|  |  |
| --- | --- |
| Sprawozdanie | |
| Marcin Żmudka 241137  Krzysztof Panek 241280 | Inżynieria Oprogramowania |
| Data odbycia ćwiczenia: 09.10.2019  Środa: 17:05 – 18:35 | Prowadzący:  prof. dr hab. inż. Jan Magott |

# Opis „świata rzeczywistego

## Opis zasobów ludzkich

Gość może utworzyć konto w systemie, wtedy staje się użytkownikiem. Gość może przeglądać dostępne pokoje hotelowe, po podaniu następujących informacji: data przyjazdu, data wyjazdu, miejscowość. Użytkownik może dokonywać rezerwacji pokoi hotelowych. Do założenia konta potrzebne są następujące informacje: imię, nazwisko, kraj pochodzenia, data urodzenia, preferowany język komunikacji. Przy rezerwacji należy podać następujące dane: liczbę osób, liczbę osób dorosłych, liczbę dzieci, data przyjazdu, data wyjazdu. Klienci mogą wybrać rodzaj pokoju hotelowego w zależności od potrzeb. Użytkownik może przeglądać swoje rezerwacje oraz odwołać je lub wyświetlić potwierdzenie jednej z nich. Pracownik administracji może dodawać i usuwać hotele z systemu, zarządzać ich dostepnością. Pokój po rezerwacji przez użytkownika systemu będzie niewidoczny dla innych klientów szukających noclegu w tym samym terminie. Hotele posiadają atrybuty takie jak: lokalizacja, ilość wolnych pokoi, ceny pokoi, rodzaje pokoi. Pokój ma swój termin dostępności, cenę, nazwę. Klient ma prawo do odwołania rezerwacji w warunkach przedstawionych przez administratora.

## Przepisy, strategia firmy

Klient nie ponosi odpowiedzialności za błędne przypisanie danych przez pracownika administracji. Za wszystkie niepoprawne informacje odpowiedzialność ponosi dany administrator. Pracownik ma dostęp do wszystkich informacji o gościach w poszczególnych hotelach.

# Dane techniczne

Klient może przeglądać katalog dostępnych hoteli za pomocą dedykowanej aplikacji. Zakłada się, że jest ograniczona liczba miejsc pokoi w hotelach, dlatego obowiązuje pierwszeństwo rezerwacji tzn. kto zarezerwuje się pierwszy temu przysługuje hotel. Dostępnych będzie około 20 hoteli, każdy będzie posiadał, niemniej niż 5 pokoi.

