Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для обміну кулінарними рецептами. Back-end
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-9

Костенко М.Р
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Онищенко К.Г.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Костенко Максиму Родіоновичу
(прізвище, ім’я, по батькові)

1. Тема роботи *«Програмна система для обміну кулінарними рецептами. Back-end»*

Затверджена наказом по університету від 20 *травня* 2024 № 471*Ст*

2. Термін подання студентом роботи до екзаменаційної комісії 17.06.2024

3. Вихідні дані до роботи *створення бекенд для мобільного кулінарного додатку, який працює на PHP з використанням фреймворку Laravel. Основне завдання — забезпечити надійне зберігання та ефективну обробку даних про рецепти, інгредієнти та користувацькі профілі.*

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	08.04.2024	*виконано*
2	Створення специфікації ПЗ	22.04.2024	*виконано*
3	Проектування ПЗ	24.04.2024	*виконано*
4	Розробка ПЗ	28.04.2024	*виконано*
5	Тестування ПЗ	30.04.2024	*виконано*
6	Оформлення пояснювальної записки	05.05.2024	*виконано*
7	Підготовка презентації та доповіді	06.05.2024	*виконано*
8	Попередній захист	09.06.2024	*виконано*
9	Нормоконтроль, рецензування	06.06.2024	*виконано*
10	Здача роботи у електронний архів	13.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	15.06.2024	*виконано*

Дата видачі завдання 08 *квітня* 2024р.

Студент (ка)_____ Костенко М.Р.
(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Онищенко К.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 61 стор., 19 рис., 15 джерел.

КУЛІНАРНА ПЛАТФОРМА, МОБІЛЬНИЙ ДОДАТОК, BACK-END, LARAVEL, MYSQL, PHP.

Об’єкт розробки – серверна частина мобільного додатка для кулінарної платформи для обміну рецептами.

Мета розробки – розробка надійної та ефективної серверної частини для обробки даних та взаємодії з мобільним додатком.

Метод рішення – серверна частина реалізована на Laravel та PHP, використовується база даних MySQL для зберігання інформації про користувачів та рецепти.

У результаті розробки була спроектована серверна платформа, яка забезпечує ефективну обробку запитів та надійне зберігання даних. Це забезпечує стабільну роботу мобільного додатку та високий рівень сервісу для користувачів.

CULINARY PLATFORM, MOBILE APPLICATION, BACK-END, LARAVEL, MYSQL, PHP.

The object of development is the server part of a mobile application for a Exchanging Culinary Recipes.

The goal of development is to create a reliable and efficient server-side for data processing and interaction with the mobile application.

Solution method – the server-side is implemented using Laravel and PHP, with a MySQL database for storing information about users and recipes.

As a result of the development, a server platform was designed that ensures efficient request processing and reliable data storage. This ensures the stable operation of the mobile application and a high level of service for users.

Я, Костенко Максим Родіонович, студент гр. ПЗПІ-20-9, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для обміну кулінарними рецептами. Back-end», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Розробка додатку "CookCraft" відповідає сучасним трендам у сфері харчування та кулінарії, пропонуючи користувачам швидкий доступ до рецептів з різних куточків світу. В умовах сучасного життя, коли часто не вистачає часу на приготування їжі, наш додаток допомагає користувачам легко знаходити рецепти та покращувати кулінарні навички через інтерактивні функції та можливість спілкування з досвідченими шеф-кухарями.

Додаток вирішує проблему інтеграції у повсякденне життя здорових харчових звичок, надає нові ідеї для кулінарних шедеврів, а також сприяє обміну кулінарним досвідом серед користувачів з різних куточків планети. Це не просто додаток, а ціла соціальна мережа для тих, хто цінує високу кухню та культуру харчування.

Нові можливості для кулінарних ентузіастів та професіоналів, об'єднуючи їх в єдиній платформі для обміну досвідом та ідеями. Цей додаток відіграє важливу роль у популяризації кулінарного мистецтва, дозволяючи користувачам відкривати нові гастрономічні горизонти без відвідування ресторанів високого класу. Він спрощує процес вибору страв, забезпечуючи користувачів необхідними інструментами для планування їжі на кожен день. Такий підхід сприяє більшій організації у щоденному житті користувачів, а також мотивує їх до здорового харчування та кулінарної креативності.

Додаток включає різноманітні функції, що дозволяють користувачам не тільки додавати власні рецепти, але й отримувати доступ до ексклюзивних кулінарних порад від відомих шеф-кухарів. Преміум-версія застосунку розкриває додаткові можливості, такі як відеоуроки та особисті кулінарні семінари, що робить кулінарний процес ще більш захоплюючим і творчим.

Основними завданнями розробки застосунку є:

– аналіз існуючих кулінарних застосунків та визначення функціональних та нефункціональних вимог для ідентифікації прогалин та можливостей, що

дозволить створити унікальний продукт, який задовольнятиме сучасні потреби користувачів;

- вибір та обґрунтування сучасних технологій для розробки мобільного клієнта та серверної частини, забезпечуючи високу продуктивність, масштабованість та безпеку даних користувачів;
- розробка інтуїтивно зрозумілого інтерфейсу, що сприяє легкому використанню застосунку, взаємодії з іншими користувачами, обміну рецептами та кулінарним досвідом;
- тестування та оптимізація продукту з метою забезпечення стабільності роботи та високої задоволеності користувачів;
- інтеграція застосунку з соціальними мережами та іншими зовнішніми платформами для забезпечення широкого розповсюдження та використання продукту.

Крім того, застосунок "CookCraft" використовує передові технології для забезпечення зручності та інтуїтивно зрозумілого інтерфейсу, що робить його доступним для користувачів будь-якого рівня кулінарних навичок. Він спроектований таким чином, щоб забезпечити максимальну ефективність та задоволення від користування, незалежно від кулінарного досвіду користувача. Це створює фундамент для стійкої спільноти, де кожен може навчитися, надихнутися та внести свій вклад у світ кулінарії.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Кулінарна галузь неперервно розвивається, з огляду на зростаючий інтерес до здорового харчування, міжнародної кухні та персоналізованого підходу до готування їжі. Розвиток мобільних технологій і збільшення числа користувачів смартфонів стимулюють попит на кулінарні додатки, які допомагають користувачам знаходити рецепти, планувати харчування та вчитися кулінарним навичкам незалежно від їхнього місцезнаходження.

Згідно з дослідженнями, кулінарні додатки займають одне з лідируючих місць у категорії домашніх застосунків. Вони не тільки спрощують процес приготування їжі, але й мотивують людей готувати вдома, що може бути здоровішим та економічним варіантом.

Наразі суспільство дедалі більше усвідомлює важливість харчування та його вплив на здоров’я. Це стимулює інтерес до освіти в галузі кулінарії та використання цифрових ресурсів для самонавчання. Цифровізація рецептів і кулінарних технік відкриває нові можливості для експериментів із різними кухнями світу, пропонуючи користувачам зручні інструменти для розширення своїх гастрономічних горизонтів.

Крім того, викликами для кулінарної індустрії залишаються потреби алергіків та людей із специфічними дієтичними обмеженнями. Використання технологій дозволяє створювати персоналізовані рецепти, що враховують індивідуальні потреби користувачів, тим самим забезпечуючи їх безпеку та здоров’я.

Враховуючи глобалізацію та зміну харчових звичок, існує постійний попит на інноваційні рішення в кулінарній сфері, що включають інтеграцію з соціальними мережами та іншими платформами для обміну досвідом. Це відкриває можливості для взаємодії користувачів не тільки в освітніх цілях, але й для створення спільнот, що сприяє обміну знаннями та кулінарними традиціями різних народів.

Однак, багато існуючих застосунків зосереджуються на стандартному наборі рецептів без можливості глибокої кастомізації під особисті потреби користувача або занадто складні для користувачів без кулінарного досвіду. Також важливо зазначити, що існуючі рішення часто ігнорують соціальний аспект кулінарії, такий як обмін рецептами з друзями або участь у кулінарних викликах. Тож перейдемо до аналізу існуючих рішень та подивимось їх сильні та слабкі сторони.

Yummly — це одне з найпопулярніших кулінарних додатків (див .рис. 1.1), яке надає користувачам доступ до величезної кількості рецептів з усього світу. Заснований у 2009 році, Yummly використовує потужні алгоритми для адаптації рецептів до персональних уподобань користувачів, враховуючи не тільки смакові переваги, але й алергії та дієтичні обмеження. Платформа також допомагає планувати прийоми їжі та автоматично генерує список покупок на основі вибраних рецептів.

Yummly не тільки пропонує рецепти, але й інтегрує функціональність "підбору рецептів", яка дозволяє користувачам відкривати нові страви на основі їхніх попередніх виборів та оцінок. Цей інтелектуальний підхід до вибору страв робить кулінарний процес не тільки простішим, але й більш захоплюючим. Користувачі можуть досліджувати міжнародні кухні, від швидких сніданків до витончених вечерь, все з врахуванням їхніх харчових звичок і переваг.

Переваги платформи:

- інтелектуальні рекомендації, що адаптуються до дієтичних потреб користувача;
- велика база даних рецептів із зручними фільтрами для пошуку;
- інтеграція з кухонними приладами, що підключені до інтернету, для автоматичного налаштування температури та часу готування.

Недоліки:

- потребує платну підписку для доступу до деяких преміальних функцій;
- іноді персоналізація не враховує всі потреби користувача через обмежену інформацію про інгредієнти.

Рисунок 1.1 – Логотип Yummly (за даними [1])

Allrecipes (див. рис. 1.2) є однією з найстаріших і найвідоміших платформ у світі кулінарії, яка з'явилася на ринку у 1997 році. Завдяки своїй широкій базі даних рецептів, сайт забезпечує користувачам унікальну можливість не лише знаходити страви за будь-яким критерієм, але й ділитися власними кулінарними досягненнями. Основною рисою Allrecipes є її спільнота: користувачі активно беруть участь у житті ресурсу, залишаючи відгуки та рекомендації до рецептів, що дозволяє новачкам уникати помилок та отримувати кращі результати.

Рисунок 1.2 – Логотип Allrecipes (за даними [2])

Переваги платформи:

- величезна кількість рецептів доступна відразу, з постійним оновленням від користувачів з усього світу;
- високий рівень взаємодії користувачів сприяє кращому вибору рецептів;
- можливість зберегти улюблені рецепти в особистий кулінарний блокнот, що полегшує доступ до них в майбутньому.

Недоліки:

- інтерфейс сайту може здатися перевантаженим новим користувачам через велику кількість інформації і рекламних блоків;
- відсутність персоналізації харчування на індивідуальному рівні може стати проблемою для людей із специфічними дієтичними вимогами.

BigOven (див. рис 1.3) — це кулінарний додаток, який пропонує широкі можливості для планування їжі, організації рецептів та покупок. Заснований у 2003 році, BigOven став популярним серед користувачів, які цінують зручність і організацію в кулінарному процесі. Значна частина привабливості BigOven полягає в тому, що він допомагає користувачам виявляти нові кулінарні ідеї на основі того, що вже є у їхньому холодильнику, зменшуючи таким чином кількість харчових відходів. Крім того, додаток забезпечує можливість збереження улюблених рецептів, планування сімейних обідів на тиждень та створення списку покупок, що можна легко поділити з іншими членами сім'ї або використати під час покупок.

Однією з унікальних особливостей BigOven є його здатність адаптуватися до індивідуальних дієтичних потреб користувачів, пропонуючи рецепти з урахуванням алергій, дієтичних обмежень або особистих переваг. Це робить додаток ідеальним інструментом для тих, хто слідує специфічним дієтам, таким як безглютенова, вегетаріанська або низькокалорійна.

Рисунок 1.3 – Логотип BigOven (за даними [3])

Переваги:

– багатофункціональний підхід забезпечує повне кулінарне планування;
– велика база даних рецептів зі зручним пошуком.

Недоліки:

– інтерфейс може здатися складним для нових користувачів через велику кількість опцій і налаштувань.

1.2 Виявлення та вирішення проблем

Розробка кулінарного застосунку виявляє кілька важливих проблем у сфері кулінарних застосунків, які потребують вирішення. Перша проблема — це недостатня інтеграція з соціальними функціями в більшості кулінарних додатків. Багато застосунків зосереджені лише на рецептах без можливості спілкування користувачів або обміну досвідом. Це створює відчуття ізоляції і обмежує можливості для взаємодії та навчання від інших кулінарів.

Наступна проблема — відсутність персоналізації рекомендацій рецептів. Більшість кулінарних застосунків пропонують загальний набір рецептів, не враховуючи індивідуальних смаків або дієтичних обмежень користувача, що може зменшувати корисність і задоволення від використання застосунку. Проблема інтерфейсу — багато кулінарних застосунків мають перевантажені або

неінтуїтивно зрозумілі інтерфейси, що ускладнює взаємодію, особливо для нових користувачів.

Вирішити ці проблеми можна запровадивши функції соціальної взаємодії, таких як коментарі, оцінки та обговорення рецептів, що дозволить користувачам ділитися порадами та особистим досвідом.

Розробити чистий та зрозумілий інтерфейс, що забезпечує легкий доступ до всіх основних функцій застосунку, роблячи використання застосунку зручнішим та приємнішим. Однією з основних проблем, яка спостерігається у багатьох кулінарних застосунках, є розрізненість та неповнота кулінарної інформації.

Користувачі часто мусять витрачати час на пошук додаткових даних про інгредієнти або техніки приготування на різних платформах. Відповідь на цю проблему полягає у створенні інтегрованої бази знань, де кожен рецепт супроводжується детальним описом інгредієнтів, їх можливих замінників, а також відео- та текстовими інструкціями. Це забезпечує користувачам всеохоплюючий досвід, дозволяючи заощадити час та зусилля.

1.3 Постановка задачі

На основі проведеного аналізу предметної галузі та виявлених проблем, метою даної кваліфікаційної роботи є розробка програмної системи "CookCraft" для кулінарних зацікавлень. Дана система покликана вирішити ключові недоліки існуючих кулінарних застосунків, забезпечуючи користувачам зручний інструмент для виявлення, зберігання та обміну рецептами.

Основні завдання роботи включають:

– система повинна забезпечити механізми для інтеграції та уніфікації кулінарної інформації з різних джерел, щоб користувачі мали доступ до повного спектру даних про рецепти, інгредієнти, кулінарні техніки та відгуки;

- персоналізація досвіду користувача, яка фільтрує та персоналізує контент, що дозволить користувачам знаходити рецепти за індивідуальними уподобаннями, дієтичними обмеженнями та доступними інгредієнтами;
- створення інструментів для залучення та участі користувачів у спільноті, таких як можливість коментування, оцінювання рецептів, а також проведення кулінарних конкурсів;
- зберігання та адміністрування даних включає розробку надійної та масштабованої бази даних для зберігання великого обсягу інформації, забезпечення її захисту та конфіденційності;
- впровадження зручного та інтуїтивно зрозумілого інтерфейсу, який буде оптимізований для різних пристроїв і платформ;
- технологічне вдосконалення включає використання сучасних технологій розробки, таких як реактивні фреймворки і API для інтеграції з зовнішніми сервісами, для забезпечення високої продуктивності та масштабованості системи.

Мета роботи — створення комплексної програмної системи, яка не тільки вирішить існуючі проблеми кулінарних застосунків, але й відкриє нові можливості для користувачів глибше досліджувати світ кулінарії.

