Projekt
Baza Hotelowa

Piotrowski Dawid
Informatyka Stosowana
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

15.01.2025r.

Strona znajduje się na serwerze wydziałowym Pascal w lokalizacji `~2piotrowski/hotel/`

Spis treści

Rozdział 1

Projekt koncepcji, założenia

1.1 Zdefiniowanie tematu projektu

Projekt dotyczy stworzenia zaawansowanej bazy danych dla systemu rezerwacji hotelowych. Celem projektu jest umożliwienie zarządzania rezerwacjami, pokojami, gośćmi, dodatkami oraz pracownikami hotelu. System ma na celu automatyzację procesów związanych z rezerwacjami, zarządzaniem stanem pokojów oraz generowaniem raportów finansowych.

1.2 Analiza wymagań użytkownika

Aby system spełniał oczekiwania użytkowników, został zidentyfikowany szereg funkcjonalności:

- Zarządzanie rezerwacjami.
- Obsługa wyposażenia pokoi.
- Zarządzanie pokojami.
- Zarządzanie typami łóżek.
- Zarządzanie klasami pokoi.
- Zarządzanie gośćmi.
- Obsługa dodatków do rezerwacji.
- Zarządzanie pracownikami.
- Generowanie raportów.
- Logowanie wszystkich operacji w systemie.
- Walidacja danych wejściowych (np. numer telefonu, zakres dat).

1.3 Zaprojektowanie funkcji

Podstawowe funkcje realizowane w bazie danych obejmują:

- `dodaj_rezerwacje_przez_nowego_goscia`: Dodawanie rezerwacji dla nowego gościa.
- `dodaj_rezerwacje_przez_goscia_public`: Dodawanie rezerwacji przez istniejącego lub nowego gościa.
- `dodaj_rezerwacje`: Dodawanie rezerwacji dla istniejącego gościa.
- `zaktualizuj_rezerwacje`: Aktualizacja danych rezerwacji.
- `usun_rezerwacje`: Usuwanie rezerwacji.
- `anuluj_rezerwacje`: Anulowanie rezerwacji.
- `wykwateruj_rezerwacje`: Wykwaterowanie rezerwacji.
- `zmien_status_pokoju`: Zmiana statusu pokoju.

- `sprawdz_dostepne_pokoje_z_parametrami`: Sprawdzanie dostępności pokojów na podstawie parametrów.
- `oblicz_koszt_rezerwacji_z_parametrami`: Obliczanie kosztów rezerwacji.
- `aktualizuj_status_platnosci`: Aktualizacja statusu płatności.

Rozdział 2

Projekt diagramów (konceptualny)

2.1 Budowa i analiza diagramu przepływu danych (DFD)

Diagram przepływu danych przedstawia sposób, w jaki dane przepływają przez system rezerwacji hotelowych. Główne elementy DFD to:

- **Wejścia:** Dane rezerwacji, dane gościa, dane płatności, dane pokoi.
- **Przetwarzania:** Dodawanie rezerwacji, aktualizacja statusów, obliczanie kosztów, generowanie raportów.
- **Wyjścia:** Potwierdzenia rezerwacji, raporty finansowe, logi systemowe.

Rysunek 2.1: Diagram przepływu danych (DFD)

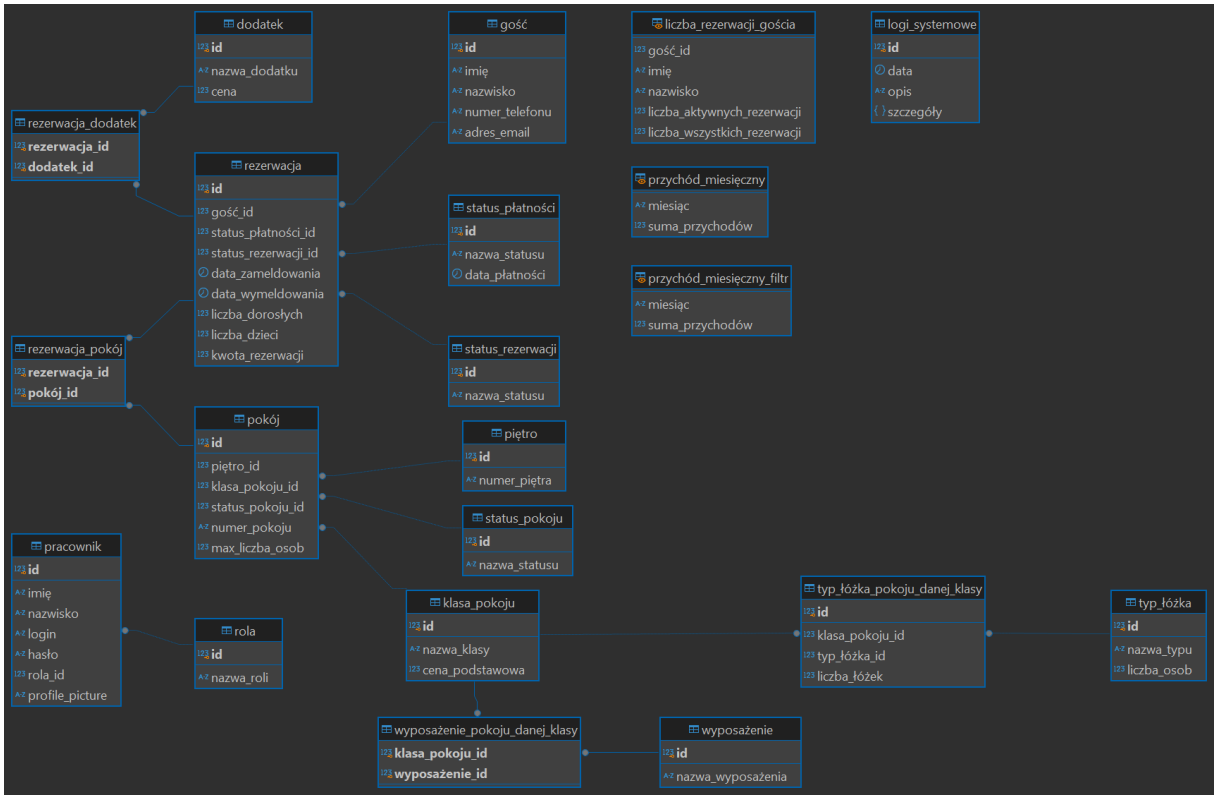
2.2 Zdefiniowanie encji (obiektów) oraz ich atrybutów

System obejmuje następujące encje:

- **Gość:** `id`, `imię`, `nazwisko`, `numer_telefonu`, `adres_email`.
- **Rezerwacja:** `id`, `gość_id`, `status_płatności_id`, `status_rezerwacji_id`, `data_zameldowania`, `data_wymeldowania`, `liczba_dorosłych`, `liczba_dzieci`, `kwota_rezerwacji`.
- **Pokój:** `id`, `piętro_id`, `klasa_pokoju_id`, `status_pokoju_id`, `numer_pokoju`, `max_liczba_osob`.
- **Dodatki:** `id`, `nazwa_dodatku`, `cena`.
- **Pracownik:** `id`, `imię`, `nazwisko`, `login`, `hasło`, `rola_id`, `profile_picture`.
- **Logi systemowe:** `id`, `data`, `opis`, `szczegóły`.
- **Typ łóżka:** `id`, `nazwa_typu`, `liczba_osob`.
- **Klasa pokoju:** `id`, `nazwa_klasy`, `cena_podstawowa`.
- **Wyposażenie:** `id`, `nazwa_wyposażenia`.
- **Piętro:** `id`, `numer_piętra`.
- **Rola:** `id`, `nazwa_roli`.

2.3 Zaprojektowanie relacji pomiędzy encjami (ERD)

Poniżej przedstawiono diagram ERD (Entity-Relationship Diagram), który ilustruje powiązania między encjami w systemie.



Rysunek 2.2: Diagram ERD

Rozdział 3

Projekt logiczny

3.1 Projektowanie tabel, kluczy, indeksów

Struktura bazy danych została zaprojektowana zgodnie z wymaganiami, zapewniając integralność danych poprzez zastosowanie kluczy głównych i obcych oraz odpowiednich ograniczeń (constraints).

