Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра математического анализа и моделирования
Направление подготовки – 01.03.02 Прикладная математика и информатика
Направленность (профиль) образовательной программы «Прикладная математика и информатика»

ДОПУСТИТЬ К ЗАЩИТЕ
И.о. зав. кафедрой
_____ Н.Н. Максимова
«_____»_____ 2023 г.

БАКАЛАВРСКАЯ РАБОТА

на тему: Создание телеграм-бота для информационного канала

Исполнитель
студент группы 952-об _____ Т.А. Пискаль
(подпись, дата)

Руководитель
доцент, канд. физ.-мат. наук _____ В.В. Сельвинский
(подпись, дата)

Нормоконтроль
ст. преподаватель _____ А.Н. Дудин
(подпись, дата)

Благовещенск 2023

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра математического анализа и моделирования

УТВЕРЖДАЮ
И.о. зав. кафедрой
_____ Н.Н. Максимова
«_____»_____2023 г.

ЗАДАНИЕ

К бакалаврской работе студента: Пискаль Тимофея Александровича

1. Тема бакалаврской работы: Создание телеграм-бота для информационного канала (утверждена приказом от 20.04.2022 г. № 591-уч)

2. Срок сдачи студентом законченной работы: 19.06.2023 г.

3. Исходные данные к бакалаврской работе: отчет по преддипломной практике, научные статьи, учебно-методические работы

4. Содержание бакалаврской работы (перечень подлежащих разработке вопросов): создание телеграм-бота на языке программирования Python.

5. Перечень материалов приложения: листинг программы

6. Консультанты по бакалаврской работе: нормоконтроль – Дудин А.М., ст. преподаватель.

7. Дата выдачи задания: 20.05.2023 г.

Руководитель бакалаврской работы: Сельвинский Владимир Владимирович, доцент, канд. физ.-мат. наук, доцент

Задание принял к исполнению (20.05.2023): _____ Пискаль Т.А.

РЕФЕРАТ

Бакалаврская работа содержит 58 с., 25 рисунков, 18 источников, 1 приложение.

ТЕЛЕГРАМ-БОТ, API, PYTHON, ФРЕЙМФОРК, HTML, PYCHARM, ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ, АВТОМАТИЗАЦИЯ, ЧАТ-БОТ, BEAUTIFUL SOUP, FLASK

В бакалаврской работе рассмотрены различные методы для создания телеграм-бота на языке программирования Python для информационного канала. Были изучены особенности использования Python-telegram-bot, Flask, Beautiful Soup и PyCharm для реализации функциональности бота. Также был проведен анализ существующих решений и выбраны наиболее подходящие методы.

Результатом работы является функционирующий телеграм-бот, готовый к номинальной эксплуатации.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время социальные сети и мессенджеры являются одними из самых популярных каналов коммуникации в интернете. Их активное использование позволяет пользователю получать информацию в режиме реального времени, а также поддерживать связь с близкими и друзьями. В связи с этим разработка телеграм-ботов становится все более востребованной задачей, поскольку такие боты позволяют автоматизировать многие процессы, связанные с информационными потоками и общением в мессенджерах.

Телеграм-боты являются удобным и быстрым способом получения информации, поскольку пользователи могут подписаться на канал и получать актуальные новости и другую информацию без необходимости перехода на другие веб-сайты. Бот может быть настроен на сбор информации с различных источников, таких как новостные сайты, блоги и социальные сети, и автоматически публиковать эту информацию в телеграм-канале. Это удобно для тех, кто хочет получать актуальную информацию из разных источников в одном месте, а также для тех, кто управляет информационными каналами и хочет оптимизировать свою работу [4].

Основная идея данного проекта заключается в том, чтобы подготовиться к созданию бота, который сможет автоматически собирать и публиковать информацию в телеграм-канале. Для этого необходимо изучить скрипты на языке программирования Python для получения информации с внешних источников, а также скрипты для автоматической публикации этой информации в телеграм-канале.

Одним из ключевых преимуществ создания телеграм-бота является автоматизация процесса сбора и публикации информации в телеграм-канале. Это позволит сократить время и усилия, затрачиваемые на ручной сбор и публикацию информации, а также увеличит скорость обновления и актуальности информации в канале. Кроме того, разработка телеграм-бота поможет

повысить качество и надежность публикуемой информации, поскольку все процессы будут автоматизированы и отслеживаемы.

Целью работы является создание телеграм-бота для информационного канала на языке Python.

Для достижения поставленной цели необходимо решить следующие задачи:

– изучить вопросы написания телеграм-бота, определить необходимые для этого инструменты;

– изучить особенности подключения программы к сервисам Telegram с помощью API, подобрать подходящие сервисы и инструменты;

– произвести тестирование и отладку работы телеграм-бота до его дальнейшего введения в номинальную эксплуатацию.

Работа состоит из введения, четырех глав, заключения, библиографического списка и приложения. Первая глава посвящена основам разработки телеграм-ботов, обзоре использования ботов в современном мире. Во второй главе описаны методы, используемые в реализации телеграм-ботов. В третьей главе отражена основная часть работы: процесс создания и написания телеграм-бота на языке Python с использованием выбранных инструментов. В четвертой главе описано тестирование функциональности и оптимизации телеграм-бота, а также отладка для более корректной и конкурентноспособной работы.

1 РАЗРАБОТКА ТЕЛЕГРАМ-БОТОВ В СОВРЕМЕННЫХ УСЛОВИЯХ

1.1 Анализ создания и эксплуатации телеграм-ботов

Боты являются незаменимыми инструментами в информационном обществе. Они предоставляют широкий спектр функций, которые могут быть использованы для улучшения эффективности работы и взаимодействия с аудиторией. Боты могут выполнять различные задачи, включая автоматизацию рутинных процессов, обработку данных, управление устройствами и коммуникацию с пользователями.

Существует несколько типов ботов, которые могут быть использованы в различных ситуациях. Телеграм-боты, например, являются одним из наиболее распространенных типов ботов, которые интегрированы в мессенджер Telegram. Они используются для общения с пользователями, предоставления информации и выполнения задач. Чат-боты, с другой стороны, часто используются для общения с пользователями на веб-сайтах или в мессенджерах. Роботы-ассистенты могут быть использованы для автоматизации рутинных процессов и управления устройствами [2].

Разработка ботов является важной темой в настоящее время, поскольку они предоставляют новые возможности для улучшения эффективности работы и взаимодействия с аудиторией. Боты могут помочь ускорить выполнение задач, уменьшить нагрузку на персонал, сократить время ответа на запросы пользователей и улучшить опыт пользователей в целом.

Telegram является одним из наиболее популярных мессенджеров в мире. Он был создан в 2013 году. С тех пор Telegram стал одним из основных инструментов коммуникации для миллионов пользователей по всему миру. Telegram имеет ряд особенностей, которые отличают его от других мессенджеров. Одна из ключевых особенностей Telegram – это возможность отправки сообщений, которые могут быть зашифрованы и защищены от не-

санкционированного доступа. Это делает Telegram одним из наиболее безопасных мессенджеров на рынке [13].

Telegram также имеет удобный и простой в использовании интерфейс, который позволяет пользователям быстро находить и обмениваться сообщениями с другими пользователями. Он также поддерживает функции группового чата, возможность отправки файлов и мультимедиа, а также многие другие функции.

Telegram также предлагает множество возможностей для разработчиков, включая API для создания приложений и ботов. Это позволяет разработчикам создавать приложения и ботов, которые могут интегрироваться с Telegram и предоставлять дополнительные функции для пользователей. Одной из ключевых особенностей Telegram является поддержка телеграм-ботов. Телеграм-боты являются автоматизированными программами, которые могут выполнять различные задачи в мессенджере. Они могут быть использованы для общения с пользователями, предоставления информации и выполнения задач.

В современном информационном обществе создание телеграм-ботов на языке программирования Python для информационных каналов является важной и актуальной темой. Телеграм-боты автоматизируют процессы общения и обмена информацией между пользователями и каналом.

Одним из основных преимуществ использования телеграм-ботов является возможность быстрого и эффективного общения с аудиторией канала. Боты позволяют автоматически отвечать на вопросы пользователей, предоставлять информацию и выполнять другие задачи, тем самым улучшая качество коммуникации и снижая нагрузку на администраторов канала. Это особенно важно в случае крупных каналов, где количество вопросов и запросов пользователей может быть значительным [3].

Кроме того, телеграм-боты имеют большой потенциал для достижения большой аудитории и распространения информации. Телеграм является одним из самых популярных мессенджеров в мире, который имеет более 500

миллионов пользователей. Боты могут использоваться для распространения информации о канале и его содержимом, а также для рекламы и монетизации контента.

1.2 Элементы качественного телеграм-бота

Выбор языка программирования является важным этапом при разработке телеграм-бота. Существует множество языков программирования, которые можно использовать для создания ботов, но в данном случае был выбран язык Python. Это связано с рядом преимуществ, которые Python обладает перед другими языками. Первым и наиболее важным преимуществом Python является его простота и удобство использования. Python имеет простой и интуитивно понятный синтаксис, что делает его легким для изучения и использования, даже для начинающих программистов. Это также делает Python очень гибким языком, который может быть использован для разработки различных типов приложений, в том числе и телеграм-ботов.

Вторым преимуществом Python является наличие богатой и развитой библиотеки для различных задач. Python имеет огромное сообщество разработчиков, которые создают и поддерживают библиотеки, предназначенные для решения различных задач. Это позволяет разработчикам быстро и легко создавать телеграм-ботов.

Третьим преимуществом Python является его мультиплатформенность. Python может быть запущен на большинстве операционных систем, включая Windows, Mac OS и Linux. Это делает его идеальным языком для создания кросс-платформенных телеграм-ботов, которые могут работать на различных устройствах и операционных системах.

Наконец, Python также имеет широкую поддержку сообщества. Python имеет огромное и активное сообщество разработчиков и пользователей, которые всегда готовы помочь новичкам и ответить на вопросы. Это делает Python идеальным выбором для разработки телеграм-ботов, поскольку существует множество ресурсов, где можно найти помощь и поддержку при создании бота на Python [1].

Важность и актуальность создания телеграм-бота на языке программирования Python для информационного канала подтверждается также тем фактом, что в настоящее время существует большое количество информационных каналов в телеграме, которые предоставляют пользователю различную информацию - от новостей и статей до различных рекламных материалов. В связи с этим, владельцы информационных каналов стараются разнообразить свой контент и сделать его более доступным и удобным для пользователей. Создание телеграм-бота на языке программирования Python является одним из наиболее эффективных и востребованных способов повышения качества обслуживания аудитории [11].

1.3 Методы создания и тестирования телеграм-бота

Для запуска и тестирования бота в Telegram следует выполнить ряд действий.

Первым шагом необходимо запустить ранее созданный скрипт на выполнение, который содержит все необходимые функции-обработчики для работы бота. Для этого следует открыть редактор кода и запустить файл скрипта.

После запуска скрипта следует найти своего бота в Telegram и отправить ему команду «/start». Это позволит проверить корректность работы приветственного сообщения, которое должно быть выведено в ответ на данную команду. Далее можно отправлять другие команды для проверки их работы, например, команду «/help», которая должна вывести инструкцию по использованию бота. Также следует проверить корректность вывода последних новостей информационного канала, для этого нужно отправить команду «/news».

Для проверки работы функций подписки и отписки пользователей необходимо отправить соответствующие команды, «/subscribe» и «/unsubscribe» соответственно, и проверить, что они работают корректно и пользователь добавляется или удаляется из списка подписчиков. При тестировании необходимо также проверить, что бот правильно обрабатывает со-

общения, которые не являются командами. Для этого можно отправить любое сообщение, которое не соответствует ни одной из заданных команд, и проверить, что бот ответит пользователю корректно.

В целом, тестирование бота в Telegram позволяет убедиться в корректности его работы и функциональности, а также выявить и исправить возможные ошибки и недочеты в скрипте [5].

2 ОБЗОР И ОПИСАНИЕ МЕТОДОВ И ИНСТРУМЕНТОВ И ДЛЯ СОЗДАНИЯ ТЕЛЕГРАМ-БОТА НА ЯЗЫКЕ PYTHON

2.1 Обзор популярных методов и инструментов, используемых для разработки ботов

На сегодняшний день, существует множество инструментов для создания телеграм-ботов, таких как BotFather, BotPress, Botium, BotMan, Telegraf, Botkit и другие. Каждый из этих инструментов имеет свои преимущества и недостатки, а также специфические особенности, которые могут быть полезными в зависимости от конкретных потребностей проекта.

BotFather является стандартным инструментом от Telegram, позволяющим создавать и управлять ботами в Telegram. BotFather предоставляет множество возможностей для настройки и кастомизации бота, включая настройки имени, аватара, команд, режимов, клавиатур и т.д. Однако, BotFather имеет некоторые ограничения, например, невозможность написания сложной бизнес-логики.

BotPress является открытым фреймворком для создания чат-ботов, который использует Node.js и предоставляет набор инструментов для разработки, обучения и тестирования ботов. BotPress имеет ряд преимуществ, таких как возможность разработки сложной бизнес-логики, использование веб-компонентов, настраиваемые варианты развертывания, обширный API и т.д. Однако, BotPress требует значительных знаний в программировании и может быть более сложным в использовании, чем другие инструменты [7].

Botium является открытым фреймворком для автоматизации тестирования чат-ботов, который позволяет создавать тестовые сценарии и проверять ботов на соответствие ожидаемому поведению. Botium имеет ряд преимуществ, таких как возможность создания комплексных тестовых сценариев, использование различных языков программирования, включая JavaScript, Python, Java, TypeScript, а также интеграцию с другими инструментами для тестирования.

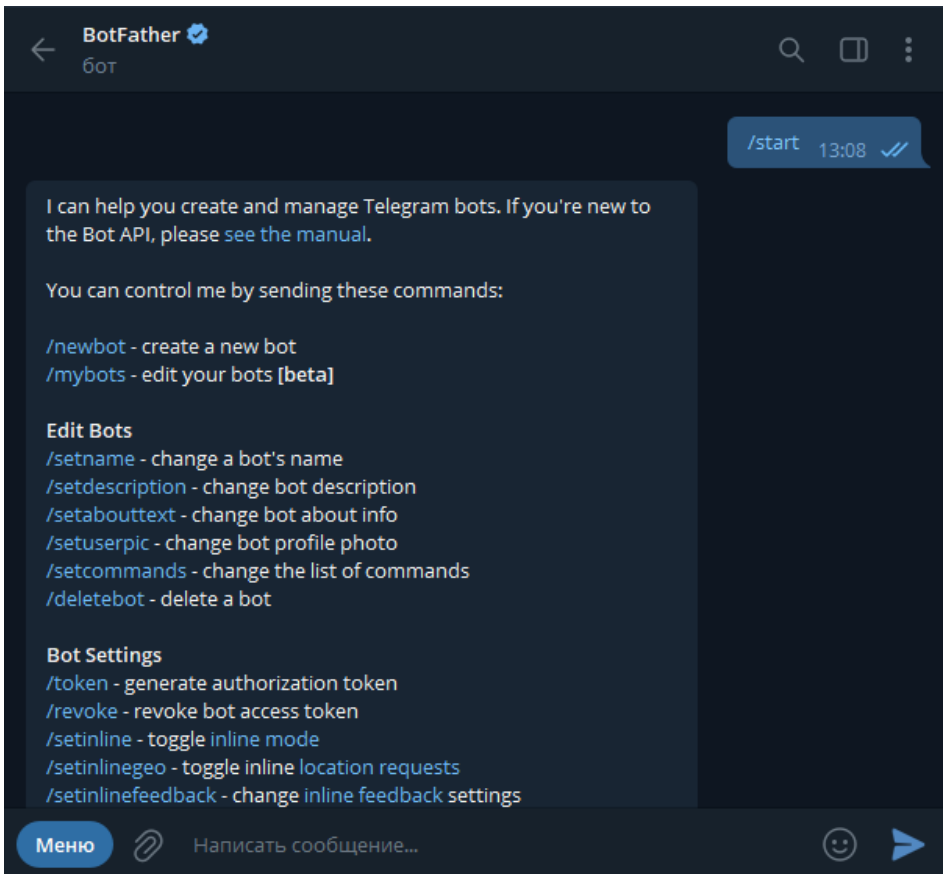

Рисунок 1 – Приветственное сообщение BotFather

Есть достаточно много готовых инструментов для создания телеграм-ботов на языке программирования Python. Однако, большинство из них предназначены для создания ботов с ограниченным функционалом.

