**Практична робота №5**

**ЧАТ-БОТИ ТА МЕРЕЖЕВА ВЗАЄМОДІЯ**

Мета роботи : Ознайомитися з підходами до інтеграції месенджеру з backend- застосунком, побудувати чат-бот, який дозволяє користувачу виконувати CRUD- операції над сутностями предметної області, отримати практичні навички асинхронного та об'єктно-орієнтованого програмування.

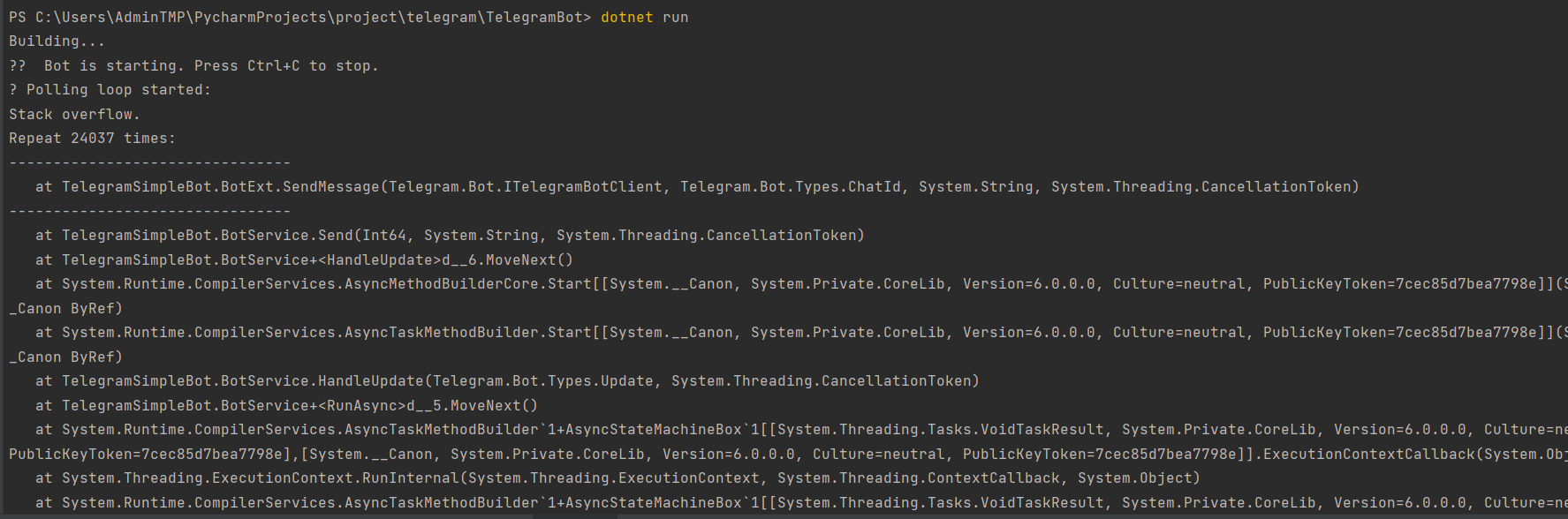
Завдання

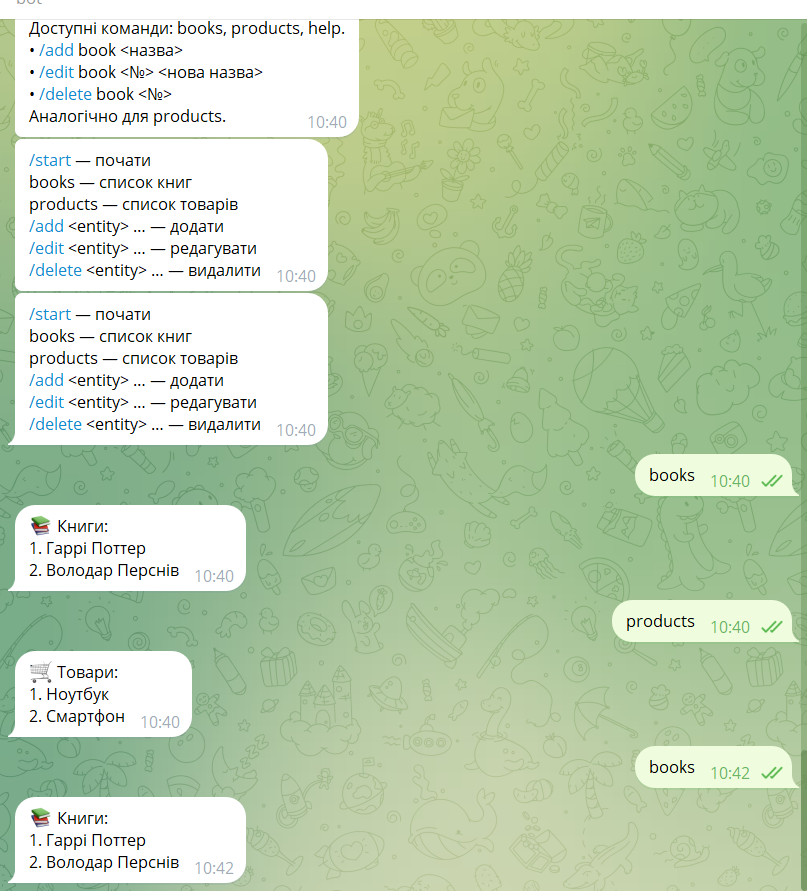
Розробити програмний чат-бот для Telegram [1], використовуючи методи та засоби паралельного програмування платформи .Net. Бот повинен мати базовий набір функціональних можливостей для взаємодії з користувачами, включаючи відправлення повідомлень, обробку команд та використання асинхронних методів для ефективної роботи. Складність і функціональність бота визначається самостійно (див. табл. 5.1) і вибирається по одному з наступних напрямків:

