

Минобрнауки России
Юго-Западный государственный университет

Кафедра программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ПО ПРОГРАММЕ БАКАЛАВРИАТА

09.03.04 Программная инженерия

(код, наименование ОПОП ВО: направление подготовки, направленность (профиль))

«Разработка программно-информационных систем»

**Бизнес-проект «Программно-информационная система конструирования
чат-ботов для Telegram». Разработка модуля конструирования.**

(название темы)

Дипломный проект

(вид ВКР: дипломная работа или дипломный проект)

Автор ВКР

Н.А. Чернов

(подпись, дата)

(инициалы, фамилия)

Группа ПО-016

Руководитель ВКР

Е.А. Петрик

(подпись, дата)

(инициалы, фамилия)

Нормоконтроль

А. А. Чаплыгин

(подпись, дата)

(инициалы, фамилия)

ВКР допущена к защите:

Заведующий кафедрой

А. В. Малышев

(подпись, дата)

(инициалы, фамилия)

Курск 2024 г.

Минобрнауки России
Юго-Западный государственный университет

Кафедра программной инженерии

УТВЕРЖДАЮ:
Заведующий кафедрой

(подпись, инициалы, фамилия)

«_____» _____ 20____ г.

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ ПО
ПРОГРАММЕ БАКАЛАВРИАТА**

Студента Чернов Н.А., шифр 20-06-0037, группа ПО-016

1. Тема «Бизнес-проект «Программно-информационная система конструирования чат-ботов для Telegram».Разработка модуля конструирования.» утверждена приказом ректора ЮЗГУ от «04» апреля 2024 г. № 1620-с.
2. Срок предоставления работы к защите «11» июня 2024 г.
3. Общая часть ВКР:
4. Индивидуальная часть ВКР:
 - 4.1. Исходные данные:
 - 4.1.1. Перечень решаемых задач
 - 1) разработать модель данных программной системы; определить варианты использования программной системы; разработать требования к программной системе;
 - 2) осуществить проектирование программной системы; разработать архитектуру программной системы; разработать пользовательский интерфейс для создания и управления чат-ботом;
 - 3) зарегистрировать бота в Telegram с помощью @BotFather и получить токен для доступа к API

4) провести тестирование модуля межпользовательского взаимодействия программной системы; провести системное тестирование компонентов программной системы.

4.1.2. Входные данные и требуемые результаты для программы: 1) Входными данными для программной системы являются: команды пользователей; сообщения пользователей; данные для интеграции с внешними сервисами.

2) Выходными данными для программной системы являются: ответы на команды; ответы на текстовые сообщения; модерация. 4.2. Содержание работы (по разделам): 4.2.1. Введение 4.2.2. Резюме стартап-проекта 4.2.3. Анализ предметной области

Лист 1. Сведения о ВКРБ

Лист 2. Цели и задачи разработки

Лист 3. Диаграммы вариантов использования

Лист 4. Диаграмма развертывания приложения

Лист 5. Заключение

Руководитель ВКР

Е.А. Петрик

(подпись, дата)

(инициалы, фамилия)

Задание принял к исполнению

Н.А. Чернов

(подпись, дата)

(инициалы, фамилия)

РЕФЕРАТ

Объем работы равен 103 страницам. Работа содержит 27 иллюстраций, 5 таблиц, 37 библиографических источников и 7 листов графического материала. Количество приложений – 2. Графический материал представлен в приложении А. Фрагменты исходного кода представлены в приложении Б.

Перечень ключевых слов: программно-информационная система, чат-бот, запрос, мессенджер, представление, модель данных, модуль, метод, пользователь, сообщение.

Объектом разработки является программно-информационная система для конструирования чат-ботов для Telegram.

Целью выпускной квалификационной работы являлась разработка модуля управления шаблонами.

При проектировании и создании программной системы применялась парадигма объектно-ориентированного программирования.

В процессе создания программной системы был разработан программный комплекс, разработаны методы и реализованы отправка, получение и обработка данных.

Данная программно-информационная система находится на стадии внедрения. Её можно использовать для помощи в группах и каналах в мессенджере Telegram.

ABSTRACT

The volume of work is 103 pages. The work contains 27 illustrations, 5 tables, 37 bibliographic sources and 7 sheets of graphic material. The number of applications is 2. The graphic material is presented in annex A. The fragment of the source code is provided in annex B.

List of keywords: software information system, chat bot, request, messenger, presentation, data model, classes, method, user, message.

The object of development is a web application of a social media platform for placing digital images.

The purpose of the final qualifying work was to develop a module for inter-user interaction for this web application.

When designing and creating a software system, the object-oriented programming paradigm was used.

In the process of creating a software system, a software package was developed, methods were developed and the sending, receiving and processing of data was implemented.

This software information system is at the implementation stage. It can be used to help in groups and channels in the Telegram messenger.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	9
РЕЗЮМЕ СТАРТАП-ПРОЕКТА	11
1 Анализ предметной области	14
2 Техническое задание	20
2.1 Основание для разработки	20
2.2 Назначение разработки	20
2.3 Требования к программной системе	20
2.3.1 Требования к программной системе	20
2.3.2 Функциональные требования к программной системе	21
2.3.2.1 Вариант использования «Попытка написать сообщение содержащие нецензурную лексику»	22
2.3.2.2 Вариант использования «Получение ежедневного прогноза погоды в чате»	22
2.3.2.3 Вариант использования «Получение прогноза погоды в чате в данный момент»	23
2.3.2.4 Вариант использования «Попытка написать сообщение содержащие нецензурную лексику»	23
2.3.3 Нефункциональные требования к программной системе	24
2.3.3.1 Требования к надежности	24
2.3.3.2 Требования к безопасности	24
2.3.3.3 Требования к программному обеспечению	24
2.4 Требования к оформлению документации	24
3 Технический проект	26
3.1 Общая характеристика организации решения задачи	26
3.2 Проектирование архитектуры программной системы	27
3.2.1 Компоненты программной системы	27
3.2.2 Язык программирования Python	28
3.2.3 OS	29
3.2.4 pyTelegramBotAPI	31

3.2.5	schedule	33
3.2.6	JSON	34
3.2.7	pathlib	36
3.3	Архитектура программной системы	38
3.4	Компоненты контроллера	38
3.5	Компоненты представления	39
3.6	Проектирование пользовательского интерфейса программной системы	40
3.6.1	Основные цели и задачи	40
3.6.2	Общий подход к проектированию интерфейса	40
3.6.3	Основные компоненты интерфейса	41
4	Рабочий проект	42
4.1	Спецификация компонентов и классов программной системы	42
4.1.1	Спецификация модуля bot_main.py	42
4.1.2	Спецификация модуля config.py	42
4.1.3	Спецификация модуля file_buffer.py	43
4.1.4	Спецификация модуля message_handlers.py	43
4.1.5	Спецификация модуля weather_handler.py	44
4.2	Тестирование программной системы	45
4.2.1	Системное тестирование программной системы	45
ЗАКЛЮЧЕНИЕ		75
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ		77
ПРИЛОЖЕНИЕ А Представление графического материала		81
ПРИЛОЖЕНИЕ Б Фрагменты исходного кода программы		89
На отдельных листах (CD-RW в прикрепленном конверте)		103
Сведения о ВКРБ (Графический материал / Сведения о ВКРБ.png)		Лист 1
Цель и задачи разработки (Графический материал / Цель и задачи разработки.png)		Лист 2

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ИС – информационная система.

ИТ – информационные технологии. ПО – программное обеспечение.

РП – рабочий проект.

ТЗ – техническое задание.

ТП – технический проект.

IDE (Integrated Development Environment) – интегрированная среда разработки программного обеспечения.

Telegram – мессенджер.

API (Application Programming Interface) – программный интерфейс приложения

ВВЕДЕНИЕ

Для упрощения или автоматизации общения с клиентами создают специальные программы - чат-боты. Чат-бот это программный консультант, основная задача которого – выполнять за вас однотипные задачи, такие как ответить клиентам по статусу их заказа, оказание технической поддержки, рекламная рассылка и другое. Со стороны пользователей это обычный чат, но на самом деле пользователь общается со специальной программой.

Цель настоящей работы – создание чат-бота, позволяющего помочь пользователям в решении их однотипных задач в группе или же канале Telegram. В процессе нужно использовать современные технологии, которые обеспечивают качественную и масштабируемую работу приложения, а также удовлетворяют потребности пользователей. Для достижения поставленной цели необходимо решить *следующие задачи*:

- провести анализ предметной области;
- - зарегистрировать бота в Telegram с помощью @BotFather и получить токен для доступа к API;
- - осуществить проектирование программной системы; разработать архитектуру программной системы;
- провести тестирование работы программно-информационной системы.

Структура и объем работы. Отчет состоит из введения, резюме стартап-проекта, 4 разделов основной части, заключения, списка использованных источников, 2 приложений. Текст выпускной квалификационной работы равен 103 страницам.

Во введении цель работы, поставлены задачи разработки, описана структура работы, приведено краткое содержание каждого из разделов.

В резюме стартап-проекта содержится основная информация о стартап-проекте: название, цели и стратегия, уникальность продукта, результаты, риски и перспективы проекта.

В первом разделе на стадии описания технической характеристики предметной области приводится сбор информации о целевой аудитории, ее интересах и потребностях.

Во втором разделе на стадии технического задания приводятся требования к разрабатываемому продукту.

В третьем разделе на стадии технического проектирования представлены проектные решения для системы.

В четвертом разделе приводится список классов, их атрибутов и методов, использованных при разработке чат-бота, производится тестирование разработанной системы.

В заключении излагаются основные результаты работы, полученные в ходе разработки.

В приложении А представлен графический материал. В приложении Б представлены фрагменты исходного кода.

РЕЗЮМЕ СТАРТАП-ПРОЕКТА

Название: «Бизнес-проект «Программно-информационная система для конструирования чат-ботов для Telegram»». Проект представляет собой чат-бота в Telegram, предназначенного для автоматизации различных задач, таких как прогноз погоды, модерация контента, ответы на часто задаваемые вопросы и другие. Были проанализированы современные аналоги и выявлены *основные проблемы*, с которыми сталкиваются пользователи медиаплатформ:

1. Сложность настройки и использования: Многие платформы требуют значительных усилий для настройки и конфигурации ботов, что делает их трудными для новичков.
2. Ограниченные возможности кастомизации: Существующие решения часто предлагают ограниченные возможности для настройки и кастомизации ботов под специфические потребности.
3. Высокая стоимость: Множество платформ требуют значительных финансовых затрат для доступа к продвинутым функциям и интеграциям.
4. Ограниченная интеграция с внешними сервисами: Поддержка интеграции с внешними сервисами и API может быть ограниченной или сложной в настройке.
5. Недостаточная поддержка и документация: Пользователи часто сталкиваются с недостаточной поддержкой и неполной документацией, что усложняет решение возникающих проблем.

Многофункциональный чат-бот призван решить эти проблемы и стать удобным пользователям.

Актуальность данного чат-бота связана с высоким спросом на автоматизированные решения, которые могут помочь в управлении и взаимодействии с участниками. Цель. Разработка чат-бота для Telegram, который будет помогать пользователям решать однотипные задачи, автоматизировать рутинные процессы и улучшать взаимодействие внутри сообществ.

Отличительные особенности приложения по сравнению с существующими решениями и технологиями:

1. Автоматизация задач. Чат-бот позволяет автоматизировать рутинные задачи, такие как ответы на часто задаваемые вопросы, модерация контента и управление задачами.
2. Централизованная обратная связь. Возможность получения и обработки обратной связи от пользователей в одном месте.
3. Эффективное управление информацией. Бот помогает структурировать и организовывать информацию, что облегчает её поиск и обработку.

В разработанном приложении реализованы следующие возможности: автоматические ответы на часто задаваемые вопросы, модерация и фильтрация контента, получение данных о погоде в своем городе.

При разработке чат-бота были решены следующие задачи: создание модулей для автоматических ответов на часто задаваемые вопросы, разработка системы модерации и фильтрации контента, реализация возможностей интеграции с популярными внешними сервисами.

Идея создания проекта возникла в сентябре 2023 года, тогда же и началась разработка. С февраля 2024 года началось тестирование чат-бота путём предоставления доступа онлайн пользователям, с июля 2024 года планируется расширение аудитории тестирования путём развёртывания приложения на общедоступном сервере для тестирования работы приложения под нагрузкой, на этом этапе принимать участие в тестировании смогут пользователи по всей стране.

Для защиты исключительных прав и правовой охраны будет осуществлена регистрация программного средства. Самыми значительными и высоковероятными рисками для проекта являются рост издержек и инфляция.

Перспективой дальнейшей разработки программной системы является наращивание аудитории, увеличение количества созданных ботов и расширение функционала посредством обновлений.

Бизнес-цели:

1. Увеличить число пользователей, которые воспользовались ботом: 1000 пользователей через год. Для достижения этой цели можно использовать различные методы привлечения пользователей, например, рекламные кампании в социальных сетях, поисковую оптимизацию и т.д.

2. Добавить не менее 5 новых ключевых функций в течение первого года на основе пользовательских отзывов и анализа рынка. Регулярно собирать обратную связь от пользователей, анализировать данные о взаимодействии с ботом, следить за новыми тенденциями и технологиями.

3. Достигнуть уровня удовлетворенности пользователей не менее 85 процентов по результатам регулярных опросов. Постоянно улучшать качество чат-бота, оперативно реагировать на жалобы и претензии, обеспечивать высокий уровень поддержки пользователей

1 Анализ предметной области

С ростом популярности мессенджеров, таких как Telegram, все больше команд и сообществ используют их для координации и общения. Это привело к необходимости автоматизации рутинных и однотипных задач, что позволяет снизить нагрузку на администраторов и повысить общую эффективность взаимодействия. Telegram-боты предоставляют отличную платформу для реализации этих целей благодаря их гибкости и широким возможностям интеграции.

Автоматизация однотипных задач включает в себя выполнение рутинных, повторяющихся действий без участия человека. В контексте Telegram-ботов это может включать:

- Ответы на часто задаваемые вопросы (FAQ): Бот могут автоматически отвечать на распространенные вопросы, экономя время администраторов.
- Прогноз погоды: Бот может определять прогноз погоды в любой точке мира.
- Модерация: Бот могут автоматически фильтровать и удалять нежелательный контент, предупреждать или блокировать нарушителей.
- Интеграция с внешними сервисами: Получение и обработка данных из различных источников для выполнения задач.

В ходе анализа целевой аудитории были составлены обобщающие модели персонажей.

На рисунке 1.1 изображена модель персонажа, пользующегося чатом.

Модель персонажа

Имя: Дмитрий Сидоров

Деятельность: Администратор

Telegram канала

Возраст: 28 лет



Мотивации и цели:

- давать людям информацию, в которой они нуждаются;
- стремление быть лучшим в своем деле;
- поиск новых способов получения информации.

Рисунок 1.1 – Модель персонажа, администратора Telegram канала

Ниже описана Journey Map пользователя по использованию для публикации цифрового искусства.

1. Определение потребности. Дмитрий, администратор популярного Telegram-канала, сталкивается с несколькими проблемами. Ему необходимо предоставить участникам точный прогноз погоды, автоматически удалять нецензурные сообщения и ссылки, а также отвечать на часто задаваемые вопросы. Осознавая, что выполнение этих задач вручную занимает много времени и сил, Дмитрий решает найти чат-бота, который сможет автоматизировать эти процессы.

2. Поиск вариантов. Дмитрий начинает искать варианты ботов для модерирования своего канала. Он может использовать поисковые системы, спросить у своих коллег, где они ищут модерацию для своих каналов или найти информацию в социальных сетях.

3. Открытие существующих сайтов. Дмитрий переходит на несколько сайтов для поиска нужного чат-бота, чтобы ознакомиться с возможностями каждого из них.

4. Окончательный выбор. После ознакомления с несколькими сайтами, Дмитрий принимает окончательное решение, какого чат-бота ему добавить к себе в канал. Он может выбрать чат-бота, который соответствует его целям и

потребностям как администратора. После этого он добавляет бота к себе на канал и пользуется.

На рисунке 1.2 изображена модель персонажа, интересующегося изысканной готовкой.

Модель персонажа

Имя: Анна Шарапова
Профессия: Шеф-повар
Возраст: 35 года

Увлечения: интересуется изысканной готовкой.



Рисунок 1.2 – Модель персонажа, интересующегося изысканной готовкой

1. Определение потребности. Анна, управляющая популярной кулинарной группой в Telegram, заметила, что участники часто задают одни и те же вопросы о рецептах, ингредиентах и технике приготовления. Кроме того, ей нужно поддерживать чистоту в группе, удаляя рекламные ссылки и неуместные сообщения. Анна понимает, что ей нужен чат-бот, который сможет автоматизировать ответы на часто задаваемые вопросы, предоставлять рецепты по запросу и модерировать контент группы[5].

2. Поиск вариантов. Анна начинает исследовать возможные варианты чат-ботов для своей группы. Она читает статьи и обзоры на специализированных сайтах, посещает форумы, посвященные кулинарным сообществам, и задает вопросы коллегам-администраторам других групп. Она также просматривает отзывы пользователей на различных платформах, чтобы понять, какие боты хорошо справляются с поставленными задачами.

3. Открытие существующих сайтов. Анна посещает несколько сайтов, предлагающих Telegram-ботов, и тщательно изучает их функционал.

4. Окончательный выбор. После ознакомления с несколькими сайтами, Анна принимает окончательное решение, каким ботом она будет пользоваться в своем канале. Она добавляет его к себе в группу и настраивает его под свои нужды. Бот начинает автоматически отвечать на вопросы участников, предоставлять рецепты и удалять неуместный контент. Благодаря этому Анна освобождает значительное количество времени, которое она теперь может посвятить созданию нового контента и взаимодействию с участниками группы.

В ходе анализа предметной области были изучены конкурентные решения. Выбор конкретного бота зависит от конкретных потребностей и приоритетов администратора канала или группы. Данный чат-бот может занять свою нишу, предлагая уникальное сочетание функциональности, простоты использования и адаптируемости под конкретные задачи пользователей.

Были выделены несколько конкурентов, которые предоставляют сходную функциональность. Они представлены ниже:

1. ManyBot.
2. Chatfuel.
3. Dialogflow.

Одним из основных конкурентов данного чат-бота является конструктор ManyBot, который позволяет создавать ботов без навыков программирования[6]. Он предоставляет функции для автоматических ответов на часто задаваемые вопросы, отправки уведомлений, создания опросов и управления контентом. Недостатки ManyBot описаны ниже:

1. Ограниченный набор команд, бот предлагает базовый набор команд и функций, что может не удовлетворять потребности пользователей, которым требуется более продвинутый функционал.
2. Отсутствие интеграции с внешними сервисами: Для пользователей, которым нужна интеграция с конкретными сервисами (например, метеорологическими API для прогноза погоды), ManyBot может быть недостаточно гибким.

3. Отсутствие автоматической модерации: ManyBot не предоставляет продвинутых инструментов для автоматической модерации, таких как анализ контента с помощью AI для выявления спама или оскорбительного контента.

Следующим конкурентом является Chatfuel – это мощная платформа для создания ботов, которая поддерживает интеграцию с Telegram. Chatfuel предоставляет множество функций, включая автоматические ответы, напоминания, опросы и модерацию контента. Платформа также предлагает интеграцию с различными внешними сервисами и API[7]. Недостатки Chatfuel описаны ниже:

1. Несмотря на наличие графического интерфейса, Chatfuel может быть сложным для новичков. Понимание логики построения бота и настройки всех необходимых блоков требует времени и усилий.

2. Хотя Chatfuel позволяет использовать JSON API, возможности для продвинутой кастомизации и программирования могут быть ограничены по сравнению с другими платформами, которые предоставляют полный доступ к исходному коду ботов.

Другим конкурентом является Dialogflow от Google позволяет создавать интеллектуальных чат-ботов с использованием машинного обучения и обработки естественного языка. Он поддерживает интеграцию с Telegram и предоставляет инструменты для создания сложных сценариев взаимодействия, автоматических ответов и модерации контента. Недостатки Dialogflow описаны ниже:

1. Использование Dialogflow часто требует интеграции с другими сервисами Google Cloud, что может привести к высоким затратам, особенно для крупных проектов с высокой нагрузкой[8].

2. Интерфейс Dialogflow может быть менее интуитивным по сравнению с другими платформами. Некоторые пользователи отмечают, что настройка агентов и управление ими может быть сложной задачей из-за недостаточно продуманного интерфейса.

3. Процесс тестирования и отладки ботов в Dialogflow может быть сложным и требовать значительных усилий. Найти и исправить ошибки в сложных сценариях взаимодействия может быть непросто.

4. Неожиданные расходы: При активном использовании платформы затраты могут быстро расти, особенно если бот обрабатывает большое количество запросов.

