System zarządzania hotelami

Autorzy:

- Jakub Kędra
- Paweł Konopka
- Krzysztof Wysocki

Spis treści

Temat:

System zarządzania hotelami - stwórz bazę danych przechowującą informacje o pokojach, rezerwacjach, klientach i pracownikach. System powinien umożliwiać zarządzanie rezerwacjami, płatnościami oraz przypisywanie zadań dla pracowników.

Linki:

link do repozytorium:
https://github.com/krzyswys/Hotel-managment-database

Nasz projekt opiera się na zastosowaniu różnych technologii, które zapewniają wydajność, funkcjonalność i interaktywność naszej aplikacji. Poniżej przedstawiamy użyte technologie w naszym stosie technologicznym.

Stos technologiczny **front-end**:

- **React**: Zdecydowaliśmy się użyć biblioteki React jako podstawy naszej aplikacji. React jest wydajnym i popularnym narzędziem do budowania interfejsów użytkownika, które umożliwia tworzenie komponentów reużywalnych i zarządzanie stanem aplikacji.
- **React-Cookie:** Wykorzystujemy bibliotekę React-Cookie do obsługi plików cookie w naszej aplikacji. Dzięki temu możemy skutecznie zarządzać danymi użytkownika, takimi jak sesje logowania czy preferencje.
- **React-DOM:** React-DOM jest biblioteką służącą do renderowania komponentów React w przeglądarce internetowej. Dzięki temu możemy skutecznie wyświetlać nasze komponenty i interakcje w przeglądarce.
- **React-Icons**: Używamy biblioteki React-Icons do dostarczania różnorodnych ikon i grafik w naszej aplikacji. Ta biblioteka zapewnia nam prosty sposób dodawania estetycznych i skalowalnych ikon, co przyczynia się do lepszego wyglądu i użyteczności interfejsu użytkownika.
- **React-Modal**: Wykorzystujemy bibliotekę React-Modal do tworzenia interaktywnych modali w naszej aplikacji. Dzięki temu możemy prezentować użytkownikom dodatkowe informacje lub opcje w przyjazny i łatwy sposób.
- **React-Google-Maps-API**: Korzystamy z biblioteki React-Google-Maps-API do integracji z usługą Google Maps.

Stos technologiczny **back-end**:

- **MongoDB**: Wybraliśmy bazę danych MongoDB jako nasze rozwiązanie do przechowywania danych. MongoDB jest nierelacyjną bazą danych, która zapewnia skalowalność, elastyczność i łatwość w obsłudze, co jest istotne dla naszego projektu.
- **Express.js**: Express.js to minimalistyczny i elastyczny framework aplikacji webowych dla Node.js. Używamy Express.js jako naszego serwera HTTP, umożliwiającego obsługę żądań klienta i dostarczanie odpowiedzi na żądania.
- **Mongoose**: Mongoose to biblioteka ODM (Object-Document Mapping) dla MongoDB, która ułatwia integrację bazy danych MongoDB z naszą aplikacją. Dzięki Mongoose możemy definiować modele danych, wykonywać zapytania i zarządzać danymi w bazie MongoDB w bardziej wygodny sposób.
- **Nodemon**: Nodemon to narzędzie deweloperskie dla Node.js, które automatycznie restartuje serwer po wykryciu zmian w kodzie. Używamy Nodemon w naszym projekcie, aby zapewnić wygodne środowisko deweloperskie i szybką iterację w trakcie tworzenia aplikacji.

