Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Юхимчука Владислава Сергійовича*
(ПІБ)

академічної групи *122-17-2*
(шифр)

спеціальності *122 Комп'ютерні науки та інформаційні технології*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки та інформаційні технології*
(назва освітньої програми)

на тему: *Розробка веб-сервісу для готельного бізнесу з використанням Java-сервлетів.*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Кабак Л.В.*			
розділів:				
спеціальний	*доц. Кабак Л.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-17-2* *Юхимчука Владислава Сергійовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-сервісу для готельного бізнесу з використанням Java-сервлетів.*

затверджена наказом ректора НТУ «ДП» від

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	

Завдання видав _____ *доц. Кабак Л.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Юхимчук В.С.*
(підпис) (прізвище, ініціали)

Дата видачі завдання:
Термін подання кваліфікаційної роботи до ЕК:

РЕФЕРАТ

Пояснювальна записка: 70 с., 17 рис., 0 табл., 3 дод., 20 джерела.

Об'єкт розробки: веб-додаток для автоматизації бізнес процесів готелю.

Мета кваліфікаційної роботи: веб додатку за допомогою мови програмування Java.

У вступі розглядається сучасний стан проблеми, конкретизується мета кваліфікаційної роботи, актуальність та галузь її застосування, уточняється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні веб-додатку, що допомагає більш ефективно вести бізнес.

Актуальність інформаційної системи визначається великим попитом на подібні розробки.

Список ключових слів: веб-додоаток, сайт, готельний бізнес, додаток, готель.

ABSTRACT

Explanatory note: 70 pages, 17 figures, 0 tables, 3 appendices, 20 sources.

Object of development: web application for automation of business processes of the hotel.

The purpose of the qualification work: web application using the Java programming language.

The introduction considers the current state of the problem, specifies the purpose of the qualification work, relevance and scope of its application, clarifies the task.

The first section analyzes the subject area, determines the relevance of the task and the purpose of development, formulates the task, specifies the requirements for software implementation, technology and software.

The second section analyzes the available solutions, selected platforms for development, designed and developed the program, describes the program, algorithm and structure of its operation, as well as calling and loading the program, determines the input and output data, describes the parameters of hardware.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical value is to create a web application that helps to run a business more efficiently.

The relevance of the information system is determined by the high demand for such developments.

List of keywords: WEB APPENDIX, SITE, HOTEL BUSINESS, APPENDIX, HOTEL.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ОС – операційна система;
VST – формат плагіна від компанії Steinberg;
AU –формат плагіна від компанії Apple;
AAX – формат плагіна від компанії Avid Technology;
RTAS – формат плагіна від компанії Digidesign;
DX – формат плагіна від компаній Micrisoft і Cakewalk;
NKI – формат плагіна від компанії Native Instruments;
PolyBLEP – Polynomial Bandlimited Step (поліноміальна смугасто-обмежена ступінчаста функція).

ВСТУП

Сьогодні склалося так, що інтернет, і все що з ним пов'язане, стали невід'ємною частиною нашого життя. За допомогою інтернету ми можемо автоматизувати, прискорити і полегшити виконання більшості завдань.

З кожним роком все частіше в різних галузях віддається перевага ПК-системам, а не людям. Також створення web-додатки є не просто даниною моді, а й необхідною для просування бізнесу. За допомогою сайту потенційний власник бізнесу зможе привернути величезну кількість клієнтів, а також зможе налаштувати можливість цілодобового взаємодії з клієнтами що позитивно позначиться на рентабельності бізнесу.

З розвитком туризму в нашій країні однією з головних сфер обслуговування став готельний сервіс. Сьогодні в цій галузі існує чимала конкуренція, тому багато лідерів цього бізнесу починають все більше і більше впроваджувати IT-технології для просування та реклами своїх послуг.

Відповідно з перерахованими вище аргументами, метою цього проекту стало створення сайту для підприємства. Як зразок, я вирішив вибрати готель «GoodZone». Сайт буде надавати клієнтам інформацію про готелі, а також клієнти зможуть забронювати номер, і це все без участі людини.

Для створення web-додатку я використовував різні системи управління і інструменти. Це все мені дозволило впровадити різноманітний функціонал.

При виконанні даної кваліфікаційної роботи поглиблення було зроблене над back-end складової, що дозволило досягти високої гнучкості сайту.

На back-end було викрористано мову програмування «Java», а при розробці front-end складової було використано HTML (від англ. HyperText Markup Language - «мова розмітки гіпертексту»), CSS (англ. Cascading Style Sheets - каскадні таблиці стилів) і мову програмування Java-script. До front-end можна віднести всі елементи які бачить користувач під час запуску web-сторінки.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості з предметної галузі

Сьогодні, управління бізнесом неможливо без використання інформаційних технологій. Величезний внесок несе правильний вибір компанії-розробника і виду програмного продукту для успішного впровадження IT-технологій в бізнесі. У сучасних умовах проблема вибору ІС (інформаційної системи) з специфічної задачі перетворюється в стандартну процедуру, але для багатьох керівників бізнесу вибір найефективнішої комп'ютерної системи управління підприємством є чимось новим. При цьому всьому дуже важливо максимально правильно підібрати систему, так як протягом декількох років це буде мати значний вплив на підприємство. Так правильно обрана система дозволить підвищити управління компанії і зможе мінімізувати витрати.

Існує безліч систем і підсистем ІС, наприклад:

- модуль менеджерів;
- модуль адміністрації;
- модуль резервування;
- модуль обслуговування.

Модуль менеджерів - це ІС готелю спрямована максимально зручне і раціональне управління готелем. Цей модуль включає в себе такі функції як:

- проведення розрахунку ціни проживання;
- визначення завантаження готелю;
- підготовка швидкої відповіді клієнту;
- виконання класифікації номерів по заданому пріоритету;
- збереження інформації про постояльців.

Модуль адміністрації обов'язково присутній в будь-якому готельному бізнесі. У ньому накопичується інформація про номерний фонд:

- статус номера (вільний, зайнятий, в ремонті і т.д.);
- технічний стан;
- вартість проживання;

Модуль резервування - ще одна важлива складова інформаційних систем. Цей модуль призначений для автоматизації робочих процесів які протікають на базі відділу резервування. Цей модуль включає в себе даний функціонал:

- контроль за вільними / зайнятими номерами;
- бронювання номерів;
- створення листа очікування;
- підтвердження заявок;
- зберігання інформації про попередніх постояльців номера;
- можливість онлайн оплати.

Модуль обслуговування - дозволяє стежити за станом номерів, отримувати відомості про зайнятих / вільних номерах, полегшує підготовку номерів до здачі (за рахунок тегів прибрано, потрібно прибирання і т.д.). Включає в себе функціонал:

- допомога в прогнозуванні зайнятості готелю.
- отримання нової інформації про статусах номера;

Етапи створення інформаційної системи:

1. визначення завдань, які будуть виконуватися підсистемою. Список цих завдань створює замовник;

2. створення моделі майбутньої інформаційної системи;

3. за готової моделі IT-компанія створює програмний продукт;

4. тестування продукції;

5. запуск інформаційної системи;

6. супровід. Згодом може знадобиться доопрацювання ІС.

Основні вимоги до ІС:

Інформаційні системи повинні володіти рядом вимог:

• оптимальний рівень функціональності, який дасть можливість отримувати повні відомості про клієнта і особливості його обслуговування;

• надійність, яка полягає в мінімальній кількості відмов і стабільності роботи;

• простота інтерфейсу, що дозволяє користувачам з різним рівнем підготовки працювати з інформаційними системами і технологіями в готельному підприємстві;

• гнучкість програмного продукту, що важливо для внесення змін в структуру системи і розширення її функціоналу.

1.2. Призначення розробки та галузь застосування

Призначення розробки та галузь застосування – це автоматизація інформаційної системи готельного бізнесу:

- читання таблиць СУБД;
- онлайн боронування номерів;
- реєстрація клієнта;
- особистий кабінет клієнта;
- особистий кабінет адміністратора.

1.3. Підстава для розробки

Підставою є наказ ректора Національного технічного університету «Дніпровська політехніка» № 317 від 07.06.2021р.

Завдання на кваліфікаційну роботу на тему «Розробка веб-сервісу для готельного бізнесу з використанням Java-сервлетів.».

1.4. Постановка завдання

Завданням кваліфікаційної роботи є проектування інформаційної системи готельного бізнесу. Програмне забезпечення призначене для надання універсального інструменту доступу до даних підприємства.

Програма повинна реалізувати наступні функції:

- отримання доступу до окремих даних;
- реєстрація клієнта;
- реєстрація акаунта адміністратора;
- пошук вільних номерів за заданими критеріями;
- онлайн боронування номерів.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Вимоги до програми – це проектування автоматизованої інформаційної системи готельного бізнесу:

- читання таблиць СУБД з даними про клієнтів, стан готельних номерів;
- створення особистого кабінету клієнта та адміністратора;
- пошук інформації про готельні номера за заданим критерієм;
- онлайн бронювання та оплата номера;

Головне вікно програми повинно давати змогу для вибору рівня доступу.

Рівнем доступу будуть виступати:

- адміністратор;
- клієнт.

Для вибору рівня доступу до програми потрібно буде ввести логін та пароль, заздалегідь зареєстрованим клієнтом або адміністратором готелю.

1.5.2. Вимоги до інформаційної безпеки

Виходячи з практичних задач, які повинен вирішувати даний програмний засіб, потрібно обмежити доступ клієнта до інформації яка доступна адміністратору готелю.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для функціонування системи необхідна клієнтська персональна ЕОМ з наступними мінімальними характеристиками:

- процесор класу Intel Core i3 2 ядра 3,9Ггц;
- монітор;
- не менше 1Гб ОЗУ;
- клавіатура;
- маніпулятор «миша».

1.5.4. Вимоги до інформаційної та програмної сумісності

Для інформаційної та програмної сумісності необхідна наявність наступних програм та систем:

- будь-який встановлений браузер;
- встановлена операційна система Microsoft Windows 7/8/10.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Призначення розробки – автоматизація процесів готельного бізнесу за допомогою веб-додатку.

Розроблений в ході кваліфікаційної роботи веб-додаток має наступні функції та призначення:

- зручний та інтуїтивно зрозумілий інтерфейс для клієнтів;
- клієнт має змогу виконати бронювання кімнат он-лайн;
- клієнт може швидко та зручно знайти інформацію щодо готелю, його номерів, тощо;
- веб-додаток також має функціонал для адміністраторів готелю;
- адміністратор готелю має змогу переглядати заявки на бронювання, переглядати інформацію щодо статусу номерів, статусу оплати номерів, тощо;
- адміністратор може переглядати інформацію про зареєстрованих клієнтів.

2.2. Опис застосованих математичних методів

В ході кваліфікаційної роботи було використано математичний метод який допомагає аналізувати інформацію та складати рейтинг популярності номерів.

2.3. Опис використаних технологій та мов програмування

Програма розроблена за допомогою мови програмування Java в середовищі програмування IntelliJ IDEA 2020. Java часто використовується в

якості мови для розробки веб-застосунків. Java – це один із найпопулярніших мов програмування, він широко використовується для розробки веб-застосунків протягом останніх 20-ти років.[1] Такого успіху даній мові програмування забезпечує її гнучкість та зручність під час розробки.

Універсальність досягається за допомогою віртуальної машини Java(JVM - Java Virtual Machine).

В багатьох мовах програмування один і той самий код може по-різному працювати на різних пристроях чи платформах. Проблема криється в логіці роботи мов програмування. В багатьох мовах програмування під-час компіляції код програми переводиться в інший код який і може працювати по-різному на різних пристроях. JVM грає роль проміжного рівня – із програми на Java вона робить код, котрий при компіляції буде давати однаковий результат незалежно від того на якій платформі чи пристрої він буде запущений.[2]

Також Java має дуже великий спектр застосувань. Наприклад Java допомагає в вирішенні багатьох задач зв’язаних із:

- розробкою мобільних додатків на Android;
- розробкою API для роботи з базами даних;
- цифровою обробкою зображень;
- програмування мережевих задач.

В якості основної технології було використана технологія Java Servlets.

