**SPRAWOZDANIE FINALNE**

Zajęcia: Bazy danych II

Prowadzący: prof. ATH dr hab. inż. Andrzej Nowak

# Rezerwacja Hotelowa

Dariusz Cebrat
Informatyka I stopień,
stacjonarne,
4 semestr,
Gr.1B

# 1. Określenie Celu i wymagań

## Cel

Celem jest stworzenie aplikacji bazodanowej przechowującej oraz przetwarzające informacje na temat rezerwacji pokoi przez użytkowników w sieci hotelowej, żeby odnaleźć oraz zarezerwować dostępne pokoje według parametrów szukającego.

## Analiza

### Analiza wycinka rzeczywistości

Aplikacja bazodanowa dotyczy sieci hotelowej chcącej wprowadzić nowe funkcjonalności oraz usprawnić działanie systemu. Umożliwienie wprowadzanie funkcji rezerwacji miejsc noclegowych ma na celu przyciągnięcie nowych klientów, optymalizacje stanu obłożenia pokoi oraz maksymalizowanie zakwaterowania. Aplikacja umożliwia rezerwacje gościom pokoi przez internet oraz dostosowanie ofert do swoich potrzeb.

### Scenariusz

Użytkownik chce znaleźć miejsce noclegowe w określonej miejscowości wyznaczonym okresie czasowym. Chce znać hotele z wolnymi pokojami ,które spełniają wymagania w zakresie wyposażenia(np. ilość łóżek w pokoju, parking) mieszczące się w jego budżecie. Użytkownik pragnie zarezerwować dany pokój określonym terminie. Wyliczony zostaje przewidywalny koszt zakwaterowania za dobę, do zapłacenia wybranym sposobem płatności. Podane zostają dane osobiste potrzebne do wykonania transakcji oraz identyfikacji gościa. Sieć hotelowa pozwala zrezygnować z rezerwacji bezkostnie, do terminu dwóch dni przed datą zakwaterowania późniejszym terminie zostaje zastosowany zapis w regulaminie, zgodnie z którym hotel pobiera od gościa opłatę za jedną noc. Pieniądze zostają ściągnięte z karty kredytowej ,która została podana w etapie rezerwacji.

## Wstępne założenia

- Baza posiada wiele hoteli o określonym wyposażeniu
- Baza posiada do kilkuset pokoi o określonym wyposażeniu
- Baza przechowuje informacje osobiste gości
- Potrzebne jest przechowywanie informacji o rezerwacji
- Przechowywane muszą być informacje o płatności
- Status pokoju musi się zmieniać zależności czy jest w zarezerwowany czy wolny
- Obliczanie kosztu pobytu gościa
- Wyszukiwanie wolnych pokoi według parametrów
- Możliwość Modyfikacji danych
- Możliwość zrezygnowania z rezerwacji

# Wymagania

## Wymagania funkcjonalne

| Opis wymagania | Źródło | Niezbędne |
|---|---|---|
| Dodanie pokoju | Osoba uprawniona | Tak |
| Dodanie hotelu | Osoba uprawniona | Tak |
| Dodanie Rezerwacji | Użytkownik | Tak |
| Dodanie danych gościa | Użytkownik | Tak |
| Podgląd danych pokoju | Użytkownik | Tak |
| Podgląd danych hotelu | Użytkownik | Tak |
| Podgląd danych rezerwacji | Użytkownik | Tak |
| Podgląd danych gościa | Użytkownik | Tak |
| Wyszukiwanie listy dostępnych pokoi w danej lokalizacji | Użytkownik | Tak |
| Wyszukiwanie listy hoteli w danej lokalizacji | Użytkownik | Tak |
| Usuwanie pokoju | Osoba uprawniona | Tak |
| Usuwanie hotelu | Osoba uprawniona | Tak |
| Rezygnacja z rezerwacji | Użytkownik | Tak |
| Modyfikacja danych pokoju | Osoba uprawniona | Tak |
| Modyfikacja danych hotelu | Osoba uprawniona | Tak |
| Modyfikacja rezerwacji | Użytkownik | Tak |
| Modyfikacja danych gościa | Użytkownik | Tak |
| Zliczanie kosztu pobytu | Użytkownik | Tak |
| Raport rezerwacji w danym terminie | Użytkownik | Tak |
| Wyszukiwanie wolnych pokoi o odpowiednim wyposażeniu | Użytkownik | Tak |
| Wyszukiwanie hotelu o odpowiednim wyposażeniu | Użytkownik | Tak |
| Wyszukiwanie dostępnych pokoi w danych terminie | Użytkownik | Tak |
| Weryfikacja danych gościa | System | Tak |
| Weryfikacja danych rezerwacji | System | Tak |
| Weryfikacja danych hotelu | System | Tak |
| Weryfikacja danych pokoju | System | Tak |
| Sporządzenie statystyk rezerwacji | Osoba uprawniona | Nie |
| Wyszukiwanie dostępnych pokoi w danej cenie | Użytkownik | Tak |
| Przetwarzanie statusu pokoi | System | Tak |

| | | |
|---|---|---|
| Blokada rezygnacji po terminie | System | Tak |
| Wyszukiwanie gościa | Osoba uprawniona | Tak |
| Wyszukiwanie rezerwacji | Osoba uprawniona | Tak |
| Wyszukiwanie pokoju | Użytkownik | Tak |
| Wyszukiwanie hotelu | Użytkownik | Tak |
| Tworzenie Faktur | Osoba uprawniona | Nie |
| Identyfikacji użytkowników | System | TAK |
| Autoryzacji | System | TAK |
| Tworzenie kopi zapasowej | System | TAK |

### Wymagania niefunkcjonalne

- Relacyjna baza danych
- Interfejs graficzny(strona WWW)
- Posiadanie domeny Internetowej
- Posiadanie Certyfikatu szyfrującego SSL ,protokół HTTPS
- Wykupić hosting lub posiadać Serwer
- Posiadanie bazy danych z środowiskiem implementacyjnym SQL Server
- Aplikacja działa z przeglądarkami Chrome, Microsoft Edge, Opera
- Dostęp do Sieci
- Dane przyjmują zmienną liczbę znaków.
- Kodowanie znaków ze standardem UNICODE.
- Szybki przyrost danych nawet do 5GB dziennie
- Musi zapewniać dostępność ciągłą w systemie 24 godziny dziennie, 7 dni w tygodniu.
- Wykonywać wiele zadań jednocześnie

## Ocena ryzyka

### Kradzież danych

Aplikacja podatna będzie na próby wykradania danych lub ataki hackerskie. Rozwiązaniem jest posiadanie odpowiednich protokołów sieciowych szyfrujących oraz walidacja wprowadzanych danych.

### Koszt utrzymania

Żeby Aplikacja działała w Internecie potrzebne jest wykupienie domeny ,licencja na oprogramowanie Serwerowe, posiadanie nowoczesnej serwerowni mogącą przechowywać sporą ilość danych oraz wyszkoloną kadrę pracowników. Wiąże się to z ogromnymi kosztami. Rozwiązaniem jest wykupienie usług od firm umożliwiających hosting.

### Prze rezerwowanie

Strona będzie podatna na ataki botów mających na celu obciążenie serwera np. przez rezerwowanie terminów z wyprzedzeniem. Rozwiązaniem jest Walidacja danych.

### Usterki Techniczne

Aplikacja mus działać 24 godziny na dobę. Wiele czynników zagraża poprawnemu działaniu, Serwerownia musi być przygotowana na okoliczności jak utrata zasilania. Możliwość rozszerzenia miejsca  na dane. W razie problemów posiadanie kopi zapasowej. Rozwiązaniem jest zrzucenie odpowiedzialności na firmy oferujące powyższe usługi.

