ЗВІТ З ПЕРЕВІРКИ НА ПЛАГІАТ

ЦЕЙ ЗВІТ ЗАСВІДЧУЄ, ЩО ПРИКРПЛЕНА РОБОТА
ФЕДОТОВ Д.С. ІПЗ-129
БУЛА ПЕРЕВІРЕНА СЕРВІСОМ ДЛЯ ЗАПОБІГАННЯ ПЛАГІАТУ MY.PLAG.COM.UA І МАЄ:

СХОЖІСТЬ

8%

РИЗИК ПЛАГІАТУ

100%

ПЕРЕФРАЗУВАННЯ

1%

НЕПРАВИЛЬНІ ЦИТУВАННЯ

0%

Назва файлу: ФЕДОТОВ Д.С. ІПЗ-129 (-).docx
Файл перевірено: 2023-06-20
Звіт створено: 2023-06-20

МІНІСТЕРСТВО ОСВІТИ НАУКИ УКРАЇНИ

ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ» (www.zieit.edu.ua)

Кафедра інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Завідувач кафедри,
д.е.н., доц.
(library.econom.zp.ua)
_____ С.І. Левицький

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

РОЗРОБКА WEB-САЙТУ ДЛЯ ГОТЕЛЬНОГО БІЗНЕСУ НА БАЗІ JAVA-ТЕХНОЛОГІЙ

Виконав

ст. гр. ІПЗ (library.econom.zp.ua) – 129
_____ Д. С. Федотов

Керівник

к.е.н., доц. _____ О. В. Шляга

Запоріжжя

2023 р.

ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ» (www.zieit.edu.ua)

Кафедра інформаційних технологій

ЗАТВЕРДЖУЮ
Зав. (library.econom.zp.ua) кафедри
д.е.н., доцент
Левицький (library.econom.zp.ua) С.І.
_____
__ __ _____ р.

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ

Студенту гр. ІПЗ – (library.econom.zp.ua) 129, спеціальності 121«Інженерія програмного забезпечення» Федотову Дмитру Сергійовичу

1. Тема: Розробка web-сайту для готельного бізнесу на базі java-технологій затверджена наказом по інституту _____

2. Термін здачі студентом закінченої роботи:_____

3. Перелік питань, що підлягають розробці

1. Розглянути питання актуальності розробки застосунку.

2. Провести огляд галузі та аналітику проблеми і її рішень загалом.

3. Провести огляд та аналіз популярних аналогів, зробити висновки про вимоги до проекту.

4. Розглянути методи та способи створення веб-застосунків та обрати направлення розробки.

5. Провести огляд стеку технологій для розробки проекту та зробити

вибір на підставі вимог.

6. Розробити проектування архітектури веб-застосунку.

7. Створити відповідний застосунок, спираючись на отримані дані.

8. Проаналізувати отримані результати.

9. Оформити звіт за результатами роботи.

4. Календарний графік підготовки бакалаврської дипломної роботи

№ етапу	Зміст	Терміни виконання	Готовність по графіку %, підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Формулювання (корегування) теми кваліфікаційної бакалаврської роботи та збір практичного матеріалу за темою	16.01.23-11.02.23		
2	**І атестація** І розділ кваліфікаційної бакалаврської роботи	27.03.23-31.03.23		
3	**ІІ атестація** ІІ розділ кваліфікаційної бакалаврської роботи	24.04.23-28.04.23		
4	**ІІІ атестація** ІІІ розділ кваліфікаційної бакалаврської роботи, висновки та рекомендації, додатки, реферат	22.05.23-26.05.23		
5	Перевірка кваліфікаційної бакалаврської роботи на оригінальність	15.05.23-12.06.23		
6	Доопрацювання кваліфікаційної бакалаврської роботи, підготовка презентації, отримання (library.econom.zp.ua) відгуку керівника і рецензії	29.05.23-12.06.23		
7	**Попередній захист кваліфікаційної бакалаврської роботи**	**12.06.23-18.06.23**		
8	Подача кваліфікаційної бакалаврської роботи на кафедру	за 3 дні до захисту		
9	**Захист кваліфікаційної бакалаврської роботи**	**19.06.23-24.06.23**		

Дата видачі завдання: ___.___._____ р.

Керівник кваліфікаційної
бакалаврської роботи _____ О. В. Шляга

Завдання отримав до виконання _____ Д. С. Федотов

РЕФЕРАТ

Бакалаврська робота містить 66 сторінок, 62 рисунки, 1 таблицю, 11 використаних джерел.

Метою роботи є розробка веб-сайту на базі Java-технологій.

Об’єктом дослідження є веб-застосунки для готельного бізнесу.

Предметом дослідження є архітектурне рішення для функціонального та зручного веб-додатку.

Проект реалізовано за допомогою таких засобів, як (library.econom.zp.ua) Java, Spring Boot, Spring Security, Hibernate, Bootstrap. Здійснене проектування моделі предметної області, програмування сутностей та алгоритмів на базі (library.econom.zp.ua) MVC-додатку.

Веб-додаток на основі вищезазначених технологій є зручним та багатофункціональним, відповідає усім потребам з точки зору дизайну.

SPRING BOOT, SPRING SECURITY, HIBERNATE, BOOTSTRAP, SQL (ela.kpi.ua)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	Повна назва	Пояснення/переклад
JSP	Java Secured Page	Технологія, що дозволяє веб-розробникам динамічно генерувати HTML, XML та інші (er.knutd.edu.ua) веб-сторінки
SQL	Structured Query Language	Декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД
NoSQL	Not Only Structured Query Language	Широкий спектр різноманітних систем управління базами даних, (elartu.tntu.edu.ua) які пропонують альтернативний підхід до роботи з даними, але також можуть включати в себе елементи SQL
HTML	Hyper Text Markup Language	Стандартизована мова розмітки документів для перегляду веб-сторінок у браузері (uk.wikipedia.org)
CSS	Cascading Style Sheets	Формальна мова декорування та опису зовнішнього вигляду документа (веб-сторінки), написаного (ela.kpi.ua) з використанням мови розмітки (найчастіше HTML або XHTML)
UX	User Experience	Сприйняття та відповідні дії користувача, що виникають в результаті використання продукції, системи (ela.kpi.ua) чи послуги
UI	User Interface	Інтерфейс, що забезпечує передачу інформації між користувачем та програмно-апаратними компонентами комп'ютерної системи (ela.kpi.ua)
ID	Identifier	Унікальна ознака об'єкта, що дозволяє відрізняти його від інших об'єктів, тобто ідентифікувати (openarchive.nure.ua)
MVC	Model View Controller	Схема поділу даних програми та керуючої логіки на три окремі компоненти: модель, уявлення та контролер – таким чином, що модифікація кожного компонента може здійснюватися незалежно (ir.nmu.org.ua)

ВСТУП

Створення веб-сайту для малого, середнього або навіть великого бізнесу є надзвичайно актуальною задачею, оскільки завдяки подібним інтернет-ресурсам замовник має можливість донести до широкої аудиторії та своїх бізнес-партнерів спектр послуг, який він надає. Готельний бізнес з цієї точки зору є надзвичайно чутливим до створення власного веб-сайту, на якому можна продемонструвати майбутнім гостям номери, прилеглу територію, можливі тури та усі інші переваги того чи іншого готелю, ознайомити їх з ціновою політикою, надати можливість здійснити оформлення броні на певні дати.

Виходячи з обраної тематики, а саме «Розробка web-сайту для готельного бізнесу на базі Java-технологій», в ході виконання бакалаврської роботи потрібно здійснити наступні кроки:

– ознайомитися з нюансами готельного бізнесу, визначити вектори для створення зручного, привабливого та функціонального web-сайту для залучення користувачів та ефективної взаємодії між бізнесом та клієнтами;

– окреслити необхідний спектр інструментів розробки на мові Java, які будуть використані на проекті, пояснити свій вибір та визначити переваги таких рішень;

– продумати логіку та архітектуру проекту, здійснити його програмну реалізацію, продемонструвати його роботу.

В цій роботі проаналізовані веб-сайти популярних готелів за наступними параметрами: дизайн сайту, його функціонал (а особливо формат надання інформації щодо варіантів номерів та способів їх бронювання), наочність та комфорт при користуванні сайтом (так званий *User Experience*), а також підхід до презентації бізнесу. Виходячи з отриманої інформації буде сформований перелік фундаментальних вимог для створення сайту, функціонуючого належним чином, та рекомендації щодо його візуальної складової.

З програмного боку проведений аналіз таких нюансів та сутностей, як: мова Java в цілому, та аргументи щодо її вибору, організація роботи з фреймворком Spring та його порівняння зі збіркою на основі «сервлетів» та jsp-сторінок, вибір Системи управління базами даних та (ism.lp.edu.ua) типу бази даних в цілому, порівняння реляційних (*Structured Query Language*, надалі – SQL) та нереляційних (NoSQL) форматів зберігання даних, а також (ism.lp.edu.ua) інструменти для візуального супроводу продукту (HTML, CSS, JavaScript, Bootstrap).

По результатам проведеного аналізу поставлені задачі щодо функціонального наповнення та вигляду продукту, на основі чого буде розроблений працездатний та високофункціональний проект.

Метою роботи є розробка веб-сайту на базі Java-технологій.

Об’єктом дослідження є веб-застосунки для готельного бізнесу.

Предметом дослідження є архітектурне рішення для функціонального та зручного веб-додатку.

Проект реалізовано за допомогою таких засобів, як Java, Spring Boot, Spring Security, Hibernate, Bootstrap. Здійснене проектування моделі предметної області, програмування сутностей та алгоритмів на базі (library.econom.zp.ua) MVC-додатку.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Оскільки предметною областю виступає готельний бізнес та написання веб-сайту до нього, потрібно провести порівняльний аналіз існуючих веб-сайтів популярних готелів і визначити потреби з боку користувачів та з боку бізнесу. В ході аналізу потрібно приділити особливу увагу дизайну сайту, розташуванню основних кнопок, формату реєстрації користувача та залишення заявки на бронювання номеру, способам подачі інформації та демонстрації цінової політики.

Для користувачів важлива приваблива візуальна складова, яка визиватиме довіру та демонструватиме усі переваги того чи іншого готелю за допомогою фотографій, детального опису локацій, умов заїзду та проживання, додаткових можливостей (актуальні екскурсії, трансфер від аеропорту, система знижок у найближчій інфраструктурі, доступ до закритих заходів тощо). Крім того, майбутнім клієнтам необхідний зручний, інтуїтивно зрозумілий сайт, в якому можна буде легко подивитися доступні номери та дати бронювання, а також здійснити безпечну передоплату.

Для замовників веб-сайту важливо, щоб сайт «продавав» їх продукт, привертаючи увагу нових клієнтів та забезпечуючи швидкий та прозорий спосіб оплати різноманітних послуг, а також надавав відповіді на всі необхідні запитання, в тому числі за допомогою спеціальних співробітників зі служби підтримки.

У якості прикладів будемо використовувати такі всесвітньо відомі готелі або мережі готелів, як: *Hyatt, Marriott, Four Seasons, Rixos, Burj Al Arab.*

Як зазначено вище, основними предметами аналізу будуть чотири наступних параметри: дизайн сайту (*User Interface*), комфорт користування (*User Experience*), його функціональне наповнення та презентація продукту.

1.1 Дизайн

Дизайн – людська або машинна діяльність, спрямована на проектування естетичних властивостей певного продукту, а також результат цієї діяльності. Включає в себе архітектурний дизайн, графічний, ландшафтний, промисловий, анімаційний, гейм-дизайн, дизайн одежі та багато інших.

Дизайн продукту привертає до нього увагу, та є відповідним за перше враження. Дуже часто від оболонки залежить, чи подобається продукт користувачу, і чи хоче він за нього платити. Дизайн може підказувати користувачу, до якої категорії – бюджетної або люксової – належить певний продукт. Це відбувається за рахунок певних кольорів, відповідних шрифтів, асоціацій тощо.

Одним з різновидів дизайну є веб-дизайн – галузь веб-розробки, в задачі якої входить проектування користувацьких веб-інтерфейсів для сайтів (www.victoria.lviv.ua) та веб-додатків.

У задачі веб-дизайнерів входить:

- проектування логічної структури веб-сторінок
- вибір найвдаліших рішень надання інформації
- художнє оформлення веб-проекту

Вдалий та сучасний дизайн сайту повинен бути зручним та гармонічним, без нагромадження елементів та відволікаючих кольорів, містити читабельні шрифти та продуману айдентику. Гарною ознакою, як і в багатьох інших речах, є мінімалізм: сайт, який поєднує в собі простоту та функціональність, завжди краще сприймається користувачем. Також вдалою рисою вважається унікальність – здатність творити щось власне та нове, а не копіювати рішення конкурентів з галузі або з інших «трендових» сайтів.

Розглянемо декілька прикладів, та відмітимо на них вищеозначені характеристики:

1. Ritz-Carlton

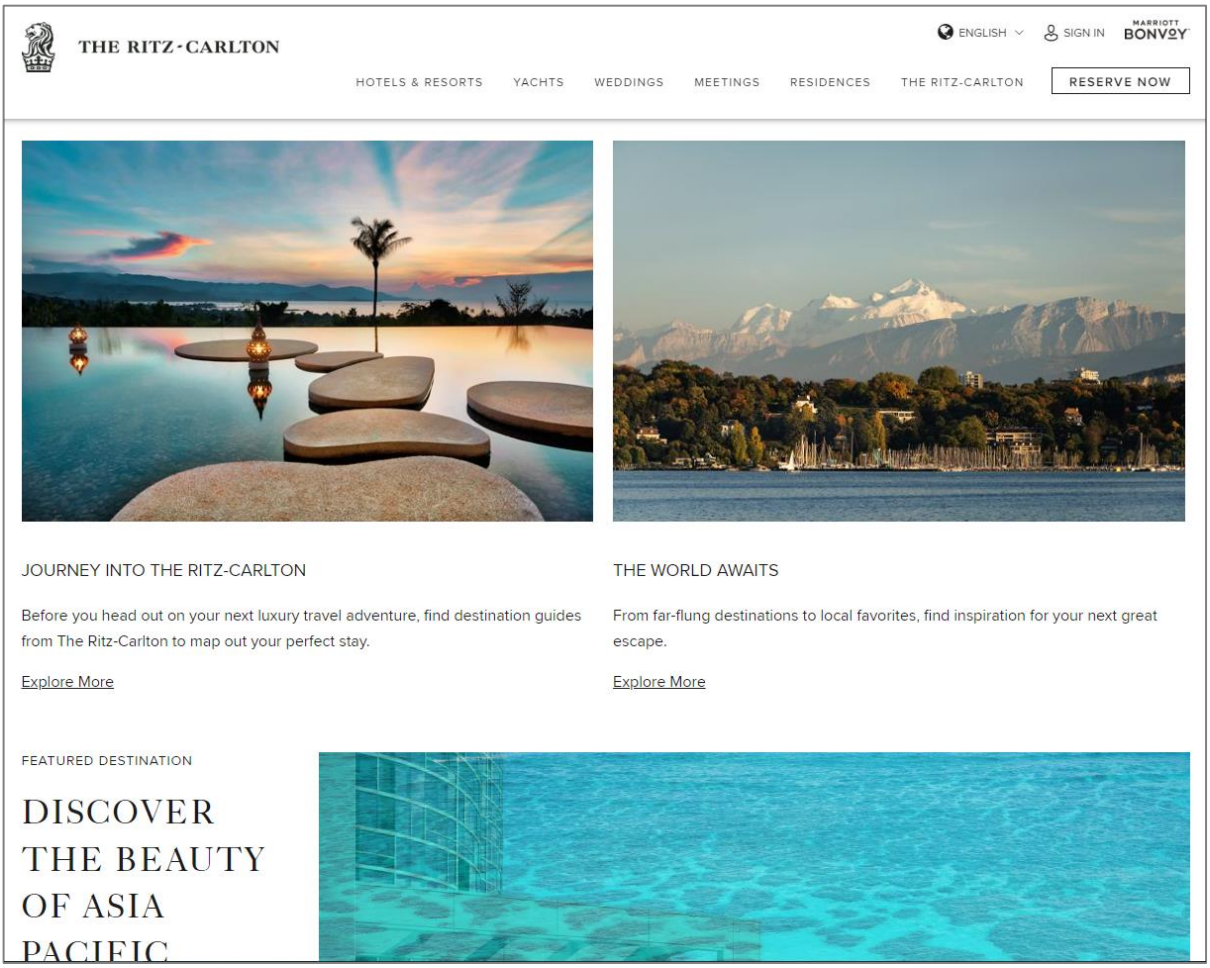


Рис. 1.1 – Сторінка з веб-сайту готелю Ritz-Carlton

Мережа готелів Ritz-Carlton привертає увагу майбутніх клієнтів яскравими фотографіями, зробленими на території готелів, привертаючими увагу заголовками та ємкими підписами. Для збалансування різнобарв'я картинок використовується мінімалістичний білий фон. Айдентика компанії виконана у строгому стилі – чорно-білий логотип з левом та класичний друкарський шрифт відсилають до глибокої історії компанії та її елітарності.

