МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Практична робота № 2**

з дисципліни « Архітектура програмного забезпечення .Net »

*назва дисципліни*

на тему: «ПАРАЛЕЛЬНІ ЗАДАЧІ ТА МЕРЕЖЕВА ВЗАЄМОДІЯ»

Виконав: студент 2 курсу групи № 622п

освітньої програми

121 інженерія програмного забезпечення

(шифр і назва ОП)

Зайченко Ярослав Ігорович

(прізвище й ініціали студента)

Прийняв: к. т. н., доцент

Лучшев Павло Олександрович

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2023

**ЗМІСТ**

[Постановка завдання 3](#_Toc159492133)

[Порядок виконання роботи 3](#_Toc159492134)

[Виконання роботи 5](#_Toc159492135)

[Висновок 11](#_Toc159492136)

Мета роботи: Вивчення особливостей розробки та програмування паралельних задач, мережева взаємодія та закріплення навичок об'єктно-орієнтованого програмування.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Складність | Вимоги до роботи | Бали | Оцінка |
|  | Базовий рівень | Створення та налаштування проекту для роботи з *Telegram API* | 1 | 1 |
|  | Використання елементів *Telegram* як меню команд, "клавіатур" для організації інтерфейсу користувача | 1 | 1 |
|  | Реалізація функціональності чат-бота відповідно до обраної предметної області (текстові повідомлення та команди) | 3 | 3 |
|  | Підвищений рівень (самостійна робота) | Використання мультимедійного контенту у процесі взаємодії з користувачем (на додаток до п.3) | 1 |  |
|  | Розгортання чат-бота на зовнішньому/хмарному хостингу | 2 | 2 |

Постановка завдання

Розробити програмний чат-бот для Telegram, використовуючи методи та засоби паралельного програмування платформи .Net. Бот повинен мати базовий набір функціональних можливостей для взаємодії з користувачами, включаючи відправлення повідомлень, обробку команд та використання асинхронних методів для ефективної роботи. Складність і функціональність бота визначається самостійно (див. табл. 2.1) і вибирається по одному з наступних напрямків:

* предметна область відповідає курсовому проекту з ООП (допускається реалізація як самостійної програми чат-бота, так і з інтеграцією безпосередньо у курсовий проект);
* предметна область визначається на основі даних з одного або декількох відкритих та загальнодоступних Web API. Чат-бот фактично організує інтерфейс користувача в Telegram для інформації, яка отримується з вибраного Web API, наприклад, за допомогою методів класу HttpClient;
* Інші варіанти предметної галузі (після обговорення з викладачем).

Порядок виконання роботи

1. У Telegram знайти спеціальний вбудований бот BotFather, в якому за допомогою команди /newbot створити власний чат-бот та отримати його API-Token. Зверніть увагу, що API-Token є секретом та не підлягає розголошенню!
2. Створити новий проект у середовищі розробки Visual Studio чи іншій IDE для .Net . Тип проекту може бути будь-яким.
3. Налаштувати проект для роботи з Telegram API (наприклад, через менеджер NuGet встановити пакет Telegram.Bot).
4. Створити у програмі екземпляр класу TelegramBotClient з отриманим API-Token.
5. У створеному у п.4. об'єкті реалізувати та підключити обробники помилок та повідомлень бота.
6. Реалізувати код для обробки команд (наприклад, /start, /help тощо на власний розсуд), які керують логікою роботи чат-бота.
7. Реалізувати код надсилання повідомлень користувачу та обробки повідомлень, які надходять від користувача.
8. Використовувати асинхронні методи для мережної взаємодії та виконання тривалих операцій (наприклад, запити до стороннього API).
9. Протестувати бота на локальній машині та/або розгорнутого на хостингу.

Виконання роботи

Опис програми:

Мова програмування: С#, операційна система Windows 11 Prо, Версія 23H2, Збірка ОС 22621.1325, процесор: Apple Silicon M1 Pro 3.20 GHz (ядер: 6), компілятор: Microsoft Visual Studio Community 2022 (64-розрядна версія ARM).

