Національний університет «Львівська політехніка»

Кафедра програмного забезпечення

**КУРСОВА РОБОТА**

**з дисципліни «Бази даних»**

на тему:

«База даних для роботи гірськолижного курорту»

Виконав: студент

напряму 121«Інженерія програмного забезпечення»

групи ПЗ-32

Багіров І.А.

Керівник:

асистент кафедри програмного забезпечення, Цимбалюк Т.М.

Оцінка:

Національна шкала\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів\_\_\_\_\_\_\_Оцінка ECTS\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_ Павич Н.Я.

(підпис)

\_\_\_\_\_\_\_\_\_\_\_\_ Цимбалюк Т.М.

(підпис)

**Львів – 2019 рік**

# РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ РОБОТИ ГІРСЬКОЛИЖНОГО КУРОРТУ

## 1.1. Опис предметної області “Гірськолижний курорт”

У 1966 році журнал «Time» опублікував статтю, яка стверджувала, що інтернет-магазини ніколи не знайдуть популярність серед жінок. Пояснювалося це тим, що жінкам просто необхідна причина для виходу з дому. А займаючись on-line шопінгом, вони цієї можливості позбавляються.  
  
Це був один з найбільш провальних прогнозів XX століття. Всім відомо, що на сьогоднішній день прибуток від on-line торгівлі в деяких сферах нічим не поступається, а іноді і перевищує дохід від off-line продажів.  
  
Інтернет-магазини мають багато переваг перед стаціонарними торговими точками. І якщо раніше вони використовувалися переважно, як підмога наявним реальним магазинам, то зараз багато людей, починаючи бізнес, створюють тільки інтернет-магазини і заробляють за допомогою них свої капітали.

Одним з найшвидких за темпом розвитку типів інтернет-магазинів можна назвати магазини електронної техніки. Наприклад, побутова техніка є неодмінним супутником життя кожної людини. Економлячи витрати часу і сил на побутове обслуговування, ми тим самим, підвищуємо якість свого життя. Безліч виробників і стрімкий прогрес в цій сфері спонукає домогосподарства постійно оновлювати свою побутову техніку. Організація бізнесу з продажу побутової техніки не вимагає спеціальних знань і досвіду роботи, скоріше вона вимагає підприємницького досвіду. Цей бізнес, зважаючи на високу конкуренцію в цьому сегменті, особливо з боку великих торгових мереж, вимагає серйозних знань в області управління людьми і вибору найкращого товарного асортименту. Також немаловажливою частиною бізнесу по продажу електронної техніки через інтернет можна назвати грамотне управління його процесами.

## 1.2. Постановка завдання роботи інтернет-магазину

До основного функціоналу програмного забезпечення відноситься: робота з базою даних клієнтів (реєстрація, активація/деактивація, підтвердження оплати, замовлення/повернення товарів), працівників (реєстрація, видача відповідних до позиції дозволів і обов’язків), управління бізнес-процесами (шлях обробки отриманого замовлення відповідними категоріями працівників, їх взаємодія).

В базі даних обов'язковими є дані про клієнтів, працівників, їх позиції, обов’язкі, які накладаються на позиції, категорії товарів і характеристики, що відносяться до нього, самі товари зі значеннями відповідних до категорії характеристик.

Доступ до незаенкріпчених даних БД не може мати ніхто. Редагувати схему БД дозволяється лише адміністратору БД.

Основними бізнес-процесами на гірськолижному курорті є:

* Замовлення і його обробка;
* Повернення товару в разі неналежного стану;

До бізнес-правил що керують роботою курорту відносяться:

* Клієнт має бути повідомленим про початок(закінчення) будь-якого кроку процесу, замовником якого виступає (підтвердження/відхилення замовлення, наджодження в відділення і т.д)
* Продавець має бути повідомленим про початок(закінчення) будь-якого кроку процесу, менеджером якого виступає (підтвердження/відхилення замовлення, наджодження в відділення і т.д)
* Продавець має інформацію тільки про процеси, на яких виступає менеджером
* Директор має інформацію про стан всіх процесів магазину
* Магазин не має піклуватися про процеси складу, вони відокремлені один від одного
* Магазин взаємодіє зі складом тільки через менеджерів, яких надає склад на обробку кожного замовлення і які реєструються в системі
* Складів може бути безліч, API — один, на нього треба здійснювати всі запити
* Для магазину склад = менеджер складу, всі питання до нього
* Рішення складу про відправлення товару не можуть бути оскарженні магазином, а лише оброблюються магазином певним чином
* Ніхто не має доступу до БД з персональними даними користтувачів в продакшні напряму

Після оформлення замовлення в першу чергу відбувається перевірка, чи дійсно даний товар є на складі. Наступним кроком продавець дозволяє або відхиляє замовлення, відправляє запит складу. В разі підтвердження замовлення в гру вступає менеджер складу, який контролюється системами складу, не зв’язаними з системую магазину. Все, що знає система магазину про менеджера складу— він відповідальний за перевезення товару зі складу. Про доставку товару зі склада у відділення магазину повідомляє менеджер складу, після чого інформацію підтверджує продавець. В разі успішного отримання магазином товару завдання на його перевезення до місця замовлення отримує кур’єр. Так само, після доставки продукту кур’єр повідомляє про це системі. З іншого боку цей факт підтверджує сам покупець платежем або запитом на повернення в разі неналежного вигляду товару. Повернення товару виконується так само, але в зворотньому напрямку. Кожна зміна стану товару (підтвердження/відхилення, наджодження в відділення) супроводжується нотіфікаціями продавцю і клієнту.

