Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 3 з дисципліни
«Сучасні технології розробки WEB-застосувань на платформі .NET (2 частина)»

«Проектування та створення веб-застосуноку в межах багаторівневої архітектури»

Виконав(ла) *ІП-14 Хільчук Артем Валерійович* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Вовк Євгеній Андрійович* _____
(прізвище, ім'я, по батькові)

Київ 2024

ЗМІСТ

1 ЗАВДАННЯ

1. У відповідності до наданих варіантів виконати тестування програмного засобу.

2. Виконати проектування архітектури програмного засобу в вигляді розприділеного моноліту або DDD, з використанням з використанням onion або clean або CQRS. Важливо при задачах фільтрації використовувати бібліотеку Ardalis.Specification, або альтернативи для побудови дерева виразів та його огортання у відповідний об’єкт.

3. Виконати моделювання системи шляхом використання C4 Model Diagram.

4. Виконати моделювання описом UML component, deployment diagram.

5. Ключові архітектурні рішення задокументувати у вигляді architectural decision record (ADR).

6. Забезпечити працездатність тестів та гарантувати що зміна архіектурного підходу не зашкодила функціональності продукту.Варіант завдання

1. Персональний блог. Авторизація/Аутентифікація: Можливість зареєструватися/увійти як автор. Створення та редагування постів: Автор може створювати, редагувати та видаляти свої пости. Перегляд постів: Відвідувачі можуть переглядати пости без реєстрації. Коментування: Відвідувачі можуть залишати коментарі під постами.

2 ВИКОНАННЯ

2.1 У відповідності до наданих варіантів виконати тестування програмного засобу.

У рамках виконання попередньої лабораторної роботи було реалізовано низку тестів – усі вони виконалися без будь-яких проблем.

2.2 Виконати проектування архітектури програмного засобу в вигляді розприділеного моноліту або DDD, з використанням з використанням onion або clean або CQRS. Важливо при задачах фільтрації використовувати бібліотеку Ardalis.Specification, або альтернативи для побудови дерева виразів та його огортання у відповідний об’єкт

У рамках даної лабораторної було вирішено використати Onion архітектуру, котра забезпечить щонайкраще зменшення зв’язності між компонентами та утримуваність проекту, з використанням CQRS, аби забезпечити найбільшого дотримання принципу єдиного призначення при розвитку застосунку.

Крім цього, буде використано низку додаткових зовнішніх пакетів. Одним з таких є Ardalis.Specification, що надає зручний інтерфейс для задання специфікацій. Іншим таким пакетом є MediatR – він реалізує патерн посередника та становить зручний засіб для управління запитами та їхніми обробниками в рамках використання патерну CQRS.

Змоделюймо отриману систему.

2.3 Виконати моделювання системи шляхом використання C4 Model Diagram.

Контекстний рівень:

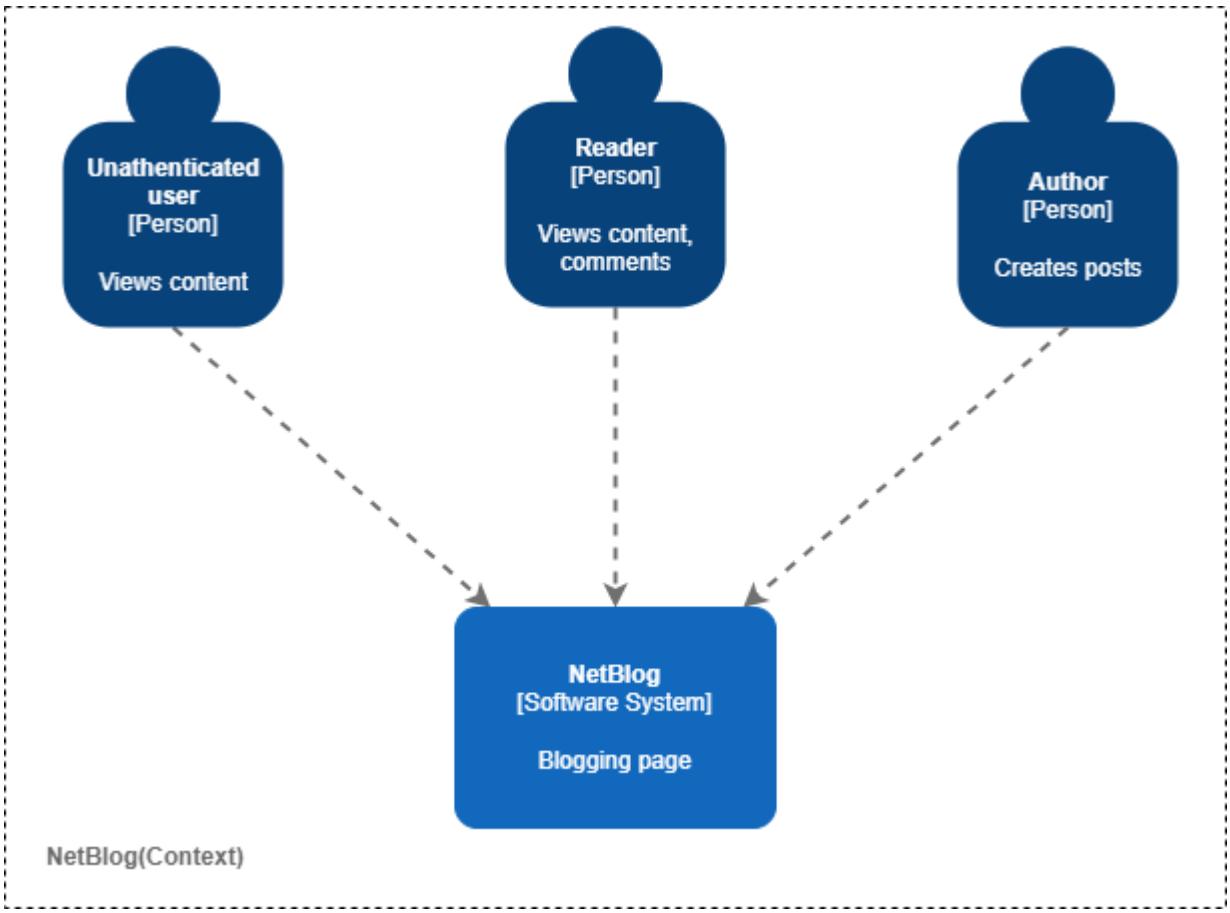

Рисунок 2.3.1 – контекстний рівень діаграми

Контейнерний рівень:

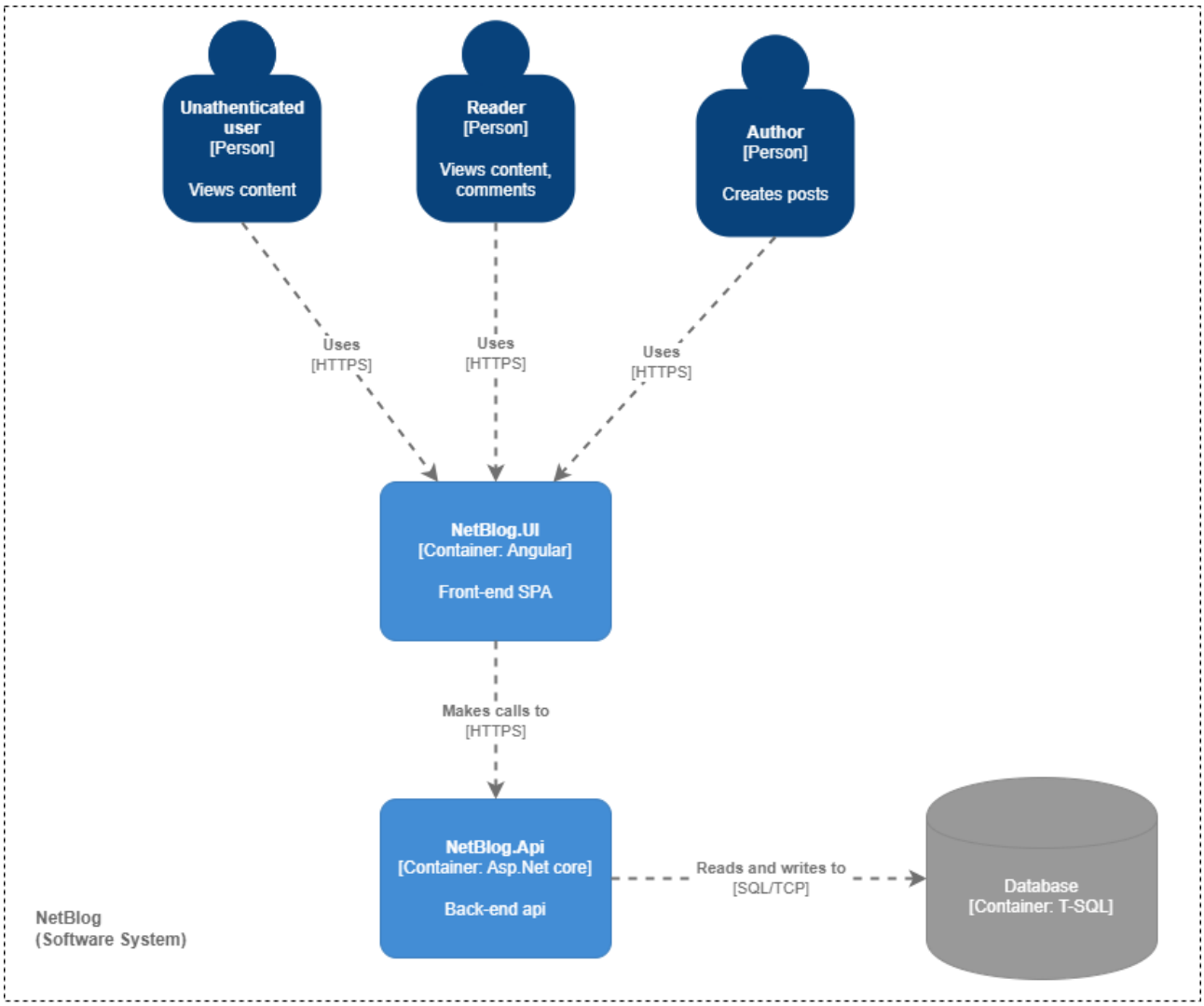

Рисунок 2.3.2 – контейнерний рівень діаграми

Компонентний рівень:

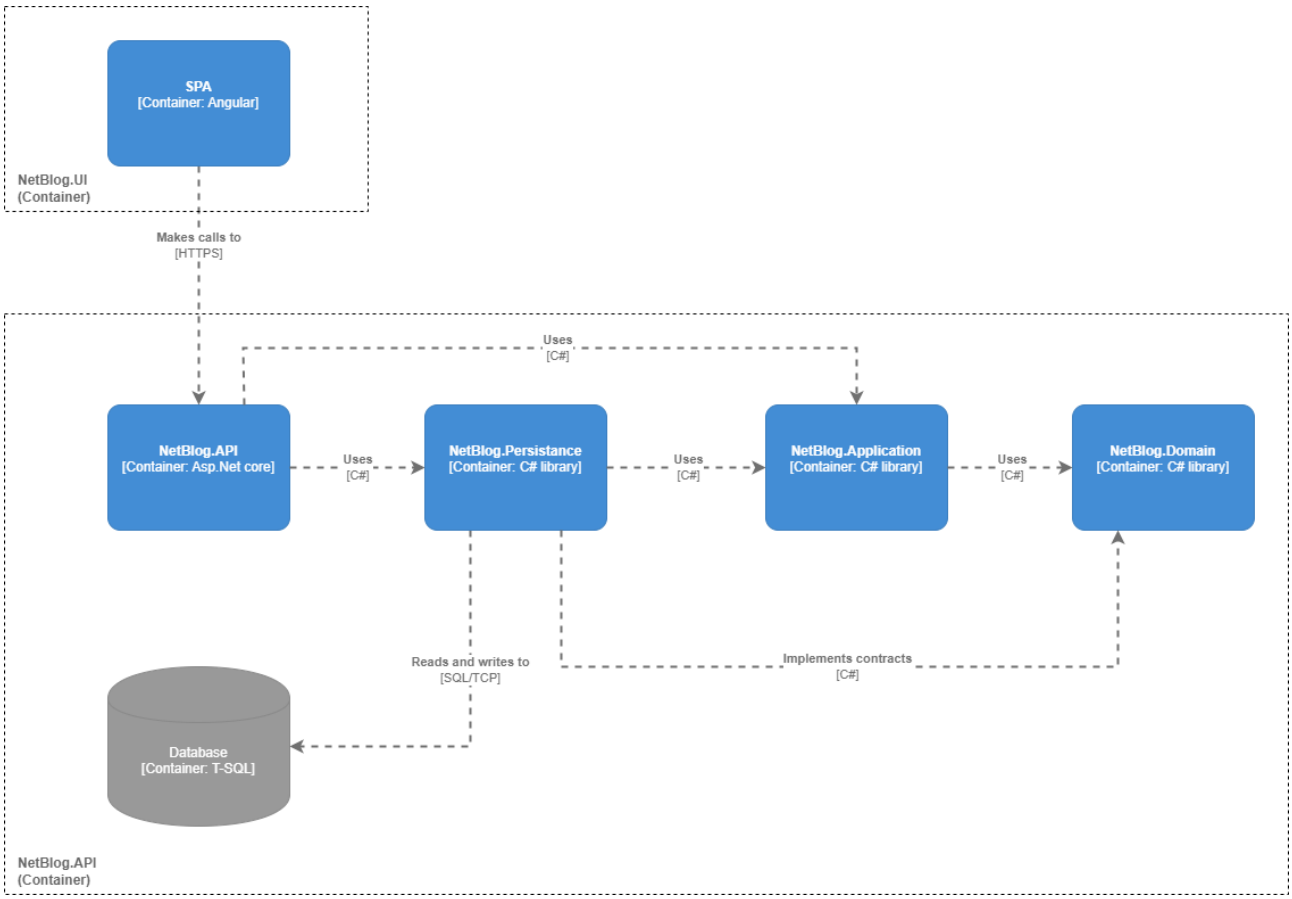

Рисунок 2.3.3 – компонентний рівень діаграми

Класовий рівень:

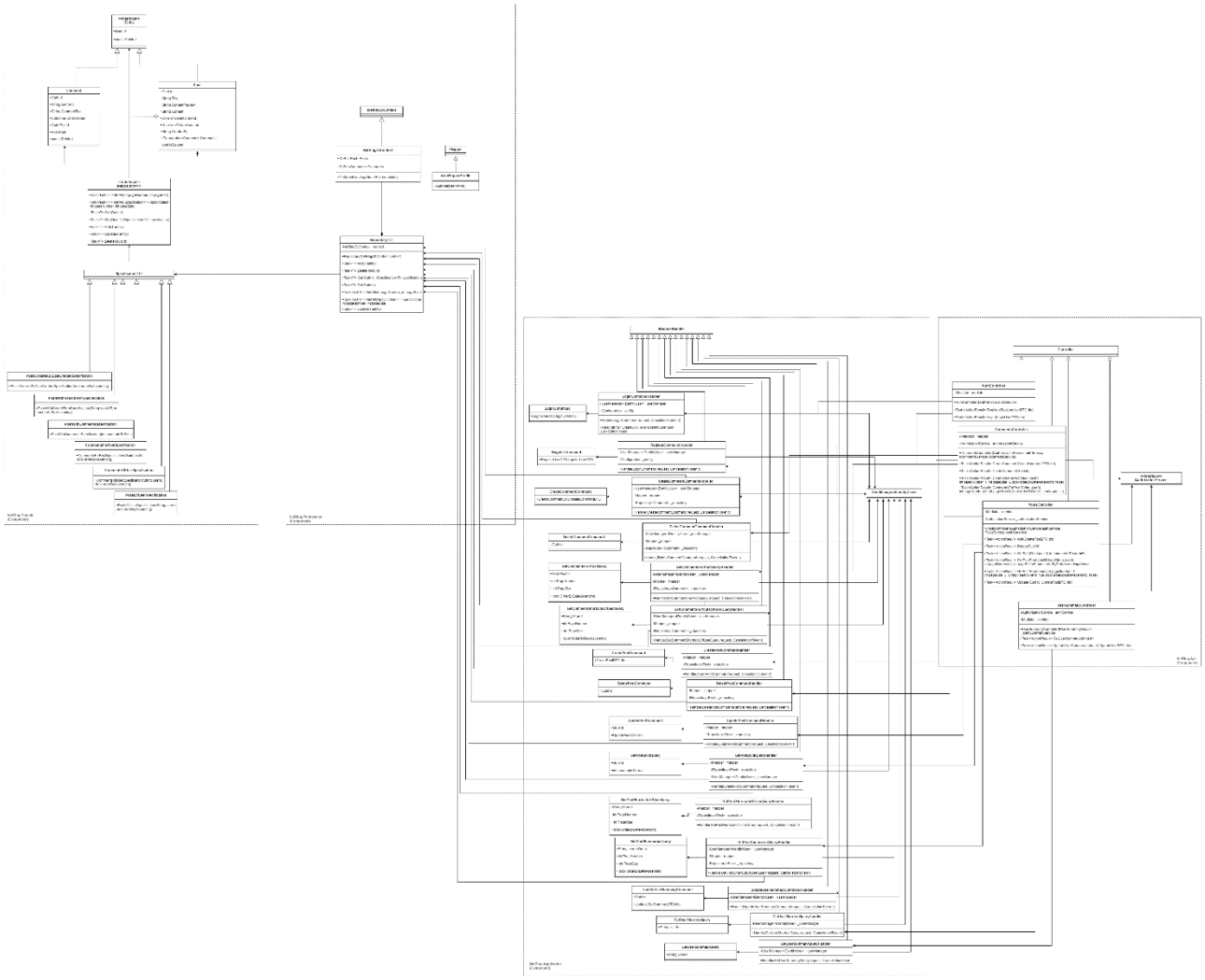

Рисунок 2.3.4 – класовий рівень діаграми

2.4 Ключові архітектурні рішення задокументувати у вигляді architectural decision record (ADR).

Записи будуть оформлюватися за шаблоном, запропонованим Майклом найгардом. Головні архітектурні рішення наведено нижче:

Id	1
Назва	Використовувати архітектуру Onion
Статус	Прийнято
Контекст	Компоненти застосунку мають численні зв’язки між собою та створюватимуть проблему під час масштабування проекту
Рішення	Було вирішено використовувати архітектуру Onion для розробки застосунку
Наслідки	• Компоненти проекту розділятимуться на шари • Шари матимуть мінімально можливу кількість зв’язків між собою • Кожен шар матиме можливість залежати лише від шарів, нижчих за ієрархією

Id	2
Назва	Використовувати Asp.Net core та C#
Статус	Прийнято
Контекст	Додаток потребує технологічне середовище для реалізації. Усі компоненти мають бути реалізовані єдиною мовою
Рішення	Було вирішено використовувати мову C# та технологічне середовище Asp.Net core
Наслідки	• Компоненти проекту будуть написані мовою C# • Технологічним середовищем для контролерів є Asp.Net core

Id	3
Назва	Розбити застосунок на 4 шари
Статус	Прийнято
Контекст	Архітектура додатку обрана, однак конкретика відсутня.
Рішення	Розбити застосунок на шари: • Api • Persistance • Application • Domain
Наслідки	• Компоненти проекту розділятимуться на шари • Шари матимуть мінімально можливу кількість зв’язків між собою • Кожен шар матиме можливість залежати лише від шарів, нижчих за ієрархією

Id	4
Назва	Використовувати патерн CQRS
Статус	Прийнято
Контекст	Додаток передбачає збереження та повернення даних, що містять значну варіативність у наборі повернених полів сутностей бізнес логіки, що призведе до значного зростання сервісів
Рішення	Було вирішено застосовувати патерн CQRS
Наслідки	• Посилюється дотримання принципу Single Responsibility • Запити на зчитування та зміну/запис даних можна оброблювати різними способами