3.1.1 Tworzenie tabel

Poniżej przedstawiono schemat tworzenia tabel wraz z kluczami głównymi i obcymi:

```sql
-- Tworzenie schematu
CREATE SCHEMA IF NOT EXISTS rezerwacje_hotelowe;

-- Tabele główne
CREATE TABLE rezerwacje_hotelowe.gość (
    id SERIAL PRIMARY KEY,
    imię VARCHAR(200) NOT NULL,
    nazwisko VARCHAR(200) NOT NULL,
    numer_telefonu VARCHAR(20) NOT NULL,
    adres_email VARCHAR(350) NOT NULL,
    CONSTRAINT unikalny_gość_email UNIQUE (adres_email),
    CONSTRAINT chk_numer_telefonu
        CHECK (numer_telefonu ~ '^(\\+[0-9]{1,3})?[0-9]{9}$')
);

CREATE TABLE rezerwacje_hotelowe.status_płatności (
    id SERIAL PRIMARY KEY,
    nazwa_statusu VARCHAR(50) NOT NULL UNIQUE,
    data_płatności TIMESTAMP
);

CREATE TABLE rezerwacje_hotelowe.status_rezerwacji (
    id SERIAL PRIMARY KEY,
    nazwa_statusu VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE rezerwacje_hotelowe.rezerwacja (
    id SERIAL PRIMARY KEY,
    gość_id INT NOT NULL,
    status_płatności_id INT NOT NULL,
    status_rezerwacji_id INT NOT NULL,
    data_zameldowania DATE NOT NULL,
    data_wymeldowania DATE NOT NULL,
    liczba_dorosłych INT CHECK (liczba_dorosłych >= 0),
    liczba_dzieci INT CHECK (liczba_dzieci >= 0),
    kwota_rezerwacji DECIMAL(10, 2) CHECK (kwota_rezerwacji >= 0),
    CONSTRAINT fk_rezerwacja_gość FOREIGN KEY (gość_id) REFERENCES rezerwacje_hotelowe.gość (id),
    CONSTRAINT fk_rezerwacja_status_płatności FOREIGN KEY (status_płatności_id) REFERENCES
        rezerwacje_hotelowe.status_płatności (id),
    CONSTRAINT fk_rezerwacja_status_rezerwacji FOREIGN KEY (status_rezerwacji_id) REFERENCES
        rezerwacje_hotelowe.status_rezerwacji (id)
);

CREATE TABLE rezerwacje_hotelowe.dodatek (
    id SERIAL PRIMARY KEY,
    nazwa_dodatku VARCHAR(100) NOT NULL,
    cena DECIMAL(10, 2) CHECK (cena >= 0)
);

CREATE TABLE rezerwacje_hotelowe.rezerwacja_dodatek (
    rezerwacja_id INT NOT NULL,
    dodatek_id INT NOT NULL,
    PRIMARY KEY (rezerwacja_id, dodatek_id),
    CONSTRAINT fk_rez_dod_rezerwacja FOREIGN KEY (rezerwacja_id) REFERENCES rezerwacje_hotelowe.
        rezerwacja (id),
    CONSTRAINT fk_rez_dod_dodatek FOREIGN KEY (dodatek_id) REFERENCES rezerwacje_hotelowe.dodatek (id)
);

CREATE TABLE rezerwacje_hotelowe.typ_łóżka (
```

```
    id SERIAL PRIMARY KEY,
    nazwa_typu VARCHAR(50) NOT NULL,
    liczba_osob INT NOT NULL DEFAULT 1
);

CREATE TABLE rezerwacje_hotelowe.klasa_pokoju (
    id SERIAL PRIMARY KEY,
    nazwa_klasy VARCHAR(100) NOT NULL,
    cena_podstawowa DECIMAL(10, 2) CHECK (cena_podstawowa >= 0)
);

CREATE TABLE rezerwacje_hotelowe.wyposażenie (
    id SERIAL PRIMARY KEY,
    nazwa_wyposażenia VARCHAR(100) NOT NULL
);

CREATE TABLE rezerwacje_hotelowe.wyposażenie_pokoju_danej_klasy (
    klasa_pokoju_id INT NOT NULL,
    wyposażenie_id INT NOT NULL,
    PRIMARY KEY (klasa_pokoju_id, wyposażenie_id),
    CONSTRAINT fk_klasa_wyposażenie_klasa FOREIGN KEY (klasa_pokoju_id) REFERENCES rezerwacje_hotelowe
        .klasa_pokoju (id),
    CONSTRAINT fk_klasa_wyposażenie_wyposażenie FOREIGN KEY (wyposażenie_id) REFERENCES
        rezerwacje_hotelowe.wyposażenie (id)
);

CREATE TABLE rezerwacje_hotelowe.typ_łóżka_pokoju_danej_klasy (
    id SERIAL PRIMARY KEY,
    klasa_pokoju_id INT NOT NULL,
    typ_łóżka_id INT NOT NULL,
    liczba_łóżek INT CHECK (liczba_łóżek > 0),
    CONSTRAINT fk_klasa_typ_łóżka_klasa FOREIGN KEY (klasa_pokoju_id) REFERENCES rezerwacje_hotelowe.
        klasa_pokoju (id),
    CONSTRAINT fk_klasa_typ_łóżka_typ FOREIGN KEY (typ_łóżka_id) REFERENCES rezerwacje_hotelowe.typ_ło
        żka (id)
);

CREATE TABLE rezerwacje_hotelowe.status_pokoju (
    id SERIAL PRIMARY KEY,
    nazwa_statusu VARCHAR(100) NOT NULL
);

CREATE TABLE rezerwacje_hotelowe.piętro (
    id SERIAL PRIMARY KEY,
    numer_piętra VARCHAR(5) NOT NULL
);

CREATE TABLE rezerwacje_hotelowe.pokój (
    id SERIAL PRIMARY KEY,
    piętro_id INT NOT NULL,
    klasa_pokoju_id INT NOT NULL,
    status_pokoju_id INT NOT NULL,
    numer_pokoju VARCHAR(10) NOT NULL UNIQUE,
    max_liczba_osob INT,
    CONSTRAINT fk_pokój_piętro FOREIGN KEY (piętro_id) REFERENCES rezerwacje_hotelowe.piętro (id),
    CONSTRAINT fk_pokój_klasa_pokoju FOREIGN KEY (klasa_pokoju_id) REFERENCES rezerwacje_hotelowe.
        klasa_pokoju (id),
    CONSTRAINT fk_pokój_status_pokoju FOREIGN KEY (status_pokoju_id) REFERENCES rezerwacje_hotelowe.
        status_pokoju (id)
);

CREATE TABLE rezerwacje_hotelowe.rezerwacja_pokój (
    rezerwacja_id INT NOT NULL,
    pokój_id INT NOT NULL,
    PRIMARY KEY (rezerwacja_id, pokój_id),
    CONSTRAINT fk_rez_pok_rezerwacja FOREIGN KEY (rezerwacja_id) REFERENCES rezerwacje_hotelowe.
        rezerwacja (id),
    CONSTRAINT fk_rez_pok_pokój FOREIGN KEY (pokój_id) REFERENCES rezerwacje_hotelowe.pokój (id)
);

CREATE TABLE rezerwacje_hotelowe.rola (
    id SERIAL PRIMARY KEY,
    nazwa_roli VARCHAR(50) NOT NULL
);

CREATE TABLE rezerwacje_hotelowe.pracownik (
    id SERIAL PRIMARY KEY,
    imię VARCHAR(200) NOT NULL,
    nazwisko VARCHAR(200) NOT NULL,
    login VARCHAR(100) UNIQUE NOT NULL,
    hasło VARCHAR(256) NOT NULL,
    rola_id INT NOT NULL,
    profile_picture VARCHAR(255) DEFAULT NULL,
    CONSTRAINT fk_pracownik_rola FOREIGN KEY (rola_id) REFERENCES rezerwacje_hotelowe.rola (id)
);

CREATE TABLE rezerwacje_hotelowe.logi_systemowe (
    id SERIAL PRIMARY KEY,
    data TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    opis VARCHAR(500),
    szczegóły JSONB
```

```
141 );
```

3.2 Słowniki danych

Poniżej przedstawiono słownik danych dla kluczowych tabel w systemie.

3.2.1 Tabela `gość`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator gościa.
imię	VARCHAR(200)	Imię gościa.
nazwisko	VARCHAR(200)	Nazwisko gościa.
numer_telefonu	VARCHAR(20)	Numer telefonu gościa. Musi spełniać format: opcjonalny prefiks (+xxx) oraz dokładnie 9 cyfr.
adres_email	VARCHAR(350)	Adres e-mail gościa. Musi być unikalny.

3.2.2 Tabela `rezerwacja`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator rezerwacji.
gość_id	INT	Klucz obcy do tabeli `gość`.
status_płatności_id	INT	Klucz obcy do tabeli `status_płatności`.
status_rezerwacji_id	INT	Klucz obcy do tabeli `status_rezerwacji`.
data_zameldowania	DATE	Data zameldowania.
data_wymeldowania	DATE	Data wymeldowania.
liczba_dorosłych	INT	Liczba dorosłych osób w rezerwacji. Musi być nieujemna.
liczba_dzieci	INT	Liczba dzieci w rezerwacji. Musi być nieujemna.
kwota_rezerwacji	DECIMAL(10,2)	Kwota rezerwacji. Musi być nieujemna.

3.2.3 Tabela `status_płatności`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator statusu płatności.
nazwa_statusu	VARCHAR(50)	Nazwa statusu płatności. Musi być unikalna.
data_płatności	TIMESTAMP	Data i czas dokonania płatności.

3.2.4 Tabela `status_rezerwacji`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator statusu rezerwacji.
nazwa_statusu	VARCHAR(50)	Nazwa statusu rezerwacji. Musi być unikalna.

3.2.5 Tabela `dodatek`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator dodatku.
nazwa_dodatku	VARCHAR(100)	Nazwa dodatku.
cena	DECIMAL(10,2)	Cena dodatku. Musi być nieujemna.

3.2.6 Tabela `rezerwacja_dodatek`

Atrybut	Typ	Opis
rezerwacja_id	INT	Klucz obcy do tabeli `rezerwacja`.
dodatek_id	INT	Klucz obcy do tabeli `dodatek`.

