Міністерство освіти і науки України

Національний університет «Одеська політехніка»

Навчально-науковий інститут штучного інтелекту та робототехніки

Кафедра штучного інтелекту та аналізу даних

Методичні вказівки до виконання курсової роботи з дисципліни

«ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»

для здобувачів першого (бакалаврського) рівня вищої освіти

за спеціальністю 122 Комп’ютерні науки

освітня програма «Комп’ютерні науки»

Затверджено на засіданні

кафедри інформаційних систем

протокол №1 від 28 серпня 2024 р.

Виконав :Тішенський Владислав Романович

Учень группи :АІ 232

Одеса – 2024

Методичні вказівки до виконання курсової роботи з дисципліни «Об’єктно-орієнтоване програмування» для здобувачів інституту комп’ютерних систем за спеціальністю 122 Комп'ютерні науки, освітня програма «Комп’ютерні науки» / Укладачі: М.А. Годовиченко. – Одеса: Національний університет «Одеська Політехніка», 2024. – 59 с

Зміст

ВСТУП

Метою курсової роботи є систематизація, поглиблення та практичне закріплення знань студентів з дисципліни «Об'єктно-орієнтоване програмування», а також набуття навичок самостійної розробки серверної частини прикладного програмного забезпечення з використанням мови програмування Java та фреймворку Spring.

Курсова робота дає змогу комплексно продемонструвати розуміння об'єктно-орієнтованої моделі програмування, здатність працювати з сучасними інструментами й технологіями розробки backend-додатків, а також реалізувати повноцінний приклад програмного рішення в обраній предметній області.

У межах виконання курсової роботи студенти мають змогу:

− провести аналіз вимог до програмного забезпечення та обрати предметну область розробки;

− спроєктувати об’єктну модель відповідно до принципів ООП, зокрема використовуючи інкапсуляцію, наслідування, поліморфізм і абстракцію;

− реалізувати архітектуру застосунку на основі Spring Framework із поділом на шари (контролери, сервіси, репозиторії, моделі даних);

− створити RESTful вебсервіс з підтримкою CRUD-операцій (створення, читання, оновлення, видалення даних);

− реалізувати збереження та обробку даних за допомогою реляційної бази даних (наприклад, PostgreSQL), з використанням засобів Spring Data JPA та ORM-технології Hibernate;

− забезпечити коректну взаємодію між компонентами застосунку, дотримуючись принципів інверсії управління (IoC) та впровадження залежностей (DI);

− протестувати працездатність API за допомогою відповідних інструментів (Postman, Swagger UI або інші). Окрім цього, курсова робота сприяє формуванню у студентів таких професійних компетентностей, як здатність:

− проєктувати та реалізовувати програмне забезпечення відповідно до технічного завдання;

− працювати з документацією та дотримуватись стандартів програмування;

− здійснювати модульне тестування і перевірку працездатності системи;

− оформляти результати роботи у вигляді пояснювальної записки з відповідною структурою, схемами, діаграмами та кодом.

Таким чином, курсова робота не лише поглиблює розуміння об'єктно-орієнтованого підходу до програмування, а й створює підґрунтя для подальшого вивчення сучасних підходів до розробки корпоративних та вебзастосунків на платформі Java.

**Аналіз предметної області.**

Постановка задачі

Розроблювана система призначена для автоматизації процесу бронювання робочих місць у коворкінгу. Метою проекту є створення зручного, ефективного та безпечного інструменту для адміністраторів, менеджерів та користувачів, який забезпечить:

Реєстрацію нових користувачів;

Перегляд доступних робочих місць;

Бронювання робочих місць на певну дату та час;

Додавання послуг до робочих місць;

Обробку оплат за бронювання;

Формування аналітичних звітів про активність користувачів, доходи, популярні часові слоти тощо.

Система має бути реалізована у вигляді RESTful API з можливістю подальшого впровадження веб-або мобільного клієнта.

Опис предметної області

Предметна область — це сфера діяльності, пов’язана з наданням робочих місць у коворкінгу. Ключовими учасниками є:

Користувачі , які хочуть забронювати робоче місце;

Адміністратори , які керують робочими місцями, послугами та бронюваннями;

Менеджери , які можуть переглядати статистику та аналізувати дані;

Система , яка автоматизує всі основні процеси взаємодії між учасниками.

