**IDC Software**

**Politechnika Koszalińska**

2015

**Zestaw narzędzi do zarządzania siecią hoteli**

**Dokument detaliczny**

**Autorzy oraz funkcje w zespole**  
  
*Mikołaj Grygorcewicz* – programista aplikacji internetowej  
*Michał Łuniewski* – programista serwera bazodanowego  
*Sebastian Pawlikowski* – bazodanowiec, tester  
*Jakub Rajski* – programista aplikacji lokalnej  
***Marcin Rodak*** – kierownik zespołu, dokumentalista  
*Maksymilian Strzelecki* – programista aplikacji lokalnej

Spis treści

[1. Wprowadzenie 4](#_Toc421304459)

[1.1. Cel dokumentu 4](#_Toc421304460)

[1.2. Zakres projektu 4](#_Toc421304461)

[2. Standardy, nazewnictwo oraz użyte narzędzia 4](#_Toc421304462)

[2.1. Standardy dokumentacyjne 4](#_Toc421304463)

[2.2. Nazewnictwo 4](#_Toc421304464)

[2.3. Użyte narzędzia oraz technologie 4](#_Toc421304465)

[3. Specyfikacja bazy danych 5](#_Toc421304466)

[3.1. Model konceptualny bazy danych 5](#_Toc421304467)

[3.2. Model fizyczny bazy danych 5](#_Toc421304468)

[3.3. Słownik bazy danych 6](#_Toc421304469)

[3.3.1. Dane: 7](#_Toc421304470)

[3.3.2. Słownik pojęć: 7](#_Toc421304471)

[3.4. Sposób połączenia do bazy danych 7](#_Toc421304472)

[4. Serwer bazodanowy 8](#_Toc421304473)

[4.1. Funkcja serwera oraz podstawowe informacje 8](#_Toc421304474)

[4.2. Pakiety obsługiwane przez serwer 8](#_Toc421304475)

[4.2.1. Login 8](#_Toc421304476)

[4.2.2. Register 8](#_Toc421304477)

[4.2.3. UserPermissionLevel 9](#_Toc421304478)

[4.2.4. Hotel 10](#_Toc421304479)

[4.2.5. Room 11](#_Toc421304480)

[4.2.6. Reservation 11](#_Toc421304481)

[4.2.7. Template 12](#_Toc421304482)

[5. Aplikacja internetowa 13](#_Toc421304483)

[5.1. Możliwości aplikacji 13](#_Toc421304484)

[5.1. Uniwersalność aplikacji 13](#_Toc421304485)

[5.2. Struktura plików oraz ich znaczenie dla aplikacji 13](#_Toc421304486)

[6. Aplikacja lokalna 13](#_Toc421304487)

[6.1. Możliwości aplikacji 13](#_Toc421304488)

[6.2. Zaimplementowane klasy 14](#_Toc421304489)

[6.2.1. AddHotelForm 14](#_Toc421304490)

[6.2.2. AddRoomForm 15](#_Toc421304491)

[6.2.3. AddUserForm 15](#_Toc421304492)

[6.2.4. Hash 16](#_Toc421304493)

[6.2.5. Hotel 16](#_Toc421304494)

[6.2.6. PermissionLevel 16](#_Toc421304495)

[6.2.7. Room 17](#_Toc421304496)

[6.2.8. HotelsData 17](#_Toc421304497)

[6.2.9. Reservation 17](#_Toc421304498)

[6.2.10. Template 18](#_Toc421304499)

[6.2.11. Login 18](#_Toc421304500)

[6.2.12. Strings 19](#_Toc421304501)

[6.2.13. ServerAPIInterface 20](#_Toc421304502)

[6.2.14. MainForm 20](#_Toc421304503)

# Wprowadzenie

## Cel dokumentu

Dokument ma za zadanie sprecyzować sposób wykonania zbioru narzędzi, a także wyjaśnić jakie technologie oraz narzędzia zostały użyte do realizacji projektu. Dokument określa także podstawowe założenia projektu.

