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 та проектування програмного забезпечення .Net »

на тему: « Паралельні задачі та мережева взаємодія »

Виконав: студент 2 курсу групи № 621п

121 «Інженерія програмного забезпечення»

(код спеціальності)

\_\_\_\_ \_Бахшалієв А. Е.\_\_\_ \_\_\_\_\_\_

(ПІБ студента)

Прийняв: ст. викладач каф. 603

\_\_\_\_\_\_Сьомочкін М. О., Дем’яненко В. А.\_\_\_\_\_\_

(ПІБ викладача)

Національна шкала:

Кількість балів:

Харків – 2024

**ЗМІСТ**

[**ПОСТАНОВКА ЗАДАЧІ** 3](#_Toc166572416)

[**ВИКОНАННЯ РОБОТИ** 3](#_Toc166572417)

[Теоретичний матеріал 3](#_Toc166572418)

[Створення телеграм-бота 5](#_Toc166572419)

[Лістинг програми 8](#_Toc166572420)

[**ВИСНОВОК** 14](#_Toc166572421)

# **ПОСТАНОВКА ЗАДАЧІ**

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

**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 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).

# **ВИКОНАННЯ РОБОТИ**

## Теоретичний матеріал

Розуміння конкурентності, багатопоточності, асинхронності та паралелізму має важливе значення при розробці програмного забезпечення, оскільки це дозволяє створювати швидкі, чуйні та ефективні програми. Ці концепції допомагають керувати ресурсами комп'ютера, такими як процесорний час та пам'ять, оптимізуючи виконання завдань. Асинхронне та паралельне програмування підвищує чуйність додатків і дозволяє їм працювати більш ефективно у багатозадачному середовищі. Крім того, використання паралелізму особливо важливе для ефективного використання багатоядерних процесорів, що забезпечує масштабованість та збільшує продуктивність додатків.

***Multithreading (Багатоточність)***

- Багатопоточність є підхід до проблеми багатозадачності, при якому кожне завдання виконується в окремому потоці.

– Потоки – це незалежні послідовності інструкцій, які можуть виконуватись паралельно всередині процесу.

– Багатопоточність дозволяє розпаралелювати виконання завдань на багатоядерних процесорах і покращувати чуйність додатків, наприклад, шляхом асинхронної обробки введення користувача.

***Конкурентність (Конкурентність)***

– Конкурентність у програмуванні означає виконання кількох завдань протягом одного часового періоду.

– Це може бути досягнуто через паралельне виконання завдань, де кілька операцій виконуються одночасно або виконання, що чергується, коли завдання перемикаються між собою для виконання.

– Конкурентність важлива для ефективного використання ресурсів та підвищення продуктивності програм під час роботи з безліччю завдань, які можуть бути виконані незалежно один від одного.

***Parallelism (Паралелізм)***

– Паралелізм у програмуванні означає виконання кількох завдань одночасно на багатоядерному процесорі або у розподіленій системі.

– Це відрізняється від конкурентності, оскільки паралелізм фактично включає одночасне виконання завдань, а не лише їхнє конкурентне виконання протягом одного часового періоду.

– Паралелізм може бути досягнутий за допомогою багатопоточності, а також за допомогою спеціалізованих інструментів та бібліотек, таких як *Parallel* класи на платформі. *Net* , які автоматично розподіляють завдання на кілька ядер процесора для більш ефективного виконання.

***Asynchronous (Асинхронність)***

– Асинхронне програмування є підходом до виконання операцій, при якому потік виконання не блокується в очікуванні завершення операції.

– Натомість програма/процес може продовжувати виконувати інші завдання або очікувати завершення операції без блокування основного потоку виконання.

– Асинхронність часто використовується для мережної взаємодії, обробки операцій введення-виводу, таких як читання файлів або виконання запитів до бази даних тощо, які можуть тривати час, але не вимагають активної участі процесора під час виконання.

Патерни *TAP*, *EAP* та *APM* надають розробникам різні інструменти для роботи з асинхронними операціями в .*Net*, і вибір конкретного патерну залежить від вимог проекту та доступних засобів розробки.

1. ***TAP (Task-based Asynchronous Pattern)*** :

– Цей патерн був введений у .*Net* *Framework 4.0* і став основним методом асинхронного програмування у *.Net*.

