ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»

**КУРСОВИЙ ПРОЕКТ**

з навчальної дисципліни «Об’єктно-орієнтоване програмування»

на тему: Система управління фінансовою стороною готельного бізнесу

Студента III курсу 217 групи

спеціальності 126

Інформаційні системи та технології

Чикалова Е.Д.

Керівник Прохоров Г.В.

Оцінка   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії:

                     \_\_\_\_\_\_\_\_\_\_\_

 (ПІПб)                                     (підпис)

                         \_\_\_\_\_\_\_\_\_\_\_

 (ПІПб)                                     (підпис)

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

 (ПІПб)                                     (підпис)

м. Чернівці – 2023

***ЗМІСТ***

Вступ

Технічний проект

Контролери

1.1 Rest Контролери  
1.2 UI Контролери

Моделі

Репозиторії

2. Service

2.1 Implementation

2.2 Interfaces

Структура html

Висновок

***Вступ:***

У сучасному світі готельний бізнес є однією з найбільш динамічних та прибуткових галузей. Забезпечення якісного обслуговування клієнтів та ефективного управління фінансовою стороною готелю є критичними факторами для досягнення успіху у цій галузі. І саме з метою полегшити та автоматизувати процес управління фінансовою стороною готелю була розроблена наша програма.

Ця програма пропонує ефективний спосіб відстеження та контролю фінансових аспектів готельного бізнесу. Вона дозволяє готельним адміністраторам та менеджерам з легкістю керувати номерами, клієнтами, поселеннями та іншими фінансовими аспектами, пов'язаними з роботою готелю.

Програма надає зручний інтерфейс, який дозволяє з легкістю керувати інформацією про клієнтів та номери, бронюваннями, поселеннями та оплатами. Вона забезпечує точний облік фінансових операцій, включаючи вартість номерів, знижки, а також можливість створення звітів та аналізу фінансових даних.

У даному звіті ми детально описуємо функціональні можливості програми, структуру бази даних, а також надаємо приклади використання та вигоди, які можуть бути отримані від впровадження такої системи управління фінансовою стороною готельного бізнесу. Знання та розуміння цих аспектів допоможуть вам краще оцінити потенціал і значення нашої програми для вашого готелю.

Нехай цей опис стане вам в пригоді під час ознайомлення з програмою та допоможе прийняти обґрунтоване рішення щодо використання системи управління фінансовою стороною готельного бізнесу для підвищення ефективності та прибутковості вашого готелю.

**Технічний проект**

**Контролери**

**1.2 Rest Контролери**

1. SettlementRestController:

Цей контролер відповідає за обробку HTTP-запитів, пов'язаних з розрахунками (Settlement) у системі готелю. Він надає наступні ендпоінти:

* + **GET /api/v1/settlements/**: Повертає список всіх розрахунків у системі готелю.
  + **GET /api/v1/settlements/{id}**: Повертає інформацію про конкретний розрахунок за його ідентифікатором.
  + **POST /api/v1/settlements/**: Створює новий розрахунок на основі наданої інформації.
  + **PUT /api/v1/settlements/**: Оновлює існуючий розрахунок з використанням наданої інформації.
  + **DELETE /api/v1/settlements/{id}**: Видаляє розрахунок за його ідентифікатором.

1. RoomRestController:

Цей контролер відповідає за обробку HTTP-запитів, пов'язаних з номерами (Room) у системі готелю. Він надає наступні ендпоінти:

* + **GET /api/v1/rooms/**: Повертає список всіх номерів у системі готелю.
  + **GET /api/v1/rooms/{id}**: Повертає інформацію про конкретний номер за його ідентифікатором.
  + **POST /api/v1/rooms/**: Створює новий номер на основі наданої інформації.
  + **PUT /api/v1/rooms/**: Оновлює існуючий номер з використанням наданої інформації.
  + **DELETE /api/v1/rooms/{id}**: Видаляє номер за його ідентифікатором.

