государственное автономное профессиональное образовательное учреждение

Самарской области

«Новокуйбышевский гуманитарно-технологический колледж»

**КУРСОВАЯ РАБОТА**

|  |  |
| --- | --- |
| по ПМ | 02 Осуществление интеграции программных модулей |
| МДК | 02.02 Инструментальные средства разработки программного обеспечения |
| Тема: | Создание функционального бота в мессенджере Telegram |

Федорченко Ивана Олеговича

студента 3 курса группы 36

специальность: 09.02.07

Информационные системы

и программирование

Форма обучения: очная

Преподаватель: Десятов И.А.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя:

\_\_\_\_\_\_\_\_\_\_\_\_\_ И.А. Десятов

2023

государственное автономное профессиональное образовательное учреждение

Самарской области

«Новокуйбышевский гуманитарно-технологический колледж»

Согласовано

на заседании предметно-цикловой комиссии

от \_\_\_\_\_\_\_протокол № \_\_\_\_

Председатель ПЦК

\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Баткова

**ЗАДАНИЕ**

для выполнения курсовой работы

|  |  |
| --- | --- |
| по ПМ | 02 Осуществление интеграции программных модулей |
| МДК | 02.02 Инструментальные средства разработки программного обеспечения |
| Тема: | Создание функционального бота в мессенджере Telegram |

курс 3 группа 36

|  |  |
| --- | --- |
| **Тема:** | Создание функционального бота в мессенджере Telegram |
| **Объект:** | Сетевые технологии. |
| **Предмет:** | Возможности сетевых технологий для разработки электронных учебных пособий. |
| **Цель:** | Создание функционального бота в Telegram. |
| **Задачи:** | 1. Изучение документации по API Telegram для разработки ботов. 2. Настройка окружения разработки для работы с Telegram ботом. 3. Реализация функционала получения и отправки сообщений ботом. 4. Разработка команды для отображения погоды по указанному городу. 5. Интеграция с соответствующим API для получения информации о погоде. 6. Создание команды для поиска страницы на Wikipedia по заданному запросу. 7. Использование соответствующего API для выполнения поиска на Wikipedia. 8. Добавление функционала отправки рандомных фотографий. 9. Интеграция с соответствующим API для получения рандомных фотографий. 10. Реализация команды "Об авторе" для отображения информации о создателе бота. 11. Тестирование функциональности бота и исправление возможных ошибок. 12. Документирование процесса разработки и результатов работы. |
| **Содержание:** | **Глава 1: Введение и изучение API Telegram** 1. Введение  1.1. Обзор темы  1.2. Цели и задачи работы  1.3. Обзор используемых технологий и инструментов  2. Изучение API Telegram  2.1. Регистрация и настройка бота  **Глава 2: Реализация функционального бота**  3. Настройка окружения разработки  3.1. Установка необходимых инструментов  4. Реализация функционала получения и отправки сообщений  4.1. Подключение к API Telegram  4.2. Обработка входящих сообщений  4.3. Отправка текстовых сообщений  5. Команда "Погода"  6. Команда "Поиск"    7. Команда "Рандомная картинка"  8. Команда "Об авторе"  9. Заключение  11. Список использованной литературы  12. Приложения  12.1. Исходный код проекта  12.2. Скриншоты и примеры работы бота |
|  |  |

специальность 09.02.07 Информационные системы и программирование

Дата выдачи «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2023 г.

Срок окончания «\_\_\_\_» \_\_\_\_\_\_\_\_2023 г.

