**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

МОСКОВСКОЙ ОБЛАСТИ

**«Ногинский колледж»**

**Дипломная работа**

**По теме: «Разработка чат-бота для абитуриентов ГБПОУ «Ногинский колледж» с использованием нейросетевых технологий»**

Выполнил (а) обучающийся

группы 4ИСПр1

Булаткин Даниил Алексеевич

(ф.и.о.)

Руководитель дипломной работы от ОУ

Путилов Евгений Александрович

(ф.и.о.)

Ногинск, 2025 год

Содержание

Введение

Современное образование активно интегрирует цифровые технологии, которые становятся неотъемлемой частью образовательного процесса. Одним из таких инструментов являются чат-боты, которые позволяют автоматизировать процессы взаимодействия с пользователями, предоставляя им оперативную и точную информацию. В контексте среднего профессионального образования, где абитуриенты часто сталкиваются с большим объемом информации о поступлении, учебных программах и условиях обучения, разработка чат-бота становится особенно актуальной. Это не только упрощает процесс коммуникации, но и повышает уровень доступности информации для потенциальных студентов.

Актуальность темы данной дипломной работы обусловлена необходимостью оптимизации процессов взаимодействия абитуриентов с администрацией образовательных учреждений. В условиях высокой конкуренции на рынке образовательных услуг, колледжи и техникумы стремятся сделать процесс поступления максимально комфортным и прозрачным для абитуриентов. Чат-бот, разработанный для ГБПОУ Ногинский колледж, может стать эффективным инструментом для решения этой задачи, обеспечивая круглосуточную поддержку и оперативное предоставление информации.

Теоретическая значимость работы заключается в изучении и применении современных технологий искусственного интеллекта и обработки естественного языка для создания интеллектуальных систем взаимодействия. Практическая целесообразность проявляется в возможности внедрения чат-бота в реальный образовательный процесс, что позволит снизить нагрузку на сотрудников приемной комиссии и повысить удовлетворенность абитуриентов.

Современное состояние проблемы в теоретическом аспекте связано с недостаточной изученностью применения чат-ботов в сфере среднего профессионального образования, особенно в контексте их адаптации к специфическим потребностям абитуриентов. В практическом аспекте проблема заключается в отсутствии эффективных инструментов автоматизации взаимодействия с абитуриентами, что приводит к увеличению временных затрат и снижению качества обслуживания.

Целью данной работы является разработка чат-бота для абитуриентов ГБПОУ Ногинский колледж, который будет предоставлять информацию о поступлении, учебных программах и других аспектах, связанных с обучением в колледже.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ существующих решений в области разработки чат-ботов для образовательных учреждений.

2. Изучить потребности абитуриентов и сотрудников колледжа в контексте автоматизации взаимодействия.

3. Разработать архитектуру и функциональные требования к чат-боту.

4. Реализовать прототип чат-бота с использованием современных технологий.

5. Провести тестирование и оценку эффективности разработанного решения.

Объектом исследования является процесс взаимодействия абитуриентов с администрацией ГБПОУ Ногинский колледж. Предметом исследования выступают методы и технологии автоматизации этого взаимодействия посредством разработки и внедрения чат-бота.

Таким образом, данная работа направлена на создание практического инструмента, который сможет улучшить качество обслуживания абитуриентов и оптимизировать работу сотрудников колледжа, что делает ее актуальной и значимой как с теоретической, так и с практической точки зрения.

**Теоретическая часть**

**1.1 Анализ предметной области**

Государственное бюджетное профессиональное образовательное учреждение Московской области «Ногинский колледж» (ГБПОУ МО «Ногинский колледж») представляет собой многопрофильное образовательное учреждение, осуществляющее подготовку специалистов среднего звена по различным направлениям. Основанный 31 декабря 2015 года, колледж входит в систему профессионального образования Московской области и подчиняется Министерству образования региона. Миссия учреждения сформулирована как помощь в профессиональной самореализации студентов через расширение границ их возможностей и раскрытие талантов.

**Организационно-управленческая структура колледжа**

Высшим органом управления колледжа является дирекция во главе с директором Чистовым Андреем Александровичем. В структуру стратегического управления входят восемь заместителей директора, курирующих ключевые направления деятельности:

1. Учебно-методическая работа (Соковикова Т.А.)
2. Учебно-воспитательная деятельность (Тихонова Ю.А.)
3. Учебно-производственные процессы (Сергеев Д.А.)
4. Экономика и финансы (Варламова Э.М.)
5. Административно-хозяйственные вопросы (Сапожкова Е.И.)
6. Информационные технологии (Лепихов Д.И.)
7. Безопасность (Строганов Д.А.)
8. Кадровая политика (Кирина В.Б.)