### Spoofing ataki typu phishing

Ważne jest uświadamianie klientów o możliwym zagrożeniu podszywania żeby nie przekazywali danych osobistych nieupoważnionym do tym osobom.

### Podgląd danych

Potrzebne jest podjęcie kroków w celu autoryzacji oraz nadania uprawnień. Ważnym aspektem jest ochrona danych gości oraz stosowanie się do zasad RODO.

### Obciążenie aplikacji

Trzeba wziąć pod uwagę obciążenie strony, aplikacji, bazy dużą ilością zadań. Dostęp do aplikacji jest powszechny. Wiele osób będzie korzystało jednocześnie. Rozwiązanie jest optymalizacja kodu , dobór odpowiednich podzespołów sprzętu, zastosowanie dodatkowych funkcji w celu przetwarzania danych.

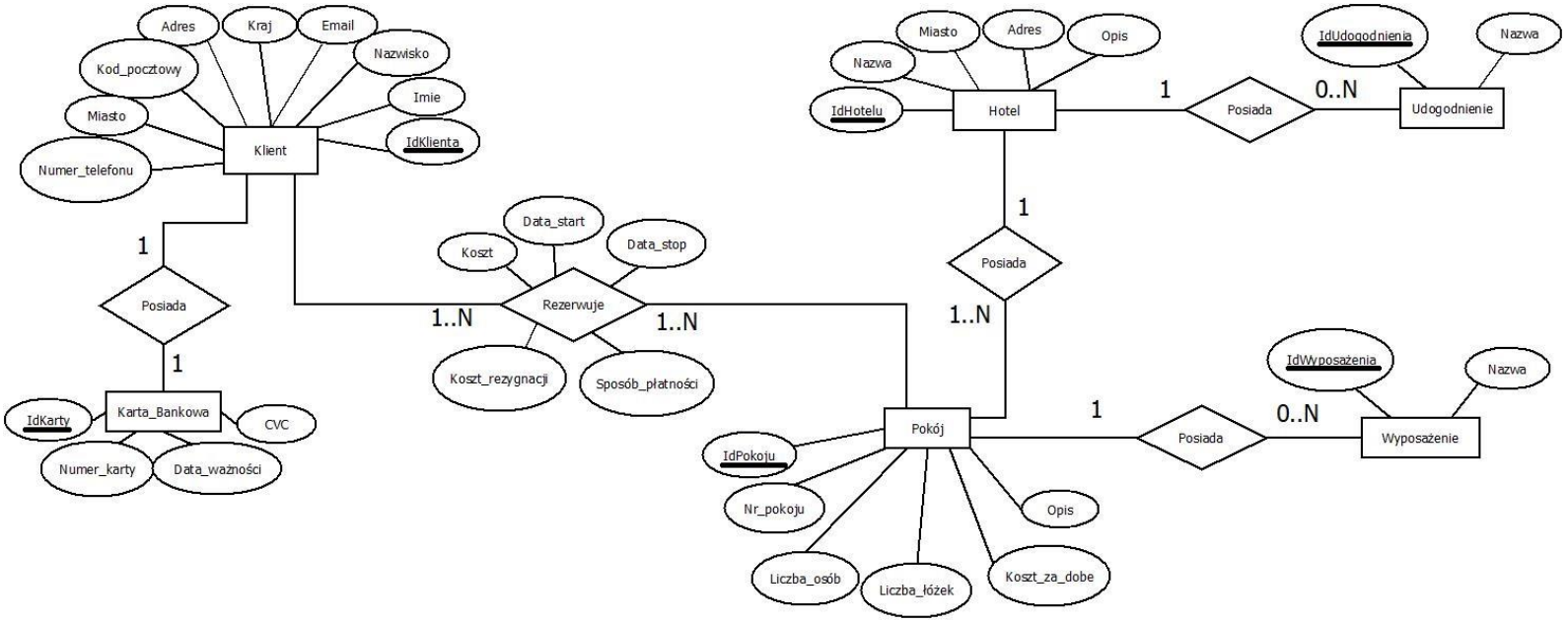
### Złożoność Projektu

Projekt jest dość złożony. Musi ze sobą jednocześnie współpracować wiele elementów. Błąd w jednym etapie projektu może doprowadzić do niemożliwości wykonanie zadania. Potrzebne są profesjonalne rozwiązania oraz opracowany plan wykonania. Ważne jest zdefiniowany pierwszorzędnych podzespołów, jak przygotowanie projektu bazy.

## Korzyści wdrożenia

- Maksymalizacja zakwaterowania
- Optymalizacja stanu obłożenia pokoi
- Przyśpieszenie etapu zameldowania
- Możliwość optymalizacji oferty
- Gwarancja zakwaterowania
- Automatyzacja harmonogramu rezerwacji
- Odciążenie pracowników z dodatkowych obowiązków
- Przejrzystość danych
- Przechowywanie danych
- Gwarancja pewności
- Analiza informacji oraz możliwość dostosowania rozwiązań