Перші сторінки сайту не обтяжують користувача розцінками, акційними пропозиціями, формами для надання імейлу і отримання промокоду – на першому місці стоять емоції та позитивне враження, відчуття комфорту та спокою, так важливі при відпочинку в ідеальному готелі.

2. Burj Al Arab

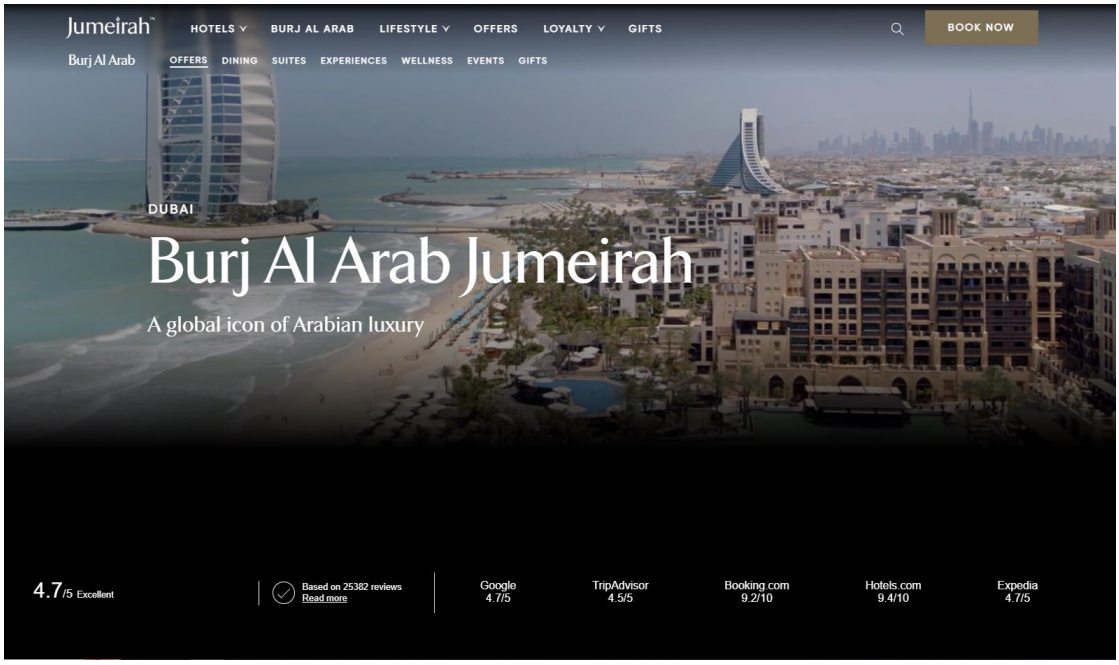


Рис. 1.2 – Вітальна сторінка з веб-сайту готелю Burj Al Arab

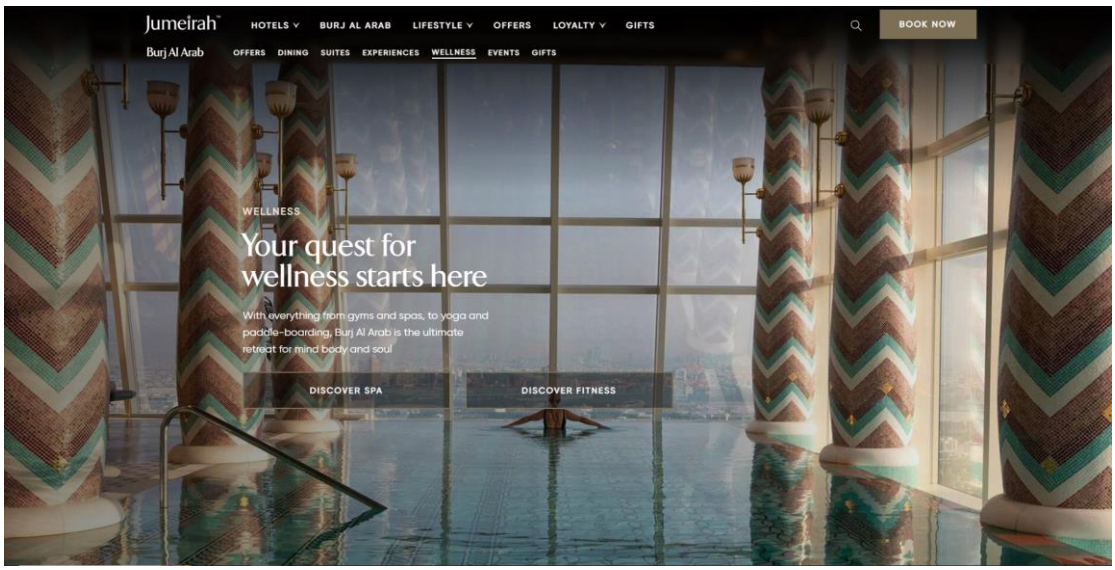


Рис. 1.3 – Сторінка з веб-сайту готелю Ritz-Carlton

Один з найдорожчих готелів світу, розташований в ОАЕ Burj Al Arab має схожий підхід, але фотографії та відео-фрагменти використовуються тут в якості фону, нагадуючи дуже популярний нині формат сайтів-візиток. Пов’язує ці сегменти чорний фон, майже завжди виконаний з ефектом «фейду», або вицвітання. Як і на минулому прикладі, сайт має невеликі обсяги

тексту, та зосереджує візитера на розкішних зображеннях інтер'єру. «Арабська вежа» також не забуває похизуватися високими оцінками різноманітних видань – вони розташовані на самому першому слайді.

3. Four Seasons

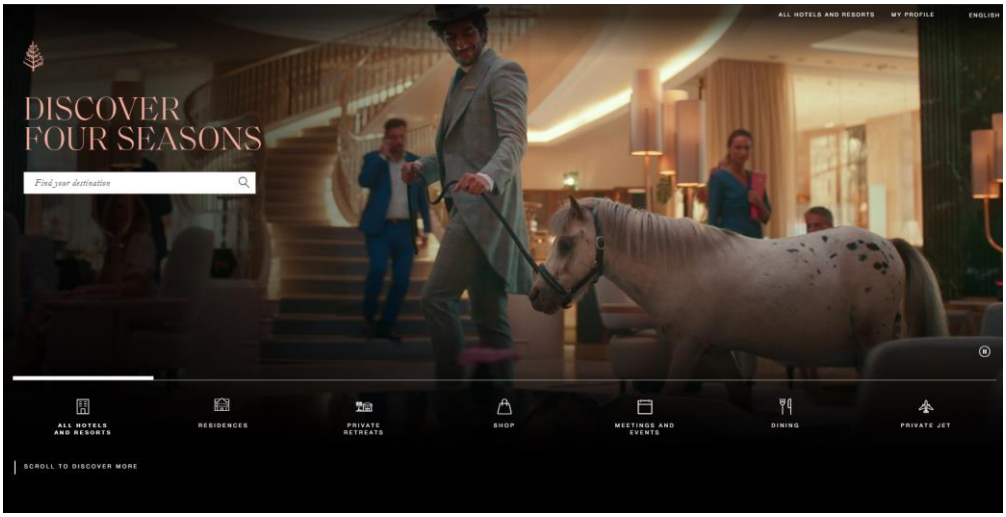


Рис. 1.4 – Вітальна сторінка з веб-сайту готелю Four Seasons

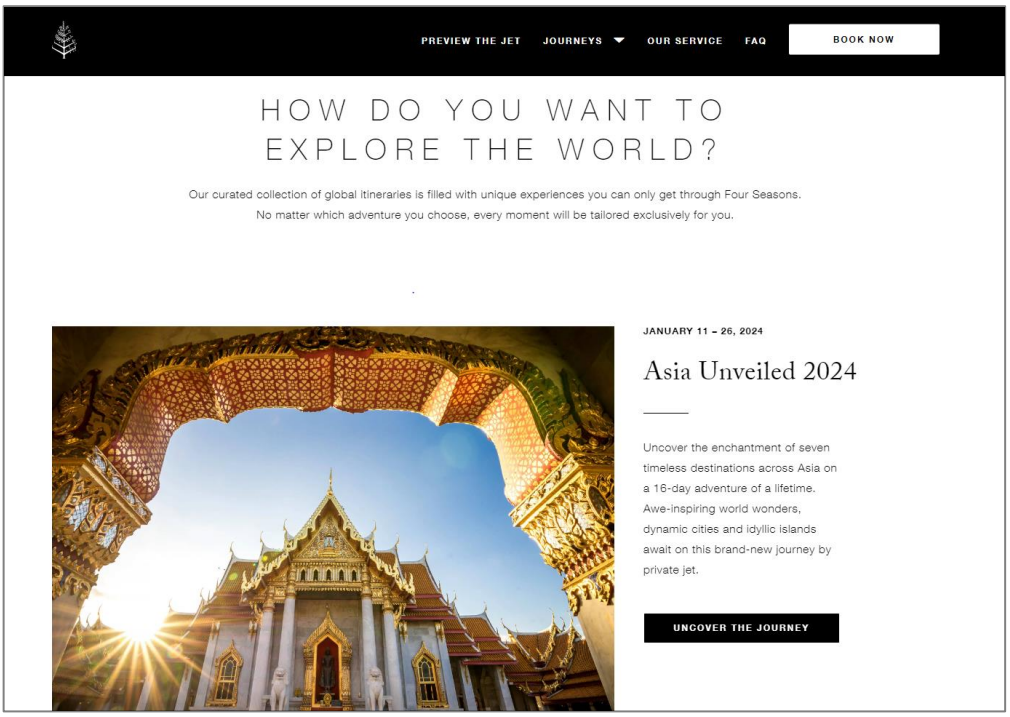


Рис. 1.5 – Сторінка з веб-сайту готелю Four Seasons

Ще одним власником вдалого сайту є всесвітня мережа готелів Four Seasons – тут використовуються ті ж самі правила, як і в попередніх випадках:

яскраві відео та зображення на весь екран, чорний та білий кольори, великі заголовки та короткі підписи. У цьому проекті дуже вдало підібрані шрифти та розроблені сучасні іконки для переходу на різні сторінки. Окрему увагу хочеться приділити логотипу, який відображає назву без будь-яких приписів та легко підлаштовується під різні стилі та кольору навколишнього наповнення сайту.

1.2 Комфорт користування

Вдалий веб-сайт повинен бути не лише приємним на вигляд, а й комфортним у користуванні. За це відповідає такий підрозділ дизайну, як User Experience (з англ. – досвід взаємодії).

Досвід користувача – сприйняття та відповідні дії, виникаючі в результаті користування або майбутнього користування продукцією, системою, послугою. Це поняття включає всі емоції, переконання, уподобання, відчуття, фізичні та психологічні реакції користувача, які виникають до, під час та після використання системи. UX-дизайн умовно поділяють на п’ять рівнів:

1. Рівень стратегії – (cad.kpi.ua) найвищий та найбільш абстрактний рівень. На цьому рівні потрібно отримати відповіді на питання, які стосуються бажань та очікувань стосовно майбутнього програмного продукту, як з боку потенційних користувачів, так і з боку замовника.
2. Рівень можливостей – створення переліку функціональних можливостей, які будуть доступні для користувачів.
3. Рівень структури – взаємне розташування сторінок веб-сайту, програмних форм, вікон тощо. Ефективна структура поліпшує навігацію та робить її інтуїтивно зрозумілою для користувачів.
4. Рівень компоновки – надає конкретну реалізацію абстрактної структури продукту. На цьому рівні вирішуються питання найбільш вдалого різноманітних візуальних елементів.

5. Рівень поверхні – Вигляд продукту з точки зору кінцевого користувача. (cad.kpi.ua)

Ознайомимося, як з цими задачами справляються деякі представники бізнесу:

1. Rixos Hotels

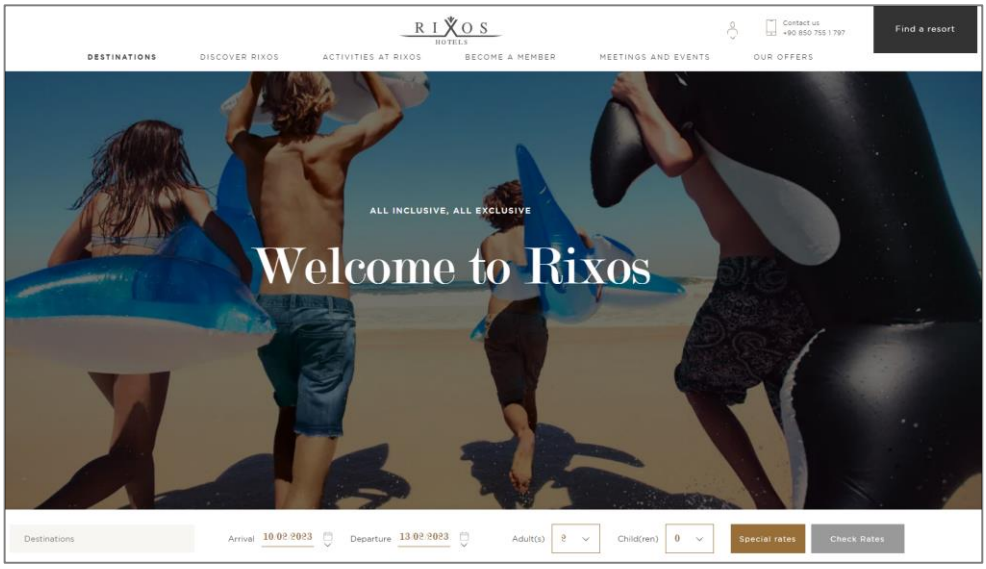


Рис. 1.6 – Вітальна сторінка з веб-сайту готелю Rixos

Перша сторінка сайту Rixos вміщує в себе класичне навігаційне меню зверху, посилання на реєстрацію або логін користувача та форму бронювання номеру в готелі. Розташування цих елементів у такій конфігурації є досить популярним, що на інтуїтивному рівні спрощує пошук потрібного розділу. Усі форми виконані в сучасному стилі, мають достатній розмір та читабельність, щоб бути помітними для користувача.