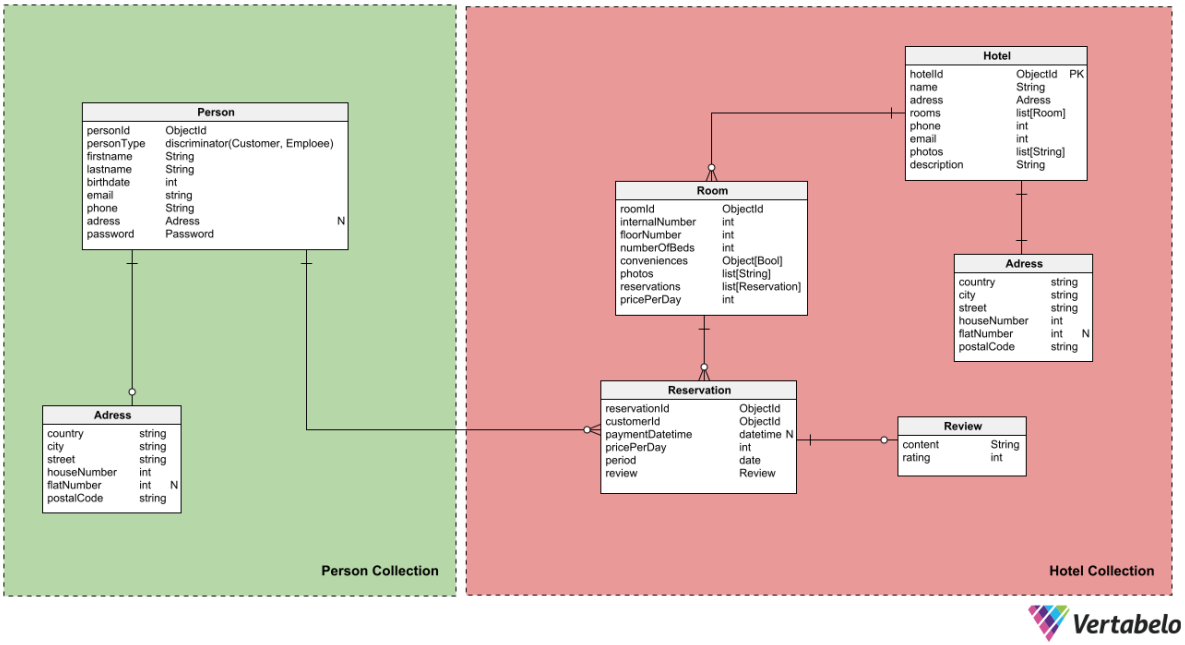
- **fuzzy-search**: jest narzędziem programistycznym, które umożliwia wyszukiwanie informacji w sposób elastyczny i tolerancyjny na błędy. Wykorzystuje ona podejście oparte na logice rozmytej, które pozwala na porównywanie tekstów, uwzględniając podobieństwo znaków i ciągów znaków, zamiast polegać na dokładnym dopasowaniu

Dodatkowo, podczas naszego projektu skorzystaliśmy z technologii **Docker**, która odegrała kluczową rolę w tworzeniu i wdrażaniu naszej aplikacji. Docker jest popularnym narzędziem do konteneryzacji, które umożliwia nam pakowanie aplikacji i jej zależności w lekkie i przenośne jednostki zwane kontenerami. Dzięki Dockerowi byliśmy w stanie zapewnić jednolite środowisko uruchomieniowe dla naszej aplikacji na różnych platformach, co zwiększyło skalowalność, przenośność i niezawodność naszego rozwiązania. Wykorzystanie Docker'a pozwoliło nam również skrócić czas konfiguracji i zapewnić izolację aplikacji, co przyczyniło się do wyższej niezawodności i bezpieczeństwa naszego systemu.

Dzięki zastosowaniu tych technologii jesteśmy w stanie stworzyć aplikację, która zapewnia użytkownikom interaktywny interfejs, efektywne zarządzanie danymi, a także integrację z usługami zewnętrznymi, takimi jak Google Maps.