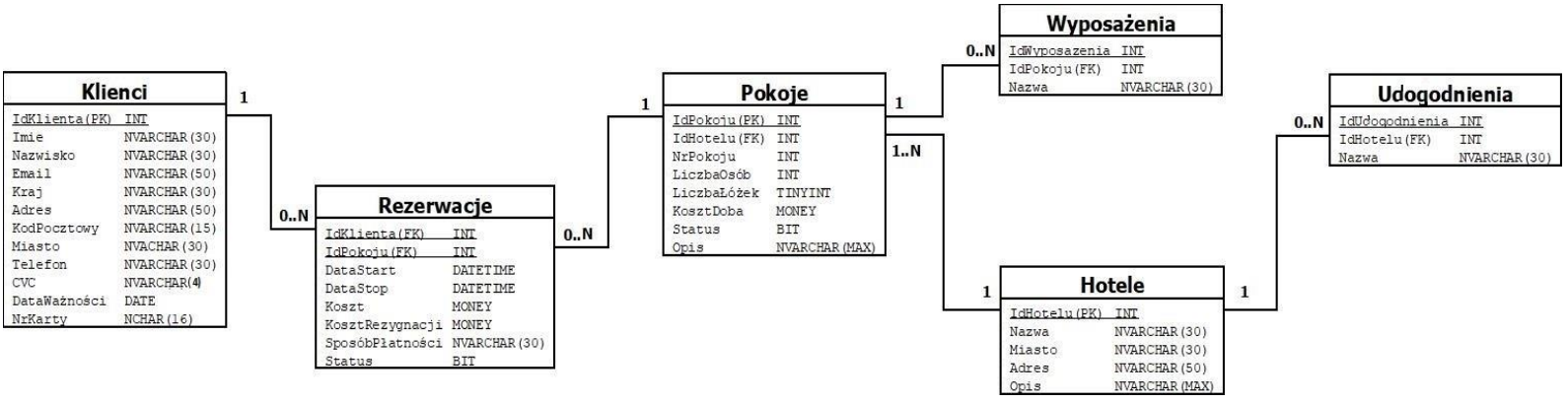
# 2. Definicja DZE



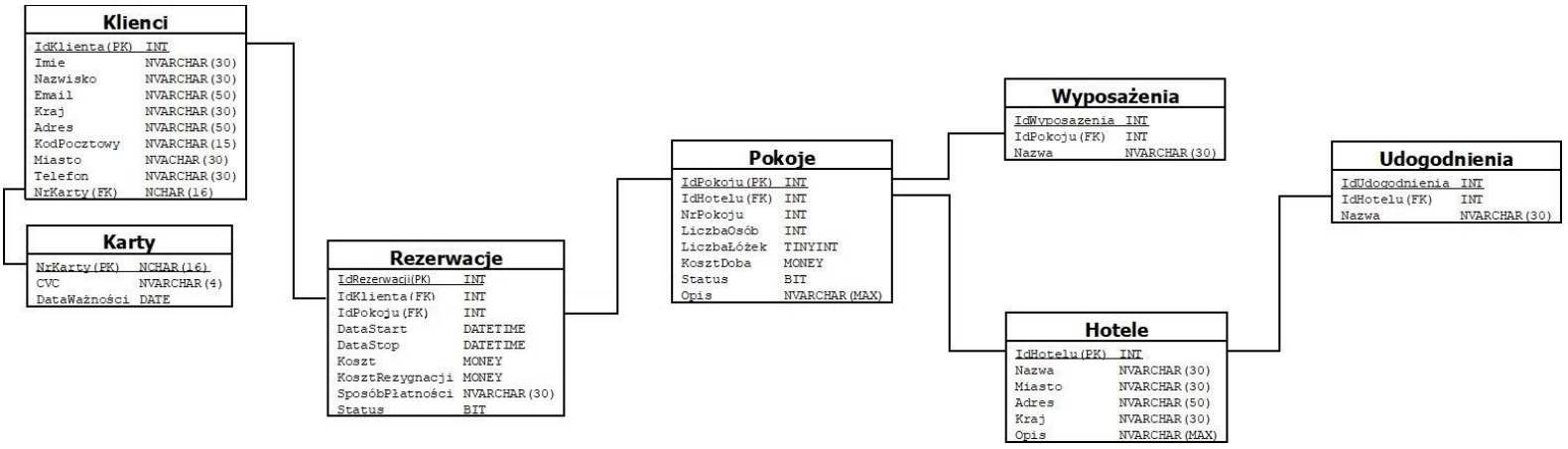

## Poprawka

Została dodana encja Wyposażenie  powiązana z Pokój. Dla Hotel powiązana została encja Udogodnienie obie z atrybutami  Nazwa oraz Id.

# 3. Transformacja DZE do modelu relacyjnego



# 4. Proces normalizacji schematów relacyjnych




Została dodana tabela wyposażenia powiązana z tabelą pokoje. Uznałem że pokoje mogą mieć indywidualne wyposażenia np. dla niepełnosprawnych lub pokoje ekskluzywne. Nie mogłem tego umieścić w opisie ani nowej kolumnie wyposażenia ponieważ wydłużyłoby oraz utrudniłoby to filtracje stworzyłoby atrybuty wielowartościowe co byłoby niepoprawne z pierwszą postacią normalną .Dodanie kolumny w tabeli pokoje z ograniczeniem do jednej wyposażenia na wiersz stworzyłoby nadmiar niepotrzebnych danych. Utrudniło by także aktualizacje danych pokoju. W relacji Klienci trzeba było przenieść CVC,DataWażności do osobnej relacji Karty według drugiej oraz trzeciej postaci normalnej.

Powyższy schemat relacji jest znormalizowany uzasadnia to poniższe reguły.

**Pierwsza postać normalna**
Atrybuty posiadają tylko wartości atomowe czyli informacje elementarne .Każde pole tabeli przechowuje jedną informację. Nie zawiera kolekcji, listy.

**Druga postać normalna**
Relacja jest w drugiej postaci normalnej wtedy i tylko wtedy, gdy jest w I postaci normalnej i żadna kolumna niekluczowa nie jest częściowo funkcyjnie zależna od jakiegokolwiek klucza potencjalnego.

**Trzecia postać normalna**
Relacja jest w trzeciej postaci normalnej wtedy i tylko wtedy, gdy jest w II postaci normalnej i żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.

## Poprawka

Tabela rezerwacje posiada IdRezerwacji jako klucz główny ,dana kolumna jest potrzebna w dalszych etapach projektowania bazy danych w celu zachowania integralności.

# 5. Definicja zasad poprawności danych

Musimy wziąć pod uwagę że Klienci będą z całego świata. Dlatego kod pocztowy nie może posiadać wzoru.
Dla prawidłowego funkcjonowania bazy danych należy zastosować poniższe zasady poprawności danych dla tabel:

- Klienci
  - Telefon: musi być wartością cyfrową z prefiksem w nawiasie.
  -NrKarty: musi posiadać wartości unikatowe. Każda wartość musi posiadać 16 cyfr.
  -Email: musi być emailem(posiadać @).
- Karty
  -NrKarty: musi posiadać wartości unikatowe. Każda wartość musi posiadać 16 cyfr.
  -CVC: Musi posiadać 3 lub 4 cyfry.
  -DataWażności: nie może być starsza niż data dodania.
- Rezerwacje
  -DataStart: nie może zawierać daty starszej niż data dodania rezerwacji.
  -DataStop: nie może zawierać daty starszej niż DataStart.
  -Status: musi posiadać domyślną wartość 0 która będzie oznaczała że rezerwacja nie jest anulowana. Wartość 1 będzie oznaczało że klient zrezygnował z rezerwacji.
  -KosztRezygnacji: Nie może być wartością ujemną. Według umowy hotelu musi wynosić cene kosztu za dobe pokoju.
  -Koszt: nie może być wartością ujemną.
  -SposóbPłatności: może mieć wartość ino (‘przelew’,’na miejscu’)
- Pokoje
  -Status: musi posiadać domyślną wartość 0 która będzie oznaczała że pokój jest wolny. Wartość 1 będzie oznaczało że pokój jest zajęty.
  -NrPokoju, LiczbaOsób, LiczbaŁóżek, KosztDoba: nie mogą mieć wartości ujemnych.

Wszystkie kolumny zawierające Id muszą być wartościami unikatowymi. Żadna kolumna oprócz opis nie może być null.

