**Dokumentacja projektu wykonanego w ramach przedmiotu   
BAZY DANYCH I**

**Hotel**

**Wykonał Kamil Sudoł**

**I. Projekt koncepcji, założenia**

**1. Zdefiniowanie tematu projektu:**

Tematem projektu jest obsługa bazy danych dla spraw hotelarskich. Dzięki napisanej aplikacji możliwe byłoby obsłużenie podstawowych działań związanych z branżą hotelarską, takich, jak możliwość dokonania rezerwacji danego pokoju, możliwość opłaty bądź rezygnacji z rezerwacji itp. Graficzny interfejs użytkownika w całości został napisany w języku Java. Wykorzystując przy tym rozne poziomy autoryzacji

**2. Analiza wymagań użytkownika:**

Użytkownik zalogowany jako klient może dokonać rezerwacji na jeden konkretny pokój z wielu dostępnych o określonej kategorii, w konkretnym terminie (o ile wybrany pokój będzie dostępny w tymże terminie) na konkretną liczbę osób uwzględniając podział na dzieci i dorosłych, mając przy tym możliwość wyboru dodatkowych usług, takich, jak dostęp do basenu, kręgielni, bilarda czy siłowni. Ponadto klient ma możliwość podglądu swoich rezerwacji, wyświetlenia szczegółowych informacji każdej z nich oraz możliwość zdecydowania, czy daną rezerwację klient chce opłacić bądź zrezygnować z niej.

Użytkownik zalogowany jako headadmin lub pracownik poza możliwościami wymienionymi dla klientów ma udostępniony panel administratora. Na omawiany panel składa się możliwość dodania nowego pracownika do obsługi hotelu a także możliwość podglądu wszystkich raportów oraz tabel hotelowej bazy danych.

**3. Zaprojektowanie funkcji:**

**Funkcje:**

*Numeric\_check* – pomocnicza funkcja wykorzystywana przy walidacji danych rejestracyjnych sprawdzająca czy podany numer telefonu składa się z samych cyfr.

*Get\_pokoje* – funkcja zwracająca z bazy listę dostępnych pokoi na podany termin i deklarowaną liczbę osób. Do tego celu został użyty kursor iterujący po tabeli *pokoj,* gdzie w pętli sprawdzany był warunek, czy wskazywany pokój na podany termin przypisany jest już do jakiejś rezerwacji – jeżeli nie, wówczas id pokoju było zapisywane do zwracanej tabeli.

*Get\_liczba\_dni* – pomocnicza funkcja wyznaczająca przedział czasowy pomiędzy podanymi datami.

*Last\_rezerwacja\_id* – funkcja zwracająca z bazy numer id ostatniej złożonej rezerwacji. W tym celu wykorzystano funkcję agregującą *MAX()*, która została wywołana na atrybucie *rezerwacja\_id* z tabeli *rezerwacje*.

*Last\_uzytkownik\_id* – funkcja zwracająca z bazy numer id ostatniego zarejestrowanego użytkownika. W tym celu wykorzystano funkcję agregującą *MAX()*, która została wywołana na atrybucie *uzytkownik\_id* z tabeli *uzytkownik*.

*OplataZaplac* – funkcja aktualizująca wartość atrybutu *status\_czy\_oplacone* na wartość *‘Oplacone’* krotki z tabeli *oplata* o podanym numerze id.

*OplataRezygnuj* – funkcja aktualizująca wartość atrybutu *status\_czy\_oplacone* na wartość *‘Nieoplacone - rezygnacja’* krotki z tabeli *oplata* o podanym numerze id.

*ZbanowanyCheck* – funkcja sprawdzająca, czy użytkownik o podanym id został z jakiegoś powodu zablokowany. W tym celu sprawdzane jest funkcja agregującą *COUNT()*, czy podany id sprawdzanego użytkownika znajduje się w tabeli *czarna\_lista*.

*Autoryzacja* – funkcja sprawdzająca, czy logujący się użytkownik jest klientem, pracownikiem czy headadminem. W tym celu sprawdzany jest atrybut *typ* z tabeli *uzytkownik*.

