

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

**«Проектування та створення веб-застосунок 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	3
2	ВИКОНАННЯ.....	4
	ВИСНОВОК	7

1 ЗАВДАННЯ

- 1.1 У відповідності до наданих варіантів виконати тестування програмного засобу
- 1.2 Обрати та аргументувати вибір типу тесту(ів) що будуть використані для контролю якості складової частини програмного коду, функціональної поведінки тощо.
- 1.3 Виконати покриття тестами не менше ніж на 80%. Використати наступні тести:
 - a) Unit.
 - b) Integration.
 - c) End-to-End

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

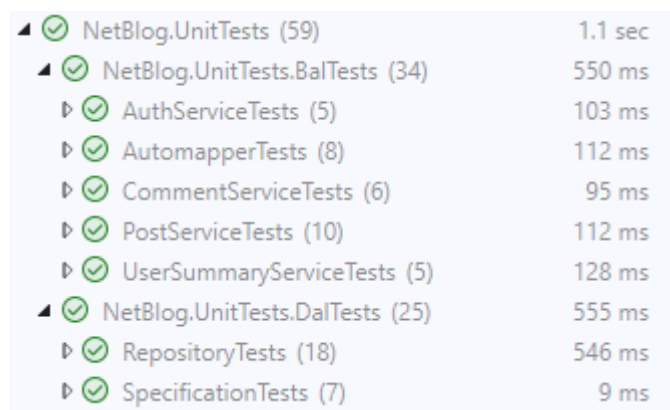
Розпочнімо з вибору відповідних тестів для компонент застосунку. Для тестування шару доступу до даних було застосовано юніт-тести з використанням бази даних в оперативній пам'яті. Таким чином можна перевірити на правильність роботи логіку класів даного шару, при цьому переконавшись у тому, що сутності дійсно зберігатимуться в базі даних, linq вирази правильно перетворюватимуться в sql-запити, тощо. Пакетом, на основі якого буде реалізовано тести, є xUnit.

Для тестування шару бізнес логіки було також застосовано юніт-тести у зв'язці із засобами бібліотеки Moq для підробки залежностей.

Для тестування шару презентації було вирішено не проводити юніт-тестування, оскільки вони не містять комплексної бізнес-логіки й повністю покладаються на відповідні сервіси для виконання завдань. Відповідно, для тестування контролерів було проведено інтеграційне тестування, аби переконатися в правильності їхньої роботи у зв'язку із сервісами шару бізнес логіки, авторизацією, тощо.

Урешті-решт, також буде проведено End-to-End тестування, аби випробувати окремі сценарії використання застосунку та переконатися в його працездатності.

Відповідно, у рамках виконання даного завдання було реалізовано 59 юніт тестів для репозиторію та окремих специфікацій шару доступу до даних, а також кожної конкретної реалізації сервісів шару бізнес-логіки, як показано на рисунку 2.1



NetBlog.UnitTests (59)	1.1 sec
NetBlog.UnitTests.BalTests (34)	550 ms
AuthServiceTests (5)	103 ms
AutomapperTests (8)	112 ms
CommentServiceTests (6)	95 ms
PostServiceTests (10)	112 ms
UserSummaryServiceTests (5)	128 ms
NetBlog.UnitTests.DalTests (25)	555 ms
RepositoryTests (18)	546 ms
SpecificationTests (7)	9 ms

Рисунок 2.1 – колекції юніт-тестів

Результати аналізу покриття коду юніт-тестами показано на рисунку 2.2. Для даної цілі використовувався пакет Coverlet.

Module	Line	Branch	Method
NetBlog.BAL	94.7%	81.25%	98.7%
NetBlog.DAL	97.14%	86.36%	97.72%
Total	95.47%	82.55%	98.34%
Average	95.92%	83.8%	98.21%

Рисунок 2.2 – результати аналізу покриття коду юніт-тестами

Як бачимо, досягнуто покриття коду за стрічками в 95.47%, 82.25% за можливими розгалуженнями, а також 98.34% за методами.