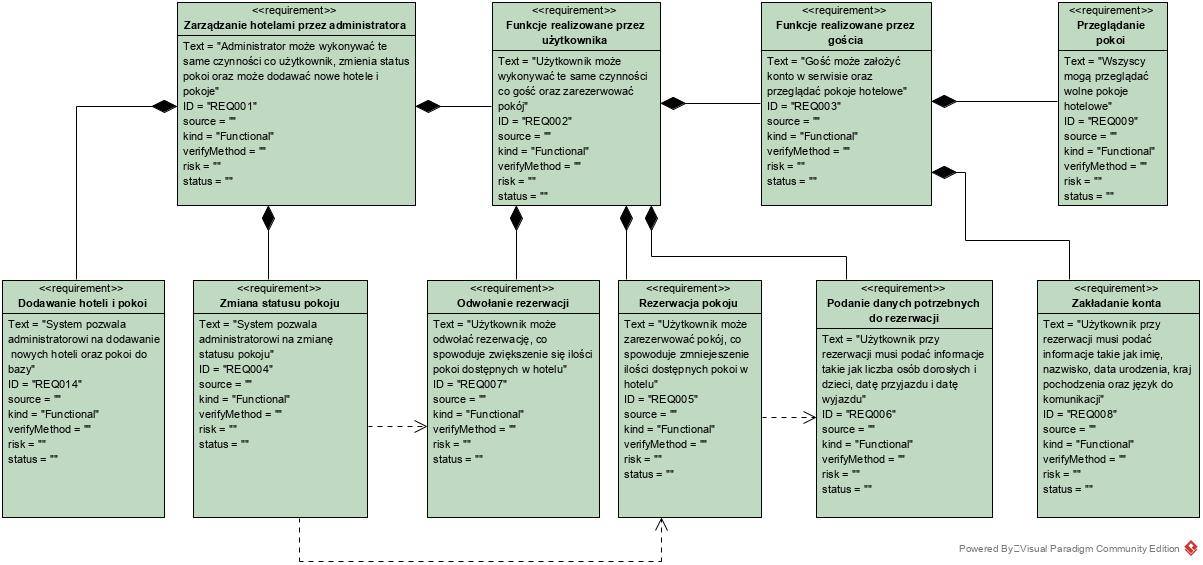
# Wymagania stawianej aplikacji

## Lista wymagań funkcjonalnych

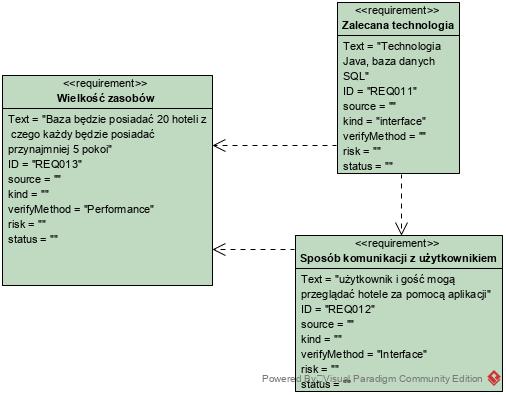
1. System zawiera katalog hoteli, do którego można wstawiać nowe hotele oraz usunąć istniejące.
2. Można zarezerwować różne pokoje, w wielu hotelach, w wielu miejscowościach, w wielu terminach.
3. Pokój w hotelu może zostać zarezerwowany i wtedy dany pokój nie jest już widoczny dla innych użytkowników.
4. Do dokonania rezerwacji potrzebne są takie dane jak: imię i nazwisko, liczba osób dorosłych, liczba dzieci, data urodzenia, preferowany język komunikacji, data przyjazdu, data wyjazdu, kraj pochodzenia.
5. Gdy rezerwacja się powiodła system drukuje potwierdzenie rezerwacji.
6. Przeglądanie hoteli możliwe jest dopiero po podaniu dat. Można listę hoteli sortować według ceny.
7. Administrator może ustawić hotel jako zajęty lub wolny w danym terminie.
8. Administrator może dodać nowy hotel wraz z nowymi pokojami oraz dodać nowe pokoje do istniejącego hotelu.
9. Istnieje możliwość odwołania rezerwacji przy zachowaniu warunków podanych przez administratora.
10. Klient może założyć konto w serwisie, podając wymagane dane – imię, nazwisko, data urodzenia, kraj pochodzenia, preferowany język komunikacji.

## Lista wymagań niefunkcjonalnych

1. Baza danych SQL.
2. Aplikacja z interfejsem graficznym wykorzystująca środowisko JAVA.
3. Rezerwacje dokonywane są przez klientów.
4. Wstawianie, usuwanie, ustawianie dostępności ( wolne/zajęte ) dokonywane jest przez pracowników administracji.



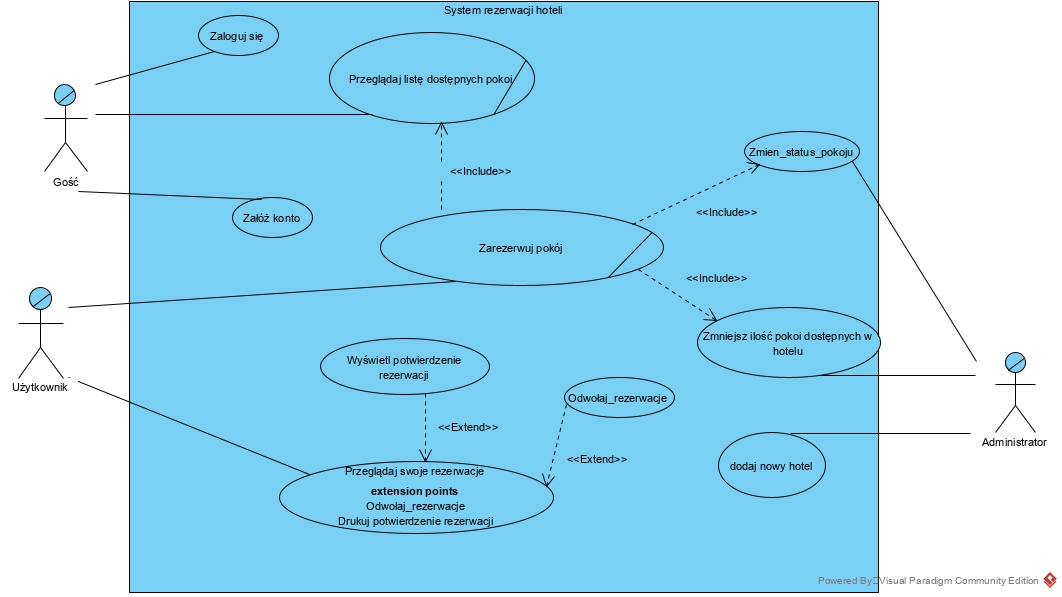
Rysunek 1 Diagram Wymagań Funkcjonalnych



