

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

PROJEKT Z BAZ DANYCH

Temat projektu

System obsługi hotelu.

Termin zajęć: Czwartek, 13:15–15:00

AUTOR/AUTORZY:

Kamil Kuczaj

Indeks: 218478

E-mail: 218478@student.pwr.edu.pl

PROWADZĄCY ZAJĘCIA:

dr inż. Roman Ptak, W4/K9

Wrocław, 2017 r.

Spis treści:

1. Wstęp	4
1.1. Cel projektu	4
1.2. Zakres projektu	4
2. Analiza wymagań	4
2.1. Opis działania i schemat logiczny systemu	4
2.2. Wymagania funkcjonalne	4
2.3. Wymagania niefunkcjonalne	5
2.3.1. Wykorzystywane technologie i narzędzia	5
2.3.2. Wymagania dotyczące rozmiaru bazy danych	5
2.3.3. Wymagania dotyczące bezpieczeństwa systemu	5
2.4. Przyjęte założenia projektowe	6
3. Projekt systemu	6
3.1. Projekt bazy danych	6
3.1.1. Analiza rzeczywistości i uproszczony model konceptualny	6
3.1.2. Model logiczny i normalizacja i model fizyczny oraz ograniczenia integralności danych	7
3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych	7
3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych	8
3.2. Projekt aplikacji użytkownika	9
3.2.1. Architektura aplikacji i diagramy projektowe	9
3.2.2. Interfejs graficzny i struktura menu	10
3.2.3. Projekt wybranych funkcji systemu	11
3.2.4. Metoda podłączania do bazy danych – integracja z bazą danych	11
3.2.5. Projekt zabezpieczeń na poziomie aplikacji	11
4. Implementacja systemu baz danych	12
4.1. Tworzenie tabel i definiowanie ograniczeń	12
4.2. Implementacja mechanizmów przetwarzania danych	13
4.3. Implementacja uprawnień i innych zabezpieczeń	13
4.4. Testowanie bazy danych na przykładowych danych	16
5. Implementacja i testy aplikacji	20
5.1. Instalacja i konfigurowanie systemu	20
5.2. Instrukcja użytkowania aplikacji	20
Panel administratora	28

5.3. Testowanie opracowanych funkcji systemu.....	32
5.4. Omówienie wybranych rozwiązań programistycznych.....	34
5.4.1. Implementacja interfejsu dostępu do bazy danych	34
5.4.2. Implementacja wybranych funkcjonalności systemu	35
5.4.3. Implementacja mechanizmów bezpieczeństwa	37
6. Podsumowanie i wnioski.....	38
Literatura	39
Spis ilustracji.....	39

1. Wstęp

1.1. Cel projektu

Budowa aplikacji webowej współpracującej z bazą danych, w celu automatyzacji rezerwacji dla małego hotelu.

1.2. Zakres projektu

Budowa bazy danych dla małej wielkości hotelu oraz obsługi rezerwacji przez Internet.

2. Analiza wymagań

2.1. Opis działania i schemat logiczny systemu

Aplikacja webowa umożliwia użytkownikowi utworzenie konta, a następnie rezerwację pobytu w hotelu. Na kompleks składają się:

- 20 pokoiów jednoosobowych
 - 10 z łazienką
 - 10 z łazienką współdzieloną na korytarzu
- 20 pokoiów dwuosobowych:
 - 10 z łózkami typu King
 - 10 z dwoma łózkami typu Single
- 4 pokoiów czteroosobowych:
 - 2 z łózkami małżeńskimi
 - 2 z jednym łóżkiem małżeńskim i dwoma łózkami typu Single
- 6 apartamentów dwuosobowych:
 - 2 z Jacuzzi
 - 2 z Sauną
 - 2 z Jacuzzi oraz Sauną
- Usługi:
 - Fryzjer (płatność przy wymeldowaniu)
 - Wypożyczalnia sprzętu narciarskiego
 - Wypożyczalnia skuterów śnieżnych
 - Masażysta
 - Basen

Wszystko odbywa się bez ingerencji obsługi hotelu. Rezerwujący może wybrać termin pobytu oraz pokój. Rezerwujący ma podgląd do dokonanej rezerwacji z możliwością anulowania do 14 dni przed planowanym pobytem. Może również wysłać zapytanie do hotelu korzystając z formularza kontaktowego.

2.2. Wymagania funkcjonalne

1. Klient może utworzyć konto w aplikacji webowej, a następnie:
 - a. dokonać rezerwacji na kalendarzu na stronie
2. Klient może przeglądać swoje rezerwacje i poznać:
 - a. informację o zarezerwowanym pobycie
 - i. termin
 - ii. rodzaj pokoju
 - b. zakupione usługi:

- i. informację, kiedy wykonano usługę
 - ii. kwota usługi
 - iii. informację o usłudze
3. Klient może anulować swoją rezerwację
 - a. anulować na maksymalnie 14 dni przed planowanym terminie
4. Kontakt z hotelem (poprzez bazę danych)
 - a. Dane kontaktowe (mail, telefon, adres itp.)
 - b. Pod danymi kontaktowymi pole tekstowe, do którego można wpisać pytanie a następnie wysłać zapytanie do recepcji hotelu.
5. Obsługa hotelowa może:
 - a. dokonać meldunku gościa
 - b. naliczyć karę dla pokoju wraz z wpisaniem informacji o tej karze
 - c. obciążyć rezerwację kosztem usługi dodatkowej dla gości pokoju wraz z wpisaniem informacji o tej usłudze
 - d. anulować rezerwację

2.3. Wymagania niefunkcjonalne

2.3.1. Wykorzystywane technologie i narzędzia

Proponowany system zarządzania bazą danych to MySQL, ze względu na szerokie grono użytkowników, dobrze napisaną dokumentację oraz dużą ilość for internetowych.

Aplikacja webowa będzie korzystać z serwera i mikro serwisu Python Flask, komunikując się z bazą danych poprzez wywoływania procedur. W wyświetleniu informacji na stronie internetowej pomogą HTML5, CSS3, JavaScript rozszerzony o bibliotekę jQuery oraz Ajax. Zdecydowano się na taki krok, ponieważ te technologie implementują jedne z największych stron internetowych takich jak: *linkedin.com*, *pinterest.com*.

2.3.2. Wymagania dotyczące rozmiaru bazy danych

Maksymalna liczba użytkowników = 182 500.

$$50 \cdot 1 \cdot 365 \cdot 10 = 182\,500$$

$$\text{ilość_pokojów} \cdot \text{osoba_wynajmująca} \cdot \text{dni_w_roku} \cdot \text{szacunkowy_czas_działania_w_latach}$$

Maksymalna liczba rezerwacji:

$$50 \cdot 365 = 18\,250$$

$$\text{ilość_pokojów} \cdot \text{dni_w_roku}$$

2.3.3. Wymagania dotyczące bezpieczeństwa systemu

- Hasła użytkowników przechowywane w formie haszów generowanych i sprawdzanych algorytmem MD-5.
- Zmniejszenie szansy błędu użytkownika przy wprowadzaniu hasła podczas rejestracji (hasło należy wpisać w dwóch polach przy rejestracji)
- Weryfikacja przeciwko SQL injection.
- Weryfikacja na podstawie wprowadzania danych do tabeli (początek rezerwacji nie może być później niż koniec jak również nie wcześniej niż dzisiejsza data)
- Zastosowanie mechanizmu sesji użytkownika
- Sygnalizacja błędów w bazie danych odpowiednimi kodami wyjątków
- Backupy bazy danych.

2.4. Przyjęte założenia projektowe

- dostarczenie wygodnego mechanizmu do zarządzania rezerwacją dla klienta
- zautomatyzowanie procesu rezerwacji i redukcja liczby obowiązków obsługi hotelowej
- umożliwienie podglądu rezerwacji i użytkowników przez personel

3. Projekt systemu

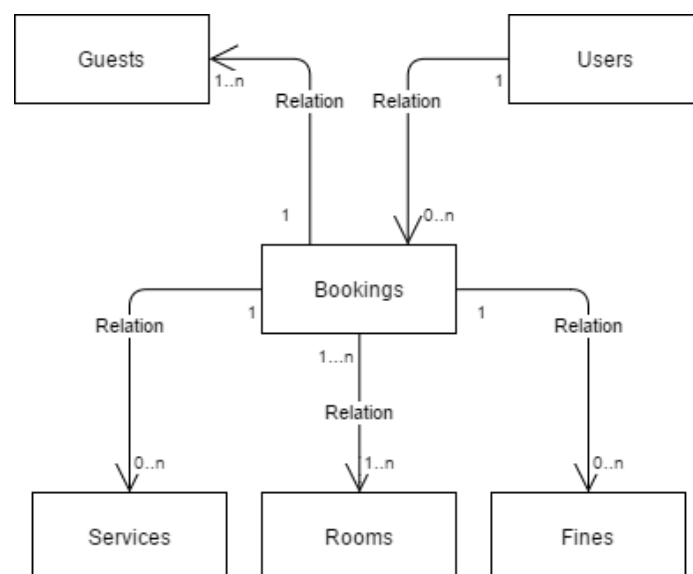
Ze względu na chęć późniejszego udostępnienia programu poprzez publiczne repozytorium zdecydowano się na angielskie nazwy w tabelach bazy danych oraz aplikacji.

