Wymagania funkcjonalne

Must have:

- Użytkownik może wyszukiwać sale pod względem:
    - Lokacji
    - Wielkości
    - Wyposażenia
    - Cenie
    - ustawienia sali
- Użytkownik może zarezerwować sale na dany dzień
- Możliwość założenia konta by odblokować benefity takie jak:
    - Historia zamówień zalogowanego użytkownika.
    - łatwe zarządzanie rezerwacją (np. Anulowanie jej, ale nie w późniejszym terminie niż 3 dni przed zarezerwowaną datą)
- Powiadomienie na E-mail o rezerwacji Sali oraz 3 dni przed planowaną rezerwacją wysyła przypomnienie o zbliżającym się terminie

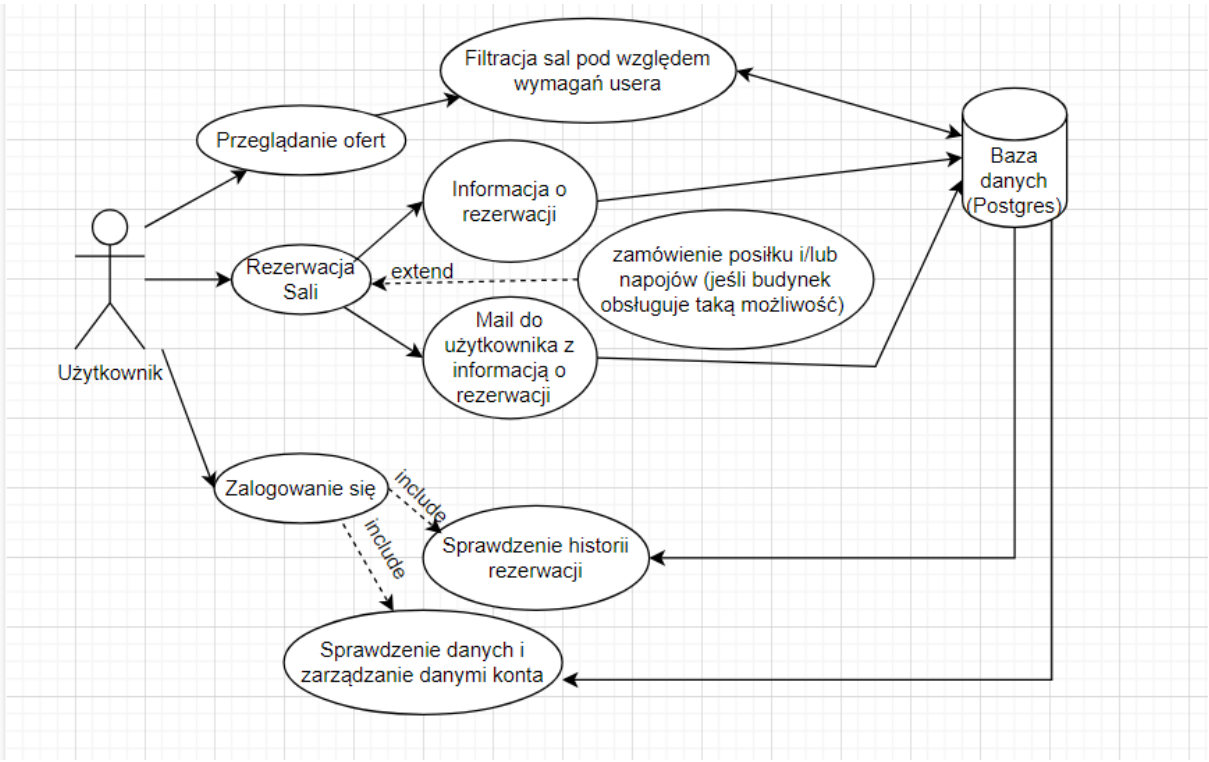
Nice to have:

- Każda sala ma swoją miniaturkę w wyszukiwarce i 2-3 zdjęcia pokazowe pomieszczenia po wyświetleniu opisu sali.
- Użytkownik może zapisywać oferty rezerwacji sal do własnych zakładek i zapisywać je do ulubionych.
- Użytkownik może wstawiać opinie w skali od 0 do 5 gwiazdek i komentarze pod danymi salami.
- .
    - .