Id	5
Назва	Використовувати патерн Mediator для управління запитами та їхніми обробниками
Статус	Прийнято
Контекст	Необхідно сформувати механізм взаємодії між компонентами системи із забезпечує слабкої зв'язності між відправниками запитів та їхніми обробниками
Рішення	Вирішено використовувати патерн Mediator, що надаватиме можливість управління запитами та їхніми обробниками, виступаючи посередником між ними та зменшуючи залежності.
Наслідки	• Зниження зв’язності між компонентами • Спрощення масштабованості додатку

Id	6
Назва	Використовувати JWT модель автентифікації
Статус	Прийнято
Контекст	Варіанти використання системи передбачають авторизацію досвіду користувачів залежно від їньої ролі, отож, постає потреба в застосуванні автентифікації для веб-додатку
Рішення	Вирішено використовувати JWT-токени для автентифікації користувачів.
Наслідки	• Зниження зв’язності між компонентами • Спрощення масштабованості додатку

Id	7
Назва	Використовувати принципи REST
Статус	Прийнято
Контекст	Стоїть потреба в стандартизованому способі взаємодії між клієнтами та сервером
Рішення	Вирішено використовувати RESTful підхід для побудови API.
Наслідки	• Зниження зв'язності між клієнтом та сервером • Підвищено стандартизованість та легкість використання застосунку

2.5 Забезпечити працездатність тестів та гарантувати що зміна архіектурного підходу не зашкодила функціональності продукту.

Для перевірки якості створеного застосунку було перенесено інтеграційні та End-to-End тести з попередньої роботи. Запуск тестів демонструє, що зміна архітектурного підходу не зашкодила функціональності проекту.

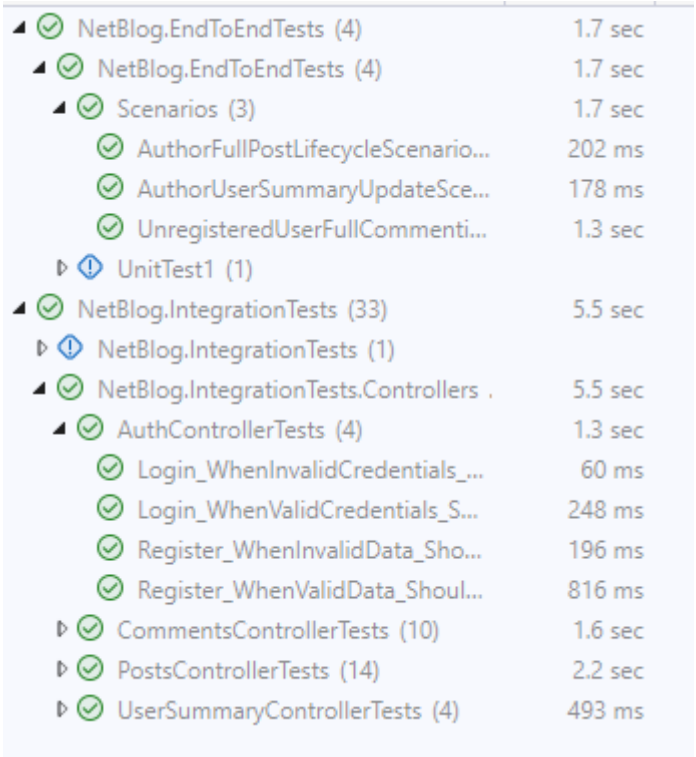

Рисунок 2.5.1 – результати виконання тестування

ВИСНОВОК

Отож, у ході виконання лабораторної роботи було обрано цільову архітектуру застосунку – Onion у сукупності з патерном CQRS – та реалізовано його у відповідності до проведених моделювань. Під час виконання роботи було описано модель програмного засобу за допомогою діаграми C4 на відповідних рівнях, а також зафіксовано головні архітектурні рішення шляхом створення низки ADR записів. Урешті-решт, було виконано розроблені інтеграційні та End-to-End тести та встановлено, що зміна архітектурного підходу не вплинула на функціональність застосунку. Набуто практичних навичок проектування програмних засобів шляхом використання сучасних архітектур та інструментів моделювання.