У процесі проведення інтеграційного тестування застосунку було створено тести для кожного з наявних контролерів, що розглядали позитивні та негативні випадки, у тому числі такі, що включали як невалідність вхідних даних, так і неавторизованість користувача до виконання дії. Усього було розроблено 32 таких випадки, розподіл числа тестів за контролерами наведено на рисунку 2.3

NetBlog.IntegrationTests (32)	5.7 sec
NetBlog.IntegrationTests.Controllers	5.7 sec
AuthControllerTests (4)	479 ms
CommentsControllerTests (10)	1.5 sec
PostsControllerTests (14)	2.2 sec
UserSummaryControllerTests (4)	1.5 sec

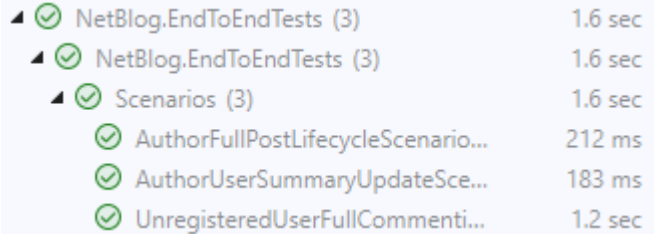
Рисунок 2.3 – кількості інтеграційних тестів за контролерами

Урешті-решт, було релізовано 3 випадки End-to-End тестування, у яких розглядалися наступні сценарії:

1. Типовий сценарій дій над постами:
 - 1.1. Автор входить у систему
 - 1.2. Автор створює пост

- 1.3. Автор читає створений пост
- 1.4. Автор гортає коментарі під постом
- 1.5. Автор видаляє коментар
- 1.6. Автор оновлює пост
- 1.7. Автор видаляє пост
2. Типовий сценарій дій читача
 - 2.1. Незареєстрований користувач читає пост
 - 2.2. Користувач реєструється як читач
 - 2.3. Користувач входить в систему
 - 2.4. Користувач коментує пост
 - 2.5. Користувач переглядає свій профіль
 - 2.6. Користувач видаляє свій коментар
3. Сценарій редагування профілю
 - 3.1. Автор входить у систему
 - 3.2. Автор переглядає свій профіль
 - 3.3. Автор оновлює дані у своєму профілі
 - 3.4. Автор переходить на один зі своїх постів, аби переконатися, що зміни дійсно збереглися

Перелік тестів наведено на рисунку 2.4



▲ ✓ NetBlog.EndToEndTests (3)	1.6 sec
▲ ✓ NetBlog.EndToEndTests (3)	1.6 sec
▲ ✓ Scenarios (3)	1.6 sec
✓ AuthorFullPostLifecycleScenario...	212 ms
✓ AuthorUserSummaryUpdateSce...	183 ms
✓ UnregisteredUserFullCommenti...	1.2 sec

Рисунок 2.4 – перелік End-to-End тестів

З повним кодом можна ознайомитись за посиланням:

<https://github.com/KhilchukArtemIP-14/KPI-DotNet-3-2/tree/Lab2>

ВИСНОВОК

Отож, у ході виконання лабораторної роботи було проведено контроль якості розробленого застосунку шляхом реалізації та виконання низки тестів: юніт-тестів, інтеграційних та End-to-End. Під час виконання роботи було обрано конкретні види тестування і їхні допоміжні засоби для різних категорій складових додатку та обгрунтовано доцільність їхнього використання. Урешті-решт, було виконано розроблені тести та проаналізовано покриття додатку юніт тестами за допомогою засобу Coverlet: усі тести успішно пройдено, а аналіз продемонстрував покриття 95% коду застосунку. Набуто практичних навичок проведення тестування додатків за допомогою ASP.Net core, а також пакетів xUnit, Moq та інших.