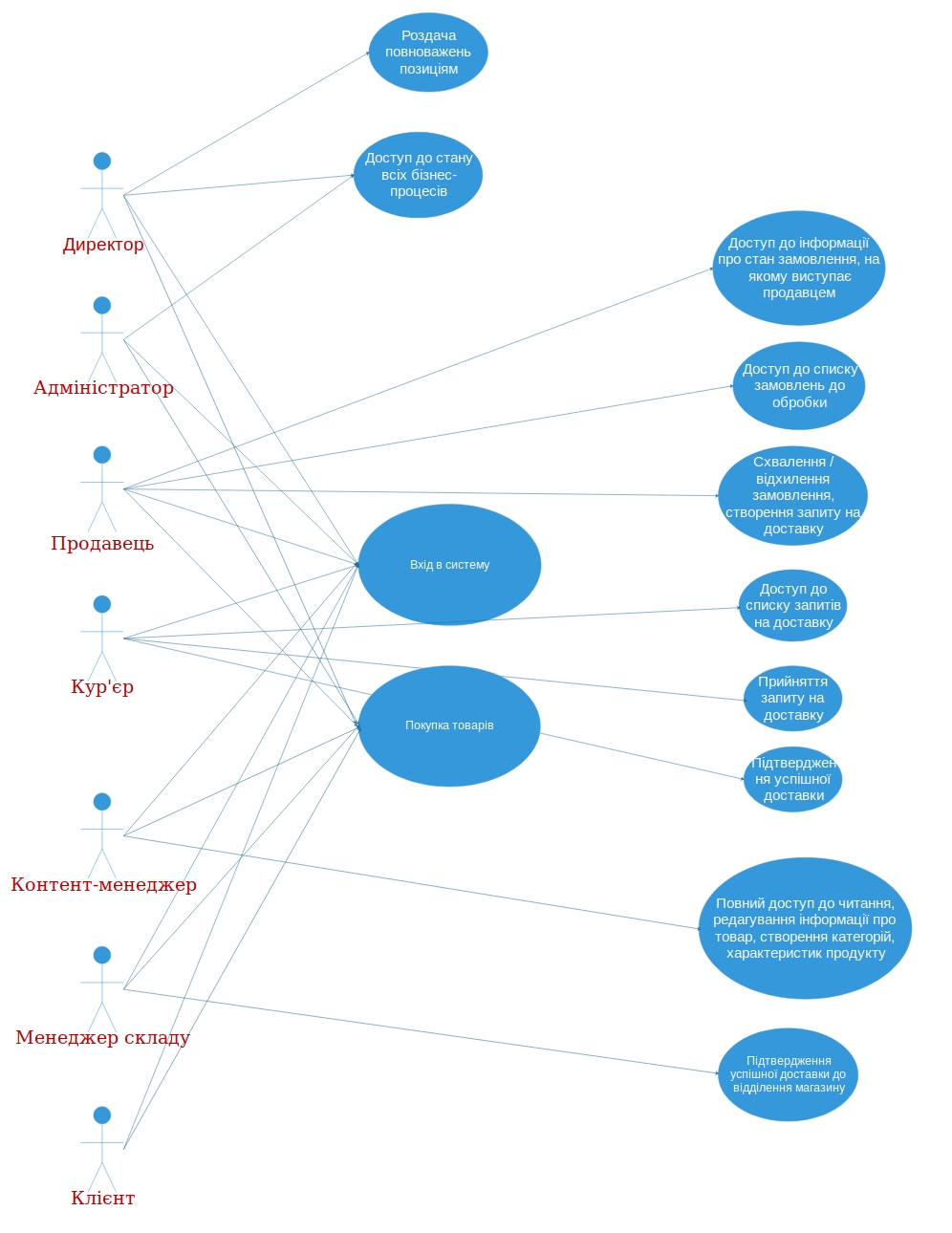


Рис.1.1. Діаграма прецедентів

## 1.3. Специфікація вимог до системи управління гірськолижним курортом

### 1.3.1. Вступ

**1.3.1.1. Призначення, мета**

Метою розробки програмного забезпечення є дозволити як гігантським інтернет-магазинам так і маленьким реалізаторам ефективно управляти внутрішніми бізнес-процесами, розподіляти ресурси, спостерігати та аналізувати стан різних частин інфраструктури.

### 1.3.2. Загальний опис

**1.3.2.1. Перспективи продукту**

Головною перспективою програмного забезпечення є використання магазинами будь-якої країни світу.

**1.3.2.2. Характеристика продукту**

Основними функціями програмного забезпечення є пов’язані з діяльністю інтернет-магазину, а саме: управління базою даних клієнтів, адміністрування товарів, управління бізнес-процесами. Основними характеристиками програмного забезпечення є надійність, гнучкість, високий рівень безпеки, зрозумілість інтерфейсу та легкість навчання нового персоналу.

**1.3.2.3. Класи користувачів та їх характеристики**

Для ефективного розподілу функціональності програмного продукту в ньому повинні бути декілька класів користувачів:

* Покупець — переглядає публічну інформацію і здійснює замовлення
* Продавець — отримує запит на замовлення, дозволяє/відхиляє замовлення, керує всім процесом продажу, супроводжує його. Має інформацію лиш по замовленням, які оброблює сам
* Менеджер складу — виступає партнером магазину, а не підлеглим, підкорюється лише самому складу, відповідає за доставку товару зі складу в відділення магазину
* Кур’єр — отримує завдання на перевезення товару і виконує його
* Директор — може створювати нові позиції, динамічно розподіляти повноваження і обов’язки на уже існуючі, як додаткову або основну, має інформацію про стан всіх процесів і може вирішувати спірні питання — наприклад, в разі загублення товару по дорозі зі складу в магазин
* Адміністратор — має інформацію про стан всіх процесів, веде облік товарів, керує контент-менеджером
* Контент-менеджер — заповнює сторінку інтернет-магазину товарами, адмініструє характеристики, придатні для опису певних типів продукції, заповнює дані цих характеристик для конкретних товарів

**1.3.2.4. Середовище функціонування**

Для функціонування системи необхідна платформа повинна відповідати наступним вимогам:

Таблиця 1.1. Вимоги до платформи

|  |  |
| --- | --- |
| Процесор | x64 сумісний процесор з частотою не менше 2 GHz |
| Пам’ять | Не менше 2 ГБ ОЗП |
| Операційна система | Windows 7+, Linux, MacOS |
| Додаткові програмні компоненти | JDK 8-12, один з інтернет-браузерів: Internet Explorer 7+, Mozilla Firefox 35.0.1+, Chrome 48.0.2564.97, Microsoft Edge |
| Інтернет | Наявність доступу до мережі інтернет є обов’язковою для роботи програми. |

**1.3.2.5. Обмеження проектування та реалізації**

Продукт повинен бути реалізований впродовж 6 місяців після старту розробки. Програмне забезпечення повинне реалізовуватись за допомогою кросплатформенної технології Java Spring 5.1.2+ та бути повноцінним веб-додатком.

**1.3.2.6. Документація користувача**

Документацією користувача є довідка яку можна знайти на офіційному сайті компанії веб-сервісу. Довідка складається з декількох частин, кожна з яких описує можливості окремих модулів: модулі для адміністраторів, рецепції готелів, паркувальників, завідуючих спорядженням та диспетчерів підйомників.

### 1.3.3. Характеристики системи

**1.3.3.1. Авторизація користувачів**

Пріоритет *високий*.

Авторизація користувача

**1.3.3.1.1. Послідовність дія/відгук**

**База даних**

**Форма**

**входу**

**Користувач**

**1.3.3.1.2. Функціональні вимоги**

REQ-1: Введення даних входу (логін/пароль).

REQ-2: Перевірка коректності вводу до передачі запиту до бази даних (використання тільки латиниці та цифр).

REQ-3: Приховування пароля користувача.

REQ-4: Автоматичне визначення ролі користувача та надання йому робочого середовища тільки з доступними йому функціями.

REQ-5: Відхилення авторизації при введенні невалідованих даних (користувач не зареєстрований адміністратором).

**1.3.3.2. Створення аккаунтів для працівників та роздача прав користування програмним забезпеченням**

Пріоритет *високий*. Тільки адміністратор може створювати аккаунти для інших працівників та роздавати їм доступ до функціоналу програми та бази даних.