./’

Schemat



Vertabelo

Kolekcje

people

Kolekcja przechowująca informację o klientach (modelu Person) oraz pracownikach (modelu Employee)

Atrybut	Opis
_id	Autogenerowane ID people
__t	Pole dodane przez discriminator, służące do rozróżnienia modelu Person od modelu Employee
address	Obiekt adres
email	adres email
password	Hasło
phone	Numer telefonu
birthdate	Data urodzenia

firstname	Imię
lastname	Nazwisko
updatedAt	Data ostatniej aktualizacji elementu
createdAt	data utworzenia elementu
position	Stanowisko w firmie (dotyczy tylko modelu Employee)
salary	Wynagrodzenie (dotyczy tylko modelu Employee)

Przyykładowy dokument modelu Person:

```
{
    "_id": "64908f5fb8f0035e7621564a",
    "firstname": "Test",
    "lastname": "Testowski",
    "birthdate": "2000-01-01T00:00:00.000Z",
    "email": "person@person.com",
    "address": {
        "country": "Polska",
        "city": "Kraków",
        "street": "Kawiory",
        "houseNumber": "21",
        "postalCode": "30055"
    },
    "password": "zaq1@WSX",
    "createdAt": "2023-06-19T17:24:47.154Z",
    "updatedAt": "2023-06-19T17:24:47.154Z",
    "__v": 0
}
```

Przykładowy dokument modelu Employee:

```
{
    "_id": "64908f5fb8f0035e76215660",
    "__v": 0,
    "address": {
        "country": "Polska",
        "city": "Kraków",
        "street": "Kawiory",
        "houseNumber": "21a",
        "postalCode": "30055"
```

```
    },
    "birthdate": "2000-01-01T00:00:00.000Z",
    "createdAt": "2023-06-19T17:24:47.180Z",
    "email": "admin@admin.com",
    "firstname": "Admin",
    "lastname": "Admin",
    "password": "admin",
    "updatedAt": "2023-06-19T17:24:47.180Z",
    "__t": "Employee",
    "position": "Admin",
    "salary": 15000
}
```


hotel

Kolekcja przechowująca informację o konkretnym hotelu. Każdy hotel ma pokoje (schema Room), każdy pokój ma przypisane rezerwacje (schema Reservation)

Atrybut	Opis
_id	Autogenerowane ID hotelu
name	Nazwa hotelu
address	Obiekt Address, przechowujący informacje o adresie hotelu
rooms	Subkolekcja Room
email	Adres email
phone	Numer telefonu
photos	Lista URL z zdjęciami
description	Opis hotelu
createdAt	Data utworzenia dokumentu
updatedAt	Data aktualizacji dokumentu

Przykładowy dokument


```
{
    "_id": "64908f5eb8f0035e762155e6",
    "__v": 126,
    "address": {
```

```
        "country": "Polska",
        "city": "Kraków",
        "street": "Styczna",
        "houseNumber": "37d",
        "postalCode": "32764"
    },
    "description": "Nasz hotel to doskonałe miejsce na wakacje...",
    "email": "contact@kraków.hotel.com.pl",
    "name": "Przytulny Dom",
    "phone": "+48514123620",
    "photos": [
        "https://picsum.photos/id/220/1200/1200",
        "https://picsum.photos/id/270/1200/1200",
        "https://picsum.photos/id/251/1200/1200"
    ],
    "rooms": [
        {
        "internalNumber": 106,
        "floorNumber": 2,
        "beds": 3,
        "conveniences": {
            "kitchen": true,
            "smoking": true,
            "pets": true,
            "balcony": true
        },
        "photos": [],
        "pricePerDay": 373,
        "maxCapacity": 4,
        "_id": "64908f5eb8f0035e762155f2",
        "reservations": [
            {
            "personId": {"$oid": "64908f5fb8f0035e7621564a"},
            "pricePerDay": 373,
            "period": {
                "startDate": "2021-09-03T15:32:15.722Z",
                "dueDate": "2021-09-03T15:32:15.595Z"
            },
            "review": {
                "content": "To był doskonały hotel na pobyt!",
                "_id": "64908f5fb8f0035e762157fe",
                "stars": 5,
                "createdAt": "2023-06-19T17:24:47.459Z",
```

```
                "updatedAt": "2023-06-19T17:24:47.459Z"
            },
            "_id": "64908f5fb8f0035e762157fd"
            }
        ]
        }
    ],
    "createdAt": {"$date": "2023-06-19T17:24:47.027Z"},
    "updatedAt": "2023-06-19T17:25:00.115Z"
}
```

Subkolekcje

Dzięki wykorzystaniu biblioteki mongoose mogliśmy wyodrębnić z schematu głównych kolekcji kilka różnych podkolekji, które prezentują się następująco:

Room

Subkolekcja, przechowująca informacje o danym pokoju. Subkolekcja ta jest przechowywana przez model Hotel pod atrybutem rooms.

