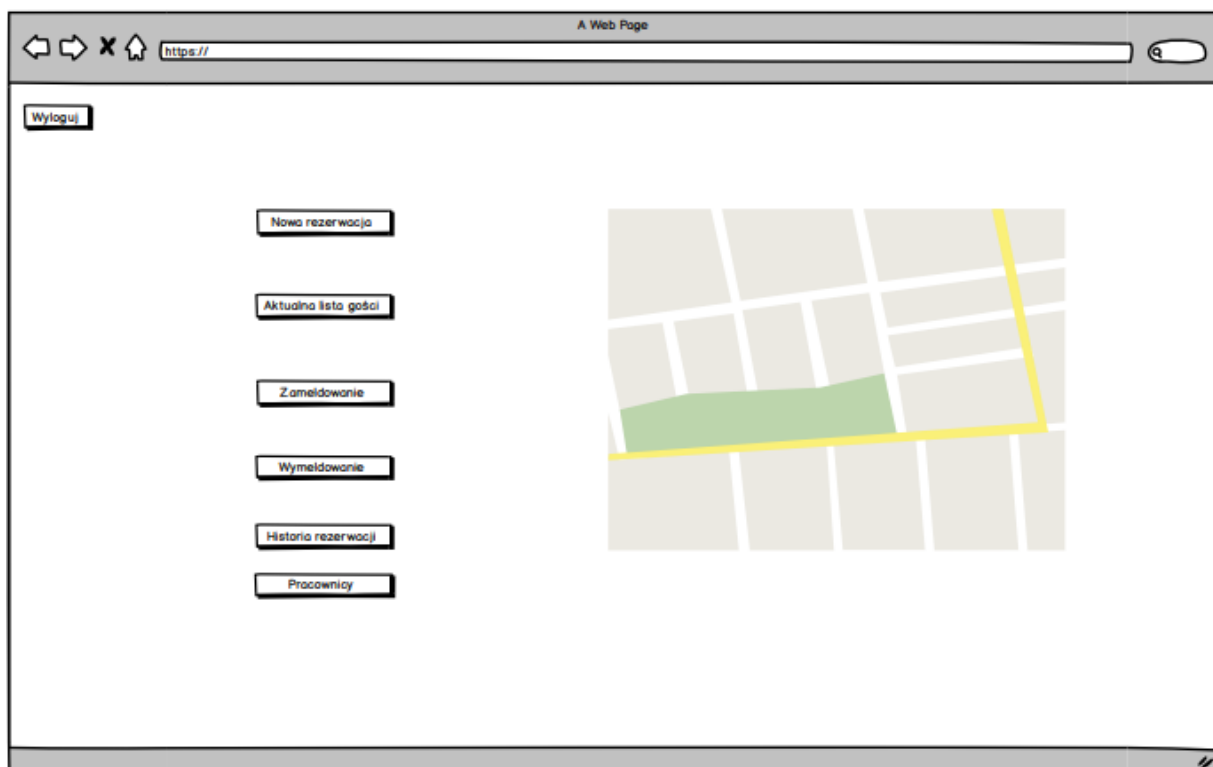
## 3.2. Projekt aplikacji użytkownika

### 3.2.1. Architektura aplikacji i diagramy projektowe



### 3.2.2. Interfejs graficzny i struktura menu





A Web Page

https://

Wyloguj

Imię

Nazwisko

Data urodzenia

Obywatelstwo

E-mail

Miejscowość

Kod pocztowy

Ulica

Nr domu

Nr mieszkania

Nr telefonu

Potwierdzenie danych

A Web Page

https://

Wyloguj

Imię

Nazwisko

Data urodzenia

Obywatelstwo

E-mail

Miejscowość

Kod pocztowy

Ulica

Nr domu

Nr mieszkania

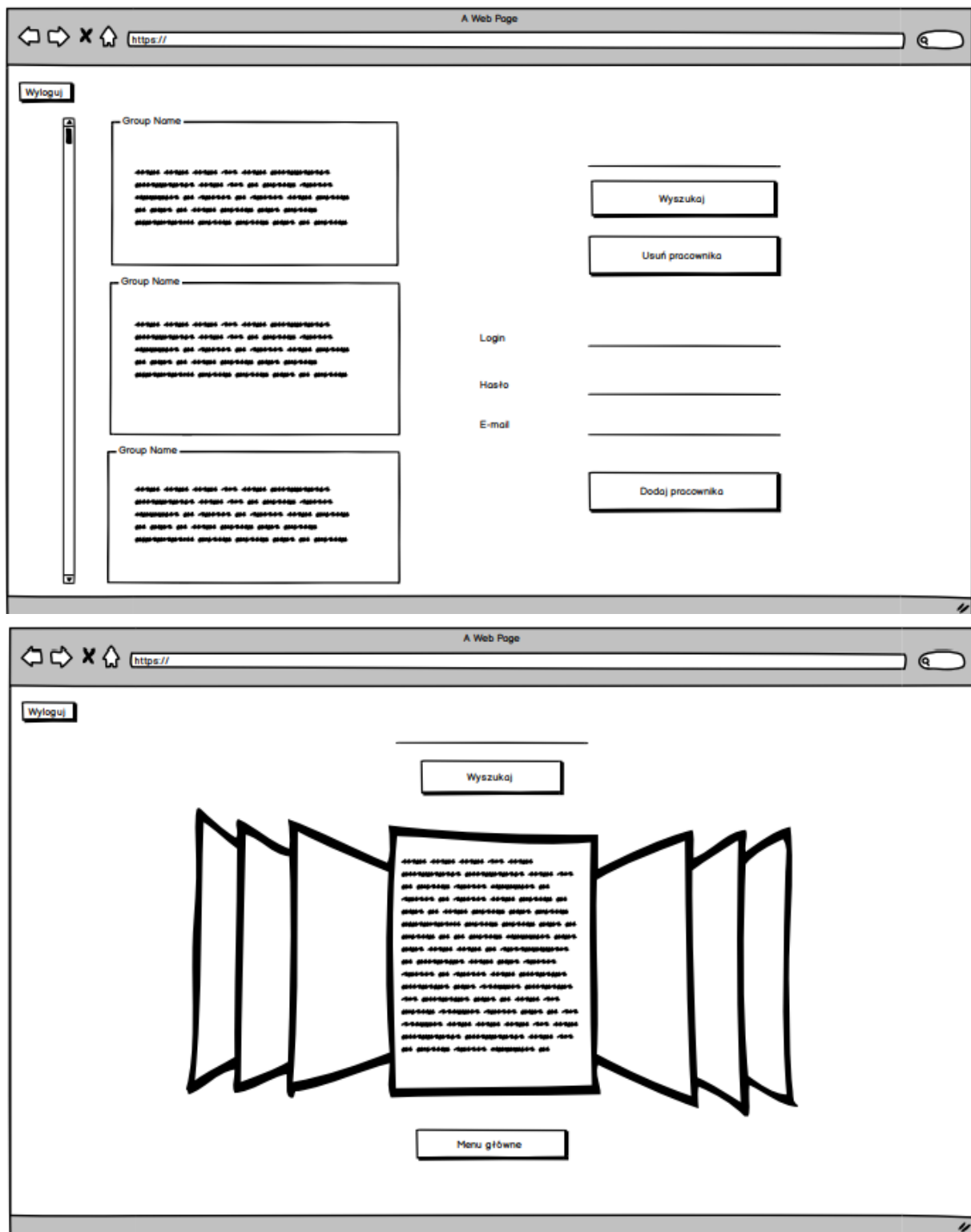
Nr telefonu

Ilość osób : 3  
Początek pobytu: 03.09.2019  
Koniec pobytu: 10.09.2019  
Szacowany koszt: 490 zł