Существует ряд более продвинутых инструментов, позволяющих создавать ботов с более широким функционалом, таких как интеграция с базами данных, работа с API различных сервисов, реализация машинного обучения и др. Некоторые из таких инструментов, например, python-telegram-bot, Telepot, aiogram, весьма популярны и широко используются сообществом разработчиков [9].

При этом, несмотря на наличие готовых решений, создание телеграм-бота с учетом специфики информационного телеграм-канала может требовать от разработчика индивидуального подхода и разработки необходимых функций и модулей. Например, важным моментом может быть организация планирования и автоматизации публикации постов, реализация возможности

аналитики посещаемости канала, работа с дополнительными сервисами и многие другие задачи.

Рисунок 2 – Пример созданной клавиатуры посредством инструмента python-telegram-bot

Также, при выборе инструментов и подходов к созданию телеграм-бота необходимо учитывать возможные ограничения и ограничивающие факторы, такие как ограничение на количество запросов в секунду, необходимость использования прокси-серверов, настройка системы безопасности и другие.

Для создания телеграм-бота на языке программирования Python с функционалом, отвечающим специфике информационного телеграм-канала, можно использовать как готовые инструменты, так и разрабатывать собственные модули и функции. При этом, необходимо учитывать возможные ограничения и требования к безопасности, а также наличие поддержки со стороны разработчиков и сообщества.

Другим важным аспектом создания телеграм-бота является выбор методов связи кода программы бота на сервере и непосредственно ботом в мессенджере Телеграм. Эта задача может быть выполнена с использованием различных программных средств и библиотек. Однако, независимо от выбранных инструментов, существует несколько основных методов, которые могут быть использованы для написания кода Python для телеграм-бота.

Первый метод – это использование библиотеки python-telegram-bot для создания телеграм-бота. Эта библиотека предоставляет удобный интерфейс для создания и настройки телеграм-ботов, а также обработку различных типов сообщений и событий. Создание телеграм-бота с использованием этой библиотеки обычно начинается с создания экземпляра класса telegram [12].

Второй метод – это использование библиотеки BeautifulSoup для парсинга HTML-страниц и получения информации из них. Эта библиотека может быть полезна для телеграм-ботов, которые получают информацию из веб-страниц и возвращают ее пользователю. Для использования этой библиотеки обычно требуется сначала загрузить веб-страницу с помощью библиотеки requests, а затем произвести парсинг HTML-кода страницы с помощью BeautifulSoup.

Также можно использовать библиотеку schedule для того, чтобы можно было формировать и отправлять пользователю, группе или в канал сообщения, согласно указанному с помощью библиотеки времени [10].

2.2 Перечень программных средств для решения исследуемой задачи

Перечень основных программных средств, используемых в данной работе, включает Python-telegram-bot, BeautifulSoup, Schedule и PyCharm. Python-telegram-bot является пакетом Python, предоставляющим удобный интерфейс для создания телеграм-ботов и работы с API Telegram. BeautifulSoup является библиотекой Python для парсинга HTML и XML документов, используемой для извлечения информации из веб-страниц. PyCharm – интегрированная среда разработки (IDE) на языке программирования Python, которая предоставляет широкий набор инструментов для разработки, отладки и тестирования приложений на Python.

PyCharm является мощной интегрированной средой разработки (IDE) на языке программирования Python, которая предоставляет широкий набор инструментов для разработки, отладки и тестирования приложений на Python. Она обладает множеством функций, которые облегчают и ускоряют

процесс разработки, таких как автодополнение кода, функции рефакторинга, инструменты для отладки и профилирования, системы контроля версий и многое другое.

Одним из наиболее полезных инструментов PyCharm для разработки телеграм-ботов является поддержка виртуальных сред, которые позволяют создавать изолированные среды разработки с заданными версиями Python и зависимостями. Это значительно облегчает управление зависимостями и позволяет создавать надежные и совместимые приложения. Не менее важным инструментом PyCharm является возможность интеграции с пакетными менеджерами, такими как pip и conda, что позволяет быстро устанавливать и управлять пакетами и зависимостями. Также есть поддержка Docker и других технологий контейнеризации, что облегчает развертывание и управление приложениями.

PyCharm также обладает мощными функциями для отладки приложений, включая поддержку отладки многопоточных приложений и профилирование кода. Это позволяет быстро находить и исправлять ошибки и повышать производительность приложений. Еще одним полезным инструментом является поддержка систем контроля версий, таких как Git и SVN. PyCharm предоставляет интегрированный интерфейс для работы с репозиториями, позволяя быстро создавать и коммитить изменения, создавать ветки, сливать их и многое другое.

Среда разработки PyCharm предоставляет широкий набор функций для автоматизации процесса разработки, таких как интеграция с системами непрерывной интеграции и развертывания, автоматическое форматирование кода, быстрое создание и настройка новых проектов и многое другое.

Python-telegram-bot – это пакет Python, который обеспечивает удобный интерфейс для работы с API Telegram и позволяет создавать телеграм-ботов для общения с пользователями в Telegram. Он является одним из наиболее популярных пакетов для создания телеграм-ботов и имеет обширную документацию. Python-telegram-bot обладает множеством возможностей для со-

здания как простых, так и сложных телеграм-ботов. Он поддерживает многопоточную обработку запросов, что обеспечивает высокую производительность и позволяет создавать ботов, которые могут обрабатывать одновременно несколько запросов. Кроме того, он предоставляет широкий набор встроенных функций для работы с сообщениями, стикерами, фото, видео и другими медиафайлами.

Пакет Python-telegram-bot также предоставляет возможность создавать кастомные клавиатуры и меню для бота, что позволяет создавать более удобный и интуитивно понятный интерфейс для пользователя. Он также поддерживает возможность работы с базами данных, что позволяет создавать более сложные боты, например, для управления задачами или для обработки заказов в интернет-магазине.

Библиотека schedule является мощным инструментом для планирования и выполнения задач по расписанию в среде программирования Python. Одним из ключевых преимуществ schedule является возможность точной настройки времени выполнения задач. Можно определить задачи, которые должны выполняться каждый день в определенное время, например, для формирования автоматического отчета по результатам работы. Библиотека также позволяет выполнять задачи с заданной периодичностью, что обеспечивает гибкость и автоматизацию процесса создания отчетов.

В целом, библиотека schedule предоставляет надежный и эффективный способ для автоматизации задач по расписанию в научных исследованиях и проектах. Ее гибкость, точность и удобный интерфейс делают ее ценным инструментом для создания отчетов по работе и повышения производительности в научной области.

BeautifulSoup – это библиотека Python, предназначенная для парсинга HTML и XML документов. Она является одним из наиболее популярных инструментов для извлечения данных из веб-страниц и использования их в других приложениях. BeautifulSoup предоставляет удобный и простой в исполь-

зовании интерфейс для поиска, извлечения и манипулирования данными в HTML-коде.

Основная функциональность BeautifulSoup включает в себя создание объекта парсера, поиск и фильтрацию элементов HTML-документа и извлечение информации из этих элементов. Для создания объекта парсера можно использовать различные типы парсеров, включая стандартные парсеры Python, такие как lxml и html5lib, а также встроенный парсер BeautifulSoup.

Поиск и фильтрация элементов HTML-документа в BeautifulSoup осуществляется с помощью методов find() и find_all(), которые могут принимать различные аргументы для определения элементов на странице. Например, можно искать элементы по имени тега, классу CSS или идентификатору, а также использовать регулярные выражения для более точного поиска. Это позволяет разработчикам быстро и легко находить нужные элементы на странице и извлекать из них данные для использования в других приложениях.

Дополнительно, BeautifulSoup позволяет обрабатывать различные ошибки, которые могут возникать при парсинге HTML-документов. Например, если в документе присутствует невалидный HTML-код, то BeautifulSoup попытается автоматически исправить его, чтобы упростить процесс парсинга.

Благодаря простому интерфейсу и богатым возможностям, BeautifulSoup является популярным инструментом для многих задач, связанных с обработкой веб-страниц. Он может использоваться для автоматизации сбора данных с веб-сайтов, анализа контента, извлечения и сбора информации из нескольких источников, в том числе и для создания более сложных веб-приложений. Кроме того, BeautifulSoup является открытым программным обеспечением и имеет широкое сообщество разработчиков, которые поддерживают его и развивают новые функции и возможности [6].

2.3 Возможности выбранных программных средств для решения задачи написания телеграм-бота на языке программирования Python для информационного телеграм-канала

PyCharm является одной из наиболее популярных интегрированных сред разработки для языка Python. Она обладает множеством инструментов для эффективной и удобной разработки программного обеспечения на языке Python. Среди этих инструментов можно выделить систему автодополнения кода, отладчик, систему управления версиями и многое другое, что облегчает процесс разработки.

Один из наиболее полезных инструментов PyCharm для создания телеграм-ботов – это его поддержка для языка Python и библиотек Python-telegram-bot и Flask. PyCharm может автоматически устанавливать и настраивать эти библиотеки, а также предоставлять подсказки и советы по их использованию.

Python-telegram-bot является библиотекой Python, которая предоставляет удобный интерфейс для создания и настройки телеграм-ботов. Эта библиотека позволяет легко настраивать обработку различных типов сообщений и событий, что позволяет создавать ботов с различной функциональностью. Для создания телеграм-бота информационного канала, одной из важных функций является обработка сообщений и команд от пользователей. Python-telegram-bot предоставляет удобный интерфейс для обработки таких сообщений и команд, а также для отправки ответов и результатов пользователю. Кроме того, библиотека может быть использована для настройки оповещений и уведомлений в информационном канале.

Flask является микро фреймворком для создания веб-приложений на языке Python. Flask позволяет легко создавать веб-страницы и API, что может быть полезно для телеграм-ботов, которые обрабатывают сообщения и возвращают результаты пользователю.

Получение информации из веб-страниц является необходимым функционалом для многих телеграмм-ботов, которые предоставляют пользователю какие-то данные или информацию из внешних источников. Для реализации этой функциональности может быть использована библиотека Beautiful-

Soup, которая предоставляет удобный интерфейс для парсинга HTML-страниц.

В качестве примера, рассмотрим задачу создания телеграмм-бота для информационного канала о погоде. Для получения данных о погоде можно использовать различные сервисы, например, сайт «OpenWeatherMap». Для получения данных о погоде с этого сайта можно использовать его API, однако для примера мы будем использовать парсинг HTML-страницы с данными о погоде.

Для этого можно написать функцию, которая будет получать данные о погоде из веб-страницы и возвращать их в удобном формате. Например, функция будет получать данные о погоде из сайта «OpenWeatherMap» и возвращать их в виде строки показана рисунке.

main.py

```python
from bs4 import BeautifulSoup
import requests

def get_weather():
    url = "https://www.weather-forecast.com/locations/Blagoveshchensk/forecasts/latest"
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    temperature = soup.find(class_='b-forecast__table-value b-forecast__table-value_type_temperature').get_text()
    description = soup.find(class_='b-forecast__table-description-content').get_text()
    return f"The weather in Blagoveshchensk is {temperature} and {description}."
```

Рисунок 3 – Функция получения данных с помощью библиотеки Beautiful-Soup

Эта функция использует библиотеку requests для получения содержимого веб-страницы и библиотеку BeautifulSoup для парсинга HTML-страницы и извлечения необходимых данных. В данном случае, функция извлекает температуру и описание погоды из HTML-страницы сайта «OpenWeatherMap» для города Благовещенск и возвращает строку с этими данными.

Для отправки сообщений пользователю телеграмм-ботом можно использовать библиотеку python-telegram-bot. Эта библиотека предоставляет

удобный интерфейс для создания и настройки телеграмм-ботов, включая возможность отправки различных типов сообщений, таких как текстовые сообщения, изображения, аудио и др.

Для реализации функционала отправки сообщений пользователю можно написать функцию, которая будет принимать на вход текст сообщения и ID чата пользователя, и отправлять это сообщение пользователю. Например, функция может отправлять текстовое сообщение пользователю с заданным ID чата.

3 РАЗРАБОТКА ТЕЛЕГРАМ-БОТА ДЛЯ ИНФОРМАЦИОННОГО КАНАЛА

3.1 Анализ требований и проектирование

Разрабатываемый телеграм-бот должен быть инструментом, способным служить помощником для студентов и преподавателей Факультета математики и информатики. Он должен обладать функционалом, который обеспечит пользователю доступ к важной и актуальной информации о факультете, а также предоставит дополнительные возможности для удобного взаимодействия с ботом [8].

Основными требованиями к функциональности телеграм-бота являются:

– предоставление информации о факультете: телеграм-бот должен быть способен предоставить пользователю информацию о структуре факультета, его кафедрах. Это может включать расписание занятий, актуальные новости и события, а также материалы для самостоятельного изучения;

– взаимодействие с ботом: телеграм-бот должен обеспечивать возможность взаимодействия с пользователем через удобный интерфейс кнопок и запросов. Пользователь должен иметь возможность запрашивать определенную информацию;

– интеграция с внешними источниками: телеграм-бот должен быть способен интегрироваться с различными внешними источниками данных, такими как веб-сайты факультета, системы расписания, репозитории учебных материалов и другие. Это позволит обеспечить актуальность и достоверность предоставляемой информации, а также расширить функциональные возможности бота. Однако на момент написания телеграм-бота, технической возможности интегрировать веб-сайт Амурского государственного университета нет, так как сайт не предоставляет пользователю данные посредством API-ключа. Таким образом, всю информацию будет необходимо обновлять вручную, что совершенно нецелесообразно. Поэтому было принято решение реа-

лизовать выдачу ботом расписания в виде ссылки на раздел сайта университета с расписанием по группам.

Касаемо использования API, можно построить механизм автоматической публикации информации о погоде на официальный телеграм-канал ФМиИ | АмГУ. Это позволит подписчикам узнавать актуальную информацию о погоде за окном, и учитывать ее для построения личных планов. В свою очередь, этот аспект позволит увеличить пользу и востребованность бота и информационного канала ФМиИ для подписчиков.

Основываясь на вышеуказанных требованиях, можно составить диаграмму взаимодействия между пользователем и ботом (внутреннее взаимодействие).

