# Laboratorium 5

# Diagram sekwencji

## Cel laboratorium

Budowa diagramów sekwencji reprezentujących kolejność w czasie wysyłania komunikatów pomiędzy różnymi obiektami w systemie. W oparciu o zdefiniowane elementy z poprzednich laboratorium należy zaproponować diagramy sekwencji dla projektowanego systemu.

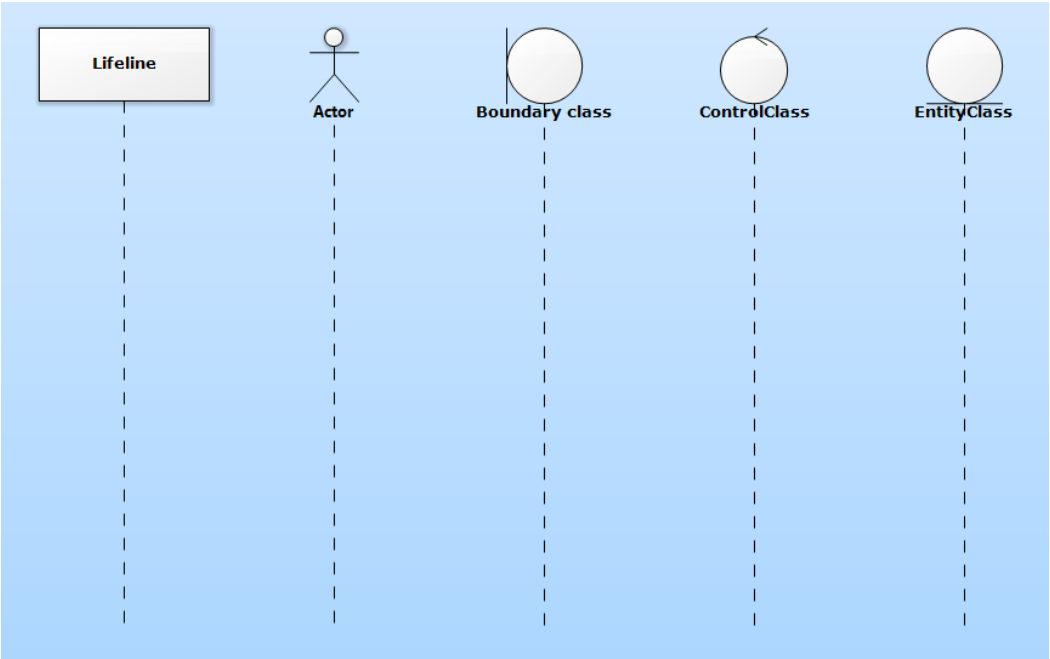
**Diagram sekwencji** - opisuje interakcje pomiędzy częściami systemu w postaci sekwencji komunikatów wymienianych między nimi. Służy do modelowania przepływu wiadomości między obiektami w określonej kolejności czasowej, przedstawiając, jak elementy systemu współdziałają ze sobą w czasie.

Zastosowanie diagramu sekwencji

- Modelowanie scenariuszy użytkowania - pomaga zrozumieć, jak system powinien działać z perspektywy użytkownika.
- Analiza wymagań - pokazuje, w jaki sposób komponenty systemu komunikują się podczas wykonywania konkretnego zadania.
- Projektowanie systemów - pomaga projektantom w definiowaniu kolejności operacji i zależności między komponentami.
- Testowanie - wspiera tworzenie przypadków testowych poprzez wizualizację interakcji systemowych.

## Notacja

- **Linia Życia** to rola uczestnika interakcji, jaką pełni w czasie jej trwania. Linia Życia reprezentuje współuczestnika interakcji i czas jego istnienia podczas realizacji scenariusza. Linie Życia reprezentują konkretne byty – obiekty, systemy i mogą przyjmować stereotypy, które świadczą o roli, jaką pełni dany obiekt w systemie.