Особенностью организационной структуры является территориальная распределённость на четыре учебных корпуса:

1. Административный корпус №1 (Ногинск, ул. Ремесленная, 15)
2. Корпус №2 (Ногинск, ул. Зенитчиков, 1)
3. Корпус №3 (Ногинск, ул. Климова, 51а)
4. Корпус №4 (Балашиха, ул. Крупешина, 5)

Каждый корпус обладает собственной учебной инфраструктурой, включая методические кабинеты, библиотеки и отделы профориентации. Управление филиалами осуществляется через систему ответственных лиц, подотчётных центральной дирекции.

**Функциональные подразделения**

Ключевые структурные элементы колледжа включают:

* Учебно-методический отдел, разрабатывающий образовательные программы
* Отдел содействия трудоустройству выпускников
* Информационный отдел, ответственный за цифровизацию процессов
* Приёмную комиссию, осуществляющую набор абитуриентов

Особое значение для реализации проекта имеет информационный отдел под руководством Лепихова Д.И., который курирует внедрение цифровых технологий в образовательный процесс. Деятельность отдела включает поддержку IT-инфраструктуры, разработку программных решений и интеграцию новых технологических платформ.

**Процессы взаимодействия с абитуриентами**

Текущий процесс взаимодействия с абитуриентами базируется на традиционных каналах:

1. Телефонные консультации (8(496)514-19-80)
2. Личные встречи в приёмной комиссии
3. Электронная почта ([noginsk-priem@eduprosvet.ru](mailto:noginsk-priem@eduprosvet.ru))

График работы приёмной комиссии ограничен будними днями с 9:00 до 16:00, что создаёт сложности для занятых абитуриентов. Отсутствие автоматизированной системы ответов приводит к перегрузке сотрудников в период приёмной кампании.

**Проблемные зоны существующей системы**

Анализ текущей ситуации выявил три ключевые проблемы:

1. Низкая доступность информации в нерабочее время
2. Дублирование однотипных запросов
3. Отсутствие системы предварительной фильтрации вопросов
4. Ограниченные возможности удалённого консультирования

Эти факторы обуславливают необходимость внедрения чат-бота как элемента цифровой трансформации процессов взаимодействия с абитуриентами.

**Технологическая инфраструктура колледжа**

Информационный отдел колледжа под руководством Лепихова Д.И. обеспечивает функционирование:

* Корпоративной сети с объединением всех корпусов
* Системы электронного документооборота
* Официального сайта (https://nogkolledzh.ru)
* Локальных баз данных студентов и преподавателей

Отсутствие интеграции чат-бота в существующую IT-инфраструктуру потребует разработки специализированных API-интерфейсов для синхронизации данных с приёмной комиссией и учебными отделами.

**Перспективы внедрения чат-бота**

Разрабатываемый чат-бот должен стать частью экосистемы цифровых сервисов колледжа, взаимодействуя с:

1. Базами данных образовательных программ
2. Системой электронной записи
3. Календарём приёмной кампании
4. Модулем онлайн-тестирования

Особое внимание потребуется уделить интеграции с подсистемой безопасности Строганова Д.А. для обеспечения защиты персональных данных пользователей.

**Ожидаемые эффекты**

Внедрение нейросетевого чат-бота позволит:

* Увеличить охват абитуриентов на 40-50%
* Сократить нагрузку на сотрудников приёмной комиссии
* Обеспечить круглосуточную поддержку поступающих
* Автоматизировать сбор статистики запросов
* Персонализировать образовательные траектории

Реализация проекта потребует тесного взаимодействия разработчиков с учебно-методическим отделом (Соковикова Т.А.) для актуализации образовательного контента и с отделом профориентации для настройки сценариев взаимодействия.

# 1.2 Техническое задание на разработку чат-бота для абитуриентов ГБПОУ «Ногинский колледж»

## 1. Цель создания продукта

Разработать интеллектуального чат-бота на базе технологий OpenAI для автоматизированного круглосуточного консультирования абитуриентов ГБПОУ «Ногинский колледж». Чат-бот будет предоставлять актуальную информацию о поступлении, учебных программах и других аспектах обучения, обеспечивая удобный и персонализированный диалог с пользователями через мессенджер Telegram.