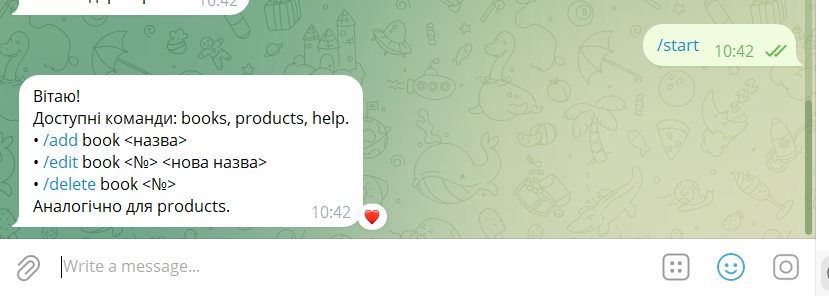
– предметна область визначається на основі Web API з свого варіанту практичної роботи №3. Чат-бот фактично організує інтерфейс користувача в Telegram з Web API, наприклад, за допомогою методів класу HttpClient [3];

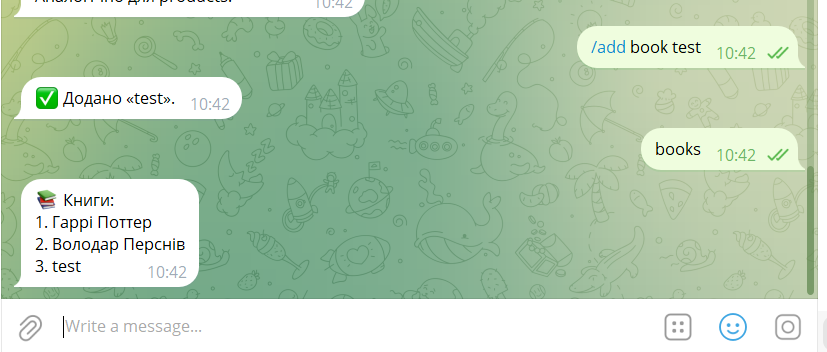
– предметна область відповідає курсовому проекту з ООП (допускається реалізація як самостійної програми чат-бота, так і з інтеграцією безпосередньо у курсовий проект);

- інші варіанти предметної галузі (після обговорення з викладачем).









using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading;  
using System.Threading.Tasks;  
using Telegram.Bot;  
using Telegram.Bot.Types;  
  
namespace TelegramSimpleBot;  
  
/// <summary>  
/// Найпростіший bot-loop на GetUpdates + in-memory «БД» книг / товарів.  
/// </summary>  
public sealed class BotService  
{  
 private readonly TelegramBotClient \_bot;  
 private int \_offset; // 🔹 offset GetUpdates (int у Bot API)  
  
 // "База даних" у пам'яті  
 private readonly List<string> \_books = new() { "Гаррі Поттер", "Володар Перснів" };  
 private readonly List<string> \_products = new() { "Ноутбук", "Смартфон" };  
  
 public BotService(string token) => \_bot = new TelegramBotClient(token);  
  
 /\*────────────────────────────────────── MAIN LOOP ──────────────────────────────────────\*/  
  
 public async Task RunAsync(CancellationToken ct)  
 {  
 Console.WriteLine("✅ Polling loop started… Ctrl-C to exit");  
  
 while (!ct.IsCancellationRequested)  
 {  
 try  
 {  
 var updates = await \_bot.GetUpdates(  
 offset : \_offset,  
 timeout : 30,  
 cancellationToken: ct);  
  
 foreach (var upd in updates)  
 {  
 \_offset = upd.Id + 1;  
 await HandleUpdate(upd, ct);  
 }  
 }  
 catch (Exception ex)  
 {  
 Console.WriteLine($"⚠️ {ex.Message}");  
 await Task.Delay(TimeSpan.FromSeconds(3), ct);  
 }  
 }  
 }  
  
 /\*────────────────────────────────────── HANDLERS ───────────────────────────────────────\*/  
  
 private async Task HandleUpdate(Update upd, CancellationToken ct)  
 {  
 if (upd.Message?.Text is not { } txt) return;  
  
 var chat = upd.Message.Chat.Id;  
 txt = txt.Trim();  
  