3.2.7 Tabela `typ_łóżka`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator typu łóżka.
nazwa_typu	VARCHAR(50)	Nazwa typu łóżka (np. Pojedyncze, Podwójne).
liczba_osob	INT	Maksymalna liczba osób, które mogą spać na danym typie łóżka. Domyślnie 1.

3.2.8 Tabela `klasa_pokoju`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator klasy pokoju.
nazwa_klasy	VARCHAR(100)	Nazwa klasy pokoju (np. Standard, Deluxe).
cena_podstawowa	DECIMAL(10,2)	Podstawowa cena za noc pobytu w pokoju. Musi być nieujemna.

3.2.9 Tabela `wyposażenie`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator wyposażenia.
nazwa_wyposażenia	VARCHAR(100)	Nazwa wyposażenia (np. Telewizor, Klimatyzacja).

3.2.10 Tabela `wyposażenie_pokoju_danej_klasy`

Atrybut	Typ	Opis
klasa_pokoju_id	INT	Klucz obcy do tabeli `klasa_pokoju`.
wyposażenie_id	INT	Klucz obcy do tabeli `wyposażenie`.

3.2.11 Tabela `typ_łóżka_pokoju_danej_klasy`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator powiązania typu łóżka z klasą pokoju.
klasa_pokoju_id	INT	Klucz obcy do tabeli `klasa_pokoju`.
typ_łóżka_id	INT	Klucz obcy do tabeli `typ_łóżka`.
liczba_łóżek	INT	Liczba łóżek danego typu w klasie pokoju. Musi być większa niż 0.

3.2.12 Tabela `status_pokoju`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator statusu pokoju.
nazwa_statusu	VARCHAR(100)	Nazwa statusu pokoju (np. Dostępny, Zajęty).

3.2.13 Tabela `piętro`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator piętra.
numer_piętra	VARCHAR(5)	Numer piętra (np. 1, 2, 3).

3.2.14 Tabela `pokój`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator pokoju.
piętro_id	INT	Klucz obcy do tabeli `piętro`.
klasa_pokoju_id	INT	Klucz obcy do tabeli `klasa_pokoju`.
status_pokoju_id	INT	Klucz obcy do tabeli `status_pokoju`.
numer_pokoju	VARCHAR(10)	Numer pokoju. Musi być unikalny.
max_liczba_osob	INT	Maksymalna liczba osób, które mogą się zakwaterować w pokoju.

3.2.15 Tabela `rezerwacja_pokój`

Atrybut	Typ	Opis
rezerwacja_id	INT	Klucz obcy do tabeli `rezerwacja`.
pokój_id	INT	Klucz obcy do tabeli `pokój`.

3.2.16 Tabela `rola`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator roli.
nazwa_roli	VARCHAR(50)	Nazwa roli (np. Administrator, Recepcjonista, Manager).

3.2.17 Tabela `pracownik`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator pracownika.
imię	VARCHAR(200)	Imię pracownika.
nazwisko	VARCHAR(200)	Nazwisko pracownika.
login	VARCHAR(100)	Unikalny login pracownika.
hasło	VARCHAR(256)	Hasło pracownika (przechowywane jako hash).
rola_id	INT	Klucz obcy do tabeli `rola`.
profile_picture	VARCHAR(255)	Ścieżka do zdjęcia profilowego pracownika.

3.2.18 Tabela `logi_systemowe`

Atrybut	Typ	Opis
id	SERIAL	Unikalny identyfikator logu.
data	TIMESTAMP	Data i czas operacji.
opis	VARCHAR(500)	Krótki opis operacji.
szczegóły	JSONB	Szczegółowe informacje o operacji.

3.3 Analiza zależności funkcyjnych i normalizacja tabel

W celu zapewnienia integralności danych oraz eliminacji redundancji, wszystkie tabele w systemie rezerwacji hotelowych zostały zaprojektowane zgodnie z zasadami normalizacji. Poniżej przedstawiono analizę zależności funkcyjnych oraz ocenę normalnej postaci dla każdej tabeli.

3.3.1 Tabela `gość`

Zależności funkcyjne

- `id` → `imię`, `nazwisko`, `numer_telefonu`, `adres_email`
- `adres_email` → `id`

Normalizacja

- **Pierwsza postać normalna (1NF):** Tabela `gość` jest w 1NF, ponieważ wszystkie atrybuty zawierają tylko wartości atomowe, a tabela posiada klucz główny (`id`).
- **Druga postać normalna (2NF):** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego (`id`).
- **Trzecia postać normalna (3NF):** Tabela jest w 3NF, ponieważ nie występują zależności przejściowe między atrybutami. `adres_email` jest unikalny i zależny bezpośrednio od `id`, a nie od innych atrybutów.

3.3.2 Tabela `rezerwacja`

Zależności funkcyjne

- `id` → `gość_id`, `status_płatności_id`, `status_rezerwacji_id`, `data_zameldowania`, `data_wymeldowania`, `liczba_dorosłych`, `liczba_dzieci`, `kwota_rezerwacji`
- `gość_id` → `gość_id` **(Tylko jako klucz obcy)**
- `status_płatności_id` → `status_płatności_id` **(Tylko jako klucz obcy)**
- `status_rezerwacji_id` → `status_rezerwacji_id` **(Tylko jako klucz obcy)**

Normalizacja

- **1NF:** Tabela `rezerwacja` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego (`id`).
- **3NF:** Tabela jest w 3NF, ponieważ nie występują zależności przejściowe. Wszystkie atrybuty są bezpośrednio zależne od klucza głównego, a klucze obce odwołują się bezpośrednio do odpowiednich tabel.

3.3.3 Tabela `status_płatności`

Zależności funkcyjne

- `id` → `nazwa_statusu`, `data_płatności`

Normalizacja

- **1NF:** Tabela `status_płatności` jest w 1NF, wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, zakładając, że `nazwa_statusu` jest unikalna. Nie występują zależności przejściowe.

3.3.4 Tabela `status_rezerwacji`

Zależności funkcyjne

- `id` → `nazwa_statusu`

Normalizacja

- **1NF:** Tabela `status_rezerwacji` jest w 1NF, wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ `nazwa_statusu` jest unikalna i zależna bezpośrednio od `id`, eliminując zależności przejściowe.

3.3.5 Tabela `dodatek`

Zależności funkcyjne

- `id` → `nazwa_dodatku`, `cena`

Normalizacja

- **1NF:** Tabela `dodatek` jest w 1NF, wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ nie występują zależności przejściowe.

3.3.6 Tabela `rezerwacja_dodatek`

Zależności funkcyjne

- `rezerwacja_id`, `dodatek_id` → `rezerwacja_id`, `dodatek_id`

Normalizacja

- **1NF:** Tabela `rezerwacja_dodatek` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny złożony z `rezerwacja_id` i `dodatek_id`.
- **2NF:** Tabela jest w 2NF, ponieważ nie posiada niekluczowych atrybutów. Wszystkie atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ nie posiada niekluczowych atrybutów oraz nie występują zależności przejściowe.

3.3.7 Tabela `typ_łóżka`

Zależności funkcyjne

- `id` → `nazwa_typu`, `liczba_osob`

Normalizacja

- **1NF:** Tabela `typ_łóżka` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ, nie występują zależności przejściowe.

3.3.8 Tabela `klasa_pokoju`

Zależności funkcyjne

- `id` → `nazwa_klasy`, `cena_podstawowa`

Normalizacja

- **1NF:** Tabela `klasa_pokoju` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ nie występują zależności przejściowe.

3.3.9 Tabela `wyposażenie`

Zależności funkcyjne

- `id` → `nazwa_wyposażenia`

Normalizacja

- **1NF:** Tabela `wyposażenie` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ nie występują zależności przejściowe.

3.3.10 Tabela `wyposażenie_pokoju_danej_klasy`

Zależności funkcyjne

- `klasa_pokoju_id`, `wyposażenie_id` → `klasa_pokoju_id`, `wyposażenie_id`

Normalizacja

- **1NF:** Tabela `wyposażenie_pokoju_danej_klasy` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny złożony z `klasa_pokoju_id` i `wyposażenie_id`.
- **2NF:** Tabela jest w 2NF, ponieważ nie posiada niekluczowych atrybutów. Wszystkie atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ nie posiada niekluczowych atrybutów oraz nie występują zależności przejściowe.

3.3.11 Tabela `typ_łóżka_pokoju_danej_klasy`

Zależności funkcyjne

- `id` → `klasa_pokoju_id`, `typ_łóżka_id`, `liczba_łóżek`

Normalizacja

- **1NF:** Tabela `typ_łóżka_pokoju_danej_klasy` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ nie posiada niekluczowych atrybutów oraz nie występują zależności przejściowe.

3.3.12 Tabela `status_pokoju`

Zależności funkcyjne

- `id` → `nazwa_statusu`

Normalizacja

- **1NF:** Tabela `status_pokoju` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ nie występują zależności przejściowe.

3.3.13 Tabela `piętro`

Zależności funkcyjne

- `id` → `numer_piętra`

Normalizacja

- **1NF:** Tabela `piętro` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ nie występują zależności przejściowe.