2.4. Опис структури системи та алгоритмів її функціонування

Структура проекту складається з багатьох пакетів (Packege) в яких знаходяться класові компоненти сайту та інтерфейси. На рис. 2.1, рис. 2.2 та рис. 2.3 зображена структура проекту:

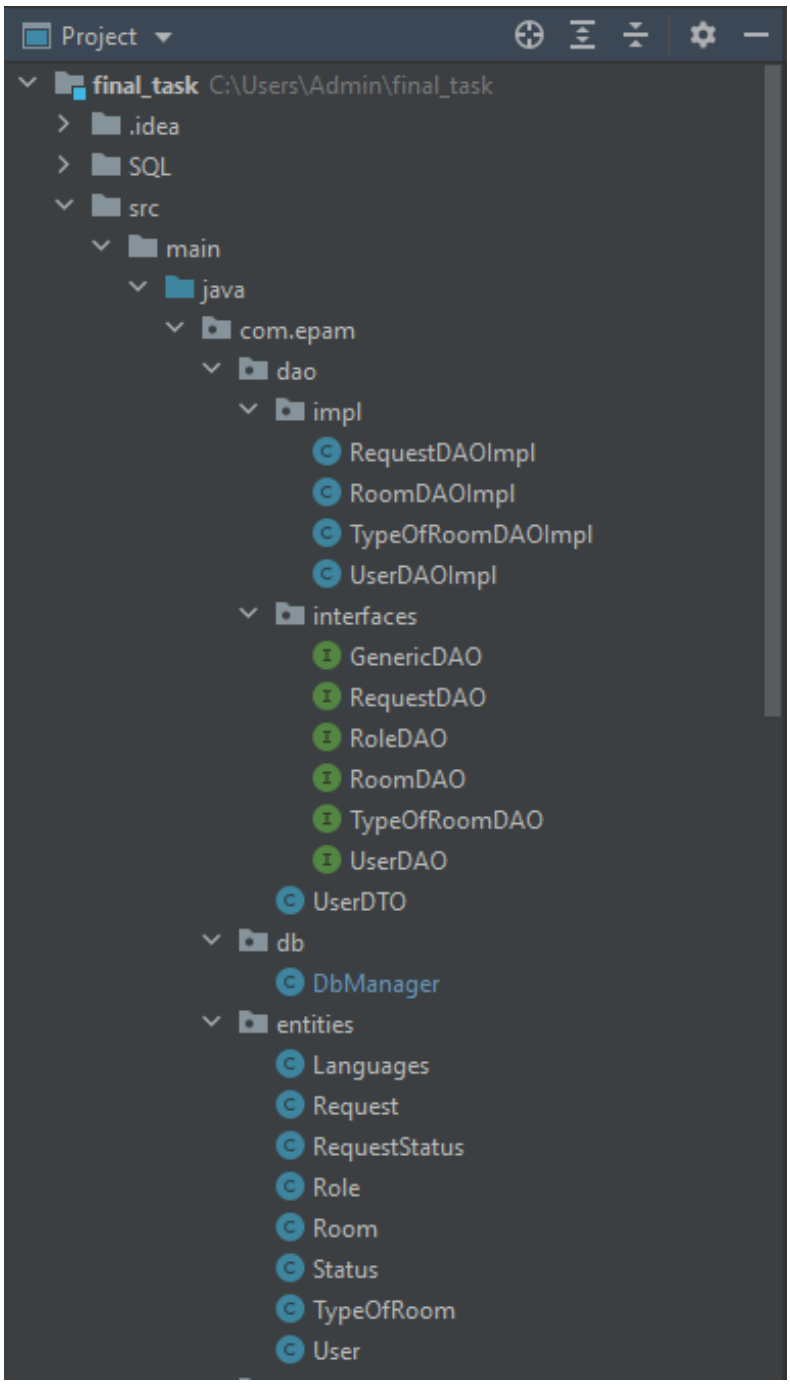


Рис. 2.1. Структура файлів проекту

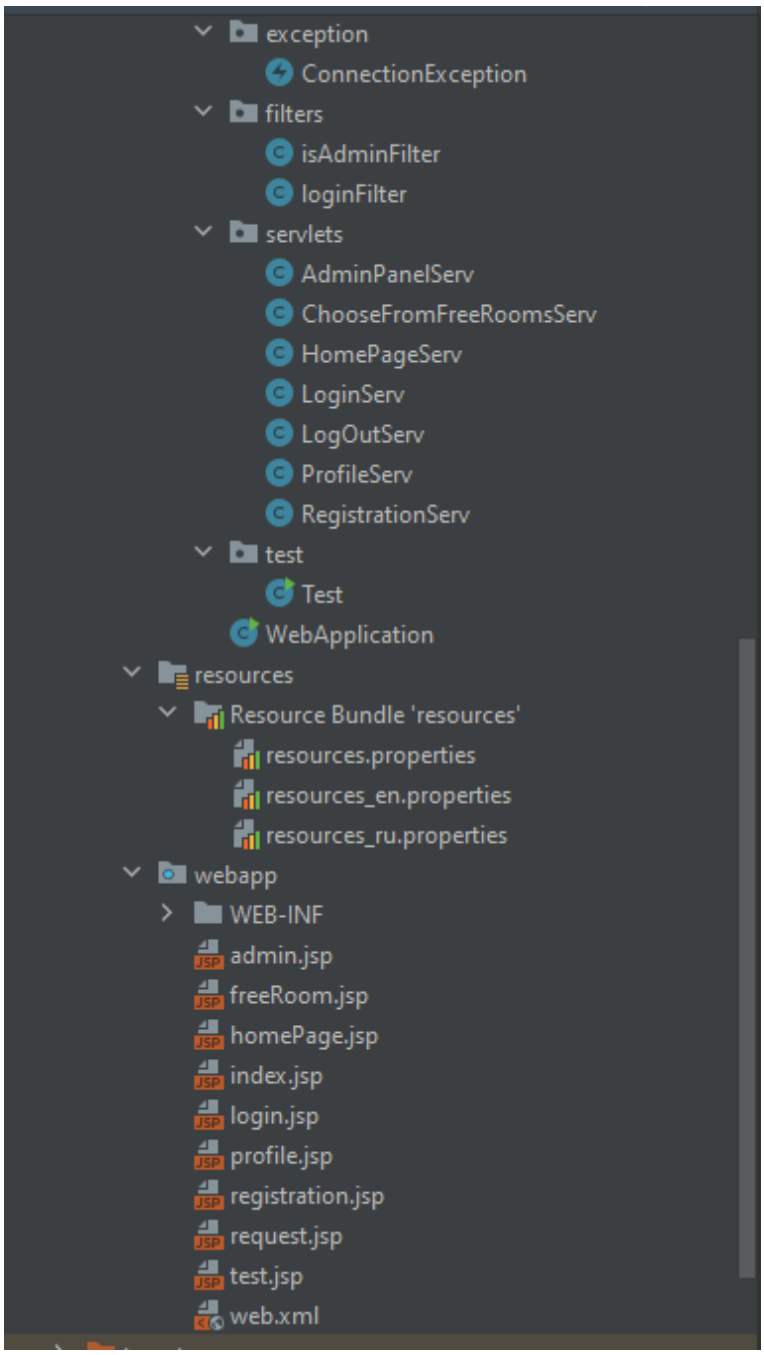


Рис. 2.2. Структура файлів проекту

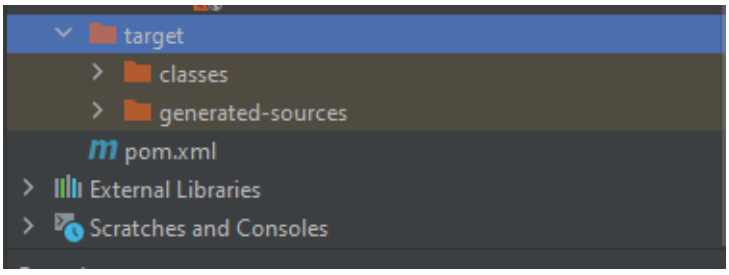


Рис. 2.3. Структура файлів проекту

Опис файлів структури:

1. Package entities: в цьому пакеті знаходяться суті проекту. Сутність – це реальний або представлений об’єкт, інформація про який повинна зберігатися та бути доступною.

2. Package interfaces: в цьому пакеті знаходяться інтерфейси проекту. Інтерфейс в мові програмування Java - це абстрактний тип, який використовується для визначення поведінки, яке повинні реалізовувати класи. Вони схожі на протоколи. Інтерфейси оголошуються з використанням ключового слова interface і можуть містити тільки сигнатуру методу і оголошення констант.[3] Серед усіх інтерфейсів є головний інтерфейс із назвою GenericDAO. Він включає в себе усі можливості які також характерні для інших інтерфейсів. Інші інтерфейси успадковуються від GenericDAO. Успадкування – це одне із ключових аспектів об'єктно-орієнтованого програмування за допомогою якого можна розширити функціонал вже наявних класів за рахунок додавання нового функціоналу або зміни старого. В цьому випадку є головний інтерфейс GenericDAO який включає в себе запрограмовані дії які також властиві для інших інтерфейсів, наприклад RoomDAO, але окрім цих дій RoomDAO має свої унікальні дії.

3. Package impl: в цьому пакеті знаходяться імплементації інтерфейсів.

4. Package db: в цьому пакеті знаходиться клас DbManager. Цей клас відповідає за роботу пов’язану із базами даних.

5. Package exception: в цьому пакеті знаходиться клас який відповідає за роботу із винятками.

6. Package filters: в цьому пакеті знаходяться класи які відповідають за фільтрацію дій користувача. Наприклад клас loginFilter відповідає за те, щоб не зареєстрований користувач не зміг отримати інформацію яка доступна тільки зареєстрованим користувачам.

7. Package servlets: в цьому пакеті знаходяться класи-сервлети які

відповідають за основні дії на сайті. Java Servlet API — стандартизований API для створення динамічного контенту до веб-сервера, використовуючи платформу Java. Сервлети — аналог технологій PHP, CGI і ASP. NET. Сервлет може зберігати інформацію між багатьма транзакціями, використовуючи HTTP кукіз, сесії або через редагування URL.[4] Servlet API, що міститься в пакеті javax. servlet, описує взаємодію веб-контейнера і сервлета. Веб-контейнер — це компонент веб-сервера, що створений для взаємодії з сервлетами. Він відповідає за управління життєвим циклом сервлетів, перетворення URL у певний сервлет та забезпечення того, щоб клієнт, який зробив URL запит, мав відповідні права доступу.[5] Наприклад RegistrationServ оброблює адресу "/registration". Він отримує інформацію від користувача на формі реєстрації після чого за допомогою sql записує інформацію до бази даних.[6]

8. Package test: в цьому пакеті знаходиться клас який використовувався для внутрішнього тестування програмного застосунку.

9. Package resources: в цьому пакеті знаходяться текстові компоненти для веб-додатку.

10. Package webapp: в цьому пакеті знаходяться компоненти які пов’язані із front-end складовою.

11. Package WEB-INF: в цьому пакеті знаходяться компоненти пов’язані із бібліотекою JSTL. JSP за замовчуванням дозволяє вбудовувати код на java в розмітку html. Однак іноді використання стандартних способів для ряду операцій, наприклад, для виведення на сторінку елементів зі списку і т.д., може бути кілька громіздким. Щоб спростити вбудовування коду java в JSP була розроблена спеціальна бібліотека - JSTL. JSTL (JSP Standard Tag Library) надає теги для базових завдань JSP (цикл, умовні вирази і т. д.)

12. Package directive: в цьому пакеті знаходиться компонент який вказує шлях до кореневих файлів бібліотеки JSTL.

13. Package lib: в цьому пакеті знаходиться кореневий файл бібліотеки

JSTL.

14. Admin. jsp, freeRoom. jsp, homepage. jsp, index. jsp, login. jsp, profile. jsp, registration. jsp, request. jsp, test. jsp, web. xml – це файли які відповідають за відображення веб-сторінок.

15. pom. xml - файл в якому знаходяться всі maven dependencies які потрібні для коректної роботи сайту.

В даній кваліфікаційній роботі було використано базу даних. Базу даних було розроблено за допомогою програми MySQL Workbench. База даних відноситься до реляційних. Реляційна база даних — база даних, заснована на реляційній моделі даних. Для роботи з реляційними БД застосовують реляційні СКБД. Інакше кажучи, реляційна база даних — це база даних, яка сприймається користувачем як набір нормалізованих відношень різного ступеня.

Реляційна база даних є сукупністю елементів даних, організованих у вигляді набору формально описаних таблиць, з яких дані можуть бути доступними або повторно зібрані багатьма різними способами без необхідності реорганізації таблиць бази даних. На рис. 2.4 зображена модель бази даних яка була розроблено для даної кваліфікаційної роботи.