2 Техническое задание

2.1 Основание для разработки

Функциональное назначение разрабатываемой программной системы заключается в предоставлении. Предполагается, что разрабатываемая программная система будет ис- пользоваться широким кругом лиц, занимающихся цифровым искусством в России.

2.2 Назначение разработки

Функциональное назначение разрабатываемой программной системы заключается в автоматизации и оптимизации однотипных задач пользователей в группах и каналах Telegram. команды пользователей; сообщения пользователей; данные для интеграции с внешними сервисами[9].

2.3 Требования к программной системе

2.3.1 Требования к программной системе

Входными данными для программной системы являются:

- id чата;
- id пользователя;
- команды пол;
- приглашение в чат;
- запрос
- запрос на создание ключевого слова;
- запрос на удаление ключевого слова;
- запрос на редактирование ключевого слова;
- запрос на список ключевых слов;
- запрос на создание ключевого слова с уведомлением;
- запрос на удаление ключевого слова с уведомлением ;
- запрос на редактирование ключевого слова с уведомлением ;
- запрос на список ключевых слов с уведомлением ;

Выходными данными для программной системы являются:

- сообщения о успешной обработке команд;
- отправка сообщений по ключевому слову;
- отправка сообщений по ключевому слову с уведомлением;
- список ключевых слов с уведомлением;
- отправка информации о погоде.

2.3.2 Функциональные требования к программной системе

В разрабатываемой программной системе для пользователя должны быть реализованы следующие функции:

1. Старт и помощь.
2. Сохранение сообщения или фото.
3. Сохранение сообщения или фото с уведомлением.
4. Редактирование сообщения или фото.
5. Редактирование сообщения или фото с уведомлением.
6. Удаление сообщения или фото.
7. Удаление сообщения или фото с уведомлением.
8. Просмотр списка ключевых слов.
9. Просмотр списка ключевых слов с уведомлением.
10. Фильтрация запрещенных слов.
11. Ответ на ключевое слово.
12. Просмотр прогноза погоды.
13. Выбор города для просмотра прогноза погоды.
14. Изменение города для получения прогноза погоды.

На рисунках 2.1 представлены функциональные требования к системе в виде диаграммы прецедентов нотации UML.

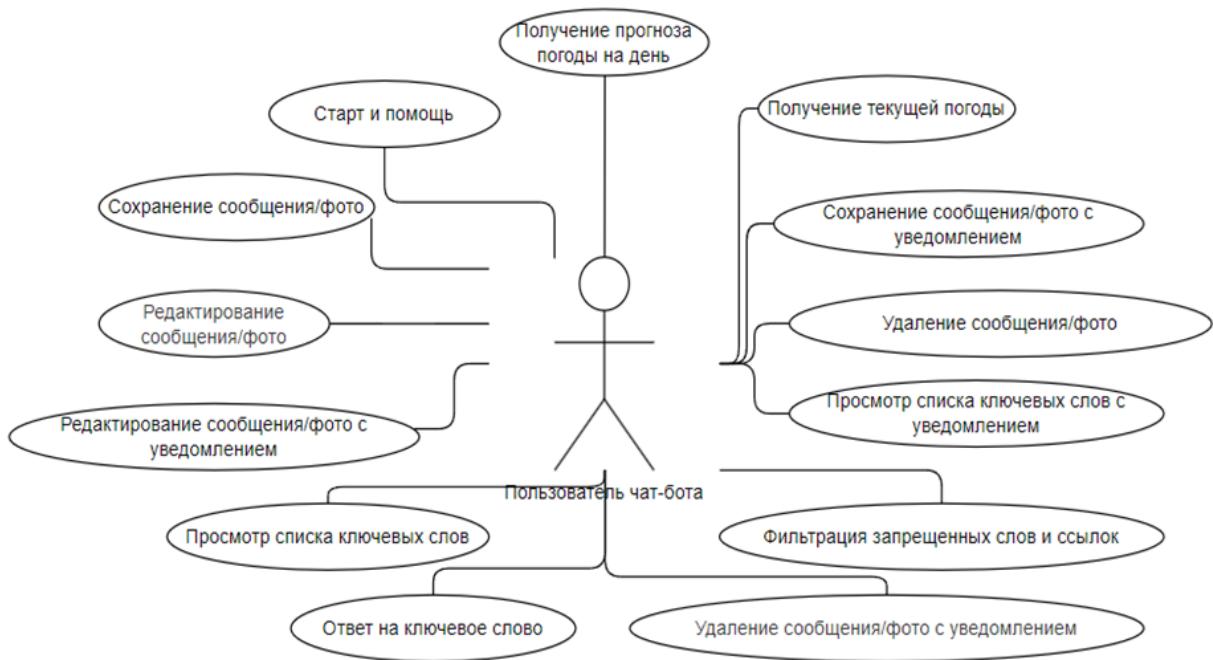


Рисунок 2.1 – Диаграмма вариантов использования

2.3.2.1 Вариант использования «Попытка написать сообщение содержащие нецензурную лексику»

Заинтересованные лица и их требования: пользователь хочет написать сообщение содержащие нецензурную лексику в чате сообщества.

Предусловие: пользователь использует чат в котором присутствует чат бот.

Постусловие: пользователь написал сообщение содержащие нецензурную лексику.

Основной успешный сценарий:

1. Пользователь отправляет сообщение содержащие нецензурную лексику.
2. Сообщение удаляется мгновенно.

2.3.2.2 Вариант использования «Получение ежедневного прогноза погоды в чате»

Заинтересованные лица и их требования: пользователь хочет настроить получение ежедневного прогноза погоды в чате города.

Предусловие: пользователь использует чат в котором присутствует чат-бот.

Постусловие: пользователь хочет получать прогноз погоды по одному существующему городу. Основной успешный сценарий:

1. Пользователь переходит на в чат, котором добавлен бот.
2. Пишет команду «/set_city».
3. Бот отправляет сообщение о успешном срабатывании команды и ожидании информации от пользователя с названием города для предоставления прогноза.
4. Пользователь отправляет название города.
5. Бот сохраняет информацию по данному чату и каждый день в 9:00 отправляет прогноз погоды на день.

2.3.2.3 Вариант использования «Получение прогноза погоды в чате в данный момент»

Заинтересованные лица и их требования: пользователь хочет получить прогноз погоды сейчас.

Предусловие: пользователь использует чат в котором присутствует чат-бот.

Постусловие: сохранена информация о городе.

Основной успешный сценарий:

1. Пользователь пишет команду «/current_weather».
2. Бот отправляет прогноз погоды.

2.3.2.4 Вариант использования «Попытка написать сообщение содержащие нецензурную лексику»

Заинтересованные лица и их требования: пользователь хочет написать сообщение содержащие нецензурную лексику в чате сообщества.

Предусловие: пользователь использует чат в котором присутствует чат-бот.

Постусловие: пользователь написал сообщение содержащие нецензурную лексику.

Основной успешный сценарий:

1. Пользователь отправляет сообщение содержащие нецензурную лексику.
2. Сообщение удаляется мгновенно.

2.3.3 Нефункциональные требования к программной системе

2.3.3.1 Требования к надежности

Система должна работать все время, за исключением плановых технических обслуживаний.

2.3.3.2 Требования к безопасности

Все пользователи должны быть зарегистрированы в Telegram. Доступ к различным функциям чат-бота должен быть основан на ролях и разрешениях в группе или канале.

2.3.3.3 Требования к программному обеспечению

Для реализации программной системы должны быть использованы следующие языки программирования:

Python 3.7 или выше: бот написан на языке программирования Python, поэтому необходимо наличие установленного интерпретатора Python версии 3.7 или выше.

Операционная система: поддерживаются любые ОС, совместимые с Python, такие как Linux, Windows, macOS.

2.4 Требования к оформлению документации

Требования к стадиям разработки программ и программной документации для вычислительных машин, комплексов и систем независимо от их

назначения и области применения, этапам и содержанию работ устанавливаются ГОСТ 19.102-77.

Программная документация должна включать в себя:

1. Техническое задание.
2. Технический проект.
3. Рабочий проект.

3 Технический проект

3.1 Общая характеристика организации решения задачи

Полное наименование системы: программно-информационная система для конструирования чат-ботов для Telegram.

Проект представляет собой телеграм-бота, выполняющего функции сохранения сообщений, связанных с ключевыми словами, и отправки уведомлений о погоде. Вся структура проекта организована в виде набора модулей, каждый из которых отвечает за определенные функциональные блоки. Рассмотрим основные модули и их функциональность.

Модуль управления ботом:

- отвечает за взаимодействие с API Telegram для обработки входящих сообщений от пользователей и отправки ответов.

- реализует логику работы бота, включая обработку команд пользователя и пересылку сообщений другим модулям для дальнейшей обработки.

Модуль сохранения сообщений:

- отвечает за сохранение сообщений, содержащих ключевые слова, в базу данных или файловую систему для последующего анализа.

- обеспечивает функционал поиска и извлечения сохраненных сообщений по заданным ключевым словам.

Модуль анализа текста:

- производит анализ входящих сообщений для выявления ключевых слов или фраз, связанных с интересующими темами.

- может использовать методы обработки естественного языка для улучшения точности выявления ключевых элементов.

Модуль уведомлений о погоде:

- получает данные о погоде из внешних источников (например, открытых API погодных сервисов).

- отправляет уведомления пользователям о текущей или прогнозируемой погоде в заданном местоположении или с использованием указанных параметров.

Модуль управления настройками:

- позволяет пользователям настраивать параметры бота, такие как географическое местоположение для уведомлений о погоде, список ключевых слов для отслеживания и другие параметры.
- обеспечивает интерфейс для изменения настроек и сохранения их в базе данных.

3.2 Проектирование архитектуры программной системы

3.2.1 Компоненты программной системы

Разрабатываемая система является ботом для мессенджера, который предназначен для обработки команд, фильтрации сообщений и управления ключевыми словами.

Для разработки frontend-модуля необходимо использовать язык высокоуровневый язык программирования общего назначения Python.

Для разработки backend-модуля будет использоваться язык программирования высокоуровневый язык программирования общего назначения Python.

Используемая среда разработки – PyCharm Community Edition 2023.2.

В качестве системы управления версиями для командной разработки необходимо использовать Git[21][22].

Диаграмма развёртывания программы представлена на рисунке 3.1.

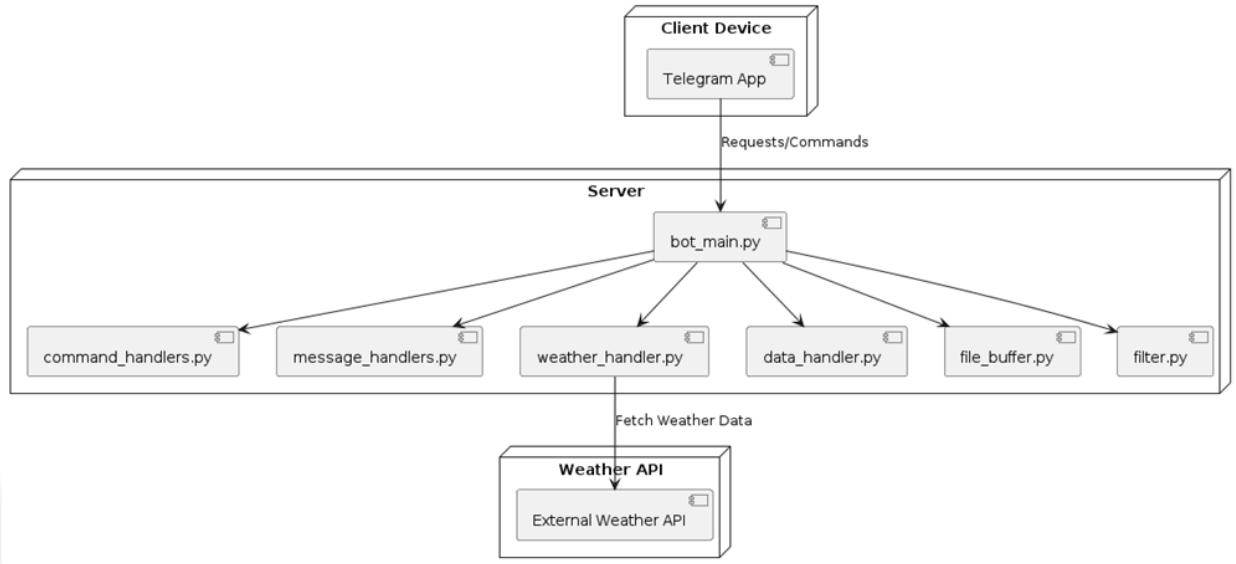


Рисунок 3.1 – Диаграмма развертывания

3.2.2 Язык программирования Python

Выбор языка программирования Python обусловлен его многочисленными преимуществами и подходящими характеристиками для разработки разнообразных приложений:

- Простота и удобство

Python предлагает простой и понятный синтаксис, что делает его привлекательным для начинающих разработчиков и обеспечивает высокую продуктивность при написании кода.

- Богатая экосистема

Python имеет огромное сообщество разработчиков и богатую экосистему библиотек и инструментов, которые позволяют быстро решать разнообразные задачи, начиная от веб-разработки и машинного обучения, и заканчивая научными вычислениями и анализом данных.

- Кроссплатформенность и переносимость

Python поддерживает все основные операционные системы и может быть запущен на различных платформах без изменений в коде, что обеспечивает легкость развертывания приложений.

- Широкое применение

Python применяется в различных областях, включая веб-разработку, научные исследования, разработку игр, автоматизацию задач, анализ данных и многое другое, что делает его универсальным инструментом для разработчиков.

- Активное сообщество и поддержка

Python имеет активное и дружественное сообщество разработчиков, готовых помочь и поддержать новичков, а также обеспечивает доступ к богатой документации, форумам обсуждений и онлайн-курсам для обучения.

В целом, Python является идеальным выбором для разработки различных приложений благодаря своей простоте, гибкости, мощным возможностям и обширной экосистеме.

3.2.3 OS

Библиотека os является одной из стандартных библиотек языка программирования Python и предоставляет набор функций для взаимодействия с операционной системой. Эта библиотека позволяет выполнять широкий спектр задач, связанных с файловой системой, процессами, переменными окружения и другими аспектами операционной системы.

Основные особенности библиотеки os Управление файловой системой: os предоставляет функции для работы с файлами и директориями, такие как создание, удаление, переименование и перемещение файлов и папок. Поддержка операций чтения и записи файлов, а также получение информации о файлах и директориях. Работа с процессами:

Возможность запуска внешних команд и программ с помощью функций os.system и os.exec. Управление процессами, включая создание, завершение и получение информации о процессах. Переменные окружения:

Доступ к переменным окружения через функции os.environ, os.getenv и os.putenv. Возможность установки, изменения и удаления переменных окружения. Работа с путями:

Подмодуль `os.path` предоставляет удобные функции для работы с путями файловой системы, такие как объединение, разделение и нормализация путей. Операции, специфичные для платформы:

`os` предоставляет функции, специфичные для различных операционных систем, что позволяет писать кроссплатформенный код. Преимущества библиотеки `os` Широкий спектр возможностей:

Библиотека предоставляет богатый набор функций для работы с файловой системой, процессами и переменными окружения. Кроссплатформенность:

`os` поддерживает множество операционных систем, включая Windows, macOS и Linux, что позволяет писать кроссплатформенные приложения. Стандартная библиотека:

`os` является частью стандартной библиотеки Python, поэтому не требует установки дополнительных пакетов и всегда доступна. Гибкость:

Библиотека предоставляет функции для выполнения как простых, так и сложных операций, связанных с операционной системой. Недостатки библиотеки `os` Ограниченная безопасность:

Некоторые функции, такие как `os.system`, могут представлять угрозу безопасности, если входные данные не проверяются должным образом, так как они могут быть использованы для выполнения произвольных команд. Сложность использования некоторых функций:

Для новичков некоторые функции библиотеки могут показаться сложными и требовать дополнительных знаний о работе операционных систем. Ограничения специфичные для платформы:

Хотя библиотека поддерживает множество операционных систем, некоторые функции могут вести себя по-разному на разных платформах, что может требовать дополнительных проверок и обработки исключений в коде.

Заключение:

Библиотека `os` является мощным инструментом для взаимодействия с операционной системой из Python-программ. Она предоставляет широкий

набор функций для работы с файловой системой, процессами и переменными окружения, что делает её незаменимой для задач автоматизации и системного программирования. Преимущества библиотеки, такие как кроссплатформенность, гибкость и доступность в стандартной библиотеке Python, значительно перевешивают её недостатки.

3.2.4 pyTelegramBotAPI

Библиотека pyTelegramBotAPI (также известная как TeleBot) является одной из наиболее популярных библиотек для разработки Telegram-ботов на языке программирования Python. Она предоставляет простой и удобный интерфейс для взаимодействия с API Telegram, что делает процесс создания ботов относительно лёгким даже для начинающих разработчиков. Давайте рассмотрим основные аспекты этой библиотеки, её преимущества и недостатки.

Основные особенности pyTelegramBotAPI Простота использования:

pyTelegramBotAPI позволяет быстро и легко начать работу с ботом благодаря интуитивно понятному интерфейсу. Основные функции библиотеки включают создание и отправку сообщений, обработку обновлений (updates), работу с клавиатурами и кнопками, а также управление чатами и пользователями. Асинхронная обработка:

Библиотека поддерживает асинхронный режим работы, что позволяет обрабатывать несколько запросов одновременно, улучшая производительность бота. Поддержка всех типов сообщений:

pyTelegramBotAPI поддерживает работу со всеми типами сообщений, включая текст, фото, видео, документы, стикеры и другие мультимедийные файлы. Обработка команд и текстовых сообщений:

Библиотека предоставляет удобные декораторы для обработки команд и текстовых сообщений, что упрощает организацию кода и улучшает его читаемость. Преимущества pyTelegramBotAPI Легкость освоения:

Библиотека имеет простую и понятную документацию, что позволяет разработчикам быстро разобраться с основными функциями и начать разработку бота. Широкая поддержка:

pyTelegramBotAPI активно поддерживается сообществом и имеет множество примеров и решений на форумах и в репозиториях, таких как GitHub. Гибкость:

Библиотека позволяет легко расширять функционал бота за счет поддержки различных типов сообщений и интеграции с другими библиотеками Python. Асинхронность:

Поддержка асинхронного режима работы позволяет эффективно обрабатывать большое количество запросов и улучшает отзывчивость бота. Недостатки pyTelegramBotAPI Производительность:

В некоторых случаях производительность библиотеки может быть ниже по сравнению с другими решениями, особенно если бот обрабатывает большое количество запросов в режиме реального времени. Зависимость от внешних библиотек:

Для работы в асинхронном режиме библиотека требует установки дополнительных зависимостей, что может усложнить настройку окружения. Ограниченнная гибкость в конфигурации:

Некоторые разработчики могут столкнуться с ограничениями в конфигурации и кастомизации поведения бота, что потребует использования дополнительных библиотек или написания собственного кода. Заключение pyTelegramBotAPI является отличным выбором для создания Telegram-ботов благодаря своей простоте, широким возможностям и активной поддержке сообщества. Она позволяет быстро начать разработку и предоставляет все необходимые инструменты для создания функциональных ботов. Несмотря на некоторые недостатки, такие как потенциальные проблемы с производительностью и зависимость от внешних библиотек, преимущества pyTelegramBotAPI значительно перевешивают эти минусы, делая её предпочтительным выбором для большинства проектов.

Для студентов и начинающих разработчиков руTelegramBotAPI может стать отличным стартом в мире разработки Telegram-ботов, предоставляя удобные инструменты и обширную документацию для быстрого освоения.

3.2.5 schedule

Библиотека `schedule` — это простая и удобная библиотека для планирования и выполнения периодических задач на языке программирования Python. Она широко используется для автоматизации рутинных задач и позволяет легко настроить выполнение функций в определенное время или через определенные интервалы. Рассмотрим основные аспекты этой библиотеки, её преимущества и недостатки.

Основные особенности `schedule` Простота использования:

Библиотека обладает интуитивно понятным синтаксисом, что делает её доступной для начинающих разработчиков. Задачи можно планировать с использованием простых и понятных команд, таких как `every`, `day`, `hour`, `minute`, `second`. Гибкость в планировании задач:

Позволяет задавать сложные графики выполнения задач, такие как выполнение задач в определенные дни недели или месяца, а также в определенное время суток. Поддерживает различные интервалы выполнения задач, от секунд до недель. Легкая интеграция:

`schedule` легко интегрируется в существующие проекты на Python и может работать совместно с другими библиотеками и инструментами. Минимальные зависимости:

Библиотека не требует установки большого количества дополнительных зависимостей, что упрощает её установку и использование. Простота и удобство:

Библиотека очень проста в использовании и настройке. Разработчики могут быстро освоить её синтаксис и начать планировать задачи. Легковесность:

`schedule` занимает мало места и не требует значительных ресурсов для работы, что делает её подходящей для небольших и средних проектов. Минимальные зависимости:

Библиотека не требует сложной установки и настройки дополнительных пакетов, что упрощает её интеграцию в проекты. Гибкость:

Поддерживает широкий спектр сценариев планирования, что позволяет создавать сложные графики выполнения задач. Недостатки `schedule` Ограничения по масштабируемости:

Библиотека не предназначена для работы с большими нагрузками и может оказаться неэффективной для задач, требующих высокой производительности и параллельного выполнения.

