Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Навчально-науковий центр заочної форми навчання*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Розробка Web-додатку для автоматизації системи управління
(тема)

Виконав:
студент 2 курсу, групи *ІМІзм-19-2*
Остапенко О.В.
(прізвище, ініціали)

Спеціальність *172 Телекомунікації та радіотехніка*
(код і повна назва спеціальності)

Тип програми *освітньо-наукова програма*

Освітня програма *Інформаційно-мережна інженерія*
(повна назва освітньої програми)

Керівник *доц. Скорик Ю.В.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Безрук В.М.*
(підпис) (прізвище, ініціали)

2021 р.

Не містить відомостей, заборонених до відкритого публікування

Студент _____ / *Остапенко О.В.*/

Керівник _____ /*Скорик Ю.В.*/

Харківський національний університет радіоелектроніки
(повна назва вищого навчального закладу)

Навчально-науковий центр заочної форми навчання

Кафедра *Інформаційно-мережної інженерії*

Освітній рівень *другий (магістерський)*

Спеціальність *172 Телекомунікації та радіотехніка*
(код і назва)

Тип програми *освітньо-наукова програма*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Інформаційно-мережна інженерія*
(назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Остапенко Олександру Володимировичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка Web-додатку для автоматизації системи Управління*

затверджена наказом університету від «*25*» *березня* 2021 року № *33 Стз*

2. Термін подання студентом роботи до екзаменаційної комісії *18 травня 2021 р.*

3. Вихідні дані до роботи *Розглянути теоретичні і практичні особливості розробки web-додатку, розробити web-додаток для автоматизації системи обліку готельних заявок*

4. Перелік питань, що потрібно опрацювати в роботі
Вступ
1. Web-програмування
2. Створення web-додатку
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____

Слайди у форматі Power Point (назва та мета роботи, завдання, термінологія, HTML, CSS, Java та MySQL, JSP, структура веб-додатку, дизайн веб-додатку, форми регістрації та автентифікації, MVC, Register.jsp, RegisterServlet.java, UserServlet.jsp, RegisterServlet.java (продовження), RegisterDao.java, DBConnection.java, БД, висновки

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів атестаційної роботи	Терміни виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	*25.03.21*	
2	*Підбір літератури за темою роботи.*	*26.03 – 30.03.21*	
3	*Виконання розділу 1*	*31.03 – 12.04.21*	
4	*Виконання розділу 2*	*13.04 – 23.04.21*	
5	*Оформлення пояснювальної записки*	*24.04 – 09.05.21*	
6	*Оформлення презентаційного матеріалу,*		
	підготовка до захисту у ЕК	*10.05 – 18.05.21*	

Дата видачі завдання *25 березня 2021 р.*

Студент _____ *Остапенко О.В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Скорик Ю.В.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: 97 с., 37 рис., 2 табл., 24 джерел

Мета роботи – розробка web-додатку та підключення його до бази даних.

Розглянуті основні технології розробки web-додатку HTML та CSS. Розглянута мова програмування Java, принцип роботи з базою даних MySQL та інтерфейсом MySQL Workbench. Розроблені структура, дизайн та функціонал веб-додатку. Проведені розробки основних елементів веб-додатку, таблиць бази даних. Написані програмні коди на мові Java згідно з архітектурою MVC. Розроблені форми регістрації та аутентифікації, а також функціонал системи бронювання номерів та оформлення експрес-заявок, пов’язані з базою даних.

WEB-ДОДАТОК, HTML, CSS, JAVA, MYSQL, JSP, MVC ,КОНТЕЙНЕР APACHE-TOMCAT, СЕРВЛЕТ, WEB-ТЕХНОЛОГІЇ, КОД, БАЗА ДАНИХ, ФОРМА, ІНТЕРНЕТ.

THE ABSTRACT

Explanatory note: 97 p., 37 fig., 2 tabl., 24 reference

The purpose of work – web application development and connection it to database.

The main technologies of web application development as HTML and CSS were considered. Java programming language, working principle of MySQL database and MySQL Workbench web interface were considered. Web application structure, design and functionality has been developed. Main elements of the web application and database tables was developed. Java programme code, with accordance to MVC architecture was written. Registration, authorization forms and the functionality of the booking/express reservation systems form, which connected to the database, were developed.

WEB APPLICATION, HTML, CSS, JAVA, JSP, MVC, CONTAINER APACHE-TOMCAT, SERVLET, MYSQL, WEB-TECHNOLOGY, CODE, DATABASE, FORM, INTERNET.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ

B2B (business to business) – бізнес модель «бізнес для бізнесу»;

B2C (business to consumer) – бізнес модель «бізнес для споживача»;

C2C (consumer to consumer) – бізнес моделі «споживач для споживача»;

CSS (Cascading Style Sheets) – каскадні таблиці стилів;

HTML (Hyper Text Markup Language) – мова розмітки гіпертекстових документів;

Java EE (Java Enterprise Edition) – обчислювальна корпоративна платформа Java;

JDBC (Java Database Connectivity) – з'єднання з базами даних на Java;

JSP (Java Servlet Pages) – технологія, що дозволяє динамічно генерувати HTML, XML та інші веб-сторінки;

MVC (Model-View-Controller) – архітектурний шаблон Java «модель-представлення-контроллер»;

SQL (Structured Query Language) – мова структурованих запитів;

URL (Uniform Recsource Locator) – єдиний вказівник на ресурс;

WWW (World Wide Web) – глобальний інформаційний простір;

БД – база даних;

СУБД – система управління базами даних.

ВСТУП

На теперішній час HTML як основа, щоб написати будь-яку веб-сторінку в Інтернет. Коли використовують код каскадних таблиць стиля (CSS), то спроектування коду на мові HTML дуже відрізняється. Коли цю мову почали використовувати й впроваджувати багато користувачів, необхідні стали дизайнери, основна робота яких – це зроблення новітнього та більш сучасного зовнішнього вигляду сторінки. Завдання верстки, як один із процесів видозміни проекту дизайнера у веб-сторінку, яку відображає браузер. Проте результат можливо отримати по різному, розробник вирішує, якому з них надається перевага. Окрім знань стосовно специфіки створення деяких ефектів треба уміти бачити результат роботи з елементами веб-сторінок та розуміти особливості різних браузерів. Від того, який вибір обере розробник залежить те, як буде працювати сайт і якість того, як буде відображатись, доступність для різного виду пристроїв та браузерів.

Для того, щоб можливості були більшими і створювалась динаміка веб-сторінок, розробники використовують мови web-програмування, такі как Java, PHP, Perl, JavaScript, ASP, а також під’єднанням їх же до бази даних.

Стрімкий розвиток глобальної мережі дає повшток і багато іншим сферам діяльності людини. Інтернет зумовив до революційних змін у сфері програмного забезпечення.

На даний час глобальна мережа Інтернет як невід’ємна частина життя великої кількості людей. Зараз майже кожна компанія має свій web-сайт, за яким користувач може здійснити замовлення і робити заявки через Інтернет. Кожного дня розробляються нові веб-служби, які за своєю функціональністю та інтерфейсами не уступають і додаткам Windows.

Дану кваліфікаційну роботу присвячено розробці веб-додатку, щоб автоматизувати систему обліку готельних заявок із взалученням технологій HTML, CSS, мови Java, та баз даних MySQL.

1 WEB-ПРОГРАМУВАННЯ

Веб-програмування це один із розділів програмування, що орієнтовано на те, щоб розробляти динамічні інтернет-додатки. Основне завдання, це створення графічного веб-дизайну з використанням супутніх технологій а також провести програмування як у клієнта, так і на сервері.

Клієнтські мови записуються у користувача (частіше в браузері), результати виконання програм цією мовою можуть бути зроблені і без відправки документу до серверу. Найбільш поширеною є JavaScript. Якшо необхідні більш широку можливості, то це вже з залученням сторони серверу[1-3].

Коли відбувається запит користувачем сторінки в мережі Інтернет, то ця сторінка на початку має обробитись на серверу. Обробляються програми, які пов'язані зі сторінкою, та далі відправляються до відвідувача як файл, переданий по мережі з розширенням (HTML, PHP, XML, ASP). Серверні скрипти мають взаємодіяти з базами даних. Найчастіше використовують СУБД Mysql, PostreSQL, MS SQL Server, Oracle. Також мови програмування: PHP, Java, Python, Perl, Ruby, Asp.net [4].

1.1 Гіпертекстова мова розмітки HTML

World Wide Web – WWW, складає у собі безліч різноманітних і зв'язаних між собою електронних документів, які є джерелом інформаційних даних, що описуються спеціальними технологічними правилами. Ці правила написані на мові HTML (HyperText Markup Language). Можна вважати, що мова розмітки HTML основа усіх в Інтернеті електронних документів. Ця мова як фундамент, на базі котрого інші мережні програмні продукти, необхідні для підвищення

загальної привабливості, ефективності і інтерактивності носіїв інформаційних даних у мережі [5].

HTML це не зовсім мова програмування. Це мова розмітки електронних документів, та тільки показує браузерам HTML-сторінок необхідну форму подання записаної у документі інформації.

HTML – це мова, що описує структуру веб-сторінок. Він дозволяє розробникам:

- опублікувати онлайн документ з заголовком, текстом, таблицею, фотографією;
- отримати онлайн дані по гіпертекстових посиланнях, натискаючи на кнопку;
- проектувати форми для того, щоб проводити операції з віддаленим сервісом, також для пошуку інформації, можливості зарезервувати замовлення продукції;
- підключити електронну таблицю, відео, аудіо та багато інших додатків з цих документів.

З HTML можна описати структуру сторінок, за допомогою розмітки.

Під гіпертекстом розуміють текст, що сформован мовою розмітки [5].

1.2 Каскадні таблиці стилів CSS

CSS (Cascading Style Sheets) – називають каскадними таблицями стилів, це технологія, за допомогою якої можна описати зовнішній вигляд документу. Переважне застосування – для оформлення веб-сторінок у форматі HTML.

CSS використовують творці веб-сторінок для того, щоб задавати кольори, шрифти, розташовувати інші аспекти представлення документу. Основна мета розробки CSS це розділення змісту який написано на HTML і представлення документа, що написано на CSS. Це розподілення може збільшити доступність документу, надавати гарну гнучкість та можливість керування цим поданням, а також зменшувати складність в структурному вмісті. Також, за допомогою CSS

можна зробити той самий документ у різноматнітних стилях та методах виведення, це й екранне уявлення і друк, і прочитання в голос [6].

1.3 Java та MySQL

Напочатку Java була універсальною мовою програмування. За призначенням ця мова не була для веб-додатків. Проте, багато користувачів вважають цю мову однією з найкращих мов веб-програмування для ознайомлення та щоб використовувати [3, 7].

Мова програмування Java не тільки упростила появу програм для Інтернет, але й вплинула на появу нових прикладних програм, які призначені для працювання у мережі і називаються аплетами, які поміняли саме поняття змісту мережного середовища. Окрім того, ця мова допомогла знайти вирішення двох найбільших проблем програмування, пов'язаних з Інтернет: переносимість та безпека [3].

Розробники провели узагальнення основних понять Java та написала перелік її особливості:

- простота;
- безпеку;
- переносимість;
- об'єктна орієнтованість;
- надійність;
- многопоточность;
- архітекгурная нейтральність;
- Java інтерпретуємість та висока продуктивність;
- динамічність.

Програми, які зроблені на Java мають певний обсяг інформації динамічного типу, що використовуються для того, щоб перевірити повноваження та дозволити доступ до об'єкту під час виконання. Це дає змогу з безпекою та раціональністю робити динамічне зв'язування коду. Це досить

необхідно для того, щоб була стійкість середовища Java, де маленьки фрагменти даних байт-коду мають можливість постійно автоматично оновлюватись у існуючій мережі [8].

Одна з найпопулярніших мов це PHP, яка береться у поєднанні з MySQL. Проте на Java більш приваблива альтернатива для додатків, яким необхідна більша масштабованість і відповідність до стандартів. Java JDBC, це інтерфейс програми, який залежить від баз даних, має добрі методи для виконання різних типів взаємодії із базами даних простору Java та має гарну підтримку для того, щоб обробляти знаявні програми.

JDBC – це стандартний прикладний інтерфейс (API) на Java, необхідний щоб організувати взаємодію між додатком та СУБД. Драйвери JDBC допомагають провести цю взаємодію. В JDBC є декілька типів драйверів, які наведено у табл. 1.1

Таблиця 1.1 – Драйвери JDBC

Типи драйверів	
Тип 1	Драйвер, який використовує інший прикладний інтерфейс, зокрема ODBC (Open Database Connectivity – програмний інтерфейс (API) доступу до баз даних), для роботи з СУБД (так званий JDBC-ODBC – міст). Стандартний драйвер першого типу sun.jdbc.odbc.JdbcOdbcDriver входить в JSDK (Java Servlet Development Kit).
Тип 2	Драйвер, який працює через нативні бібліотеки (тобто клієнта) СУБД.
Тип 3	Драйвер, який працює з мережевого і незалежного від СУБД протоколу з проміжним Java-сервером, який, в свою чергу, підключається до потрібної СУБД.
Тип 4	Мережний драйвер, який працює безпосередньо з потрібною СУБД і не вимагає установки native-бібліотек.

Зазвичай, перевага надається другому типу, але якщо додаток робиться на машині, де не має клієнта СУБД, то перевага надається третьому або четвертому типу. Перший тип використовують не частно, тільки коли СУБД немає свій драйвер JDBC, проте має драйвер ODBC.