Створення нового боту

Для виконання цього завданян необхідно зкористатись <https://t.me/BotFather>. Результат на рисунку 1.

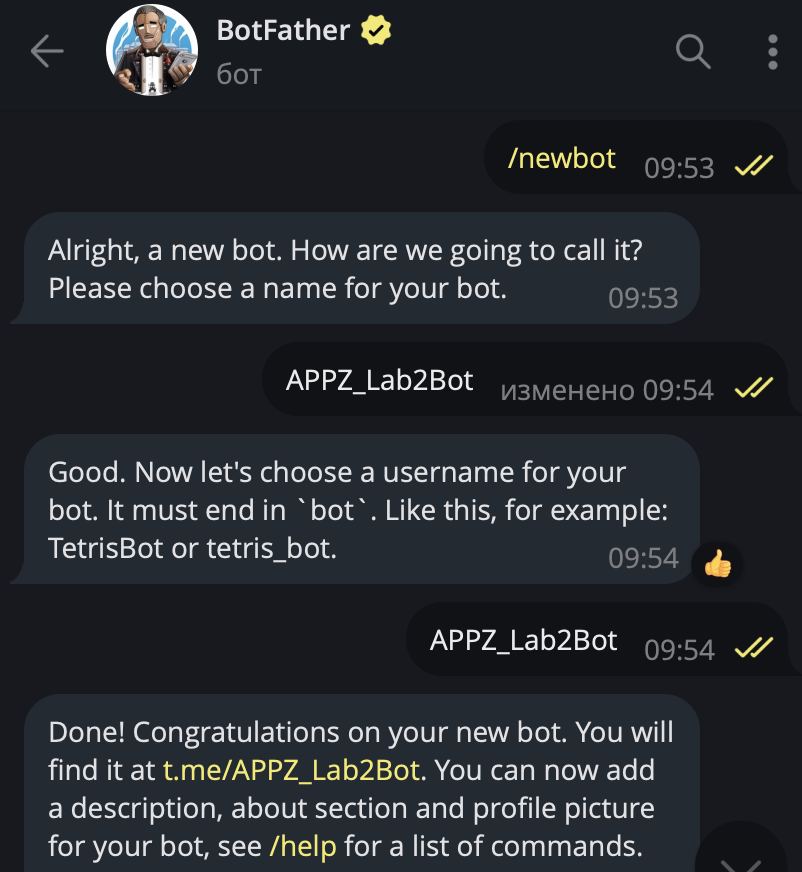


Рисунок 1 – створення бота

**Налаштування проєкту для роботи з Telegram API**

Для роботи з ботом в Телеграмі необхідно встановити NuGet пакет з назвою “Telegram.Bot”. Для цього було використано команди NuGet менеджеру, а саме: перехід до директорії проєкту та команда встановлення пакету. Результат встановлення на рисунку 2.

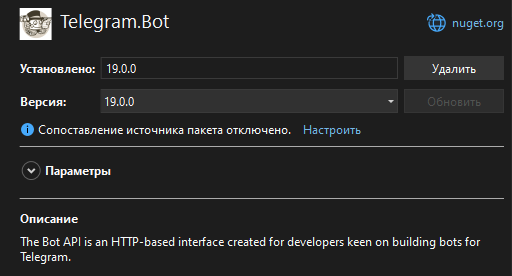


Рисунок 2 – встановлення пакету для роботи з ботами

Створення у програмі екземпляру класу TelegramBotClient з отриманим API-Tokenʼом від BotFather.

Поданий нижче фрагмент коду реалізує поставлену задачу

TelegramBotClient botClient = new TelegramBotClient("69\*\*\*\*\*\*86:AAH5Yt\*\*\*\*\*\*\*Xf\_3x\_CNGVc\_B5\*\*\*\*pyAE");

Реалізація та підключення обробників помилок та повідомлень бота.