Wprowadź rezerwację

Popraw dane

Anuluj rezerwację



### 3.2.3. Projekt wybranych funkcji systemu

**Rejestracja Pracownika.** Funkcja rejestrująca pracownika w systemie. Administrator po podaniu przez pracownika danych (imię, nazwisko, email) zakłada konto dla pracownika.

**Logowanie pracownika/administratora.** Funkcja logowania administratora w systemie. Jej celem jest sprawdzenie poprawności login'u i hasła. Po poprawnym zidentyfikowaniu następuje przekierowanie użytkownika do panelu pracownika/administratora.

**Przypomnienie hasła.** Funkcja umożliwiająca zmianę hasła konta administratora/pracownika systemu. W wyniku podania poprawnego email wysyła za pomocą poczty elektronicznej wygenerowany link umożliwiający zmianę hasła.

**Przegląd historii pobytu.** Funkcja umożliwia przeglądanie pobytów, które się już zakończyły. Umożliwiać będzie przeglądanie pobytów w określonych ramach czasowych, dla danego gościa, dla danego pokoju.

**Przegląd rezerwacji.** Funkcja umożliwia przeglądanie złożonych rezerwacji.

**Przegląd wolnych pokoi.** Funkcja umożliwia przeglądanie wolnych pokoi w wybranym terminie.

**Przegląd rezerwacji dla danego pokoju.** Funkcja umożliwia przegląd rezerwacji konkretnego pokoju.

### 3.2.4. Metoda podłączania do bazy danych – integracja z bazą danych

Wymiana danych pomiędzy interfejsem użytkownika (przeglądarka WWW), a bazą danych będzie odbywać się przy pomocy modułu PHP. Moduł ten jest odpowiedzialny za walidację danych, ich przetwarzanie oraz dwukierunkową wymianę z bazą danych. System bazodanowy aplikacji będzie oparty o bazę danych MySQL w wersji 5.2 lub wyższej uruchomiony pod systemem operacyjnym Linux z zainstalowanym serwerem HTTP Apache 1.3.x lub wyższą z obsługą PHP w wersji 4.3.x lub wyższej. Komunikacja użytkownika z aplikacją będzie odbywać się przy pomocy przeglądarki stron.

### 3.2.5. Projekt zabezpieczeń na poziomie aplikacji

Głównym zabezpieczeniem systemu będzie login pracownika i hasło. Hasła będą przechowywane w oddzielnej bazie. Przechowywane hasło będzie zhaszowane.

## 4. Implementacja systemu baz danych

Do zaimplementowania bazy danych użyto zestawu aplikacji Xampp + phpMyAdmin. Wersja silnika bazy danych: 10.1.38-MariaDB

### 4.1. Tworzenie tabel i definiowanie ograniczeń

Elementy bazy danych takie jak Tablice i definiowanie ograniczeń zostały stworzone przy pomocy narzędzi dostępnych w pakiecie phpMyAdmin.

W poniższym przykładzie ukazano sposób w jaki zostały stworzone tabele „klient” oraz „adres” oraz jak zaimplementowano ograniczenie dla tabeli „klient”.

```
CREATE TABLE `klient` (  
  `NrKlienta` int(10) NOT NULL,  
  `ImieKlienta` varchar(20) NOT NULL,  
  `ImieDrugie` varchar(20) DEFAULT NULL,  
  `NazwiskoKlienta` varchar(40) NOT NULL,  
  `DataUrodzenia` date DEFAULT NULL,  
  `Obywatelstwo` varchar(40) NOT NULL,  
  `Email` varchar(40) DEFAULT NULL,  
  `FkAdres` int(10) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
ALTER TABLE `klient`  
  ADD CONSTRAINT `klient_ibfk_1` FOREIGN KEY (`FkAdres`) REFERENCES `adres` (`IdAdres`);  
  
CREATE TABLE `adres` (  
  `IdAdres` int(10) NOT NULL,  
  `Miejscowosc` varchar(40) NOT NULL,  
  `KodPocztowy` char(6) NOT NULL,  
  `Ulica` varchar(20) DEFAULT NULL,  
  `NrDomu` varchar(10) NOT NULL,  
  `NrMieszkania` varchar(10) DEFAULT NULL,  
  `NrTelefonu` char(12) DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

## 4.2. Implementacja mechanizmów przetwarzania danych

Niektóre mechanizmy przetwarzania danych takie jak indeksy oraz sekwencje zostały zaimplementowane przy pomocy gotowych narzędzi pakietu phpMyAdmin już podczas tworzenia tabel. Poniżej przedstawiono indeksy oraz sekwencje dla tabeli „rezerwacja”.

```
ALTER TABLE `rezerwacja`  
  ADD PRIMARY KEY (`NrRezerwacji`),  
  ADD KEY `PocRezIdx` (`PoczatekRezerwacji`),  
  ADD KEY `KonRezIdx` (`KoniecRezerwacji`),  
  ADD KEY `FkKlient` (`FkKlient`);  
  
ALTER TABLE `rezerwacja`  
  MODIFY `NrRezerwacji` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;
```

Mechanizmy przetwarzania danych takie jak widoki, wyzwalacze oraz procedury składowane utworzone zostały przy pomocy czystego języka SQL. Poniżej przedstawiono po jednym przykładzie każdego z nich.

```
CREATE VIEW `wolne_ten_miesiac` AS  
  select `t1`.`NrPokoju` AS `NrPokoju`,  
         `t1`.`LiczbaMiejsc` AS `LiczbaMiejsc`,  
         `t1`.`Pietro` AS `Pietro`,  
         `t1`.`TypPokoju` AS `TypPokoju`,  
         `t1`.`CenaPokoju` AS `CenaPokoju`  
  from ((`pokoj` `t1` join `rezerwacja` `t2`)  
        join `rezerwacja_pokoj` `t3`)  
 where ((`t1`.`NrPokoju` = `t3`.`NrPokoju`)  
        and (`t2`.`NrRezerwacji` = `t3`.`NrRezerwacji`)  
        and (month(`t2`.`PoczatekRezerwacji`) <> month(curdate()))  
        and (`t1`.`DostepnyTeraz` = '1')) ;
```

Dodatkowo: Platnosc\_pokoj, hotel\_dane, historia\_rezerwacji, goscie\_teraz.

