МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
Аліна САВЧЕНКО
"___"_____2022 р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

"БАКАЛАВРА"

ЗА СПЕЦІАЛЬНІСТЮ 122 «КОМП'ЮТЕРНІ НАУКИ»

Тема: " Back-end частина Web-сервісу надання перукарських послуг"

Виконавиця: Лебеденко Ганна Юріївна

Керівник: к.т.н., доцент Куклінський Максим Володимирович

Нормоконтролер: Олександр ШЕВЧЕНКО
(підпис)

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

Освітній ступінь: "Бакалавр"

Галузь знань, спеціальність, освітньо-професійна програма:

12 "Інформаційні технології, 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології" .

ЗАТВЕРДЖУЮ

Завідувач кафедри

Аліна САВЧЕНКО

"____" _____ 2022 р.

ЗАВДАННЯ

на виконання дипломного проекту студентки

Лебеденко Ганни Юріївни

(прізвище, ім'я, по батькові)

1. **Тема роботи:** «Back-end частина Web-сервісу надання перукарських послуг» затверджена наказом ректора №454/ст від 29.04.2022
2. **Термін виконання роботи:** 10.05.2022 – 13.06.2022.
3. **Вихідні дані до роботи:** дані про веб-застосунки, документація, методи, засоби та техніки тестування
4. **Зміст пояснювальної записки (перелік питань, що підлягають розробці):** вступ, аналіз загальнотеоретичних питань веб-сервісів, огляд використаних технологій в ході розробки веб-додатку, пояснення розробки веб-сервісу, представлення роботи веб-сервісу.
5. **Перелік обов'язкового графічного матеріалу:** слайди презентації MS PowerPoint.

6. Календарний план-графік

№ з/п	*Завдання*	*Термін виконання*	*Підпис керівника*
1.	Аналіз та дослідження предметної області	10.05.2022р. – 12.05.2022р.	
2	Огляд та аналіз технологій і фреймворків	13.05.2022р. – 20.05.2022р.	
3	Проектування рівнів веб-сервісу. Написання розділу 1 дипломної роботи	21.05.2022р. – 27.05.2022р.	
4	Розробка веб-сервісу з використанням оглянутих технологій. Написання розділу 2 дипломної роботи.	28.05.2022р. – 31.05.2022р.	
5	Написання Розділу 3 дипломної роботи. Завершення створення пояснювальної записки дипломної роботи.	01.06.2022р. – 05.06.2022р.	
6	Оформлення та друк пояснювальної записки.	05.06.2022р. – 07.06.2022р.	
7	Підготовка презентації та доповіді	07.06.2022р. – 13.06.2022р.	

Дата видачі завдання: 10.05.2022 р.

Керівник дипломного проекту _____ Максим КУКЛІНСЬКИЙ

(підпис керівника)

Завдання прийняла до виконання _____ Ганна ЛЕБЕДЕНКО

(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Back-end частина Web-сервісу надання перукарських послуг» представлена на 41 сторінках, містить 26 рисунків, 1 таблицю, 12 наукових джерел.

Мета дипломного проекту: написання повноцінного веб-сервісу з використанням популярних технологій для надання перукарських послуг. Аналіз даної предметної області, визначення вимог, огляд наявних технологій.

Об'єкт дослідження: веб-сервіс надання перукарських послуг.

Предмет дослідження: Back-end розробка.

Метод дослідження: аналіз структурі веб-сервісу, аналіз існуючих технологій для розробки

Результат проекту: розроблений веб-сервіс, який надає функціонал для сфери послуг.

ВЕБ-СЕРВІС, BACK-END РОЗРОБКА, SPRING FRAMEWORK, JAVA

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Мапер (Mapper) – підсистема, яка відповідає за те, щоб взяти якийсь об'єкт і перетворити (скопіювати його значення) його на інший.

POJO (Plain Old Java Object) – це простий Java-об'єкт, не успадкований від якогось специфічного об'єкта і який не реалізує жодних службових інтерфейсів окрім тих, які потрібні для бізнес-моделі..

DTO (Data Transfer object) – один із шаблонів проектування, який використовують для передачі даних між підсистемами програми.

DAO (Data Access Object) – об'єкт що надає абстрактний інтерфейс до деяких видів баз даних , реалізуючи певні операції без розкриття деталей бази даних.

Бін (Bean) – це об'єкти, які є основою програми та управляються Spring IoC контейнером.

DI (Dependency injection) – це процес надання програмному компоненту зовнішньої залежності.

ВСТУП

Інтернет — найскладніша з будь-коли створених систем, у якій мільйони незалежних програмних та апаратних компонентів взаємодіють на різних рівнях. Це масштабування стало можливим завдяки геніальним та простим архітектурним принципам та інженерним рішенням.

Коли в 1990-х роках Всесвітня павутина (WWW) увірвалася в життя людей, люди раптово отримали в свої руки величезну кількість інформації. Існує безліч типів програм, які можна розглядати як веб-служби, але взаємодія між програмами більшою мірою покращується за рахунок використання знайомих технологій, таких як XML та HTTP. Ці технології дозволяють програмам, що використовують різні мови та платформи, взаємодіяти знайомим чином.

Всесвітню мережу можна розглядати як єдину інформаційну систему, яка є найважливішим результатом розвитку інтернет-технологій. Користувачі WWW безпосередньо стикаються з роботою таких програмних компонентів, як браузери, веб-сервери, пошукові системи. Всі ці програми, розвиваються самостійно, проте постійно набувають нових можливостей, не втрачаючи сумісності одна з одною.

Сьогодні є безліч сервісів – від пошуку інформації та покупки повсякденних речей до супутникової навігації логістичних компаній. Одна з найпопулярніших сфер – сфера послуг. Сфера послуг та технології прийшли до успішного взаємозв’язку, який дозволяє покращити робочий процес та має ряд переваг як для клієнтів, так і для робітників. Розробка для сфери послуг наразі - дуже популярна та високо затребувана. Веб-сервіси спрощують онлайн-запис, надають всю потрібну інформацію, дозволяють залишити відгуки тощо.

РОЗДІЛ 1
ЗАГАЛЬНОТЕОРЕТИЧНІ ПИТАННЯ ВЕБ-СЕРВІСІВ

1.1. Поняття веб-сервісу

Веб-сервіс — це набір відкритих протоколів і стандартів, які дозволяють обмінюватися даними між різними програмами або системами. Веб-служби можуть використовуватися програмами, написаними різними мовами програмування та запущеними на різних платформах.

Будь-яке програмне забезпечення, додаток або хмарна технологія, що використовує стандартизовані веб-протоколи (HTTP або HTTPS) для підключення, взаємодії та обміну повідомленнями даних (зазвичай XML (Extensible Markup Language)) через Інтернет, вважається веб-сервісом.

1.2. Переваги веб-сервісів

Одна з переваг – програми, розроблені на різних мовах, можуть з’єднуватися одна з одною шляхом обміну даними через веб-сервіс між клієнтами і серверами. Клієнт викликає веб-сервіс, надсилаючи запит XML, на який сервер відповідає XML-відповіддю.

Наступна перевага - веб-сервіс не вимагає установки на комп’ютер і може працювати під будь-якою операційною системою (Windows, MacOS, Linux). Все, що необхідно -встановлений браузер останніх версій.

Бази даних, створені в онлайн-системі, зберігаються на сервері. Завдяки резервному копіюванню він надійно захищений від втрати даних, а зашифрований канал передачі гарантує конфіденційність цих даних і захист від перехоплення.

Веб-сервер обслуговується професіоналами, доступ до нього суворо регламентований, тому не доводиться нести витрати, пов’язані з резервним копіюванням даних, обслуговуванням та налаштуванням обладнання. Все відбувається автоматично, тож не треба слідкувати за оновленням програми.

Оскільки всі складні обчислення відбуваються на стороні сервера - веб-сервіс стає менш вимогливим до комп'ютерних ресурсів, ніж «настільний» додаток.

Єдина обов’язкова умова для роботи з онлайн-системою – наявність стабільного та швидкого інтернету.

1.3. Технології веб-сервісів

Веб-сервіси засновані на XML і трьох інших основних технологій: WSDL, SOAP і UDDI. Перед створенням веб-служб її розробники створюють її визначення у формі документа WSDL, який описує розташування сервісу в Інтернеті та функціональні можливості, надані цим сервісом. Інформація про сервіс згодом може бути введена в реєстрі UDDI, що дозволяє користувачам веб-служби шукати та знаходити потрібні їм сервіси. Спираючись на інформацію в реєстрі UDDI, розробник клієнтських веб-служб використовує інструкції в WSDL для створення SOAP повідомлень задля обміну даними з сервісом за протоколом HTTP. Нижче описано популярний тип протоколу та архітектурне рішення.

1.2.1. SOAP (Simple Object Access Protocol)

SOAP — це протокол W3C на основі XML для обміну даними через HTTP. Він забезпечує простий, заснований на стандартах метод для надсилання XML-повідомлень між додатками. Веб-сервіси використовують SOAP для надсилання повідомлень між службою та її клієнтами. Оскільки HTTP підтримується всіма веб-серверами та браузерами, повідомлення SOAP можна надсилати між програмами незалежно від їхньої платформи чи мови програмування.

Дані передаються між клієнтом і веб-сервісом за допомогою SOAP-повідомлень запитів і відповідей, формат яких зазначено у визначенні WSDL. Оскільки клієнт і сервер під час створення повідомлень SOAP дотримуються контракту WSDL, ці повідомлення гарантовано будуть сумісні.

Завдяки використанню протоколу SOAP сервіс-орієнтована архітектура є універсальним способом організації взаємодії сервісів, оскільки під час передачі повідомлень об'єкти передаються в універсальному стандартизованому форматі XML. Однак внаслідок цього значно збільшується розмір повідомлення та знижується швидкість його обробки.

1.2.2. REST(representational state transfer)

REST — це набір архітектурних принципів, налаштованих на потреби веб-сервісів і мобільних додатків. Оскільки це набір рекомендацій, реалізація цих рекомендацій залишається на стороні розробників.

Коли запит даних надсилається до REST API, це зазвичай виконується через протокол передачі гіпертексту (званий HTTP). Після отримання запиту API, розроблені для REST (так звані RESTful API або веб-сервіси RESTful), можуть повертати повідомлення в різних форматах: HTML, XML, звичайний текст і JSON.

JSON (JavaScript Object Notation) є перевагою як формат повідомлення, оскільки його можна прочитати будь-якою мовою програмування (незважаючи на назву), він може бути легко прочитаний як людиною так і комп’ютером. RESTful API є доволі гнучким і його легко налаштовувати.