Rysunek 2 Diagram wymagań niefunkcjonalnych

|  |  |
| --- | --- |
| Krzysztof Panek 241280  Marcin Żmudka 241137 | Inżynieria Oprogramowania |
| Środa godz: 17:05 |
| Sprawozdanie 3  Diagram Przypadków Użycia | Prowadzący  prof. dr hab. inż. Jan Magott |
| System rezerwacji miejsc w hotelach | |

Diagram przypadków użycia



|  |  |  |
| --- | --- | --- |
| **AKTOR** | **OPIS** | **PRZYPADKI UŻYCIA** |
| Gość | Gość może przeglądać dostępne pokoje i założyć konto w serwisie | - PU Załóż konto  - PU Zaloguj się  - PU Przeglądaj listę dostępnych pokoi jest powiązane przez <<include>> z PU Zarezerwuj pokój |
| Użytkownik | Użytkownik może zarezerwować pokój i odwołać rezerwację | - PU Zarezerwuj Pokój powiązane przez <<include>> z PU Zmień status pokoju oraz powiązane przez <<include>> z PU zmniejsz ilość dostępnych pokoi w hotelu  - PU Przeglądaj swoje rezerwacje powiązane przez <<extend>> z PU Wyświetl swoje rezerwacje oraz połączone przez <<extend>> z PU Odwołaj rezerwację |
| Administrator | Zarządza może edytować stan dostępności pokoi w hotelu | - PU Zmień status Pokoju powiązane przez  <<include>> z PU zarezerwuj pokój  - PU Zmniejsz Ilość Pokoi Dostępnych w Hotelu  powiązane przez <<include>> z PU Zarezerwuj Pokój  -PU Dodaj nowy hotel |

PU Zaloguj się

OPIS

CEL: Logowanie do systemu

WS: inicjalizacja przez uruchomienie programu

WK: Zalogowania się do swojego konta. Możliwość dokonywania rezerwacji, przeglądania rezerwacji etc.

PRZEBIEG:

1. Kliknięcie przycisku “Zaloguj:.
2. Podanie loginu oraz hasła
3. Jeśli powyższe dane były prawidłowe, nastąpi zalogowanie.

PU Zmień status pokoju

OPIS

CEL: Zmiana statusu pokoju

WS: jest wywoływany podczas wywołania PU Zarezerwuj pokój

WK: Zmiana statusu pokoju na zajęty lub wolny w zależności od decyzji użytkownika

PRZEBIEG:

1. Po decyzji użytkownika o zarezerwowaniu lub rezygnacji z pokoju, jego status zmieni się na zajęty lub wolny. Jeśli to administrator wywołał ten przypadek, może on zmienić status pokoju na dowolny.

2. Zaktualizowany status będzie widoczny w aplikacji.

PU Zmniejsz ilość pokoi dostępnych w hotelu

OPIS

CEL: Informacja o dostępnych pokojach w danym hotelu!

WS: jest konieczny podczas wywołania PU Zarezerwuj pokój

WK: Zmniejszenie ilości pokoi dostępnych w hotelu

PRZEBIEG:

1. Po zarezerwowaniu pokoju przez użytkownika, ilość dostępnych pokoi zmniejszy się

2. Zarezerwowane pokoje nie będą widoczne w systemie.

PU Odwołaj rezerwację

OPIS

CEL: Odwołanie rezerwacji

WS: może być wywołany z PU Przeglądaj swoje rezerwacje

WK: Zwiększenie się ilości wolnych pokoi w hotelu.

PRZEBIEG:

1. Kliknięcie przycisku, “Odwołaj rezerwacje”.
2. Po odwołaniu rezerwacji ilość pokoi w hotelu zwiększy się.

PU Załóż konto

OPIS

CEL: Założenie konta w systemie

WS: inicjalizacja poprzez uruchomienie programu.

WK: Założenie konta użytkownika, poprzez podanie wymaganych danych oraz utworzenie loginu wraz z hasłem. Gość staje się użytkownikiem, zostaje natychmiastowo zalogowany.

