System hotelu

Dziedzina problemowa: Projektowany system jest przykładowym systemem dla zarządzania niewielkim hotelem.

Cel: System ma ułatwić dostęp do danych dotyczących m.in. klientów, pracowników, rezerwowanych pokojach, czy też do informacji o grafiku sprzątania pokoju.

Zakres odpowiedzialności systemu: System powinien umożliwiać zarządzanie informacjami dotyczącymi:

- osób pracujących w hotelu(przede wszystkim pracowników i menedżerów), klientów;
- rezerwacji;
- pokojach.

Oprócz tego, system ma umożliwiać dostęp do danych na temat sprzątania pokoi.

Użytkownicy systemu:
Potencjalnymi użytkownikami systemu będą pracownicy hotelu (menedżer albo pracownik), użytkownicy niezarejestrowani (klienci) oraz podsystem czasu.

Wymagania użytkownika:

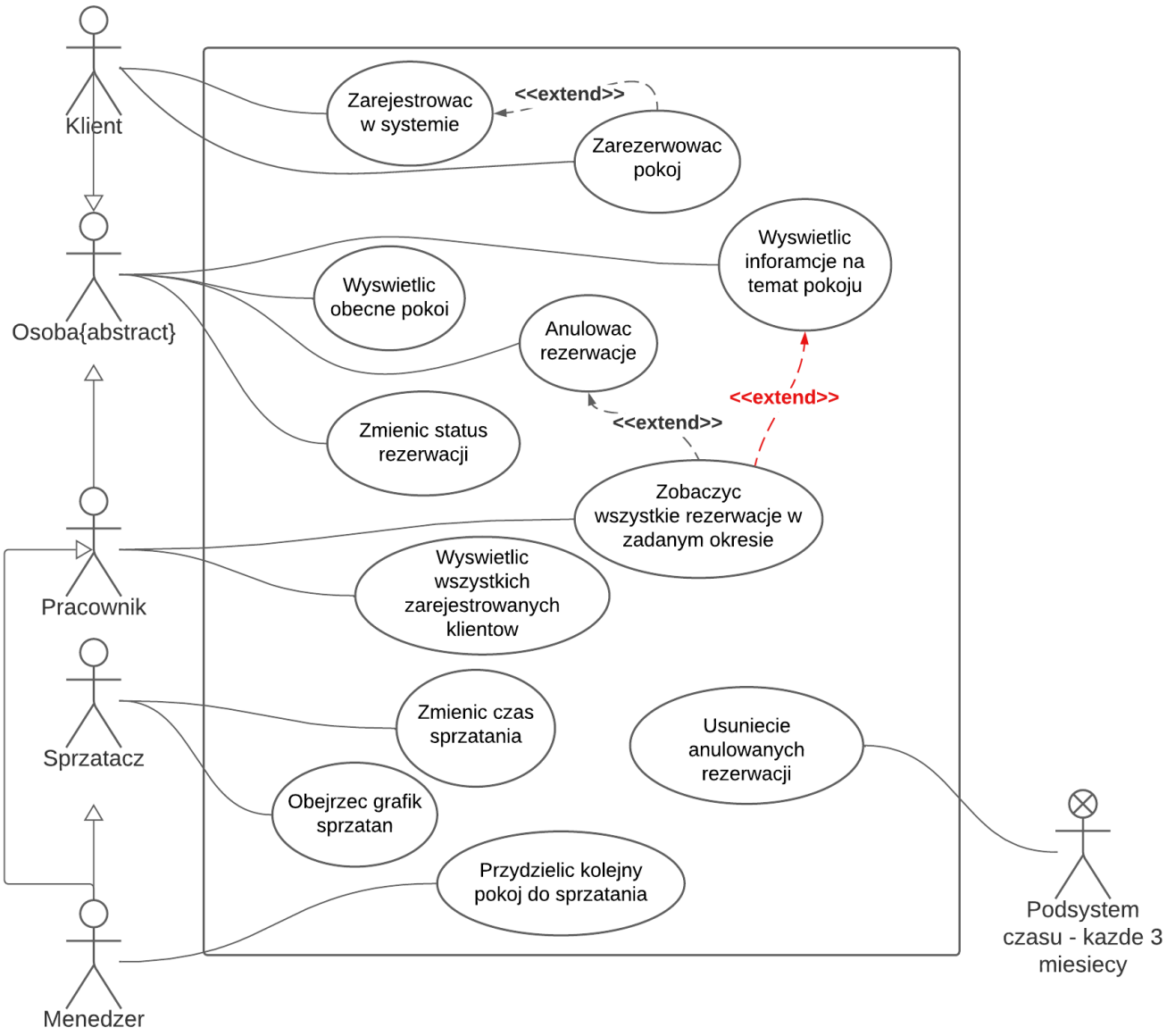
System musi przechowywać informację odnośnie hotelu. Musi tam się znajdować informacja dotycząca nazwy, adresu(Kraj, Miasto, Ulica, nr. Budynku), liczbie pokoi, liczba gwiazdek. W hotelu znajdują się co najmniej 1 pokój, są różnych typów m.in.: prezydencki, zwykły. Te pokoi muszą mieć: unikatowe numery, ceny, powierzchnię, liczbę pojemności ludzi. Prezydencki pokój mieści w sobie sejf, powinniśmy pamiętać hasło od niego. W zwykłych pokojach są różne opcję, takie jak: czy są klimatyzowane, czy posiadają TV, natomiast w prezydenckim te opcję są włączeni. Przed rezerwacją, klient może wybrać sobie opcję, które będą mu pasować. Obowiązkowo powinniśmy pamiętać dane każdego klienta: imię, nazwisko, PESEL. Każdy klient może zarezerwować maksymalnie 3 pokoi jednocześnie. Oczywiście dla rezerwacji mamy pamiętać datę przyjazdu, datę odjazdu, pokoi, które zarezerwował, kwotę, a także status. Statusy są podzieleni na 3 kategorie: rezerwacją złożona(powinniśmy pamiętać wysokość zaliczki oraz datę złożenia rezerwacji), rezerwacją