Стандарт J2EE дає засіб використання Java у комерційному та відкритому веб-сервері або сервері додатку, щоб створити масштабовані та ефективні веб-додатки, які будуть використовувати MySQL у якості серверу баз даних[3].

Об’єднання Java та баз даних MySQL досить зручний підхід до динамічного веб-конструювання, яке дає змогу зробити сайти, які налаштовуються та веб-додатки, які мають інформацію, яка змінюється у реальному часі [3, 9].

MySQL має швидку, надійну систему керування реляційними базами даних (РСКБД). Ця база даних дає змогу ефективного зберігання, пошуку, сортування і отримування даних. Сервер MySQL керує доступом до інформації, дозволяє працювати з нею одночасно кільком користувачам, а також дає змогу забезпечення швидкого доступу до даних та дає гарантію надання доступа користувачам. MySQL є багатопотоковим сервером. Він застосовує SQL (Structured Query Language), мову структурованих запитів, яка має використання у всьому світі [9].

1.4 JSP

JSP (Java Server Pages)– є стандартним розширенням Java. Метою JSP є спрощення створення та керування динамічними Web сторінками.

Сторінка JSP може забезпечити розподіл динамічних та статичних частин сторінки, як результат – можливість змінити дизайн сторінки, при цьому не зачіпати динамічний зміст. Це використовують при розроблянні та підтримуванні сторінок [10].

Java Server Pages – є однією з технологій J2EE, яка надає представлення розширення технологій сервлетів для того, щоб спростити роботу з Web-

змістом. Сторінка JSP дає можливість з Web-вмісту зробити дву частини, це статична та динамічна частина, яка дає можливість багаторазового використання деяких компонентів.

Під динамічним та статичним змістом розуміють динамічні ресурси (сервлети і JSP) та статичні ресурси (HTML, JavaScript).

Сторінки JSP групируються web-контейнером в сервлети – Java-класи та робляться на сервері [10].

Веб-контейнер має у складі веб-серверу. За допомогою веб-серверу можна звернутися до ресурсів на інших серверах. Це може бути і статичні ресурси, і динамічні. Динамічні ресурси мають бути у середині веб-додатку, отже говорять сервер додатків (application server), проте і веб-сервер також.

На теперішній час найпопулярніші веб-сервера це: Apache Tomcat, Jetty, Oracle GlassFish, Oracle WebLogic, IBM WebSphere, JBoss WildFly.

В усіх цих серверах є базова функціональність, але вони можуть відрізнятися за конфігурацією та додатковими можливостями. Тому, коли додаток потребує використанняє веб-технологій, то необхідно обирати один із варіантів [11].

Також при виборі веб-серверу необхідне і оточення. Якщо розробнику необхідно проводити запуск і тестування програм зі свого комп'ютера, тоді достатньо мати базову функціональность. Або ж веб-додаток необхідно запустити на production сервері, то необхідні інші можливості, це: безпека, керування ресурсами, гнучка настройка і розгортання (deployment), моніторинг додатків, кластеризація (clustering) і масштабованість, розподіл навантаження (load balancing) і відмовостійкість (failover) [11].

У даному проекті необхіден легкий веб-сервер, котрий буде швидко і легко налаштувати для запуску. Підходить Apache Tomcat чи Jetty. Для розробки веб-додатку обраний Tomcat. Критерії за якими обрано цей сервер:

- стабільність і надійність;

- підтримка останніх версій всіх веб-технологій (Servlets, JSP, EL, Web Sockets);

- наявність плагінів під Maven і інші системи збирання;

- простота налаштування.

Порядок обробки сервером JSP сторінок показаний на рис. 1.1. Коли відкривається сторінка jsp браузер надсилає http запит до серверу "GET /page_name.jsp". Сервер (WEB-контейнер) згідно запиту компілює JSP сторінку та робить Java сервлет, котрий повертається користувачеві як відповідна сторінка. Ця процедура має 6 кроків: запит від користувача; прочитання jsp сторінки сервером; генерацію java класу на основі jsp сторінки; компіляцію в class файл; виконання class файлів; отримання відповіді користувачем у вигляді html сторінки [12].

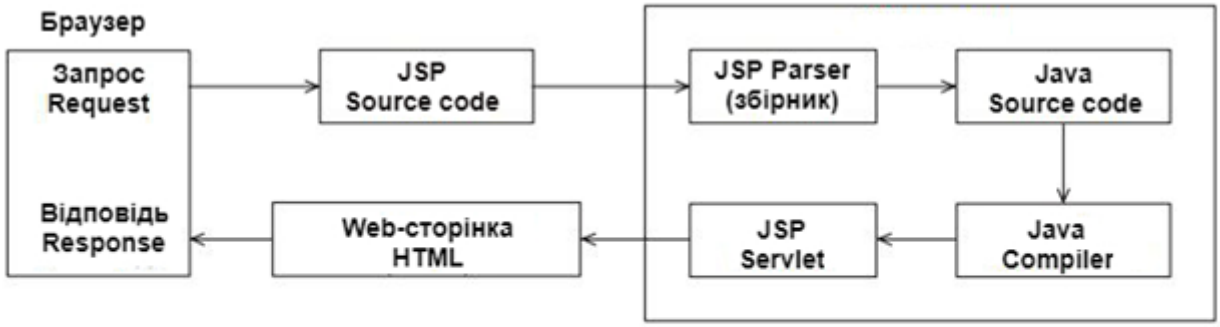


Рисунок 1.1 – Робочий цикл JSP

Формування class файлу, тобто перетворення в сервлет, сервер робить один раз, коли зверненя вперше до jsp сторінки. Якщо ж наступні звернення, то сервер робить скомпільований сервлет [3].

Структура JSP сторінки – це як середнє між сервлетом та HTML сторінкою. JSP-код Java полягає в спеціальних тегах, які вказують контейнеру, щоб він мог використати код, щоб згенерувати сервлет чи його частину. Також є документ, що зразу ж містить і сторінку, і код Java, що управляє цією сторінкою.

JSP включає усі стандартні XTMЛ теги, JSP-теги та призначені для користувача ЖСП теги. У специфікації JSP 1.1 є шість основних тегів:

```
<% @ Директива%>
<%! оголошення%>
<% Скріптлет%>
<% = Обчислюється вираз%>
<% - JSP-коментар -%>
<! - HTML-коментар ->
```

Маємо дуже простий JSP приклад, який має звичайний Java виклик для того, щоб отримати поточний час в мілісекундах, який надалі ділиться на 1000 для того, щоб отримати час у секундах. Тому що використовується JSP вираз (<% =), результат обчислень змінюється у рядок та далі у сгенерировану Web сторінку:

```
<html>
    <body>
        <h1> The time in seconds is:
<%= System.currentTimeMillis()/1000%>
        </h1>
    </body>
</html>
```

Коли створюється запит до JSP сторінки, web-сервер необхідно налаштувати, щоб направити запит на JSP контейнер, який далі додає сторінку [12].

ВИСНОВКИ ДО РОЗДІЛУ 1

У даному розділі розглянуто основи web-програмування, технології і засоби, які використовують для веб-програмувані (HTML, CSS, Java). Проведен детальний аналіз цих технологій, розглянуто етапи розробки веб-додатку та проведено работу з програмним забезпеченням.

2 СТВОРЕННЯ ВЕБ-ДОДАТКУ

2.1 Структура та дизайн веб-додатку

Логічна структура веб-додатку готелю представлена на рис. 2.1.

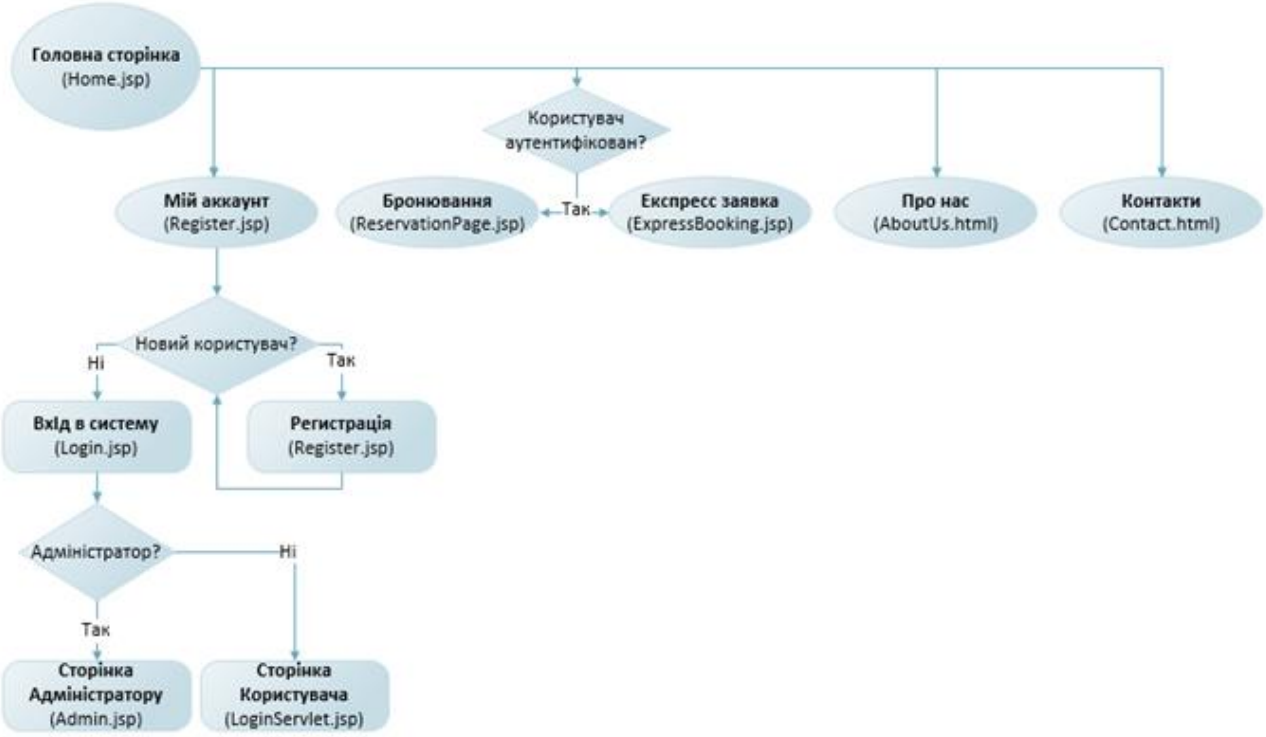


Рисунок 2.1 – Структура веб-додатку

В табл. 2.1 описані основні функції розділів головної сторінки.

Таблиця 2.1 – Розділи головної сторінки

Розділи	Пояснення
«Головна сторінка»	рекурсія на саму себе
«Мій аккаунт»	містить форму для реєстрації/аутентифікації користувача в системі

Продовження табл. 2.1

«Бронювання»	містить таблицю з інформацією про номери (а саме про ціну, кількість мість, класс та статус), а також дозволяє зробити бронь доступних номерів
«Експес заявка»	містить форму для оформлення швидкої заявки
«Про нас», «Контакти»	містить загальну інформацію

За наведеною сторінкою спроектована головна сторінка сайту і зображена на рис.2.2. Вона має основні розділи-сторінки веб-додатку, перехід до яких можна здійснити за посиланнями. Для того, щоб створити логотип та додаткові елементи меню залучалось програмне забезпечення Adobe Photoshop (рис.2.3).

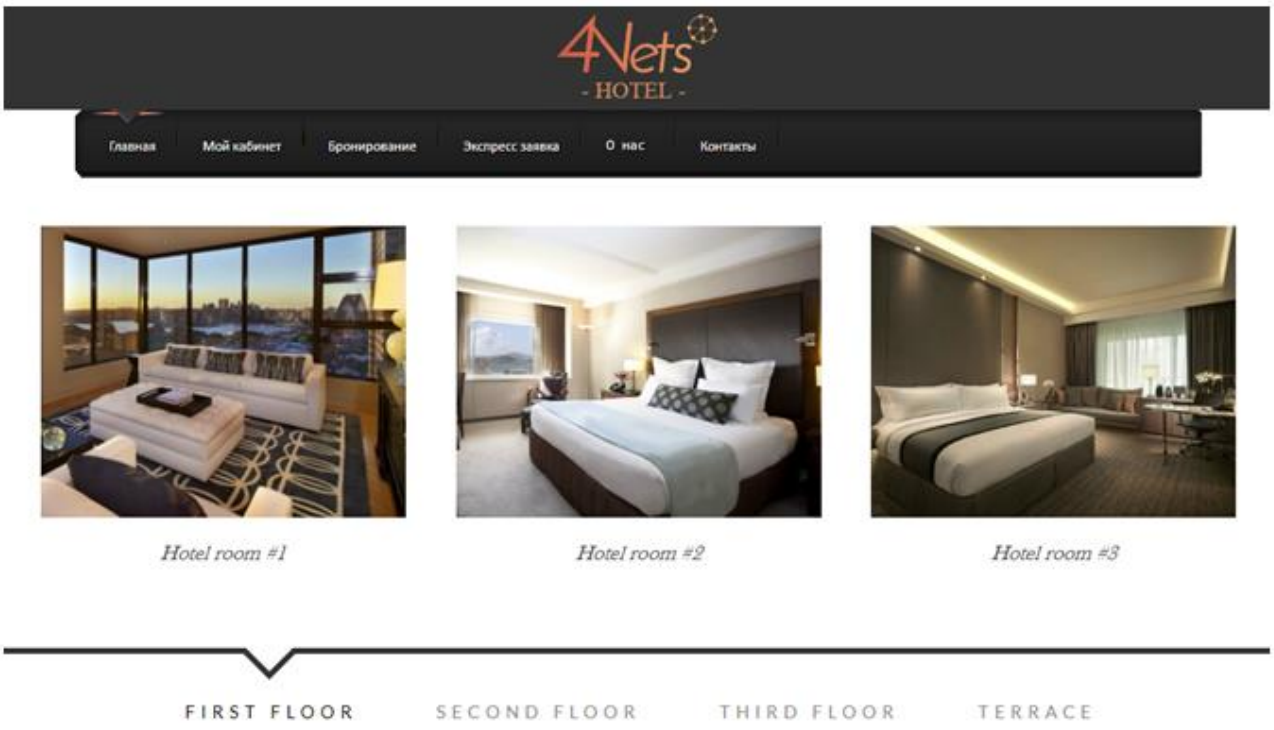


Рисунок 2.2 – Головна стрінка

Дизайн веб-додатку або веб-сайту ґрунтувався завжди на двох складових: зпроектування інтерфейсу і художній дизайн.

