Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота № 3

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи проектування розгортання»

Виконав
Студент групи ІА-31: Губар Б. О.

Перевірив: Мягкий М. Ю.

Київ 2025

Зміст

1. **Мета:**

Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

2. **Хід роботи:**

Тема : CI server (state, command, decorator, mediator, visitor, soa)

1) Ознайомитись з короткими теоретичними відомостями.
2) Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи та спроєктувати діаграму розгортання використання відповідно до обраної теми лабораторного циклу

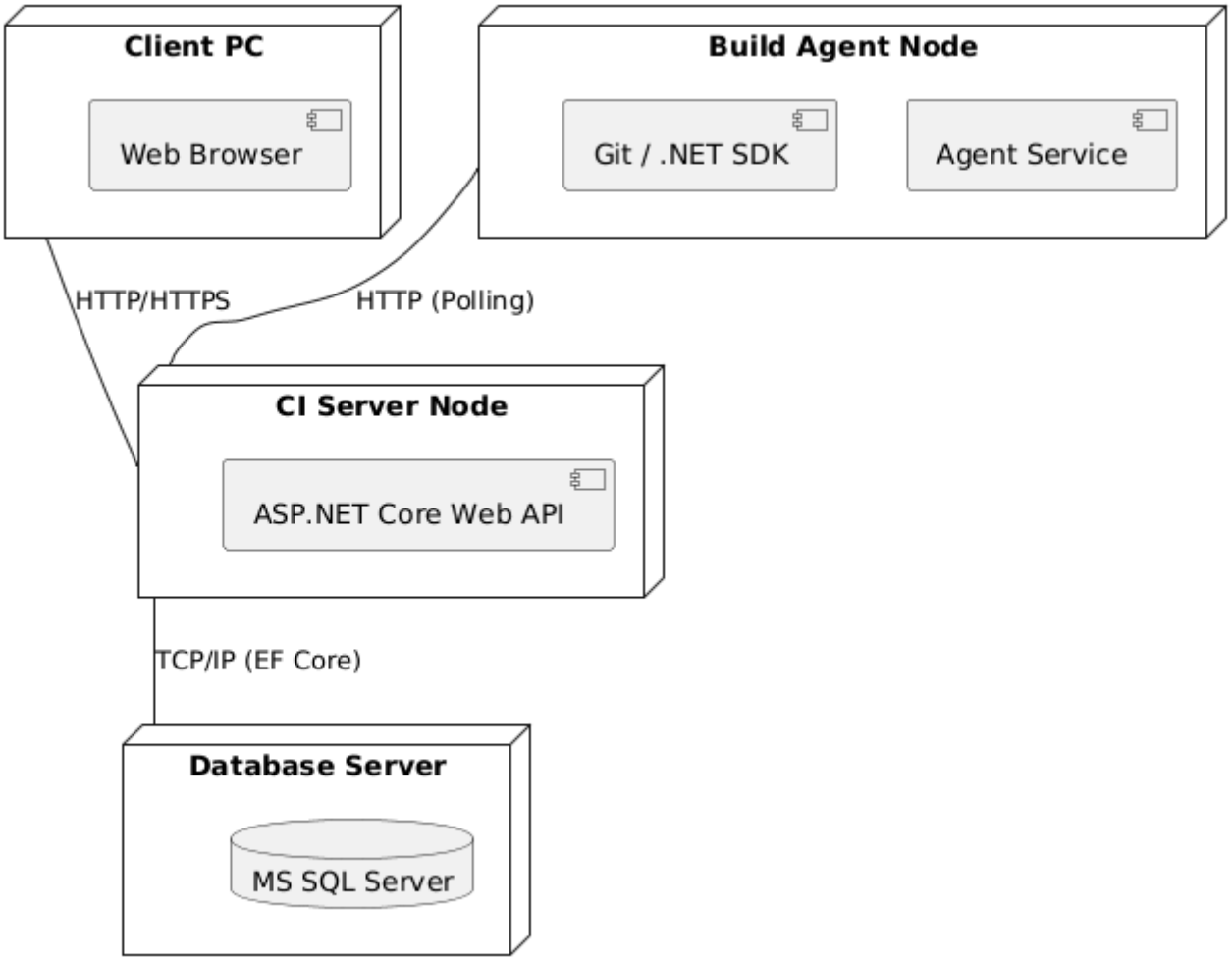


Рис.1 – Діаграма розгортання

На діаграмі розгортання (Рис. 1) зображено фізичну архітектуру системи "CI Server". Система побудована за розподіленою моделлю і включає такі вузли:

- Client PC (Вузол розробника): Персональний комп'ютер користувача. Взаємодія відбувається через веб-браузер, який відправляє HTTP-запити до головного сервера (для перегляду логів, запуску збірок).
- CI Server Node (Master): Центральний сервер, на якому розгорнуто ASP.NET Core Web API. Він приймає запити, керує чергою завдань та зберігає результати.
- Database Server: Окремий (або віртуальний) сервер із СУБД Microsoft SQL Server, де зберігаються налаштування проєктів, історія білдів та логи.
- Build Agent Node: Окремий обчислювальний вузол (або контейнер), де безпосередньо відбувається компіляція коду. На ньому встановлено .NET SDK, Git та Agent Service (консольний додаток), який спілкується з сервером через HTTP/TCP.

3) Розробити діаграму компонентів для проєктованої системи

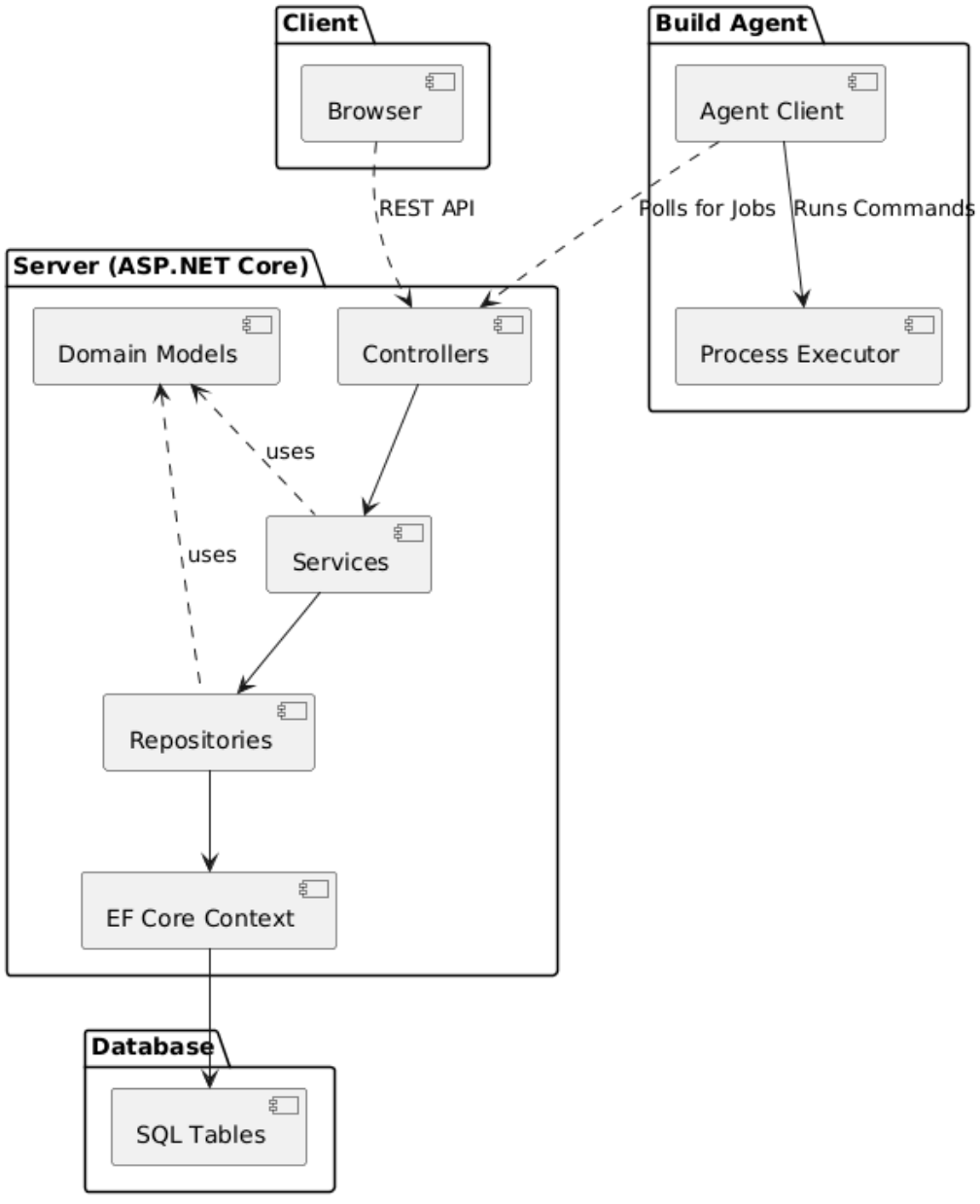


Рис.2 – Діаграма компонентів

На діаграмі компонентів зображено логічну структуру системи, розділену на шари:

Presentation Layer (Client): Компоненти інтерфейсу (Browser), які звертаються до API.

Service Layer (Server Middleware):

Controllers: Приймають REST-запити.

BuildService: Відповідає за бізнес-логіку створення та керування білдами.

ProjectService: Керує налаштуваннями проєктів.

Data Layer (Server):

ApplicationDbContext: Контекст Entity Framework Core.

Repositories: Абстракція доступу до даних.

Agent Layer:

AgentClient: Компонент, що опитує сервер на наявність задач.

ProcessExecutor: Компонент, що запускає реальні процеси ОС (git.exe, dotnet.exe).

4) Розробити як мінімум дві діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі

Автоматичний запуск збірки	
Передумови	Проєкт налаштований у CI сервері, у VCS налаштований Webhook.
Постумови	Створено нову задачу (Build Job) у черзі, статус збірки "Pending".
Взаємодіючі сторони	VCS (GitHub/GitLab), CI Server.
Короткий опис	Розробник робить push у репозиторій, VCS повідомляє сервер, сервер створює задачу.
Основний потік подій	Розробник відправляє код у репозиторій.
	VCS відправляє HTTP POST запит (Webhook) на CI Server.
	CI Server валідує запит.
	CI Server створює запис Build у базі даних зі статусом Pending.
	CI Server додає задачу в чергу на виконання.
Винятки	Невалідний токен Webhook. Сервер відхиляє запит (403 Forbidden).