potwierdzona(musi mieścić w sobie datę opłaty) i w końcu anulowana rezerwacja (tylko datę anulowania). Pracownicy powinni mieć: imię nazwisko, pesel, datę zatrudnienia, pensję. Pracownicy hotelu, a dokładnie sprzątaczki, mają swój grafik sprzątania pokoi, maksymalnie może mieć 5 pokoi do sprzątania w jednym dniu. Powinniśmy mieć informację odnośnie czasu początku sprzątania oraz czasu skończenia sprzątania, a także datę.

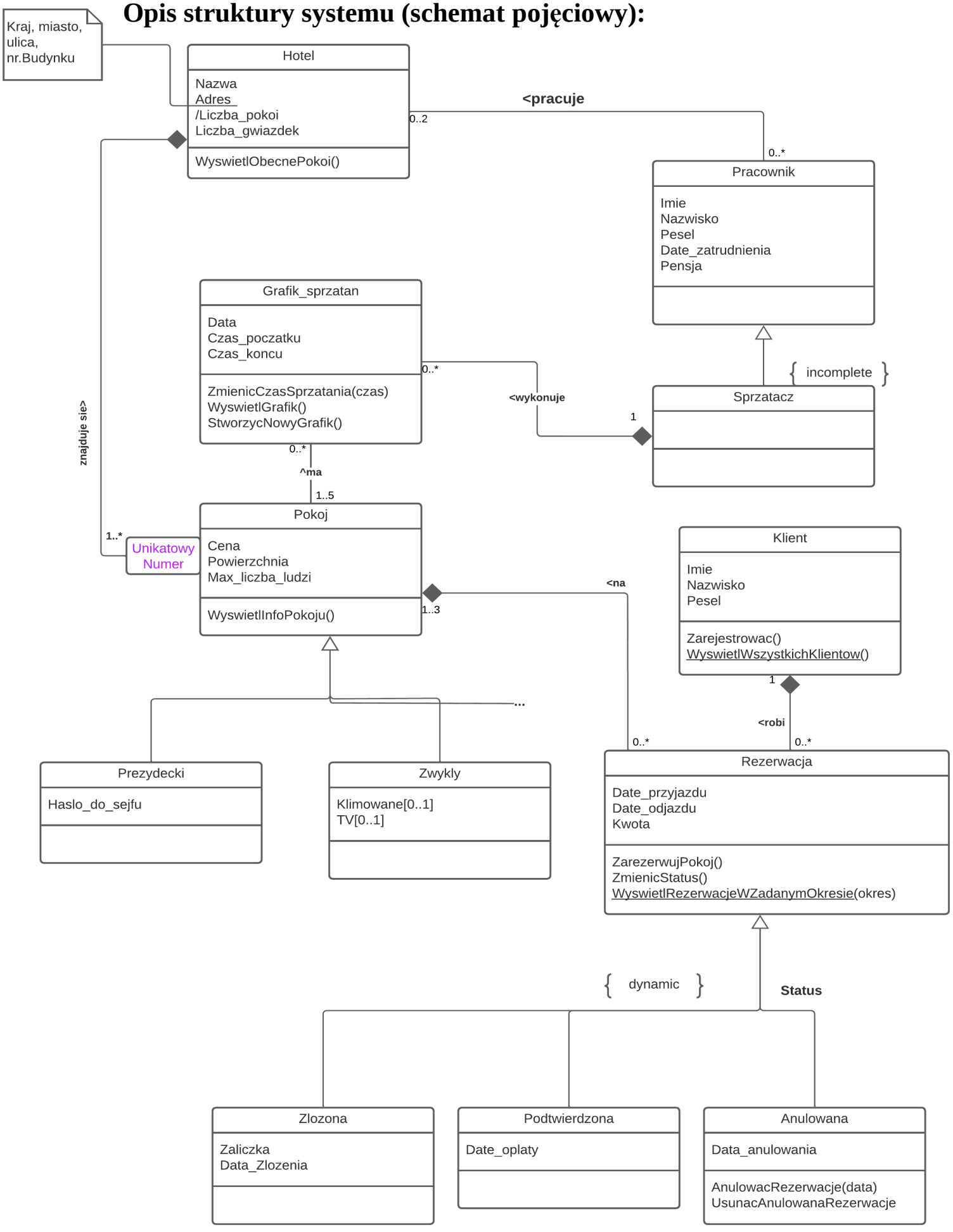
Oczekuję, że system będzie wspomagał pracownikom hotelu oraz klientom w łatwej obsłudze systemu, rezerwacji pokoi. Każdy klient może zarejestrować w systemie z możliwością zarezerwować dowolny pokój, jaki tylko życzy. Klient może wyświetlić obecne pokoi, spojrzeć jaki rodzaj ma ten pokoi, ile powierzchnię, oczywiście cenę (jak i pracownik). Po rezerwacji klient albo pracownik może zmienić status rezerwacji, anulować rezerwację. Pracownik hotelu może wyświetlić; wszystkie rezerwację w zadanym okresie z ewentualnym anulowaniem rezerwacji, wszystkich zarejestrowanych klientów. Sprzątacz może zmienić czas sprzątania albo obejrzeć grafik sprzątania. Tylko menedżer może przypisać sprzątaczu pokój do sprzątania. Co 3 miesięcy system musi zadbać o usunięcie anulowanych rezerwacji. Menedżer ma dostęp do każdego rekordu w systemie oraz możliwość ich edycji i usunięcia.

System powinien być dostępny dla dowolnego użytkownika w dowolnym czasie. Oczywiście bazę danych, w której będą przechowywane dane także kopię zapasową na wypadek awarii. Oraz średnie poziom zabezpieczenia.