## 2. Состав подсистем и функциональные задачи

## Основные подсистемы:

* **Модуль обработки естественного языка (NLP)**  
  Использует API OpenAI для понимания запросов и генерации ответов на русском языке с учётом контекста диалога.
* **Интерфейс взаимодействия с пользователем**  
  Чат-бот, интегрированный в мессенджер Telegram, обеспечивающий удобный доступ к функционалу на мобильных и десктопных устройствах.
* **Подсистема управления базой знаний**  
  Хранит и обновляет структурированную информацию о поступлении, специальностях, расписании и правилах колледжа.
* **Административный модуль**  
  Позволяет сотрудникам колледжа обновлять базу знаний, просматривать статистику обращений и корректировать сценарии диалога.

## Функциональные задачи:

* Приём и обработка вопросов абитуриентов в режиме реального времени через Telegram.
* Предоставление точной и актуальной информации по тематике поступления и обучения.
* Поддержка FAQ и сценариев для типовых запросов.
* Возможность передачи сложных вопросов оператору при необходимости.
* Сбор и анализ статистики запросов для улучшения качества обслуживания.

## 3. Требования к подсистемам

* **NLP-модуль** должен обеспечивать высокую точность распознавания запросов и генерации ответов с учётом контекста диалога, используя возможности OpenAI API.
* **Интерфейс пользователя** реализуется как Telegram-бот с удобным и интуитивно понятным интерфейсом.
* **База знаний** должна быть структурирована и легко обновляемая сотрудниками колледжа без необходимости программирования.
* **Административный модуль** должен иметь защищённый доступ с разграничением прав пользователей.

## 4. Требования к информационной базе, программному и техническому обеспечению

* Использование облачного API OpenAI для обработки естественного языка и генерации ответов.
* Хранение базы знаний и логов взаимодействия в защищённой базе данных с резервным копированием.
* Серверная часть должна обеспечивать стабильное соединение с API OpenAI и поддерживать одновременную работу не менее 100 пользователей.
* Интеграция с Telegram Bot API для обмена сообщениями с пользователями.
* Обеспечение безопасности данных пользователей в соответствии с законодательством РФ о персональных данных.

## 5. Общие требования к проектируемой системе

* Круглосуточная доступность чат-бота с минимальной задержкой ответа (не более 2 секунд).
* Высокая отказоустойчивость и масштабируемость системы.
* Интуитивно понятный интерфейс для пользователей и администраторов.
* Возможность дальнейшего расширения функционала (например, добавление голосового ввода).
* Соответствие требованиям информационной безопасности и конфиденциальности.

## 6. Перечень задач создания системы и исполнителей

| **Задача** | **Исполнитель** |
| --- | --- |
| Анализ требований и проектирование | Студент, научный руководитель |
| Разработка архитектуры системы | Студент, IT-специалисты колледжа |
| Интеграция с OpenAI API | Студент, разработчики |
| Создание Telegram-бота | Студент, веб-разработчики |
| Настройка базы знаний и административного модуля | Сотрудники колледжа, студент |
| Тестирование и отладка | Студент, IT-отдел колледжа |
| Внедрение и сопровождение | IT-отдел колледжа |

## 7. Этапы создания системы и сроки выполнения

| **Этап** | **Сроки выполнения** | **Результат** |
| --- | --- | --- |
| Сбор и анализ требований | 1 неделя | Документ с требованиями |
| Проектирование архитектуры | 2 недели | Технический проект |
| Разработка и интеграция OpenAI API | 3 недели | Рабочий прототип |
| Создание Telegram-бота и базы знаний | 2 недели | Полнофункциональный интерфейс и база знаний |
| Тестирование и исправление ошибок | 2 недели | Отчёт о тестировании |
| Внедрение и запуск | 1 неделя | Рабочая система |

Данное техническое задание отражает использование OpenAI API как ядра интеллектуального чат-бота с интеграцией в Telegram, что обеспечивает удобство доступа для абитуриентов и эффективное взаимодействие с приёмной комиссией колледжа.