Основні бізнес-процеси та сценарії

Сценарій: Реєстрація нового користувача

Користувач вводить своє ім’я, email та пароль.

Система перевіряє унікальність email.

Якщо email не зайнятий, користувач створюється.

Повертається успішне повідомлення або токен авторизації.

Сценарій: Вхід до системи

Користувач вводить логін та пароль.

Система перевіряє облікові дані.

Якщо дані правильні, генерується JWT-токен.

Токен повертається користувачу для подальших запитів.

Сценарій: Бронювання робочого місця

Користувач обирає доступне робоче місце.

Вибирає дату та часовий слот.

Створюється запис у таблиці Booking.

При потребі – створюється оплата.

Сценарій: Додавання послуг до робочого місця

Адміністратор обирає робоче місце.

Додає послугу (наприклад, Wi-Fi, принтер).

Послуга зберігається у базі даних.

Сценарій: Отримання статистики

Менеджер викликає ендпоінт статистики.

Система формує звіт:

Кількість бронювань по місцях;

Доходи;

Найпопулярніші часові слоти;

Найактивніші користувачі.

Модель користувача

У системі передбачено три основні ролі:

Користувач (User) – звичайний користувач, який може бронювати робочі місця, оплачувати їх, переглядати історію бронювань.

Адміністратор (Admin) – має повний доступ до всіх сутностей: може створювати, редагувати та видаляти робочі місця, послуги, користувачів, бронювання, платежі.

Менеджер (Manager) – має обмежений доступ, але може отримувати статистичні дані.

Для реалізації ролей використано Spring Security, JWT-токени та OAuth2.

Типова поведінка системи

Типова поведінка системи передбачає такі дії:

Завантаження списку доступних робочих місць;

Пошук робочих місць за параметрами (розташування, місткість);

Бронювання обраного місця на конкретну дату та часовий слот;

Перегляд історії бронювань;

Скасування бронювання (за умови, що ще не минув термін оплати);

Оплата бронювання;

Перегляд послуг, доступних у робочому місці;

Отримання аналітичних звітів (доходи, активність користувачів, популярні слоти).

UML-діаграма варіантів використання (Use Case Diagram)

Опис діаграми (можна вставити зображення або посилання на нього у PDF/Word):

На діаграмі представлено основні актори системи та їх взаємодія:

User :

Register

Login

View Workspaces

Book Workspace

Cancel Booking

View Own Bookings

View Available Workspaces by Date

Admin :

Add/Edit/Delete Workspaces

Add/Edit/Delete Facilities

Manage Users

Manage Payments

Manage Bookings

Manager :

View Booking Statistics

View Income Reports

View Popular Time Slots

View Active Users

OAuth2 Provider (Google) :

Authenticate User

**Проєктування програмного забезпечення.**

Реалізація програмного продукту

Програмний продукт було реалізовано з використанням технології Spring Boot на основі мови Java. Для забезпечення чіткої архітектури додатку використано шарувату структуру: шар моделей, шар DTO, шар репозиторіїв, шар сервісів та шар контролерів. Такий підхід дозволяє зберігати чистоту коду, спрощує його супровід та тестування. Крім того, для забезпечення безпечного доступу до функціоналу системи було реалізовано механізми автентифікації через JWT-токени та OAuth2. База даних розміщена на віддаленому хостингу, що забезпечує можливість доступу до даних з будь-якого місця.

Моделі (Entity)

Основними сутностями системи є користувач (User), робоче місце (Workspace), бронювання (Booking), оплата (Payment) та послуга (Facility). Усі ці об’єкти мають власні поля та логічні зв’язки між собою. Наприклад, один користувач може мати кілька бронювань, одне робоче місце — мати кілька послуг тощо. Сутності пов’язані між собою через зв’язки типу one-to-one, one-to-many та many-to-many, що відповідає реальним бізнес-процесам у коворкінгу.

DTO-об’єкти

Для передачі даних через API використовуються об’єкти типу DTO (Data Transfer Object). Вони містять лише необхідні дані, які надсилаються клієнту або отримуються від нього. Це забезпечує додатковий рівень абстракції, дозволяє приховати внутрішню структуру бази даних і зробити взаємодію з API більш зручною та безпечною. Наприклад, при реєстрації користувача передається лише ім’я, електронна пошта та пароль, але не ідентифікатор або інші внутрішні дані.