Takimi stereotypami mogą być:

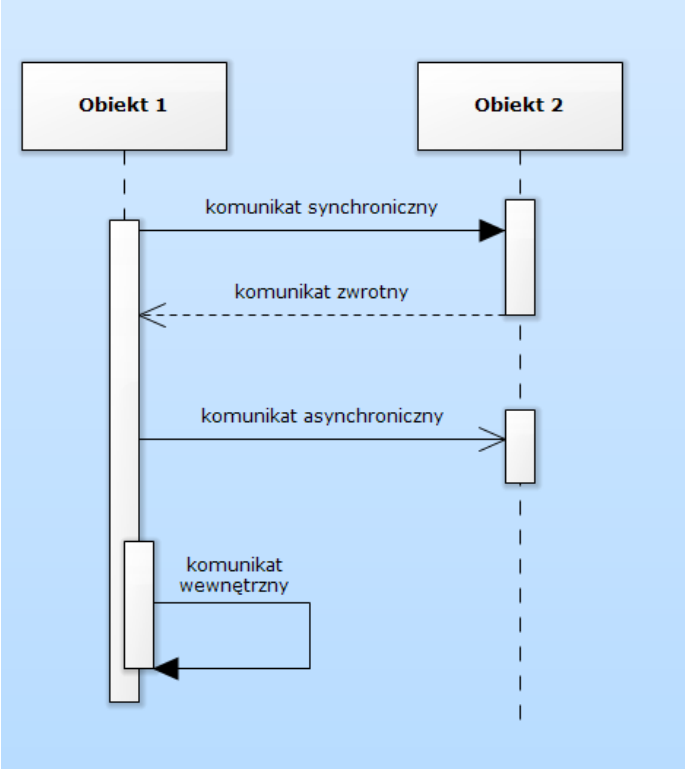
- aktor (ang. actor),

- obiekt klasy granicznej (ang. boundary class),
- obiekt klasy sterującej (ang. control class),
- obiekt klasy danych (ang. entity class).

Wymienione rodzaje stereotypów obiektów służą do wskazania, jaką rolę w systemie pełni obiekt. Stereotyp aktora informuje, że obiekt ten pełni funkcję zewnętrzną w stosunku do systemu. Obiekt klasy granicznej reprezentuje interfejs między systemem a bytami występującymi poza nim. Obiekt klasy sterującej używany jest do sterowania działaniem jednej lub wielu klas.

- **Komunikat (ang. message)** jest to informacja przesyłana pomiędzy obiektami. W języku UML można korzystać z różnych typów komunikatów. **Komunikat synchroniczny** oznacza, ż obiekt musi czekać na odpowiedź. **Komunikat asynchroniczny** nie wymaga oczekiwania na odpowiedź.

Stosując na diagramie komunikaty synchroniczne, przyjmuje się, że natychmiast po jego wywołaniu przychodzi odpowiedź. Taka konwencja pozwala na zmniejszenie liczby elementów na diagramie. W sytuacji, gdy modelowany fragment rzeczywistości jest inny niż założenia konwencji, należy umieścić komunikat zwrotny.