Atrybut	Opis
_id	Autogenerowane ID room
internalNumber	Wewnętrzny numer pokoju
floorNumber	Numer piętra, na którym dany pokój się znajduje
beds	liczba łóżek
photos	Lista adresów url do zdjęć danego pokoju
reservations	Subkolekcja reservations
pricePerDay	Cena za wynajem (na dzień)
maxCapacity	Maksymalna pojemność danego pokoju/maksymalna liczba ludzi, którzy

Reservation

Subkolekcja przechowująca informację o rezerwacjach w danym pokoju

Atrybut	Opis
_id	autogenerowane ID rezerwacji
personId	id użytkownika
paymentDate	Data płatności (nie ustawiona, jeśli jeszcze nie dokonano płatności)
pricePerDay	Cena za dzień (kopiowana w momencie utworzenia rezerwacji z rodzica - Room)
period.startDate	Okres rezerwacji
period.dueDate	
review.content	Opinia o danej rezerwacji

review.stars	Liczba gwiazdek (od 1 do 5)

Address

Subkolekcja/Obiekt, przechowujący informację o adresie. Obie kolekcje - hotels oraz people - posiadają po jednym takim obiekcie na dokument

Atrybut	Opis
country	Kraj
city	Miasto
street	Ulica
houseNumber	Numer budynku
flatNumber	Numer mieszkania (opcjonalny)
postalCode	Kod pocztowy

API:

Hotele

post /hotel

Ta funkcja obsługuje żądanie POST na ścieżce "/hotel". Po otrzymaniu żądania, funkcja pobiera dane z ciała żądania, takie jak nazwa, adres, numer telefonu, adres e-mail, zdjęcia i opis hotelu. Następnie wywołuje funkcję addHotel, przekazując pobrane dane. Funkcja addHotel zajmuje się dodawaniem informacji o hotelu do systemu lub bazy danych. Jeśli operacja dodawania przebiegnie pomyślnie, funkcja zwraca odpowiedź JSON o statusie "ok". Jeśli wystąpi błąd, funkcja zwraca odpowiedź JSON z kodem statusu 400 i informacją o błędzie.

Paramter	**Opis**
name	Nazwa hotelu
address	Adres hotelu
phone	Telefon kontaktowy hotelu
email	Email kontaktowy hotelu
photos	Lista URLi do zdjęć hotelu
description	Opis hotelu

get /hotel/:hotelId

Funkcja obsługuje żądanie GET na ścieżce "/hotel/:hotelId". Pobiera parametry z żądania, wyszukuje hotel o określonym ID w bazie danych, tworzy tablicę opinii na podstawie danych rezerwacji, oblicza średnią ocenę i zwraca odpowiedź JSON zawierającą hotel, średnią ocenę i tablicę opinii. W przypadku błędu, zwraca odpowiedni kod statusu i komunikat o błędzie.

Paramter	**Opis**
hotelId	Id hotelu
startDate	Adres hotelu
dueDate	Telefon kontaktowy hotelu

get /hotels

Ta funkcja obsługuje żądania GET na ścieżce "/hotels" i wyszukuje hotele spełniające określone kryteria. Pobiera parametry zapytania, wykonuje wyszukiwanie hoteli, sprawdza dostępność pokoi i oblicza średnią ocenę dla każdego hotelu. Ostatecznie zwraca listę znalezionych hoteli w formacie JSON.