# 1.3 Анализ используемого программного и технического обеспечения

## Программное обеспечение (ПО)

В процессе разработки чат-бота для абитуриентов ГБПОУ «Ногинский колледж» использовался следующий набор программных средств:

* **PyCharm** — интегрированная среда разработки (IDE) для Python, обеспечивающая удобное написание, отладку и тестирование кода чат-бота.
* **GitHub Desktop** — клиент для работы с системой контроля версий Git, позволяющий управлять исходным кодом и вести совместную разработку.
* **Mozilla Firefox** — веб-браузер, использовавшийся для тестирования веб-интерфейсов, проверки документации и взаимодействия с онлайн-сервисами.
* **Telegram Desktop** — официальный клиент Telegram, применялся для тестирования функционала чат-бота и отладки взаимодействия с пользователями.
* **Postman** — инструмент для тестирования API-запросов, в частности для проверки корректности взаимодействия с OpenAI API и Telegram Bot API.
* **Python-библиотеки**:
  + aiogram — библиотека для удобной работы с Telegram Bot API.
  + openai — библиотека для взаимодействия с OpenAI API.
  + requests — для отправки HTTP-запросов.
  + dotenv — для управления конфиденциальными данными (ключи API).

Данный набор ПО обеспечивает полный цикл разработки, тестирования и поддержки чат-бота, а также интеграцию с внешними сервисами OpenAI и Telegram.

## Техническое обеспечение (ТО)

Разработка и тестирование чат-бота выполнялись на персональном компьютере со следующими характеристиками:

* **Процессор:** AMD Ryzen 5 2600
* **Оперативная память:** 16 ГБ DDR4
* **Графический процессор:** NVIDIA GeForce GTX 1060
* **Операционная система:** Windows 10

Для размещения серверной части чат-бота используется **локальный сервер**, принадлежащий колледжу, что обеспечивает контроль над инфраструктурой и безопасность данных. В колледже имеется выделенный сервер и IT-инфраструктура, поддерживающая работу цифровых сервисов.

Связь с Telegram API и OpenAI API осуществляется через стабильное интернет-соединение.На данный момент дополнительное оборудование для обеспечения безопасности (например, VPN или аппаратные средства шифрования) не используется, но может быть рассмотрено в дальнейшем.

# 1.4 Проектирование программного продукта

## Концептуальная модель программного продукта

Для разработки чат-бота для абитуриентов ГБПОУ «Ногинский колледж» была создана концептуальная модель, отражающая основные бизнес-процессы, информационные потоки и функциональные компоненты системы.

Основная задача чат-бота — автоматизация консультаций абитуриентов по вопросам поступления, учебных программ и организации учебного процесса через мессенджер Telegram. Взаимодействие пользователя с ботом основано на обработке естественного языка с помощью API OpenAI.

## Анализ информационной модели и бизнес-процессов

* **Пользователь (абитуриент)** отправляет запрос в Telegram-бот с вопросом о поступлении, специальностях, документах и пр.
* **Чат-бот** принимает запрос, формирует запрос к OpenAI API, получает ответ и отправляет его пользователю.
* **Администратор** может обновлять базу знаний и просматривать статистику обращений (планируемая функциональность).

В рамках проекта база данных в классическом понимании не создаётся, так как основная логика обработки запросов реализуется через API OpenAI. Однако локально сохраняются логи взаимодействий и настройки бота.

## Основные объекты и данные:

* **Пользователь**: уникальный идентификатор Telegram, история сообщений.
* **Запрос**: текст сообщения пользователя.
* **Ответ**: текст ответа, сформированный нейросетью.
* **Настройки бота**: параметры работы (температура генерации, максимальная длина ответа).
* **Статистика**: количество запросов, частота тематики.

## Логическая модель системы

Логическая модель отражает структуру компонентов и их взаимодействие.

* **Модуль взаимодействия с Telegram API** — принимает и отправляет сообщения.
* **Модуль обработки запросов** — формирует запросы к OpenAI API.
* **Модуль интеграции с OpenAI API** — отправляет запросы и получает ответы.
* **Модуль логирования** — сохраняет историю диалогов.
* **Административный модуль** (планируется) — управление настройками и анализ статистики.

## Диаграммы

Для формализации модели можно использовать:

* **Диаграмму прецедентов (Use Case)**, показывающую взаимодействие пользователя и администратора с системой.
* **Диаграмму компонентов**, отражающую модули и их связи.
* **Диаграмму последовательности**, демонстрирующую процесс обработки запроса от пользователя до получения ответа.