Даний фрагмент коду відповідає за прийняття та оброблення повідомлень ботом:

async Task HandleUpdateAsync(ITelegramBotClient botClient, Update update, CancellationToken cancellationToken)

{

try

{

if (update.Type == UpdateType.Message)

{

// Only process Message updates: https://core.telegram.org/bots/api#message

if (update.Message is not { } message)

return;

// Only process text messages

if (message.Text is not { } messageText)

return;

var chatId = message.Chat.Id;

Debug.WriteLine($"Received a '{messageText}' message in chat {chatId} from {message.Chat.FirstName}.");

**//подальше оброблення повідомлень**

}

}

}

Даний фрагмент коду відповідає за оброблення помилок в процесі роботи з Telegram API:

Task HandlePollingErrorAsync(ITelegramBotClient botClient, Exception exception, CancellationToken cancellationToken)

{

var ErrorMessage = exception switch

{

ApiRequestException apiRequestException

=> $"Telegram API Error:\n[{apiRequestException.ErrorCode}]\n{apiRequestException.Message}",

\_ => exception.ToString()

};

Debug.WriteLine(ErrorMessage);

return Task.CompletedTask;

}

Реалізація коду для обробки команд

Для обробки команд було створено наступні команди:

* private async Task HandleStartCommandAsync(ITelegramBotClient botClient, long chatId, CancellationToken cancellationToken) – обробка команди /start;
* private async Task HandleWeatherCommandAsync(ITelegramBotClient botClient, long chatId, string cityName, CancellationToken cancellationToken) – обробка команди /weather;
* private async Task HandleHelpCommandAsync(ITelegramBotClient botClient, long chatId, CancellationToken cancellationToken) – обробка команди /help;
* private async Task HandleChangeCityCommandAsync(ITelegramBotClient botClient, long chatId, CancellationToken cancellationToken) – обробка команди /changecity;

Реалізація коду надсилання повідомлень користувачу та обробки повідомлень

Для надсилання повідомлень користувачеві використовується метод SendTextMessageAsync з ITelegramBotClient.

Для обробки вхідних повідомлень використовується метод HandleUpdateAsync. Він аналізує тип оновлення та виконує відповідні дії.

Використання асинхронних методів

В коді программи було створено наступні асинхронні методи:

* async Task HandleUpdateAsync(ITelegramBotClient botClient, Update update, CancellationToken cancellationToken)
* private async Task HandleChangeCityCommandAsync(ITelegramBotClient botClient, long chatId, CancellationToken cancellationToken)
* private async Task HandleStartCommandAsync(ITelegramBotClient botClient, long chatId, CancellationToken cancellationToken)
* private async Task HandleWeatherCommandAsync(ITelegramBotClient botClient, long chatId, string cityName, CancellationToken cancellationToken)
* private async Task HandleHelpCommandAsync(ITelegramBotClient botClient, long chatId, CancellationToken cancellationToken)
* private async Task HandleWeatherRequest(ITelegramBotClient botClient, long chatId, string cityName) – метод для звернення до відкритого API.

Розгортання програми на хостингу (використано Docker)

Щоб розгорнути програму на сервері було обрано метод створення DockerImage, що дозволяє ефективно та в подальшому безпомилково розгортати додаток на великій кількості хостингів.

Для цього спочатку необхідно написати інструкції в файлі з назвою Dockerfile. Його вміст поданий нижче:

# Use the official .NET SDK image as a build stage

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build

WORKDIR /src

COPY ["cmd\_noGUI\_for\_Docker\_Bot.csproj", "./"]

RUN dotnet restore "cmd\_noGUI\_for\_Docker\_Bot.csproj"

COPY . .

RUN dotnet build "cmd\_noGUI\_for\_Docker\_Bot.csproj" -c Release -o /app/build

RUN dotnet publish "cmd\_noGUI\_for\_Docker\_Bot.csproj" -c Release -o /app/publish

# Use the official .NET runtime image for the final stage

FROM mcr.microsoft.com/dotnet/runtime:8.0

WORKDIR /app

COPY --from=build /app/publish .