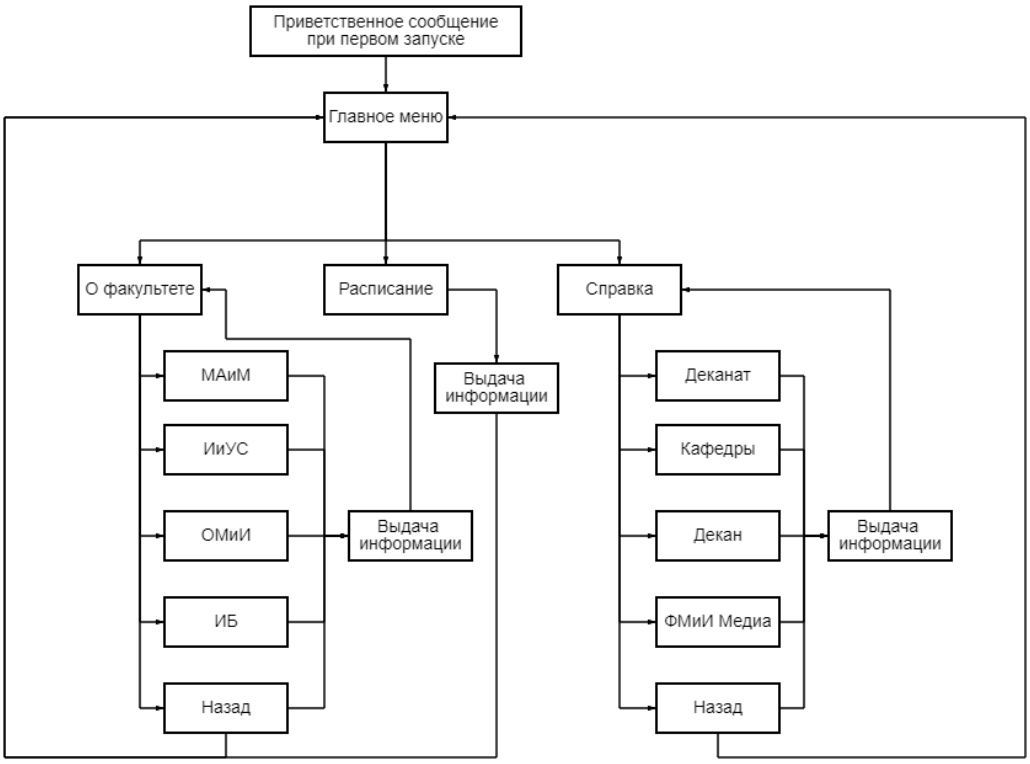

Рисунок 4 – Схема взаимодействия между пользователем и ботом

Взаимодействие должно происходить посредством кнопок, реализованных с помощью библиотеки python-telegram-bot.

Что касается внешних взаимодействий, то есть подключение к самому мессенджеру Telegram, к различным сервисам, то тут должны использоваться

ключи API. С помощью них программе бота получается взаимодействовать с нужными ресурсами и производить деятельность.

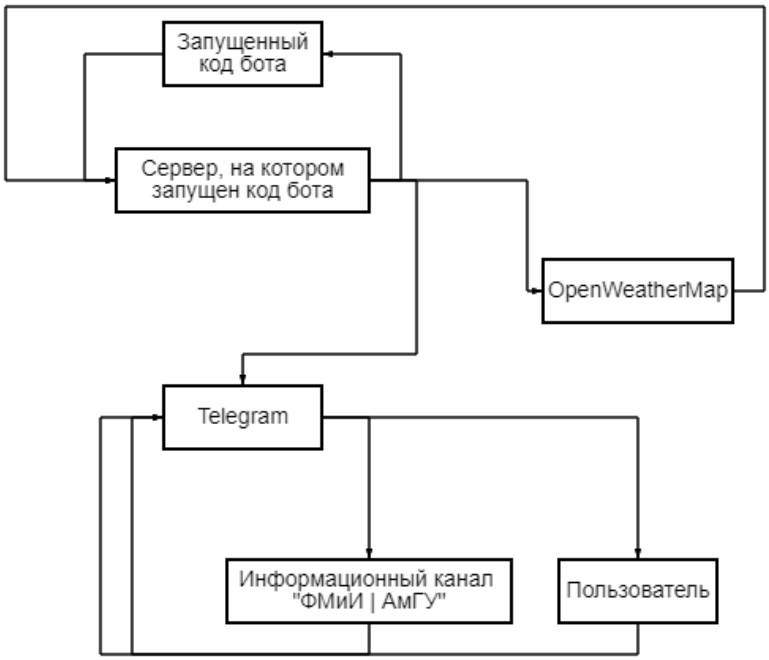

Рисунок 5 – Схема внешнего взаимодействия

Логика бота заключается в том, чтобы отвечать на команды, которые отправляет пользователь. Для удобного и эффективного взаимодействия используются кнопки.

Запуск и тестирование бота проводится с персонального компьютера, на котором разрабатывается продукт. Запуском бота можно считать начало выполнения кода, который записан в файле с расширением .py.

3.2 Реализация функциональности бота с помощью Python-telegram-bot

Для разработки телеграм-бота на языке программирования Python необходимо предварительно установить интерпретатор данного языка на свое устройство. Интерпретатор Python представляет собой программное обеспечение, которое позволяет выполнять программы, написанные на языке Python, путем интерпретации их исходного кода.

Установка интерпретатора Python может быть осуществлена путем скачивания соответствующего дистрибутива с официального веб-сайта Python (https://www.python.org). На сайте доступны дистрибутивы для различных операционных систем, включая Windows, macOS и Linux. Рекомендуется выбрать последнюю стабильную версию Python, совместимую с операционной системой, используемой на персональном компьютере. После загрузки дистрибутива Python необходимо запустить установочный файл и следовать инструкциям мастера установки. В процессе установки можно выбрать дополнительные компоненты, такие как pip (инструмент управления пакетами Python) и добавление Python в переменную окружения PATH. Эти компоненты облегчат управление пакетами и запуск программ на языке Python [16].

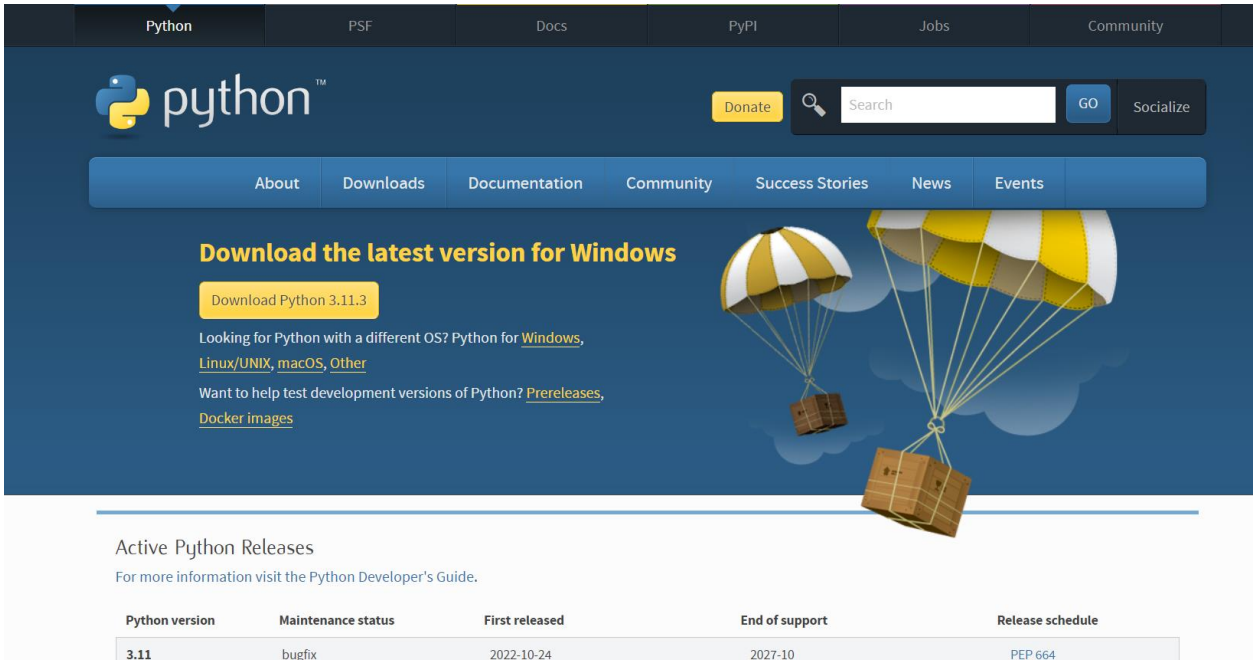

Рисунок 6 – Официальный сайт Python

После успешной установки интерпретатора Python можно проверить его работоспособность, выполнив простую команду в командной строке. Для этого необходимо открыть командную строку (терминал) и ввести команду «python.exe». Если все настроено правильно, должно отобразиться приглашение интерпретатора Python, готового к выполнению команд. Для проверки

работоспособности, можно написать код для вывода стандартной фразы «Привет, Мир!».

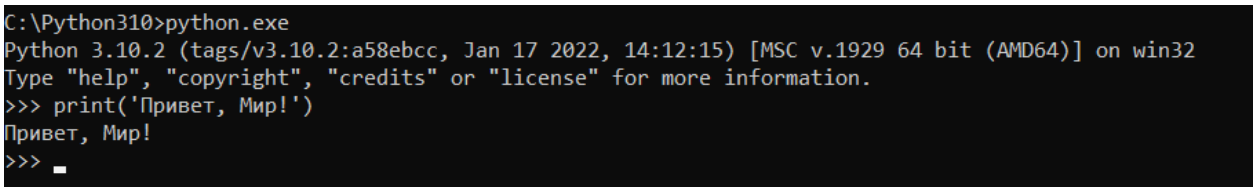

Рисунок 7 – Проверка работоспособности интерпретатора

Также для того, чтобы приступить к разработке бота, необходимо установить нужные библиотеки в папку «Scripts» корневой папки «Python310». Для этого, с помощью встроенной команды терминала «cd» (change directory) переходим в нужную папку и прописываем следующую команду:

pip install python-telegram-bot==13.7

Этой командой устанавливается пакет python-telegram-bot версии 13.7.

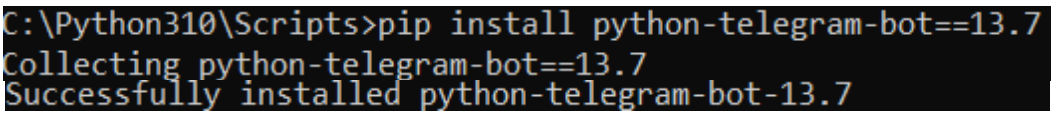

Рисунок 8 – Сообщение об успешной установке библиотеки

Таким же образом необходимо установить библиотеки BeautifulSoap и Schedule. Для этого в этой же папке необходимо написать следующие команды:

– pip install beautifulsoap4;

– pip install schedule.

Теперь, когда интерпретатор Python установлен и работает, установлены необходимые библиотеки, можно приступать к разработке телеграм-бота на языке Python. Это включает написание исходного кода бота, использование соответствующих библиотек и инструментов для работы с Telegram API, а также тестирование и развертывание бота на сервере [18].

В ходе разработки телеграм-бота нужно сохранять различные версии кода на разных этапах процесса разработки. Это позволит иметь возможность

вернуться к предыдущему варианту кода в случае необходимости. Сохранение версий кода обеспечивает следующие преимущества:

– возврат к предыдущему состоянию: если в процессе доработки кода возникли проблемы или ошибки, можно вернуться к предыдущей версии кода, которая работала корректно. Это позволяет сохранить стабильность функциональности бота и избежать потенциальных проблем;

– отслеживание изменений: сохранение различных версий кода позволяет отслеживать внесенные изменения на каждом этапе разработки. Это полезно для анализа и оценки влияния каждого изменения на работу бота. Также это помогает команде разработчиков сотрудничать и согласовывать свои действия при работе;

– эксперименты и новые функциональности: сохранение версий кода позволяет проводить эксперименты и вносить новые функциональности, не беспокоясь о потенциальных проблемах. Если новая функциональность не сработает или приведет к нежелательным результатам, всегда можно вернуться к предыдущей версии кода и продолжить работу;

– важно сохранять версии кода на регулярной основе, особенно перед внесением значительных изменений или экспериментами. Это помогает минимизировать риски и обеспечивает гибкость в работе над проектом.

Первым этапом создания телеграм-бота является создание самого бота через официального бота Telegram под названием «BotFather». «BotFather» представляет собой специального бота, который позволяет пользователям создавать и настраивать своих собственных телеграм-ботов. Для того чтобы создать бота, необходимо выполнить следующие шаги:

– открыть приложение Telegram на своем устройстве и воспользоваться функцией поиска, чтобы найти бота «BotFather». Необходимо ввести запрос «@botfather» в поле поиска и нажать на найденный результат;

– запустить бота «BotFather», нажав на его имя в результате поиска;

– начать взаимодействие с ботом «BotFather», отправив ему сообщение.

Далее «BotFather» предоставит список доступных команд и опций. Один из важных шагов – создание нового бота. Для этого необходимо ввести команду «/newbot» и следовать инструкциям «BotFather».

«BotFather» запросит название бота, поэтому нужно придумать уникальное название для телеграм-бота и ввести его. В нашем случае, бот называется «bot_for_info_channel».

Затем «BotFather» запросит уникальное имя пользователя (username) для бота. Имя пользователя должно оканчиваться на «_bot». Было придумано имя «tgchanneltest_bot».

После ввода имени пользователя «BotFather» выдаст токен доступа (API token). Токен является уникальным идентификатором бота и позволяет взаимодействовать с Telegram API.

После успешного создания телеграм-бота через «BotFather» и получения токена доступа, можно приступать к разработке функциональности бота с использованием языка программирования Python. Токен будет использоваться в коде для аутентификации и связи с Telegram API, позволяя боту отправлять и получать сообщения, обрабатывать команды и выполнять другие функции взаимодействия с пользователями.

Для удобства необходимо создать папку телеграм-бота на персональном компьютере, а в ней создать файл формата .py. Назовем его v01.py. Затем следует открыть файл в IDE «PyCharm».

Для разработки телеграм-бота необходимо импортировать определенные библиотеки языка программирования Python. Эти библиотеки предоставляют набор функций и инструментов, которые позволяют взаимодействовать с Telegram API и реализовывать функциональность бота. Вот список импортируемых библиотек и их роли:

– logging: данная библиотека предоставляет возможность ведения журнала событий и отладочной информации. Она позволяет записывать сообщения различных уровней важности, что упрощает процесс отладки и обеспечивает информацию о происходящих событиях;

– requests: библиотека requests предоставляет возможности для выполнения HTTP-запросов. Она используется для взаимодействия с внешними API и получения данных из внешних источников;

– time: библиотека time предоставляет функции для работы с временем и задержками в коде. Она используется для управления временными интервалами, планирования задач и организации циклов выполнения;

– schedule: данная библиотека предоставляет возможности для планирования и выполнения задач по расписанию. Она позволяет установить определенное время или интервалы для запуска функций и методов;

– datetime: библиотека datetime предоставляет классы и функции для работы с датой и временем. Она используется для работы с датами, временными интервалами и форматированием даты и времени;

– telegram: эта библиотека предоставляет классы и методы для взаимодействия с Telegram API. Она позволяет отправлять и получать сообщения, управлять клавиатурой бота, обрабатывать команды и другие события;

– telegram.error: библиотека telegram.error предоставляет классы исключений, связанных с ошибками взаимодействия с Telegram API. Она используется для обработки и управления возможными ошибками, возникающими при работе с API;

– telegram.ext: эта библиотека предоставляет расширенный функционал для работы с Telegram API. Она предоставляет дополнительные классы и методы для создания обработчиков команд, событий и сообщений, а также для настройки и запуска бота [17].

Используя указанные библиотеки, можно создать телеграм-бота с необходимым функционалом и взаимодействовать с Telegram API для обработки и отправки сообщений, управления клавиатурой бота, планирования задач и других операций.

```
import logging
import requests
import time
import schedule
from datetime import datetime, timedelta
from telegram import Bot, Update, ReplyKeyboardMarkup, InlineKeyboardMarkup, InlineKeyboardButton
from telegram.error import BadRequest
from telegram.ext import Updater, CommandHandler, CallbackContext, MessageHandler, Filters, CallbackQueryHandler
```

Рисунок 9 – Импорт нужных библиотек и модулей в файл

Функция logging.basicConfig() используется для настройки системы логирования в приложении. В данном случае, вызов basicConfig() устанавливает следующие настройки:

format: Формат сообщения лога. В данном случае, формат сообщения задан как '%(asctime)s - %(name)s - %(levelname)s - %(message)s'. Здесь:

– %(asctime)s – дата и время в формате строки;

– %(name)s – имя логгера;

– %(levelname)s – уровень логирования;

– %(message)s – текст сообщения лога;

– level: уровень логирования. Установлен уровень logging.INFO, что означает, что будут записываться сообщения с уровнем логирования INFO и выше. Уровни логирования в порядке возрастания: DEBUG, INFO, WARNING, ERROR, CRITICAL [15].