*StartUpdate* – funkcja wywoływana na bazie danych przy każdym starcie aplikacji. Funkcja ta wykorzystuje kursor iterujący po widoku *OplataRezerwacje* sprawdzając między innymi:  
- czy dzisiejszego dnia następuje jakieś zakwaterowanie,  
- czy dzisiejszego dnia następuje jakieś wykwaterowanie,  
- czy dzisiejszego dnia mija termin opłaty rezerwacji któregoś z klientów.

W przypadku zakwaterowania czy wykwaterowania modyfikowana jest krotka z tabeli *zakwaterowani\_goscie\_info* aktualizująca wartość atrybutu *status\_czy\_zakwaterowany*, zaś w przypadku braku uregulowania opłaty w terminie użytkownik ten automatycznie jest przenoszony na czarną listę, innymi słowy wykonywany jest *INSERT* do tabeli *czarna\_lista* zawierająca id tegoż użytkownika oraz powód o treści *‘Nieoplacenie rezerwacji w terminie.’*.

**Wyzwalacze:**

*Register\_validator* – wyzwalacz uruchamiany przed *INSERT’em* do tabeli *użytkownik*. Wyzwalacz ten sprawdza poprawność wprowadzonych danych, czyli sprawdza, czy wszystkie pola są wypełnione, czy adres email jest poprawny bądź unikatowy oraz czy podany numer telefonu jest składa się z cyfr. W przypadku niepowodzenia do aplikacji wysyłany jest odpowiedni komunikat z bazy.

Panel\_validator – wyzwalacz uruchamiany przed *INSERT’em* do tabeli *panel*. Wyzwalacz ten sprawdza poprawność wprowadzonych danych, czyli sprawdza, czy wszystkie pola są wypełnione, czy login oraz hasło są dłuższe, niż 5 znaków oraz czy nazwa użytkownika jest unikatowa. W przypadku niepowodzenia do aplikacji wysyłany jest odpowiedni komunikat z bazy.

*Platnosc\_rezerwacja* – wyzwalacz uruchamiany po *INSER’cie* do tabeli *rezerwacje*. Wyzwalacz ten oblicza sumaryczny koszt pobytu w zależności od ilości dni uwzględniając dzieci oraz dorosłych, wykonując *INSERT* do tabeli *oplata*.

*Platnosc\_update* – wyzwalacz uruchamiany po *INSER’cie* do tabeli *dodatkowe\_uslugi*. Wyzwalacz ten aktualizuje sumaryczny koszt pobytu w hotelu z tabeli *oplata* o koszt za dodatkowe usługi uwzględniając dzieci oraz dorosłych.

*OplataStatus* – wyzwalacz uruchamiany przed *UPDATE’em* do tabeli *oplata*. Wyzwalacz ten odpowiedzialny jest za aktualizacje statusu opłaty. W celu uniknięcia problemów z aktualizacjami wynikającymi z dwóch wcześniej opisanych wyzwalaczy sprawdzane jest najpierw, czy nowa kwota rożni się od kwoty poprzedniej, wówczas wyzwalacze *Platnosc\_rezerwacje* oraz *Platnosc\_update* mogą spokojnie się wykonać. W przeciwnym wypadku sprawdzany jest warunek, czy stary status jest określony jako *‘Nieoplacone’*. Jeżeli tak, wówczas w przypadku nowego statusu *‘Oplacone’* wykonywany jest *INSERT* do tabeli *zakwaterowani\_goscie\_info* zawierający id rezerwacji wraz z informacja *‘Oczekiwanie na zakwaterowanie’*, zaś w przypadku nowego statusu *‘Nieoplacone - rezygnacja’* wykonywany jest *INSERT* do tabeli *rezygnacja\_z\_rezerwacji\_info* zawierający id rezerwacji oraz id użytkownika. Gdy jednak żaden z wyżej wymienionych warunków nie jest spełniony, do aplikacji wysyłany jest komunikat informujący o braku możliwości modyfikacji wcześniej zmodyfikowanego statusu opłaty rezerwacji.