# 6. Definicja schematu bazy danych (SQL), utworzenie bazy danych.

```
USE master
--TWORZENIE BAZY DANYCH
GO
IF EXISTS(
	SELECT name FROM master.dbo.sysdatabases
	WHERE name = N'BazaHotel'
	)
BEGIN
	DROP DATABASE BazaHotel
END
GO
CREATE DATABASE BazaHotel
GO
USE BazaHotel
--TWORZENIE TABEL
GO
DROP TABLE IF EXISTS dbo.Karty
GO
CREATE TABLE Karty(
[NrKarty] NCHAR(16) NOT NULL PRIMARY KEY,
[CVC] NVARCHAR(4) NOT NULL,
[DataWażności] DATE NOT NULL
)
GO
DROP TABLE IF EXISTS dbo.Klienci
GO
CREATE TABLE Klienci(
[IdKlienta] INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
[Imie] NVARCHAR(30) NOT NULL,
[Nazwisko] NVARCHAR(30) NOT NULL,
[Email] NVARCHAR(50) NOT NULL,
[Kraj] NVARCHAR(30) NOT NULL,
[Adres] NVARCHAR(50) NOT NULL,
[KodPocztowy] NVARCHAR(15) NOT NULL,
[Miasto] NVARCHAR(30) NOT NULL,
[Telefon] NVARCHAR(30) NOT NULL,
[NrKarty] NCHAR(16) NOT NULL
)
GO
DROP TABLE IF EXISTS dbo.Rezerwacje
GO
CREATE TABLE Rezerwacje(
[IdRezerwacji] INT IDENTITY(1,1) NOT NULL PRIMARY KEY, --poprawka
[IdKlienta] INT NOT NULL,
[IdPokoju] INT NOT NULL,
[DataStart] DATETIME NOT NULL,
[DataStop] DATETIME NOT NULL,
[Koszt] MONEY NOT NULL,
[KosztRezygnacji] MONEY NOT NULL DEFAULT(0),--poprawka
[SposóbPłatności] NVARCHAR(30) NOT NULL,
[Status] BIT NOT NULL DEFAULT(0)
)
GO
DROP TABLE IF EXISTS dbo.Pokoje
GO
CREATE TABLE Pokoje(
[IdPokoju] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
```

```
[IdHotelu] INT NOT NULL,
[NrPokoju] INT NOT NULL,
[LiczbaOsób] INT NOT NULL,
[Liczbałóżek] TINYINT NOT NULL,
[KosztDoba] MONEY NOT NULL,
[Status] BIT NOT NULL DEFAULT(0),
[Opis] NVARCHAR(MAX) NULL
)
GO
DROP TABLE IF EXISTS dbo.Wyposażenia
GO
CREATE TABLE Wyposażenia
(
[IdWyposażenia] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
[IdPokoju] INT NOT NULL,
[Nazwa] NVARCHAR(30) NOT NULL
)
GO
DROP TABLE IF EXISTS dbo.Hotele
GO
CREATE TABLE Hotele(
[IdHotelu] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
[Nazwa] NVARCHAR(30) NOT NULL,
[Miasto] NVARCHAR(30) NOT NULL,
[Adres] NVARCHAR(50) NOT NULL,
[Kraj] NVARCHAR(30) NOT NULL,
[Opis] NVARCHAR(MAX) NULL
)
GO
DROP TABLE IF EXISTS dbo.Udogodnienia
GO
CREATE TABLE Udogodnienia(
[IdUdogodnienia]  INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
[IdHotelu] INT NOT NULL,
[Nazwa] NVARCHAR(30) NOT NULL
)
GO
USE BazaHotel
--usuniecie POWIĄZAŃ jesli istnieja
GO
IF EXISTS (
       SELECT *
       FROM sys.foreign_keys
       WHERE object_id = OBJECT_ID('dbo.FK_Klienci_Karty')
       AND parent_object_id=OBJECT_ID('dbo.Klienci')
)

ALTER TABLE dbo.Klienci
DROP CONSTRAINT FK_Klienci_Karty
GO
IF EXISTS (
       SELECT *
       FROM sys.foreign_keys
       WHERE object_id = OBJECT_ID('dbo.FK_Udogodnienia_Hotele')
       AND parent_object_id=OBJECT_ID('dbo.Udogodnienia')
)
ALTER TABLE Udogodnienia
DROP CONSTRAINT FK_Udogodnienia_Hotele
GO
IF EXISTS (
       SELECT *
       FROM sys.foreign_keys
```

```
	WHERE object_id = OBJECT_ID('dbo.FK_Pokoje_Hotele')
	AND parent_object_id=OBJECT_ID('dbo.Pokoje')
)
ALTER TABLE dbo.Pokoje
DROP CONSTRAINT FK_Pokoje_Hotele
GO
IF EXISTS (
	SELECT *
	FROM sys.foreign_keys
	WHERE object_id = OBJECT_ID('dbo.FK_Wyposażenia_Pokoje')
	AND parent_object_id=OBJECT_ID('dbo.Wyposażenia')
)


ALTER TABLE dbo.Wyposażenia
DROP CONSTRAINT FK_Wyposażenia_Pokoje
GO
IF EXISTS (
	SELECT *
	FROM sys.foreign_keys
	WHERE object_id = OBJECT_ID('dbo.FK_Rezerwacje_Pokoje')
	AND parent_object_id=OBJECT_ID('dbo.Rezerwacje')
)

ALTER TABLE dbo.Rezerwacje
DROP CONSTRAINT FK_Rezerwacje_Pokoje
GO
IF EXISTS (
	SELECT *
	FROM sys.foreign_keys
	WHERE object_id = OBJECT_ID('dbo.FK_Rezerwacje_Klienci')
	AND parent_object_id=OBJECT_ID('dbo.Rezerwacje')
)


ALTER TABLE dbo.Rezerwacje
DROP CONSTRAINT FK_Rezerwacje_Klienci
GO
--DODANIE POWIĄZAŃ
ALTER TABLE Klienci
ADD CONSTRAINT FK_Klienci_Karty
	FOREIGN KEY(NrKarty) REFERENCES Karty(NrKarty)
GO

ALTER TABLE Rezerwacje
ADD CONSTRAINT FK_Rezerwacje_Klienci
	FOREIGN KEY (IdKlienta) REFERENCES Klienci(IdKlienta)
GO

ALTER TABLE Rezerwacje
ADD CONSTRAINT FK_Rezerwacje_Pokoje
	FOREIGN KEY (IdPokoju) REFERENCES Pokoje(IdPokoju)
GO

ALTER TABLE Wyposażenia
ADD CONSTRAINT FK_Wyposażenia_Pokoje
FOREIGN KEY (IdPokoju) REFERENCES Pokoje(IdPokoju)
GO

ALTER TABLE Pokoje
ADD CONSTRAINT FK_Pokoje_Hotele
FOREIGN KEY (IdHotelu) REFERENCES Hotele(IdHotelu)
```

```
GO

ALTER TABLE Udogodnienia
ADD CONSTRAINT FK_Udogodnienia_Hotele
FOREIGN KEY (IdHotelu) REFERENCES Hotele(IdHotelu)
GO
```

# 7. Definicja niedeklaratywnych mechanizmów sprawdzania poprawności danych

Trzeba utworzyć skrypty które będą miały za zadanie:

W tabeli Klienci
-Sprawdza czy email posiada @.

W tabeli Karty
-Sprawdza czy DataWażności nie jest starsza niż data dodania.

W tabeli Rezerwacje
-Sprawdza czy DataStart nie jest starsza niż data dodania.
-Sprawdza czy DataStop nie jest starsza niż DataStart.
-Sprawdza czy KosztRezygnacji, Koszt są większe lub równe 0
-Sprawdza czy SposóbPłatności zawiera wartość ‘przelew’ lub ’na miejscu’.

W tabeli Pokoje
-Sprawdza czy NrPokoju, LiczbaOsób, LiczbaŁóżek, KosztDoba są większe lub równe 0.

Można dany efekty uzyskać poprzez dodanie wyzwalaczy na tabele przy Update oraz Insert, jak i procedur oraz dodaniu constraint(to jest mechanizm deklaratywny).