Однопоточная работа:

`schedule` работает в однопоточном режиме, что может стать проблемой для приложений, требующих параллельной обработки задач.

Отсутствие встроенного управления ошибками:

Библиотека не предоставляет встроенных средств для обработки ошибок и восстановления после сбоев, что требует дополнительной обработки исключений в коде.

Заключение: Библиотека `schedule` является отличным инструментом для простого и удобного планирования задач в Python. Её простота и интуитивно понятный интерфейс делают её доступной для разработчиков любого уровня, а гибкость в настройке задач позволяет использовать её в широком спектре проектов. Однако для приложений, требующих высокой производительности и параллельной обработки задач, могут потребоваться более сложные решения.

3.2.6 JSON

Выбор JSON для хранения данных

JSON (JavaScript Object Notation) - это легкий формат обмена данными, который легко читается и записывается людьми и машинами. JSON широко

используется в веб-разработке и является популярным выбором для хранения данных благодаря своей простоте и универсальности.

Преимущества использования JSON для хранения данных: Читаемость и простота:

Читаемость: JSON легко читается и понимается как людьми, так и машинами. Это делает его удобным для разработчиков и администраторов. Простота: JSON использует минимальный синтаксис, что упрощает создание и парсинг данных. В отличие от XML, JSON не требует закрывающих тегов и лишних элементов. Универсальность:

Совместимость: JSON является языконезависимым форматом и может использоваться в большинстве языков программирования, таких как JavaScript, Python, Java, C#, PHP и другие. Широкая поддержка: JSON поддерживается многими библиотеками и фреймворками, что облегчает его интеграцию в различные проекты. Легковесность:

JSON занимает меньше места по сравнению с другими форматами хранения данных, такими как XML. Это позволяет уменьшить объем передаваемых данных и повысить производительность приложений. Гибкость:

JSON позволяет легко иерархически структурировать данные, что удобно для хранения сложных объектов и массивов. Возможность вложенностии позволяет эффективно организовывать данные. Недостатки использования JSON для хранения данных: Отсутствие схемы:

В отличие от XML, JSON не поддерживает встроенные схемы для валидации структуры данных. Это может привести к ошибкам, если данные не соответствуют ожидаемому формату. Безопасность:

JSON-данные могут быть уязвимы для атак, таких как JSON-инъекции, если не применяются правильные меры безопасности. Это особенно важно при работе с пользовательским вводом. Ограничения в типах данных:

JSON поддерживает ограниченное количество типов данных: строки, числа, объекты, массивы, логические значения и null. Более сложные типы

данных, такие как даты, необходимо представлять в виде строк или других форматов. Производительность:

Парсинг и сериализация JSON может быть медленнее по сравнению с бинарными форматами данных, такими как Protocol Buffers или MessagePack. Это может быть критично для высоконагруженных систем. Заключение JSON является отличным выбором для хранения данных благодаря своей простоте, читаемости и универсальности. Он идеально подходит для веб-разработки и обмена данными между клиентом и сервером. Однако необходимо учитывать его ограничения, такие как отсутствие встроенной схемы и возможные проблемы с безопасностью. Для приложений с высокими требованиями к производительности и безопасности могут потребоваться дополнительные меры и альтернативные форматы данных.

Таким образом, выбор JSON для хранения данных обоснован в большинстве случаев, особенно когда важны простота и совместимость.

3.2.7 **pathlib**

Библиотека `pathlib` является частью стандартной библиотеки Python и предоставляет объектно-ориентированный интерфейс для работы с файловыми системами. Она была введена в Python 3.4 как более современная и удобная альтернатива модулям `os` и `os.path` для работы с путями файловой системы.

Основные особенности библиотеки `pathlib` Объектно-ориентированный интерфейс:

`pathlib` предоставляет классы для работы с путями, такие как `Path` и `PurePath`, что делает код более читабельным и удобным. Кроссплатформенность:

Библиотека автоматически адаптируется к операционной системе, с которой работает, обеспечивая корректную обработку путей на разных платформах (Windows, macOS, Linux). Удобство работы с путями:

Простое объединение, разделение и изменение путей с помощью перегруженных операторов и методов класса Path. Расширенные возможности работы с файлами и директориями:

Поддержка создания, удаления, перемещения и копирования файлов и директорий. Возможность получения информации о файлах и директориях, такой как размер, время последнего изменения и права доступа. Работа с содержимым файлов: Удобные методы для чтения и записи текстовых и бинарных файлов.

Преимущества библиотеки pathlib

Простота и удобство: Объектно-ориентированный интерфейс делает работу с путями интуитивно понятной и удобной, улучшая читаемость и поддержку кода. Кроссплатформенность:

pathlib автоматически обрабатывает различия между файловыми системами разных операционных систем, что упрощает разработку кроссплатформенных приложений. Широкие возможности:

Библиотека предоставляет богатый набор методов для работы с путями, файлами и директориями, что позволяет выполнять большинство задач, связанных с файловой системой, без необходимости использования дополнительных библиотек. Современный подход:

Включение pathlib в стандартную библиотеку Python отражает современные тенденции в разработке, делая её предпочтительным выбором для новых проектов. Недостатки библиотеки pathlib Переход с os и os.path:

Для разработчиков, привыкших использовать модули os и os.path, переход на pathlib может потребовать некоторого времени и усилий для адаптации. Производительность:

Заключение:

Библиотека pathlib представляет собой мощный и удобный инструмент для работы с файловыми системами в Python. Её объектно-ориентированный интерфейс, кроссплатформенная совместимость и богатый набор возможно-

стей делают её предпочтительным выбором для большинства задач, связанных с обработкой путей, файлов и директорий.

3.3 Архитектура программной системы

На рисунке 3.2 в виде UML-диаграммы показана архитектура программной системы.

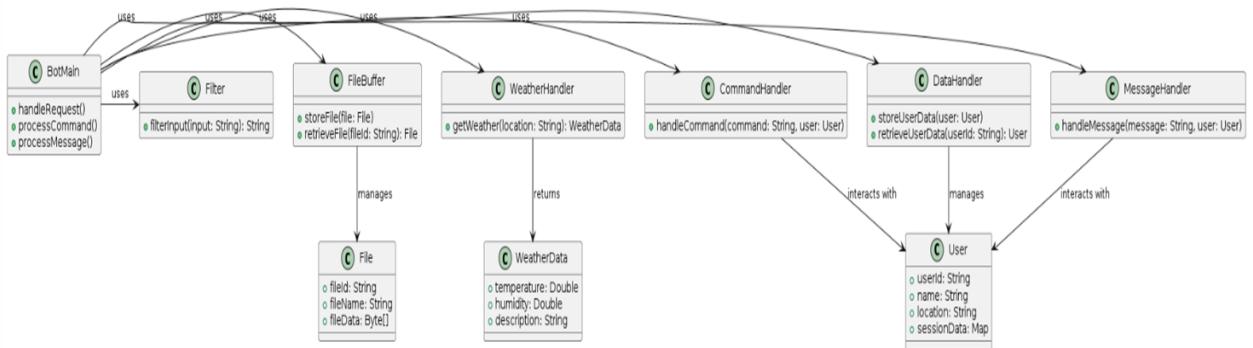


Рисунок 3.2 – Архитектура программной системы

Программно-информационная система содержит следующие компоненты:

1. User Этот модуль представляет пользователя системы. Он хранит информацию о пользователе, такую как идентификатор, имя и местоположение.
2. WeatherData Этот модуль представляет данные о погоде. Он хранит информацию о текущих погодных условиях для определенного местоположения.
3. File Этот модуль представляет загруженные файлы. Он хранит информацию о файле, такую как идентификатор, имя и данные файла.
4. UserPreferences Этот модуль хранит предпочтения пользователя, такие как предпочитаемый язык и настройки уведомлений.
5. UserActivity Этот модуль хранит информацию о действиях пользователя в системе.

3.4 Компоненты контроллера

На рисунке 3.3 представлена UML-диаграмма классов-контроллеров.



Рисунок 3.3 – Архитектура программной системы

1. UserController – Класс отвечает за управление пользователями системы. Он обрабатывает запросы, связанные с регистрацией, аутентификацией и управлением пользовательскими данными.
2. WeatherController – Класс отвечает за получение и обработку данных о погоде. Он обрабатывает запросы, связанные с текущими погодными условиями для определенного местоположения.
3. FileController - Класс отвечает за управление файлами, загруженными пользователями. Он обрабатывает запросы, связанные с загрузкой, скачиванием и удалением файлов.
4. PreferencesController – Класс отвечает за управление предпочтениями пользователя. Он обрабатывает запросы, связанные с обновлением и получением настроек пользователя.
5. ActivityController – Класс отвечает за управление действиями пользователя. Он обрабатывает запросы, связанные с получением и отслеживанием активности пользователя в системе.

3.5 Компоненты представления

На рисунке 3.4 представлена UML-диаграмма классов- компонентов представления

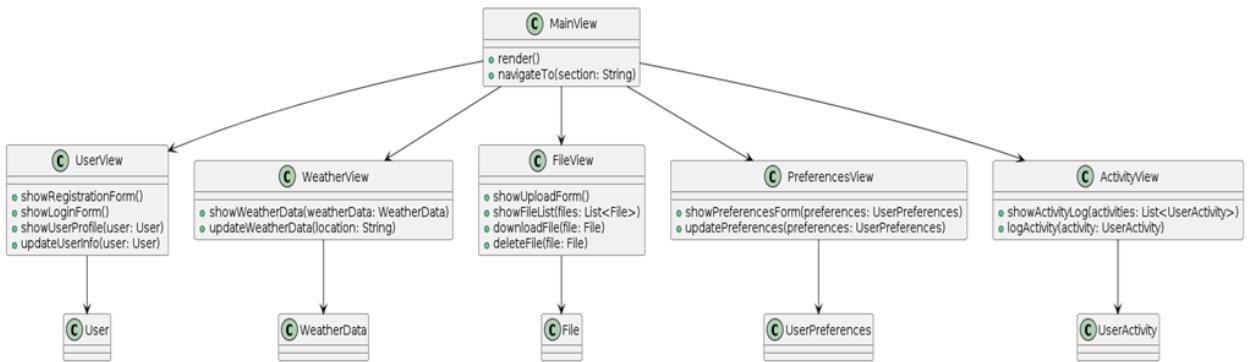


Рисунок 3.4 – Архитектура программной системы

1. **MainView** - Основной компонент интерфейса, который отображает общую структуру и основные разделы приложения.
2. **UserView** – Компонент интерфейса для управления пользователем, включая регистрацию, аутентификацию и обновление информации.
3. **WeatherView** – Компонент интерфейса для отображения данных о погоде.
4. **FileView** – Компонент интерфейса для управления файлами, загруженными пользователем.
5. **PreferencesView** – Компонент интерфейса для управления предпочтениями пользователя.
6. **ActivityView** – Компонент интерфейса для отображения активности

3.6 Проектирование пользовательского интерфейса программной системы

3.6.1 Основные цели и задачи

Цель проектирования пользовательского интерфейса — создать удобный, интуитивно понятный и эффективный интерфейс для пользователей, который позволяет легко взаимодействовать с системой и выполнять необходимые действия.

3.6.2 Общий подход к проектированию интерфейса

Подход к проектированию интерфейса включает:

- Анализ пользовательских требований и задач.

- Создание макетов интерфейса.
- Обеспечение интуитивной навигации и взаимодействия.
- Тестирование интерфейса с реальными пользователями.
- Внесение улучшений на основе отзывов пользователей.

3.6.3 Основные компоненты интерфейса

Для данного проекта, интерфейс будет включать несколько ключевых компонентов, обеспечивающих взаимодействие пользователя с ботом через мессенджер:

1. Главное меню:
 - Команда /start для запуска бота.
 - Команда /help для получения списка доступных команд.
2. Команды для работы с погодой:
 - Команда /set_city для установки города для получения прогноза погоды.
 - Команда /current_weather для получения текущей погоды.
 - Команда /change_city для изменения города.
3. Обработка медиа сообщений: - Возможность отправки текстовых сообщений, фотографий, видео и документов для их сохранения с ключевыми словами.

4 Рабочий проект

4.1 Спецификация компонентов и классов программной системы

Десктоп-приложение разделено на модули, исходя из выполняемых ими функций. Далее приведено описание основных модулей и методов этих модулей. Фрагменты исходного кода приведены в приложении Б.

4.1.1 Спецификация модуля `bot_main.py`

Основной файл для запуска Telegram-бота. Таблица 4.1 – Поля модуля `"bot_main.py"`.

Таблица 4.1 – Поля модуля `"bot_main.py"`

Имя поля	Область видимости	Тип данных	Описание
1	2	3	4
<code>bot_token</code>	<code>global</code>	<code>str</code>	Токен для доступа к API Telegram.
<code>bot</code>	<code>global</code>	объект бота от библиотеки « <code>telebot</code> »	Объект бота, инициализированный с токеном.
<code>data_file_path</code>	<code>global</code>	<code>str</code>	Путь к файлу с данными ключевых слов.

4.1.2 Спецификация модуля `config.py`

Конфигурационный файл для хранения токена API. Поля модуля `config.py` описаны в таблице 4.2

Таблица 4.2 – Поля модуля config.py

Имя поля	Область видимости	Тип данных	Описание
1	2	3	4
API_TOKEN	global	str	Конфигурационный файл для хранения токена API.

4.1.3 Спецификация модуля file_buffer.py

Этот модуль управляет буфером файлов для хранения и управления файлами, полученными через бота. Поля модуля buffer.py описаны в таблице

Таблица 4.3 – file_buffer.py

Имя поля	Область видимости	Тип данных	Описание
1	2	3	4
FILE_BUFFER_DIR	global	str	Директория для хранения файлов.
initialize_buffer()	global	function	Инициализация директории буфера, если она не существует.
save_file()	public	function	Сохранение файла, полученного от пользователя.
get_file_path()	global	function	Получение пути к сохраненному файлу.
remove_file()	global	function	Удаление файла.

4.1.4 Спецификация модуля message_handlers.py

Этот модуль обрабатывает входящие сообщения и сохраняет их, используя ключевые слова. Поля модуля message_handlers.py описаны в таблице 4.4.

Таблица 4.4 – file_buffer.py

Имя поля	Область видимости	Тип данных	Описание
1	2	3	4
chat_keywords	global	list	Загрузка ключевых слов из данных.
register_message_handler	global	function	Регистрация обработчиков сообщений.
handle_media_message	function	function	Обработка мультимедийных сообщений (текст, фото, видео, документ).

4.1.5 Спецификация модуля weather_handler.py

Модуль для работы с API погоды и отправки прогнозов погоды. Поля модуля handler.py описаны в таблице 4.5.

Таблица 4.5 – Спецификация методов модуля «main.py»

Имя поля	Область видимости	Тип данных	Описание
1	2	3	4
WEATHER_API_URL	global	str	URL для API прогноза погоды.
CURRENT_WEATHER_API_URL	global	str	URL для текущей погоды.
API_KEY	global	str	API ключ для доступа к сервису погоды.
WEATHER_FILE	global	function	Файл конфигурации для хранения городов.
load_weather_config()	global	function	Загрузка конфигурации погоды.
save_weather_config(data)	global	function	Сохранение конфигурации погоды.

Продолжение таблицы 4.5

1	2	3	4
get_weather	global	function	Получение прогноза погоды для города.
get_current_weather	global	function	Получение текущей погоды для города.
format_weather_message	local	list	Форматирование сообщения с прогнозом погоды.
format_current-weather_message	local	list	Форматирование сообщения с текущей погодой.
send_daily_weather	global	function	Отправка ежедневного прогноза погоды.
set_weather_city	global	function	Установка города для получения погоды.
get_and_send-current_weather	global	function	Получение и отправка текущей погоды.
register_weather-handlers(bot)	global	function	Регистрация обработчиков команд для погоды.

4.2 Тестирование программной системы

4.2.1 Системное тестирование программной системы

Для проверки работоспособности программной системы было выполнено тестирование разработанного функционала межпользовательского взаимодействия. Результаты проведенного тестирования[25][26] представлены в данном разделе в виде снимков экрана при работе программной системы.

При начала работы необходимо добавить бота в чат который нужно администрировать. На рисунке 4.1 предоставлено окно добавления бота.



Рисунок 4.1 – окно добавления бота

Так же для корректной работы необходимо дать права администратора (рисунок 4.2).

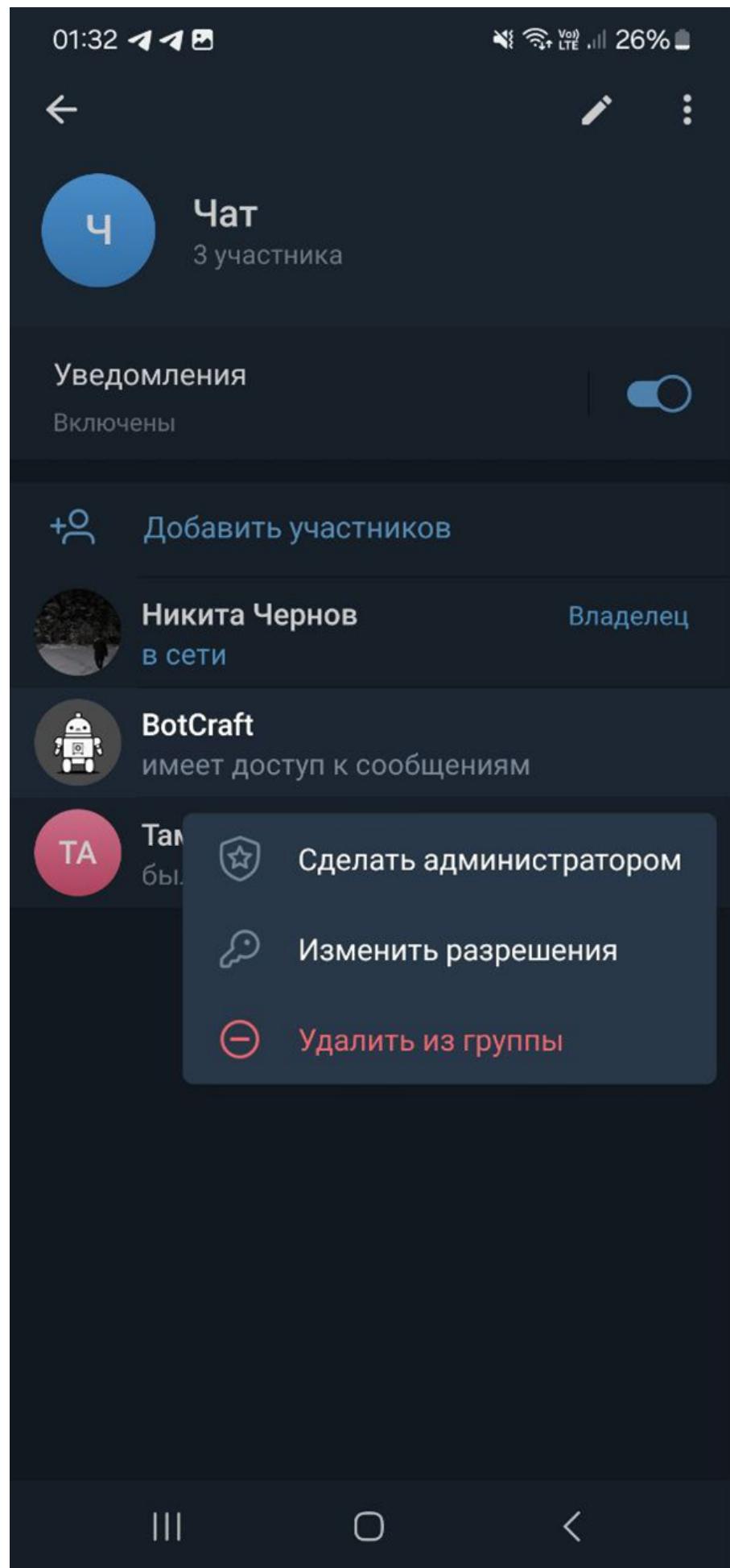


Рисунок 4.2 – Выдача боту необходимых прав

Пользователь может ознакомиться с функционалом бота написав сообщение /start или /help (рисунок 4.3.1 и 4.3.2).