## 1. Диаграмма прецедентов (Use Case)

@startuml

actor Абитуриент

actor Администратор

Абитуриент --> (Отправить запрос в чат-бот)

(Отправить запрос в чат-бот) --> (Получить ответ от чат-бота)

Администратор --> (Обновить базу знаний)

Администратор --> (Просмотреть статистику обращений)

@enduml

## 2. Диаграмма компонентов

@startuml

package "Telegram" {

[Telegram API]

}

package "Чат-бот" {

[Модуль взаимодействия с Telegram API]

[Модуль обработки запросов]

[Модуль интеграции с OpenAI API]

[Модуль логирования]

[Административный модуль]

}

package "OpenAI" {

[OpenAI API]

}

[Telegram API] --> [Модуль взаимодействия с Telegram API]

[Модуль взаимодействия с Telegram API] --> [Модуль обработки запросов]

[Модуль обработки запросов] --> [Модуль интеграции с OpenAI API]

[Модуль интеграции с OpenAI API] --> [OpenAI API]

[Модуль обработки запросов] --> [Модуль логирования]

[Административный модуль] --> [Модуль логирования]

@enduml

## 3. Диаграмма последовательности (Sequence)

@startuml

actor Абитуриент

participant "Telegram Bot" as Bot

participant "OpenAI API" as OpenAI

Абитуриент -> Bot : Отправить сообщение

Bot -> Bot : Обработать сообщение

Bot -> OpenAI : Отправить запрос с текстом сообщения

OpenAI --> Bot : Вернуть сгенерированный ответ

Bot -> Абитуриент : Отправить ответ

@enduml

## Пример упрощённой логической структуры (компоненты):

[Telegram API] <--> [Модуль взаимодействия] <--> [Модуль обработки запросов] <--> [OpenAI API]

|

v

[Модуль логирования]

|

v

[Административный модуль]

**Практическая часть**

# Создание программного продукта с использованием специализированного программного обеспечения

## Этап 1: Установка и настройка интегрированной среды разработки PyCharm

Первым шагом в процессе создания чат-бота для абитуриентов ГБПОУ «Ногинский колледж» стала установка и настройка интегрированной среды разработки PyCharm. PyCharm представляет собой одну из наиболее популярных и мощных IDE для разработки на языке Python, предоставляющую комплексные инструменты для написания, тестирования и отладки кода. Выбор именно этой среды разработки обусловлен её широкими возможностями автодополнения кода, встроенными инструментами отладки и удобным интерфейсом для работы с виртуальными окружениями.

Процесс установки PyCharm включал загрузку дистрибутива с официального сайта JetBrains и выполнение стандартной процедуры инсталляции. При установке были выбраны опции создания ярлыков на рабочем столе и ассоциации файлов с расширением .py с данной IDE. После завершения установки была выполнена первоначальная настройка среды разработки, включающая выбор темы оформления и базовых параметров редактора кода.

## Этап 2: Создание нового проекта в PyCharm

После успешной установки PyCharm был создан новый проект для разработки Telegram-бота. При создании проекта была выбрана опция "New Project", что позволило настроить проект с нуля в соответствии с требованиями технического задания. Структура проекта была организована таким образом, чтобы обеспечить логическое разделение компонентов системы и упростить дальнейшую разработку и сопровождение кода.

В процессе создания проекта особое внимание было уделено выбору расположения файлов проекта и настройке базовых параметров. Проект был размещён в отдельной директории на локальном диске, что обеспечивает изоляцию от других разработческих проектов и упрощает управление файлами.

## Этап 3: Создание виртуального окружения Python

Следующим критически важным этапом стало создание изолированного виртуального окружения для проекта. Виртуальное окружение представляет собой изолированную среду Python, которая позволяет устанавливать и использовать различные версии пакетов для разных проектов без конфликтов между зависимостями. Это особенно важно при разработке приложений, использующих специфические библиотеки, такие как aiogram и openai.

Для создания виртуального окружения была использована стандартная команда python -m venv mybotnew. Данная команда создаёт новое виртуальное окружение в директории с именем "mybotnew", используя встроенный модуль venv. Модуль venv является частью стандартной библиотеки Python 3.3+ и не требует дополнительной установки. Команда python -m venv инструктирует Python запустить модуль venv как скрипт, где флаг -m указывает на запуск модуля.

Создание виртуального окружения обеспечивает несколько ключевых преимуществ для проекта разработки чат-бота:

* Изоляция зависимостей проекта от системных пакетов Python
* Возможность использования конкретных версий библиотек без влияния на другие проекты
* Упрощение развёртывания приложения на различных системах
* Обеспечение воспроизводимости среды разработки

## Этап 4: Настройка интерпретатора Python в PyCharm

После создания виртуального окружения возникла необходимость ручной настройки интерпретатора Python в среде разработки PyCharm. Эта процедура требовалась для того, чтобы IDE корректно работала с созданным виртуальным окружением и имела доступ к устанавливаемым в него библиотекам.

