Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3

Технології розроблення програмного забезпечення

«Діаграма розгортання. Діаграма компонентів.

Діаграма взаємодій та послідовностей.»

Варіант 24

Виконав
студент групи ІА–33:
Вовчок М. М.

Перевірив
Мягкий М. Ю.

Київ 2025

Тема

Flexible Automation Tool (автоматизація дій за правилами та макросами, використання поведінкових патернів: strategy, command, observer тощо).

Короткі теоретичні відомості

Діаграма розгортання відображає фізичну архітектуру системи: вузли, на яких виконуються програмні компоненти, середовища виконання та мережеві з’єднання між ними. Для Flexible Automation Tool це клієнтський пристрій користувача, сервер застосунку та сервер бази даних, між якими передаються HTTP/API-запити та SQL-запити.

Діаграма компонентів показує логічну структуру програмного забезпечення: окремі модулі (компоненти), їхні інтерфейси та залежності. У нашому випадку виділяються компоненти клієнтського інтерфейсу, серверна частина (обробка правил, макросів, планувальник) та компоненти доступу до бази даних.

Діаграма послідовностей демонструє динамічну взаємодію між об’єктами системи в часі. Вона дозволяє детально описати порядок викликів при створенні правила автоматизації, автоматичному запуску правила за розкладом або виконанні макросу користувача.