```
CREATE TRIGGER `wstaw_date` BEFORE INSERT ON `rezerwacja` FOR EACH ROW BEGIN  
  SET NEW.DataOperacji=NOW();  
END
```

Dodatkowo: ustaw\_dostepny(tabela pokoj), ustaw\_niedostepny(tabela pokoj)



```

CREATE DEFINER='root'@'localhost' PROCEDURE `generuj_rachunek` (IN `id_klient` INT) NO SQL
BEGIN
DECLARE x, y, z INT DEFAULT 0;
SELECT (pobyt.KoniecZameldowania - pobyt.PoczatekZameldowania),
       (pobyt.KoniecZameldowania - pobyt.PoczatekZameldowania) *
       pokoj.CenaPokoju, pobyt.NrPobytu
       INTO x,y,z
FROM pobyt, pokoj
WHERE pobyt.FkKlient=id_klient
      AND pobyt.FkPokoj=pokoj.NrPokoju;

INSERT INTO rachunek VALUES (null, y, x,id_klient);
INSERT INTO rachunek_pobyt VALUES(z, LAST_INSERT_ID());

END$$

```

Dodatkowo: dostępne\_pokoje, wykonaj\_rezerwacje, znajdz\_klienta

### 4.3. Implementacja uprawnień i innych zabezpieczeń

Dostęp do bazy danych będzie miał jedynie administrator oraz pracownik.

Uprawnienia administratora:

```

1 • GRANT ALL PRIVILEGES ON *.* TO 'Administrator'@'localhost'
2   IDENTIFIED BY PASSWORD '*05A8A1A73083F816772592F13D11C8AA5CCD9681' WITH GRANT OPTION;
3
4 • GRANT ALL PRIVILEGES ON `obsługa_hotelu`.* TO 'Administrator'@'localhost' WITH GRANT OPTION;
5
6 • GRANT SELECT, INSERT (FkNazwa, LiczbaMiejsc, DostepnyTeraz, Pietro, TypPokoju, CenaPokoju),
7   UPDATE (FkNazwa, LiczbaMiejsc, DostepnyTeraz, Pietro, TypPokoju, CenaPokoju),
8   DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER
9   ON `obsługa_hotelu`.`pokoj` TO 'Administrator'@'localhost' WITH GRANT OPTION;
10
11 • GRANT SELECT, INSERT (FkAdres, HotelNazwa, Email),
12   UPDATE (FkAdres, HotelNazwa, Email),
13   DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER
14   ON `obsługa_hotelu`.`hotel` TO 'Administrator'@'localhost' WITH GRANT OPTION;
15
16 • GRANT SELECT, INSERT (CzasPobytu, Oplata, FkKlient),
17   UPDATE (CzasPobytu, Oplata, FkKlient),
18   DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER
19   ON `obsługa_hotelu`.`rachunek` TO 'Administrator'@'localhost' WITH GRANT OPTION;
20
21 • GRANT SELECT, INSERT (KodPocztowy, Miejscowosc, NrTelefonu, NrDomu, Ulica, NrMieszkania),
22   UPDATE (KodPocztowy, Miejscowosc, NrTelefonu, NrDomu, Ulica, NrMieszkania),
23   DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER
24   ON `obsługa_hotelu`.`adres` TO 'Administrator'@'localhost' WITH GRANT OPTION;

```

```

26 • GRANT SELECT, INSERT (KoniecRezerwacji, PoczątekRezerwacji, FkKlient, DataOperacji),
27 UPDATE (KoniecRezerwacji, PoczątekRezerwacji, FkKlient, DataOperacji),
28 DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER
29 ON `obsługa_hotelu`.`rezerwacja` TO 'Administrator'@'localhost' WITH GRANT OPTION;
30
31 • GRANT SELECT, INSERT (NazwiskoKlienta, ImieKlienta, Obywatelstwo, ImieDrugie, FkAdres, DataUrodzenia, Email),
32 UPDATE (NazwiskoKlienta, ImieKlienta, Obywatelstwo, ImieDrugie, FkAdres, DataUrodzenia, Email),
33 DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER
34 ON `obsługa_hotelu`.`klient` TO 'Administrator'@'localhost' WITH GRANT OPTION;
35
36 • GRANT SELECT, INSERT (KoniecZameldowania, PoczątekZameldowania, FkRezerwacja, FkPokoje, FkKlient),
37 UPDATE (KoniecZameldowania, PoczątekZameldowania, FkRezerwacja, FkPokoje, FkKlient),
38 DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER ON `obsługa_hotelu`.`pobyt`
39 TO 'Administrator'@'localhost' WITH GRANT OPTION;

```

## Uprawnienia pracownika:

```

1 • GRANT SELECT, INSERT, UPDATE, DELETE, FILE ON *.* TO 'pracownik'@'localhost'
2 IDENTIFIED BY PASSWORD '*05A8A1A73083F816772592F13D11C8AA5CCD9681';
3
4 • GRANT ALL PRIVILEGES ON `obsługa_hotelu`.* TO 'pracownik'@'localhost';
5
6 • GRANT SELECT ON `obsługa_hotelu`.`hotel` TO 'pracownik'@'localhost';
7
8 • GRANT SELECT,
9 INSERT (KoniecZameldowania, PoczątekZameldowania, FkRezerwacja, FkPokoje, FkKlient),
10 UPDATE (KoniecZameldowania, PoczątekZameldowania, FkRezerwacja, FkPokoje, FkKlient),
11 DELETE ON `obsługa_hotelu`.`pobyt` TO 'pracownik'@'localhost';
12
13 • GRANT SELECT,
14 INSERT (NazwiskoKlienta, ImieKlienta, Obywatelstwo, ImieDrugie, FkAdres, DataUrodzenia, Email),
15 UPDATE (NazwiskoKlienta, ImieKlienta, Obywatelstwo, ImieDrugie, FkAdres, DataUrodzenia, Email),
16 DELETE ON `obsługa_hotelu`.`klient` TO 'pracownik'@'localhost';
17
18 • GRANT SELECT,
19 INSERT (KodPocztowy, Miejscowosc, NrTelefonu, NrDomu, Ulica, NrMieszkania),
20 UPDATE (KodPocztowy, Miejscowosc, NrTelefonu, NrDomu, Ulica, NrMieszkania),
21 DELETE ON `obsługa_hotelu`.`adres` TO 'pracownik'@'localhost';
22
23 • GRANT SELECT,
24 INSERT (KoniecRezerwacji, PoczątekRezerwacji, FkKlient, DataOperacji),
25 UPDATE (KoniecRezerwacji, PoczątekRezerwacji, FkKlient, DataOperacji),
26 DELETE ON `obsługa_hotelu`.`rezerwacja` TO 'pracownik'@'localhost';
27
28 • GRANT SELECT,
29 INSERT (FkNazwa, LiczbaMiejsc, DostepnyTeraz, Pietro, TypPokoju, CenaPokoju),
30 UPDATE (FkNazwa, LiczbaMiejsc, DostepnyTeraz, Pietro, TypPokoju, CenaPokoju)
31 ON `obsługa_hotelu`.`pokoje` TO 'pracownik'@'localhost';
32
33 • GRANT SELECT,
34 INSERT (CzasPobytu, Oplata, FkKlient),
35 UPDATE (CzasPobytu, Oplata, FkKlient),
36 DELETE
37 ON `obsługa_hotelu`.`rachunek` TO 'pracownik'@'localhost';

```

## 4.4. Testowanie bazy danych na przykładowych danych

Wstawianie danych:

Poprawne:

✓ Wstawionych rekordów: 1.  
Id wstawionego wiersza: 4