Paramter	**Opis**
capacity	Szukana liczba osób
startDate	Adres hotelu
dueDate	Telefon kontaktowy hotelu
search	Pattern do fuzzysearch -przeszukuje po nazwie i adresie hotelu

delete /hotel/:hotelId

Ta funkcja obsługuje żądanie DELETE na ścieżce "/hotel/:hotelId". Usuwa dokument hotelu o podanym "hotelId" za pomocą metody "findByIdAndDelete" i zwraca odpowiedź JSON o statusie "ok" lub kodzie błędu 400 w przypadku wystąpienia błędu.

Paramter	**Opis**
hotelID	Id hotelu do wykonania operacji

Pokoje

get /hotel/:hotelId/rooms

Pobranie aktualnie dostępnych pokoi. Domyślnie pobiera wszystkie pokoje. Po ustawieniu parametrów startDate oraz endDate

Paramter	**Opis**
startDate	Data początkowa
dueDate	Data końcowa
…conveniences	Wszystkie keywordy, będące oficjalnie wspieranymi udogodnieniami pokoju

post /room

Dodanie nowego pokoju do hotelu. Jako body należy przesłać

Parametr	Opis
hotelId	Id hotelu
maxPeople	Maksymalna liczba ludzi w danym pokoju
conveniencesa	Obiekt z udogodnieniami danego pokoju
internalNumber	Numer wewnętrzny danego pokoju
floor	Piętro, na którym dany pokój się znajduje
photos	Lista url ze zdjęciami
pricePerDay	Cena pokoju za dzień

get /hotel/:hotelId/room/:roomId

Funkcja ta przyjmuje parametry z żądania oznaczone jako ":hotelId" i ":roomId". Następnie próbuje odnaleźć hotel o podanym "hotelId" przy użyciu metody findOne z biblioteki Mongoose, która komunikuje się z bazą danych. W zapytaniu do bazy danych wykorzystuje się obiekt mongoose.Types.ObjectId, aby przekształcić wartość "hotelId" na typ ObjectId.

Jeśli hotel nie zostanie znaleziony, funkcja wyrzuca wyjątek z komunikatem "Hotel not found". W przeciwnym przypadku funkcja odnajduje pokój o podanym "roomId" w znalezionym hotelu przy użyciu metody id z biblioteki Mongoose.

Na końcu, funkcja zwraca odpowiedź w formacie JSON zawierającą znaleziony pokój jako obiekt "room". Jeśli wystąpił błąd, funkcja zwraca odpowiedź z kodem statusu 400 (Bad Request) i komunikatem błędu w formacie JSON.

Paramter	**Opis**
hotelID	Id szukanego hotelu
roomId	Id szukanego pokoju

DELETE /hotel/:hotelId/room/:roomId

Ta funkcja obsługuje żądanie DELETE dla trasy "/hotel/:hotelId/room/:roomId" w aplikacji. Usuwa pokój o określonym hotelId i roomId. Jeśli operacja przebiegnie pomyślnie, zwraca odpowiedź JSON z polem "status" o wartości "ok". W przypadku błędu, zwraca odpowiedź JSON o statusie 400 i zawiera błąd w polu "error".

Paramter	Opis
hotelID	Id szukanego hotelu
roomId	Id szukanego pokoju

Rezerwacje

post /reservation

Utworzenie nowej rezerwacji. Jako body należy przesłać:

Parametr	Opis
personId	Id użytkownika
roomId	id pokoju
startDate	Data początkowa pobytu
dueDate	Data końcowa pobytu

Utworzenie nowej rezerwacji dla aktualnie zalogowanego użytkownika.

delete reservations/:reservationId/:personId

Ta funkcja jest endpointem DELETE dla ścieżki "/reservations/:reservationId/:personId" w aplikacji. Po otrzymaniu żądania DELETE, funkcja pobiera parametry reservationId i personId z żądania. Następnie wywołuje funkcję deleteRoomReservation z tymi parametrami, aby usunąć rezerwację pokoju dla określonej osoby. Jeśli operacja przebiegnie pomyślnie, zwracana jest odpowiedź JSON z statusem "ok". W przypadku wystąpienia błędu, funkcja zwraca odpowiedź z kodem statusu 400 i obiektem JSON zawierającym informacje o błędzie.