3.1. Projekt bazy danych

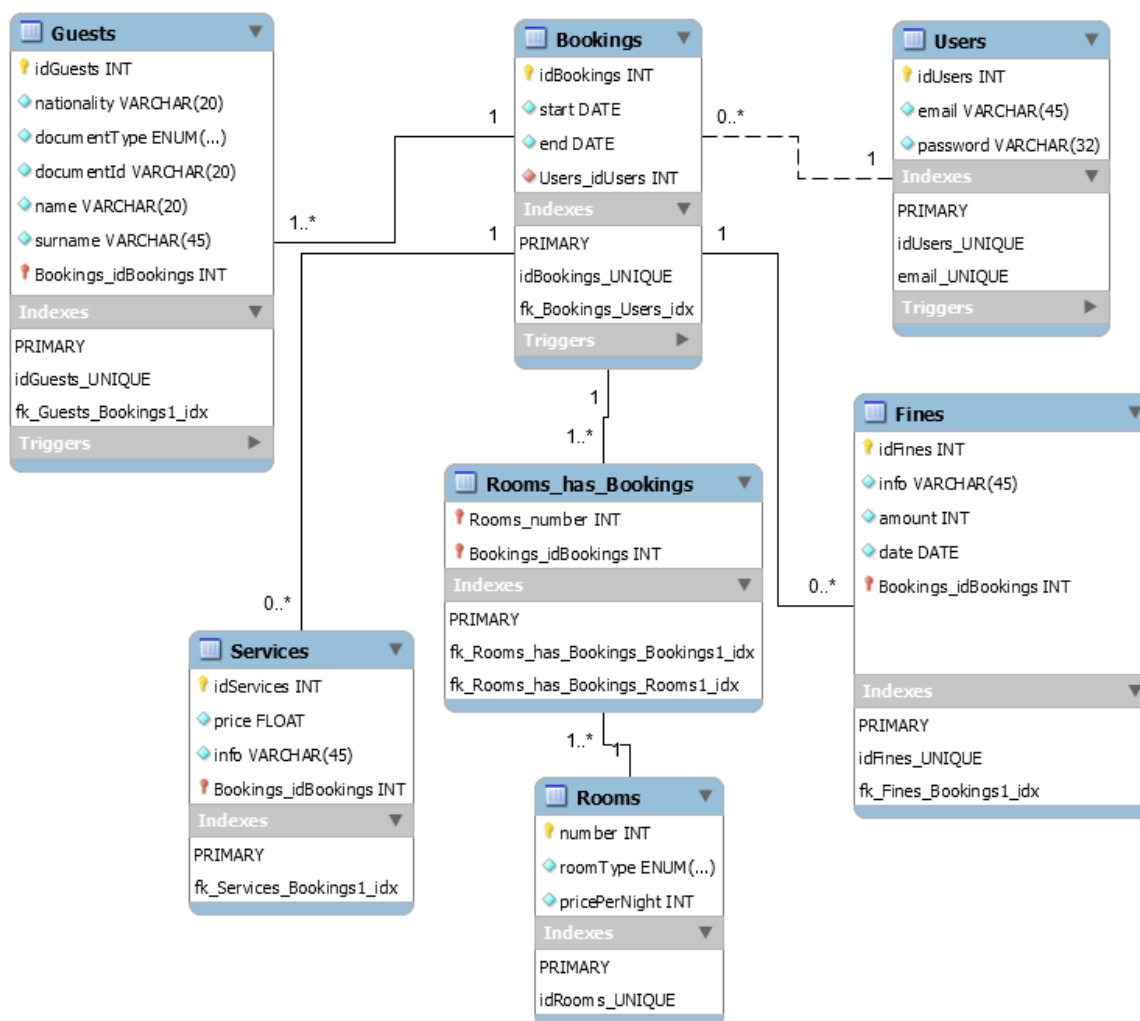
Użytkownik wchodzi na stronę internetowej. Strona umożliwi utworzenie konta lub zalogowanie się na swój profil (korzystając z założonego uprzednio konta). Po weryfikacji poprawności hasła, przeglądarka wczytuje nową stronę z trzema zakładkami:

1. Rezerwacja
 - a. Kalendarz, na którym widoczne są wolne terminy (tzn. jeden kalendarz dla wszystkich pokoi). Jeżeli pokój jest zajęty w wybranym terminie, to pokój znika z opcji wyboru.
 - b. Klikamy w dzień zameldowania a następnie na dzień wymeldowania – wybrane dni zostają zaznaczone kolorem.
 - c. Wybieramy pokój, klikamy zatwierdź i przechodzimy do strony z możliwością zaznaczenia opcji obiadów oraz podsumowania rezerwacji
2. Obsługa hotelu
 - a. Podgląd rezerwacji dokonanych przez wszystkich użytkowników
 - b. Możliwość wprowadzenia danych gości i przypisanie ich do rezerwacji
 - c. Pogląd gości widniejących w bazie hotelu
 - d. Podgląd zarejestrowanych użytkowników z wykluczeniem wyświetlania hasła użytkownika

3.1.1. Analiza rzeczywistości i uproszczony model konceptualny



3.1.2. Model logiczny i normalizacja i model fizyczny oraz ograniczenia integralności danych



Rysunek 1 Model logiczny oraz model fizyczny

3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych

Ponieważ swoją funkcjonalność opieram o procedury składowane, model bazy danych posiada niewielką ilość widoków. To widoki udostępnione dla użytkowników:

1. Pokazanie wszystkich rezerwacji wraz z przypisanymi do nich pokojami

To połączenie dwóch tabel [tabela (wybrane pola tabeli)]:

- Bookings (idBookings, Users_idUsers, start, end)
- Rooms (number, roomType, pricePerNight)

idBookings	Users_idUsers	number	start	end	roomType	pricePerNight
...

2. Wyświetlenie typów pokoi

Zwróci wartości przechowywane w polu roomType w tabeli Rooms.

roomType	pricePerNight
...	...

3. Pokazanie ilości pokoi lub ilości rezerwacji

Zwróci ilość pokoi dostępnych w hotelu.

count(*)
...

Procedury:

Dostęp do bazy danych z poziomu strony internetowej będzie odbywał się poprzez wywołanie procedur, np.:

1. utwórz rezerwację w okresie *start* do *end* na *pokój* dla *użytkownika*
2. utwórz gościa i przypisz do rezerwacji
3. utwórz użytkownika
4. usuń rezerwację
5. zwróć wszystkie rezerwacje
6. zwróć wszystkie pokoje
7. zwróć wszystkie rezerwacje
8. zwróć wszystkich gości
9. pokaż wolne pokoje w zadanym okresie czasu
10. pokaż zajęte pokoje w zadanym okresie czasu
11. sprawdź, czy hasło jest poprawne dla danego użytkownika
12. zwróć pierwszy wolny numer pokoju o podanym typie (np. *pojedynczy z dzieloną łazienką*)
13. zwróć rezerwacje dla danego użytkownika

Indeksy:

Indeksy w tabelach są posortowane rosnąco. Każdy główny i obcy klucz jest indeksem.

Będzie istniała funkcja, która sama uzupełni tabelę pokoje, gdyż nie przewiduje się rozwoju placówki w ciągu najbliższych lat. Tym samym nie zmienią się też pokoje.

3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych

Trzy typy użytkowników oraz ich uprawnienia:

- admin (administrator)
 - pełne uprawnienia
- user (użytkownik – gość hotelu)
 - możliwość wstawiania i usuwania rekordów w tabelach:
 - Users
 - Bookings
- staff (pracownik hotelu)

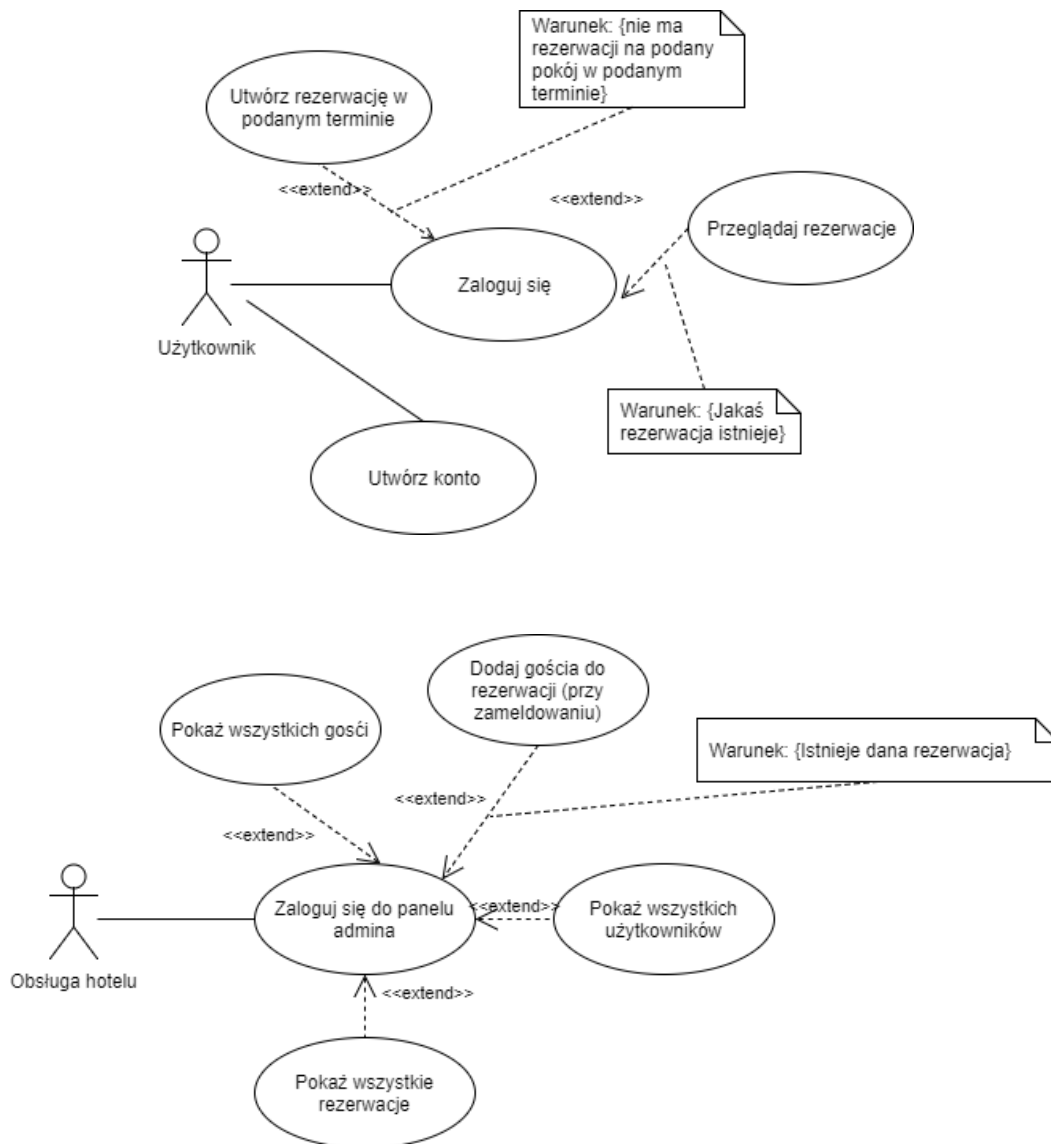
- o możliwość wstawiania i usuwania rekordów w tabelach:
 - Guests
 - Bookings
 - Fines
 - Services