Преподаватель\_\_\_\_\_\_\_\_ И.А. Десятов

Зам. директора \_\_\_\_\_\_\_\_ О.С. Макарова

**СОДЕРЖАНИЕ**

**Глава 1: Введение и изучение API Telegram**  
1. Введение

1.1. Обзор темы

1.2. Цели и задачи работы

1.3. Обзор используемых технологий и инструментов

2. Изучение API Telegram

2.1. Регистрация и настройка бота

**Глава 2: Реализация функционального бота**

3. Настройка окружения разработки

3.1. Установка необходимых инструментов

4. Реализация функционала получения и отправки сообщений

4.1. Подключение к API Telegram

4.2. Обработка входящих сообщений

4.3. Отправка текстовых сообщений

5. Команда "Погода"

6. Команда "Поиск"

7. Команда "Рандомная картинка"

8. Команда "Об авторе"

9. Заключение

11. Список использованной литературы

12. Приложения

12.1. Исходный код проекта

12.2. Скриншоты и примеры работы бота

**ВВЕДЕНИЕ**

В современном мире мессенджеры стали неотъемлемой частью нашей коммуникации. Они предоставляют удобный и быстрый способ общаться с другими людьми, а также получать различную информацию. Одним из популярных мессенджеров является Telegram, который предлагает широкие возможности для разработки и использования ботов.

Целью данной курсовой работы является создание функционального бота в мессенджере Telegram. Бот будет обладать набором полезных функций, которые позволят пользователям получать информацию о погоде, выполнять поиск в Википедии, получать случайные фотографии и многое другое. Создание такого бота требует знания языка программирования, использования API Telegram Bot, а также взаимодействия с внешними сервисами.  
**Объект** курсовой работы – создание функционального бота в мессенджере Telegram.  
**Предмет** курсовой работы – разработка и реализация бота для мессенджера Telegram с целью предоставления различных функциональных возможностей.  
**Цель** курсовой работы – cоздание и реализация функционального бота в мессенджере Telegram.   
**Задачи**:

1. Изучение API Telegram: изучение документации и функциональных возможностей API Telegram для разработки ботов.
2. Реализация базового функционала: создание скелета бота, обработка команд и сообщений от пользователей, отправка текстовых сообщений.
3. Интеграция дополнительных функций: добавление функций, таких как отправка изображений, получение информации из внешних сервисов, выполнение поисковых запросов и т.д.
4. Тестирование и отладка: проверка функциональности бота, исправление ошибок и улучшение производительности.

**Глава 1: Введение и изучение API Telegram**

* 1. **Обзор темы**

API Telegram представляет собой набор программных интерфейсов и методов, предоставляемых Telegram, популярным мессенджером, для разработки ботов и приложений. Он предоставляет разработчикам возможность взаимодействовать с платформой Telegram, отправлять и получать сообщения, управлять чатами, работать с медиафайлами и другими функциями.

Основные черты API Telegram включают:

1. Асинхронное взаимодействие: API Telegram поддерживает асинхронные вызовы, что позволяет разработчикам выполнять несколько операций одновременно и эффективно обрабатывать большой поток сообщений.
2. Широкий функционал: API Telegram предоставляет различные методы для работы с сообщениями, чатами, контактами, медиафайлами и другими элементами платформы Telegram. Разработчики могут создавать ботов, которые выполняют разнообразные задачи, от отправки уведомлений до обработки команд пользователей.
3. Безопасность и приватность: API Telegram обеспечивает высокий уровень безопасности и приватности данных. Все сообщения передаются по протоколу HTTPS, а идентификация и аутентификация осуществляются с использованием токенов доступа и ключей API.
4. Расширяемость: API Telegram предоставляет возможность разработчикам создавать собственные библиотеки и инструменты для работы с платформой Telegram, а также интегрировать ботов с другими сервисами и приложениями.

В целом, API Telegram является мощным инструментом для разработки функциональных ботов в мессенджере Telegram. Он предоставляет широкий набор возможностей и гибкость в создании разнообразных приложений, обеспечивая удобство и эффективность взаимодействия с пользователем.