Для того, щоб створити веб-сторінку на початку розпочинають працювати з дизайном, тобто із проектуванням зовнішнього вигляду себ-сайту, а саме, інтерфейса. Інтерфейс веб-сайту має бути доступним та легким для використання користувача, а найголовніше, необхідно, щоб він відповідав завданню, яке стоїть перед веб-сайтом [13].

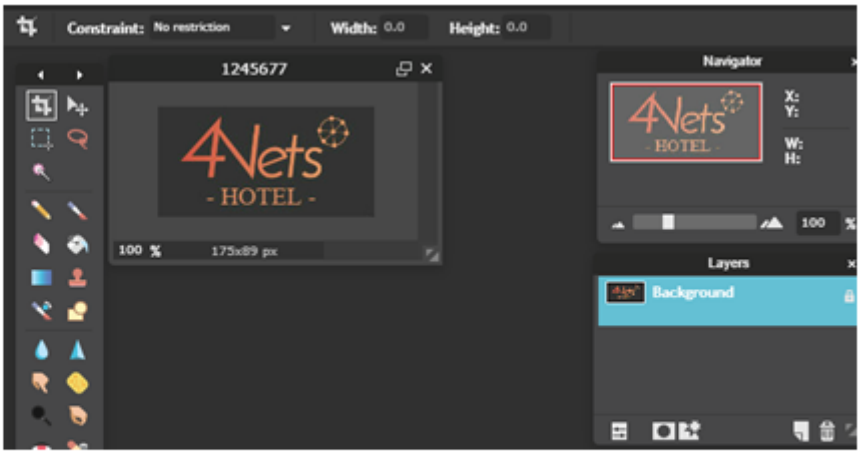


Рисунок 2.3 – Створення логотипу

У даній розробці основна сторінка з каталогом (рис. 2.2), розроблена, щоб приваблювати відвідувачів та не відволікати їх, та наступні сторінки зроблені у зпрощеному стилі та мають призначення донести інформацію про готель і контактні дані (рис. 2.4) [14].

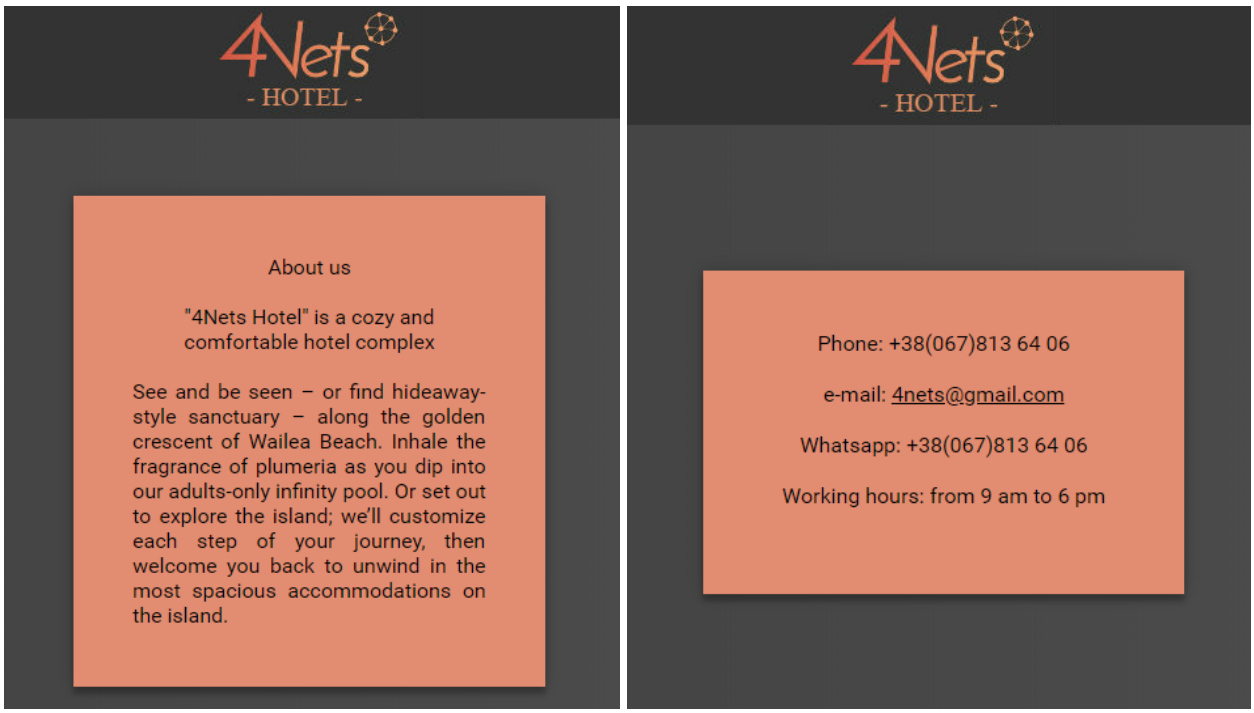


Рисунок 2.4 – HTML-сторінки «Про нас» та «Контакти»

2.2 Форми регістрації та аутентифікації користувачів

У деяких додатках використовують не сервлети чи JSP, а їх поєднання. У JSP відповідає за те, як буде виглядати результат, а сервлет за виклики класу бізнес-логіки та відправлення результатів з виконанням бізнес-логіки у відповідні JSP. Отже сервлети не можуть зенерувати відповідь, а можуть проконтролювати запити. Така процедура створювання додатків називається MVC (Model-View-Controller). Model – клас бізнесу логікі та тривалішого зберігання, View – сторінка JSP, Controller – сервлет [2, 15].

На рис 2.5 показані дві форми для того, щоб заповнити дані. Ці форми зроблені за допомогою HTML та CSS. Для того, щоб зареєструватись користувачу необхідно записати своє ім'я, email, нікнейм та пароль. Якщо користувач вже зареєстровався, тож може аутентифікуватись у вкладці «Логін».

Рисунок 2.5 – Форми аутентифікації та регистрації

З архітектурою Model View Controller (MVC) можна робити три речі: логіка програм (Model), показ усіх даних програми користувачу (View), роюити обробку введення та дій користувача (Controller). Також може бути гарним поясненням за діаграмою, що на рис. 2.6 [16].

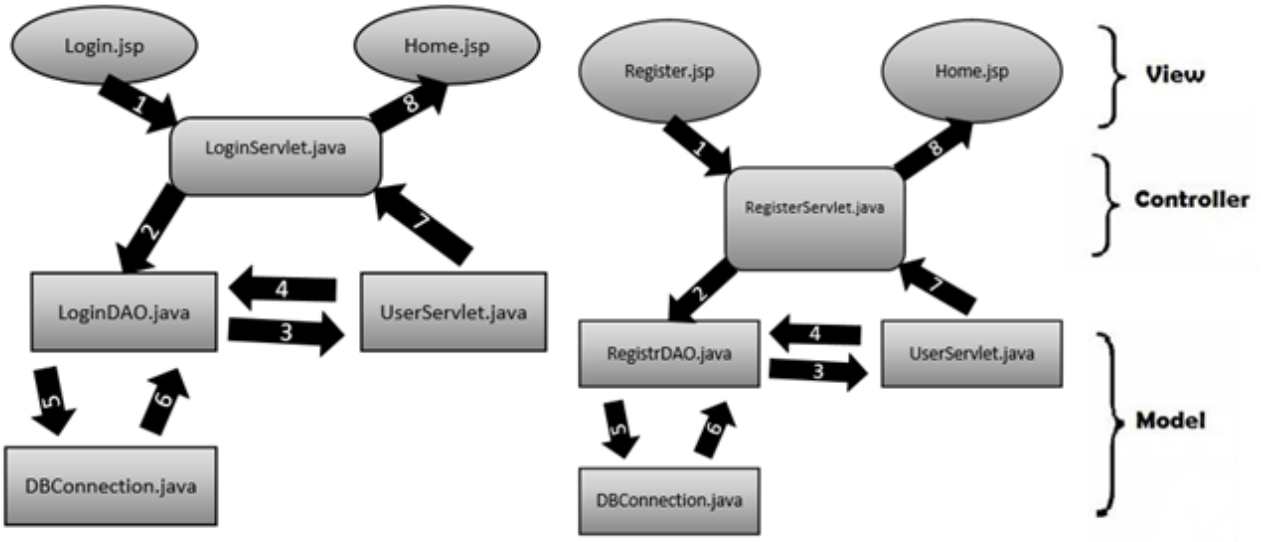


Рисунок 2.6 – Діаграми архітектур MVC для процесу аутентифікації та регистрації користувачів

З архітектурою MVC зв’язані і форми експрес-заявок та систем бронювання номеру (рис. 2.7).

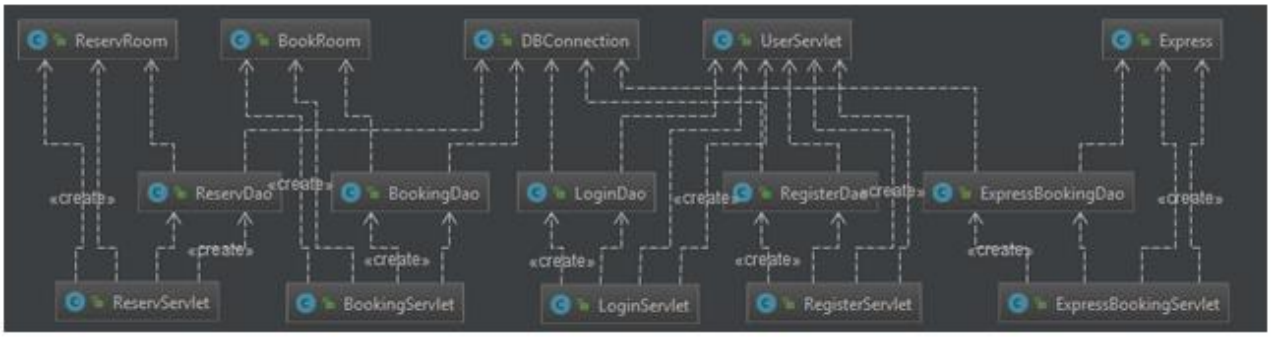
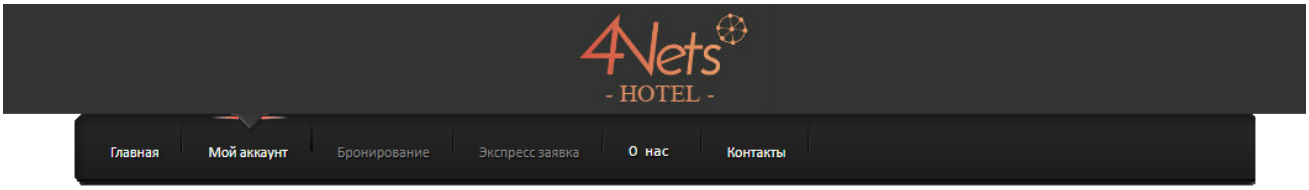


Рисунок 2.7 – Діаграми залежностей JSP та сервлетів веб-додатку

Для прикладу розглядаємо програмний код реєстрації нового користувача у системі. Сеанс (сесія) необхіден, щоб з'єднати клієнт і сервер, який встановлюють на певний час, за котрий користувач має можливість додати до серверу безліч запитів. Сеанси використовують, щоб забезпечити зберігання даних, коли проводяться запити Web-сторінки чи обробки інформації, яку записано у форму користувача. У веб-додатку нікнейм має записуватися до сессії, коли користувач підключається до системи. Щоб це зробити, користувачу необхідно перейти з головної сторінкі веб-сайту у розділ «Мій аккаунт» (рис. 2.8) [17].



```jsp
<%@ page import="ua.nure.faryha.fp.UserServlet" %><%--
  Created by IntelliJ IDEA.
  User: Julia
  Date: 05.11.2018
  Time: 12:34
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>


<html>
<title>Главная</title>
<header id="header">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <link href="css/index1.css" rel="stylesheet" type="text/css">
    <link rel="stylesheet" href="font-awesome-4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/modernizr.custom.63321.js"></script>
    <img src="images/1245677.jpg"/>
</header>

<body>
```