1.3.1 Цільова аудиторія

Програмна система "CookCraft" для кулінарії орієнтована на широку цільову аудиторію, що охоплює різні категорії любителів кулінарії. Детальний аналіз цільової аудиторії допоможе забезпечити ефективне задоволення потреб та очікувань користувачів під час розробки системи. Нижче наведено перелік зацікавлених груп користувачів:

- початківці кулінари, які тільки починають свій шлях у кулінарії. Для них важливо мати доступ до простих та зрозумілих рецептів з докладними інструкціями та візуальними покроковими підказками;

- професійні кухари або досвідчені кулінари, які шукають нові ідеї та складні рецепти. Вони цінують високу якість контенту, можливість обмінюватися досвідом та отримувати відгуки від інших користувачів;
- батьки, які планують щоденне харчування для своїх сімей і шукають здорові, швидкі та економні рецепти. Для них корисними будуть функції планування харчування, списки покупок та рекомендації щодо дитячого харчування.
- люди, які дотримуються особливих дієт (наприклад, веганської, безглютенової, палео тощо) та шукають рецепти, що відповідають їхнім харчовим обмеженням. Вони вітають персоналізацію рецептів з можливістю адаптації під свої потреби;
- користувачі, які створюють контент для соціальних мереж та блогів. Їм потрібні інструменти для легкого створення та ділення контенту, інтеграції з соціальними мережами та залучення аудиторії;
- організації, які використовують кулінарний контент для навчальних цілей, включаючи викладання кулінарного мистецтва. Вони можуть використовувати систему як навчальний ресурс з багатим набором інструментів для вивчення і практики.

Система «CookCraft» створена для забезпечення максимально широкого охоплення різних кулінарних інтересів, з особливим акцентом на індивідуальні потреби користувачів. Чи то новачок у світі кулінарії, який шукає базові рецепти, чи досвідчений шеф, який прагне експериментувати з новими та складними стравами, кожен знаходить тут щось для себе. Особлива увага приділяється забезпеченню доступності та зручності користування, що дозволяє всім користувачам легко орієнтуватися в додатку та використовувати його можливості на повну.

Платформа активно підтримує спільноту кулінарів, надаючи їм інструменти для обміну знаннями та досвідом, що сприяє розвитку кулінарних навичок і зміцненню зав’язків між учасниками.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Вимоги до серверної частини кулінарної платформи акцентують увагу на створенні надійної та ефективної системи, що забезпечує швидке та безпечне оброблення даних. Основна увага приділяється розробці API, який повинен бути стабільним та здатним витримувати високе навантаження, надаючи користувачам безперервний доступ до функцій платформи. Наступні аспекти враховують технічні потреби системи та спрямовані на оптимізацію процесів взаємодії з базою даних:

- платформа повинна забезпечити ефективне управління базою даних з використанням сучасних методик оптимізації запитів та кешування даних, що забезпечить швидкий доступ до рецептів та управління кулінарними проектами. Інтерфейс серверної частини має бути розроблений так, щоб підтримувати легке масштабування та інтеграцію з зовнішніми сервісами.
- система повинна забезпечити захист даних користувачів та їх конфіденційність через впровадження передових механізмів автентифікації та авторизації, шифрування даних та захист від несанкціонованого доступу;
- з урахуванням міжнародного ринку користувачів, система має підтримувати кілька мовних версій, щоб забезпечити зручність користування для осіб з різних країн. Це допоможе уникнути мовного бар’єру та залучити більшу аудиторію;
- для зручності користувачів, платформа "CookCraft" інтегрується з платіжною системою PayPal, що дозволяє здійснювати безпечні платежі за підписки та інші преміум-функції. Це не тільки спрощує процес оплати, але й забезпечує додатковий рівень безпеки для фінансових транзакцій. Використання PayPal дозволяє користувачам швидко підписуватися на преміум-послуги, отримуючи ексклюзивний доступ до особливих рецептів, персоналізованих кулінарних порад та інших корисних ресурсів.

Вибір PHP та фреймворку Laravel обумовлений потребою в гнучкій, але потужній платформі, яка може ефективно масштабуватися та адаптуватися до змінних вимог замовників і ринку. Ці інструменти дозволяють швидко реагувати на зміни, вносити необхідні корективи у проект та підтримувати високий рівень продуктивності системи.

Необхідно розробити систему використовуючи перевірену роками СУБД - MySQL, яка відома своєю надійністю та високою продуктивністю при роботі з великими обсягами інформації. Використання MySQL забезпечує стабільність і швидкість обробки даних, що є критично важливим для бекенд-частини будь-якої інформаційної системи.

Для забезпечення зв'язку між серверною частиною та клієнтськими застосунками, "CookCraft" реалізує RESTful API, який спрощує інтеграцію та розширення функціоналу платформи. API дозволяє легко обмінюватися даними між різними компонентами системи та підтримує уніфікацію інтерфейсів, що є важливим для забезпечення стійкості та надійності в роботі.

Необхідно забезпечити безпечну систему авторизації на базі Sanctum, який використовується для аутентифікації та авторизації, забезпечуючи безпеку на основі токенів. Це дозволяє ефективно захищати дані користувачів від несанкціонованого доступу, використовуючи сучасні підходи до аутентифікації, які підходять як для веб-так і для мобільних додатків. Впровадження сучасних технологій захисту допомагає оберігати від SQL-ін'єкцій, XSS та інших загроз. Заходи безпеки включають шифрування комунікацій та використання безпечних методів обробки запитів, що сприяє забезпеченню конфіденційності та цілісності даних користувачів.

Завершуючи розділ про формування вимог до програмної системи, можна відзначити, що всебічний підхід до розробки, з акцентом на забезпечення гнучкості, продуктивності та безпеки, є ключовим для задоволення потреб і очікувань різноманітної аудиторії.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННІЯ

3.1 UML проектування ПЗ

Діаграма варіантів використання (Use Case Diagram) вважається ключовою у наборі діаграм UML та служить для моделювання та аналізу функціональних вимог системи через лінзу її зовнішніх користувачів або акторів. Ця діаграма ілюструє головні типи взаємодії між користувачами та функціями системи.

Нижче наведено Use Case діаграму (див. рис. 3.1) для авторизованого користувача системи, яка містить наступні елементи:

- користувач може переглядати деталі свого профілю;
- користувач може редагувати свої персональні дані в профілі;
- користувач може переглянути різні плани членства, доступні в системі, та обирати найбільш підходящий для себе;
- користувач має можливість скасувати свою підписку та відмовитися від стягування плати за неї;
- користувач має можливість переглядати перелік доступних рецептів на платформі;
- користувач може переглядати детальну інформацію про конкретний рецепт, включаючи інгредієнти та інструкції;
- користувачі можуть залишати свої коментарі та враження щодо рецептів;
- користувачі можуть оцінювати рецепти за допомогою рейтингової системи;
- користувачі можуть зберігати улюблені рецепти в спеціальній секції для швидкого доступу;
- користувач може переглядати інформацію про автора рецепту, його інші рецепти та рейтинг;
- користувачі можуть підписатися на оновлення від улюблених авторів рецептів, щоб отримувати сповіщення про нові публікації;
- автори рецептів мають можливість видаляти свої публікації з системи;

– автори можуть вносити зміни до своїх рецептів, оновлюючи інгредієнти чи методи приготування;
– автори можуть додавати нові рецепти, надаючи всю необхідну інформацію та інструкції;
– автори можуть переглядати список усіх своїх опублікованих рецептів для керування ними;

Рисунок 3.1 – Use-Case діаграма (Рисунок виконано самостійно)

На рисунку 3.2 відображена діаграма послідовності на якій представлено процес взаємодії користувача з системою публікації рецептів від авторизації до публікації рецепта. Діаграма послідовностей зображує взаємодію між користувачем і різними компонентами системи. Ось кроки, що представлені на діаграмі:

- користувач вводить дані для входу, що ініціює процес авторизації в системі;
- після успішної авторизації користувач вводить дані про рецепт, який він хоче створити. Це включає введення інгредієнтів, методів приготування тощо;
- зібрані дані про рецепт відправляються на сервер для перевірки та публікації;
- система перевіряє надані дані на відповідність стандартам публікації (наприклад, наявність усіх необхідних інгредієнтів, коректність опису процесу приготування);
- якщо дані про рецепт відповідають усім критеріям, відбувається публікація рецепта. Користувач отримує повідомлення про успішну публікацію;

Діаграма також включає альтернативний шлях, де користувач може отримати повідомлення про помилки у рецепті та необхідність їх виправлення перед повторним надсиланням на публікацію. Це дозволяє користувачам коригувати та адаптувати свої рецепти згідно з вимогами системи. Ця діаграма допомагає зрозуміти, як система реагує на дії користувача та як організована логіка обробки запитів від введення даних до публікації рецепта, ілюструючи основні взаємодії між користувачем і системою.

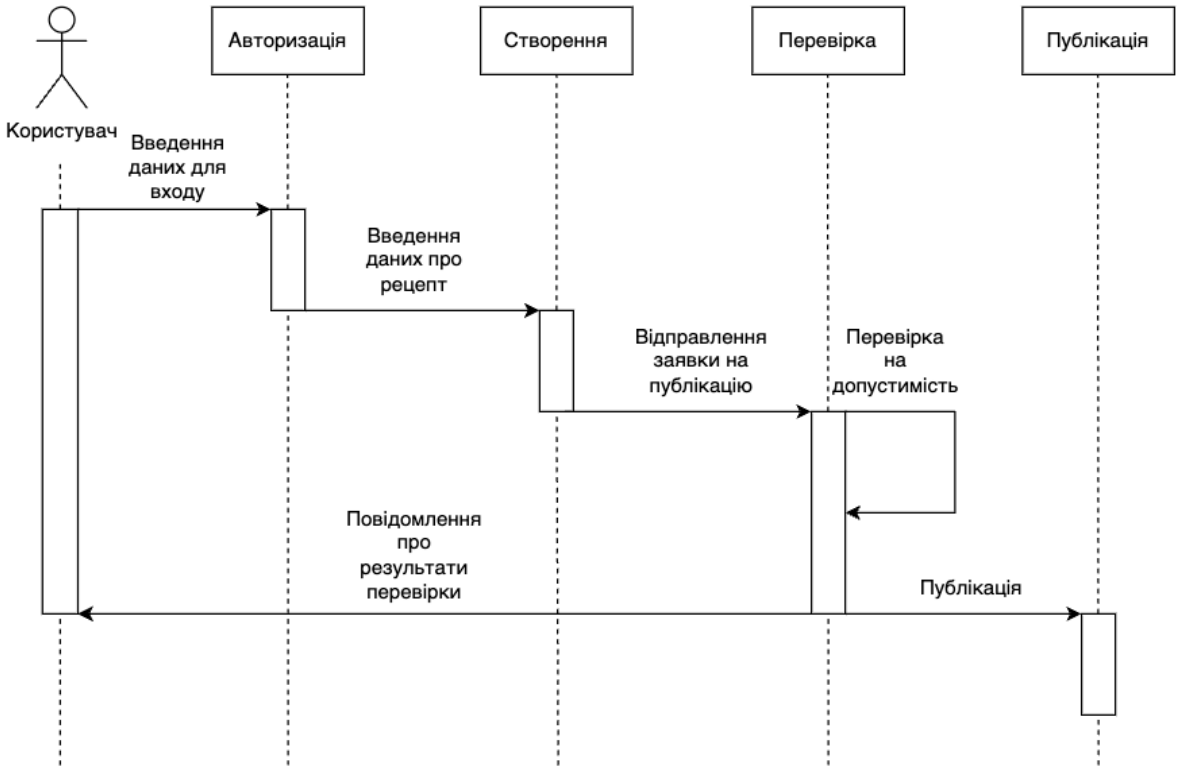

Рисунок 3.2 – Діаграма послідовності публікації рецепта (виконано самостійно)

3.2 Проектування архітектури ПЗ

Архітектура клієнт-сервер є фундаментальною для системи [6], оскільки вона визначає взаємодію між мобільним додатком (клієнт) і сервером. Основна концепція полягає у тому, що клієнт (мобільний додаток) та сервер (бекенд) комунікують через мережу Інтернет за протоколом HTTPs.

Бекенд, реалізований за допомогою Laravel та PHP, обробляє всі серверні запити, займається зберіганням даних, виконує бізнес-логіку, і надає дані фронтенду через RESTful API. Laravel як PHP фреймворк підтримує розробку з використанням MVC патерна, що сприяє чистоті коду та його модульності. На бекенді застосовується архітектура MVC (Model-View-Controller), що є стандартом для фреймворку Laravel. Ця архітектура допомагає в організації коду шляхом розділення бізнес-логіки (модель), користувацького інтерфейсу (вид) та управління вхідними даними (контролер).

Бекенд, реалізований за допомогою Laravel та PHP, обробляє всі серверні запити, займається зберіганням даних, виконує бізнес-логіку, і надає дані фронтенду через RESTful API. Laravel як PHP фреймворк підтримує розробку з використанням MVC патерна, що сприяє чистоті коду та його модульності. На бекенді застосовується архітектура MVC (Model-View-Controller), що є стандартом для фреймворку Laravel. Ця архітектура допомагає в організації коду шляхом розділення бізнес-логіки (модель), користувацького інтерфейсу (вид) та управління вхідними даними (контролер).

Моделі в системі представляють структуру даних, виконують валідацію та управління станами даних, а також взаємодіють з базою даних через ORM (Object-Relational Mapping) функції Laravel, це забезпечує оптимальну роботу з даними.

Контролери відповідають за обробку вхідних запитів від користувачів, виклик відповідних методів моделей для обробки даних та передачу даних до відображень. Вони є головним зв'язуючим компонентом між користувацьким інтерфейсом та моделями.

Представлення у Laravel зазвичай є шаблонами, що використовуються для генерації відповідей користувачу. Хоча в контексті API-центричної архітектури, представлення можуть не використовуватися так активно, замість цього формуючи відповіді JSON безпосередньо з контролерів.

Сервіси у Laravel дозволяють винести частину бізнес-логіки з контролерів для забезпечення більшої перевикористання коду та легшого тестування. Сервісні класи можуть включати валідацію даних, операції над даними та інші специфічні дії, які потрібно виконати.

Репозиторії використовуються для абстрагування та інкапсуляції логіки доступу до даних, забезпечуючи більш чистий та організований код у контролерах. Репозиторії дозволяють легко змінювати джерела даних без впливу на бізнес-логіку.

Класи для валідації використовуються для перевірки коректності вхідних даних до їх обробки або збереження в базі даних. Laravel надає потужні засоби для валідації, що можуть бути легко використані в будь-якому компоненті системи.

Використання MySQL для бази даних забезпечує надійне зберігання та швидкий доступ до даних. Використання MySQL як реляційної системи управління базами даних (РСУБД), яка забезпечує високу продуктивність, надійність, і широкі можливості для керування даними, що важливо для забезпечення цілісності даних і їх безпечного зберігання.

Взаємодія між клієнтом і сервером відбувається через HTTP-запити, де фронтенд відправляє запити до бекенду, а бекенд надсилає запитувані дані назад. Застосування Laravel Sanctum для аутентифікації та авторизації користувачів. Sanctum забезпечує легку інтеграцію з системами на основі токенів, що включає в себе захист від загальнопоширених вразливостей.

Цей підхід забезпечує чітке розділення відповідальностей між фронтендом та бекендом, спрощує розширення та супровідність системи та забезпечує високу продуктивність та масштабованість архітектури. Було спроектовано діаграму розгортання (див. рис. 3.3) яка показує розподіл системних компонентів по фізичних серверах і клієнтських пристроях.