**1.3.3.2.1. Послідовність дія/відгук**

**Форма**

**редагування даних**

**База даних**

**Користувач**

**1.3.3.2.2. Функціональні вимоги**

REQ-1: Самостійна реєстрація користувачем, який отримує роль переглядача (клієнта).

REQ-2: Юзер з повноваженням “user-management” (за замовчуванням — директор) назначає позицію новому користувачу, який отримує відповідні позиції повноваження автоматично.

### 1.3.4. Вимоги зовнішніх інтерфейсів

**1.3.4.1. Апаратні інтерфейси**

Комп'ютер на якому буде виконуватись дана програма повинен мати наступні апаратні характеристики:

* 64-х бітний процесор з тактовою частотою 2 ГГц, або більше;
* Не менше 2 ГБ ОЗП;
* 100 МБ вільного дискового простору;

**1.3.4.2. Програмні інтерфейси**

Для коректної роботи програми бажана наяність JVM версії 8-12, сучасний браузер (бажано Mozilla Firefox або створений на двигуні Chromium) та мережева ОС що підтримує ці браузери (ОС Windows 7 і вище, Linux, MacOS).

**1.3.4.3. Комунікаційні інтерфейси**

Для коректної роботи програми ПК повинен мати модуль WiFi або ж порт Ethernet для стабільного підключення до мережі Інтернет.

### 1.3.5. Нефункціональні вимоги до проекту

**1.3.5.1. Вимоги продуктивності**

Програмне забезпечення повинно стабільно працювати на ПК середньої вартості. Затримка при запитах до бази даних або через мережу повинні бути відсутні або мінімальні.

**1.3.5.2. Вимоги надійності**

Програмне забезпечення повинно час від часу зберігати дані, щоб уникнути втрат даних під час непередбачених ситуацій (перебої електрики). Програмне забезпечення повинно реагувати без помилок на раптове зникнення доступу до мережі Інтернет.

**1.3.5.3. Вимоги безпеки**

Програмне забезпечення повинно запобігати зовнішньому зловмисному втручанню в базу даних особливо інформацію, критичну для функціонування бізнес-процесів. Також програмне забезпечення повинно розподіляти функціонал та відповідальність різних членів персоналу в залежності від їх ролі та запобігати можливості отримати права адміністратора некомпетентним користувачам.

**1.3.5.4. Атрибути якості програмного продукту**

Програмне забезпечення повинно мати найбільш можливу зручність інтерфейсу, бути легким у навчанні для нового персоналу, мати комунікаційну підтримку користувачів. Програмне забезпечення повинне підтримуватися після релізу та стабільно обновлятись для усунення помилок.

# РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ

## 2.1. Концептуальне моделювання предметної області

### 2.1.1. Опис сутностей

Моя база даних містить наступні сутності: «Країна», «Місто», «Регіон», «Продукт», «Товар», «Повідомлення», «Позиція», «Повноваження» , «Адреса», «Операція», «Профіль», «Юзер», «Характеристика», «Паспорт», «Цілочесельна характеристика», «Булева характеристика», «Характеристика з плаваючою комою», «Текстова характеристика».

«Країна» - країна, яку магазин обслуговує:

Рис.2.1. Таблиця «Країна»

• id – унікальний номер, призначений кожній країні;

• name – назва країни;

«Регіон» – регіон (область) країни, який магазин обслуговує:

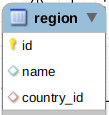


Рис.2.3. Таблиця «Регіон»

• id – унікальний номер, призначений кожному регіону;

• name – назва регіону;

• country\_id – ідентифікатор країни, в якій знаходиться регіон;

«Місто» – місто, яке магазин обслуговує:

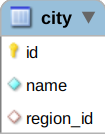


Рис.2.2. Таблиця «Місто»

• id – унікальний номер, призначений кожному місту;

• name – назва міста;

• region\_id – ідентифікатор регіону, в якому знаходиться місто;

«Продукт» – клас товару (комп’ютер, ноутбук, холодильник...) :

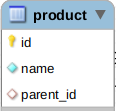


Рис.2.4. Таблиця «Місто»

• id – унікальний номер, призначений кожному продукту;

• name – назва продукта.

• parent\_id – батьківський продукт (комп’ютер-батьківський для ноутбука, зварювальний апарат — батьківський для електро- і газозварювального апарату).

«Товар» – екземпляр продукту:

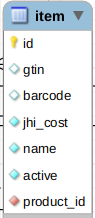


Рис.2.5. Таблиця «Товар»

• id – унікальний номер, призначений кожній одиниці товару;

• gtin – глобальний номер товарної позиції;

• barcode – штрих-код;

• jhi\_cost – ціна;

• name – ім’я.

• active – чи є товар активним.

• product\_id – продукт, до я кого відноситься товар.

«Повідомлення» – повідомлення користувачу від системи:

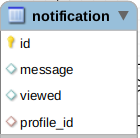


Рис.2.6. Таблиця «Повідомлення»

* id – унікальний номер, призначений кожному повідомленню;
* message – саме повідомлення в текстовому форматі;
* viewed – чи є повідомлення новим;
* profile\_id – профіль отримувача повідомлення;

«Позиція» – позиція в магазині:

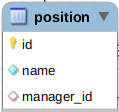


Рис.2.7. Таблиця «Позиція»

• id – унікальний номер, призначений кожній позиції;

• name – назва позиції.

• manager\_id – ідентифікатор позиції, що є наступною вгору по їєрархії позицій компанії.

«Повноваження» – повноваження, якими наділяється позиція:

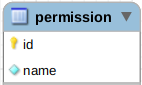


Рис.2.8. Таблиця «Повноваження»

• id – унікальний номер, призначений повноваженню;

• name – назва повноваження;

«Адреса» – адреса (доставки або профіля):

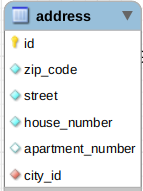


Рис.2.9. Таблиця «Адреса»

• id – унікальний номер, призначений кожній адресі;

• zip\_code - поштовий індекс;

• street - назва вулиці;

• house\_number - номер будівлі;

• apartment\_number - номер квартири;

• city\_id - ідентифікатор міста;

«Операція» – операція над товаром:

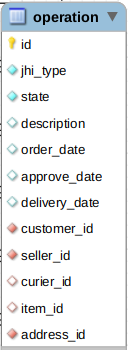


Рис.2.10. Таблиця «Операція»

• id – унікальний номер, призначений кожній операції;

• jhi\_type – тип операції.

• state – стан операції.

• description – додатковий опис операції.

• order\_date – дата замовлення.

• approve\_date – дата обробки продавцем.

• delivery\_date – дата доставки.

• customer\_id – ідентифікатор профіля користувача.

• seller\_id – ідентифікатор профіля продавця.

• curier\_id – ідентифікатор профіля кур’єра.

• item\_id – ідентифікатор товару, до якого відноситься операція.

• address\_id – ідентифікатор адреси доставки.

«Профіль» – профіль користувача:

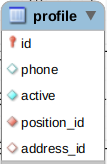


Рис.2.11. Таблиця «Профіль»

• id – унікальний номер, призначений кожному профілю;

• phone – телефон користувача;

• active – чи активний користувач;

• position\_id – ідентифікатор позиції користувача.