PRZEBIEG:

1. Kliknięcie przycisku “Załóż konto”.

2. Zakładanie konta wymaga, aby Gość podał następujące dane: imię, nazwisko, data urodzenia, kraj pochodzenia, preferowany język komunikacji. Dodatkowo klient musi wymyślić login, który nie istnieje już w bazie oraz hasło o minimalnej długości 8 znaków.

3. Jeśli założenie przebiegło pomyślnie. Gość staje się użytkownikiem, zostaje natychmiastowo zalogowany na nowe konto. ( Wywołaj PU Zaloguj ).

PU Przeglądaj listę dostępnych pokoi

OPIS

CEL: Znalezienie pokoju najbardziej odpowiadającego wymaganiom Gościa bądź Użytkownika

WS: Inicjalizacja poprzez uruchomienie programu

WK: Podanie pokoju o atrybutach obowiązkowych: cena, data przyjazdu, data wyjazdu, miejscowość.

PRZEBIEG:

1. Użytkownik podaje następujące parametry: nazwa, cena, data przyjazdu, data wyjazdu.
2. Kliknięcie przycisku „Wyszukaj rezerwacje”.
3. Jeśli istnieje pokój o podanych atrybutach jest zwracany, w przeciwnym razie zwracana jest informacja o braku dostępnych pokoi.

PU Wyświetl potwierdzenie rezerwacji

OPIS

CEL: Wyświetlenie potwierdzenia dokonania rezerwacji

WS: wywołania z PU Zarezerwuj pokój

WK: Wyświetlenie potwierdzenia rezerwacji dokonanej przez użytkownika.

PRZEBIEG

1. Wybranie rezerwacji.
2. Kliknięcie przycisk “Wyświetl potwierdzenie”.

PU Zarezerwuj Pokój

OPIS

CEL: Dokonanie rezerwacji pokoju o podanych wymaganiach

WS: wywołać PU Przeglądaj listę dostępnych pokoi

WK: Zmniejszenie liczby dostępnych pokoi w hotelu, zmienienie statusu pokoju

PRZEBIEG

1. Wywołaj PU Przeglądaj listę dostępnych pokoi.
2. Kliknięcie przycisku “Zarezerwuj pokój”
3. Podanie liczby osób dorosłych, liczby dzieci.
4. Wywołaj PU Zmień Status Pokoju
5. Wywołaj PU Zmniejsz ilość pokoi w hotelu.

PU Przeglądaj Swoje Rezerwacje

CEL: Dokonanie przeglądu dokonanych rezerwacji

WS: Zalogowany użytkownik

WK: Wyświetlenie dokonanych rezerwacji

PRZEBIEG:

1. Kliknięcie przycisku “Wyświetl swoje rezerwacje” dostępnego w głównym menu programu.
2. Jeśli użytkownik posiada rezerwacje, zostaną one wyświetlone, w przeciwnym razie otrzyma informacje, iż nie posiada jeszcze dokonanych rezerwacji.
3. Jeśli wszystko przebiegnie pomyślnie, można wywołać PU Wyświetl Potwierdzenie Rezerwacji.
4. Jeśli wszystko przebiegnie pomyślnie, można wywołać PU Odwołaj Rezerwacje.

PU Dodaj nowy Hotel

CEL: Dodanie nowego pokoju wraz z pokojami do systemu

WS: Zalogowany administrator

WK: Zmienienie liczby dostępnych hoteli

PRZEBIEG:

1. Kliknięcie przycisku “Dodaj nowy hotel”.
2. Wypełnienie formularza dotyczącego hotelu, zawierający m.in nazwę hotelu, liczbę pokoi, nazwę każdego pokoju, ceną każdego pokoju za dobę, krótki opis każdego pokoju.
3. Kliknięcie przycisku “Zatwierdź”.
4. Jeśli wszystko przebiegło pomyślnie, hotel powinien pojawić się w aplikacji.