Parametr	Opis
reservationId	Id rezerwacji do usuniecia
personId	id osoby usuwającej rezerwacje

get /hotel/:hotelId/reservations

Gdy funkcja zostanie wywołana, odczytuje parametr "hotelId" z żądania, a następnie używa go do wywołania funkcji "getHotelReservations" w celu pobrania rezerwacji dla danego

hotelu. Po otrzymaniu odpowiedzi z rezerwacjami, funkcja zwraca je jako odpowiedź JSON. Jeśli wystąpi jakiś błąd podczas przetwarzania żądania, funkcja zwróci odpowiedź z kodem 500 (Internal Server Error) i wiadomością o błędzie w formacie JSON.

Parametr	**Opis**
hotelId	Id hotelu

get /reservation/:reservationId

Szuka rezerwacji o podanym identyfikatorze w bazie danych hoteli. Jeśli rezerwacja zostanie znaleziona, zwraca jej dane w formacie JSON. W przeciwnym razie zwraca błąd 404, jeśli rezerwacja nie została znaleziona, lub błąd 500 w przypadku problemu z serwerem.

Parametr	**Opis**
reservationId	Id rezerwacji szukanej

get /reservations

Pobranie wszystkich dostępnych rezerwacji.

get /person/:personId/reservations

Ta funkcja jest obsługą żądania GET, która otrzymuje parametr "personId" z URL-a. Funkcja pobiera rezerwacje osoby o określonym "personId" przy użyciu funkcji "getPersonReservations". Następnie zwraca te rezerwacje jako odpowiedź w formacie JSON. Jeśli wystąpi błąd podczas pobierania rezerwacji, funkcja zwraca odpowiedź z kodem błędu 500 i komunikatem "Internal Server Error".

Parametr	**Opis**
reservationId	Id osoby

Autoryzacja i Autentykacja

post /login

Ta funkcja obsługuje żądania POST na ścieżce "/login". Przyjmuje dane logowania (adres e-mail i hasło) z żądania, sprawdza ich poprawność i zwraca odpowiedź w formacie JSON zawierającą informacje o użytkowniku lub komunikat błędu, w zależności od wyniku weryfikacji.

Parametr	Opis
email	email logującego
password	hasło logującego

post /register

Ta funkcja obsługuje żądania HTTP typu POST na ścieżce "/register". Głównym celem tej funkcji jest rejestracja nowej osoby w aplikacji. Funkcja odbiera dane przesłane w żądaniu (firstname, lastname, birthdate, email, phone, address, password) za pomocą req.body. Następnie sprawdza poprawność danych i zapisuje nową osobę w bazie danych. Jeśli rejestracja powiedzie się, funkcja zwraca odpowiedź JSON zawierającą informacje o walidacji (validation: 'true'), identyfikatorze osoby (personId) oraz jej imieniu (name). Jeśli wystąpi błąd podczas rejestracji, funkcja zwraca odpowiedź z kodem statusu 400 i komunikatem błędu w formacie JSON.

Parametr	Opis
email	email rejestrujacego
password	hasło rejestrujacego
firstname	imie nowej osoby
lastname	nazwisko osoby
birthdate	data urodzenia osoby
phone	telefon kontaktowy osoby
address	adress osoby

post /registerEmployee

Ta funkcja działa tak samo tak register ale przyjmuje dodatkowe dwa argumenty: position i salary mówiące o pozycji pracownika oraz jego wypłacie

Osoby:

GET /persons:

Ta funkcja zwraca wszystkie osoby z bazy

GET/persons/:personId

Ta funkcja zwraca osobę o zadanym ID

Parametr	**Opis**
personId	Id szukanej osoby

DELETE/persons/:personId

Ta funkcja usuwa osobę o zadanym ID z bazy

Parametr	**Opis**
personId	Id szukanej osoby

Procedury

addHotel:

Funkcja addHotel dodaje nowy hotel do systemu, sprawdzając czy wszystkie wymagane pola są wypełnione. Następnie zapisuje hotel w bazie danych i opcjonalnie rejestruje operację w dzienniku.