1. RegularClientRestController:

Цей контролер відповідає за обробку HTTP-запитів, пов'язаних з постійними клієнтами (RegularClient) у системі готелю. Він надає наступні ендпоінти:

* + **GET /api/v1/regular-clients/**: Повертає список всіх постійних клієнтів у системі готелю.
  + **GET /api/v1/regular-clients/{id}**: Повертає інформацію про конкретного постійного клієнта за його ідентифікатором.
  + **POST /api/v1/regular-clients/**: Створює нового постійного клієнта на основі наданої інформації.
  + **PUT /api/v1/regular-clients/**: Оновлює існуючого постійного клієнта з використанням наданої інформації.
  + **DELETE /api/v1/regular-clients/{id}**: Видаляє постійного клієнта за його ідентифікатором.

1. EvictionOfRoomsRestController:

Цей контролер відповідає за обробку HTTP-запитів, пов'язаних з виселеннями (EvictionOfRooms) у системі готелю. Він надає наступні ендпоінти:

* + **GET /api/v1/evictions/**: Повертає список всіх виселень у системі готелю.
  + **GET /api/v1/evictions/{id}**: Повертає інформацію про конкретне виселення за його ідентифікатором.
  + **POST /api/v1/evictions/**: Створює нове виселення на основі наданої інформації.
  + **PUT /api/v1/evictions/**: Оновлює існуюче виселення з використанням наданої інформації.
  + **DELETE /api/v1/evictions/{id}**: Видаляє виселення за його ідентифікатором.

1. ClientRestController:

Цей контролер відповідає за обробку HTTP-запитів, пов'язаних з клієнтами (Client) у системі готелю. Він надає наступні ендпоінти:

* + **GET /api/v1/clients/**: Повертає список всіх клієнтів у системі готелю.
  + **GET /api/v1/clients/{id}**: Повертає інформацію про конкретного клієнта за його ідентифікатором.
  + **POST /api/v1/clients/**: Створює нового клієнта на основі наданої інформації.
  + **PUT /api/v1/clients/**: Оновлює існуючого клієнта з використанням наданої інформації.
  + **DELETE /api/v1/clients/{id}**: Видаляє клієнта за його ідентифікатором.

Ці контролери реалізують CRUD-операції (створення, отримання, оновлення, видалення) для відповідних моделей у системі готелю. Вони використовують анотації **@RestController**, **@RequestMapping** та анотації HTTP-методів (**@GetMapping**, **@PostMapping**, **@PutMapping**, **@DeleteMapping**) для визначення ендпоінтів та способів їх обробки. Залежно від типу запиту та шляху, контролери викликають відповідні методи сервісів для виконання необхідних операцій з даними.

Цей код демонструє підхід RESTful архітектури у розробці API для системи готелю. Кожен контролер відповідає за конкретну сутність у системі та надає набір ендпоінтів для взаємодії з цією сутністю.

**1.2 UI Контролери**

1. SettlementUiController:
   * **getAll**: Метод обробляє HTTP-запит GET на шляху **/ui/v1/settlements/**. Він викликає сервіс **settlementService.getAll()** для отримання списку всіх розрахунків. Далі, він додає цей список до моделі та повертає ім'я шаблону "settlements", яке відповідає відображенню списку розрахунків.
2. RoomUiController:
   * **getAll**: Метод обробляє HTTP-запит GET на шляху **/ui/v1/rooms/**. Він викликає сервіс **roomService.getAll()** для отримання списку всіх кімнат. Далі, він додає цей список до моделі та повертає ім'я шаблону "rooms", яке відповідає відображенню списку кімнат.
3. RegularClientUiController:
   * **getAll**: Метод обробляє HTTP-запит GET на шляху **/ui/v1/regular-clients/**. Він викликає сервіс **regularClientService.getAll()** для отримання списку всіх регулярних клієнтів. Далі, він додає цей список до моделі та повертає ім'я шаблону "regular-clients", яке відповідає відображенню списку регулярних клієнтів.
4. EvictionOfRoomsUiController:
   * **getAll**: Метод обробляє HTTP-запит GET на шляху **/ui/v1/evictions/**. Він викликає сервіс **evictionService.getAll()** для отримання списку всіх виселень кімнат. Далі, він додає цей список до моделі та повертає ім'я шаблону "evictions", яке відповідає відображенню списку виселень кімнат.
5. ClientUiController:
   * **getAll**: Метод обробляє HTTP-запит GET на шляху **/ui/v1/clients/**. Він викликає сервіс **clientService.getAll()** для отримання списку всіх клієнтів. Далі, він додає цей список до моделі та повертає ім'я шаблону "clients", яке відповідає відображенню списку клієнтів.
6. BillUiController:
   * **getAll**: Метод обробляє HTTP-запит GET на шляху **/ui/v1/bills/**. Він викликає сервіс **billService.getAll()** для отримання списку всіх рахунків. Далі, він додає цей список до моделі та повертає ім'я шаблону "bills", яке відповідає відображенню списку рахунків.