```
INSERT INTO `klient` (`NrKlienta`, `ImieKlienta`, `ImieDrugie`, `NazwiskoKlienta`,  
`DataUrodzenia`, `Obywatelstwo`, `Email`, `FkAdres`) VALUES (NULL, 'Przemysław', 'Stanisław',  
'Kwiatkowski', '1992-05-21', 'Polska', 'kwiatkowski@wp.pl', '1');
```

✓ Wstawionych rekordów: 1.  
Id wstawionego wiersza: 2

```
INSERT INTO `adres` (`IdAdres`, `Miejscowosc`, `KodPocztowy`, `Ulica`, `NrDomu`, `NrMieszkania`,  
`NrTelefonu`) VALUES (NULL, 'Poznan', '12-130', 'Kwiatkowa', '5', '2', NULL);
```

✓ Wstawionych rekordów: 1.

```
INSERT INTO `pokoj` (`NrPokoju`, `LiczbaMiejsc`, `Pietro`, `TypPokoju`, `CenaPokoju`,  
`DostepnyTeraz`, `FkNazwa`) VALUES ('6', '2', '2', 'dwuosobowy', '100', '1', '1');
```

Niepoprawne:

```
1 INSERT INTO `klient` (`NrKlienta`, `ImieKlienta`, `ImieDrugie`, `NazwiskoKlienta`,  
`DataUrodzenia`, `Obywatelstwo`, `Email`, `FkAdres`) VALUES (NULL, 'Piotr', NULL,  
'Bałtroczyk', NULL, 'Polska', NULL, NULL);
```

#1048 - Kolumna 'FkAdres' nie może być null

```
1 INSERT INTO `rezerwacja` (`NrRezerwacji`, `PoczątekRezerwacji`, `KoniecRezerwacji`,  
`DataOperacji`, `FkKlient`) VALUES (NULL, '2019-05-16', '2019-05-25', NULL, '33');
```

#1452 - Cannot add or update a child row: a foreign key constraint fails (`obsługa\_hotelu`.`rezerwacja`, CONSTRAINT `rezerwacja\_ibfk\_1` FOREIGN KEY (`FkKlient`) REFERENCES `klient` (`NrKlienta`))

-brak klienta o nr 33 w tabeli klient.

```
1 INSERT INTO `rachunek` (`NrRachunku`, `Oplata`, `CzasPobytu`, `FkKlient`) VALUES (NULL,  
'200', '3', '50');
```

#1452 - Cannot add or update a child row: a foreign key constraint fails (`obsługa\_hotelu`.`rachunek`, CONSTRAINT `rachunek\_ibfk\_1` FOREIGN KEY (`FkKlient`) REFERENCES `klient` (`NrKlienta`))

- brak klienta o nr 50 w tabeli klient

Modyfikacja danych:

Poprawna:

✓ Zmodyfikowanych rekordów: 1.

```
UPDATE `rachunek` SET `Oplata` = '500' WHERE `rachunek`.`NrRachunku` = 4;
```

✓ Zmodyfikowanych rekordów: 1.

```
UPDATE `pokoj` SET `CenaPokoju` = '200' WHERE `pokoj`.`NrPokoju` = 6;
```

✓ Zmodyfikowanych rekordów: 1.

```
UPDATE `klient` SET `ImieDrugie` = 'Sebastian', `Email` = 'wladzimirzomerski@wp.pl' WHERE `klient`.`NrKlienta` = 5;
```

Niepoprawna:

❗ Zmodyfikowanych rekordów: 0.

Warning: #1265 Data truncated for column 'KodPocztowy' at row 1

```
UPDATE `adres` SET `KodPocztowy` = '98-2144' WHERE `adres`.`IdAdres` = 3;
```

- zbyt wiele znaków w kodzie pocztowym

❗ Zmodyfikowanych rekordów: 0.

Warning: #1265 Data truncated for column 'KoniecRezerwacji' at row 1

```
UPDATE `rezerwacja` SET `KoniecRezerwacji` = 'NULL' WHERE `rezerwacja`.`NrRezerwacji` = 4;
```

- Pole KoniecRezerwacji jest wymagane

✓ Zmodyfikowanych rekordów: 0. (Wykonanie zapytania trwało 0,0000 sekund(y).)

```
UPDATE `pokoj` SET `DostepnyTeraz` = '10' WHERE `pokoj`.`NrPokoju` = 3
```

⚠ Warning: #1265 Data truncated for column 'DostepnyTeraz' at row 1

- nie wprowadzono zmian. Pole DostepnyTeraz jest typu bool.

Usuwanie danych:

Poprawne:

✓ Zmodyfikowanych rekordów: 1. (Wykonanie zapytania trwało 0,0000 sekund(y).)

```
DELETE FROM adres WHERE adres.IdAdres = 22
```

✓ Zmodyfikowanych rekordów: 1. (Wykonanie zapytania trwało 0,0000 sekund(y).)

```
DELETE FROM klient WHERE klient.NrKlienta=17
```

✓ Zmodyfikowanych rekordów: 0. (Wykonanie zapytania trwało 0,0000 sekund(y).)

```
DELETE FROM rezerwacja WHERE NrRezerwacji=5
```

Niepoprawne:

```
DELETE FROM klient  
WHERE klient.NrKlienta=2
```

**MySQL zwrócił komunikat:** ?

```
#1451 - Cannot delete or update a parent row: a foreign key constraint fails  
(`obsługa_hotelu`.`rezerwacja`, CONSTRAINT `rezerwacja_ibfk_1` FOREIGN KEY (`FkKlient`)  
REFERENCES `klient` (`NrKlienta`))
```

- klient o nr 2 posiada rezerwacje i nie mogą zostać usunięte jego dane.

```
DELETE FROM adres  
WHERE IdAdres=1
```

**MySQL zwrócił komunikat:** ?

```
#1451 - Cannot delete or update a parent row: a foreign  
key constraint fails (`obsługa_hotelu`.`hotel`,  
CONSTRAINT `hotel_ibfk_1` FOREIGN KEY (`FkAdres`)  
REFERENCES `adres` (`IdAdres`))
```

- Do adresu o id=2 są przypisani klienci

```
DELETE FROM pokoj  
WHERE NrPokoju=1
```

**MySQL zwrócił komunikat:** ?