# 8. Implementacja niedeklaratywnych mechanizmów sprawdzania poprawności danych.

```
use BazaHotel
--sprawdzanie czy email jest poprawny
IF OBJECt_ID('Klienci','U') IS NOT NULL
begin
	ALTER TABLE Klienci DROP CONSTRAINT IF EXISTS SprawdzEmail;
	ALTER TABLE Klienci
	ADD CONSTRAINT SprawdzEmail CHECK (Email like '%@%');
end
go
--sprawdzanie czy koszt nie jest wartością ujemną
IF OBJECt_ID('Rezerwacje','U') IS NOT NULL
begin
	ALTER TABLE Rezerwacje DROP CONSTRAINT IF EXISTS SprawdzKoszt;
	ALTER TABLE Rezerwacje
	ADD CONSTRAINT SprawdzKoszt CHECK (Koszt >=0);
end
go
--sprawdzanie czy koszt rezygnacji nie jest wartością ujemną
IF OBJECt_ID('Rezerwacje','U') IS NOT NULL
begin
	ALTER TABLE Rezerwacje DROP CONSTRAINT IF EXISTS SprawdzKosztRezygnacji;
	ALTER TABLE Rezerwacje
	ADD CONSTRAINT SprawdzKosztRezygnacji CHECK (KosztRezygnacji >=0);
end
go
--sprawdzenie czy SposóbPłatności ma poprawną wartość
IF OBJECt_ID('Rezerwacje','U') IS NOT NULL
begin
	ALTER TABLE Rezerwacje DROP CONSTRAINT IF EXISTS SprawdzSposóbPłatności;
	ALTER TABLE Rezerwacje
	ADD CONSTRAINT SprawdzSposóbPłatności CHECK (SposóbPłatności in ('przelew','na
miejscu'));
end
go
--sprawdzanie czy nr pokoju nie wartością ujemną lub zerem
IF OBJECt_ID('Pokoje','U') IS NOT NULL
begin
	ALTER TABLE Pokoje DROP CONSTRAINT IF EXISTS SprawdzNrPokoju;
	ALTER TABLE Pokoje
	ADD CONSTRAINT SprawdzNrPokoju CHECK (NrPokoju >0);
end
go
--sprawdzanie czy liczba osób nie wartością ujemną
IF OBJECt_ID('Pokoje','U') IS NOT NULL
begin
	ALTER TABLE Pokoje DROP CONSTRAINT IF EXISTS SprawdzLiczbaOsób;
	ALTER TABLE Pokoje
	ADD CONSTRAINT SprawdzLiczbaOsób CHECK (LiczbaOsób >= 0);
end
go
--sprawdzanie czy liczba łóżek nie wartością ujemną
IF OBJECt_ID('Pokoje','U') IS NOT NULL
begin
	ALTER TABLE Pokoje DROP CONSTRAINT IF EXISTS SprawdzLiczbałóżek;
	ALTER TABLE Pokoje
	ADD CONSTRAINT SprawdzLiczbałóżek CHECK (Liczbałóżek >= 0);
end
```

```
go
--sprawdzanie czy koszt za dobe nie wartością ujemną
IF OBJECt_ID('Pokoje','U') IS NOT NULL
begin
	ALTER TABLE Pokoje DROP CONSTRAINT IF EXISTS SprawdzKosztDoba;
	ALTER TABLE Pokoje
	ADD CONSTRAINT SprawdzKosztDoba CHECK (KosztDoba >= 0);
end
go
--TWORZENIE WYZWALACZA KTORY SPRAWDZA CZY DATA WAŻNOŚCI KARTY NIE JEST STARSZA NIŻ
DATA DODANIA przy update
DROP TRIGGER IF EXISTS SprawdzKartaDataWażnościUpdate;
go
CREATE TRIGGER SprawdzKartaDataWażnościUpdate ON dbo.Karty
FOR UPDATE
AS
BEGIN
	IF UPDATE(DataWażności)
	BEGIN
	DECLARE @DataDodania DATE;
	SET @DataDodania = GETDATE();
	IF((SELECT DataWażności FROM INSERTED)<@DataDodania)
		BEGIN
		PRINT 'PRZETERMINOWANA DATA'
		UPDATE Karty
		SET DataWażności = (SELECT DataWażności FROM deleted)
		WHERE NrKarty = (SELECT NrKarty FROM inserted)
		END
	END
END
go
--TWORZENIE WYZWALACZA KTORY SPRAWDZA CZY DATA WAŻNOŚCI KARTY NIE JEST STARSZA NIŻ
DATA DODANIA przy insert
DROP TRIGGER IF EXISTS dbo.SprawdzKartaDataWażnościInsert;
go
CREATE TRIGGER SprawdzKartaDataWażnościInsert ON dbo.Karty
instead of insert
AS
	begin
	DECLARE @DataDodania DATE;
	SET @DataDodania = GETDATE();
	IF((SELECT DataWażności FROM INSERTED)>@DataDodania)
		BEGIN
		insert into Karty
		values((select NrKarty from inserted),(select CVC from inserted),(select
DataWażności from inserted));
		END
	end
go
--sprawdza czy data start jest pozniejsza niz data dodania oraz czy data stop jest
pozniejsza od data start przy insert
--oblicza koszt rezerwacji
drop trigger if exists dbo.SprawdzDataStartStopInsert;
go
CREATE TRIGGER SprawdzDataStartStopInsert ON dbo.Rezerwacje
INSTEAD OF INSERT
AS
BEGIN
	DECLARE @DataDodania DATE;
	SET @DataDodania = GETDATE();
	IF(((SELECT DataStart FROM INSERTED)>@DataDodania )AND ((SELECT DataStop FROM
inserted)  > (SELECT DataStart FROM inserted)        ) )
```

```
        BEGIN
        INSERT INTO Rezerwacje
        VALUES((SELECT IdKlienta FROM inserted),(SELECT IdPokoju FROM inserted),
        (SELECT DataStart FROM inserted),(SELECT DataStop FROM inserted)
        ,(SELECT (KosztDoba*(DATEDIFF(day,DataStart,DataStop))) FROM inserted i join
Pokoje p on i.IdPokoju = p.IdPokoju
        ),0,
        (SELECT SposóbPłatności FROM inserted),
        0);
        END
END
Go
```

Do metod niedeklaratywnych mechanizmów sprawdzania poprawności danych zaliczamy: wyzwalacze (Triggers), procedury składowane (Stored Procedure).