Кожен контролер також має метод **delete**, який обробляє HTTP-запит GET з параметром **{id}** на відповідний шлях (наприклад, **/ui/v1/settlements/{id}**). Цей метод викликає відповідний сервіс для видалення об'єкта з вказаним ідентифікатором **{id}**. Після видалення він повертає редирект на відповідну сторінку списку об'єктів (наприклад, **/ui/v1/settlements/**).

Ці контролери використовуються для відображення списків об'єктів (розрахунків, кімнат, регулярних клієнтів, виселень кімнат, клієнтів, рахунків) у веб-інтерфейсі готелю, а також для видалення об'єктів за їх ідентифікатором.

**Моделі**

1. Settlement:
   * Поля:
     + **id**: унікальний ідентифікатор розрахунку.
     + **client**: об'єкт типу **Client**, який представляє клієнта, що здійснив розрахунок.
     + **room**: об'єкт типу **Room**, який представляє кімнату, в якій проживав клієнт.
     + **dateOfSettlement**: дата і час поселення клієнта в кімнату.
     + **dateOfEviction**: дата і час виселення клієнта з кімнати.
     + **note**: додаткові нотатки або коментарі щодо розрахунку.
     + **createdAt**: дата і час створення розрахунку.
     + **updatedAt**: дата і час останнього оновлення розрахунку.
   * Конструктори:
     + Порожній конструктор.
     + Конструктор, який приймає параметри **client**, **room** і **note** для створення нового розрахунку.
   * Перевизначені методи:
     + **equals()**: порівнює розрахунки за їх ідентифікатором (**id**).
     + **hashCode()**: обчислює хеш-код розрахунку за його ідентифікатором (**id**).
2. Room:
   * Поля:
     + **id**: унікальний ідентифікатор кімнати.
     + **number**: номер кімнати.
     + **numberOfPeople**: кількість осіб, які можуть проживати в кімнаті.
     + **comfort**: перерахування **Comfort**, що вказує рівень комфорту кімнати.
     + **price**: вартість проживання в кімнаті.
     + **createdAt**: дата і час створення кімнати.
     + **updatedAt**: дата і час останнього оновлення кімнати.
   * Конструктори:
     + Порожній конструктор.
     + Конструктор, який приймає параметри **number**, **numberOfPeople**, **comfort** і **price** для створення нової кімнати.
   * Перевизначені методи:
     + **equals()**: порівнює кімнати за їх ідентифікатором (**id**).
     + **hashCode()**: обчислює хеш-код кімнати за її ідентифікатором (**id**).
3. RegularClient:
   * Поля:
     + **id**: унікальний ідентифікатор регулярного клієнта.
     + **client**: об'єкт типу **Client**, який представляє клієнта, який є регулярним.
     + **registration**: дата і час реєстрації регулярного клієнта.
     + **updatedAt**: дата і час останнього оновлення регулярного клієнта.
   * Конструктори:
     + Порожній конструктор.
     + Конструктор, який приймає параметр **client** для створення нового регулярного клієнта.
   * Перевизначені методи:
     + **equals()**: порівнює регулярних клієнтів за їх ідентифікатором (**id**).
     + **hashCode()**: обчислює хеш-код регулярного клієнта за його ідентифікатором (**id**).
4. EvictionOfRooms:
   * Поля:
     + **id**: унікальний ідентифікатор виселення з кімнати.
     + **settlement**: об'єкт типу **Settlement**, який представляє розрахунок, пов'язаний з виселенням.
   * Конструктори:
     + Конструктор, який приймає параметр **settlement** для створення нового виселення з кімнати.
   * Перевизначені методи:
     + **equals()**: порівнює виселення з кімнати за їх ідентифікатором (**id**).
     + **hashCode()**: обчислює хеш-код виселення з кімнати за його ідентифікатором (**id**).
5. Client:
   * Поля:
     + **id**: унікальний ідентифікатор клієнта.
     + **lastName**: прізвище клієнта.
     + **firstName**: ім'я клієнта.
     + **patronymic**: по батькові клієнта.
     + **passportData**: паспортні дані клієнта.
     + **comment**: додатковий коментар щодо клієнта.
     + **clientStatus**: перерахування **ClientStatus**, що вказує статус клієнта.
     + **createdAt**: дата і час створення клієнта.
     + **updatedAt**: дата і час останнього оновлення клієнта.
   * Конструктори:
     + Порожній конструктор.
     + Конструктор, який приймає параметри **lastName**, **firstName**, **patronymic**, **passportData**, **comment** і **clientStatus** для створення нового клієнта.
   * Перевизначені методи:
     + **equals()**: порівнює клієнтів за їх ідентифікатором (**id**).
     + **hashCode()**: обчислює хеш-код клієнта за його ідентифікатором (**id**).
6. Bill:
   * Поля:
     + **id**: унікальний ідентифікатор рахунку.
     + **client**: об'єкт типу **Client**, який представляє клієнта, для якого створений рахунок.
     + **settlements**: список об'єктів типу **Settlement**, які представляють розрахунки, пов'язані з рахунком.
     + **rooms**: список об'єктів типу **Room**, які представляють кімнати, пов'язані з рахунком.
     + **discount**: знижка на рахунок.
     + **totalSum**: загальна сума рахунку.
     + **createdAt**: дата і час створення рахунку.
     + **updatedAt**: дата і час останнього оновлення рахунку.
   * Конструктори:
     + Конструктор, який приймає параметри **client**, **settlements**, **rooms**, **discount** і **totalSum** для створення нового рахунку.
   * Перевизначені методи:
     + **equals()**: порівнює рахунки за їх ідентифікатором (**id**).
     + **hashCode()**: обчислює хеш-код рахунку за його ідентифікатором (**id**).