Функция basicConfig() устанавливает эти настройки для модуля logging во всем приложении. После настройки можно использовать методы модуля logging, такие как logging.debug(), logging.info(), logging.warning() и т.д., для записи логов с заданным форматом и уровнем логирования.

В данном коде логирование будет производиться в консоль, так как не указан параметр filename (имя файла для сохранения логов).

Теперь необходимо ввести переменную TOKEN, которая представляет собой токен бота Telegram. Этот токен используется для авторизации и связи с API Telegram.

В коде объект Bot создается с использованием токена TOKEN. Вот соответствующая строка кода:

– bot = Bot(token=TOKEN).

Эта строка создает экземпляр класса Bot из модуля telegram и присваивает его переменной bot. При создании объекта Bot, параметр token устанавливается равным значению переменной TOKEN. Таким образом, токен, который был определен в переменной TOKEN, используется для инициализации объекта Bot.

Объект Bot предоставляет методы для взаимодействия с API Telegram, такие как отправка сообщений, удаление сообщений, получение информации о чате и другие. После создания объекта Bot, можно вызывать соответствующие методы для выполнения операций с ботом Telegram.

Теперь необходимо разработать функции кнопок и взаимодействий бота и пользователя. Для этого необходимо создать функцию main, в которой были бы обработчики каждой из кнопок. Также необходимо прописать функции самих кнопок.

Функция delete_message(message_id): данная функция используется для удаления сообщения из чата в Telegram. Она принимает в качестве аргумента message_id – идентификатор сообщения, которое необходимо удалить. Внутри функции вызывается метод bot.delete_message(chat_id, message_id), где bot является объектом класса Bot и предоставляет доступ к API Telegram. Этот метод отправляет запрос на удаление сообщения с указанным идентификатором из указанного чата (chat_id).

Функция start(update: Update, context: CallbackContext): данная функция обрабатывает команду /start, которая является одной из стандартных команд для запуска бота в Telegram. Она принимает объекты update и context, предоставляемые Telegram при обновлении состояния бота. Функция используется для отправки главного меню с кнопками пользователю. Внутри функции создается объект ReplyKeyboardMarkup, который содержит раскладку кнопок меню, и вызывается метод update.message.reply_text, чтобы отправить сообщение с текстом и кнопками меню пользователю.

Функция open_timetable(update: Update, context: CallbackContext): эта функция обрабатывает нажатие на кнопку «Расписание». Она отправляет пользователю ссылку на расписание, а затем вызывает функцию clear_chat, чтобы удалить предыдущие сообщения, и функцию send_main_menu, чтобы отправить кнопки главного меню пользователю.

Функция help_menu(update: Update, context: CallbackContext): эта функция обрабатывает нажатие на кнопку «Справка». Она отправляет пользователю меню справки, содержащее кнопки с различными вариантами запросов. Внутри функции создается объект ReplyKeyboardMarkup, содержащий раскладку кнопок меню справки, и вызывается метод update.message.reply_text, чтобы отправить сообщение с текстом и кнопками пользователю. Затем вызываются функции clear_chat и help_menu, чтобы удалить предыдущие сообщения и отправить кнопки меню справки.

Функция deanery_info(update: Update, context: CallbackContext): данная функция обрабатывает нажатие на кнопку «Деканат». Она отправляет информацию о деканате пользователю и вызывает функцию help_menu, чтобы отправить кнопки меню справки.

Функция departments_info(update: Update, context: CallbackContext): эта функция обрабатывает нажатие на кнопку «Кафедры». Она отправляет информацию о кафедрах пользователю и вызывает функцию help_menu, чтобы отправить кнопки меню справки.

Функция fmi_media_info(update: Update, context: CallbackContext): данная функция обрабатывает нажатие на кнопку «ФМиИ Медиа». Она отправляет информацию о социальных сетях факультета пользователю и вызывает функцию help_menu, чтобы отправить кнопки меню справки.

Функция back_to_main_menu(update: Update, context: CallbackContext): функция обрабатывает нажатие на кнопку «Вернуться в главное меню». Она вызывает функцию clear_chat, чтобы удалить предыдущие сообщения, и функцию send_main_menu, чтобы отправить кнопки главного меню пользователю.

Функция info_menu(update: Update, context: CallbackContext): данная функция обрабатывает нажатие на кнопку «Информация». Она отправляет пользователю меню информации, содержащее кнопки с различными видами информации. Внутри функции создается объект ReplyKeyboardMarkup, содержащий раскладку кнопок меню информации, и вызывается метод update.message.reply_text, чтобы отправить сообщение с текстом и кнопками пользователю. Затем вызываются функции clear_chat и info_menu, чтобы удалить предыдущие сообщения и отправить кнопки меню информации.

Функция maim_info(update: Update, context: CallbackContext): функция обрабатывает нажатие на кнопку «МАиМ». Она отправляет информацию о кафедре МАиМ пользователю и вызывает функцию info_menu, чтобы отправить кнопки меню информации.

Функция iius_info(update: Update, context: CallbackContext): данная функция обрабатывает нажатие на кнопку «ИиУС». Она отправляет информацию о кафедре ИиУС пользователю и вызывает функцию info_menu, чтобы отправить кнопки меню информации.

Функция omii_info(update: Update, context: CallbackContext): эта функция обрабатывает нажатие на кнопку «ОМиИ». Она отправляет информацию о кафедре ОМиИ пользователю и вызывает функцию info_menu, чтобы отправить кнопки меню информации.

Функция handle_message(update: Update, context: CallbackContext): данная функция обрабатывает входящие сообщения от пользователя. Она вызывает функцию clear_chat, чтобы удалить предыдущие сообщения, и функцию send_main_menu, чтобы отправить кнопки главного меню пользователю.

Функция clear_chat(update: Update, context: CallbackContext): функция важна и используется для удаления предыдущих сообщений в чате. Она принимает два аргумента: update и context. update содержит информацию о входящем сообщении, а context предоставляет различные полезные методы и атрибуты для обработки сообщений. Внутри функции происходит следующий процесс удаления сообщений:

– получение идентификатора чата: chat_id = update.effective_chat.id. Здесь используется атрибут effective_chat объекта update, чтобы получить идентификатор текущего чата, в котором происходит взаимодействие с пользователем. Этот идентификатор будет использоваться для удаления сообщений из этого чата;

– получение идентификатора последнего сообщения: last_message_id = update.message.message_id. В объекте update содержится информация о входящем сообщении, и метод message_id позволяет получить идентификатор последнего сообщения в чате;

– удаление предыдущих сообщений: for message_id in range(last_message_id - 1, 0, -1): context.bot.delete_message(chat_id, message_id). В этом цикле происходит удаление предыдущих сообщений, начиная с предпоследнего и до самого первого сообщения. Используется функция delete_message объекта bot из context, чтобы удалить каждое сообщение по его идентификатору. Цикл выполняется в обратном порядке с помощью функции range, чтобы удалить сообщения в правильной последовательности;

– удаление текущего сообщения: context.bot.delete_message(chat_id, last_message_id). Наконец, последнее сообщение (текущее сообщение) также удаляется с помощью вызова функции delete_message с его идентификатором.

Таким образом, функция clear_chat позволяет удалить предыдущие сообщения в чате, включая текущее сообщение, обеспечивая таким образом более чистый и актуальный интерфейс взаимодействия с пользователем.

Функция send_main_menu(update: Update, context: CallbackContext): данная функция отправляет пользователю главное меню с кнопками. Внутри функции создается объект ReplyKeyboardMarkup, содержащий раскладку кнопок главного меню, и вызывается метод update.message.reply_text, чтобы отправить сообщение с текстом и кнопками пользователю.

Общий подход к реализации данных функций состоит в использовании объектов класса Update и CallbackContext, предоставляемых Telegram, для

обработки входящих обновлений и отправки сообщений пользователю. Каждая функция отвечает за определенную логику обработки команд, нажатий на кнопки и отправки сообщений. Кроме того, функции clear_chat и send_main_menu вызываются внутри других функций для удобства удаления предыдущих сообщений и отправки главного меню.

В функции main вызываются функции add_handler для каждого обработчика команд и сообщений. Обработчики определяются с помощью класса CommandHandler и MessageHandler из модуля telegram.ext.

Обработчик команды /start: dp.add_handler(CommandHandler("start", start)). Здесь функция start будет вызываться при получении команды /start от пользователя.

Обработчики нажатия на кнопки главного меню, функции info_menu, open_timetable и help_menu будут вызываться соответственно при нажатии на кнопки «О факультете», «Расписание» и «Справка».

Функции maim_info, iius_info, omii_info и back_to_main_menu будут вызываться соответственно при нажатии на кнопки «МАиМ», «ИиУС», «ОМиИ» и «Назад».

Функции deanery_info, departments_info и fmi_media_info будут вызываться соответственно при нажатии на кнопки «Деканат», «Кафедры» и «ФМиИ Медиа».

Функция handle_message будет вызываться для любых текстовых сообщений, которые не соответствуют предыдущим обработчикам. Для этой функции также необходимо написать отдельную функцию. Функция handle_message принимает два аргумента: update и context. update содержит информацию о входящем сообщении, а context предоставляет различные полезные методы и атрибуты для обработки сообщений.

В начале функции выполняется проверка наличия сообщения в update. Если сообщение присутствует, то это означает, что пользователь отправил текстовое сообщение, которое не было обработано предыдущими обработчиками. В таком случае, функция выполняет следующие действия:

Отправка сообщения об ошибке: update.message.reply_text('Извините, я понимаю только кнопки.'). Здесь вызывается метод reply_text объекта update.message, чтобы отправить ответное сообщение пользователю с текстом «Извините, я понимаю только кнопки.» Это сообщение предупреждает пользователя о том, что его ввод не поддерживается.

Вызов функции back_to_main_menu: back_to_main_menu(update, context). Данная функция используется для возврата пользователя к главному меню после отправки сообщения об ошибке.

Удаление предыдущих сообщений: clear_chat(update, context). Здесь вызывается функция clear_chat, которая используется для удаления предыдущих сообщений бота. Это позволяет очистить чат от предыдущих ответов бота и обеспечить более чистый и организованный интерфейс.

Таким образом, если пользователь отправляет текстовое сообщение, которое не соответствует ни одному из ожидаемых вариантов, функция handle_message отправляет сообщение об ошибке, возвращает пользователя к главному меню и удаляет предыдущие сообщения бота для обеспечения более понятного и удобного взаимодействия с пользователем [14].

3.3 Интеграция парсера BeautifulSoup для получения информации из веб-страниц

Необходимо наладить механизм публикации в телеграм-канал информации о погоде в городе Благовещенск согласно установленному разработчиком расписанием для того, чтобы пользователи канала могли чаще заходить на канал, узнавая полезную для них информацию.

Функция get_weather() является ключевой для получения информации о погоде в заданном городе и публикации её в телеграм-канале. Рассмотрим основные особенности и важность каждого метода внутри этой функции:

Получение текущей погоды через OpenWeatherMap API:

– datetime.now(): получает текущую дату и время для использования в формировании запроса;

– city: заданный город, для которого будет получена информация о погоде;

– url: формирует URL-адрес для запроса к OpenWeatherMap API, включая город и ключ API.

Веб-сайт OpenWeatherMap предоставляет пользователям бесплатные ключи API после регистрации, которые позволяют получать доступ к различным метеорологическим данным. Для использования функционала получения информации о погоде в своем телеграм-боте, нам необходимо получить ключ API от OpenWeatherMap.

Чтобы получить ключ API OpenWeatherMap, нужно проследовать этим шагам:

– перейти на официальный веб-сайт OpenWeatherMap по адресу: https://openweathermap.org/;

– нажать на кнопку «Sign Up» или «Create Account» (Зарегистрироваться), чтобы создать новую учетную запись на OpenWeatherMap;

– после успешной регистрации и входа в свою учетную запись на OpenWeatherMap, найти страницу управления ключами API или API Keys. Обычно она находится в разделе «API Keys» или «My Account» (Моя учетная запись);

– нажать на кнопку «Generate Key» или «Create New Key» (Создать новый ключ), чтобы сгенерировать новый ключ API. Могут быть доступны различные типы ключей, в зависимости от требуемого функционала и ограничений;

– скопировать сгенерированный ключ API. Он выглядит как длинная строка символов, в нашем случае: «b30b2100af102ed7f6702f41cdbbf415»;

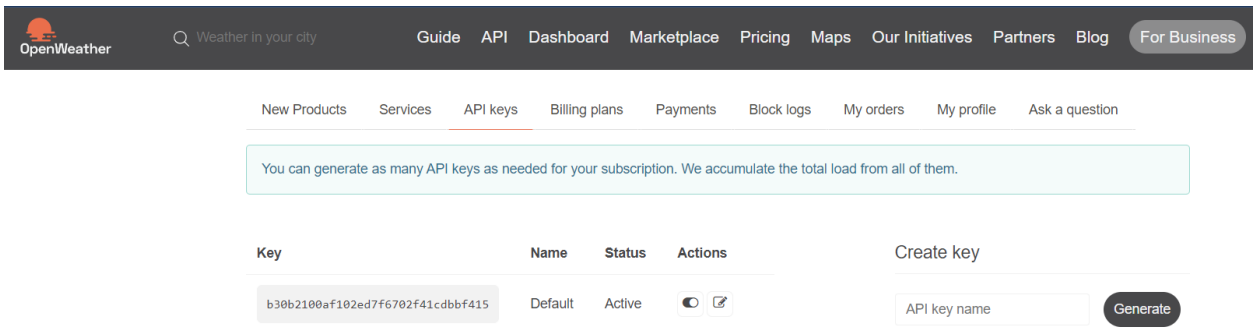

Рисунок 11 – API ключ на сайте OpenWeatherMap

– вставить скопированный ключ API в свой код телеграм-бота, в переменную WEATHER_API_KEY.

Теперь телеграм-бот сможет использовать ключ API OpenWeatherMap для получения актуальной информации о погоде в указанном городе.

Отправка запроса и обработка JSON-ответа:

– requests.get(url): отправляет GET-запрос на указанный URL-адрес для получения данных о погоде;

– response.json(): преобразует ответ от сервера в формате JSON в структуру данных Python.

JSON (JavaScript Object Notation) – это легкий формат обмена данными, основанный на синтаксисе объектов JavaScript. Он широко используется для передачи данных между клиентскими веб-приложениями и сервером, а также в других контекстах, включая API.

Формат JSON представляет данные в виде пар «ключ-значение», где ключи – это строки, а значения могут быть различных типов данных, таких как строки, числа, логические значения, массивы или другие вложенные объекты. В контексте запроса requests.get(url) он отправляет GET-запрос по указанному URL-адресу, в нашем случае к OpenWeatherMap API, для получения данных о погоде. Полученный ответ от сервера представляет собой строку в формате JSON. Затем, с помощью метода response.json(), эта строка преобра-

зуется в структуру данных Python, которую можно легко использовать и извлекать необходимую информацию.

Формирование сообщения о погоде:

– создание сообщения, включающего информацию о погоде в заданном городе;

– добавление дополнительного сообщения (admessage) на основе условий температуры и влажности.