API Telegram имеет большую значимость в современном программировании и разработке мессенджеров. Вот несколько причин, почему API Telegram является важным:

1. Гибкость и масштабируемость: API Telegram предоставляет разработчикам гибкость в создании различных типов ботов и приложений, от простых информационных до сложных интерактивных. Благодаря масштабируемости и возможности обработки большого количества запросов, API Telegram идеально подходит для создания ботов с большим числом пользователей.
2. Интеграция с другими сервисами: API Telegram позволяет интегрировать ботов с другими сервисами и приложениями. Разработчики могут использовать API для автоматизации процессов, обмена данными с другими платформами и создания синхронизированных решений для пользователей.
3. Широкая аудитория: Telegram имеет миллионы активных пользователей по всему миру. Создание бота с использованием API Telegram дает возможность достичь большой аудитории и обеспечить ей удобный доступ к вашим сервисам и функциям.
4. Безопасность и приватность: Telegram известен своей сильной защитой данных и приватностью пользователей. API Telegram также обеспечивает высокий уровень безопасности передачи данных и конфиденциальность пользователей, что делает его предпочтительным выбором для разработки приложений, где безопасность является приоритетом.

**1.2 Цели и задачи работы**

**Цель работы**: Создание функционального бота в мессенджере Telegram с использованием API Telegram.

В написании бота для Telegram возможны следующие задачи:

1. Анализ требований: Определение функциональности и основных возможностей бота, а также изучение потребностей целевой аудитории.
2. Изучение API Telegram: Ознакомление с документацией Telegram API, понимание его возможностей и ограничений.
3. Настройка и подключение к API Telegram: Создание бота в Telegram, получение токена для доступа к API и настройка обработки входящих сообщений.
4. Реализация функций бота: Написание кода для основных функций бота, таких как отправка и получение сообщений, обработка команд, работа с базой данных и т.д.
5. Тестирование и отладка: Проверка работоспособности бота, исправление ошибок и устранение проблем, связанных с взаимодействием с API Telegram.
6. Развитие и улучшение бота: Анализ обратной связи пользователей, добавление новых функций, оптимизация производительности, улучшение интерфейса и взаимодействия с пользователем.
7. Документирование и «деплой»: Написание документации по использованию бота, подготовка к «деплою» на выбранную платформу или хостинг.
8. Обеспечение безопасности: Разработка механизмов обработки и хранения пользовательской информации, защита от нежелательной активности и обеспечение безопасности данных.
9. Поддержка и сопровождение: Предоставление технической поддержки пользователям, регулярное обновление и обслуживание бота для поддержания его работоспособности.
   1. **Обзор технологий и инструментов**

Для разработки Telegram-бота на языке C# использованы следующие технологии и инструменты:

1. Язык программирования C#: C# (C-Sharp) является объектно-ориентированным языком программирования, разработанным Microsoft. Он широко используется для создания приложений на платформе .NET.
2. .NET Framework/.NET Core: .NET Framework и .NET Core представляют собой платформы разработки, которые обеспечивают инфраструктуру для создания и выполнения приложений на языке C#. Они предоставляют различные классы, библиотеки и инструменты для разработки приложений.
3. Telegram.Bot API: Это библиотека для работы с Telegram Bot API на языке C#. Она предоставляет удобные методы и классы для взаимодействия с API Telegram и разработки функционального бота.
4. Visual Studio или Visual Studio Code: Это интегрированные среды разработки (IDE), которые предоставляют широкий набор инструментов для разработки на языке C#. Они обеспечивают подсветку синтаксиса, отладку, автодополнение и другие полезные функции для удобной разработки.
5. NuGet: NuGet - это пакетный менеджер для платформы .NET, который позволяет управлять зависимостями и устанавливать необходимые библиотеки и пакеты для проекта.
6. JSON.NET: JSON.NET (Newtonsoft.Json) - это популярная библиотека для работы с форматом данных JSON. Она обеспечивает сериализацию и десериализацию объектов JSON, что может быть полезно при взаимодействии с Telegram API.