Перерахування (enums):

1. Comfort: вказує рівень комфорту кімнати і має значення "SUITE", "JUNIOR\_SUITE" або "ORDINARY".
2. ClientStatus: вказує статус клієнта і має значення "REGULAR", "NOT\_REGULAR" або "VIP".

Ці моделі допоможуть вам представити дані клієнтів, кімнат, розрахунків та рахунків у вашій готельній системі.

**Репозиторії**

1. SettlementRepository:
   * Пакет: **com.example.hotel.repository**
   * Опис: Репозиторій для доступу до даних про розрахунки (Settlement).
   * Розширює: **MongoRepository<Settlement, String>**.
   * Анотація **@Repository**: Позначає клас як репозиторій, що забезпечує доступ до даних.
2. RoomRepository:
   * Пакет: **com.example.hotel.repository**
   * Опис: Репозиторій для доступу до даних про кімнати (Room).
   * Розширює: **MongoRepository<Room, String>**.
   * Анотація **@Repository**: Позначає клас як репозиторій, що забезпечує доступ до даних.
3. RegularClientRepository:
   * Пакет: **com.example.hotel.repository**
   * Опис: Репозиторій для доступу до даних про регулярних клієнтів (RegularClient).
   * Розширює: **MongoRepository<RegularClient, String>**.
   * Анотація **@Repository**: Позначає клас як репозиторій, що забезпечує доступ до даних.
4. EvictionOfRoomsRepository:
   * Пакет: **com.example.hotel.repository**
   * Опис: Репозиторій для доступу до даних про виселення з кімнат (EvictionOfRooms).
   * Розширює: **MongoRepository<EvictionOfRooms, String>**.
   * Анотація **@Repository**: Позначає клас як репозиторій, що забезпечує доступ до даних.
5. ClientRepository:
   * Пакет: **com.example.hotel.repository**
   * Опис: Репозиторій для доступу до даних про клієнтів (Client).
   * Розширює: **MongoRepository<Client, String>**.
   * Анотація **@Repository**: Позначає клас як репозиторій, що забезпечує доступ до даних.
6. BillRepository:
   * Пакет: **com.example.hotel.repository**
   * Опис: Репозиторій для доступу до даних про рахунки (Bill).
   * Розширює: **MongoRepository<Bill, String>**.
   * Анотація **@Repository**: Позначає клас як репозиторій, що забезпечує доступ до даних.