Навігаційне меню та сторінка реєстрації доступні з усіх сторінок, що надає можливість користувачу вільно переходити в будь-який край логічного устрою сайту.

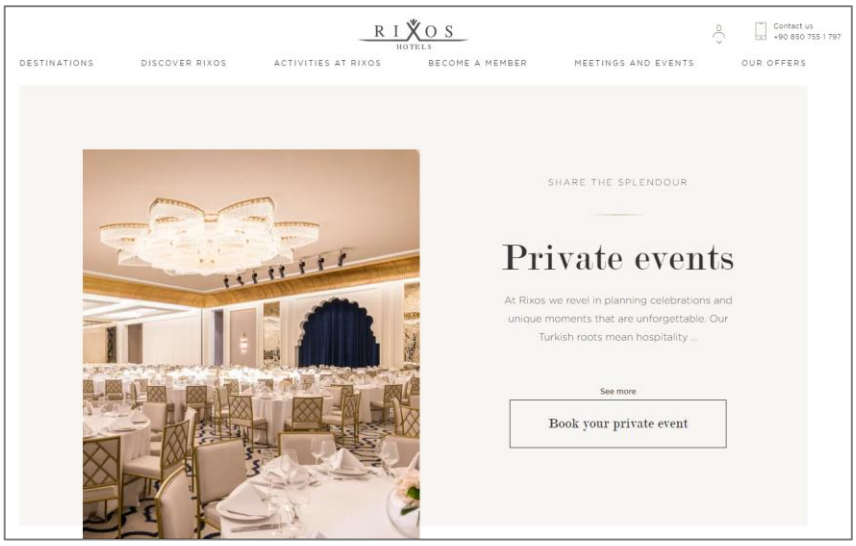


Рис. 1.7 – Сторінка з веб-сайту готелю Rixos

2. Marriott

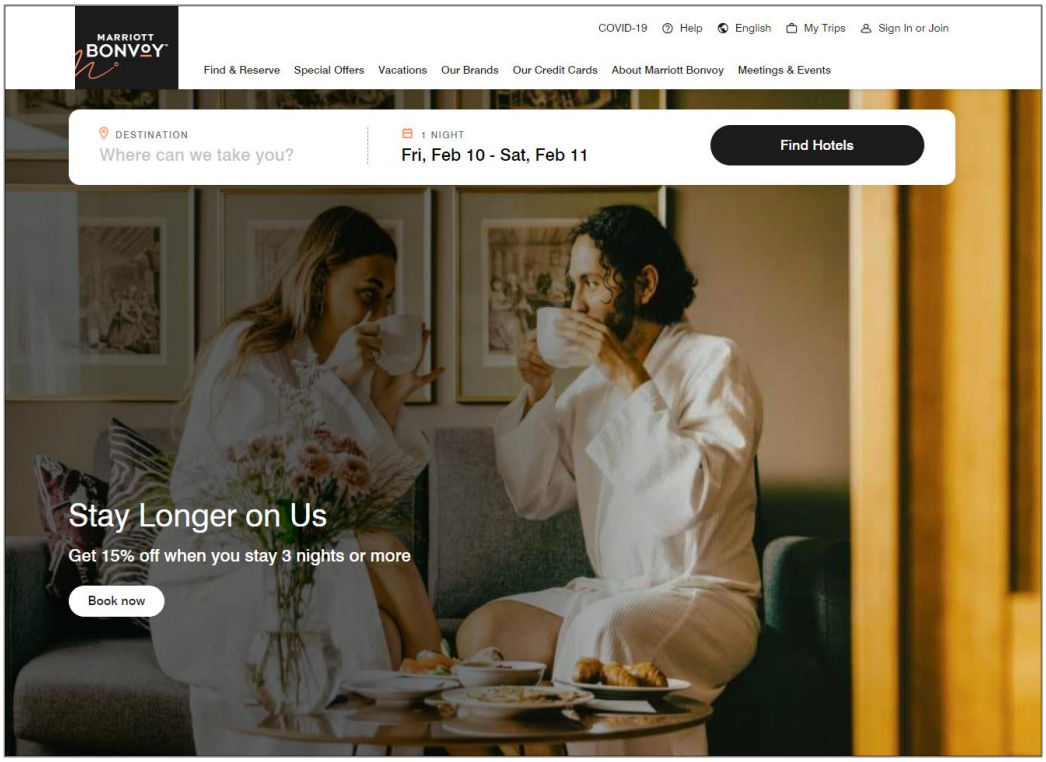


Рис. 1.8 – Вітальна сторінка з веб-сайту готелю Marriott

Готель Marriott має таку ж саму структуру: верхнє навігаційне меню, посилання на реєстрацію акаунту, можливість зміни мови та форма для бронювання на першій сторінці.

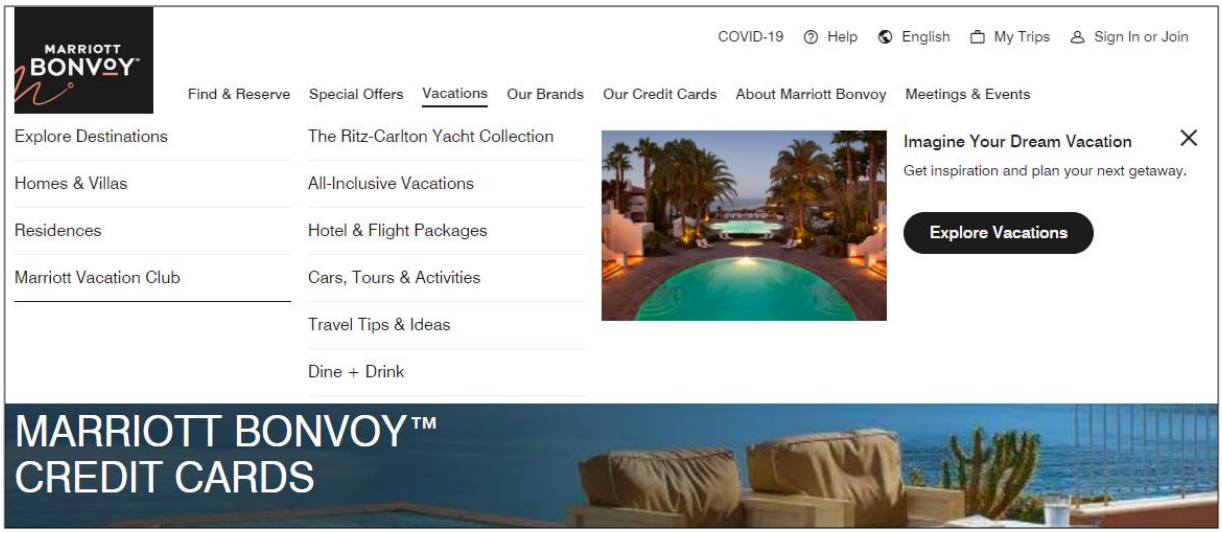


Рис. 1.9 – Сторінка з веб-сайту готелю Marriott

При натисканні на пункти з меню відбувається випадіння додаткової панелі, яка відсилає на окремі сторінки залежно від специфікації.

1.3 Функціональне наповнення

Враження від користування сервісом в багато в чому залежить від успішності взаємодії з формами. Вірною ознакою правильно функціонуючої форми є відсутність зайвих запитань, таких як:

– Як ввести ці дані?
– Навіщо вони потрібні?
– Чому форма не пропускає введену інформацію?

Для того, щоб форма виглядала та працювала належним чином, потрібно зосередити увагу на наступних нюансах:

1. Елементи інтерфейсу повинні відповідати типу очікуваної від користувача інформації. Принцип роботи елементів інтерфейсу має зчитуватися за його зовнішнім виглядом.

2. Перевірка полів повинна виконуватися по мірі заповнення форми. Повідомлення про помилки мають містити рекомендації щодо їх усунення.

Гарним тоном вважається, якщо система не видає помилок, доки відбувається заповнення окремого поля.

3. Кнопка відправки форми повинна бути неактивна, доки не заповнені усі необхідні поля. Система повинна вказувати, чому відправка форми неможлива.

4. Усі неочевидні поля потрібно супроводжувати відповідними поясненнями

5. Елементи форми розташовуються на одній прямій та групуються за сенсом задля наочності логічної структури форми, її загального об’єму та послідовності заповнення.

6. Форма повинна забезпечувати фокус уваги на поточному полі за допомогою підсвітки.

7. Форма має пам’ятати дані, які туди вводяться, задля економії часу в таких випадках, як перезавантаження або помилкове закриття сторінки.

8. Якщо форма містить секретні дані, такі, як пароль або інформація про банківську картку, вони мають бути скритими під символами.

9. Будь-який ввід завершується кнопкою, яка відправляє дані на сервер. Кнопка повинна бути помітною, та в більшості випадків має напис в вигляді дієслова (наприклад, «Зареєструватися»).

Зупинимося докладніше на формах реєстрації та логіну користувачів, а також формі бронювання номеру.

1. Реєстрація та логін

Реєстрація майже у всіх випадках містить такі поля, як ім’я та прізвище, імейл та номер телефону, а також пароль та додаткове поле для його підтвердження. Імейл використовується для підтвердження реєстрації, та в подальшому буде виступати в якості логіну для входу. Також інколи система потребує додаткову інформацію, таку, як країна проживання, поштовий код, вік користувача, номер картки члену клубу тощо.

Кожне поле повинно приймати дані в певному форматі, та містити перевірки щодо правильності вводу, оскільки ці дані потрапляють в базу

даних, в якому зберігаються в зазначеному типі даних, відносно якого функціонує програма. Особлива увага приділяється таким ключовим полям, як номер телефону та імейл.

Телефон повинен відповідати певному формату залежно від країни, та складатися лише з цифр. Такі символи, як плюс, дужки, дефіси між цифрами система повинна підставляти власноруч та наочно демонструвати це користувачу.

Імейл також повинен проходити хоча б первинну перевірку, маючи певну довжину, та спеціальні символи, як «@» та «.». Крім того, імейл має бути записаний на латиниці, та належати до існуючих доменів (наприклад, gmail.com).

Специфічний формат також має пароль: він має бути певної складності, що забезпечується його довжиною та певним різноманіттям символів: великі та малі літери, цифри, спеціальні символи тощо.

На цьому зображенні можемо ознайомитися з типовою формою реєстрації на веб-сайті готелю: власні дані, імейл та його дублікат для підтвердження створення аккаунту та надсилання новин або виписок по бронюванню, країна проживання, код номеру залежно від країни та безпосередньо сам номер, а також двічі написаний пароль. Перед тим, як зареєструватися, треба ознайомитися з умовами користування.

У разі незаповнення деяких обов’язкових полів, система підсвічує проблемну область червоним та надає пояснення, що треба виправити. Після натискання кнопки «Приєднатися», у разі вдалої перевірки на існування введених імейлу та номеру телефону, користувачу приходить повідомлення на пошту, після підтвердження якого реєстрацію можна вважати завершеною.

Форма логіну, звичайно, мінімалістична: для входу потрібен лише ID в екосистемі готелю або імейл, та пароль. Пароль, як і належить полям з секретною інформацією, прихований за символами, але має функцію підглядання (для людей, які мають складності зі сліпим вводом). Також існують функції запам’ятовування даних та відновлення або зміни паролю.

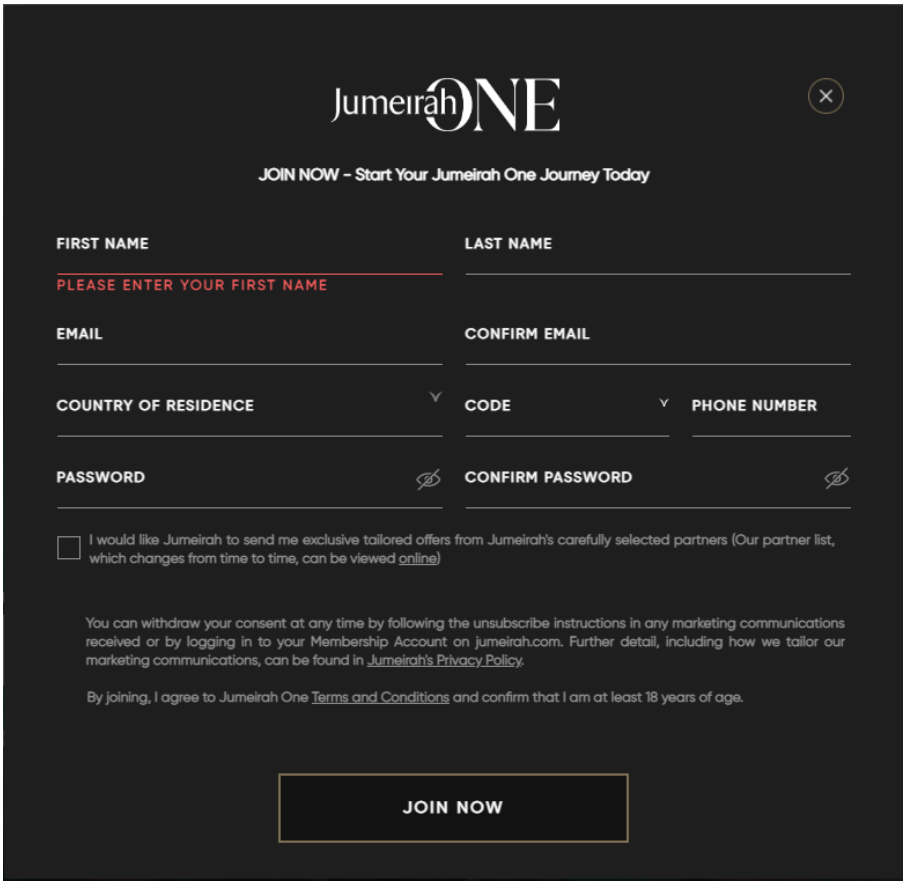


Рис. 1.10 – Форма реєстрації з веб-сайту готелю Burj Al Arab

2. Форма бронювання

Форма отримує від користувача таку інформацію, як кількість людей, які планують проживати в готелі, дати заїзду та виїзду з готелю, клас номеру та багато іншого. Оскільки подібна інформація передбачувана, та повинна вкладатися в певні рамки, для заповнення бронювання в більшості випадків використовуються випадаючі списки. Для заповнення дати використовується ускладнений варіант списку – електронний календар з можливостю обирати дати в певному проміжку. Для відображення дати у якості календаря використовуються спеціальні віджети.