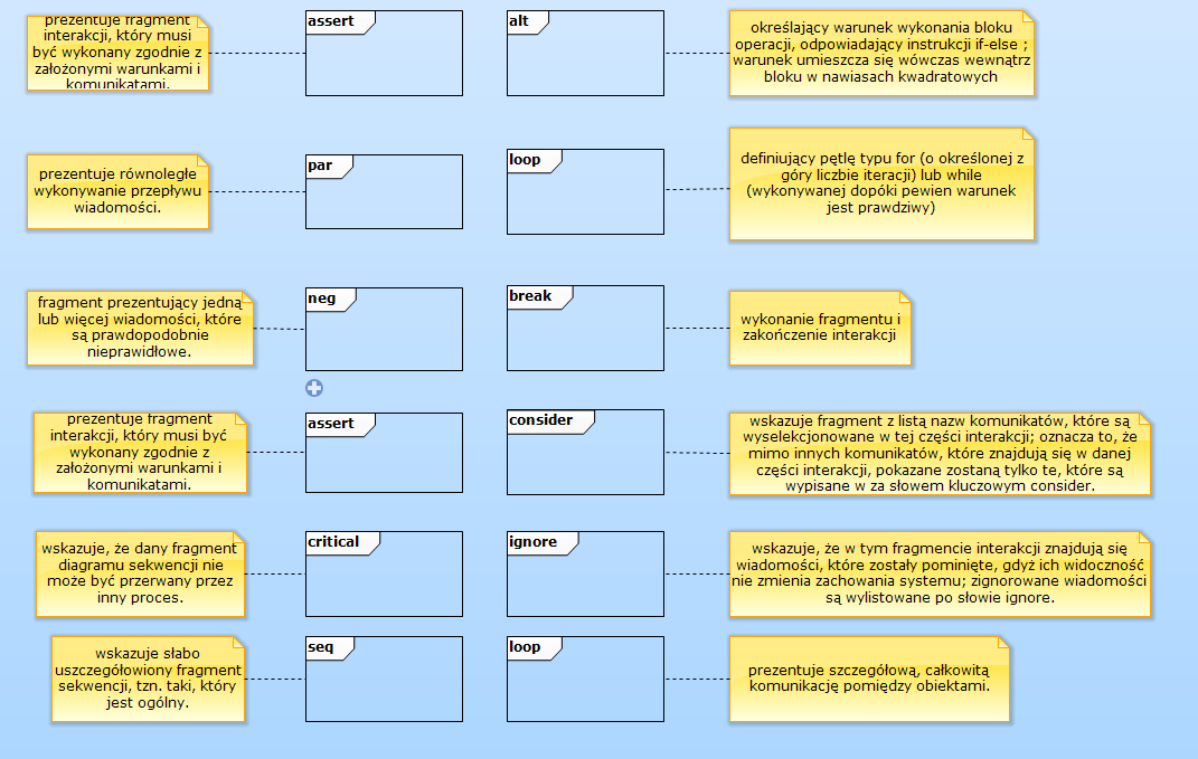



- **Fragment (ang. combined fragment)** to konceptualnie zamknięta część diagramu sekwencji, która rozszerza możliwości obejmowanego przez siebie obszaru diagramu sekwencji. Fragment może zawierać w sobie pętle, powtórzenia, scenariusze alternatywne lub wskazywać poziom abstrakcji modelowanego fragmentu systemu.

Rodzaj fragmentu jest określany poprzez umieszczenie odpowiedniego słowa kluczowego w lewym górnym rogu. Poniżej opis wszystkich słów kluczowych, które mogą wystąpić we fragmentach.

- **alt** – dzieli fragment interakcji zgodnie z warunkami logiki Boole'a na dwa alternatywne scenariusze; każda z alternatyw musi być opatrzona warunkiem dozoru, którego spełnienie gwarantuje wykonanie danej alternatywy.
- **assert** – prezentuje fragment interakcji, który musi być wykonany zgodnie z założonymi warunkami i komunikatami.

- **break –** wskazuje fragment diagramu sekwencji, który realizowany jest po spełnieniu warunku dozoru; spełnienie warunku dozoru skutkuje wykonaniem sekwencji komunikatów zawartych we fragmencie, a następnie wyjście ze scenariusza; w przypadku, gdy warunek dozoru nie jest spełniony, komunikaty zawarte we fragmencie są pomijane.
- **consider** – wskazuje fragment z listą nazw komunikatów, które są wyselekcjonowane w tej części interakcji; oznacza to, że mimo innych komunikatów, które znajdują się w danej części interakcji, pokazane zostaną tylko te, które są wypisane w za słowem kluczowym consider.
- **critical** – wskazuje, że dany fragment diagramu sekwencji nie może być przerwany przez inny proces.
- **ignore** – wskazuje, że w tym fragmencie interakcji znajdują się wiadomości, które zostały pominięte, gdyż ich widoczność nie zmienia zachowania systemu; zignorowane wiadomości są wylistowane po słowie ignore.
- **loop –** powtórzenie fragmentu interakcji określoną warunkiem liczbę razy.
- **neg –** fragment prezentujący jedną lub więcej wiadomości, które są prawdopodobnie nieprawidłowe.
- **opt –** wskazuje opcjonalny fragment interakcji, który jest wykonywany po spełnieniu warunku dozoru.
- **par –** prezentuje równoległe wykonywanie przepływu wiadomości.
- **seq** – wskazuje słabo uszczegółowiony fragment sekwencji, tzn. taki, który jest ogólny.
- **strict** – prezentuje szczegółową, całkowitą komunikację pomiędzy obiektami.