Хід роботи

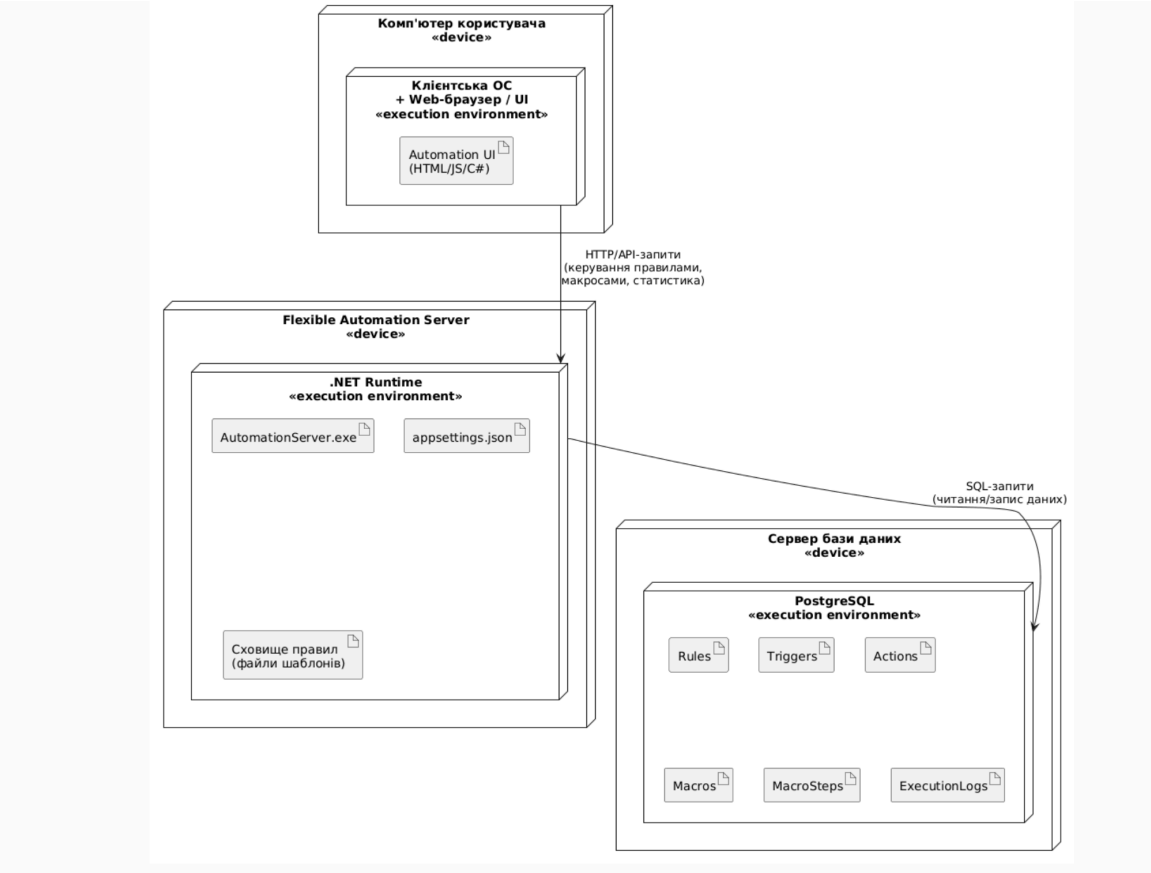
Завдання

1. Ознайомитися з короткими теоретичними відомостями.
2. Розробити діаграму розгортання для проектованої системи Flexible Automation Tool.
3. Розробити діаграму компонентів для проектованої системи.