Database Server включає артефакт MySQL Database, що підказує, що він використовується для зберігання і управління всіма даними системи. MySQL є реляційною базою даних, яка забезпечує високу продуктивність, надійність і зручність у масштабованості;

На Web Server розміщено API, що вказує на те, що він обробляє всі запити від клієнтів через веб-інтерфейс. API може бути розроблений з використанням будь-яких сучасних технологій для створення RESTful, який дозволяє обмінюватися даними між клієнтами і сервером;

Клієнтська частина системи представлена через пристрій, що працює на операційній системі Android. Цей додаток забезпечує користувачеві інтерфейс для взаємодії з системою, наприклад, для перегляду, створення чи редагування даних через мобільний пристрій.

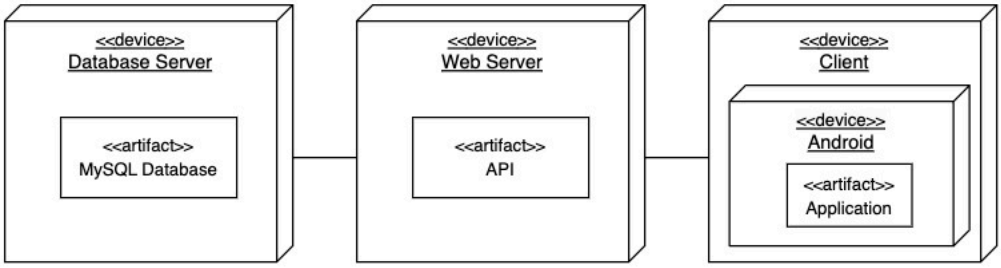

Рисунок 3.3 – Діаграма розгортання (Рисунок виконано самостійно)

3.3 Проектування структури зберігання даних

У рамках розробки серверної частини мобільного додатку для кулінарної платформи, велика увага приділялася проектуванню структури зберігання даних. Ефективна організація бази даних є критично важливою для забезпечення високої продуктивності, швидкого доступу до даних та їх безпеки. Ретельно спланована структура бази даних допомагає уникнути дублювання даних, спрощує процеси масштабування системи та оптимізує обробку запитів.

Створено надійну та масштабовану базу даних для кулінарної платформи, яка обслуговує різноманітні потреби користувачів – від управління профілями та рецептами до обробки платежів та підписок. Використання реляційної бази даних MySQL дозволяє ефективно зберігати та керувати об'ємними даними, а також забезпечує швидкий доступ до них через оптимізовані запити та індексацію.

Ігнорування ретельного проектування структури бази даних може мати серйозні наслідки для будь-якої інформаційної системи, включаючи кулінарні платформи. Недостатньо продумана структура БД може вплинути на продуктивність, масштабованість і надійність системи. Перш за все, відсутність належного проектування може призвести до низької продуктивності системи. Неефективні запити до бази даних і погано структуровані таблиці можуть сповільнювати обробку даних, особливо коли кількість користувачів і обсяги інформації зростають. Це може призвести до затримок у завантаженні інформації,

що негативно впливає на загальне враження користувачів від використання платформи.

Крім того, неправильно спроектована база даних може ускладнити масштабування системи. З непродуманою архітектурою БД стає важче адаптувати систему до зростаючих потреб користувачів, оскільки вносити зміни або доповнення стає технічно складніше і витратніше. Відсутність гнучкості у структурі бази даних може призвести до потреби її повної переробки, що може бути дуже витратним процесом. Наостанок, погано спроектована база даних створює ризики для безпеки даних. Неправильне використання або відсутність використання відповідних обмежень і заходів безпеки, таких як шифрування даних та налаштування прав доступу, можуть призвести до витоку конфіденційної інформації. Це не тільки порушує приватність користувачів, але й може завдати шкоди репутації компанії та привести до юридичних наслідків.

Нижче наведено ER-діаграму (див. рис. 3.4), яка відображає комплексну структуру бази даних.

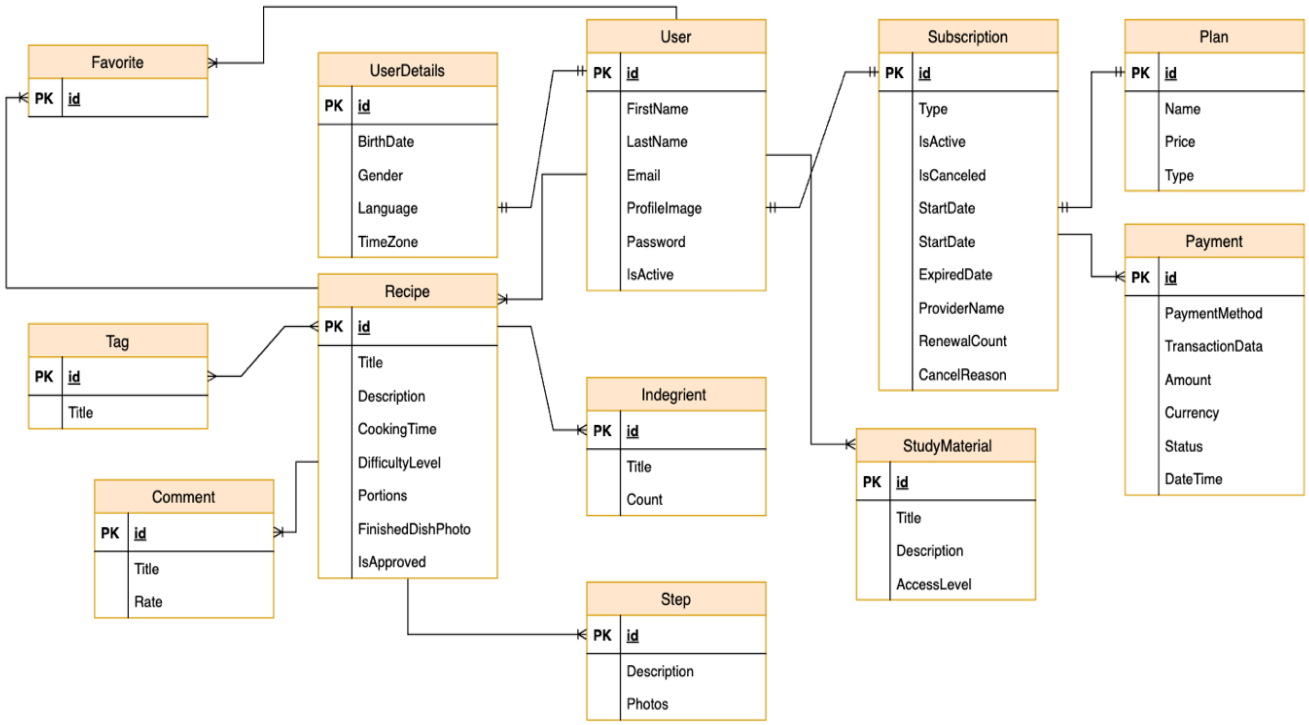

Рисунок 3.4 – ER-діаграма (Рисунок виконано самостійно)

В базі даних кулінарної платформи існує кілька основних таблиць, кожна з яких відіграє свою роль у зберіганні і обробці даних. Таблиця "Users" є центральним сховищем для інформації про користувачів, зберігаючи такі дані, як ім'я, прізвище, електронну адресу, зображення профілю, пароль та статус активності користувача.

Для детальнішої інформації про кожного користувача створена таблиця "UserDetails", де зберігаються додаткові особисті відомості, такі як дата народження, стать, мова та часовий пояс.

Управління підписками здійснюється через таблицю "Subscription", яка містить інформацію про статус підписки (активна чи скасована), дати початку та закінчення підписки, назву провайдера, кількість поновлень, а також причину скасування підписки. Таблиця "Plan" описує доступні плани підписок, зазначаючи назву плану, його ціну та тип.

Операції з платежами фіксуються в таблиці "Payment", де реєструються всі деталі транзакцій, включно з методом оплати, датою транзакції, сумою, валютою та статусом оплати.

Таблиця "Recipe" є основою для зберігання рецептів, включаючи назву, опис, час приготування, рівень складності, кількість порцій, фотографію готової страви та статус затвердження рецепту. Для кожного рецепту зберігається список інгредієнтів у таблиці "Ingredient", де кожен інгредієнт має свій ідентифікатор, назву, опис та кількість. Таблиця "Step" визначає кроки приготування для рецептів, зазначаючи ідентифікатор рецепту як зовнішній ключ, опис кожного кроку та супутні фотографії.

Також у базі даних спроектована таблиця "Comments", призначена для зберігання коментарів користувачів до рецептів. Кожен запис у цій таблиці містить унікальний ідентифікатор, заголовок коментаря, оцінку, яку може поставити користувач, ідентифікатори рецепту та користувача, які пов'язані через зовнішні ключі з таблицями "Recipes" та "Users" відповідно. Важливим аспектом є

використання каскадного видалення, що гарантує видалення всіх коментарів, пов'язаних з видаленим рецептом або користувачем.

Інша важлива таблиця - "Favorites", яка дозволяє користувачам зберігати улюблені рецепти. Вона містить ідентифікатори користувача та рецепту, а також мітки часу створення і останнього оновлення запису. Зв'язки з таблицями "Users" та "Recipes" також використовують каскадне видалення, щоб усунути улюблені рецепти при видаленні профілю користувача або рецепту.

3.4 Приклад найцікавіших алгоритмів та методів

Аутентифікація та авторизація в системі реалізовані за допомогою Laravel Sanctum, який забезпечує розширені можливості для управління сесіями та безпечного доступу користувачів до ресурсів. Sanctum використовує механізм хешування паролів з додаванням "солі" (див. рис. 3.5), що значно підвищує безпеку зберігання.

```
Maksym-Kostenko-Leasoft
public function login(LoginRequest $request) : JsonResponse
{
    $validated = $request->validated();
    $user = $this->userRepository->getUserByEmail($validated['email']);

    if (!$user || !Hash::check($validated['password'], $user->password)) {
        return $this->respondUnAuthenticated(__( key: "The provided credentials are incorrect."));
    }
    $token = $user->createToken( name: 'auth_token')->plainTextToken;
```

Рисунок 3.5 – Реалізація системи валідації пароля (виконано самостійно)

При реєстрації нового користувача, "сіль" генерується автоматично та додається до пароля перед його хешуванням, формуючи таким чином унікальний хеш. Під час процесу аутентифікації, система витягує "сіль" зі збереженого хешу, застосовує її до введеного пароля та повторно виконує хешування для порівняння з існуючим хешем в базі даних. Цей підхід дозволяє точно ідентифікувати користувачів та забезпечити надійний захист їхніх акаунтів.

Для підвищення взаємодії та залученості користувачів, програмна система інтегрована з системою подій Laravel, яка автоматизує процес комунікації між авторами рецептів та їхніми підписниками (див. рис. 3.6).

```php
class EventServiceProvider extends ServiceProvider
{
    /**
     * The event to listener mappings for the application.
     *
     * @var array<class-string, array<int, class-string>>
     */
    no usages
    protected $listen = [
        Registered::class => [
            SendEmailVerificationNotification::class,
        ],
        RecipePublished::class => [
            SendRecipePublishedNotification::class,
        ],
    ];
```

Рисунок 3.6 – Клас для обробки подій (виконано самостійно)

Коли автор викладає новий рецепт, всі користувачі, які підписані на нього, автоматично отримують повідомлення електронною поштою (див. рис. 3.7).

```php
class SendRecipePublishedNotification
{
    new *
    public function handle(RecipePublished $event): void
    {
        $subscribers = Subscriber::where( column: 'author_id', $event->recipe->user_id)->get();

        foreach ($subscribers as $subscriber) {
            Mail::to($subscriber->user->email)->send(new RecipePublishedEmail($event->recipe));
        }
    }
```

Рисунок 3.7 – Реалізація відправки повідомлень (виконано самостійно)

Це забезпечує активне залучення користувачів та підтримує їхній інтерес до платформи, стимулюючи регулярні відвідування та взаємодію з контентом. Така інтеграція сприяє більш персоналізованому користувацькому досвіду, дозволяючи користувачам отримувати оновлення від улюблених авторів безпосередньо на свою електронну адресу, що робить взаємодію з платформою більш зручною та приємною. Система подій Laravel також допомагає зменшити затримки у відповіді сервера, оптимізувавши відправку сповіщень без додаткового навантаження.

Ця функціональність робить платформу не лише інструментом для пошуку та ділення рецептами, а й міцним засобом для підтримки спільноти, де кожен користувач відчуває особистий зв'язок з контентом, що вони вибирають слідувати.

Зберігання та обробка даних відбувається за допомогою системи управління базами даних MySQL, що забезпечує надійність і швидкий доступ до великих обсягів інформації. Використання реляційної бази даних дозволяє ефективно керувати даними, забезпечуючи їх цілісність та безпечне зберігання, а також оптимізувати обробку запитів через раціонально сплановані індекси та запити;

4 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ

4.1 Опис вибору технологій

Для розробки серверної частини кулінарної платформи було обрано мову програмування PHP та фреймворк Laravel. PHP є широко використовуваною мовою, яка відома своєю гнучкістю та потужними можливостями для створення динамічних веб-сторінок. Laravel, зі свого боку, є одним з найпопулярніших PHP фреймворків, що забезпечує швидку розробку завдяки своїй модульності та ряду передбачених функцій, таких як міграції баз даних, сидінг, Eloquent ORM для роботи з базою даних.

PHP, мова програмування, яку було обрано для серверної частини кулінарної платформи, має багаторічну історію успіхів у веб-розробці. Створена у 1995 році Расмусом Лердорфом, PHP орієнтована на генерацію динамічного контенту на веб-сторінках. З того часу PHP еволюціонувала з простого скриптового мови до повноцінної мови програмування, яка підтримує об'єктно-орієнтоване програмування, функціональне програмування та процедурний стиль.

Laravel, фреймворк, що став справжнім відкриттям у світі PHP розробки, був створений Тейлором Отвеллом у 2011 році з метою надати розробникам кращий інструмент для побудови веб-додатків. Цей фреймворк швидко здобув популярність завдяки своїй здатності забезпечувати чистоту коду, швидкість розробки та ефективність виконання. Однією з ключових особливостей Laravel є використання архітектури Model-View-Controller (MVC), яка сприяє відділенню бізнес-логіки від користувацького інтерфейсу.

MVC архітектура у Laravel сприяє ефективній організації коду за допомогою відокремлення даних моделі (Model), користувацького інтерфейсу (View) та бізнес-логіки додатку (Controller). Це розподіл дозволяє розробникам легше управляти комплексними програмними проектами, роблячи код легшим для розуміння та підтримки. Завдяки цьому розділенню, команда розробників може працювати ефективніше, зосереджуючись на конкретних аспектах додатку без ризику створення зав'язаних один на одному завдань.

Laravel фокусується на модульності та повторному використанні коду, надаючи розробникам гнучкість, необхідну для створення застосунків, які легко масштабуються та адаптуються до змінних вимог бізнесу. Ця платформа не тільки прискорює розробку, але й скорочує час, необхідний для впровадження нових функцій, що робить Laravel переважним вибором для багатьох проектів, які прагнуть швидко вийти на ринок зі стабільними та ефективними рішеннями.