Процесс настройки интерпретатора включал следующие шаги:

1. Открытие настроек проекта через меню File → Settings → Project → Python Interpreter
2. Нажатие на кнопку "Add Interpreter" для добавления нового интерпретатора
3. Выбор опции "Add Local Interpreter" для указания локального интерпретатора
4. Указание пути к исполняемому файлу python.exe в созданном виртуальном окружении

Правильная настройка интерпретатора критически важна для корректной работы среды разработки. Она обеспечивает правильное автодополнение кода, корректную работу отладчика и доступ к установленным в виртуальном окружении библиотекам. Без этой настройки PyCharm не сможет корректно анализировать код и предоставлять разработчику необходимые инструменты для эффективной работы.

## Этап 5: Решение проблем с установкой pip

В процессе настройки среды разработки возникла проблема с доступностью пакетного менеджера pip. Pip (Pip Installs Packages) является стандартным инструментом для установки пакетов Python и обычно устанавливается автоматически вместе с интерпретатором Python. Однако в некоторых случаях, особенно при использовании определённых дистрибутивов Python или при неправильной настройке переменных окружения, pip может быть недоступен.

Проблема с недоступностью pip может возникать по нескольким причинам:

* Неправильная настройка переменной PATH в операционной системе
* Использование дистрибутива Python, который не включает pip по умолчанию
* Конфликты между различными версиями Python, установленными в системе

Для решения этой проблемы потребовалась ручная установка pip. Процесс ручной установки включал загрузку скрипта get-pip.py с официального сайта и его выполнение с помощью интерпретатора Python. Эта процедура обеспечила корректную установку пакетного менеджера и его интеграцию с используемой версией Python.

## Этап 6: Установка необходимых библиотек для проекта

После решения проблем с pip была предпринята попытка установки необходимых для проекта библиотек с помощью команды pip install aiogram openai. Эта команда должна была установить две ключевые библиотеки:

* **aiogram** - мощная асинхронная библиотека для создания Telegram-ботов на Python
* **openai** - официальная библиотека для взаимодействия с API OpenAI

Библиотека aiogram предоставляет удобный интерфейс для работы с Telegram Bot API и поддерживает асинхронное программирование, что критически важно для обеспечения высокой производительности чат-бота. Она включает в себя инструменты для обработки различных типов сообщений, управления состояниями диалога и интеграции с внешними сервисами.

Библиотека openai необходима для взаимодействия с API OpenAI, который обеспечивает возможности обработки естественного языка и генерации ответов на запросы пользователей. Эта библиотека предоставляет удобные методы для отправки запросов к различным моделям OpenAI и обработки полученных ответов.

## Этап 7: Решение проблем с автоматической установкой библиотек

Несмотря на корректную настройку pip, автоматическая установка библиотек через команду pip install не увенчалась успехом. Такие проблемы могут возникать по различным причинам:

* Конфликты зависимостей между различными версиями пакетов
* Проблемы с сетевым подключением к репозиториям PyPI
* Ограничения прав доступа в операционной системе
* Конфликты с уже установленными версиями библиотек

Для решения этой проблемы была выполнена ручная установка необходимых библиотек. Ручная установка библиотек включает несколько альтернативных подходов:

1. **Загрузка исходного кода библиотек** с официальных репозиториев GitHub или с сайта PyPI
2. **Использование установочных файлов setup.py** для компиляции и установки библиотек из исходного кода
3. **Применение альтернативных менеджеров пакетов** таких как easy\_install

Процесс ручной установки потребовал выполнения команды python setup.py install в директории каждой загруженной библиотеки. Эта команда запускает скрипт установки, который компилирует исходный код библиотеки и размещает её файлы в соответствующих директориях виртуального окружения.

## Заключение этапа настройки среды разработки

Успешное завершение всех описанных выше этапов обеспечило создание полнофункциональной среды разработки для проекта чат-бота. Настроенная среда включает:

* Интегрированную среду разработки PyCharm с корректно настроенным интерпретатором Python
* Изолированное виртуальное окружение для предотвращения конфликтов зависимостей
* Установленный и функционирующий пакетный менеджер pip
* Необходимые библиотеки aiogram и openai для реализации функционала чат-бота

Данная конфигурация обеспечивает надёжную основу для дальнейшей разработки, тестирования и отладки программного продукта. Правильно настроенная среда разработки критически важна для эффективного процесса создания программного обеспечения и минимизации потенциальных проблем на последующих этапах разработки.