*RezerwacjaNaTydzienPrzed* – wyzwalacz uruchamiany przed *INSERT’em* do tabeli *rezerwacje*. Wyzwalacz ten sprawdza, czy wybrany termin zakwaterowania jest określony na więcej niż siedem dni od dnia złożenia rezerwacji. W przeciwnym wypadku do aplikacji wysyłany jest odpowiedni komunikat z bazy.

*RezerwacjaOsobyValidator* - wyzwalacz uruchamiany przed *INSERT’em* do tabeli *rezerwacje*. Wyzwalacz ten sprawdza, czy podana liczba osób w rezerwacji jest poprawna. Jeżeli sumaryczna liczba osób wynosi zero bądź przekracza pojemność wybranego pokoju, wówczas do aplikacji wysyłany jest odpowiedni komunikat z bazy.

*RezygnacjaChecker* – wyzwalacz uruchamiany po *INSER’cie* do tabeli *rezygnacja\_z\_rezerwacji\_info*. Wyzwalacz ten sprawdza liczbę rekordów w tejże tabeli dla danego użytkownika. Jeżeli liczba rezygnacji przekroczy 5 rekordów, wówczas użytkownik zostaje dodany na czarną listę, czyli wykonywany jest *INSERT* do tabeli *czarna\_lista* zawierający id użytkownika oraz powód o treści *‘Ciagle rezygnowanie ze skladanych rezerwacji.’*.

*CzarnaListaChecker* - wyzwalacz uruchamiany przed *INSERT’em* do tabeli *czarna\_lista*. Wyzwalacz ten jest głównie używany w czasie manualnego blokowania użytkownika przez administratora – sprawdzane jest, czy powód blokady jest uzupełniony oraz czy blokowany użytkownik nie jest headadminem lub pracownikiem – w przeciwnym wypadku do aplikacji zostanie wysłany odpowiedni komunikat z bazy.

TransakcjaRezerwacjaChecker – wyzwalacz uruchamiany przed *INSERT’em* do tabeli *dodatkowe\_uslugi*. Wyzwalacz ten jest dosyć prostym i prymitywnym zabezpieczeniem przed wymieszaniem się danych w momencie, kiedy dwie osoby w tym samym czasie próbują dokonać rezerwacji na ten sam pokój w pokrywających się terminach. Jeżeli warunki te zostaną spełnione, do aplikacji zostanie wysłany odpowiedni komunikat z bazy. Dlaczego wyzwalacz ten jest podpięty pod tabelę *dodatkowe\_uslugi*? Otóż proces rezerwacji został zaimplementowany jako procedura transakcji, gdzie INSERT do tabeli *dodatkowe\_uslugi* wykonuje się zaraz po *INSER’cie* do tabeli *rezerwacje* i INSERT ten jest ściśle związany z poprzednim poprzez wartość atrybutu *rezerwacja\_id*. Jest to mało prawdopodobny scenariusz, jednakże, jeżeli w tym samym momencie dwie osoby przy wyżej wymienionych warunkach będą chciały dokonać rezerwacji, wówczas bez tego zabezpieczenia może nastąpić przypisanie informacji o dodatkowych usługach do zupełnie innej rezerwacji.

TransakcjaRejestracjChecker – wyzwalacz uruchamiany przed *INSERT’em* do tabeli *panel*. Wyzwalacz ten spełnia podobną rolę, co wyzwalacz opisany powyżej. Sprawdzane jest, czy nie następuje dopisanie informacji do istniejącego już użytkownika, a w przypadku spełnienia warunku do aplikacji zostanie wysłany odpowiedni komunikat z bazy.