3.3.14 Tabela `pokój`

Zależności funkcyjne

- `id` → `piętro_id`, `klasa_pokoju_id`, `status_pokoju_id`, `numer_pokoju`, `max_liczba_osob`
- `numer_pokoju` → `id`, `piętro_id`, `klasa_pokoju_id`, `status_pokoju_id`, `max_liczba_osob` **(Ponieważ numer pokoju jest unikalny)**

Normalizacja

- **1NF:** Tabela `pokój` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, zakładając, że `numer_pokoju` jest unikalny. Nie występują zależności przejściowe.

3.3.15 Tabela `rezerwacja_pokój`

Zależności funkcyjne

- `rezerwacja_id`, `pokój_id` → `rezerwacja_id`, `pokój_id`

Normalizacja

- **1NF:** Tabela `rezerwacja_pokój` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny złożony z `rezerwacja_id` i `pokój_id`.
- **2NF:** Tabela jest w 2NF, ponieważ nie posiada niekluczowych atrybutów. Wszystkie atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ nie posiada niekluczowych atrybutów oraz nie występują zależności przejściowe.

3.3.16 Tabela `rola`

Zależności funkcyjne

- `id` → `nazwa_roli`

Normalizacja

- **1NF:** Tabela `rola` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ, nie występują zależności przejściowe.

3.3.17 Tabela `pracownik`

Zależności funkcyjne

- `id` → `imię`, `nazwisko`, `login`, `hasło`, `rola_id`, `profile_picture`
- `login` → `id`, `imię`, `nazwisko`, `hasło`, `rola_id`, `profile_picture` **(Ponieważ login jest unikalny)**

Normalizacja

- **1NF:** Tabela `pracownik` jest w 1NF, ponieważ wszystkie atrybuty są atomowe, a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, zakładając, że `login` jest unikalny. Nie występują zależności przejściowe.

3.3.18 Tabela `logi_systemowe`

Zależności funkcyjne

- `id` → `data`, `opis`, `szczegóły`

Normalizacja

- **1NF:** Tabela `logi_systemowe` jest w 1NF, ponieważ wszystkie atrybuty są atomowe (JSONB jest traktowany jako atomowy typ danych), a tabela posiada klucz główny (`id`).
- **2NF:** Tabela jest w 2NF, ponieważ wszystkie niekluczowe atrybuty są w pełni zależne od klucza głównego.
- **3NF:** Tabela jest w 3NF, ponieważ nie występują zależności przejściowe. Wszystkie atrybuty są bezpośrednio zależne od klucza głównego.

3.4 Denormalizacja struktury tabel

Denormalizacja została zastosowana w tabelach `wyposażenie_pokoju_danej_klasy` oraz `typ_łóżka_pokoju_danej_klasy`, aby umożliwić szybkie zapytania dotyczące wyposażenia pokoi oraz typów łóżek bez konieczności wielokrotnego łączenia tabel.

3.5 Zaprojektowanie operacji na danych

Funkcje oraz procedury składowane zostały zaprojektowane, aby umożliwić realizację wszystkich wymaganych operacji. Poniżej przedstawiono przykładowe kwerendy oraz funkcje:

3.5.1 Funkcja: dodaj_rezerwacje_przez_nowego_goscia

```
CREATE OR REPLACE FUNCTION rezerwacje_hotelowe.dodaj_rezerwacje_przez_nowego_goscia(
    imie VARCHAR,                   -- Imię nowego gościa
    nazwisko VARCHAR,               -- Nazwisko nowego gościa
    numer_telefonu VARCHAR,         -- Wymagany numer telefonu
    adres_email VARCHAR,            -- Wymagany adres e-mail
    pokoje INT[],                   -- Tablica ID pokoi
    start_date DATE,                -- Data zameldowania
    end_date DATE,                  -- Data wymeldowania
    liczba_doroslych INT,           -- Liczba dorosłych
    liczba_dzieci INT,              -- Liczba dzieci
    dodatki INT[] DEFAULT NULL      -- Opcjonalna tablica dodatków
)
RETURNS VOID AS $$
DECLARE
    new_gosc_id INT;                -- ID nowo utworzonego gościa
BEGIN
    -- Walidacja danych wejściowych dla gościa
    IF imie IS NULL OR nazwisko IS NULL OR numer_telefonu IS NULL OR adres_email IS NULL THEN
        RAISE EXCEPTION 'Brak wymaganych danych: imię, nazwisko, numer telefonu lub adres e-mail.';
    END IF;

    -- Tworzenie nowego gościa
    INSERT INTO rezerwacje_hotelowe.gość (imię, nazwisko, numer_telefonu, adres_email)
    VALUES (imie, nazwisko, numer_telefonu, adres_email)
    RETURNING id INTO new_gosc_id;

    -- Wywołanie funkcji dodaj_rezerwacje dla nowo utworzonego gościa
    PERFORM rezerwacje_hotelowe.dodaj_rezerwacje(
        new_gosc_id,
        pokoje,
        start_date,
        end_date,
        liczba_doroslych,
        liczba_dzieci,
        dodatki
    );

    -- Logowanie dodania gościa i rezerwacji
    INSERT INTO rezerwacje_hotelowe.logi_systemowe (opis, szczegóły)
    VALUES (
        'Dodano nowego gościa i jego rezerwację',
        jsonb_build_object(
            'gosc', jsonb_build_object(
                'imię', imie,
                'nazwisko', nazwisko,
                'numer_telefonu', numer_telefonu,
                'adres_email', adres_email
            ),
            'rezerwacja', jsonb_build_object(
                'pokoje', pokoje,
                'data_zameldowania', start_date,
                'data_wymeldowania', end_date,
                'liczba_dorosłych', liczba_doroslych,
                'liczba_dzieci', liczba_dzieci,
                'dodatki', dodatki
            )
        )
    );

    -- Informacja o zakończonej operacji
    RAISE NOTICE 'Gość % został dodany z ID: %, a jego rezerwacja została utworzona.', imie || ' ' ||
        nazwisko, new_gosc_id;
END;
$$ LANGUAGE plpgsql;
```

3.5.2 Funkcja: dodaj_rezerwacje

```
CREATE OR REPLACE FUNCTION rezerwacje_hotelowe.dodaj_rezerwacje(
    gosc_id INT,
    pokoje INT[],
    start_date DATE,
    end_date DATE,
    liczba_doroslych INT,
    liczba_dzieci INT,
    dodatki INT[] DEFAULT NULL
)
RETURNS VOID AS $$
DECLARE
    new_rezerwacja_id INT;
    koszt_pokoju NUMERIC;
    pokoj_id INT;
    liczba_dni INT;
BEGIN
    -- Oblicz liczbę dni pobytu
    SELECT (end_date - start_date) INTO liczba_dni;
```

```
    IF liczba_dni <= 0 THEN
        RAISE EXCEPTION 'Nieprawidłowy zakres dat: data wymeldowania musi być późniejsza niż data zameldowania.';
    END IF;

    -- Iteracja przez pokoje
    FOREACH pokoj_id IN ARRAY pokoje LOOP
        -- Dodanie nowej rezerwacji
        INSERT INTO rezerwacje_hotelowe.rezerwacja (
            gość_id, status_płatności_id, status_rezerwacji_id, data_zameldowania, data_wymeldowania,
            liczba_dorosłych, liczba_dzieci, kwota_rezerwacji
        )
        VALUES (
            gosc_id,
            (SELECT id FROM rezerwacje_hotelowe.status_płatności WHERE nazwa_statusu = 'Oczekująca'),
            (SELECT id FROM rezerwacje_hotelowe.status_rezerwacji WHERE nazwa_statusu = 'Oczekująca'),
            start_date, end_date, liczba_doroslych, liczba_dzieci, 0
        )
        RETURNING id INTO new_rezerwacja_id;

        -- Logowanie dodania rezerwacji
        INSERT INTO rezerwacje_hotelowe.logi_systemowe (opis, szczegóły)
        VALUES (
            'Dodano nową rezerwację',
            jsonb_build_object(
                'rezerwacja_id', new_rezerwacja_id,
                'gość_id', gosc_id,
                'pokoje', pokoje,
                'data_zameldowania', start_date,
                'data_wymeldowania', end_date,
                'liczba_dorosłych', liczba_doroslych,
                'liczba_dzieci', liczba_dzieci
            )
        );

        -- Obliczanie kosztów
        koszt_pokoju := rezerwacje_hotelowe.oblicz_koszt_rezerwacji_z_parametrami(pokoje, start_date,
            end_date, dodatki);

        -- Przydzielenie pokoju do rezerwacji
        INSERT INTO rezerwacje_hotelowe.rezerwacja_pokój (rezerwacja_id, pokój_id)
        VALUES (new_rezerwacja_id, pokoj_id);

        -- Dodanie dodatków
        IF dodatki IS NOT NULL THEN
            INSERT INTO rezerwacje_hotelowe.rezerwacja_dodatek (rezerwacja_id, dodatek_id)
            SELECT new_rezerwacja_id, unnest(dodatki);
            INSERT INTO rezerwacje_hotelowe.logi_systemowe (opis, szczegóły)
            VALUES (
                'Dodano dodatki do rezerwacji',
                jsonb_build_object(
                    'rezerwacja_id', new_rezerwacja_id,
                    'dodatki', dodatki
                )
            );
        END IF;

        -- Aktualizacja kosztów rezerwacji
        UPDATE rezerwacje_hotelowe.rezerwacja
        SET kwota_rezerwacji = koszt_pokoju
        WHERE id = new_rezerwacja_id;
    END LOOP;
END;
$$ LANGUAGE plpgsql;
```