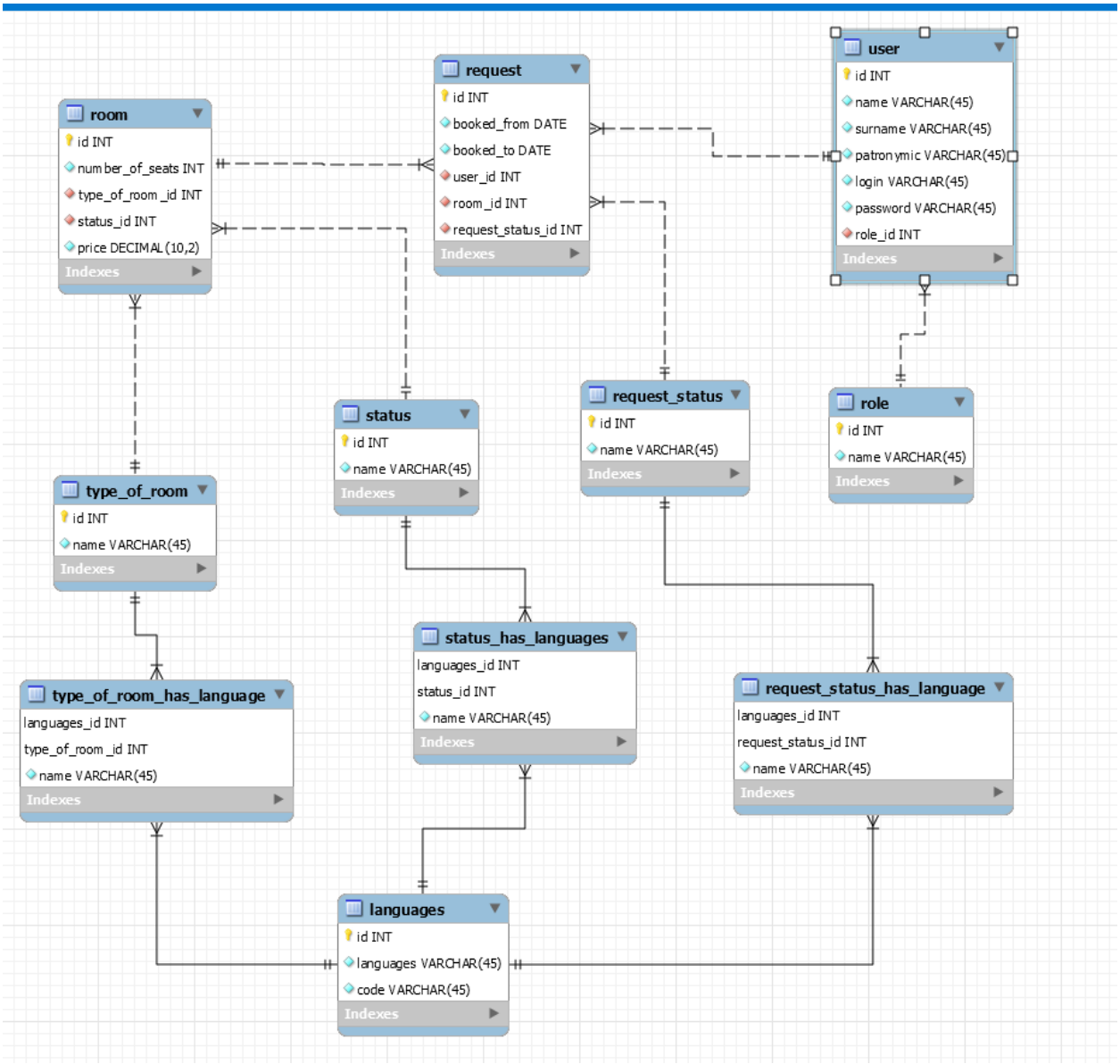


Рис. 2.4. Структура бази даних

Згідно з сутями було розроблено базу даних, таблиці якої бережуть дані про:

1. Room – зберігає інформацію про кімнати готелю. Id – індикаційний номер кімнат, number_of_seats – число яке означає на яку кількість людей призначена кімната, type_of_room_id – це ключове поле(КП) завдяки якому ми можемо отримати інформацію з таблиці type_of_room, status_id – КП за допомогою якого ми отримаємо інформацію з таблиці status, price – число яке відображає вартість кімнати за добу.

2. User – зберігає інформацію про зареєстрованих користувачів. Id -

індикаційний номер кожного зареєстрованого клієнта, name – ім’я клієнта, surname – прізвище клієнта, patronymic – ім’я по-батькові клієнта, login – видуманий користувачем логін для входу, password – видуманий користувачем пароль для входу, role_id – КП задля отримання інформації з таблиці role.

3. Request – зберігає інформацію про заявки які користувач робить для бронювання кімнати. Boocked_from – це поле зберігає інформацію про дату з якої клієнт хоче забронювати номер, Boocked _to - це поле зберігає інформацію про дату по яку клієнт буде бронювати номер, user_id – КП що дозволяє отримати інформацію з таблиці user, room_id – КП завдяки якому ми можемо отримати інформацію з таблиці room, request_status_id – КП за допомогою якого ми можемо отримати інформацію з таблиці request_status.

4. Таблиці type_of_room, type_of_room_has_language, status , status_has_language, request_status, request_status_has_language, languages використовуються для локалізації веб-додатку.

5. Type_of_room – використовується для збереження назв видів номерів;

6. Status – використовується для збереження назв стану номерів (вільний, заборонований, занятий, недоступний);

7. Request_status – використовується для збереження назв статусів оплати заявок (оплачена, не оплачена);

8. Role – зберігає інформацію про ролі користувачів на сайті (адміністратор , звичайний користувач).

Для виконання даної кваліфікаційної роботи було також використано Git. Це розподілена система керування версіями файлів та спільної роботи. Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів. На рис. 2.5

зображена схема розробки проекту[7].

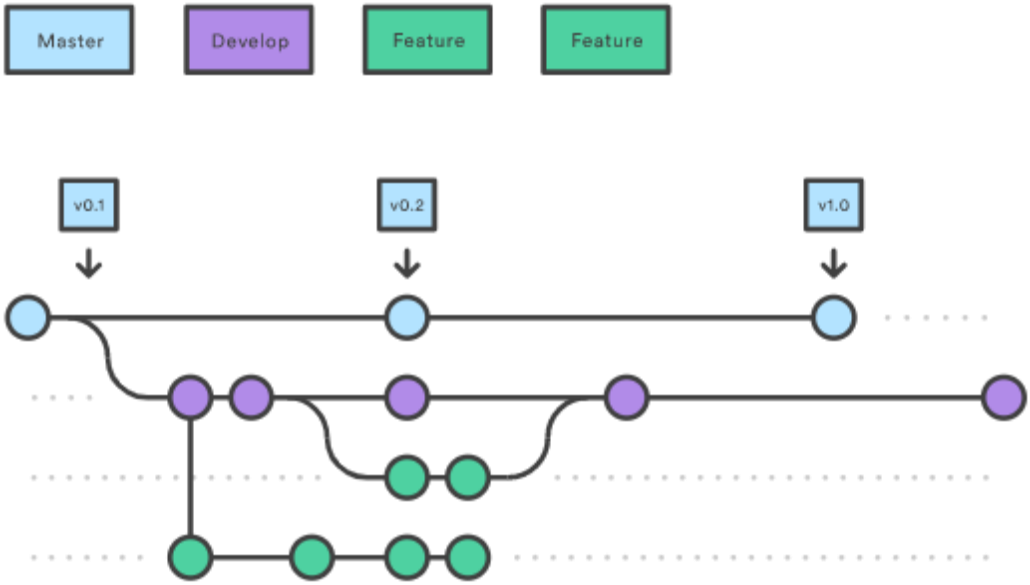


Рис. 2.5. Git структура

Структура проекту включає в себе Master branch – це основний проект який доступний для усіх учасників проекту. Кожний учасник проекту створює побічні гілки на яких він працює над своєю складовою продукту. Після виконання завдань учасник робить merge request, і після цього його зміни будуть додані до основного проекту[8].

При створенні цього веб-додатку було використано Apache Maven. Це фреймворк для автоматизації зборки проектів на базі опису їх структури в файлах на мові POM.

Maven забезпечує декларативну, а не імперативну збірку проекту. У файлах опису проекту міститься його специфікація, а не окремі команди виконання.[9] Всі завдання по обробці файлів, описані в специфікації, Maven виконує за допомогою їх обробки послідовністю вбудованих і зовнішніх плагінів.

Maven використовується для побудови і управління проектами, написаними на Java, C #, Ruby, Scala, та іншими мовами[10].

Серед примітних альтернатив - система автоматичного складання Gradle,

побудована на принципах Apache Ant і Maven, але використовує спеціалізований DSL на Groovy замість POM-конфігурації. Разом з Maven я використовував Maven dependencies. Найголовніші з них:

1. Mysql-connector-java – завдяки якому було підключено базу даних до проекту;

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.16</version>
</dependency>
```

2. Liquibase-core – це плагін який потрібен для управління liquibase з Maven;

```
<dependency>
  <groupId>org.liquibase</groupId>
  <artifactId>liquibase-core</artifactId>
  <version>3.8.4</version>
</dependency>
```

3. Jaxb-api – створений для використовування маршалінгу. Маршалінг (marshalling) - це процес перетворення об'єктів Java в документи XML. Unmarshalling - це процес читання документів XML в об'єкти Java. Клас JAXBContext забезпечує точку входу клієнта в API JAXB. Він надає API для маршалінга, демаршалінга (unmarshal) і перевірки[11];

```
<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.3.1</version>
</dependency>
```

4. Lombok - проект по додаванню додаткової функціональності в Java за допомогою зміни вихідного коду перед Java компіляцією. Він дозволяє

відмовитись від багатослівності в Java, бо додає велику кількість анотацій за допомогою яких можна дуже легко скоротити кількість строк коду, та покращити читаність коду[12].

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.8</version>
    <scope>provided</scope>
</dependency>
```

5. Log4j – це дуже корисний плагін який дозволяє дуже просто налаштувати логування, що допоможе виявляти проблеми під час розробки та адміністрування веб-додатку.

```
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
</dependency>
```

Також для виконання даної дипломної роботи було використано Java Servlets. Java Servlet API - стандартизований API, призначений для реалізації на сервері і роботи з клієнтом по схемі запит-відповідь. На рис. 2.5 та рис. 2.6 зображено схему роботи Java сервлетів.

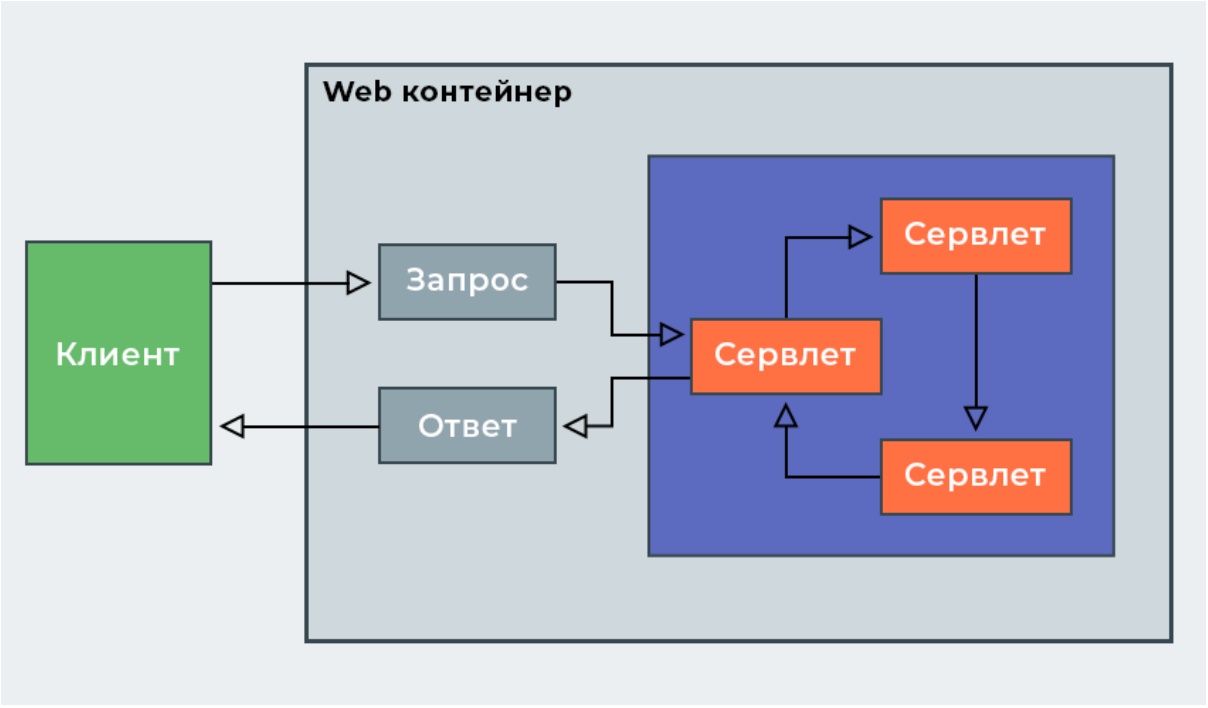


Рис. 2.5 Схема роботи Java сервлетів

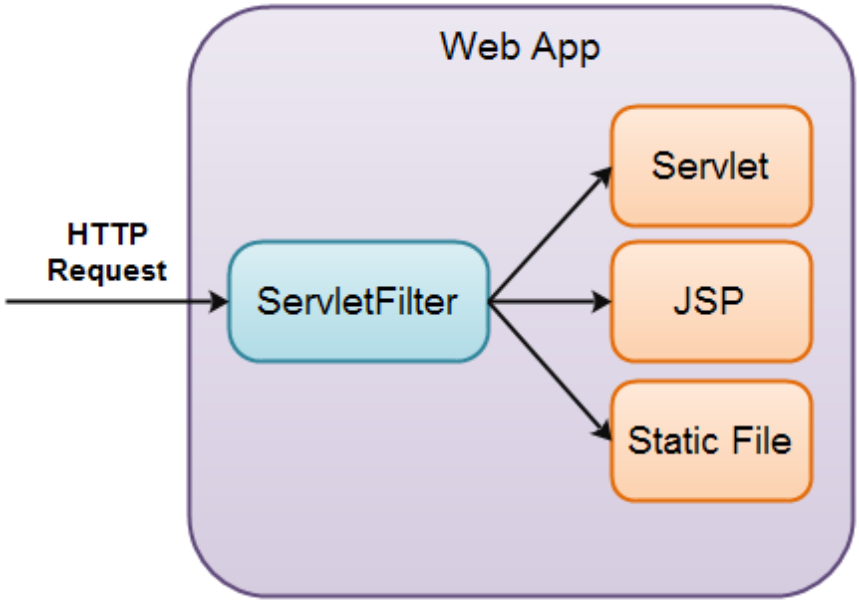


Рис. 2.6 Схема роботи Java сервлетів

Також існує такий термін як життєвий цикл сервлетів який зображено на рис. 2.7.