Wyzwalacze (ang. *Triggers*):

- walidacja adresu email przy użyciu wyrażenia regularnego
 - o $\wedge[A-Z0-9_ \% -] + @ [A-Z0-9 -] + \wedge [A-Z] \{2,4\} \$$
- Zmiana pierwszej litery imienia i nazwiska oraz zapisanie jej wielką literą przy uzupełnianiu tabeli Guests
- Sprawdzenie czy rezerwowana data jest odpowiednia, tj. czy koniec rezerwacji przypada po początku rezerwacji oraz czy rezerwujemy okres w przyszłości, aby wykluczyć możliwość błędu

3.2. Projekt aplikacji użytkownika

3.2.1. Architektura aplikacji i diagramy projektowe



Rysunek 2 Diagramy przypadków użycia dla użytkownika oraz pracownika hotelu

3.2.2. Interfejs graficzny i struktura menu



Zaloguj

e-mail
hasło

Utwórz konto

Rysunek 3 Strona główna hotelu

Zaznacz date pobytu

FotografiaDlaCiekawych.pl

2012

Styczeń

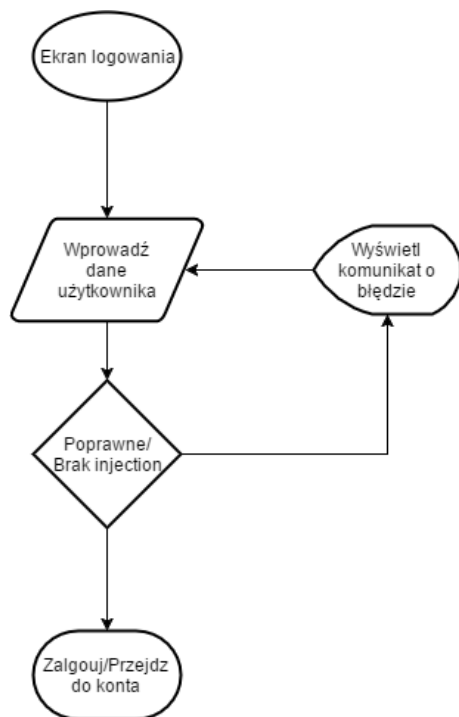
Poniedziałek	Wtorek	Środa	Czwartek	Piątek	Sobota	Niedziela
						1 Nowy Rok
2	3	4	5	6 Trzech Króli	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21 Dzień Babci	22 Dzień Dziadka
23	24	25	26	27	28	29
30	31					

Lista dostępnych pokoiów

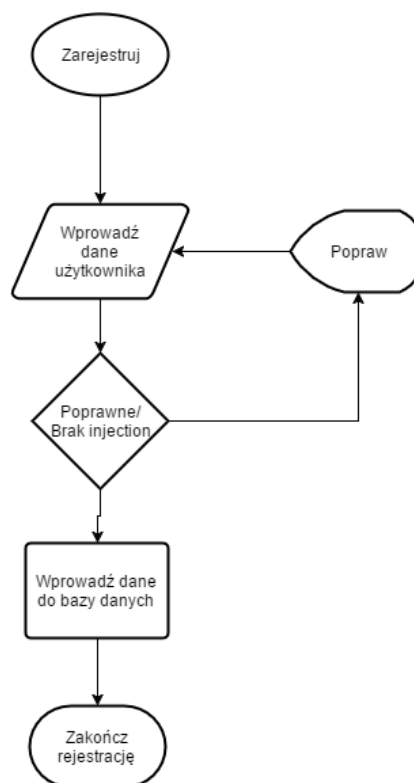
Przejdź do wypełnienia
danych gości i płatności

Rysunek 4 Strona rezerwacji

3.2.3. Projekt wybranych funkcji systemu



Rysunek 6 Logowanie



Rysunek 5 Rejestracja

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

Podłączenie przez API Flask oraz Flask_mysql i port 5000. Zapytania do bazy danych wprowadzane przy użyciu jQuery.

3.2.5. Projekt zabezpieczeń na poziomie aplikacji

- Każdy *input* użytkownika zostaje sprawdzany przeciwko zjawisku SQL Injection.
- Weryfikacja poprawności emaila (np. czy zgadza się z *wyrażeniem regularnym*: `^*@*$`).
- Hashowanie hasła algorytmem MD5.
- Weryfikacja złożoności hasła.
- Otwarty jeden port (5000).
- Mechanizm sesji będzie zabezpieczał przed niechcianym dostępem

4. Implementacja systemu baz danych

Implementacja i testy bazy danych w wybranym systemie zarządzania bazą danych.

4.1. Tworzenie tabel i definiowanie ograniczeń

Przykładowe kody MySQL tworzenia niektórych tabel:

```
-- Table `HotelDatabase`.`Users`
```

```
CREATE TABLE IF NOT EXISTS `HotelDatabase`.`Users` (  
  `idUsers` INT UNSIGNED NOT NULL AUTO_INCREMENT,  
  `username` VARCHAR(20) NOT NULL,  
  `email` VARCHAR(45) NOT NULL,  
  `password` VARCHAR(32) NOT NULL,  
  PRIMARY KEY (`idUsers`),  
  UNIQUE INDEX `idUsers_UNIQUE` (`idUsers` ASC),  
  UNIQUE INDEX `email_UNIQUE` (`email` ASC),  
  UNIQUE INDEX `username_UNIQUE` (`username` ASC))  
ENGINE = InnoDB;
```

```
-- Table `HotelDatabase`.`Bookings`
```

```
CREATE TABLE IF NOT EXISTS `HotelDatabase`.`Bookings` (  
  `idBookings` INT NOT NULL AUTO_INCREMENT,  
  `start` DATE NOT NULL,  
  `end` DATE NOT NULL,  
  `Users_idUsers` INT UNSIGNED NOT NULL,  
  PRIMARY KEY (`idBookings`),  
  UNIQUE INDEX `idBookings_UNIQUE` (`idBookings` ASC),  
  INDEX `fk_Bookings_Users_idx` (`Users_idUsers` ASC),  
  CONSTRAINT `fk_Bookings_Users`  
    FOREIGN KEY (`Users_idUsers`)  
      REFERENCES `HotelDatabase`.`Users` (`idUsers`)  
      ON DELETE NO ACTION  
      ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
-- Table `HotelDatabase`.`Rooms`
```

```
CREATE TABLE IF NOT EXISTS `HotelDatabase`.`Rooms` (  
  `number` INT NOT NULL,  
  `roomType` ENUM('single_bath', 'single_shared', 'double_king', 'double_twobeds', 'four_twokings',  
  'four_kingandtwobeds', 'apartment_jacuzzi', 'apartment_sauna', 'apartment_jacuzzi_sauna') NOT NULL,  
  `pricePerNight` INT NOT NULL,  
  PRIMARY KEY (`number`),  
  UNIQUE INDEX `idRooms_UNIQUE` (`number` ASC))  
ENGINE = InnoDB;
```

```
-- Table `HotelDatabase`.`Rooms_has_Bookings`
```

```
CREATE TABLE IF NOT EXISTS `HotelDatabase`.`Rooms_has_Bookings` (  
  `Rooms_number` INT NOT NULL,  
  `Bookings_idBookings` INT NOT NULL,  
  PRIMARY KEY (`Rooms_number`, `Bookings_idBookings`),  
  INDEX `fk_Rooms_has_Bookings_Bookings1_idx` (`Bookings_idBookings` ASC),  
  INDEX `fk_Rooms_has_Bookings_Rooms1_idx` (`Rooms_number` ASC),  
  CONSTRAINT `fk_Rooms_has_Bookings_Rooms1`  
    FOREIGN KEY (`Rooms_number`)
```

```

REFERENCES `HotelDatabase`.`Rooms` (`number`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_Rooms_has_Bookings_Bookings1`
FOREIGN KEY (`Bookings_idBookings`)
REFERENCES `HotelDatabase`.`Bookings` (`idBookings`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

4.2. Implementacja mechanizmów przetwarzania danych

Pokazanie wszystkich rezerwacji wraz z przypisanymi do nich pokojami

```

-----
-- View `HotelDatabase`.`showFullBookings`
-----
DROP TABLE IF EXISTS `HotelDatabase`.`showFullBookings`;
USE `HotelDatabase`;
CREATE OR REPLACE VIEW `showFullBookings` AS
select
    bookings.idBookings,
    bookings.Users_idUsers,
    rooms.number,
    bookings.start,
    bookings.end,
    rooms.roomType,
    rooms.pricePerNight
from bookings
join Rooms_has_bookings on bookings.idBookings = rooms_has_bookings.bookings_idBookings
join rooms on rooms.number = rooms_has_bookings.rooms_number;

```

Wyświetlenie typów pokoiów wraz z ich cenami

```

-----
-- View `HotelDatabase`.`showRoomTypesAndPrices`
-----
DROP TABLE IF EXISTS `HotelDatabase`.`showRoomTypesAndPrices`;
USE `HotelDatabase`;
CREATE OR REPLACE VIEW `showRoomTypesAndPrices` AS
    select distinct(rooms.roomType), rooms.pricePerNight from rooms;
USE `HotelDatabase`;

```

4.3. Implementacja uprawnień i innych zabezpieczeń

Wyzwalacze dla tabeli Users:

```

DELIMITER $$
USE `HotelDatabase`$$
CREATE TRIGGER `trg_entity_email_insert` BEFORE INSERT ON `Users` FOR EACH ROW
BEGIN
    IF NOT (SELECT NEW.email REGEXP '[A-Z0-9_%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$') THEN
        -- bad data
        SIGNAL SQLSTATE '40000';
    END IF;
END$$

USE `HotelDatabase`$$
CREATE TRIGGER `trg_entity_email_update` BEFORE UPDATE ON `Users` FOR EACH ROW
BEGIN
    IF NOT (SELECT NEW.email REGEXP '[A-Z0-9_%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$') THEN
        -- bad data
        SIGNAL SQLSTATE '40000';
    END IF;
END$$

```

Wyzwalacze dla tabeli Bookings:

```
USE `HotelDatabase` $$
CREATE TRIGGER `trg_entity_datecheck_insert` BEFORE INSERT ON `Bookings` FOR EACH ROW
BEGIN
    IF NOT (SELECT NEW.start < New.end ) THEN
        -- bad date
        SIGNAL SQLSTATE '41000';
    END IF;
    IF NOT (SELECT NEW.start > Curdate()) THEN
        -- bad date
        SIGNAL SQLSTATE '41000';
    END IF;
END $$