3.5.3 Funkcja: oblicz_koszt_rezerwacji_z_parametrami

```
CREATE OR REPLACE FUNCTION rezerwacje_hotelowe.oblicz_koszt_rezerwacji_z_parametrami(
    p_room_ids INT[],        -- Tablica ID pokoi
    p_start_date DATE,       -- Data zameldowania
    p_end_date DATE,         -- Data wymeldowania
    p_addons INT[] DEFAULT NULL -- Opcjonalna tablica dodatków
)
RETURNS NUMERIC AS $$
DECLARE
    total_cost NUMERIC := 0; -- Całkowity koszt
    liczba_dni INT;          -- Liczba dni pobytu
    addons_cost NUMERIC := 0; -- Koszt dodatków
BEGIN
    -- Oblicz liczbę dni pobytu
    SELECT (p_end_date - p_start_date) INTO liczba_dni;

    IF liczba_dni <= 0 THEN
        RAISE EXCEPTION 'Nieprawidłowy zakres dat: data wymeldowania musi być późniejsza niż data zameldowania.';
    END IF;

    -- Oblicz koszt pokoi
```

```
21      SELECT SUM(k.cena_podstawowa * liczba_dni)
22      INTO total_cost
23      FROM rezerwacje_hotelowe.pokój p
24      JOIN rezerwacje_hotelowe.klasa_pokoju k ON p.klasa_pokoju_id = k.id
25      WHERE p.id = ANY(p_room_ids);
26
27      -- Oblicz koszt dodatków, jeśli są podane
28      IF p_addons IS NOT NULL AND array_length(p_addons, 1) > 0 THEN
29          SELECT SUM(d.cena * liczba_dni)
30          INTO addons_cost
31          FROM rezerwacje_hotelowe.dodatek d
32          WHERE d.id = ANY(p_addons);
33      END IF;
34
35      -- Dodaj koszt dodatków do całkowitego kosztu
36      total_cost := total_cost + addons_cost;
37
38      -- Zwróć całkowity koszt
39      RETURN COALESCE(total_cost, 0);
40  END;
41  $$ LANGUAGE plpgsql;
```

Rozdział 4

Projekt funkcjonalny

4.1 Interfejsy do prezentacji, edycji i obsługi danych

Projekt zakłada stworzenie intuicyjnych interfejsów użytkownika, które umożliwią:

- **Prezentację danych:** Wyświetlanie informacji o dostępnych pokojach, rezerwacjach, gościach oraz raportów finansowych.
- **Edycję danych:** Formularze umożliwiające dodawanie, edytowanie oraz usuwanie rezerwacji, gości, pokojów i pracowników.
- **Obsługę danych:** Interfejsy do zarządzania statusami płatności i rezerwacji oraz przypisywania pokoi do rezerwacji.

4.2 Wizualizacja danych

System umożliwia generowanie różnorodnych raportów, takich jak:

- **Liczba rezerwacji na gościa:** Raport przedstawiający liczbę aktywnych i wszystkich rezerwacji dla każdego gościa.
- **Przychód miesięczny:** Raport pokazujący sumę przychodów z rezerwacji w poszczególnych miesiącach.
- **Przychód miesięczny z filtrem:** Raport przedstawiający przychody tylko z tych miesięcy, w których suma przychodów przekroczyła 3000 zł.

4.3 Zdefiniowanie panelu sterowania aplikacji

Panel sterowania aplikacji został zaprojektowany w taki sposób, aby umożliwić:

- **Zarządzanie rezerwacjami:** Tworzenie, edycja, usuwanie oraz anulowanie rezerwacji.
- **Zarządzanie pokojami:** Dodawanie nowych pokoi, aktualizacja statusów oraz przypisywanie do rezerwacji.
- **Zarządzanie typami łóżek:** Dodawanie, edytowanie oraz usuwanie łóżek.
- **Zarządzanie klasami pokojów:** Dodawanie, edytowanie oraz usuwanie łóżek.
- **Zarządzanie gośćmi:** Dodawanie nowych gości oraz aktualizacja ich danych.
- **Generowanie raportów:** Dostęp do raportów finansowych i statystycznych.
- **Zarządzanie pracownikami:** Dodawanie nowych pracowników oraz przypisywanie ról.
- **Zarządzanie wyposażeniem:** Dodawanie, edycja oraz usuwanie wyposażenia.
- **Zarządzanie dodatkami:** Dodawanie, edycja oraz usuwanie dodatków.
- **Zarządzanie logami:** Wersja DEMO przedstawia podstawowy sposób rejestrowania zmian w bazie danych. Pozwala na podgląd dotyczący szczegółów logu, usuwanie logu, bądź całkowitą "eradykacje"logów z bazy danych.

4.4 Makropolecenia

Makropolecenia zostały zaprojektowane, aby ułatwić codzienną obsługę systemu:

- **Automatyczne logowanie operacji:** Wszystkie operacje wykonywane w systemie są automatycznie logowane w tabeli `logi_systemowe`.
- **Automatyczna aktualizacja statusów pokojów:** Przypisanie pokoju do rezerwacji automatycznie zmienia jego status na "Zajęty", a usunięcie rezerwacji lub wykwaterowanie zmienia status na "Dostępny".
- **Walidacja danych:** System automatycznie sprawdza poprawność danych wejściowych, takich jak numer telefonu czy zakres dat.

Rozdział 5

Dokumentacja

5.1 Wprowadzanie danych

Dane mogą być wprowadzane do systemu na kilka sposobów:

- **Ręczne:** Bezpośrednie wpisywanie danych przez interfejs użytkownika.
- **Automatyczne:** Funkcje składowane umożliwiające dodawanie i aktualizację danych poprzez wywołania SQL(np. przy pomocy wcześniej zaprojektowanych triggerów).

5.2 Dokumentacja użytkownika

Dokumentacja użytkownika jest dostępna na stronie w zakładce "Funkcjonalności"po zalogowaniu się jako administrator. Zawiera szczegółowe instrukcje dotyczące korzystania z systemu, zarządzania rezerwacjami, pokojami, gośćmi oraz generowania raportów.

Rozdział 6

Załączniki

6.1 Cały projekt w języku SQL

Poniżej znajduje się pełny skrypt SQL użyty do tworzenia i konfiguracji bazy danych.