```
#1451 - Cannot delete or update a parent row: a foreign  
key constraint fails (`obsługa_hotelu`.`pobyt`,  
CONSTRAINT `pobyt_ibfk_2` FOREIGN KEY (`FkPokoj`)  
REFERENCES `pokoj` (`NrPokoju`))
```

- W pokoju o nr 2 istnieją pobyty

## Działanie Mechanizmów przetwarzania:

Triger „ustaw\_dostepny”:

Przed operacją update:

NrPokoju	LiczbaMiejsc	Pietro	TypPokoju	CenaPokoju	DostepnyTeraz	FkNazwa
3	2	1	dwuosobowy	100	0	1

Po operacji update:

```
UPDATE `pobyt` SET `KoniecZameldowania` = '2019-05-08' WHERE `pobyt`.`NrPobytu` = 16;
```

NrPokoju	LiczbaMiejsc	Pietro	TypPokoju	CenaPokoju	DostepnyTeraz	FkNazwa
3	2	1	dwuosobowy	100	1	1

-po wymeldowaniu klienta z danego pokoju zmienia on swój status na dostępny.

```
INSERT INTO `pobyt` (`NrPobytu`, `PoczątekZameldowania`, `KoniecZameldowania`, `FkPokoj`,  
`FkKlient`, `FkRezerwacja`) VALUES (NULL, '2019-05-08', '2019-05-08', '3', '12', '6');
```

NrPokoju	LiczbaMiejsc	Pietro	TypPokoju	CenaPokoju	DostepnyTeraz	FkNazwa
3	2	1	dwuosobowy	100	0	1

-Po wykonaniu operacji insert wraz z data wymeldowania pokój widnieje w bazie jako niedostępny, tymczasem nie jest zajęty.

Widok „historia\_rezerwacji”:

✓ Pokazano wiersze 0 - ... (Wykonanie zapytania trwało 0,0000 sekund(y).)				
SELECT * FROM `historia_rezerwacji`				
PoczątekRezerwacji	KoniecRezerwacji	ImieKlienta	ImieDrugie	NazwiskoKlienta
2019-04-25	2019-04-30	Tomasz	NULL	Kubat
2019-04-25	0000-00-00	Przemysław	Stanisław	Kwiatkowski
2019-04-28	2019-05-05	Patryk	NULL	Dziatkiewicz
2019-04-28	2019-05-05	Janusz	NULL	Szczepaniak
2019-05-20	2019-05-25	Janusz	NULL	Szczepaniak
2019-05-11	2019-05-13	Janusz	NULL	Tracz
2019-05-16	2019-05-25	Józef	NULL	Poniatowski

Procedura „generuj\_rachunek”:

Wiersz w tabeli „pokoj”:

NrPokoju	LiczbaMiejsc	Pietro	TypPokoju	CenaPokoju	DostepnyTeraz	FkNazwa
4	2	1	dwuosobowy	100	0	1

Wiersz w tabeli „pobyt”:

NrPobytu	PoczatekZameldowania	KoniecZameldowania	FkPokoj	FkKlient	FkRezerwacja
18	2019-05-05	2019-05-08	4	16	7

Wiersz w tabeli „klient”:

NrKlienta	ImieKlienta	ImieDrugie	NazwiskoKlienta	DataUrodzenia	Obywatelstwo	Email	FkAdres
16	Adam	NULL	Nawałka	NULL	Polska	NULL	21

Wywołanie procedury:

```
✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0,0000 sekund(y).)

call generuj_rachunek(16)
```

Efekt w postaci wygenerowanego wiersza w tabeli „rachunek”

NrRachunku	Oplata	CzasPobytu	FkKlient
7	300	3	16

W przypadku kiedy klient ma aktywny pobyt , wywołanie procedury się nie powiedzie, ponieważ nie możliwe jest obliczenie kosztu pobytu bez znanej daty zakończenia.

```
call generuj_rachunek(6)

