**1. Введение**

Тема проекта: Telegram-бот

Цель проекта - изучить принципы создания Telegram-бота с использованием языка C#, реализовать работающее приложение, взаимодействующее с Telegram Bot API.

Функциональность:

• Обработка команд /start, /help, /info.

• Ответы на сообщения.

• Возможность расширения (например, ведение дневника, напоминания и др.

**2. Выбор технологий**

|  |  |  |
| --- | --- | --- |
| Компонент | Технология | Описание |
| Язык программирования | C# | Современный язык с мощной экосистемой |
| Telegram API | Telegram.Bot | Популярная .NET-библиотека для работы с Bot API |
| IDE | Visual Studio / Rider | Для разработки |
| Хранение данных | JSON / SQLite / PostgreSQL | Выбирается в зависимости от проекта |
| Хостинг | Render / Railway / VPS | Развёртывание Telegram-бота |
| Документация | Markdown, Draw.io, PlantUML | Для технического описания и UML |

**3. Исследование: как работает Telegram-бот на C#**

**3.1 Как работает Telegram Bot API**

Telegram Bot API — это HTTP-интерфейс, через который можно получать и отправлять сообщения, обрабатывать команды и взаимодействовать с пользователями. Есть два основных способа получения обновлений:

• Long Polling — бот запрашивает новые сообщения с интервалом

• Webhook — Telegram сам отправляет обновления на указанный URL

Для простоты мы будем использовать Long Polling, так как он не требует публичного сервера на первом этапе.

**3.2 Основы библиотеки Telegram.Bot для C#**

Telegram.Bot (https://github.com/TelegramBots/Telegram.Bot) — официальная .NET-библиотека для Telegram Bot API. Возможности:

• Отправка и получение сообщений.

• Обработка команд.

• Поддержка клавиатур и inline-кнопок.

• Работа с медиа и API Telegram.

**4. Пошаговое создание Telegram-бота на C#**

**Шаг 1: Регистрация бота в Telegram**

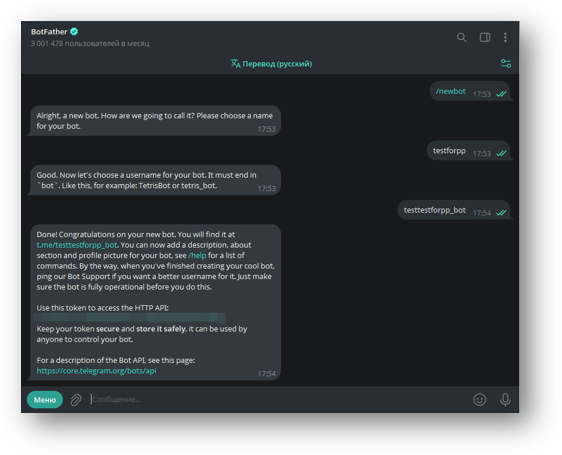
1. Откройте Telegram и найдите бота @BotFather (https://t.me/BotFather)

2. Отправьте команду /newbot

3. Укажите имя и username (например, MyTestBot)

4. Скопируйте токен, он выглядит так:

123456789:ABCDefGhIJKLmNoPQRsTUVwxyZ



**Шаг 2: Создание проекта**

1. Откройте Visual Studio

2. Создайте консольное приложение:

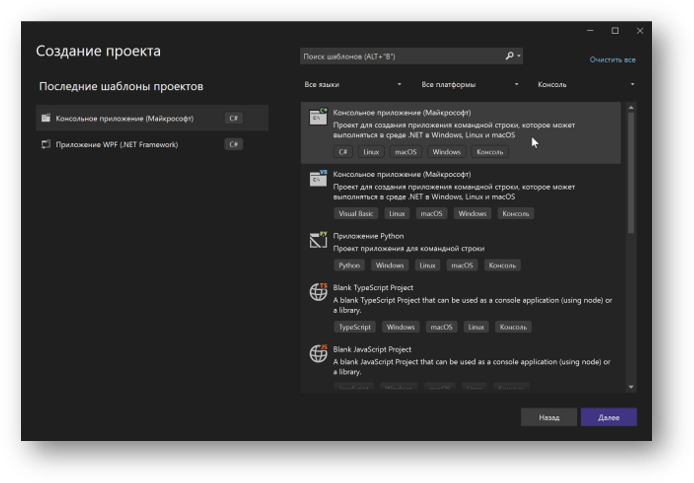
File → New → Project → Console App (.NET Core или .NET 6/8)