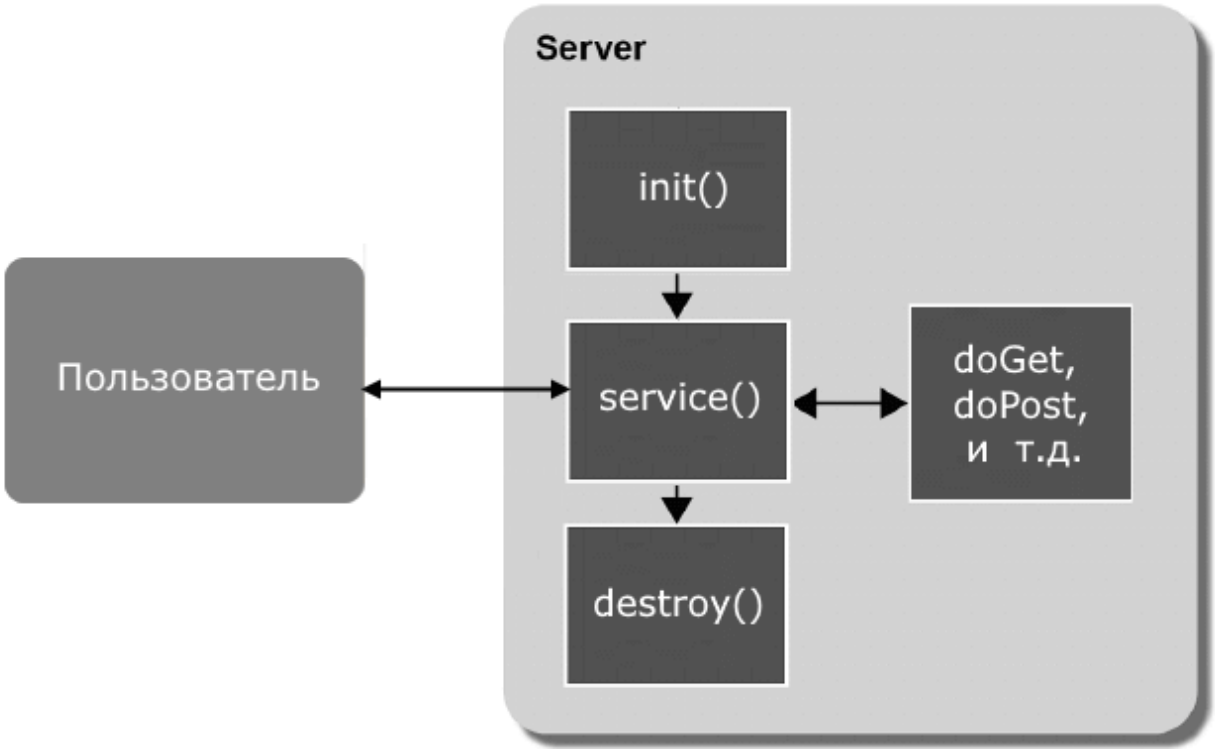


рис. 2.7. Життєвий цикл сервлетів

Сервлет складається з 2-х частин – методів doGet та doPost. DoGet - опрацьовує запит Get (отримання даних), doPost - опрацьовує запит Post (відправка даних). Також важливою складовою є методи init() та destroy(). Метод init() викликається коли створюється об’єкт сервлету, а метод destroy() викликається коли об’єкт сервлету довгий час не використовується (до нього нема ніяких запитів), або якщо відбувається завершення роботи движка сервлетів, то движок сервлетів вивантажує з пам'яті всі створені екземпляри сервлетів. Однак до вивантаження сервлету з пам'яті у сервлету викликається метод destroy ()[13].

В ході написання кваліфікаційної роботи було створено наступні сервлети:

1. Сервлет Registration – цей сервлет оброблює адресу "/registration"

та виконує таку важливу роль, як реєстрація на сайті. У методі doGet відображається jsp сторінка registration.jsp.

```
@Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    req.getRequestDispatcher("registration.jsp").forward(req, resp);
  }
```

У методі doPost отримується інформація з форми та записується до бази даних.

```
@Override
  protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String login = req.getParameter("Login");
    String password = req.getParameter("Password");
    String name = req.getParameter("Name");
    String surname = req.getParameter("Surname");
    String patronymic = req.getParameter("Patronymic");

    if (login != null && password != null && name != null && surname != null && patronymic != null) {
      User userCe = User.builder()

.login(login).name(name).surname(surname).patronymic(patronymic).password(password).build();
      userDAO.createNew(user);
    } else {
      req.getRequestDispatcher("registration.jsp").forward(req, resp);
      return;
    }
    req.getRequestDispatcher("homePage.jsp").forward(req, resp);
```

2. Сервлет Login – цей сервлет оброблює адресу "/login". У методі doGet відображається jsp сторінка login.jsp.

```
@Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    req.getRequestDispatcher("login.jsp").forward(req, resp);
  }
```

У методі doPost отримується інформація з форми та звіряється з інформацією із бази даних. В разі успішної звірки даних створюється сесія куди записується дані користувача та інші дані які можуть бути корисними під

час роботи програми. Сесія буде активною весь час до моменту коли користувач не вийде з профілю[14]. Після чого користувач відправляється на сторінку особистого кабінету. В разі неуспішної звірки – користувач потрапляє знову на сторінку логіну.

```
@Override
  protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String login = req.getParameter("Login");
    String password = req.getParameter("Password");
    User = userDAO.findByName(login).orElseThrow(RuntimeException::new);
    if (user.getLogin().equals(login) && user.getPassword().equals(password)) {
      HttpSession session = req.getSession();
      session.setAttribute("login", user.getLogin());
      session.setAttribute("role", user.getRole().getName());
      session.setAttribute("roleId" , user.getRole().getId());
      session.setAttribute("user", user);
      session.setAttribute("name", user.getName());
      session.setAttribute("surname", user.getSurname());
      session.setAttribute("patronymic", user.getPatronymic());
      session.setAttribute("id" , user.getId());
      PrintWriter out = resp.getWriter();
      resp.sendRedirect("/profile");
      out.close();
    } else {
      resp.sendRedirect("login.jsp");
    }
  }
```

3. Сервлет Profile - цей сервлет оброблює адресу /profile". У методі doGet отримується інформація про користувача з вже створеної сесії та передається на форму. Після чого користувач відправляється на сторінку profile.jsp[15].

```
@Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    req.setCharacterEncoding("UTF-8");
    HttpSession session = req.getSession();
    User = (User) session.getAttribute("user");
    req.setAttribute("login", user.getLogin());
```

```
            RequestDAOImpl = new RequestDAOImpl();
            List<Request> requests = new ArrayList<>();
            requests = requestDAOImpl.findRequestByUserId(user.getId());
            requests.forEach(requ -> requ.getRoom().setPrice(new BigDecimal(
                ((requ.getBookedTo().toEpochDay() -
requ.getBookedFrom().toEpochDay())*requ.getRoom().getPrice().longValue(
))
            )));
            req.setAttribute("requests" , requests);
            req.setAttribute("roleId" , session.getAttribute("roleId"));
            req.getRequestDispatcher("profile.jsp").forward(req, resp);
        }
```

На формі особистого кабінету користувача окрім інформації про самого користувача також є інформація про його заявки. У особистому кабінеті користувач може оплатити бронювання номеру, за це і відповідає метод doPost[16].

```
@Override
   protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
     Longreques tBookId = Long.valueOf(req.getParameter("requestBookId"));
     requestDAO.updateInformationAboutPayment(requestBookId);
     resp.sendRedirect("/profile");
   }
```

4. Сервлет ChooseFromFreeRooms. Також користувач може серед вільних номерів забронювати номер. У методі doGet за допомогою методу findFreeRooms() отримується інформація з бази даних про вільні кімнати та передається на форму. Після чого клієнта відправляється на форму freeRoom.jsp.

```
    @Override
      protected void doGet(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
            RoomDAOImpl = new RoomDAOImpl();
            TypeOfRoomDAOImpl typeOfRooms = new
TypeOfRoomDAOImpl();
            List<Room> rooms = roomDAOimpl.findFreeRooms();
            req.setAttribute("rooms" , rooms);
            req.getRequestDispatcher("freeRoom.jsp").forward(req, resp);
        }
```

В методі doPost було використано патерн Builder. Він є патерном

створення об'єктів (creational pattern). Суть його полягає в тому, щоб відокремити процес створення деякого складного об'єкта від його уявлення. Таким чином, можна отримувати різні уявлення об'єкта, використовуючи один і той же "технологічний" процес. За допомогою цього патерну я створюю об’єкт request, після чого записую його до бази даних[16].

```
@Override
   protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
      String typeOfRoom = req.getParameter("typeOfRoom");
      LocalDate dateFrom = LocalDate.parse(req.getParameter("dateFrom"));
      LocalDate dateTo = LocalDate.parse(req.getParameter("dateTo"));
      BigDecimal price = new BigDecimal(req.getParameter("price"));
      Long roomId = Long.valueOf(req.getParameter("bookId"));
      HttpSession session = req.getSession();
      User = (User) session.getAttribute("user");
      Long userId = user.getId();
      Request = Request.builder()
            .bookedFrom(dateFrom)
            .bookedTo(dateTo)
            .room(Room.builder().id(roomId).build())
            .user(User.builder().id(userId).build())
            .build();
      requestDAO.createNew(request);
      roomDAO.updateRoomStatusById(roomId);
      System.out.println(request);
      req.getRequestDispatcher("profile.jsp").forward(req, resp);
   }
```

5. Сервлет LogOut – відповідає за те, щоб користувач зміг вийти з акаунту. Метод doPost – нічого не робить, бо сервлет не відображає жодну з форм. Метод doGet – завершує сесію та відправляє користувача на сторінку логіну[17].

```
@Override
   protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
      super.doGet(req, resp);
   }

   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse
```

```
resp) throws ServletException, IOException {
        HttpSession session = req.getSession();
        session.invalidate();
        req.getRequestDispatcher("login.jsp").forward(req, resp);
    }
```

6. Сервлет AdminPanel – цей сервлет оброблює адресу "/admin". У методі doGet за допомогою sql запиту отримується інформація про зареєстрованих клієнтів та їх заявки та передає цю інформацію на форму. Після чого відправляє користувача на форму admin.jsp. Так як це інформаційна сторінка, то метод doPost нічого не робить[18].

```
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
        RequestDAOImpl = new RequestDAOImpl();
        List<Request> requests;
        requests = requestDAOImpl.findAll();
        req.setAttribute("request", requests);
        Map<UserDTO, Long> counted = requests.stream()
            .collect(Collectors.groupingBy(request -> new
UserDTO(request.getUser().getId(),
                    request.getUser().getName(),
request.getUser().getSurname(),
                    request.getUser().getPatronymic()),
Collectors.counting()));
        List<UserDTO> collect = counted.keySet().stream().map(userDTO
-> new UserDTO(userDTO.getId(),
                userDTO.getName(), userDTO.getSurname(),
                userDTO.getPatronymic(),
counted.get(userDTO))).collect(Collectors.toList());
        req.setAttribute("collect" , collect);
        req.getRequestDispatcher("admin.jsp").forward(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
        super.doPost(req, resp);}
```

Також щоб неавторизований користувач не зміг перейти на сторінку бронювання кімнат або на інші сторінки окрім головної я використовував фільтри. В даному випадку я перевіряю чи є активна сесія, якщо її нема, то це

означає, що користувач не авторизований тому треба перенаправити його на сторінку логіна.

```
@WebFilter({"/profile" , "/admin" , "/order"})
public class loginFilter implements Filter {
   @Override
   public void init(FilterConfig filterConfig) throws ServletException {

   }

   @Override
   public void doFilter(ServletRequest, ServletResponse, FilterChain
filterChain) throws IOException, ServletException {
      HttpServletRequest request = (HttpServletRequest) servletRequest;
      HttpServletResponse response =  (HttpServletResponse) servletResponse;
      HttpSession session = request.getSession(false);

      String loginURI = request.getContextPath() + "/login";
      boolean loggedIn = session!=null && session.getAttribute("login") !=
null  && session.getAttribute("role") != null;
      boolean loginRequest = request.getRequestURI().equals(loginURI);

      if(loggedIn || loginRequest){
         filterChain.doFilter(request , response);
      }else {
         response.sendRedirect(loginURI);
      }
   }
```

Щоб користувач який не є адміністратором не зміг потрапити на сторінку адміністратора я також зробив фільтр. З сесії я отримую інформацію про роль користувача і перевіряю чи вона адміністратор. Якщо так, то відправляю користувача на сторінку адміністратора, якщо ні, то до особистого кабінету користувача[19].

```
@WebFilter("/admin")
public class isAdminFilter implements Filter {
   @Override
   public void init(FilterConfig filterConfig){

   }

   @Override
```

```
    public void doFilter(ServletRequest, ServletResponse, FilterChain filterChain) throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) servletRequest;
        HttpServletResponse response = (HttpServletResponse) servletResponse;
        HttpSession session = request.getSession(false);

        String role = String.valueOf(session.getAttribute("role"));
        if (role != null && role.equals("Admin")) {
            filterChain.doFilter(request, response);
        } else {
            response.sendRedirect("/profile");
        }
    }
```

Також для швидкого пошуку інформації на сторінці для бронювання вільних номерів за допомогою Java-script створено пошуковий фільтр[20].

```
<script>
    function findByTypeOfRoom(cells) {
        var selec = document.getElementById("myInput");
        var table = document.getElementById("table");
        for (var i = 1; i < table.rows.length; i++) {
            if (selec.value == table.rows[i].cells[cells].innerText.trim()) {
                table.rows[i].style.display = "";
            } else {
                table.rows[i].style.display = "none";
            }
        }
    }
    function reset() {
        var table = document.getElementById("table");
        for (var i = 1; i < table.rows.length; i++) {
            table.rows[i].style.display = "";
        }
    }
</script>
```

2.5. Обґрунтування та організація вхідних та вихідних даних програми

В якості вхідних даних виступають дані які вводить користувач, наприклад під час реєстрації користувач заповнює форму в якій логін, пароль, ім’я, прізвище, ім’я по-батькові – це дані які вводить користувач.