Програма називається RESTful, якщо вона відповідає 6 архітектурним рекомендаціям, таким як:

1. Архітектура клієнт-сервер, що складається з клієнтів, серверів і ресурсів.

2. Стан сеансу зберігається на стороні клієнта, тобто вміст контенту клієнта не зберігається на сервері між запитами.

3. Кешування даних усуває потреби в деяких взаємодіях між клієнт-сервером.

4. Єдиний інтерфейс між компонентами, щоб інформація передавалася в стандартизованій формі, а не з відповідністю до потреб програми. Це описує Рой Філдінг, розробник REST, як «головну особливість, яка відрізняє архітектурний стиль REST від інших мережевих стилів».

Існує набір операцій, з допомогою яких цими ресурсами можна управляти. Цей набір обмежений методами HTTP. У таблиці 1.1. представлені основні методи HTTP, використовувані під час реалізації REST-архітектури, зі своїми коротким цільовим призначенням.

Таблиця 1.1.

Метод	Опис
OPTIONS	Використовується для запиту про підтримувані методи, адреси, а також додаткову інформацію
GET	Використовується для отримання та читання даних з ресурсу
HEAD	Використовується для отримання метаданних стану ресурсу
PUT	Створює або заміняє зміст ресурсу
POST	Додає вміст у ресурс

1.3.1. Переваги REST перед SOAP

REST пропонує низку переваг у порівнянні з SOAP:

• REST пропонує більшу різноманітність форматів даних, тоді як SOAP має лише XML.

• У поєднанні з JSON (який зазвичай краще працює з даними та пропонує швидший аналіз) REST зазвичай вважається більш простим для роботи.

• Завдяки JSON REST пропонує кращу підтримку браузерних клієнтів.

• REST забезпечує чудову продуктивність, особливо завдяки кешування незмінної та статичної інформації.

- REST це протокол, який найчастіше використовується у великих сервісів, таких як Yahoo, Ebay, Amazon і навіть Google.
- REST зазвичай швидше. Крім того, його простіше інтегрувати з існуючими веб-сайтами без необхідності рефакторингу інфраструктури сайту. Це дозволяє розробникам працювати швидше, а не витрачати час на переписування сайту з нуля. Натомість вони можуть просто додати вже існуючі додаткові функції та сервіси до своєї інфраструктури.

РОЗДІЛ 2
ЗАСОБИ РЕАЛІЗАЦІЇ ВЕБ-СЕРВІСУ

2.1. Мова програмування Java

Мови програмування спростили життя людей. Кожен аспект нашого життя залежить від коду, тому не дивно, що сьогодні програмування є однією з основних навичок, необхідних більшості високооплачуваних робіт. Навички програмування особливо цінні в сегментах IT, аналізу даних, веб-дизайну та інженерії. Судячи з даних PYPL, перше місце посідає Python, на другому ж – Java.

Java була розроблена компанією Sun Microsystems (яка зараз є дочірньою компанією Oracle) у 1995 році. Java - це мова програмування та платформа. Java - це надійна, об'єктно-орієнтована і безпечна мова програмування. Поряд з бізнес-програмами Java широко використовується в мобільній операційній системі Android. З використанням Java створено багато великих веб-додатків та систем, таких як LinkedIn, Amazon, Twitter та інші.

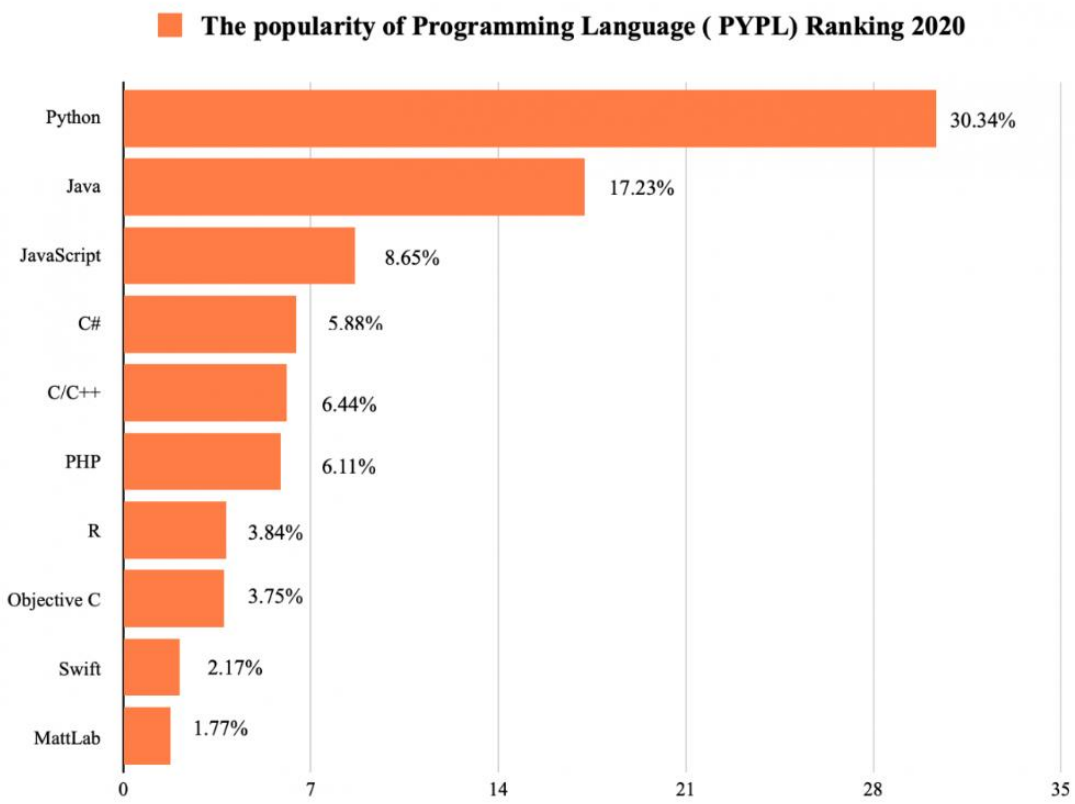


Рис. 2.1. Популярні мови програмування за рейтингом PYPL

Плюси: Java - класичний вибір багатьох бекенд-розробників по всьому світу і єдина офіційна мова для розробки програм для Android. Java розроблений як слабо зв'язана мова програмування, а це означає, що програма, написана на Java, може працювати на будь-якій платформі, що її підтриму. В результаті Java описується як мова програмування «напиши один раз, працюй будь-де». Також для платформи Java створений дуже відомий та зручний фреймворк - Spring Framework, який буде розглянутий далі.

Мінуси: мова програмування Java не є ідеальною для програм, які працюють у хмарних сховищах, на відміну від серверів (що є характерним для бізнес-додатків). Також Java повільніше, ніж деякі інші мови програмування;

У своєму дипломному проекті я вирішила використовувати мову програмування Java, адже я добре знаю цю мову, і не даремно почала її вивчення, відштовхуючись від плюсів цієї мови та її попиту.

2.2. Середовище розробки IntelliJ IDEA

Інтегроване середовище розробки (IDE) - це програмний додаток, який надає програмістам комплексні засоби для розробки програмного забезпечення. IDE зазвичай складається щонайменше з редактора вихідного коду, засобів автоматизації збирання та дебаггера.

IntelliJ IDEA – це спеціальне середовище програмування або інтегроване середовище розробки (IDE), в основному призначене для Java. Він розроблений компанією JetBrains, яка раніше називалася IntelliJ. Він доступний у двох версіях: Community Edition, ліцензованої Apache 2.0, та комерційної версії, відомої як Ultimate Edition. Обидва вони можуть бути використані для створення комерційного програмного забезпечення. Що відрізняє IntelliJ IDEA від своїх аналогів - так це простота використання, гнучкість та надійний дизайн.

Найбільша причина, через яку він вважається одним з найкращих інструментів програмування на основі Java, - це його допоміжні функції, які спрощують його використання та роблять програми, створені за його допомогою, гнучкішими та з можливістю подальшого розширення. Він також має розширені функції перевірки помилок, які дозволяють швидше та простіше рефакторити код.

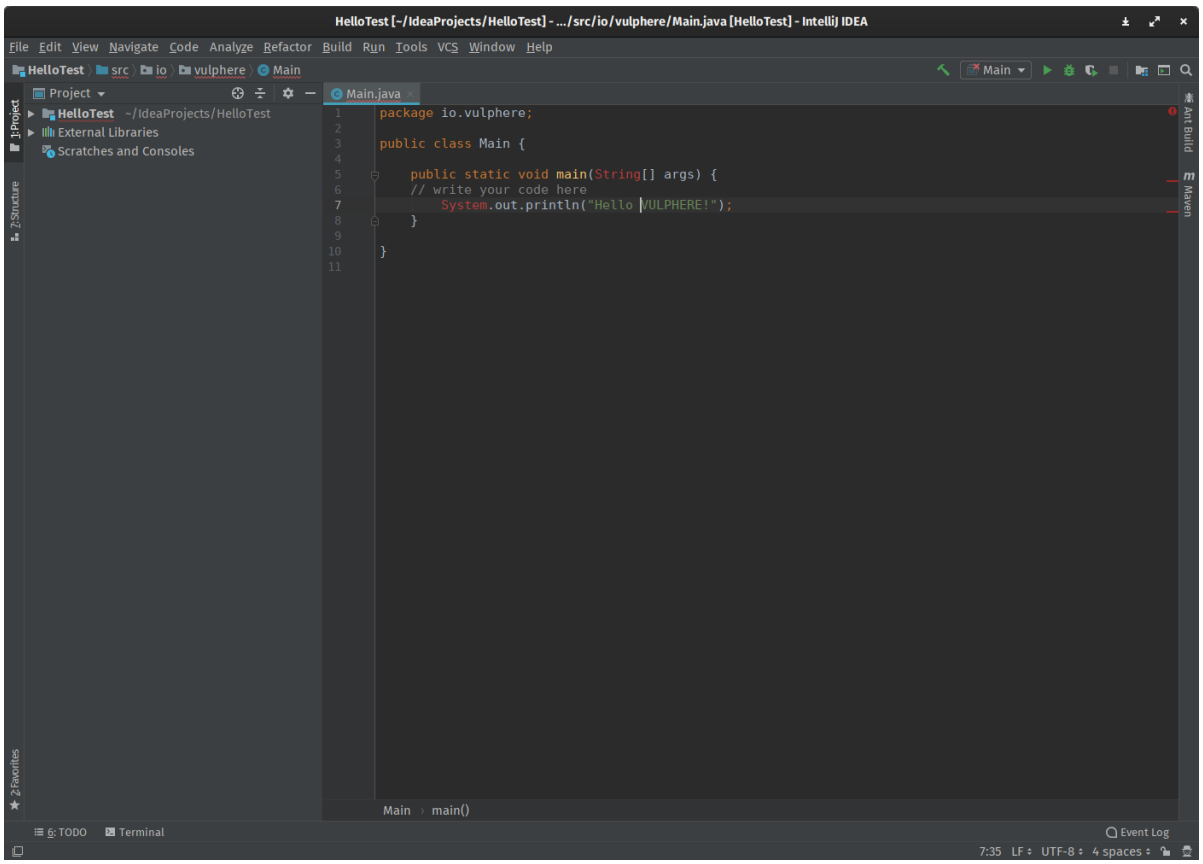


Рис. 2.2. UI вигляд середовища розробки Intellij IDEA

2.3. СУБД PostgreSQL

PostgreSQL – одна з найпопулярніших СУБД, що має багато спільного з MySQL. У недавньому опитуванні вона була обрана другою за популярністю СУБД після MySQL. PostgreSQL - це система управління реляційними базами даних (СУБД), яка створює складніші структури даних шляхом об'єднання певних об'єктів та табличних процедур. Його мета полягає в тому, щоб покращити дотримання та розширюваність стандартів.