Система маршрутизації в Laravel (див. рис. 4.1) дозволяє розробникам легко та ефективно визначати шляхи (маршрути), по яких користувацькі запити досягають відповідних контролерів, що управляють логікою додатка. Вона підтримує засоби для визначення простих маршрутів, маршрутів з параметрами, групування маршрутів за атрибутами, такими як середній програмний шар (middleware), простори імен, і переадресації. Це забезпечує чистоту і організацію в обробці HTTP запитів і відповідей, а також дозволяє легко інтегрувати безпеку, як-то перевірку аутентифікації чи прав доступу, безпосередньо в маршрути. Така гнучка система маршрутизації робить Laravel винятково потужним інструментом для сучасної веб-розробки.

```php
Route::controller( controller: AuthorizationController::class)->group(function () {
    Route::post( uri: '/auth/register', action: 'register');
    Route::post( uri: '/auth/login', action: 'login');
    Route::post( uri: '/auth/logout', action: 'logout')->middleware( middleware: 'auth:sanctum');
});

Route::group(['middleware' => ['auth:sanctum']],  function() {

    Route::controller( controller: SubscriberController::class)->group(function () {
        Route::post( uri: '/subscribe', action: 'subscribe');
        Route::post( uri: '/unsubscribe', action: 'unsubscribe');
    });

    Route::controller( controller: RecipeController::class)->group(function () {
        Route::get( uri: '/recipes', action: 'index');
        Route::get( uri: '/recipes/{id}', action: 'show');
        Route::post( uri: '/recipes', action: 'store');
        Route::post( uri: '/recipes/{id}/update', action: 'update');
        Route::delete( uri: '/recipes/{id}', action: 'destroy');
    });
```

Рисунок 4.1 – Система маршрутизації Laravel (виконано самостійно)

Eloquent ORM (Object-Relational Mapping) є одним із ключових компонентів фреймворку Laravel, що дозволяє розробникам працювати з базою даних у об'єктно-орієнтованому стилі. Цей інструмент є чудовим прикладом реалізації шаблону проектування Active Record, де кожен об'єкт моделі в Laravel відповідає запису в базі даних, а клас моделі відповідає таблиці.

Eloquent надає зручний, інтуїтивно зрозумілий API для виконання запитів до бази даних, що робить його винятково потужним у поєднанні з лаконічністю PHP. Розробники можуть виконувати складні запити до бази даних, використовуючи простий і виразний синтаксис, не турбуючись про безпосереднє написання SQL-коду. Eloquent дозволяє виконувати створення, читання, оновлення та видалення (CRUD) операцій дуже легко, об'єднуючи зручність Ruby on Rails або Django ORM з продуктивністю PHP.

Однією з головних переваг Eloquent є його спроможність до глибокої інтеграції з Laravel, зокрема автоматичне управління відносинами між базами даних (див. рис. 4.2). Розробники можуть легко визначати відносини як методи всередині моделей Eloquent, що дозволяє з легкістю реалізовувати запити між пов'язаними таблицями без потреби звертатися до складних JOIN операцій SQL. Така особливість робить Eloquent ідеальним вибором для веб-додатків, де потрібно часто звертатися до пов'язаних даних.

```
1 usage   Maksym-Kostenko-Leasoft
protected function createRecipe(RecipeData $recipeData): Recipe
{
    $recipeDataArray = $recipeData->toArray();

    if (!empty($recipeData->coverPhoto)) {
        $filename = $this->uploadFile($recipeData->coverPhoto, path: 'recipe_cover_photos');
        $recipeDataArray['cover_photo'] = $filename;
    }

    return $this->recipeModel->create($recipeDataArray);
}

1 usage   Maksym-Kostenko-Leasoft
protected function createIngredients(int $recipeId, array $ingredients): void
{
    foreach ($ingredients as $ingredientData) {
        $this->ingredientModel->create(array_merge($ingredientData->toArray(), ['recipe_id' => $recipeId]));
    }
}
```

Рисунок 4.2 – Створення рецепту використовуючи Eloquent ORM (виконано самостійно)

З іншого боку, використання Eloquent може спричинити певні труднощі, коли мова йде про дуже великі та складні запити до бази даних або коли потрібно дуже детально оптимізувати виконання запитів заради максимальної продуктивності. В таких випадках розробникам може знадобитися більш прямий контроль над SQL-кодом, що може вимагати використання не Eloquent, а більш низькорівневих компонентів Laravel або навіть чистого SQL. Однак для більшості додатків середнього та невеликого масштабу Eloquent пропонує ідеальне співвідношення продуктивності та простоти використання.

У проекті використовується система валідації (див. рис. 4.3) даних для забезпечення точності та повноти інформації, яка подається через API. Ця система допомагає переконатися, що всі дані, введені користувачами, відповідають визначеним критеріям, перш ніж обробляти їх далі в системі. Вона включає правила, які забезпечують обов'язковість та формат поля, такі як максимальна довжина тексту, чи поле є числовим та перевірка на мінімальне та максимальне значення.

```php
public function rules(): array
{
    return [
        'title' => 'required|string|max:255',
        'description' => 'required|string|max:1000',
        'cooking_time' => 'required|integer|min:1',
        'difficulty_level' => 'required|in:easy,medium,hard',
        'portions' => 'required|integer|min:1',
        'is_approved' => 'required|boolean',
        'is_published' => 'required|boolean',
        'cover_photo' => 'required|file|max:2048',
        'ingredients' => 'required|array|min:1',
        'ingredients.*.title' => 'required|string|max:255',
        'ingredients.*.measure' => 'required|string|max:100',
        'ingredients.*.count' => 'required|integer|min:1',
        'steps' => 'required|array|min:1',
        'steps.*.description' => 'required|string|max:1000',
        'steps.*.photos' => 'array|nullable',
        'steps.*.photos.*' => 'file|mimes:jpg,jpeg,png|max:2048',
        'tags.*' => 'nullable|exists:tags,id'
    ];
}
```

Рисунок 4.3 – Приклад валідації для рецепту (виконано самостійно)

MySQL, обрана як система управління базами даних, вирізняється своєю високою надійністю та ефективністю у роботі з великими обсягами даних, що робить її ідеальною для застосунків, які вимагають швидкого доступу до даних та їх надійного зберігання. Завдяки своїй широкій підтримці та великій спільноті, MySQL є однією з найпопулярніших реляційних баз даних у світі, що постійно розвивається та оптимізується для роботи з сучасними веб-технологіями. Її здатність ефективно обробляти транзакції, забезпечувати цілісність даних та легко інтегруватися з різними програмними мовами і фреймворками робить MySQL незамінним інструментом для будь-якої великої системи, особливо для кулінарної платформи, де потреба в швидкому доступі до рецептів та користувацької інформації є критичною.

MySQL є однією з найстаріших та найбільш надійних систем управління базами даних, яка була розроблена у 1995 році шведськими програмістами Майклом Віденіусом та Девідом Аксмарком. Від самого початку MySQL призначалася для швидкого створення та обслуговування баз даних, що дозволяло їй швидко знайти широке використання в розробці веб-сайтів і веб-додатків.

Однією з найбільш значущих переваг MySQL є її висока продуктивність при роботі з великими обсягами даних. Це досягається завдяки ефективним механізмам індексування, які значно прискорюють пошук та сортування даних. Крім того, MySQL підтримує різноманітні типи табличного зберігання, включаючи InnoDB, який забезпечує підтримку транзакцій, атомарність, консистентність, ізоляцію та стійкість (ACID), що робить її надійною вибором для систем, де критично важливі ці характеристики.

У проекті кулінарної платформи активно використовуються патерни проектування, що значно підвищує ефективність розробки та гнучкість системи.

Один з таких патернів — фабрика (див. рис. 4.4), який застосовується для створення об'єктів провайдера платежів. Цей паттерн дозволяє системі динамічно вибирати метод оплати залежно від параметрів, що надходять у запиті, тим самим спрощуючи інтеграцію різноманітних способів обробки платежів та забезпечуючи легше масштабування платформи.

Використання паттернів проектування в коді не тільки підвищує якість продукту, але й забезпечує його надійність, відтворюваність та легкість обслуговування. Це стає ключовим фактором успіху у розробці сучасних програмних систем, де важливо швидко адаптуватися до змінних вимог користувачів і ринкових умов.

```php
public function createSubscription(CreateSubscriptionRequest $request) : JsonResponse
{
    $data = $this->extractRequestData($request);

    try {
        if (!is_null($this->subscriptionRepository->getActiveSubscription(Auth::id()))) {
            return throw new \Exception(__( key: "You already have an active subscription!"));
        }
        $paymentProvider = PaymentProviderFactory::create($data['paymentMethod']);
        $subscriptionData = $paymentProvider->createSubscription($data['planId']);

    } catch (\Exception $e) {
        Log::error( message: 'Payment error: ' . $e->getMessage());
        return $this->respondError($e->getMessage());
    }

    return $this->respondWithSuccess([
        'message' => __( key: 'Subscription created successfully'),
        'data' => $subscriptionData
    ]);
}
```

Рисунок 4.4 – Використання патерну Фабричний метод (виконано самостійно)

Під час розробки кулінарної платформи велике значення мало використання інструментів для тестування API, таких як Postman, який дозволяє виконувати запити до сервера, імітуючи роботу клієнтської частини, що сприяє перевірці та налагодженню функціональності API перед її впровадженням в живу систему.

На рисунку 4.5 демонструється використання Postman для створення рецепту в системі. Postman допомагає переконатися, що кожен аспект запиту відповідає очікуванням і специфікаціям API, що забезпечує якість та стабільність роботи платформи.

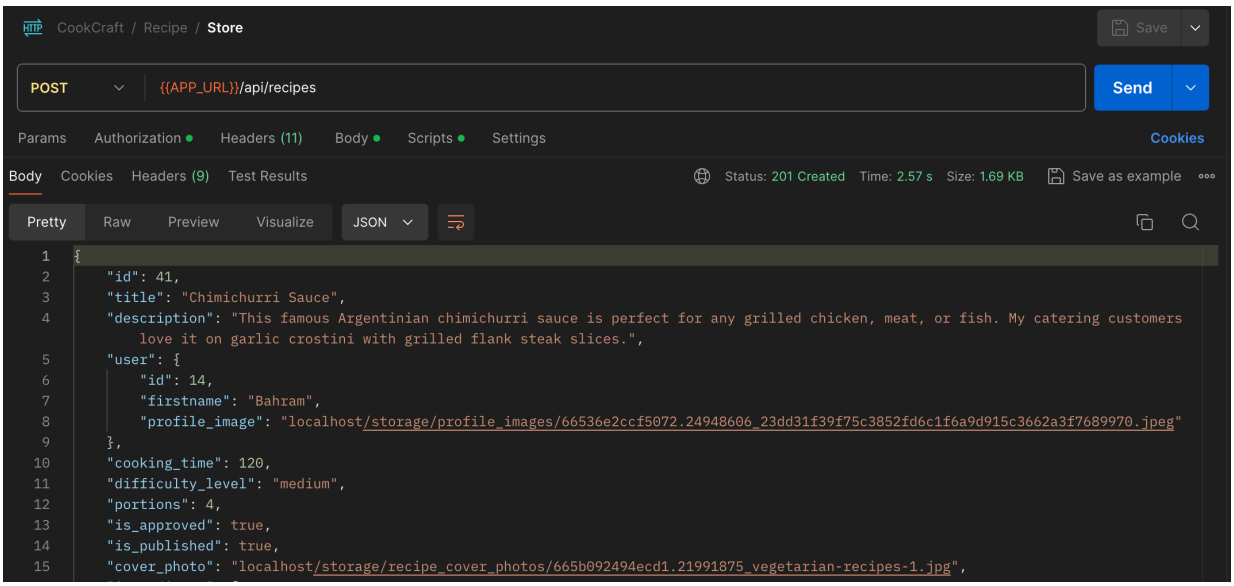

Рисунок 4.5 – Виконання запиту на створення рецепту через Postman (виконано самостійно)

В рамках кулінарної платформи активно використовується система подій Laravel для оповіщення користувачів. Коли автор публікує новий рецепт, подія RecipePublished (див. рис. 4.6) автоматично спрацьовує, ініціюючи розсилку електронних листів усім підписникам цього автора. Така модель забезпечує високу взаємодію та заохочує користувачів до активнішого використання платформи, завдяки своєчасним оновленням та персоналізованому контенту.

```php
public function handle(RecipePublished $event): void
{
    $subscribers = Subscriber::where('author_id', $event->recipe->user_id)->get();

    foreach ($subscribers as $subscriber) {
        Mail::raw( text: "A new recipe titled '{$event->recipe->title}' has been published by {$event->recipe->user->name}.
            $message->to($subscriber->user->email)
                ->subject('New Recipe Published');
        });
    }
}
```

Рисунок 4.6 – Виконання події на відправку підписникам листа стосовно нового рецепту від їх автора (виконано самостійно)

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Під час розробки було створено набір тестових сценаріїв. Тестування здійснювалося за методологією BlackBox. Кожен тестовий сценарій, або тестовий випадок, включає назву, опис тесту, компонент, який був протестований, пріоритет тестового випадку, критичність, кроки, необхідні для відтворення тесту, очікуваний та фактичний результати. Пріоритет тестового випадку визначається найвищим значенням P1 і найнижчим - P4. Критичність потенційної помилки оцінюється від S1 (некритичний) до S4 (критичний). У таблиці 5.1 наведено список створених тестових сценаріїв.

Таблиця 5.1 – Список тестових випадків для кулінарної системи

Тест № 1	
Назва тесту:	Валідація вхідних даних для створення рецепту
Опис тесту:	Перевірка валідації вхідних даних при створенні нового рецепту, включаючи обов'язкові поля та формат даних.
Компонент системи:	API створення рецепту
Пріоритет:	P2
Критичність:	S3
Кроки відтворення:	Викликати API для створення рецепту з некоректними даними. Передати запит без заповнення обов'язкових полів. Передати запит із неправильним форматом даних (наприклад, текст замість числових значень).
Очікуваний результат:	Система поверне помилку валідації з відповідним повідомленням для кожного некоректного поля
Фактичний результат:	Система прийняла некоректні дані без помилок валідації.
Результат:	Знайдено баг, виправлено 05.05.2024

Продовження таблиці 5.1

Тест № 2	
Назва тесту:	Перевірка функціональності відправки повідомлення підписникам автора
Опис тесту:	Перевірити відправку повідомлення користувачу який підписан на автора при додаванні автором нового рецепту.
Компонент системи:	API обробки подій
Пріоритет:	P1
Критичність:	S4
Кроки відтворення:	Викликати API для авторизації у систему. Підписатися на автора. Викликати API для авторизації у систему від імені автора. Додати новий рецепт та зробити його публічним.
Очікуваний результат:	Система опрацює подію додавання нового рецепту та відправить лист на пошту всім підписникам автора.
Фактичний результат:	Система повернула помилку підключення до поштового сервісу.
Результат:	Знайдено баг, виправлено 15.05.2024

У розробленому API кулінарної платформи передбачена вбудована система валідації даних, що забезпечує перевірку вхідних даних на коректність перед їх обробкою (див. рис. 5.1). Це включає перевірку на наявність необхідних полів, валідність електронних адрес та правильність інших введених даних, як то стать користувача. Використання цієї системи дозволяє уникнути помилок у обробці даних та забезпечує надійність і безпеку при роботі з користувацькою інформацією. Валідація в API є важливим компонентом для забезпечення цілісності даних і відповідає загальним стандартам якості в розробці сучасних веб-систем.