Публикация сообщения в телеграм-канале:

– bot.send_message(): отправка сообщения в указанный чат (канал) с помощью объекта bot;

– использование переменной CHANNEL_ID для определения идентификатора канала, в который будет отправлено сообщение.

Идентификатор телеграм-канала или пользователя (также известный как Chat ID) – это уникальный числовой идентификатор, который присваивается каждому каналу или пользователю в Telegram. Он используется для идентификации конкретного чата или пользователя при взаимодействии с Telegram API.

Для получения идентификатора телеграм-канала или пользователя можно воспользоваться ботом «IDBot». Необходимо просто отправить ссылку на нужный телеграм-канал или пользователя боту, и он ответит числовым идентификатором. Идентификатор телеграм-канала или пользователя необходим для отправки сообщений или выполнения других действий от имени бота в указанном чате или с пользователем. В коде выше идентификатор канала используется в функции bot.send_message(chat_id=CHANNEL_ID, text=message), чтобы опубликовать сообщение о погоде в заданном канале.

Идентификаторы являются уникальными для каждого чата или пользователя, и необходимо обеспечить точность и актуальность использования правильного идентификатора для правильного канала или пользователя.

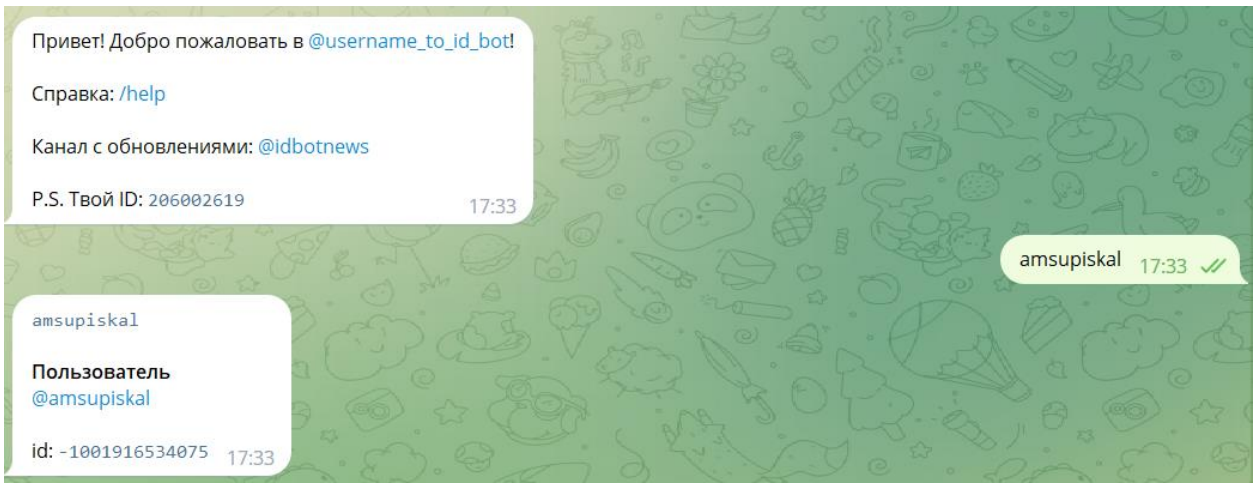

Рисунок 12 – Определение идентификатора канала

Задержка и удаление сообщения: time.sleep(10): Ожидание 10 секунд после публикации сообщения для предотвращения слишком частых обновлений.

Расписание (schedule): используется для планирования удаления сообщения через 2 часа с помощью функции delete_message(). Задержка обеспечивает, что сообщение о погоде будет отображаться в телеграм-канале в течение определенного времени, так как оно потеряет свою актуальность и станет «мусором», засоряющим пространство для публикаций.

Функция delete_message(message_id) отвечает за удаление сообщения из телеграм-канала. Она принимает идентификатор сообщения (message_id) и использует объект bot для выполнения удаления с помощью метода bot.delete_message().

В функции main добавляется код для планирования выполнения функции get_weather() в определенное время каждый день с помощью schedule.every().day.at(). Это позволяет автоматически запускать функцию get_weather() в установленное время, чтобы обновлять информацию о погоде в телеграм-канале.

Важность этих методов заключается в обеспечении регулярного обновления информации о погоде в канале, что позволяет пользователям получать актуальные данные и заинтересовывает их возвращаться на канал для получения полезной информации. Кроме того, использование планировщика за-

дач schedule позволяет автоматизировать процесс публикации и удаления сообщений, освобождая разработчика от необходимости выполнять эти действия вручную. В качестве времени для публикации информации о погоде, лучше всего выбрать следующее время: 8:00, 15:00, 19:00. Эти временные точки являются хорошим выбором, так как они представляют разные моменты дня и могут быть интересными для пользователей, желающих узнать о текущей погоде:

– время публикации информации о погоде в 8:00 утра может быть полезным для пользователей, которые хотят получить прогноз на весь день и спланировать свои активности;

– время публикации в 15:00 является удобным для тех, кто хочет узнать о прогнозе на вторую половину дня и может влиять на свои планы или решения;

– и, наконец, время публикации в 19:00 может быть интересным для пользователей, которые хотят получить последнюю информацию о погоде перед вечерними мероприятиями или планами.

Выбор этих временных точек позволяет обеспечить регулярное и информативное обновление о погоде в разные части дня, что может привлечь больше пользователей на телеграм-канал.

В функции main после добавления всех обработчиков команд и сообщений, следующим шагом является вызов метода start_polling() объекта updater. Этот метод запускает процесс получения обновлений от Telegram для бота.

После вызова start_polling(), мы входим в бесконечный цикл while True, который выполняется до тех пор, пока не будет прерван или остановлен. Внутри этого цикла выполняется две основные задачи: выполнение отложенных задач и пауза.

Выполнение отложенных задач: schedule.run_pending(). Так как в коде используется планировщик задач schedule, то вызов run_pending() позволяет выполнить все отложенные задачи, которые должны быть выполнены в те-

кущий момент времени. Это включает отправку автоматических сообщений, выполнение задач по расписанию и т.д.

Пауза: time.sleep(1). После выполнения отложенных задач происходит пауза на 1 секунду с помощью функции sleep() из модуля time. Это необходимо для снижения нагрузки на процессор и ограничения частоты опроса обновлений от Telegram.

Таким образом, цикл while True вместе с вызовом start_polling() позволяет боту постоянно ожидать и обрабатывать входящие обновления от Telegram, выполнять отложенные задачи и поддерживать активное взаимодействие с пользователями.

По мере написания кода, естественно, он запускался для проверки тех или иных методов. Были сохранены различные этапы кода:

– ptg.py был разработкой взаимодействия бота с пользователем;

– bs4test.py был разработкой получения информации с сайта OpenWeatherMap с помощью библиотеки BeautifulSoap и API ключа;

– bs4test_pub.py являлся отработкой алгоритма публикации информации о погоде с сайта OpenWeatherMap в определенное время, указанное разработчиком;

– v04.py является попыткой объединения кода ptg.py и bs4test_pub.py;

– v05.py является оптимизированной версией v04, что позволит коду работать быстро и без задержек;

– v06.py является версией с добавленным текстом, необходимым для понимания уже для пользователя.

Так как версия кода v06 и является основной для тестирования, и ее уже можно показывать потенциальному пользователю, то дадим коду префикс beta. Префикс «beta» широко используется в программной индустрии для обозначения версий программного обеспечения, которые находятся в стадии бета-тестирования. Бета-версия является промежуточной стадией между разработкой и окончательным выпуском продукта. Она обозначает, что программа находится в активной фазе тестирования.

4 ТЕСТИРОВАНИЕ И ДЕПЛОЙ БОТА НА СЕРВЕР

4.1 Тестирование функциональности и производительности телеграм-бота

Тестирование телеграм-бота представляет важную часть процесса разработки программного обеспечения. Тестирование бота позволяет выявить и исправить потенциальные ошибки, убедиться в правильной работе его функционала и обеспечить высокое качество и надежность программы перед ее деплоем на сервер.

Тестирование телеграм-бота включает в себя несколько этапов и подходов, которые помогают проверить различные аспекты его функционирования. При тестировании телеграм-бота также важно уделить внимание тестированию пользовательского интерфейса. Тестирование пользовательского интерфейса может включать проверку отображения и расположения элементов, обработку ввода пользователя, а также проверку правильности вывода информации и отправки сообщений.

Важным аспектом тестирования является также учет требований к боту и ожидаемого поведения. Необходимо проверить, что бот выполняет свои функции согласно спецификации и обрабатывает входные данные и команды пользователя соответствующим образом.

После проведения тестирования и обнаружения возможных ошибок и проблем, необходимо внести исправления и повторно протестировать бот, чтобы убедиться в правильной работе после внесенных изменений.

Тестирование телеграм-бота является важным этапом в процессе его разработки и деплоя на сервер. Это позволяет обеспечить качество и надежность бота, улучшить пользовательский опыт и предотвратить возможные ошибки и проблемы при его использовании.

Во время тестирования бота, будем запускать его, используя командную строку (терминал), чтобы наблюдать его поведение и получать обратную

связь. Для тестирования функции взятия информации о погоде и публикации, установим время публикации на 5 минут позже запуска, на 8 и на 9 минут.

```
schedule.every().day.at("20:35").do(get_weather)
schedule.every().day.at("20:38").do(get_weather)
schedule.every().day.at("20:39").do(get_weather)
```

Рисунок 13 – Настройка времени публикации в коде

Для запуска бота в режиме тестирования, необходимо:

– открыть командную строку (терминал) на компьютере;

– перейти в каталог, где находится код бота. Это может быть каталог, содержащий файлы скрипта бота, или путь к директории проекта, в зависимости от вашей организации файловой системы;

– запустить бота, выполнив команду, которая вызывает функцию main(): beta_v06.py.

```
Командная строка - beta_v06.py

C:\>cd telegram

C:\Telegram>beta_v06.py
2023-05-23 20:41:14,451 - apscheduler.scheduler - INFO - Scheduler started
```

Рисунок 14 – Запуск кода бота в терминале

После запуска бота, необходимо обратить внимание на возможные ошибки или предупреждения, которые могут появиться в выводе.

В нашем случае, запуск кода ошибок не выдал. Поэтому следует перейти в Telegram, чтобы протестировать бота. Нужно протестировать функциональность бота, нажимая на кнопки, отправляя команды и сообщения в чат. Нужно убедиться, что бот правильно реагирует на команды, обрабатывает сообщения и отправляет ожидаемые ответы.

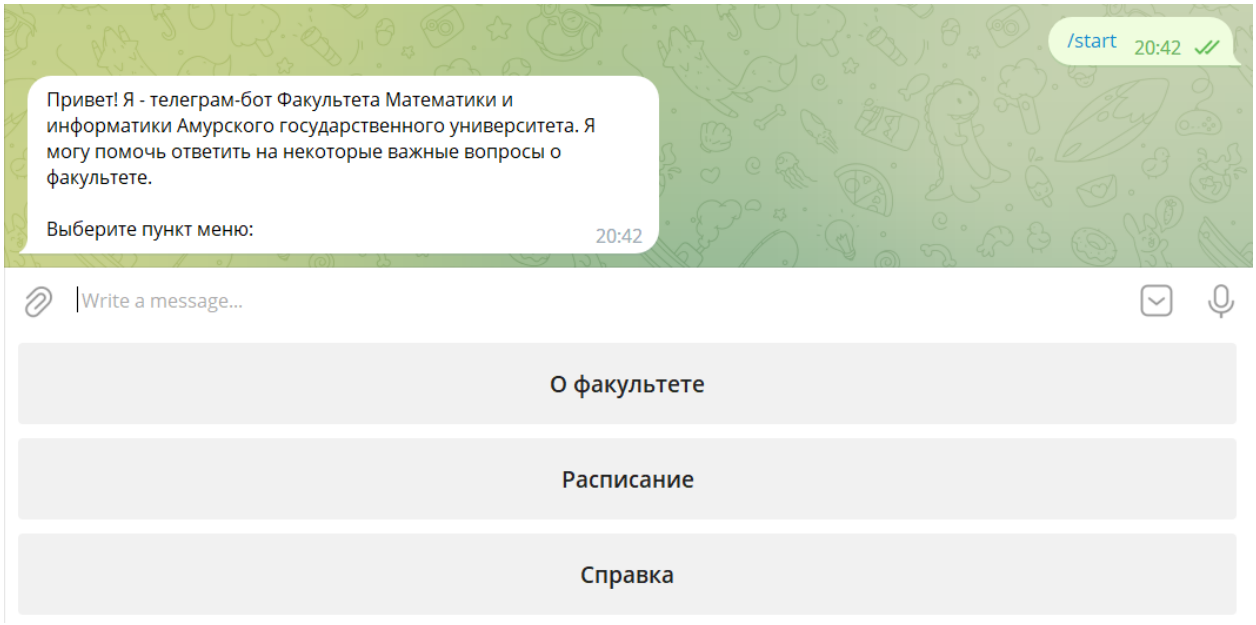

Рисунок 15 – Приветственное сообщение телеграм-бота

Во время тестирования необходимо обращать внимание на любые ошибки, неправильные ответы или нежелательное поведение бота.

Были проверены все разделы бота, вся информация отображается корректно, кнопки работают без ошибок.

Рисунок 16 – Выдача после нажатия кнопки «О факультете»

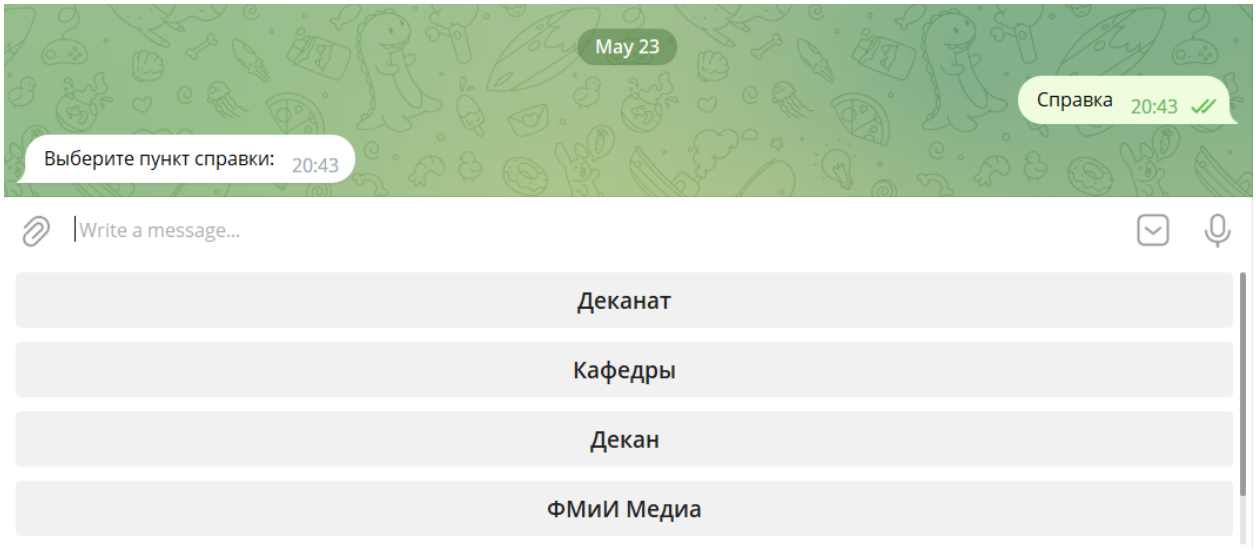

Рисунок 17 – Выдача после нажатия кнопки «Справка»