ENTRYPOINT ["dotnet", "cmd\_noGUI\_for\_Docker\_Bot.dll"]

Далі необхідно запустити програму Docker та в папці з проєктом відкрити термінал/консоль, та ввести наступні команди:

* docker build -t my-telegram-weather-bot . – будуємо образ контейнеру (рис. 3)
* docker run -d --name my-telegram-bot-container my-telegram-weather-bot – команда запускає створений контейнер з програмою (це також можна зробити і через графічний інтерфейс DockerDesktop)

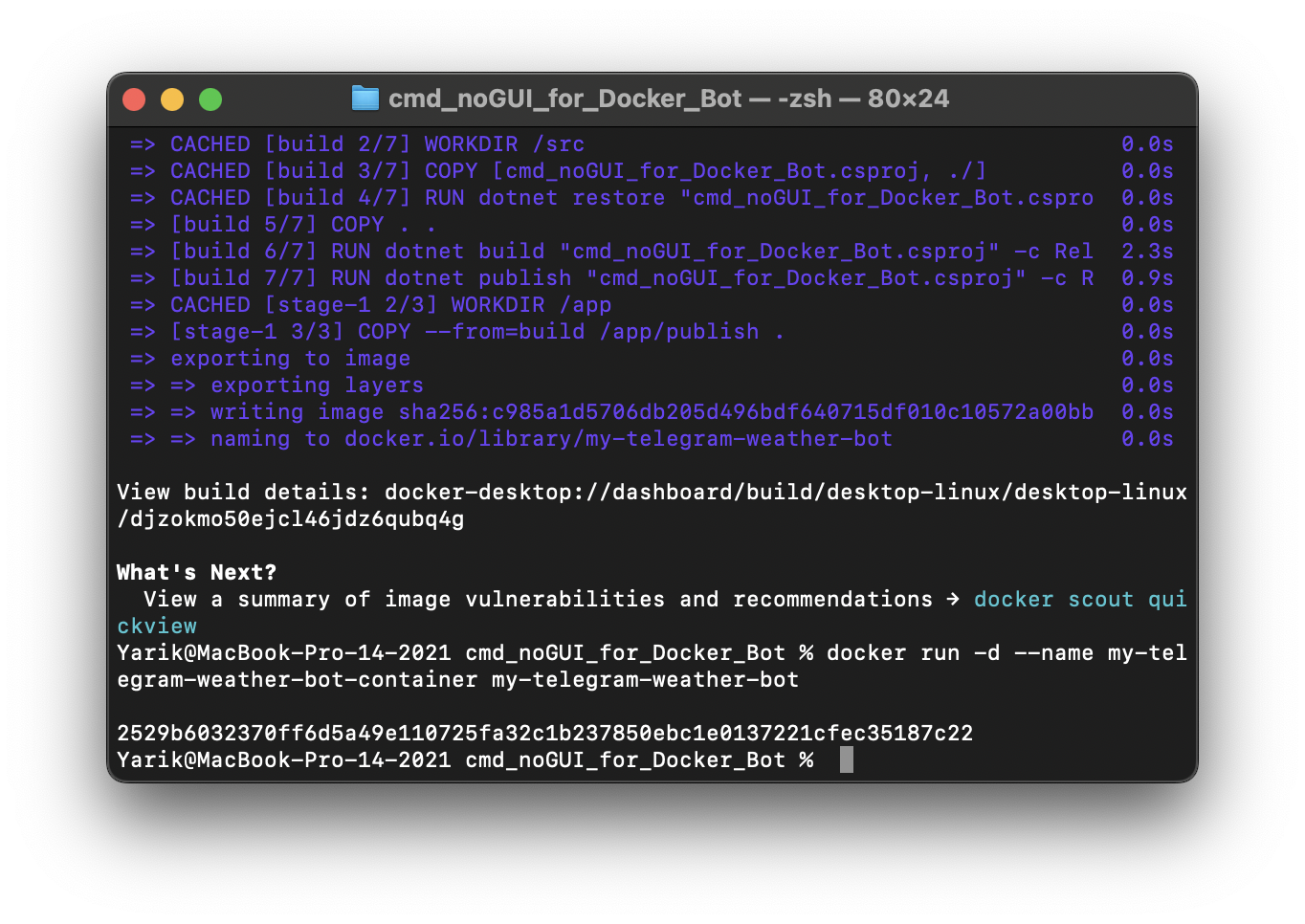


Рисунок 3 – результат побудови образу

Запущений контейнер має такий вигляд в програмі DockerDesktop (рис. 4)