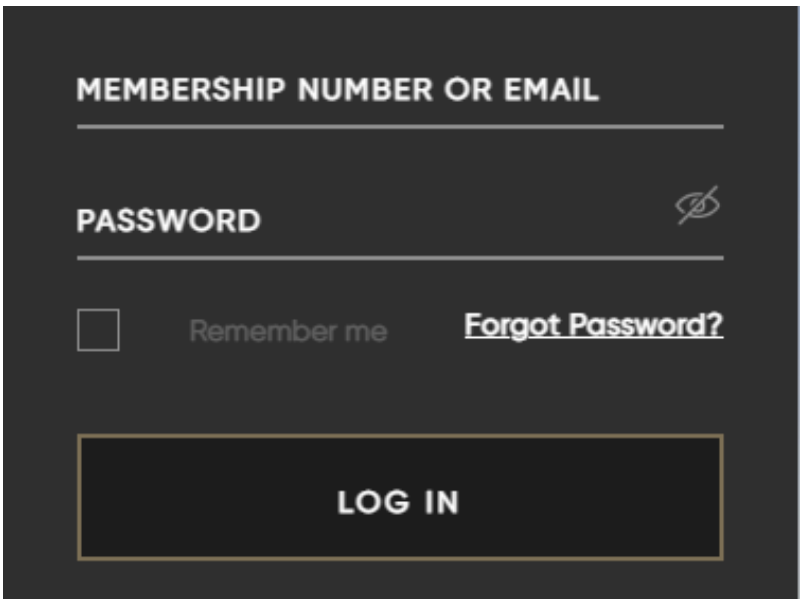


Рис. 1.11 – Форма логіну з веб-сайту готелю Burj Al Arab

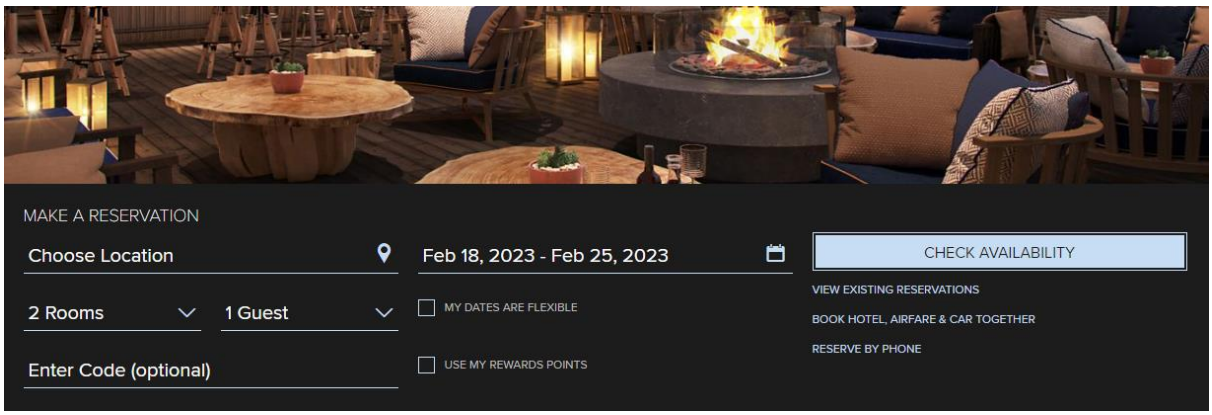


Рис. 1.12 – Форма бронювання з веб-сайту готелю Ritz-Carlton

У цьому випадку форма припускає бронювання одразу деяких номерів та має відповідне поле вводу у вигляді списку для цього, а також поле для вводу спеціального коду на знижку. Для введення інформації про кількість також використовується випадаючий список (від одного до шести гостей на номер). Для задання дати використовується віджет календаря, який відображає обрану область між двома датами, та не дає можливості обрати минулу дату та дату в далекому майбутньому, а також виставити дату виїзду раніше дати заїзду.

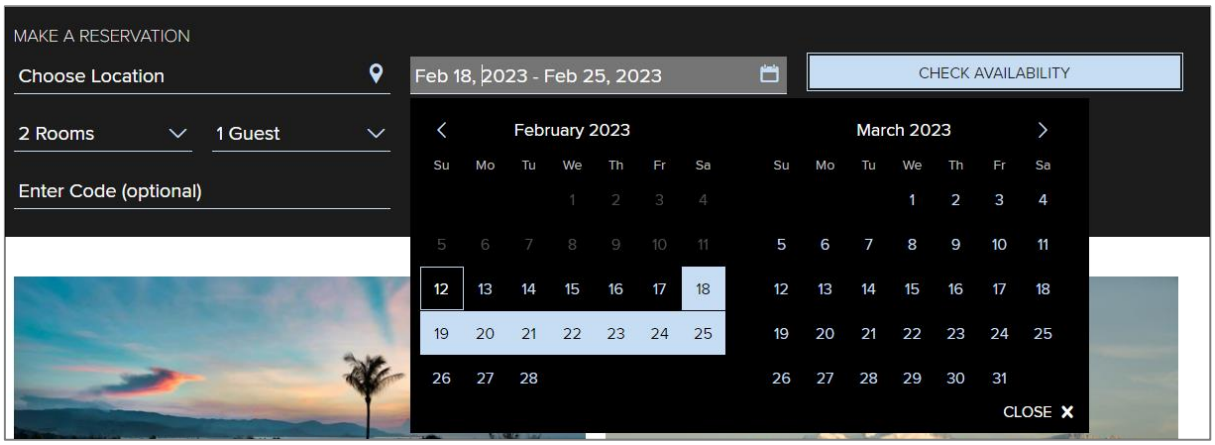


Рис. 1.13 – Календар з форми бронювання з веб-сайту готелю Ritz-Carlton

1.4 Презентація продукту

У останньому розділі пропоную привернути увагу до способів надання інформації щодо переваг тих чи інших типів номерів та відповідних розцінок. Номери можуть відрізнятися за площею, кількістю місць, «люксовістю», видом з вікна тощо. В залежності від цих особливостей може назначатися різна ціна на номери. Демонстрація наповнення номеру суттєво спрощує підбір кімнати для найбільш комфортного користування.

Цінова політика в люксовому сегменті готелів майже завжди не афішується, та надається наприкінці, перед самою оплатою. Так, наприклад, веб-сайт Ritz-Carlton надає фото та відео з номеру, інформацію про площу, краєвид, інші особливості та навіть планування кімнат, але ціну неможливо побачити, навіть якщо натиснути «Забронювати». Лише після вибору дат та перевірки на доступність номерів у цей період будуть надані ціни.

Бюджетні готелі не завжди вдаються до створення власного сайту, покладаючись на більше охоплення аудиторії на букінг-сайтах. Такі сайти агрегують усі готелі в регіоні та розташовують їхні послуги за певну комісію від кожної угоди. Якщо недорогий готель все ж таки має веб-сайт, розцінки, навпаки, демонструються, бо це є його суттєвою перевагою.

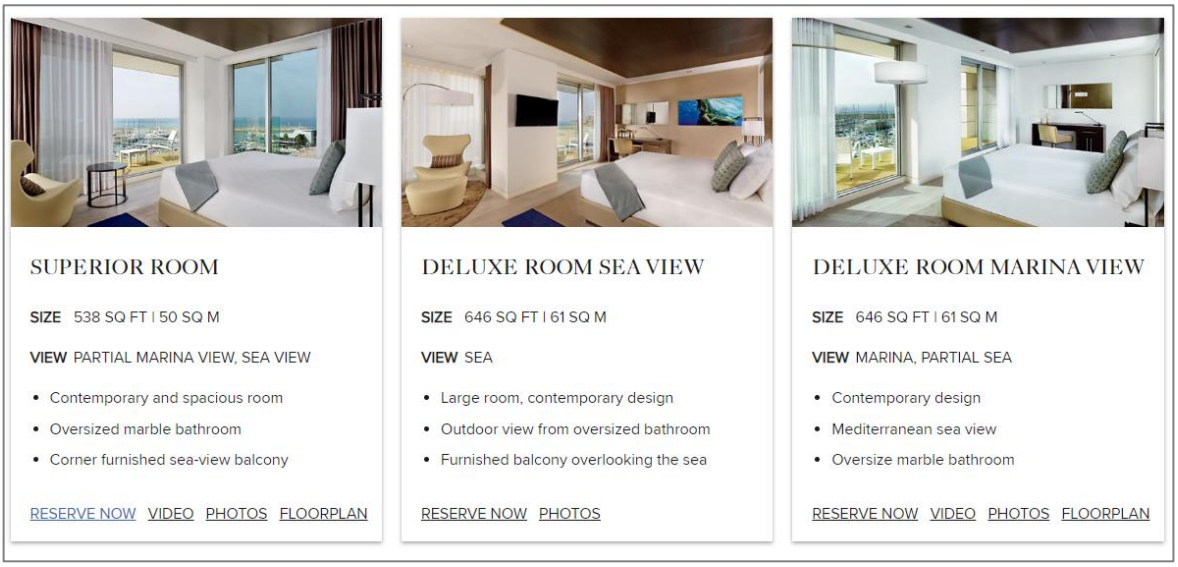


Рис. 1.14 – Таблички з описом номерів з веб-сайту готелю Ritz-Carlton

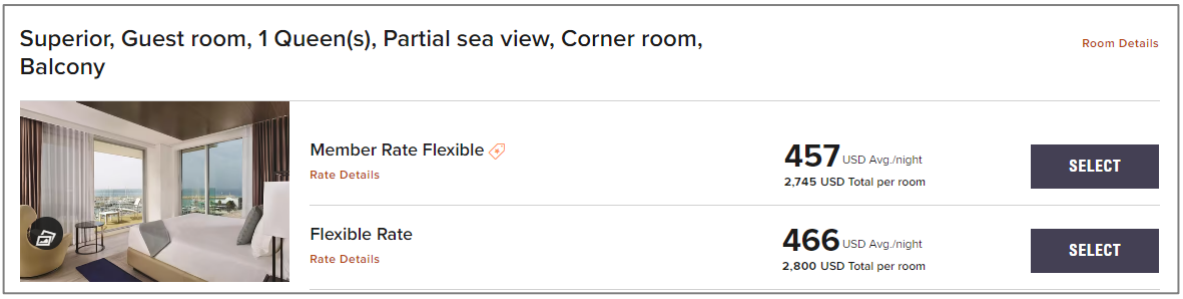


Рис. 1.15 – Табличка з цінами з веб-сайту готелю Ritz-Carlton

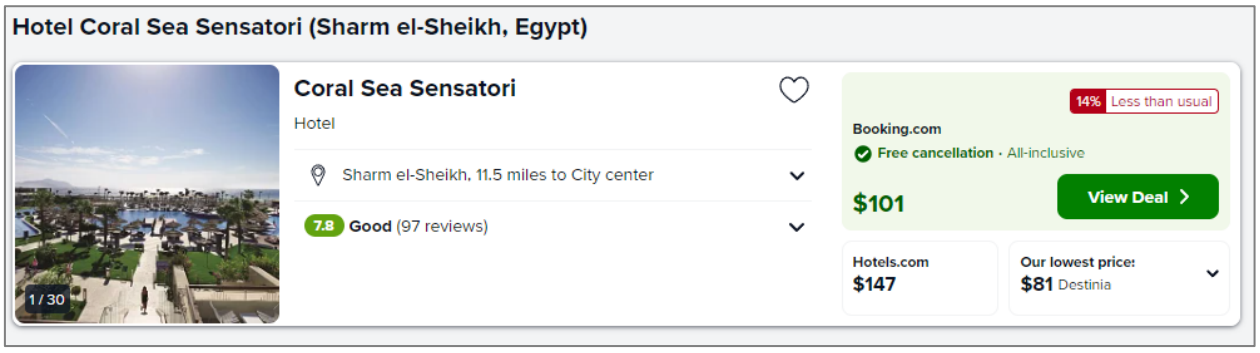


Рис. 1.16 – Табличка з описом готелю з веб-сайту букінгу Trivago

Таким чином, презентація продукту може суттєво відрізнятися в залежності від того, які цілі переслідує замовник веб-ресурсу, але повинна надавати усю необхідну для користувачів інформацію у належному вигляді.

РОЗДІЛ 2 АНАЛІЗ ІНСТРУМЕНТІВ РОЗРОБКИ

При розробці будь-якого продукту вкрай важливо наперед визначитися с основним переліком інструментів розробки, та надати аргументацію, чому саме ці рішення є найбільш вдалими в конкретному випадку. Від обраних технологій буде залежати те, наскільки ефективно буде проходити робота над проектом, та чи вдасться виконати всі поставлені задачі. До аргументів щодо використання тих чи інших інструментів можна віднести широкий функціонал, простоту використання, рівень володіння інструментом, його доступність та сумісність з іншими технологіями, а також багато іншого.

Одним з стандартних рішень для архітектури програмного забезпечення є розділення програмної системи на «front-end» і «back-end». Воно пов'язане з принципом поділу відповідальності між зовнішнім представленням та внутрішньою реалізацією.

Бек-енд розробка представляє собою проектування внутрішньої частини сайту та серверу, та відповідає за логіку і коректну роботу програми. У спрощеному вигляді логіка роботи веб-сайту виглядає наступним чином:

1. Отримання даних від користувача;
2. Обробка даних на сервері;
3. Отримання відповіді від серверу;
4. Вивід результатів користувачу.

Для обробки та передачі даних використовуються різноманітні мови програмування, бази даних, різні інтерфейси та архітектури. У даному проекті стек бек-енд технологій виглядає наступним чином:

– Мова програмування Java

– SQL База даних (у якості СУБД використовується MySQL)

– Фреймворк Spring (такі модулі, як Spring Boot та Spring Security) для створення логіки програми та взаємодії з базою даних та сайтом

Фронт-енд – це поверхневий рівень проектування, відповідний за зовнішній вигляд веб-сайту, оптимізацію елементів на ньому, масштабування тощо. По суті, фронт-енд представляє собою все те, що бачить користувач при відкритті веб-сторінки. Функціонал цього рівню розробки зазвичай виконується на боці клієнта.

Основною задачею фронт-енд розробника є пов'язання представлених дизайнером графічних макетів веб-додатків (сторінки сайту) з бек-ендом. При необхідності реалізація деякого обчислювального функціоналу також відбувається на стороні користувача. Фронт-енд розробка в більшості випадків відбувається за допомогою трьох елементів:

– HTML (HyperText Markup Language) – створення макетів сторінок;

– CSS (Cascading Style Sheets) – (ela.kpi.ua) конфігурація стилів відображення сторінок;

– JavaScript – написання скриптів, які оживляють сайт, додаючи в нього функціонал інтерактивної взаємодії з користувачем.

Розглянемо детальніше усі необхідні для створення проекту інструменти бек-енд та фронт-енд розробки – від мови програмування та системи управління базами даних до фреймворків для візуального наповнення сайту.

2.1 Мова програмування JAVA

У якості мови програмування на проекті буде використовуватися Java. Java це строго-типізована об’єктно-орієнтована мова загального призначення, розроблена компанією Sun Microsystems, яка згодом була поглинена Oracle. Вона компілюється у байт-код, який при виконанні інтерпретується віртуальною машиною (JVM) для конкретної платформи. Тим самим Java відповідає принципу WORA – Write Once, Run Anywhere (з англ. – Написане один раз, запускається де завгодно). Java широко використовується у розробці програм на Android, десктопних та веб-додатків, а також має деяку популярність при роботі з BigData. На Java написані або частково

користуються нею такі відомі проекти, як Android, Jira, Facebook, YouTube, Netflix, Ebay, Amazon і багато інших.