– *TAP* заснований на типі ***Task*** та використовує ключові слова ***async*** та ***await*** .

– Він забезпечує простий та зручний спосіб для створення асинхронних методів та очікування їх виконання без блокування потоку.

2. ***EAP (Event-based Asynchronous Pattern)*** ,

– Цей патерн був попередником *TAP* і часто використовується у старих *API* та бібліотеках.

– *EAP* базується на асинхронних подіях та зворотних викликах (***callback***).

– В *EAP* для кожної асинхронної операції зазвичай визначається пара подій: одна подія, що вказує на успішне завершення операції, та інша – на помилку.

3. ***APM (Asynchronous Programming Model)*** :

– Цей патерн також використовується у старих *API* та бібліотеках, і він передував як *TAP* , так і *EAP* .

– *APM* заснований на методах із суфіксами *Begin* та *End* , які запускають та завершують асинхронні операції відповідно.

– Він вимагає більш багатослівного та складного коду порівняно з *TAP* та *EAP* , і його використання вважається менш кращим у сучасному програмуванні.

На даний момент патерн *TAP* є найбільш рекомендованим та кращим підходом асинхронного програмування. *Net* , оскільки він забезпечує простоту, читання та масштабованість коду.

## Створення телеграм-бота



Рисунок 1 – Перевірка роботи бота з фразою “Give duck!”

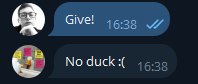


Рисунок 2 – Перевірка роботи бота з фразою “Give!