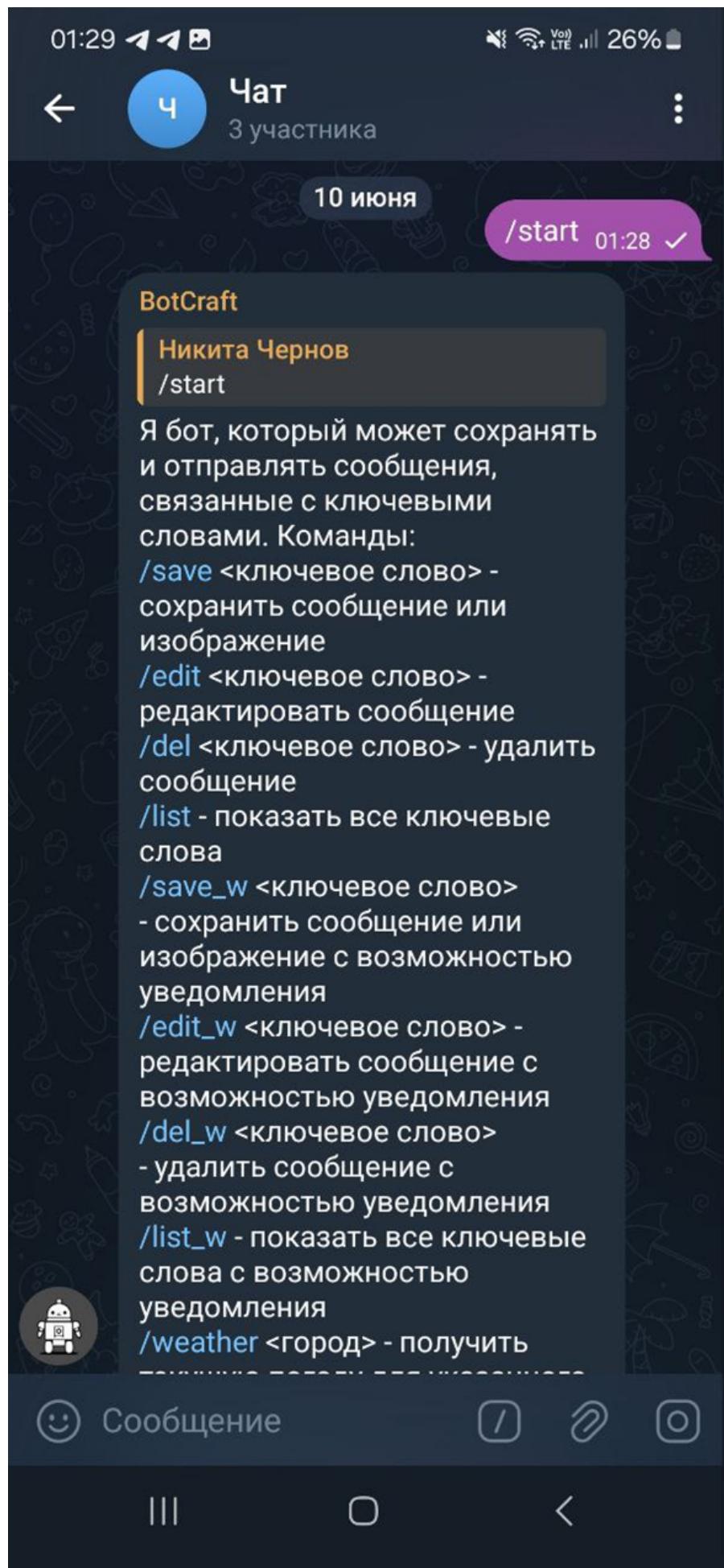


Рисунок 4.3 – Информация о функционале бота

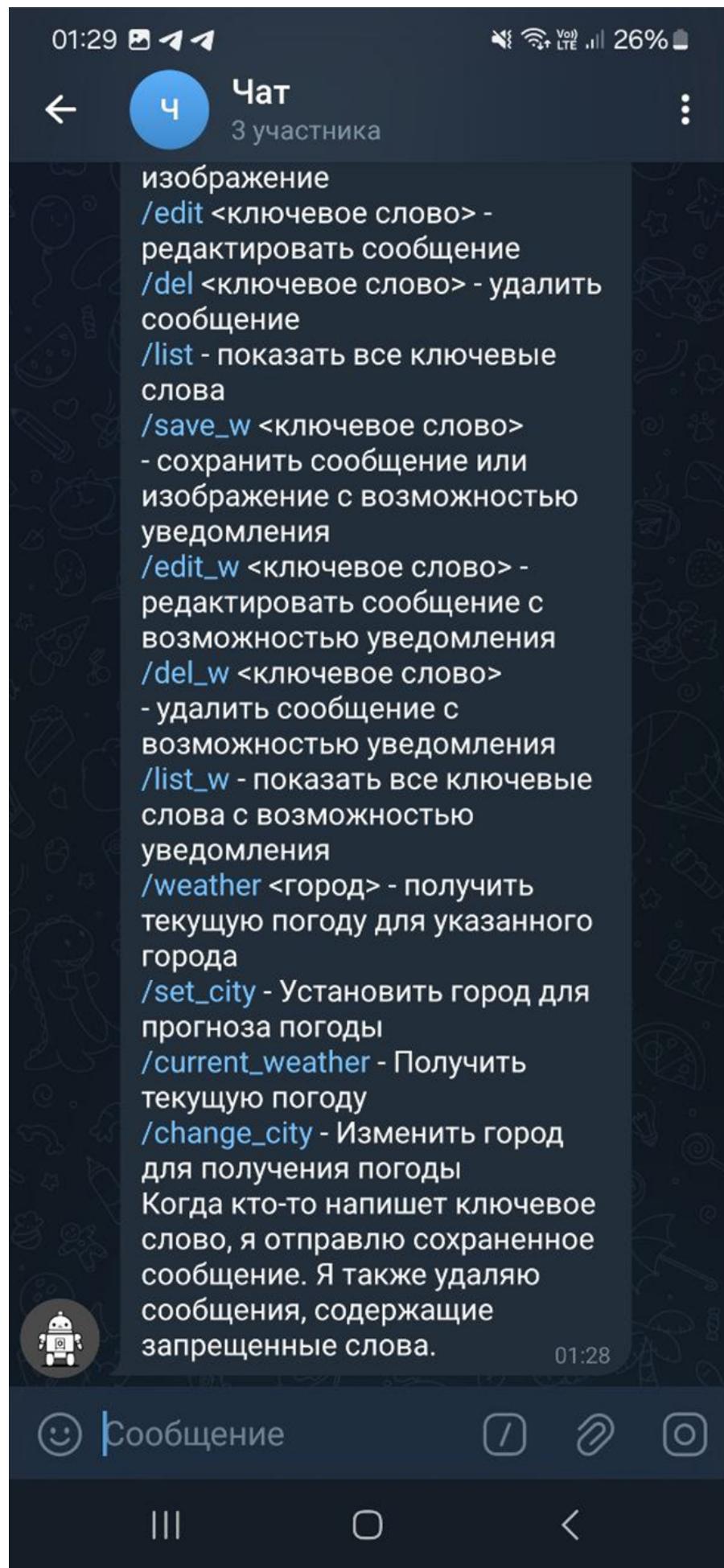


Рисунок 4.4 – информация о функционале бота

Пользователь может дать боту информацию о городе для получения прогноза погоды. Для того необходимо написать в чат команду /set_city (рисунок 4.4).

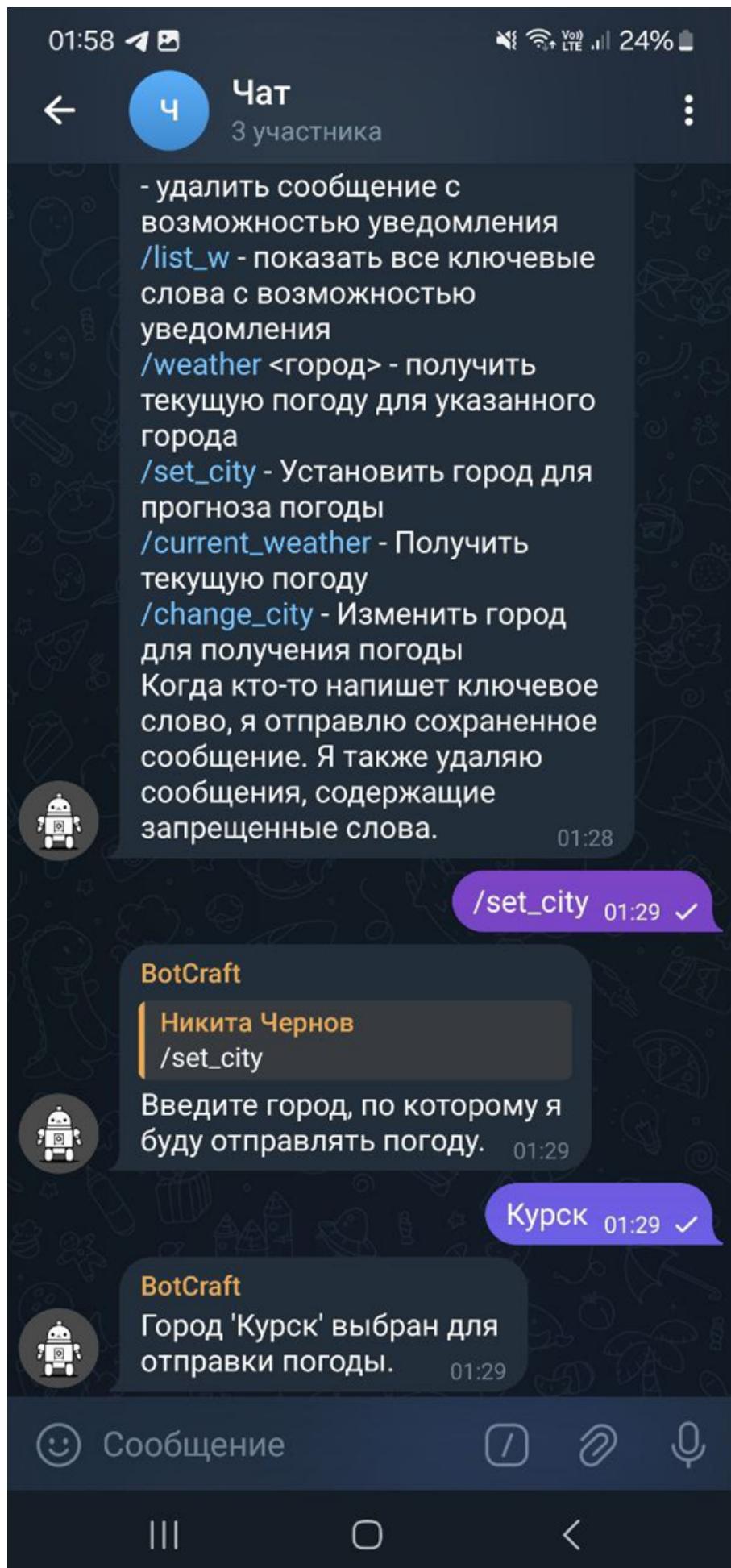


Рисунок 4.5 – получение ботом информации о городе для предоставления погоды

При написании в чате команды /current_weather Бот отправляет информацию о погоде в данный момент(рисунок 4.5).

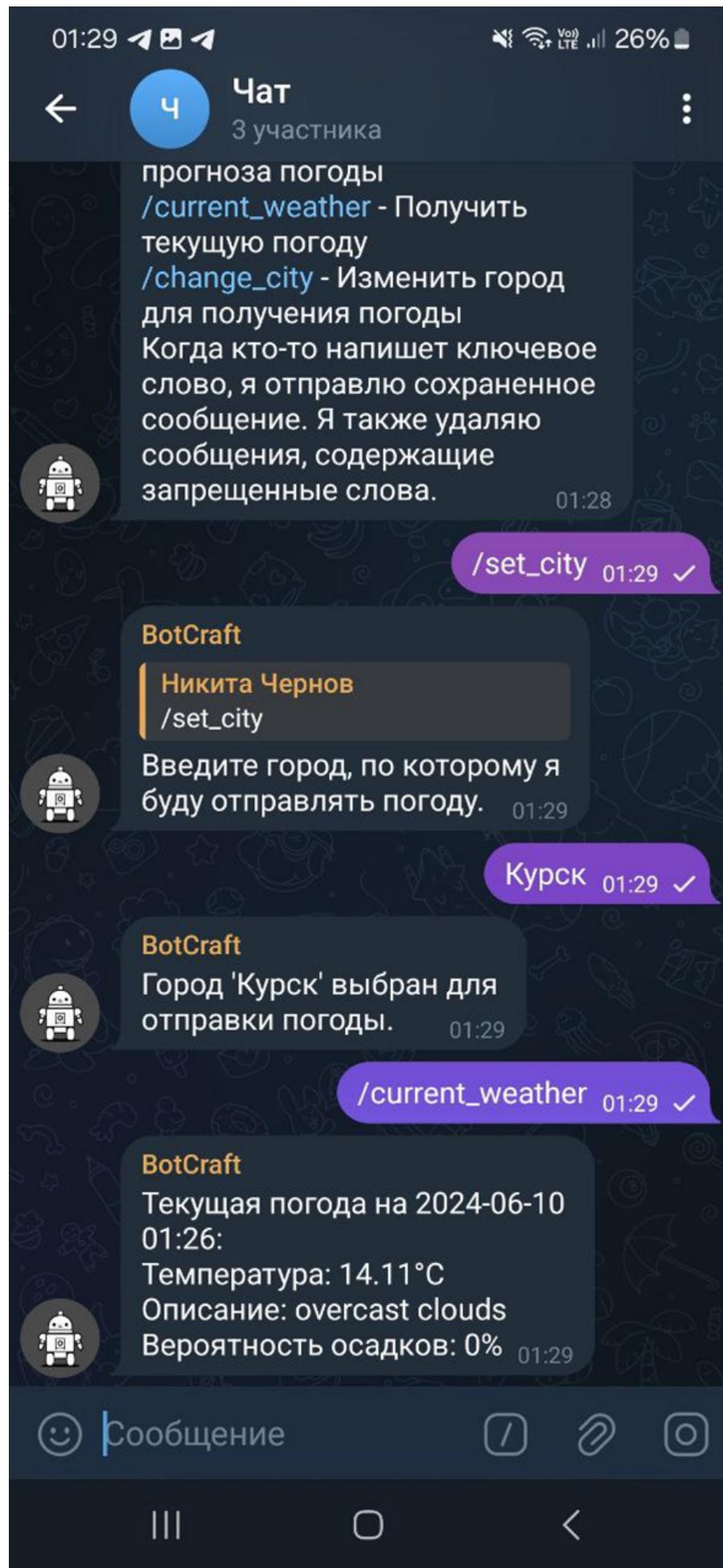


Рисунок 4.6 – Отправление информации о погоде в данный момент.

Пользователь может изменить город для получения прогноза погоды при помощи команды /change_city (рисунок 4.6.1 и 4.6.2).



Рисунок 4.7 – Изменение города для прогноза погоды



Рисунок 4.8 – Изменение города для прогноза погоды

Пользователь может попытаться отправить сообщение содержащие запрещенные слова (рисунок 4.7.1, 4.7.2 и 4.7.3)