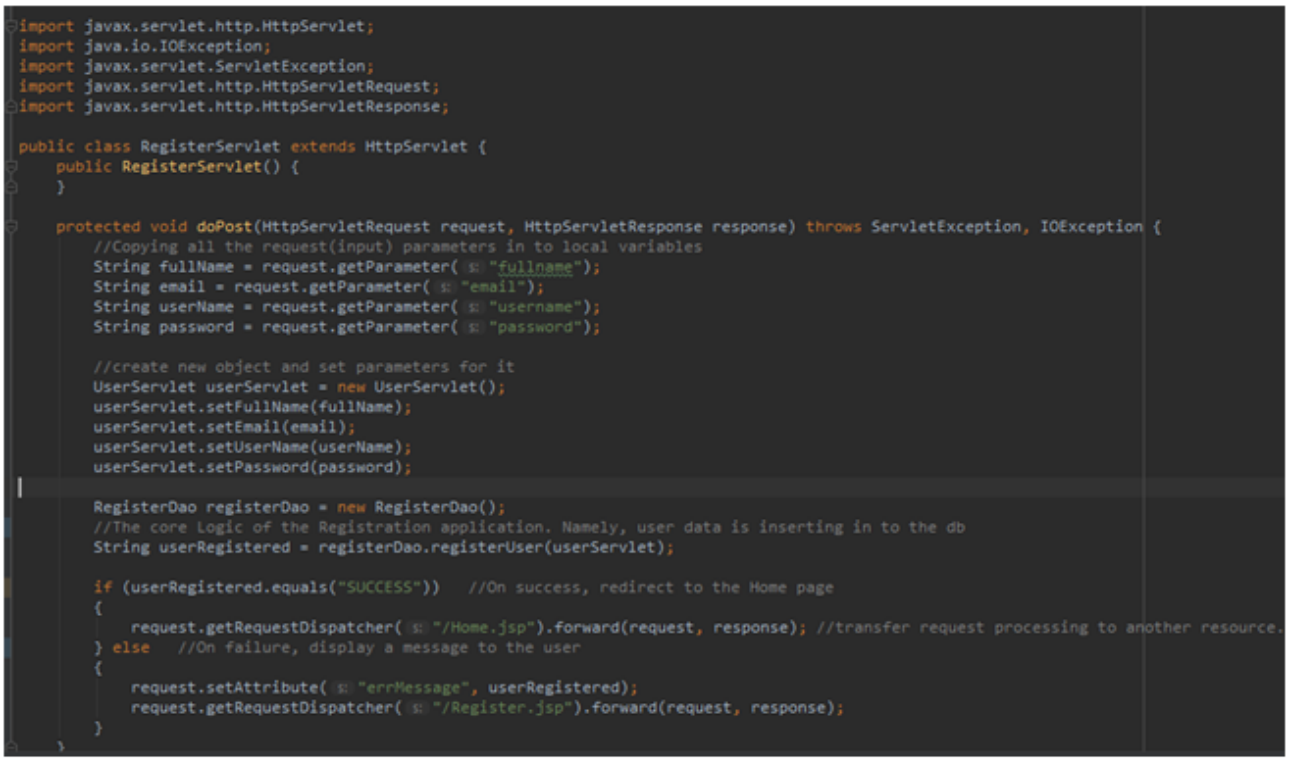
```jsp
<nav>
    <ul id="nav">
        <li><a href="Home.jsp">Главная</a></li>

        <!-- If user is found, he can go to the user cabinet; if not - to registration/login form" -->
        <% if (session.getAttribute("User") != null) {%>
        <li><a href="JSP/User.jsp"> Мой кабинет</a></li>
        <%} else {%>
        <li><a href="Register.jsp">Мой аккаунт</a></li>
        <% }%>

        <!-- If user is found , he can go to the booking and express booking pages;
        otherwise - pages are not available" -->
        <% if (session.getAttribute("User") != null) {%>
        <li><a href="ReservationPage.jsp?par=8">Бронирование</a></li>
        <li><a href="ExpressBooking.jsp">Экспресс заявка</a></li>
        <%} else {%>
        <li><a href="#" title="сперва войдите в аккаунт" style="...">Бронирование</a></li>
        <li><a href="#" title="сперва войдите в аккаунт" style="...">Экспресс заявка</a></li>
        <% }%>

        <li><a href="AboutUs.html">
            <pre>О нас </pre>
        </a></li>
        <li><a href="contact.html">Контакты</a></li>
        <div id="lavalamp"></div>
    </ul>
</nav>
```

Рисунок 2.8 – Розділи меню та відповідний програмний код Home.jsp

На рис. 2.9 показана форма реїстрації клієнта і її програмний код Register.jsp. Якщо клієнт натискає «Create», система проводить повернення користувача до головної сторінки, у тому випадку, коли дані, що ввели (повне ім’я, email, нікнейм та пароль) правильно оброблені у класі RegisterServlet.java [17].


```
<%--
  Created by IntelliJ IDEA.
  User: Julia
  Date: 05.11.2018
  Time: 11:15
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<header id="header">
    <meta charset="utf-8">
    <link href="css/loginform.css" media="screen" rel="stylesheet">
    <link href='http://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic' rel='stylesheet' type='text/css'>
    <a href="Home.jsp" title="Главная"><img class="logo" src="images/1245677.jpg"/></a>
</header>
<body>
<section>
    <div class="container mregister login-page">
        <div id="login" class="form">
            <form action="RegisterServlet" id="registerform" method="post" name="form" class="login-form">
                <input class="input" id="full_name" name="fullname" size="32"  type="text" value="" placeholder="full name">
                <input class="input" id="email" name="email" size="32" type="email" value="" placeholder="email">
                <input class="input" id="username" name="username" size="20" type="text" value="" placeholder="username">
                <input class="input" id="password" name="password" size="32"   type="password" value="" placeholder="password">
                <input class="input" id="cpassword" name="cpassword" size="32"   type="password" value="" placeholder="confirm password">
                <tr>
                    <td><%=(request.getAttribute("errMessage") == null) ? ""
                            : request.getAttribute("errMessage")%></td>
                </tr>
                <button class="submit button" name="register" type="submit" id="register" value="Register">create</button>
                <p class="regtext message">Already registered? <a href= "JSP/Login.jsp">Sign In</a></p>
            </form>
        </div>
    </div>
</section>

</body>
</html>
```

Рисунок 2.9 – Форма регістрації користувача та відповідний програмний код Register.jsp



```
import javax.servlet.http.HttpServlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class RegisterServlet extends HttpServlet {
    public RegisterServlet() {
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //Copying all the request(input) parameters in to local variables
        String fullName = request.getParameter( s: "fullname");
        String email = request.getParameter( s: "email");
        String userName = request.getParameter( s: "username");
        String password = request.getParameter( s: "password");

        //create new object and set parameters for it
        UserServlet userServlet = new UserServlet();
        userServlet.setFullName(fullName);
        userServlet.setEmail(email);
        userServlet.setUserName(userName);
        userServlet.setPassword(password);

        RegisterDao registerDao = new RegisterDao();
        //The core Logic of the Registration application. Namely, user data is inserting in to the db
        String userRegistered = registerDao.registerUser(userServlet);

        if (userRegistered.equals("SUCCESS"))   //On success, redirect to the Home page
        {
            request.getRequestDispatcher( s: "/Home.jsp").forward(request, response); //transfer request processing to another resource.
        } else   //On failure, display a message to the user
        {
            request.setAttribute( s: "errMessage", userRegistered);
            request.getRequestDispatcher( s: "/Register.jsp").forward(request, response);
        }
    }
```

Рисунок 2.10 – Лістинг RegisterServlet.java

Для підсумку роботи RegisterServlet.java (рис. 2.10) маємо такі етапи [18]:

– вхідні дані клієнта фіксуються у локальні змінні;
– робиться об’єкт відповідно до класу UserServlet.java (рис. 2.11), для встановлення усіх даних користувача через налаштування Java;
– далі робиться перехід до класу RegisterDao.java (рис. 2.12), де дані, що отримано зберігаються і встановлюються до бази даних за допомогою registerUser. RegisterDao.java з'єднується з шаром бази даних jfp_datab у класі DBConnection.java (рис. 2.13) та додає дані про користувача у відповідне поле таблиці до бази даних MySQL (рис. 2.14).
– якщо обробка успішна, то користувача перенаправляють до головної сторінки веб-додатку готелю.

```java
public class UserServlet {
    private String fullName;
    private String email;
    private String userName;
    private String password;
    private int mobile;

    public String getUserName() { return userName; }

    public void setUserName(String userName) { this.userName = userName; }

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }

    public void setFullName(String fullName) { this.fullName = fullName; }

    public String getFullName() { return fullName; }

    public void setEmail(String email) { this.email = email; }

    public String getEmail() { return email; }

    public int getMobNumb() { return mobile; }

    public void setMobile(int mobile) { this.mobile = mobile; }
}
```

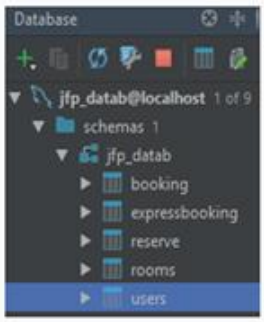
Рисунок 2.11 – Лістинг UserServlet.java

```java
import ...

public class RegisterDao {
    public String registerUser(UserServlet userServlet) {
        //Copying all the get parameters in to local variables
        String fullName = userServlet.getFullName();
        String email = userServlet.getEmail();
        String userName = userServlet.getUserName();
        String password = userServlet.getPassword();

        Connection con = null;
        PreparedStatement preparedStatement = null;

        try {
            con = DBConnection.createConnection();
            String query = "insert into users(id,fullName,email,username,password) values (NULL,?,?,?,?)"; //Insert info into the table
            preparedStatement = con.prepareStatement(query); //Making use of prepared statements here to insert bunch of data
            preparedStatement.setString( parameterIndex: 1, fullName); //Sets the designated parameter to the given Java String value.
            preparedStatement.setString( parameterIndex: 2, email);
            preparedStatement.setString( parameterIndex: 3, userName);
            preparedStatement.setString( parameterIndex: 4, password);

            int i = preparedStatement.executeUpdate(); //Returns the number of rows affected by the crud operation as a result.

            if (i != 0)  //Just to ensure that data has been inserted into the users database
                return "SUCCESS";
            con.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return "Oops.. Something went wrong there..!";  // If failure, than send a message
    }
}
```

Рисунок 2.12 – Лістинг RegisterDao.java

```java
package ua.nure.faryha.fp;

import java.sql.Connection;
import java.sql.DriverManager;

public class DBConnection {
    public static Connection createConnection() {
        Connection con = null;
        String url = "jdbc:mysql://localhost:3306/jfp_datab?useUnicode=true&\" +\n" +
                "            \"useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC&" +
                "verifyServerCertificate=false&useSSL=true";
        String username = "root";
        String password = "1234";

        try {
            try {
                Class.forName("com.mysql.jdbc.Driver"); // initialize the Driver class, and in turn the class
                // registers itself with the java.sql.DriverManager per the JDBC specification.
                //in other words, loading MySQL drivers.
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
            con = DriverManager.getConnection(url, username, password); //trying to connect to database
            System.out.println("Post establishing a DB connection - " + con);
        } catch (Exception e) {
            e.printStackTrace();
        }

        return con;
    }
}
```

Рисунок 2.13 – Лістинг DBConnection.java



2	Don Sun	donsun@gmail.com	dons	don1234	User
4	Jen Frank	jenFrank@gmail.com	jennyf	111111	User
5	Jhon Miks	jm@gmail.com	jmy	hht333	User
7	Persik Van	persik@gmail.com	pers	persik1111	User
8	Persik VanTwo	persik2@gmail.com	pers2	persik222	User
13	One one	one@gmail.com	one	one11111	User
14	two two	two@gmail.com	two	two1111	User
27	prov prov	prov@gmail.com	prov	prov111	User
28	epam epam	epam@gmail.com	epam	111111	User
30	jjj	jjj@gmail.com	jjj	jjj	User

Рисунок 2.14 – Локальна база даних jfp_datab та таблиця «users»

Для збереження даних використано бази даних. В додатках Java можуть бути використані різноманітні системи керування базами даних. Проте, на сьогодні, одна з найбільш популярних у додатках Java – СУРБД MySQL [17].

Один із засобів працювання з базою даних це використання JDBC. JDBC (Java Database Connectivity) це спеціалізований інтерфейс додатку, через який можемо використати один інтерфейс з роботою для різних СУБД. Щоб працювати з базою даних у додатку на Java через JDBC необхідно мати драйвер. В проекті використовується драйвер mysql-connector-java-5.1.47.jar.

Загальна схема роботи з базою даних буде такою: налаштувати з'єднання, отримати statement, виконатизапиту, отримати і обробити дані.

Послідовність дій:

– Необхідне завантаження класу драйвера:

String driverName = "com.mysql.jdbc.Driver";

Потім робиться власне завантаження драйвера до пам'яті:

Class.forName (driverName);

Чи менший варіант:

Class.forName ("com.mysql.jdbc.Driver");

і можливез'єднання з СУБД.

– Встановлюється з'єднання з БД:

Connection cn = DriverManager.getConnection ("jdbc:mysql://localhost :3306/ jfp_datab", "login", "password");

Як результат, повертається об'єкт Connection і залишиться одне з'єднання з БД jfp_datab.

