МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

МЕЖИНСТИТУТСКАЯ БАЗОВАЯ КАФЕДРА

**КУРСОВОЙ ПРОЕКТ**

по дисциплине

«Междисциплинарный проектный практикум»

на тему:

«Разработка telegram бота с распознаванием речи при помощи Python и с последующим переводом в текст на основе нейронных сетей»

|  |  |
| --- | --- |
|  | **Выполнил:**  Магомедов Имран Борисович  студент 2 курса  группы ПИЖ-б-о-22-1  направления подготовки 09.03.04 «Программная инженерия»  направленность (профиль) «Разработка и сопровождение программного обеспечения»  очной формы обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  **Руководитель работы:**  И. В. Свистунов, профессор межинститутской базовой кафедры |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Работа допущена к защите | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись руководителя) | | \_\_\_\_\_\_\_\_\_\_\_\_\_  (дата) |
| Работа выполнена и  Защищена с оценкой | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | Дата защиты \_\_\_\_\_\_\_\_\_\_\_ | |
| Члены комиссии:  зав. межинститутской  базовой кафедрой | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. Н. Новикова  (подпись) | | | |
| доцент МИБК | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. В. Свистунов  (подпись) | | | |
| доцент МИБК | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ З.М. Альбекова  (подпись) | | | |

Ставрополь, 2024 г.

Министерство науки и высшего образования и Российской Федерации Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Цифрового развития\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра межинститутская базовая \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление подготовки 09.03.04 «Программная инженерия» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность (профиль) «Разработка и сопровождение программного обеспечения»\_\_\_

ЗАДАНИЕ

на курсовую работу

|  |  |  |
| --- | --- | --- |
| студента | Магомедов Имран Борисович | |
| по дисциплине | | (фамилия, имя, отчество)  «Междисциплинарный проектный практикум» | |
| 1. Тема работы: разработка мобильного приложения для учёта клиентов фитнес-тренера\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
| 2. Цель: повышение уровня профессиональной подготовки путем углубления и закрепления теоретических знаний и практических навыков, приобретенных в результате изучения дисциплины «Междисциплинарный проектный практикум»; подготовка к самостоятельной разработке программного обеспечения с использованием современных информационных технологий.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
| 3. Задачи:  3.1 Анализ прикладной задачи и методов ее решения. Обоснование выбора средств, технологий и алгоритмов решения прикладной задачи.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  3.2 Разработка алгоритмов решения задачи.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  3.3 Реализация программного кода \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  3.4 Отладка и тестирование программного кода\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
| 4. Перечень подлежащих разработке вопросов:  а) теоретической части: изучение и анализ литературы, постановка условия задачи, выбор и описание методов и библиотек для ее решения и технологий, среды программирования  б) проектная часть: проектирование UML-диаграмм, IDEF, интерфейса и пр.  в) реализация: описание структуры проекта, описание файлов проекта, разработка программного кода, тестирование программы | | | |
| 5. Исходные данные:  а) по литературным источникам: ГОСТы, международные стандарты, программные средства, используемые при разработке программного обеспечения  б) исходные данные, подготовленные для тестирования объекта профессиональной деятельности (информационной системы/приложения/программного продукта)  6. Список рекомендуемой литературы: монографии, диссертации, научные статьи, учебнометодические материалы, ссылки на официальные сайты, содержащие информацию необходимую для решения поставленных в работе задач (перечислить 4 – 5 основных литературных источников, которые применялись при выполнении КР) | | | |

7. Контрольне сроки представления отдельных разделов курсового проекта:

25 % - представление первого раздела «07» марта 2024 г.

50 % - представление второго раздела «15» апреля 2024 г.

75 % - представление третьего раздела «29» апреля 2024 г.

100 % - представление работы на отзыв «10» мая 2024 г.

8. Срок защиты студентом курсового проекта «23» мая 2024 г.