Вихідними даними є дані які користувач отримує з форми, наприклад з форми реєстрації користувач отримує дані про види кімнат, на яку кількість людей розрахована кімната та її вартість за день.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Веб додаток працює в будь-якому браузері на комп’ютері та в будь-якому браузері на телефоні чи інших пристроях.

2.6.2. Використані програмні засоби

Даний веб-застосунок було розроблено для будь-яких пристроїв де э браузер та стабільне інтернет з’єднання.

2.6.3. Виклик та завантаження програми

Для запуску веб-додатку потрібно завантажити будь-який браузер, після чого створити нове вікно та знайти сайт у будь-якій з пошукових систем.

2.6.4. Опис інтерфейсу користувача

На рис. 2.8 зображено інтерфейс стартової сторінки. У майбутньому її можна буде якось змінити. На цій сторінці є дві кнопки: Enter та Registration. Натиснувши на першу кнопку користувач має змогу перейти на сторінку для авторизації. Натиснувши на іншу кнопку користувач переходить на сторінку для реєстрації.

Рис. 2.8 інтерфейс стартової сторінки

На рис. 2.9 зображено інтерфейс сторінки реєстрації. Так як дана кваліфікаційна робота була націлена саме на Back-end складову, то дизайн інтерфейсу виглядає дещо примітивно. На даному рисунку зображено 5 полів (Login, Password, Name, Surname, Patronymic) які користувач заповнює. Після заповнення всіх даних клієнт натискає на кнопку confirm. На цьому реєстрація завершується.

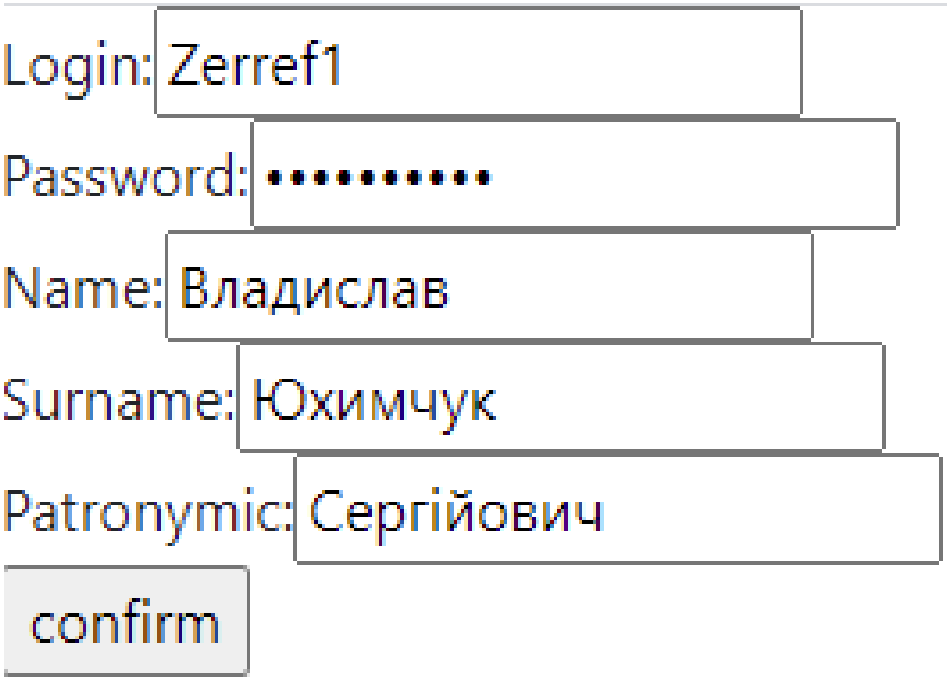


Рис. 2.9 Інтерфейс сторінки реєстрації

Після завершення реєстрації користувач переходить на форму логіна рис. 2.10.

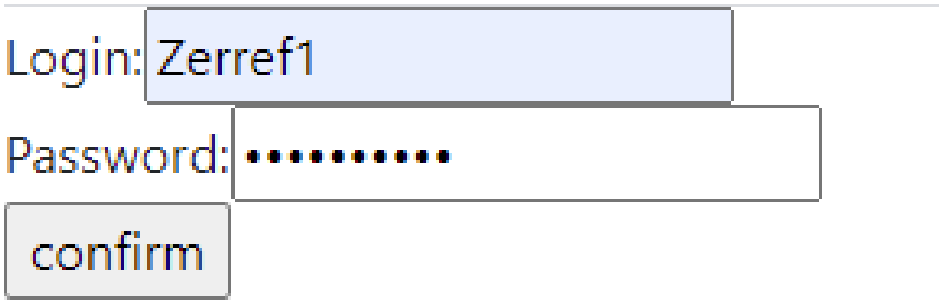


Рис. 2.10 Інтерфейс сторінки авторизації

На сторінці авторизації знаходиться 2 текстових поля – Login, Password. Після заповнення інформації, користувач натискає кнопку confirm, після чого він переходить на сторінку особистого кабінету.

На рис. 2.11 зображено інтерфейс особистого кабінету користувача.

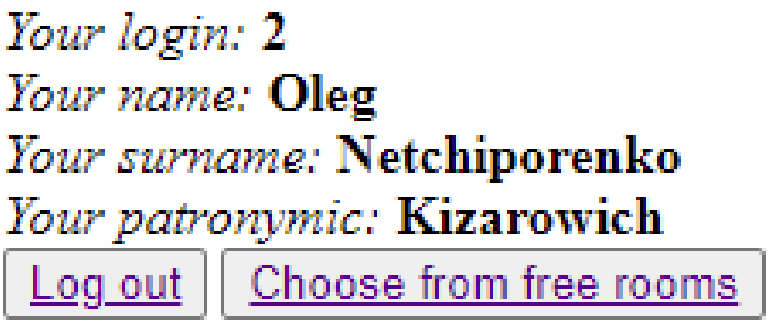


Рис. 2.11 Інтерфейс особистого кабінету користувача

В особистому кабінеті користувача зображена інформація про користувача (логін, прізвище, ім’я, ім’я по-батькові) та дві кнопки: Log out та Choose from free rooms. Натиснувши на кнопку Log out користувач вийде з

свого акаунту та перейде на форму логіну. Натиснувши на кнопку Choose from free rooms користувач перейде на форму для бронювання кімнат.

На рис. 2.12 зображено інтерфейс форми бронювання кімнат.

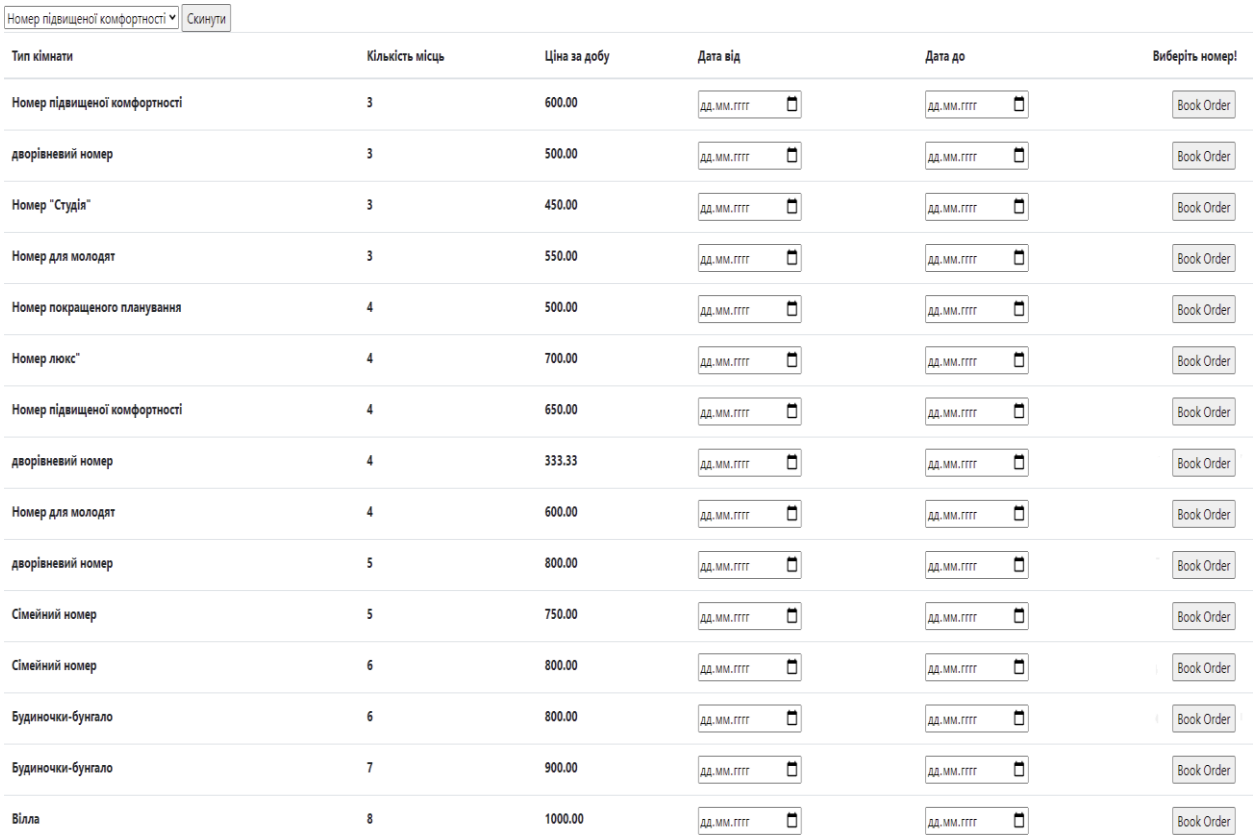


Рис. 2.12 Інтерфейс форми бронювання

В колонці під назвою тип кімнат записана інформація про назви типів номерів, у колонці під назвою кількість місць – інформація про кількість місць в кожній з кімнат, в колонці під назвою ціна за добу – ціна яку повинен заплатити клієнт за добу перебування в цій кімнаті, в колонках під назвами дата від та дата по користувач вибирає діапазон дат бронювання на рис. 2.13 зображено інтерфейс вибору дат.

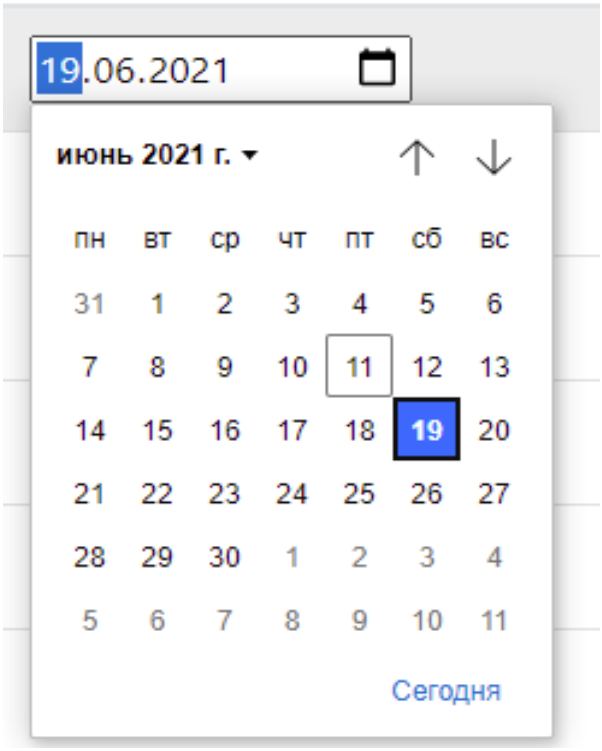


Рис. 2.13 Інтерфейс вибору дати

В колонці під назвою виберіть номер знаходиться кнопка яку користувач натисне після вибору дат бронювання. Після чого користувача перенаправить на сторінку його особистого кабінету де він зможе побачити номери які він забронював. На рис 2.14 зображено інтерфейс забронюваних номерів.

Ідентифікатор запиту	Дата від	Дата до	Ціна	Кількість місць	Тип кімнати	Статус запиту	Оплата
15	2020-11-12	2020-11-21	3600	3	Номер покращеної планування	Оплачено	
13	2020-11-26	2020-12-06	3330	2	Номер покращеної планування	Не оплачено	Оплатити
12	2020-11-19	2020-11-27	2400	1	Номер "Студія"	Не оплачено	Оплатити

Рис. 2.14 Інтерфейс забронюваних номерів

В колонці під назвою ідентифікатор запиту зображено ідентифікатор запиту у базі даних, у колонках під назвами дата від та дата по зображено діапазон дат бронювання номеру, в колонці під назвою ціна зображена сумарна ціна бронювання номеру, в колонці під назвою кількість місць зображена кількість місць в номері, в колонці під назвою тип кімнат зображено

назву типу кімнати, в колонці під назвою статус запиту зображено статус запиту, в колонці під назвою оплата в залежності від статусу запиту відображається чи не відображається кнопка оплатити. Після натискання на цю кнопку зміниться статус запиту з не оплачено на оплачено та кнопка зникне рис. 2.15.