Paramter	**Opis**
name	Nazwa hotelu
address	Adres hotelu
phone	Telefon kontaktowy do hotelu
email	Adres email hotelu
photos	Lista URLi do zdjęć w hotelu

addRoomReservation:

Jest to asynchroniczna funkcja, która dodaje nową rezerwację pokoju na podstawie przekazanych parametrów. Sprawdza poprawność danych, a następnie tworzy i zapisuje rezerwację do bazy danych.

Paramter	**Opis**

personId	Id osoby która dokonuje rezerwacji
roomId	Id hotelu dla którego jest dokonywana rezerwacja
params	Krótka zawierająca datę początkową i końcową rezerwacji

deleteRoomReservation:

Jest to asynchroniczna funkcja, która dodaje usuwa rezerwację pokoju na podstawie przekazanych parametrów. Sprawdza poprawność danych, a następnie usuwa rezerwację z bazy danych.

Paramter	**Opis**
personId	Id osoby która usuwa swoją rezerwację
reservationId	Id rezerwacji kotra ma zostac usunieta

checkEmployeePositionById:

Jest to asynchroniczna funkcja, sprawdza czy osoba o przekazanym id to menadżer

Paramter	**Opis**
personId	Id osoby

deleteRoom:

Jest to asynchroniczna funkcja, która dodaje usuwa pokój z hotelu.

Paramter	**Opis**
hotelId	Id hotelu z którego będzie usuwany pokój
roomId	Id pokoju do usuniecia

Funkcje

getPersonReservations

Ta funkcja pobiera identyfikator osoby i zwraca jej rezerwacje. Przeszukuje ona hotele, pokoje i rezerwacje, aby znaleźć odpowiednie rezerwacje dla danej osoby. Jeśli wystąpi błąd, zostanie wyświetlony komunikat o błędzie i rzucony dalej.

Paramter	Opis
personID	Id szukanej osoby

getHotelReservations

Funkcja jest asynchroniczną funkcją, która pobiera rezerwacje hotelowe dla określonego hotelu. Przeszukuje pokoje hotelowe i zwraca wszystkie rezerwacje znalezione w tych pokojach. Jeśli wystąpi błąd, jest on obsługiwany i ponownie rzucony.

Paramter	Opis
hotelID	Id szukanego hotelu

getOverlappingRoomIds

To funkcja, która przyjmuje identyfikator hotelu, datę rozpoczęcia i datę zakończenia, a następnie zwraca identyfikatory pokoi w hotelu, które mają nakładające się rezerwacje w podanym przedziale czasowym. Funkcja korzysta z biblioteki mongoose do interakcji z bazą danych MongoDB.

Paramter	Opis
hotelID	Id przeszukiwanego hotelu
stardDate	Data rozpoczęcia
endDate	Data zakończenia

fuzzySearchHotel

Funkcja przeprowadza wyszukiwanie rozmyte wśród hoteli na podstawie podanego wzorca, zwracając pasujące hotele.

Paramter	Opis
hotels	Lista przeszukiwanych hoteli
pattern	Wzorzec przeszukiwania

getAvailableRooms

Funkcja pobiera dostępne pokoje w hotelu na podstawie określonych kryteriów, takich jak data rozpoczęcia i zakończenia pobytu oraz wymagane wygody.

Paramter	**Opis**
hotelID	Id przeszukiwanego hotelu
stardDate	Data rozpoczęcia
endDate	Data zakończenia
conveniences	Lista udogodnień

Oznaczenia

Person - klient/użytkownik
Employee - pracownik, jego struktura jest oparta o model Person
Hotel - hotel
Room - pokój
Reservation - rezerwacja
Address - adres

Założenia:

- jedna rezerwacja zezwala na rezerwacje w tylko jednym z hoteli (każda rezerwacja dotyczy tylko jednego pokoju)