 // → команда та (можливо) аргументи:  
 var parts = txt.Split(' ', 3, StringSplitOptions.RemoveEmptyEntries);  
 var cmd = parts[0].ToLowerInvariant();  
  
 switch (cmd)  
 {  
 /\*──────────── базові ────────────\*/  
 case "/start":  
 await Send(chat,  
 "Вітаю!\n" +  
 "Доступні команди: books, products, help.\n" +  
 "• /add book <назва>\n" +  
 "• /edit book <№> <нова назва>\n" +  
 "• /delete book <№>\n" +  
 "Аналогічно для products.",  
 ct);  
 break;  
  
 case "help":  
 await Send(chat,  
 "/start — почати\n" +  
 "books — список книг\n" +  
 "products — список товарів\n" +  
 "/add <entity> … — додати\n" +  
 "/edit <entity> … — редагувати\n" +  
 "/delete <entity> … — видалити",  
 ct);  
 break;  
  
 /\*──────────── вивід списків ────────────\*/  
 case "books":  
 await Send(chat, FormatList("📚 Книги", \_books), ct);  
 break;  
  
 case "products":  
 await Send(chat, FormatList("🛒 Товари", \_products), ct);  
 break;  
  
 /\*──────────── CRUD ────────────\*/  
 case "/add":  
 await HandleAdd(parts, chat, ct);  
 break;  
  
 case "/edit":  
 await HandleEdit(parts, chat, ct);  
 break;  
  
 case "/delete":  
 await HandleDelete(parts, chat, ct);  
 break;  
  
 /\*──────────── fallback ────────────\*/  
 default:  
 await Send(chat, "Невідома команда. Напишіть help.", ct);  
 break;  
 }  
 }  
  
 /\*────────────────────────────────────── HELPERS ───────────────────────────────────────\*/  
  
 #region CRUD-helpers  
 private async Task HandleAdd(string[] parts, long chat, CancellationToken ct)  
 {  
 if (parts.Length < 3)  
 {  
 await Send(chat, "Синтаксис: /add book|product <назва>", ct);  
 return;  
 }  
  
 var target = parts[1].ToLowerInvariant();  
 var name = parts[2].Trim();  
  
 var list = target switch  
 {  
 "book" or "books" => \_books,  
 "product" or "products" => \_products,  
 \_ => null  
 };  
  
 if (list is null)  
 {  
 await Send(chat, "Можна додавати лише book або product.", ct);  
 return;  
 }  
  
 list.Add(name);  
 await Send(chat, $"✅ Додано «{name}».", ct);  
 }  
  
 private async Task HandleEdit(string[] parts, long chat, CancellationToken ct)  
 {  
 if (parts.Length < 4 || !int.TryParse(parts[2], out var idx) || idx < 1)  
 {  
 await Send(chat, "Синтаксис: /edit book|product <№> <нова назва>", ct);  
 return;  
 }  
  
 var target = parts[1].ToLowerInvariant();  
 var name = parts[3].Trim();  
  
 var list = target switch  
 {  
 "book" or "books" => \_books,  
 "product" or "products" => \_products,  
 \_ => null  
 };  
  
 if (list is null) { await Send(chat, "Редагувати можна book або product.", ct); return; }  
 if (idx > list.Count) { await Send(chat, "Індекс виходить за межі списку.", ct); return; }  
  
 list[idx - 1] = name;  
 await Send(chat, $"✏️ Запис №{idx} змінено на «{name}».", ct);  
 }  
  
 private async Task HandleDelete(string[] parts, long chat, CancellationToken ct)  
 {  
 if (parts.Length < 3 || !int.TryParse(parts[2], out var idx) || idx < 1)  
 {  
 await Send(chat, "Синтаксис: /delete book|product <№>", ct);  
 return;  
 }  
  
 var target = parts[1].ToLowerInvariant();  
  
 var list = target switch  
 {  
 "book" or "books" => \_books,  
 "product" or "products" => \_products,  
 \_ => null  
 };  
  
 if (list is null) { await Send(chat, "Видаляти можна book або product.", ct); return; }  
 if (idx > list.Count) { await Send(chat, "Індекс виходить за межі списку.", ct); return; }  
  
 var removed = list[idx - 1];  
 list.RemoveAt(idx - 1);  
 await Send(chat, $"🗑️ Видалено «{removed}».", ct);  
 }  
 #endregion  
  
 private static string FormatList(string title, IReadOnlyList<string> items) =>  
 items.Count == 0  
 ? $"{title}: (порожньо)"  
 : $"{title}:\n" +  
 string.Join('\n', items.Select((v, i) => $"{i + 1}. {v}"));  
  
 /\*\* єдиний виклик Bot API, який ми насправді використовуємо \*/  
 private Task Send(long chat, string text, CancellationToken ct) =>  
 \_bot.SendMessage(chat, text, cancellationToken: ct);  
}