MySQL zwrócił komunikat: ?
#1048 - Kolumna 'Oplata' nie może być null
```

**Wszystkie powyższe operacje przeprowadzano na bazie danych której tablice miały średnią ilość rekordów w przedziale 50-150.**

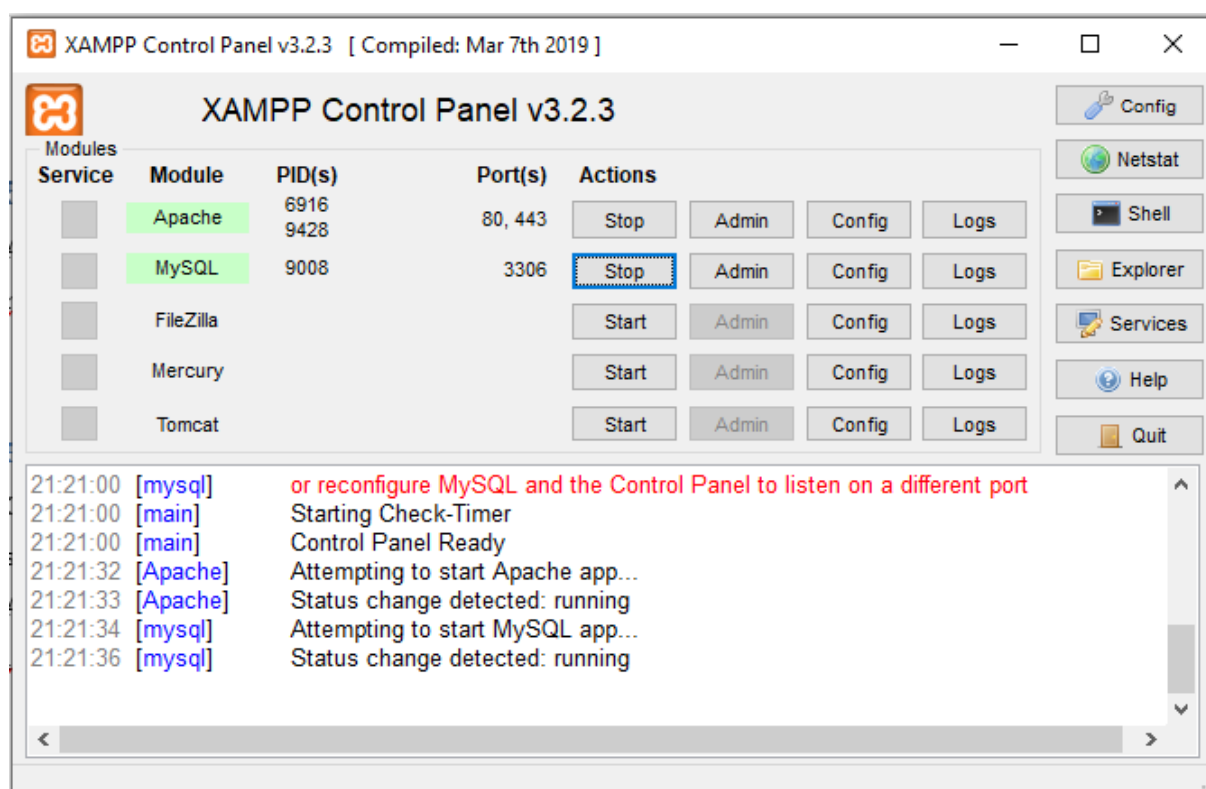


## 5. Implementacja i testy aplikacji

Aplikacja webowa została napisana w całości przy pomocy języka PHP w wersji 5.2.13 release oraz HTML w wersji 5.

### 5.1. Instalacja i konfigurowanie systemu

Całość aplikacji zbudowana w oparciu o aplikację XAMPP dzięki której uruchomiono serwer Apache aby w prosty sposób zaimplementować aplikację webową.



Rys.1a. Konfiguracja systemu w pakiecie XAMPP.

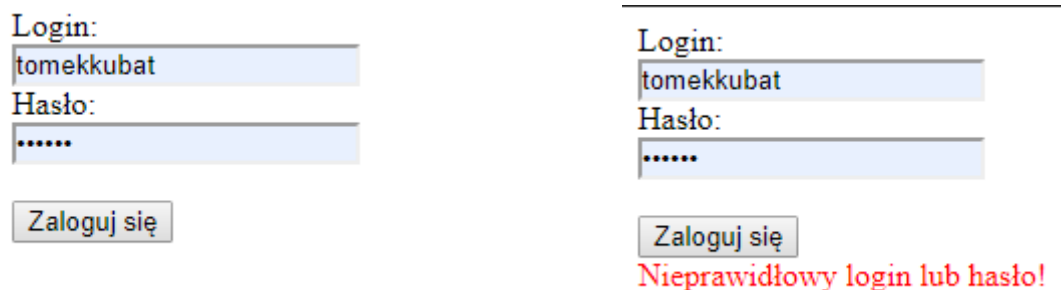
Aby rozpocząć tworzenie aplikacji konieczne jest uruchomienie serwera Apache aby w ten sposób móc uruchamiać naszą aplikację webową na stronie internetowej.

Aplikacja webowa jest dostępna jedynie na lokalnej maszynie pod adresem localhost/obslugahotelu. Uruchomienie pakietu XAMPP nie jest konieczne z punktu widzenia samej implementacji strony, jednak aby móc się do niej odwołać i w trakcie tworzenia aplikacji debugować konieczne jest jego uruchomienie. Dzięki temu wystarczy, że umieścimy domyślnie pliki w folderze XAMPP/htdocs/ aby w ten sposób uzyskać dostęp poprzez przeglądarkę internetową do naszej aplikacji bazodanowej.



## 5.2. Instrukcja użytkowania aplikacji

Po wpisaniu odpowiedniego adresu jesteśmy proszeni o wpisanie loginu i hasła. W przypadku podania błędnego loginu lub hasła, ekran logowania zgłosi błąd i możliwa będzie kolejna próba logowania.



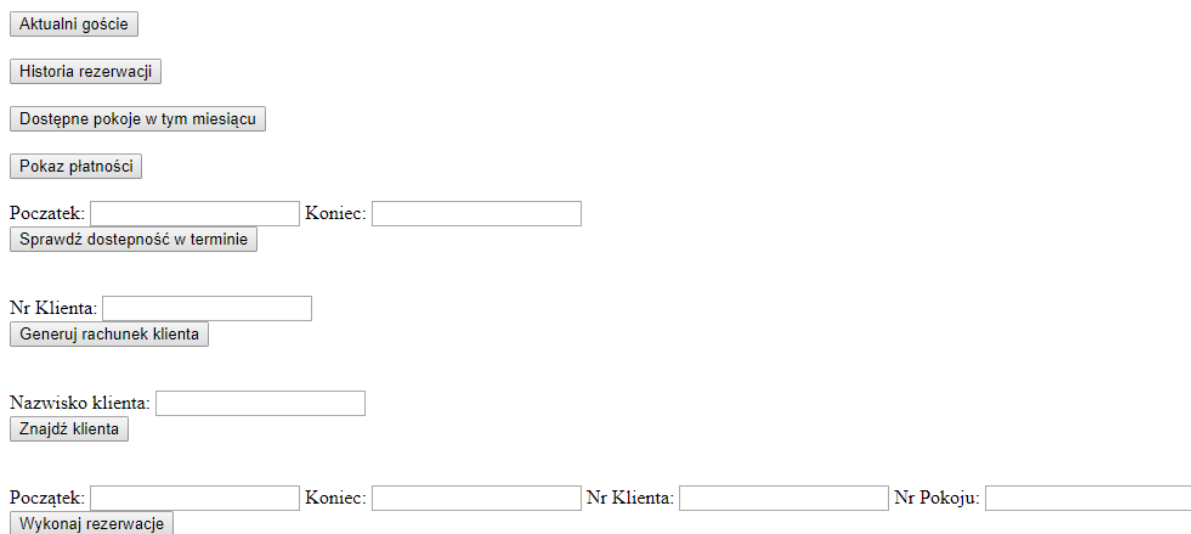
Two screenshots of the login interface. The left screenshot shows the login form with fields for 'Login:' (containing 'tomekkubat') and 'Hasło:' (containing '\*\*\*\*\*'), and a 'Zaloguj się' button. The right screenshot shows the same form with an additional red error message 'Nieprawidłowy login lub hasło!' below the 'Zaloguj się' button.

Rys.2 Ekran logowania.

Po zalogowaniu ukaże nam się ekran z kilkunastoma funkcjonalnościami służącymi do obsługi hotelowej bazy danych. Nazwy poszczególnych przycisków jasno obrazują działanie poszczególnych elementów.

Witaj adam! [ [Wyloguj się!](#) ]

### Hotel - panel pracownika



The screenshot displays the 'Hotel - panel pracownika' interface. It includes several buttons: 'Aktualni goście', 'Historia rezerwacji', 'Dostępne pokoje w tym miesiącu', and 'Pokaż płatności'. Below these are input fields for 'Początek:' and 'Koniec:', followed by a 'Sprawdź dostępność w terminie' button. Further down are fields for 'Nr Klienta:' and 'Nazwisko klienta:', each with a corresponding button ('Generuj rachunek klienta' and 'Znajdź klienta'). At the bottom, there are fields for 'Początek:', 'Koniec:', 'Nr Klienta:', and 'Nr Pokoju:', followed by a 'Wykonaj rezerwacje' button.

Rys.2 Wygląd strony głównej.

Po kliknięciu przycisku **Goście teraz** naszym oczom ukaże się lista aktualnie przebywających gości w hotelu lub informacja o ich braku jeśli w tej chwili nie mamy aktywnych pobyków.

Po wybraniu opcji **Historia rezerwacji** naszym oczom ukaże się lista wszystkich rezerwacji jakie kiedykolwiek miały miejsce w naszym hotelu.

Po wybraniu opcji **Dostępne pokoje w tym miesiącu** otrzymamy listę pokoi dostępnych do końca bieżącego miesiąca, lub aplikacja wyświetli odpowiedni komunikat w przypadku braku wolnych pokoi.

Po wybraniu opcji **Pokaż płatności** otrzymamy widok wszystkich płatności jakie zostały wygenerowane (wygenerowany faktury) oraz informacje na temat użytkowników, pobyków odbytych oraz należności jakie są zobowiązani zapłacić nam klienci naszego hotelu.

Po wypełnieniu odpowiednich pól i wybraniu opcji **Pokaż dostępne w tym terminie** ukażą nam się dostępne pokoje w wybranym przedziale czasowym.

Po wybraniu opcji **Generuj rachunek** oraz podaniu numeru klienta w bazie danych zostanie automatycznie wygenerowany rachunek dla tego klienta.

Po wybraniu opcji **Znajdź klienta** jesteśmy w szybki sposób odnaleźć informacje o kliencie znajdującym się w bazie danych po podaniu jego nazwiska. Wyświetlone zostaną jego dane osobowe.

Po wybraniu przycisku **Wykonaj rezerwację** i wypełnieniu odpowiednich pól zostanie dodana nowa rezerwacja do bazy danych uwzględniająca podany przedział czasowy, numer klienta oraz numer pokoju w który został zarezerwowany.

Administrator posiada dwie dodatkowe opcje.

Jedną z nich jest dodawanie nowego pracownika po podaniu jego loginu i hasła w odpowiednich miejscach oraz kliknięcia w przycisk **Dodaj pracownika**.

Po podaniu loginu w odpowiednim polu oraz kliknięciu w przycisk **Usuń pracownika** administrator ma możliwość usunięcia pracownika o podanym loginie z bazy danych przez co nie będzie on w stanie się zalogować do aplikacji.

### 5.3. Testowanie opracowanych funkcji systemu

Po kliknięciu w przycisk **Historia rezerwacji** ukazuje nam się widok na którym widnieją wszystkie rezerwacje umieszczone w bazie danych.

#### Hotel - Historia rezerwacji

<a href="#">Powrót</a>			