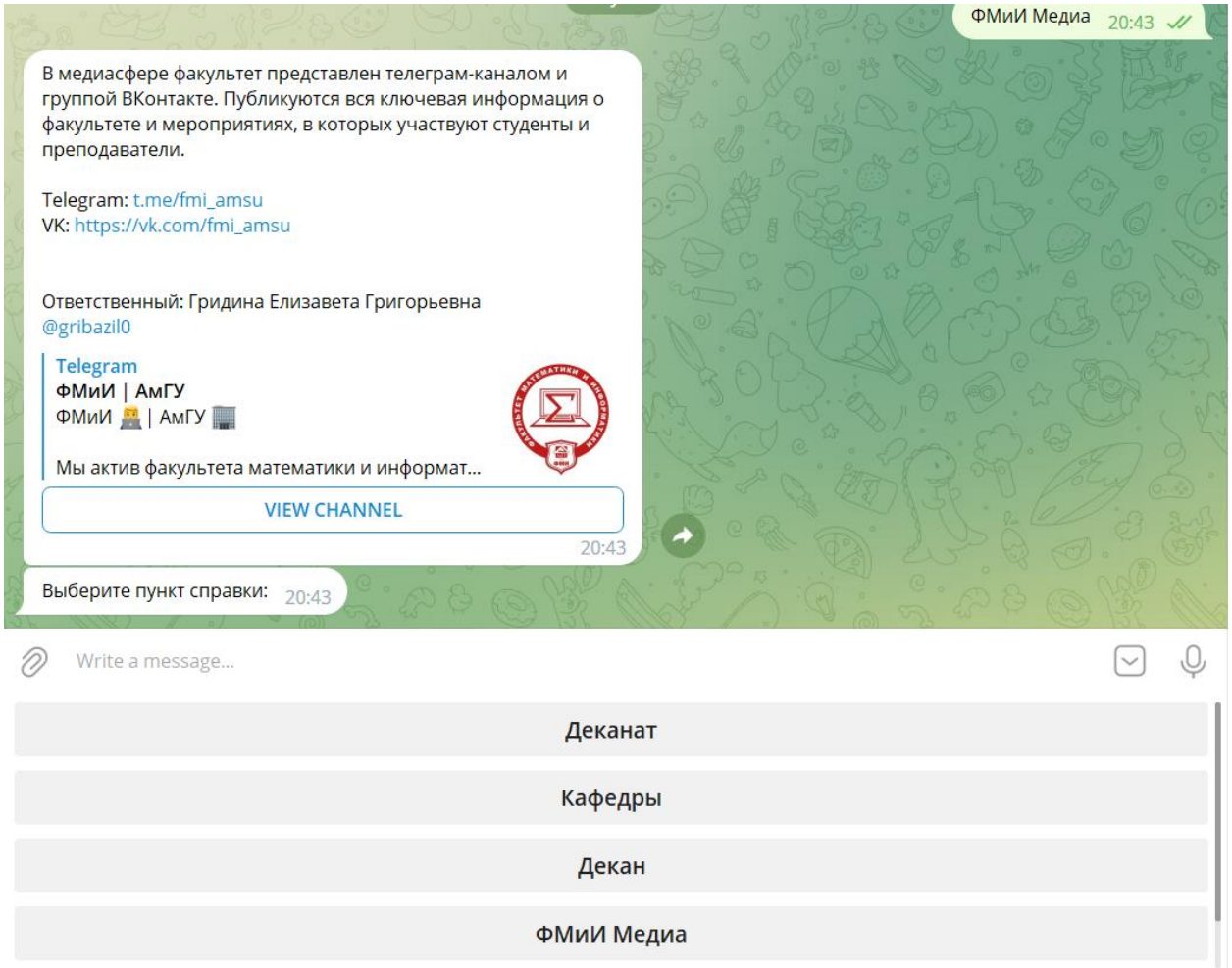

Рисунок 18 – Выдача после нажатия кнопки «ФМиИ Медиа» в меню «Справка»

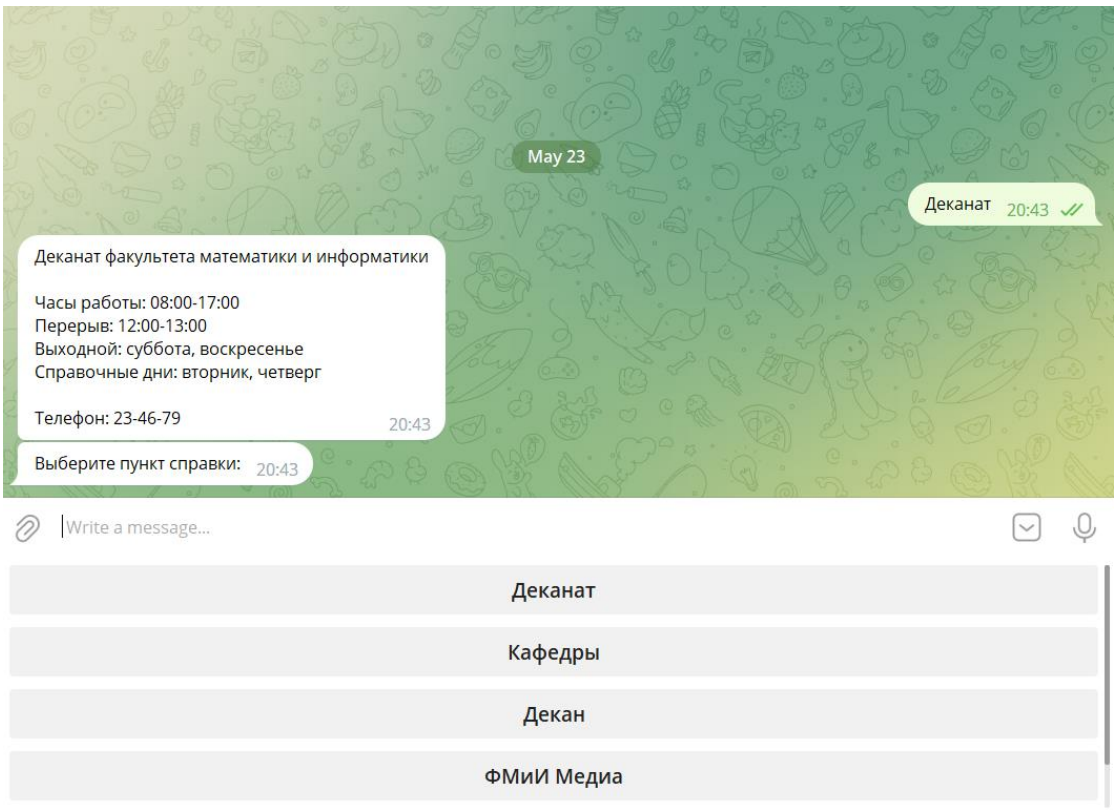

Рисунок 19 – Выдача после нажатия кнопки «Деканат» в меню «Справка»

Видно, что кнопки работают корректно, информация выводится без ошибок, а также чистится чат, чтобы пользователю было удобно пользоваться ботом. Во время тестирования взаимодействия бота с пользователем, также было проверена и публикация информационного сообщения в тестовый телеграм-канал.

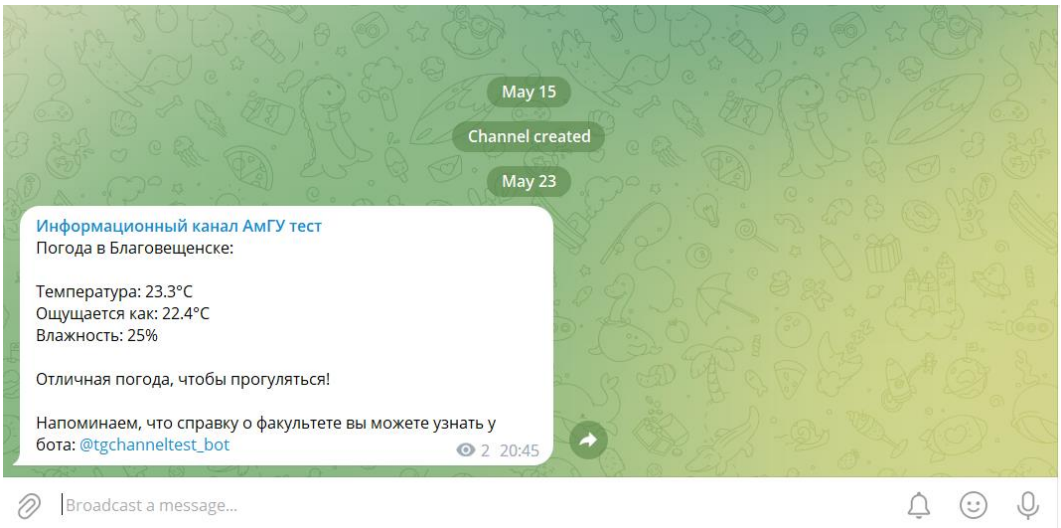

Рисунок 20 – Выдача после нажатия кнопки «О факультете»

4.2 Деплой бота на сервер

Деплой (развертывание) телеграм-бота на сервере является важным шагом в разработке и предназначен для обеспечения доступности и функционирования бота в онлайн-среде. Вот несколько основных причин, почему деплой бота на сервере является необходимым:

– постоянная доступность: размещение бота на сервере обеспечивает его непрерывную работу и доступность для пользователей в любое время. Бот будет готов к обработке запросов и предоставлению информации 24/7, даже если устройство, на котором был запущен, выключено или отключено от интернета;

– масштабируемость: серверное развертывание позволяет масштабировать бота для обработки большого количества запросов и обслуживания большого числа пользователей. Выделенные ресурсы сервера позволяют боту работать более эффективно и эластично, а также обрабатывать больший объем данных;

– улучшенная производительность: запуск бота на сервере может повысить его производительность, поскольку сервер обычно обладает большей вычислительной мощностью и ресурсами, чем отдельное устройство. Это позволяет боту отвечать на запросы быстрее и более отзывчиво;

– системная стабильность: серверное развертывание обеспечивает более стабильное окружение для работы бота. Серверы обычно поддерживаются и обновляются для обеспечения безопасности и стабильности работы, что уменьшает вероятность возникновения сбоев или проблем связанных с окружением;

– удобное обновление и сопровождение: размещение бота на сервере упрощает процесс обновления и сопровождения. Вы можете легко внести изменения в код и обновить бота на сервере без прерывания его работы. Это облегчает поддержку и дальнейшее развитие бота;

– безопасность: при правильной настройке серверного окружения можно обеспечить высокий уровень безопасности для бота и его пользователей. Можно применить различные меры безопасности, такие как шифрование данных и защита от несанкционированного доступа.

Деплой бота на сервере обеспечивает его стабильную работу, доступность и производительность, а также упрощает его обновление и сопровождение. Это важный шаг для предоставления качественного сервиса пользователям и обеспечения успешного функционирования телеграм-бота в онлайн-среде.

Для деплоя бота был выбран сервис Beget. Beget – это хостинг-провайдер, предоставляющий услуги размещения и хостинга веб-сайтов и приложений. Он является одним из популярных и надежных хостинг-провайдеров, предлагающих широкий спектр услуг для хостинга веб-сайтов, включая такие функциональности, как размещение файлов, баз данных, почтовые сервисы и многое другое.

Beget обеспечивает высокую доступность и стабильность своих серверов, что гарантирует непрерывную работу веб-сайтов и приложений. Они используют современное оборудование и осуществляют регулярное обслуживание для обеспечения оптимальной производительности.

Серверы Beget обладают высокой вычислительной мощностью, быстрым доступом к данным и оптимизированным программным обеспечением. Это позволяет обеспечить быструю загрузку веб-страниц и отзывчивость веб-приложений. Также Beget предоставляет простой и интуитивно понятный интерфейс управления хостингом, который позволяет пользователям легко настраивать и управлять своими веб-сайтами и приложениями. Они также предоставляют широкий спектр инструментов и функций для работы с файлами, базами данных, электронной почтой и другими сервисами.

Сервис обеспечивает высокий уровень безопасности для хостинга веб-сайтов и приложений. Они применяют меры защиты от несанкционированно-

го доступа, регулярно обновляют программное обеспечение и предоставляют инструменты для управления безопасностью данных.

Beget является популярным выбором для размещения веб-сайтов и приложений благодаря своей надежности, высокой производительности, удобству использования и качественной технической поддержке.

Для того, чтобы развернуть функционирование бота на сервере, необходимо зарегистрироваться на сервисе и ввести платежную информацию. После этого нужно выбрать подходящий тариф, в нашем случае хватит минимального.

После получения места на сервере и автоматической установки выбранной операционной системы.

Сразу после выбора тарифа, сервис предложит выбрать сервер и операционную систему, которую необходимо установить в нашу ячейку. Была выбрана ОС Ubuntu. Ubuntu – это один из наиболее популярных и широко используемых дистрибутивов операционной системы Linux. Он основан на Debian и предоставляет пользователю открытую и бесплатную операционную систему с обширным набором функций и возможностей. Ubuntu широко используется как на персональных компьютерах, так и на серверах. Он предлагает гибкость, надежность и возможность настройки под различные потребности пользователей.

Далее сервис переводит на экран управления хостингом (Рис. 21). После входа в хостинг-панель Beget и выбора нужного аккаунта, вы будете перенаправлены на экран управления хостингом. На этом экране необходимо найти и выбрать раздел «Файловый менеджер». При выборе данного раздела, появится файловая система, представляющая арендованное место на сервере.

Далее потребуется создать новую папку с именем «telegrambot». Для этого, в файловой системе, нужно найти соответствующую кнопку или опцию, позволяющую создать новую папку. После создания папки «telegrambot», нужно перейти в нее.

Затем, с помощью кнопки на панели управления, обозначенной как «Загрузить Файлы», необходимо загрузить python-файл телеграм-бота с названием «beta_v06.py». При нажатии на эту кнопку, будет предоставлена возможность выбрать файл на компьютере для загрузки на сервер.