**II. Projekt diagramów (konceptualny)**

**4. Budowa i analiza diagramu przepływu danych (DFD):**

**5. Zdefiniowanie encji (obiektów) oraz ich atrybutów:**

***Panel:*** *uzyt\_id*, *login*, *haslo*,

***Uzytkownik***: *uzytkownik\_id*, *imie*, *nazwisko*, *e\_mail*, *numer*, *typ*,

***Rezerwacje***: *rezerwacja\_id*, *uzytkownik\_id*, *pokoj\_id*, *data\_rezerwacji*, *od\_kiedy*, *do\_kiedy*, *liczba\_dzieci*, *liczba\_doroslych*,

***Dodatkowe\_uslugi***: *dodatkowe\_uslugi\_id*, *nazwa\_uslugi*, *cena\_od\_osoby*, *rezerwacja\_id*,

***Oplata***: *oplata\_id*, *rezerwacja\_id,* *status\_czy\_oplacone*, *kwota*,

***Pokoj***: *pokoj\_id*, *numer\_pokoju*, *pietro, liczba\_miejsc*, *kategoria\_id*,

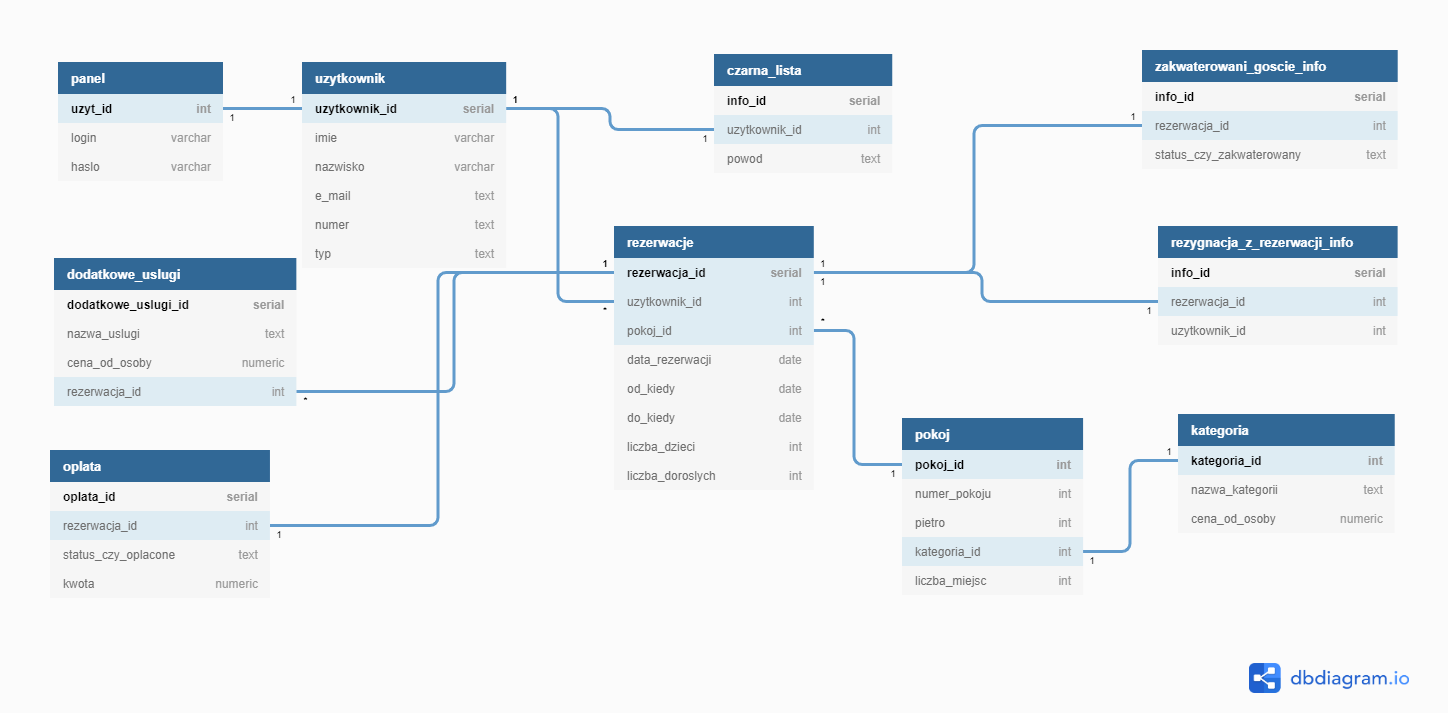
***Kategoria***: *kategoria\_id*, *nazwa\_kategorii*, *cena\_od\_osoby*,

