УДК:004.415:004.75:551.5

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КЛІЄНТСЬКОГО МОДУЛЯ ОТРИМАННЯ КЛІМАТИЧНИХ ДАНИХ У WPF-ЗАСТОСУНКУ

Грибенко Д.М., студент
Київський національний університет технологій та дизайну
Мительська О.В., кандидат технічних наук, доцент
Київський національний університет технологій та дизайну
Афтанділянц В.Є., кандидат економічних наук, доцент
Київський національний університет технологій та дизайну

Ключові слова: WPF-застосунок, клієнт-серверна архітектура, асинхронне програмування, доменна модель даних, кліматичні дані, API-клієнт.

Модуль отримання кліматичних даних реалізовано із застосуванням класу HttpClient, який у середовищі .NET виступає стандартним засобом організації HTTP-комунікації. Виокремлення API-клієнта в окремий програмний компонент забезпечує логічну ізоляцію мережевої взаємодії від інтерфейсного та аналітичного рівнів системи. Такий підхід відповідає принципу *separation of concerns*, згідно з яким функціональні блоки, відповідальні за інтеграцію із зовнішніми сервісами, не повинні перетинатися з модулями візуалізації або обробки даних [1-4].

У межах кліматичної аналітичної системи це має критичне значення, оскільки зовнішні API характеризуються потенційною нестабільністю: можливі зміни структури відповіді, затримки або тимчасова недоступність сервісу. У зв'язку з цим внутрішня предметна логіка застосунку не повинна напряму оперувати JSON-структурами, а має використовувати уніфіковану доменну модель даних.

Процес отримання даних ініціюється вибором регіону в графічному інтерфейсі користувача. У WPF-реалізації елементи ComboBox містять асоційовані географічні координати, які зберігаються у властивості Tag. Наприклад, для м. Київ використовуються значення 50.4501 (широта) та 30.5234 (довгота). Це дозволяє відокремити візуальне представлення регіону від технічних параметрів API-запиту та забезпечує коректне відображення концепції адаптера між UI-рівнем і географічною моделлю даних.

Наступним етапом є формування часових меж аналізу відповідно до обраного періоду спостереження. У системі реалізовано інтервали тривалістю 30, 90 та 180 днів. При цьому кінцева дата зміщується відносно поточного моменту часу, що зумовлено особливостями архівних кліматичних сервісів, де актуалізація даних може відбуватися із затримкою. Таким чином, враховується часовий лаг між фактичними метеорологічними вимірюваннями та їх публікацією у відкритих API.

Формування HTTP-запиту здійснюється динамічно на основі вибраних параметрів: географічних координат, інтервалу дат, переліку метеорологічних показників та часової зони. Для аналізу обрано набір змінних: temperature_2m_mean, precipitation_sum та wind_speed_10m_max, що забезпечує комплексне відображення температурного режиму, рівня опадів та інтенсивності вітрової активності. Сукупність цих параметрів дозволяє виконувати базовий багатофакторний аналіз кліматичної динаміки.

Після формування запиту API-клієнт виконує асинхронний HTTP GET-запит. Використання асинхронної моделі на основі await є принципово важливим для WPF-додатків, оскільки запобігає блокуванню UI-потоку. У разі синхронного виконання мережевих операцій інтерфейс втрачає реактивність, що негативно впливає на користувацький досвід. Асинхронний підхід забезпечує реалізацію принципу *responsive UI*.

Після отримання відповіді виконується перевірка HTTP-статусу. У разі успішного виконання запиту (код 200 OK) JSON-відповідь передається до методу ParseOpenMeteo, який виконує її структурну нормалізацію. Для обробки використовується JsonDocument, що дозволяє працювати з ієрархічними JSON-структурами без жорсткого зв'язування з класами десеріалізації.

Ключовим елементом відповіді API є об'єкт daily, що містить паралельні масиви значень (дати, температури, опади, швидкість вітру). На етапі нормалізації ці масиви трансформуються у набір об'єктів доменної моделі ClimateRecord. Кожен об'єкт формується шляхом об'єднання відповідних індексів масивів (наприклад, time[i], temperature_2m_mean[i], precipitation_sum[i], wind_speed_10m_max[i]), що забезпечує перехід від табличної JSON-структури до об'єктно-орієнтованого представлення даних.

Така трансформація є важливою з архітектурної точки зору, оскільки дозволяє відокремити аналітичний та інтерфейсний рівні від формату зовнішнього API та забезпечує стабільність внутрішньої моделі даних.

На рис. 1 подано sequence-діаграму взаємодії компонентів системи, яка відображає послідовність обміну даними між користувачем, WPF-застосунком, API-клієнтом, зовнішнім сервісом Open-Meteo, аналітичним модулем та компонентом візуалізації.