## Zakres projektu

Projekt składa się z czterech podstawowych elementów stanowiących spójną całość, są to:

* Baza danych – zawiera wszystkie informacje niezbędne do funkcjonowania aplikacji,
* Serwer danych – zapewnia komunikacje między aplikacjami a bazą danych,
* Aplikacja lokalna – skierowana do pracowników oraz właścicieli hoteli,
* Aplikacja internetowa – skierowana głównie do klientów hoteli.

Każdy z tych elementów stanowi część całości systemu, który zapewnia możliwość zarządzaniem siecią hoteli, tj. edycje liczby pokoi, ich standardów, dodawanie nowych hoteli czy też ich usuwanie. System pozwala także na zarządzanie rezerwacjami, w tym tworzenie nowych rezerwacji, edycje istniejących rezerwacji, zdalne rezerwacje przez klientów z dowolnej platformy przy pomocy strony internetowej. Prawidłowe działanie systemu zapewnia integracja obu aplikacji z serwerem danych,   
a co za tym idzie z bazą danych.

# Standardy, nazewnictwo oraz użyte narzędzia

## Standardy dokumentacyjne

Stworzenie dokumentu detalicznego zostało oparte o powszechne, ogólnie przyjęte normy   
i wytyczne do prowadzenia dokumentacji technicznej.

## Nazewnictwo

Nazewnictwo użyte w dokumencie oparte jest o ogólne normy dokumentacji technicznej, a także   
o nazewnictwo typowe dla technologii .NET Framework, MySQL, JSON oraz WWW.

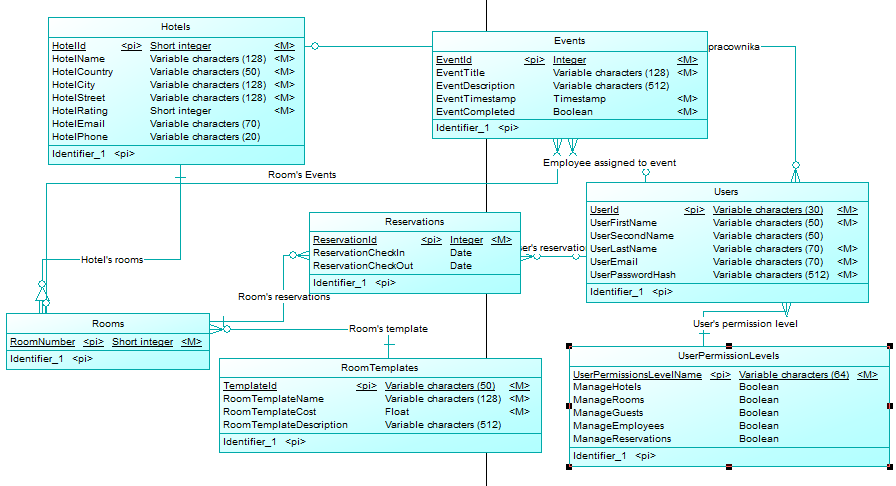
## Użyte narzędzia oraz technologie

Do realizacji projektu zostały użyte następujące narzędzia oraz technologie:

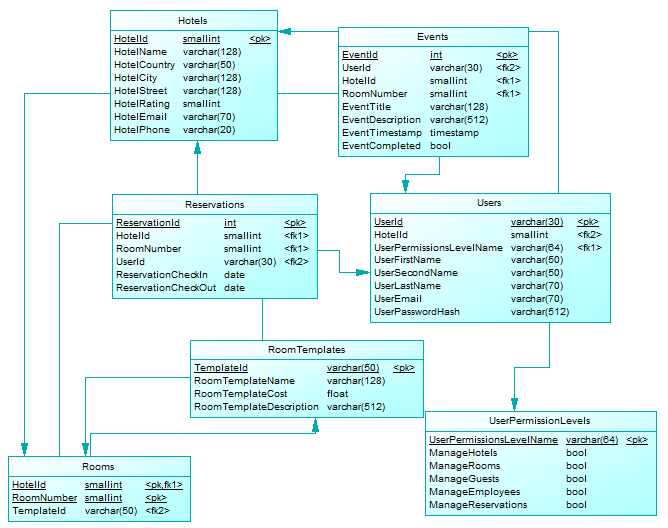
* SybasePowerDesigner – stworzenie modelu konceptualnego oraz fizycznego bazy danych, wygenerowanie kodu bazy danych dla MySQL 5.0,
* Microsoft VisualStudio – stworzenie aplikacji lokalnej oraz aplikacji internetowej,
* HTML5, CSS 3.0 – wygląd i treść stron www
* JQuery 1.11.1, JavaScript - logika aplikacji internetowej, nawigacja na stronie
* Websocket – komunikacja aplikacji internetowej z serwerem bazodanowym
* Sublime Text + Node JS – stworzenie kodu serwera
* Microsoft Office 2007 – stworzenie plików dokumentacyjnych