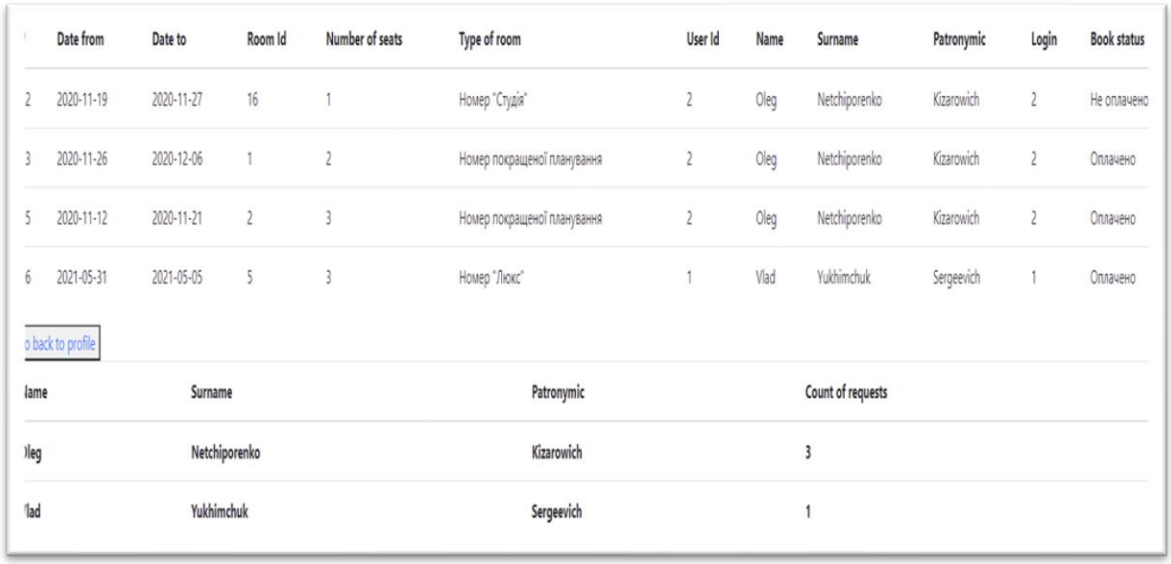
Ідентифікатор запиту	Дата від	Дата до	Ціна	Кількість місць	Тип кімнати	Статус запиту	Оплата
15	2020-11-12	2020-11-21	3600	3	Номер покращеної планування	Оплачено	
13	2020-11-26	2020-12-06	3330	2	Номер покращеної планування	Оплачено	
12	2020-11-19	2020-11-27	2400	1	Номер "Студія"	Не оплачено	Оплатити

Рис. 2.15 Інтерфейс заброньованих номерів

Між особистим кабінетом звичайного користувача та адміністратора є різниця рис. 2.16.

Рис. 2.16 Різниця між особистим кабінетом адміністратора та звичайним клієнтом

Натиснувши на кнопку Go to admin panel адміністратор потрапляє до сторінки адміністратора рис. 2.17.



	Date from	Date to	Room Id	Number of seats	Type of room	User Id	Name	Surname	Patronymic	Login	Book status
2	2020-11-19	2020-11-27	16	1	Номер "Студія"	2	Oleg	Netchiporenko	Kizarowich	2	Не оплачено
3	2020-11-26	2020-12-06	1	2	Номер покращеної планування	2	Oleg	Netchiporenko	Kizarowich	2	Оплачено
5	2020-11-12	2020-11-21	2	3	Номер покращеної планування	2	Oleg	Netchiporenko	Kizarowich	2	Оплачено
6	2021-05-31	2021-05-05	5	3	Номер "Люкс"	1	Vlad	Yukhimchuk	Sergeevich	1	Оплачено

o back to profile

iame	Surname	Patronymic	Count of requests
leg	Netchiporenko	Kizarowich	3
lad	Yukhimchuk	Sergeevich	1

Рис. 2.17 Інтерфейс сторінки адміністратора

На рис. 2.17 зображено інформацію про кожний запит на бронювання номеру та його статус. Далі є кнопка Go back to profile після натискання цієї кнопки адміністратор потрапляє на сторінку особистого кабінету. Також на рис. 2.17 зображено таблицю яка містить інформацію про кількість запитів від кожного з користувачів.

РОЗДІЛ 3

ЕКОНОМІЧНИЙ РОЗДІЛ

Вихідні дані розробки програмного забезпечення:

1. передбачуване число операторів - 1370
2. коефіцієнт складності програми – 1.5
3. коефіцієнт кореляції програми в ході її розробки – 0.09
4. середня годинна заробітна плата програміста, грн/год – 76
5. коефіцієнт кваліфікації програміста, обумовлений від стажу – 0.8
6. вартість машино-години ЕОМ, грн/год – 4.2

3.1. Визначення трудомісткості розробки програмного забезпечення

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = to + tu + ta + tn + totл + tд \text{ , людино-годин,} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 60 людино-годин);

t_u – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

t_n – витрати праці на програмування по готовій блок-схемі,

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ,

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C(1 + p), \text{ людино-годин,} \qquad (3.2)$$

де q – передбачуване число операторів,

C – коефіцієнт складності програми,

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1370 \cdot 1{,}50 \cdot (1 + 0{,}09) = 2239{,}95$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,} \qquad (3.3)$$

де B, яке дорівнює 1,4, – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі,

k, яке дорівнює 0,8, – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності.

$$t_u = \frac{2239{,}95 \cdot 1{,}4}{80 \cdot 0{,}8} = 48{,}99, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20..25) \cdot k} \qquad (3.4)$$

$$t_a = \frac{2239.95}{23 \cdot 0{,}8} = 121{,}74, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20..25) \cdot k} \qquad (3.5)$$

$$t_n = \frac{2239,95}{22 \cdot 0,8} = 127.27, \text{ людино-годин}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) \cdot k} \tag{3.6}$$

$$t_{\text{отл}} = \frac{2239.95}{4 \cdot 0,8} = 699,9, \text{ людино-годин,}$$

– за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1,4 \cdot t_{отл} \tag{3.8}$$

$$t_{\text{отл}}^{\text{к}} = 1,4 \cdot 699,9 = 979,86, \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_{д} = t_{др} + t_{до} \tag{3.8}$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15..20) \cdot k} \tag{3.9}$$

$$t_{\text{др}} = \frac{2239,95}{20 \cdot 0,8} = 139,9, \text{ людино-годин.}$$

де $t_{до}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0,75 \cdot t_{д} \tag{3.10}$$

$$t_{\text{до}} = 0{,}75 \cdot 139{,}9 = 104{,}93, \text{ людино-годин.}$$

$$t_{\text{д}} = 139{,}9 + 104{,}93 = 244{,}83, \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

$$t_{\text{д}} = 60{+}48.99{+}121.74{+}117{+}699.9{+}244.83{=}\ 1292{,}46, \text{ людино-годин.}$$

У результаті ми розрахували, що в загальній складності необхідно 1292,46 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $K_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного для налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \qquad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \qquad (3.12)$$

де t – загальна трудомісткість, людино-годин,

$C_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

$$З_{ЗП} = 1292{,}46 \cdot 76 = 91404{,}4 \text{ грн.}$$

$З_{МВ}$ – вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{МЧ}\text{ , грн,} \qquad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 699{,}9 \cdot 8 = 5599{,}2\text{, грн,}$$

$$К_{ПО} = 91404{,}4 + 5599 = 97{,}003\text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}\text{ , мес,} \qquad (3.14)$$

де B_k – число виконавців,

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні Fp = 176 годин).

$$T = \frac{1292{,}46}{1 \cdot 176} \approx 7{,}34\text{ міс.}$$

Висновки: час розробки даного програмного забезпечення складає 1292,46 людино-годин. Таким чином, очікувана тривалість розробки складе 7,34 місяця при 40 годинному робочому тижні (місячний фонд робочого часу 176 годин), а витрати на створення програмного забезпечення складатимуть 97003 грн.

ВИСНОВКИ

Метою даного проекту є створення інструменту, який допоможе автоматизувати бізнес-процеси готелю. Веб-додаток буде корисним як для клієнтів готелю так і для його керівництва.

Сайт дає можливість вести бізнес цілодобово та відкриває нові можливості для автоматизації та контролю над бізнес процесами готелю.

Для досягнення мети були вирішені наступні завдання:

- аналіз предметної галузі;
- створення основного функціоналу;
- створення інтерфейсу веб-додатку;
- тестування роботи сайту.

Відповідно до проведеного аналізу та завдання в роботі було поставлено та вирішено всі функціональні задачі та вимоги до розробленого веб-додатку.

Визначено трудомісткість розробленої інформаційної системи (1292,46 людино-годин), проведений підрахунок вартості роботи по створенню програми (97003, грн.) та розраховано час на його створення (7,34 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Core Java Volume I – Fundamentals / Cay S. Horstmann 2018. – 148p.
2. Effective Java / Joshua Bloch, 3rd Edition / Joshua Bloch 2017. – 380p.
3. Java: A Beginner's Guide, 8th Edition / Herbert Schildt 2018. – 429p.
4. Java - The Complete Reference, 11th Edition / Herbert Schildt 2018. – 544p.
5. Head First Java, 2nd Edition / Kathy Sierra & Bert Bates 2005. – 314p.
6. Java Concurrency in Practice, 1st Edition / Brian Goetz with Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes, and Doug Lea 2006. – 113p.
7. Test-Driven: TDD and Acceptance TDD for Java Developers, 1st Edition / Lasse Koskela 2007. – 294p.
8. Head First Object-Oriented Analysis Design, 1st Edition / Brett D. McLaughlin, Gary Pollice & David West 2004. – 337p.
9. Java Performance: The Definite Guide, 1st Edition / Scott Oaks 2014. – 344p.
10. Head First Design Patterns, 10th Anniversary Edition / Eric Freeman & Elisabeth Robson with Kathy Sierra & Bert Bates 2014. – 235p.
11. Clean Code – A Handbook of Agile Software Craftsmanship, 1st Edition / Robert Cecil Martin, a.k.a. Uncle Bob 2008. – 420p.
12. Head First Java, 2nd Edition / Kathy Sierra & Bert Bates 2005. – 244p.
13. Test-Driven: TDD and Acceptance TDD for Java Developers, 1st Edition / Lasse Koskela 2007. – 150p.
14. Core Java Volume I – Fundamentals / Cay S. Horstmann 2018. – 216p.
15. Head First Java, 2nd Edition / Kathy Sierra & Bert Bates 2005. – 272p.
16. Clean Code – A Handbook of Agile Software Craftsmanship, 1st Edition / Robert Cecil Martin, a.k.a. Uncle Bob 2008. – 336p.
17. Java Concurrency in Practice, 1st Edition / Brian Goetz with Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes, and Doug Lea 2006. – 215p.

18. Head First Design Patterns, 10th Anniversary Edition / Eric Freeman & Elisabeth Robson with Kathy Sierra & Bert Bates 2014. – 317p.

19. Head First Design Patterns, 10th Anniversary Edition / Eric Freeman & Elisabeth Robson with Kathy Sierra & Bert Bates 2014. – 257p.