**2. Изучение API Telegram**

Изучение API Telegram является важной частью разработки бота для данного мессенджера. API Telegram предоставляет разработчикам возможность взаимодействовать с ботами и управлять ими через определенные методы и функции. В процессе изучения API Telegram рекомендуется уделить внимание следующим аспектам:

1. Ознакомление с документацией: Официальная документация Telegram API предоставляет подробную информацию о доступных методах, параметрах и объектах API. Изучение этой документации поможет понять основные концепции и возможности, предоставляемые Telegram API.
2. Работа с webhook или Long Polling: API Telegram поддерживает два основных метода получения обновлений - webhook и Long Polling. Webhook позволяет Telegram API отправлять обновления непосредственно на указанный веб-сервер, в то время как Long Polling предполагает периодические запросы к API для получения обновлений. Изучение и понимание работы обоих методов поможет определить наиболее подходящий способ получения обновлений для вашего бота.
3. Работа с сообщениями и командами: Основным взаимодействием бота с пользователями являются сообщения и команды. Изучение методов Telegram API, связанных с обработкой сообщений, позволит разработать функционал, который будет отвечать на сообщения пользователей, обрабатывать команды и предоставлять информацию.
4. Отправка медиа-файлов и клавиатур: API Telegram также предоставляет возможности отправки медиа-файлов (фотографии, видео, документы и т.д.) и настройки клавиатуры для взаимодействия с пользователем. Изучение соответствующих методов API позволит создать более интерактивный и функциональный бот.
5. Управление чатами и пользователями: Telegram API также предоставляет методы для управления чатами и пользователями, такие как получение информации о пользователях, управление административными правами и другие функции. Изучение этих методов поможет создать бота, способного эффективно взаимодействовать с пользователями.

В процессе изучения API Telegram рекомендуется также пробовать различные методы и экспериментировать с ними в рамках разработки прототипов и тестовых проектов. Это поможет лучше понять функциональность API и применений.

**2.1** **Регистрация и настройка бота**

Регистрация и настройка бота в Telegram включает несколько шагов:

1. Создание бота: Для создания бота вам понадобится аккаунт в Telegram. Зайдите в приложение Telegram и найдите бота @BotFather. Начните диалог с ним и следуйте его инструкциям для создания нового бота. В процессе создания вам будет предоставлен API-токен, который необходим для взаимодействия с API Telegram от имени вашего бота.
2. Получение API-токена: После создания бота, @BotFather выдаст вам API-токен. Сохраните этот токен в безопасном месте, так как он будет использоваться для авторизации вашего бота при отправке запросов к Telegram API.
3. Настройка параметров бота: Вы можете настроить различные параметры вашего бота, такие как имя, описание, аватар и т.д. Для этого обратитесь к @BotFather и следуйте инструкциям для изменения параметров вашего бота.
4. Подключение к API Telegram: Для взаимодействия с API Telegram от имени вашего бота, вам понадобится использовать Telegram Bot API. В зависимости от выбранного языка программирования, вам потребуется использовать соответствующую библиотеку или SDK для работы с Telegram API. Например, для языка C# можно использовать библиотеку Telegram.Bot.
5. Авторизация с помощью API-токена: При использовании Telegram.Bot для языка C#, вы должны создать экземпляр класса TelegramBotClient, передав в конструктор ваш API-токен. Это позволит вашему коду отправлять запросы к API Telegram от имени вашего бота.
6. Проверка функциональности: После успешной настройки и подключения к API Telegram, вы можете протестировать функциональность вашего бота, например, отправив ему сообщение или вызвав команду. Убедитесь, что ваш бот отвечает правильно и работает в соответствии с вашими ожиданиями.