***Rezygnacja\_z\_rezerwacji\_info:*** *info\_id*, *rezerwacja\_id*, *uzytkownik\_id*,

***Zakwaterowani\_goscie\_info:*** *info\_id*, *rezerwacja\_id*, *status\_czy\_zakwaterowany*,

**C*zarna\_lista***: *info\_id*, *uzytkownik\_id, powod.*

**6. Zaprojektowanie relacji pomiędzy encjami:**

****

**III. Projekt logiczny**

**7. Projektowanie tabel, kluczy, indeksów:**

Schemat ***projekt***:

- tabela ***panel:***

- *uzyt\_id* INT PRIMARY KEY,

- *login* VARCHAR,

- *haslo* VARCHAR

- tabela ***uzytkownik***:

- *uzytkownik\_id* SERIAL PRIMARY KEY,

- *imie* VARCHAR,

- *nazwisko* VARCHAR,

- *e\_mail* TEXT,

- *numer* TEXT,

- *typ* TEXT

- tabela ***rezerwacje***:

- *rezerwacja\_id* SERIAL PRIMARY KEY,

- *uzytkownik\_id* INT,

- *pokoj\_id* INT,

- *data\_rezerwacji* DATE,

- *od\_kiedy* DATE,

- *do\_kiedy* DATE,

- *liczba\_dzieci* INT,

- *liczba\_doroslych* INT

- tabela ***dodatkowe\_uslugi***:

- *dodatkowe\_uslugi\_id* SERIAL PRIMARY KEY,

- *nazwa\_uslugi* TEXT,

- *cena\_od\_osoby* NUMERIC,

- *rezerwacja\_id* INT,

- tabela ***oplata***:

- *oplata\_id* SERIAL PRIMARY KEY,

- *rezerwacja\_id* INT,

- *status\_czy\_oplacone* TEXT,

- *kwota* NUMERIC

- tabela ***pokoj***:

- *pokoj\_id* INT PRIMARY KEY,

- *numer\_pokoju* INT,

*- pietro INT,*

- *liczba\_miejsc* INT,

- *kategoria\_id* INT

- tabela ***kategoria***:

- *kategoria\_id* INT PRIMARY KEY,

- *nazwa\_kategorii* TEXT,

- *cena\_od\_osoby* NUMERIC

- tabela ***rezygnacja\_z\_rezerwacji\_info:***

- *info\_id* SERIAL PRIMARY KEY,

- *rezerwacja\_id* INT,

- *uzytkownik\_id* INT

- tabela ***zakwaterowani\_goscie\_info:***

- *info\_id* SERIAL PRIMARY KEY,

- *rezerwacja\_id* INT,

- *status\_czy\_zakwaterowany* TEXT

- tabela ***czarna\_lista***:

- *info\_id* SERIAL PRIMARY KEY,

- *uzytkownik\_id* INT,

- *powod* TEXT

Panel (uzyt\_id) klucz obcy do Uzytkownik (uzytkownik\_id).

Rezerwacje (uzytkownik\_id) klucz obcy do Uzytkownik (uzytkownik\_id).

Rezerwacje (pokoj\_id) klucz obcy do Pokoj (pokoj\_id).

Dodatkowe\_uslugi (rezerwacja\_id) klucz obcy do Rezerwacje (rezerwacja\_id).

Oplata (rezerwacja\_id) klucz obcy do Rezerwacje (rezerwacja\_id).

Pokoj (kategoria\_id) klucz obcy do Kategoria (kategoria\_id).

Zakwaterowani\_goscie\_info (rezerwacja\_id) klucz obcy do Rezerwacje (rezerwacja\_id).

Rezygnacja\_z\_rezerwacji\_info (rezerwacja\_id) klucz obcy do Rezerwacje (rezerwacja\_id).

Czarna\_lista (uzytkownik\_id) klucz obcy do Uzytkownik (uzytkownik\_id).