USE `HotelDatabase` $$
CREATE DEFINER = CURRENT_USER TRIGGER `HotelDatabase`.`Guests_BEFORE_INSERT`
BEFORE INSERT ON `Guests` FOR EACH ROW
BEGIN
    SET NEW.name = CONCAT(UCASE(LEFT(New.name, 1)),
        LCASE(SUBSTRING(New.name, 2)));
    SET NEW.surname = CONCAT(UCASE(LEFT(New.surname, 1)),
        LCASE(SUBSTRING(New.surname, 2)));
END $$

DELIMITER ;
```

Przykładowe nadanie przywilejów użytkownikowi staff:

```
GRANT SELECT, INSERT, TRIGGER ON TABLE `HotelDatabase`.* TO 'staff';
GRANT DELETE, INSERT, SELECT, UPDATE ON TABLE `HotelDatabase`.`Fines` TO 'staff';
GRANT INSERT, SELECT, UPDATE, DELETE ON TABLE `HotelDatabase`.`Bookings` TO 'staff';
GRANT DELETE, INSERT, SELECT, UPDATE ON TABLE `HotelDatabase`.`Guests` TO 'staff';
GRANT SELECT ON TABLE `HotelDatabase`.`Rooms` TO 'staff';
GRANT DELETE, INSERT, SELECT, UPDATE ON TABLE `HotelDatabase`.`Services` TO 'staff';
GRANT DELETE, INSERT, SELECT, UPDATE ON TABLE `HotelDatabase`.`Users` TO 'staff';
```

Procedura składowana, która tworzy rezerwację:

```
-- -----
-- procedure sp_createBooking
-- -----

DELIMITER $$
USE `HotelDatabase` $$
CREATE DEFINER='root'@`localhost` PROCEDURE `sp_createBooking`(
    IN p_start date,
    IN p_end date,
    IN p_idUser int,
    IN p_roomNumber int
)
BEGIN
    if ( select exists (select 1 from showFullBookings where start = p_start and end = p_end and number =
p_roomNumber) ) THEN
        set @known_id = (select 1 from showFullBookings where start = p_start and end = p_end and
number = p_roomNumber);
    else
        set @known_id=0;
    end if;
    if (select exists (select 1 from Rooms_has_bookings where rooms_number=p_roomNumber and
Bookings_idBookings=@known_id)) then
        select 'Booking exists !!';
    else
        begin
            insert into `bookings` set `start`=p_start, `end`=p_end, `Users_idUsers`=p_idUser;
            SET @out_param = LAST_INSERT_ID();
            insert into `Rooms_has_bookings` set `Rooms_number`=p_roomNumber, `Bookings_idBookings`=
@out_param;
        end;
    end if;
END $$

DELIMITER ;
```

Procedura zwracająca wolne pokoje w zadanym okresie czasu

```
CREATE DEFINER='root'@'localhost' PROCEDURE `sp_showVacantRooms` (  
    IN p_start DATE,  
    IN p_end DATE  
)  
BEGIN  
    drop table if exists t1;  
    create table t1  
        select * from showFullBookings where ( start >= p_start and end <= p_end)  
            or ( start >= p_start and end >= p_end)  
            or ( start <= p_start and end >= p_end)  
            or ( start <= p_start and end >= p_start);  
  
    drop table if exists helper_VacantRoomsTable;  
    create table helper_VacantRoomsTable  
        SELECT Rooms.number, Rooms.roomType  
        FROM Rooms  
        LEFT JOIN t1  
            ON Rooms.number= t1.number  
        WHERE t1.number IS NULL  
        UNION  
        SELECT t1.number, t1.roomType  
        FROM t1  
        LEFT JOIN Rooms  
            ON Rooms.number = t1.number  
        WHERE Rooms.number IS NULL;  
  
    select distinct(helper_VacantRoomsTable.roomType) from helper_VacantRoomsTable;  
END
```

Procedura tworząca nowego gościa:

```
CREATE DEFINER='root'@'localhost' PROCEDURE `sp_createGuest`(  
    IN p_name VARCHAR(20),  
    IN p_surname VARCHAR(45),  
    IN p_nationality VARCHAR(20),  
    IN p_documentType VARCHAR(20),  
    IN p_documentId VARCHAR(20),  
    IN p_bookingsId INT  
)  
BEGIN  
    if ( select exists (select 1 from guests where name = p_name and surname = p_surname and  
        documentId = p_documentId and documentType = p_documentType and  
        Bookings_idBookings = p_bookingsId) ) THEN  
        select 'Guest exists !!';  
    ELSE  
        insert into Guests  
        (  
            nationality,  
            documentType,  
            documentId,  
            name,  
            surname,  
            Bookings_idBookings  
)  
        values  
        (  
            p_nationality,  
            p_documentType,  
            p_documentId,  
            p_name,  
            p_surname,  
            p_bookingsId  
)  
    );  
  
    END IF;  
END
```

4.4. Testowanie bazy danych na przykładowych danych

```
mysql> select * from showRoomTypesAndPrices;
```

roomType	pricePerNight
Jednoosobowy z lazienka	30
Jednoosobowy z lazienka dzielona	25
Dwuosobowy z lozkiem malzenskim	40
Dwuosobowy z dwoma oddzielnymi lozkami	50
Czterooosobowy z dwoma lozkami malzenskimi	65
Czterooosobowy - lozko malzenskie i dwa lozka pojedyncze	75
Apartament dwuosobowy z Jacuzzi	150
Apartament dwuosobowy z sauna	160
Apartament dwuosobowy z Jacuzzi i sauna	250

```
9 rows in set (0.01 sec)
```

Rysunek 7 Wyświetlenie wszystkich rodzajów pokoi

```
mysql> select * from showFullBookings;
```

idBookings	Users_idUsers	number	start	end	roomType	pricePerNight
21	2	300	2017-06-11	2017-06-22	Apartament dwuosobowy z Jacuzzi	150
22	2	301	2017-06-11	2017-06-23	Apartament dwuosobowy z Jacuzzi	150

Rysunek 8 Wyświetlenie wszystkich rezerwacji

```
mysql> call sp_getBookingsForUser(2);
```

idBookings	Users_idUsers	number	start	end	roomType	pricePerNight
21	2	300	2017-06-11	2017-06-22	Apartament dwuosobowy z Jacuzzi	150
22	2	301	2017-06-11	2017-06-23	Apartament dwuosobowy z Jacuzzi	150
23	2	209	2017-08-01	2018-08-14	Dwuosobowy z dwoma oddzielnymi lozkami	50

```
3 rows in set (0.00 sec)
```

Query OK, 0 rows affected (0.02 sec)

Rysunek 9 Wyświetlenie wszystkich rezerwacji dla danego użytkownika

24	1	209	2017-08-01	2018-08-14	Dwuosobowy z dwoma oddzielnymi lozkami	50
----	---	-----	------------	------------	--	----

```
4 rows in set (0.00 sec)
```

```
mysql> call sp_deleteBooking(24);
```

Query OK, 1 row affected (0.02 sec)

```
mysql> select * from showFullBookings;
```

idBookings	Users_idUsers	number	start	end	roomType	pricePerNight
21	2	300	2017-06-11	2017-06-22	Apartament dwuosobowy z Jacuzzi	150
22	2	301	2017-06-11	2017-06-23	Apartament dwuosobowy z Jacuzzi	150
23	2	209	2017-08-01	2018-08-14	Dwuosobowy z dwoma oddzielnymi lozkami	50

```
3 rows in set (0.00 sec)
```

Rysunek 10 Usunięcie rezerwacji


```
mysql> call sp_getAllUsers();
```

idUsers	email	password
1	admin@hotel-pod-galazka.com	58b4e38f66bcd546380845d6af27187
2	test@test.com	098f6bcd4621d373cade4e832627b4f6

```
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> call sp_createUser('adres@email.com','haslo');
Query OK, 1 row affected (0.03 sec)

mysql> call sp_getAllUsers();
```

idUsers	email	password
1	admin@hotel-pod-galazka.com	58b4e38f66bcd546380845d6af27187
2	test@test.com	098f6bcd4621d373cade4e832627b4f6
3	adres@email.com	haslo

```
3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

Rysunek 11 Dodanie nowego użytkownika oraz procedura zwracająca wszystkich użytkowników

```
mysql> call sp_getAllGuests();
```

idGuests	nationality	documentType	documentId	name	surname	Bookings_idBookings
1	San Escobar	Passport	abc12345	Kamil	Kowalski	21
2	Brzuch	Passport	124135	Noga	Reka	22

```
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> call sp_createGuest('Imie','Nazwisko','Polska','Paszport','EG1234',23);
ERROR 1265 (01000): Data truncated for column 'documentType' at row 1
mysql> call sp_createGuest('Imie','Nazwisko','Polska','Passport','EG1234',23);
Query OK, 1 row affected (0.02 sec)

mysql> call sp_getAllGuests();
```

idGuests	nationality	documentType	documentId	name	surname	Bookings_idBookings
1	San Escobar	Passport	abc12345	Kamil	Kowalski	21
2	Brzuch	Passport	124135	Noga	Reka	22
3	Polska	Passport	EG1234	Imie	Nazwisko	23

```
3 rows in set (0.00 sec)
```

Rysunek 12 Dodanie nowego gościa oraz procedura zwracająca wszystkich gości

```
mysql> select * from showFullBookings;
```

idBookings	Users_idUsers	number	start	end	roomType	pricePerNight
21	2	300	2017-06-11	2017-06-22	Apartament dwuosobowy z Jacuzzi	150
22	2	301	2017-06-11	2017-06-23	Apartament dwuosobowy z Jacuzzi	150

```
2 rows in set (0.00 sec)

mysql> call sp_createBooking('2017-08-01','2018-08-14',2,209);
Query OK, 1 row affected (0.01 sec)

mysql> select * from showFullBookings;
```

idBookings	Users_idUsers	number	start	end	roomType	pricePerNight
21	2	300	2017-06-11	2017-06-22	Apartament dwuosobowy z Jacuzzi	150
22	2	301	2017-06-11	2017-06-23	Apartament dwuosobowy z Jacuzzi	150
23	2	209	2017-08-01	2018-08-14	Dwuosobowy z dwoma oddzielnymi lozkami	50

```
3 rows in set (0.00 sec)
```