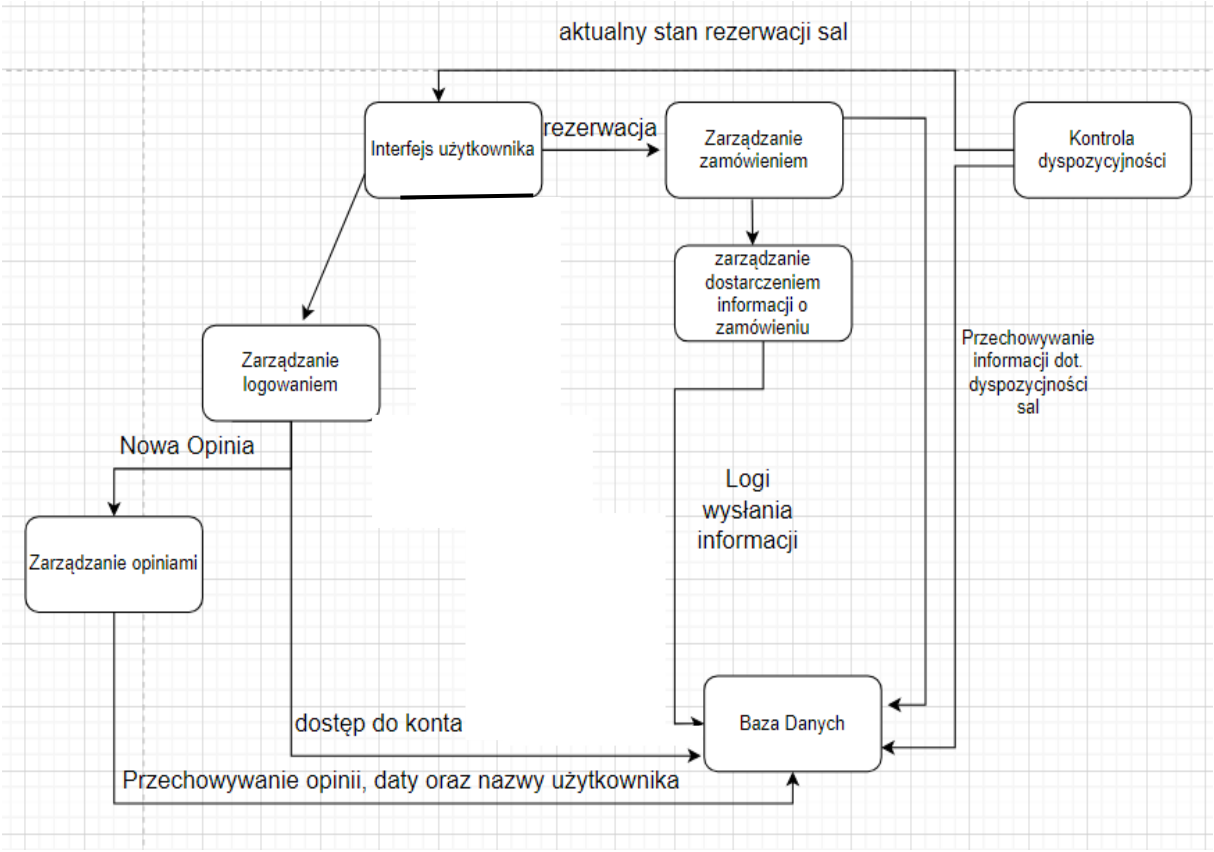
Nie funkcjonalne wymagania

- Wysoka dostępność
- Wysoka niezawodność
- Duża skalowalność
- Bezpieczeństwo

Diagram przypadków użycia:



Projekt wysokiego poziomu:



Interfejs Użytkownika:

- Wyszukiwanie sal
- Rezerwacja sal
- Wstawianie opinii tylko od zalogowanych użytkowników

Zarządzanie zamówieniem:

- Anulowanie zamówienia

Zarządzanie dostarczaniem informacji o zamówieniu:

- Wysyłanie maila z potwierdzeniem
- Wysyłanie informacji do budynku z salą
- Logi do bazy danych

Zarządzanie logowaniem:

- Logowanie się na istniejące konto
- Rejestracja

- przypominanie hasła

Zarządzanie opiniami:

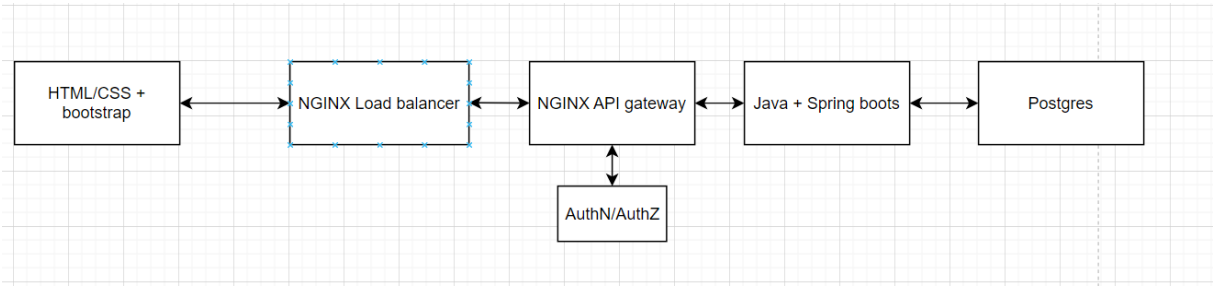
- Filtracja nieodpowiednich treści

- Dodawanie opinii do danych sal oraz budynków

Kontrola dyspozycyjności

- Sprawdzenie czy sala jest dalej dostępna w real time

Projekt niskiego poziomu:



HTML/CSS + bootstrap:

- front-end aplikacji

NGINX Load balancer:

- Rozkłada ruch użytkowników

NGINX API GATEWAY:

- Większe bezpieczeństwo i ochrona back-endu przed nie upoważnionymi osobami.