• address\_id – ідентифікатор адреси користувача.

«Юзер» – користувач системи:

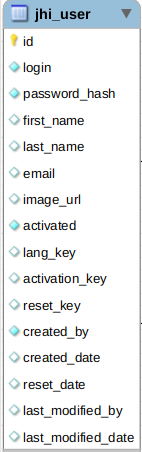


Рис.2.12. Таблиця «Юзер»

* id – унікальний номер, призначений кожному користувачу;
* login – логін для входу в систему;
* password\_hash – пароль для входу в систему;
* first\_name – ім’я користувача;
* last\_name – прізвище користувача;
* email – пошта користувача;
* image\_url – посилання на фотографію користувача.
* activated – чи є користувач активованим.
* lang\_key – мова користувача.
* activation\_key – код активації користувача, який шлеться на пошту.
* reset\_key – посада користувача.
* created\_by – той, хто створив користувача.
* created\_date – дата створення користувача.
* reset\_date – дата, до якої можна змінити пароль по активаційному коду.
* last\_modified\_by – хто в останнє редагував дані про користувача.
* last\_modified\_date – дата останнього редагування користувача.

«Характеристика» – характеристика продукту (класу товарів):

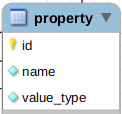


Рис.2.13. Таблиця «Характеристика»

* id – унікальний номер, призначений кожній характеристиці;
* name – назва характеристики.
* value\_type — тип характеристики (ціле число, булева характеристика, стрічка...)

«Паспорт» – паспорт профіля:

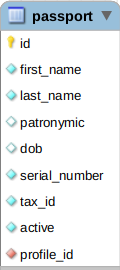


Рис.2.14. Таблиця «Паспорт»

* id – унікальний номер, призначений кожному паспорту;
* first\_name – ім’я;
* last\_name – прізвище;
* patronymic – по батькові;
* dob – дата народження;
* serial\_number – серійний номер паспорту;
* tax\_id — індивідуальний податковий номер
* active — чи є паспорт дійсним
* profile\_id — ідентифікатор профіля власника паспорту

«Повноваження позиції» – зв’язка позиції і наданого повноваження:

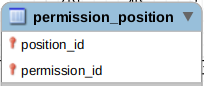


Рис.2.14. Таблиця «Повноваження позиції»

* position\_id – ідентифікатор позиції з наданим повноваженням;
* permission\_id – ідентифікатор повноваження, наданого позиції;

«Характеристика продукту» – зв’язка характеристики і продукту:

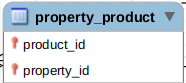


Рис.2.14. Таблиця «Характеристика продукту»

* product\_id – ідентифікатор продукту;
* property\_id – ідентифікатор характеристики;

«Булева характеристика товару» – характеристика товару з можливими станами так/ні (наприклад, наявність веб-камери у ноутбука):

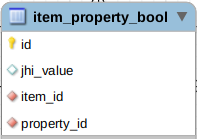


Рис.2.14. Таблиця «Булева характеристика товару»

* id – унікальний номер, призначений кожномій характеристиці;
* jhi\_value – значення характеристики для данного товару;
* item\_id – ідентифікатор товару, якому належить характеристика;
* property\_id – ідентифікатор характеристики продукту;

«Стрічкова характеристика товару» – характеристика товару, значення якої є стрічка (наприклад, додатковий коментар з описом):

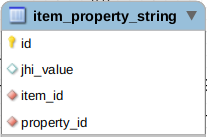


Рис.2.14. Таблиця «Стрічкова характеристика товару»

* id – унікальний номер, призначений кожномій характеристиці;
* jhi\_value – значення характеристики для данного товару;
* item\_id – ідентифікатор товару, якому належить характеристика;
* property\_id – ідентифікатор характеристики продукту;

«Цілочисельна характеристика товару» – характеристика товару, значення якої є ціле число (наприклад, кількість ядер у процессора):

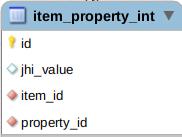


Рис.2.14. Таблиця «Цілочисельна характеристика товару»

* id – унікальний номер, призначений кожномій характеристиці;
* jhi\_value – значення характеристики для данного товару;
* item\_id – ідентифікатор товару, якому належить характеристика;
* property\_id – ідентифікатор характеристики продукту;

«Числова характеристика товару з плаваючою комою» – характеристика товару, значенням якої є число з плаваючою комою (наприклад, діагональ екрану в монітора):

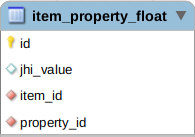


Рис.2.14. Таблиця «Числова характеристика товару з плаваючою комою»

* id – унікальний номер, призначений кожномій характеристиці;
* jhi\_value – значення характеристики для данного товару;
* item\_id – ідентифікатор товару, якому належить характеристика;
* property\_id – ідентифікатор характеристики продукту;

«Числова характеристика товару з плаваючою комою подвійної точності» – характеристика товару, значенням якої є число з плаваючою комою подвійної точності (наприклад, вага):

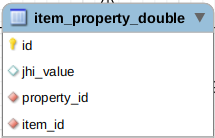


Рис.2.14. Таблиця «Числова характеристика товару з плаваючою комою подвійної точності»

* id – унікальний номер, призначений кожномій характеристиці;
* jhi\_value – значення характеристики для данного товару;
* item\_id – ідентифікатор товару, якому належить характеристика;
* property\_id – ідентифікатор характеристики продукту;

### 2.1.2. Опис зв’язків між сутностями

Між сутностями було встановлено наступні зв’язки:

* Між сутностями Регіон та Країна було встановлено зв’язок багато-до-одного. (Країна має багато регіонів, регіон відноситься лише до одної країни);
* Між сутностями Місто та Регіон було встановлено зв’язок багато-до-одного. (Регіон має багато міст, місто відноситься лише до одного ругіону);
* Між сутностями Місто та Адреса було встановлено зв’язок багато-до-одного. (Місто має багато адрес, адреса відноситься лише до одного міста);
* Між сутностями Адреса та Профіль було встановлено зв’язок багато-до-одного. (По одній адресі можуть мешкати багато користувачів);
* Між сутностями Операція та Адреса було встановлено зв’язок багато-до-одного. (Може бути багато операцій на одну адресу);
* Між сутностями Операція та Продавець було встановлено зв’язок багато-до-одного. (Може бути багато операцій з одним продавцем);
* Між сутностями Операція та Покупець було встановлено зв’язок багато-до-одного. (Може бути багато операцій з одним покупцем);
* Між сутностями Операція та Кур’єр було встановлено зв’язок багато-до-одного. (Може бути багато операцій з одним кур’єром);
* Між сутностями Операція та Товар було встановлено зв’язок багато-до-одного. (Може бути багато операцій з товаром);
* Між сутностями Паспорт та Профіль було встановлено зв’язок багато-до-одного. (У людей буває багато паспортів, при закінченні терміну дії попереднього він становиться неактивним, але в історії має зберігатись в будь-якому разі);
* Між сутностями Повідомлення та Профіль було встановлено зв’язок багато-до-одного. (Багато повідомлень відправляється одному профілю);
* Між сутностями Профіль та Позиція було встановлено зв’язок багато-до-одного. (Профіль може находитись лише на одній Позиції, на одній Позиції може бути багато Профілей);
* Між сутностями Повноваження та Позиція було встановлено зв’язок багато-до-багатьох. (Позиція може мати багато Повноважень, Повноваження може бути в багатьох Позицій);
* Між сутностями Продукт та Продукт було встановлено зв’язок багато-до-одного(Продукт може мати один батьківський, від якого успадковує базовий набір характеристик, один продукт може мати багато нащадків);
* Між сутностями Товар та Продукт було встановлено зв’язок багато-до-одного (Товар є екземпляром продукту, в одного продукту може бути багато екземплярів);
* Між сутностями Характеристика та Продукт було встановлено зв’язок багато-до-одного. (Один продукт може мати багато характеристик);
* Між сутностями Булева характеристика товару та Характеристика було встановлено зв’язок багато-до-одного. (може бути багато характеристик, які набувають значення так/ні);
* Між сутностями Стрічкова характеристика товару та Характеристика було встановлено зв’язок багато-до-одного. (може бути багато характеристик, які набувають текстового значення);
* Між сутностями Цілочисельна характеристика товару та Характеристика було встановлено зв’язок багато-до-одного. (може бути багато характеристик, які набувають цілочисельне значення);
* Між сутностями Числова характеристика товару з плаваючою комою та Характеристика було встановлено зв’язок багато-до-одного. (може бути багато характеристик, які набувають значення числа з плаваючою комою);
* Між сутностями Числова характеристика товару з плаваючою комою подвійної точності та Характеристика було встановлено зв’язок багато-до-одного. (може бути багато характеристик, які набувають значення числа з плаваючою комою подвійної точності);
* Між сутностями Булева характеристика товару та Товар було встановлено зв’язок багато-до-одного. (Товар може бути багато характеристик, які набувають значення так/ні);
* Між сутностями Стрічкова характеристика товару та Товар було встановлено зв’язок багато-до-одного. (Товар може мати багато характеристик, які набувають текстового значення);
* Між сутностями Цілочисельна характеристика товару та Товар було встановлено зв’язок багато-до-одного. (Товар може мати багато характеристик, які набувають цілочисельне значення);
* Між сутностями Числова характеристика товару з плаваючою комою та Товар було встановлено зв’язок багато-до-одного. (Товар може мати багато характеристик, які набувають значення числа з плаваючою комою);
* Між сутностями Числова характеристика товару з плаваючою комою подвійної точності та Товар було встановлено зв’язок багато-до-одного. (Товар може мати багато характеристик, які набувають значення числа з плаваючою комою подвійної точності);

## 2.2. Логічне проектування бази даних

Логічну та фізичне проектування бази даних було здійснено за допомогою MySQL Workbench. Спочатку було створено логічну модель (створено всі сутності та налаштування зв’язків між ними). Після цього тип моделі було змінено на Physical, в властивостях сутностей вибрано пункт “Extended”.

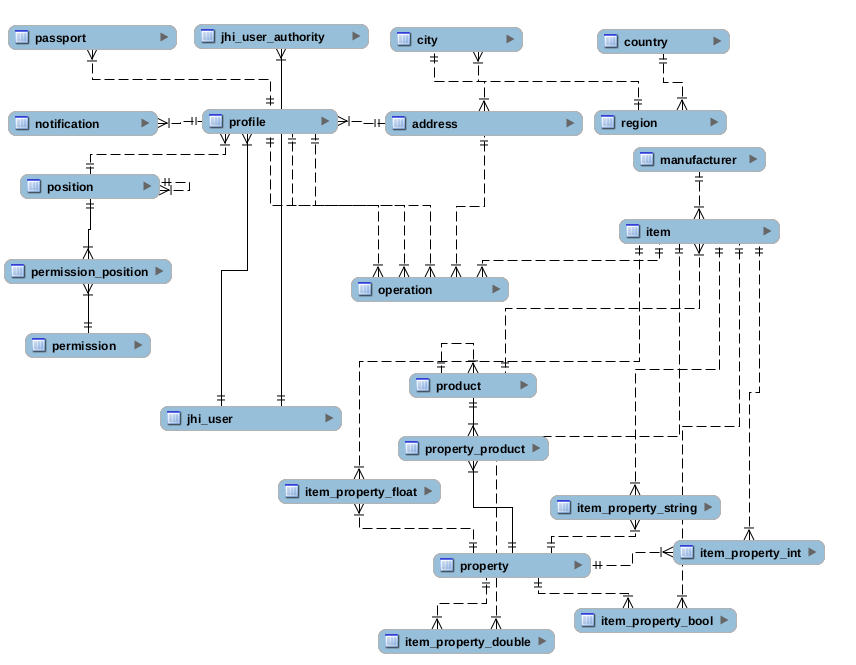


Рис.2.15. Логічна модель БД

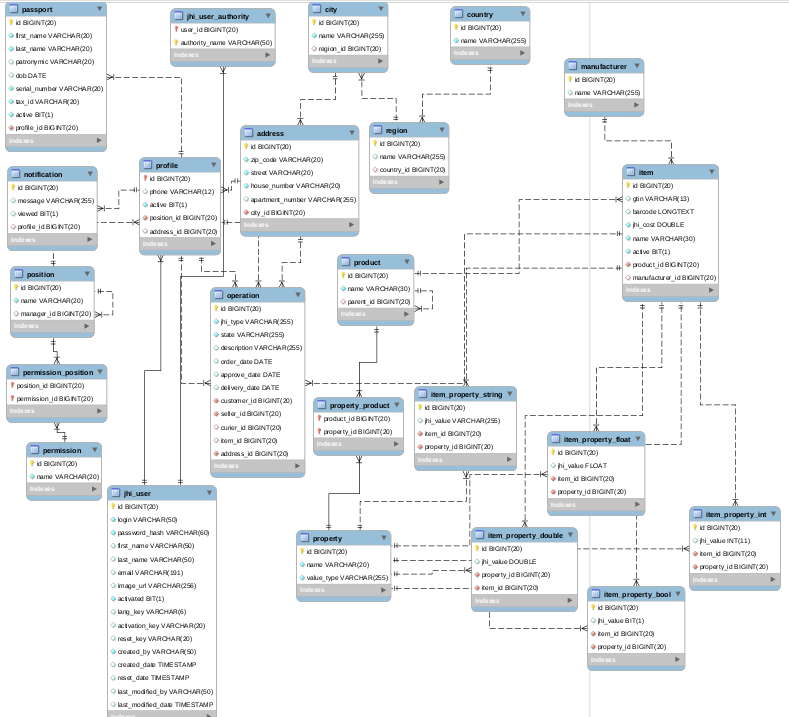


Рис.2.16. Фізична модель БД

Після створення логічної та фізичної моделей їх перенесено в середовище Microsoft SQL Server Management Studio:

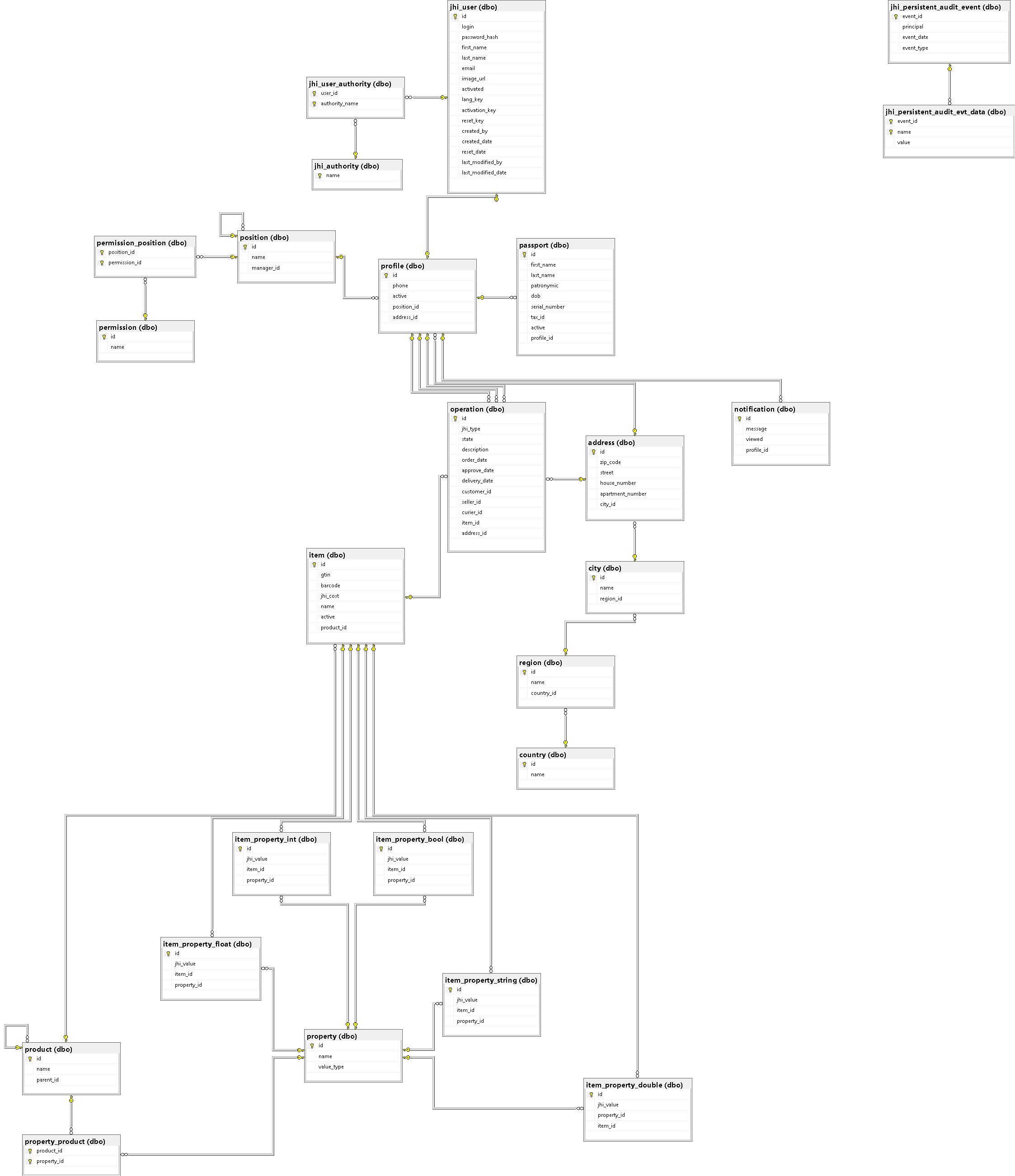


Рис.2.17. Діаграма бази даних в Microsoft SQL Server Management Studio

Для полегшення роботи з даними, надійності створено такі віртуальні таблиці :

* clientFeeList - віртуальна таблиця яка видає детальну інформацію про рахунки клієнтів за час візиту на курорт;
* clientFullInfoList - віртуальна таблиця, яка показує детальну інформацію про клієнтів зібрану зі всіх пов’язаних таблиць в базі даних;
* hotelStateList - віртуальна таблиця, яка показує детальну інформацію про готелі з зайнятості кімнат стану кімнат.

Створено такі обмеження:

* На усі атрибути (RoomId и FoodTypeId в Reservation) з вторинними ключами було накладено обмеження з умовою - не рівні NULL.
* Для всіх таблиць ідентифікатори мають властивість auto increment.

## 2.3. Проектування типових запитів і транзакцій

У роботі з БД мною було спроектовано наступні запити та транзакції:

* Запити, які додають записи в таблицях «Сlient», «Reservation», «Room», «Hotel», «RoomType», «FoodType», «PayStatus», «Employee», «EmployeeType», «ResortObject», «ResortObjectType», «ResortObjectState», «StateChange», «City».
* Запити, які редагують записи в таблицях «Сlient», «Reservation», «Room», «Hotel», «RoomType», «FoodType», «PayStatus», «Employee», «EmployeeType», «ResortObject», «ResortObjectType», «ResortObjectState», «StateChange», «City».
* Запити, які видаляють записи в таблицях «Сlient», «Reservation», «Room», «Hotel», «RoomType», «FoodType», «PayStatus», «Employee», «EmployeeType», «ResortObject», «ResortObjectType», «ResortObjectState», «StateChange», «City».
* Вибірка всієї інформації про певного клієнта (за допомогою багатотабличного запиту).
* Вибірка всієї інформації про готель(стан кімнат, мешканці).
* Вибірка інформації про відгуки клієнта стосовно обраного об’єкту.
* Вибірка інформації про рахунок та стан оплати певного клієнта.
* Вибірка інформації про працівників та їх посади.
* Вибірка інформації підйомник, його стан та тип.
* Транзакція для видалення запису з таблиці «Client» і всіх резервувань на його ім’я з обробкою можливої помилки.
* Транзакція для видалення запису з таблиці «Reservation» тільки якщо рахунок оплачений.

Також для зручності подальшого використання бази даних в програмі були створені такі збережені процедури та триггери:

* Процедура для розрахунку загального рахунку клієнта.
* Процедура для визначення власника резервування по його ID.
* Транзакція для одночасного додавання клієнта в таблицю «Client» та резервування на його ім’я в таблицю «Reservation» з обробкою можливої помилки.
* Процедура для швидкого додавання запису в таблицю «Hotel».
* Триггер для звільнення кімнати при видалення «Reservation» та заселення при створення «Reservation».
* Триггер для додавання на рахунок ціни за SkiPass.