# Specyfikacja bazy danych

## Model konceptualny bazy danych



## Model fizyczny bazy danych



## Słownik bazy danych

|  |  |  |
| --- | --- | --- |
| **Nazwa Tabeli** | **Opis** | **Typ danych** |
| **Hotels** | | |
| HoteId | Zawiera Id hotelu | Short integer NOT NULL |
| HotelName | Zawiera nazwę hotelu | Vchar (128) NOT NULL |
| HotelCountry | Zawiera kraj w którym znajduje się hotel | Vchar (50) NOT NULL |
| HotelCity | Zawiera miasto w którym znajduje się hotel | Vchar (128) NOT NULL |
| HotelStreet | Zawiera adres ulicy na której znajduje się hotel | Vchar (128) NOT NULL |
| HotelRating | Zawiera informacje o ilości gwiazdek hotelu | Short integer |
| HotelEmail | Mail do hotelu | Vchar (70) |
| HotelPhone | Nr telefonu hotelu | Vchar (20) |
| **Rooms** | | |
| RoomNumber | Zawiera numer pokoju | Short integer NOT NULL |
| **Reservations** | | |
| ReservationId | Zawiera Id rezerwacji | Vchar (10) NOT NULL |
| ReservationCheckIn | Zawiera datę początku rezerwacji | Date |
| ReservationCheckOut | Zawiera datę końca rezerwacji | Date |
| **RoomTemplates** | | |
| TemplateId | Przechowuje Id template'u | Vchar (50) NOT NULL |
| RoomTemplateName | Nazwa template'u | Vchar (128) NOT NUL |
| RoomTemplateCost | Cena template'u | Float NOT NUL |
| RoomTemplateDescription | Opis template'u | Vchar (512) |
| **Events** | | |
| EventId | Przechowuje Id wydarzenia | Integer NOT NULL |
| EventTitle | Przechowuje tytuł wydarzenia | Vchar (128) NOT NUL |
| EventDescription | Przechowuje opis, wydarzenia | Vchar (512) |
| EventTimestamp | Przechowuje timestamp wydarzenia | Timestamp NOT NULL |
| EventCompleted | Czy wydarzenie zostało zakończone czy nie | Boolean NOT NULL |

|  |  |  |
| --- | --- | --- |
| **Nazwa Tabeli** | **Opis** | **Typ danych** |
| **Users** | | |
| UserId | Przechowuje Id użytkownika | Vchar (30) NOT NULL |
| UserFirstName | Pierwsze imię użytkownika | Vchar (50) NOT NULL |
| UserSecondName | Drugie imię użytkownika | Vchar (50) |
| UserLastName | Nazwisko użytkownika | Vchar (70) NOT NULL |
| UserEmail | Adres Email użytkownika | Vchar (70) NOT NULL |
| UserPasswordHash | Hash hasła użytkownika | Vchar (512) NOT NULL |
| **UserPermissionLevels** | | |
| UserPermissionsLevelName | Nazwa uprawnień | Vchar (64) NOT NULL |
| ManageHotels | Zarządzanie hotelami | Boolean |
| ManageRooms | Zarządzanie pokojami | Boolean |
| ManageGuests | Zarządzanie gośćmi | Boolean |
| ManageEmployees | Zarządzanie pracownikami | Boolean |
| ManageReservations | Zarządzanie rezerwacjami | Boolean |

### Dane:

* Boolean - zmienna przechowująca wartości true/false (tak/nie),
* Date - zmienna reprezentująca datę w formacie RRRR-MM-DD,
* Float - zmienna reprezentująca liczby zmiennoprzecinkowe,
* Integer - zmienna reprezentująca duże liczby całkowite zarówno dodatnie jak i ujemne,
* Short integer - zmienna reprezentująca średnie liczby całkowite zarówno dodatnie jak i ujemne,
* Vchar (x) - Zmienna reprezentująca ciąg znaków bez spacji gdzie x określa ilość znaków,
* Timestamp - Data i czas wstawienia rekordy do bazy.