Сильними рисами цієї мови є багатоплатформність і продуктивність, а також те, що Java має величезний вибір бібліотек під будь-які задачі, що робить її найпоширенішою мовою для back-end розробки та однією з самих популярних мов в цілому. З цього витікають такі додаткові переваги, як велика база коду та добре розвинена спільнота, що суттєво спрощує пошук прикладів реалізації потрібної ідеї та відповідей на виниклі запитання.

До недоліків Java можна віднести багатослівність, що частково виправляється сучасними середовищами розробки і впровадженням нових сутностей у нових версіях мови, та велику кількість застарілого коду, що пояснюється довгою історією цієї мови та небажанням або неможливістю бізнеса підтримувати та розвивати свій продукт.

Для комфортного використання будь-якої мови важливе зручне середовище розробки. Для Java, наприклад, існує Intellij IDEA, яка є дуже функціональною, ергономічною та комфортною, та має великий вибір інструментів для роботи з кодом, а також інструменти для спільної віддаленої роботи

2.2 Фреймворк SPRING

Spring – універсальний фреймворк з відкритим кодом для Java-платформи, призваний полегшити розробникам процес проектування і створення додатків. Spring Framework не пов’язаний з конкретною парадигмою або моделлю програмування, що дозволяє використовувати його, як каркас для різних видів додатків.

Основним принципом Spring є універсальність – Spring представляє собою платформу, на основі якої створюється додаток, і він може конфігуруватися як завгодно. Spring часто називають «полегшеним» - він дозволяє реалізовувати задачі з меншою кількістю коду та мінімальною

залежністю від фреймворку. Основна увага у фреймворку приділена налаштуванню залежностей і зв'язків між технологіями.

По своїй сутності Spring є модульним фреймворком, який складається з багатьох компонентів. Одним з основних вважається модуль інверсії контролю, якому делеговано управління залежностями та конфігурація різних компонентів. Він використовує технологію Dependency Injection – ін'єкція залежностей. При такому підході створення залежностей виноситься за рамки самого об'єкта, в спеціальний метод або конструктор.

Також варто детальніше розглянути модуль доступу до даних, який відповідає за взаємодію з системами управління базами даних. За допомогою цього модулю задається логіка взаємодії додатку з базою даних, та доступ різних часток коду до БД. Для надання доступу використовується стандарт Java Database Connectivity (JDBC). База даних в цьому випадку підключається за допомогою драйверів по унікальному URL-посиланню. Крім цього у Spring підтримується ORM – технологія об'єктно-реляційного відображення, яка пов'язує таблиці бази даних до відповідних сутностей ООП.

Ще одним важливим модулем для розробки веб-додатків є модуль MVC, який надає реалізації схеми розподілу додатку на три частини (Модель–Представлення–Контролер) Така схема дозволяє модифікувати кожний компонент окремо. Перевагами цього модулю є чіткий та прозорий розподіл між рівнями, можливість швидко підставляти один інтерфейс на місце іншого.

До інших основних модулів відносяться: Модуль АОП (аспектно-орієнтованого програмування), Модуль транзакцій, Модуль авторизації та аутентифікації (на основі якого розроблений підфреймворк Spring Security), а також модуль віддаленого управління та модуль тестування.

Ще одним важливим доповненням в екосистемі Spring є Spring Boot, який дозволяє спростити процес конфігурації Spring, який часто займає багато часу. Spring Boot функціонує за допомогою стартових пакетів, які включають в себе певний функціонал, пов'язаний з відповідним модулем Spring. Стартові пакети обираються на початку створення додатку або можуть бути

додані/змінені у процесі розробки. Spring Boot ніяк не обмежує розробника, оскільки за необхідності можливо змінити конфігурацію в середині компонента, що дозволяє налаштувати фреймворк саме так, як він повинен працювати. Також Spring Boot підтримує вбудований сервер для розгортання додатків, та надає можливість автоматично створювати та налаштовувати базу даних.

Розглянемо основні переваги Spring:

– Універсальність. Spring підтримує величезну кількість компонентів і технологій, його можна використовувати комплексно для вирішення складних задач.

– Прискорення роботи. Як і більшість інших фреймворків, основною задачею Spring є поліпшення та прискорення роботи. Фреймворк виступає каркасом, в якому багато компонентів та логіки вже мають реалізацію, і розробнику потрібно правильно їх використати.

– Масштабність. Екосистема Spring є дуже масштабною, має додаткові модулі, ПЗ та технології для більшості сучасних напрямків – веб, мікросервіси, робота з даними і багато іншого

– Велика спілка. Spring це сучасний та популярний фреймворк, він має великий попит на ринку праці. Через те навкруги нього сформувалася сукупність людей, які постійно поліпшують та доповнюють фреймворк, та приймають активну участь в обговореннях, що допоможе починаючому програмісту знайти відповіді на більшість запитань.

– Відкритий вихідний код. Ще однією важливою особливістю є те, що Spring безкоштовний, та його код відкритий для всіх охочих, що значно знижує поріг входу та поліпшує програмування.

До недоліків Spring можна віднести:

– Довге налаштування. Spring потребує довготривалої конфігурації з нуля для кожного проекту, що займає багато часу та потребує деяких знань. Для спрощення конфігурації проекту існує Spring Boot, який надає можливість

обрати необхідні стартові пакети, які містять необхідну реалізацію, і в разі необхідності змінити їх налаштування.

– Складний старт. Повноцінне освоювання фреймворку Spring потребує багато часу, та здебільшого відбувається практичним шляхом, оскільки усвідомити механізми роботи Spring в теорії буде досить складно.

2.3 Бази даних

Для того, щоб сайт був не просто «візиткою», а мав обширний функціонал та здатність взаємодіяти з даними користувача, потрібно підключати базу даних. База даних – комп'ютеризована система, яка зберігає організовану інформацію. Управління базами даних відбувається за допомогою Системи Управління Базами Даних (скорочено – СУБД). В залежності від типу зберігання та звернення до даних виділяють реляційні (SQL) та нереляційні (NoSQL – Not Only SQL) бази даних.

Реляційна модель даних представляє інформацію у вигляді взаємопов'язаних таблиць, кожна з яких містить поля та записи, які не повторюються. Їх унікальність забезпечує первинний ключ – мінімальний набір атрибутів, сукупність значень яких однозначно визначає кортеж у відношенні. Інформаційний зв'язок будується трьома типами відношень: «один до одного», «один до багатьох», «багато до багатьох». Такий спосіб організації даних є досить простим для розуміння користувача через представлення даних у вигляді звичайної таблиці, а реляції між ними забезпечують цілісність та безпеку даних.

Концепція NoSQL надає альтернативний погляд на способи зберігання даних. Основною ціллю цього підходу є створення зручних рішень для сфер, в яких організація даних реляційним способом призводить до великих затрат пам'яті, нагромадження схем або повільного функціонування. Ця концепція не заперечує SQL, а лише пропонує альтернативні способи організації даних, інколи частково використовуючи функціонал та стандарти мови SQL. NoSQL

не є представленням якоїсь конкретної моделі, і об’єднує в собі багато різних підходів до проектування бази даних в нереляційному форматі в залежності від поставленої задачі.

Розглянемо основні показники, за якими порівняють ці два підходи:

1. Масштабування. SQL масштабується вертикально, тобто за рахунок збільшення потужності за допомогою використання більш потужних серверів. NoSQL, в свій час, масштабується горизонтально, шляхом додавання додаткових вузлів до вже існуючих, що робить можливості масштабування майже безмежними.

2. Гнучкість. БД на SQL не дають розробникам особливої гнучкості: для роботи з ними спочатку треба продумати та реалізувати схему бази даних, а змінити її буде не так просто. NoSQL, навпаки, легко адаптується до нових схем даних, але може бути складною для налаштування, якщо потребуються жорсткі схеми даних.

3. Структури даних. У SQL дані зберігаються в таблицях, які поєднуються «реляціями», тобто відношеннями. У NoSQL використовуються різні підходи, серед них: бази даних типу «ключ-значення», документо-орієнтовані, стовпчикові та графові.

4. Цілісність даних. SQL слідує принципам ACID – скорочено від Atomicity, Consistency, Isolation, Durability (Атомарність, Узгодженість, Ізольованість, Надійність). NoSQL користується теоремою CAP – Consistency, Availability, Partition Tolerance (Узгодженість, Доступність, Стійкість до (ena.lpnu.ua) розділу).

5. Підтримка та спілка. Оскільки SQL має довгу історію та широкий спектр застосування, навколо нього склалася велика спілка людей, які створюють навчальні посібники, відеоматеріали, статті, відповідають на питання тощо. NoSQL значно молодший, та має багато різних підходів, що ускладнює пошук потрібної інформації.

Оскільки зберігання даних про користувачів веб-сайту є доволі класичним випадком, для більшої наочності буде використовуватися реляційна база даних з СУБД у вигляді MySQL від Oracle.

2.4 Інструменти FRONT-END

Фронт-енд розробка відбувається за допомогою трьох основних компонентів:

– HTML – мова розмітки всіх елементів та документів на сторінці, та їх взаємодія у структурі сторінки.

– CSS – мова стилізації зовнішнього вигляду документа, за допомогою якої браузер розуміє, як необхідно відображати елементи. CSS створює шрифти, кольори, визначає порядок розташування блоків сайту та (www.fasshotel.ch) є відповідним за адаптацію вигляду документа в різних умовах.

– JavaScript – мова, яка робить сторінки більш інтерактивними. За допомогою скриптів та функцій, написаних на цій мові, з'являється можливість відгукуватись на дії користувача, обробляти натискання клавіш, переміщення курсору, кліки мишкою. JavaScript також надає можливість вводити повідомлення, надсилати запити на сервер, а також завантажувати дані без перезавантаження сторінки, і так далі.

Ще одним корисним інструментом фронт-енд розробки виступає Bootstrap – фреймворк, який використовує вищезазначені технології, та значно прискорює та пришвидшує розробку, надаючи можливість розробнику користуватися різноманітними макетами та скриптами, які широко використовується на багатьох сучасних сайтах.

РОЗДІЛ 3 ПРОГРАМНИЙ МОДУЛЬ

На основі проведеного аналізу предметної області та вибору інструментів розробки потрібно сформулювати основні вимоги до функціональної та візуальної складових проекту, а саме:

1) У проекті повинні бути реалізовані повноцінні форми логіну та реєстрації, з перевіркою на коректність заповнених полів.
2) Проект повинен реалізовувати повний цикл створення та оплати букінгу користувачем, аналізуючи введені дані та перевіряючи номери, які відповідають заданим умовам, на предмет доступності.
3) У проекті має бути реалізована роль адміністратора, який має доступ до більшості даних з бази даних та можливість додавати/видаляти/змінювати їх.
4) Дизайн сайту повинен відповідати сучасним тенденціям, мати зручну навігацію та прозорий для користувача інтерфейс резервації та оплати номеру.

Практична реалізація проекту включає в себе усі стадії розробки, починаючи зі створення моделей для бази даних і закінчуючи контентом на html-сторінках. Для того, щоб створити правильно функціонуючий та зручний сайт, потрібно наперед продумати більшість аспектів, які повинні бути реалізовані, і основні шляхи користувачів по цьому сайту, та на основі цього визначитися з архітектурою проекту.

3.1 Логіка програми

Перш за все визначимося з основною схемою роботи програми. В базі даних зберігаються три основні сутності, за допомогою яких функціонує сайт:

– User (користувач), містить необхідні для реєстрації та логіну поля та роль для доступу на сайт.
– Room (номер готелю), містить інформацію про номер.

– Booking (запис про резервацію певної кімнати певним користувачем), містить деталі букінгу, ID зарезервованої кімнати та ID користувача, який оформив резервацію.

Користувач авторизується та заповнює інформацію для букінгу: дати заселення та виселення, а також кількість гостей. Програма перевіряє кімнати та існуючі по ним броні на співпадіння заданим критеріям, та у разі наявності вільного простору у графіку створює букінг. Після цього користувачу необхідно його сплатити.

Усього існує три формати доступу на сайт:

1) Неавторизовані користувачі бачать основну сторінку сайту, наповнену контентом та інформацією, неактивну форму створення букінгу та кнопку для логіну/реєстрації.
2) Авторизований користувач з правами «USER» бачить основну сторінку та має можливість зробити резервацію, має доступ до власного кабінету, де він може змінити інформацію про себе та сплатити букінг.
3) Адміністратор (який має права «ADMIN») має такий же функціонал, як і звичайний користувач, та додаткову панель адміністратора, де виводяться усі дані про користувачів, букінги та кімнати, існує можливість додавати або видаляти, змінювати інформацію.

3.2 Архітектура проекту

Класичні веб-додатки на основі Spring функціонують за схемою MVC, яка складається з трьох сутностей:

– Model (модель) надає дані та реагує на команди контролера, змінюючи свій стан.

– View (представлення) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі.

– Controller (контролер) інтерпретує дії користувача, сповіщаючи модель про необхідність змін. (dut.edu.ua)

При створенні Spring-додатку розробник не має необхідності створювати таблиці бази даних своїми руками, оскільки для цього існує бібліотека Hibernate, яка є найбільш поширеною реалізацією JPA (Java Persistence API – специфікація мови, яка надає можливість зберігати Java-об’єкти у базі даних).

Замість таблиці у базі даних створюється Java-клас, помічений спеціальною анотацією @Entity . Його обов’язковими атрибутами є ID, конструктори (пустий та порожній) та методи доступу до полів класу (Get- та Set-методи). За допомогою бібліотеки Lombok можна суттєво скоротити шаблонний код конструкторів та методів, замінивши його додатковими анотаціями. Готовий для обробки за допомогою Hibernate клас має такий вигляд (рис. 3.1).

```java
@Entity
@Table(name = "rooms")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Room {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private int number;
    private String level;
}
```

Рис. 3.1 – Приклад моделі

Hibernate, зчитавши необхідні анотації, сприймає java-клас як шаблон для створення таблиці у базі даних: таблиця отримує назву класу, змінні Java-класу стають її полями. У Hibernate генерація ID відбувається автоматично. У базі даних це виглядає наступним чином:

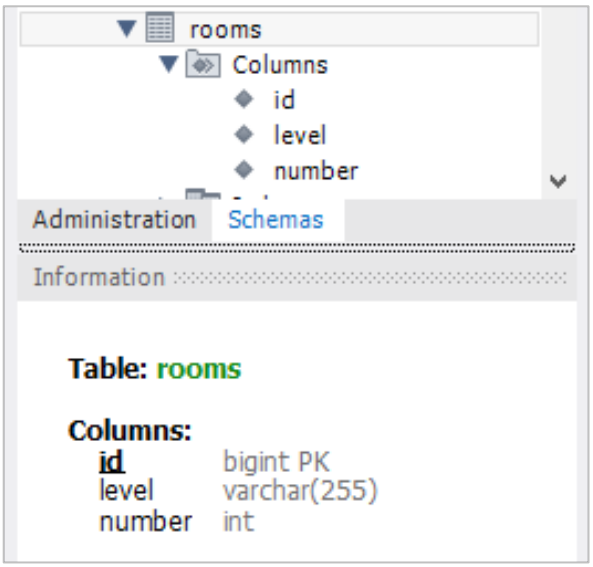


Рис. 3.2 – Вигляд таблиці у БД

Для виведення інформації з бази даних на сайт необхідні ще представлення та контролер. Представлення являє собою html-сторінки, які і бачить користувач при роботі з сайтом. Передача даних на сторінку відбувається у контролері, а саме за допомогою додавання необхідного атрибуту (пара назва-об’єкт). На сторінці ці дані виводяться за допомогою додаткових інструментів зі специфічним синтаксисом. В цьому проекті для цього використовується FreeMarker.

```
@Controller
@RequiredArgsConstructor
public class RoomController {
    private final RoomService roomService;

    @GetMapping("/room/{id}")
    public String roomInfo(@PathVariable Long id, Model model){
        model.addAttribute( attributeName: "room", roomService.getRoomById(id));
        return "room-info";
    }
```

Рис. 1.3 – Вигляд контролеру

Контролер також помічається відповідною анотацією @Controller, що дає сигнал для Spring обробляти цей java-файл специфічним чином. Контролер

зберігає в собі два типи методів – «GetMapping» та «PostMapping», за допомогою яких відловлює певні URL-посилання, та надає відповідні до них представлення.