Rysunek 13 Utworzenie nowej rezerwacji

```
mysql> select * from showFullBookings;
+-----+-----+-----+-----+-----+-----+-----+
| idBookings | Users_idUsers | number | start | end | roomType | pricePerNight |
+-----+-----+-----+-----+-----+-----+-----+
| 21 | 2 | 300 | 2017-06-11 | 2017-06-22 | Apartament dwuosobowy z Jacuzzi | 150 |
| 22 | 2 | 301 | 2017-06-11 | 2017-06-23 | Apartament dwuosobowy z Jacuzzi | 150 |
| 23 | 2 | 209 | 2017-08-01 | 2018-08-14 | Dwuosobowy z dwoma oddzielnymi lozkami | 50 |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> call sp_showVacantRooms('2017-06-10','2017-06-23');
+-----+
| roomType |
+-----+
| Jednoosobowy z lazienka |
| Jednoosobowy z lazienka dzielona |
| Dwuosobowy z lozkiem malzenskim |
| Dwuosobowy z dwoma oddzielnymi lozkami |
| Czterooosobowy z dwoma lozkami malzenskimi |
| Czterooosobowy - lozko malzenskie i dwa lozka pojedyncze |
| Apartament dwuosobowy z sauna |
| Apartament dwuosobowy z Jacuzzi i sauna |
+-----+
8 rows in set (0.13 sec)

Query OK, 0 rows affected (0.14 sec)

mysql> call sp_showBusyRooms('2017-06-10','2017-06-23');
+-----+-----+-----+-----+-----+-----+-----+
| idBookings | Users_idUsers | number | start | end | roomType | pricePerNight |
+-----+-----+-----+-----+-----+-----+-----+
| 21 | 2 | 300 | 2017-06-11 | 2017-06-22 | Apartament dwuosobowy z Jacuzzi | 150 |
| 22 | 2 | 301 | 2017-06-11 | 2017-06-23 | Apartament dwuosobowy z Jacuzzi | 150 |
| 23 | 2 | 209 | 2017-08-01 | 2018-08-14 | Dwuosobowy z dwoma oddzielnymi lozkami | 50 |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

Rysunek 14 Wyświetlenie wszystkich rezerwacji oraz wolnych i zarezerwowanych pokoiów

Testy, które nie wymagały przedstawienia testów w postaci graficznej.

Umieszczenie rezerwacji w bazie danych:

insert into hoteldatabase.bookings value (1,'2017-08-08','2017-09-09',1,1);

Brak pokoju w encji Rooms powoduje wyświetlenie błędu:

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('hoteldatabase`.`bookings`, CONSTRAINT `fk_Bookings_Rooms1` FOREIGN KEY (`Rooms_idRooms`) REFERENCES `rooms` (`idRooms`) ON DELETE NO ACTION ON UPDATE NO ACTION)

Aktualizacja tabeli Users:

insert into hoteldatabase.users value (1, 'kkuczaj','abc@gmail.com','haselko');

Dwukrotna próba wpisania tych danych powoduje błąd:

Error Code: 1062. Duplicate entry 'abc@gmail.com' for key 'email_UNIQUE'

Sprawdzenie funkcji auto inkrementacji wierszy (POPRAWNIE):

insert into hoteldatabase.users value (NULL,'jnowak','j@gmail.com','haslo');

Błędny typ enum przy wprowadzaniu danych do tabeli Rooms:

insert into hoteldatabase.rooms value (NULL,1,'sgs',30);

Rezultat:

Error Code: 1265. Data truncated for column 'roomType' at row 1

Wprowadzenie rezerwacji:

insert into hoteldatabase.bookings value (NULL,'2017-8-01','2017-8-14',1);

Sprawdzenie wyświetlenia odpowiednich numerów pokoiów przypisanych do rezerwacji:

<pre>select Rooms_number from rooms_has_bookings where rooms_has_bookings.Bookings_idBookings = 1 LIMIT 0, 1000;</pre>
<p>Walidacja adresu email: <pre>insert into users value (NULL,'abcde','should@a.c','passwd');</pre></p> <p>Rezultat: <i>Error Code: 1644. Unhandled user-defined exception condition</i></p>
<p>Kapitalizacja imienia i nazwiska w tabeli Guests: <pre>insert into guests value (NULL,'pl','passport','123','kamil','kuczaj',1);</pre></p> <p>Rezultat: <i>Nazwisko zmienione na Kamil oraz nazwisko na Kuczaj</i></p>
<p>Sprawdzenie czy data startu i końca rezerwacji jest prawidłowa: <pre>-- end wcześniejszy niż start insert into bookings value (NULL,'2017-08-01','2008-08-01',1); -- start wcześniejszy niż obecna data insert into bookings value (NULL,'2016-08-01','2017-08-01',1);</pre></p> <p>Rezultat: <i>Error Code: 1644. Unhandled user-defined exception condition</i></p>

5. Implementacja i testy aplikacji

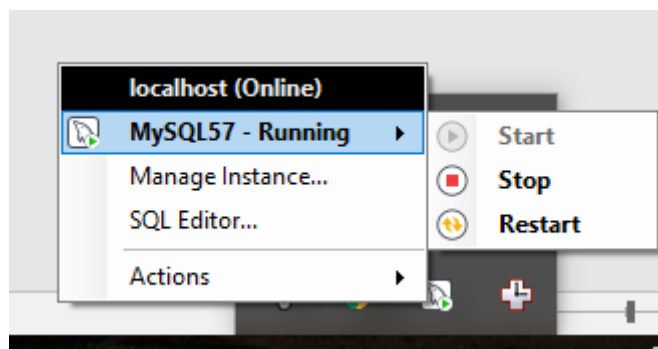
Wskutek tego, że praca była wykonywana w grupie jednoosobowej nie udało się zaimplementować niektórych funkcjonalności na czas. Do pełnej funkcjonalności brakuje możliwości anulowania rezerwacji oraz możliwości rezerwacji kilku pokoi w ramach jednej rezerwacji.

5.1. Instalacja i konfigurowanie systemu

Do instalacji niezbędne są:

1. serwer MySQL w wersji 5.7 (podana została taka wersja, ponieważ to na niej przeprowadzone zostały testy. Aplikacja najprawdopodobniej powinna zadziałać na starszych wersjach, jednak autor pracy nie gwarantuje tego)
2. Interpretator Python2.7 z zainstalowanymi bibliotekami Flask, Flask_MySQL oraz Flask_wtf

Aby uruchomić serwer należy włączyć bazę danych. W moim przypadku jest to upewnienie się, że serwis MySQL57 działa.



Następnie należy uruchomić aplikację poleceniem:

```
> python2.7 app.py
```

```
[Done] exited with code=1 in 9743.36 seconds

[Running] python "c:\Python27\Scripts\PythonApp\app.py"
c:\Python27\Scripts\PythonApp\app.py:5: ExtDeprecationWarning: Import
  from flask.ext.mysql import MySQL
C:\Python27\lib\site-packages\flask\exthook.py:106: ExtDeprecationWar
  .format(x=modname), ExtDeprecationWarning
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

5.2. Instrukcja użytkowania aplikacji

Aplikacja (strona internetowa) została napisana w przyjazny dla użytkownika sposób, który nie wymaga specjalnych instrukcji. Pomocą przy korzystaniu ze strony internetowej są wyświetlane komunikaty (zarówno w przypadku błędu jak i sukcesu) oraz podświetlanie aktywnych zakładek na niebiesko.



Rezerwacja i informacje o pobycie

Załącz konto już dzisiaj

Rysunek 15 Strona startowa. Zdjęcia to pokaz slajdów z estetyczną animacją.

Na stronie startowej użytkownik może podziwiać piękne hotelu poprzez pokaz slajdów zdjęć hotelu. Ma też do wyboru trzy zakładki. Pierwsza, na której się znajduje to *Strona Główna*. Następne dwie służą do kolejno: *logowania użytkownika* oraz *rejestracji użytkownika*.

Zarejestruj się

Email

Hasło

Powtórz hasło

Rejestracja

Rysunek 16 Formularz rejestracji

Po kliknięciu w formularz rejestracji, naszym oczom ukazuje się interfejs pozwalający na założenie konta na stronie, które następnie wymagane jest do zalogowania się i umożliwia nam dokonanie rezerwacji.

**Konto zostało
poprawnie
stworzone !**

abc@gmail.com

...

...

Rejestracja

Rysunek 17 Poprawna rejestracja

Poprawne wypełnienie formularza skutkuje wyświetleniem komunikatu tak jak na rysunku wyżej. W przypadku błędu aplikacja wyświetli komunikat, który zawierał będzie odpowiednie instrukcje.

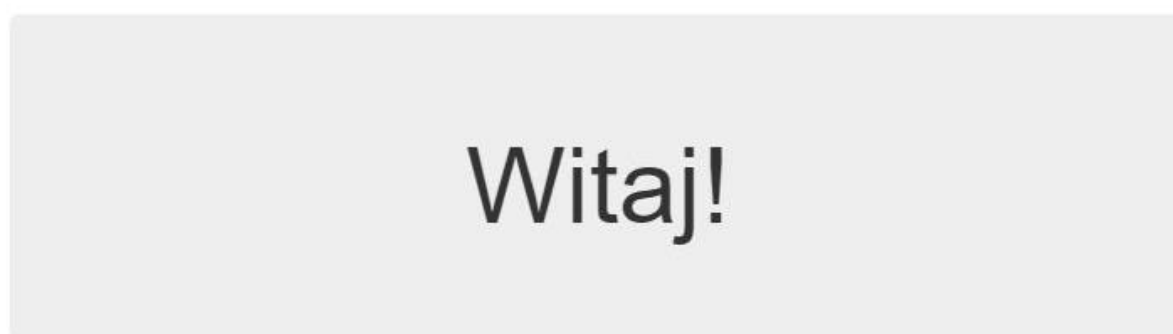


The login screen features a light gray background. At the top, the text 'Zaloguj się' is displayed in a large, bold, black font. Below this, there are two input fields: the first contains the email address 'abc@gmail.com', and the second contains three dots '...', indicating a password field. A blue button with the text 'Zaloguj' is positioned below the input fields.

Rysunek 18 Ekran logowania

Po kliknięciu na opcję logowania, wyświetla się interfejs do zalogowania użytkownika. Tak jak poprzednio, w przypadku wprowadzenia poprawnych danych, zostaniemy przeniesieni do strony domowej użytkownika, jednak w przypadku błędu zostaniemy poinformowani, czy np. aplikacja nie może odnaleźć podanego maila w bazie danych, lub czy też hasło nie zgadza się z poprawnym.