Wymagania funkcjonalne:



Opis struktury systemu (schemat pojęciowy):



Dostępność - czas, w jakim system powinien być dostępny dla użytkowników: 24 godziny przez 7 dni w tygodniu przez 365 dni w roku (dostępność w trybie 24/7/365);

Bazę danych - serwerem dla naszego systemu będzie Microsoft SQL Server 2000 lub nowszy;

Bezpieczeństwo - używamy narzędzie od Cloudflare , dla zabezpieczenia od DDOS atak.

Opis przyszłej ewolucji systemu:

Tak, w przyszłości można zrobić możliwość potwierdzenia rezerwacji przez pracownika oraz dodać możliwość wjazdu do pokoju bez rezerwacji, a bezpośrednio przez receptorom.

Przypadek użycia:

Scenariusz dotyczący przypadku użycia "*Wyświetlić rezerwacje w zadanym okresie*":

Warunki wstępne: w systemie są co najmniej 2 użytkownika (klient, menedżer) i choć 1 rezerwacja.

Warunki końcowe: przypadek kończy się po potwierdzeniu użytkownika ze skończył działanie z p.u. albo przejście na inny p.u..

1. Menedżer uruchamia p.u. "Wyświetlić rezerwacje w zadanym okresie"
2. System prosi wpisać okres
3. Użytkownik wpisuje datę od i datę do
4. System wyświetla listę rezerwacji
5. Użytkownik kończy p.u. "Wyświetlić rezerwacje w zadanym okresie"

Alternatywny przepływ zdarzeń:

3a. Użytkownik wprowadził niewłaściwą datę

4a. Nie ma rezerwacji w zadanym okresie

4b. Użytkownik chce zmienić okres

5a. Użytkownik wybiera z listy rezerwacje i uruchamia p.u. „Wyświetlić informacje na temat pokoju"

Analiza dynamiczna:

Diagram aktywności:

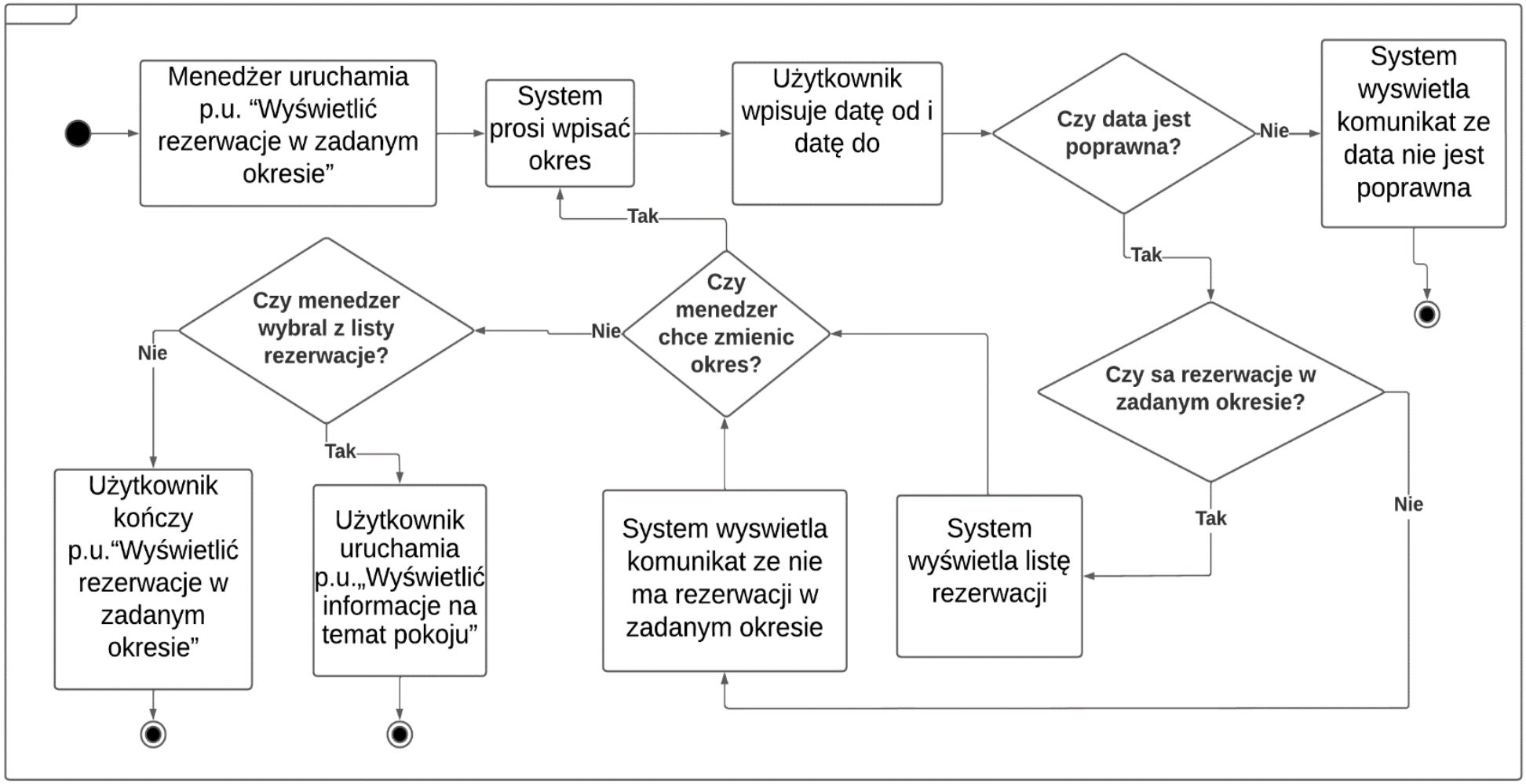


Diagram stanów:

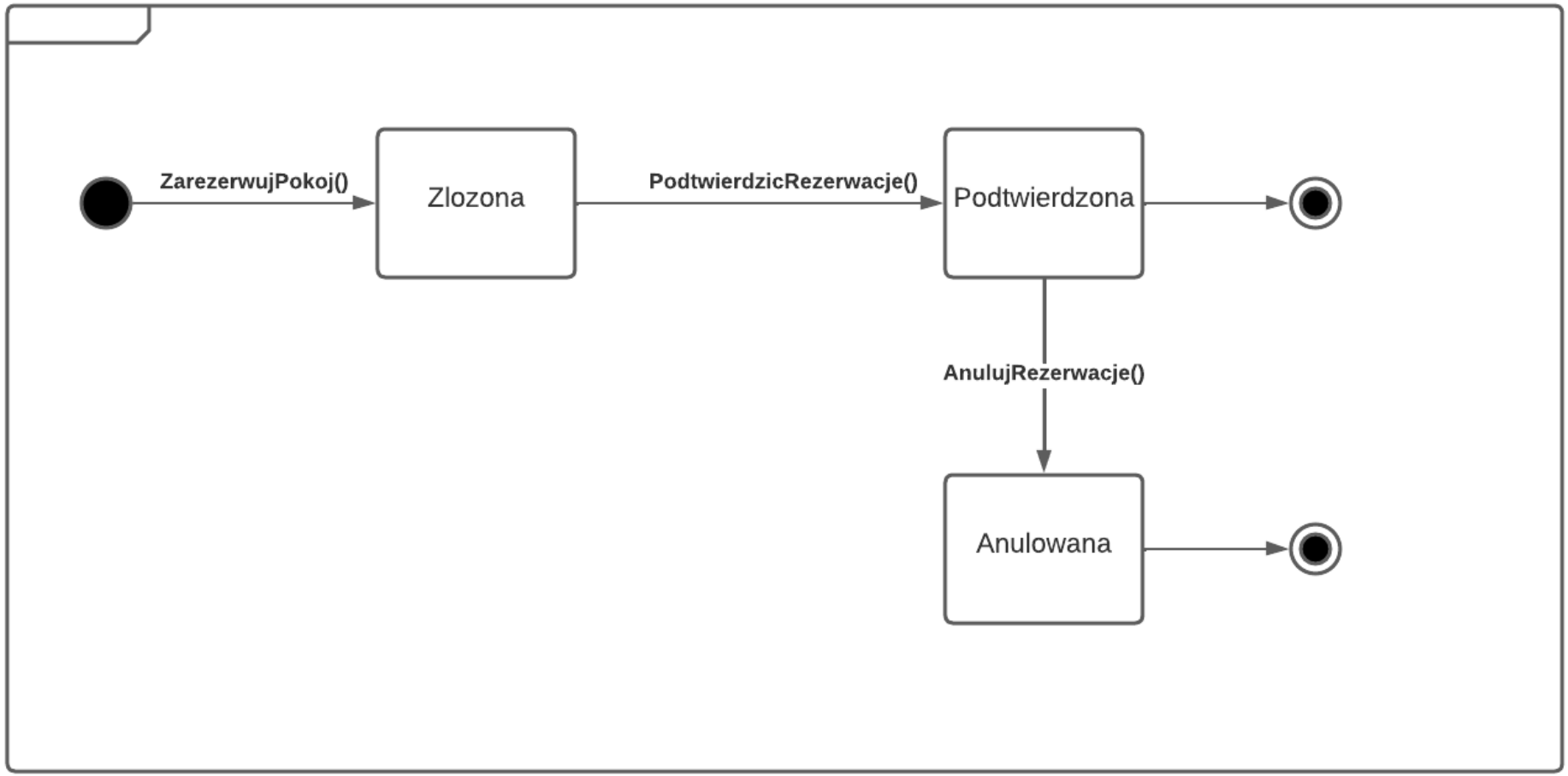