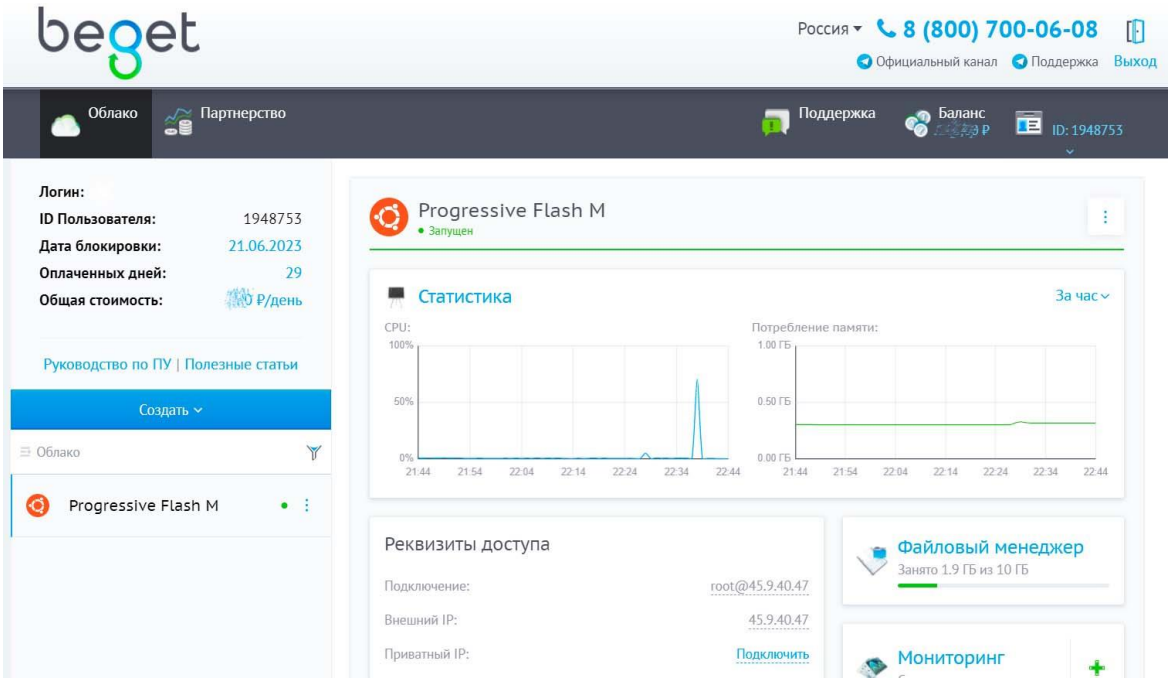

Рисунок 21 – Экран управления хостингом Beget

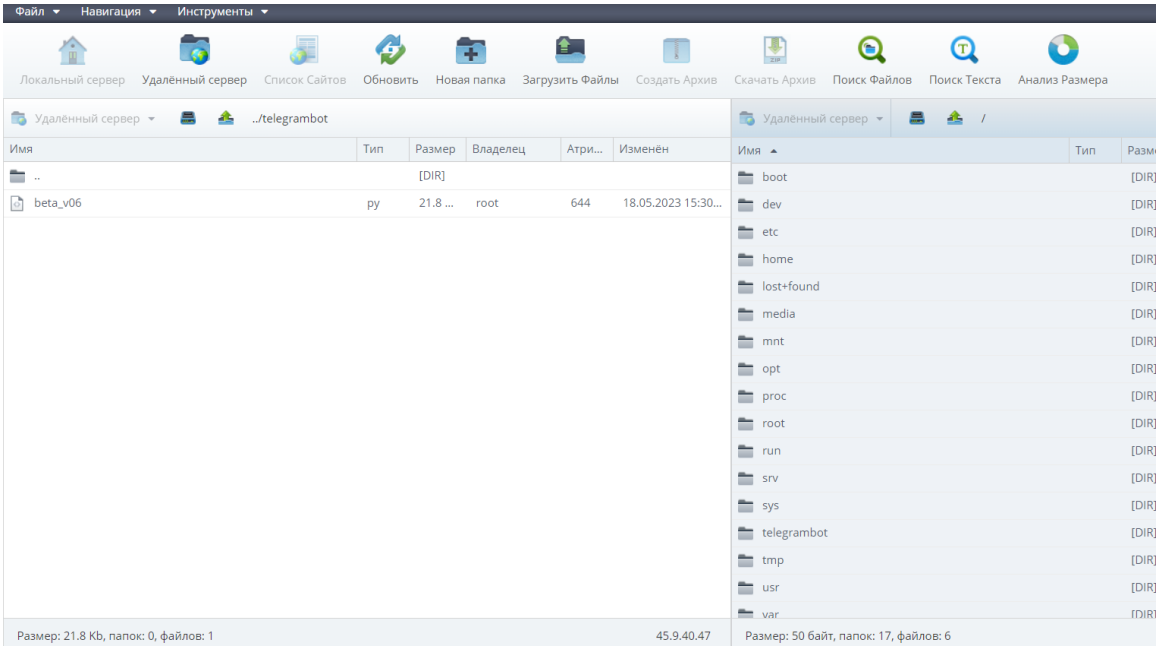

Рисунок 22 – Файловый менеджер хостинга Beget, папка «telegrambot» с файлом «beta_v06.py»

Затем нужно вернуться на панель управления и выбрать раздел «Терминал». В нем необходимо получить доступ к нашему месту на сервере с помощью логина «root» и пароля, отправленному на почту при регистрации. После этого нужно прописать путь к файлу бота.

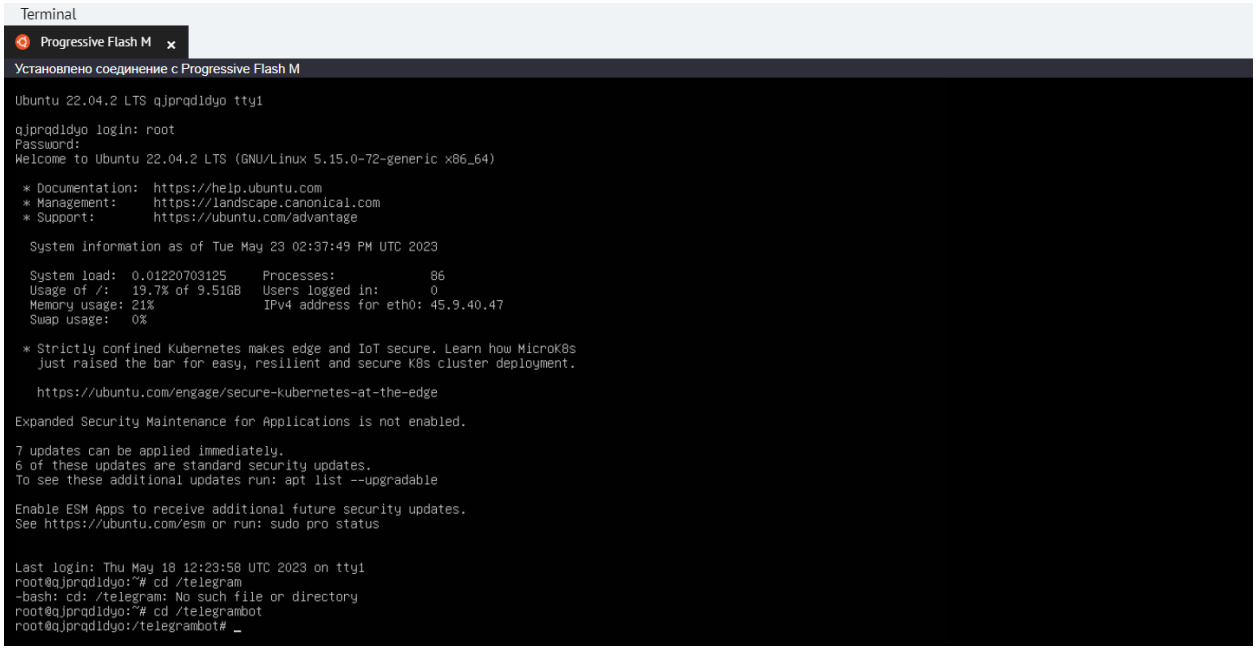

Рисунок 23 – Авторизация на сервере и переход в папку «telegrambot»

После этого шага нам необходимо узнать время на сервере с помощью команды «date», чтобы понять, какое время нужно установить в коде для планирования публикаций с помощью библиотеки schedule. По результатам запроса стало ясно, что время на сервере отличается от времени в городе Благовещенск на минус 9 часов. Поэтому в файле кода, который был загружен на сервер, меняем с времени 08:00, 15:00 и 19:00 на 23:00, 06:00 и 10:00 соответственно.

Сохранив файл, необходимо, вернувшись на вкладку терминала, запустить файл телеграм-бота с помощью следующей команды: nohup python3 beta_v06.py.

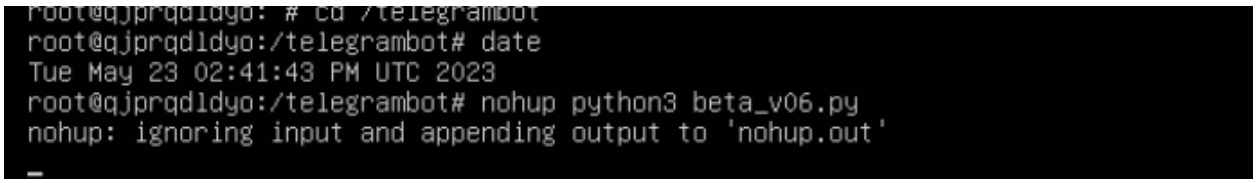

Рисунок 24 – Запуск бота на сервере

Осталось проверить работу бота в Telegram. После введенной команды «/start» телеграм-бот отвечает, значит он полностью функционирует и готов помогать студентам и преподавателям факультета.

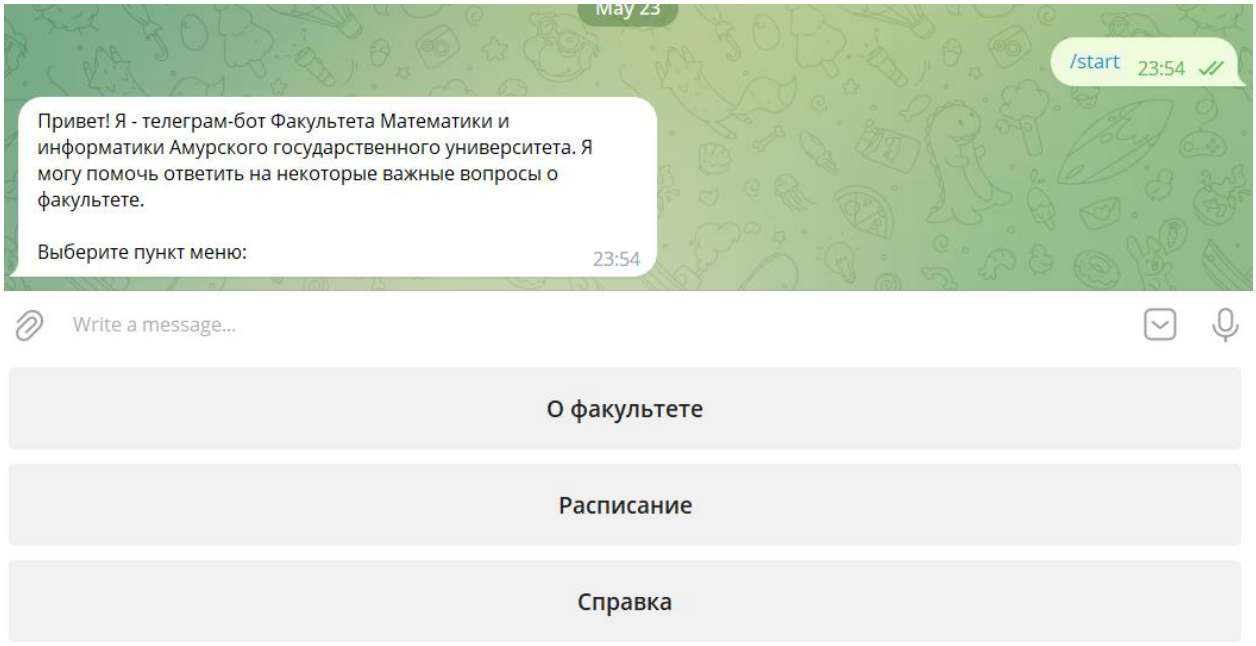

Рисунок 25 – Функционирующий телеграм-бот, запущенный с сервера

ЗАКЛЮЧЕНИЕ

Телеграм-бот является современным инструментом коммуникации, обладающим широким спектром применений и преимуществами. Использование телеграм-ботов имеет значительную важность в наше время. Они облегчают коммуникацию, предоставляют быстрый доступ к информации, а также способствуют автоматизации определенных задач.

В рамках данной бакалаврской работы была проведена разработка и реализация телеграм-бота, предназначенного для студентов и преподавателей Факультета математики и информатики Амурского государственного университета. Основная цель разработки телеграм-бота заключалась в обеспечении пользователей актуальной информацией о погоде в городе, а также предоставлении справочных данных о факультете. Бот был разработан с использованием библиотек Python-telegram-bot, BeautifulSoup и Schedule, что позволило реализовать функциональность получения погоды, обработку команд и расписание публикации сообщений.

Результатом работы является функционирующий телеграм-бот, который уже сейчас может быть полезен студентам и преподавателям Факультета математики и информатики. Бот предоставляет информацию о погоде, что помогает планировать активности на улице, а также предоставляет справочные данные о факультете, упрощая доступ к необходимой информации. Возможности бота могут быть расширены и усовершенствованы в дальнейшем. Например, можно добавить функцию получения новостей и анонсов факультета, создать интерактивные опросы или оповещения о важных событиях. Такое усовершенствование позволит боту стать полезным инструментом для всего университетского сообщества.

Таким образом, бакалаврская работа демонстрирует успешную разработку и функционирование телеграм-бота, а также отработку библиотек Python-telegram-bot, BeautifulSoup и Schedule. Разработанный бот представляет полезный инструмент для студентов и преподавателей Факультета мате-

матики и информатики Амурского государственного университета, а его потенциальные усовершенствования позволят расширить его полезность для всего университетского сообщества.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Беднов, А.В. Инструменты и этапы построения архитектуры расширяемого чат-бота для изучения английского языка / А.В. Беднов // Молодой ученый. – 2022. – №. 30. – С. 4-5.

2 Бородин, А.И. Методы обработки текста при создании чат-ботов / А.И. Бородин, Р.Р. Вейнберг, О.В. Литвишко // Гуманитарные балканские исследования. – 2019. – №. 3. – С. 108-111.

3 Галузина, В.И. Интеллектуальные чат-боты, технологии их создания и применение / В.И. Галузина // Инновации. Наука. Образование. – 2021. – №. 34. – С. 1406-1410.

4 Головченко, Н.А. Чат-боты в образовательной среде и бизнесе : ст. в сб. тр. конф. / Н.А. Головченко // Образование, наука, культура: традиции и современность. – Краснодар. : Изд-во Юг, 2022. – С. 78-81.

5 Доусон, М. Программируем на Python : учеб. пособие / М. Доусон. – СПб. : изд-во Питер, 2022. – 416 с.

6 Захарьева, Д.В. Разработка чат-ботов на платформе телеграм / Д.В. Захарьева, А.Г. Мацкевич // Сборник трудов XII моложедного научного форума МТУСИ телекоммуникации и информационные технологии реали возможности перспективы. – М. : МТУСИ, 2021. – С. 76-84.

7 Каспарова, Д.А. Искусственный интеллект в соцсетях: чат-боты как новый инструмент достижения маркетинговых целей / Д.А. Каспарова, К.А. Голубева, С.А. Мусатова // Шаг в будущее: искусственный интеллект и цифровая экономика. : ст. в сб. тр. конф. – М. : гос. ун-т управления, 2017. – С. 6-12.

8 Корнев, М.С. Создание контента для интеллектуальных чат-ботов на примере чат-бота «Эли» проекта ЮНЕСКО / М.С. Корнев // Журналистика в 2020 году: творчество, профессия, индустрия. : сб. материалов международной науч.-практич. конф. – М., 2021. – С. 201-202.

9 Кучерук, А.А. Применение нейросетевых технологий для разработки чат-ботов и диалоговых систем : науч. ст. / А.А. Кучерук // Безопасность. Управление. Искусственный интеллект. – 2022. – №. 1. – С. 28-34.

10 Лутц, М. Изучаем Python : учеб. пособие / М. Лутц. – изд-во Вильямс, 2020. – 720 с.

11 Лысенко, А.В. Технологии чат-ботов в CRM-технологиях / А.В. Лысенко // Карельский научный журнал. – 2019. – №. 1. – С. 84-85.

12 Моисеев, Я.Р. Использование телеграм-ботов для маркетинга / Я.Р. Моисеев, И.Н. Горбачёва // Проблемы экономики и управления: социокультурные, правовые и огранизационные аспекты. : сб. ст. – Кемерово. : Кузбасский гос. ун-т., 2022. – С. 430-436.

13 Петрова, А.О. Использование чат-ботов в интернет-коммуникациях / А.О. Петрова // Актуальные проблемы коммуникации: теория и практика. : ст. в сб. тр. конф. – Уфа. : Башкирский гос. ун-т., 2017. – С. 188-190.

14 Руководство : Бот в Telegram на Питоне от А до Я [Электронный ресурс]. – Режим доступа: https://otus.ru/journal/bot-v-telegram-na-pitone-ot-a-do-ya.

15 Руководство : Простой Telegram-бот на Python за 30 минут [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/442800.