# Diagramy czynności

Diagram czynności przypadku użycia PU Wyświetl Potwierdzenie Rezerwacji

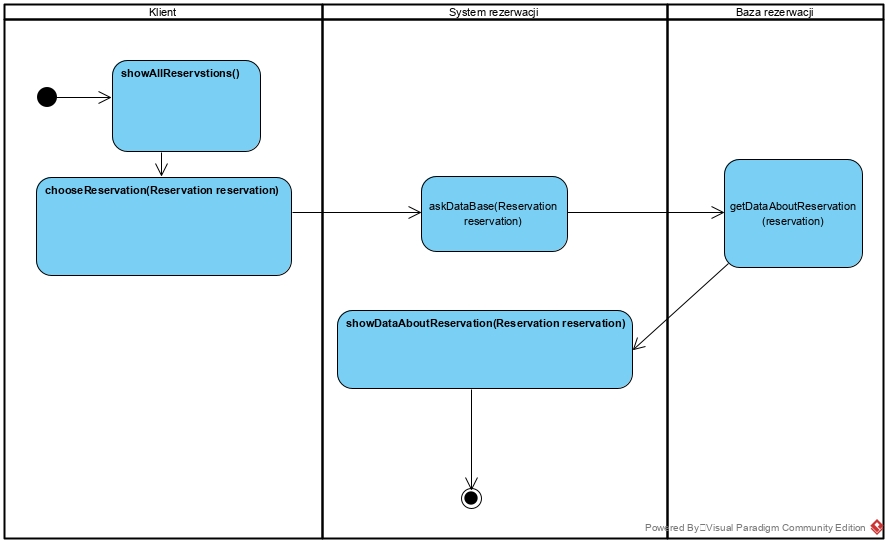


Diagram czynności świata biznesowe, dla dokonania rezerwacji:

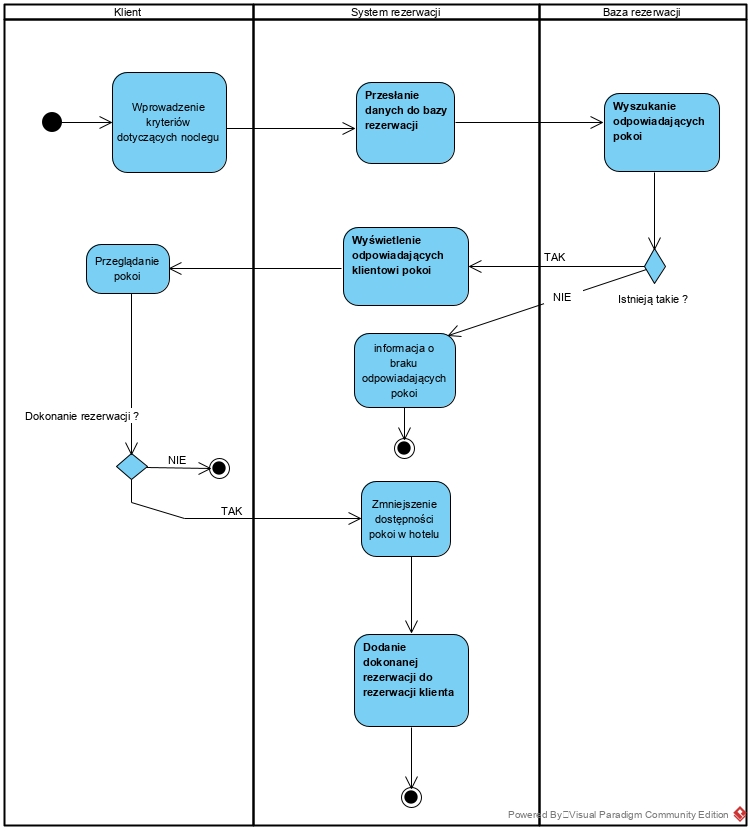


Diagram czynności dla PU Dodaj nowy Hotel:

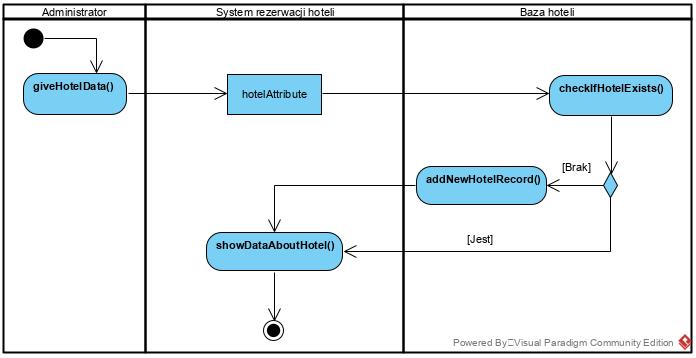
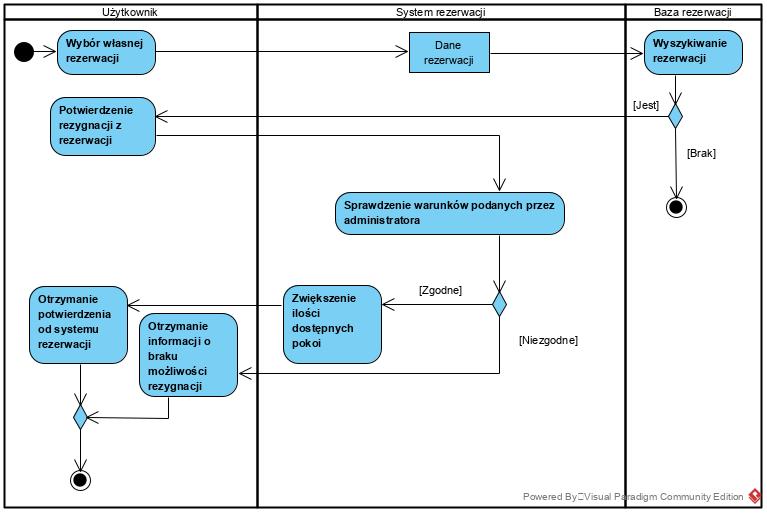
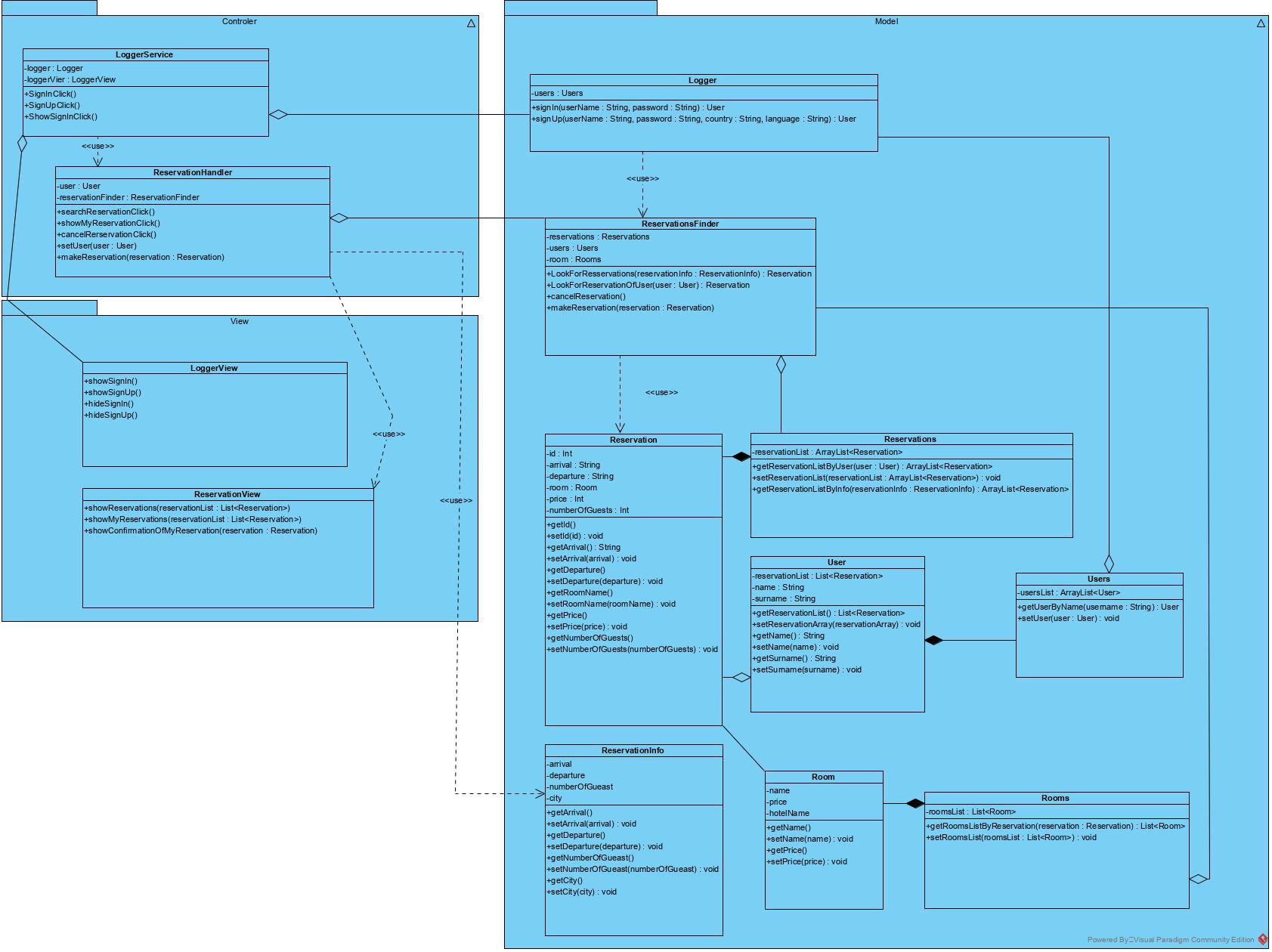