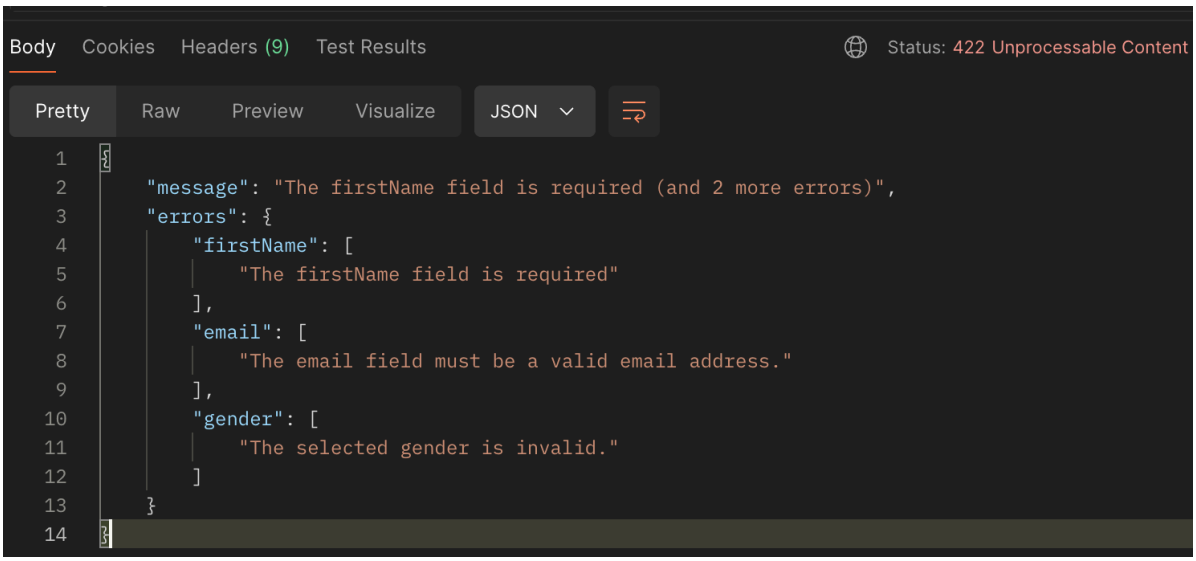

Рисунок 5.1 – Приклад валідації на сервері (виконано самостійно)

У розробленому API кулінарної платформи активно використовуються HTTP-відповіді для індикації результатів валідації даних. При виникненні помилок у вхідних даних, система надсилає відповіді з відповідними статусними кодами HTTP, такими як 422 Unprocessable Entity. Це допомагає клієнтам API чітко розуміти, що запит містить помилки, які потребують виправлення. Використання таких відповідей не тільки підвищує безпеку та надійність системи, але й сприяє кращому досвіду користувачів, оскільки вони отримують зрозумілу інформацію про помилки в запитах.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Аналіз та планування:

- оцінка вимог користувачів до кулінарної платформи та адаптація під їхні потреби;
- дослідження необхідних функцій та встановлення технічних специфікацій;
- розробка стратегії реалізації проекту та визначення критеріїв успіху;
- організація робочого процесу та вироблення чіткого часового плану.

Розробка програмного забезпечення:

- конструювання структури баз даних для оптимального збереження всіх потрібних даних;
- програмування логіки роботи системи, забезпечення її масштабованості та ефективності;
- створення зручних і функціональних API для взаємодії клієнтської частини з сервером;
- інтеграція рішень для аутентифікації та забезпечення безпеки даних.

Тестування:

- виконання широкомасштабних тестів для забезпечення стабільності та надійності рішень;
- аналіз відповідності системи встановленим вимогам та очікуванням;
- ідентифікація та виправлення виявлених в ході тестування проблем.

Впровадження:

- оптимізація серверної інфраструктури для запуску та підтримки системи;
- детальне налаштування та конфігурація програмного забезпечення для оптимальної роботи;
- міграція існуючих даних у нову систему і верифікація їх цілісності;
- проведення фінального тестування для гарантії стабільної роботи платформи.

Навчання та підтримка:

– організація тренінгів для користувачів та адміністраторів системи;
– розробка інструкцій та методичних матеріалів для користувачів;
– забезпечення оперативної технічної підтримки та оновлень системи.

Моніторинг та оптимізація:

– встановлення інструментів для моніторингу продуктивності та стабільності;
– періодичний аналіз системи на предмет виявлення можливостей для її удосконалення;
– впровадження змін, що сприяють підвищенню ефективності використання ресурсів.

Під час розробки програмного забезпечення активно використовувалась трекінгова платформа Jira для планування та декомпозиції функціоналу проекту (див. рис 6.1). Задачі планувалися на спринти та мали прив'язку до епіків (див. рис. 6.2) та оцінювались у сторіпоінтах.

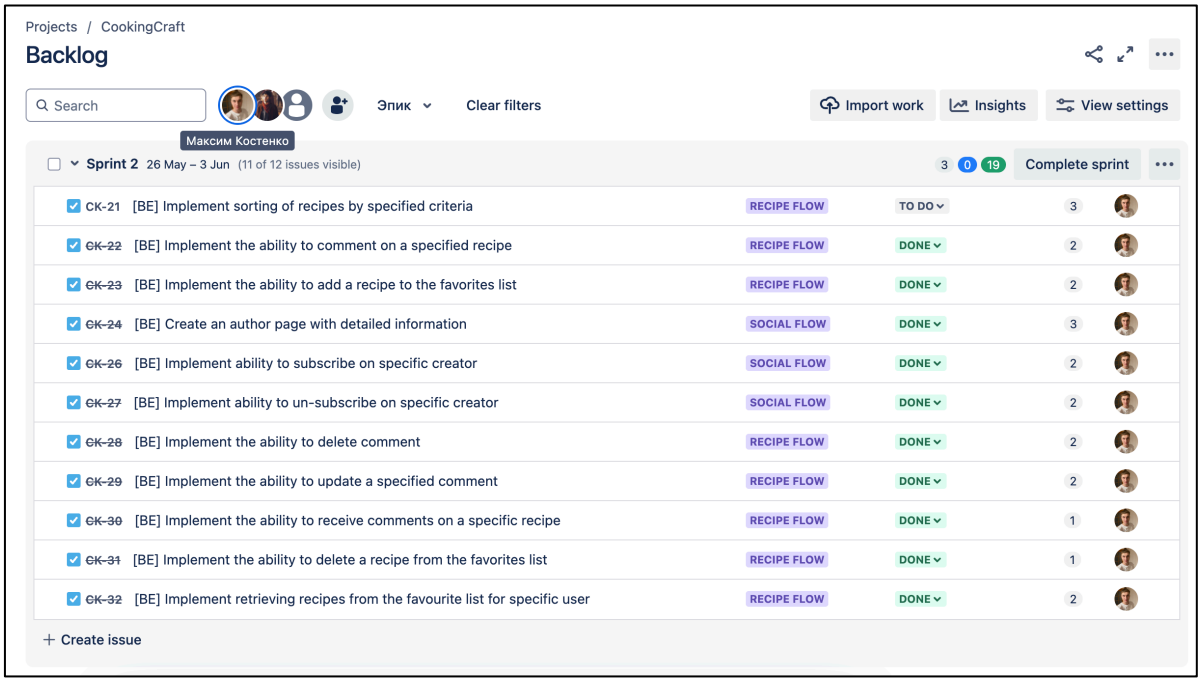

Рисунок 6.1 – Backlog у Jira (виконано самостійно)

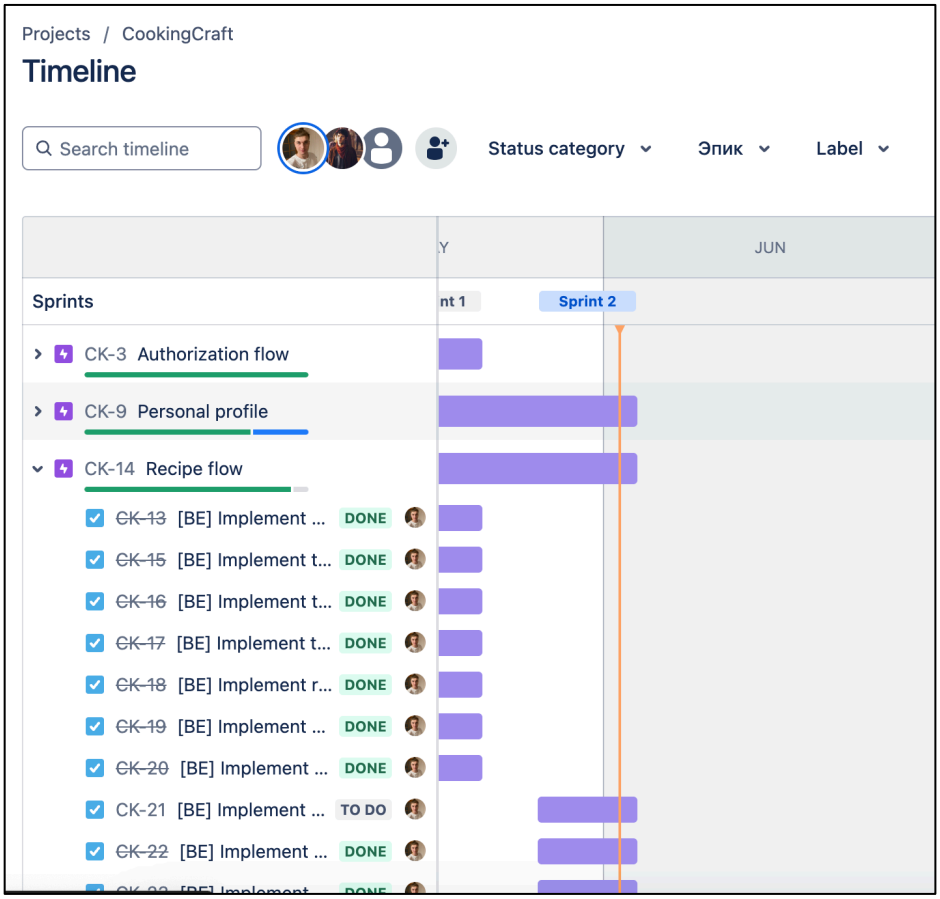

Рисунок 6.2 – Групування задач на функціональні блоки у Jira (виконано самостійно)

Приймав участь у наукові конференції «Цифровізація науки та сучасні тренди її розвитку», опублікував тезу на тему «Патерн Стратегія, як засіб проектування масштабованих систем» та отримав сертифікат який додано у ДОДАТКУ В.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра розроблено комплексне кулінарне програмне забезпечення, що охоплює взаємодію між сервером та клієнтом, забезпечуючи користувачам доступ до великої кількості рецептів та кулінарних інструкцій.

Мета проекту досягнута повністю, забезпечивши інтуїтивно зрозумілий та легкий доступ до кулінарної інформації, а також можливості персоналізації та взаємодії зі спільнотою.

Під час реалізації проекту було здійснено глибокий аналіз кулінарної галузі та існуючих рішень, виявлено недоліки поточних систем та визначено основні вимоги до нової платформи. Цей аналіз допоміг встановити напрямки для покращення та розвитку програмного забезпечення.

Для реалізації серверної частини обрано використання PHP та фреймворку Laravel, що забезпечили модульність та гнучкість у розробці. Laravel як потужний фреймворк сприяв швидкій розробці та високій продуктивності бекенду. Для зберігання даних використано реляційну базу даних MySQL, яка забезпечила надійне та ефективне зберігання великих обсягів інформації, необхідних для кулінарної платформи. Реалізація клієнтської частини здійснювалася за допомогою сучасних технологій веб-розробки, забезпечуючи користувачам гнучкий та зручний інтерфейс.

Завдяки впровадженню гнучкої системи управління контентом, платформа забезпечує адміністраторам потужні інструменти для керування контентом, модерації рецептів та взаємодії з користувачами, що підвищує якість та безпеку вмісту.

Розроблена програмна система не тільки виконує всі заплановані функціональні завдання, але й має високий потенціал для майбутнього масштабування та вдосконалення, відповідаючи сучасним трендам і вимогам користувачів у сфері кулінарії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Yummly – Wikipedia [Електронний ресурс] – URL: https://en.wikipedia.org/wiki/Yummly (дата звернення: 25.04.2024)

2. Allrecipes.com – Wikipedia [Електронний ресурс] – URL: https://en.wikipedia.org/wiki/Allrecipes.com (дата звернення: 26.04.2024)

3. BigOven Review – [Електронний ресурс] – URL: https://www.pcmag.com/reviews/bigoven (дата звернення 20.05.2024 р.)

4. Офіційна документація Laravel. Laravel - PHP Framework. URL: https://laravel.com/docs (дата звернення: 19.04.2024 р.)

5. Redis Documentation. Redis. URL: https://redis.io/documentation (дата звернення: 21.04.2024 р.)

6. Amazon S3 документація. Amazon Simple Storage Service (S3) — Scalable Storage in the Cloud. URL: https://aws.amazon.com/s3/ (дата звернення: 19.04.2024 р.)

7. Офіційна документація MySQL. MySQL Database Service Guide. URL: https://dev.mysql.com/doc/ (дата звернення: 19.04.2024 р.)

8. REST API Principles | A Comprehensive Overview URL: https://restfulapi.net/ (дата звернення: 05.04.2024 р.)

9. Алгоритми і структура даних: Навчальний посібник / В.М.Ткачук. - Івано-Франківськ : Видавництво Прикарпатського національного університету імені Василя Стефаника, 2016. 286 с.

10. Лавріщева К. М. Програмна інженерія / К. М. Лавріщева. – К. : Академперіодика, 2008. – 319 с.

11. Масштабована веб-архітектура та розподілені системи / Кейт Мацудаїра. – O'Reilly Media, 2011.

12. Мистецтво планування ємності: масштабування веб-ресурсів / Джон Оллспо. – O'Reilly Media, 2018. – 156 с.

13. Кулінарні дані: великі дані в кулінарії та їхній вплив на харчову промисловість / Аміт Зоран, Марсело Коельйо. – Журнал харчових технологій, 2022.

14. Кулінарна еволюція: історія та наука за традиційними стравами. Журнал кулінарних інновацій, 2021.Лавріщева К. М. Програмна інженерія / К. М. Лавріщева. – К. : Академперіодика, 2008. – 319 с.

15. Соціальні медіа і кулінарія: вплив онлайн-спільнот на харчові звички. Журнал цифрової культури, 2023.