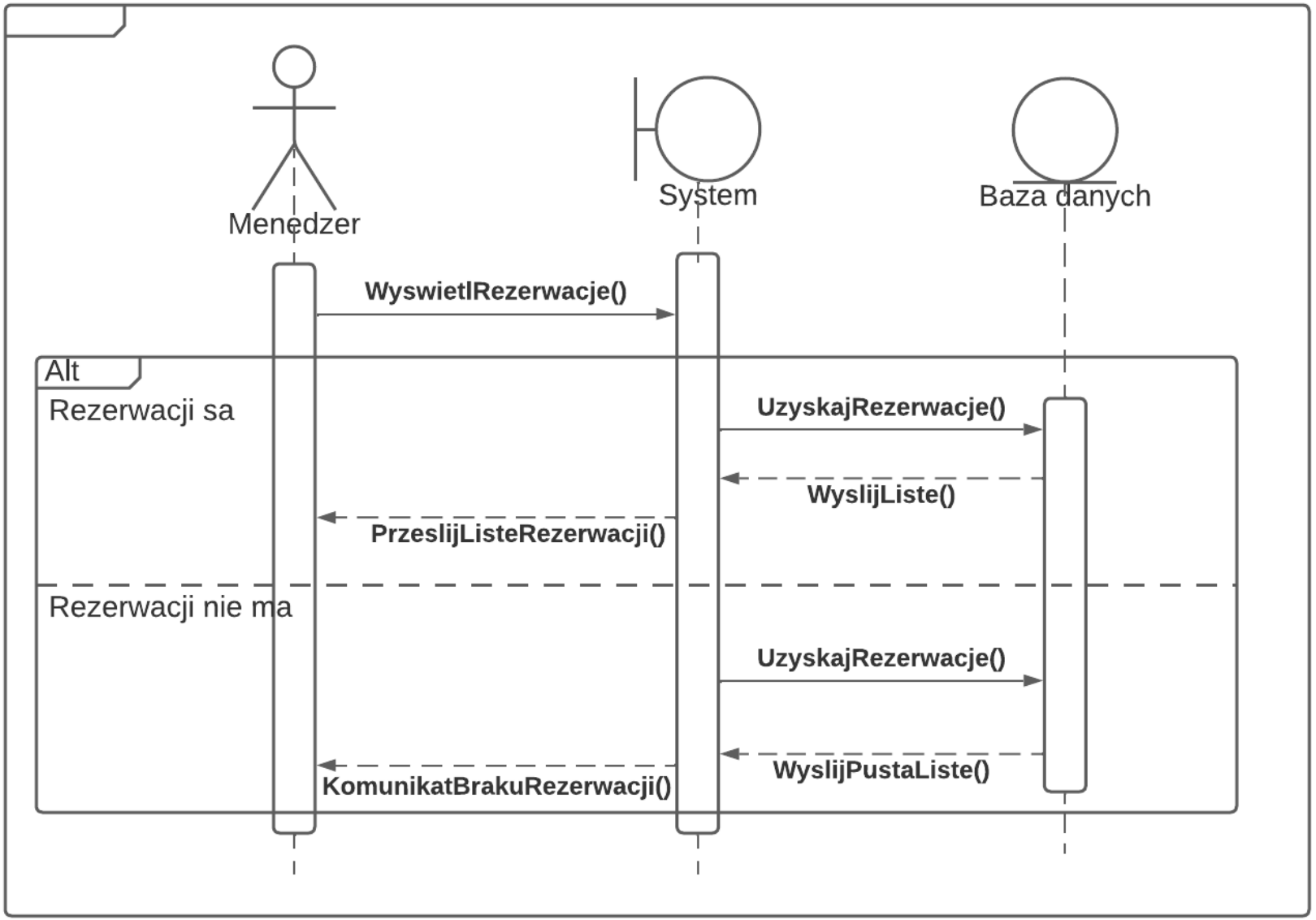