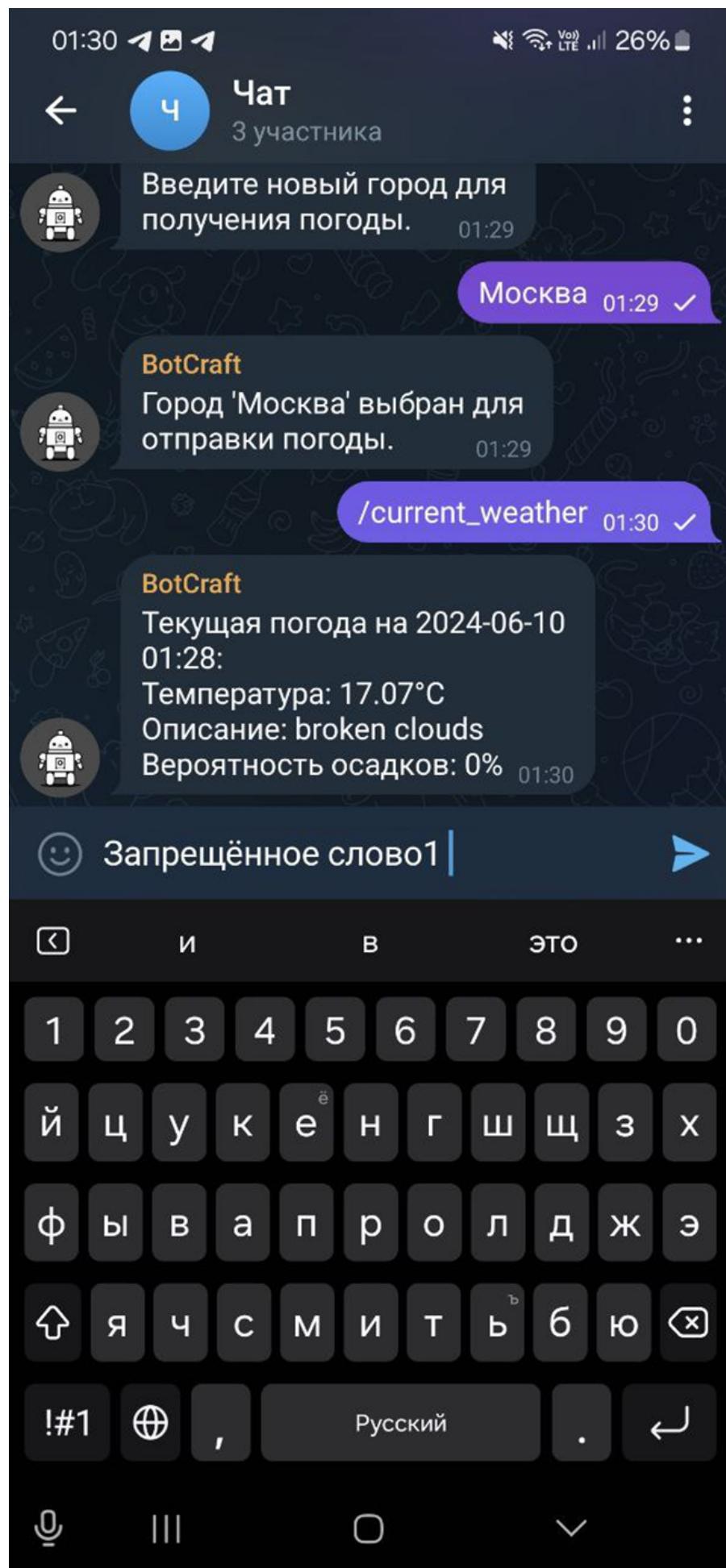


Рисунок 4.9 – Сообщение содержащие запрещенное слово

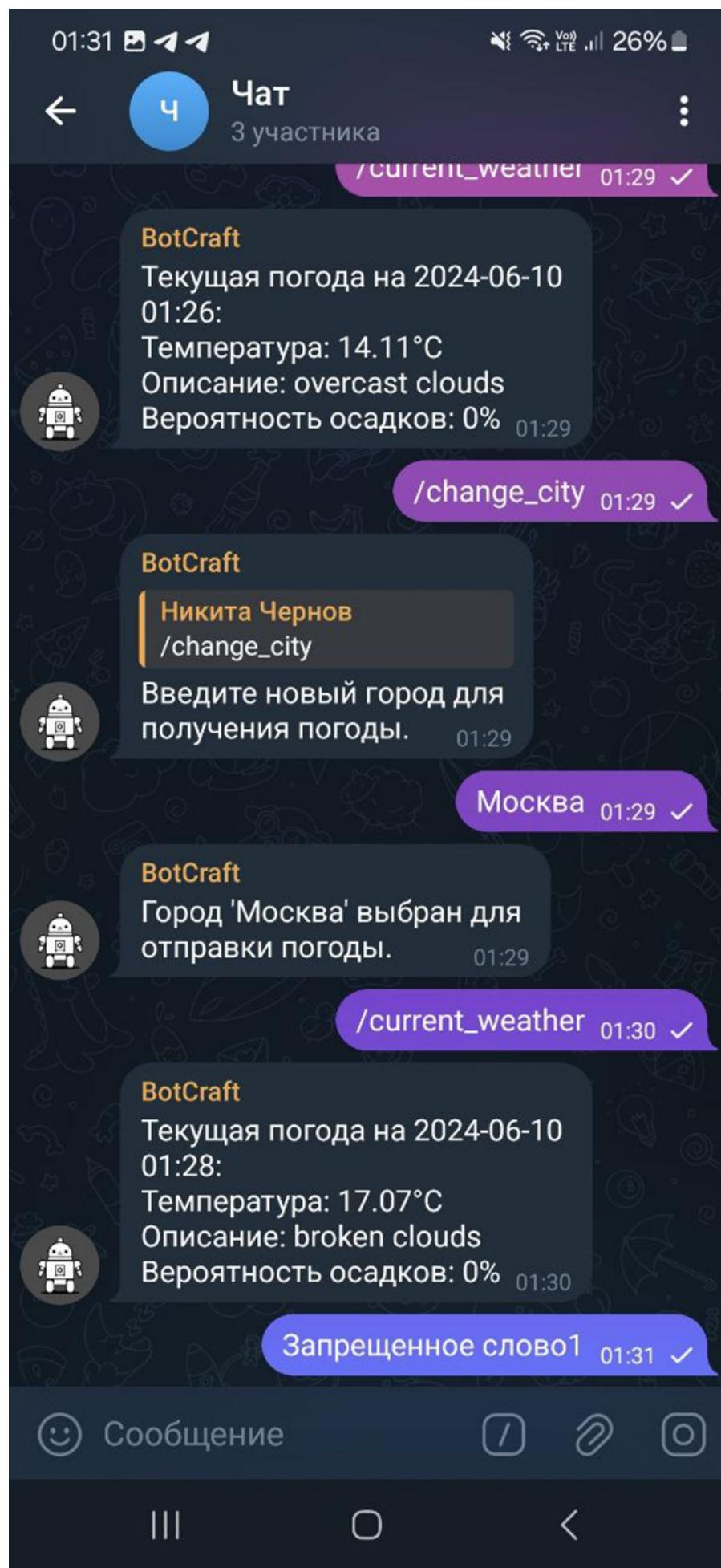


Рисунок 4.10 – Сообщение содержащие запрещенное слово попадает в чат

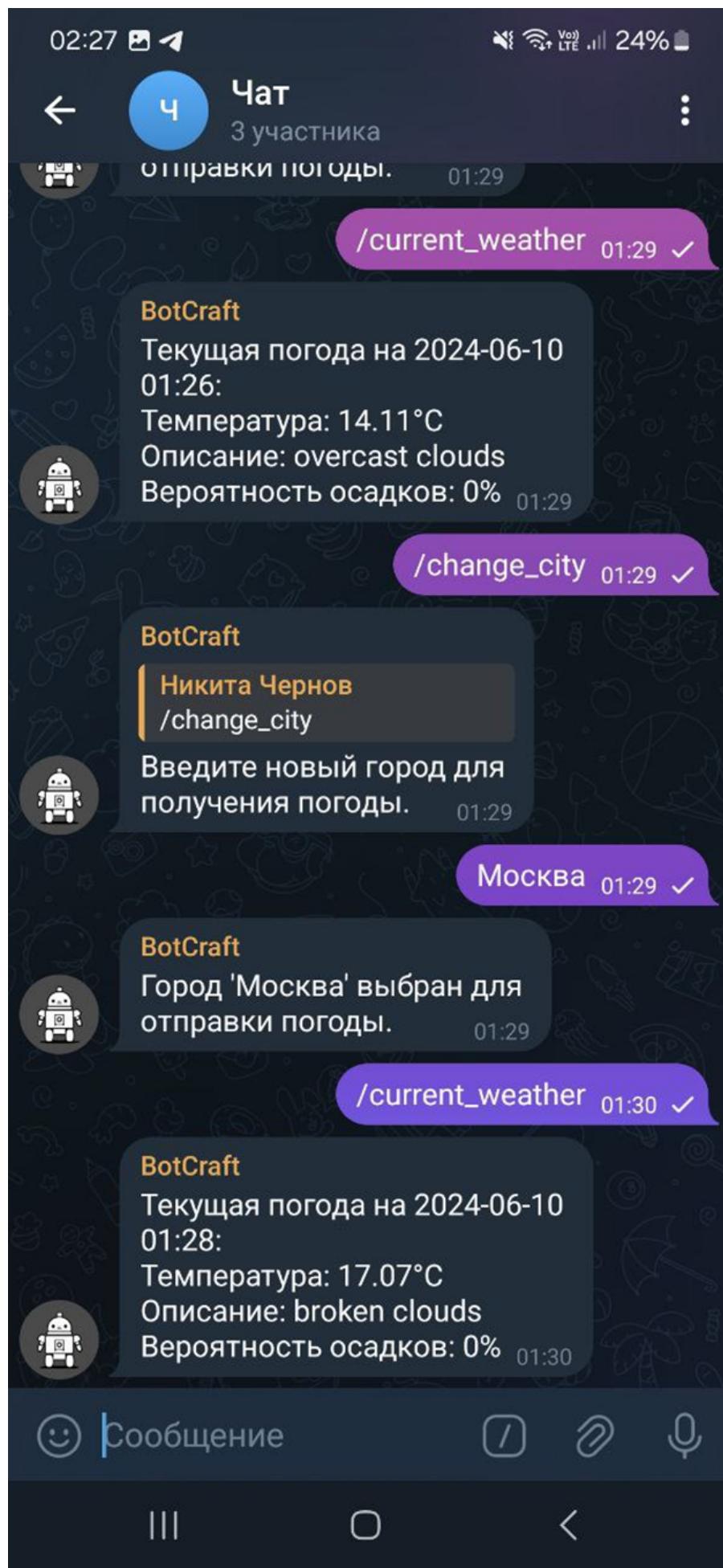


Рисунок 4.11 – Удаление запрещенного слова
62

Пользователь пытается отправить сообщение содержащие запрещенные слова замаскировав их текстом (рисунок 4.8.1,4.8.2,4.8.3).

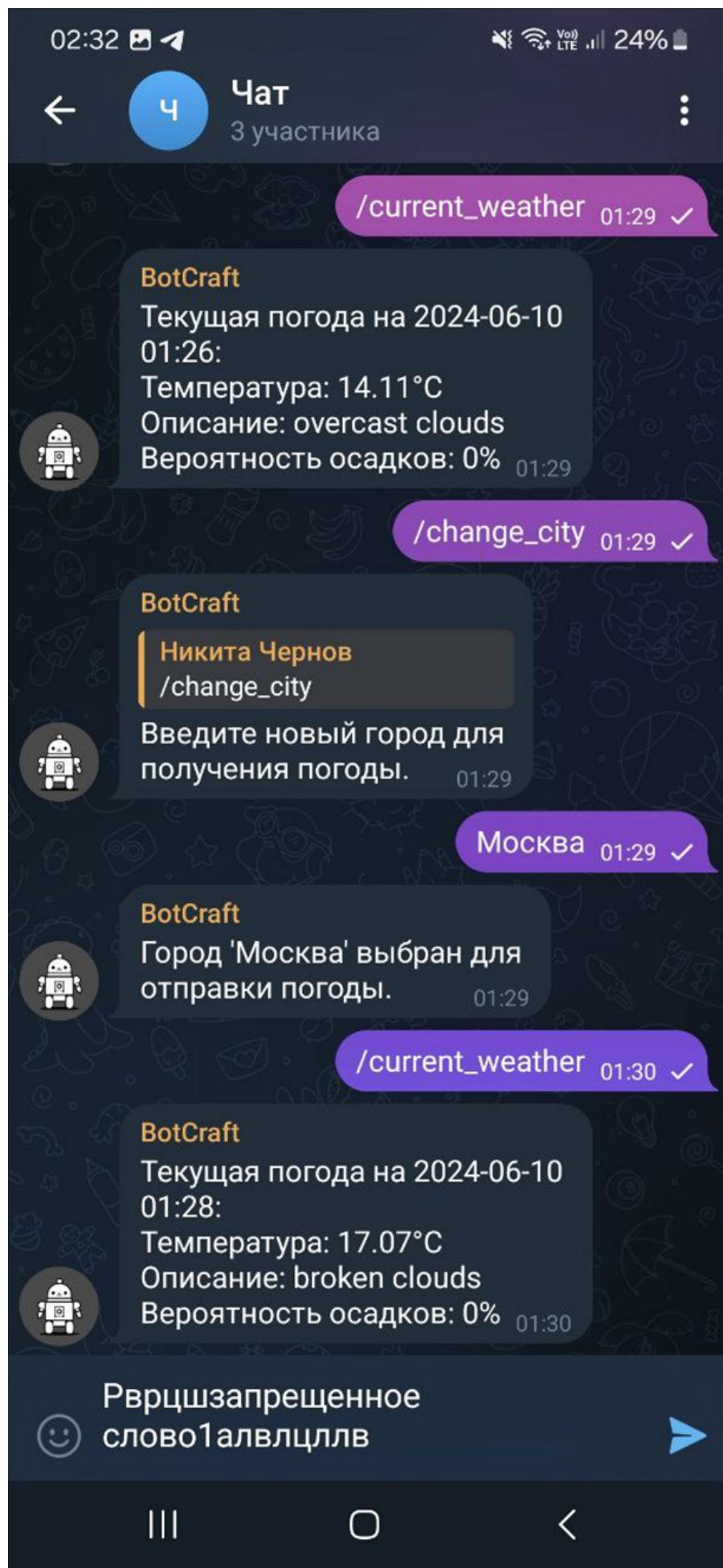


Рисунок 4.12 – Сообщение содержащие запрещенное слово

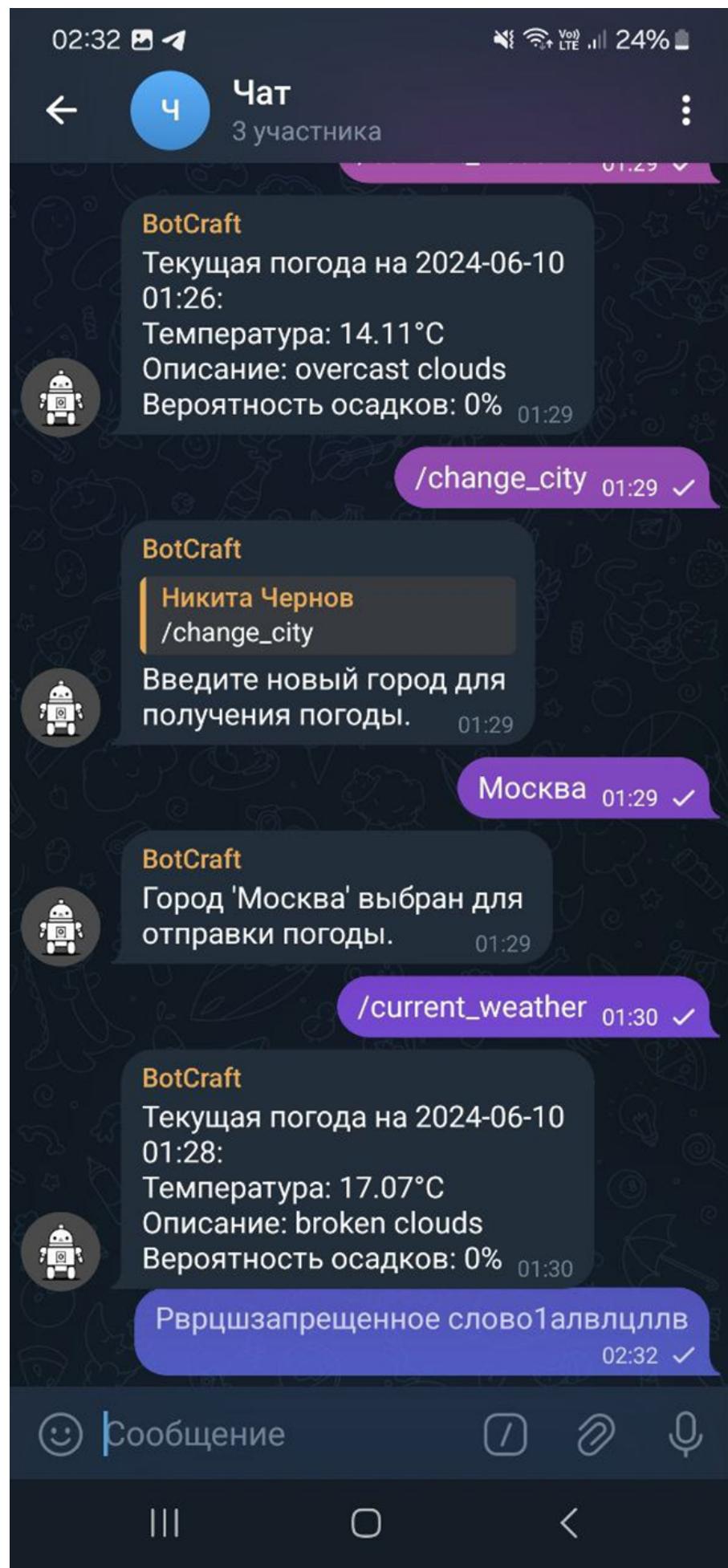


Рисунок 4.13 – Сообщение содержащие запрещенное слово

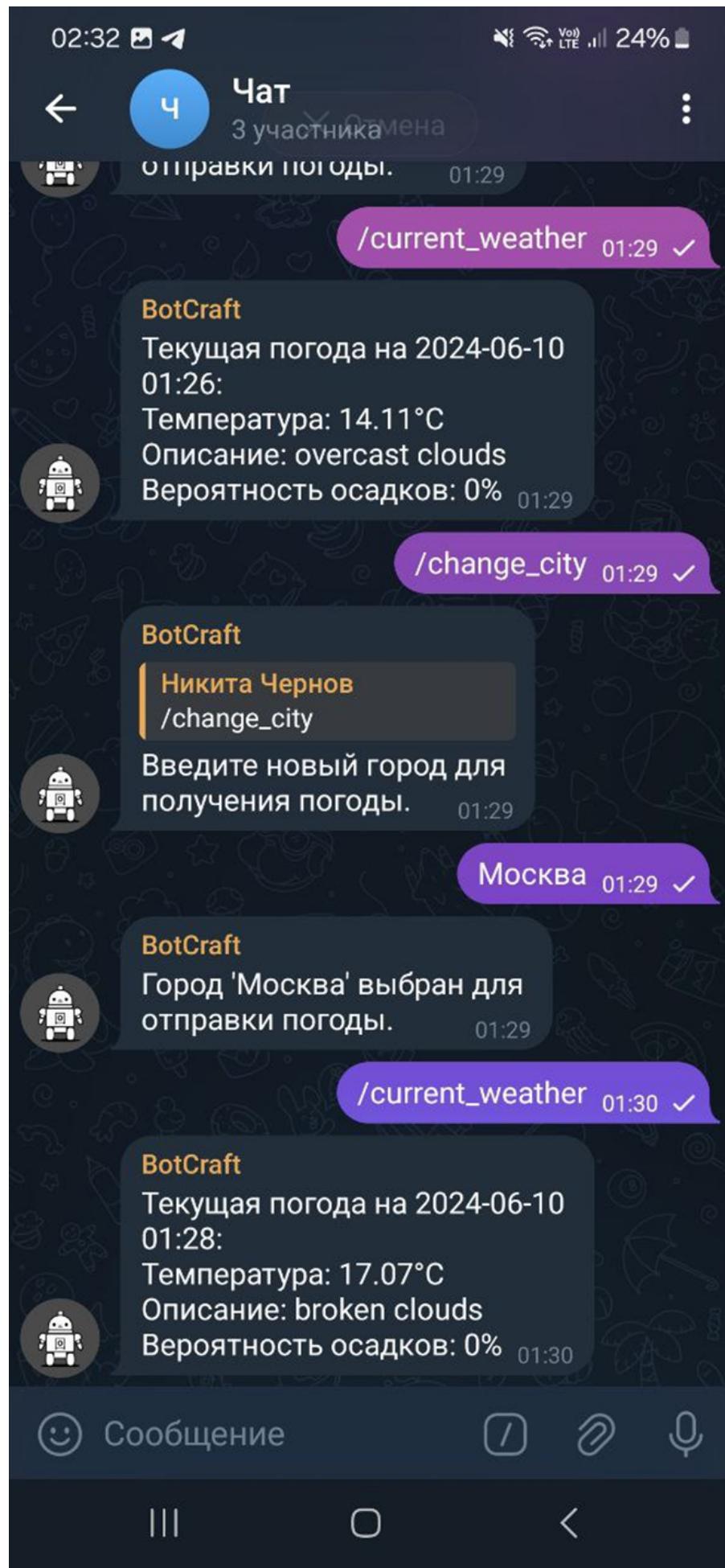


Рисунок 4.14 – Сообщение содержащие запрещенное слово удалено из чата

Пользователь собирается отправить сообщение содержащие ссылку(рисунок 4.9.1,4.9.2,4.9.3)