Кожен репозиторій надає стандартні методи для збереження, оновлення, видалення та отримання даних з бази даних, завдяки спадкуванню від **MongoRepository**.

**Service**

**1.1 Implementation**

1. SettlementServiceImpl:
   * Пакет: **com.example.hotel.service.impls**
   * Опис: Реалізація сервісу **SettlementService**.
   * Анотація **@Service**: Позначає клас як сервіс, що надає бізнес-логіку.
   * Залежності:
     + **SettlementRepository**: Репозиторій для доступу до даних про розрахунки.
     + **EvictionOfRoomsRepository**: Репозиторій для доступу до даних про виселення з кімнат.
     + **ClientService**: Сервіс для роботи з даними про клієнтів.
     + **RoomService**: Сервіс для роботи з даними про кімнати.
   * Методи:
     + **getAll()**: Повертає список всіх розрахунків.
     + **getOne(String id)**: Повертає розрахунок за заданим ідентифікатором.
     + **create(Settlement settlement)**: Створює новий розрахунок.
     + **update(Settlement settlement)**: Оновлює існуючий розрахунок.
     + **delete(String id)**: Видаляє розрахунок за заданим ідентифікатором.
2. RoomServiceImpl:
   * Пакет: **com.example.hotel.service.impls**
   * Опис: Реалізація сервісу **RoomService**.
   * Анотація **@Service**: Позначає клас як сервіс, що надає бізнес-логіку.
   * Залежності:
     + **RoomRepository**: Репозиторій для доступу до даних про кімнати.
   * Методи:
     + **getAll()**: Повертає список всіх кімнат.
     + **getOne(String id)**: Повертає кімнату за заданим ідентифікатором.
     + **create(Room room)**: Створює нову кімнату.
     + **update(Room room)**: Оновлює існуючу кімнату.
     + **delete(String id)**: Видаляє кімнату за заданим ідентифікатором.
3. RegularClientServiceImpl:
   * Пакет: **com.example.hotel.service.impls**
   * Опис: Реалізація сервісу **RegularClientService**.
   * Анотація **@Service**: Позначає клас як сервіс, що надає бізнес-логіку.
   * Залежності:
     + **RegularClientRepository**: Репозиторій для доступу до даних про постійних клієнтів.
     + **ClientService**: Сервіс для роботи з даними про клієнтів.
   * Методи:
     + **getAll()**: Повертає список всіх постійних клієнтів.
     + **getOne(String id)**: Повертає постійного клієнта за заданим ідентифікатором.
     + **create(RegularClient regularClient)**: Створює нового постійного клієнта.
     + **update(RegularClient regularClient)**: Оновлює існуючого постійного клієнта.
     + **delete(String id)**: Видаляє постійного клієнта за заданим ідентифікатором.
4. EvictionServiceImpl:
   * Пакет: **com.example.hotel.service.impls**
   * Опис: Реалізація сервісу **EvictionService**.
   * Анотація **@Service**: Позначає клас як сервіс, що надає бізнес-логіку.
   * Залежності:
     + **EvictionOfRoomsRepository**: Репозиторій для доступу до даних про виселення з кімнат.
     + **SettlementService**: Сервіс для роботи з даними про розрахунки.
   * Методи:
     + **getAll()**: Повертає список всіх виселень з кімнат.
     + **getOne(String id)**: Повертає виселення з кімнати за заданим ідентифікатором.
     + **create(EvictionOfRooms evictionOfRooms)**: Створює нове виселення з кімнати.
     + **update(EvictionOfRooms evictionOfRooms)**: Оновлює існуюче виселення з кімнати.
     + **delete(String id)**: Видаляє виселення з кімнати за заданим ідентифікатором.
5. ClientServiceImpl:
   * Пакет: **com.example.hotel.service.impls**
   * Опис: Реалізація сервісу **ClientService**.
   * Анотація **@Service**: Позначає клас як сервіс, що надає бізнес-логіку.
   * Залежності:
     + **ClientRepository**: Репозиторій для доступу до даних про клієнтів.
   * Методи:
     + **getAll()**: Повертає список всіх клієнтів.
     + **getOne(String id)**: Повертає клієнта за заданим ідентифікатором.
     + **create(Client client)**: Створює нового клієнта.
     + **update(Client client)**: Оновлює існуючого клієнта.
     + **delete(String id)**: Видаляє клієнта за заданим ідентифікатором.
6. BillServiceImpl:
   * Пакет: **com.example.hotel.service.impls**
   * Опис: Реалізація сервісу **BillService**.
   * Анотація **@Service**: Позначає клас як сервіс, що надає бізнес-логіку.
   * Залежності:
     + **BillRepository**: Репозиторій для доступу до даних про рахунки.
     + **ClientService**: Сервіс для роботи з даними про клієнтів.
     + **SettlementService**: Сервіс для роботи з даними про розрахунки.
     + **RoomService**: Сервіс для роботи з даними про кімнати.
   * Методи:
     + **getAll()**: Повертає список всіх рахунків.
     + **getOne(String id)**: Повертає рахунок за заданим ідентифікатором.
     + **create(Bill bill)**: Створює новий рахунок.
     + **update(Bill bill)**: Оновлює існуючий рахунок.
     + **delete(String id)**: Видаляє рахунок за заданим ідентифікатором.
     + **getTotalDiscount(Client client, List<Settlement> settlementList)**: Обчислює загальну знижку для клієнта.
     + **getTotalSum(Client client, List<Settlement> settlementList, List<Room> roomList)**: Обчислює загальну суму для клієнта.
     + **getBill(Client client, List<Settlement> settlementList, List<Room> roomList)**: Створює рахунок для клієнта з врахуванням знижок і загальної суми.