Aplikacja dopilnuje również, aby pola zostały wypełnione. W przeciwnym wypadku nie prześle danych do bazy.



Rysunek 19 Po podaniu poprawnych danych wyświetla się komunikat powitalny i jesteśmy zalogowani na nasze konto.

Ekran powitalny użytkownika. Wynik poprawnej operacji logowania się użytkownika.

Zarezerwuj pokój

The screenshot shows a reservation form for a hotel. At the top, there's a navigation bar with 'Hotel pod Gałązką', a blue 'Rezerwuj' button, and links for 'Moje rezerwacje' and 'Wyloguj'. Below this is a section titled 'Zarezerwuj pokój'. Inside this section is a light gray box containing a calendar for June 2017. The calendar has days of the week as headers (MON, TUE, WED, THU, FRI, SAT, SUN) and dates from 1 to 30. The date 12 is highlighted with a dark blue circle. Below the calendar are two input fields labeled 'Od:' and 'Do:'. At the bottom of the gray box is a blue button with the text 'Pokaż wolne pokoje na ten okres'.

Rysunek 20 Ekran rezerwacji pokoju. Aby wybrać obszar możemy skorzystać z pomocniczego kalendarza lub wpisać datę w formacie RRRR-MM-DD

Po poprawnym zalogowaniu możemy wybrać pierwszą zakładkę, która umożliwi nam złożenie rezerwacji. Najwygodniejszym sposobem wyboru daty pobytu to skorzystanie z kalendarza. Należy w tym celu kliknąć na datę rozpoczęcia pobytu a następnie datę zakończenia pobytu. Aplikacja wygodnie wpisze do formularza odpowiednie dane.

Użytkownik też może jednak wybrać metodę wprowadzenia danych ręcznie, zachowując odpowiedni format: RRRR-MM-DD, gdzie

- RRRR to roku, np. 2017 lub 1995
- MM to miesiąc, np. 08 (Sierpień) lub 01 (Styczeń)
- DD – to dzień, np. 24 (24 sierpnia 2017 roku)

Zarezerwuj pokój

The screenshot displays a booking interface for June 2017. At the top, a calendar header shows 'JUN 2017' with navigation arrows. Below the header, the days of the week are listed: MON, TUE, WED, THU, FRI, SAT, SUN. The calendar grid shows dates from 1 to 30. The 12th and 14th are highlighted with dark blue circles, indicating the selected stay dates. Below the calendar, there are two input fields: 'Od: 2017-06-12' and 'Do: 2017-06-14'. A blue button labeled 'Pokaż wolne pokoje na ten okres' is positioned below the date fields.

Rysunek 21 Poprawny wygląd danych pobytu

W aplikacji zastosowano dobrze znane informatykom i programistom podejście, tzw. *dummy-proof*. Nie pozwoli nam zarezerwować pokoju z datą wcześniejszą niż dzisiaj oraz sama wybierze datę wcześniejszą i wprowadzi do pola w formularzu *Od:* oraz *Do:*

Aby rezerwacja była prawidłowa nie można zaznaczyć tego samego dnia jako chęci spędzenia jednej nocy w hotelu. Zastosowano rozwiązanie, które można znaleźć na większości stron internetowych hoteli – zaznaczyć dzień zameldowania i dzień wymeldowania.

Zarezerwuj pokój

Od: 2017-06-12

Do: 2017-06-14

Typ pokoju

Jednoosobowy z łazienka

Zarezerwuj

Rysunek 22 Strona, która wyświetli się po kliknięciu przycisku "Pokaż wolne pokoje na ten okres". Informacje o wybranym terminie są przekazywane automatycznie.

Dane z poprzedniej strony są przekazywane automatycznie, a po kliknięciu w menu wyświetli się dostępna lista typów pokoi w podanym okresie. Ta informacja jest pobierana automatycznie z bazy danych. W przypadku, gdy nie będzie już pokoi podanego typu (np. *Apartament z Jacuzzi*), nie zostanie on wyświetlony na wysuwanej liście i nie wyświetli się żadne komunikat.

Zarezerwuj pokój

Od: 2017-06-12

Do: 2017-06-14

Typ pokoju

Jednoosobowy z łazienka
Jednoosobowy z łazienka
Jednoosobowy z łazienką dzieloną
Dwuosobowy z łóżkiem małżeńskim
Dwuosobowy z dwoma oddzielnymi łóżkami
Czterooosobowy z dwoma łóżkami małżeńskimi
Czterooosobowy - łóżko małżeńskie i dwa łóżka pojedyncze
Apartament dwuosobowy z sauna
Apartament dwuosobowy z Jacuzzi i sauna

Rysunek 23 Ekran z listą wysuwaną, na której znajdują się typy dostępnych pokoi w podanym okresie.

Rezerwacja zapisana!

© Hotel pod Gałązką

Rysunek 24 Poprawna rezerwacja kończy się wyświetleniem powyższego komunikatu.

Gdy wszystkie pola formularza zostaną wypełnione oraz zostanie poprawnie wybrany rodzaj pokoju zostaniemy przekierowani na stronę, gdzie wyświetli się powyższy komunikat. W przypadku gdy dane formularza zostaną nieprawidłowo wypełnione lub niektóre rubryki pozostaną puste zamiast powyższego komunikatu wyświetli się odpowiedni komunikat błędu, który powie użytkownikowi gdzie wystąpił błąd.

Zarządzaj swoimi rezerwacjami

Nr rezerwacji	Nr pokoju	Od	Do	Pokój	Cena za dobę
21	300	Sun, 11 Jun 2017 00:00:00 GMT	Thu, 22 Jun 2017 00:00:00 GMT	Apartament dwuosobowy z Jacuzzi	150
22	301	Sun, 11 Jun 2017 00:00:00 GMT	Fri, 23 Jun 2017 00:00:00 GMT	Apartament dwuosobowy z Jacuzzi	150
23	209	Tue, 01 Aug 2017 00:00:00 GMT	Tue, 14 Aug 2018 00:00:00 GMT	Dwuosobowy z dwoma oddzielnymi łózkami	50
26	100	Mon, 12 Jun 2017 00:00:00 GMT	Wed, 14 Jun 2017 00:00:00 GMT	Jednoosobowy z łazienką	30

Rysunek 25 Ekran, na którym wyświetlone są wszystkie rezerwacje, których dokonał użytkownik..

Po kliknięciu w lewym górnym rogu na zakładkę *Moje rezerwacje* wyświetli nam się powyższy obraz, który ukaże wszystkie dokonane przez nas informacje. Informacje te są otrzymywane poprzez zapytanie do bazy danych.

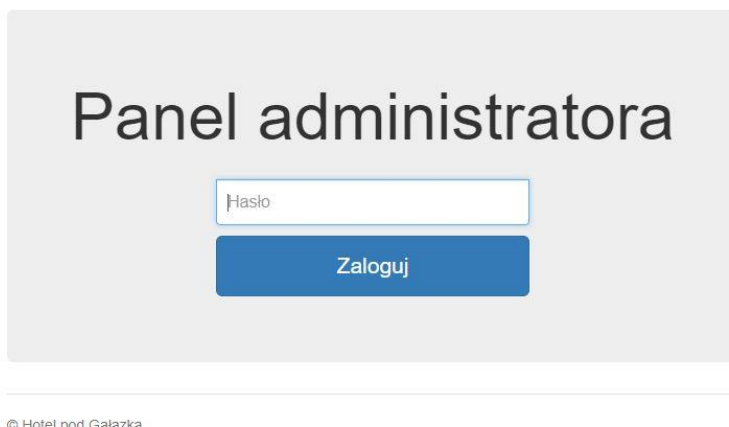
Po kliknięciu na przycisk *Wyloguj* zostaniemy przeniesieni z powrotem na stronę domową hotelu. Nasza sesja wygaśnie i pomimo tego, że będziemy mogli wrócić na stronę z takim samym interfejsem jak dla zalogowanego użytkownika (jest to spowodowane pamięcią *cache* przeglądarki) dane z bazy danych nie zostaną pobrane gdyż sesja użytkownika nie będzie aktywna.

Panel administratora

Aby obsługa hotelu mogła zalogować się do części przeznaczanej dla nich musi do pasku adresu dopisać */admin*.

 127.0.0.1:5000/admin

Rysunek 26 Przykładowy pasek adresu. W naszym przypadku serwer przypisany jest do adresu hosta lokalnego i portu 5000.



Rysunek 27 Ekran logowania administratora

Konto administratora jest tworzone podczas instalacji systemu. Domyślnie przypisywane jest do niego hasło *qwerty1234*. W przypadku chęci jego zmiany należy skontaktować się z osobą z działu technicznego.

Po wpisaniu hasła zostaniemy przekierowani na stronę ze wszystkimi wykonanymi rezerwacjami. Dodatkowo mamy opcję czterech innych zakładek. Menu nawigacyjne znajduje się w prawym górnym rogu. Możemy wylogować się, dodać dane gości do rezerwacji (przydatne podczas meldunku gości), wyświetlenia wszystkich zarejestrowanych gości, czy też wyświetlenia danych wszystkich

użytkowników zarejestrowanych poprzez stronę internetową (z pominięciem hasła – Ustawa o Zachowaniu Prywatności).

Panel Wyświetl gości Wyświetl użytkowników Dodaj gości Pokaż rezerwacje Wyloguj

Rezerwacje użytkowników

Nr rezerwacji	Id użytkownika	Nr pokoju	Od	Do	Pokój	Cena za dobę
21	2	300	Sun, 11 Jun 2017 00:00:00 GMT	Thu, 22 Jun 2017 00:00:00 GMT	Apartament dwuosobowy z Jacuzzi	150
22	2	301	Sun, 11 Jun 2017 00:00:00 GMT	Fri, 23 Jun 2017 00:00:00 GMT	Apartament dwuosobowy z Jacuzzi	150
23	2	209	Tue, 01 Aug 2017 00:00:00 GMT	Tue, 14 Aug 2018 00:00:00 GMT	Dwuosobowy z dwoma oddzielnymi łózkami	50
26	2	100	Mon, 12 Jun 2017 00:00:00 GMT	Wed, 14 Jun 2017 00:00:00 GMT	Jednoosobowy z łazienką	30
27	4	101	Mon, 12 Jun 2017 00:00:00 GMT	Fri, 30 Jun 2017 00:00:00 GMT	Jednoosobowy z łazienką	30

Rysunek 28 Podgląd wszystkich rezerwacji

Tabela przedstawia nr rezerwacji, który potrzebny jest pracownikowi hotelu, aby poprawnie dodać dane gości do wybranej rezerwacji. Dzięki tabeli Id użytkownika widzimy kto dokonał rezerwacji. Mamy również dostęp do tego jaki pokój został wybrany oraz jaki jest nr pokoju, który przyporządkowała baza danych i jaki klucz należy przekazać klientowi. Wiemy również jaka jest cena za dobę spędzoną w podanym pokoju oraz od kiedy obowiązuje rezerwacja.