Дата выдачи задания «05» февраля 2024 г.

|  |  |  |
| --- | --- | --- |
| Руководитель курсовой работы |  |  |
| К. технических н., доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | И. В. Свистунов |
| *(учебная степень, звание)* | *(личная подпись)* | *(инициалы, фамилия)* |
| Задание принял к исполнению студент очной формы обучения 3 курса  группы ПИЖ-б-о-21-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Магомедов И.Б\_\_ | | |
| *(личная подпись) (инициалы, фамилия)* . | | |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc169001019)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 6](#_Toc169001020)

[1.1 Предметная область 6](#_Toc169001021)

[1.2 Выявление проблем 6](#_Toc169001022)

[1.3 Техническое задание 7](#_Toc169001023)

[1.3.1 Требования к функциональным характеристикам программы 8](#_Toc169001024)

[1.3.2 Требования к техническим характеристикам программы 8](#_Toc169001025)

[1.3.3 Требования к программным характеристикам программы 9](#_Toc169001026)

[1.3.4 Этапы разработки программы и график выполнения работ 9](#_Toc169001027)

[1.4 Технологии для разработки программы 10](#_Toc169001028)

[1.4.1 Среда разработки 10](#_Toc169001029)

[1.4.2 Языки программирования и фреймворки 10](#_Toc169001030)

[1.4.3 Библиотеки 10](#_Toc169001031)

[1.4.4 Инструменты для контроля версий и совместной работы 11](#_Toc169001032)

[1.4.5 Нейронные сети и алгоритмы 11](#_Toc169001033)

[Вывод 12](#_Toc169001034)

[2 ПРОЕКТНАЯ ЧАСТЬ 13](#_Toc169001035)

[2.1 Алгоритм работы программы. 13](#_Toc169001036)

[2.2. Стандарт разработки 13](#_Toc169001037)

[2.2.1 Проектирование: 13](#_Toc169001038)

[2.2.2 Кодирование: 14](#_Toc169001039)

[2.2.3 Тестирование: 14](#_Toc169001040)

[2.2.4 Документирование: 14](#_Toc169001041)

[2.3. Проектирование структуры программы 15](#_Toc169001042)

[2.4. Блок-схемы 15](#_Toc169001043)

[2.5. UML-диаграмма классов 19](#_Toc169001044)

[2.6. Диаграмма прецедентов 20](#_Toc169001045)

[Вывод 22](#_Toc169001046)

[3. ПРАКТИЧЕСКАЯ ЧАСТЬ 23](#_Toc169001047)

[3.1 Реализация программы 23](#_Toc169001048)

[3.1.1 Основные компоненты программы 23](#_Toc169001049)

[3.1.2 Основные функции программы 23](#_Toc169001050)

[3.1.3 Реализация базы данных 24](#_Toc169001051)

[3.2 Описание готового продукта 24](#_Toc169001052)

[3.2.1 Пользовательский интерфейс 25](#_Toc169001053)

[3.2.2 Методы, функции и файлы. 29](#_Toc169001054)

[3.3 Руководство для пользователя 37](#_Toc169001055)

[Вывод 38](#_Toc169001056)

[ЗАКЛЮЧЕНИЕ 39](#_Toc169001057)

[СПИСОК ЛИТЕРАТУРЫ 40](#_Toc169001058)

[ПРИЛОЖЕНИЕ 41](#_Toc169001059)

[Приложение A 41](#_Toc169001060)

# ВВЕДЕНИЕ

Современные технологии стремительно развиваются, и искусственный интеллект становится неотъемлемой частью нашего повседневного взаимодействия с цифровыми устройствами. Одной из наиболее востребованных областей применения этих технологий является распознавание речи, которое значительно упрощает взаимодействие пользователей с различными сервисами и устройствами. В этой связи, создание telegram ботов с функцией распознавания речи и преобразования её в текст открывает новые возможности для автоматизации задач и повышения удобства использования мессенджеров.

В данной работе будет использован готовый API от EDENAI для реализации функции распознавания речи, что позволит сосредоточиться на интеграции этой технологии в telegram бота и обеспечить высокое качество работы приложения.

Объект исследования: процессы распознавания и преобразования речи в текст с использованием готовых API в telegram ботах.