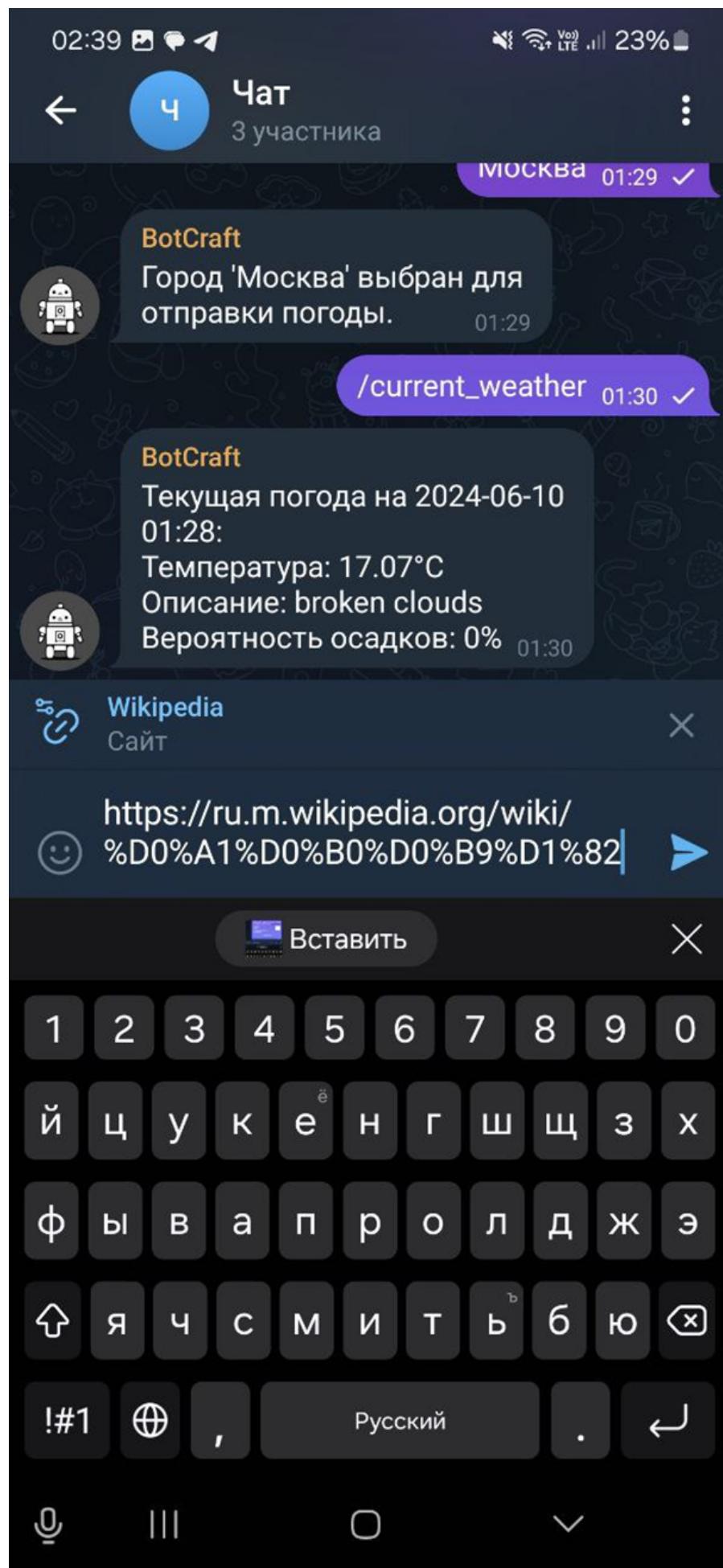


Рисунок 4.15 – Сообщение содержащие ссылку
68

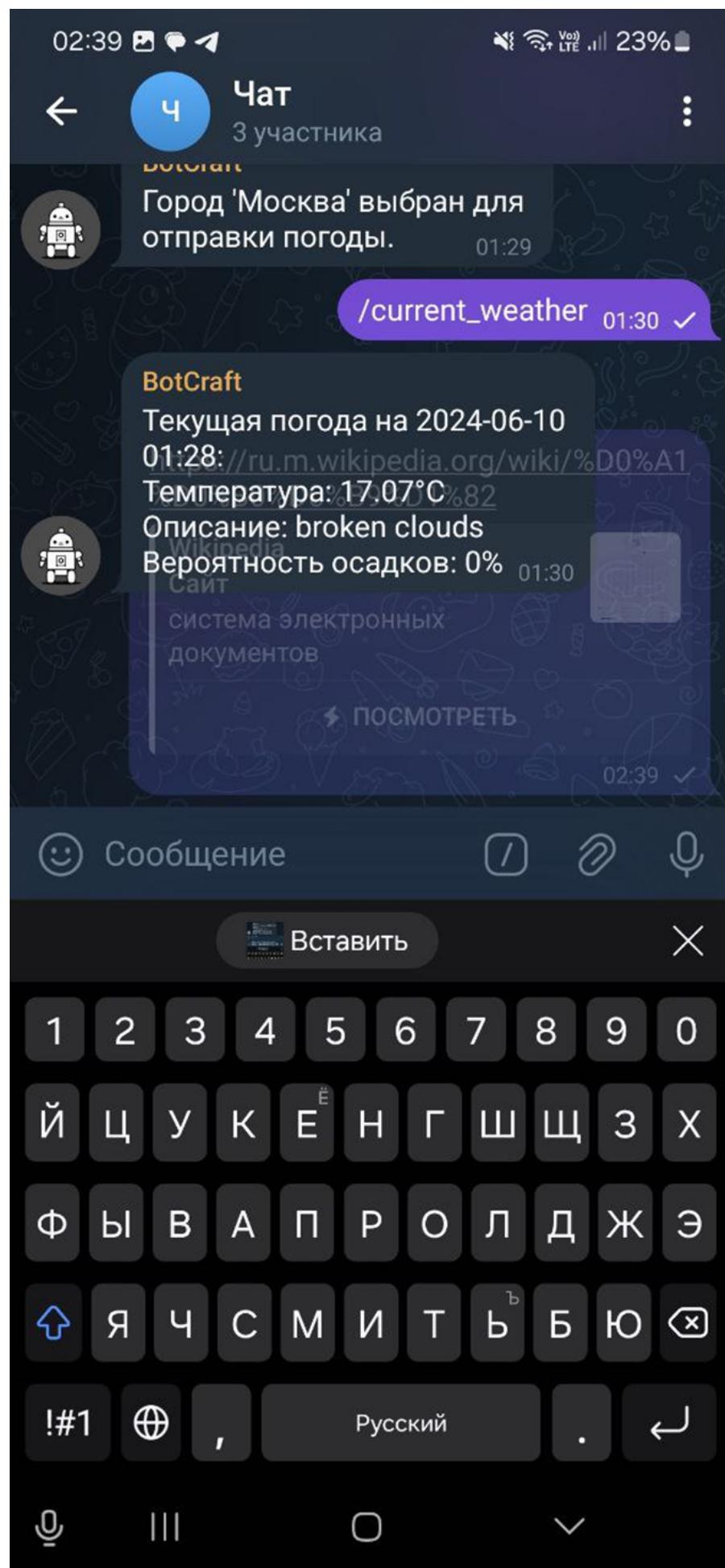


Рисунок 4.16 – Сообщение попадает в чат

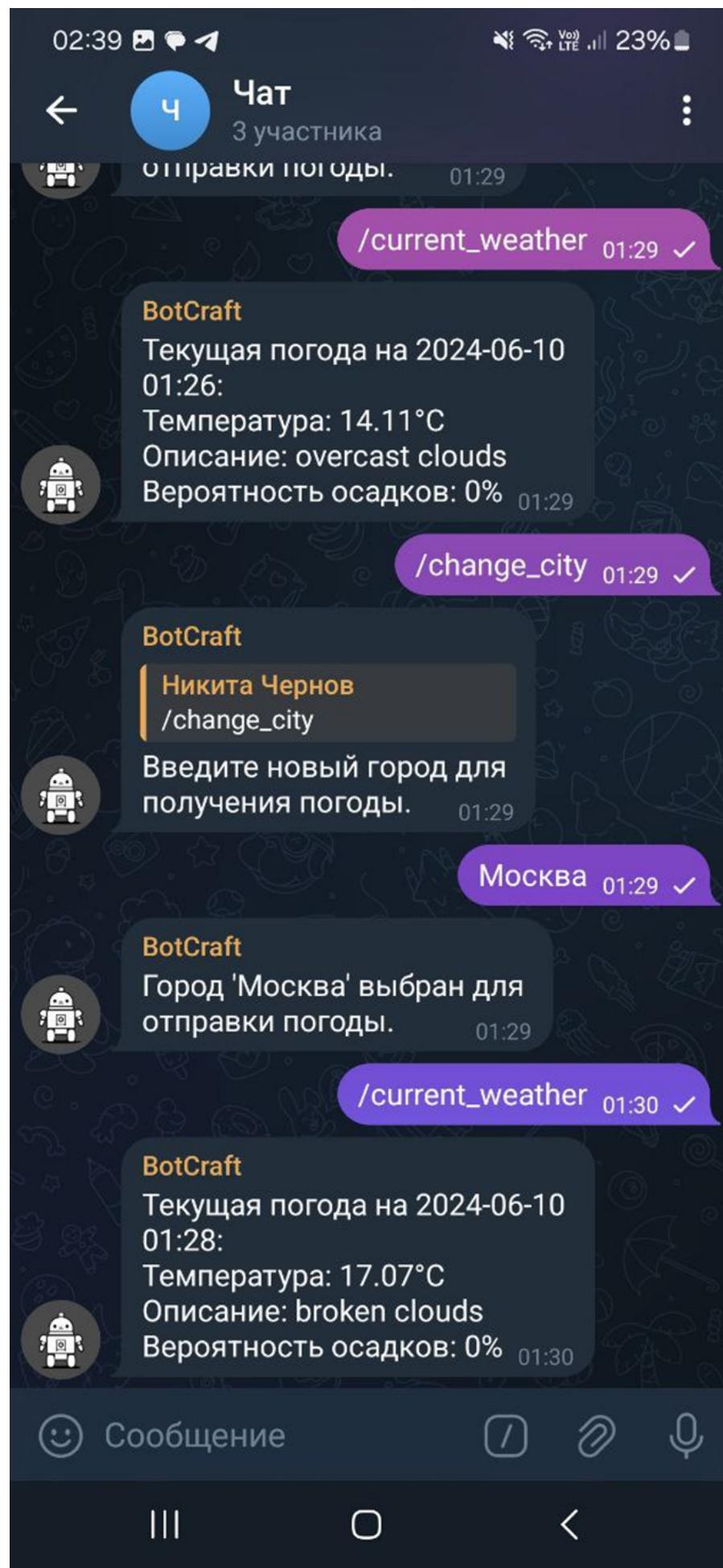


Рисунок 4.17 – Сообщение удаляется из чата

Пользователь пытается отправить сообщение содержащие ссылку замаскировав его текстом (рисунок 4.10.1, 4.10.2, 4.10.3).

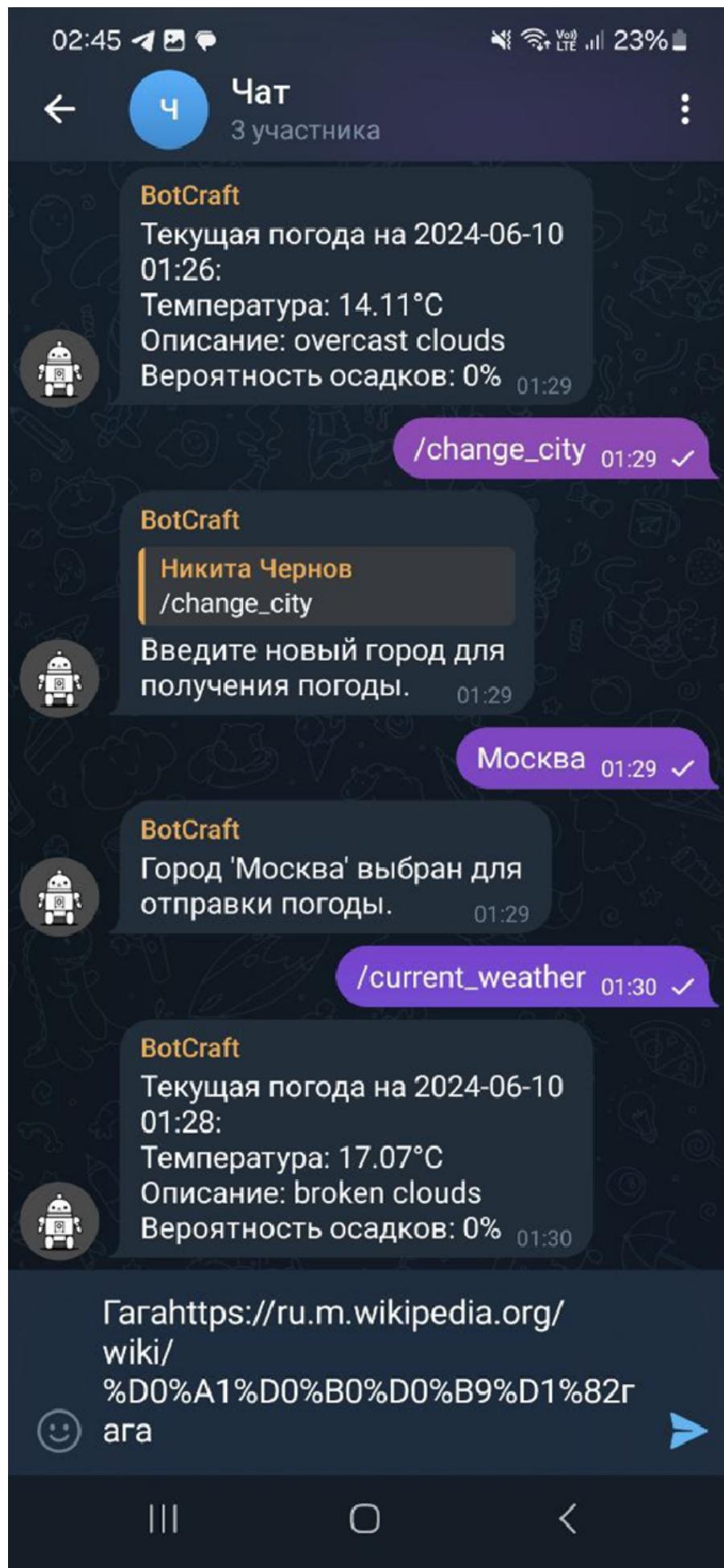


Рисунок 4.18 – Сообщение содержащие ссылку

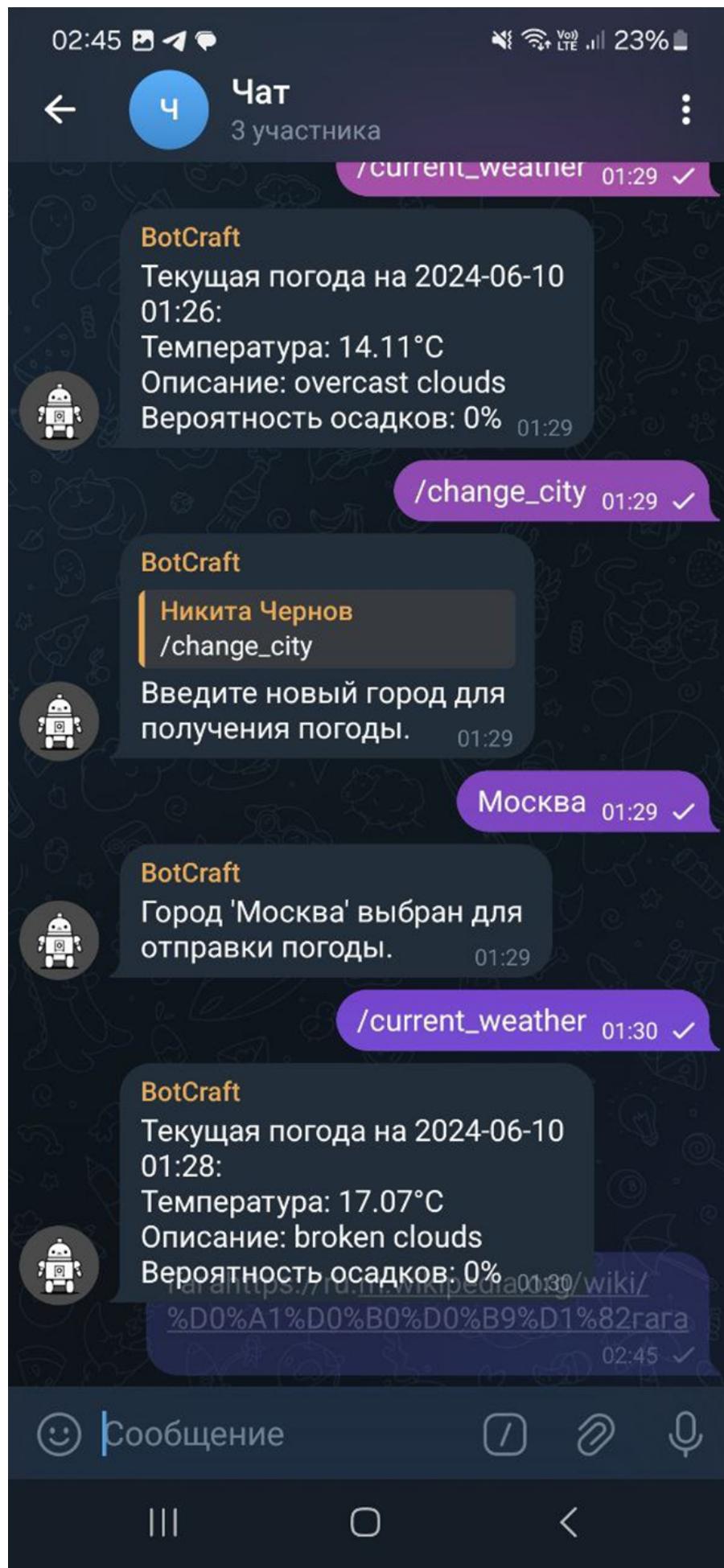


Рисунок 4.19 – Сообщение попадает в чат

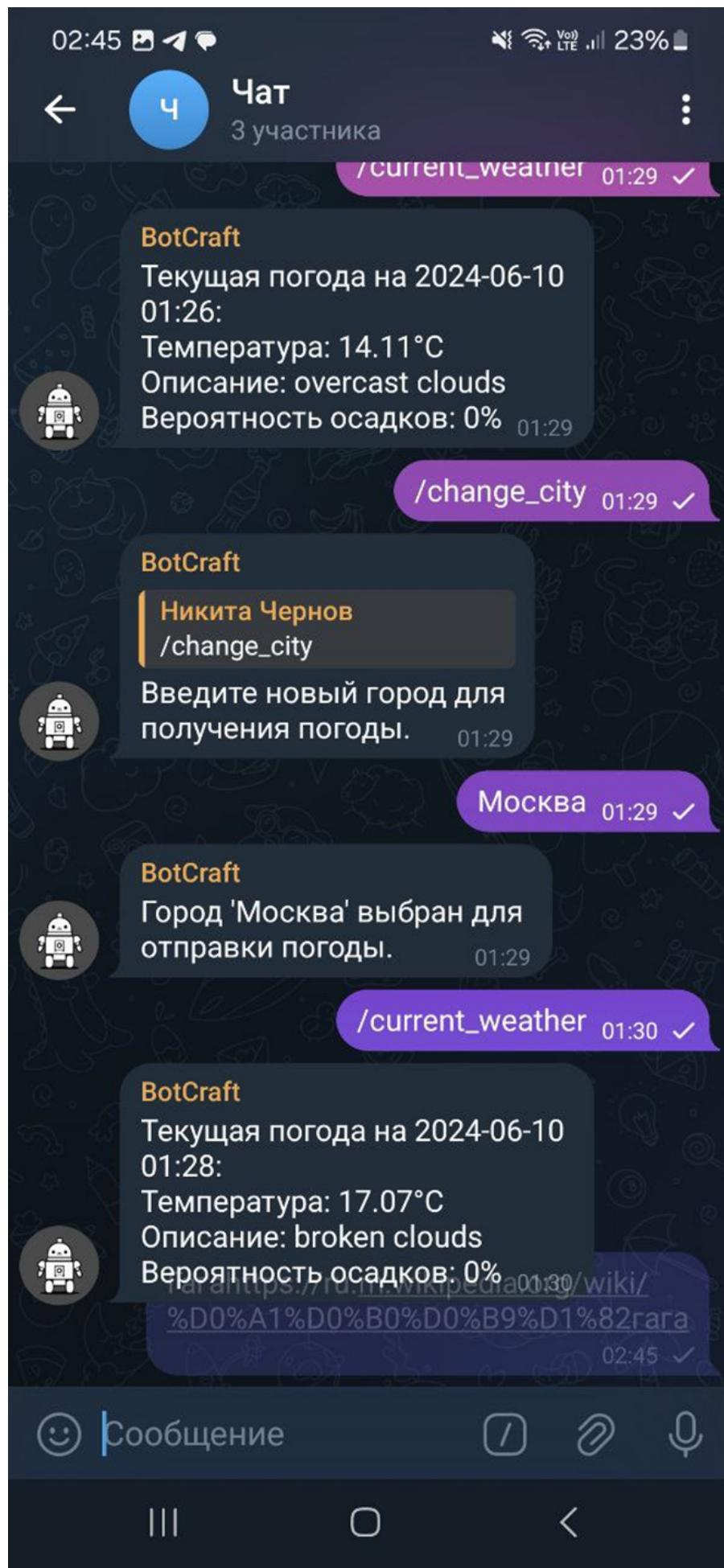


Рисунок 4.20 – Сообщение удаляется из чата

ЗАКЛЮЧЕНИЕ

Разработанный проект Telegram bot представляет собой комплексное решение для управления взаимодействием в чате с различными функциональными возможностями, включая обработку сообщений на основе ключевых слов, фильтрацию запрещенных слов и обновления погоды. Этот проект демонстрирует применение Python и библиотеки pyTelegramBotAPI для создания надежного чат-бота, который может обрабатывать команды пользователя, сохранять и извлекать данные, а также взаимодействовать с внешними API.

- Каждый чат может быть настроен индивидуально, что позволяет адаптировать функционал бота под конкретные нужды участников чата. Возможность настройки ежедневных уведомлений о погоде в заданное время обеспечивает актуальную информацию без необходимости лишних действий со стороны пользователей. Функция фильтрации запрещенных слов помогает поддерживать чистоту и безопасность чатов, автоматически удаляя нежелательные сообщения. Это особенно полезно в больших группах и публичных каналах, где важно поддерживать определенный уровень общения. Архитектура бота позволяет легко добавлять новый функционал и адаптировать существующие возможности под новые требования пользователей. Использование JSON-файлов для хранения ключевых слов и запрещенных слов делает управление данными простым и удобным. Индивидуальные ключевые слова и сообщения: Каждый чат может иметь свой уникальный набор ключевых слов, связанных с определенными сообщениями или медиафайлами. Пользователи могут сохранять и редактировать сообщения, привязанные к ключевым словам, что позволяет эффективно управлять информацией в чате.