- **Wystąpienie interakcji (ang. interaction occurence)** jest odniesieniem (referencją) do interakcji, której obraz przedstawiony jest na innym diagramie. Użycie referencji pomaga uczynić diagram czytelniejszym, gdyż pozwala na ukrycie szczegółów nieistotnych z punktu widzenia modelowanej sytuacji.
- **Punkt końcowy (ang. end point**) i punkt startowy (ang. found point) to elementy prezentujące byty spoza diagramu sekwencji, który jest nieznanym źródłem wiadomości lub nieznanym celem wiadomości. Punkt końcowy wskazuje na nieznany byt poza diagramem sekwencji. Natomiast punkt startowy umożliwia dostarczenie zewnętrznego komunikatu bez przedstawiania bytu.

- **Brama (ang. gate)** to element stanowiący łącznik pomiędzy diagramami sekwencji lub fragmentami diagramów sekwencji. Brama jest elementem granicznym, który łączy interakcję z jego uszczegóławiającym fragmentem. Brama jest specjalizowaną formą punktu końcowego. Element brama jest używany jako punkt graniczny pomiędzy dwoma diagramami sekwencji bądź między diagramem sekwencji a fragmentem go uszczegóławiającym.

Narzędzia do tworzenia diagramów sekwencji:

- Software Ideas Modeler
- Sparx Enterprise Architect (Enterprise Architect)
- PlantUML (tekstowy sposób definiowania diagramów).
- Lucidchart (wizualny edytor).
- Draw.io (łatwe tworzenie diagramów).
- Visual Paradigm (profesjonalne narzędzie do UML).
- StarUML (narzędzie dedykowane do UML).

Diagramy sekwencji są potężnym narzędziem do modelowania dynamicznych aspektów systemu. Pozwalają na szczegółowe przedstawienie interakcji między obiektami w czasie, co jest kluczowe dla zrozumienia i projektowania złożonych systemów informatycznych. Ich elastyczność w reprezentowaniu różnych scenariuszy, w tym alternatywnych ścieżek i powtórzeń, czyni je nieocenionym narzędziem w procesie analizy i projektowania oprogramowania.