16 Соболева, Е.Д. Обзор технологий создания чат-ботов / Е.Д. Соболева, М.А. Постникова, Р.Р. Маркарян, Ю.Е. Гапанюк // Естественные и технические науки. – М. : Изд-во «Спутник+», 2019. – №. 5. – С. 207-210.

17 Шариков, Д.В. Разработка Telegram-бота для администрирования небольшого сообщества в социальной сети ВКонтакте на языке программирования Python : науч. ст. / Д.В. Шариков // Системный администратор. – 2020 – №. 3. – С. 84-87.

18 Шафиев, Т.Р. Интеграция Telegram-ботов в информационных системах / Т.Р. Шафиев // Молодой ученый. – 2018. – №. 19. – С. 123-126.

ПРИЛОЖЕНИЕ А

Листинг программы бота

```
import logging

import requests

import time

import schedule

from datetime import datetime, timedelta

from telegram import Bot, Update, ReplyKeyboardMarkup, InlineKeyboardMarkup, InlineKeyboardButton

from telegram.error import BadRequest

from telegram.ext import Updater, CommandHandler, CallbackContext, MessageHandler, Filters, CallbackQueryHandler



logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s', level=logging.INFO)


TOKEN = "6133893198:AAF5aVO9iFD1MoaMe2yF54QhUHyBJYua8YI"

WEATHER_API_KEY = "b30b2100af102ed7f6702f41cdbbf415"

CHANNEL_ID = "-1001916534075"


bot = Bot(token=TOKEN)



def get_weather():

    # Получаем текущую дату и время
```

```
  now = datetime.now()
        # Формируем URL для запроса к OpenWeatherMap API
        city = "Благовещенск"
        url =
f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={WEATHER_API_KEY}"
        response = requests.get(url)
        data = response.json()

        # Извлекаем необходимые данные из JSON-ответа
        weather = data['weather'][0]['description'].capitalize()
        temperature = round(data['main']['temp'] - 273.15, 1)
        feels_like = round(data['main']['feels_like'] - 273.15, 1)
        humidity = data['main']['humidity']

        if temperature >= 17 and humidity <= 60:
                admessage = f"Отличная погода, чтобы прогуляться!"
        elif temperature >= 14 and humidity <= 70:
                admessage = f"Неплохая погода, чтобы прогуляться"
        else:
                admessage = f"Возможно, стоит посидеть в помещении.."

        # Формируем сообщение о погоде
        message = f"Погода в {city}е:\n\nТемпература: {temperature}°C\nОщущается как:
{feels_like}°C\nВлажность: {humidity}%\n\n{admessage}\n\nНапоминаем, что справку о
факультете вы можете узнать у бота: @tgchanneltest_bot"
```

```
    # Публикуем сообщение в канал
    sent_message = bot.send_message(chat_id=CHANNEL_ID, text=message)
    # Ждем 10 секунд
    time.sleep(10)

    # Удаляем сообщение из канала через 2 часа
    deletion_time = datetime.now() + timedelta(hours=2)
    schedule.every().day.at(deletion_time.strftime("%H:%M")).do(delete_message, message_id=sent_message.message_id)

def delete_message(message_id):
    # Удаляем сообщение из канала
    bot.delete_message(chat_id=CHANNEL_ID, message_id=message_id)

def start(update: Update, context: CallbackContext):
    # Отправляем главное меню с кнопками пользователю
    keyboard = [['О факультете'], ['Расписание'], ['Справка']]
    reply_markup = ReplyKeyboardMarkup(keyboard, one_time_keyboard=True)
    update.message.reply_text('Привет! Я - телеграм-бот Факультета Математики и информатики Амурского государственного университета. Я могу помочь ответить на некоторые важные вопросы о факультете.\n\nВыберите пункт меню:', reply_markup=reply_markup)

def open_timetable(update: Update, context: CallbackContext):
    # Ссылка на расписание
    timetable_url = 'https://www.amursu.ru/obrazovanie/timetable/timetable-group/'
```

```
        # Отправить пользователю ссылку на расписание
        update.message.reply_text(f"Расписание доступно по ссылке:\n{timetable_url}")

        # Отправить пользователю кнопки главного меню
        back_to_main_menu(update, context)

        # Удалить сообщения
        clear_chat(update, context)

def help_menu(update: Update, context: CallbackContext):
        # Отправляем меню справки
        keyboard = [['Деканат'], ['Кафедры'], ['Декан'], ['ФМиИ Медиа'], ['Назад']]
        reply_markup = ReplyKeyboardMarkup(keyboard, one_time_keyboard=True)
        update.message.reply_text('Выберите пункт справки:', reply_markup=reply_markup)
        # Удалить сообщения
    clear_chat(update, context)

def deanery_info(update: Update, context: CallbackContext):
        # Отправляем информацию о деканате
        update.message.reply_text('Деканат факультета математики и информатики\n\nЧасы работы: 08:00-17:00\nПерерыв: 12:00-13:00\nВыходной: суббота, воскресенье\nСправочные дни: вторник, четверг\n\nТелефон: 23-46-79')
        # Отправляем кнопки меню справки
        help_menu(update, context)

def departments_info(update: Update, context: CallbackContext):
```

```
    # Отправляем информацию о кафедрах
        update.message.reply_text('Кафедра информационных и управляющих систем\n\nТелефон: +7 (4162) 23-46-74\nEmail: ciius@amursu.ru\n\n\nКафедра информационной безопасности\n\nТелефон: +7 (4162) 23-46-97\nEmail: cib@amursu.ru\n\n\nКафедра математического анализа и моделирования\n\nТелефон: +7 (4162) 23-46-76\nEmail: cmaim@amursu.ru\n\n\nКафедра Общей математики и информатики\n\nТелефон: +7 (4162) 23-46-77\nEmail: comim@amursu.ru\n\n')

  # Отправляем кнопки меню справки
    help_menu(update, context)

def decan_info(update: Update, context: CallbackContext):
    # Отправляем информацию о кафедрах
        update.message.reply_text('Декан факультета математики и информатики кандидат технических наук,  доцент\nСамохвалова Светлана Геннадьевна\n\nТелефон: +7 (4162) 23-46-80\nEmail: 'sgs@amursu.ru')

    # Отправляем кнопки меню справки
    help_menu(update, context)

def fmi_media_info(update: Update, context: CallbackContext):
    # Отправляем информацию о ФМиИ Медиа
        update.message.reply_text('В медиасфере факультет представлен телеграм-каналом и группой ВКонтакте. Публикуются вся ключевая информация о факультете и мероприятиях, в которых участвуют студенты и преподаватели. '\n\nTelegram:  t.me/fmi_amsu\nVK:  https://vk.com/fmi_amsu\n\n\nОтветственный: Гридина Елизавета Григорьевна\n@gribazil0')
```

```
    # Отправляем кнопки меню справки
    help_menu(update, context)


def back_to_main_menu(update: Update, context: CallbackContext):
    # Удаляем ключ 'in_info_menu' из context.user_data
    if 'in_info_menu' in context.user_data:
        del context.user_data['in_info_menu']

        # Отправляем главное меню с кнопками пользователю
        send_main_menu(update, context)

        # Удалить сообщения
        clear_chat(update, context)


def info_menu(update: Update, context: CallbackContext):
    if 'in_info_menu' not in context.user_data:
        # Пользователь только что вошел в меню информации о...
        context.user_data['in_info_menu'] = True

    update.message.reply_text('Факультет математики и информатики был образован в Амурском государственном университете по приказу ректора Б. А. Виноградова 11 января 1995 года.\n3а годы работы факультет выпустил более 1700 специалистов, из них 273 получили дипломы с отличием, более двадцати выпускников получили ученую степень кандидата наук, один выпускник стал доктором физ.-мат. наук, более 15 остались работать в Alma-mater. Высокий уровень подготовки выпускников позво-
```

ляет успешно получать образование, обучаясь в магистратуре различных университетов нашей страны, а также и за рубежом и конкурировать на рынке труда различных городов не только России. Наши выпускники работают в организациях Амурской области (ПАО Росбанк, Главное управление МЧС России по Амурской области, АО ДРСК, ФГБОУ ВО "АмГУ", ПАО Сбербанк и др.) и в других регионах России, а также в сфере информационных технологий в Германии, Новой Зеландии, Китае, США, Австрии.')

Отправляем меню о факультете...

keyboard = [['Математического анализа и моделирования'], ['Информационных и управляющих систем'], ['Общей математики и информатики'], ['Информационной безопасности'], ['Назад']]

reply_markup = ReplyKeyboardMarkup(keyboard, one_time_keyboard=True)

update.message.reply_text('Узнайте о кафедре:', reply_markup=reply_markup)

Удалить сообщения

clear_chat(update, context)

def maim_info(update: Update, context: CallbackContext):

Отправляем информацию о МАиМ

update.message.reply_text('Кафедра математического анализа и моделирования основана 28 мая 1993 года.\n\nВ настоящее время на кафедре осуществляется подготовка бакалавров по направлению 01.03.02 – Прикладная математика и информатика и магистров по направлению 01.04.02 – Прикладная математика и информатика.\n\nЗа время существования кафедры подготовлено более 450 молодых специалистов. Выпускники кафедры работают на предприятиях и в учреждениях области, региона и страны в целом, обучаются в аспирантуре, занимаются научной деятельностью.\n\nКоллектив кафедры представляет собой сплав опытных и молодых преподавателей и ученых. Здесь работали и работают выпускники ведущих университетов страны: МГУ им. М.В. Ломоносова, НГУ (г. Новосибирск), ИГУ (г. Иркутск),

ГГУ (г. Нижний Новгород), ТГУ (г. Ташкент). Кафедра пополняется молодыми сотрудниками – своими вчерашними студентами: ассистентами, магистрантами, аспирантами. Многие бывшие выпускники уже защитили диссертации и стали ведущими преподавателями кафедры.\n\nСтуденты активно участвуют в учебной и научной деятельности кафедры. Ежегодно проводятся студенческие научные конференции, математические олимпиады, где наши студенты регулярно становятся победителями или занимают призовые места.\n\nВ настоящее время на кафедре работают 6 преподавателей. Пять преподавателей имеют ученое звание доцента и ученую степень кандидата или доктора наук.')

Отправляем кнопки второго меню

info_menu(update, context)

def iius_info(update: Update, context: CallbackContext):

Отправляем информацию об ИиУС

update.message.reply_text('14 февраля 1992 года слиянием двух кафедр – прикладной математики и КУЭС была организована кафедра автоматизированных систем обработки информации и управления. В период с 1997 по 1998 гг. кафедрой руководил А.Д. Плутенко. В 1998 году А.Д. Плутенко поступил в докторантуру, затем после отзыва из докторантуры был избран ректором АмГУ. С 1998 года заведующим кафедрой стал кандидат технических наук, доцент А.В. Бушманов.\n\nВ настоящее время кафедра выпускает бакалавров по следующим направлениям подготовки:\n09.03.01 «Информатика и вычислительная техника», профиль «Информатика и вычислительная техника»\n09.03.02 «Информационные системы и технологии», профиль «Информационные системы и технологии»\n09.03.04 «Программная инженерия», профиль «Программная инженерия»\n10.03.01 «Информационная безопасность», профиль «Безопасность автоматизированных систем (по отрасли или в сфере профессиональной деятельности)»\n\nКафедра обладает серьезным научным потенциалом. В ее составе три доктора наук, три профессора, 6 кандидатов наук, 8 доцентов.\n\nЗа последние пять лет профессорско-преподавательским составом кафедры написано более двухсот научных и научно-методических работ, множество статей и тезисов.')

```
    # Отправляем кнопки второго меню
    info_menu(update, context)


def omii_info(update: Update, context: CallbackContext):
    # Отправляем информацию об ОМиИ
    update.message.reply_text('История создания кафедры начинается с октября 1961г., когда приказом министерства образования Высшей школы был образован Благовещенский общетехнический факультет Хабаровского политехнического института.\n\nС февраля 2000г. кафедра «Высшей математики» была реорганизована в кафедру «Общая математика и информатика».\n\nС 01.09.2014г. и по настоящее время заведующей кафедрой является Юрьева Татьяна Александровна, кандидат педагогических наук, доцент.')

    # Отправляем кнопки второго меню
    info_menu(update, context)


    def ib_info(update: Update, context: CallbackContext):
    # Отправляем информацию об ОМиИ
    update.message.reply_text('В соответствии с решением Ученого совета Амурского государственного университета с 16.05.2019 в структуре факультета математики и информатики создана кафедра информационной безопасности.\nПервым заведующим была назначена Самохвалова Светлана Геннадьевна. \n\nС 01.09.2022 исполнение обязанностей заведующего кафедрой возложены на Никифорову Ларису Владимировну.')

    # Отправляем кнопки второго меню
    info_menu(update, context)
```

```
def handle_message(update: Update, context: CallbackContext):
    if update.message:
        # Отправляем сообщение об ошибке, если получено неподдерживаемое сообщение
        update.message.reply_text('Извините, я понимаю только кнопки.')
        back_to_main_menu(update, context)

  # Удалить сообщения
        clear_chat(update, context)

def clear_chat(update: Update, context: CallbackContext):
    # Удаляем предыдущие сообщения
    chat_id = update.effective_chat.id
    last_message_id = update.message.message_id

    # Удаление сообщений, начиная со второго
    for message_id in range(last_message_id - 1, 0, -1):
        context.bot.delete_message(chat_id, message_id)

    # Удаление текущего сообщения
     context.bot.delete_message(chat_id, last_message_id)

def send_main_menu(update: Update, context: CallbackContext):
    # Отправляем главное меню с кнопками пользователю
```

```
    keyboard = [['О факультете'], ['Расписание'], ['Справка']]
    reply_markup = ReplyKeyboardMarkup(keyboard, one_time_keyboard=True)
    update.message.reply_text('Выберите пункт меню:', reply_markup=reply_markup)

def main():
    updater = Updater(TOKEN, use_context=True)
    dp = updater.dispatcher

    # Добавляем обработчик команды /start
    dp.add_handler(CommandHandler("start", start))

    # Добавляем обработчик нажатия на кнопки главного меню
    dp.add_handler(MessageHandler(Filters.regex('^О факультете$'), info_menu))
    dp.add_handler(MessageHandler(Filters.regex('^Расписание$'), open_timetable))
    dp.add_handler(MessageHandler(Filters.regex('^Справка$'), help_menu))

    # Добавляем обработчики нажатий на кнопки второго меню
    dp.add_handler(MessageHandler(Filters.regex('^Математического анализа и моделирования$'), maim_info))
    dp.add_handler(MessageHandler(Filters.regex('^Информационных и управляющих систем$'), iius_info))
    dp.add_handler(MessageHandler(Filters.regex('^Общей математики и информатики$'), omii_info))
    dp.add_handler(MessageHandler(Filters.regex('^Информационной безопасности$'), ib_info))
    dp.add_handler(MessageHandler(Filters.regex('^Назад$'), back_to_main_menu))
```

```
        dp.add_handler(MessageHandler(Filters.regex('^Деканат$'), deanery_info))
        dp.add_handler(MessageHandler(Filters.regex('^Кафедры$'), departments_info))
        dp.add_handler(MessageHandler(Filters.regex('^Декан$'), decan_info))
        dp.add_handler(MessageHandler(Filters.regex('^ФМиИ Медиа$'), fmi_media_info))

        # Добавляем обработчик текстовых сообщений
        dp.add_handler(MessageHandler(Filters.text, handle_message))

        schedule.every().day.at("08:00").do(get_weather)
        schedule.every().day.at("15:00").do(get_weather)
        schedule.every().day.at("19:00").do(get_weather)
        updater.start_polling()

        while True:
            # Запускаем планировщик задач
            schedule.run_pending()
            time.sleep(1)

if __name__ == '__main__':
    main()
```