# 9. Implementacja kodu wspomagającego aplikację użytkową.

```
use BazaHotel
go

drop procedure if exists p_szukaj_wolnego_pokoju;
go
--procedura znajdujaca nie zarezerwowane pokoje w danym miescie,
--w  danym przedziale czasu  o danej liczbie osob i łóżek.
--Posegreguje po cenie rosnąco
create procedure p_szukaj_wolnego_pokoju
@data_od as Datetime,
@data_do as Datetime,
@Miasto as nvarchar(30),
@Liczba_Osob as int,
@liczba_Lozek as tinyint
as
begin
select Distinct w.Hotel,w.[Nr Pokoju],w.[koszt za dobe]
from (select r.IdPokoju,p.NrPokoju[Nr Pokoju],h.Nazwa[Hotel],p.KosztDoba[koszt za
dobe]
from Rezerwacje r
join Pokoje p on r.IdPokoju = p.IdPokoju
join Hotele h on p.IdHotelu = h.IdHotelu
where p.Liczbałóżek = @liczba_Lozek and p.LiczbaOsób =@Liczba_Osob and h.Miasto =
@Miasto
and p.Status=0
) as w
left join
(
select r.IdPokoju from Rezerwacje r join Pokoje p on r.IdPokoju = p.IdPokoju
join Hotele h on p.IdHotelu = h.IdHotelu
where  (DataStop>=@data_od) and (DataStart <= @data_do) and  (r.Status = 0)
and (h.Miasto = @Miasto) and (p.Liczbałóżek = @liczba_Lozek) and
(p.LiczbaOsób =@Liczba_Osob)
)as wn
on w.IdPokoju = wn.IdPokoju
where wn.IdPokoju is  null
order by w.[koszt za dobe]
end
go
--procedura wyswietlajaca wyposazenie pokoju
drop procedure if exists p_pokaz_wyposazenie_pokoju;
```

```
go
create procedure p_pokaz_wyposazenie_pokoju
@hotel as nvarchar(30),
@nr_pokoju as int
as
begin
select w.Nazwa[Wyposazenie pokoju],h.Nazwa[hotel]
from Pokoje p join Hotele h on h.IdHotelu = p.IdHotelu
join Wyposażenia w on w.IdPokoju = p.IdPokoju
where h.Nazwa = @hotel and NrPokoju = @nr_pokoju
end
-- procedura wyswietlajaca wyposazenie hotelu
go
drop procedure if exists p_pokaz_wyposazenie_hotelu;
go
create procedure p_pokaz_wyposazenie_hotelu
@hotel as nvarchar(30)
as
begin
select u.Nazwa[Udogodnienia hotelu]
from Hotele h join Udogodnienia u on h.IdHotelu = u.IdHotelu
where h.Nazwa = @hotel;
end
go
--wyzwalacz który sprawdza czy nie zrezygnowana za późno z rezerwacji,
--w razie rezygnacji od 2 dni przed terminem naliczane są koszta 1 doby pokoju
drop trigger if exists Rezygnacja;
go
create trigger Rezygnacja
on Rezerwacje
for update
as
begin
	if UPDATE(Status)
	begin
		DECLARE @DATA AS DATE= GETDATE();
		IF  DATEDIFF(DAY,@DATA,(SELECT DataStart FROM INSERTED))<=2
		begin
		update Rezerwacje
		set KosztRezygnacji =
		(select p.KosztDoba
		from inserted i join Pokoje p on p.IdPokoju = i.IdPokoju)
		where IdRezerwacji in (select IdRezerwacji from inserted)
		end
	end
end
go
--tworzy widok dla histori odbytych rezerwacji z szczegółowymi danymi
drop view if exists v_Rezerwacje;
go
create view v_Rezerwacje as
select r.IdRezerwacji,h.Nazwa[Nazwa hotelu],
NrPokoju,DataStart,DataStop,r.Koszt,Imie,Nazwisko,Telefon
from Rezerwacje r join Klienci k
on k.IdKlienta = r.IdKlienta
join Pokoje p on r.IdPokoju = p.IdPokoju
join Hotele h on h.IdHotelu = p.IdHotelu
where r.Status = 0;
go
--tworzy widok histori rezerwacji gdzie osoby zrezygnowały z rezerwacji
drop view if exists v_RezerwacjeZrezygnowane
go
```

```
create view v_RezerwacjeZrezygnowane as
select r.IdRezerwacji,h.Nazwa[Nazwa hotelu],NrPokoju,DataStart[Data Start],
DataStop[Data Stop],
r.KosztRezygnacji[Koszt Rezygnacji],Imie,Nazwisko,Telefon
from Rezerwacje r join Klienci k
on k.IdKlienta = r.IdKlienta
join Pokoje p on r.IdPokoju = p.IdPokoju
join Hotele h on h.IdHotelu = p.IdHotelu
where r.Status = 1
go
--procedura pomagająca znaleść liste hoteli z danymi udogodnieniami
drop procedure if exists p_SearchHotel;
go
create procedure p_SearchHotel
@Miejscowosc as nvarchar(30),
@Udogodnienie1 as nvarchar(30)=null,
@Udogodnienie2 as nvarchar(30) =null,
@Udogodnienie3 as nvarchar(30)=null,
@Udogodnienie4 as nvarchar(30)=null,
@Udogodnienie5 as nvarchar(30)=null,
@Udogodnienie6 as nvarchar(30)=null,
@Udogodnienie7 as nvarchar(30)=null,
@Udogodnienie8 as nvarchar(30)=null
as
begin
select h.Nazwa[nazwa hotelu],h.Miasto,
count(u.Nazwa)[Liczba posiadających szukanych udogodnień]
from Hotele h join Udogodnienia u
on h.IdHotelu = u.IdHotelu
where h.Miasto = @Miejscowosc  and
u.Nazwa in
(@Udogodnienie1,@Udogodnienie2,@Udogodnienie3,@Udogodnienie4,@Udogodnienie5,
@Udogodnienie6,@Udogodnienie7,@Udogodnienie8)
group by h.Nazwa,h.Miasto
end
go
--tworzenie funkcji zwracajace dane dla faktur
drop function if exists f_faktura
go
create function f_faktura(@idRezerwacji int)
returns table
as
       return(select k.Imie,k.Nazwisko,k.Adres,k.Miasto,k.KodPocztowy,
       k.Kraj,k.NrKarty,r.Koszt,r.SposóbPłatności,
       r.DataStart[data zameldowania],h.Nazwa[hotel],
       h.Adres[adres hotelu],h.Miasto[miasto hotelu]
       from Rezerwacje r join Klienci k
       on r.IdKlienta = k.IdKlienta