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

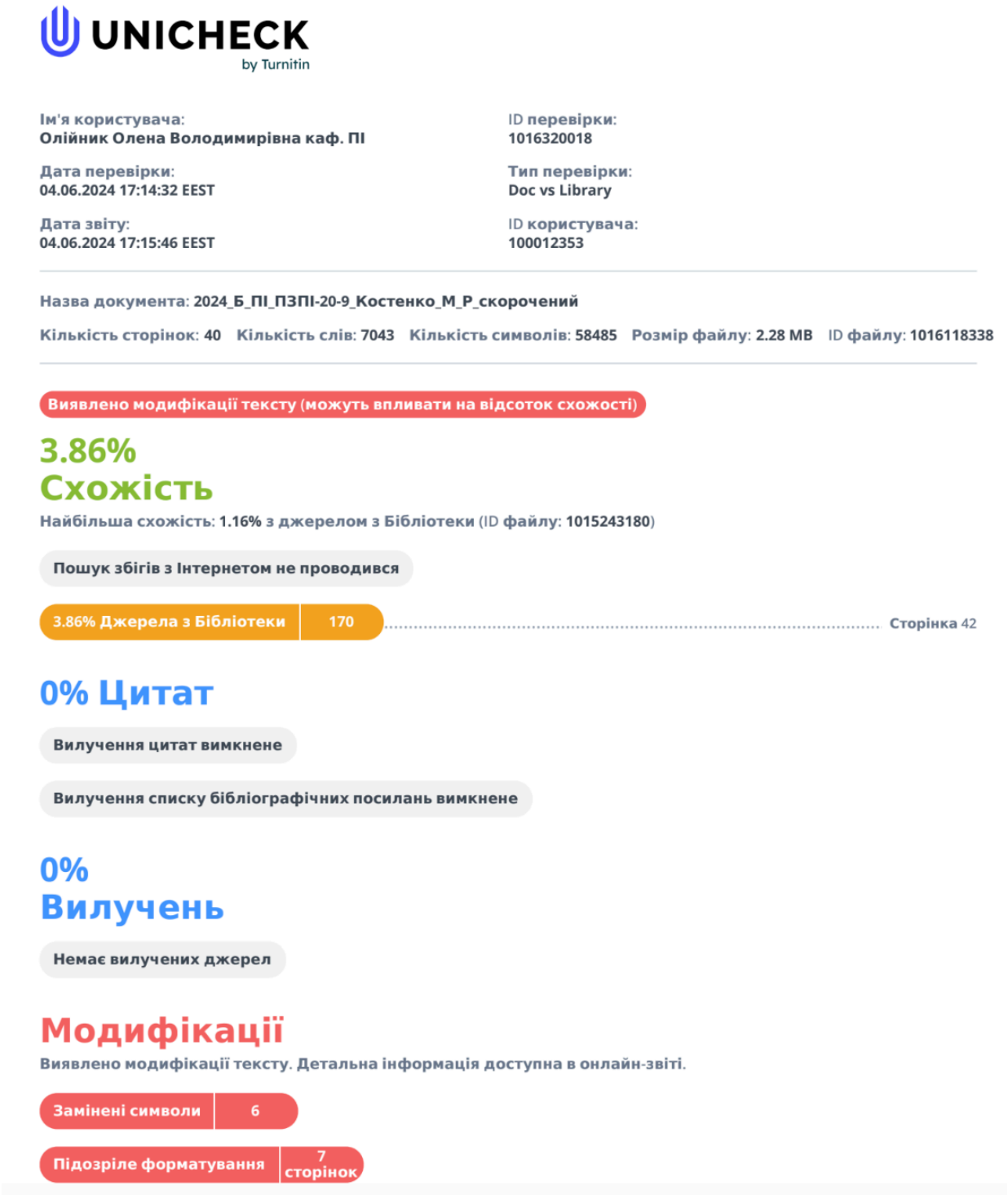

Рисунок А.1 – Перевірка на плагіат

ДОДАТОК Б

Слайди презентації

Рисунок Б.1 – Слайд 1

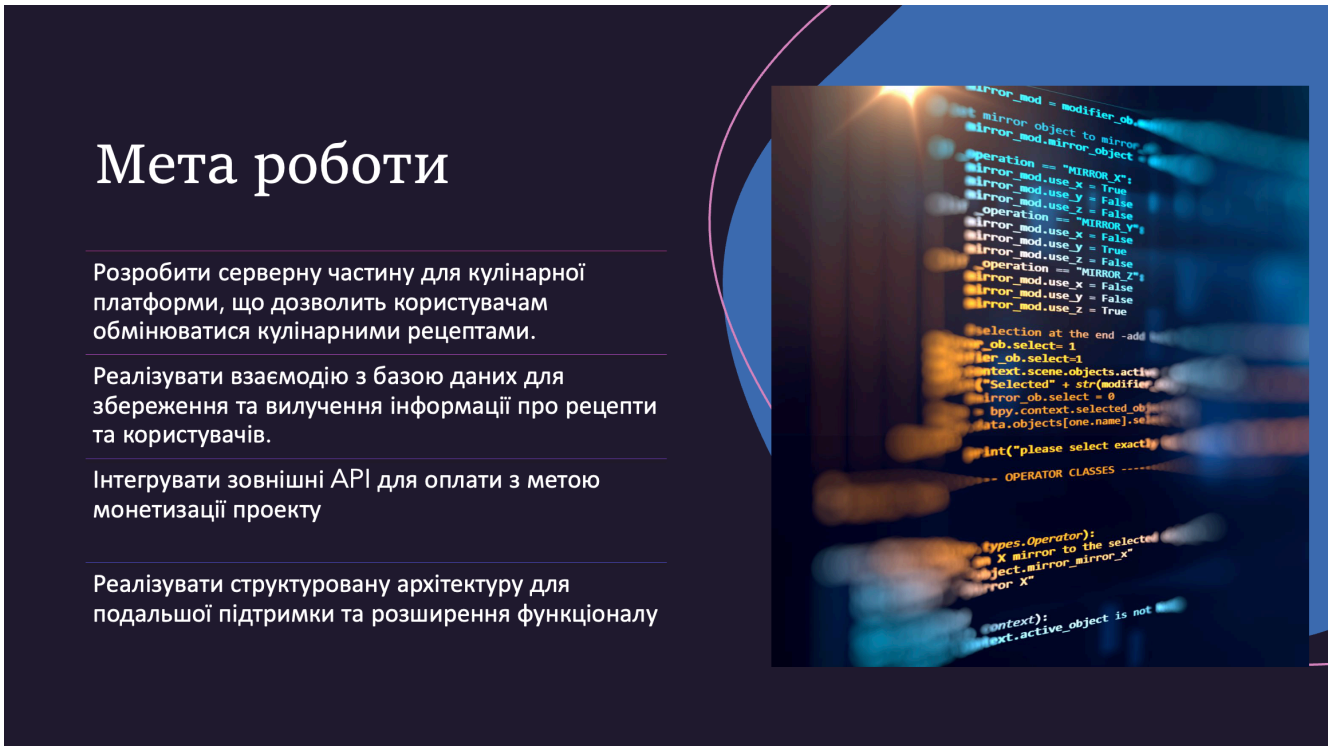

Рисунок Б.2 – Слайд 2

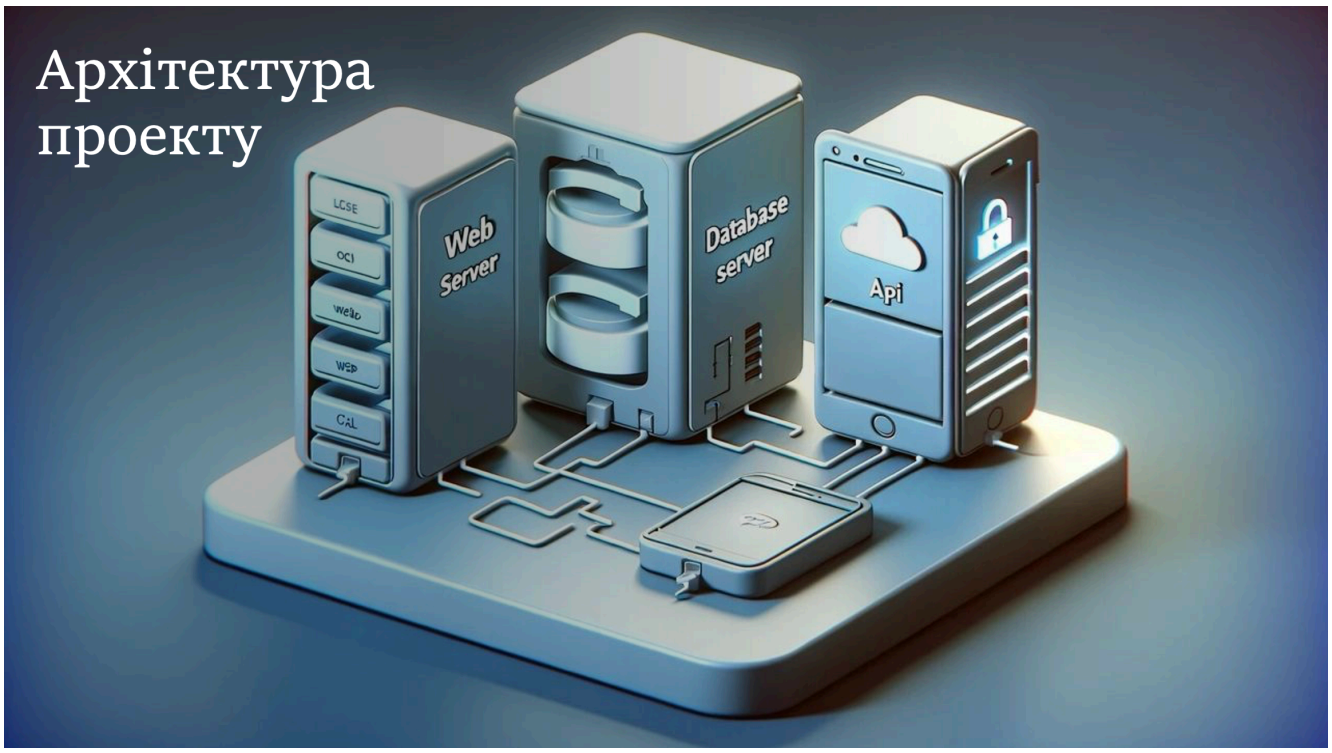

Рисунок Б.3 – Слайд 3

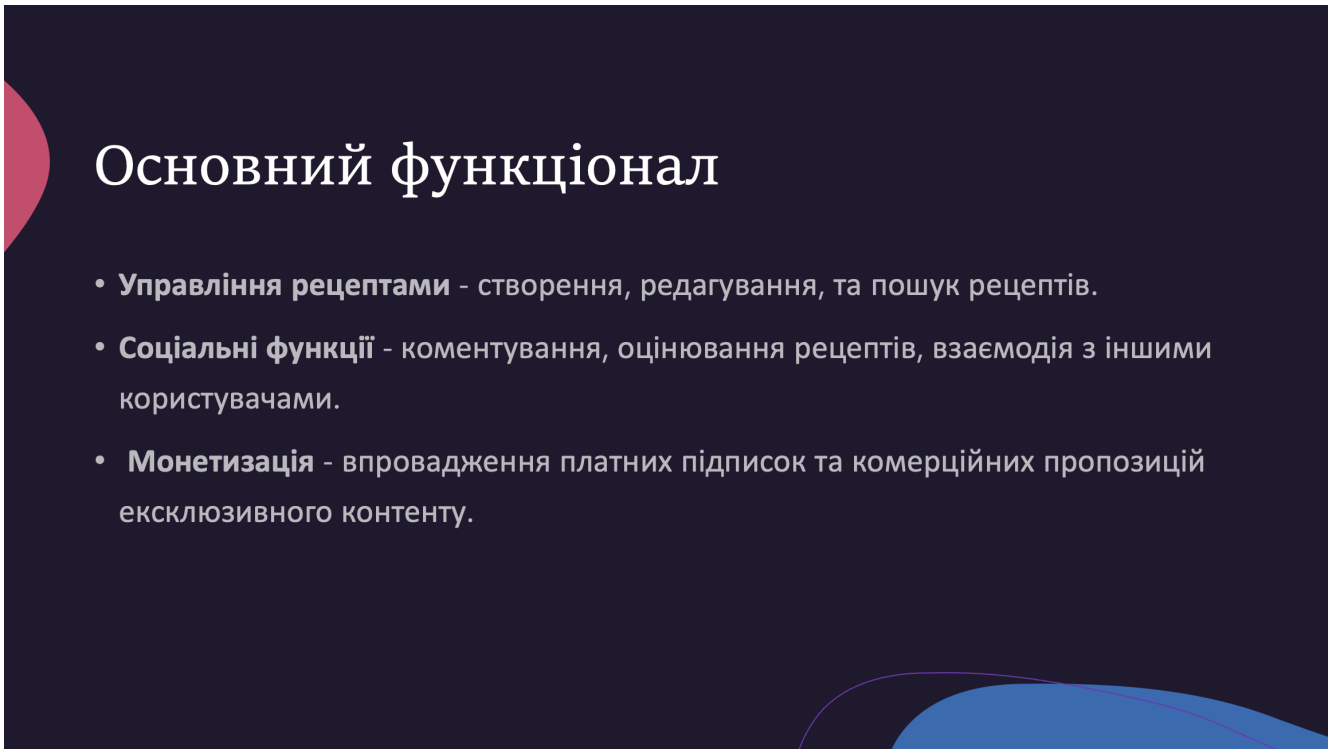

Рисунок Б.4 – Слайд 4

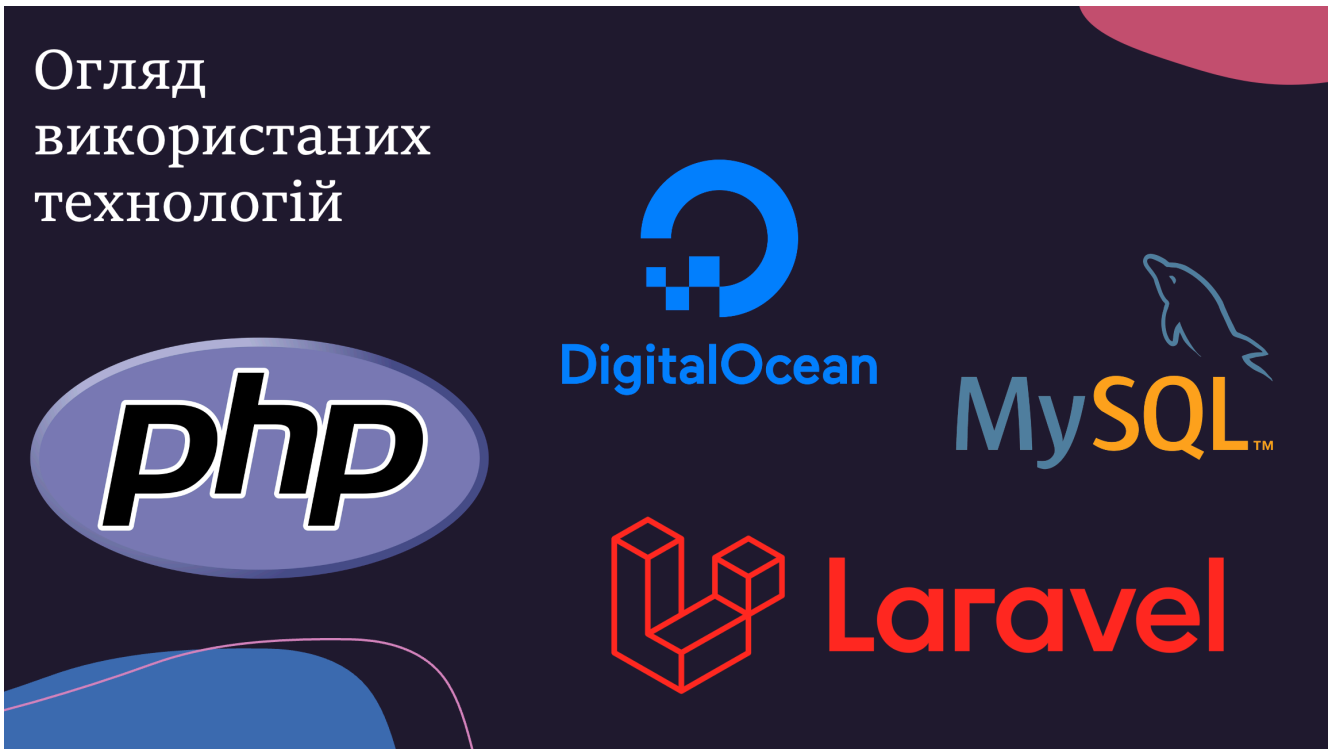

Рисунок Б.5 – Слайд 5

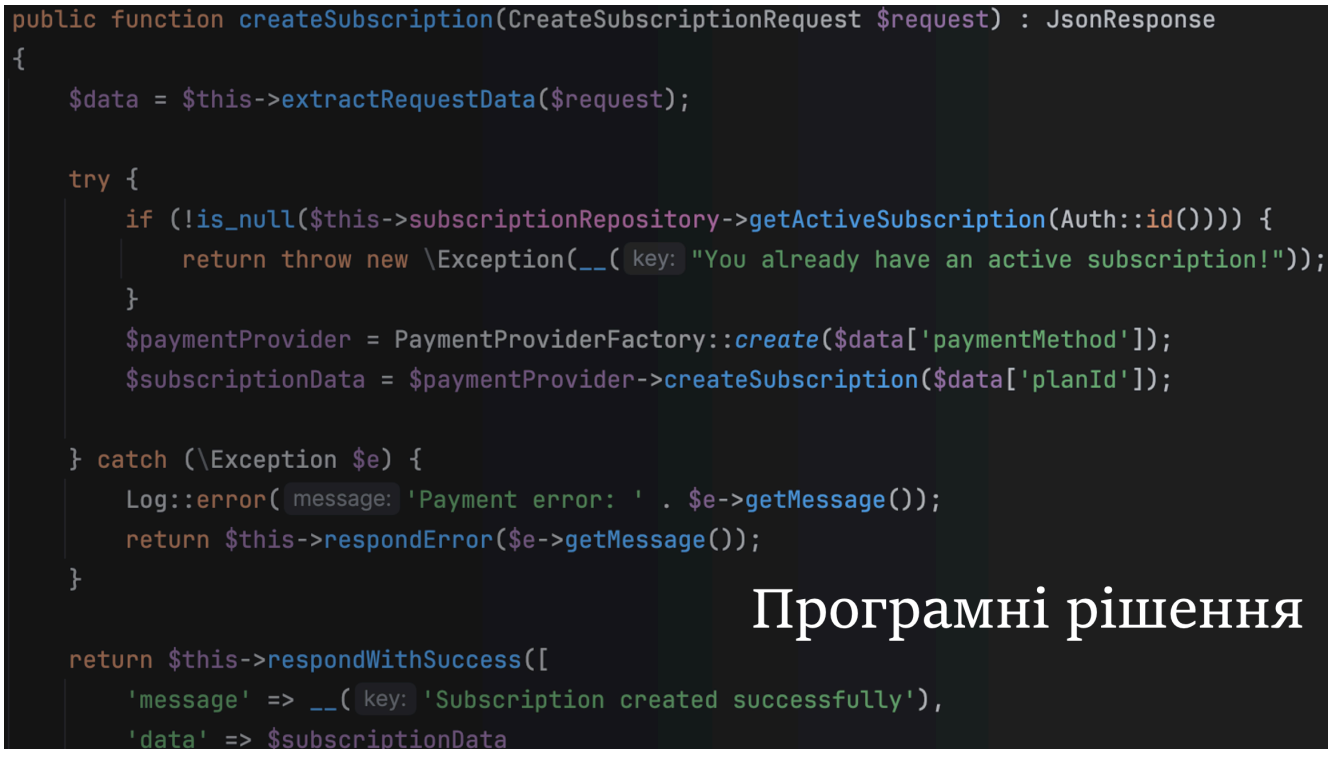

Рисунок Б.6 – Слайд 6

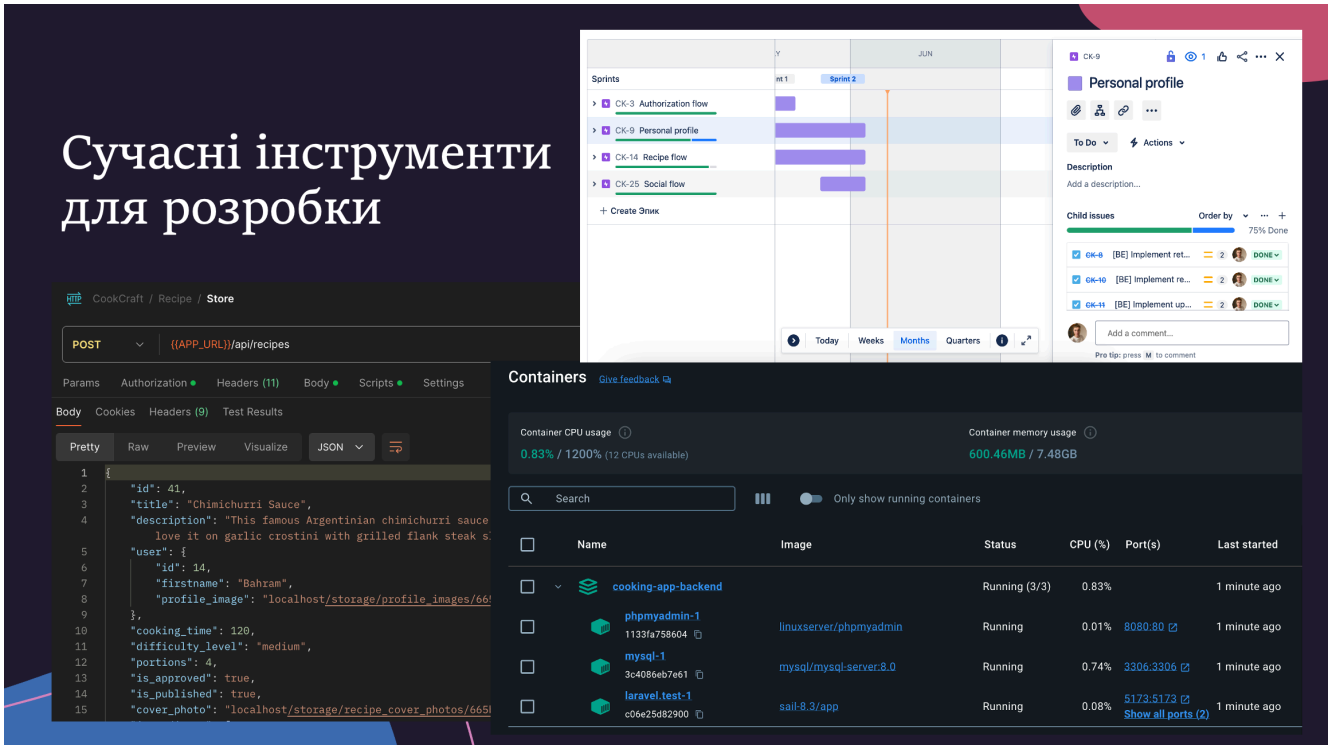

Рисунок Б.7 – Слайд 7

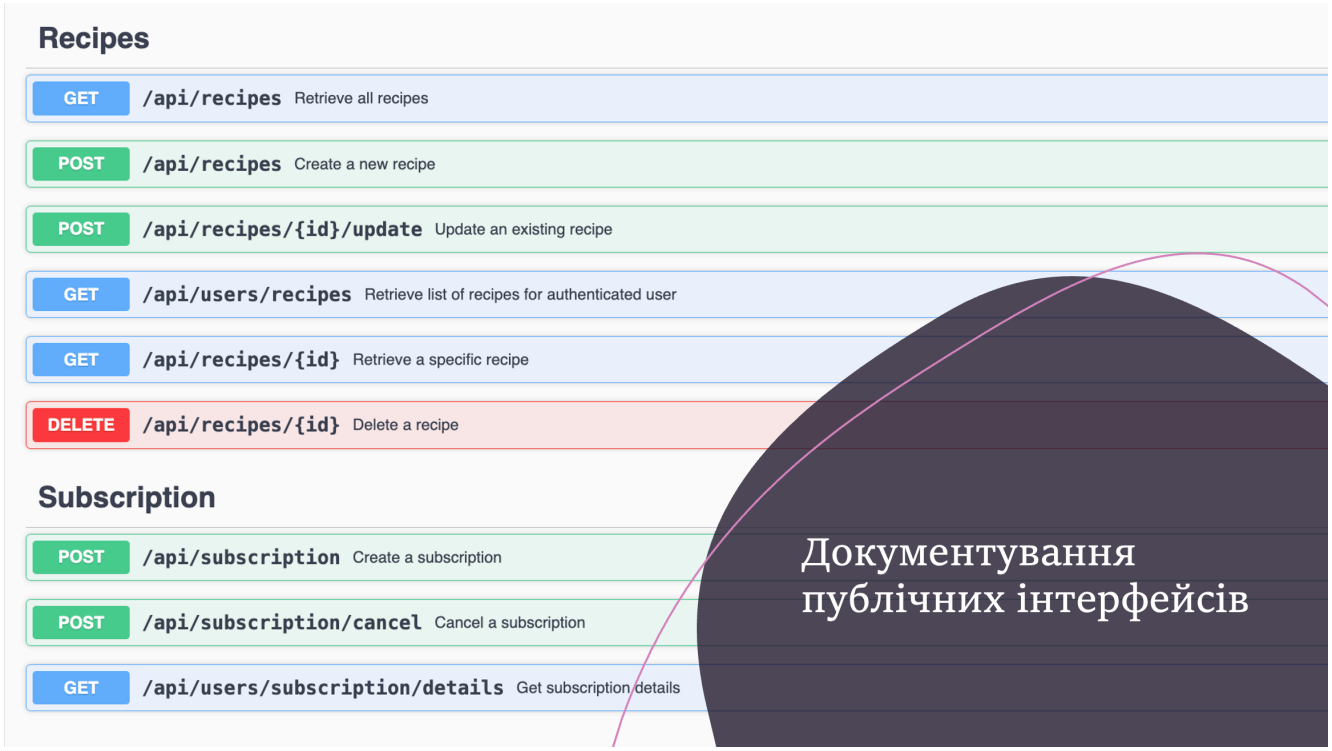

Рисунок Б.8 – Слайд 8

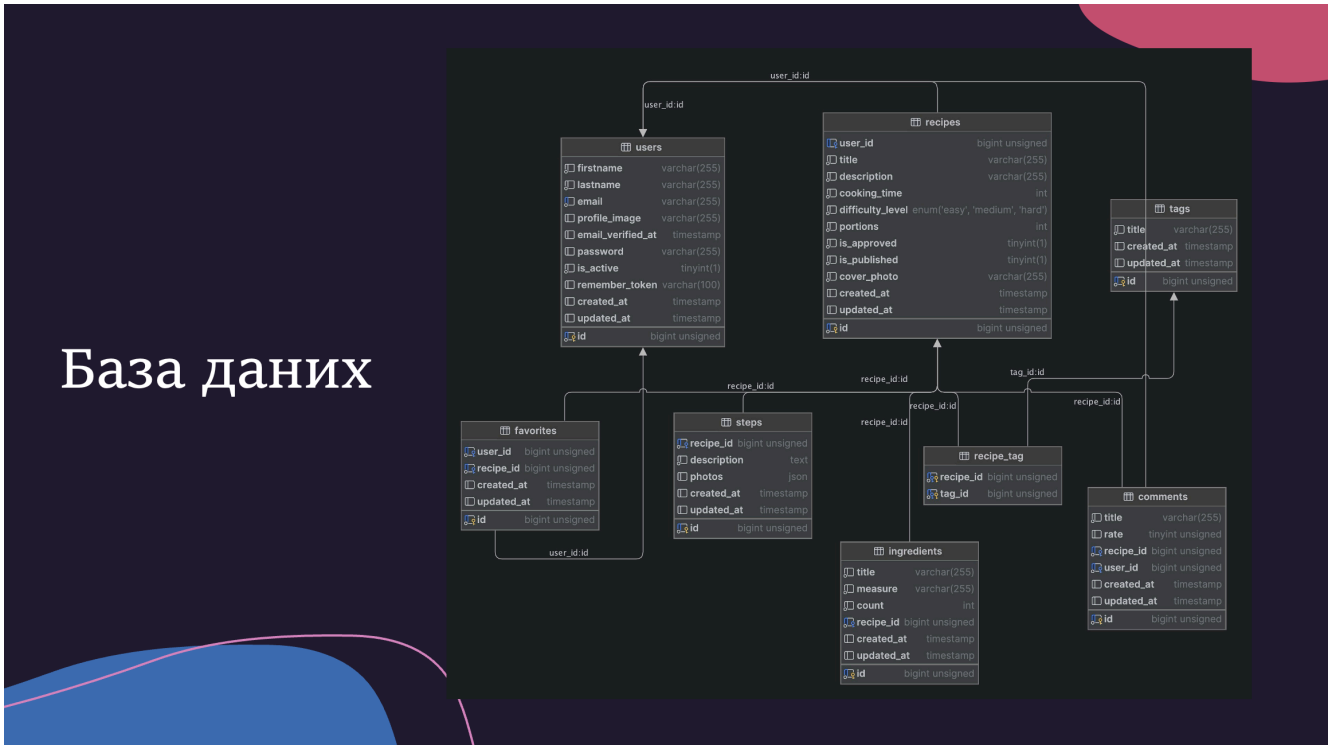

Рисунок Б.9 – Слайд 9

Рисунок Б.10 – Слайд 10

Рисунок Б.11 – Слайд 11

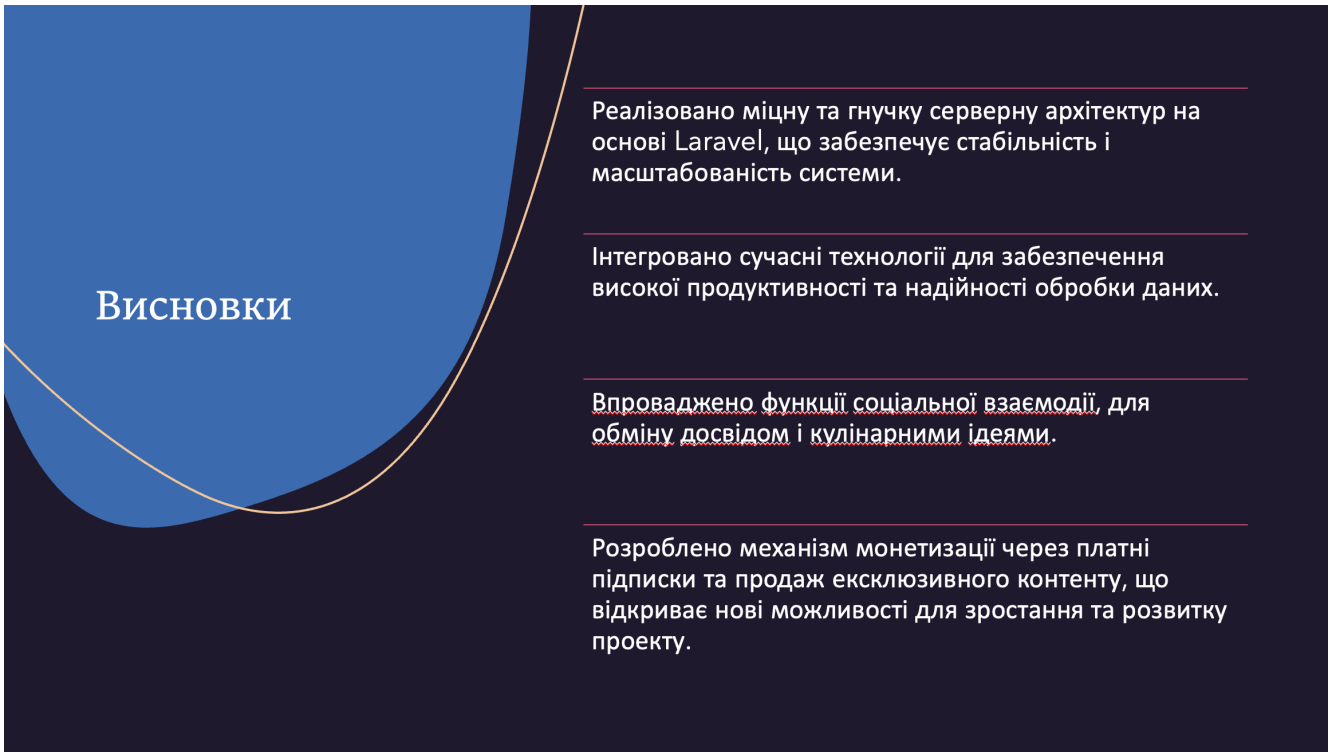

Рисунок Б.12 – Слайд 12

ДОДАТОК В

Наукова публікація

Костенко Максим Родіонович, студент кафедри Програмної інженерії
Харківський національний університет радіоелектроніки, Україна

Науковий керівник: Онищенко Костянтин Георгійович, асистент кафедри Програмної інженерії
Харківський національний університет радіоелектроніки, Україна

ПАТЕРН «СТРАТЕГІЯ» ЯК ЗАСІБ ПРОЕКТУВАННЯ МАСШТАБОВАНИХ СИСТЕМ

В сучасних умовах розробки програмного забезпечення ключовим аспектом є здатність систем адаптуватися до змінних вимог. Це вимагає від професіоналів не лише технічних навичок, а й глибокого розуміння концептів абстракції та модульного проектування. У цьому контексті, знання патернів проектування відіграє важливу роль у розробці гнучких та масштабованих систем. Використання цих патернів дозволяє розробникам ефективно вирішувати типові проектні завдання, що, в свою чергу, сприяє підвищенню якості та надійності програмного продукту.