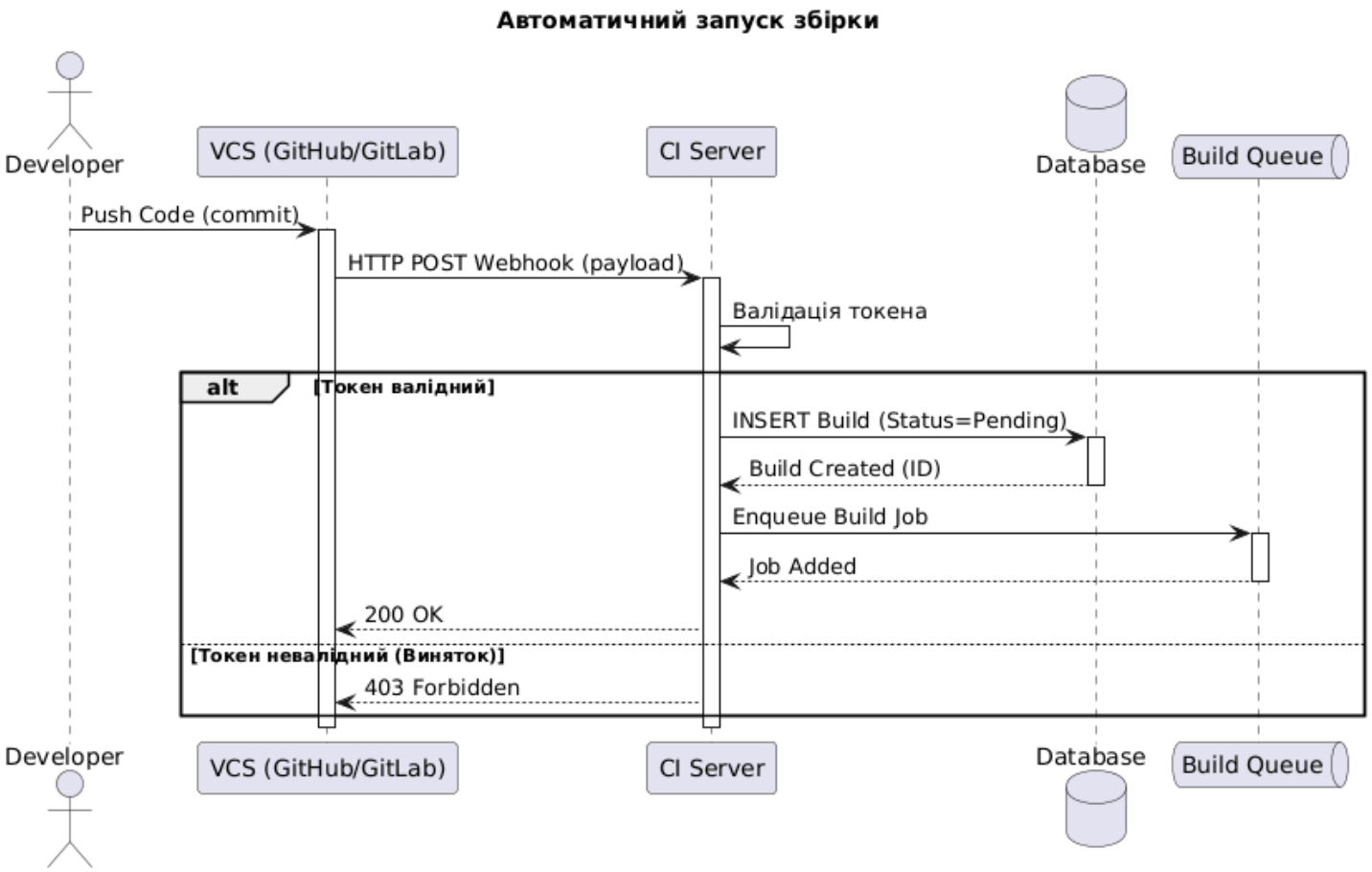


Рисунок 3.1. Діаграма послідовності «Автоматичний запуск збірки»

Виконання пайплайну	
Передумови	У черзі є задача зі статусом Pending, є вільний Агент.
Постумови	Збірка завершена зі статусом Success або Failed, логи збережені.
Взаємодіючі сторони	CI Server, Build Agent.
Короткий опис	Агент отримує задачу та виконує послідовність команд.
Основний потік подій	1. Агент запитує задачу у Сервера.
	2. Сервер змінює статус збірки на Running.
	3. Агент клонує репозиторій.
	4. Агент виконує кроки (Build, Test).
	5. Агент відправляє логи на Сервер у реальному часі.
	6. Агент завершує роботу і оновлює фінальний статус у базі даних.
Винятки	Помилка компіляції. Агент перериває виконання, ставить статус Failed і записує це в БД.