       join Pokoje p on p.IdPokoju = r.IdPokoju
       join Hotele h on h.IdHotelu = p.IdHotelu
       where IdRezerwacji = @idRezerwacji
       )
go
drop procedure if exists p_szukaj_wolnego_pokoju_w_hotelu;
go
--procedura znajdujaca nie zarezerwowane pokoje w danym hotelu
--,w  danym przedziale czasu  o danej liczbie osob i łóżek.
--Posegreguje po cenie rosnąco
create procedure p_szukaj_wolnego_pokoju_w_hotelu
@data_od as Datetime,
@data_do as Datetime,
@hotel as nvarchar(30),
```

```
@Liczba_Osob as int,
@liczba_Lozek as tinyint
as
begin
select Distinct w.Hotel,w.[Nr Pokoju],w.[koszt za dobe]
from (select r.IdPokoju,p.NrPokoju[Nr Pokoju],h.Nazwa[Hotel],p.KosztDoba[koszt za
dobe]
from Rezerwacje r
join Pokoje p on r.IdPokoju = p.IdPokoju join Hotele h on p.IdHotelu = h.IdHotelu
where p.Liczbałóżek = @liczba_Lozek and p.LiczbaOsób =@Liczba_Osob
and h.Nazwa = @hotel and p.Status=0
) as w
left join
(
select r.IdPokoju from Rezerwacje r join Pokoje p on r.IdPokoju = p.IdPokoju
join Hotele h on p.IdHotelu = h.IdHotelu
where  (DataStop>=@data_od) and (DataStart <= @data_do) and  (r.Status = 0)
and (h.Nazwa = @hotel) and (p.Liczbałóżek = @liczba_Lozek)
and (p.LiczbaOsób =@Liczba_Osob)
)as wn
on w.IdPokoju = wn.IdPokoju
where wn.IdPokoju is  null
order by w.[koszt za dobe]
end
go
drop procedure if exists p_szukaj_wolnego_pokoju_w_hotelu_z_wyposazeniem;
go
--procedura znajdujaca nie zarezerwowane pokoje w danym hotelu,
--w  danym przedziale czasu  o danej liczbie osob i łóżek oraz wyposazeniu.
--Posegreguje po cenie rosnąco
create procedure p_szukaj_wolnego_pokoju_w_hotelu_z_wyposazeniem
@data_od as Datetime,
@data_do as Datetime,
@hotel as nvarchar(30),
@Liczba_Osob as int,
@liczba_Lozek as tinyint,
@wyposazenie1 as nvarchar(30)
as
begin
select Distinct w.Hotel,w.[Nr Pokoju],w.[koszt za dobe]
from (select r.IdPokoju,p.NrPokoju[Nr Pokoju],h.Nazwa[Hotel],p.KosztDoba[koszt za
dobe]
from Rezerwacje r
join Pokoje p on r.IdPokoju = p.IdPokoju
join Hotele h on p.IdHotelu = h.IdHotelu
join Wyposażenia w on p.IdPokoju = w.IdPokoju
where p.Liczbałóżek = @liczba_Lozek and p.LiczbaOsób =@Liczba_Osob and h.Nazwa =
@hotel
and p.Status=0 and w.Nazwa in(@wyposazenie1)
) as w
left join
(
select r.IdPokoju from Rezerwacje r join Pokoje p on r.IdPokoju = p.IdPokoju
join Hotele h on p.IdHotelu = h.IdHotelu
join Wyposażenia w on p.IdPokoju = w.IdPokoju
where  (DataStop>=@data_od ) and (DataStart <= @data_do ) and  (r.Status = 0)
and (h.Nazwa = @hotel) and (p.Liczbałóżek = @liczba_Lozek) and (p.LiczbaOsób
=@Liczba_Osob)
 and (w.Nazwa in(@wyposazenie1))
)as wn
on w.IdPokoju = wn.IdPokoju
where wn.IdPokoju is  null
```

```
group by w.Hotel,w.[Nr Pokoju],w.[koszt za dobe]
order by w.[koszt za dobe]
end
go
drop procedure if exists p_szukaj_wolnego_pokoju_w_miescie_z_wyposazeniem;
go
--procedura znajdujaca nie zarezerwowane pokoje w danym miescie,
--w  danym przedziale czasu  o danej liczbie osob i łóżek oraz wyposazeniu.
--Posegreguje po cenie rosnąco
create procedure p_szukaj_wolnego_pokoju_w_miescie_z_wyposazeniem
@data_od as Datetime,
@data_do as Datetime,
@miasto as nvarchar(30),
@Liczba_Osob as int,
@liczba_Lozek as tinyint,
@wyposazenie1 as nvarchar(30)
as
begin
select Distinct w.Hotel,w.[Nr Pokoju],w.[koszt za dobe]
from (select r.IdPokoju,p.NrPokoju[Nr Pokoju],h.Nazwa[Hotel],p.KosztDoba[koszt za
dobe]
from Rezerwacje r
join Pokoje p on r.IdPokoju = p.IdPokoju
join Hotele h on p.IdHotelu = h.IdHotelu
join Wyposażenia w on p.IdPokoju = w.IdPokoju
where p.Liczbałóżek = @liczba_Lozek and p.LiczbaOsób =@Liczba_Osob and h.Miasto =
@miasto
and p.Status=0 and w.Nazwa in(@wyposazenie1)
) as w
left join
(
select r.IdPokoju from Rezerwacje r join Pokoje p on r.IdPokoju = p.IdPokoju
join Hotele h on p.IdHotelu = h.IdHotelu
join Wyposażenia w on p.IdPokoju = w.IdPokoju
where  (DataStop>=@data_od ) and (DataStart <= @data_do ) and  (r.Status = 0)
and (h.Miasto = @miasto) and (p.Liczbałóżek = @liczba_Lozek) and (p.LiczbaOsób
=@Liczba_Osob)
 and (w.Nazwa in(@wyposazenie1))
)as wn
on w.IdPokoju = wn.IdPokoju
where wn.IdPokoju is  null
group by w.Hotel,w.[Nr Pokoju],w.[koszt za dobe]
order by w.[koszt za dobe]
end
go
--procedura dodajaca nowego klienta do bazy danych
drop procedure if exists p_dodaj_klienta;
go
create procedure p_dodaj_klienta
@Imie as nvarchar(30),
@Nazwisko as nvarchar(30),
@Email as nvarchar(50),
@Kraj as nvarchar(30),
@Adres as nvarchar(50),
@KodPocztowy as nvarchar(15),
@Miasto as nvarchar(30),
@Telefon as nvarchar(30),
@NrKart as nchar(16),
@CVC as nvarchar(4),
@DataWażnosci as date
as
begin
```

```
insert into Karty
values(@NrKart,@CVC,@DataWażnosci);
insert into Klienci
values(@Imie,@Nazwisko,@Email,@Kraj,@Adres,@KodPocztowy,@Miasto,@Telefon,@NrKart);
end
go
```