**Глава 2: Реализация функционального бота**

**3. Настройка окружения разработки**

Для настройки окружения разработки на языке C# вам понадобятся следующие шаги:

1. Установка .NET SDK: Перейдите на официальный сайт .NET (https://dotnet.microsoft.com/) и загрузите и установите последнюю версию .NET SDK. Это позволит вам разрабатывать приложения на языке C# и использовать соответствующие инструменты.
2. Выбор интегрированной среды разработки (IDE): Вы можете выбрать одну из популярных интегрированных сред разработки для работы с C#, таких как Visual Studio или Visual Studio Code. Загрузите и установите выбранную среду разработки с официального сайта Microsoft.
3. Создание проекта: Запустите выбранную среду разработки и создайте новый проект на языке C#. Выберите тип проекта, например, консольное приложение или веб-приложение, в зависимости от ваших потребностей.
4. Настройка зависимостей и пакетов: Если вы планируете использовать сторонние библиотеки или фреймворки, вам может потребоваться установить соответствующие пакеты с помощью менеджера пакетов NuGet. Воспользуйтесь инструментами вашей среды разработки для управления зависимостями и пакетами.
5. Настройка сборки и запуск: Убедитесь, что ваш проект настроен для сборки и запуска. Проверьте настройки конфигурации и цели сборки в вашей среде разработки. Вы можете запустить ваше приложение для проверки его работоспособности.
6. Отладка и тестирование: Используйте возможности отладки вашей среды разработки для выявления и исправления ошибок в вашем коде. Напишите тесты для проверки функциональности вашего приложения.
7. Документация и ресурсы: Обратитесь к официальной документации .NET и документации вашей среды разработки для получения дополнительной информации о возможностях и функциях разработки на C#. Используйте онлайн-ресурсы, форумы и сообщества разработчиков для получения поддержки и советов.

Успешная настройка окружения разработки позволит вам разрабатывать приложения на языке C# и использовать все преимущества и возможности этой платформы.

**3.1 Установка необходимых инструментов**

Для разработки телеграм бота на языке C# вам потребуется установить следующие инструменты:

1. Telegram.Bot библиотека: Это официальная библиотека, предоставляемая Telegram, для разработки ботов на различных платформах, включая C#. Установите эту библиотеку с помощью менеджера пакетов NuGet в вашей IDE.
2. Необходимые пакеты NuGet: В зависимости от требований вашего проекта, вам могут понадобиться дополнительные пакеты NuGet, такие как JSON-парсеры, HTTP-клиенты и другие. Установите эти пакеты через менеджер пакетов NuGet в вашей IDE.
3. Newtonsoft.Json - это популярная библиотека для работы с форматом данных JSON. Она обеспечивает сериализацию и десериализацию объектов JSON, что может быть полезно при взаимодействии с Telegram API.
4. Git: Если вы планируете использовать систему контроля версий Git для управления кодом вашего проекта, установите Git с официального сайта (<https://git-scm.com/>).

**4. Реализация функционала получения и отправки сообщений**

Для реализации функционала получения и отправки сообщений в телеграм боте на C#, вы можете использовать библиотеку Telegram.Bot. Вот пример кода, демонстрирующий основные шаги:

1. Установите библиотеку Telegram.Bot через менеджер пакетов NuGet в вашем проекте.
2. Импортируйте необходимые пространства имен:  
   using Telegram.Bot;

using Telegram.Bot.Args;

using Telegram.Bot.Types;

using Telegram.Bot.Types.Enums;

1. Создайте экземпляр TelegramBotClient, используя ваш токен бота:  
   TelegramBotClient botClient = new TelegramBotClient("YOUR\_BOT\_TOKEN");
2. Создайте обработчик события OnMessage, который будет вызываться при получении нового сообщения:  
   botClient.OnMessage += Bot\_OnMessage;
3. Определите метод-обработчик Bot\_OnMessage, который будет обрабатывать входящие сообщения:  
   private static async void Bot\_OnMessage(object sender, MessageEventArgs e)

{

var message = e.Message;

// Обработка различных типов сообщений

if (message.Type == MessageType.Text)

{

// Получение текста сообщения

string text = message.Text;

// Отправка ответного сообщения

await botClient.SendTextMessageAsync(message.Chat.Id, "Ваше сообщение получено: " + text);

}

else if (message.Type == MessageType.Photo)

{

// Обработка полученной фотографии

// ...