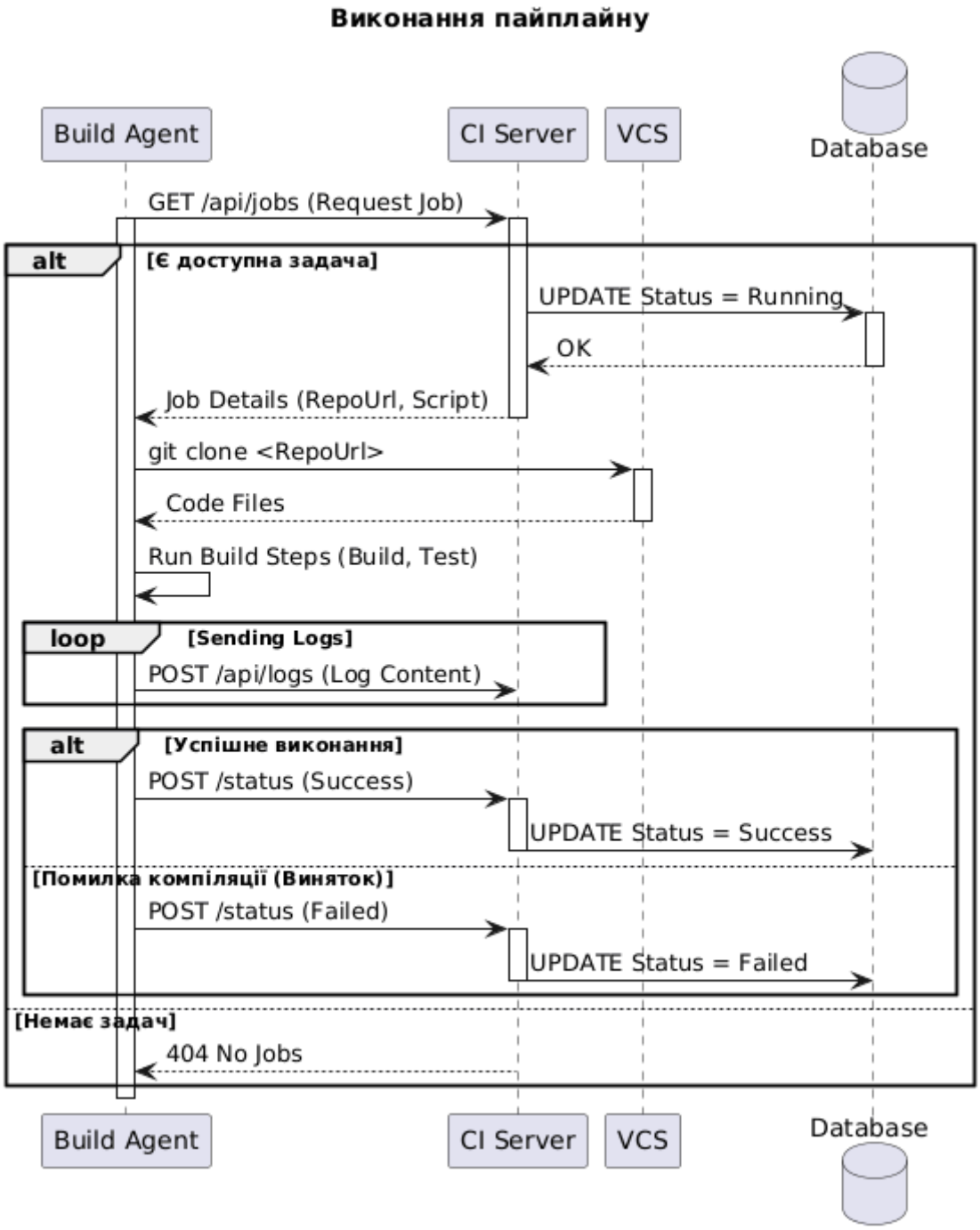


Рисунок 3.2. Діаграма послідовності «Виконання пайплайну»

Перегляд логів	
Передумови	Збірка вже запущена або завершена, у базі є записи логів.
Постумови	Розробник бачить текстовий лог виконання.
Взаємодіючі сторони	Розробник, CI Server.
Короткий опис	Розробник відкриває сторінку збірки, щоб дізнатися причину помилки або хід виконання.
Основний потік подій	1. Розробник обирає конкретну збірку зі списку історії.
	2. Розробник натискає "View Logs".
	3. Система робить запит до БД в таблицю BuildLogs.

	4. Система відображає список рядків логу, відсортованих за часом.
Винятки	Збірка не знайдена. Система видає помилку 404.