Клас DriverManager дає засоби для керування набором драйверів баз даних.

getConnection() треба надіслати тип та фізичне місце знаходження БД, логін та пароль для доступа.

У даному веб-додатку параметри підключення MySQL наступні: Ім'я користувача: root; Пароль: 1234.

За допомогою registerDriver () драйвера можна зареєструвати, а getDrivers () отримується список усіх драйверів.

– Зроблення об'єкту для передачі запитів

Statement st = con.createStatement ();

Statement необхідне, щоб виконати запит без його попередньої підготовки та команди SQL. Можна застосувати і операторів для того, щоб виконати підготовлені запити і збережені процедуру PreparedStatement. Об'єкт, який створено використовується для виконання запиту [18].

– Виконання запиту

Результат виконання запиту поміщається у ResultSet:

ResultSet rs = st.executeQuery ("SELECT * FROM my_table");

Для того, щоб додати чи змінити інформацію у таблиці замість executeQuery() цей запит поміщають у executeUpdate().

– Обробка результатів з виконання запиту робиться методами інтерфейсом ResultSet, де поширеними є next() і getString() та аналогічні методи, що починаються з getТип() і updateТип().

Якщо вдало виконувати ці пункти (вдала регістрація користувача у системі), клієнта буде переправлено на головну сторінку веб-додатку [18].

Далі клієнт робить вхід до системи (рис. 2.15).

```
<%--
  Created by IntelliJ IDEA.
  User: Julia
  Date: 05.11.2018
  Time: 11:16
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<header>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Login</title>
    <link href="../css/loginform.css" media="screen" rel="stylesheet">
    <link href='http://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic' rel='stylesheet' type='text/css'>
    <a href="../Home.jsp"  title="Главная"><img class="logo" src="../images/1245677.jpg"/></a>
</header>
<body>

<section>
    <div class="container mlogin login-page">
        <span class="s"><pre> <%=(request.getAttribute("errMessage") == null) ? "" : request.getAttribute("errMessage")%></pre></span>
        <div id="login" class="form">
            <form action="<%=request.getContextPath()%>
/LoginServlet" id="loginform" method="post" name="form" class="login-form">
                <input class="input" id="username" name="username" size="20" type="text" value="" placeholder="username">
                <input class="input" id="password" name="password" size="20" type="password" value="" placeholder="password">
                <button class="submit button" name="Login" type="submit" value="Login">login</button>
                <button class="reset button" value="Reset" type="reset">Reset</button>
                <!--    <p class="submit"><input class="button" name="login"type= "submit" value="Log In"></p>     -->
                <p class="regtext message">Not registered?<a href= "../Register.jsp">Create an account</a></p>
            </form>
        </div>

    </div>
</section>
</body>
</html>
```

Рисунок 2.15 – Форма входу та відповідний лістинг коду Login.jsp

Якщо аутентифікація вдала, клієнт може перейти до особистого кабінету, в якій міститься таблиця бронювання та експрес заявки користувача (рис. 2.16) [18].

Рисунок 2.16 – Власний кабінет користувача

```
<p>Your booking list:</p>
<% String usr = (String) session.getAttribute("User");
    String ordr = "";
    String chIn = "";
    String chOt = "";
    String rn = "";
    Connection con = null;
    Statement statement = null;
    ResultSet resultSet = null;
    con = DBConnection.createConnection();
```

```
    try {
        statement = con.createStatement();
        String str = "select orderNumber,checkIn,checkOut,roomNumb " +
                "from booking where nameOfUser = ('" + usr + "');";
        resultSet = statement.executeQuery(str);
```

```
        while (resultSet.next()) {
            try {
                ordr = resultSet.getString("orderNumber");
                chIn = resultSet.getString("checkIn");
                chOt = resultSet.getString("checkOut");
                rn = resultSet.getString("roomNumb");
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
%>
```

```
<table border="1" style="...">
    <tr>
        <td style="...">order number</td>
        <td style="...">check in</td>
        <td style="...">check out</td>
        <td style="...">room number</td>
    </tr>
    <tr>
        <td><p style="..."><%=ordr %>
        </p></td>
        <td><p style="..."><%=chIn %>
        </p></td>
        <td><p style="..."><%=chOt %>
        </p></td>
        <td><p style="..."><%=rn %>
        </p></td>
    </tr>
</table>

<!-- -------------------- ---->
</br>
<p>Your express orders list:</p>
```

```
<%
    String idr = "";
    String rnum = "";
    String mes = "";
    String fn = "";
    try {
        statement = con.createStatement();
        String str = "select fullname from users where username = ('" + usr + "');";
        resultSet = statement.executeQuery(str);
        while (resultSet.next()) {
            try {
                fn = resultSet.getString("fullname");
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    try {
        statement = con.createStatement();
        String str = "select idreserve,roomNum,message from reserve where userFN= ('" + fn + "');";
        resultSet = statement.executeQuery(str);
        while (resultSet.next()) {
            try {
                idr = resultSet.getString("idreserve");
                rnum = resultSet.getString("roomNum");
                mes = resultSet.getString("message");
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
%>
```

```
<table border="1" style="...">
    <tr>
        <td style="...">id of reserv</td>
        <td style="...">room number</td>
        <td style="...">message</td>
    </tr>
    <tr>
        <td style="..."><%=idr %>
        </td>
        <td style="..."><%=rnum %>
        </td>
        <td style="..."><%=mes %>
        </td>

    </tr>
</table>


<button style="...">
    <a style="..." href="<%=request.getContextPath()%>/LogoutServlet">Logout</a>
</button>
</div>
</div>
</div>

</body>
</html>
```

Рисунок 2.17 – Лістинг User.jsp («Особистий кабінет»)

Процедура аутентифікації за архітектурою MVC дуже схожа до процедури регістрації, проте у ньому фігурують Login.jsp (рис. 2.15), LoginDao.java (рис. 2.18) та LoginServlet.java (рис. 2.19) [19].

```java
public class LoginDao {
    public String authenticateUser(UserServlet userS) {
        String userName = userS.getUserName();
        String password = userS.getPassword();

        Connection con = null;
        Statement statement = null;
        ResultSet resultSet = null;

        String userNameDB = "";
        String passwordDB = "";
        String roleDB = "";

        //create connection, taking user parameters, and check if it is admin or user
        try {
            con = DBConnection.createConnection();
            statement = con.createStatement();
            resultSet = statement.executeQuery( sql: "select username,password,role from users");

            while (resultSet.next()) {
                userNameDB = resultSet.getString( columnLabel: "username");
                passwordDB = resultSet.getString( columnLabel: "password");
                roleDB = resultSet.getString( columnLabel: "role");

                if (userName.equals(userNameDB) && password.equals(passwordDB) && roleDB.equals("Admin"))
                    return "Admin_Role";
                else if (userName.equals(userNameDB) && password.equals(passwordDB) && roleDB.equals("User"))
                    return "User_Role";
            }
            con.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return "Invalid user credentials";
    }
}
```

Рисунок 2.18 – Лістинг LoginDao.java

```java
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public LoginServlet() {
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //get request parameters and set them for local variables
        String userName = request.getParameter( s: "username");
        String password = request.getParameter( s: "password");

        //create new user and set name and passw for it
        UserServlet userS = new UserServlet();

        userS.setUserName(userName);
        userS.setPassword(password);

        LoginDao loginDao = new LoginDao();

        try {
            String userValidate = loginDao.authenticateUser(userS);

            if (userValidate.equals("Admin_Role")) {
                System.out.println("Admin's Home");

                HttpSession session = request.getSession(); //Creating a session
                session.setAttribute( s: "Admin", userName); //setting session attribute
                request.setAttribute( s: "userName", userName);

                request.getRequestDispatcher( s: "/JSP/Admin.jsp").forward(request, response);
            } else if (userValidate.equals("User_Role")) {
                System.out.println("User's Home");
```

```
                HttpSession session = request.getSession();
                session.setMaxInactiveInterval(10 * 60);
                session.setAttribute( s: "User", userName);
                request.setAttribute( s: "userName", userName);

                request.getRequestDispatcher( s: "/JSP/User.jsp").forward(request, response);
            } else {
                System.out.println("Error message = " + userValidate);
                request.setAttribute( s: "errMessage", userValidate);

                request.getRequestDispatcher( s: "/JSP/Login.jsp").forward(request, response);
            }
        } catch (IOException e1) {
            e1.printStackTrace();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
}
```

Рисунок 2.19 – Лістинг LoginServlet.java

У процедурі аутентифікації нікнейм клієнта записується у сесію. Це зроблене для того, що коли користувач переходе до цншого розділу веб-додатку, сервер знає хто саме робе процедуру бронювання, і там підтримується сеанс з користувачем. Як відомо, HTTP – є протоколом без статуса, це значить, що зв'язок між сервером та браузером закінчується, коли транзакція завершена. І отже не має можливості відстежити, хто насправді зробив запит та коли цей запит припинено. Сеанс має допомогти підтримати зв’язок між клієнтом і сервером. Обидва HTTP і веб-сервер не мають статуса, тому, щоб створити сеанс використовують інформацію (sessionID) [20].

HTTP сесія має інформацію, яка має зберігатись на сервері з конкретним користувачем. Сесію була отримано викликанням на HttpSessionRequest методу getSession:

HttpSession s = request.getSession ();

За відповідними методами set/getAttribute встановлюються та визначаються нікнейм користувача у системі.

– setAttribute (String name, Object value) має атрибут value під ім'ям name;
– Object getAttribute (String name) повертає атрибут на ім'я.

Для того, щоб вийти з системи застосовують LogoutServlet (рис. 2.20). Примусово зробити завершення сеансу можливо методом invalidate(). Для цього необхідно знайти поточну сесію, потім має застосуватися цей метод і за результатом цього метода, для сеансу удаляються усі зв'язки з об'єктами, що

використовувались, і дані, що були збережені мають бути втрачені для користувача та усіх додатків [21].

```java
package ua.nure.faryha.fp;

import javax.servlet.http.HttpServlet;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class LogoutServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        HttpSession session = request.getSession( b: false); //Fetch session object

        if (session != null) //If session is not null
        {
            session.invalidate(); //removes all session attributes bound to the session
            request.setAttribute( s: "errMessage",  o: "You have logged out successfully");
            RequestDispatcher requestDispatcher = request.getRequestDispatcher( s: "/JSP/Login.jsp");
            requestDispatcher.forward(request, response);
            System.out.println("Logged out");
        }
    }
}
```

Рисунок 2.20 – Лістинг LogoutServlet.java

2.3 Бронювання

Після вдалої аутентифікації, клієнт може зробити бронювання номеру готелю та оформлення експресс-заявки (яка потім розглядається менеджером).

Клієнт має можливість зайти у розділ «Бронювання» на головній сторінці веб-додатку [22].

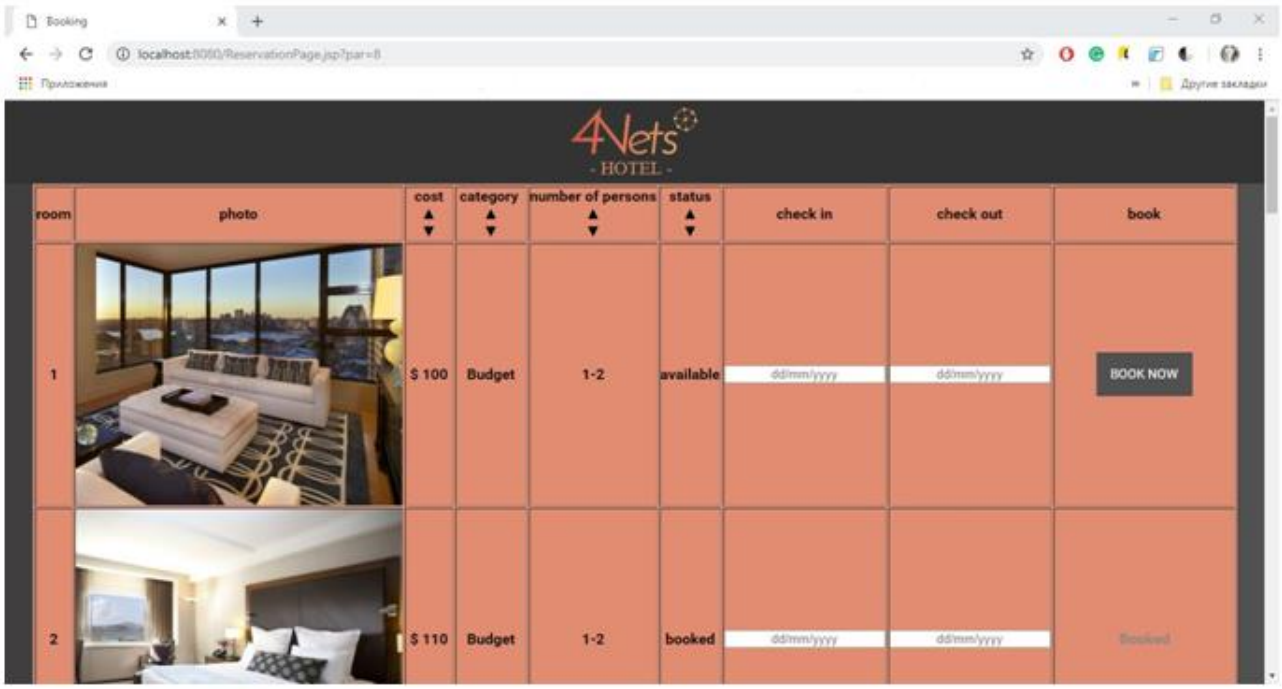


Рисунок 2.21 – «Бронювання»

У вкладці бронювання номеру, користувач має можливість переглянути усі номери і відсортувати номери [23].