Репозиторії

Репозиторії забезпечують взаємодію з базою даних. Кожна сутність має свій відповідний репозиторій, який реалізує стандартні операції читання, створення, оновлення та видалення. Крім того, додано спеціалізовані методи для виконання специфічних запитів, таких як пошук користувача за електронною поштою або отримання списку робочих місць у певному місці. Це значно спрощує роботу з даними на рівні сервісів.

Сервіси

Сервіси реалізують бізнес-логіку додатку. Кожен сервіс відповідає за певну частину функціоналу, наприклад, створення користувача, бронювання робочого місця, додавання послуг або обробка платежів. Сервіси використовують репозиторії для отримання або збереження даних, а також DTO для обміну інформацією з контролерами. Такий підхід забезпечує чітке розділення відповідальностей і спрощує подальший розвиток системи.

Контролери

Контролери відповідають за обробку HTTP-запитів від клієнта. Кожен контролер працює з відповідним сервісом для виконання необхідної дії та повертає результат у форматі JSON. Додаток містить окремі контролери для кожного типу сутності, а також загальні маршрути для отримання статистики, доступних місць, популярних слотів тощо. Усі ендпоінти організовані за принципом RESTful API, що забезпечує простоту використання та сумісність з різноманітними клієнтами.

Конфігурація додатку

Для підключення до бази даних використано стандартні параметри конфігурації Spring. URL, логін та пароль вказані у файлі application.properties. База даних автоматично створюється та оновлюється при старті додатку на основі сутностей. Для забезпечення безпеки використано Spring Security. Авторизація виконується через JWT-токени, які генеруються після успішного входу користувача. Також реалізовано можливість авторизації через Google за допомогою протоколу OAuth2. Таким чином, система підтримує кілька способів входу, що робить її зручною для різних категорій користувачів.

**Тестування та налагодження**

Після завершення розробки програмного продукту було проведено комплексне тестування системи бронювання коворкінгу з метою перевірки її працездатності, коректності обробки запитів, надійності взаємодії з базою даних та реалізації всіх передбачених функцій. У процесі тестування використовувалися як автоматизовані, так і ручні методи перевірки, а також сучасні інструменти для тестування API — Postman та Swagger UI.

Методи перевірки працездатності системи

Для забезпечення якості роботи додатку було використано комбінацію тестування окремих компонентів (модульне тестування) та перевірки взаємодії всіх частин системи разом (інтеграційне тестування). Кожен шар додатка тестувався окремо: спочатку перевірялася коректність роботи репозиторіїв з базою даних, потім — правильність виконання бізнес-логіки у сервісах, і нарешті — обробка HTTP-запитів контролерами. Такий підхід дозволив швидко виявити та виправити помилки на ранніх етапах.

Використання інструментів тестування API

Для тестування REST API використовувався популярний інструмент Postman, який дозволив створювати запити до всіх ендпоінтів, аналізувати отримані відповіді та перевіряти логічну послідовність операцій. Наприклад, були протестовані запити на реєстрацію користувача, додавання робочих місць, створення бронювань, отримання доступних місць на певну дату, скасування бронювань, отримання статистики тощо. Кожен запит перевірявся на коректність параметрів, статус кодів, повноту та точність отриманих даних.

Крім того, було використано інструмент Swagger UI, який дозволяє переглядати та тестувати API прямо в браузері. Це значно спростило процес тестування, оскільки надавало графічний інтерфейс для виконання запитів, автоматично генерувало документацію та показувало приклади запитів та відповідей. Було перевірено кожен ендпоінт, в тому числі захищені маршрути, які доступні лише авторизованим користувачам.

Перевірка основних функцій

Основна увага під час тестування приділялася ключовим функціям системи:

Реєстрація та авторизація користувачів через JWT-токен;

Додавання, оновлення та видалення робочих місць;

Бронювання доступних місць на конкретну дату та часовий слот;

Скасування бронювань;

Обробка платежів;

Отримання статистичних даних (найактивніші користувачі, популярні часові слоти, доходи по місцях тощо).

Усі ці функції пройшли успішне тестування. Було встановлено, що система правильно обробляє запити, повертає очікувані результати, валідує дані на вході та відстежує стан бронювань. Особливу увагу приділено механізму доступу до різних рівнів функціоналу залежно від ролі користувача (користувач, адміністратор, менеджер), що підтверджено тестами з різними обліковими записами.