Для бази даних використовуються індекси створені за замовчуванням Microsoft SQL Server (кластеризовані індекси що одночасно є первинними ключами). Відмова від інших індексів пов’язана з сповільненням операцій видалення, додавання та редагування полів, що є індексами, а саме ці операції є основними бізнес процесами в програмному забезпечення, виграш в пошуку і сортуванні по значущим полям буде незначною перевагою.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ РОБОТИ З БАЗОЮ ДАНИХ

* 1. Реалізація доступу до бази даних

Для реалізації доступу до даних БД я обрав технологію Hibernate — засіб відображення між об'єктами та реляційними структурами (object-relational mapping, ORM) для платформи Java. Hibernate є вільним програмним забезпеченням, яке поширюється на умовах GNU Lesser General Public License. Hibernate надає легкий для використання каркас (фреймворк) для відображення між об'єктно-орієнтованою моделлю даних і традиційною реляційною базою даних.

Метою Hibernate є звільнення розробника від значних типових завдань із програмування взаємодії з базою даних. Розробник може використовувати Hibernate як при розробці з нуля, так і для вже існуючої бази даних.

Hibernate піклується про зв'язок класів з таблицями бази даних (і типів даних мови програмування із типами даних SQL), і надає засоби автоматичної побудови SQL запитів й зчитування/запису даних, і може значно зменшити час розробки, який зазвичай витрачається на ручне написання типового SQL і JDBC коду. Hibernate генерує SQL виклики і звільняє розробника від ручної обробки результуючого набору даних, конвертації об'єктів і забезпечення сумісності із різними базами даних.

Hibernate забезпечує прозору підтримку збереження даних, тобто їхньої персистентності (англ. persistence) для «POJO»-об'єктів, себто для звичайних Java-об'єктів; єдина сувора вимога до класу, що зберігається — конструктор за умовчанням (Для коректної поведінки у деяких застосуваннях потрібно приділити особливу увагу до методів equals() і hashCode()

Як інструмент для міграцій схеми бази даних використано Liquibase. Система міграцій схеми бази даних Liquibase хороша тим, що дозволяє використовувати системи контролю версій, VCS, (наприклад, Git) для управління ревізіями бази даних програми. Говорячи більш точно, VCS містить опис змін, необхідні для міграції схеми бази даних з однієї ревізії в іншу.

Liquibase використовує так звані «чейнджсети» (changeset - набір змін), XML-код для опису операторів DDL. Вони складають файли чейнджлогов (changelog).

Приклад ченджлогу для створення таблиці Позиція:

<?xml version="1.0" encoding="utf-8"?>

<databaseChangeLog

xmlns="http://www.liquibase.org/xml/ns/dbchangelog"

xmlns:ext="http://www.liquibase.org/xml/ns/dbchangelog-ext"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.5.xsd

http://www.liquibase.org/xml/ns/dbchangelog-ext http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-ext.xsd">

<property name="now" value="now()" dbms="h2"/>

<property name="now" value="now()" dbms="mysql"/>

<property name="autoIncrement" value="true"/>

<property name="floatType" value="float4" dbms="postgresql, h2"/>

<property name="floatType" value="float" dbms="mysql, oracle, mssql"/>

<!--

Added the entity Position.

-->

<changeSet id="20190430162315-1" author="jhipster">

<createTable tableName="position">

<column name="id" type="bigint" autoIncrement="${autoIncrement}">

<constraints primaryKey="true" nullable="false"/>

</column>

<column name="name" type="varchar(20)">

<constraints nullable="false" unique="true" uniqueConstraintName="ux\_position\_name" />

</column>

<column name="manager\_id" type="bigint">

<constraints nullable="true" />

</column>

</createTable>

</changeSet>

</databaseChangeLog>

Створення зовнішніх ключів і автоматичне завантаження даних з .csv-таблиці:

<?xml version="1.0" encoding="utf-8"?>

<databaseChangeLog

xmlns="http://www.liquibase.org/xml/ns/dbchangelog"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.5.xsd">

<changeSet id="20190430162315-2" author="jhipster">

<addForeignKeyConstraint baseColumnNames="manager\_id"

baseTableName="position"

constraintName="fk\_position\_manager\_id"

referencedColumnNames="id"

referencedTableName="position"/>

<loadData encoding="UTF-8"

file="config/liquibase/positions.csv"

separator=";"

tableName="position">

</loadData>

</changeSet>

</databaseChangeLog>

Клас, створений для даної таблиці:

@Entity

@Table(name = "position")

@Cache(usage = CacheConcurrencyStrategy.*NONSTRICT\_READ\_WRITE*)

public class Position implements Serializable {

private static final long *serialVersionUID* = 1L;

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

private Long id;

@NotNull

@Size(min = 1, max = 20)

@Column(name = "name", length = 20, nullable = false, unique = true)

private String name;

@ManyToOne

@JsonIgnoreProperties("positions")

private Position manager;

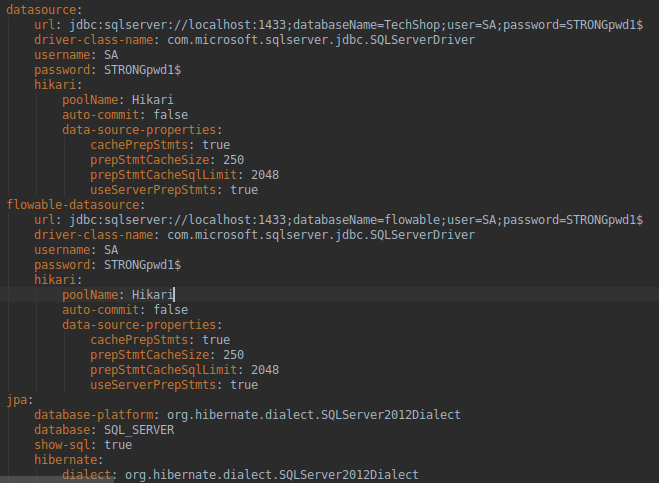
@ManyToMany(mappedBy = "positions")

@Cache(usage = CacheConcurrencyStrategy.*NONSTRICT\_READ\_WRITE*)

@JsonIgnore

private Set<Permission> permissions = new HashSet<>();

Підключення до БД в файлах ресурсів:



Також на даному етапі була створена діаграма компонентів для наглядного відображення структури майбутнього Web-додатку:

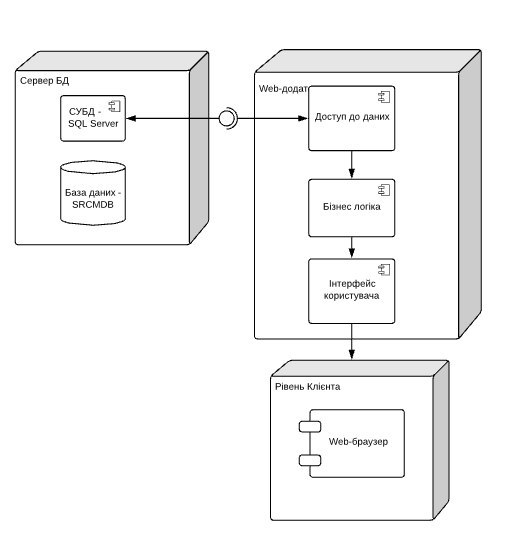


Рис.3.4. Діаграма компонент

3.2.  Реалізація функціональних характеристик системи

### 3.2.1. Авторизація користувачів

Функції авторизації реалізовані на основі JSON Web Token. JWT - це відкритий стандарт (RFC 7519) для створення токенов доступу, заснований на форматі JSON. Як правило, використовується для передачі даних для аутентифікації в клієнт-серверних додатках. Токени створюються сервером, підписуються секретним ключем і передаються клієнту, який в подальшому використовує даний токен для підтвердження своєї особи.