### Słownik pojęć:

* Id - identyfikator, indywidualny numer nadawany obiektowi, rzeczy, osobie,
* Login - ciąg znaków oznaczający identyfikator służący do logowania się,
* Hash - ciąg znaków zawierający zakodowane hasło,
* Template - termin oznaczający opis wyposażenia pokoju ustalony wcześniej przez administratora systemu.

## Sposób połączenia do bazy danych

Żadna aplikacja nie łączy się bezpośrednio do bazy danych. Do tego celu użyty został serwer bazodanowy, za którego pośrednictwem z bazą może połączyć się zarówno aplikacja internetowa jak i lokalna. Upraszcza to znacznie kwestie programistyczne, a także uniemożliwia niepożądane zapytania czy też próby uszkodzenia bazy danych.

# Serwer bazodanowy

## Funkcja serwera oraz podstawowe informacje

Serwer bazodanowy pełni funkcje pośrednika pomiędzy aplikacjami a bazą danych. Aplikacje łączą się jedynie z serwerem, ten natomiast przekazuje dane pomiędzy bazą danych a konkretną aplikacją.

Serwer bazodanowy jest serwerem WebSocket. Do jego działania wymagane jest środowisko Node JS z zainstalowanymi pakietami ws i MySQL. Jako, że jest to aplikacja dostępowa do bazy danych, wymagana jest też działająca baza danych MySQL, załączoną do aplikacji. Komunikacja odbywa się za pomocą tzw. pakietów opisanych w języku JSON.

Każdy pakiet posiada pole command, które determinuje pośrednio tablicę, na której chce operować użytkownik. Niektóre pakiety mają pole action, które pozwala na standardowe operacje CRUD na wskazanych polem command danych.

## Pakiety obsługiwane przez serwer

### Login

Pakiet zaloguje użytkownika na konto o podanych danych.

#### Przykładowy pakiet:

{   
 "command": "login",   
 "loginData":  
 {  
 "userEmail": "a@a.com",  
 "userPasswordHash": "abcd"  
 }  
}

#### Pola

* command – "login"
* loginData.userEmail – string
* loginData.userPasswordHash - string

### Register

Pakiet pozwala na rejestracje użytkowników. Pole userHotelId wskazuje na jego charakter, jeśli jest null – jest to gość, jeśli nie – jest to pracownik hotelu o podanym id.

#### Przykładowy pakiet:

{  
 "command": "register",   
 "registerData":  
 {  
 "userId": "B12345",   
 "userPermissionLevel": "guest",   
 "userFirstName": "Jan",  
 "userSecondName": "",  
 "userLastName": "Kowalski",   
 "userEmail": "v@a.com",  
 "userPasswordHash": "zxcasd",  
 "userHotelId": null   
 }

}

#### Pola

* command – "register"
* registerData.userId – string
* registerData.userPermissionLevel – string
* registerData.userFirstName – string
* registerData.userSecondName – string
* registerData.userLastName – string
* registerData.userEmail – string
* registerData.userPasswordHash – string
* registerData.userHotelId - int

### UserPermissionLevel

Pakiet pozwala dokonywać operacji na poziomach uprawnień, jakie można przydzielić użytkownikowi.