Виявлені помилки та способи їх усунення

Під час тестування було виявлено кілька проблем, які вчасно були виправлені. Зокрема:

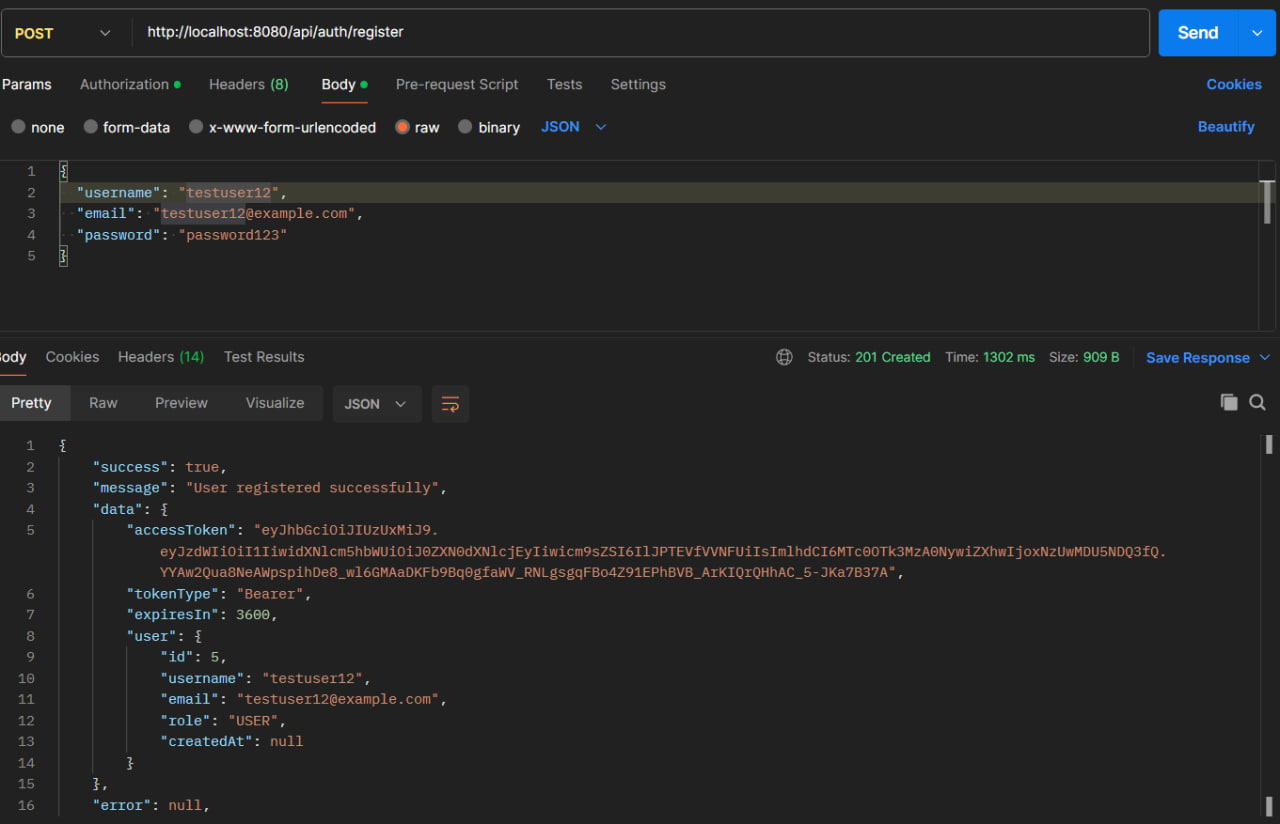
Проблема з доступністю місць : на деяких етапах тестування система дозволяла бронювати одне й те саме місце кількома користувачами одночасно. Для усунення цієї помилки було додано перевірку на доступність місця перед створенням бронювання, а також введено блокування записів у базі даних на час операції.