4. Розробити діаграму послідовностей для основних сценаріїв роботи

системи.

5. Скласти звіт про виконану роботу.

Діаграма розгортання



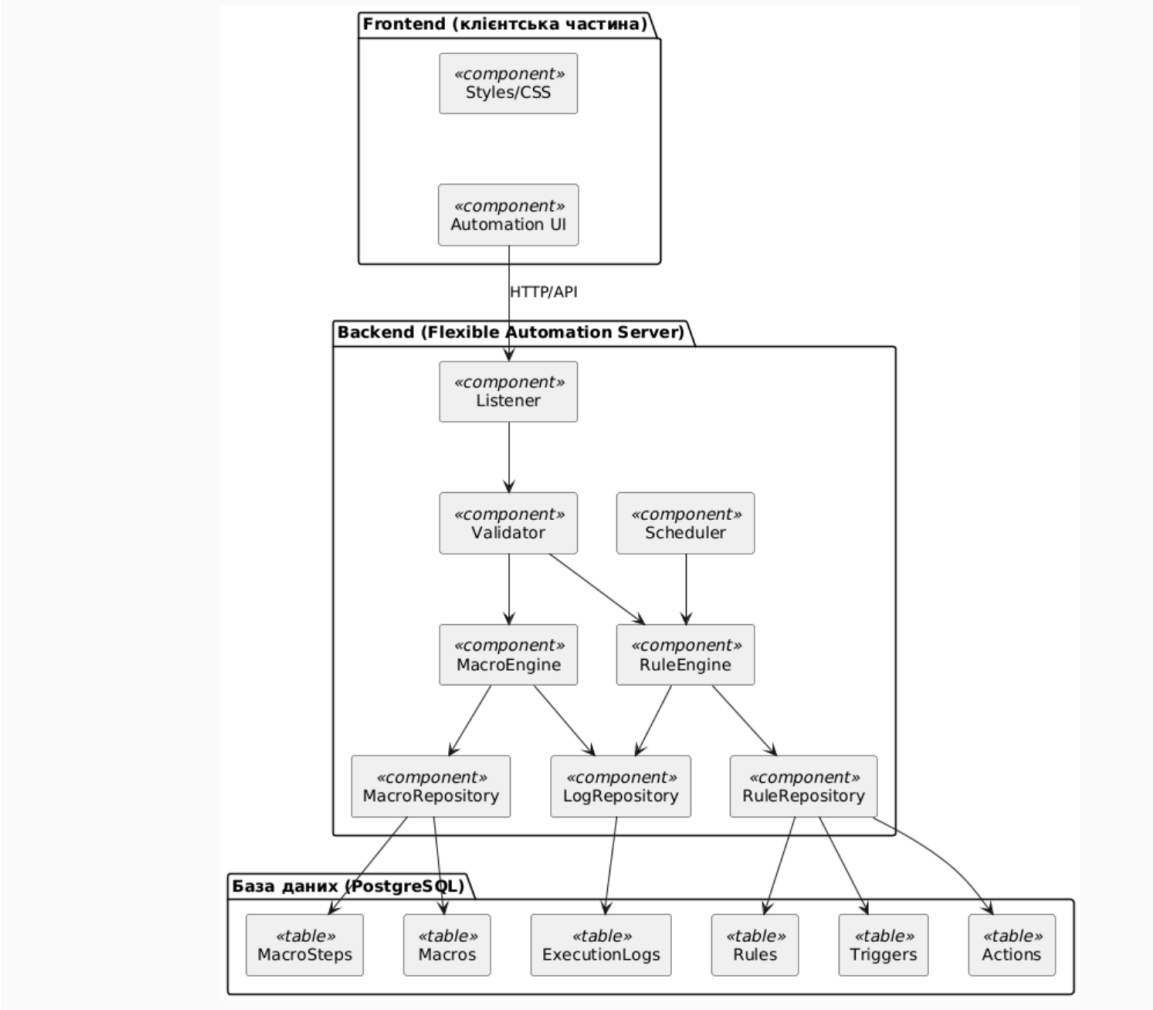
Діаграма розгортання (рис. 1)