Diagram czynności świata rzeczywistego dla odwołania rezerwacji:



# Diagram Klas



# Szkielet kodu źródłowego

**package** Controler;

**public** **class** LoggerService{

**private** Logger logger;

**private** LoggerView loggerView;

**public** SignInClick()

**public** SignUpClick()

**public** ShowSignInClick()

}

**package** Controler;

**public** **class** ReservationHandler{

**private** User user;

**private** ReservationFinder reservationFinder;

**public** searchReservationClick()

**public** showMyReservationClick()

**public** cancelRerservationClick()

**public** setUser(User)

**public** makeReservation(Reservation reservation)

}

**package** View;

**public** **class** LoggerView{

**public** showSignIn()

**public** showSignUp()

**public** hideSignIn()

**public** hideSignUp()

}

**package** View;

**public** **class** Reservation View{

**public** showReservations(Reservation reservationArray)

**public** showMyReservations(Reservation reservationArray)

**public** showConfirmationOfMyReservation(Reservation reservation)

}

**package** Model;

**public** **class** Logger{

**private** Users users;

**public** User signIn(String userName, String password)

**public** User signUp(String userName, String password, String country, String language)

}

**package** Model;

**public** **class** ReservationFinder{

**private** Reservations reservations;

**private** Users users;

**private** Rooms rooms;

**public** Reservation LookForResservations(ReservationInfo reservationInfo)

**public** Reservation LookForReservationOfUser()

**public** cancelReservation()

**public** makereservation(Reservation reservation)

}

**package** Model;

**public** **class** Reservation{

**private** id;

**private** arrival;

**private** departure;

**private** room;

**private** price;

**private** numberOfGuests;

**public** getId()

**public** **void** setId(id)

**public** getArrival()

**public** **void** setArrival(arrival)

**public** getDeparture()

**public** **void** setDeparture(departure)

**public** getRoomName()

**public** **void** setRoomName(roomName)

**public** getPrice()

**public** **void** setPrice(price)

**public** getNumberOfGuests()

**public** **void** setNumberOfGuests(numberOfGuests)

}

**package** Model;

**public** **class** Reservations{

**private** List <ReservationList> Reservation = **new** ArrayList<>();

**public** ArrayList<Reservation> getResevationList(User user);

**public** **void** setReservationList(ArrayList<Reservation> reservationList)

**public** ArrayList<Reservation> getREservationListByInfo(ReservationInfo reservationInfo)

}

**package** Model;

**public** **class** User{

**private** List<Reservation> reservationList;

**private** name;

**private** surname;

**public** getReservationList(List<ReservationList>)

**public** **void** setReservationArray(reservationArray)

**public** getName()

**public** **void** setName(name)

**public** getSurname()

**public** **void** setSurname(surname)

}

**package** Model;

**public** **class** Users{

**private** List <UserList> User = **new** ArrayList<>();

**public** User getUserByName(String username);

**public** **void** setUser(User user)

}