Крім того, клієнт має можливість зробити бронювання доступного номеру. Цей процес зроблений на архітектурі MVC (рис 2.22).

Щоб сортувати, до URL гіперпосилань були додані параметри 0-8 (у форматі: /ReservationPage.jsp?par=5) (рис. 2.23).

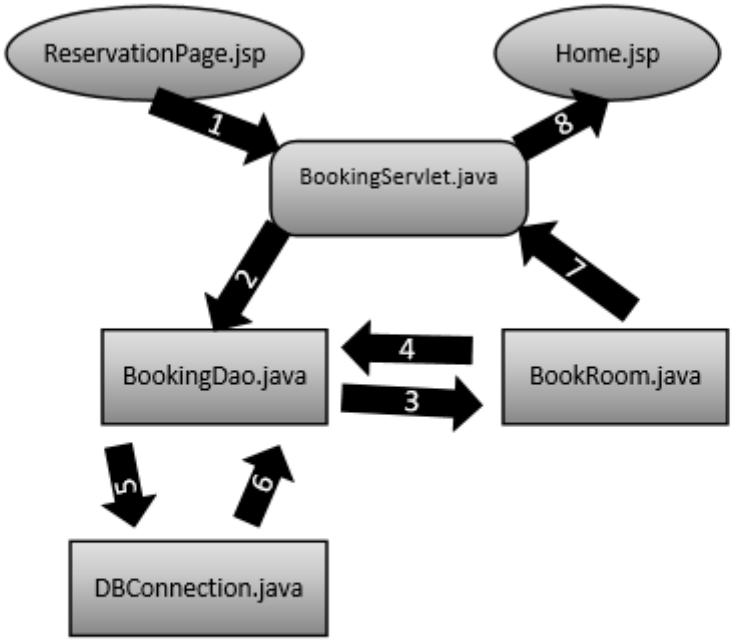


Рисунок 2.22 – MVC архітектура процесу бронювання номерів

```
<%@ page import="ua.nure.faryha.fp.DBConnection" %>
<%@ page import="java.sql.*" %>
<%--
  Created by IntelliJ IDEA.
  User: Julia
  Date: 13.11.2018
  Time: 16:41
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<header id="header">
    <title>Booking</title>
    <link href="css/contact.css" rel="stylesheet" type="text/css">
    <link href="css/loginform.css" media="screen" rel="stylesheet">
    <link href="css/reservation.css" rel="stylesheet" type="text/css">
    <h1>
        <a href="Home.jsp"><img class="logo" src="images/1245677.jpg"/></a>
    </h1>
</header>
```

```
<body>
<table border="1">
    <tr>
        <th> room</th>
        <th> photo</th>
        <th>cost<br>
            <a href="ReservationPage.jsp?par=0">&#9650;</a><br> <!--ascending-->
            <a href="ReservationPage.jsp?par=1">&#9660;</a><!--descending-->
        </th>
        <th>category <br>
            <a href="ReservationPage.jsp?par=2"> &#9650;</a><br>
            <a href="ReservationPage.jsp?par=3"> &#9660;</a>
        </th>
        <th>number of persons<br>
            <a href="ReservationPage.jsp?par=4"> &#9650;</a><br>
            <a href="ReservationPage.jsp?par=5"> &#9660;</a>
        </th>
        <th>status<br>
            <a href="ReservationPage.jsp?par=6">&#9650;</a><br>
            <a href="ReservationPage.jsp?par=7">&#9660;</a>
        </th>
        <th> check in</th>
        <th> check out</th>
        <th> book</th>
    </tr>
```

```
    <%
        //sorting by the request parameter
        int x = Integer.parseInt(request.getParameter("par"));
        String s = "";
        switch (x) {
            case 0:
                s = "SELECT * FROM rooms ORDER BY price";
                break;
            case 1:
                s = "SELECT * FROM rooms ORDER BY price DESC";
                break;
            case 2:
                s = "SELECT * FROM rooms ORDER BY category";
                break;
            case 3:
                s = "SELECT * from rooms ORDER BY category DESC";
                break;
            case 4:
                s = "SELECT * from rooms ORDER BY numberOfPersons";
                break;
            case 5:
                s = "SELECT * from rooms ORDER BY numberOfPersons DESC";
                break;
            case 6:
                s = "SELECT * from rooms ORDER BY status";
                break;
```

```
            case 7:
                s = "SELECT * from rooms ORDER BY status DESC";
                break;
            default:
                s = "SELECT * FROM rooms"; //if par==8, sorting by price
        }

        try {
            Connection connection = DBConnection.createConnection();
            PreparedStatement pstmt = connection.prepareStatement(s); //"s" from certain case
            ResultSet rs = pstmt.executeQuery(s);
            while (rs.next()) {
                //Copying all the parameters in to local variables
                String roomid = rs.getString("roomID");
                String roomprice = rs.getString("price");
                String roomcategory = rs.getString("category");
                String numOfPers = rs.getString("numberOfPersons");
                String stat = rs.getString("status");
    %>
```

```
    <!-- finding images for each room -->
    <tr>
        <th><% out.print(roomid); %></th>
        <th><%if (roomid.equals("1")) { %> <img src="images/1948.jpg" width="350px" alt="img01"/>
            <% ;
            } else if (roomid.equals("2")) {%> <img src="images/b.jpg" width="350px" alt="img01">
            <% ;
            } else if (roomid.equals("3")) { %> <img src="images/c.jpg" width="350px" alt="img01">
            <% ;
            } else if (roomid.equals("4")) { %> <img src="images/d.jpg" width="350px" alt="img01">
            <% ;
            } else if (roomid.equals("5")) { %> <img src="images/e.jpg" width="350px" alt="img01">
            <% ;
            } else if (roomid.equals("6")) { %> <img src="images/f.jpg" width="350px" alt="img01">
            <% ;
            } else if (roomid.equals("7")) { %> <img src="images/g.jpg" width="350px" alt="img01">
            <% ;
            } else if (roomid.equals("8")) {%> <img src="images/h.jpg" width="350px" alt="img01">
            <% ;
            } else if (roomid.equals("9")) {%> <img src="images/i.jpg" width="350px" alt="img01">
            <% ;
            } else if (roomid.equals("10")) { %> <img src="images/j.jpg" width="350px" alt="img01">
            <% ;
            } else if (roomid.equals("11")) { %> <img src="images/a.jpg" width="350px" alt="img01">
            <% ;
            } else if (roomid.equals("12"))%> <img src="images/k.jpg" width="350px" alt="img01">
        </th>
        <th> $ <% out.print(roomprice); %></th>
        <th><% out.print(roomcategory); %></th>
        <th><% out.print(numOfPers);%></th>
        <th><% out.print(stat);%></th>
```

```
        <form action="BookingServlet" method="post">
            <th><input id="checkIn" name="checkIn" type="text" placeholder="dd/mm/yyyy" style="...">
            </th>
            <th><input id="checkOut" name="checkOut" type="text" placeholder="dd/mm/yyyy" style="...">
            </th>

            <!-- if room is available, then button is available too -->
            <th class="form" style="...">
                <% if (stat.equals("available")) {%>
                <button class="submit button" name="roomID" type="submit" id="roomID" value="<%=roomid%>">Book now
                </button>
                <% } else if (stat.equals("reserved")) {%>
                <p class="submit button" style="...">Reserved</p>
                <% } else { %>
                <p class="submit button" style="...">Booked</p>
            </th>
        </form>
    </tr>
    <% }
    }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    %>
</table>
</body>
</html>
```

Рисунок 2.23 – Лістинг ReservationPage.jsp

На початку рис. 2.23 бачимо стандартний header з підключеннями файлу css, встановленнмя логотипу і сформуванням таблиці із гіперпосиланнями, які передають параметри від 0 до 8. Потім додається java код у вигляді скриплету, який починається з оператору вибору switch і параметрами. Потім класс має під’єднатися до бази даних та записати дані, які отримано у локальні змінні.

З’являється сторінка з ціною, категорією, кількістю персон, та статусом номеру [24].

На BookingServlet.java у рис. 2.24 додаються параметри дати і номеру кімнати (яка буде заброньована) у локальні змінні. Далі робиться екземпляр класу BookRoom, в який за налаштуваннями Java – сеттерами (set) встановлюються поля дати, номеру кімнати, та ім’я клієнта. Далі робиться об’єкт класу BookingDao, на якому викликається booking. Ця функція створює дані локальних змінних, які отримуються через геттери класу BookRoom; створюється зв’язок з базою даних, додаються до таблиці booking відповідні значення та оновлюються статус кімнати як «заброньована», повертаючи повідомлення «SUCCESS» у разі вдалої операції [22].

```
package ua.nure.faryha.fp;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

public class BookingServlet extends HttpServlet {
    public BookingServlet() {
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //set local variables
        String checkIn = request.getParameter( s: "checkIn");
        String checkOut = request.getParameter( s: "checkOut");
        String roomID = request.getParameter( s: "roomID");

        BookRoom bookRoom = new BookRoom();

        //get name of user from the current session
        HttpSession session = request.getSession();
        String nameOfUser = (String) session.getAttribute( s: "User");
        String nou = nameOfUser;

        bookRoom.setCheckIn(checkIn);
        bookRoom.setCheckOut(checkOut);
        bookRoom.setRoomID(roomID);
        bookRoom.setNameOfUser(nameOfUser);

        BookingDao bookingDao = new BookingDao();
        String roombooked = bookingDao.booking(bookRoom);
        if (roombooked.equals("SUCCESS"))   //On success, redirect to home.jsp
        {
            request.getRequestDispatcher( s: "/Home.jsp").forward(request, response);
        } else
            request.getRequestDispatcher( s: "ReservationPage.jsp?par=8").forward(request, response);
    }
}
```

Рисунок 2.24 – Лістинг BookingServlet.java

```java
package ua.nure.faryha.fp;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class BookingDao {
    public String booking(BookRoom bookRoom) {
        //set local variables
        String nameOfUser = bookRoom.getNameOfUser();
        String checkIn = bookRoom.getCheckIn();
        String checkOut = bookRoom.getCheckOut();
        String roomID = bookRoom.getRoomID();

        try {

            Connection con;
            PreparedStatement preparedStatement;
            con = DBConnection.createConnection();

            //Insert user details into the table books
            String query = "insert into booking(orderNumber,nameOfUser, checkIn,checkOut,roomNumb) values (NULL,?,?,?,?)";
            //Making use of prepared statements here to insert bunch of data
            preparedStatement = con.prepareStatement(query);
            preparedStatement.setString( parameterIndex: 1, nameOfUser);
            preparedStatement.setString( parameterIndex: 2, checkIn);
            preparedStatement.setString( parameterIndex: 3, checkOut);
            preparedStatement.setString( parameterIndex: 4, roomID);
            int i = preparedStatement.executeUpdate();

            int rid = Integer.parseInt(roomID);
            String upd = "UPDATE rooms SET status = 'booked' WHERE roomID = (" + rid + ");";
            preparedStatement = con.prepareStatement(upd);
            preparedStatement.executeUpdate();

            if (i != 0)  //Just to ensure data has been inserted into the database
                return "SUCCESS";
            con.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return "Oops.. Something went wrong there..!";  // On failure, send a message
    }
}
```

Рисунок 2.25 – Лістінг BookingDao.java

```java
package ua.nure.faryha.fp;

public class BookRoom {

    private String nameOfUser;
    private String checkIn;
    private String checkOut;
    private String roomID;

    public void setNameOfUser(String nameOfUser) { this.nameOfUser = nameOfUser; }

    public String getNameOfUser() { return nameOfUser; }

    public void setCheckIn(String checkIn) { this.checkIn = checkIn; }

    public String getCheckIn() { return checkIn; }

    public void setCheckOut(String checkOut) { this.checkOut = checkOut; }

    public String getCheckOut() { return checkOut; }

    public void setRoomID(String roomNumb) { this.roomID = roomNumb; }

    public String getRoomID() { return roomID; }

}
```

Рисунок 2.26 – Лістінг BookRoom.java

У разі вдалого бронювання номеру, у базу даних додаються дані про букінг (рис. 2.27), також у особовому кабінеті клієнта веб-додатку з’являється наступний запис (рис. 2.28), до якого входить номер заказу, дата заселення і виселення з номеру готелю, а також і номер, який забраньовано[21].

Рисунок 2.27 – Запис про заброньований номер в таблиці

У базі даних характеристики номерів мають такий вигляд (рис. 2.28).