Zarejestrowani użytkownicy

Id użytkownika	Email
1	admin@hotel-pod-galazka.com
2	test@test.com
3	adres@email.com
4	abc@gmail.com

© Hotel pod Gałązką

Rysunek 29 Podgląd zarejestrowanych użytkowników

Dzięki powyższej zakładce obsługa hotelu może stwierdzić czy konto rzeczywiście zostało poprawnie zarejestrowane (w razie wątpliwości klienta).

Dodaj gościa

Imię	<input type="text" value="Jan"/>
Nazwisko	<input type="text" value="Kowalski"/>
Narodowość	<input type="text" value="Polska/USA"/>
Typ dokumentu	<input type="text" value="Wybierz..."/>
ID dokumentu	<input type="text" value="ABC123456"/>
Nr rezerwacji	<input type="text" value="12"/>

Dodaj gościa

Rysunek 30 Zakładka dodawania gości. Wykorzystywana podczas zameldowania gości w hotelu

Dzięki zakładce, której widok przedstawiony jest na Rys. 44 obsługa hotelu może dodać dane gości podczas meldunku gości. W przypadku gdy użytkownik wprowadzi błędnie dane gościa i spróbuje zatwierdzić je przez kliknięcie przycisku *Dodaj gościa* na stronie ukaże się odpowiedni komunikat, w którym użytkownik zostanie poinformowany w jaki sposób powinien naprawić błąd.

Panel Wyświetl gości Wyświetl użytkowników Dodaj gościa Pokaż rezerwacje Wyloguj

Zapisani goście

Narodowość	Typ dokumentu	ID dokumentu	Imię	Nazwisko	Nr rezerwacji
San Escobar	Passport	abc12345	Kamil	Kowalski	21
Brzuch	Passport	124135	Noga	Reka	22
Polska	Passport	EG1234	Imie	Nazwisko	23
Kanada	DrivingLicense	KA12543	John	Smith	27

Rysunek 31 Podgląd zapisanych gości

Dzięki powyższemu widokowi, obsługa hotelu może sprawdzić i zweryfikować poprawność wprowadzonych danych do bazy danych.

Poprawne działanie takiej strony internetowej powinno być również wyposażone o interfejs usuwania rekordów z bazy danych. Taki *feature* nie został zaimplementowany ze względu na zbyt słabe zaplecze programistyczne oraz duże braki zespołowe w zespole jednoosobowym.

5.3. Testowanie opracowanych funkcji systemu

Hotel pod Gałązką

[Strona główna](#)

[Zaloguj](#)

[Rejestracja](#)



The screenshot shows a login form titled "Zaloguj się". It contains two input fields: the first contains "abc@gmail.com" and the second is labeled "Hasło" (Password). Below the password field is a blue button labeled "Zaloguj". A yellow tooltip with an exclamation mark icon is positioned over the button, displaying the message "Please fill out this field.".

Rysunek 32 W przypadku niepodania hasła serwer nie dokona operacji logowania

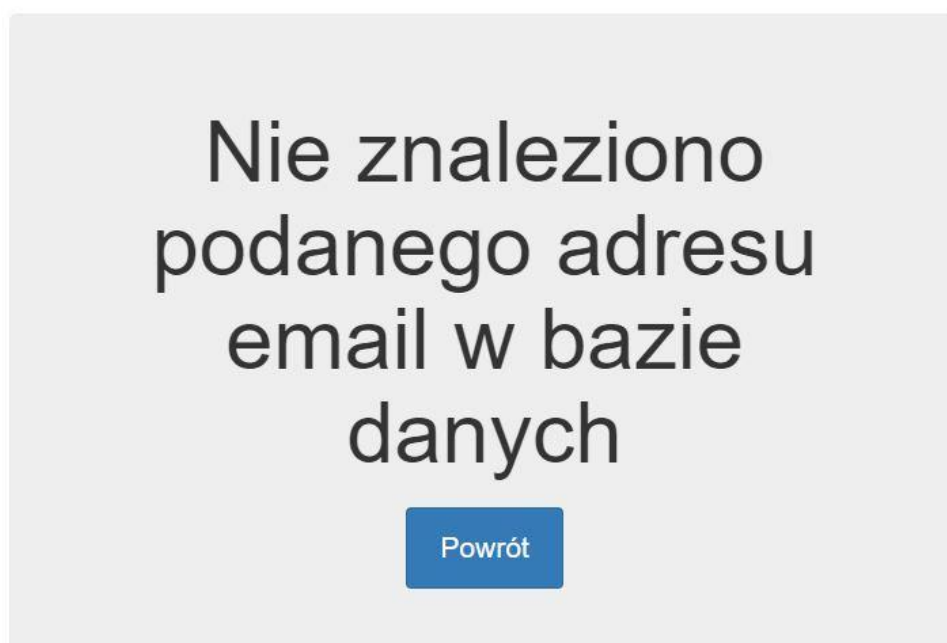
W przypadku nieuzupełnienia danych formularza logowania zostaniemy już o tym poinformowani już na etapie wpisywania. danych Dzięki temu nie trzeba będzie wysyłać zapytania do bazy danych, co zaoszczędzi czas reakcji operacji na stronie internetowej.

Hotel pod Gałązką

[Strona główna](#)

[Zaloguj](#)

[Rejestracja](#)



The screenshot shows a large gray box with the text "Nie znaleziono podanego adresu email w bazie danych" (Email address not found in the database) in a large, bold, black font. Below the text is a blue button labeled "Powrót" (Return).

Rysunek 33 Błędne adres email podczas logowania

Powyższy komunikat wyświetli się gdy użytkownik będzie próbował zalogować się do konta nie korzystając z operacji rejestracji (jego adres email nie będzie znajdował się w bazie danych).

W przypadku błędnego logowania (niepoprawnego wpisania pola *email* lub *hasło* zostanie wypisany poniższy komunikat:



Rysunek 34 Rezultat podania poprawnego adresu email, jednak nieprawidłowe hasło

Wybranie terminu rezerwacji jest walidowane na poziomie wprowadzania. Zastosowanie kalendarza umożliwia ustawienie, aby użytkownik nie był w stanie wybrać daty z przeszłości. Dodatkowym zabezpieczeniem jest to, że miejsca, w których pojawiają się daty wyświetlane są w trybie do odczytu – użytkownik nie może ich ręcznie zmienić.

The image shows a web page header with the text "Hotel pod Gałązką" on the left and three links, "Rezerwuj", "Moje rezerwacje", and "Wyloguj", on the right. The "Rezerwuj" link is highlighted with a blue background. Below the header is a large light gray rectangular box. At the top of this box is the text "Zarezerwuj pokój" in a large, bold, black font. Below this text are two input fields: "Od: 2017-06-12" and "Do: 2017-06-14". Below these fields is a label "Typ pokoju" and a dropdown menu showing "Jednoosobowy z łazienką". At the bottom of the box is a blue button with the white text "Zarezerwuj".

Rysunek 35 Ekran wyboru pokoju po wybraniu daty pobytu.

Po kliknięciu na przycisk *Pokaż wolne pokoje na ten okres* wyświetla się kolejna strona typu *dummy-proof*. Pola do odczytu, które zawierają mechanizmy zabezpieczeń przed błędem. Jednakże w najgorszym wypadku, gdy użytkownik znajdzie jakiś sposób, aby oszukać te zabezpieczenia, strona nie zawiesi się ani nie wykona nieoczekiwanego zamknięcia serwera tylko wyświetli komunikat o błędzie.

```
(1452, u'Cannot add
or update a child
row: a foreign key
constraint fails
(`hoteldatabase`.`guests`
CONSTRAINT
`fk_Guests_Bookings1`
FOREIGN KEY
(`Bookings_idBookings`)
REFERENCES
`bookings`
(`idBookings`) ON
```

Rysunek 36 Przykładowy komunikat błędu uzyskany dopiero po usunięciu zabezpieczeń na stronie internetowej.

5.4. Omówienie wybranych rozwiązań programistycznych

5.4.1. Implementacja interfejsu dostępu do bazy danych

Serwer przechowuje dane konfiguracyjne bazy danych i jej połączenia. Flask umożliwia również zaszyfrowanie pliku wykonywalnego, aby uniemożliwić hakerom dostęp do nich. W tym celu podaje się tzw.:

```
app.secret_key = 'why would I tell you my secret key?'
```

Odwołujemy się do bazy danych poprzez wywoływanie procedur. Jeżeli baza danych zwraca nam dane, zapisywane są one do pliku JSON, który następnie w kodzie JavaScript jest parsowany i przetwarzany.

Komunikaty błędu zazwyczaj wyświetlane są na osobnej stronie *error.html*, ale czasem również zmieniany jest nagłówek strony i w ten sposób użytkownik może dowiedzieć się o rezultacie swoich akcji.

Jest to technika jakie stosują jedne z poważniejszych stron internetowych takich jak *linkedin.com* czy *pinterest.com*.

5.4.2. Implementacja wybranych funkcjonalności systemu

Dostęp do każdej ze stron jest niesamowicie prosty dzięki API Flask:

```
@app.route("/")
def main():
    return render_template('index.html')

@app.route("/showIndex")
def showIndex():
    return render_template('index.html')

@app.route('/showSignUp')
def showSignUp():
    return render_template('signup.html')

@app.route('/showSignIn')
def showSignIn():
    return render_template('signin.html')

@app.route('/userHome')
def userHome():
    if session.get('user'):
        return render_template('userHome.html')
    else:
        return render_template('error.html', error='Nieautoryzowana próba dostępu')
```

Rysunek 37 Listing fragmentu kodu serwera, który odpowiedzialny jest za wyświetlanie odpowiednik plików .html przez przeglądarkę.