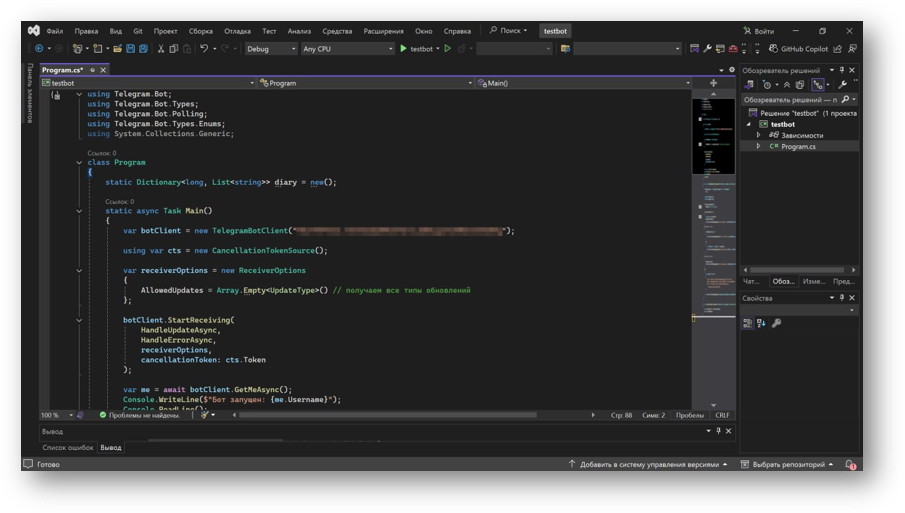
3. Установите NuGet-пакет:

dotnet add package Telegram.Bot

Или через Visual Studio:

Project → Manage NuGet Packages → Telegram.Bot



**Шаг 3: Базовая реализация бота**

Создайте файл Program.cs и вставьте следующий код:

using Telegram.Bot;

using Telegram.Bot.Types;

using Telegram.Bot.Polling;

using Telegram.Bot.Types.Enums;

class Program

{

static async Task Main()

{

var botClient = new TelegramBotClient("YOUR\_TOKEN\_HERE");

using var cts = new CancellationTokenSource();

var receiverOptions = new ReceiverOptions

{

AllowedUpdates = Array.Empty<UpdateType>() // получаем все типы обновлений

};

botClient.StartReceiving(

HandleUpdateAsync,

HandleErrorAsync,

receiverOptions,

cancellationToken: cts.Token

);

var me = await botClient.GetMeAsync();

Console.WriteLine($"Бот запущен: {me.Username}");

Console.ReadLine();

cts.Cancel();

}

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

{

if (update.Message is not { } message || message.Text is not { } messageText)

return;

Console.WriteLine($"Сообщение от {message.Chat.Id}: {messageText}");

string response = messageText switch

{

"/start" => "Привет! Я Telegram-бот на C#.",

"/help" => "Доступные команды: /start, /help, /info",

"/info" => "Этот бот написан на C# с использованием Telegram.Bot.",

\_ => $"Вы написали: {messageText}"

};

await botClient.SendTextMessageAsync(

chatId: message.Chat.Id,

text: response,

cancellationToken: cancellationToken);

}

static Task HandleErrorAsync(ITelegramBotClient botClient, Exception exception, CancellationToken cancellationToken)

{

Console.WriteLine($"Ошибка: {exception.Message}");

return Task.CompletedTask;

}

}

**Шаг 4: Запуск**

1. Вставьте свой токен в строку:

var botClient = new TelegramBotClient("YOUR\_TOKEN\_HERE");

2. Запустите проект: Ctrl + F5

3. Напишите боту в Telegram команду /start

**Итог:**

На этом этапе у нас есть базовый Telegram-бот, который:

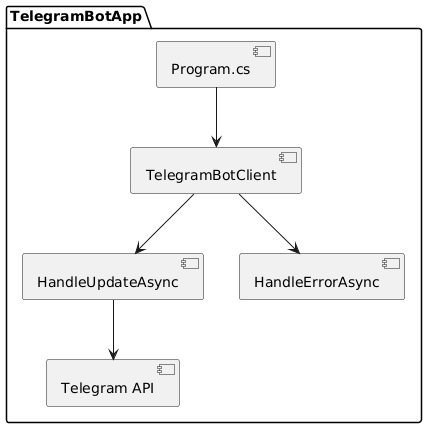
• Работает через Long Polling

• Обрабатывает команды /start, /help, /info

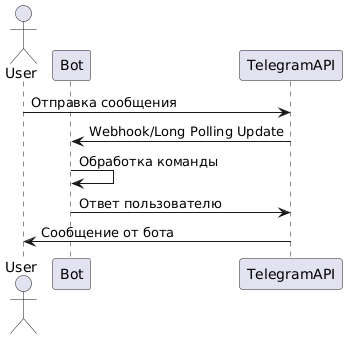
• Отвечает на любые текстовые сообщени

**5. Иллюстрации и диаграммы**

**5.1. UML-диаграмма компонентов**

****

**5.2. Диаграмма последовательности обработки сообщений**



**5.3. Схема логики обработки команд**

[Старт]

[Получено сообщение]

[Это команда?] -- нет --> [Ответ: "Вы написали..."]

да

[Команда /start?] --> Ответ: "Привет!"

[Команда /help?] --> Ответ: "Доступные команды..."

[Команда /info?] --> Ответ: "Информация о боте"

**6. Модификация проекта: Ведение дневника**

**Цель модификации**

Добавить в Telegram-бота возможность сохранять и просматривать дневниковые записи пользователя. Пользователь сможет:

• Добавить запись: /add Сегодня был тяжёлый день

• Посмотреть все записи: /list

• Очистить записи: /clear

**Техническая реализация**

Мы используем:

• Словарь Dictionary<long, List<string>> для хранения записей в памяти

• Команды: /add, /list, /clear

Позже это можно заменить на SQLite или сохранение в файл.

**Пример кода**

Добавим в Program.cs:

static Dictionary<long, List<string>> diary = new();

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

{

if (update.Message is not { } message || message.Text is not { } messageText)

return;

var chatId = message.Chat.Id;

var text = messageText.Trim();

if (!diary.ContainsKey(chatId))

diary[chatId] = new List<string>();

if (text.StartsWith("/add "))

{

string entry = text.Substring(5);

diary[chatId].Add(entry);

await botClient.SendTextMessageAsync(chatId, "Запись добавлена!", cancellationToken: cancellationToken);

}

else if (text == "/list")

{

if (diary[chatId].Count == 0)

{

await botClient.SendTextMessageAsync(chatId, "У вас нет записей.", cancellationToken: cancellationToken);

}

else

{

string allEntries = string.Join("\n— ", diary[chatId]);

await botClient.SendTextMessageAsync(chatId, $"Ваши записи:\n— {allEntries}", cancellationToken: cancellationToken);

}

}

else if (text == "/clear")

{

diary[chatId].Clear();

await botClient.SendTextMessageAsync(chatId, "Все записи удалены.", cancellationToken: cancellationToken);

}

else

{

string response = text switch

{

"/start" => "Привет! Я бот-дневник. Используйте /add, /list, /clear",

"/help" => "Команды: /add <запись>, /list, /clear",

"/info" => "Этот бот сохраняет ваши личные записи.",

\_ => "Неизвестная команда. Введите /help"

};

await botClient.SendTextMessageAsync(chatId, response, cancellationToken: cancellationToken);

}

}

**Идеи для будущей доработки**

• Добавить фильтрацию по дате

• Сохранять данные в базу

• Напоминания на основе записей

• Экспорт в PDF или TXT

**7 и 8 пункты по подключению кода к гиту и записыванию видео**

****