Він був розроблений як СУБД із відкритим вихідним кодом. Ця СУБД була створена з використанням мови програмування C та сумісна з Microsoft, iOS, Android, NetBSD, Linux та іншими платформами.

Переваги PostgreSQL

- Покращена масштабованість

PostgreSQL оптимізовано для швидкої та безперервної масштабованості, і ця якість дає йому перевагу перед MySQL. Завдяки своїм високим можливостям масштабування PostgreSQL може бути розгорнутий у великих корпораціях та для проектів із високими вимогами.

- Його повністю відкритий вихідний код

PostgreSQL, можливо, має найкращу та найактивнішу ідеологію відкритого вихідного коду. Він пропонує організаційну ефективність та безмежні можливості зростання. Користувачі PostgreSQL також можуть брати активну участь у житті спільноти, публікуючи повідомлення про помилки та проблеми та ділитись ними.

- Широка підтримка даних користувача

Загальновідомо, що PostgreSQL добре підтримує модулі даних noSQL. За замовчуванням, він підтримує широкий спектр типів даних, включаючи JSON, XML, H-Store і т.д.

- Ефективна спільна робота з іншими інструментами

Існує широкий спектр інструментів управління базами даних, що доповнюють СУБД. На жаль, багато СУБД мають обмежені можливості інтеграції. PostgreSQL вирішує цю проблему, легко та ефективно інтегруючись з багатьма іншими незалежними інструментами.

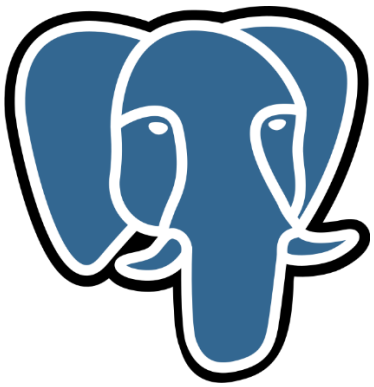


Рис. 2.3. Логотип PostgreSQL

Для зручності зміни схем реляційної бази даних існують міграції.

2.3. Міграції бази даних

Міграції бази даних, також відомі як міграції схеми, міграції схеми бази даних або просто міграції, є контрольовані набори змін, розроблені для зміни структури об'єктів в реляційній базі даних. Міграції допомагають перевести схеми бази даних з їхнього поточного стану в новий бажаний стан, будь то додавання таблиць і стовпців, видалення елементів, розділення полів або зміна типів та обмежень.

Міграції керують додатковими, часто оборотними змінами в структурах даних програмним способом. Програмне забезпечення для міграції баз даних призначене для того, щоб зробити зміни у базі даних відтвореними, загальнодоступними без втрати даних. Як правило, програмне забезпечення міграції створює артефакти, що описують точний набір операцій, необхідних перетворення бази даних з відомого стану в новий стан. Їх можна перевірити та керувати за допомогою звичайного програмного забезпечення для контролю версій, щоб відстежувати зміни та ділитися ними між членами команди.

Хоча запобігання втраті даних, як правило, є однією з цілей програмного забезпечення для міграції, зміни, які видаляють або деструктивно

модифікують структури, в яких зберігаються дані, можуть призвести до видалення. Щоб впоратися з цим, міграція часто є контрольованим процесом, що включає перевірку результуючих сценаріїв змін та внесення будь-яких змін, необхідних для збереження важливої інформації. Одним з таких інструментарієв, які були використані в дипломному проекті є Flyway.

Flyway — інструментарій для супроводу баз даних і синхронізації їхньої структури з пов'язаним із нею програмним забезпеченням. Flyway можна розглядати як аналог системи контролю версій для БД, який виконує завдання автоматизації відображення змін у структурі бази даних для відповідності версії БД і версії програмного забезпечення, що працює з цією БД.

Рис. 2.4. Логотип Flyway

2.4. Spring Framework

Для розробки веб-сервісу зручніше всього використовувати перевірені фреймворки, наприклад, як Spring Framework.

Spring дозволяє створювати веб-програми швидко і без проблем. Видаливши більшу частину стандартного, шаблонного коду та налаштувань, пов'язаних з веб-розробкою - отримали сучасну модель веб-програмування, яка спрощує розробку серверних HTML та REST API – додатків.

Spring - найпопулярніше середовище розробки програм для Java. Мільйони розробників по всьому світу використовують Spring Framework для

створення високопродуктивного, коду, що легко тестується та багатаразово використовується.

Spring Framework – це платформа Java з відкритим вихідним кодом. Спочатку його було написано Родом Джонсоном і вперше випущено під ліцензією Apache 2.0 у червні 2003 року. Spring доволі легкий щодо розміру - базова версія фреймворку Spring важить близько 2 МБ.

Основні функції Spring Framework можна використовувати при розробці будь-якої програми Java, але також є розширення для створення веб-додатків поверх платформи Java EE. Платформа Spring просуває передові методи програмування, використовуючи модель програмування на основі POJO.

2.4.1. Основні технології фреймворку

Spring використовує потужні та корисні технології, які допомагають розробникам швидше, якісніше писати код та тестувати його. Нижче описано деякі з них, які є найголовнішими в Spring, та без яких Spring Framewok не зміг би існувати.

Впровадження залежностей (DI)

Технологія, з якою найбільше асоціюється та представляє собою серце Spring - це впровадження залежностей (DI) , особливість Inversion of Control. Інверсія управління (IoC) - це загальна концепція, яку можна висловити по-різному. Впровадження залежностей — це лише один конкретний приклад інверсії управління.

При написанні складної Java-програми класи та програми повинні бути якомога незалежнішими від інших класів Java, щоб підвищити можливість повторного використання цих класів та їх тестування незалежно від інших класів під час модульного тестування. Впровадження залежностей допомагає склеїти ці класи разом і водночас зберегти їхню незалежність.

Впровадження залежностей може відбуватися шляхом передачі параметрів у конструктор або шляхом пост-конструкції з використанням методів геттерів-сеттерів(методів, що повертають значення та встановлюють його).

Аспектно-орієнтоване програмування (АОП)

Одним із ключових компонентів Spring є інфраструктура аспектно-орієнтованого програмування (АОП). Функції, які охоплюють кілька частин програми, називаються наскрізними завданнями, і ці наскрізні завдання концептуально відокремлені від бізнес-логіки програми. Існують різні приклади аспектів, включаючи ведення журналу, декларативні транзакції, кешування тощо.

Ключовою одиницею модульності ООП є клас, тоді як в АОП одиницею модульності є аспект. DI допомагає відокремити об'єкти програми один від одного, а АОП допомагає відокремити наскрізні завдання від об'єктів, на які вони впливають.

Модуль АОП Spring Framework надає реалізацію аспектно-орієнтованого програмування, що дозволяє визначати перехоплюючі методи та pointcut (предикат, який відповідає точкам з'єднання) для чіткого поділу коду, які реалізують функції на які програма має бути розділена.

2.4.2. Переваги та недоліки використання Spring Framework

Переваги Spring:

- Можливість комплексного використання. Завдяки величезній кількості компонентів та технологій, які підтримує Spring, він є універсальним. Його можна використовувати комплексно, наприклад, для різних частин архітектури MVC або вирішення інших складних завдань. Не потрібно збирати великий стек технологій. Якщо

чогось виявиться недостатньо, допоможуть додаткові інструменти та інше програмне забезпечення фреймворку Spring.

- Полегшення та прискорення роботи. Мета будь-якого фреймворку — робити роботу ефективніше та швидше. Те, на що пішло б кілька місяців чистою мовою, з фреймворком робиться за лічені дні, а в комерційній розробці час має ключове значення. Фреймворк - як каркас, навколо якого пишеться решта коду програми. Деякі компоненти та логіка вже реалізовані, і завдання програміста – у тому, щоб грамотно ними скористатися.
- Масштабність. Завдяки величезній екосистемі можна підібрати рішення для будь-якого завдання. Додаткові модулі, технології та програмне забезпечення від розробників є для більшості актуальних напрямків: веб-розробки, створення мікросервісів, реалізації тієї чи іншої програмної моделі.
- Велика спільнота. Spring затребуваний і необхідний ринку праці. З ним працює багато ентузіастів, тому ви, швидше за все, зможете знайти відповідь на своє запитання на тематичних порталах. У нього велика документація, частина якої перекладена російською мовою. Ентузіасти постійно покращують фреймворк або доповнюють його новими технологіями – у довготривалій перспективі це полегшує програмування.
- Відкритий вихідний код. Spring безкоштовний, а його код відкритий усім охочим. Технології, пов'язані з Java, найчастіше бувають закритими, тому відкритість Spring особливо важлива – вона знижує поріг входу у розробку додатків та полегшує вивчення фреймворку.

Недоліки Spring:

- Довге налаштування. "Чистий" Spring вимагає тривалої конфігурації з нуля для кожного проекту. Це забирає час та сили. Для вирішення проблеми існує Spring Boot. Він спрощує роботу з фреймворком, але може спричинити плутанину через схожість назв.

- Складний старт. Java і пов'язані з нею технології – не найпростіші для старту. Тому новачкам може бути складно розпочати вивчення програмування саме зі Spring та дочірніх інструментів.

2.5. Платформа контейнерізації Docker

Docker — це платформа контейнерізації з відкритим вихідним кодом, використовуваною для розробки, розвертання та управління додатками в полегшених віртуальних середовищах, які називаються контейнерами.