Dane skrypty pomagają użytkownikowi korzystającego z bazy danych.
Gotowy zbiór procedur z najczęściej używanymi zapytaniami oraz przyśpieszające wprowadzanie danych.
Specjalnie przygotowane widoki w celu bezpieczeństwa jak i przejrzystości danych.
Funkcje pozwalające na ponowne użycie przygotowanego kodu.
Wyzwalacze automatyzujące większość danych w celu braku nieprawidłowości.

# 10. Testowanie bazy danych

```
use BazaHotel
go
--wprowadzanie danych do tabeli Hotele
insert into Hotele(Nazwa,Miasto,Adres,Kraj,Opis)
values('Dębnica','Kraków','ul. Kościuszki 24','Polska','Wypocznij w cichej okolicy i zrelaksuj się korzystając z Hotelu Debnica.'),
('Pod Jeleniem','Kraków','ul. Władysława 14','Polska','Uroczy Hotel z jeszcze bardziej uroczą okolicą.'),
('Nad Jeziorem','Kraków','ul. Młyńska 24','Polska','Obiecujemy nie zapomniane przeżycia.'),
('Blisko','Gdańsk','ul. Czerwonych maków 23','Polska','Zawsze jesteśmy Blisko!')
go
--wprowadzanie danych do tabeli Karty
insert into Karty
values('1234567890123456','12IO','2022-05-05');
insert into Karty
values('0987654321123456','3E1','2021-12-12');
insert into Karty
values('1087654321123456','3A1','2023-12-12');
go
--wprowadzanie danych Klientów
insert into Klienci
values('Adrian','Karpicki','adam@gmail.com','Polska','ul. Magórka 22 ','22-321','Kraków','123456789','1234567890123456'),
('Marek','Kowalski','maro@o2.pl','Polska','ul. Kwiatowa 21 ','20-321','Zakopane','213456119','0987654321123456'),
('Alex','White','whiteALex@gmail.com','USA','882 Terminal St','CA 92123','San Diego','212256119','1087654321123456')
go
--wprowadzanie danych do tabeli Pokoje
insert into Pokoje(IdHotelu,NrPokoju,LiczbaOsób,Liczbałóżek,KosztDoba,Opis)
values(1,1,1,1,20,'wygodny pokój z widokiem na Rynek oraz wawel'),
(1,2,1,1,20,'wygodny pokój z widokiem na Rynek'),
(1,3,2,1,30,'wygodny pokój z widokiem na Rynek dla par'),
(1,4,3,2,45,'wygodny pokój z widokiem na Wawel,który pomieści całą rodzinę'),
(1,5,5,4,45,'Przestrony pokój z dwoma łazienkami dla dużych rodzin'),
(1,6,5,5,45,'Przestrony pokój z trzema łazienkami stworzony z myślą o wycieczkach szkolnych'),
(1,7,5,5,70,'Przestrony pokój z trzema łazienkami stworzony z myślą o wycieczkach szkolnych'),
(1,8,5,5,70,'Przestrony pokój z trzema łazienkami stworzony z myślą o wycieczkach szkolnych'),
(2,1,1,1,22,'wygodny pokój z widokiem na Rynek oraz wawel'),
(2,2,1,1,25,'wygodny pokój z widokiem na Rynek'),
(2,3,2,1,33,'wygodny pokój z widokiem na Rynek dla par'),
(2,4,2,1,44,'wygodny pokój z widokiem na Wawel,dla nowożeńców'),
(2,5,3,2,45,'Przestrony pokój'),
(3,1,1,1,30,'wygodny pokój z widokiem na Rynek oraz wawel'),
(3,2,1,1,20,'Pokój idealny do postoju podczas zwiedzania'),
(3,3,3,2,33,'wygodny pokój z widokiem na Rynek dla rodziny'),
(3,4,4,3,45,'wygodny pokój z widokiem na Wawel'),
(3,5,5,5,50,'Przestrony pokój'),
(4,1,2,1,55,'wygodny pokój z widokie na morze'),
(4,2,1,1,20,'Pokój idealny do postoju podczas wyjazdów służbowych'),
(4,3,3,2,39,'wygodny pokój z widokiem na Rynek dla rodziny'),
(4,4,4,3,45,'wygodny pokój z widokiem na morze'),
(4,5,2,1,70,'Przestrony apartament z widokiem na całą okolice')
go
insert into Pokoje(IdHotelu,NrPokoju,LiczbaOsób,Liczbałóżek,KosztDoba,Status,Opis)
values(1,9,1,1,50,1,'Pokój dla palaczy');
--24 pokoje
```

```
--wprowadzenie danych do tabeli Udogodnienia
go
insert into Udogodnienia(IdHotelu,Nazwa)
values(1,'Parking'),(2,'Parking'),(3,'Parking'),
(1,'Basen'),(3,'Basen'),(1,'Sauna'),(2,'Sauna'),
(1,'Siłownia'),(2,'Siłownia'),(3,'Siłownia'),(4,'Siłownia'),
(1,'Bar'),(4,'Bar');
go
--wprowadzanie danych do tabeli wyposażenia
insert into Wyposażenia(IdPokoju,Nazwa)
values(1,'Sejf'),(2,'Sejf'),(3,'Sejf'),(4,'Sejf'),
(5,'Sejf'),(6,'Sejf'),(7,'Sejf'),(8,'Sejf'),(20,'Sejf'),
(21,'Sejf'),(22,'Sejf'),(23,'Sejf'),(24,'Sejf'),
(24,'Jacuzzi'),(11,'Dla palaczy'),(21,'Dla palaczy'),
(22,'Dla palaczy'),(21,'Dla palaczy'),(16,'Dla palaczy'),
(13,'Dla palaczy'),(21,'Dla niepełnosprawnych'),(22,'Dla niepełnosprawnych'),
(17,'Dla niepełnosprawnych'),(1,'Mini Barek'),(24,'Mini Barek'),(3,'Mini Barek'),
(9,'Telewizor'),(10,'Telewizor'),(11,'Telewizor'),(12,'Telewizor'),(13,'Telewizor')
(24,'Telewizor'),(14,'Telewizor'),(15,'Telewizor'),(16,'Telewizor'),
(24,'Lodówka'),(24,'Sauna'),(16,'Sejf'),(16,'Sejf'),(18,'Sejf'),(17,'Telewizor'),
(3,'Telewizor'),(4,'Telewizor'),(19,'Jacuzzi');
go
--wprowadzanie danych do tabeli rezerwacje

insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,1,'2021-07-06 7:00','2021-07-11 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,2,'2021-07-11 7:00','2021-07-13 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,3,'2021-07-01 7:00','2021-07-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,4,'2021-07-01 7:00','2021-08-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,5,'2021-06-21 7:00','2021-07-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,6,'2021-07-11 7:00','2021-07-26 8:00','Na miejscu');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,7,'2021-07-01 7:00','2021-07-03 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,8,'2021-08-01 7:00','2021-08-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,9,'2021-06-11 7:00','2021-07-01 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,10,'2021-07-01 7:00','2021-07-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,10,'2021-07-09 7:00','2021-07-16 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,11,'2021-07-11 7:00','2021-07-19 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,11,'2021-08-01 7:00','2021-08-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,12,'2021-07-01 7:00','2021-07-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,12,'2021-07-11 7:00','2021-07-26 8:00','Na miejscu');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,13,'2021-07-14 12:00','2021-07-28 18:00','Na miejscu');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,13,'2021-07-01 17:00','2021-07-16 8:00','Na miejscu');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,14,'2021-06-21 7:00','2021-06-26 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,14,'2021-08-01 7:00','2021-08-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,15,'2021-07-01 7:00','2021-07-06 8:00','Na miejscu');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,15,'2021-07-09 7:00','2021-07-16 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,15,'2021-07-18 7:00','2021-07-23 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,16,'2021-08-01 7:00','2021-08-06 8:00','Na miejscu');
```

```
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,16,'2021-07-01 7:00','2021-07-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,17,'2021-07-11 7:00','2021-07-13 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,17,'2021-07-01 7:00','2021-07-06 8:00','Na miejscu');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,18,'2021-07-01 7:00','2021-08-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,19,'2021-06-21 7:00','2021-07-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,19,'2021-07-21 7:00','2021-07-26 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,20,'2021-08-01 7:00','2021-08-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,20,'2021-07-01 7:00','2021-07-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,21,'2021-07-09 7:00','2021-07-16 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,21,'2021-06-18 7:00','2021-06-23 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,21,'2021-08-01 7:00','2021-08-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,22,'2021-07-19 7:00','2021-07-26 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,22,'2021-06-18 7:00','2021-06-23 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,22,'2021-07-01 7:00','2021-07-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,23,'2021-07-14 12:00','2021-07-28 18:00','Na miejscu');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,23,'2021-07-01 17:00','2021-07-12 8:00','Na miejscu');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,23,'2021-06-21 7:00','2021-06-26 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,24,'2021-08-01 7:00','2021-08-06 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(1,24,'2021-07-19 7:00','2021-07-26 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(3,24,'2021-06-18 7:00','2021-06-23 8:00','Przelew');
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,24,'2021-07-01 7:00','2021-07-11 8:00','Przelew');
```

```
--wprowadzanie niepoprawnych danych,sprawdzanie zabezpieczeń
go
insert into Klienci
values('Michał','Karpicki','michalo2.pl','Polska','ul. Armi 22 ','29-321','Katowice','333456789','1234567890123456')
--zły email dany klient nie zostanie wprowadzony.Jeżeli znajdował by się w większym zestawie to cały zestaw klientów
--nie zostanie wprowadzony do póki nie usunie się niepoprawnych danych
go
insert into Karty
values ('1122334411223344','11F','2019-02-03');--Zła data
--dana karta nie zostanie wprowadzona
go
insert into Karty
values ('11223344112233223','11F','2024-02-03');--niepoprawny numer karty
--dana karta nie zostanie wprowadzona
go
insert into Pokoje(IdHotelu,NrPokoju,LiczbaOsób,Liczbałóżek,KosztDoba,Opis)
values(1,-20,1,1,10,'');--nie poprawny numer pokoju
insert into Pokoje(IdHotelu,NrPokoju,LiczbaOsób,Liczbałóżek,KosztDoba,Opis)
values(1,10,-11,1,10,'');--nie poprawna liczba osób
insert into Pokoje(IdHotelu,NrPokoju,LiczbaOsób,Liczbałóżek,KosztDoba,Opis)
values(1,100,11,-1,10,'');--nie poprawna liczba łóżek
insert into Pokoje(IdHotelu,NrPokoju,LiczbaOsób,Liczbałóżek,KosztDoba,Opis)
values(1,19,11,1,-30,'');--nie poprawny koszt za dobe
--żaden z pokoji nie zostanie wprowadzony
--sprawdzanie poprawnosci wprowadzanych danych z tabeli Rezerwacje
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,24,'2021-07-01 7:00','2021-07-11 8:00','wtf');
--nie pozwala na nie poprawny sposob platnosci
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,24,'2021-10-21 7:00','2021-07-29 8:00','Przelew');
--nie wprowadza danych z data startowa pozniejsza niz data koncowa
insert into Rezerwacje(IdKlienta,IdPokoju,DataStart,DataStop,SposóbPłatności)
values(2,24,'2080-01-01 7:00','2021-07-29 8:00','Przelew');
--nie został wprowadzony wiersz z data pozniejsza niz data dzisiejszego dnia
```

Pierwsza część skryptów wprowadzają dane do tabel. Druga część skryptów sprawdzają zabezpieczenia przed nieoprawnymi danymi.

Etapy przedstawiają pełne przygotowanie poprawnie funkcjonującej bazy danych. Od etapu określenia wymagań i zaprojektowania konceptualnego do etapu stworzenia i testowania bazy danych.