20. Head First Java, 2nd Edition / Kathy Sierra & Bert Bates 2005. – 269p.

ДОДАТОК А

КОД ПРОГРАМИ

RequestDAOImpl.java

```
package com.epam.dao.impl;
import com.epam.dao.interfaces.RequestDAO;
import com.epam.db.DbManager;
import com.epam.entities.*;
import javax.net.ssl.HttpsURLConnection;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpSession;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.TimeZone;
public class RequestDAOImpl implements RequestDAO {

    public static final String Querry_For_Getting_All_Info_About_Request = "SELECT \n" +
            "    r.id as requestId , r.booked_from as bookedFrom , r.booked_to as bookedTo , \n" +
            "    user.id as userId, user.name , user.surname , user.patronymic , user.login, user.password ,
user.role_id as roleId,\n" +
            "    room.id as roomId , room.price , room.number_of_seats as numberOfSeats,\n" +
            "    type_of_room.name as typeOfRoom,  \n" +
            "    request_status.id as requestStatusId,\n" +
            "    request_status.name as requestStatus\n" +
            "from request r\n" +
            "join user on r.user_id = user.id\n" +
            "join room on r.room_id = room.id\n" +
            "join type_of_room on room.type_of_room_id = type_of_room.id\n" +
            "join request_status on r.request_status_id = request_status.id";
    public static final String QUERY_FOR_CREATING_NEW_REQUEST = "Insert into request \n" +
            "(booked_from, \n" +
            "booked_to,\n" +
            "user_id,\n" +
            "room_id,\n" +
            "request_status_id)\n" +
            "values (? , ? , ? , ? , 2)";
    public static final String FIND_ALL_REQUESTS_BY_USER_ID = "SELECT r.id as requestId ,
r.booked_from as bookedFrom , r.booked_to as bookedTo ,\n" +
            "\tuser.id as userId,\n" +
            "\troom.price , room.number_of_seats as numberOfSeats,\n" +
            "\ttype_of_room.name as typeOfRoom,\n" +
            "\trequest_status.name as requestStatus\n" +
            "from request r\n" +
            "\tjoin user on r.user_id = user.id\n" +
            "\tjoin room on r.room_id = room.id\n" +
            "\tjoin type_of_room on room.type_of_room_id = type_of_room.id\n" +
            "\tjoin request_status on r.request_status_id = request_status.id\n" +
            "where user_id = ?";
    public static final String UPDATE_REQUEST_STATUS_AFTER_PEYMENT = "update request set
request_status_id = 1 where id = ?";
    private Connection;
    @Override
    public Optional<Request> findById(Long id) {
       return Optional.ofNullable(null);
    }
```

```
public RequestDAOImpl() {
   this.connection = DbManager.getConnection();
}
@Override
public List<Request> findAll() {
   String sql = Querry_For_Getting_All_Info_About_Request;
   TimeZone.setDefault(TimeZone.getTimeZone("UTC"));
   List<Request> requests = new ArrayList<>();
   try (PreparedStatement = connection.prepareStatement(sql)) {
      ResultSet = preparedStatement.executeQuery();
      while (resultSet.next()) {
         requests.add(new Request(
               resultSet.getLong("requestId"),
               resultSet.getDate("bookedFrom").toLocalDate(),
               resultSet.getDate("bookedTo").toLocalDate(),
               new Room(resultSet.getLong("roomId"),
                     resultSet.getInt("numberOfSeats"),
                     null,
                     new TypeOfRoom(null,
                           resultSet.getString("typeOfRoom")),
                     resultSet.getBigDecimal("price")),
               new User(resultSet.getLong("userId"),
                     resultSet.getString("name"),
                     resultSet.getString("surname"),
                     resultSet.getString("patronymic"),
                     resultSet.getString("login"), resultSet.getString("password"),
                     null),
               new RequestStatus(resultSet.getLong("requestStatusId"),
                     resultSet.getString("requestStatus"))));
      }
      return requests;
   } catch (SQLException e) {
      throw new RuntimeException(e);
   }
}
@Override
public Request createNew(Request entity) {
   String sql = QUERY_FOR_CREATING_NEW_REQUEST;
   TimeZone.setDefault(TimeZone.getTimeZone("UTC"));
   try {
      PreparedStatement = connection.prepareStatement(sql);
      preparedStatement.setDate(1, Date.valueOf(entity.getBookedFrom()));
      preparedStatement.setDate(2, Date.valueOf(entity.getBookedTo()));
      preparedStatement.setLong(3, entity.getUser().getId());
      preparedStatement.setLong(4, entity.getRoom().getId());
      preparedStatement.executeUpdate();
   } catch (SQLException e) {
      throw new RuntimeException(e);
   }
   return null;
}
@Override
public Request update(Request entity, Long id) {
   return null;
}

@Override
public void deleteById(Long id) {
}
@Override
public void delete(Request entity) {
```

```
    }
    @Override
    public Optional<com.epam.entities.Request> findByName(String name) {
        return Optional.empty();
    }
    @Override
    public List<Request> findRequestByUserId(Long id) {
        String sql = FIND_ALL_REQUESTS_BY_USER_ID;
        TimeZone.setDefault(TimeZone.getTimeZone("UTC"));
        List<Request> requests = new ArrayList<>();
        try (PreparedStatement = connection.prepareStatement(sql)) {
            preparedStatement.setLong(1, id);
            ResultSet = preparedStatement.executeQuery();
            while (resultSet.next()) {
                requests.add(new Request(
                        resultSet.getLong("requestId"),
                        resultSet.getDate("bookedFrom").toLocalDate(),
                        resultSet.getDate("bookedTo").toLocalDate(),
                        new Room(null,
                                resultSet.getInt("numberOfSeats"),
                                null,
                                new TypeOfRoom(null,
                                        resultSet.getString("typeOfRoom")),
                                resultSet.getBigDecimal("price")),
                        null,
                        new RequestStatus(null,
                                resultSet.getString("requestStatus"))));
            }
            return requests;
        } catch (SQLException e) {
            throw new RuntimeException();
        }
    }
    @Override
    public Boolean updateInformationAboutPayment(Long id) {
        String sql = UPDATE_REQUEST_STATUS_AFTER_PEYMENT;
        try (PreparedStatement = connection.prepareStatement(sql)) {
            preparedStatement.setLong(1 ,id);
            preparedStatement.executeUpdate();
            return true;
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
    @Override
    public Integer countOfRequests(Long id) {
        return null;
    }
}
```

RoomDAOImpl.java

```
package com.epam.dao.impl;
import com.epam.dao.interfaces.RoomDAO;
import com.epam.db.DbManager;
import com.epam.entities.Room;
import com.epam.entities.Status;
import com.epam.entities.TypeOfRoom;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
```

```
import java.util.List;
import java.util.Optional;
import java.util.TimeZone;
public class RoomDAOImpl implements RoomDAO {
   public static final String Find_Free_Rooms = "SELECT\n" +
         "\tr.id , r.number_of_seats as numberOfSeats,\n" +
         "\ttype_of_room.name as typeOfRoom, r.price,\n" +
         "\tstatus.name as statusName\n" +
         "From room r\n" +
         "join status on r.status_id = status.id\n" +
         "join type_of_room on r.type_of_room_id = type_of_room.id\n" +
         "where status.id = 1\n" +
         "order by r.number_of_seats";
   public static final String UPDATE_STATUS_OD_ROOM = "update room set status_id = 2 where id =
?";
   public RoomDAOImpl() {
      this.connection = DbManager.getConnection();
   }
   private Connection;
   @Override
   public Optional<Room> findById(Long aLong) {
      return Optional.empty();
   }
   @Override
   public List<Room> findAll() {
      return null;
   }
   @Override
   public Room createNew(Room entity) {
      return null;
   }
   @Override
   public Room update(Room entity, Long aLong) {
      return null;
   }
   @Override
   public void deleteById(Long aLong) {
   }
   @Override
   public void delete(Room entity) {
   }
   @Override
   public List<Room> findFreeRooms() {
      String sql = Find_Free_Rooms;
      TimeZone.setDefault(TimeZone.getTimeZone("UTC"));
      Room = null;
      List<Room> rooms = new ArrayList<>();
      try (PreparedStatement = connection.prepareStatement(sql)) {
         ResultSet = preparedStatement.executeQuery();
         while (resultSet.next()) {
            rooms.add(new Room(
                  resultSet.getLong("id"),
                  resultSet.getInt("numberOfSeats"),
                  new Status(resultSet.getLong("id"), resultSet.getString("statusName")),
                  new TypeOfRoom(resultSet.getLong("id"), resultSet.getString("typeOfRoom")),
                  resultSet.getBigDecimal("price")));
         }
         return rooms;
      } catch (SQLException e) {
         throw new RuntimeException(e);
      }
   }
```

```
    @Override
    public Boolean updateRoomStatusById(Long id) {
        String sql = UPDATE_STATUS_OD_ROOM;
        TimeZone.setDefault(TimeZone.getTimeZone("UTC"));
        try (PreparedStatement = connection.prepareStatement(sql)) {
            preparedStatement.setLong(1 ,id);
            preparedStatement.executeUpdate();
            return true;
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
}
```

TypeOfRoomDAOImpl.java

```
package com.epam.dao.impl;
import com.epam.dao.interfaces.TypeOfRoomDAO;
import com.epam.db.DbManager;
import com.epam.entities.TypeOfRoom;
import java.awt.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
public class TypeOfRoomDAOImpl implements TypeOfRoomDAO {
    public static final String SELECT_FROM_HOTEL_TYPE_OF_ROOM = "SELECT * FROM
hotel.type_of_room";
    private Connection;
    public TypeOfRoomDAOImpl() {
        this.connection = DbManager.getConnection();
    }
    @Override
    public Optional<TypeOfRoom> findById(Long aLong) {
        return null;
    }
    @Override
    public List<TypeOfRoom> findAll(){
        String sql = SELECT_FROM_HOTEL_TYPE_OF_ROOM;
        List <TypeOfRoom> typeOfRooms = new ArrayList<>();
        TypeOfRoom = null;
        try (PreparedStatement = connection.prepareStatement(sql)){
            ResultSet = preparedStatement.executeQuery(sql);
            while (resultSet.next()){
                typeOfRooms.add(new TypeOfRoom(resultSet.getLong("id"),
                        resultSet.getString("name")));
            }return typeOfRooms;
        }catch (SQLException e){
            throw new RuntimeException(e);
        }
    }
    @Override
    public TypeOfRoom createNew(TypeOfRoom entity) {
        return null;
    }
    @Override
    public TypeOfRoom update(TypeOfRoom entity, Long aLong) {
        return null;
    }
```

```
    @Override
    public void deleteById(Long aLong) {
    }
    @Override
    public void delete(TypeOfRoom entity) {
    }
}
```

UserDAOImpl.java

```
package com.epam.dao.impl;
import com.epam.dao.interfaces.UserDAO;
import com.epam.db.DbManager;
import com.epam.entities.Role;
import com.epam.entities.User;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import java.util.Optional;
public class UserDAOImpl implements UserDAO {
    public static final String
INSERT_INTO_USER_NAME_SURNAME_PATRONYMIC_LOGIN_PASSWORD_ROLE_VALUES_2 =
"INSERT INTO user (name , surname , patronymic , login , password , role_id) VALUES (? , ? , ? , ? , ? , 2 )";
    public static final String
SELECT_U_ID_U_NAME_U_SURNAME_U_PATRONYMIC_U_LOGIN_U_PASSWORD_R_ID_AS_ROLE_ID_R_NAME_AS_ROLE_FROM_USER_U_JOIN_ROLE_R_ON_U_ROLE_ID_R_ID = "SELECT u.id, u.name ,
u.surname , u.patronymic, u.login ,\n" +
            "u.password, r.id as role_id , r.name as role FROM user u \n" +
            "JOIN role r ON u.role_id = r.id\n" +
            "where u.login = ?";
    private Connection;
    private Object User;

    public UserDAOImpl() {
        this.connection = DbManager.getConnection();
    }
    @Override
    public Optional<User> findById(Long aLong) {
        return Optional.empty();
    }
    @Override
    public List<User> findAll() {
        return null;
    }
    @Override
    public User createNew(User entity) {
        String sql =
INSERT_INTO_USER_NAME_SURNAME_PATRONYMIC_LOGIN_PASSWORD_ROLE_VALUES_2;
        try {
            PreparedStatement = connection.prepareStatement(sql);
            preparedStatement.setString(1, entity.getName());
            preparedStatement.setString(2, entity.getSurname());
            preparedStatement.setString(3, entity.getPatronymic());
            preparedStatement.setString(4, entity.getLogin());
            preparedStatement.setString(5, entity.getPassword());
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return null;
    }
```

```
    @Override
    public Optional<User> findByName(String name) {
        String sql =
SELECT_U_ID_U_NAME_U_SURNAME_U_PATRONYMIC_U_LOGIN_U_PASSWORD_R_ID_AS_ROLE_I
D_R_NAME_AS_ROLE_FROM_USER_U_JOIN_ROLE_R_ON_U_ROLE_ID_R_ID;
        User = null;
        try (PreparedStatement = connection.prepareStatement(sql)) {
            preparedStatement.setString(1, name);
            ResultSet = preparedStatement.executeQuery();
            if (resultSet.next()) {
                user = new User(
                        resultSet.getLong("id"),
                        resultSet.getString("name"),
                        resultSet.getString("surname"),
                        resultSet.getString("patronymic"),
                        resultSet.getString("login"),
                        resultSet.getString("password"),
                        new Role(resultSet.getLong("role_id"),
                                resultSet.getString("role")));
            }
            return Optional.ofNullable(user);
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
    @Override
    public User update(User entity, Long aLong) {
        return null;
    }
    @Override
    public void deleteById(Long aLong) {
    }
    @Override
    public void delete(User entity) {

    }
}
```

GenericDAOImpl.java

```
package com.epam.dao.interfaces;
import java.sql.SQLException;
import java.util.List;
import java.util.Optional;
public interface GenericDAO<T , ID> {
    Optional<T> findById(ID id);
    List<T> findAll() throws SQLException;
    T createNew(T entity);
    T update (T entity, ID id);
    void deleteById(ID id);
    void delete(T entity);
}
```

RequestDAO.java

```
package com.epam.dao.interfaces;

import com.epam.entities.Request;
import java.util.List;
import java.util.Optional;
public interface RequestDAO extends GenericDAO<Request , Long> {
    Optional<Request> findByName(String name);
    List<Request> findRequestByUserId(Long id);
    Boolean updateInformationAboutPayment(Long id);
```

```
    Integer countOfRequests(Long id);
}
```

RoleDAO.java

```
package com.epam.dao.interfaces;
import com.epam.entities.Role;
public interface RoleDAO extends GenericDAO<Role , Long>{
}
```

RoomDAO.java

```
package com.epam.dao.interfaces;
import com.epam.entities.Room;
import java.util.List;
import java.util.Optional;
public interface RoomDAO extends GenericDAO <Room, Long> {
    List<Room> findFreeRooms ();
    Boolean updateRoomStatusById(Long id);
}
```