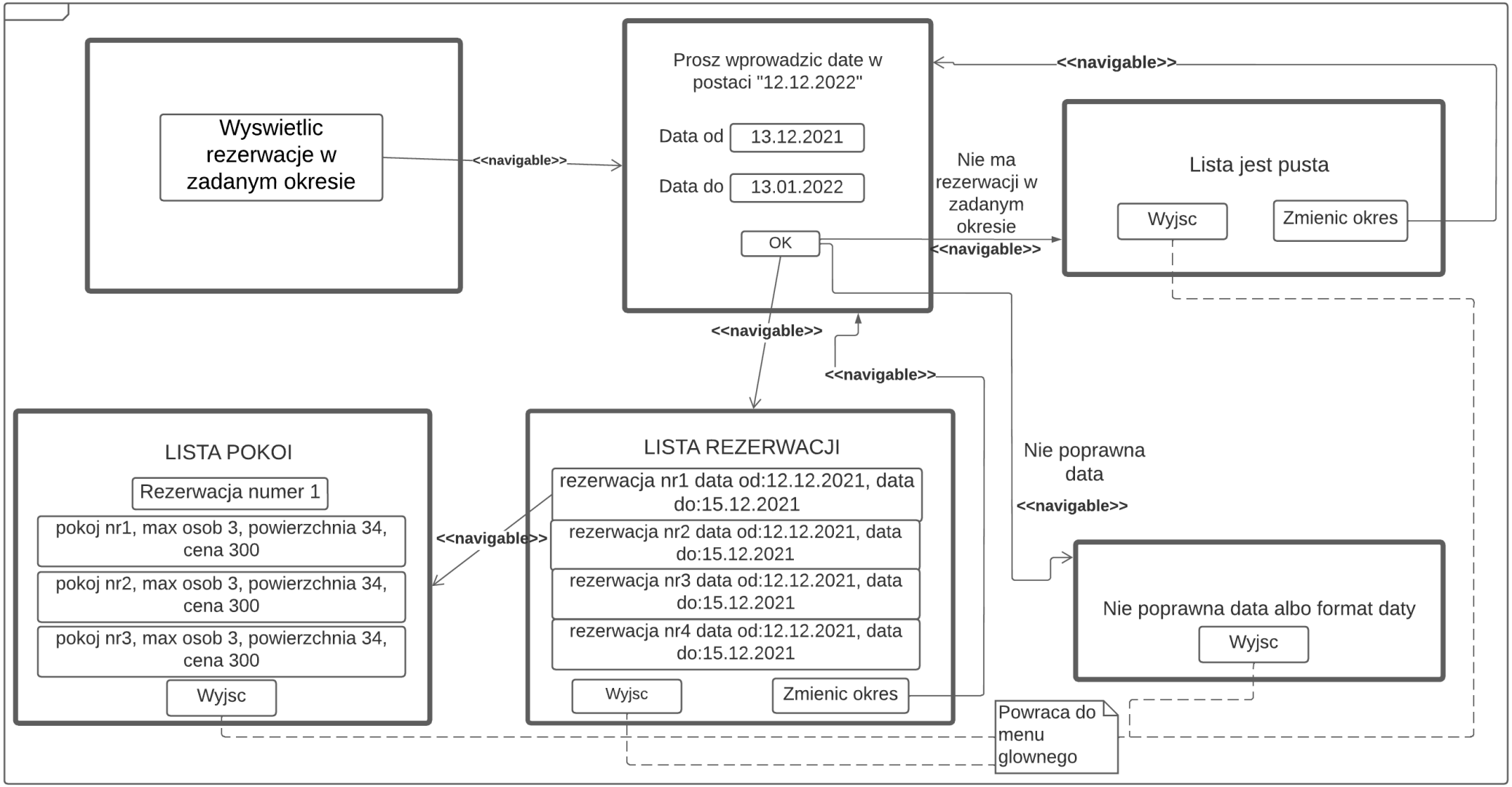