PoczątekRezerwacji	KoniecRezerwacji	ImięKlienta	NazwiskoKlienta
2019-04-25	2019-04-30	Tomasz	Kubat
2019-04-25	0000-00-00	Przemysław	Kwiatkowski
2019-04-28	2019-05-05	Patryk	Dziatkiewicz
2019-04-28	2019-05-05	Janusz	Szczepaniak
2019-05-20	2019-05-25	Janusz	Szczepaniak
2019-05-11	2019-05-13	Janusz	Tracz
2019-05-16	2019-05-25	Józef	Poniatowski
2019-12-15	2019-12-15	Jakub	Błaszczykowski
2019-12-15	2019-12-15	Zbigniew	Boniek
2019-12-15	2019-12-15	Robert	Lewandowski
2019-12-15	2019-12-15	Tomasz	Hajto
2019-12-15	2019-12-15	Adam	Nawałka
2019-12-15	2019-12-15	Piotr	Bartosz
2019-12-15	2019-12-15	Włodzimierz	Zomerski
2019-12-15	2019-12-15	Włodzimierz	Zomerski
2019-12-15	2019-12-15	Radosław	Zomerski
2019-12-15	2019-12-15	Włodzimierz	Urbański
2019-12-15	2019-12-15	Radosław	Klich
2019-05-15	2019-12-10	Izydor	Chich
2019-05-15	2019-12-13	Danuta	Ziobłowski
2019-06-15	2019-12-12	Agnieszka	Stachurski
2019-09-15	2019-12-19	Sylvia	Stroski

Podobnie w przypadku przycisków **Aktualni goście**, **Dostępne pokoje w tym miesiącu**, **Pokaż płatności**. Każdy z tych przycisków odpowiada konkretnej procedurze składowanej w bazie danych i wyświetla odpowiednie dane lub ich brak gdy takie nie występują. Użycie tych przycisków zawsze jest poprawne (w przypadku działającej bazy danych i możliwemu dostępowi do niej).

Po wybraniu przycisku **Aktualni goście** wyświetli się nam lista osób aktualnie przebywających w hotelu. W przypadku poniżej otrzymujemy informację o braku takich osób, ponieważ nikt w momencie przeprowadzania testu nie posiadał aktualnie pobytu.

#### Hotel - Aktualni goście

[Powrót](#)

Aktualnie w hotelu nie przebywa żaden gość!

*Rys.3c. Aktualni goście- komunikat o braku aktualnych pobytów.*

W poniższym przypadku otrzymujemy komunikat o braku wolnych pokoi w tym miesiącu, wynika to z faktu, iż mimo, że w momencie testów nikt nie przebywał w naszym hotelu to znajdowało się bardzo dużo rezerwacji na koniec miesiąca, stąd wszystkie pokoje były zajęte.

## Hotel - Wolne w tym miesiącu

Powrót

brak wolnych pokoi w tym miesiącu!

*Rys.3d. Komunikat o braku wolnych pokoi w bieżącym miesiącu.*

Na poniższym przykładzie widzimy efekt działania przycisku **Pokaż płatności**

## Hotel - Płatności

Powrót

NrKlienta	Początek zameldowania	Koniec zameldowania	Czas pobytu	Oplata
3	2019-05-05	2019-05-07	2	200
3	2019-05-05	2019-05-07	2	500
12	2019-05-08	2019-05-08	0	0
12	2019-05-08	2019-05-08	0	0
12	2019-05-08	2019-05-08	0	0
16	2019-05-05	2019-05-08	3	300
6	2019-05-04	2019-05-17	13	1300
6	2019-05-04	2019-05-17	13	1300
6	2019-05-04	2019-05-17	13	1300

*Rys.3e. Widok płatności.*

Zaimplementowano również kilka funkcji które wchodzi w interakcje z użytkownikiem aplikacji. Może on wpisywać dowolne dane i dzięki nim wyszukiwać odpowiednie informacje znajdujące się w bazie danych.

Początek:  Koniec:

Rys.4 Miejsce wyszukiwania dostępnych pokoi w danym przedziale czasowym.

## Hotel - Dostępne w terminie

<input type="button" value="Powrót"/>				
Nr Pokoju	Liczba Miejsc	Piętro	Typ pokoju	Cena pokoju
1	2	1	dwuosobowy	100
3	2	1	dwuosobowy	100
2	2	1	dwuosobowy	100
4	2	1	dwuosobowy	100
5	2	1	dwuosobowy	100

Rys.5. Wyświetlane informacje o dostępnych pokojach.

W podobny sposób działają pozostałe funkcjonalności tj. **Generuj rachunek klienta, Znajdź klienta, wykonaj rezerwacje.**

## Hotel - Rachunek klienta

<input type="button" value="Powrót"/>				
Nr rachunku	Oplata	Czas Pobytu	FkKlient	
67	1300	13	6	

Rys.5a. Efekt działania funkcjonalności generującej rachunek.

## Hotel - Informacje o kliencie

[Powrót](#)

Nr Klienta	Imię	Drugie Imię	Nazwisko	Data Urodzenia	Obywatelstwo	Email
14	Robert		Lewandowski		Polska	

Rys.5b. Efekt działania funkcjonalności wyszukiwania klienta.

## Hotel - Rezerwacja

[Powrót](#)

Wykonano rezerwację!

Nr Rezerwacji	Początek Rezerwacji	Koniec Rezerwacji	Data Rezerwacji	Nr Klienta
102	2019-05-10	2019-05-15	2019-06-04	4

Rys.5c. Efekt poprawnie wykonanej rezerwacji – informacja zwrotna.

W przypadku podania błędnych bądź niekompletnych danych, każda z tych funkcjonalności wyświetli odpowiedni komunikat.

Początek:  Koniec:  Nr Klienta:  Nr Pokoju:   
[Wykonaj rezerwację](#)

Rys.6. Przykład podania niekompletnych danych.

## Hotel - Rezerwacja

[Powrót](#)

Nie udało się wykonać rezerwacji!

Rys.7. Efekt działania, informacja o błędzie.

# Hotel - Informacje o kliencie

[Powrót](#)

brak klienta!

*Rys.7a. Komunikat w przypadku wyszukiwania nieistniejącej osoby w bazie.*

# Hotel - Rachunek klienta

[Powrót](#)

błędne dane!

*Rys.7b. Komunikat w przypadku błędnie podanego numeru klienta*

## 5.4. Omówienie wybranych rozwiązań programistycznych

### 5.4.1. Implementacja interfejsu dostępu do bazy danych

Cała mechanika oraz wygląd strony została zaprojektowana przy użyciu języka HTML oraz PHP. Połączenie z bazą danych jest wykonywane przy pomocy polecenia znajdującego się w składni języka PHP a mianowicie **mysqli(args)**. Operacje na bazie danych są wykonywane poprzez polecenie **query(args)**.