Listing 6.1: Projektowy plik SQL

```sql
-- Tworzenie schematu
CREATE SCHEMA IF NOT EXISTS rezerwacje_hotelowe;

-- Dropy istniejących wyzwalaczy, funkcji, widoków, oraz tabel
DROP TRIGGER IF EXISTS trig_aktualizuj_status_pokoju_po_rezerwacji ON rezerwacje_hotelowe.
    rezerwacja_pokój CASCADE;
DROP TRIGGER IF EXISTS trig_aktualizuj_status_po_rezerwacji ON rezerwacje_hotelowe.rezerwacja_pokój
    CASCADE;
DROP TRIGGER IF EXISTS trig_aktualizuj_status_po_wymeldowaniu ON rezerwacje_hotelowe.rezerwacja_pokój
    CASCADE;
DROP TRIGGER IF EXISTS trig_aktualizuj_date_platnosci ON rezerwacje_hotelowe.status_płatności;

DROP FUNCTION IF EXISTS rezerwacje_hotelowe.oblicz_koszt_rezerwacji CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.przypisz_pokoj CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.sprawdz_dostepne_pokoje CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.zmien_status_pokoju CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.dodaj_rezerwacje CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.dodaj_rezerwacje_przez_nowego_goscia CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.dodaj_rezerwacje_przez_goscia_public CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.zaktualizuj_rezerwacje CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.usun_rezerwacje CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.anuluj_rezerwacje CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.wykwateruj_rezerwacje CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.aktualizuj_date_platnosci CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.aktualizuj_status_platnosci CASCADE;
DROP FUNCTION IF EXISTS rezerwacje_hotelowe.oblicz_koszt_rezerwacji_z_parametrami CASCADE;

DROP VIEW IF EXISTS rezerwacje_hotelowe.liczba_rezerwacji_gościa CASCADE;
DROP VIEW IF EXISTS rezerwacje_hotelowe.przychód_miesięczny CASCADE;
DROP VIEW IF EXISTS rezerwacje_hotelowe.przychód_miesięczny_filtr CASCADE;

DROP TABLE IF EXISTS rezerwacje_hotelowe.rezerwacja_pokój CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.pokój CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.piętro CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.status_pokoju CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.typ_łóżka_pokoju_danej_klasy CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.wyposażenie_pokoju_danej_klasy CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.wyposażenie CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.klasa_pokoju CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.typ_łóżka CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.rezerwacja_dodatek CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.dodatek CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.rezerwacja CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.status_płatności CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.status_rezerwacji CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.gość CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.pracownik CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.rola CASCADE;
DROP TABLE IF EXISTS rezerwacje_hotelowe.logi_systemowe CASCADE;

-- Tworzenie tabel
CREATE TABLE rezerwacje_hotelowe.gość (
    id SERIAL PRIMARY KEY,
    imię VARCHAR(200) NOT NULL,
    nazwisko VARCHAR(200) NOT NULL,
    numer_telefonu VARCHAR(20) NOT NULL,
    adres_email VARCHAR(350) NOT NULL,
    CONSTRAINT unikalny_gość_email UNIQUE (adres_email),
    CONSTRAINT chk_numer_telefonu
        CHECK (numer_telefonu ~ '^(\\+[0-9]{1,3})?[0-9]{9}$')
);

CREATE TABLE rezerwacje_hotelowe.status_płatności (
    id SERIAL PRIMARY KEY,
```

```
    nazwa_statusu VARCHAR(50) NOT NULL,
    data_płatności TIMESTAMP
);

CREATE TABLE rezerwacje_hotelowe.status_rezerwacji (
    id SERIAL PRIMARY KEY,
    nazwa_statusu VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE rezerwacje_hotelowe.rezerwacja (
    id SERIAL PRIMARY KEY,
    gość_id INT NOT NULL,
    status_płatności_id INT NOT NULL,
    status_rezerwacji_id INT NOT NULL,
    data_zameldowania DATE NOT NULL,
    data_wymeldowania DATE NOT NULL,
    liczba_dorosłych INT CHECK (liczba_dorosłych >= 0),
    liczba_dzieci INT CHECK (liczba_dzieci >= 0),
    kwota_rezerwacji DECIMAL(10, 2) CHECK (kwota_rezerwacji >= 0),
    CONSTRAINT fk_rezerwacja_gość FOREIGN KEY (gość_id) REFERENCES rezerwacje_hotelowe.gość (id),
    CONSTRAINT fk_rezerwacja_status_płatności FOREIGN KEY (status_płatności_id) REFERENCES
        rezerwacje_hotelowe.status_płatności (id),
    CONSTRAINT fk_rezerwacja_status_rezerwacji FOREIGN KEY (status_rezerwacji_id) REFERENCES
        rezerwacje_hotelowe.status_rezerwacji (id)
);

CREATE TABLE rezerwacje_hotelowe.dodatek (
    id SERIAL PRIMARY KEY,
    nazwa_dodatku VARCHAR(100) NOT NULL,
    cena DECIMAL(10, 2) CHECK (cena >= 0)
);

CREATE TABLE rezerwacje_hotelowe.rezerwacja_dodatek (
    rezerwacja_id INT NOT NULL,
    dodatek_id INT NOT NULL,
    PRIMARY KEY (rezerwacja_id, dodatek_id),
    CONSTRAINT fk_rez_dod_rezerwacja FOREIGN KEY (rezerwacja_id) REFERENCES rezerwacje_hotelowe.
        rezerwacja (id),
    CONSTRAINT fk_rez_dod_dodatek FOREIGN KEY (dodatek_id) REFERENCES rezerwacje_hotelowe.dodatek (id)
);

CREATE TABLE rezerwacje_hotelowe.typ_łóżka (
    id SERIAL PRIMARY KEY,
    nazwa_typu VARCHAR(50) NOT NULL,
    liczba_osob INT NOT NULL DEFAULT 1
);

CREATE TABLE rezerwacje_hotelowe.klasa_pokoju (
    id SERIAL PRIMARY KEY,
    nazwa_klasy VARCHAR(100) NOT NULL,
    cena_podstawowa DECIMAL(10, 2) CHECK (cena_podstawowa >= 0)
);

CREATE TABLE rezerwacje_hotelowe.wyposażenie (
    id SERIAL PRIMARY KEY,
    nazwa_wyposażenia VARCHAR(100) NOT NULL
);

CREATE TABLE rezerwacje_hotelowe.wyposażenie_pokoju_danej_klasy (
    klasa_pokoju_id INT NOT NULL,
    wyposażenie_id INT NOT NULL,
    PRIMARY KEY (klasa_pokoju_id, wyposażenie_id),
    CONSTRAINT fk_klasa_wyposażenie_klasa FOREIGN KEY (klasa_pokoju_id) REFERENCES rezerwacje_hotelowe
        .klasa_pokoju (id),
    CONSTRAINT fk_klasa_wyposażenie_wyposażenie FOREIGN KEY (wyposażenie_id) REFERENCES
        rezerwacje_hotelowe.wyposażenie (id)
);

CREATE TABLE rezerwacje_hotelowe.typ_łóżka_pokoju_danej_klasy (
    id SERIAL PRIMARY KEY,
    klasa_pokoju_id INT NOT NULL,
    typ_łóżka_id INT NOT NULL,
    liczba_łóżek INT CHECK (liczba_łóżek > 0),
    CONSTRAINT fk_klasa_typ_łóżka_klasa FOREIGN KEY (klasa_pokoju_id) REFERENCES rezerwacje_hotelowe.
        klasa_pokoju (id),
    CONSTRAINT fk_klasa_typ_łóżka_typ FOREIGN KEY (typ_łóżka_id) REFERENCES rezerwacje_hotelowe.typ_łó
        żka (id)
);

CREATE TABLE rezerwacje_hotelowe.status_pokoju (
    id SERIAL PRIMARY KEY,
    nazwa_statusu VARCHAR(100) NOT NULL
);

CREATE TABLE rezerwacje_hotelowe.piętro (
    id SERIAL PRIMARY KEY,
    numer_piętra VARCHAR(5) NOT NULL
);

CREATE TABLE rezerwacje_hotelowe.pokój (
    id SERIAL PRIMARY KEY,
```

```
    piętro_id INT NOT NULL,
    klasa_pokoju_id INT NOT NULL,
    status_pokoju_id INT NOT NULL,
    numer_pokoju VARCHAR(10) NOT NULL UNIQUE,
    max_liczba_osob INT,
    CONSTRAINT fk_pokój_piętro FOREIGN KEY (piętro_id) REFERENCES rezerwacje_hotelowe.piętro (id),
    CONSTRAINT fk_pokój_klasa_pokoju FOREIGN KEY (klasa_pokoju_id) REFERENCES rezerwacje_hotelowe.
        klasa_pokoju (id),
    CONSTRAINT fk_pokój_status_pokoju FOREIGN KEY (status_pokoju_id) REFERENCES rezerwacje_hotelowe.
        status_pokoju (id)
);

CREATE TABLE rezerwacje_hotelowe.rezerwacja_pokój (
    rezerwacja_id INT NOT NULL,
    pokój_id INT NOT NULL,
    PRIMARY KEY (rezerwacja_id, pokój_id),
    CONSTRAINT fk_rez_pok_rezerwacja FOREIGN KEY (rezerwacja_id) REFERENCES rezerwacje_hotelowe.
        rezerwacja (id),
    CONSTRAINT fk_rez_pok_pokój FOREIGN KEY (pokój_id) REFERENCES rezerwacje_hotelowe.pokój (id)
);

CREATE TABLE rezerwacje_hotelowe.rola (
    id SERIAL PRIMARY KEY,
    nazwa_roli VARCHAR(50) NOT NULL
);

CREATE TABLE rezerwacje_hotelowe.pracownik (
    id SERIAL PRIMARY KEY,
    imię VARCHAR(200) NOT NULL,
    nazwisko VARCHAR(200) NOT NULL,
    login VARCHAR(100) UNIQUE NOT NULL,
    hasło VARCHAR(256) NOT NULL,
    rola_id INT NOT NULL,
    profile_picture VARCHAR(255) DEFAULT NULL,
    CONSTRAINT fk_pracownik_rola FOREIGN KEY (rola_id) REFERENCES rezerwacje_hotelowe.rola (id)
);

CREATE TABLE rezerwacje_hotelowe.logi_systemowe (
    id SERIAL PRIMARY KEY,
    data TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    opis VARCHAR(500),
    szczegóły JSONB
);
```