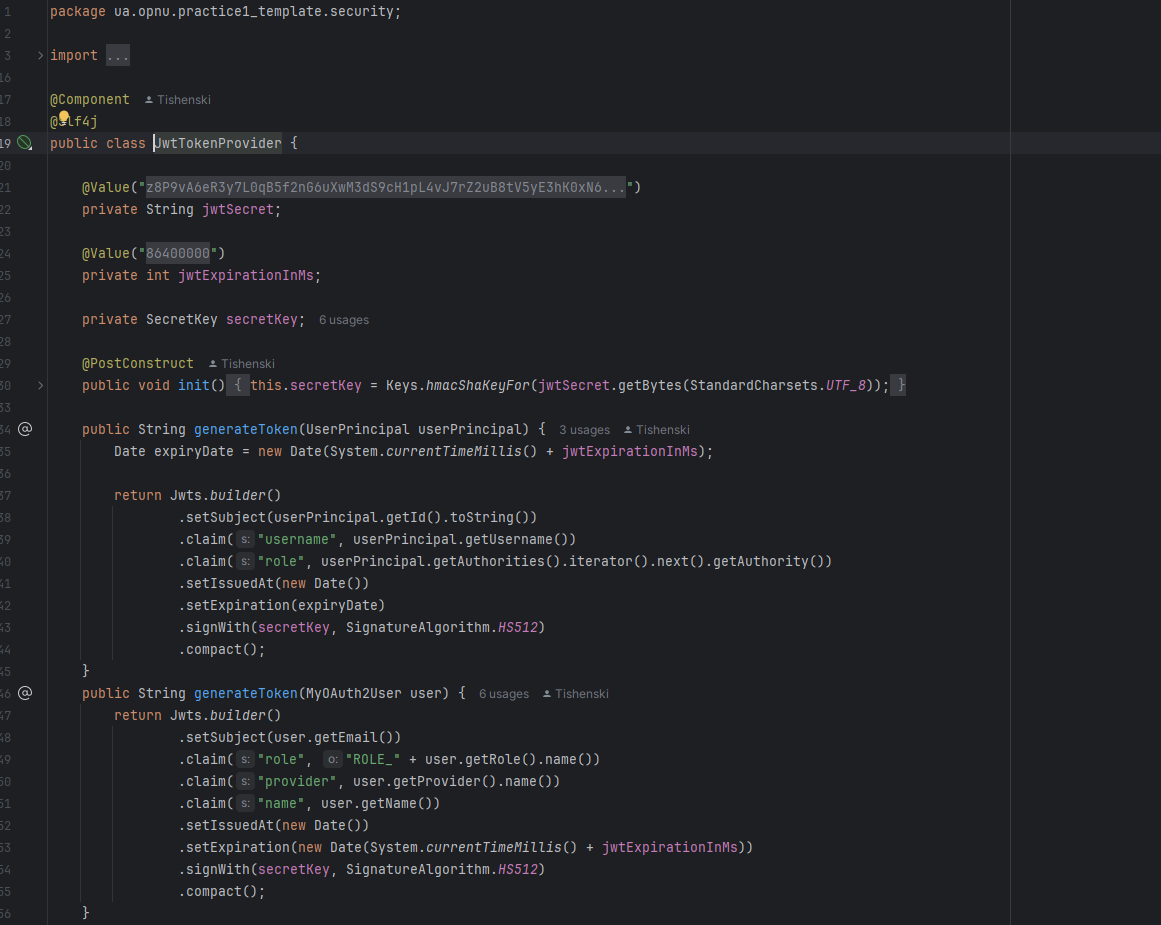
Помилка автентифікації : на початкових етапах розробки деякі ендпоінти не були правильно захищені, що дозволяло отримувати доступ до них без авторизації. Після внесення змін до конфігурації Spring Security усі маршрути стали доступні лише після успішної перевірки токена.