AuthN/AuthZ:

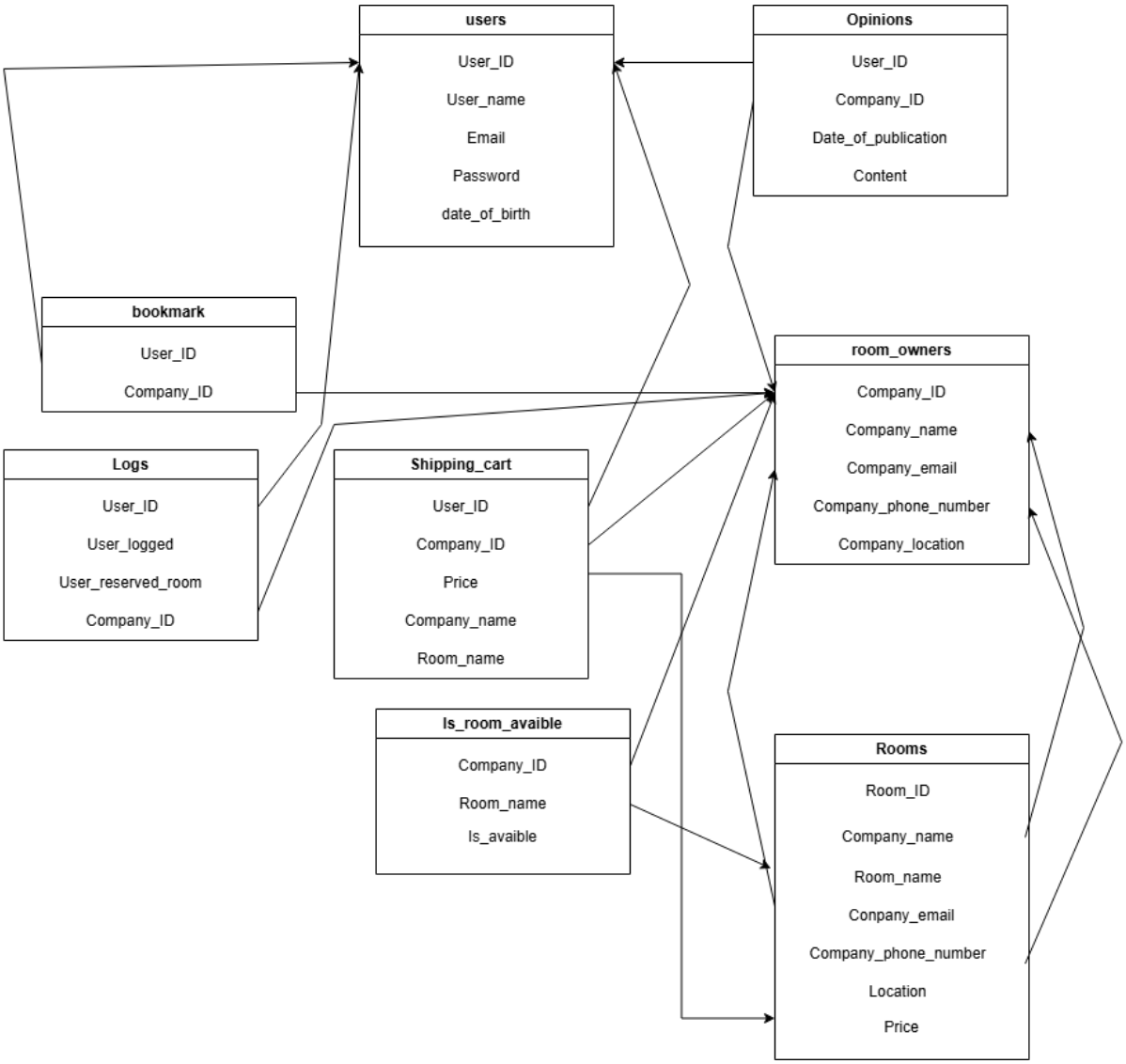
- Nadanie użytkownikowi praw do rezerwacji sali gdy jest poprawnie zalogowany.

Java + Spring boots:

- W javie będzie pisany backend i będzie się posługiwał frameworkiem Spring boots do połączeń z bazą danych, tworzenia rest api i nadawania tokenów JWT zalogowanym użytkownikom oraz adekwatne im prawa (dla zwykłego użytkownika tylko rezerwacja sal)

Postgres:

- Baza danych SQL'owa będzie wykorzystywana do przechowywania danych o użytkownikach, logach, opiniach użytkowników o salach oraz datach ich utworzenia, aktualnym stanie Sali (zarezerwowana, wolna).



Users{

- Tablica która będzie przechowywać info o użytkowniku.

Informacje dotyczące kolumn:

- User_ID(int, auto_increment, primary key)
- User_name(varchar(40),not null)
- Email(varchar(60), not null, unique)
- Password(varchar(60), not null)
- date_of_birth(date)

}