В основному він використовується в якості платформ розробки програмного забезпечення що розроблюють розподілені додатки, які ефективно працюють у різних середовищах. Роблячи програмну систему незалежною, розробникам не потрібно турбуватись про проблеми сумісності. Упаковка додатків в ізольовані засоби (контейнери) також спрощує розробку, розвертання, обслуговування та використання додатків.

Оскільки Docker використовує віртуалізацію для створення контейнерів для зберігання додатків, концепція може відображатися відповідно до віртуальних машин. Хоча вони являють собою ізольовані засоби, використовувані для розробки програмного забезпечення, між віртуальними контейнерами та віртуальними машинами є відмінності. Найважливіше відмінність — це те, що контейнери Docker легше, швидше та більш ефективно використовують ресурси, ніж віртуальні машини.

Контейнери Docker – це легкі віртуалізовані середовища виконання для запуску програм. Кожен контейнер є пакетом програмного забезпечення, який містить код, системні інструменти, середовище виконання, бібліотеки, залежності та файли конфігурації, необхідні для запуску певної програми. Вони незалежні та ізольовані від хоста та інших екземплярів, що працюють на хості.

На одному обладнанні може розміщуватися кілька контейнерів. На відміну від віртуальних машин, контейнери віртуалізуються на рівні додатків.

Тому ділять ядро ОС з хостом і віртуалізують операційну систему поверх нього.

Докер використовується для:

- Запуск кількох навантажень на меншу кількість ресурсів.
- Ізоляція та поділ додатків.
- Стандартизація середовищ для забезпечення узгодженості циклів розробки та випуску.
- Оптимізація життєвого циклу розробки та підтримка робочих процесів CI/CD.
- Розробка переносних навантажень, які можуть працювати на мультихмарних платформах.

Крім того, він використовується як:

- Економічна альтернатива віртуальним машинам.
- Система контролю версій програми.

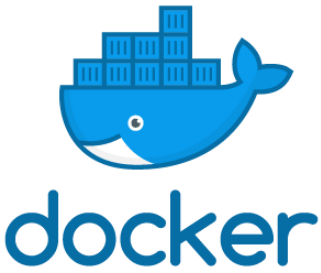


Рис. 2.5. Логотип Docker

РОЗДІЛ 3
РОЗРОБКА ВЕБ-СЕРВІСУ

3.1. Загальний опис системи

Розроблюваний веб сервіс - це повноцінний веб-сайт салону, з використанням популярних технологій, які допомагають автоматизувати управління, спростити написання сервісу.

У розроблюваному сервісів передбачено чотири ролі :

- Адміністратор
- Користувач
- Робітник
- Гість

Гість

Гість – це неавторизований користувач, який має доступ для сторінки авторизації, реєстрації та перегляду наявних послуг/салонів/майстрів. Він не має доступ до бронювання.

Адміністратор

Роль адміністратор – це адміністратор сайту, який має доступ до всього, наприклад, до додавання або видалення робітника, салону, послуг.

Користувач

Користувач – це юзер, який зареєструвався, підтвердив свою пошту та авторизувався. Тобто він міститься в базі даних, а також має доступ до перегляду таких сторінок як: домашня сторінка, вибір салону, вибір перукаря, перелік послуг перукаря, бронювання. Також може змінювати свій профіль, фото профілю тощо.

Робітник

Робітник – це користувач, але з розширеними можливостями. Оскільки робітник працює в певному салоні – він має доступ до планування свого робочого графіку виходячи з графіку салону.

Реєстрація користувача відбувається наступним чином: натискаючи кнопку Sign Up, гість потрапляє на сторінку з формою реєстрації, заповнює її, і якщо введена пошта та нікнейм ще не зареєстровано, на вказану пошту відсилається посилання, яке її підтверджує. Після цього гостя переадресовує на головну сторінку, де він може перейти до авторизації.

3.2. Архітектура системи

В даному проекті використовувалась мова програмування Java, і найзручніший фреймворк Spring. Даний веб-сервіс використовує клієнт сервісну архітектуру

Архітектура клієнт-сервера являє собою мережевий додаток, який розподіляє задачі між клієнтами та серверами, які знаходяться в одній системі або пов’язані з комп’ютерною системою.

Архітектура клієнт-сервер зазвичай включає в себе кілька робочих станцій користувачів, ПК або інших пристроїв, підключених до центрального сервера через Інтернет-з'єднання або іншу мережу. Клієнт відправляє запит даних, сервер приймає та обробляє запит, відправляє пакети даних, тому користувачу, який в них потребує.

В розробленому застосунку зв'язок між клієнтом та сервером відбувається через RestAPI, отримуючи від сервера JSON відповідь.

Для побудування моделі застосунку було обрано Controller-Repository-Service, а в якості стандарту доступу до БД – JDBC, оскільки для мене було важливо самостійне написання та регулювання запитів, а також JDBC вважається більш швидкішим в плані виконання запитів.

Шаблон Controller-Service-Repository переважає в багатьох програмах Spring Boot. Одна з важливих причин, чому мені подобається цей шаблон, полягає в тому, що він відмінно справляється з поділом завдань: рівень контролера у верхній зображення несе виняткову відповідальність за надання функціональності, щоб її могли використовувати зовнішні об'єкти. Шар відповідає за зберігання та вилучення деякого набору даних. На сервісному рівні має бути вся бізнес-логіка. Якщо бізнес-логіка вимагає отримання/збереження даних, вона підключається до репозиторію. Якщо хтось хоче отримати доступ до цієї бізнес-логіки, необхідно звернутись до контролера, щоб до неї дістатися.

Це досить простий поділ завдань. Якщо код пов'язаний із зберіганням/витягом, він має бути поміщений у репозиторій. Якщо йдеться про розкриття функціональності, це відбувається у контролері. Все унікальне в бізнес-лозіці буде на рівні сервісу. Репозиторію байдуже, який компонент його викликає він просто робить те, що його просять. Рівень сервісу не піклується про те, як отримати доступ до нього, він просто виконує свою роботу, використовуючи репозиторій, там, де це необхідно. Контролер зазвичай просто передає роботу сервісному шару, тому він зазвичай виглядає дуже простим і компактним.

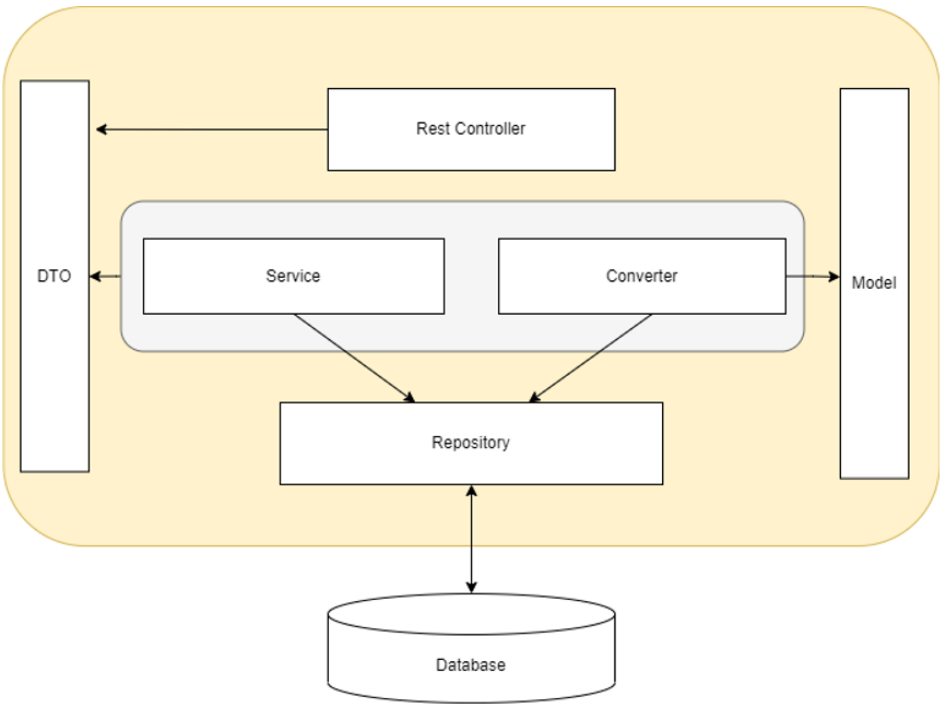


Рис. 3.1. Модель рівнів веб-сервісу

Технології, використані в даному проекті:

- PostgreSql
- Spring Framework
- MailHog
- MinIO
- Thymeleaf

3.3. Проектування та створення бази даних

Для правильного і коректного проектування бази даних необхідно дотримуватися деяких правил, які включають створення таблиць та впровадження відношень між ними, забезпечуючи захист даних та роблячи базу даних більш гнучкою.

Перша нормальна форма полягає в тому, що атрибути відношення були атомарними, тобто їх неможливо поділити на більш прості атрибути, які відповідають іншим властивостям описуваної сутності.

Друга нормальна форма полягає в тому, що відношення знаходиться в першій нормальній формі і всі його не ключові атрибути залежать від первинного ключа.

І третя нормальна форма – відношення знаходиться в другій нормальній формі і всі атрибути, не пов’язані з первинним ключом залежать один від одного.