містить три основні вузли: пристрій користувача, сервер застосунку Flexible Automation Server та сервер бази даних. Користувацький пристрій містить клієнтський застосунок або Web-браузер, через який користувач створює правила, керує макросами та переглядає статистику. Сервер застосунку виконує бізнес-логіку: обробляє HTTP/API-запити, зберігає правила, запускає планувальник та взаємодіє із зовнішніми сервісами. Сервер бази даних забезпечує збереження правил, тригерів, дій, макросів та журналу виконань.

На рисунку 1 відображені мережеві з’єднання між вузлами:

• HTTP/API-запити від клієнта до сервера застосунку для керування правилами та макросами;
• SQL-запити від сервера застосунку до сервера бази даних для збереження та читання даних.

Діаграма компонентів

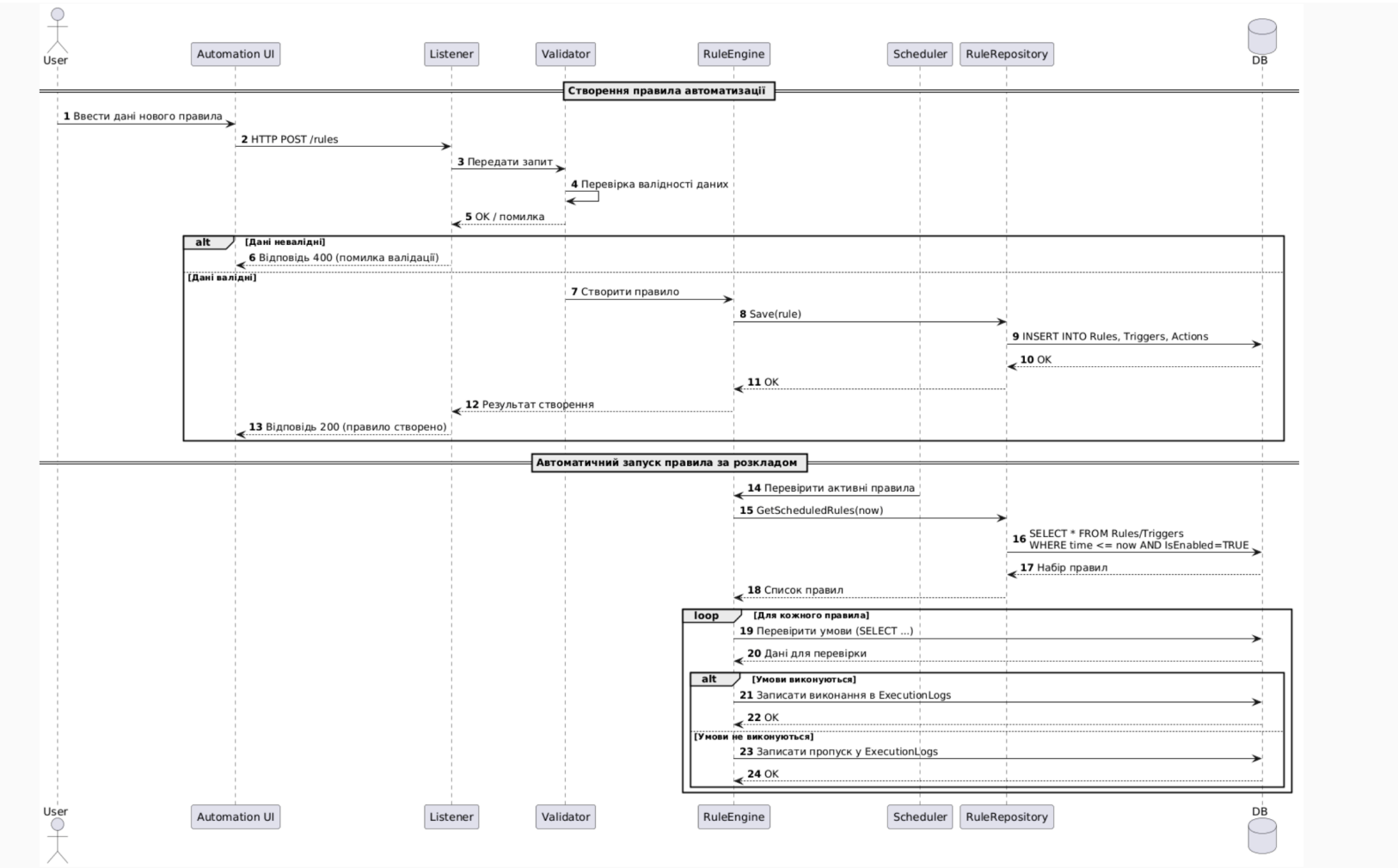


Діаграма компонентів (рис. 2)

Виділено три основні частини системи: клієнтський інтерфейс (Frontend), сервер застосунку (Backend) та база даних. Frontend містить компонент Automation UI, який відповідає за відображення форм створення правил, управління макросами та перегляд статистики. Backend включає компоненти Listener, Validator, RuleEngine, MacroEngine, Scheduler та RuleRepository. База даних представлена компонентами таблиць Rules, Triggers, Actions, Macros, MacroSteps та ExecutionLogs.

Компонент Listener отримує вхідні HTTP/API-запити. Validator перевіряє коректність даних і визначає тип запиту: створення/ оновлення правила, запуск макросу або запит статистики. Компонент RuleEngine відповідає за перевірку тригерів, умов та виконання пов’язаних дій. MacroEngine виконує макроси, що складаються з послідовності кроків. Scheduler відповідає за періодичний запуск правил за розкладом. RuleRepository і інші репозиторії інкапсулюють доступ до таблиць бази даних.

Діаграма послідовностей



Діаграма послідовностей (рис. 3)

Розглянуто два основні сценарії роботи системи: створення нового правила автоматизації користувачем та автоматичний запуск правила за розкладом.

1. Створення правила автоматизації:
• Користувач через Automation UI надсилає запит на створення правила до сервера.
• Listener приймає запит і передає його до Validator, який перевіряє коректність даних.
• Після успішної валідації запит передається до RuleEngine, який формує модель правила.
• RuleEngine через RuleRepository надсилає команду на збереження правила у базі даних.
• База даних підтверджує збереження, після чого користувачу повертається відповідь про успішне створення правила.

2. Автоматичний запуск правила за розкладом:
• Scheduler періодично ініціює перевірку активних правил.
• RuleEngine отримує перелік правил, термін яких настав, через RuleRepository.
• Для кожного правила RuleEngine перевіряє умови та виконує дії (наприклад, надсилання запиту до зовнішнього сервісу).
• Результати виконання записуються в ExecutionLog через репозиторій.
• При потребі користувач може переглянути статистику виконання правила через Automation UI.

Висновок

У ході виконання даної лабораторної роботи було побудовано UML-діаграми для системи Flexible Automation Tool: діаграму розгортання, діаграму компонентів та діаграму послідовностей. Діаграма розгортання відобразила фізичну структуру системи та взаємодію між клієнтом, сервером та базою даних. Діаграма компонентів дозволила виділити основні модулі фронтенду, бекенду та бази даних. Діаграма послідовностей показала динаміку обробки запитів на створення правила та запуск правил за розкладом. Отримані результати можуть бути використані як основа для подальшої реалізації та документування системи.