#### Przykładowy pakiet:

{  
 "command": "permissionLevel",  
 "action": "update",  
 "UserPermissionLevelName": "superadmin",  
 "ManageHotels": true,   
 "ManageRooms": true,   
 "ManageGuests": true,  
 "ManageEmployees": true,   
 "ManageReservations": true  
}

#### Pola

* command – "permissionLevel"
* action - string
* UserPermissionsLevelName – string, identyfikator
* ManageHotels - bool
* ManageRooms - bool
* ManageGuests – bool
* ManageEmployees – bool
* ManageReservations – bool

#### Dostępne akcje:

* get – zwraca listę dostępnych poziomów uprawnień. Pole z listą ma nazwę "list", a poszczególne pola wierszy są tożsame z polami w bazie danych (i tym samym z tymi w tym pakiecie poza command i action),
* add – dodaje nowy poziom uprawnień. Wszystkie pola muszą być sprecyzowane,
* update – uaktualnia istniejący już poziom uprawnień. Nie można zmienić jego nazwy, w takim wypadku należy usunąć stary poziom i dodać nowy, poprawny. Wszystkie pola muszą być sprecyzowane,
* delete – usuwa istniejący poziom uprawnień. Wszystkie pola poza nazwą nie są wymagane i zostaną zignorowane.

### Hotel

Pakiet pozwala operować na hotelach w bazie danych.

#### Przykładowy pakiet:

{  
 "command": "hotel",   
 "action": "add",  
 "HotelName": "Fajny hotel",  
 "HotelCountry": "pl",  
 "HotelCity": "Koszalin",  
 "HotelStreet": "Majowa 5",  
 "HotelRating": 5,  
 "HotelEmail": fajny@hotel.pl,  
 "HotelPhone": "555-444-333"  
}

#### Pola

* command – "hotel"
* action - string
* HotelId – liczba (autoincrement)
* HotelName – string
* HotelCountry – string
* HotelCity – string
* HotelStreet – string
* HotelRating – int
* HotelEmail – string
* HotelPhone - string

#### Dostępne akcje:

* get – zwraca listę dostępnych hoteli. Pole z listą ma nazwę "list", a poszczególne pola wierszy są tożsame z polami w bazie danych (i tym samym z tymi w tym pakiecie poza command i action),
* add – dodaje nowy hotel. Wszystkie pola muszą być sprecyzowane poza HotelId, które zostanie zignorowane,
* update – uaktualnia istniejący już hotel. Nie można zmienić jego id, w takim wypadku należy usunąć stary poziom i dodać nowy, poprawny. Wszystkie pola muszą być sprecyzowane,
* delete – usuwa istniejący hotel. Wszystkie pola poza HotelId nie są wymagane i zostaną zignorowane.

### Room

Pakiet pozwala operować na pokojach w bazie danych.

#### Przykładowy pakiet:

{  
 "command": "room",  
 "action": "delete",  
 "HotelId": 1,  
 "RoomNumber": 201  
}

#### Pola

* command – "room"
* action - string
* HotelId – int
* RoomNumber – int
* TemplateId - string

#### Dostępne akcje:

* get – zwraca listę dostępnych pokoi we wszystkich hotelach (można po stronie klienta rozróżniać po HotelId). Pole z listą ma nazwę "list", a poszczególne pola wierszy są tożsame z polami w bazie danych (i tym samym z tymi w tym pakiecie poza command i action),
* add – dodaje nowy pokój. Wszystkie pola muszą być sprecyzowane. Warto zauważyć, że HotelId i TemplateId powinny być wybierane z ComboBoxów, bo zależą od innych tabel,
* update – uaktualnia istniejący już pokój. Nie można zmienić jego hotelu ani numeru, w takim wypadku należy usunąć stary poziom i dodać nowy, poprawny. Wszystkie pola muszą być sprecyzowane,
* delete – usuwa istniejący pokój. Wymagane są tylko pola HotelId i RoomNumber.

### Reservation

Pakiet pozwala na otrzymanie informacji o rezerwacjach bądź dodać nową.

#### Przykładowy pakiet:

{  
 "command": "reservation",  
 "action": "get"  
}

#### Pola

* command – "reservation"
* action - string
* ReservationId – int (autoincrement)
* HotelId – int
* RoomNumber – int
* UserId – string
* ReservationCheckIn – string w postaci "YYYY-MM-DD"
* ReservationCheckOut – string w postaci "YYYY-MM-DD"