## Этап 8: Написание основного кода чат-бота с интеграцией ИИ

После успешной настройки среды разработки и установки необходимых библиотек был приступлен к созданию основного кода Telegram-бота для абитуриентов ГБПОУ «Ногинский колледж». Данный этап является ключевым в процессе разработки, поскольку именно здесь реализуется вся логика взаимодействия бота с пользователями и внешними сервисами искусственного интеллекта.

## Импорт необходимых модулей и библиотек

Первым шагом в создании кода стал импорт всех необходимых модулей и библиотек. Процесс начался с подключения ключевых компонентов библиотеки aiogram, которая представляет собой мощную асинхронную библиотеку для создания Telegram-ботов на Python. Библиотека aiogram 3.x обеспечивает удобный интерфейс для работы с Telegram Bot API и поддерживает асинхронное программирование, что критически важно для обеспечения высокой производительности чат-бота при одновременной обработке множества запросов.

Импорт включал следующие основные компоненты:

* **Bot** — основной класс для взаимодействия с API Telegram
* **Dispatcher** — менеджер событий, который управляет поступающими от Telegram сообщениями
* **types** — модуль, содержащий типы данных для сообщений и других объектов
* **Command** — фильтр для обработки команд от пользователя
* **Message** — тип данных для работы с сообщениями

Дополнительно был подключён модуль **asyncio** для работы с асинхронными функциями, а также библиотека **openai** для интеграции с API искусственного интеллекта.

## Конфигурация токенов и API-ключей

Следующим важным этапом стала настройка конфиденциальных данных, необходимых для работы бота. Процесс включал конфигурацию двух типов ключей доступа:

**Токен Telegram-бота** — уникальный идентификатор, полученный от BotFather в Telegram. Этот токен обеспечивает аутентификацию бота в системе Telegram и предоставляет доступ к Bot API для отправки и получения сообщений. Токен представляет собой строку формата "числа:буквенно-цифровая последовательность" и является критически важным элементом безопасности системы.

**API-ключ для сервиса AIML API** — специальный ключ доступа к внешнему сервису, обеспечивающему функциональность искусственного интеллекта. Сервис AIML API представляет собой революционную платформу, предоставляющую доступ к более чем 100 моделям ИИ через единый интерфейс. Переход от стандартного OpenAI к AIML API осуществляется с помощью всего одной строки кода, что делает интеграцию быстрой и простой.

## Инициализация клиента OpenAI-совместимого API

Для обеспечения взаимодействия с языковой моделью был инициализирован клиент OpenAI, настроенный на работу с внешним сервисом AIML API. Данная конфигурация позволяет использовать стандартную библиотеку OpenAI для Python, но направлять запросы к альтернативному провайдеру услуг ИИ. Это решение обеспечивает гибкость в выборе поставщика услуг искусственного интеллекта при сохранении привычного интерфейса разработки.

Конфигурация включала указание базового URL сервиса AIML API и соответствующего ключа аутентификации. Такой подход позволяет легко переключаться между различными провайдерами ИИ-услуг без существенных изменений в коде приложения.

## Инициализация объектов бота и диспетчера

После настройки доступов был создан основной объект бота с использованием токена Telegram и настройкой параметров по умолчанию. Особое внимание было уделено конфигурации **DefaultBotProperties**, которая определяет режим парсинга сообщений. Выбор HTML-режима парсинга обеспечивает возможность форматирования текста в ответах бота, включая использование жирного шрифта, курсива и других элементов разметки.

Диспетчер (**Dispatcher**) был инициализирован для управления обработкой входящих событий и сообщений. Диспетчер представляет собой центральный компонент системы, который принимает входящие обновления от Telegram API и направляет их к соответствующим обработчикам на основе настроенных фильтров и маршрутов.

## Реализация обработчика команды /start

Первым функциональным компонентом бота стал обработчик команды /start, который является стандартной точкой входа для взаимодействия пользователей с Telegram-ботами. Обработчик реализован как асинхронная функция с использованием декоратора @dp.message(Command("start")), что обеспечивает автоматическое срабатывание при получении соответствующей команды от пользователя.

Функция отправляет приветственное сообщение "Привет! Я чат-бот Ногинского колледжа. Чем могу помочь?", которое информирует пользователя о назначении бота и предлагает начать взаимодействие. Использование асинхронного подхода с ключевым словом async и await обеспечивает неблокирующую обработку запросов, позволяя боту одновременно обслуживать множество пользователей.

## Создание универсального обработчика сообщений

Центральным элементом функциональности бота стал универсальный обработчик всех текстовых сообщений, реализованный через декоратор @dp.message() без дополнительных фильтров. Этот обработчик активируется при получении любого текстового сообщения, не являющегося командой, и обеспечивает основную логику взаимодействия с искусственным интеллектом.