6.1.1 Inicjalizacja danych

Poniżej przedstawiono przykładowe dane wstawione do tabel:

```
-- Dodanie przykładowych statusów
INSERT INTO rezerwacje_hotelowe.status_rezerwacji (nazwa_statusu) VALUES
('Oczekująca'),
('Potwierdzona'),
('W trakcie'),
('Zrealizowana'),
('Anulowana');

-- Dodawanie ról
INSERT INTO rezerwacje_hotelowe.rola (nazwa_roli) VALUES ('Administrator'), ('Recepcjonista'), ('
    Manager');

-- Dodawanie statusów płatności
INSERT INTO rezerwacje_hotelowe.status_płatności (nazwa_statusu) VALUES ('Oczekująca'), ('Zrealizowana
    '), ('Anulowana');

-- Dodawanie statusów pokoju
INSERT INTO rezerwacje_hotelowe.status_pokoju (nazwa_statusu) VALUES ('Dostępny'), ('Zajęty'), ('W
    trakcie sprzątania');

-- Dodawanie wyposażenia
INSERT INTO rezerwacje_hotelowe.wyposażenie (nazwa_wyposażenia) VALUES
('Telewizor'),
('Klimatyzacja'),
('Wi-Fi'),
('Minibar');

-- Dodawanie typów łóżek
INSERT INTO rezerwacje_hotelowe.typ_łóżka (nazwa_typu, liczba_osob) VALUES
('Pojedyncze', 1),
('Podwójne', 2),
('Królewskie', 2),
('Kanapa', 1);

-- Dodawanie klas pokoi z wariantami
INSERT INTO rezerwacje_hotelowe.klasa_pokoju (nazwa_klasy, cena_podstawowa) VALUES
('Standard', 200.00),
('Standard z kanapą', 250.00),
('Standard z 3 łóżkami pojedynczymi', 270.00),
('Deluxe', 350.00),
```

```
('Deluxe z 2 łóżkami podwójnymi', 400.00),
('Deluxe z łóżkiem królewskim i pojedynczym', 380.00),
('Apartament', 500.00),
('Apartament 5-osobowy', 600.00),
('Apartament 8-osobowy', 800.00);

-- Dodawanie wyposażenia dla pokoi każdej klasy
INSERT INTO rezerwacje_hotelowe.wyposażenie_pokoju_danej_klasy (klasa_pokoju_id, wyposażenie_id)
    VALUES
(1, 1), -- Standard z Telewizorem
(1, 3), -- Standard z Wi-Fi
(2, 1), -- Standard z kanapą i Telewizorem
(2, 3), -- Standard z kanapą i Wi-Fi
(3, 1), -- Standard z 3 łóżkami pojedynczymi i Telewizorem
(3, 3), -- Standard z 3 łóżkami pojedynczymi i Wi-Fi
(4, 1), -- Deluxe z Telewizorem
(4, 2), -- Deluxe z Klimatyzacją
(4, 3), -- Deluxe z Wi-Fi
(5, 1), -- Deluxe z 2 łóżkami podwójnymi i Telewizorem
(5, 2), -- Deluxe z 2 łóżkami podwójnymi i Klimatyzacją
(5, 3), -- Deluxe z 2 łóżkami podwójnymi i Wi-Fi
(6, 1), -- Deluxe z łóżkiem królewskim i pojedynczym
(6, 2),
(6, 3), -- Deluxe z Wi-Fi
(7, 1), -- Apartament z Telewizorem
(7, 2), -- Apartament z Klimatyzacją
(7, 3), -- Apartament z Wi-Fi
(7, 4), -- Apartament z Minibarem
(8, 1),
(8, 2), -- Apartament 5-osobowy z Telewizorem
(8, 3), -- Apartament 5-osobowy z Wi-Fi
(8, 4),
(9, 1), -- Apartament 8-osobowy z Telewizorem
(9, 2), -- Apartament 8-osobowy z Klimatyzacją
(9, 3), -- Apartament 8-osobowy z Wi-Fi
(9, 4); -- Apartament 8-osobowy z Minibarem

-- Dodawanie typów łóżek do klas pokoi
INSERT INTO rezerwacje_hotelowe.typ_łóżka_pokoju_danej_klasy (klasa_pokoju_id, typ_łóżka_id, liczba_łó
    żek) VALUES
-- Standard
(1, 1, 2), -- Standard z 2 łóżkami pojedynczymi
(2, 4, 1), -- Standard z 1 kanapą
(2, 1, 2),
(3, 1, 3), -- Standard z 3 łóżkami pojedynczymi
-- Deluxe
(4, 2, 1), -- Deluxe z 1 łóżkiem podwójnym
(5, 2, 2), -- Deluxe z 2 łóżkami podwójnymi
(6, 3, 1), -- Deluxe z 1 łóżkiem królewskim
(6, 1, 1), -- Deluxe z 1 łóżkiem pojedynczym
-- Apartament
(7, 3, 1), -- Apartament z 1 łóżkiem królewskim
(8, 1, 5), -- Apartament 5-osobowy z 5 łóżkami pojedynczymi
(9, 1, 8); -- Apartament 8-osobowy z 8 łóżkami pojedynczymi

-- Dodawanie pięter
INSERT INTO rezerwacje_hotelowe.piętro (numer_piętra) VALUES ('1'), ('2'), ('3');

-- Dodawanie pokoi
INSERT INTO rezerwacje_hotelowe.pokój (piętro_id, klasa_pokoju_id, status_pokoju_id, numer_pokoju)
    VALUES
-- Standard
(1, 1, 1, '101'),
(1, 2, 1, '102'), -- Standard z kanapą
(1, 3, 1, '103'), -- Standard z 3 łóżkami pojedynczymi
-- Deluxe
(2, 4, 1, '201'), -- Deluxe
(2, 5, 1, '202'), -- Deluxe z 2 łóżkami podwójnymi
(2, 6, 1, '203'), -- Deluxe z łóżkiem królewskim i pojedynczym
-- Apartament
(3, 7, 1, '301'), -- Apartament
(3, 8, 1, '302'), -- Apartament 5-osobowy
(3, 9, 1, '303'); -- Apartament 8-osobowy

-- Dodawanie dodatków
INSERT INTO rezerwacje_hotelowe.dodatek (nazwa_dodatku, cena) VALUES (' niadanie ', 30.00), ('Parking'
    , 20.00), ('Spa', 100.00);

-- Dodawanie gości
INSERT INTO rezerwacje_hotelowe.gość (imię, nazwisko, numer_telefonu, adres_email) VALUES
('Adam', 'Nowak', '123456789', 'adam.nowak@example.com'),
('Ewa', 'Kowalska', '987654321', 'ewa.kowalska@example.com');

-- Dodawanie rezerwacji
INSERT INTO rezerwacje_hotelowe.rezerwacja (gość_id, status_płatności_id, status_rezerwacji_id,
    data_zameldowania, data_wymeldowania, liczba_dorosłych, liczba_dzieci, kwota_rezerwacji)
VALUES
(1, 1, (SELECT id FROM rezerwacje_hotelowe.status_rezerwacji WHERE nazwa_statusu = 'Oczekująca'), '
    2023-12-01', '2023-12-05', 2, 0, 800.00),
(2, 2, (SELECT id FROM rezerwacje_hotelowe.status_rezerwacji WHERE nazwa_statusu = 'Potwierdzona'), '
    2023-12-10', '2023-12-15', 2, 1, 1500.00);
```

```
122
123 -- Przypisywanie pokoi do rezerwacji
124 INSERT INTO rezerwacje_hotelowe.rezerwacja_pokój (rezerwacja_id, pokój_id) VALUES
125 (1, 1),
126 (2, 3);
127
128 -- Przypisywanie dodatków do rezerwacji
129 INSERT INTO rezerwacje_hotelowe.rezerwacja_dodatek (rezerwacja_id, dodatek_id) VALUES
130 (1, 1), --  niadanie  do rezerwacji 1
131 (1, 2), -- Parking do rezerwacji 1
132 (2, 1), --  niadanie  do rezerwacji 2
133 (2, 3); -- Spa do rezerwacji 2
134
135 -- Dodawanie pracowników
136 INSERT INTO rezerwacje_hotelowe.pracownik (imię, nazwisko, login, hasło, rola_id)
137 VALUES
138 ('Dawid', 'Piotrowski', 'dawid.piotrowski', md5('admin123'),
139  (SELECT id FROM rezerwacje_hotelowe.rola WHERE nazwa_roli = 'Administrator')),
140 ('Ewa', 'Kowalska', 'ewa.kowalska', md5('recep123'),
141  (SELECT id FROM rezerwacje_hotelowe.rola WHERE nazwa_roli = 'Recepcjonista')),
142 ('Jan', 'Nowak', 'jan.nowak', md5('manager123'),
143  (SELECT id FROM rezerwacje_hotelowe.rola WHERE nazwa_roli = 'Manager')),
144 ('Anna', 'Wiśniewska', 'anna.wisniewska', md5('recep456'),
145  (SELECT id FROM rezerwacje_hotelowe.rola WHERE nazwa_roli = 'Recepcjonista'));
```

6.1.2 Tworzenie widoków

```
1  -- Tworzenie widoków
2  CREATE OR REPLACE VIEW rezerwacje_hotelowe.liczba_rezerwacji_gościa AS
3  SELECT
4      g.id AS gość_id,
5      g.imię,
6      g.nazwisko,
7      COUNT(r.id) FILTER (WHERE sr.nazwa_statusu NOT IN ('Anulowana')) AS liczba_aktywnych_rezerwacji,
8      COUNT(r.id) AS liczba_wszystkich_rezerwacji
9  FROM
10     rezerwacje_hotelowe.gość g
11 LEFT JOIN
12     rezerwacje_hotelowe.rezerwacja r ON g.id = r.gość_id
13 LEFT JOIN
14     rezerwacje_hotelowe.status_rezerwacji sr ON r.status_rezerwacji_id = sr.id
15 GROUP BY
16     g.id, g.imię, g.nazwisko;
17
18 CREATE OR REPLACE VIEW rezerwacje_hotelowe.przychód_miesięczny AS
19 SELECT
20     TO_CHAR(DATE_TRUNC('month', r.data_zameldowania), 'YYYY-MM') AS miesiąc,
21     SUM(r.kwota_rezerwacji) AS suma_przychodów
22 FROM rezerwacje_hotelowe.rezerwacja r
23 JOIN rezerwacje_hotelowe.status_rezerwacji sr ON r.status_rezerwacji_id = sr.id
24 WHERE sr.nazwa_statusu = 'Zrealizowana'
25 GROUP BY DATE_TRUNC('month', r.data_zameldowania)
26 ORDER BY DATE_TRUNC('month', r.data_zameldowania);
27
28 CREATE OR REPLACE VIEW rezerwacje_hotelowe.przychód_miesięczny_filtr AS
29 SELECT
30     TO_CHAR(DATE_TRUNC('month', r.data_zameldowania), 'YYYY-MM') AS miesiąc,
31     SUM(r.kwota_rezerwacji) AS suma_przychodów
32 FROM rezerwacje_hotelowe.rezerwacja r
33 JOIN rezerwacje_hotelowe.status_rezerwacji sr ON r.status_rezerwacji_id = sr.id
34 WHERE sr.nazwa_statusu = 'Zrealizowana'
35 GROUP BY miesiąc
36 HAVING SUM(r.kwota_rezerwacji) > 3000
37 ORDER BY miesiąc;
```