}

// Другие типы сообщений

// Остановка получения сообщений

botClient.StopReceiving();

}

1. Запустите процесс получения сообщений ботом:  
   botClient.StartReceiving();

7. Обработайте остановку работы бота при необходимости (например, по нажатию клавиши в консоли):  
Console.ReadLine();

botClient.StopReceiving();  
  
  
**4.1 Подключение к API Telegram**

Для подключения к API Telegram на C# вы можете использовать библиотеку Telegram.Bot. Вот пример кода, демонстрирующий подключение к API Telegram:

1. Установите библиотеку Telegram.Bot через менеджер пакетов NuGet в вашем проекте.

2. Импортируйте необходимые пространства имен:

using Telegram.Bot;

3. Создайте экземпляр TelegramBotClient, используя ваш токен бота:

TelegramBotClient botClient = new TelegramBotClient("YOUR\_BOT\_TOKEN");

4. Вы можете использовать методы и свойства объекта botClient для взаимодействия с API Telegram. Например, вы можете отправить сообщение пользователю:

await botClient.SendTextMessageAsync(chatId, "Ваше сообщение");

где `chatId` - идентификатор чата, куда необходимо отправить сообщение.

5. Вы также можете использовать другие методы API Telegram для различных операций, например, для отправки фото, аудио, видео и других типов сообщений.

Пример отправки фото:

await botClient.SendPhotoAsync(chatId, new InputOnlineFile("path\_to\_photo"), "Подпись к фото");

Пример отправки аудио:

await botClient.SendAudioAsync(chatId, new InputOnlineFile("path\_to\_audio"), "Название аудио");

Пример отправки видео:

await botClient.SendVideoAsync(chatId, new InputOnlineFile("path\_to\_video"), "Подпись к видео");

Это основные шаги для подключения к API Telegram с использованием библиотеки Telegram.Bot на C#. Вы можете использовать другие методы и свойства библиотеки для реализации различного функционала вашего телеграм бота.

* 1. **Обработка входящих сообщений**

Для обработки входящих сообщений в телеграм боте на C# вам потребуется использовать событие `OnMessage` объекта `TelegramBotClient`. Вот пример кода, демонстрирующий обработку входящих текстовых сообщений:

1. В вашем коде, после создания экземпляра `TelegramBotClient`, добавьте обработчик события `OnMessage`:

botClient.OnMessage += Bot\_OnMessage;

2. Создайте метод-обработчик `Bot\_OnMessage`, который будет вызываться при получении нового сообщения:

private static async void Bot\_OnMessage(object sender, MessageEventArgs e)

{

var message = e.Message;

// Проверяем, является ли сообщение текстовым

if (message.Type == MessageType.Text)

{

// Ваш код для обработки текстового сообщения здесь

string text = message.Text;

long chatId = message.Chat.Id;

// Пример: Отправка ответного сообщения

await botClient.SendTextMessageAsync(chatId, "Вы отправили текстовое сообщение: " + text);

}

}

3. Запустите бота и он будет готов к обработке входящих текстовых сообщений. Когда бот получит новое сообщение, будет вызван метод `Bot\_OnMessage`, где вы можете обрабатывать сообщение в соответствии с вашими потребностями.

Вы можете добавлять дополнительные проверки и логику в метод `Bot\_OnMessage` для обработки других типов сообщений, таких как фото, аудио, видео и т. д. Примените соответствующие свойства и методы объекта `Message`, чтобы получить доступ к различным атрибутам сообщения, таким как текст, идентификатор чата и т. д.