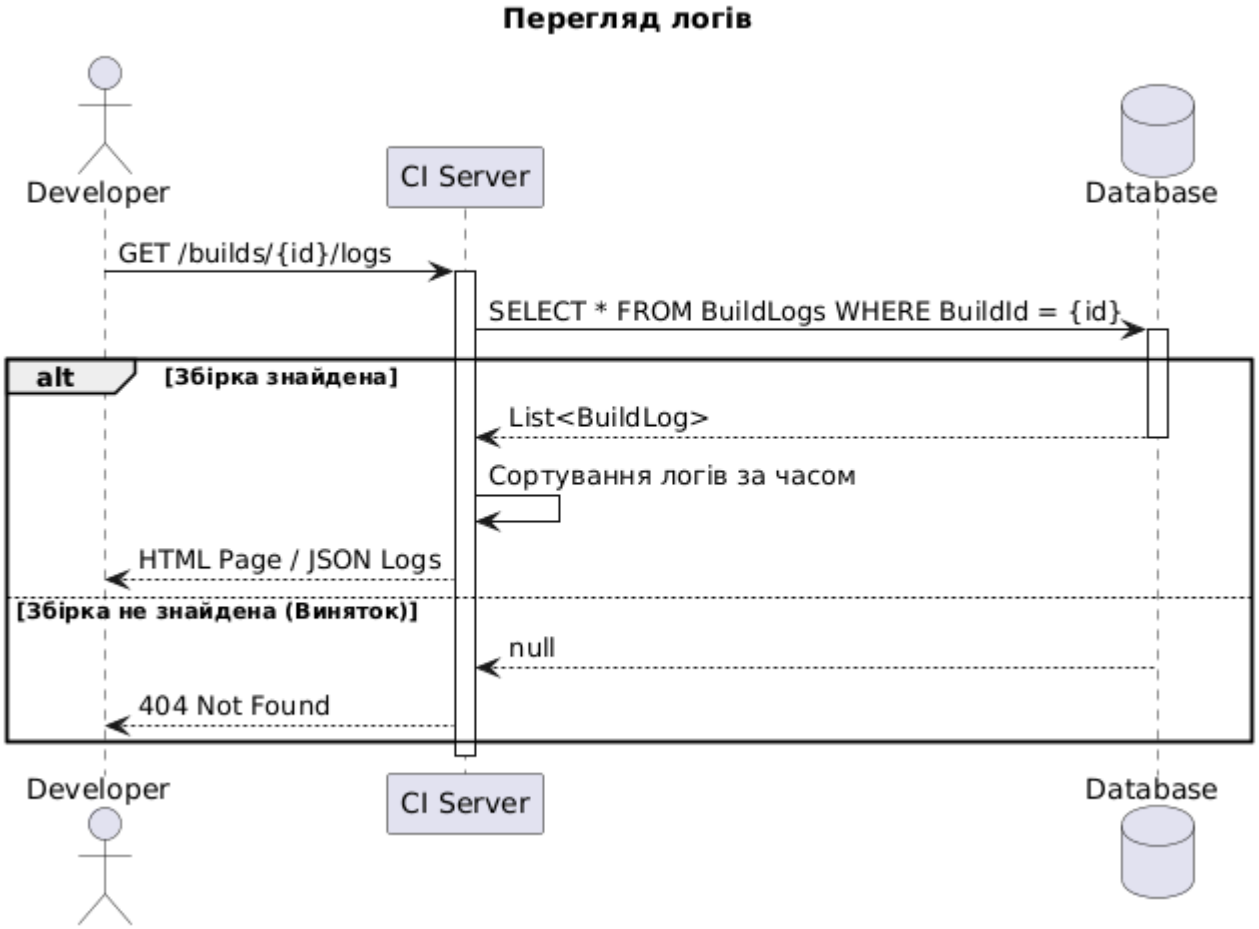


Рисунок 3.3. Діаграма послідовності «Перегляд логів»

5) На основі спроєктованих діаграм розгортання та компонентів доопрацювати програмну частину системи. Реалізація системи, додатково до попередньої реалізації, повинна містити як мінімум дві візуальні форми. В системі вже повинен бути повністю реалізована архітектура (повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД та відображенням на UI).