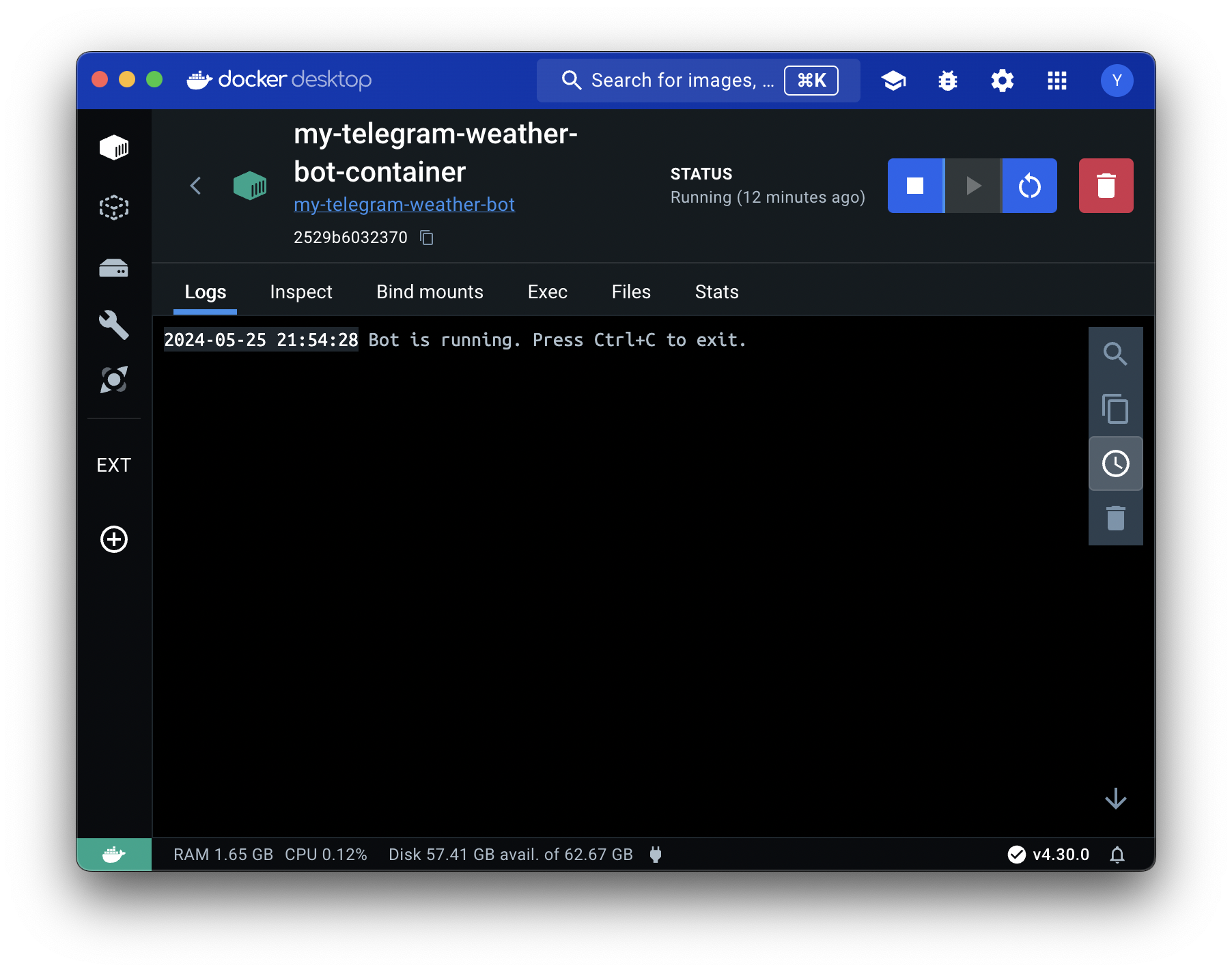
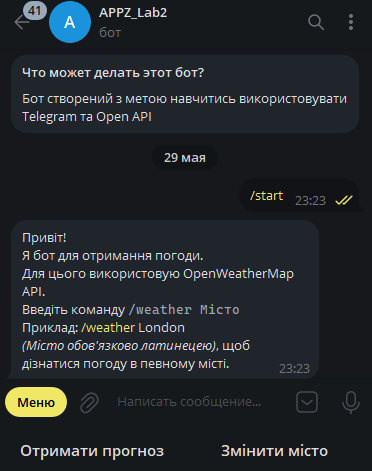