Запрещенные слова: Для каждого чата можно настроить свой список запрещенных слов, что позволяет учитывать специфику аудитории и поддерживать необходимый уровень общения. Обновление списка запрещенных слов может производиться легко и быстро, что обеспечивает гибкость в управлении содержимым чата. Погодные уведомления:

- Пользователи могут настраивать время и частоту получения уведомлений о погоде, что позволяет

ет каждому чату получать информацию в наиболее удобное для участников время. • Возможность выбора города для погодных уведомлений позволяет адаптировать бота под географические предпочтения участников чата. Дан- ный проект представляет собой многофункциональный Телеграм-бот, кото- рый существенно улучшает взаимодействие пользователей с системой. Бла- годаря возможности индивидуальной настройки для каждого чата, бот адап- тируется под конкретные нужды и предпочтения пользователей. Он обеспе- чивает удобное управление сообщениями и медиафайлами, информирование о погодных условиях и поддерживает безопасность чатов посредством фильтрации запрещенных слов. Его архитектура и использование современных библиотек и технологий делают проект легко расширяемым и поддер- живае- мым, что является значительным преимуществом для его дальнейшего раз- вития и интеграции в различные системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Макконнелл , С. Совершенный код. Практическое руководство по разработке программного обеспечения / С. Макконнелл . – СПб. : БХВ, 2022. – 896 с. – ISBN 978-5-9909805-1-8. – Текст : непосредственный.
2. Марков А.А., "Чат-боты. Что такое и зачем нужны". Статья из журнала "Хакер №8 (2017), с. 64-69
3. Херманс , Ф. Ум программиста. Как понять и осмыслить любой код. / Ф. Херманс . – СПб. : БХВ, 2023. – 272 с. – ISBN 978-5-9775-1176-6. – Текст : непосредственный.
4. Брайант, Э. Компьютерные системы. Архитектура и програмирование / Э. Брайант, О'Халларон Р. – Москва : ДМК Пресс, 2022. – 994 с. – ISBN 978-5-97060-492-2. – Текст : непосредственный.
5. Иванов И.П., "Разработка и обучение чат-бота для Telegram на PHP". Статья из журнала "Web-программист №4 (2018), с. 20-25. – Текст : непосредственный.
6. Соколов А.К., "Использование Telegram API для создания чат-ботов". Книга. Москва: Техносфера, 2019. 240 с. ISBN 978-5-94836-777-2. – Текст : непосредственный.
7. Мартин, Р. С. Чистый код. Создание, анализ и рефакторинг / Р. С. Мартин. – Санкт-Петербург : Питер, 2021. – 464 с. – ISBN 978-5-4461-0960– Текст : непосредственный.
8. Белов Д.М., "Применение машинного обучения для улучшения функционала чат-ботов". Статья из журнала "Искусственный интеллект №2 (2020), с. 45-52. - Текст : непосредственный.
9. Петцольд, Ч. Код. Тайный язык информатики / Ч. Петцольд. – Москва : Манн, Иванов и Фербер, 2019. – 448 с. – ISBN 978-5-00117-545-2. – Текст : непосредственный.
10. Мартин, Р. С. Чистая архитектура. Искусство разработки программного обеспечения / Р. С. Мартин. – Санкт-Петербург : Питер, 2018. – 351 с. – ISBN 978-5-4461-0772-8.– Текст : непосредственный

11. Петров Н.С., "Особенности использования API Telegram при разработке чат-ботов". Материалы конференции "Современные информационные технологии 2019", с. 112-118 – Текст : непосредственный.
12. Джувел, Л. Совершенный софт / Л. Джувел. – Санкт-Петербург : Питер, 2020. – 480 с. – ISBN 978-5-4461-1621-8. – Текст : непосредственный.
13. Фленов, М.Е. Библия C#. 5-е издание. / М.Е. Фленов. – СПб. : БХВ, 2022. – 464 с. – ISBN 978-5-9775-6827-2. – Текст : непосредственный.
14. Троелсен, Э. Язык программирования C# 9 и платформа .NET 5: основные принципы и практики программирования, 10-е издание. / Э. Троелсен, Ф. Джепикс. – Москва : Диалектика, 2022. – 1392 с. – ISBN 978-5-907458-67-3. – Текст : непосредственный.
15. Джепикс, Ф. Язык программирования C# 7 и платформы .NET и .NET Core / Ф. Джепикс, Э. Троелсен. – Москва : Вильямс, 2018. – 1328 с. – ISBN: 978-1-4842-3017-6. – Текст : непосредственный.
16. Дронов, В.А. JavaScript. 20 уроков для начинающих. / В.А. Дронов. – СПб. : БХВ, 2020. – 352 с. – ISBN 978-5-9775-6589-9. – Текст : непосредственный.
17. Дронов , В.А. JavaScript. Дополнительные уроки для начинающих. / В.А. Дронов . – СПб. : БХВ, 2021. – 352 с. – ISBN 978-5-9775-6781-7. – Текст : непосредственный.
18. Фримен, А. Практикум по программированию на JavaScript / А. Фримен. – Москва : Вильямс, 2013. – 960 с. – ISBN 978-5-8459-1799-7. – Текст : непосредственный.
19. Умрихин, Е.Д. Разработка веб-приложений с помощью ASP.Net Core MVC. / Е.Д. Умрихин. – СПб. : БХВ, 2023. – 416 с. – ISBN 978-5-9775-1206-0. – Текст : непосредственный.
20. Вишняков Ю.М., Кодачигов В.И., Родзин С.И. Учебно–методическое пособие по курсам «Системы искусственного интеллекта», «Методы распознавания образов». Таганрог: Из–во ТРТУ, 2021. - 30 с. – ISBN: 978-8521479630. – Текст : непосредственный.

21. Ганди, Р. Head First. Git / Р. Ганди. – СПб. : БХВ, 2023. – 464 с. – ISBN 978-5-9775-1777-5. – Текст : непосредственный.
22. Чакон, С. Git для профессионального программиста / С. Чакон. – Санкт-Петербург : Питер, 2016. – 496 с. – ISBN 978-5-496-01763-3. – Текст : непосредственный.
23. Хориков, В. Принципы юнит-тестирования / В. Хориков. – СПб. : Питер, 2022. – 320 с. – ISBN 978-5-4461-1683-6. – Текст : непосредственный.
24. Гома Х. UML. Проектирование систем реального времени, параллельных и распределенных приложений: Пер. с англ. М.: ДМК Пресс, 2022. - 87 с. – ISBN: 978-1234567894. – Текст : непосредственный.
25. Маурисио, А. Эффективное тестирование программного обеспечения / А. Маурисио. – Москва : ДМК Пресс, 2023. – 370 с. – ISBN 978-5-97060-997-2. – Текст : непосредственный.
26. Гульяев А.К. MATLAB 5.3. Имитационное моделирование в среде Windows, М.: Корона прнт, 2021. - 33 с. – ISBN: 978-1234567896. – Текст : непосредственный.
27. Игнатьев, А. В. Тестирование программного обеспечения / А. В. Игнатьев. – Москва : Лань, 2021. – 56 с. – ISBN 978-5-8114-8072-2. – Текст : непосредственный.
28. Плаксин, М. А. Тестирование и отладка программ для профессионалов будущих и настоящих / М. А. Плаксин. – Москва : БИНОМ, 2020. – 170 с. – ISBN 978-5-00101-810-0. – Текст: непосредственный.
29. Дьяконов В. Mathematica 4: учебный курс. СПб: Питер, 2022. - 162 с. – ISBN: 978-1234567899. – Текст : непосредственный.
30. Шмитт, Э. Применение Web-стандартов. CSS и Ajax для больших сайтов / Э. Шмитт, К. Блессинг, Р. Черни . – СПб. : Корона-Принт, 2016. – 224 с. – ISBN 978-5-7931-0844-7. – Текст : непосредственный.
31. Вайсфельд, М. Объектно-ориентированное мышление / М. Вайсфельд. – СПб : Питер, 2014. – 304 с. – ISBN 978-5-496-00793-1. – Текст : непосредственный.

32. Джонсон, Р. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Р. Джонсон, Г. Эрих, Р. Хелм, Д. Влисседес. – Санкт-Петербург : Питер, 2016. – 366 с. – ISBN 978-5-459-01720-5. – Текст : непосредственный.
33. Вайсфельд, М. Объектно-ориентированное мышление / М. Вайсфельд. – СПб : Питер, 2014. – 304 с. – ISBN 978-5-496-00793-1. – Текст : непосредственный.
34. Гаско, Р. Объектно Ориентированное Программирование / Р. Гаско. – Москва : Солон-Пресс, 2021. – 298 с. – ISBN 978-5-91359-285-9. – Текст : непосредственный.
35. Ларман, К. Применение UML и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку: учебник и практикум для бакалавриата и магистратуры / К. Ларман. – М.: ООО “И.Д. Вильямс”, 2013. – 426 с. – ISBN 978-5-8459-1185-8.. – Текст : непосредственный.
36. Агальцов, В.П. Базы данных. Локальные базы данных: учебник / В.П. Агальцов. - М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2016. - 352 с. - ISBN 978-5-16-011625-9. - Текст: непосредственный
37. Вайсфельд, М. Объектно-ориентированное мышление / М. Вайсфельд. - СПб.: Питер, 2014. - 304 с. - ISBN 978-5-496 -00793-1. - Текст: непосредственный.

ПРИЛОЖЕНИЕ А
Представление графического материала

Графический материал, выполненный на отдельных листах, изображен на рисунках А.1–А.9.

610424030908525288

1

Сведения о ВКРБ

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА ПО ПРОГРАММЕ БАКАЛАВРИАТА

«Бизнес-проект «Программно-информационная система для
конструирования чат-ботов для Telegram,
разработка модуля конструирования»

Руководитель ВКР
К.Ф.-м.н., доцент
Елена Анатольевна Петрик

Автор ВКР
студент группы ПО-01б
Чернов Никита Анатольевич

ВКРБ 2068443.09.03.04.24.019			
Фамилия И. О.	Иванова	Д.И.	Иванова
Имя отчества	Наталья Ильинична		
Родившийся	1991 год 10 мес.		
Национальность	Немецкая		
Место жительства	г. Самара		
Место обучения	ЮЗУПИ		
Место прохождения практики	ЮЗУПИ		

Рисунок А.1 – Сведения о ВКРБ

Цели и задачи разработки

Цель и задачи разработки

Цель настоящей работы – создание программно-информационной системы для конструирования чат-ботов для Telegram, а также узнать мнение о них у любителей и заинтересованных в этом сообществах. Необходимо создать чат-бота с использованием современных технологий, которые обеспечивают качественную и масштабируемую работу приложения, а также удовлетворяют потребности пользователей. Для достижения поставленной цели необходимо решить следующие задачи:

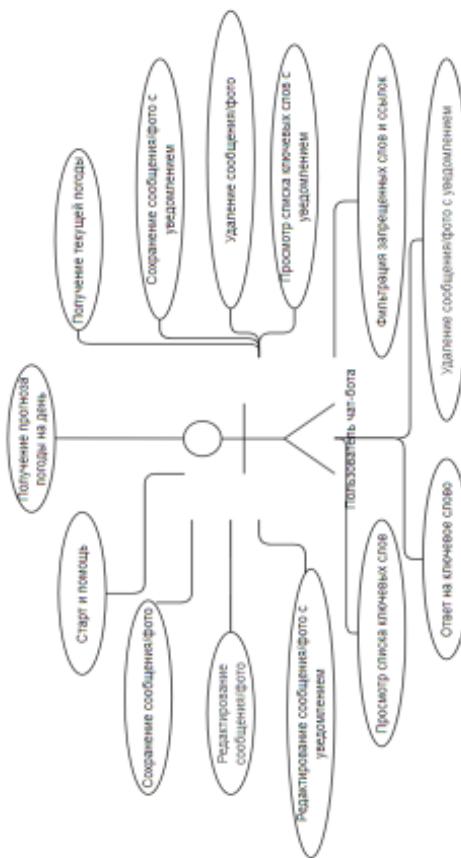
- разработать модель данных программной системы; определить варианты использования
- программной системы; разработать требования к программной системе;
- осуществить проектирование программной системы; разработать архитектуру
- программной системы; разработать пользовательский интерфейс межпользовательского взаимодействия;
- зарегистрировать бота в Telegram с помощью @Botfather и получить токен для доступа к API;
- провести тестирование модуля межпользовательского взаимодействия программной системы.

ВКРБ 206843.09.03.04.24.019					
Задача	Цель	Задачи	Методы	Инструменты	Личные
1. Постановка задачи	Постановка задачи	1. Постановка задачи	Документ	План и задачи разработки	Борисов А. С.
2. Планирование	Планирование	2. Планирование	Документ	План и задачи разработки	Борисов А. С.
3. Выполнение	Выполнение	3. Выполнение	Документ	План и задачи разработки	Борисов А. С.
4. Оценка	Оценка	4. Оценка	Документ	План и задачи разработки	Борисов А. С.
5. Выводы	Выводы	5. Выводы	Документ	План и задачи разработки	Борисов А. С.

Рисунок А.2 – Цель и задачи разработки

Диаграмма использования

3

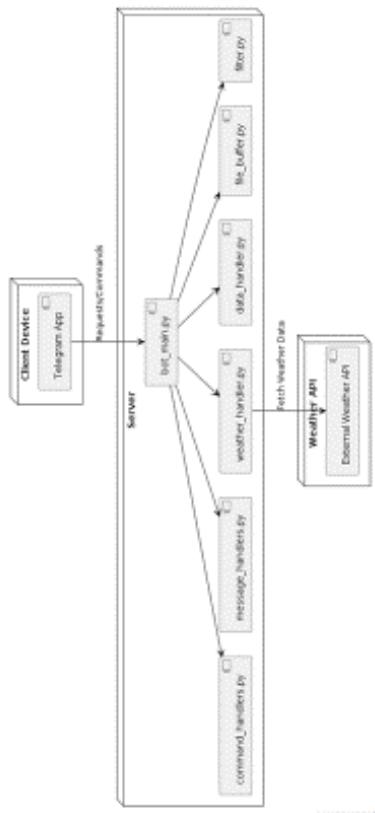


БКРБ 2068443_09_03_04_24_0	
Фамилия И. А.	Иванов
Номер т.т.с.	1234567890
Родительство	Дети
Парентство	Паренты
Парентальное	Паренты: И. А.
Имя	Иван
Фамилия	Иванов
Отчество	Иванович
Пол	Мужчина
Возраст	24
Город	Москва
Парентальный	Паренты: И. А.
Имя	Иван
Фамилия	Иванов
Отчество	Иванович
Пол	Мужчина
Возраст	24
Город	Москва

Рисунок А.3 – Диаграмма использования

Диаграмма развертывания приложения

4



ВКРБ 206843.09.03.04.24.019				
Фамилия и Имя	Логин	Пароль	Имя	Фамилия
Андрейкин	Andrey@A.S.			
Романовна	Romanova@R.S.			
Наталья	Natalya@N.S.			
			Андрей	Андрейкин
			Мария	Марковна
			Роман	Романов
			Наталья	Натальевна

Рисунок А.4 – Диаграмма развертывания

Диаграмма классов модели

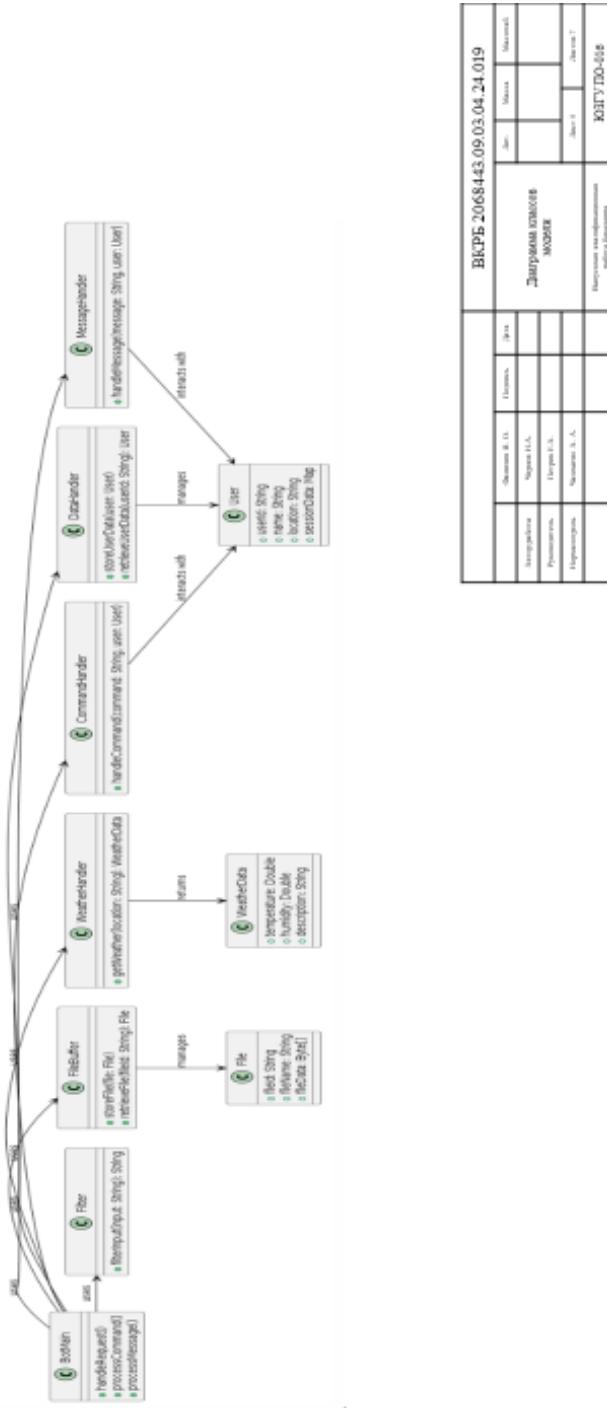


Рисунок А.5 – Диаграмма классов модели

Диаграмма классов-контроллеров

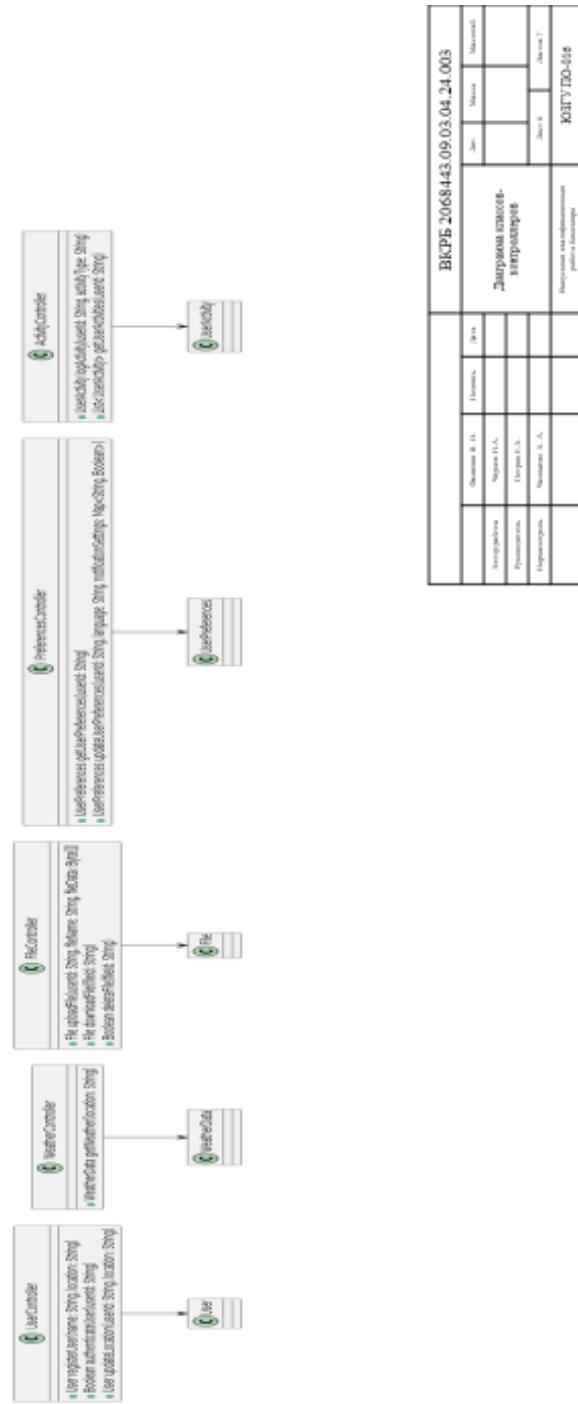


Рисунок А.6 – Диаграмма классов модули-контроллеров

Заключение

Разработка программно-информационной системы для конструирования чат-ботов для Telegram является актуальной и перспективной задачей.

Основные результаты работы:

1. Разработана модель данных программной системы. Определены варианты использования программной системы. Разработаны требования к программной системе.
 2. Осуществлено проектирование программной системы. Разработана архитектура программной системы. Разработан пользовательский интерфейс межпользовательского взаимодействия.
 3. Зарегистрирован бот в Telegram с помощью @Botfather и получен токен для доступа к API.
- Проведено тестирование модуля межпользовательского взаимодействия программной системы. Проведено системное тестирование компонентов программной системы.