**8. Słowniki danych:**

Schemat ***projekt***:

- tabela ***panel:***

- *uzyt\_id* INT PRIMARY KEY > 0,

- *login* VARCHAR > 5,

- *haslo* VARCHAR > 5

- tabela ***uzytkownik***:

- *uzytkownik\_id* SERIAL PRIMARY KEY > 0,

- *imie* VARCHAR,

- *nazwisko* VARCHAR,

- *e\_mail* TEXT,

- *numer* TEXT,

- *typ* TEXT

- tabela ***rezerwacje***:

- *rezerwacja\_id* SERIAL PRIMARY KEY > 0,

- *uzytkownik\_id* INT > 0,

- *pokoj\_id* INT > 0,

- *data\_rezerwacji* DATE,

- *od\_kiedy* DATE,

- *do\_kiedy* DATE,

- *liczba\_dzieci* INT >= 0,

- *liczba\_doroslych* INT >= 0

- tabela ***dodatkowe\_uslugi***:

- *dodatkowe\_uslugi\_id* SERIAL PRIMARY KEY > 0,

- *nazwa\_uslugi* TEXT,

- *cena\_od\_osoby* NUMERIC >= 0,

- *rezerwacja\_id* INT > 0,

- tabela ***oplata***:

- *oplata\_id* SERIAL PRIMARY KEY > 0,

- *rezerwacja\_id* INT > 0,

- *status\_czy\_oplacone* TEXT,

- *kwota* NUMERIC >= 0

- tabela ***pokoj***:

- *pokoj\_id* INT PRIMARY KEY > 0,

- *numer\_pokoju* INT > 0,

*- pietro INT* >= 0*,*

- *liczba\_miejsc* INT > 0,

- *kategoria\_id* INT > 0

- tabela ***kategoria***:

- *kategoria\_id* INT PRIMARY KEY > 0,

- *nazwa\_kategorii* TEXT,

- *cena\_od\_osoby* NUMERIC >= 0

- tabela ***rezygnacja\_z\_rezerwacji\_info:***

- *info\_id* SERIAL PRIMARY KEY > 0,

- *rezerwacja\_id* INT > 0,

- *uzytkownik\_id* INT > 0

- tabela ***zakwaterowani\_goscie\_info:***

- *info\_id* SERIAL PRIMARY KEY > 0,

- *rezerwacja\_id* INT > 0,

- *status\_czy\_zakwaterowany* TEXT

- tabela ***czarna\_lista***:

- *info\_id* SERIAL PRIMARY KEY > 0,

- *uzytkownik\_id* INT > 0,

- *powod* TEXT

**9. Analiza zależności funkcyjnych i normalizacja tabel:**

Pierwsza postać normalna (1NF):

W każdej tabeli jest jednoznacznie definiujący encję klucz główny. Wszystkie dane są atomowe.

Druga postać normalna (2NF):

Każda tabela przechowuje dane dotyczące tylko konkretnej klasy obiektów. Każdy niekluczowy atrybut jest nieredukowalnie zależny od klucza głównego.

Trzecia postać normalna (3NF):

Wszystkie niekluczowe atrybuty (niewchodzące w skład klucza głównego) są

**10. Denormalizacja struktury tabel:**

**11. Zaprojektowanie operacji na danych:**

**IV. Projekt funkcjonalny**

**12.Interfejsy do prezentacji, edycji i obsługi danych:**

**13.Wizualizacja danych:**

**14.Zdefiniowanie panelu sterowania aplikacji:**

**15.Makropolecenia:**

**V. Dokumentacja**

**16. Wprowadzanie danych:**

**17. Dokumentacja użytkownika:**

**18.Opracowanie dokumentacji technicznej:**

Kod został udokumentowany w kodzie źródłowym.

**19.Wykaz literatury.**

- <https://dbdiagram.io/>,

- <https://newton.fis.agh.edu.pl/~antek/index.php?sub=db_wykl>,

- <https://www.postgresql.org/docs/>,

- <https://docs.oracle.com/javase/8/docs/>,

- <https://stackoverflow.com/>,

OplataRezerwacje, która jest zlaczeniem tabel oplata oraz rezerwacja po atrybucie rezerwacja\_id