Метод з анотацією @GetMapping по суті є звичайним переходом на іншу сторінку, для якої потрібно підвантажити необхідні дані, додавши їх у модель даних. В парадигмі роботи з даними CRUD (Create, Read, Update, Delete) GetMapping являє собою функцію «Read» – читання даних. За три інших – додавання, зміну та видалення даних відповідає @PostMapping.

Зазвичай ці методи потребують додаткової обробки або фільтрації даних, і для цього гарним тоном є введення допоміжної ланки – сервісів, або служб. Сервіси також мають спеціальну анотацію @Service, та являють собою набір методів роботи з даними. Дані для сервісів достаються з відповідних репозиторіїв, які є нащадками JpaRepository – інтерфейсу, який забезпечує основні операції по пошуку, збереженню, видаленню та зміні даних.

```java
import org.springframework.data.jpa.repository.JpaRepository;

public interface RoomRepository extends JpaRepository<Room, Long> {
    Room findByNumber(int number);
}
```

Рис. 3.4 – Вигляд інтерфейсу

```java
@Service
@RequiredArgsConstructor
public class RoomService {
    private final RoomRepository roomRepository;


    public List<Room> listRooms() { return roomRepository.findAll(); }
```

Рис. 3.5 – Вигляд сервісу

3.3 Програмна реалізація

Створюємо за допомогою Spring Initalizr проект, додаємо у pom.xml усі необхідні залежності:

Рис. 3.6 – Залежності у pom.xml

Створюємо моделі номеру, користувача та букінгу. Прописуємо необхідні анотації та задаємо потрібний нам формат даних та відповідні назви змінних.

Модель кімнати готелю містить записи щодо її номеру та місткості, на основі чого буде змінюватися ціна. В цьому проекті використаний формат зв'язку таблиць «uni-directional», оскільки специфіка створення букінгу та частота запитів до таблиць користувача та кімнати не потребують сильної зв'язності.

```java
@Entity
@Table(name = "rooms")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Room {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private int number;
    private String level;
}
```

Рис. 3.7 – Модель «Room»

Таблиця Booking по суті є збіркою записів щодо бронювання, на основі якої складається графік завантаженості готелю. Містить такі поля, як: дата заселення, дата виселення, статус оплати та ціна, яка вираховується в залежності від часу знаходження в готелі та типу обраного номеру.

Задаємо для букінгу зв’язки з сутностями користувача та кімнати у форматі «ManyToOne», оскільки один й той самий користувач може робити багато букінгів, та на одну й ту саму кімнату буде зроблено багато резервацій протягом певного проміжку часу.

Сутність користувача містить інформацію, яка запитується при реєстрації: повне ім’я, поштова адреса, пароль (пароль зберігається у закодованому форматі) та номер телефону. Крім цього, User має статус активності, змінюючи який його можна «забанити» та обмежити доступ до ресурсу. Оскільки додаток розділений на частину звичайного користувача та частину адміністратора, тут також створене поле для ролей, яке зберігає значення зі спеціального Enum. Для того, щоб використовувати цей клас у якості облікового запису при авторизації на сайт, потрібно розширити цей клас інтерфейсом UserDetails, та перевизначити відповідні функції.

Рис. 3.8 – Модель «Booking»

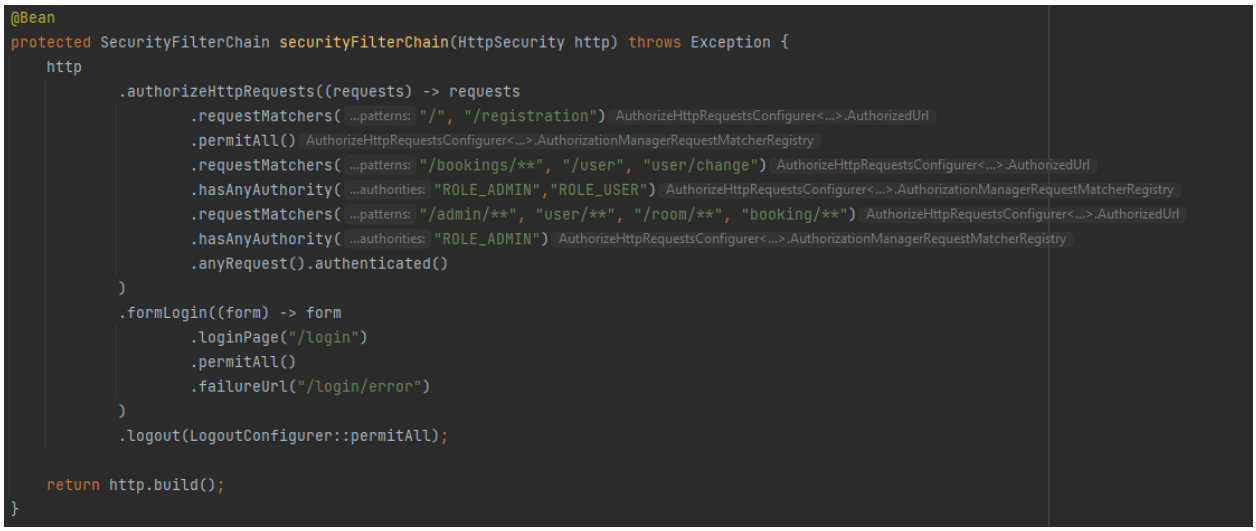
Для того, щоб модернізувати протокол «Spring security», потрібно створити власну конфігурацію, та перевизначити механізм завантаження користувача для подальших перевірок логіну та паролю.

Рис. 3.9 – Модель «User»

```java
@Service
@RequiredArgsConstructor
public class CustomUserDetailsService implements UserDetailsService {
    private final UserRepository userRepository;
    @Override
    public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {
        return userRepository.findByEmail(email);
    }
}
```

Рис. 3.10 – Перевизначення конфігурації UserDetailsService

Крім того, у файлі конфігурації потрібно записати ланцюг фільтрів дозволу, який надає користувачу можливість заходити на ту чи іншу частину сайту в залежності від того, чи він авторизований, та яку роль має.



```java
@Bean
protected SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    http
            .authorizeHttpRequests((requests) -> requests
                    .requestMatchers( ...patterns: "/", "/registration") AuthorizeHttpRequestsConfigurer<...>.AuthorizedUrl
                    .permitAll() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
                    .requestMatchers( ...patterns: "/bookings/**", "/user", "user/change") AuthorizeHttpRequestsConfigurer<...>.AuthorizedUrl
                    .hasAnyAuthority( ...authorities: "ROLE_ADMIN","ROLE_USER") AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
                    .requestMatchers( ...patterns: "/admin/**", "user/**", "/room/**", "booking/**") AuthorizeHttpRequestsConfigurer<...>.AuthorizedUrl
                    .hasAnyAuthority( ...authorities: "ROLE_ADMIN") AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
                    .anyRequest().authenticated()
            )
            .formLogin((form) -> form
                    .loginPage("/login")
                    .permitAll()
                    .failureUrl("/login/error")
            )
            .logout(LogoutConfigurer::permitAll);

    return http.build();
}
```

Рис. 3.11 – Ланцюг фільтрів доступу

Створюємо за допомогою спеціальної анотації клас-контролер, в якому перевизначимо мапінг логіну та додамо механізм реєстрації. За допомогою GetMapping-ів відловлюємо посилання, які стосуються логіну чи реєстрації, та повертаємо свої html-сторінки з відповідними формами. Прописуємо додаткові умови, які у разі спрацювання передаватимуть на сторінку повідомлення про невдалу дію – спробу створити користувача з вже використаним логіном або неправильне введення даних для входу.

Для обробки даних також потрібні деякі додаткові перевірки і маніпуляції, які доречно буде вивести у окремий сервіс UserService. При реєстрації в методі створення користувача відбуваються запити у репозиторій та заповнення технічних змінних, які не вказуються користувачем при реєстрації.

Рис. 3.12 – Контролер «UserController»

```
@Service
@RequiredArgsConstructor
public class UserService {
    private final UserRepository userRepository;
    private final PasswordEncoder passwordEncoder;

    public boolean createUser(User user){
        if(userRepository.findByEmail(user.getEmail()) != null ){
            return false;
        }
        user.setActive(true);
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        user.getRoles().add(Role.ROLE_USER);
        userRepository.save(user);
        return true;
    }
```

Рис. 3.13 – Сервіс «UserService»

Створюємо представлення – сторінки логіну і реєстрації. Інформація на цих сторінках заповнюється за допомогою атрибуту form, в який ми прописуємо:

- посилання;
- тип методу;
- спеціальний csrf-токен, який захищає від зловмисного підроблення форми та подальшого використання даних користувача;
- поля для введення з відповідними унікальними іменами;
- кнопку для відправки форми.

Рис. 3.14 – Post-форма логіну

Сторінка реєстрації виглядає схожим чином, але містить поля для введення додаткової інформації (телефон чи ім’я). Крім того, на клієнтському боці відбувається перевірка на співпадіння паролю з його повторним введенням у полі підтвердження.

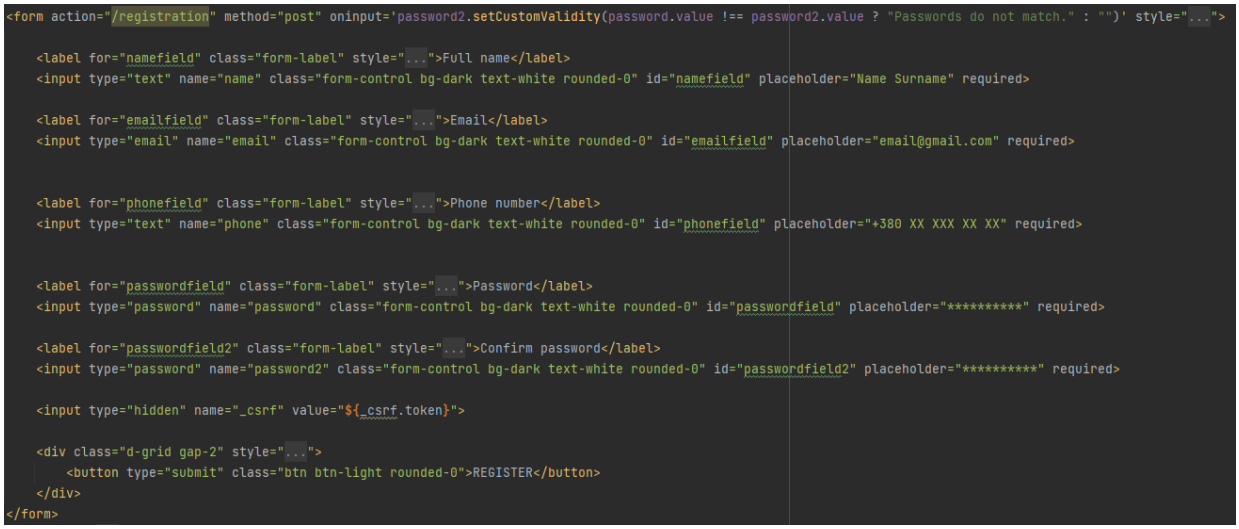


Рис. 3.15 – Post-форма реєстрації

На кожній сторінці сайту задля комфорту та візуальної орієнтації існують меню навігації та футер, які містять корисні посилання на розділи сайту чи сторонні ресурси, а також кнопки навігації. Крім того, сторінки містять посилання на стилі Bootstrap та додаткові бібліотеки, такі, як jQuery.

Рис. 3.16 – html-код футеру

За допомогою такого елементу FreeMarker, як блок <#if>, відбувається показ тих кнопок, які зараз необхідні користувачу. Якщо в моделі не існує інформації про користувача, йому висвічується кнопка реєстрації, в іншому випадку – кнопка закінчення сесії (logout). Якщо користувач є адміністратором, йому доступна кнопка, яка відсилає його у кабінет адміністратора.

Рис. 3.17 – html-код навігації

Переходимо до реалізації основної сторінки сайту, на якій відбувається створення букінгу користувачем. Для цього прописуємо post-метод, який приймає у себе усі параметри, що стосуються букінгу, та звертається до методу checkFreeRoom() сервісу BookingService. В залежності від перевірки на наявність вільних номерів можливі дві логічні гілки: збереження букінгу та перенаправлення користувача у його кабінет, де він може сплатити букінг, або виведення повідомлення про відсутність вільних кімнат та подальші рекомендації.

```
@PostMapping("/bookings/create")
public String createBooking(String dateIn, String dateOut, String level, Principal principal, Model model, RedirectAttributes redirectAttrs){
    Room freeRoom = bookingService.checkFreeRoom(dateIn, dateOut, level);
    if(freeRoom != null){
        bookingService.saveBooking(principal, bookingService.bookingPreset(dateIn, dateOut, b: false, freeRoom, bookingService.countPrice(dateIn, dateOut,level)));
        redirectAttrs.addFlashAttribute( attributeName: "posted",  attributeValue: true);
        return "redirect:/user#bookingarea";
    }
    else{
        System.out.println("No free rooms");
        model.addAttribute( attributeName: "bookings", bookingService.listBookings());
        model.addAttribute( attributeName: "rooms", roomService.listRooms());
        model.addAttribute( attributeName: "currentUser", bookingService.getUserByPrincipal(principal));
        model.addAttribute( attributeName: "errorMessage",  attributeValue: "No free rooms. Try another dates or number of guests");
        return "bookings";
    }
}
```

Рис. 3.18 – Мапінг створення букінгу

Метод аналізу при цьому виглядає наступним чином: відбувається фільтрація кімнат та букінгів на предмет тих, що задовольняють параметрам

(букінг – по датам, кімната – по рівню місткості), та порівняння дат букінгів з датами, вказаними у параметрі методу. Якщо виявляється колізія, кімната, на яку оформлений цей букінг, видаляється зі списку доступних. Якщо за підсумком перевірки у масиві залишаються кімнати, букінгу користувача присвоюється ID цієї кімнати. Якщо кімнат немає, метод повертає null.

Рис. 3.19 – Алгоритм пошуку вільних кімнат

У збереженні букінгу приймають участь три методи:

– countPrice() вираховує кількість днів, протягом яких буде зайнята кімната, та рахує ціну в залежності від типу номеру.

– bookingPreset() збирає окремі параметри у єдиний об’єкт класу Booking.