БКРБ 2068443/09/03/04/24/01/9			
Фамилия И.О.	Имя	Отчество	Место
Андрющик Наталья Н.С.			
Романчик Юрий Н.А.			
Нарышкин Петр Петрович	Петр Петрович		
			Бюро 7 ЮИГУПО-45
			ЮИГУПО-45

Рисунок А.7 – Заключение

ПРИЛОЖЕНИЕ Б

Фрагменты исходного кода программы

`bot_main.py`

```
1 import os
2 from telebot import TeleBot
3 from command_handlers import register_command_handlers
4 from weather_handler import register_weather_handlers, send_daily_weather
5 from data_handler import load_data
6 import schedule
7
8 # Инициализация бота
9 bot_token = '6844853072:AAE9E21vq017QAWP_9ZSNBF-0460bLBFChM'
10 bot = TeleBot(bot_token)
11 data_file_path = 'chat_keywords.json'
12
13 # Загрузка сохраненных данных
14 chat_keywords = load_data(data_file_path)
15
16 # Регистрация командных обработчиков
17 register_command_handlers(bot, chat_keywords, data_file_path)
18 register_weather_handlers(bot)
19
20 # Настройка расписания для ежедневного отправления погоды
21 schedule.every().day.at("09:00").do(send_daily_weather, bot=bot)
22
23 # Запуск бота
24 if __name__ == '__main__':
25     while True:
26         schedule.run_pending()
27         bot.polling(none_stop=True)
```

`filter.py`

```
1 import json
2
3 def load_banned_words(file_path):
4     try:
5         with open(file_path, 'r', encoding='utf-8') as f:
6             banned_words = json.load(f)
7             return banned_words
8     except Exception as e:
9         print(f"Error loading banned words from {file_path}: {e}")
10        return []
```

```

12 def contains_banned_words(message, banned_words):
13     return any(word in message for word in banned_words)
14
15 def contains_keywords(message, keywords):
16     return any(keyword in message for keyword in keywords)

    file_buffer.py

1 # file_buffer.py
2 import os
3 import pathlib
4 from pathlib import Path
5
6 FILE_BUFFER_DIR = "files"
7
8
9 def initialize_buffer():
10     if not os.path.exists(FILE_BUFFER_DIR):
11         os.makedirs(FILE_BUFFER_DIR)
12
13
14 def save_file(chat_id, file_info, bot, file_type):
15     Path(f'{FILE_BUFFER_DIR}/{chat_id}/').mkdir(parents=True, exist_ok=True)
16     file_path = file_info.file_path
17     downloaded_file = bot.download_file(file_path)
18     src = f'{FILE_BUFFER_DIR}/{chat_id}/' + file_path.replace('photos/', '').
19         replace('videos/', '').replace(
20             'animations/', '')
21     with open(src, 'wb') as new_file:
22         new_file.write(downloaded_file)
23
24     return src
25
26
27 def get_file_path(chat_id, file_name):
28     file_path = f'{FILE_BUFFER_DIR}/{chat_id}/{file_name}'
29     if os.path.exists(file_path):
30         return file_path
31     return None
32
33
34 def remove_file(chat_id, file_name):
35     file_path = get_file_path(chat_id, file_name)
36     if file_path and os.path.exists(file_path):
37         os.remove(file_path)

```

message_handlers.py

```
1 from data_handler import load_chat_keywords, save_chat_keywords
2
3 chat_keywords = load_chat_keywords()
4
5
6 def register_message_handlers(bot):
7     @bot.message_handler(content_types=['text', 'photo', 'video', 'document'])
8
9     def handle_media_message(message):
10         chat_id = message.chat.id
11         keyword = 'your_keyword_here' # Определите, как вы хотите получить
12             Ключевое слово
13         content = None
14
15         if message.content_type == 'photo':
16             content = message.photo[-1].file_id
17         elif message.content_type == 'video':
18             content = message.video.file_id
19         elif message.content_type == 'document':
20             content = message.document.file_id
21         elif message.content_type == 'text':
22             content = message.text
23
24         if chat_id not in chat_keywords:
25             chat_keywords[chat_id] = {}
26
27             chat_keywords[chat_id][keyword] = {'type': message.content_type,
28                                         'content': content}
29             save_chat_keywords(chat_keywords)
30
31             bot.send_message(chat_id, f'Media with keyword "{keyword}" has been
32                                         saved.')
33
```

weather_handler.py

```
1 import requests
2 import json
3 from datetime import datetime
4 from telebot import TeleBot
5
6 WEATHER_API_URL = "http://api.openweathermap.org/data/2.5/forecast"
7 CURRENT_WEATHER_API_URL = "http://api.openweathermap.org/data/2.5/weather"
8 API_KEY = 'b95abfcde186f22560a271a013b1d2b'
9 WEATHER_FILE = 'weather_config.json'
10
```

```

11
12 def load_weather_config():
13     try:
14         with open(WEATHER_FILE, 'r', encoding='utf-8') as file:
15             return json.load(file)
16     except FileNotFoundError:
17         return {}
18     except json.JSONDecodeError:
19         return {}
20
21
22 def save_weather_config(data):
23     with open(WEATHER_FILE, 'w', encoding='utf-8') as file:
24         json.dump(data, file, ensure_ascii=False, indent=4)
25
26
27 def get_weather(city):
28     params = {
29         'q': city,
30         'appid': API_KEY,
31         'units': 'metric',
32         'cnt': 8 # Получаем прогноз на 24 часа (8 временных точек по 3 часа)
33     }
34     response = requests.get(WEATHER_API_URL, params=params)
35     data = response.json()
36
37     if response.status_code == 200:
38         weather_data = []
39         for entry in data['list']:
40             dt = datetime.fromtimestamp(entry['dt'])
41             temp = entry['main']['temp']
42             description = entry['weather'][0]['description']
43             rain = entry.get('rain', {}).get('3h', 0)
44             weather_data.append((dt, temp, description, rain))
45     return weather_data
46 else:
47     return None
48
49
50 def get_current_weather(city):
51     params = {
52         'q': city,
53         'appid': API_KEY,
54         'units': 'metric'
55     }

```

```

56     response = requests.get(CURRENT_WEATHER_API_URL, params=params)
57     data = response.json()
58
59     if response.status_code == 200:
60         dt = datetime.fromtimestamp(data['dt'])
61         temp = data['main']['temp']
62         description = data['weather'][0]['description']
63         rain = data.get('rain', {}).get('1h', 0)
64         return (dt, temp, description, rain)
65     else:
66         return None
67
68
69 def format_weather_message(weather_data):
70     message = "Погода на сегодня:\n"
71     for dt, temp, description, rain in weather_data:
72         time_of_day = dt.strftime("%H:%M")
73         message += f"{time_of_day}: Температура {temp}°C, {description},\n        вероятность осадков {rain}%\n"
74     return message
75
76
77 def format_current_weather_message(weather_data):
78     dt, temp, description, rain = weather_data
79     message = f"Текущая погода на {dt.strftime('%Y-%m-%d %H:%M')}:\n        Температура: {temp}°C\n        Описание: {description}\n        Вероятность осадков: {rain}%"
80     return message
81
82
83 def send_daily_weather(bot: TeleBot):
84     config = load_weather_config()
85     for chat_id, city in config.items():
86         weather_data = get_weather(city)
87         if weather_data:
88             message = format_weather_message(weather_data)
89             bot.send_message(chat_id, message)
90
91
92 def set_weather_city(bot: TeleBot, message):
93     chat_id = str(message.chat.id)
94     city = message.text.strip()
95
96     config = load_weather_config()
97     config[chat_id] = city

```

```

98     save_weather_config(config)
99
100    bot.send_message(chat_id, f"Город '{city}' выбран для отправки погоды.")
101
102
103 def get_and_send_current_weather(bot: TeleBot, message):
104     chat_id = str(message.chat.id)
105     config = load_weather_config()
106     city = config.get(chat_id)
107
108     if city:
109         weather_data = get_current_weather(city)
110         if weather_data:
111             message = format_current_weather_message(weather_data)
112             bot.send_message(chat_id, message)
113         else:
114             bot.send_message(chat_id, f"Не удалось получить текущую погоду
115                         для города '{city}'.")
116     else:
117         bot.send_message(chat_id, "Сначала установите город с помощью команды
118                         /set_city")
119
120     def register_weather_handlers(bot: TeleBot):
121         @bot.message_handler(commands=['set_city'])
122         def handle_set_city(message):
123             bot.reply_to(message, "Введите город, по которому я буду отправлять
124                         погоду.")
125             bot.register_next_step_handler(message, lambda msg: set_weather_city(
126                         bot, msg))
127
128         @bot.message_handler(commands=['current_weather'])
129         def handle_current_weather(message):
130             get_and_send_current_weather(bot, message)
131
132         @bot.message_handler(commands=['change_city'])
133         def handle_change_city(message):
134             bot.reply_to(message, "Введите новый город для получения погоды.")
135             bot.register_next_step_handler(message, lambda msg: set_weather_city(
136                         bot, msg))

```

command_handlers.py

```

1 import os
2 import pathlib
3 from telebot import TeleBot

```

```

4 from data_handler import save_data, load_banned_words
5 from filter import contains_banned_words, contains_keywords
6 from pathlib import Path
7 from weather_handler import register_weather_handlers
8
9 user_states = {}
10
11
12 def register_command_handlers(bot: TeleBot, chat_keywords, file_path):
13     # Регистрация всех командных обработчиков
14     register_weather_handlers(bot)
15
16     @bot.message_handler(commands=['start', 'help'])
17     def send_welcome(message):
18         bot.reply_to(message, (
19             "Я бот, который может сохранять и отправлять сообщения, связанные
20             с ключевыми словами. "
21             "Команды:\n"
22             "/save <ключевое слово> - сохранить сообщение или изображение\n"
23             "/edit <ключевое слово> - редактировать сообщение\n"
24             "/del <ключевое слово> - удалить сообщение\n"
25             "/list - показать все ключевые слова\n"
26             "/save_w <ключевое слово> - сохранить сообщение или изображение с
27                 возможностью уведомления\n"
28             "/edit_w <ключевое слово> - редактировать сообщение с
29                 возможностью уведомления\n"
30             "/del_w <ключевое слово> - удалить сообщение с возможностью
31                 уведомления\n"
32             "/list_w - показать все ключевые слова с возможностью уведомления
33                 \n"
34             "/weather <город> - получить текущую погоду для указанного города
35                 \n"
36             "/set_city - Установить город для прогноза погоды\n"
37             "/current_weather - Получить текущую погоду\n"
38             "/change_city - Изменить город для получения погоды\n"
39             "Когда кто-то напишет ключевое слово, я отправлю сохраненное
40             сообщение. "
41             "Я также удаляю сообщения, содержащие запрещенные слова."
42         ))
43
44     @bot.message_handler(commands=['save'])
45     def save_keyword(message):
46         try:
47             chat_id = str(message.chat.id)
48             user_id = message.from_user.id

```

```

42     keyword = message.text.split(' ', 1)[1].strip()
43
44     if chat_id not in chat_keywords:
45         chat_keywords[chat_id] = {}
46
47     if keyword in chat_keywords[chat_id]:
48         bot.reply_to(message,
49                         f"Ключевое слово '{keyword}' уже существует.
50                         Используйте команду /edit для редактирования.
51                         ")
52
53     else:
54         bot.reply_to(message,
55                         f"Ключевое слово '{keyword}' сохранено, ожидаю
56                         информацию или изображение для сохранения.")
57         user_states[(chat_id, user_id)] = ('save', keyword)
58
59     except IndexError:
60         bot.reply_to(message, "Пожалуйста, укажите ключевое слово после
61                         команды /save.")
62
63 @bot.message_handler(commands=['save_w'])
64 def save_w_keyword(message):
65     try:
66         chat_id = str(message.chat.id)
67         user_id = message.from_user.id
68         keyword = message.text.split(' ', 1)[1].strip()
69
70         if chat_id not in chat_keywords:
71             chat_keywords[chat_id] = {}
72             if 'w' not in chat_keywords[chat_id]:
73                 chat_keywords[chat_id]['w'] = {}
74
75             if keyword in chat_keywords[chat_id]['w']:
76                 bot.reply_to(message,
77                             f"Ключевое слово '{keyword}' уже существует.
78                             Используйте команду /edit_w для
79                             редактирования.")
80
81         else:
82             bot.reply_to(message,
83                             f"Ключевое слово '{keyword}' сохранено, ожидаю
84                             информацию или изображение для сохранения.")
85             user_states[(chat_id, user_id)] = ('save_w', keyword)
86
87     except IndexError:

```

```

79         bot.reply_to(message, "Пожалуйста, укажите ключевое слово после
80             команды /save_w.")
81
82     @bot.message_handler(commands=['edit'])
83     def edit_keyword(message):
84         try:
85             chat_id = str(message.chat.id)
86             user_id = message.from_user.id
87             keyword = message.text.split(' ', 1)[1].strip()
88
89             if chat_id not in chat_keywords or keyword not in chat_keywords[chat_id]:
90                 bot.reply_to(message, f"Ключевое слово '{keyword}' не найдено
91                         .")
92             else:
93                 bot.reply_to(message,
94                             f"Редактирование ключевого слова '{keyword}'.
95                             Ожидаю новое сообщение или изображение для
96                             сохранения.")
97                 user_states[(chat_id, user_id)] = ('edit', keyword)
98
99     except IndexError:
100         bot.reply_to(message, "Пожалуйста, укажите ключевое слово после
101             команды /edit.")
102
103     @bot.message_handler(commands=['edit_w'])
104     def edit_w_keyword(message):
105         try:
106             chat_id = str(message.chat.id)
107             user_id = message.from_user.id
108             keyword = message.text.split(' ', 1)[1].strip()
109
110             if chat_id not in chat_keywords or 'w' not in chat_keywords[chat_id] or keyword not in \
111                 chat_keywords[chat_id]['w']:
112                 bot.reply_to(message, f"Ключевое слово '{keyword}' не найдено
113                         .")
114             else:
115                 bot.reply_to(message,
116                             f"Редактирование ключевого слова '{keyword}' с
117                             возможностью уведомления. Ожидаю новое
118                             сообщение или изображение для сохранения.")
119                 user_states[(chat_id, user_id)] = ('edit_w', keyword)
120
121     except IndexError:

```

```

114     bot.reply_to(message, "Пожалуйста, укажите ключевое слово после
115         команды /edit_w.")
116
116 @bot.message_handler(commands=['del'])
117 def del_keyword(message):
118     try:
119         chat_id = str(message.chat.id)
120         keyword = message.text.split(' ', 1)[1].strip()
121
122         if chat_id in chat_keywords and keyword in chat_keywords[chat_id]:
123             del chat_keywords[chat_id][keyword]
124             bot.reply_to(message, f"Ключевое слово '{keyword}' удалено.")
125             save_data(chat_keywords, file_path)
126         else:
127             bot.reply_to(message, f"Ключевое слово '{keyword}' не найдено
128                         .")
129
129 except IndexError:
130     bot.reply_to(message, "Пожалуйста, укажите ключевое слово после
131         команды /del.")
131
132 @bot.message_handler(commands=['del_w'])
133 def del_w_keyword(message):
134     try:
135         chat_id = str(message.chat.id)
136         keyword = message.text.split(' ', 1)[1].strip()
137
138         if chat_id in chat_keywords and 'w' in chat_keywords[chat_id] and
139             keyword in chat_keywords[chat_id]['w']:
140             del chat_keywords[chat_id]['w'][keyword]
141             bot.reply_to(message, f"Ключевое слово '{keyword}' с
142                 возможностью уведомления удалено.")
143             save_data(chat_keywords, file_path)
144         else:
145             bot.reply_to(message, f"Ключевое слово '{keyword}' не найдено
146                         .")
147
147 except IndexError:
148     bot.reply_to(message, "Пожалуйста, укажите ключевое слово после
149         команды /del_w.")
150
150 @bot.message_handler(commands=['list'])
151 def list_keywords(message):
152     chat_id = str(message.chat.id)

```

```

151     if chat_id in chat_keywords and chat_keywords[chat_id]:
152         keywords = '\n'.join(chat_keywords[chat_id].keys())
153         bot.reply_to(message, f"Ключевые слова:\n{keywords}")
154     else:
155         bot.reply_to(message, "Нет сохраненных ключевых слов.")
156
157 @bot.message_handler(commands=['list_w'])
158 def list_w_keywords(message):
159     chat_id = str(message.chat.id)
160     if chat_id in chat_keywords and 'w' in chat_keywords[chat_id] and
161         chat_keywords[chat_id]['w']:
162         keywords = '\n'.join(chat_keywords[chat_id]['w'].keys())
163         bot.reply_to(message, f"Ключевые слова с возможностью уведомления
164             :\n{keywords}")
165     else:
166         bot.reply_to(message, "Нет сохраненных ключевых слов с
167             возможностью уведомления.")
168
169 @bot.message_handler(content_types=['photo'])
170 def handle_photo(message):
171     try:
172         chat_id = str(message.chat.id)
173         user_id = message.from_user.id
174         state = user_states.get((chat_id, user_id))
175
176         # Если пользователь в состоянии сохранения или редактирования,
177         # сохранить изображение
178         if state:
179             action, keyword = state
180             Path(f'files/{chat_id}/').mkdir(parents=True, exist_ok=True)
181             file_info = bot.get_file(message.photo[-1].file_id)
182             downloaded_file = bot.download_file(file_info.file_path)
183             src = f'files/{chat_id}/' + file_info.file_path.replace(
184                 'photos/', '')
185             with open(src, 'wb') as new_file:
186                 new_file.write(downloaded_file)
187
188             if action == 'save':
189                 chat_keywords[chat_id][keyword] = {'type': 'photo', 'path':
190                     src}
191                 bot.reply_to(message, f"Фотография для ключевого слова '{
192                     keyword}' сохранена.")
193             elif action == 'save_w':
194                 chat_keywords[chat_id]['w'][keyword] = {'type': 'photo',
195                     'path': src}

```

```

188         bot.reply_to(message,
189                         f"Фотография для ключевого слова '{keyword}'"
190                         " с возможностью уведомления сохранена.")
191
192     elif action == 'edit':
193         chat_keywords[chat_id][keyword] = {'type': 'photo', 'path':
194             ': src'}
195         bot.reply_to(message, f"Фотография для ключевого слова '{{
196             keyword}' обновлена.")
197
198     elif action == 'edit_w':
199         chat_keywords[chat_id]['w'][keyword] = {'type': 'photo',
200             'path': src}
201         bot.reply_to(message,
202                         f"Фотография для ключевого слова '{keyword}'"
203                         " с возможностью уведомления обновлена.")
204
205
206     save_data(chat_keywords, file_path)
207     del user_states[(chat_id, user_id)]
208
209 else:
210     bot.reply_to(message, "Фото получено, но не указано ключевое
211                     слово.")
212
213 except Exception as e:
214     bot.reply_to(message, f"Произошла ошибка при обработке фото: {e}")
215
216
217 @bot.message_handler(content_types=['text'])
218 def handle_text(message):
219     chat_id = str(message.chat.id)
220     user_id = message.from_user.id
221     state = user_states.get((chat_id, user_id))
222
223     banned_words = load_banned_words("banned_words.json")
224
225     if contains_banned_words(message.text, banned_words):
226         bot.delete_message(chat_id, message.message_id)
227         bot.reply_to(message, "Ваше сообщение было удалено, так как оно
228                     содержит запрещенные слова.")
229
230     return
231
232
233     if state:
234         action, keyword = state
235         if action in ['save', 'save_w']:
236             bot.reply_to(message, f"Сообщение для ключевого слова '{{
237                 keyword}' сохранено.")
238             if action == 'save':

```

```

224         chat_keywords[chat_id][keyword] = {'type': 'text', 'content': message.text}
225     elif action == 'save_w':
226         chat_keywords[chat_id]['w'][keyword] = {'type': 'text', 'content': message.text}
227     elif action in ['edit', 'edit_w']:
228         bot.reply_to(message, f"Сообщение для ключевого слова '{keyword}' обновлено.")
229     if action == 'edit':
230         chat_keywords[chat_id][keyword] = {'type': 'text', 'content': message.text}
231     elif action == 'edit_w':
232         chat_keywords[chat_id]['w'][keyword] = {'type': 'text', 'content': message.text}
233
234     save_data(chat_keywords, file_path)
235     del user_states[(chat_id, user_id)]
236
237     elif chat_id in chat_keywords:
238         for keyword, content in chat_keywords[chat_id].items():
239             if keyword in message.text:
240                 if content['type'] == 'text':
241                     bot.reply_to(message, content['content'])
242                 elif content['type'] == 'photo':
243                     with open(content['path'], 'rb') as photo:
244                         bot.send_photo(chat_id, photo)
245
246             if 'w' in chat_keywords[chat_id]:
247                 for keyword, content in chat_keywords[chat_id]['w'].items():
248                     if contains_keywords(message.text, [keyword]):
249                         if content['type'] == 'text':
250                             bot.reply_to(message, content['content'])
251                         elif content['type'] == 'photo':
252                             with open(content['path'], 'rb') as photo:
253                                 bot.send_photo(chat_id, photo)
254
255     return bot

```

data_handler.py

```

1 import json
2
3 def load_data(file_path):
4     try:
5         with open(file_path, 'r', encoding='utf-8') as file:
6             data = json.load(file)

```

```

7         print(f"Данные успешно загружены из {file_path}")
8         return data
9     except FileNotFoundError:
10        print(f"{file_path} не найден. Начало с пустого словаря.")
11        return {}
12    except json.JSONDecodeError:
13        print(f"Ошибка декодирования JSON из {file_path}. Начало с пустого
14             словаря.")
15        return {}

16 def save_data(data, file_path):
17     with open(file_path, 'w', encoding='utf-8') as file:
18         json.dump(data, file, ensure_ascii=False, indent=4)
19         print(f"Данные успешно сохранены в {file_path}")

20
21 def load_banned_words(file_path):
22     try:
23         with open(file_path, 'r', encoding='utf-8') as file:
24             banned_words = json.load(file)
25             print(f"Запрещенные слова успешно загружены из {file_path}")
26             return banned_words
27     except FileNotFoundError:
28         print(f"{file_path} не найден. Начало с пустого списка запрещенных
29             слов.")
30     return []
31     except json.JSONDecodeError:
32         print(f"Ошибка декодирования JSON из {file_path}. Начало с пустого
33             списка запрещенных слов.")
34     return []

```

config.py

```
1 API_TOKEN = '6844853072:AAE9E21vq017QAWP_9ZSNBF-0460bLBFCChM'
```

МЕСТО ДЛЯ ДИСКА