W `@app.route(url)` zostajemy przekierowani na stronę `url`.

Ponieważ twórcy architektury Flask stosowali zasadę *Clean Code*¹, nie zdecydowano się na umieszczenie komentarzy w kodzie gdyż byłoby to bezcelowe.

¹ Kod napisany przejrzystie, zrozumiale. Jest wtedy samodokumentujący się i nie wymaga dodatkowych komentarzy.

Poniżej została zaprezentowana metoda dostępu i wywołania procedury MySQL, która zwróci obsłudze hotelu wszystkie rekordy tablicy Guests:

```
@app.route('/getAllGuests')
def getAllGuests():
    try:
        app.logger.info("Trying to get user session")
        if session.get('user'):
            _user = session.get('user')
            app.logger.info("User session established: " + str(_user))
            con = mysql.connect()
            cursor = con.cursor()
            cursor.callproc('sp_getAllGuests',)
            bookings = cursor.fetchall()
            bookings_dict = []
            for booking in bookings:
                app.logger.info("booking: " + str(booking))
                booking_dict = {
                    'nationality': booking[1],
                    'documentType': booking[2],
                    'documentId': booking[3],
                    'name': booking[4],
                    'surname': booking[5],
                    'bookingsId': booking[6],}
                bookings_dict.append(booking_dict)

            return json.dumps(bookings_dict)
        else:
            return render_template('error.html', error='Nieautoryzowana proba dostepu')
    except Exception as e:
        return render_template('error.html', error=str(e))
```

Kod jQuery, która otrzymuje i parsuje otrzymany plik JSON:

```
<script>
$(function () {
    $.ajax({
        url: '/getAllGuests',
        type: 'GET',
        success: function (res) {
            var data = JSON.parse(res);

            $('tbody tr').not(':first').not(':last').remove();
            var html = "";
            for (var i = 0; i < data.length; i++)
                html += '<tr><td>' + data[i].nationality +
                    '</td><td>' + data[i].documentType +
                    '</td><td>' + data[i].documentId +
                    '</td><td>' + data[i].name +
                    '</td><td>' + data[i].surname +
                    '</td><td>' + data[i].bookingsId + '</td></tr>';
            $('tbody tr').first().after(html);
        },
        error: function (error) {
            console.log(error);
        }
    });
});
</script>
```

5.4.3. Implementacja mechanizmów bezpieczeństwa

Komunikacja na stronie odbywa zawsze poprzez metody POST. Jest to trudniejsze w implementacji, jednak gwarantuje większą odporność przeciwko atakom hackerskim oraz bardziej estetyczny wygląd pasku adresu.

```
$('#btnSignUp').click(function(){
    var temp;
    $.ajax({
        url: '/signUp',
        data: $('form').serialize(),
        type: 'POST',
        success: function(response){
            console.log(response);
            $('#signUpHeader').text(JSON.parse(response).message);
            $('#signUpHeader').text(JSON.parse(response).error);
        },
        error: function(error){
            console.log(error);
            $('#signUpHeader').text(JSON.parse(error).error);
        }
    });
});
```

Każda funkcja dostępna po zalogowaniu się opiera swoje bezpieczeństwo na sesji użytkownika. Kod funkcji sprawdza, czy użytkownik dokonał poprawnego logowania się. Nawet jeżeli niechciany intruz wejdzie na prawidłowy adres, nie otrzyma informacji z poziomu bazy danych, gdyż jest to zablokowane z poziomu kodu serwera.

```
@app.route('/signIn', methods=['POST'])
def signIn():
    try:
        # read the posted values from the UI
        _email = request.form['inputEmail']
        _password = request.form['inputPassword']
        m = md5.new()
        m.update(_password)
        _hashed_password = m.hexdigest()
        conn = mysql.connect()
        cursor = conn.cursor()
        cursor.callproc('sp_validateLogin', (_email,))
        data = cursor.fetchall()
        print "data[0][2] = " + str(data[0][2])
        if len(data) > 0:
            if str(data[0][2]) == _hashed_password:
                session['user'] = data[0][0]
                return render_template('userHome.html', message='Witaj!')
            else:
                return render_template('error.html', error='Wrong Email address or Password.')
        else:
            return render_template('error.html', error='Wrong Email address or Password.')

    except Exception as e:
        return render_template('error.html', error=str(e))
```

6. Podsumowanie i wnioski

Prezentowana i udokumentowana aplikacja to przedwstępna wersja systemu rezerwacji pokoi przez Internet dla hotelu. Została napisana w sposób, który umożliwia łatwe rozwinięcie projektu, aby wspierał dodatkowo możliwość zarządzania rezerwacjami zarówno przez użytkowników jak i pracowników hotelu – usuwanie rezerwacji czy zmiana terminu rezerwacji. Projekt można rozszerzyć o możliwość płacenia za pobyt, dodanie informacji o placówce wypoczynkowej czy wielu innych funkcjonalności. Nie zostały one dodane, gdyż wykracza to poza możliwości pojedynczego dewelopera z małym stażem

Jest to mój pierwszy projekt, który wykorzystuje bazę danych, aplikację sieciową, integruje te dwa komponenty. Wielką pomocą okazał się program MySQL Workbench, który jest swoistym warsztatem do tworzenia baz danych. Umożliwił szybsze wdrożenie mnie w świat baz danych.

Flask to mikro framework (lub mikro serwis) oparty na Werkzeug, Jinja2. Jest to typ **Web Server Gateway Interface (WSGI)**. To specyfikacja, który mówi o tym, że aplikacja posiada prosty i uniwersalny interfejs dla tworzenia serwerów sieciowych, aplikacji sieciowych w języku sieciowym.

Dzięki pracy włożonej w realizację tego projektu poznałem dogłębnie w jaki sposób tworzy się strony internetowe, w jaki sposób następuje wymiana danych na serwerze, pomiędzy stroną internetową, bazą danych a użytkownikiem. Dzięki zdobytej wiedzy jest mi łatwiej zrozumieć i rozwiązywać problemy, które zdarzały pojawiać się nie tylko podczas implementacji tego projektu, ale również przed jego implementacją.

Zadanie, które sobie postawiłem przyczyniło się do poznania *tajników programowania i projektowania sieciowego*, które w obecnych czasach jest dużym atutem na rozmowach kwalifikacyjnych oraz jako wpis w CV.

Literatura

Ponieważ moja praca związana była z przeglądaniem dokumentacji postanowiłem zamieścić adresy stron internetowych, które okazały się bardzo przydatne przy tworzeniu aplikacji.

1. <https://dev.mysql.com/doc/refman/5.7/en/>
2. <http://flask.pocoo.org/>
3. <https://api.jquery.com/>
4. <https://www.w3schools.com/html/>
5. <https://code.tutsplus.com/tutorials/creating-a-web-app-from-scratch-using-python-flask-and-mysql--cms-22972>
6. <http://www.pigno.se/barn/PIGNOSE-Calendar/>

Spis ilustracji

Rysunek 1 Model logiczny oraz model fizyczny.....	7
Rysunek 2 Diagramy przypadków użycia dla użytkownika oraz pracownika hotelu.....	9
Rysunek 3 Strona główna hotelu.....	10
Rysunek 4 Strona rezerwacji	10
Rysunek 5 Rejestracja.....	11
Rysunek 6 Logowanie	11
Rysunek 7 Wyświetlenie wszystkich rodzajów pokoi.....	16
Rysunek 8 Wyświetlenie wszystkich rezerwacji	16
Rysunek 9 Wyświetlenie wszystkich rezerwacji dla danego użytkownika	16
Rysunek 10 Usunięcie rezerwacji	16
Rysunek 11 Dodanie nowego użytkownika oraz procedura zwracająca wszystkich użytkowników	17
Rysunek 12 Dodanie nowego gościa oraz procedura zwracająca wszystkich gości.....	17
Rysunek 13 Utworzenie nowej rezerwacji	17
Rysunek 14 Wyświetlenie wszystkich rezerwacji oraz wolnych i zarezerwowanych pokoi.....	18
Rysunek 15 Strona startowa. Zdjęcia to pokaz slajdów z estetyczną animacją	21
Rysunek 16 Formularz rejestracji	22
Rysunek 17 Poprawna rejestracja	22
Rysunek 18 Ekran logowania.....	23
Rysunek 19 Po podaniu poprawnych danych wyświetla się komunikat powitalny i jesteśmy zalogowani na nasze konto.	23
Rysunek 20 Ekran rezerwacji pokoju. Aby wybrać obszar możemy skorzystać z pomocniczego kalendarza lub wpisać datę w formacie RRRR-MM-DD	24
Rysunek 21 Poprawny wygląd danych pobytu.....	25
Rysunek 22 Strona, która wyświetli się po kliknięciu przycisku "Pokaż wolne pokoje na ten okres". Informacje o wybranym terminie są przekazywane automatycznie.....	26
Rysunek 23 Ekran z listą wysuwaną, na której znajdują się typy dostępnych pokoi w podanym okresie.	26

Rysunek 24 Poprawna rezerwacja kończy się wyświetleniem powyższego komunikatu.	27
Rysunek 25 Ekran, na którym wyświetlone są wszystkie rezerwacje, których dokonał użytkownik...	27
Rysunek 26 Przykładowy pasek adresu. W naszym przypadku serwer przypisany jest do adresu hosta lokalnego i portu 5000.	28
Rysunek 27 Ekran logowania administratora	28
Rysunek 28 Podgląd wszystkich rezerwacji.....	29
Rysunek 29 Podgląd zarejestrowanych użytkowników.....	30
Rysunek 30 Zakładka dodawania gości. Wykorzystywana podczas zameldowania gości w hotelu.....	30
Rysunek 31 Podgląd zapisanych gości.....	31
Rysunek 32 W przypadku niepodania hasła serwer nie dokona operacji logowania	32
Rysunek 33 Błędne adres email podczas logowania	32
Rysunek 34 Rezultat podania poprawnego adresu email, jednak nieprawidłowe hasło.....	33
Rysunek 35 Ekran wyboru pokoju po wybraniu daty pobytu.	33
Rysunek 36 Przykładowy komunikat błędu uzyskany dopiero po usunięciu zabezpieczeń na stronie internetowej.....	34
Rysunek 37 Listing fragmentu kodu serwera, który odpowiedzialny jest za wyświetlanie odpowiednik plików .html przez przeglądarkę.	35