Після нормалізації бази даних, можна виділити такі основні сутності:

- Салон (hairdressing_salon) з атрибутами *id, name, email, description*
- Користувач (users) з атрибутами *id, username, password, email, salon_id*
- Роль (role) з атрибутом name
- Профіль (profile) з атрибутами *id, name, last_name, sex*
- Послуга (service) з атрибутами *id, type_of_service, price, description*
- Замовлення (order) з атрибутами *id, order_date, description, user_id*
- Розклад (schedule) з атрибутами *id, day, time_from, time_to, hairdressing_salon_id*
- Одиниця часу бронювання (time_slot) з атрибутами *id, date, time_start, time_end, employee_id, order_id*

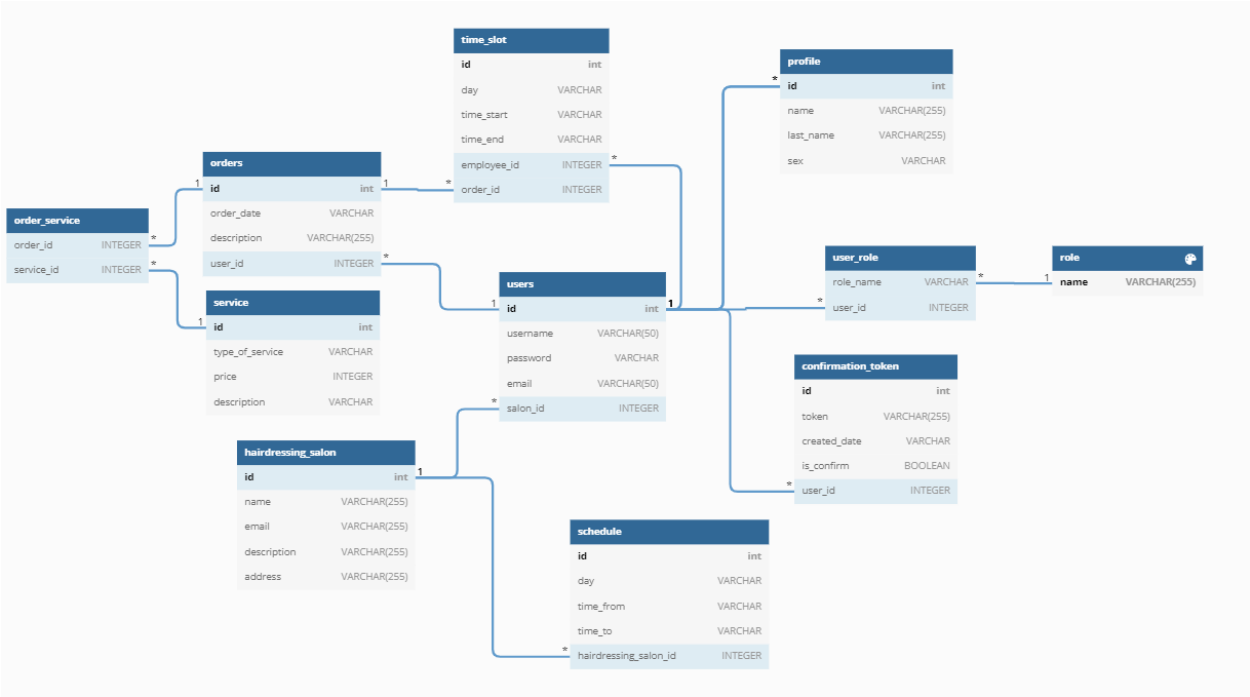


Рис. 3.2. Модель бази даних веб-сервісу

Для, так би мовити, контролю версій бази даних було використано міграції за допомогою інструментарію Flyway. Міграції бувають двух типів : версіонні та ті, що повторюються. Версіонні створюють міграцію лише один раз при піднятті серверу бази даних.

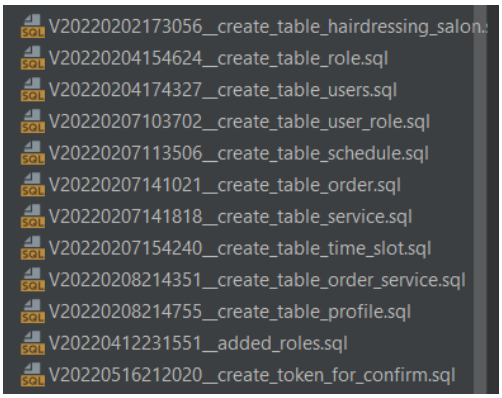


Рис. 3.3. Список міграцій

Кожна з цих міграцій автоматично записується у таблицю flyway_schema_history, де отримує номер checksum, при зміні міграції який

теж змінюється, що допомагає слідкувати за правильністю написання і редагування схеми БД.

3.4. Створення рівнів веб-сервісу

3.4.1. Реалізація моделей

В першу чергу, необхідно визначити сутності, на базі яких буде побудована модель. Ці сутності ми отримали при побудові схеми бази даних. Model зберігає дані (сутності), які використовуються для взаємодії з репозиторієм.

Оскільки використовувалось JDBC, необхідно було самостійно визначити «мапери», які використовуються при виконанні запитів для перетворення результуючого набору даних в об’єкт моделі, для цього створюється клас який імплементує інтерфейс RowMapper. Далі перевизначається метод mapRow , в якому створюється об’єкт необхідної моделі та присвоюються відповідні поля з ResultSet. ResultSet – це клас, який представляє собою результуючий набір даних і забезпечує додатку порядковий доступ до результату запита.

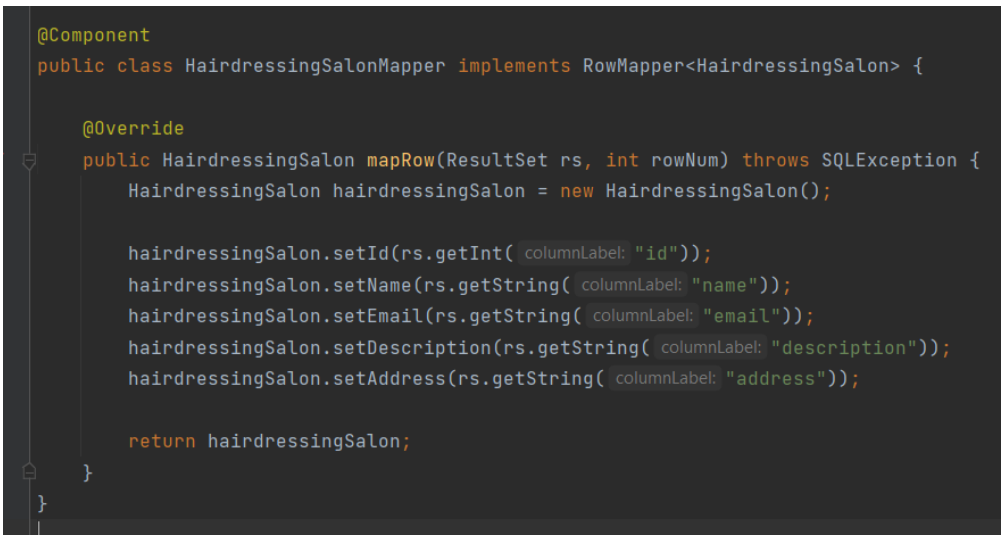


Рис. 3.4. Приклад створення мапера

3.4.2. Створення репозиторію

Далі необхідно побудувати найнижчий рівень додатку – репозиторій. Ціллю репозиторію є прямий доступ до бази даних. Це єдиний рівень, який відповідає за такий зв'язок. Звичайно він містить в собі CRUD функціональність. Для кожної з моделей необхідний свій репозиторій, в якому описуються запити до БД. Спочатку створюється інтерфейс, де визначаються необхідні методи, а потім клас, який ці методи реалізує.

Клас репозиторію позначається анотацією @Repository, яка вказує, що даний оформлений клас є репозиторієм, а також дана анотація являє собою спеціалізацією анотації @Component.

В приведеному скриншоті (рис. 3.5.) описані базові CRUD операції та реалізація одного з методів репозиторію (рис. 3.6.).

Рис. 3.5

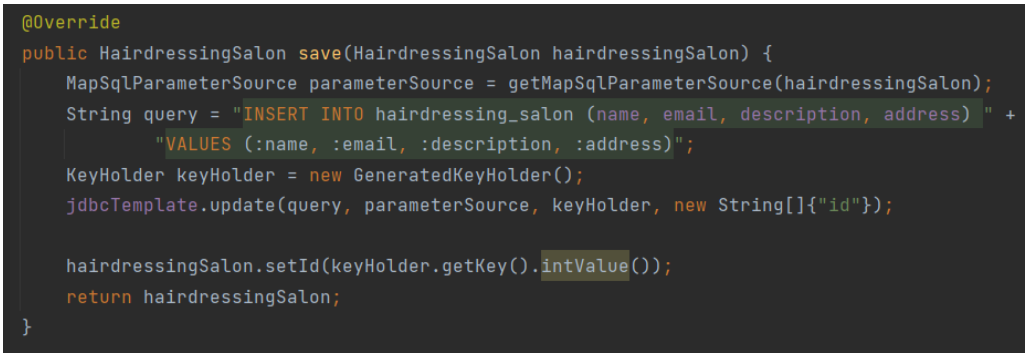


Рис. 3.6

3.4.3. Створення сервісу

Наступним рівнем є service. Цей рівень описує саму бізнес логіку, і призначений для двох цілей : запросити один або декілька репозиторіїв та реалізувати свої власні функції, які корисні , коли вказані функції мають справу з кількома репозиторіями. Оскільки рівень репозиторію призначений тільки для вилучення та збереження елементів в сховище даних, рівень сервісу ідеально призначений для будь-яких потрібних модифікацій. Спочатку створюється інтерфейс, де визначаються необхідні методи, а потім клас, який ці методи реалізує. В прикладі, наведеному нижче, в методі get сервісу салону, викликається метод репозиторію, після чого відбувається фільтрація

результату: якщо салон з необхідним id знайдено – повертається об’єкт салону, якщо ні – викидується помилка.

Клас сервісу позначається анотацією @Service, яка вказує, що даний оформлений клас є саме сервісом, а також дана анотація являє собою спеціалізацією анотації @Component.

```
@Override
public HairdressingSalon get(int id) {
    Optional<HairdressingSalon> salonOptional = hairdressingSalonRepository.findById(id);
    if (salonOptional.isPresent()) {
        return salonOptional.get();
    } else {
        throw new SalonDoesntExistException("salon with id " + id + " dont exist");
    }
}
```

Рис. 3.7. Приклад реалізації одного з метода сервісу.