Diagram sekwencji:



Projekt GUI:


Wyswietlic rezerwacje w zadanym okresie
<<navigable>>
Prosz wprowadzic date w postaci "12.12.2022"
Data od
13.12.2021
Data do
13.01.2022
OK
<<navigable>>
Nie ma rezerwacji w zadanym okresie
<<navigable>>
Lista jest pusta
Wyjsc
Zmienic okres
<<navigable>>
<<navigable>>
<<navigable>>
LISTA POKOI
Rezerwacja numer 1
pokoj nr1, max osob 3, powierzchnia 34, cena 300
pokoj nr2, max osob 3, powierzchnia 34, cena 300
pokoj nr3, max osob 3, powierzchnia 34, cena 300
Wyjsc
<<navigable>>
LISTA REZERWACJI
rezerwacja nr1 data od:12.12.2021, data do:15.12.2021
rezerwacja nr2 data od:12.12.2021, data do:15.12.2021
rezerwacja nr3 data od:12.12.2021, data do:15.12.2021
rezerwacja nr4 data od:12.12.2021, data do:15.12.2021
Wyjsc
Zmienic okres
Nie poprawna data
<<navigable>>
Nie poprawna data albo format daty
Wyjsc
Powraca do menu glownego

Podstawę do implementacje:

Dziedziczenie: dodawanie po nazwie klasy „extend Osoba”, gdzie „Osoba” – nazwa klasy, po której dziedziczy

(np. class Pracownik extend Osoba)

Dziedziczenie dynamiczne: wykorzystujemy metody „Sprytnego” kopiowanie obiektów.

Asocjacja zwykła: tworzenie w klasie „List<Osoba>” z typem klasy z kim ma asocjacje wiele albo tworzenie obiektu „Osoba”, jeżeli asocjacja jeden

Asocjacja kwalifikowana: wykorzystać TreeMap<Klucz, Obiekt> (), gdzie kluczem jest atrybut unikalny

(np. TreeMap<String, Pokoj>(), kluczem jest numer unikatowy pokoju)

Asocjacja kompozycja: dodajemy do konstruktora obowiązkowe przekazanie Obiektu całość

(np. Pokoj(Hotel hotel), pokój jest częścią oraz hotel jest całość)

Asocjacja z atrybutem: zastępujemy atrybut asocjacji, klasą pośredniczącą, otrzymujemy dwie „zwykłe” asocjacje.