Реалізація системи

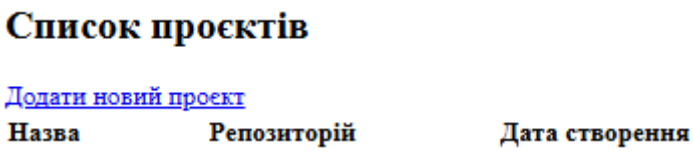


Рисунок 4.1. Список наших проєктів

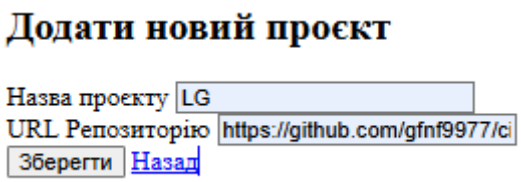


Рисунок 4.2. Додаємо новий проєкт

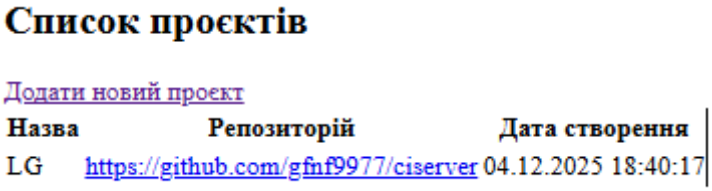
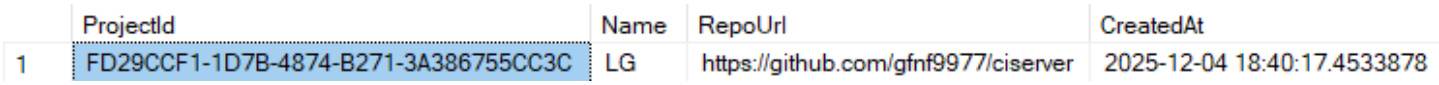


Рисунок 4.3. Оновлена сторінка



	ProjectId	Name	RepoUrl	CreatedAt
1	FD29CCF1-1D7B-4874-B271-3A386755CC3C	LG	https://github.com/gfnf9977/ciserver	2025-12-04 18:40:17.4533878

Рисунок 4.4. Таблиця в SQL з доданим записом

3. Висновок

У процесі виконання лабораторної роботи було досягнуто визначеної мети: освоєно проєктування фізичної та логічної архітектури розподіленої системи "CI Server". Було розроблено діаграму розгортання, яка демонструє розподіл вузлів (Server, Database, Build Agent), та діаграму компонентів, що відображає модульну структуру додатку. Крім того, опановано розробку діаграм взаємодії, зокрема діаграм послідовності для сценаріїв запуску збірки та процесу логування, що дозволило деталізувати логіку обміну повідомленнями між розподіленими компонентами системи. Важливим етапом стала практична реалізація спроєктованої архітектури: доопрацьовано програмну частину системи на базі ASP.NET Core MVC, створено візуальні форми для введення та відображення даних, а також забезпечено повний цикл взаємодії з базою даних MS SQL (CRUD-операції), що підтверджує коректність обраних архітектурних рішень.

4. Контрольні питання

1. Що таке діаграма розгортання в UML? Діаграма розгортання — це тип UML-діаграми, який ілюструє фізичну архітектуру системи: як програмні компоненти розподілені на апаратних або програмних вузлах та як вони взаємодіють у реальному середовищі.

2. Які типи вузлів можуть бути представлені на діаграмі розгортання? На діаграмі розгортання виділяють два основні типи вузлів: — Апаратні вузли (фізичні пристрої, наприклад, сервери, комп’ютери, смартфони). — Програмні вузли (середовища виконання, такі як операційні системи, контейнери або віртуальні машини).

3. Які зв’язки використовуються на діаграмі розгортання? На діаграмі розгортання застосовуються такі зв’язки: — Асоціація — показує фізичне або логічне з’єднання між вузлами. — Залежність — вказує, що один вузол або компонент використовує ресурси іншого.

4. Які елементи містить діаграма компонентів? Діаграма компонентів включає такі елементи: — Компоненти, інтерфейси, порти, залежності, пакети та підсистеми.

5. Що означають зв’язки на діаграмі компонентів? Зв’язки на діаграмі компонентів відображають взаємозалежність між компонентами, наприклад, який компонент використовує або реалізує інтерфейс іншого компонента.

6. Які існують види діаграм взаємодії? До діаграм взаємодії належать: — Діаграма послідовностей. — Діаграма комунікацій. — Діаграма часу. — Діаграма взаємодії огляду.

7. Яке призначення діаграми послідовностей? Діаграма послідовностей призначена для візуалізації порядку викликів і обміну повідомленнями між об’єктами під час виконання певного сценарію або процесу.

8. Які ключові елементи можуть бути присутні на діаграмі послідовностей? На діаграмі послідовностей виділяють такі ключові елементи: — Активи (актори), об’єкти, лінії життя, повідомлення (синхронні та асинхронні), а також області активації.

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання? Діаграми послідовностей деталізують кроки, які виконуються для реалізації кожного варіанту використання, описаного на діаграмі варіантів використання.

10. Як діаграми послідовностей пов’язані з діаграмами класів? Об’єкти, що взаємодіють на діаграмі послідовностей, є екземплярами класів, описаних на діаграмі класів. Повідомлення між об’єктами відповідають викликам методів цих класів.