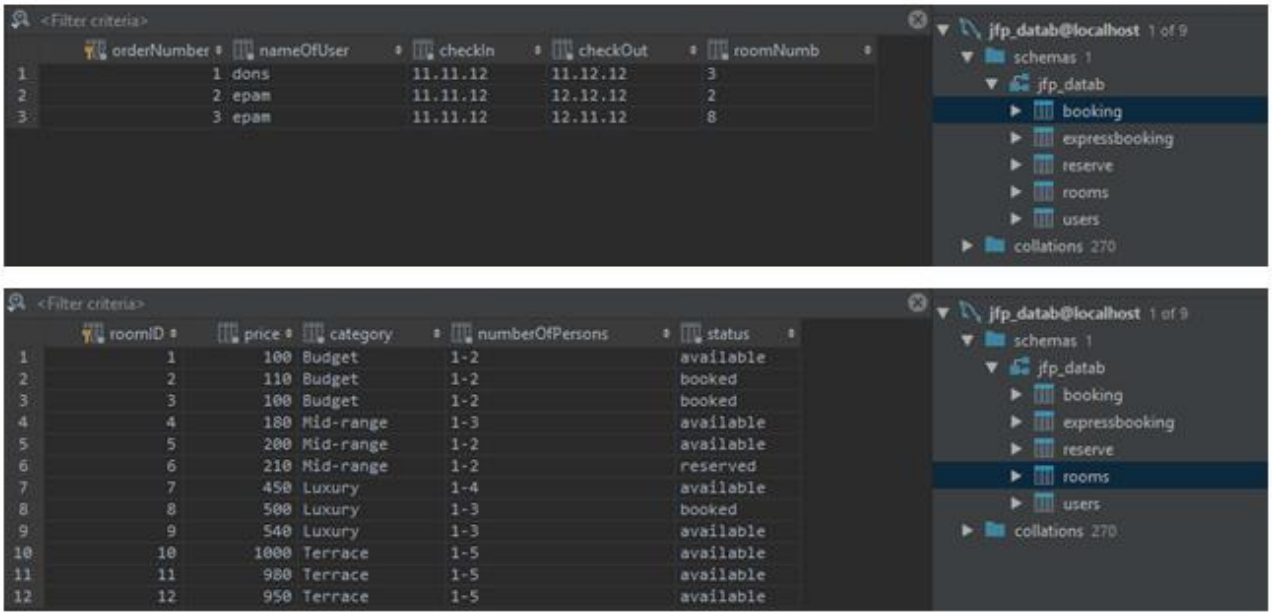


Рисунок 2.28 – Запис про заброньований номер в базі даних

2.4 Експрес заявка

З головної сторінки користувач, що пройшов аутентифікацію може зайти у розділ «експрес заявка». У цьому розділі він має можливість заповнити форму для номеру за власними побажаннями (рис. 2.29). Цю заявку буде передано на розгляд менеджера [23].

Рисунок 2.29 – Форма для експрес заявки

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>

<header id="header">
    <meta charset="utf-8">
    <link href="css/loginform.css" media="screen" rel="stylesheet">
    <link href='http://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic' rel='stylesheet' type='text/css'>
    <title>Express Booking</title>
    <a href="Home.jsp" title="Главная"><img class="logo" src="images/1245677.jpg"/></a>
</header>

<body>
<section>
    <div class="container mlogin login-page">
        <div id="login" class="form">
            <form action="ExpressBookingServlet" id="loginform" method="post" name="form" class="login-form">
                <input class="input" id="fullname" name="fullname" size="40" type="text" value=""  placeholder="fullname">
                <input class="input" id="email" name="email" size="40" type="email" value="" placeholder="email">
                <input class="input" id="numberOfGuests" name="numberOfGuests" size="40" type="number" min="1"
                       max="5" value="" placeholder="number of guests">
                <input class="input" id="typeOfRoom" name="typeOfRoom" size="40" type="text" placeholder="category">
                <input class="input" id="timeOfVisit" name="timeOfVisit" size="40" type="text" value="" placeholder="duration period">

                <button class="submit button" name="Book" type="submit" value="Book">reserve</button>
            </form>
        </div>

    </div>
</section>
</body>
</html>
```

Рисунок 2.30 – Лістинг ExpressBooking.jsp

Потім дані перенаправляються до обробки у ExpressBookingServlet.java (рис. 2.31).

```java
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class ExpressBookingServlet extends HttpServlet {
    public ExpressBookingServlet() {
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //set local variables
        String fullName = request.getParameter( s: "fullname");
        String email = request.getParameter( s: "email");
        String numberOfGuests = request.getParameter( s: "numberOfGuests");
        String typeOfRoom = request.getParameter( s: "typeOfRoom");
        String timeOfVisit = request.getParameter( s: "timeOfVisit");

        Express express = new Express();
        express.setFullName(fullName);
        express.setEmail(email);
        express.setNumberOfGuests(numberOfGuests);
        express.setTypeOfRoom(typeOfRoom);
        express.setTimeOfVisit(timeOfVisit);

        ExpressBookingDao expressBookingDao = new ExpressBookingDao();

        String bookRegistered = expressBookingDao.registerExpress(express);

        if (bookRegistered.equals("SUCCESS"))   //On success, redirect to the home.jsp
        {
            request.getRequestDispatcher( s: "/Home.jsp").forward(request, response);
        } else if (bookRegistered.equals("NONONO")) //On failure, display a message to the user
        {
            request.setAttribute( s: "errMessage", bookRegistered);
            request.getRequestDispatcher( s: "/ExpressBooking.jsp").forward(request, response);
        }
```

Рисунок 2.31 – Лістинг ExpressBookingServlet.java

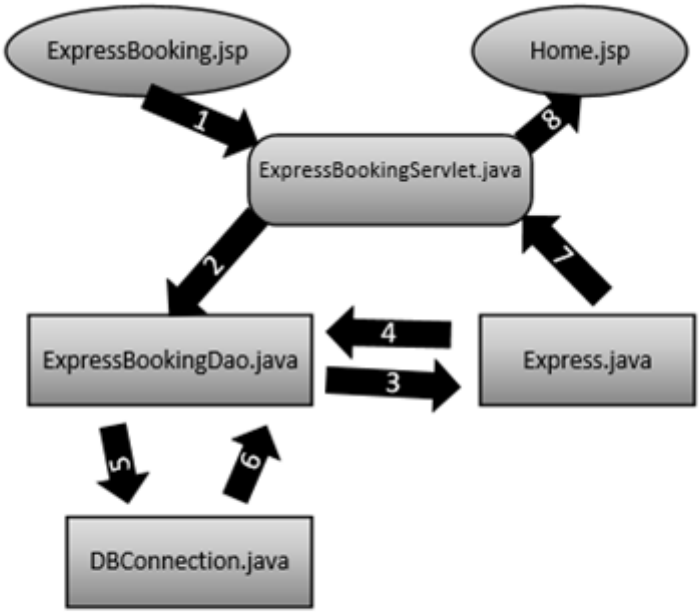


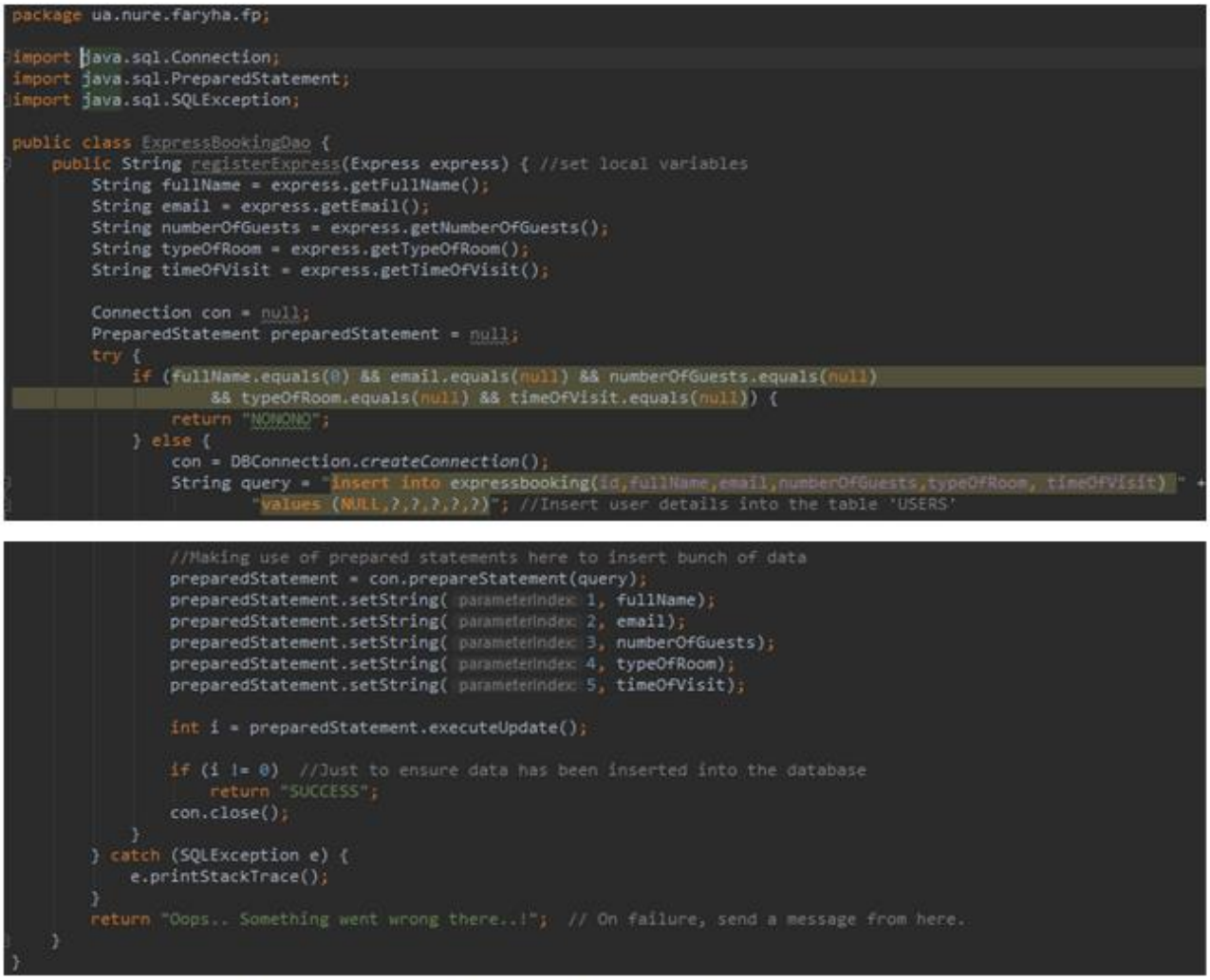
Рисунок 2.32 – Архітектура MVC для експрес заявок

Експрес заявки пов’язані з моделлю MVC (рис. 2.32). Тому, дані, що отримано записують у локальні змінні. Робиться новий об’єкт класу Express, в якому через сеттери записуються значення імені, email, кількість клієнтів, тип номеру та часу перебування клієнта (рис. 2.33) [14].

```java
package ua.nure.faryha.fp;

public class Express {

    private String fullName;
    private String email;
    private String numberOfGuests;
    private String typeOfRoom;
    private String timeOfVisit;

    public String getFullName() { return fullName; }

    public void setFullName(String fullName) { this.fullName = fullName; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public String getNumberOfGuests() { return numberOfGuests; }

    public void setNumberOfGuests(String numberOfGuests) { this.numberOfGuests = numberOfGuests; }

    public String getTypeOfRoom() { return typeOfRoom; }

    public void setTypeOfRoom(String typeOfRoom) { this.typeOfRoom = typeOfRoom; }

    public String getTimeOfVisit() { return timeOfVisit; }

    public void setTimeOfVisit(String timeOfVisit) { this.timeOfVisit = timeOfVisit; }
}
```

Рисунок 2.33 – Лістинг Express.java

Создається екземпляр класа ExpressBookingDao, який може викликати registerExpress (рис. 2.34). Ця функція за допомогою геттерів записує значення поля (повне ім’я, email, кількість клієнтів, клас номеру і час перебування) об’єкту Express, і фіксує їх у локальні змінні. Створюється підключення у базу даних, робиться перевірка локальних змінних, чи значення ненульові і відбувається запис даних у поля бази даних expressboking [14].