Для реалізації динамічного розподілення повноважень, вони серіалізуються і записуються в клейми токену в залежності від позиції користупача, після чого задається тривалість життя токену і він підписуються ключем, збереженим в ресурс-файлах проекту.

public String createToken(AuthenticationPrincipalData authenticationPrincipalData, boolean rememberMe) {

PositionGrantedAuthority positionGrantedAuthority = employeeService.getEmployeeAuthorityByEmail(authenticationPrincipalData.getAuthentication().getName());

String authorities;

try {

authorities = positionGrantedAuthority.serializeToString();

} catch (IOException e) {

e.printStackTrace();

throw new PermissionsDeniedException();

}

long now = (new Date()).getTime();

Date validity;

if (rememberMe) {

validity = new Date(now + this.tokenValidityInMillisecondsForRememberMe);

} else {

validity = new Date(now + this.tokenValidityInMilliseconds);

}

JwtBuilder builder = Jwts.*builder*();

builder.claim(*AUTHORITIES\_KEY*, authorities);

Long userId = authenticationPrincipalData.getUserId();

Authentication authentication = authenticationPrincipalData.getAuthentication();

builder.setSubject(authentication.getName());

if (userId != null) {

builder.claim(*USER\_KEY*, String.*valueOf*(userId));

}

builder.signWith(SignatureAlgorithm.*HS512*, secretKey);

builder.setExpiration(validity);

return builder.compact();

}

Після видачі токена клієну, фронт-енд запам’ятовує його в LocalStorage і передає при кожному запиті в header-ах. На стороні бек-енду повноваження перевіряється за допомогою кастомного PreAuthorize скрипта:

@PreAuthorize("hasPermission('" + ModulesConstants.*PROFILE\_MANAGEMENT* + "')")

@GetMapping("/all")

public ResponseEntity getAllProfiles(ProfileCriteria criteria, Pageable pageable) {

log.debug("REST request to get Employees by criteria: {}", criteria);

Page<EmployeeDTO> page = profileQueryService.findByCriteria(criteria, pageable);

HttpHeaders headers = PaginationUtil.*generatePaginationHttpHeaders*(page, "/api/profiles");

return ResponseEntity.*ok*().headers(headers).body(PaginationUtil.*generatePaginationBody*(page));

}

### 3.2.2. Вибірка даних

Функції вибірки даних реалізовані в усіх get-методах контролерів. Вибірка здійснюється за допомогою Spring data Specification. Приклад вибірки з класу Продукт:

*/\*\**

*\* GET /products : get all the products.*

*\**

*\** ***@param*** *pageable the pagination information*

*\** ***@param*** *criteria the criterias which the requested entities should match*

*\** ***@return*** *the ResponseEntity with status 200 (OK) and the list of products in body*

*\*/*

@GetMapping("/products")

public ResponseEntity<List<ProductDTO>> getAllProducts(ProductCriteria criteria, Pageable pageable) {

log.debug("REST request to get Products by criteria: {}", criteria);

Page<ProductDTO> page = productQueryService.findByCriteria(criteria, pageable);

HttpHeaders headers = PaginationUtil.*generatePaginationHttpHeaders*(page, "/api/products");

return ResponseEntity.*ok*().headers(headers).body(page.getContent());

}

Тут ProductCriteria містить всі критерії, по яким має здійснюватись фільтрація:

public class ProductCriteria implements Serializable {

private static final long *serialVersionUID* = 1L;

private LongFilter id;

private StringFilter name;

private LongFilter propertyId;

private LongFilter parentId;

}

Фільтрування в сервісі:

@Transactional(readOnly = true)

public Page<ProductDTO> findByCriteria(ProductCriteria criteria, Pageable page) {

log.debug("find by criteria : {}, page: {}", criteria, page);

final Specification<Product> specification = createSpecification(criteria);

return productRepository.findAll(specification, page)

.map(productMapper::toDto);

}

private Specification<Product> createSpecification(ProductCriteria criteria) {

Specification<Product> specification = Specification.*where*(null);

if (criteria != null) {

if (criteria.getId() != null) {

specification = specification.and(buildSpecification(criteria.getId(), Product\_.*id*));

}

if (criteria.getName() != null) {

specification = specification.and(buildStringSpecification(criteria.getName(), Product\_.*name*));

}

if (criteria.getPropertyId() != null) {

specification = specification.and(buildSpecification(criteria.getPropertyId(),

root -> root.join(Product\_.*properties*, JoinType.*LEFT*).get(Property\_.*id*)));

}

if (criteria.getParentId() != null) {

specification = specification.and(buildSpecification(criteria.getParentId(),

root -> root.join(Product\_.*parent*, JoinType.*LEFT*).get(Product\_.*id*)));

}

}

return specification;

}

### 3.2.3. Додавання даних

Для кожної сутності було створено контролер, сервіс з бізнес-логікою і репозиторій для забезпечення багаторівневої архітектури системи. Методи додавання даних знаходяться в методах save() сервісів відповідних сутностей. Спочатку дані передаються з форми на контролер, де перед цим перевіряються на коректність, після чого за допомогою методу методу save() репозиторія записуються в контекст. Остаточне збереження відбувається після завершення транзакції (робота зі всіма важливими данними проводиться виключно в окремих транзакціях за допомогою hibernate Transactional. Приклад додавання товару:

@Transactional

@Override

public ItemDTO save(ItemDTO itemDTO) {

log.debug("Request to save Item : {}", itemDTO);

Item item = itemMapper.toEntity(itemDTO);

item = itemRepository.save(item);

return itemMapper.toDto(item);

}

### 3.2.4. Видалення даних

Для кожної сутності було створено контролер, сервіс з бізнес-логікою і репозиторій для забезпечення багаторівневої архітектури системи. Методи видалення даних знаходяться в методах delete() сервісів відповідних сутностей. Спочатку дані передаються з форми на контролер, де перед цим перевіряються на коректність, після чого за допомогою методу методу delete() репозиторія операція записується в контекст. Остаточне видалення відбувається після завершення транзакції (робота зі всіма важливими данними проводиться виключно в окремих транзакціях за допомогою hibernate Transactional. Приклад додавання товару:

@Override

public void delete(Long id) {

log.debug("Request to delete Item : {}", id);

itemRepository.deleteById(id);

}