– saveBooking() зберігає об’єкт у репозиторії.

Букінг заповнюється через форму, яка містить дату заселення, дату виселення та тип номеру. Дані для букінгу приймаються через календар та розкривний список select, що унеможливлює неправильне введення.

Рис. 3.20 – BookingService

Для календарів прописаний додатковий скрипт, який робить дату заселення доступною лише починаючи з нинішнього дня, а для дати виселення скрипт зчитує дані з першого календаря та ставить їх як мінімальну дату. Таким чином, неможливо обрати дату виселення, яка буде раніше, ніж дата заселення, або обрати дати з минулого.

Рис. 3.21 – Форма створення букінгу

```
<script>
    var today = new Date();
    var dd = today.getDate();
    var mm = today.getMonth() + 1; //January is 0!
    var yyyy = today.getFullYear();
    if (dd < 10) {
        dd = '0' + dd;
    }
    if (mm < 10) {
        mm = '0' + mm;
    }
    today = yyyy + '-' + mm + '-' + dd;
    document.getElementById("checkIn").setAttribute("min", today);


    function updatedate() {
        var firstdate = document.getElementById("checkIn").value;
        document.getElementById("checkOut").value = "";
        document.getElementById("checkOut").setAttribute("min",firstdate);
    }
</script>
```

Рис. 3.22 – Скрипт для календаря

Після створення букінгу програма перенаправляє користувача у його власний кабінет, де йому виводиться список букінгів, які він зробив, та статус їх оплати. У кабінеті також міститься уся інформація про користувача, та є можливість змінити власні дані або пароль.

Рис. 3.23 – UserController

```
public User getUserByPrincipal(Principal principal) {
    if(principal == null){
        return new User();
    }
    return userRepository.findByEmail(principal.getName());
}

public void deleteBooking(Long id) { bookingRepository.deleteById(id); }

public Booking getBookingById(Long id) { return bookingRepository.findById(id).orElse( other: null); }

public List<Booking> getBookingsByUserId(Long id) {
    List<Booking> bookingList = bookingRepository.findAll();
    bookingList.removeIf(booking -> !booking.getUser().getId().equals(id));
    return  bookingList;
}
```

Рис. 3.24 – UserService

Оплата букінгу відбувається на окремій сторінці, яка веде користувача на банківську форму.

```
@GetMapping("/bookings/pay/{id}")
public String bookingPayPage(@PathVariable Long id, Model model, Principal principal){
    User user = bookingService.getUserByPrincipal(principal);
    List<Booking> bookings = bookingService.listBookings();
    for(Booking booking : bookings){
        if(booking.getId().equals(id) && booking.getUser().equals(user)){
            model.addAttribute( attributeName: "booking", bookingService.getBookingById(id));
            return "booking-pay";
        }
    }

    return "redirect:/user";
}

@PostMapping("/bookings/pay/{id}")
public String bookingPay(@PathVariable Long id, Principal principal){
    Booking booking = bookingService.getBookingById(id);
    booking.setPaid(true);
    bookingService.saveBooking(principal, booking);
    return "redirect:/user";
}
```

Рис. 3.25 – BookingController

Адміністратор має можливості банити користувача чи змінювати його роль (наприклад, надати йому роль адміністратора). Бан відбувається за допомогою зміни булевої властивості isActive, після цього при спробі логіну на сайт доступ буде заблоковано.

Рис. 3.26 – AdminController

Крім того, у своєму кабінеті адміністратор має можливість переглядати усі дані крім паролів, та робити з ними будь-які дії.

Рис. 3.27 – html-таблиця

Дані на сторінці зберігаються у форматі таблиць, а пошук по ним, сортування та пагінація відбуваються за допомогою плагіну DataTable.

```
<!-- DataTable-->
<link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/5.2.0/css/bootstrap.min.css">
<link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.13.4/css/dataTables.bootstrap5.min.css">
<script type="text/javascript" src="https://code.jquery.com/jquery-3.5.1.js"></script>
<script type="text/javascript" src="https://cdn.datatables.net/1.13.4/js/jquery.dataTables.min.js"></script>
<script type="text/javascript" src="https://cdn.datatables.net/1.13.4/js/dataTables.bootstrap5.min.js"></script>

<script>
    $(document).ready(function () {
        $('#example').DataTable();
        $('#example1').DataTable();
        $('#example2').DataTable();
    });
</script>
```

Рис. 3.28 – Завантаження скриптів

Видалення даних відбувається на окремих сторінках, які містять додаткову інформацію та посилання на пов’язані записи. Такий підхід мінімізує можливість випадкового видалення даних при перегляді таблиці. Так, наприклад контролер для номерів виглядає наступним чином:

```
@Controller
@RequiredArgsConstructor
public class RoomController {
    private final UserService userService;
    private final BookingService bookingService;
    private final RoomService roomService;


    @GetMapping("/room/{id}")
    public String roomInfo(@PathVariable Long id, Model model){
        model.addAttribute( attributeName: "room", roomService.getRoomById(id));
        model.addAttribute( attributeName: "bookings", bookingService.getBookingByRoomId(id));
        return "room-info";
    }

    @PostMapping("/room/create")
    public String createRoom(Room room, Model model){
        model.addAttribute( attributeName: "bookings", bookingService.listBookings());
        model.addAttribute( attributeName: "rooms", roomService.listRooms());
        model.addAttribute( attributeName: "users", userService.list());
        if(!roomService.saveRoom(room)){
            model.addAttribute( attributeName: "errorMessage",
                    attributeValue: "Room number already exists (" + room.getNumber() + ")");
            return "/admin";
        }
        return "redirect:/admin#roomarea";
    }

    @PostMapping("/room/delete/{id}")
    public String deleteRoom(@PathVariable Long id){
        roomService.deleteRoom(id);
        return "redirect:/admin#roomarea";
    }
}
```

Рис. 3.29 – RoomController

Додавання кімнат відбувається за допомогою спеціальної форми на сторінці адміністратора, яка знаходиться під таблицею з кімнатами. При спробі створити кімнату з вже існуючим номером буде виведене відповідне повідомлення.

```
<form action="/room/create" method="post">
    <h1 style="...">Add room:</h1>
<div class="row">
    <div class="col">
        <input type="text" class="form-control bg-dark text-white rounded-0" placeholder="number" name="number" required/>
    </div>
    <div class="col">
        <select name="level" id="level" class="form-control bg-dark text-white rounded-0">
            <option value="1-2 Guests">1-2 Guests</option>
            <option value="3-4 Guests">3-4 Guests</option>
            <option value="5+ Guests">5+ Guests</option>
        </select>
    </div>
    <div class="col">
        <input type="hidden" name="_csrf" value="${_csrf.token}">
        <span class="d-grid gap-2" tabindex="0">
            <button type="submit" class="btn btn-secondary rounded-0">ADD ROOM</button>
        </span>
    </div>
</div>
</form>
<#if errorMessage??>
    <h2 style="...">${errorMessage}</h2>
</#if>
```

Рис. 3.30 – Форма створення кімнати

Схоже наповнення мають і окремі сторінки букінгу або інформації про користувача:

```
<div class="div-2" style="...">
    <div class="container-xl bg-dark text-light">
        <div class="row" style="...">
            <div class="col-8">
                <h2>More about booking:</h2>
                <p> ID: <b>${booking.id?c}</b></p>
                <p> Date In: <b>${booking.dateIn}</b></p>
                <p> Date Out: <b>${booking.dateOut}</b></p>
                <p> Price: <b>${booking.price} $</b></p>
                <p> Payment: <#if (booking.paid)><b>PAID</b><#else><b>NOT PAID</b></#if></p>
                <p> Room number: <b><a href="/room/${booking.room.id?c}">${booking.room.number}</a></b></p>
                <p> Room capacity: <b>${booking.room.level}</b></p>
                <p>Booking author: <b><a href="/user/${booking.user.id?c}">${booking.user.name}</a></b></p>
                <p> User email: <b>${booking.user.email}</b></p>
                <p> User phone number: <b>${booking.user.phone}</b></p>
                <br>
                <form action="/booking/delete/${booking.id?c}" method="post">
                    <input type="hidden" name="_csrf" value="${_csrf.token}">
                    <input type="submit" style="..." class="btn btn-lg rounded-0" value="Delete booking"/>
                </form>
            </div>
        </div>
    </div>
</div>
```

Рис. 3.31 – Сторінка букінгу

Рис. 3.32 – Сторінка юзера

3.4 Демонстрація продукту

При відкритті сайту користувача зустрічає вітальна сторінка, вона містить навігаційну панель та форму для бронювання (неактивну без авторизації).

При прокручуванні сторінки користувач має змогу ознайомитися з особливостями готелю, зображеннями номерів та прилеглої території, прочитати опис кімнат та доступних турів.

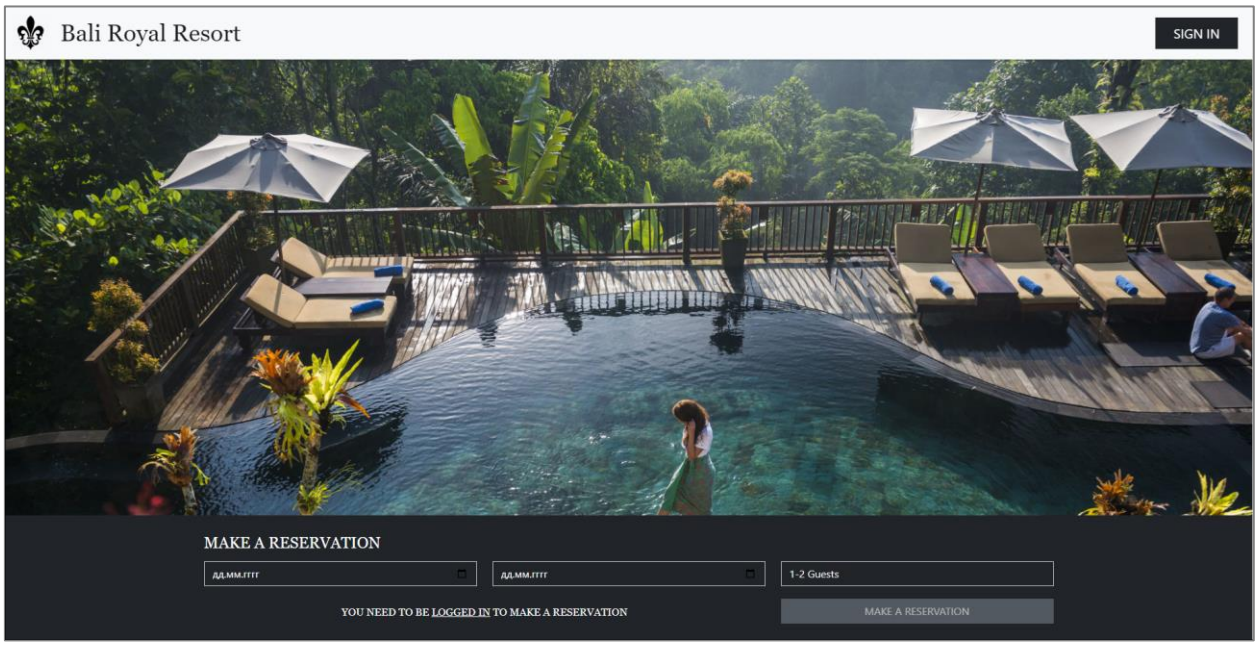


Рис. 3.33 – Вітальна сторінка

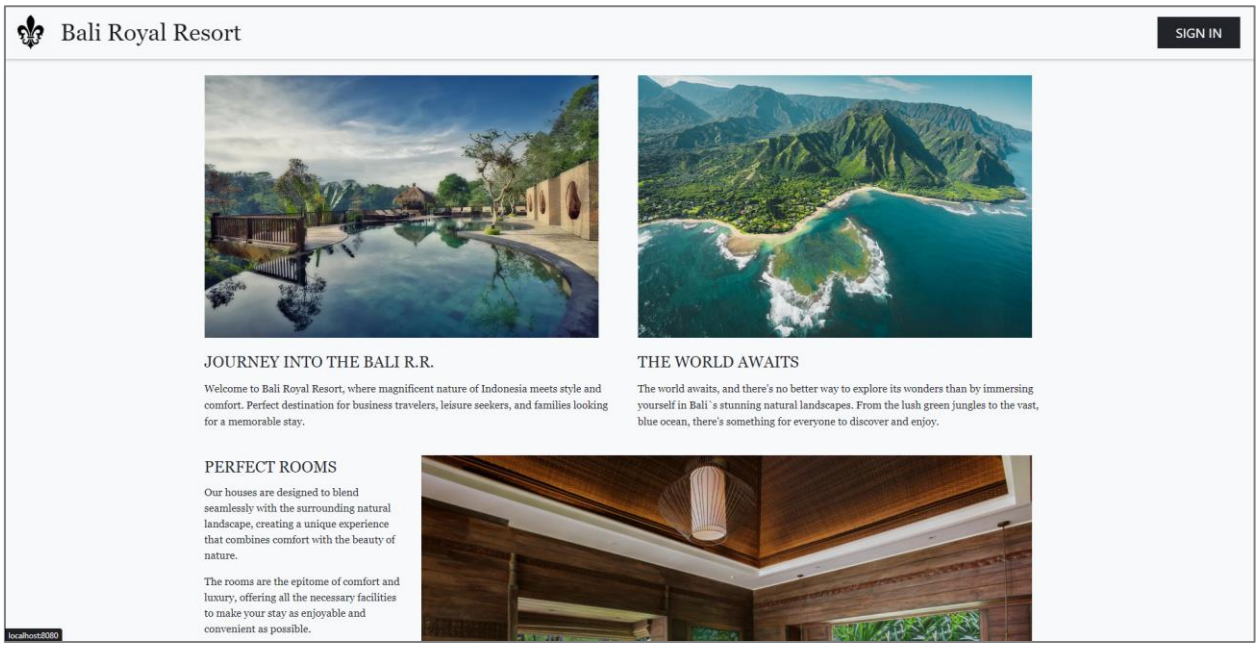


Рис. 3.34 – Наповнення сайту

Рис. 3.35 – Наповнення сайту (2)

Сторінки логіну та реєстрації мають схожий вигляд, та містять у центрі форму, яка приймає дані користувача. У разі введення некоректних даних користувачу виводиться повідомлення, в якому вказана причина.