#### Dostępne akcje

* get – zwraca listę wszystkich rezerwacji. Pole z listą ma nazwę "list", a poszczególne pola wierszy są tożsame z polami w bazie danych (i tym samym z tymi w tym pakiecie poza command i action),
* add – dodaje nową rezerwację. Wszystkie pola muszą być sprecyzowane. Pola HotelId, RoomNumber i UserId powinny być w postaci ComboBoxów. Jeśli UserId nie istnieje, należy go wcześniej dodać do tablicy z uprawnieniami gościa (w tle podając tylko numer jego dowodu),
* update – uaktualnia istniejącą już rezerwację. Nie można zmienić jej id, w takim wypadku należy usunąć starą i dodać nową. Wszystkie pola muszą być sprecyzowane,
* delete – usuwa istniejącą już rezerwację. Wszystkie pola poza ReservationId nie są wymagane i zostaną zignorowane.

### Template

Pakiet pozwala na zarządzanie szablonami pokoi.

#### Przykładowy pakiet

{  
 "command": "template",  
 "action": "add",  
 "TemplateId": "lux1",  
 "RoomTemplateName": "Apartament luksusowy 1",  
 "RoomTemplateCost": 119,99,  
 "RoomTemplateDescription": "dowolny opis"  
}

#### Pola

* command – "hotel"
* action - string
* HotelId – liczba (autoincrement)
* HotelName – string
* HotelCountry – string
* HotelCity – string
* HotelStreet – string
* HotelRating – int
* HotelEmail – string
* HotelPhone ***–*** string

#### Dostępne akcje:

* get – zwraca listę wszystkich szablonów pokoi. Pole z listą ma nazwę "list", a poszczególne pola wierszy są tożsame z polami w bazie danych (i tym samym z tymi w tym pakiecie poza command i action),
* add – dodaje nowy szablon pokoju. Wszystkie pola muszą być sprecyzowane,
* update – uaktualnia istniejący już szablon pokoju. Nie można zmienić jego TemplateId, w takim wypadku należy usunąć stary i dodać nowy. Wszystkie pola muszą być sprecyzowane,
* delete – usuwa istniejący szablon. Wszystkie pola poza TemplateId nie są wymagane i zostaną zignorowane.

# Aplikacja internetowa

## Możliwości aplikacji

Aplikacja internetowa służy głównie gościom hotelowym do zarządzania swoimi rezerwacjami,   
tj. dodawania, usuwania czy też modyfikowania swoich rezerwacji. Udostępnia ona także informacje o hotelach, pozwala na spersonalizowanie swojej rezerwacji poprzez podstawowe funkcje takie jak dobór terminu rezerwacji, a także dodatkowe, jak na przykład dobór niestandardowych elementów pokoju. Aplikacja umożliwia także skontaktowanie się z personelem hotelu oraz wpisanie się do księgi gości.

## Uniwersalność aplikacji

Dzięki skorzystaniu z modelu bootstrap aplikacja internetowa jest uniwersalna, to znaczy dopasowuje swój wygląd do urządzenia, na jakim strona jest wyświetlana. Dzięki temu goście hotelu mogą zarządzać swoimi rezerwacjami bez względu na to gdzie są, ani jakiego urządzenia używają – wystarczy dostęp do Internetu oraz smartfon, tablet czy jakiekolwiek inne urządzenie pozwalające przeglądać strony www.

## Struktura plików oraz ich znaczenie dla aplikacji

Aplikacja internetowa składa się z kilku plików, z których każdy odpowiada za konkretne działanie aplikacji. Są to następujące pliki:

* index.html - plik strony www, zawiera wszystkie widoki i podstrony
* requests.js - testowy plik sprawdzający, czy komunikacja z serwerem została nawiązana
* modal.js - plik pokazujący obrazy po kliknięciu na miniaturkę
* navigate.js - plik nawigacji po stronie, określa logikę nawigacji, odpowiada za wyświetlanie odpowiednich treści na stronie
* bootstrap-datepicker.js - plik logiki i funkcji kalendarza
* bootstrap-datepicker.css - plik stylu dla kalendarza
* map.svg - mapa Polski w formacie grafiki wektorowej
* extended-sheet.css - plik stylu niektórych elementów, między innymi okienka wyświetlającego obraz oraz wygląd miniaturki obrazu na stronie

# Aplikacja lokalna

## Możliwości aplikacji

Aplikacja lokalna skierowana jest do pracowników oraz właścicieli, czy też zarządców hoteli. Pozwala ona na zarządzanie wszystkimi dostępnymi parametrami, czy to tymi dotyczącymi pokoi   
w jednym z hoteli, konkretnego hotelu, czy też użytkownikami, w tym pracownikami i ich uprawnieniami. Dzięki aplikacji lokalnej możemy dodać nowy hotel do naszej oferty, co automatycznie będzie widoczne dla gości hotelowych, czy też usunąć hotel – wówczas nie będzie on już dostępny z poziomu aplikacji internetowej.