**Przykład 1.**

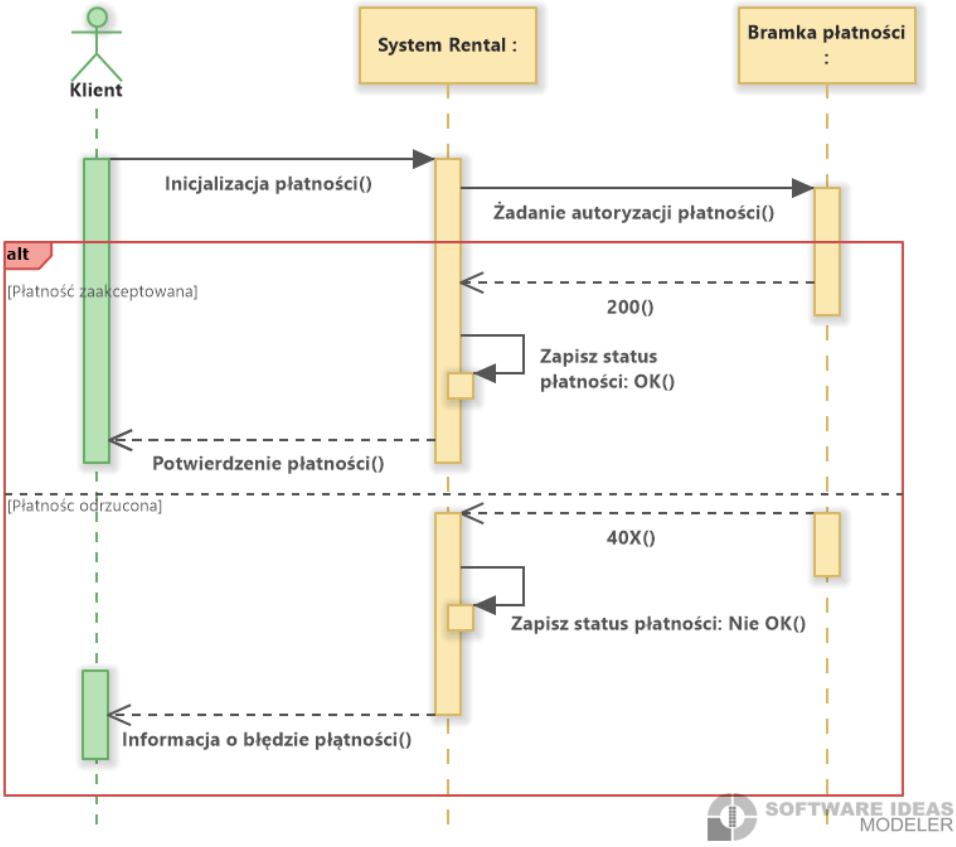
Na Rys. 1 przedstawiono diagram pokazujący asynchroniczną komunikację między systemami, co jest widoczne przez przerywane linie strzałek powrotnych. Prezentuje on również alternatywne ścieżki wykonania (zaakceptowana vs odrzucona płatność) za pomocą fragmentu oznaczonego jako „alt”. Warto zauważyć, że diagram ten przedstawia uproszczony proces płatności, skupiając się na kluczowych interakcjach między uczestnikami systemu.

Główne elementy diagramu to:

- Klient – inicjator procesu płatności
- System Rental – system obsługujący wynajem
- Bramka płatności – zewnętrzny system autoryzacji płatności

Przebieg interakcji:

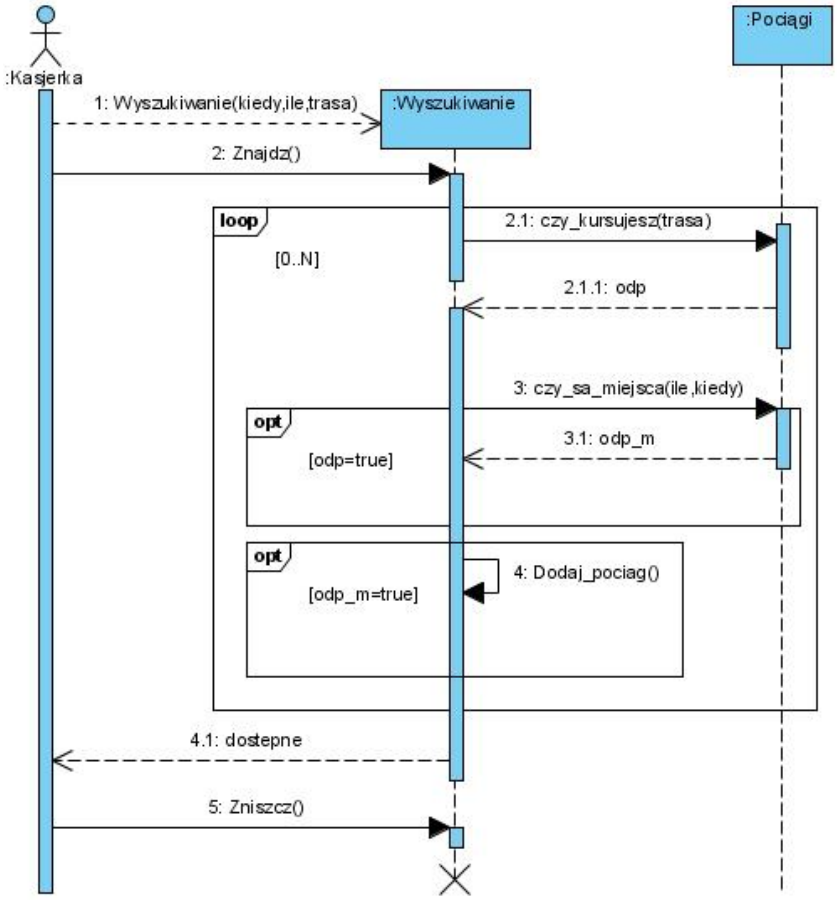
- Klient inicjuje płatność.
- System Rental wysyła żądanie autoryzacji płatności do bramki płatności.
- Bramka płatności przetwarza żądanie i zwraca odpowiedź. Diagram pokazuje dwa możliwe scenariusze:
    a) Płatność zaakceptowana:
        - Bramka zwraca kod 200 (sukces)
        - System Rental zapisuje status płatności jako OK
        - System potwierdza płatność klientowi
    b) Płatność odrzucona:
        - Bramka zwraca kod 40X (błąd)
        - System Rental zapisuje status płatności jako Nie OK
        - System informuje klienta o błędzie płatności