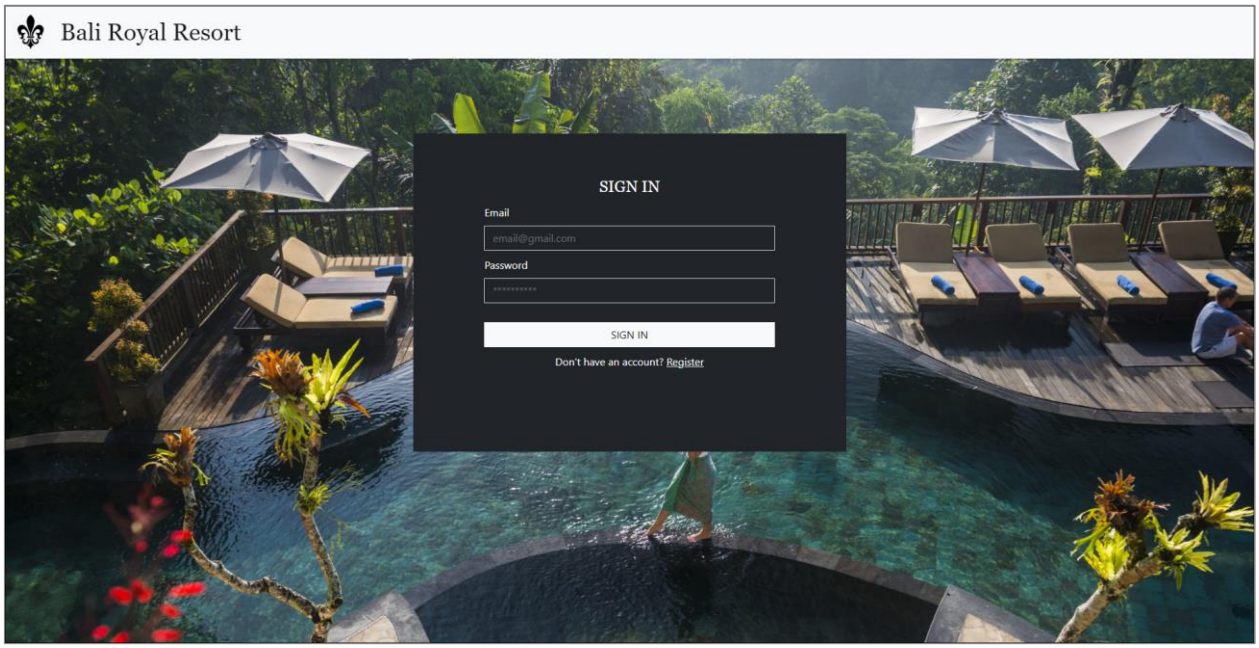


Рис. 3.36 – Логін

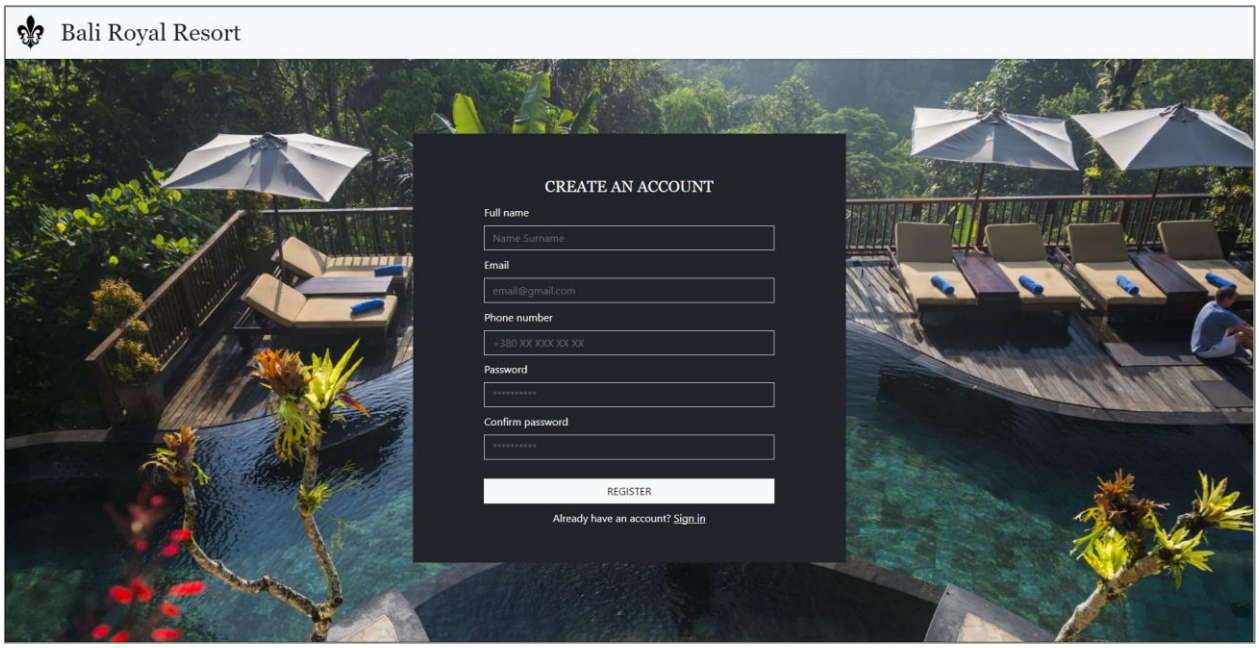


Рис. 3.37 – Реєстрація

При вдалій авторизації користувач отримує змогу зробити резервацію номеру, заповнивши дати виселення та заселення, а також обравши тип кімнати. У календарі можливо вибрати тільки дати.

Рис. 3.38 – Календарі

У випадку відсутності доступних номерів користувачу висвічується повідомлення, яке містить рекомендації змінити дати або місткість номеру.

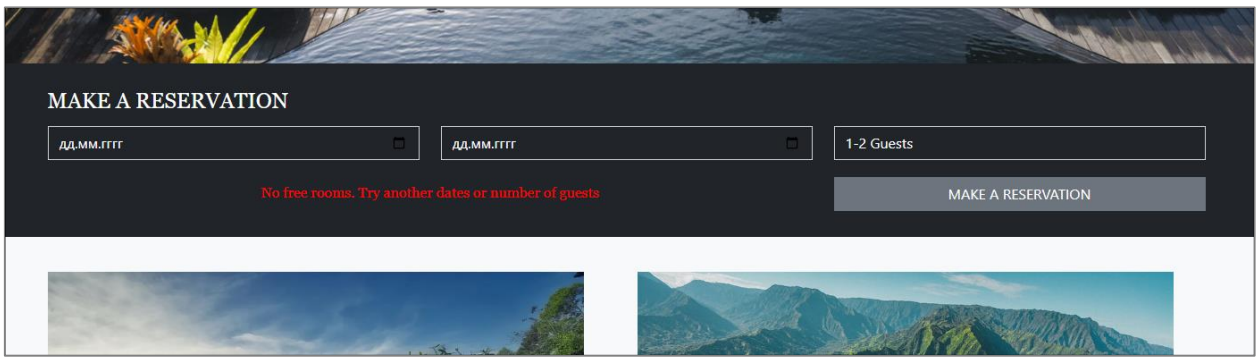


Рис. 3.39 – Повідомлення про відсутність вільних номерів

Якщо бронювання номеру відбувається успішно, система перенаправляє користувача на його акаунт, де тільки що зроблений букінг тимчасово підсвічується зеленим кольором.

Рис. 3.40 – Кабінет користувача

Після цього користувачу пропонується здійснити оплату, після чого запис про букінг горить як сплачений.

Рис. 3.41 – Оплата букінгу

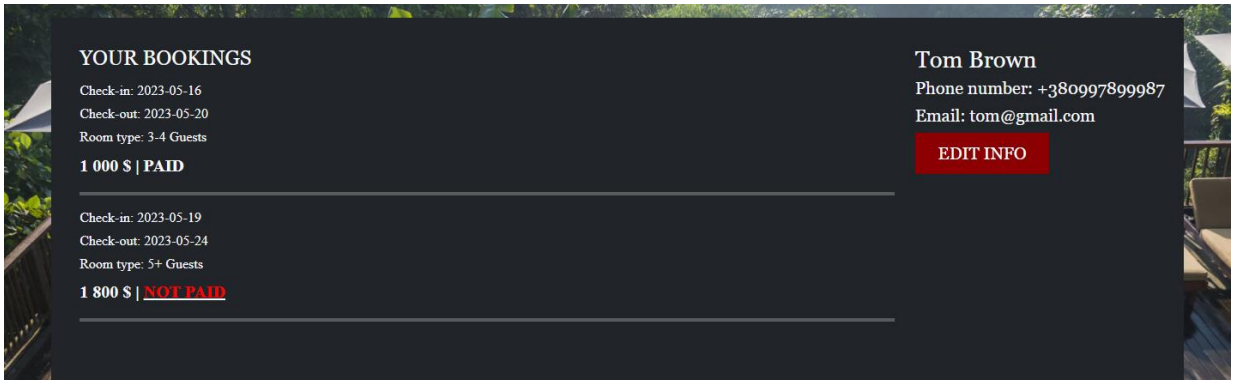


Рис. 3.42 – Вигляд букінгів у кабінеті

На цій же сторінці користувач має змогу редагувати інформацію про себе, зокрема, редагувати повне ім’я, телефон чи пароль.

Кабінет адміністратора містить таблиці користувачів, номерів та букінгів. За допомогою плагіну DataTable ці таблиці можна відфільтрувати по будь-якому параметру, здійснити пошук по ключовому слову та обрати

кількість записів, які будуть одночасно відображені в цьому блоці інформації (за замовчуванням – 10 записів).

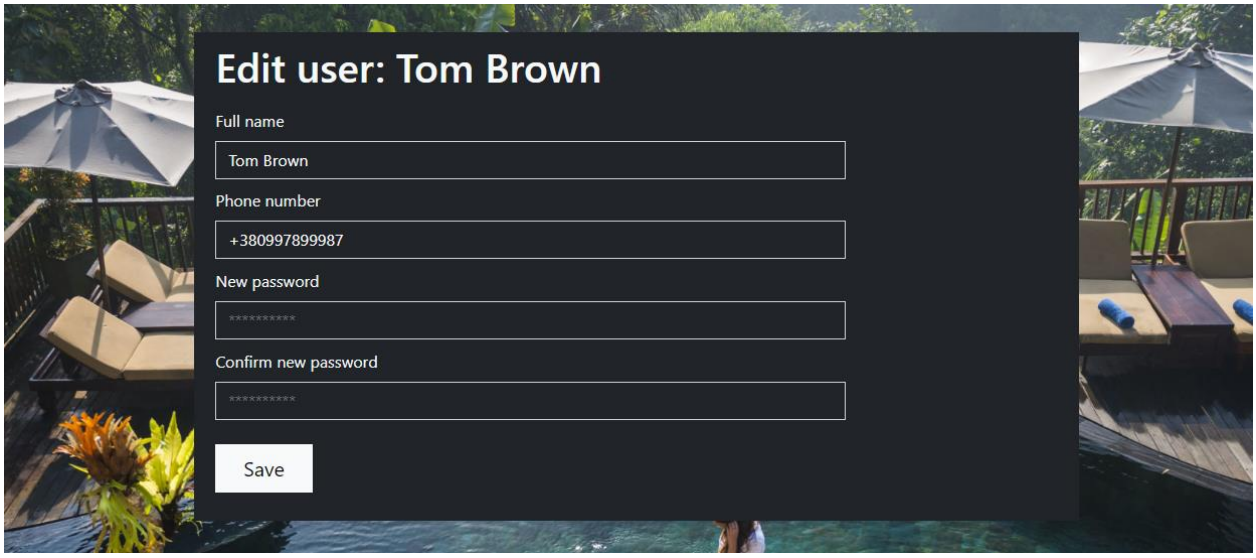


Рис. 3.43 – Зміна інформації про користувача

Рис. 3.44 – Кабінет адміністратора

У кожній таблиці у випадку, якщо зустрічається сутність (користувач, номер, букінг), про яку можна отримати більше інформації, або її можна змінити, на неї можна натиснути та перейти на окрему сторінку. Ці сторінки також містять пов’язану інформацію, та кнопку видалення запису з бази даних. Такі сторінки виглядають наступним чином:

Рис. 3.45 – Сторінки інформації про сутності

Також у адміністратора є можливість зміни ролі користувача. Для виключення можливості помилок ця функція також реалізована на окремій сторінці.

Рис. 3.46 – Зміна ролі користувача

Загалом, інтерфейс продукту є інтуїтивно зрозумілим: він виконаний з оглядкою на сучасні тенденції будування сайтів, містить усі необхідні кнопки навігації, а також підказки та повідомлення, які дають користувачу розуміння, що потрібно вводити у те чи інше поле, та чому щось пішло не так.

ВИСНОВКИ

В ході виконання дипломної роботи був здійснений аналіз сфери інтересу згідно до обраної тематики, а саме «Розробка web-сайту для готельного бізнесу на базі Java-технологій», та інструментів розробки, необхідних при створенні практичної частини роботи.

На основі проведеного аналізу був визначений стек технологій для розробки, а саме:

- мова програмування Java;
- фреймворк Spring, а саме його частина Spring Boot;
- модуль Spring Security;
- база даних у форматі SQL;
- візуальний фреймворк Bootstrap;
- обробник шаблонів FreeMarker.

Була розроблена концепція проекту, сформовані вимоги до нього. Після визначення з основною логікою проекту та його архітектурою була почата розробка програми. Проект заснований на схемі MVC, в ньому використано багато сучасних технологій, які спрощують розробку та роблять інтерфейс сайту більш функціональним або візуально приємним.

Отриманий продукт реалізує повний цикл створення та оплати букінгу з оглядкою на наявність номерів, а також подальшу обробку інформації адміністратором. Веб-сайт має можливість реєстрації та логіну з перевіркою введених даних, а також різні рівні доступу для авторизованих чи неавторизованих користувачів, а також для користувачів з роллю адміністратора.

Інтерфейс сайту вийшов функціональним, інтуїтивно зрозумілим та візуально привабливим. На сайті реалізований сучасний підхід до дизайну, що робить цей веб-сайт схожим на найкращих представників готельного бізнесу.

Створений проект легко масштабується, в нього можна без проблем інтегрувати додаткові технології. Серед ідей для розширення:

– Додати механізм автоматичного видалення букінгу після певного проміжку часу.

– Надати можливість користувачу редагувати вносити зміни у букінг перед його оплатою.

– Розширити спектр умов при створенні номеру (локація, готель, рівень комфорту у номері тощо).

– Додати можливість авторизуватися за допомогою сторонніх сервісів (Google, Facebook, Telegram та інших).

В ході роботи над проектом були значно поширені знання в таких областях, як:

1) робота з мовою програмування Java;
2) використання та конфігурація фреймворку Spring та окремих його частин;
3) продумування логіки та (elib.hduht.edu.ua) створення бази даних на основі технології SQL;
4) створення дизайну сайту за допомогою сучасних інструментів, таких, як Bootstrap, jQuery та інших.

Здобуті в процесі виконання дипломного проекту знання та навички виступають гарним фундаментом для створення наступних робіт та будуть корисними в подальшому освоєнні професії Інженера програмного забезпечення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Marriott [Електронний ресурс] – Режим доступу до ресурсу: https://www.marriott.com/default.mi
2. Burj Al Arab [Електронний ресурс] – Режим доступу до ресурсу: https://www.jumeirah.com/en/stay/dubai/burj-al-arab-jumeirah
3. Four Seasons [Електронний ресурс] – Режим доступу до ресурсу: https://www.fourseasons.com/
4. Ritz Carlton [Електронний ресурс] – Режим доступу до ресурсу: https://www.ritzcarlton.com/
5. Trivago [Електронний ресурс] – Режим доступу до ресурсу: https://www.trivago.com/
6. Rixos [Електронний ресурс] – Режим доступу до ресурсу: https://www.rixos.com/en
7. Ethan Marcotte – Responsive Web Design
8. Andreas Meier, Michael Kaufmann – SQL and NoSQL Databases
9. Craig Walls – Spring In Action (5^{th} edition)
10. Craig Walls – Spring Boot In Action
11. Joshua Bloch – Effective Java