```java
package ua.nure.faryha.fp;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class ExpressBookingDao {
    public String registerExpress(Express express) { //set local variables
        String fullName = express.getFullName();
        String email = express.getEmail();
        String numberOfGuests = express.getNumberOfGuests();
        String typeOfRoom = express.getTypeOfRoom();
        String timeOfVisit = express.getTimeOfVisit();

        Connection con = null;
        PreparedStatement preparedStatement = null;
        try {
            if (fullName.equals(0) && email.equals(null) && numberOfGuests.equals(null)
                    && typeOfRoom.equals(null) && timeOfVisit.equals(null)) {
                return "NONONO";
            } else {
                con = DBConnection.createConnection();
                String query = "insert into expressbooking(id,fullName,email,numberOfGuests,typeOfRoom, timeOfVisit)" +
                        "values (NULL,?,?,?,?,?)"; //Insert user details into the table 'USERS'
                //Making use of prepared statements here to insert bunch of data
                preparedStatement = con.prepareStatement(query);
                preparedStatement.setString( parameterIndex: 1, fullName);
                preparedStatement.setString( parameterIndex: 2, email);
                preparedStatement.setString( parameterIndex: 3, numberOfGuests);
                preparedStatement.setString( parameterIndex: 4, typeOfRoom);
                preparedStatement.setString( parameterIndex: 5, timeOfVisit);

                int i = preparedStatement.executeUpdate();

                if (i != 0)  //Just to ensure data has been inserted into the database
                    return "SUCCESS";
                con.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return "Oops.. Something went wrong there..!";  // On failure, send a message from here.
    }
}
```

Рисунок 2.34 – Лістинг ExpressBookingDao.java

Якщо додавання даних у базу даних було вдалим, повертається до классу (який його викликав) ExpressBookingServlet.java повідомлення «SUCCESS». Далі, робиться редірект (перенаправлення) клієнта до головної сторінки [14].

2.5 Сторінка менеджеру

У процессі аутентифікації даного веб-додатку є розподіл на дві основні ролі. Це – користувач та менеджер. Коли відбіюється регістрація, клієнт отримує статус user у відповідній базі даних, проте у тій же базі є і admin (це видно на рис. 2.14). Отже після аутентифікації, система має перевірити роль, та

видає сторінку менеджера (Admin.jsp) (рис. 2.35) чи клієнта (User.jsp) (рис. 2.16).

Функції менеджера веб-додатку мають обмеження тільки однією сторінкою – Admin.jsp (рис. 2.35).

У менеджера є можливість розглянути побажання клієнта та обрати зі списку номери, які доступні підходящий варіант. Коли номер обрано, менеджер вводе повідомлення для користувача, також номер кімнати, яку обрано та надає результат клієнтові. Потім, номер обраної кімнати записується, як reserved[24].

Рисунок 2.35 – Сторінка менеджера

```jsp
<%@ page import="ua.nure.faryha.fp.DBConnection" %>
<%@ page import="java.sql.*" %><%--
  Created by IntelliJ IDEA.
  User: Julia
  Date: 05.11.2018
  Time: 12:00
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<header>
    <title>Admin Page</title>
    <link href="../css/contact.css" rel="stylesheet" type="text/css">
    <h1>
        <a> <!--href="../Home.jsp" title="Главная"--><img class="logo" src="../images/1245677.jpg"/></a>
    </h1>
</header>


<% //In case, if admin session is not set, redirect to Login page
    if ((request.getSession(false).getAttribute("Admin") == null)) {
%>
<jsp:forward page="/JSP/Login.jsp"></jsp:forward>
<%} %>
```

```jsp
<script>
    document.createElement('aside');
    document.createElement('article');
</script>
<style>
    aside {
        background: #f0f0f0;
        margin-top: 15px;
        padding: 10px;
        width: 200px;
        float: right;
    }

    article {
        margin-right: 240px;
        display: block;
    }
</style>

<aside>
    <p>Available rooms</p>
    <table border="1">
        <tr>
            <td>ID</td>
            <td>Price</td>
            <td>Categ.</td>
            <td>Numb.</td>
        </tr>
```

```jsp
        <%
            String a = "available";
            String s = "SELECT * FROM rooms WHERE status='" + a + "';";
            try {
                Connection connection = DBConnection.createConnection();
                PreparedStatement pstmt = connection.prepareStatement(s); //"s" from certain case
                ResultSet rs = pstmt.executeQuery(s);
                while (rs.next()) {
                    //Copying all the parameters in to local variables
                    String roomid = rs.getString("roomID");
                    String roomprice = rs.getString("price");
                    String roomcategory = rs.getString("category");
                    String numOfPers = rs.getString("numberOfPersons");
                    String stat = rs.getString("status");

        %>
        <tr>
            <td><% out.print(roomid); %></td>
            <td>$<% out.print(roomprice); %></td>
            <td><% out.print(roomcategory); %></td>
            <td><% out.print(numOfPers);%></td>
            <% }
            } catch (SQLException e) {
                e.printStackTrace();
            }
            %>
```

```
        </tr>
    </table>

</aside>


<body>
<div class="login-page" style="...">
    <div class="form" style="...">
        <div class="vcard">
            <h3><b> Welcome <%=request.getAttribute("userName") %>!</b></h3>
            </br>
            <p>Booking list of all users:</p>

            <table border="1" style="...">
                <tr style="...">
                    <td style="...">order number</td>
                    <td style="...">user name</td>
                    <td style="...">check in</td>
                    <td style="...">check out</td>
                    <td style="...">room number</td>
                </tr>
```

```
                <%
                    //local variables
                    String ordr = "";
                    String nou = "";
                    String chIn = "";
                    String chOt = "";
                    String rn = "";

                    //create db connection
                    Connection con = null;
                    Statement statement = null;
                    ResultSet resultSet = null;
                    con = DBConnection.createConnection();

                    try {
                        statement = con.createStatement();
                        String str = "select * from booking;";
                        resultSet = statement.executeQuery(str);
                        while (resultSet.next()) {
                            try {
                                ordr = resultSet.getString("orderNumber");
                                nou = resultSet.getString("nameOfUser");
                                chIn = resultSet.getString("checkIn");
                                chOt = resultSet.getString("checkOut");
                                rn = resultSet.getString("roomNumb");
                            } catch (SQLException e) {
                                e.printStackTrace();
                            }
                %>
```

```
                <!-- while we getting data from db, insert it in table -->
                <tr style="...">
                    <td><p style="..."><%=ordr %>
                    </p></td>
                    <td><p style="..."><%=nou %>
                    </p></td>
                    <td><p style="..."><%=chIn %>
                    </p></td>
                    <td><p style="..."><%=chOt %>
                    </p></td>
                    <td><p style="..."><%=rn %>
                    </p></td>
                </tr>

                <% }
                } catch (SQLException e) {
                    e.printStackTrace();
                }%>
            </table>
            <!-- end of the first table -->
            </br>
```

```
            <!--second table with the same principle: -->
            <p>ExpressBooking list of all users:</p>
```

```
<table border="1"
       style="...">
    <tr style="...">
        <td style="...">id</td>
        <td style="...">full name</td>
        <td style="...">email</td>
        <td style="...">number of guests</td>
        <td style="...">type of room</td>
        <td style="...">time of visit</td>
        <td style="...">approve/<br>
            disapprove
        </td>
        <td style="...">room num</td>
        <td></td>
    </tr>
```

```
<%
    int id = 0;
    String fn = "";
    String em = "";
    String nog = "";
    String tor = "";
    String tov = "";

    try {
        statement = con.createStatement();
        String str = "select * from expressbooking;";
        resultSet = statement.executeQuery(str);
        while (resultSet.next()) {
            try {
                id = resultSet.getInt("id");
                fn = resultSet.getString("fullName");
                em = resultSet.getString("email");
                nog = resultSet.getString("numberOfGuests");
                tor = resultSet.getString("typeOfRoom");
                tov = resultSet.getString("timeOfVisit");
            } catch (SQLException e) {
                e.printStackTrace();
            }
%>
```

```
<tr style="...">
    <td><p style="..."><%=id %>
    </p></td>
    <td><p style="..."><%=fn %>
    </p></td>
    <td><p style="..."><%=em %>
    </p></td>
    <td><p style="..."><%=nog %>
    </p></td>
    <td><p style="..."><%=tor %>
    </p></td>
    <td><p style="..."><%=tov %>
    </p></td>
    <form action="ReservServlet" method="post">
        <td><input id="message" name="message" type="text" placeholder="text"></td>
        <td><input id="roomnum" name="roomnum" type="text" placeholder="room"></td>
        <td>
            <button name="fn" type="submit" id="fn" value="<%=fn%>">send</button>
        </td>
    </form>
</tr>

<% }
} catch (SQLException e) {
    e.printStackTrace();
}%>
</table>


<button style="...">
    <a style="..." href="<%=request.getContextPath()%>/LogoutServlet">Logout</a>
</button>
</div>
</div>
</div>
</body>
</html>
```

Рисунок 2.36 – Лістинг Admin.jsp

ВИСНОВКИ ДО РОЗДІЛУ 2

У даному розділі розглянуті і проаналізовані головні етапи для проведення створення веб-додатка із використанням HTML, CSS, Java, MySQL, середою розробки IntelliJ IDEA та сервером Tomcat.

Тематика веб-додатку цієї роботи, це готельний букінг. Клієнт-користувач може зареєструватися у системі та залишити заявку, в якій фіксує кількість місць у номері, дату перебування та якого типу апартаменті необхідні. Також має можливість обрати номер із списка наявних номерів та зробити бронювання номеру.

ВИСНОВКИ

У даній роботи розглянуті технології і засоби, які використовують при веб-програмуванні (HTML, CSS, Java). Розроблено структуру, дизайн веб-додатку.

Веб-додаток, який розроблено призначений для готельної сфери діяльності. Щоб реалізувати його функціонал і зв’язок із базою даних, використані мови программування Java і SQL. Засобами Java зроблене з’єднання та налаштування веб-сторінок, а також зроблені форми регістрації і аутентифікації. Для працювання обрано СУРБД MySQL, так як відрізняється достатньою надійністю, швидкістю роботи і гнучкістю. Програма MySQL Workbench допомогла створити таблиці у базі даних, у якій є інформація про номери, і таблиці для систем букінгу, оформленні різного типу експрес-заявок, резервування номерів та регістрації клієнтів. Для налаштування проекту використано Maven, який вбудовано у інтегровану систему розробки програмного забезпечення IntelliJIDEA. Веб-сервером (контейнер сервлетів) обрано Apache Tomcat, як досить стабільний та гарний сервер, який має останні версії усіх веб-технологій (Servlets, JSP, EL, Web Sockets), також має різноманітні плагіни.

У ході роботи проаналізовано веб-технології, розглянуто етапи розробки веб-додатку та проведено работу з програмним забезпеченням. Веб-додаток, який створено було протестовано локально.

ПЕРЕЛІК ПОСИЛАНЬ

1. Акимов С. В. Технологии Interner. Intranet в почтовой связи [Текст] : учеб. пособие / С. В. Акимов. – СПб. : СПбГУТ, 2005. – 350с.

2. Блинов И.Н. Java. Промышленное программирование: практ. пособие [Текст] / И.Н. Блинов, В.С. Романчик. – Минск : УниверсалПресс, 2007. – 704с.

3. Блинов, И.Н. Java 2: практ. рук. [Текст] / И.Н. Блинов, В.С. Романчик. – Мн.: УниверсалПресс, 2005. – 400 с.

4. Введение в Java EE [Электронный ресурс] Режим доступа: https://metanit.com/java/javaee/1.1.php

5. Гончаров А. Г. Самоучитель HTML [Текст] / А. Г. Гончаров. – СПб. : Питер, 2002. – 240с.

6. Ерёмин В. Н. Маркетинг : основы и маркетинг информации [Текст] : учеб. пособие / В. Н. Ерёмин. – М. : КНОРУС, 2006. – 656с.

7. Козлова И. С. Информатика: конспект лекций [Текст] : учеб. пособие для вузов / И. С. Козлова. – М. : Высшее издание, 2005. – 128с.

8. Любимова Е. М. Разработка Web-приложений средставами языка PHP [Текст] : учеб. пособие для вузов / Е. М. Любимова. – Елабуга : Филиал К(П)ФУ, 2014. – 148с.

9. Макфарланд Д. Большая книга CSS3 [Текст] / Д. Макфарланд, 3-е изд., знач. доп. – СПб. : Питер, 2014. – 608с.

10. Мержевич В. В. HTML и CSS на примерах [Текст] / В. В. Мержевич. – СПб. : БХВ-Петербург, 2005. – 448с.

11. Несвижинский А. И. Современные веб-технологии [Текст] / А. И. Несвижинский, В. А. Рябов. – М. Национальный Открытый Университет «ИНТУИТ», 2016. – 1002с.

12. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 [Текст] / Р. Никсон, – 3-е изд., знач. доп. – СПб. : Питер, 2015. – 688с.

13. Петюшкин А. В. HTML. Экспресс-курс [Текст] / А. В. Петюшкин. – СПб. : БХВ-Петербург, 2003. – 256с.

14. Пинягина О. В. Разработка электронного магазина на PHP и MySQL [Текст] : учеб. пособие / О. В. Пинягина. – Казань: Казанский государственный университет, 2010. – 108с.

15. Разработка Java приложений [Електроний ресурс] Режим доступу: http://it-simulator.com/#/article/1/102

16. Томсон Л. Разработка Web-приложений на PHP и MySQL [Текст] / Л. Томсон, Л. Веллинг. – 2-е изд., испр. : перевод с англ. – СПб. : ООО «ДиаСофтЮП», 2003. – 672с.

17. Ульман Л. PHP и MySQL: создание интернет-магазинов [Текст] / Л. Ульман, – 2-е изд. : перевод с англ. – М. : ООО «И. Д. Вильямс», 2015. – 544с.

18. Шилдт Г. Java 8. Полное руководство; 9-е изд.: Пер. с англ. [Текст] / Г. Шилдт. – М. : ООО "И.Д. Вильяме", 2015. - 1376 с.

19. Ecommerce Foundation [Электронный ресурс] Режим доступа: www.ecommercefoundation.org/reports – 29.05.2017г. – Загл. с экрана.

20. HTML & CSS [Электронный ресурс] Режим доступа: www.w3.org/standarts/webdesign/htmlcss – 29.05.2017г. – Загл. с экрана.

21. Java WEB [Электронный ресурс] Режим доступа: http://java-online.ru/java-web.xhtml

22. Thinking in Java Enterprise [Электронный ресурс] Режим доступа: http://javatutor.net/books/tiej/servlets

23. Wiki-учебник по веб-технологиям [Электронный ресурс] Режим доступа: www.webmasterwiki.ru – 29.05.2017г. – Загл. с экрана.

24. Скорик Ю.В., Остапенко О.В. Створення інтернет сайту з продажу мережного обладнання / Одинадцята міжнародна науково-технічна конференція «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління» 8-9 квітня 2021 року. Баку – Харків – Київ – Жиліна – 2021. – С. 105.