## Zaimplementowane klasy

### AddHotelForm

Jest to klasa formularza dodania nowego hotelu. Zawiera wszystkie etykiety, przyciski oraz pola tekstowe niezbędne do poprawnego działania formularza, a także metody, które zostają wywołane w momencie użycia jednego z przycisków.



### AddRoomForm

Jest to klasa formularza dodania nowego pokoju. Zawiera wszystkie etykiety, przyciski, pola tekstowe oraz pola wyboru (ComboBox) niezbędne do poprawnego działania formularza, a także metody, które zostają wywołane w momencie użycia jednego z przycisków.



### AddUserForm

Jest to klasa formularza dodania nowego użytkownika. Zawiera wszystkie etykiety, przyciski, pola tekstowe oraz pola wyboru (ComboBox) niezbędne do poprawnego działania formularza, a także metody, które zostają wywołane w momencie użycia jednego z przycisków.



### Hash

Jest to klasa odpowiedzialna za szyfrowanie haseł, dopiero po wykonaniu metody tej klasy hasło jest przesyłane do serwera bazodanowego.



### Hotel

Jest to klasa reprezentująca encje *Hotel* bazy danych, każde z ustawień odnosi się do konkretnego pola w encji bazy danych, o tej samej nazwie.



### PermissionLevel

Jest to klasa reprezentująca encje *PermissionLevel* bazy danych, każde z ustawień odnosi się do konkretnego pola w encji bazy danych, o tej samej nazwie.



### Room

Jest to klasa reprezentująca encje *Room* bazy danych, każde z ustawień odnosi się do konkretnego pola w encji bazy danych, o tej samej nazwie.



### HotelsData

Jest to klasa reprezentująca encje *HotelsData* bazy danych, każde z ustawień odnosi się do konkretnego pola w encji bazy danych, o tej samej nazwie.



### Reservation

Jest to klasa reprezentująca encje *Reservation* bazy danych, każde z ustawień odnosi się do konkretnego pola w encji bazy danych, o tej samej nazwie.



### Template

Jest to klasa reprezentująca encje *Template* bazy danych, każde z ustawień odnosi się do konkretnego pola w encji bazy danych, o tej samej nazwie.



### Login

Jest to klasa okna logowania. Zawiera wszystkie etykiety, przyciski oraz pola tekstowe niezbędne do poprawnego działania formularza, a także metody, które zostają wywołane w momencie użycia jednego z przycisków. Klasa ta po poprawnym zalogowaniu wywołuje główne okno aplikacji.



### Strings

Jest to klasa przechowująca wszystkie stringi, które są pobierane z bazy danych.



### ServerAPIInterface

Jest to klasa odpowiadająca za komunikacje z serwerem bazodanowym, występujące w niej metody zawierają wszystkie używane w programie zapytania do bazy danych. Zawarte w tej klasie metody   
są wywoływane w odpowiednich klasach lub formularzach aplikacji lokalnej. Stworzenie osobnej klasy obsługującej zapytania do serwera bazodanowego sprawia, że kod aplikacji jest bardziej czytelny oraz nie zawiera zbędnej nadmiarowości.

### MainForm

Jest to klasa głównego widoku aplikacji. Zawiera ona wszystkie użyte w interfejsie graficznym pola tekstowe, deklaracje menu użytkownika, komponenty .NET Framework oraz metody pozwalające na obsługę zdarzeń, takich jak załadowanie odpowiednich widoków, czy wykonanie akcji po wybraniu jednego z przycisków (np. spośród opcji dostępnych w menu). Obsługuje ona także edycje pól bazy danych.