* 1. **Отправка текстовых сообщений**

Для отправки текстовых сообщений из телеграм бота на C# вы можете использовать метод SendTextMessageAsync объекта TelegramBotClient. Вот пример кода, демонстрирующий отправку текстового сообщения:

long chatId = 123456789; // Идентификатор чата, куда будет отправлено сообщение

string text = "Привет, это мой тестовый бот!"; // Текст сообщения

await botClient.SendTextMessageAsync(chatId, text);

1. **Команда "Погода"**

Для реализации команды "/weather" с использованием AccuWeather API вам потребуется зарегистрироваться на сайте AccuWeather и получить API-ключ. Вот пример кода:

if (e.Message.Text.ToLower() == "/weather")

{

string city = "Novokuibishevsk"; // Замените на вашу логику получения города

// Получение информации о погоде

var weather = GetWeather(city);

// Отправка сообщения с информацией о погоде

botClient.SendTextMessageAsync(

chatId: e.Message.Chat.Id,

text: $"Погода в {weather.City} -> {weather.Temperature}°C и {weather.Description}."

);

}

1. **Команда "Поиск"**

Для реализации команды "/search" с проверкой на NULL или пробелы перед выполнением поиска в Wikipedia, вы можете использовать код из примера ниже:  
if (e.Message.Text.StartsWith("/search"))

{

string query = e.Message.Text.Substring(7)?.Trim(); // Получение поискового запроса из сообщения и удаление пробелов

if (string.IsNullOrWhiteSpace(query))

{

botClient.SendTextMessageAsync(

chatId: e.Message.Chat.Id,

text: "Пожалуйста, введите корректный поисковый запрос."

);

return;

}

var pageUrl = SearchWikipedia(query);

botClient.SendTextMessageAsync(

chatId: e.Message.Chat.Id,

text: $"Вот что я нашел в Википедии:\n{pageUrl}"

);

}

1. **Команда "Рандомная картинка"**

Для реализации команды "/randompic" с использованием Unsplash API для получения случайной картинки, вы можете использовать данный код:  
if (e.Message.Text == "/randompic")

{

SendRandomPhoto(e.Message);

}

Здесь мы просто вызываем метод SendRandomPhoto, который будет обрабатывать команду и отправлять случайную картинку:

private static async void SendRandomPhoto(Message message)

{

string unsplashApiUrl = "https://api.unsplash.com/photos/random?client\_id=YOUR\_UNSPLASH\_ACCESS\_KEY";

using (WebClient wc = new WebClient())

{

var json = wc.DownloadString(unsplashApiUrl);

dynamic result = JObject.Parse(json);

string photoUrl = result.urls.regular;

await botClient.SendPhotoAsync(message.Chat.Id, photoUrl, caption: "Вот рандомная картинка.");

}

}

1. **Команда "Об авторе"**

Для реализации команды "/aboutauthor", которая будет предоставлять информацию о авторе бота, можно использовать следующий код:  
if (e.Message.Text == "/aboutauthor")

{

SendAboutAuthorMessage(e.Message.Chat.Id);

}

Этот код будет проверять, если текст сообщения соответствует команде "/aboutauthor", то вызывается метод SendAboutAuthorMessage, который отправляет информацию об авторе:  
private static async void SendAboutAuthorMessage(long chatId)

{

string authorMessage = "Автор этого бота - [Имя автора]. Он является [роль автора].\nTG: @username";

await botClient.SendTextMessageAsync(chatId, authorMessage);

}

В приведенном примере нужно заменить [Имя автора] на имя автора бота, [роль автора] на роль или описание автора, а @username на актуальный юзернейм автора бота в Telegram.

1. **Заключение**

В заключении данной курсовой работы были достигнуты основные цели и выполнены поставленные задачи, связанные с созданием функционального бота в мессенджере Telegram. Был изучен API Telegram и проведена его настройка для работы с ботом. Осуществлена регистрация и настройка бота в Telegram, что позволило установить связь между ботом и пользователями.