*Rys. 1 Diagram sekwencji - przykład*

**Przykład 2.**

Obiekt Kasjerka szuka, czy na danej trasie w danym terminie jest dostępna określona ilość biletów. W tym celu tworzy obiekt Wyszukiwanie i zadaje mu odpowiednie pytanie. Obiekt ten pyta się wszystkie N pociągów czy jeżdżą daną trasą. Jeśli pociąg jeździ tą trasą, jest pytany o dostępność biletów. Jeśli bilety są dostępne, Wyszukiwanie dodaje pociąg do swojej wewnętrznej listy. Po przeszukaniu listy pociągów Wyszukiwanie zwraca Kasjerce informację o pociągach, po czym Kasjerka pozbywa się obiektu Wyszukiwanie.




***Rys. 2 Diagram sekwencji – przykład***

**Przykład 3.**

Poniżej przedstawiono scenariusz przypadku użycia Systemu rezerwacji hotelowej, który zawiera następujące kroki:

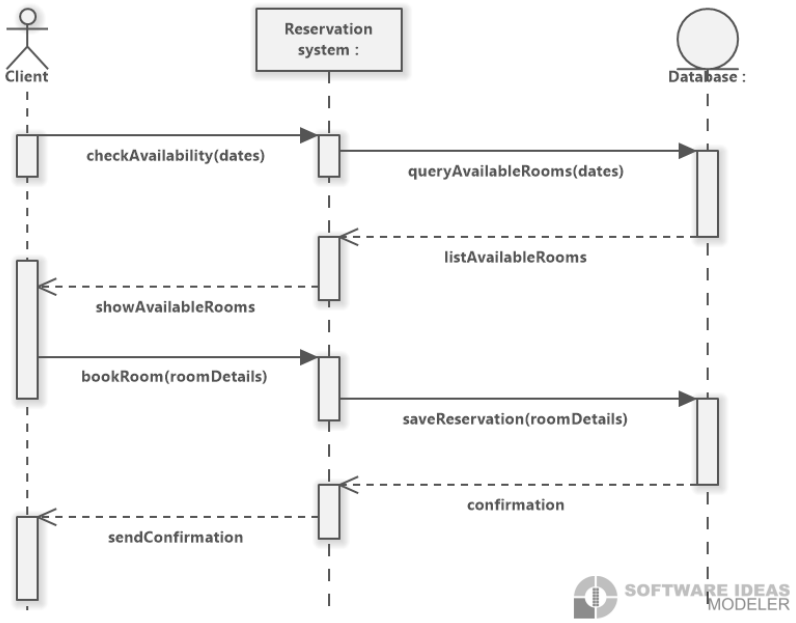
- Klient wybiera daty pobytu i sprawdza dostępność pokoi.
- System wysyła zapytanie do bazy danych o dostępne pokoje.
- Baza danych zwraca listę dostępnych pokoi.
- Klient wybiera pokój i dokonuje rezerwacji.
- System potwierdza rezerwację i zapisuje szczegóły w bazie danych.
- Klient otrzymuje potwierdzenie rezerwacji.