```

<?php
    require_once "connect.php";

    $polaczenie = @new mysqli($host, $db_user, $db_password, $db_name);

    $poczatek = $_POST['poczatek'];
    $koniec = $_POST['koniec'];
    $klient = $_POST['klient'];
    $pokoj = $_POST['pokoj'];

    if ($polaczenie->connect_errno!=0)
    {
        echo "Error: ".$polaczenie->connect_errno;
    }
    else
    {
        if ($rezultat = @$polaczenie->query(sprintf("call wykonaj_rezerwacje
        ('%s','%s',%d,%d);", $poczatek, $koniec, $klient, $pokoj)))
        {
            if($rezultat->num_rows>0)
            {

```

Rys.8. Przykładowe kod ilustrujący uzyskanie dostępu do bazy.

#### 5.4.2. Implementacja wybranych funkcjonalności systemu

Podczas użytkowania aplikacji korzystamy z interfejsu sesji, dzięki czemu możliwa jest każdorazowa weryfikacja stanu logowania danego użytkownika.

```

<?php

    session_start();

    if (!isset($_SESSION['zalogowany']))
    {
        header('Location: index.php');
        exit();
    }

?>

```

Rys.9. Fragment kodu implementujący sesje w pliku.

Dzięki użyciu mechanizmu sesji niemożliwe jest użycie przyciska wstecz w przypadku wylogowania się aby w ten sposób powrócić nieuprawniony do aplikacji webowej.



### 5.4.3. Implementacja mechanizmów bezpieczeństwa

Istnieją 2 rodzaje użytkowników dla aplikacji webowej: Administrator oraz pracownik. Administrator posiada dodatkową opcję zarządzania pracownikami której nie posiada zwykły pracownik.

Witaj tomekkubat! [ [Wyloguj się!](#) ]

## Hotel - panel administracyjny

Aktualni goście

Historia rezerwacji

Dostępne pokoje w tym miesiącu

Pokaz płatności

Początek:  Koniec:

Sprawdź dostępność w terminie

Nr Klienta:

Generuj rachunek klienta

Nazwisko klienta:

Znajdź klienta

Początek:  Koniec:  Nr Klienta:  Nr Pokoju:

Wykonaj rezerwację

Nowy login tomekkubat Nowe hasło ..... Dodaj użytkownika

login  Usun użytkownika

Rys.10. Panel administratora(czerwone opcje dostępne tylko w tym panelu).

Dzięki tej opcji administrator ma możliwość zarządzania pracownikami.

Dodatkowo każdy z użytkowników posiada możliwość wylogowania się z aplikacji co powoduje natychmiastowe zakończenie sesji i powrót do ekranu logowania.

Mechanizm sesji zapobiega dostępowi osób nieuprawnionych do aplikacji webowej

Wykonywane funkcjonalności zaimplementowane w aplikacji są odporne na **SQL injections** dzięki walidacji danych użytych do konstruowania zapytań do bazy danych.

## 6. Podsumowanie i wnioski

Stworzenie pełnoprawnego systemu bazodanowego nie jest zagadnieniem prostym. Istnieje wiele różnych kombinacji pakietów/frameworków/środowisk w których można tworzyć systemy bazodanowe. W tym projekcie skupione się przede wszystkim na technologii MySQL oraz PHP wraz z HTML. Stworzony system bazodanowy nie jest wolny od wad. Z pewnością można byłoby go udoskonalić. Niestety było to moje pierwsze zderzenie z systemami bazodanowymi wskutek czego brakowało mi doświadczenia aby wszystko zaimplementować zgodnie ze sztuką i przyjętymi dogmatami. Subiektywnie oceniam, że dobór narzędzi MySQL oraz PHP jest jednym z prostszych wykorzystania przy tego typu projektach, szczególnie dla osób początkujących. Środowiska te nie wymagają instalacji wielu skomplikowanych programów. Tak naprawdę skorzystano w trakcie projektowania projektu z 3 aplikacji: VisualParadigm, Notepad++ oraz XAMPP. Dobór tych aplikacji znacząco wpłynął na wygodę tworzenia całego systemu.

## Literatura

[\*https://stackoverflow.com/\*](https://stackoverflow.com/)

[\*https://miroslawzelent.pl/kurs-html/\*](https://miroslawzelent.pl/kurs-html/)

[\*https://miroslawzelent.pl/kurs-php/\*](https://miroslawzelent.pl/kurs-php/)

[\*https://miroslawzelent.pl/kurs-mysql/\*](https://miroslawzelent.pl/kurs-mysql/)

[\*https://www.techotopia.com/index.php/MySQL\\_Essentials\*](https://www.techotopia.com/index.php/MySQL_Essentials)

[\*https://www.techotopia.com/index.php/PHP\\_Essentials\*](https://www.techotopia.com/index.php/PHP_Essentials)