TypeOfRoomDAO.java

```
package com.epam.dao.interfaces;
import com.epam.entities.TypeOfRoom;
public interface TypeOfRoomDAO extends GenericDAO <TypeOfRoom, Long> {
}
```

UserDAO.java

```
package com.epam.dao.interfaces;
import com.epam.entities.User;
import java.util.Optional;
public interface UserDAO extends GenericDAO <User, Long> {
    Optional<User> findByName(String name);
}
```

UserDTO.java

```
package com.epam.dao;
import com.epam.entities.Role;
import lombok.*;
@Getter
@AllArgsConstructor
@EqualsAndHashCode
public class UserDTO {
    private Long id;
    private String name;
    private String surname;
    private String patronymic;
    private Long requestCount;
    public UserDTO(Long id, String name, String surname, String patronymic) {
        this.id = id;
        this.name = name;
        this.surname = surname;
        this.patronymic = patronymic;
    }
```

DbManager.java

```
package com.epam.db;
import com.epam.exception.ConnectionException;
import org.apache.log4j.Logger;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DbManager {
    private static final String URL =
"jdbc:mysql://localhost:3306/hotel?user=root&password=root123&serverTimezone=UTC";
```

```
private static final String CONNECTION_FAILURE_MSG = "Connection to DB failed";
private static final Logger log = Logger.getLogger(DbManager.class);
private static DbManager instance;
public static synchronized DbManager getInstance() {
    if (instance == null)
        instance = new DbManager();
    return instance;
}
/**
 * Returns a DB connection from the Pool Connections. Before using this
 * method you must configure the Date Source and the Connections Pool in your
 * WEB_APP_ROOT/META-INF/context.xml file.
 *
 * @return A DB connection.
 */
public static Connection getConnection() {
    try {
        return DriverManager.getConnection(URL);
    } catch (SQLException e) {
        log.error(CONNECTION_FAILURE_MSG, e);
        throw new ConnectionException(CONNECTION_FAILURE_MSG, e);
    }
}
private DbManager() {
}
// ////////////////////////////////////////////////////////
// DB util methods
// ////////////////////////////////////////////////////////
/**
 * Commits and close the given connection.
 *
 * @param con
 *            Connection to be committed and closed.
 */
public void commitAndClose(Connection con) {
    try {
        con.commit();
        con.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}
/**
 * Rollbacks and close the given connection.
 *
 * @param con
 *            Connection to be rollbacked and closed.
 */
public void rollbackAndClose(Connection con) {
    try {
        con.rollback();
        con.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}
```

AdmimPanelServ.java

```
package com.epam.servlets;
import com.epam.dao.UserDTO;
import com.epam.dao.impl.RequestDAOImpl;
import com.epam.dao.impl.RoomDAOImpl;
```

```
import com.epam.dao.impl.UserDAOImpl;
import com.epam.dao.interfaces.UserDAO;
import com.epam.entities.Request;
import com.epam.entities.Room;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.function.Function;
import java.util.stream.Collectors;
@WebServlet("/admin")
public class AdminPanelServ extends HttpServlet {
   private UserDAO = new UserDAOImpl();
   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
      RequestDAOImpl = new RequestDAOImpl();
      List<Request> requests;
      requests = requestDAOImpl.findAll();
      req.setAttribute("request", requests);
      Map<UserDTO, Long> counted = requests.stream()
            .collect(Collectors.groupingBy(request -> new UserDTO(request.getUser().getId(),
                  request.getUser().getName(), request.getUser().getSurname(),
                  request.getUser().getPatronymic()), Collectors.counting()));
      List<UserDTO> collect = counted.keySet().stream().map(userDTO -> new UserDTO(userDTO.getId(),
            userDTO.getName(), userDTO.getSurname(),
            userDTO.getPatronymic(), counted.get(userDTO))).collect(Collectors.toList());
      req.setAttribute("collect" , collect);
      req.getRequestDispatcher("admin.jsp").forward(req, resp);
   }
   @Override
   protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
      super.doPost(req, resp);
   }
}
```

ChooseFromFreeRoomsServ.java

```
package com.epam.servlets;
import com.epam.dao.impl.RequestDAOImpl;
import com.epam.dao.impl.RoomDAOImpl;
import com.epam.dao.impl.TypeOfRoomDAOImpl;
import com.epam.entities.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.math.BigDecimal;
import java.text.DecimalFormat;
import java.time.LocalDate;
import java.util.*;
@WebServlet("/order")
public class ChooseFromFreeRoomsServ extends HttpServlet {
   private RequestDAOImpl requestDAO = new RequestDAOImpl();
```

```
    private RoomDAOImpl roomDAO = new RoomDAOImpl();
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        RoomDAOImpl = new RoomDAOImpl();
        TypeOfRoomDAOImpl typeOfRooms = new TypeOfRoomDAOImpl();
        List<Room> rooms = roomDAOimpl.findFreeRooms();
        req.setAttribute("rooms" , rooms);
        req.getRequestDispatcher("freeRoom.jsp").forward(req, resp);
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        String typeOfRoom = req.getParameter("typeOfRoom");
        LocalDate dateFrom = LocalDate.parse(req.getParameter("dateFrom"));
        LocalDate dateTo = LocalDate.parse(req.getParameter("dateTo"));
        BigDecimal price = new BigDecimal(req.getParameter("price"));
        Long roomId = Long.valueOf(req.getParameter("bookId"));
        HttpSession session = req.getSession();
        User = (User) session.getAttribute("user");
        Long userId = user.getId();
        Request = Request.builder()
            .bookedFrom(dateFrom)
            .bookedTo(dateTo)
            .room(Room.builder().id(roomId).build())
            .user(User.builder().id(userId).build())
            .build();
        requestDAO.createNew(request);
        roomDAO.updateRoomStatusById(roomId);
        System.out.println(request);
        req.getRequestDispatcher("profile.jsp").forward(req, resp);
    }
}
```

HomePageServ.java

```
package com.epam.servlets;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
@WebServlet("/homePage")
public class HomePageServ extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        req.getRequestDispatcher("homePage.jsp").forward(req , resp);
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        super.doGet(req, resp);
    }
}
```

LoginServ.java

```
package com.epam.servlets;
import com.epam.dao.impl.UserDAOImpl;
import com.epam.dao.interfaces.UserDAO;
import com.epam.entities.User;
import javax.servlet.ServletException;
```

```
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.io.PrintWriter;
@WebServlet("/login")
public class LoginServ extends HttpServlet {
   private UserDAO = new UserDAOImpl();

   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
      req.getRequestDispatcher("login.jsp").forward(req, resp);
   }
   @Override
   protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
      String login = req.getParameter("Login");
      String password = req.getParameter("Password");
      User = userDAO.findByName(login).orElseThrow(RuntimeException::new);
      if (user.getLogin().equals(login) && user.getPassword().equals(password)) {
         HttpSession session = req.getSession();
         session.setAttribute("login", user.getLogin());
         session.setAttribute("role", user.getRole().getName());
         session.setAttribute("roleId" , user.getRole().getId());
         session.setAttribute("user", user);
         session.setAttribute("name", user.getName());
         session.setAttribute("surname", user.getSurname());
         session.setAttribute("patronymic", user.getPatronymic());
         session.setAttribute("id" , user.getId());
         PrintWriter out = resp.getWriter();
         resp.sendRedirect("/profile");
         out.close();
      } else {
         resp.sendRedirect("login.jsp");
      }
   }
}
```

LogOutServ.java

```
package com.epam.servlets;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
@WebServlet("/logOut")
public class LogOutServ extends HttpServlet {
   @Override
   protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
      super.doGet(req, resp);
   }
   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
      HttpSession session = req.getSession();
      session.invalidate();
```

```
            req.getRequestDispatcher("login.jsp").forward(req, resp);
        }
    }
```

ProfileServ.java

```
package com.epam.servlets;
import com.epam.dao.impl.RequestDAOImpl;
import com.epam.dao.impl.UserDAOImpl;
import com.epam.dao.interfaces.UserDAO;
import com.epam.entities.Request;
import com.epam.entities.User;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;
@WebServlet("/profile")
public class ProfileServ extends HttpServlet {
    private UserDAO = new UserDAOImpl();
    private RequestDAOImpl requestDAO = new RequestDAOImpl();
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        req.setCharacterEncoding("UTF-8");
        HttpSession session = req.getSession();
        User = (User) session.getAttribute("user");
        req.setAttribute("login", user.getLogin());
        RequestDAOImpl = new RequestDAOImpl();
        List<Request> requests = new ArrayList<>();
        requests = requestDAOImpl.findRequestByUserId(user.getId());
        requests.forEach(requ -> requ.getRoom().setPrice(new BigDecimal(
                ((requ.getBookedTo().toEpochDay() -
requ.getBookedFrom().toEpochDay())*requ.getRoom().getPrice().longValue())
        )));
        req.setAttribute("requests" , requests);
        req.setAttribute("roleId" , session.getAttribute("roleId"));
        req.getRequestDispatcher("profile.jsp").forward(req, resp);
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        Long requestBookId = Long.valueOf(req.getParameter("requestBookId"));
        requestDAO.updateInformationAboutPayment(requestBookId);
        resp.sendRedirect("/profile");
    }
}
```

RegistrationServ.java

```
package com.epam.servlets;
import com.epam.dao.impl.UserDAOImpl;
import com.epam.dao.interfaces.UserDAO;
import com.epam.entities.User;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import java.io.IOException;
@WebServlet("/registration")
public class RegistrationServ extends HttpServlet {
   private UserDAO = new UserDAOImpl();

   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
      req.getRequestDispatcher("registration.jsp").forward(req, resp);
   }
   @Override
   protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
      String login = req.getParameter("Login");
      String password = req.getParameter("Password");
      String name = req.getParameter("Name");
      String surname = req.getParameter("Surname");
      String patronymic = req.getParameter("Patronymic");
      if (login != null && password != null && name != null && surname != null && patronymic != null) {
         User = User.builder()

.login(login).name(name).surname(surname).patronymic(patronymic).password(password).build();
         userDAO.createNew(user);
      } else {
         req.getRequestDispatcher("registration.jsp").forward(req, resp);
         return;
      }
      req.getRequestDispatcher("homePage.jsp").forward(req, resp);
   }
}
```

WebApplication.java

```
package com.epam;
import org.apache.catalina.WebResourceRoot;
import org.apache.catalina.WebResourceSet;
import org.apache.catalina.core.StandardContext;
import org.apache.catalina.startup.Tomcat;
import org.apache.catalina.webresources.DirResourceSet;
import org.apache.catalina.webresources.EmptyResourceSet;
import org.apache.catalina.webresources.StandardRoot;
import java.io.File;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
public class WebApplication {
   private static File getRootFolder() {
      try {
         File root;
         String runningJarPath =
WebApplication.class.getProtectionDomain().getCodeSource().getLocation().toURI().getPath().replaceAll("\\\\",
"/");
         int lastIndexOf = runningJarPath.lastIndexOf("/target/");
         if (lastIndexOf < 0) {
            root = new File("");
         } else {
            root = new File(runningJarPath.substring(0, lastIndexOf));
         }
         System.out.println("application resolved root folder: " + root.getAbsolutePath());
         return root;
      } catch (URISyntaxException ex) {
         throw new RuntimeException(ex);
      }
```

```
        }
        public static void main(String[] args) throws Exception {
            File root = getRootFolder();
            System.setProperty("org.apache.catalina.startup.EXIT_ON_INIT_FAILURE", "true");
            Tomcat = new Tomcat();
            Path tempPath = Files.createTempDirectory("tomcat-base-dir");
            tomcat.setBaseDir(tempPath.toString());

            String webPort = System.getenv("PORT");
            if (webPort == null || webPort.isEmpty()) {
                webPort = "8080";
            }
            tomcat.setPort(Integer.valueOf(webPort));
            File webContentFolder = new File(root.getAbsolutePath(), "src/main/webapp/");
            if (!webContentFolder.exists()) {
                webContentFolder = Files.createTempDirectory("default-doc-base").toFile();
            }
            StandardContext ctx = (StandardContext) tomcat.addWebapp("", webContentFolder.getAbsolutePath());
            ctx.setParentClassLoader(WebApplication.class.getClassLoader());
            System.out.println("configuring app with basedir: " + webContentFolder.getAbsolutePath());
            File additionWebInfClassesFolder = new File(root.getAbsolutePath(), "target/classes");
            WebResourceRoot resources = new StandardRoot(ctx);
            WebResourceSet resourceSet;
            if (additionWebInfClassesFolder.exists()) {
                resourceSet = new DirResourceSet(resources, "/WEB-INF/classes",
additionWebInfClassesFolder.getAbsolutePath(), "/");
                System.out.println("loading WEB-INF resources from as '" +
additionWebInfClassesFolder.getAbsolutePath() + "'");
            } else {
                resourceSet = new EmptyResourceSet(resources);
            }
            resources.addPreResources(resourceSet);
            ctx.setResources(resources);
            tomcat.start();
            tomcat.getServer().await();
        }
    }
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Yukhimchuk_Diploma.docx	Пояснювальна записка кваліфікаційної роботи. Документ Word.
Yukhimchuk_Diploma.pdf	Пояснювальна записка кваліфікаційної роботи в форматі PDF.
Програма	
Hotel.zip	Архів. Містить коди програми.
Презентація	
Yukhimchuk_Diploma.pptx	Презентація кваліфікаційної роботи.