**package** Model;

**public** **class** Room{

**private** name;

**private** price;

**private** hotelName;

**public** getName()

**public** **void** setName(name)

**public** getPrice()

**public** **void** setPrice(price)

}

**package** Model;

**public** **class** Rooms{

**private** List<Room> roomList;

**public** List<Room> getRoomsListByReservation(Reservation reservation)

**public** **void** setRoomsList(List<Room> roomList)

}

**package** Model;

**public** **class** ReservationInfo{

**private** arrival;

**private** departure;

**private** numberOfGuests;

**private** city;

**public** getArrival()

**public** **void** setArrival(arrival)

**public** getDeparture()

**public** **void** setDeparture(departure)

**public** getNumberOfGuests()

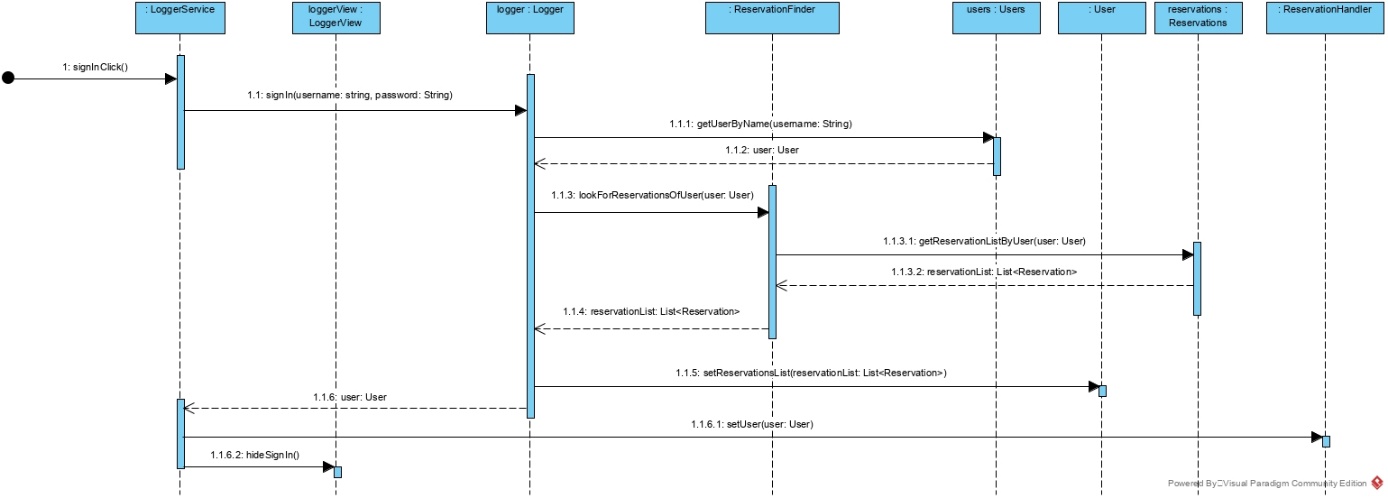
**public** **void** setNumberOfGuests(numberOfGuests)

**public** getCity()

**public** **void** setCity(city)

}

# Diagram Sekwencji dla PU Zaloguj Się



# Funkcje dla klas PU Zaloguj Się

## Klasa Logger:

|  |
| --- |
| public class Logger{ |
|  | public Users users; |
|  | public User signIn(String userName, String password){ |
|  | User user = users.getUserByName(username, password); |
|  | ReservationFinder reservationFinder = new ReservationFinder(); |
|  | ArrayList listOfReservation = reservationFinder.lookForReservationOfUser(user); |
|  |  |
|  | user.setReservationsList(listOfReservation); |
|  | return User; |
|  |  |
|  | } |
|  | public User signUp(String userName, String password, String country, String language) |
|  | } |

## Klasa LoggerService:

|  |
| --- |
| public class LoggerService{ |
|  |  |
|  | private Logger logger; |
|  | private LoggerView loggerView; |
|  | private ReservationHandler reservationHandler; |
|  | public SignInClick(){ |
|  | String username = loggerView.getUsername(); |
|  | String password = loggerView.getPassword(); |
|  | User user = logger.SignIn(username, password); |
|  | loggerView.hideSignIn(); |
|  | reservationHandler.setUser(user); |
|  | } |
|  | public SignUpClick() |
|  | } |

## Klasa ReservationFinder:

|  |
| --- |
| public class ReservationFinder{ |
|  |  |
|  | private Reservations reservations; |
|  | public ArrayList<Reservation> LookForResservations() |
|  | public ArrayList<Reservation> LookForReservationOfUser(User user){ |
|  | ArrayList reservationsList = reservations.getReservationListByUser(user); |
|  | return reservationsList; |
|  | } |
|  | public cancelReservarion() |
|  | public operation() |
|  | } |

## Klasa Reservations:

|  |
| --- |
| public class Reservations{ |
|  |  |
|  | private ArrayList<Reservation> reservationList; |
|  | getReservationListByUser(User user){ |
|  | ArrayList reservationsOfUser = ArrayList<Reservation>(); |
|  | Iterator<Reservation> iterator = reservationList.iterator(); |
|  | while (iterator.hasNext()) { |
|  | Reservation reservation = iterator.next(); |
|  | if (reservation.getUserName().equals(user.getName)) { |
|  | reservationsOfUser.add(reservation); |
|  | } |
|  | } |
|  | return reservationsOfUser; |
|  | } |
|  | } |