Rozdział 7

Podsumowanie i uwagi końcowe

7.1 Walidacje i Constraints

System zawiera liczne walidacje i ograniczenia danych, które zapewniają integralność oraz poprawność danych:

- **Unikalność:** Unikalne e-maile gości, loginy pracowników oraz numery pokojów.
- **Formaty danych:** Sprawdzanie poprawności numerów telefonów.
- **Zakresy wartości:** Ograniczenia na liczby dorosłych, dzieci oraz kwoty rezerwacji.
- **Zależności referencyjne:** Klucze obce zapewniające powiązania między tabelami.

7.2 Logowanie

Wszystkie operacje związane z zarządzaniem rezerwacjami, gośćmi, pokojami oraz pracownikami są logowane w tabeli `logi_systemowe`. Umożliwia to śledzenie zmian oraz audytowanie działań w systemie.

7.3 Automatyczne aktualizacje statusów pokojów

Dzięki wyzwalaczom (triggers), przypisanie pokoju do rezerwacji automatycznie zmienia jego status na "Zajęty". Usunięcie rezerwacji lub wykwaterowanie zmienia status pokoju na "Dostępny".

7.4 Generowanie raportów

System umożliwia generowanie raportów finansowych poprzez widoki `przychód_miesięczny` oraz `przychód_miesięczny_`, co pozwala na analizę przychodów w poszczególnych miesiącach oraz identyfikację okresów o wysokim przychodzie. Natomiast na stronie hotelowej jest to możliwe dzięki bibliotece **Charts.js**.

7.5 Bezpieczeństwo danych

Zastosowane funkcje składowane oraz wyzwalacze zabezpieczają dane przed nieautoryzowanymi modyfikacjami oraz zapewniają poprawność wprowadzanych informacji. Hasła pracowników są przechowywane jako hash (funkcja `md5`).

7.6 Automatyczne aktualizacje statusów rezerwacji

Aby uwzględnić rezerwacje w raportach finansowych, statusy płatności oraz statusy rezerwacji są aktualizowane na "Zrealizowana"za pomocą dedykowanych funkcji. Umożliwia to automatyczne uwzględnienie tych rezerwacji w raportach.

Rozdział 8

Przykłady wywołań

8.1 Przykładowe zapytania testowe

Poniżej przedstawiono przykładowe zapytania testowe używane do weryfikacji funkcjonalności systemu.

8.1.1 Dodawanie rezerwacji przez nowego gościa

```
SELECT rezerwacje_hotelowe.dodaj_rezerwacje_przez_nowego_goscia(
    'Marcin',                                -- imię
    'Grabowski',                             -- nazwisko
    '999123456',                             -- numer telefonu (9 cyfr OK)
    'marcin.grab@example.com',               -- adres e-mail
    ARRAY[3],                                 -- pokoje: np. pokój_id = 3
    '2024-03-10',                             -- data zameldowania
    '2024-03-14',                             -- data wymeldowania
    2,                                       -- liczba dorosłych
    1,                                       -- liczba dzieci
    ARRAY[1,3]                                -- dodatki: np. id=1 (Sniadanie), id=3 (Spa)
);
```

8.1.2 Dodawanie rezerwacji przez istniejącego gościa

```
SELECT rezerwacje_hotelowe.dodaj_rezerwacje_przez_goscia_public(
    'Adam',                                  -- imię
    'Nowak',                                 -- nazwisko
    '+48123456789',                          -- numer telefonu
    'adam.nowak@example.com',                -- adres e-mail
    ARRAY[2,3],                               -- rezerwuje dwa pokoje na raz
    '2024-07-01',                             -- data zameldowania
    '2024-07-05',                             -- data wymeldowania
    2,                                       -- liczba dorosłych
    2,                                       -- liczba dzieci
    ARRAY[1,2,3]                              -- trzy różne dodatki
);
```

8.1.3 Przykłady niepowodzeń

```
-- Próba z błędnym numerem telefonu (7 cyfr)
-- SELECT rezerwacje_hotelowe.dodaj_rezerwacje_przez_nowego_goscia(
--     'Zbigniew',
--     'Niedziela',
--     '1234567',          -- tylko 7 cyfr, constraint powinien wyrzucić błąd
--     'z.niedziela@example.com',
--     ARRAY[1],
--     '2024-05-01',
--     '2024-05-05',
--     2,
--     0,
--     NULL
-- );
```

Listing 8.1: Próba dodania rezerwacji z niepoprawnym numerem telefonu

```
-- Próba z niepoprawną datą (end_date <= start_date)
-- SELECT rezerwacje_hotelowe.dodaj_rezerwacje_przez_nowego_goscia(
--     'Helena',
--     'Sowa',
--     '+48988999111',
--     'helena.sowa@example.com',
--     ARRAY[2],
--     '2024-06-10',  -- start
```

```
--      '2024-06-09',  -- koniec dzień wcześniej => błąd
--      2,
--      2,
--      ARRAY[2]
-- );
```

Listing 8.2: Próba dodania rezerwacji z niepoprawnym zakresem dat

```
-- Próba z dodatkiem, który nie istnieje
-- SELECT rezerwacje_hotelowe.dodaj_rezerwacje_przez_goscia_public(
--      'Adam',
--      'Nowak',
--      '+48123456789',
--      'adam.nowak@example.com',
--      ARRAY[1],
--      '2024-11-10',
--      '2024-11-15',
--      2,
--      1,
--      ARRAY[999]                    -- dodatek_id=999 nie istnieje
-- );
```

Listing 8.3: Próba dodania rezerwacji z nieistniejącym dodatkiem

8.2 Przykładowe wywołania funkcji

8.2.1 Dodawanie rezerwacji

Listing 8.4: Dodawanie rezerwacji przez nowego gościa

```
SELECT rezerwacje_hotelowe.dodaj_rezerwacje_przez_nowego_goscia(
    'Marcin',                              -- imię
    'Grabowski',                           -- nazwisko
    '999123456',                           -- numer telefonu (9 cyfr OK)
    'marcin.grab@example.com',             -- adres e-mail
    ARRAY[3],                               -- pokoje: pokój_id = 3
    '2024-03-10',                           -- data zameldowania
    '2024-03-14',                           -- data wymeldowania
    2,                                     -- liczba dorosłych
    1,                                     -- liczba dzieci
    ARRAY[1,3]                              -- dodatki: id=1 (Sniadanie), id=3 (Spa)
);
```

8.2.2 Modyfikacja rezerwacji

Listing 8.5: Modyfikacja rezerwacji

```
SELECT rezerwacje_hotelowe.zaktualizuj_rezerwacje(
    1,                      -- rezerwacja_id
    1,                      -- gość_id = 1 (Adam Nowak)
    '2023-12-02',           -- nowa data zameldowania
    '2023-12-06',           -- nowa data wymeldowania
    ARRAY[2],               -- zmiana pokoju na pokój_id=2
    2,                      -- status_płatności_id
    2,                      -- status_rezerwacji_id (np. 'Potwierdzona')
    1300.00,               -- tymczasowa kwota
    ARRAY[1,2]              -- dodatki: Sniadanie + Parking
);
```

8.2.3 Anulowanie rezerwacji

```
SELECT rezerwacje_hotelowe.anuluj_rezerwacje(3);
```

Listing 8.6: Anulowanie rezerwacji

8.2.4 Wykwaterowanie rezerwacji

```
SELECT rezerwacje_hotelowe.wykwateruj_rezerwacje(2);
```

Listing 8.7: Wykwaterowanie rezerwacji

8.2.5 Zmiana statusu pokoju

```
SELECT rezerwacje_hotelowe.zmien_status_pokoju(1, 'W trakcie sprzątania');
```

Listing 8.8: Zmiana statusu pokoju

8.2.6 Przykład wywołania widoków

```
SELECT *
FROM rezerwacje_hotelowe.liczba_rezerwacji_gościa;
```

Listing 8.9: Wywołanie widoku liczba_rezerwacji_gościa

```
SELECT *
FROM rezerwacje_hotelowe.przychód_miesięczny;
```

Listing 8.10: Wywołanie widoku przychód_miesięczny

```
SELECT *
FROM rezerwacje_hotelowe.przychód_miesięczny_filtr;
```

Listing 8.11: Wywołanie widoku przychód_miesięczny_filtr