Послідовність взаємодії демонструє реалізацію багатошарової архітектури, де кожен компонент виконує чітко визначену роль: інтерфейс ініціює запит, API-клієнт забезпечує комунікацію із зовнішнім сервісом, аналітичний модуль виконує нормалізацію та обробку даних, а модуль візуалізації формує графічне представлення результатів.

Окрему увагу приділено механізму обробки помилок. У разі недоступності зовнішнього API, некоректної відповіді або виникнення винятків під час HTTP-комунікації система не завершує роботу аварійно.

Натомість активується режим *demo fallback*, у межах якого викликається метод LoadDemoData, що генерує синтетичні часові ряди з урахуванням сезонності, випадкових коливань та аномальних значень.

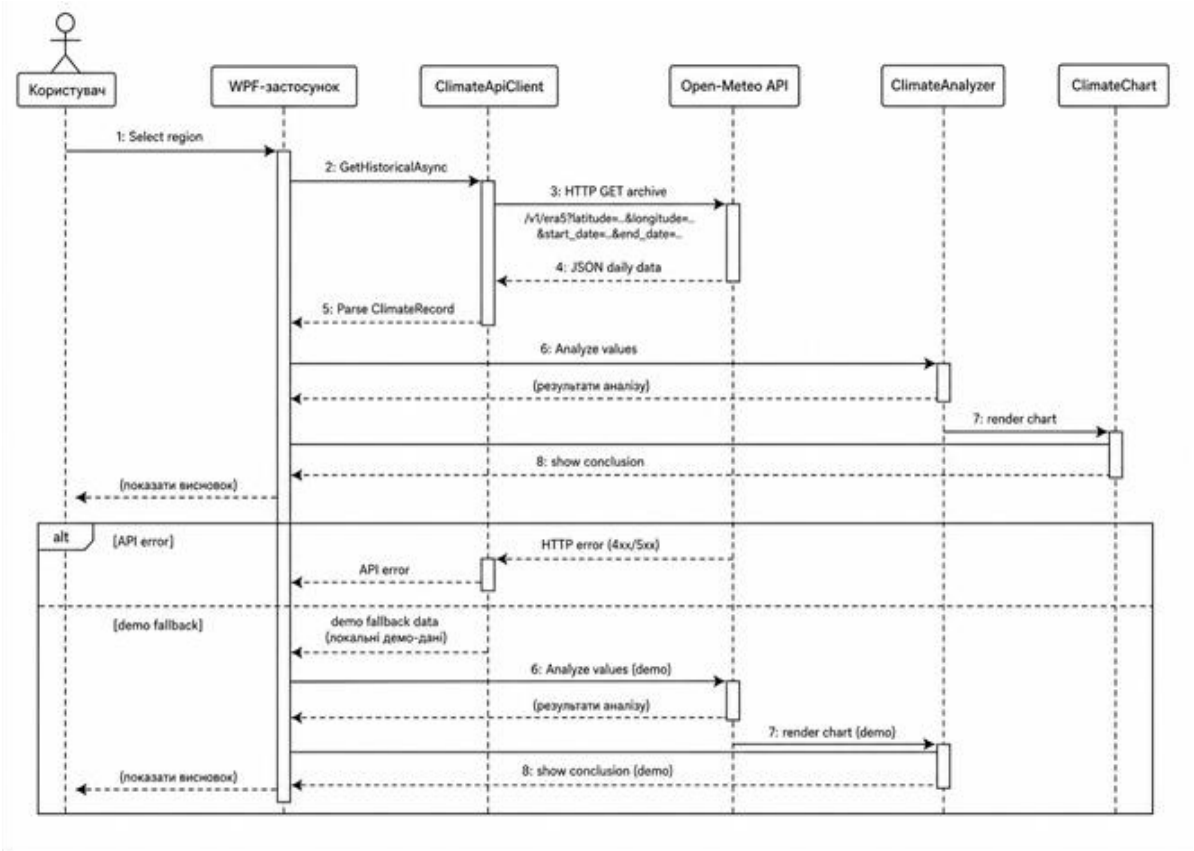

Рисунок 1 – Sequence-діаграма взаємодії компонентів системи

Такий підхід реалізує принцип *graceful degradation*, відповідно до якого система при частковій відмові зовнішніх компонентів зберігає базову функціональність. Це дозволяє забезпечити безперервність роботи інтерфейсу та незалежне тестування аналітичних і візуалізаційних модулів навіть за відсутності реальних зовнішніх даних.

Список використаних джерел

1. Troelsen A., Japikse P. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming. – New York: Apress, 2022. – 1640 p.

2. Price M. C# 12 and .NET 8 – Modern Cross-Platform Development Fundamentals. – Birmingham: Packt Publishing, 2023. – 828 p.

3. Robert C. Martin Series. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston: Prentice Hall, 2018. – 420 p.

4. Бородкіна І. Л. Інженерія програмного забезпечення: Посібник для студентів вищих навчальних закладів /І. Л. Бородкіна, Г. О. Бородкін. – К.: Центр учбової літератури, 2020. – 204 с.