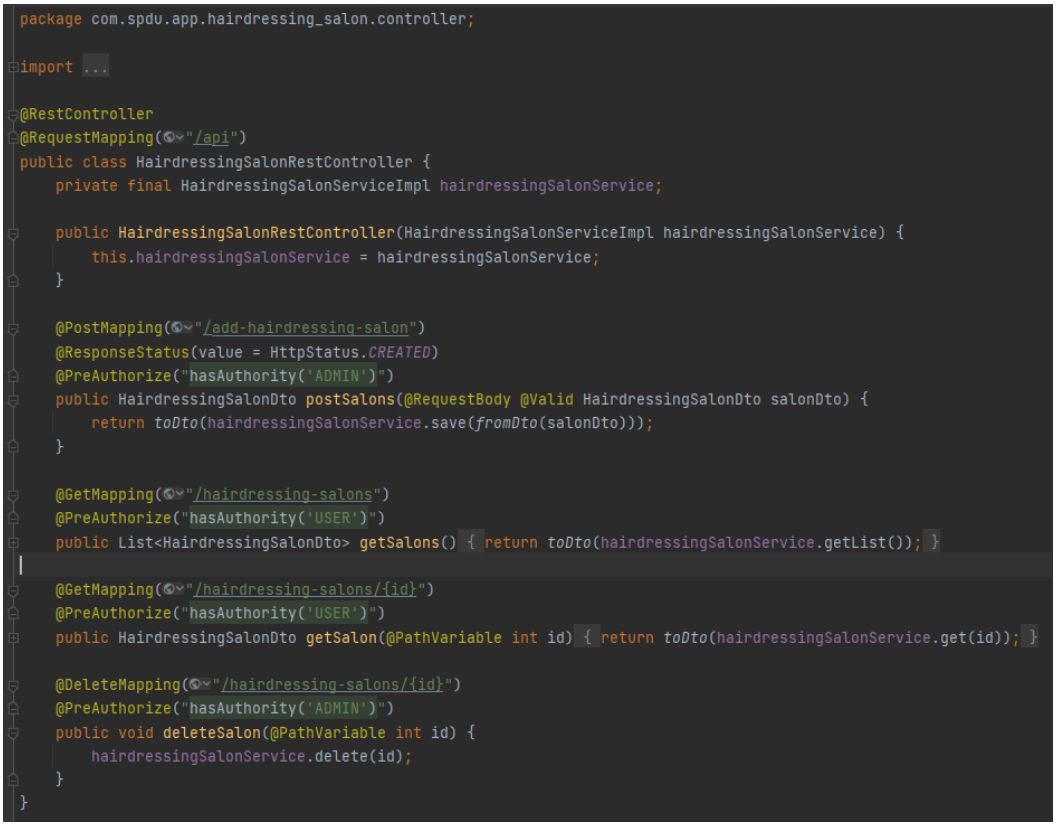
3.4.4. Створення контролерів

Оскільки зв’язок між клієнтом і сервером відбувається за рахунок RestAPI – необхідно створити контролери, енд-поінти яких будуть задовольняти архітектурні принципи RestAPI. В Spring Framework існує анотація @RestController, яка вказує, що сервер буде надсилати файли JSON. Доступ до енд-поінтів (методів контролеру) здійснюється за допомогою HTTP-запитів (GET, POST, DELETE), які вказані за допомогою анотації @GetMapping, @PostMapping, @DeleteMapping. Зазвичай, гарним тоном є застосування паттерна DTO. DTO або об'єкти передачі даних – це об'єкти, які визначають, як саме будуть передаватись дані між веб-додатком.

Основною метою шаблону є скоротити кількість звернень до сервера за рахунок групування декількох параметрів в одному виклику.

Ще однією перевагою є інкапсуляція логіки серіалізації (механізм, який переводить структуру об'єкта та дані у певний формат, який можна зберігати

та передавати). Причина використовувати DTO, полягає в тому, що клієнти не зв'язуються з внутрішніми структурами даних напряму. Це дозволяє вільно змінювати і розвивати внутрішні компоненти, не порушуючи роботу клієнтів.



```
package com.spdu.app.hairdressing_salon.controller;

import ...

@RestController
@RequestMapping("/api")
public class HairdressingSalonRestController {
    private final HairdressingSalonServiceImpl hairdressingSalonService;

    public HairdressingSalonRestController(HairdressingSalonServiceImpl hairdressingSalonService) {
        this.hairdressingSalonService = hairdressingSalonService;
    }

    @PostMapping("/add-hairdressing-salon")
    @ResponseStatus(value = HttpStatus.CREATED)
    @PreAuthorize("hasAuthority('ADMIN')")
    public HairdressingSalonDto postSalons(@RequestBody @Valid HairdressingSalonDto salonDto) {
        return toDto(hairdressingSalonService.save(fromDto(salonDto)));
    }

    @GetMapping("/hairdressing-salons")
    @PreAuthorize("hasAuthority('USER')")
    public List<HairdressingSalonDto> getSalons() { return toDto(hairdressingSalonService.getList()); }

    @GetMapping("/hairdressing-salons/{id}")
    @PreAuthorize("hasAuthority('USER')")
    public HairdressingSalonDto getSalon(@PathVariable int id) { return toDto(hairdressingSalonService.get(id)); }

    @DeleteMapping("/hairdressing-salons/{id}")
    @PreAuthorize("hasAuthority('ADMIN')")
    public void deleteSalon(@PathVariable int id) {
        hairdressingSalonService.delete(id);
    }
}
```

Рис. 3.8. Загальний вигляд контролеру HairdressingSalonController.

3.5. Розробка авторизації

Крім надання різних вбудованих параметрів аутентифікації та авторизації, Spring Security дозволяє налаштовувати процес аутентифікації за потреби розробника. Починаючи з сторінки входу, що настроюється, і закінчуючи власними налаштованими постачальниками аутентифікації та фільтрами аутентифікації, можна значною мірою налаштувати кожен аспект процесу аутентифікації, визначити власний процес аутентифікації, який може змінюватись від простої аутентифікації з використанням імені користувача та пароля до складної, такої як двофакторна аутентифікація з використанням токенів та одноразових паролів. Також можна використовувати різні бази

даних — як реляційні, так і нереляційні, використовувати різні кодувальники паролів.

Spring Security може запобігти доступу неавторизованих або шкідливих запитів до захищених ресурсів сервісу, не дозволяючи їм перейти на сторінку. Таким чином розроблений додаток та ресурси залишаються захищеними.

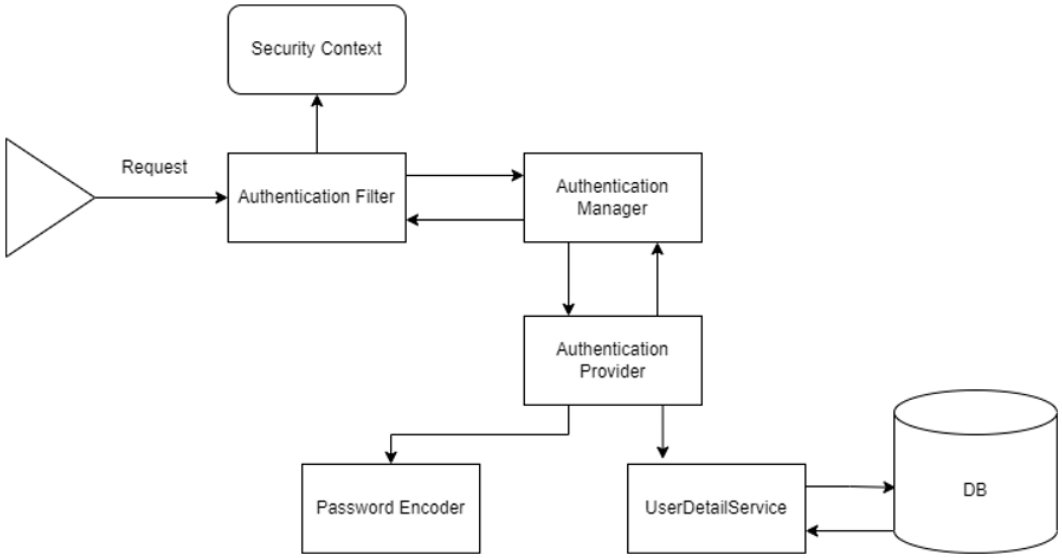


Рис. 3.9. Базові компоненти Spring Security і зв'язок між ними

AuthenticationFilter - це фільтр, який перехоплює запити та намагається їх аутентифікувати. У Spring Security він перетворює запит на об'єкт аутентифікації та делегує аутентифікацію AuthenticationManager.

AuthenticationManager - це основний інтерфейс стратегії автентифікації. Він використовує єдиний метод authentication() для автентифікації запиту. Метод authentication() виконує аутентифікацію та повертає об'єкт аутентифікації у разі успішної аутентифікації або генерує помилку AuthenticationException у разі збою аутентифікації.

У сценарії базової автентифікації пароль, наданий користувачем, звіряється з паролем у базі даних. Якщо вони збігаються один з одним, то автентифікація проходить успішно.

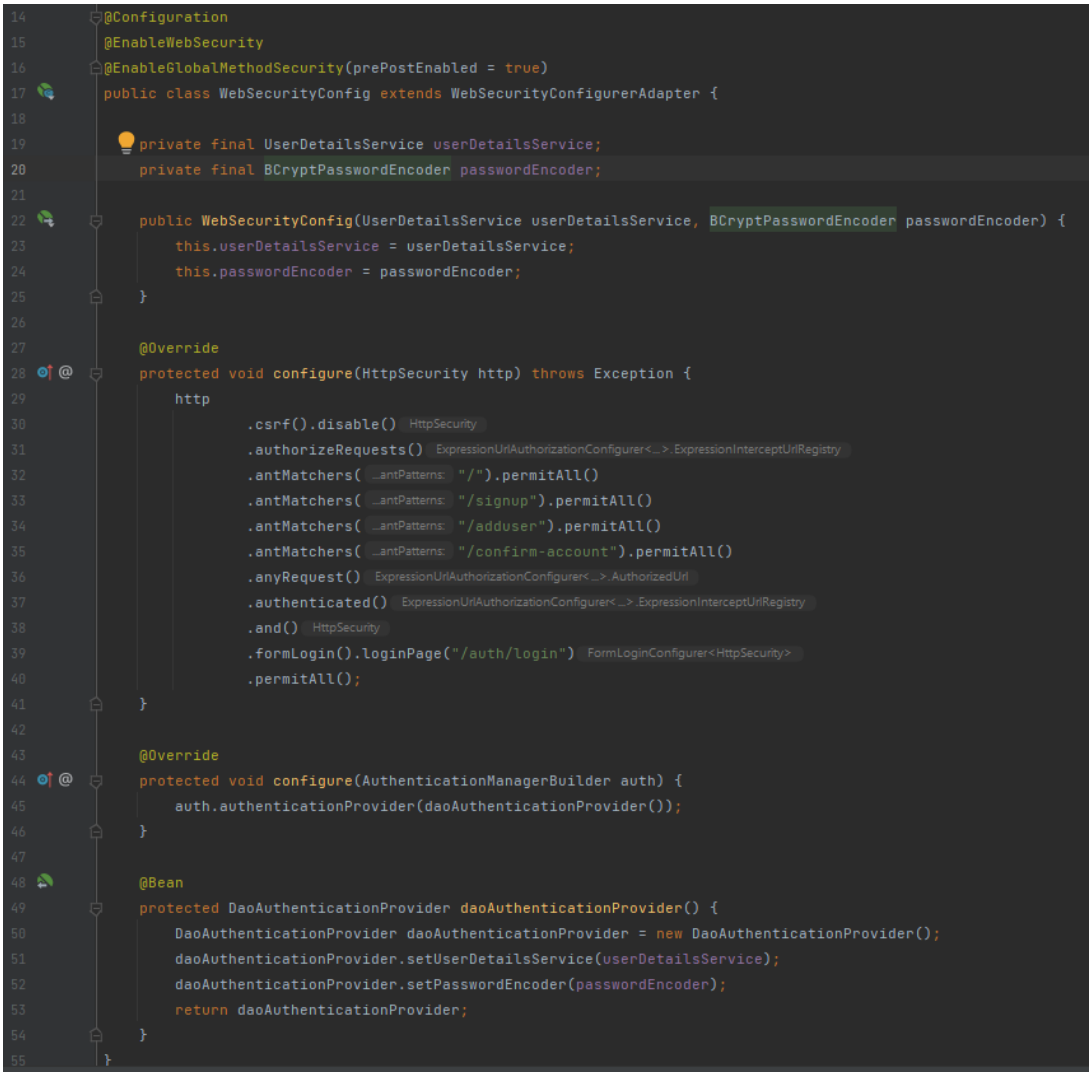


Рис. 3.10. Конфігурація класу WebSecurityConfig

В своєму проекті я створила клас WebSecurityConfig, який є конфігурацією (про що каже анотація @Configuration) реалізації аутентифікації. Він успадковується від WebSecurityConfigurerAdapter та перевизначає два перевантажених метода. В ньому створюється налаштований бін DaoAuthentocationProvider, який відповідає за аутентифікацію саме за допомогою бази даних, а також визначаються сторінки, до яких мають доступ гості веб-сайту.