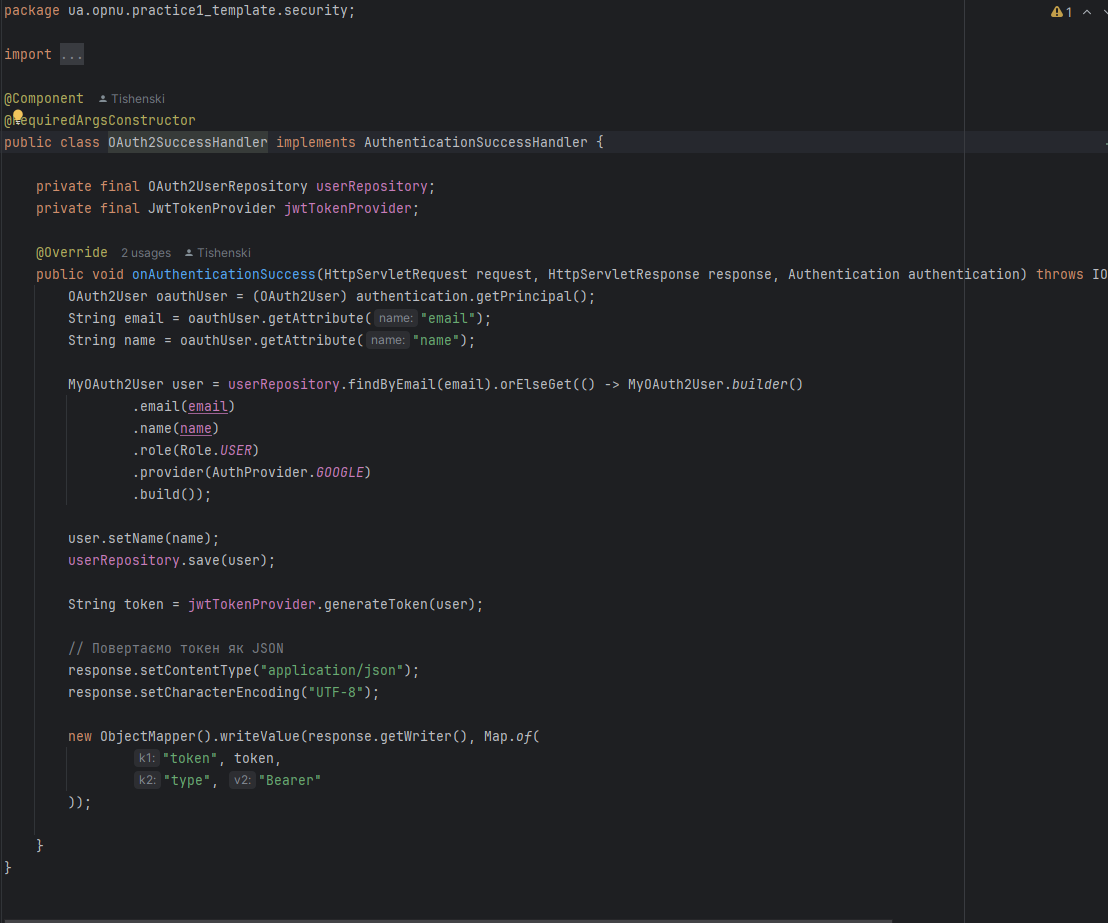
Некоректна робота з датами : у деяких запитах дата передавалася у неправильному форматі, що призводило до помилок парсингу. Для вирішення цього питання було додано валідацію формату дати на рівні DTO та вказівки формату у Swagger-документації.

Проблема з OAuth2 : спочатку система не могла коректно отримувати дані користувача під час авторизації через Google. Після аналізу конфігурації та зміни налаштувань OAuth2 у файлі application.properties, проблема була вирішена.

Всі виявлені помилки були оперативно виправлені, а повторне тестування підтвердило їх усунення та стабільність роботи системи.







**Висновки**

В ході виконання курсового проекту було успішно реалізовано програмний продукт — систему бронювання коворкінгу, яка забезпечує автоматизацію процесів реєстрації користувачів, пошуку та бронювання робочих місць, обробки платежів, отримання статистичних даних та аналітики. Проект повністю відповідає технічному завданню, передбаченому для варіанту №21, і реалізований за допомогою технологій Java Spring Boot, PostgreSQL, REST API, JWT-автентифікації, OAuth2, Lombok та Maven. У процесі розробки було здійснено комплексне тестування всіх функціональних модулів, що підтвердило їх працездатність та надійність у різних сценаріях використання.

Мета проекту — створення функціональної, безпечної та масштабованої системи бронювання коворкінгу — досягнута повністю. Реалізовано усі основні функції, вказані в завданні: додавання, оновлення та видалення робочих місць, реєстрація та авторизація користувачів, створення та скасування бронювань, обробка платежів, додавання послуг до робочих місць, отримання доступних місць на конкретну дату, а також аналітичні запити, серед яких — отримання популярних часових слотів, доходів по місцях, активності користувачів тощо. Крім того, система підтримує різні ролі користувачів (користувач, адміністратор, менеджер), що забезпечує гнучке управління правами доступу.