Рисунок 4 – робота контейнеру

Тестування роботи бота

Робота бота подана на рисунках нижче (рис. 5-10)

Рисунок 5 – команда старт

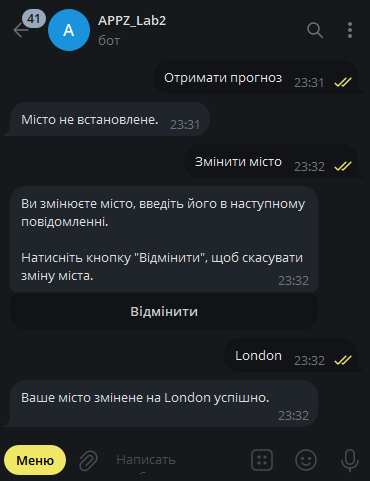


Рисунок 1 – команда встановлення/зміни міста

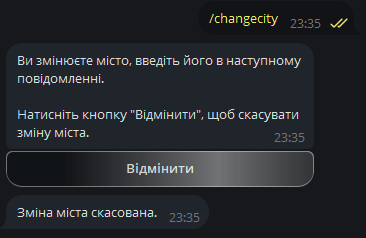


Рисунок 1 – відміна встановлення/зміни міста

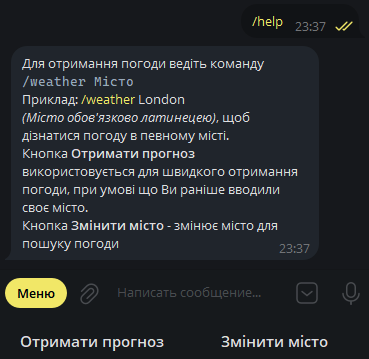


Рисунок 1 – команда для виклику повідомлення з допомогою

Висновок

У результаті виконання завдання було розроблено програмний чат-бот для Telegram, який відповідає вимогам щодо використання методів та засобів паралельного програмування платформи .Net. Бот має базовий набір функціональних можливостей для взаємодії з користувачами, включаючи відправлення повідомлень, обробку команд та використання асинхронних методів для ефективної роботи.

Спершу у Telegram був створений чат-бот за допомогою BotFather, що дозволило отримати необхідний API-Token для подальшої роботи. У середовищі розробки Visual Studio був створений новий проект та налаштований для роботи з Telegram API шляхом встановлення пакету Telegram.Bot через менеджер NuGet.

У програмі був створений екземпляр класу TelegramBotClient з отриманим API-Token, а також реалізовані та підключені обробники помилок та повідомлень бота. Реалізовані обробники для команд /start, /help, /weather та інших, які керують логікою роботи чат-бота та забезпечують взаємодію з користувачами. Реалізований код для надсилання повідомлень користувачу та обробки повідомлень, які надходять від користувачів. Були використані методи `SendTextMessageAsync` для відправки повідомлень та `HandleUpdateAsync` для обробки вхідних повідомлень.

Для мережної взаємодії та виконання тривалих операцій, таких як запити до стороннього API (наприклад, OpenWeatherMap API), використовувалися асинхронні методи, що забезпечило ефективну роботу бота. Бот був протестований на локальній машині, що дозволило переконатися у його коректній роботі та відповідності поставленим вимогам.

Завдяки виконанню даного завдання було закріплено навички паралельного програмування та асинхронної роботи у .Net, а також вивчено особливості створення та використання Telegram чат-ботів. Отриманий досвід дозволяє зручно створювати інтерактивні додатки, які можуть ефективно взаємодіяти з користувачами через месенджери, зокрема, Telegram.