UserDetailsService - це один із основних інтерфейсів Spring Security. Аутентифікація будь-якого запиту залежить від реалізації інтерфейсу UserDetailsService. Найчастіше він використовується для аутентифікації

використовуючи бази даних для отримання даних. Дані витягуються з реалізацією єдиного методу loadUserByUsername(), де ми надаємо нашу логіку для отримання інформації про користувача. Метод викине помилку UsernameNotFoundException, якщо користувача не буде знайдено. На рисунку 3.11 зображена реалізація класу, який імплементує UserDetailsService.

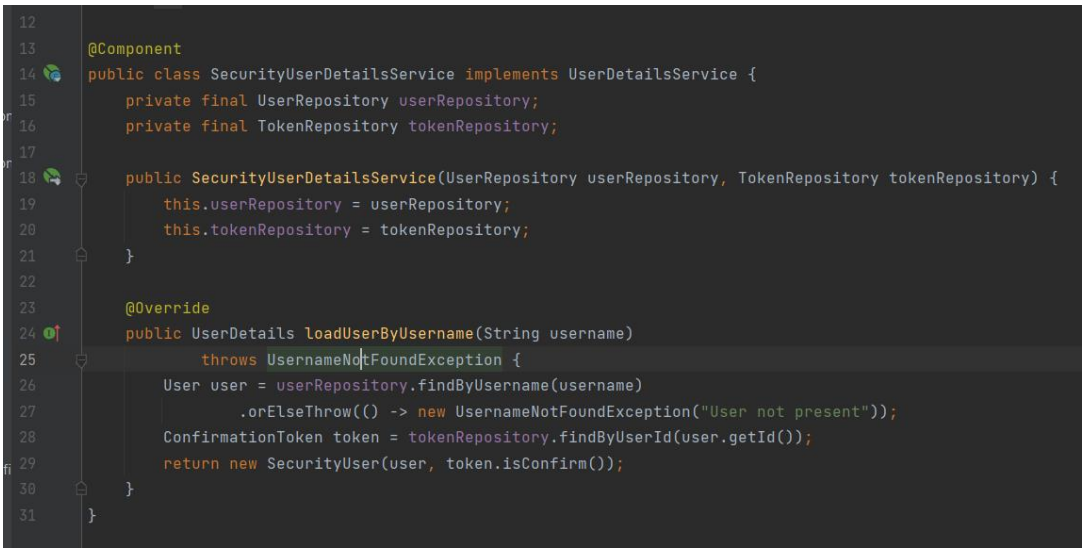


Рис. 3.11

Spring Security 5 вимагає використання PasswordEncoder для зберігання паролів. Це кодує пароль користувача, використовуючи одну з його численних реалізацій. Найбільш поширеною з його реалізацій є BCryptPasswordEncoder.

Я використовувала BcryptPasswordEncoder з силою кодування 12, створивши окремий його бін задля легкості у майбутньому використанні (рис. 3.12.).

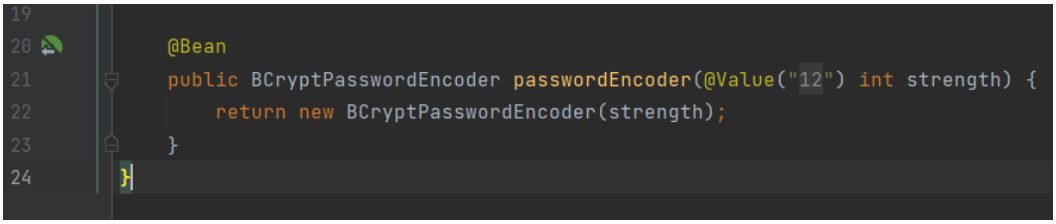


Рис. 3.12

Також, необхідно створити сутність юзера, тобто клас, який імплементує UserDetails інтерфейс (рис. 3.13.). Цей клас надає основну інформацію про користувача. Він використовується для зберігання даних, які згодом будуть інкапсульовані в об'єкти аутентифікації.

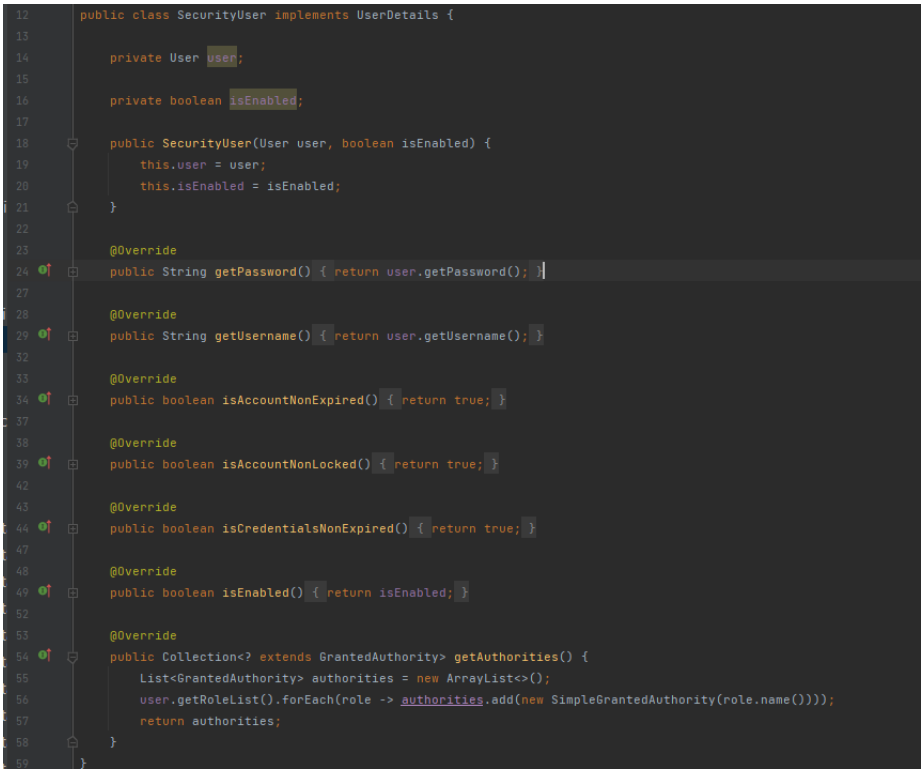


Рис. 3.13

В якості authorities в даному проекті виступає список ролей, притаманні користувачу, оскільки зв’язок між користувачем і ролями я визначила як багато-до-багатьох.

3.6. Розробка реєстрації

З реєстрацією все набагато легше, оскільки необхідно лише реалізувати запис користувачів до БД. Відбувається це через end-point “/signup”, метод якого вертає сторінку з формою реєстрації. Після натискання кнопки «зареєструватись» користувачу надсилається посилання з верифікацією

пошти та відбувається переадресація на сторінку, яка інформує користувача про необхідність підтвердження електронної пошти. Посилання складається з імені хоста, порта та токена, згенерованого за допомогою UDDI.

Рис. 3.14. Загальний вигляд моделі ConfirmationToken

Доки пошта не верифікована, доступ до акаунту заблокований. Якщо через один день токен не був підтверджений – користувач видаляється з бази даних без можливості аутентифікації, та в майбутньому буде потребувати повторної реєстрації.

3.7. SMTP розсилка

Простий протокол передачі пошти (SMTP) — це стандартний інтернет-протокол для передачі електронної пошти. Поштові сервери та інші агенти передачі повідомлень використовують SMTP для надсилання та отримання поштових повідомлень.

Існують сервіси для тестування email-розсилки. Приклад такого сервісу, який я використовувала - mailHog. MailHog – це інструмент для тестування електронної пошти з підробленим SMTP-сервером. Він інкапсулює протокол SMTP з розширеннями та не вимагає спеціальних внутрішніх реалізацій.

MailHog запускає дуже простий SMTP-сервер, який збирає вихідні листи, надіслані на нього. Перехоплені листи можна побачити у веб-інтерфейсі. Підроблений SMTP-сервер дозволяє уникнути проблем, пов'язаних із конфігурацією сервера або чимось подібним.

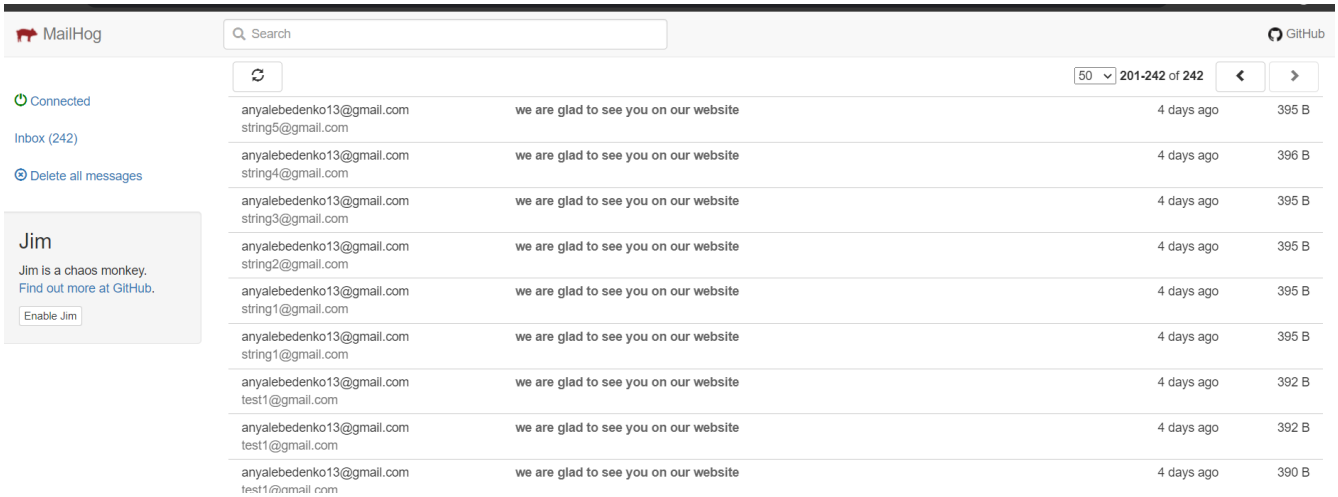


Рис. 3.15. UI вигляд MailHog

В своєму проекті я використовую розсилку при привітанні нового користувача, про повідомлення користувачів про закриття салону і відміну їх бронювання, та для відправки підтвердження пошти.