**1.2 Interfaces**

1. SettlementService:
   * Методи:
     + **getAll()**: Повертає список всіх розрахунків.
     + **getOne(String id)**: Повертає розрахунок за заданим ідентифікатором.
     + **create(Settlement settlement)**: Створює новий розрахунок.
     + **update(Settlement settlement)**: Оновлює існуючий розрахунок.
     + **delete(String id)**: Видаляє розрахунок за заданим ідентифікатором.
2. RoomService:
   * Методи:
     + **getAll()**: Повертає список всіх кімнат.
     + **getOne(String id)**: Повертає кімнату за заданим ідентифікатором.
     + **create(Room room)**: Створює нову кімнату.
     + **update(Room room)**: Оновлює існуючу кімнату.
     + **delete(String id)**: Видаляє кімнату за заданим ідентифікатором.
3. RegularClientService:
   * Методи:
     + **getAll()**: Повертає список всіх постійних клієнтів.
     + **getOne(String id)**: Повертає постійного клієнта за заданим ідентифікатором.
     + **create(RegularClient regularClient)**: Створює нового постійного клієнта.
     + **update(RegularClient regularClient)**: Оновлює існуючого постійного клієнта.
     + **delete(String id)**: Видаляє постійного клієнта за заданим ідентифікатором.
4. EvictionService:
   * Методи:
     + **getAll()**: Повертає список всіх виселень з кімнат.
     + **getOne(String id)**: Повертає виселення з кімнати за заданим ідентифікатором.
     + **create(EvictionOfRooms evictionOfRooms)**: Створює нове виселення з кімнати.
     + **update(EvictionOfRooms evictionOfRooms)**: Оновлює існуюче виселення з кімнати.
     + **delete(String id)**: Видаляє виселення з кімнати за заданим ідентифікатором.
5. ClientService:
   * Методи:
     + **getAll()**: Повертає список всіх клієнтів.
     + **getOne(String id)**: Повертає клієнта за заданим ідентифікатором.
     + **create(Client client)**: Створює нового клієнта.
     + **update(Client client)**: Оновлює існуючого клієнта.
     + **delete(String id)**: Видаляє клієнта за заданим ідентифікатором.
6. BillService:
   * Методи:
     + **getAll()**: Повертає список всіх рахунків.
     + **getOne(String id)**: Повертає рахунок за заданим ідентифікатором.
     + **create(Bill bill)**: Створює новий рахунок.
     + **update(Bill bill)**: Оновлює існуючий рахунок.
     + **delete(String id)**: Видаляє рахунок за заданим ідентифікатором.

Ці інтерфейси визначають специфікацію методів, які повинні бути реалізовані в класах сервісів, що забезпечують роботу з моделями даних у системі готелю.