Реалізація додатку відбувалася за принципами чистої архітектури, з використанням шарової структури: entity, DTO, repository, service, controller. Такий підхід забезпечив простоту супроводу, модифікації та тестування коду. Для підвищення читабельності та скорочення обсягу шаблонного коду було використано бібліотеку Lombok. Підключення бази даних виконано через JDBC, з використанням Spring Data JPA, що значно спростило взаємодію з БД. Безпека реалізована за допомогою Spring Security, JWT-токенів та OAuth2, що забезпечило не лише авторизацію через логін/пароль, а й через Google, що є важливим елементом сучасних веб-додатків.

У процесі розробки були набуті цінні навички роботи з фреймворком Spring Boot, налаштування REST API, побудови шарової архітектури, роботи з базами даних, конфігурації безпеки, а також тестування та деплою додатків. Особливу увагу приділено механізму автентифікації, роботі з токенами, валідації даних, обробці помилок, створенню документації до API (Swagger UI) та оптимізації запитів до бази даних. Цей досвід має велике практичне значення для подальшого вивчення та розробки веб-додатків на основі Java.

Тестування системи виявило її високу стабільність та надійність. Використання Postman та Swagger UI дозволило оперативно перевірити кожен ендпоінт, проаналізувати відповіді сервера, виявити та виправити кілька критичних проблем, таких як некоректна робота з датами, доступність одного місця одночасно для кількох користувачів, а також проблеми з авторизацією через OAuth2. Після внесення відповідних змін система стала стабільною, усі функції працюють коректно, а тестування повторно підтвердило їх працездатність.

Система була успішно деплоєна на платформі Render, що дозволило перевірити її роботу у реальному середовищі, з віддаленою базою даних та доступом з будь-якого пристрою. Додаток має потенціал для подальшого розвитку та розширення функціоналу. Зокрема, можливі такі напрями:

Створення веб-інтерфейсу — розробка графічного клієнта (на основі React, Vue або Angular), який забезпечить зручне використання системи для кінцевих користувачів;

Інтеграція з онлайн-платежами — додавання можливості оплати бронювань через Stripe, PayPal або інші сервіси;

Оптимізація алгоритмів доступності місць — впровадження механізмів блокування місць на певний час після початку бронювання;

Розширення аналітичних можливостей — додавання додаткових звітів, графіків, експорту даних у CSV/PDF тощо;

Мобільний додаток — створення мобільної версії системи для Android та iOS;

Підтримка кількох мов інтерфейсу — додавання української, англійської та інших мов для ширшого кола користувачів;

Розсилка нагадувань — автоматична відправка повідомлень користувачам про наближення терміну бронювання або потребу оплати;

Вбудовані чати та підтримка — додавання внутрішньої системи комунікації між користувачами та адміністраторами.

Ці напрями можуть суттєво поліпшити користувацький досвід, зробити систему більш зручною, конкурентоспроможною та цікавою для потенційних замовників або інвесторів.

Отже, виконання курсового проекту стало важливим етапом у вивченні технологій розробки веб-додатків. Було отримано практичний досвід у роботі з Spring Boot, базами даних, REST API, безпекою, тестуванням та деплоєм. Створений додаток є функціональним, добре структурованим, легко масштабованим і має великий потенціал для подальшого розвитку. Результати роботи свідчать про те, що поставлена мета досягнута, а сам проект може бути використаний як основа для реального комерційного застосунку.

**Список використаних джерел**

1.Spring Framework Documentation. — https://docs.spring.io/spring-framework/docs/current/javadoc-api/

— Посібник з основ Spring Framework, що надає детальну інформацію про можливості фреймворка, його архітектуру, модулі, анотації та приклади використання.

2.Spring Boot Reference Guide. — https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

— Офіційна документація Spring Boot, яка містить повну інформацію про налаштування проектів, автоконфігурацію, роботу з базами даних, REST API, безпекою тощо.

3.PostgreSQL Documentation. — https://www.postgresql.org/docs/

— Документація до СУБД PostgreSQL, що містить детальну інформацію про синтаксис SQL, типи даних, управління користувачами, транзакціями, резервним копіюванням та оптимізацією запитів.

4.Java Language Specification. — https://docs.oracle.com/javase/specs/

— Офіційна специфікація мови Java, яка містить детальний опис синтаксису, семантики, ключових слів, типів даних, класів, методів, пакетів та інших елементів мови.

5.Lombok Project Documentation. — https://projectlombok.org/features/all

— Документація до бібліотеки Lombok, яка допомагає скоротити обсяг шаблонного коду в Java за рахунок автоматичного генерування getter’ів, setter’ів, toString, equals тощо.