Розглянемо типове завдання, з яким стикаються розробники - реалізацію системи оплати. Цей процес є ключовим елементом більшості продуктів, оскільки механізми монетизації лежать в основі концептуального проектування будь-якої системи. Початковий етап роботи над такою задачею часто включає дослідження документації специфічних платіжних сервісів, таких як PayPal або Stripe, та їх інтеграцію в бізнес-логіку продукту. Однак цей підхід може призвести до сильної залежності від конкретних реалізацій, що ускладнює адаптацію системи до змінних вимог, наприклад, при додаванні нових методів оплати. Така ситуація вимагає значних зусиль на перевірку та тестування, аби забезпечити сумісність нового функціоналу з уже існуючим кодом, що може бути особливо складним, якщо вся логіка реалізована в межах одного класу.

Тому розглянемо як уникнути такої ситуації та дослідимо патерн Strategy. Патерн Strategy (Стратегія) дозволяє об'єктам не самим виконувати певні дії, а делегувати їх виконання іншим об'єктам, визначеним стратегією. В контексті нашої задачі із системою оплати це означає, що замість прямого виклику методів конкретного провайдера платежів, ми визначаємо загальний інтерфейс для всіх потенційних способів оплати, а конкретна імплементація вибирається в момент виконання в залежності від умов (див. рис 1).

Рис. 1. **Візуальне представлення патерну Strategy**

Рисунок В.1 – Матеріал тез (сторінка 1)

Переваги цього підходу в тому, що він забезпечує гнучкість - легко додавати нові способи оплати, не змінюючи існуючий код. Наприклад, якщо з'являється потреба інтегрувати новий сервіс оплати, достатньо лише розробити нову стратегію, що реалізує інтерфейс способу оплати. Підтримується принцип відкритості/закритості - система відкрита для розширення, але закрита для модифікації. Це означає, що ми можемо додавати нові функціональні можливості, не змінюючи старий код. Також це спрощує тестування та підтримку коду - кожен спосіб оплати може бути протестований окремо від інших частин системи. Також патерн Strategy добре поєднується з патерном Factory (Фабрика), що дозволяє ефективно управляти створенням об'єктів конкретних стратегій. Використання Фабрики спрощує процес вибору та ініціалізації відповідної стратегії оплати, роблячи код більш гнучким і читабельним. Фабрика може бути реалізована як окремий компонент, який на основі вхідних даних (наприклад, типу платіжної системи, вибраної користувачем) виробляє екземпляр потрібного класу, що реалізує інтерфейс.

Необхідно зазначити, що не існує ідеального рішення поставленої задачі, завжди необхідно оцінювати наскільки ваше рішення гарно вбудовується до існуючої системи, наведемо недоліки патерну Strategy:

- Перевантаження клієнтського коду - у деяких випадках клієнтський код може отримати надмірну відповідальність за вибір та керування стратегіями, що може порушити принцип єдиної відповідальності.
- Складність вибору стратегії - необхідність вибору відповідної стратегії може додати додаткову складність у логіку програми, особливо в системах з великою кількістю можливих операцій та стратегій.

Наведемо рекомендації щодо застосування патерну Strategy.

- Чітке визначення інтерфейсів - забезпечте, що всі стратегії мають чітко визначений спільний інтерфейс. Це спрощує заміну стратегій та їх взаємозамінність.
- Мінімізація залежностей - уникайте створення сильних залежностей між конкретними стратегіями та клієнтським кодом. Використовуйте засоби вбудовування залежностей (dependency injection) для зниження зв'язності.
- Обмеження обсягу стратегії - забезпечте, щоб кожна стратегія вирішувала лише одну конкретну задачу або набір тісно пов'язаних задач, дотримуючись принципу єдиної відповідальності.

Дотримуючись цих рекомендацій, можна ефективно впровадити патерн Strategy у систему, підвищивши її гнучкість, масштабованість та легкість підтримки.

Список використаних джерел:

1. Патерн "Стратегія". [Електронний ресурс]. URL: https://refactoring.guru/uk/design-patterns/strategy (Дата звернення 30.03.2024).
2. Gamma, E., Helm, R., Johnson, R., Vlissides, J. "Design Patterns: Elements of Reusable Object-Oriented Software". Addison-Wesley Professional, 1994. – 395 с.
3. Мартін Фаулер. "Patterns of Enterprise Application Architecture". Addison-Wesley Professional, 2002. – 533 с.
4. Принципи SOLID та патерни проектування. [Електронний ресурс]. URL: https://medium.com/@krystsina_kviatkovska/solid-principles-simple-and-easy-explanation-f57d86c47a7f (Дата звернення 30.03.2024).

Рисунок В.2 – Матеріал тез (сторінка 2)

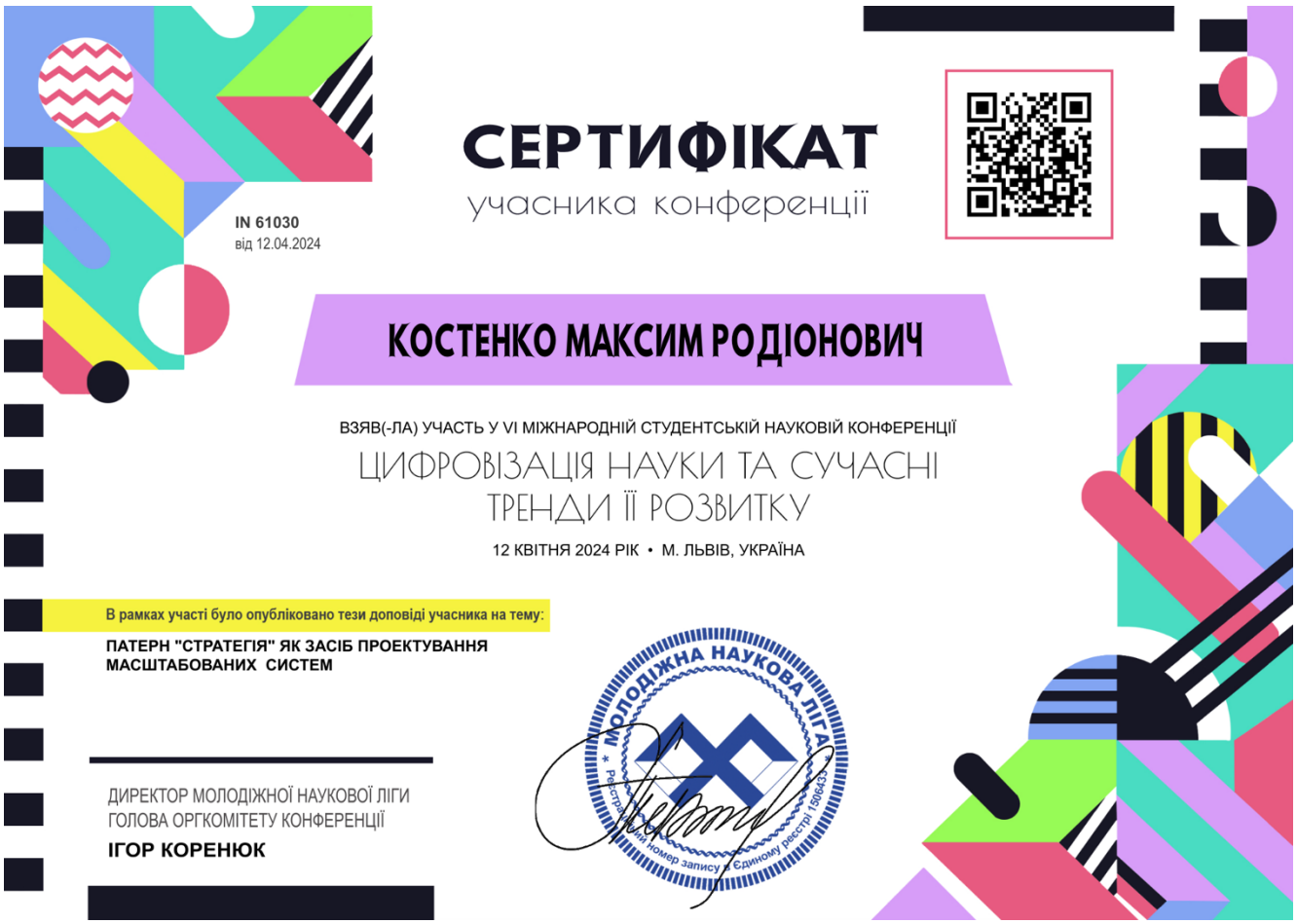

Рисунок В.3 – Сертифікат за участь у науковій конференції

ДОДАТОК Г

Фрагменти програмного коду

```
<?php

use App\Http\Controllers\API\AuthorizationController;
use App\Http\Controllers\API\CommentController;
use App\Http\Controllers\API\FavouriteController;
use App\Http\Controllers\API\PaymentCallbackController;
use App\Http\Controllers\API\PaymentWebhookController;
use App\Http\Controllers\API\PlanController;
use App\Http\Controllers\API\RecipeController;
use App\Http\Controllers\API\SubscriberController;
use App\Http\Controllers\API\SubscriptionController;
use App\Http\Controllers\API\TagController;
use App\Http\Controllers\API\UserController;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/

Route::controller(AuthorizationController::class)->group(function () {
    Route::post('/auth/register', 'register');
    Route::post('/auth/login', 'login');
```

```
    Route::post('/auth/logout', 'logout')->middleware('auth:sanctum');
});

Route::group(['middleware' => ['auth:sanctum']],  function() {

    Route::controller(RecipeController::class)->group(function () {
        Route::get('/recipes', 'index');
        Route::get('/recipes/{id}', 'show');
        Route::post('/recipes', 'store');
        Route::post('/recipes/{id}/update', 'update');
        Route::delete('/recipes/{id}', 'destroy');
    });

    Route::controller(FavouriteController::class)->group(function () {
        Route::get('/users/favourites', 'show');
        Route::post('/favourites', 'store');
        Route::delete('/favorites/recipes/{id}', 'destroy');
    });

    Route::controller(CommentController::class)->group(function () {
        Route::get('/recipes/{id}/comments', 'show');
        Route::post('/comments', 'store');
        Route::post('/comments/{id}/update', 'update');
        Route::delete('/comments/{id}', 'destroy');
    });

    Route::prefix('users')->controller(UserController::class)-
>group(function () {
        Route::get('/profile', 'getProfileData');
        Route::post('/update', 'updateProfile');
        Route::get('/recipes', 'getUserRecipes');
        Route::get('/subscriptions', 'getUserSubscriptions');
```

```
    });

    Route::get('/author-profile/{id}',              [UserController::class,
'getAuthorProfile']);


    Route::controller(SubscriptionController::class)->group(function () {

        Route::post('/subscription', 'createSubscription');

        Route::post('/subscription/cancel', 'cancelSubscription');

        Route::get('/users/subscription/details',
'getUserSubscriptionInfo');

    });

    Route::get('/plans', [PlanController::class, 'index']);

    Route::get('/tags', [TagController::class, 'index']);


    Route::controller(SubscriberController::class)->group(function () {

        Route::post('/subscribe', 'subscribe');

        Route::post('/unsubscribe', 'unsubscribe');

    });

});


Route::any('/webhooks/payment/{service}', PaymentWebhookController::class);

Route::any('/callback/payment/{service}',
PaymentCallbackController::class);


<?php


namespace App\Services;


use App\DataTransferObjects\IngredientData;

use App\DataTransferObjects\RecipeData;

use App\DataTransferObjects\StepData;

use App\Models\Recipe;

use App\Repositories\RecipeRepository;

use Illuminate\Support\Facades\Log;
```

```
class RecipeService

{

    public function __construct(protected RecipeRepository $recipeRepository)

    {

    }


    public function createRecipe($validatedData): Recipe

    {

        $recipeData = $this->mapToRecipeData($validatedData);


        try {

            return $this->recipeRepository->createRecipeWithDetails($recipeData);

        } catch (\Exception $exception) {

            Log::error('Error in creating recipe: ' . $exception->getMessage());

            throw $exception;

        }

    }


    public function updateRecipe(int $recipeId, array $validatedData): Recipe

    {

        $recipeData = $this->mapToRecipeData($validatedData);


        try {

            return $this->recipeRepository->updateRecipeWithDetails($recipeId, $recipeData);

        } catch (\Exception $exception) {

            Log::error('Error in creating recipe: ' . $exception->getMessage());

            throw $exception;

        }

    }
```

```
    public function mapToRecipeData(array $validatedData): RecipeData
    {
        return new RecipeData(
            userId: auth()->user()->id,
            title: $validatedData['title'],
            description: $validatedData['description'],
            cooking_time: $validatedData['cooking_time'],
            difficulty_level: $validatedData['difficulty_level'],
            portions: $validatedData['portions'],
            coverPhoto: $validatedData['cover_photo'],
            ingredients: array_map(fn($ingredient) => new IngredientData(
                $ingredient['title'],
                $ingredient['measure'],
                $ingredient['count']
            ), $validatedData['ingredients']),
            steps: array_map(fn($step) => new StepData(
                $step['description'],
                array_map(fn($photo) => $photo, $step['photos'])
            ), $validatedData['steps']),
            tags: $validatedData['tags'] ?? null
        );
    }
}
```