**Нtml опис**

1. Settlements:
   * Ця сторінка відображає список розрахунків.
   * У таблиці виводяться наступні колонки: "Client", "Room", "DateOfSettlement", "DateOfEviction", "Note", "Delete".
   * За допомогою директиви **<#list settlements as settlement>...</#list>**, дані про розрахунки заповнюються в таблицю.
   * Кожен рядок таблиці містить дані про один розрахунок, які виводяться за допомогою синтаксису **${settlement.fieldName}**.
2. Rooms:
   * Ця сторінка відображає список кімнат.
   * У таблиці виводяться наступні колонки: "Number", "NumberOfPeople", "Comfort", "Price", "Delete".
   * За допомогою директиви **<#list rooms as room>...</#list>**, дані про кімнати заповнюються в таблицю.
   * Кожен рядок таблиці містить дані про одну кімнату, які виводяться за допомогою синтаксису **${room.fieldName}**.
3. Regular-Clients:
   * Ця сторінка відображає список постійних клієнтів.
   * У таблиці виводиться колонка "Client", "Delete".
   * За допомогою директиви **<#list regulars as regular>...</#list>**, дані про постійних клієнтів заповнюються в таблицю.
   * Кожен рядок таблиці містить дані про одного постійного клієнта, які виводяться за допомогою синтаксису **${regular.fieldName}**.
4. Evictions:
   * Ця сторінка відображає список виселень з кімнат.
   * У таблиці виводяться колонки "Settlement", "Delete".
   * За допомогою директиви **<#list evictions as eviction>...</#list>**, дані про виселення заповнюються в таблицю.
   * Кожен рядок таблиці містить дані про одне виселення, які виводяться за допомогою синтаксису **${eviction.fieldName}**.
5. Clients:
   * Ця сторінка відображає список клієнтів.
   * У таблиці виводяться колонки "LastName", "FirstName", "Patronymic", "PassportData", "Comment", "ClientStatus", "Delete".
   * За допомогою директиви **<#list clients as client>...</#list>**, дані про клієнтів заповнюються в таблицю.
   * Кожен рядок таблиці містить дані про одного клієнта, які виводяться за допомогою синтаксису **${client.fieldName}**.
6. Bills:
   * Ця сторінка відображає список рахунків.
   * У таблиці виводяться колонки "Client", "Settlements", "Rooms", "Discount", "TotalSum", "Delete".
   * За допомогою директиви **<#list bills as bill>...</#list>**, дані про рахунки заповнюються в таблицю.
   * Кожен рядок таблиці містить дані про один рахунок, які виводяться за допомогою синтаксису **${bill.fieldName}**.

Ці HTML описи використовують шаблонний двигун FreeMarker, що дозволяє вбудовувати динамічні дані у сторінки. Дані отримуються з відповідних сервісів та моделей, які реалізують інтерфейси, зазначені у вашому початковому коді.

**Висновок**

В цій курсовій роботі була розроблена готельна система, яка включає набір клієнтських, адміністративних та функціональних можливостей. Система дозволяє управляти клієнтами, номерами готелю, розрахунками та заселенням, а також зберігати інформацію про регулярних клієнтів.

У процесі розробки були використані об'єктно-орієнтоване програмування, мова програмування Java та фреймворк Spring Boot. Система побудована на основі різних модулів та інтерфейсів, що дозволяє забезпечити гнучкість та розширюваність системи.

Кожен модуль системи має визначені контракти (інтерфейси) для виконання основних операцій: створення, читання, оновлення та видалення об'єктів. Це дозволяє легко взаємодіяти з різними компонентами системи та розширювати її функціональність у майбутньому.

Веб-інтерфейс системи забезпечує зручний спосіб взаємодії з користувачем. Кожна сторінка має свій HTML-шаблон, який динамічно відображає дані, отримані з серверної частини. Користувач може переглядати інформацію про заселення, кімнати, регулярних клієнтів, розрахунки та клієнтів, а також виконувати операції видалення за допомогою відповідних кнопок.

Усі ці компоненти разом утворюють функціональну готельну систему, яка може бути використана для ефективного управління клієнтами та готельними ресурсами. За допомогою розробленої системи готель зможе забезпечити зручне та надійне обслуговування своїх клієнтів та впорядковане управління своїми процесами.