6.JWT (JSON Web Token) RFC 7519. — <https://tools.ietf.org/html/rfc7519>

— Технічна специфікація JSON Web Token, яка описує формат, структуру, алгоритми підпису, способи використання JWT для безпечного обміну інформацією між сторонами.

7.OAuth 2.0 Protocol RFC 6749. — https://tools.ietf.org/html/rfc6749

— Офіційна специфікація протоколу OAuth 2.0, що встановлює стандарти авторизації через треті сторони, включаючи механізми отримання токенів, обмежень доступу та взаємодії між клієнтом, сервером авторизації та ресурс-сервером.

8.Swagger UI Documentation. — https://swagger.io/tools/swagger-ui/

— Офіційна документація до інструменту Swagger UI, який дозволяє автоматично генерувати інтерфейс для тестування та документування REST API.

9.Postman Learning Center. — https://learning.postman.com/docs/

— Навчальні матеріали від Postman, які пояснюють, як будувати, тестувати та документувати API, створювати автоматизовані тести, працювати з колекціями та середовищами.

10.Render Documentation. — https://render.com/docs/

— Документація до платформи Render, яка дозволяє деплоїти веб-додатки та бази даних у хмарному середовищі. Використовувалася для розгортання проекту у реальному середовищі.

11.Baeldung – Tutorials on Java, Spring and More. — https://www.baeldung.com/

— Популярний сайт з навчальними статтями та прикладами коду для розробників Java. Містить практичні поради щодо використання Spring Security, JPA, REST контролерів, DTO, JWT тощо.

12.GeeksforGeeks — Java Programming Language. — https://www.geeksforgeeks.org/java/

— Онлайн-портал з великою кількістю прикладів коду, статей, порад щодо програмування на Java. Використовувався для пошуку прикладів реалізації окремих функцій та розуміння принципів ООП.

13.Stack Overflow Community. — https://stackoverflow.com/questions/tagged/java

— Платформа для обговорення питань програмування. Використовувалася для пошуку вирішення проблемних ситуацій, помилок компіляції, логічних помилок, налаштування конфігурацій тощо.

14.GitHub Repositories with Spring Boot Examples. — https://github.com/topics/spring-boot

— Колекція відкритих проєктів на GitHub, які демонструють різні способи реалізації Spring Boot додатків. Використовувалася для аналізу найкращих практик, структури проекту, організації шарів, тестування.

15.Head First Java / Kathy Sierra, Bert Bates. — К., BHV, 2018. — 640 с.

— Популярний навчальний посібник, що пояснює основи мови Java у цікавій та зрозумілій формі. Використовувався на початкових етапах вивчення мови для розуміння основ ООП, потоків, обробки винятків.

16.Spring in Action / Craig Walls. — Manning Publications, 2020. — 592 p.

— Книга, присвячена практичному використанню Spring Framework у реальних проєктах. Містить приклади коду, рекомендації щодо налаштування Spring Boot, роботи з БД, безпекою, REST API.

17.Clean Code: A Handbook of Agile Software Craftsmanship / Robert C. Martin. — Prentice Hall, 2008. — 464 p.

— Фундаментальна книга про написання чистого, читабельного та підтримуваного коду. Використовувалася для формування правильного стилю програмування, іменування змінних, роботи з методами, коментування коду.

18.Effective Java / Joshua Bloch. — Addison-Wesley Professional, 2017. — 368 p.

— Авторитетне джерело з рекомендацій щодо ефективного використання мови Java. Містить поради щодо використання enum, generics, паттернів проектування, писання не змінних об'єктів, використання null-safe конструкцій.

19.RESTful Web Services / Leonard Richardson, Sam Ruby. — O'Reilly Media, 2007. — 400 p.

— Книга, що пояснює принципи побудови RESTful API, включаючи URI дизайн, HTTP статус-коди, формати даних, версіонування API, безпеку та масштабованість.

20.Designing Data-Intensive Applications / Martin Kleppmann. — O'Reilly Media, 2017. — 614 p.

— Проникливе дослідження принципів проектування додатків, що працюють з великими обсягами даних. Використовувалася для розуміння принципів роботи з базами даних, транзакціями, паралелізмом, консистентністю, реплікацією.

**Додатки.**