Обработчик построен по принципу "запрос-ответ", где каждое сообщение пользователя передаётся для анализа языковой модели, которая генерирует соответствующий ответ на основе контекста и обученных данных. Такой подход обеспечивает естественное взаимодействие пользователя с ботом без необходимости изучения специальных команд или синтаксиса.

## Интеграция с моделью Mistral-7B

Ключевой особенностью реализации стало использование языковой модели **Mistral-7B-Instruct-v0.2** для генерации ответов. Mistral 7B представляет собой языковую модель с 7.3 миллиардами параметров, разработанную компанией Mistral AI, которая превосходит модель Llama 2 13B по всем показателям и приближается к производительности CodeLlama 7B в задачах, связанных с кодом.

Модель использует передовые механизмы **Grouped-query attention (GQA)** для ускорения вывода и **Sliding Window Attention (SWA)** для эффективной обработки длинных последовательностей текста. Это обеспечивает высокое качество генерируемых ответов при относительно низких вычислительных требованиях.

## Конфигурация параметров генерации

Для обеспечения оптимального качества ответов были настроены ключевые параметры генерации текста:

**Параметр temperature (0.7)** — определяет степень "творчества" или вариативности в ответах модели. Значение 0.7 обеспечивает баланс между детерминированностью и креативностью, позволяя модели генерировать естественные и разнообразные ответы, избегая при этом чрезмерной непредсказуемости. Температура влияет на выбор слов и фраз, делая текст более "живым" и менее машинным.

**Параметр max\_tokens (256)** — ограничивает максимальную длину генерируемого ответа. Это значение выбрано для обеспечения достаточно подробных ответов при сохранении читаемости в формате мессенджера и контроле расходов на API-запросы.

## Система ролей и контекста

Для обеспечения релевантности ответов была реализована система ролей, включающая:

**Системная роль (system)** — определяет контекст и специализацию бота через сообщение "Вы помогаете абитуриентам Ногинского колледжа". Это обеспечивает фокусировку ответов модели на тематике поступления и обучения в колледже.

**Пользовательская роль (user)** — содержит фактический текст запроса от пользователя. Каждое сообщение обрабатывается как отдельный запрос с сохранением контекста специализации.

## Обработка ошибок и исключений

Критически важным элементом надёжности системы стала реализация комплексной обработки ошибок через конструкцию try-except. Данный механизм обеспечивает graceful degradation — корректное поведение системы при возникновении непредвиденных ситуаций.

Обработка исключений покрывает различные типы потенциальных проблем:

* Сетевые ошибки при обращении к API
* Превышение лимитов запросов или квот
* Временная недоступность внешних сервисов
* Ошибки парсинга или валидации данных

При возникновении любой ошибки пользователь получает информативное сообщение "Произошла ошибка при обработке вашего запроса. Попробуйте позже.", что обеспечивает понятную обратную связь и не прерывает пользовательский опыт.

## Реализация главной функции запуска

Завершающим элементом кода стала функция main(), реализующая механизм **long polling** для получения обновлений от Telegram API. Long polling представляет собой метод получения обновлений, при котором бот периодически опрашивает серверы Telegram на предмет новых сообщений.

Функция dp.start\_polling(bot) запускает бесконечный цикл обработки входящих сообщений, обеспечивая непрерывную работу бота. Асинхронная природа этого процесса позволяет эффективно обрабатывать множество одновременных запросов без блокировки выполнения.

## Точка входа программы

Конструкция if \_\_name\_\_ == '\_\_main\_\_': обеспечивает корректный запуск бота только при прямом выполнении файла, предотвращая нежелательное поведение при импорте модуля. Вызов asyncio.run(main()) инициализирует асинхронное выполнение главной функции в рамках event loop.

## Заключение этапа разработки кода

Созданный код представляет собой полнофункциональный Telegram-бот, интегрированный с современными технологиями искусственного интеллекта через сервис AIML API. Архитектура решения обеспечивает:

* Высокую производительность благодаря асинхронной обработке запросов
* Надёжность через комплексную систему обработки ошибок
* Качественные ответы за счёт использования продвинутой языковой модели Mistral-7B
* Масштабируемость для обслуживания множества одновременных пользователей
* Простоту сопровождения благодаря чёткой структуре кода

Данная реализация создаёт прочную основу для дальнейшего развития функционала бота и интеграции дополнительных возможностей для обслуживания абитуриентов ГБПОУ «Ногинский колледж».