Na xx przedstawiono diagram przedstawia przepływ wiadomości pomiędzy obiektami: Klient, System Rezerwacji i Baza Danych. Diagram ten pokazuje proces interakcji podczas dokonywania rezerwacji. W diagramie odpowiednio:

- Client (Aktor): Reprezentuje użytkownika korzystającego z systemu.
- Reservation System (System): Główny komponent odpowiadający za logikę biznesową.
- Database (Baza Danych): Odpowiada za przechowywanie i zwracanie danych o pokojach i rezerwacjach.
- Wiadomości: Strzałki pokazują kierunek przepływu komunikacji:
  - Żądania od klienta do systemu.
  - Zapytania i odpowiedzi pomiędzy systemem a bazą danych.

**Sposób działania diagramu**

- Klient inicjuje proces, wysyłając żądanie sprawdzenia dostępności pokoi.
- System komunikuje się z bazą danych, aby pobrać listę dostępnych pokoi.
- Klient dokonuje wyboru, a system zapisuje rezerwację w bazie danych.
- Na końcu klient otrzymuje potwierdzenie swojej rezerwacji.




*Rys. 3 Diagram sekwencji - przykład*

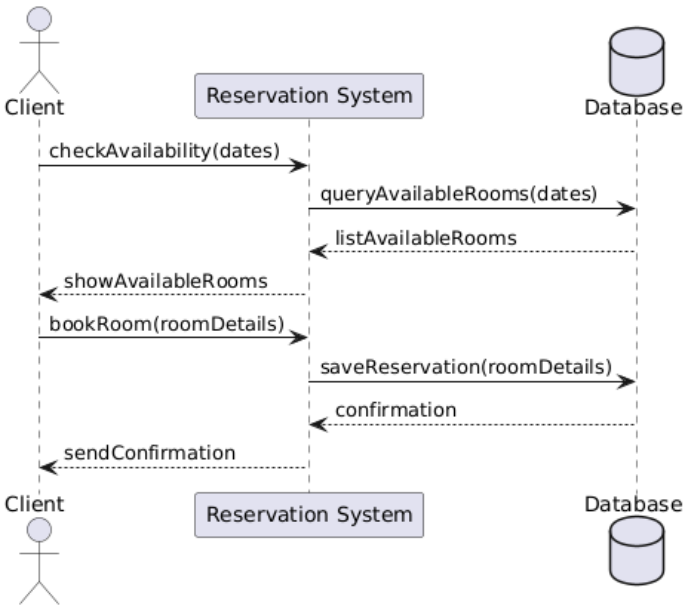
### Diagram UML w PlantUML

W celu generowania diagramów w PlanUML można skorzystać z https://plantuml.com/ poniżej przedstawiono zapis diagramu sekwencji w plantuml, natomiast na  przedstawiono wygenerowany rysunek z kodu.

```
@startuml
actor Client
participant "Reservation System" as System
database "Database" as DB

Client -> System: checkAvailability(dates)
System -> DB: queryAvailableRooms(dates)
DB --> System: listAvailableRooms
System --> Client: showAvailableRooms

Client -> System: bookRoom(roomDetails)
System -> DB: saveReservation(roomDetails)
DB --> System: confirmation
System --> Client: sendConfirmation
@enduml
```




*Rys. 4 Diagram sekwencji - przykład*