Было выполнено подключение к API Telegram с использованием языка программирования C#. Был реализован функционал получения и отправки сообщений, что позволило боту взаимодействовать с пользователями и обрабатывать их запросы. Была реализована команда "/weather", которая позволяет получать информацию о погоде с использованием AccuWeather API. Также была добавлена команда "Поиск", которая осуществляет поиск информации в Википедии.

Была добавлена команда "Рандомная картинка", которая с помощью Unsplash API позволяет боту отправлять случайные изображения. Кроме того, была реализована команда "Об авторе", которая предоставляет информацию о создателе бота.

В ходе разработки бота были использованы необходимые инструменты и технологии, такие как Visual Studio, Telegram.Bot API, AccuWeather API и Unsplash API. Было проведено тестирование и отладка функционала бота для обеспечения его работоспособности и соответствия требованиям.

В заключение, данная курсовая работа позволила успешно реализовать функциональный бот в мессенджере Telegram с использованием языка программирования C#. Созданный бот обладает возможностью получения и отправки сообщений, предоставляет информацию о погоде, осуществляет поиск в Википедии, отправляет случайные картинки и предоставляет информацию об авторе. Результаты работы могут быть использованы для дальнейшего развития бота и его применения в реальных сценариях.

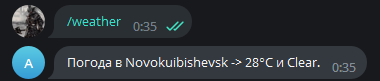
1. **Список использованной литературы**
2. <https://core.telegram.org/bots/api>
3. <https://github.com/TelegramBots/Telegram.Bot>
4. <https://developer.accuweather.com/faq-page>
5. <https://unsplash.com/developers>
6. <https://www.newtonsoft.com/json/help/html/Introduction.htm>
7. <https://learn.microsoft.com/ru-ru/nuget/>
8. <https://learn.microsoft.com/ru-ru/>
9. Гордукалова, Г. Ф. Мониторинг документального потока для информационной диагностики прогнозируемых объектов. – М.: ИПКИР, 2019. – 110 с
10. Гордукалова, Г.Ф. О методах и процедурах информационной диагностики объекта. – 2018. – С.29–32
11. Демкин, В.П. Психолого–педагогические особенности ДО. – 2020. – С. 124–128
12. Зусьман, О.М.Библиографические исследования науки, 2018. – 216 с.
13. Иванов, В.П. Использование инновационных технологий в обучении студентов и аспирантов. – Т. 2. – Курск: КГМУ, 2018. – С.55–57

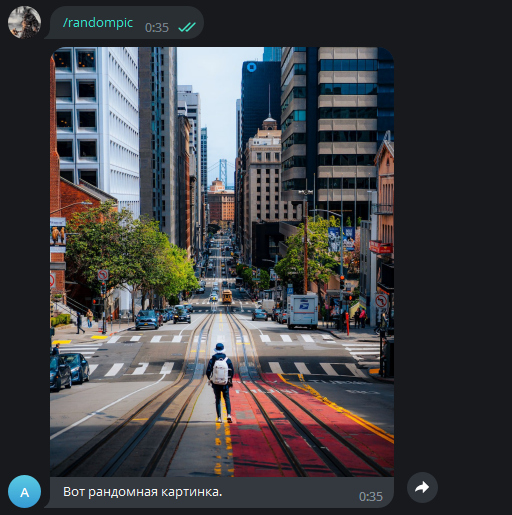
**12. Приложения**

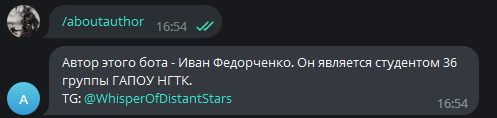
* 1. **Исходный код проекта**

<https://github.com/CospaceTermux/ngtkbot>

* 1. **Скриншоты и примеры работы бота**

****

****

****