Для того, щоб налаштувати SMTP розсилку необхідно налаштувати властивості веб-додатку у файлі application.properties.

```
spring.mail.host=localhost
spring.mail.port=2525
```

Рис. 3.16. Налаштування властивостей задля реалізації розсилки

А також створити клас MailService, який буде відповідати саме за розсилку (рис. 3.17.).

```java
@Async("mailSenderExecutorService")
public void sendUserNotification(User customer, String message, String subject) {
    SimpleMailMessage mailMessage = new SimpleMailMessage();
    mailMessage.setSubject(subject);
    mailMessage.setText(message);
    mailMessage.setTo(customer.getEmail());
    mailMessage.setFrom("anyalebedenko13@Gmail.com");
    mailSender.send(mailMessage);

    long l = atomicLong.incrementAndGet();
    logger.info("Send message from thread {}, total count {}", Thread.currentThread().getName(), l);
}
```

Рис. 3.17

Також, в якості реального SMTP- сервера я використовувала Gmail-SMTP. Конфігурація майже нічим не відрізняється від конфігурації використання MailHog, за винятком властивостей. У властивостях додатку необхідно вказати хост - smtp.gmail.host, вказати свою пошту, отриманий smtp пароль та порт.

Для того, щоб підіймати сервер mailHog локально та з подальшою зручністю у розгортанні проекту, необхідно скористатися можливостями Docker. Для цього необхідно описати та встановити image, описаний в docker-compose.yml (рис. 3.18.).

```yaml
smtp-server:
  container_name: mailhog
  image: mailhog/mailhog
  restart: on-failure
  ports:
    - "2525:1025"
    - "8025:8025"
```

Рис. 3.18

3.8. Робота з локальним сховищем об’єктів

Від хмарних рішень для резервного копіювання даних до мереж мереж доставки контенту (CDN), що забезпечують безперебійну роботу, можливість зберігання неструктурованих великих бінарних об'єктів (blob) з можливістю доступу до цих об'єктів через HTTP API, відоме як сховище об'єктів, є невід'ємною частиною сучасного технологічного середовища.

Minio - це популярний сервер зберігання об'єктів з відкритим вихідним кодом, сумісний із хмарним сховищем Amazon S3. Програми, які були налаштовані для зв'язку з Amazon S3, можуть також бути налаштовані для зв'язку з Minio, що дозволяє Minio служити життєздатною альтернативою S3. Служба може зберігати неструктуровані дані, такі як фото, відео, файли журналу, резервні копії та образи контейнерів/віртуальних машин, і навіть може надати один сервер зберігання об'єктів, що об'єднує в пул багато дисків, розміщених на різних серверах.

В своєму проекті я вибрала саме MinIO по перше, через його легкість налаштування та використання. В моєму додатку передбачено додавати, видаляти, отримувати файли, відправляючи запит до контролеру.

```
@Override
public byte[] getFile(String fileName, String bucketName) {
    if (!isBucketExists(bucketName)) {
        throw new IllegalArgumentException("bucket with name" + bucketName + "don`t exist");
    }
    try {
        GetObjectResponse object = minioClient.getObject(GetObjectArgs.builder()
                .bucket(bucketName)
                .object(fileName)
                .build());
        return object.readAllBytes();
    } catch (Exception e) {
        throw new IllegalArgumentException(e.getMessage());
    }
}
```

Рис. 3.19. Приклад реалізації одного з методу сервісу MinIO

MinIO Object Storage використовує бакети для організації об'єктів. Бакет схожий на папку або каталог у файловій системі, де кожен бакет може містити довільну кількість об'єктів. Бакети MinIO мають ту ж функціональність, що і бакети AWS S3.

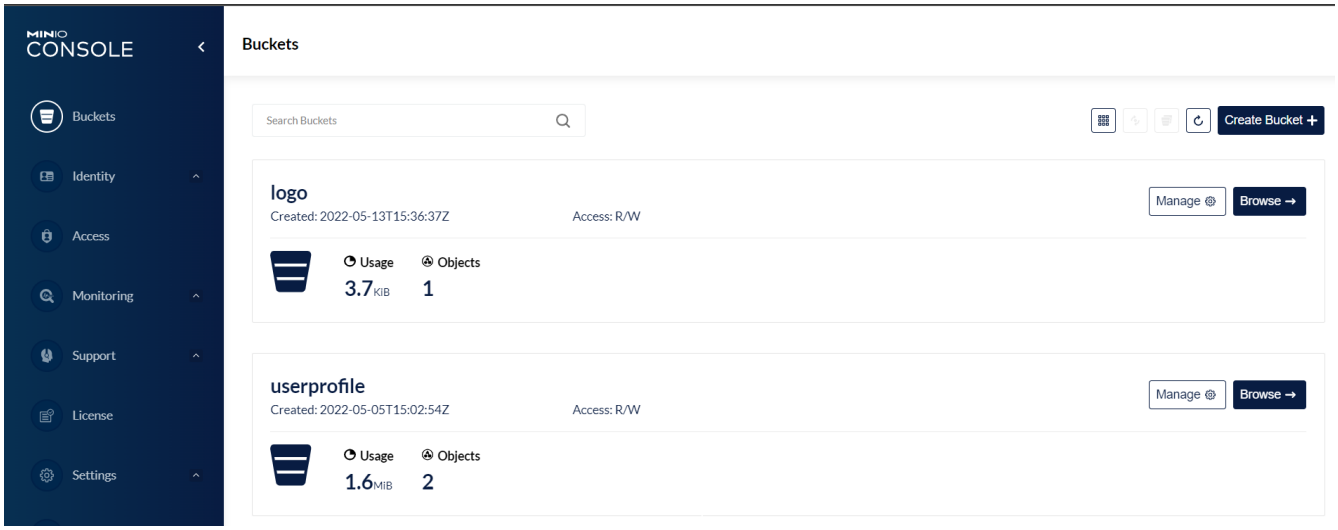


Рис. 3.20. UI вигляд MailHog

Для того, щоб підіймати сервер minIO локально та з подальшою зручністю у розгортанні проекту, необхідно скористатися можливостями Docker. Для цього необхідно описати та встановити image, описаний в docker-compose.yml (рис. 3.21.).

```yaml
minio:
  image: minio/minio:latest
  command: server --console-address ":9001" /data/
  ports:
    - "9000:9000"
    - "9001:9001"
  environment:
    MINIO_ROOT_USER: hanna13
    MINIO_ROOT_PASSWORD: minio1313
```

Рис. 3.21

ВИСНОВКИ

В першому розділі досліджено веб-сервіси та їх популярні технології. Можна зробити висновок, що веб-служби засновані виключно на відкритих та загальноприйнятих інтернет-протоколах. У цьому їхня головна сила, але й істотна слабкість. Плюси веб-сервісів не у всьому, але, безумовно, є категорією програмних продуктів, які швидко розвиваються, та за якими стоїть майбутнє.

При структуруванні веб-застосунків зазвичай рекомендується створювати їх за допомогою архітектури RESTful. Часто гарною практикою є планування колекцій та ресурсів RESTful із самого початку, адже це не тільки покращить архітектуру програми, а також дозволить у майбутньому легко створити API для інтеграції з іншими службами чи програмами.

В другому розділі було розглянуто технології, які використовувались у дипломному проекті. Ці технології роблять розробку веб-додатку швидше та легше. Було визначено, що міграціїї даних дуже важливі, оскільки вони допомогають легко змінювати та створювати схему бази даних, а середовище Spring - це дуже просте, легке та надійне середовище для створення сучасних програм на основі Java, воно надає безліч функціональних можливостей за модульним принципом, розробник може використовувати ті, які йому дійсно потрібні, а програмне забезпечення. Так само Docker є важливим інструментом для кожного сучасного розробника як основа апаратної віртуалізації додатків. Ця технологія має широкий функціонал і можливості для контролю процесів.

В третьому розділі було представлено реалізацію розробленого веб-сервісу, з використанням технологій контейнеризації, smtp-розсилки, сховищем об’єктів та досліджено функціональність Sping Security, за допомогою якого було реалізовано авторизацію та реєстрацію. В ході розробки веб-сервісу було набуто навички побудови рівнів веб-сервісу,

використання Spring Framework, а також більш детально ознайомлено з REST – архітектурою.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Хабибуллин И. Разработка Web - служб средствами Java / И. Хабибуллин. – Санкт-Петербург : БХВ-Петербург, 2003. – 400 с.
2. Walls C. Spring in action / Craig Walls. – 2-ге вид. – Greenwich, CT : Manning Publications Co., 2008. – 730 с.
3. Волш А. И. Основы программирования на Java для World Wide Web / А. И. Волш. – Київ : Диалектика, 1996. – 512 с.
4. Docker Desktop overview [Електронний ресурс] – Режим доступу: https://docs.docker.com/desktop/ (дата звернення: 03.06.2022). – Назва з екрана.
5. Spring Framework Documentation [Електронний ресурс] – режим доступу: https://docs.spring.io/spring-framework/docs/current/reference/html/ (дата звернення: 10.02.2022). - Назва з екрану
6. MinIO | The MinIO Quickstart Guide [Електронний ресурс] – Режим доступу: https://docs.min.io/ (дата звернення: 10.05.2022). - Назва з екрану
7. Flyway by Redgate • Database Migrations Made Easy. [Електронний ресурс – Режим доступу: https://flywaydb.org/documentation/ (дата звернення: 08.05.2022). - Назва з екрану
8. Spring Framework - Overview [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/spring/spring_overview.htm (дата звернення: 08.05.2022). – Назва з екрану.
9. Index of /spring-security/site/docs/3.0.x/reference [Електронний ресурс] – Режим доступу: https://docs.spring.io/spring-security/site/docs/3.0.x/reference/ (дата звернення: 08.06.2022). – Назва з екрана