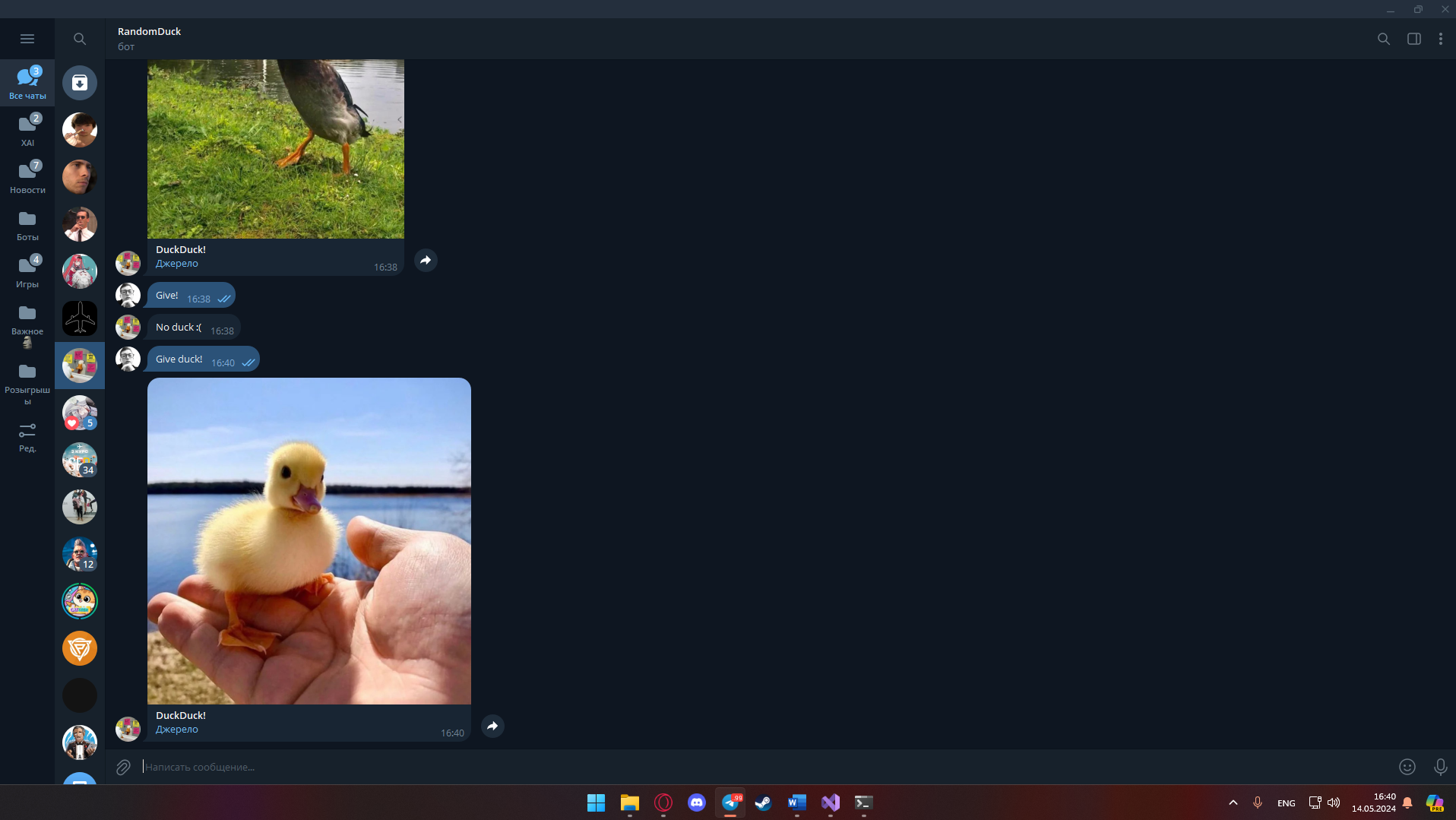


Рисунок 3 – Вид бота в телеграмі

## Лістинг програми

**Program.cs**

using Telegram.Bot.Types.ReplyMarkups;

using Telegram.Bot.Types;

using Telegram.Bot;

using Net\_Lab\_2;

class Program

{

static void Main(string[] args)

{

var client = new TelegramBotClient("6706397450:AAHX0OHzhpgSNsIF6UhG1t58b2YGjYJaNNg");

client.StartReceiving(Update, Error);

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"> Бота увiмкнено! ({DateTime.UtcNow} UTC)");

while (true)

{

string exitKeyword = "exit";

string input = Console.ReadLine();

if (input.ToLower() == exitKeyword)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"> Бота вимкнено! ({DateTime.UtcNow} UTC)");

break;

}

else

{

Console.ReadLine();

}

}

}

static async Task Error(ITelegramBotClient client, Exception, CancellationToken token)

{

}

static async Task Update(ITelegramBotClient botClient, Update update, CancellationToken token)

{

var message = update.Message;

if (message != null)

{

switch (message.Text)

{

case "Give duck!":

Console.WriteLine($"({message.Date.ToLocalTime()}) {message.Chat.Id}, {message.Chat.Username}: {message.Text}");

await BotActions.GetDuckImageOrGif(botClient, message.Chat.Id, token);

break;

default: botClient.SendTextMessageAsync(message.Chat.Id, "No duck :(");

break;

}

}

}

}

**BotActions.cs**

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Telegram.Bot;

using Telegram.Bot.Types.Enums;

using Telegram.Bot.Types;

using static System.Net.WebRequestMethods;

namespace Net\_Lab\_2

{

public static class BotActions {

public static async Task GetDuckImageOrGif(ITelegramBotClient botClient, long chatId, CancellationToken token)

{

var root = "https://random-d.uk/api/random";

using (HttpClient client = new HttpClient())

{

try

{

try

{

HttpResponseMessage response = await client.GetAsync(root);

if (response.IsSuccessStatusCode)

{

string jsonResult = await response.Content.ReadAsStringAsync();

var duckData = JsonConvert.DeserializeObject<dynamic>(jsonResult);

string imageUrl = duckData.url;

Console.WriteLine($"Завантажена свiтлина: {imageUrl}");

if (imageUrl != null)

{

await botClient.SendPhotoAsync(chatId,

photo: InputFile.FromUri(imageUrl),

caption: $"<b>DuckDuck!</b>\n<a href = '{imageUrl}'>Джерело</a>",

parseMode: ParseMode.Html,

cancellationToken: token);

}

}

else

{

Console.WriteLine($"Ошибка при загрузке изображения: {response.StatusCode}");

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при выполнении запроса: {ex.Message}");

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при выполнении запроса: {ex.Message}");

}

}

}

}

}

# **ВИСНОВОК**

Під час виконання 2 практичної роботи, було створено телеграм бота, використовуючи бібліотеку telegram.bot, та api для отримання інформації. Оформлено звіт.