Предмет исследования: разработка telegram бота с функцией распознавания речи с использованием Python и готового API от EDENAI для последующего перевода речи в текст.

Цель работы: разработка telegram бота, который сможет эффективно распознавать речь пользователя и преобразовывать её в текст, обеспечивая тем самым улучшенное взаимодействие пользователей с мессенджером.

Для достижения этой цели необходимо решить следующие задачи:

* Определить требования к функциональности и архитектуре telegram бота.
* Изучить возможности и функционал API от EDENAI для распознавания речи.
* Интегрировать готовый API в telegram бота с использованием языка программирования Python.

Методы исследования

В процессе разработки будут использованы следующие методы:

* Проектирование и моделирование. Создание прототипа telegram бота, разработка архитектуры.
* Программирование. Реализация функциональности telegram бота с использованием языка программирования Python и готового API от EDENAI.

Таким образом, данная курсовая работа направлена на создание telegram бота, который позволит пользователям эффективно взаимодействовать с мессенджером посредством голосовых команд, улучшая качество и удобство использования цифровых сервисов. Разработка и внедрение такого бота способствует дальнейшему развитию технологий распознавания речи и интеграции искусственного интеллекта в повседневную жизнь.

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В теоретической части работы освещены основные аспекты разработки telegram бота с распознаванием речи и переводом в текст на основе нейронных сетей с использованием API от EDENAI. Это включает в себя выбор технических средств, описание функциональных и технических характеристик бота, а также этапы разработки и график выполнения работ. Ключевыми моментами являются обоснование выбора технологий, использование API от EDENAI для повышения точности и эффективности распознавания речи, определение этапов разработки и выработка стратегии обеспечения качества работы бота.

## 1.1 Предметная область

Предметная область разработки telegram бота с распознаванием речи и переводом в текст на основе нейронных сетей от EDENAI охватывает процессы взаимодействия пользователя с ботом для распознавания речи и озвучивания текста. Это включает в себя получение голосовых сообщений от пользователей, их обработку с использованием алгоритмов распознавания речи, перевод аудио в текст или озвучивание текста и отправку текстового или голосового ответа пользователю.

Важными аспектами предметной области являются точность и скорость распознавания речи, корректность перевода речи в текст, удобство использования бота для пользователей, а также интеграция с платформой Telegram для обеспечения бесперебойной работы и взаимодействия с пользователями.

## 1.2 Выявление проблем

С увеличением количества пользователей мессенджеров и их потребностей в удобных средствах общения возникает необходимость в разработке инструментов, которые упрощают взаимодействие и делают его более эффективным. Одной из таких задач является преобразование голосовых сообщений в текст и наоборот. Однако существующие решения часто сталкиваются с рядом проблем:

* Низкая точность распознавания речи. Существующие алгоритмы могут допускать ошибки в распознавании, особенно при наличии фонового шума или различных акцентов.
* Низкая скорость обработки. Пользователи ожидают мгновенного отклика от бота, и задержки в обработке могут негативно сказаться на их опыте взаимодействия.
* Ограниченная функциональность. Многие боты предлагают либо только распознавание речи, либо только синтез текста в голос, что ограничивает их применение.
* Интеграция с платформами. Не все решения легко интегрируются с популярными мессенджерами, такими как Telegram, что создает дополнительные сложности для разработчиков и пользователей.
* Безопасность и конфиденциальность данных. Обработка голосовых сообщений требует соблюдения высоких стандартов защиты данных, что часто игнорируется в существующих решениях.

Эти проблемы подчеркивают необходимость разработки telegram бота с использованием нейронных сетей от EDENAI, который бы обеспечивал высокую точность и скорость распознавания речи, а также поддерживал как преобразование речи в текст, так и озвучивание текста. Решение данных проблем позволит улучшить качество взаимодействия пользователей с ботом, сделает общение в мессенджере более удобным и эффективным, а также обеспечит высокий уровень защиты персональных данных.

## 1.3 Нейронные сети и алгоритмы

Для повышения точности и эффективности преобразования речи в текст и обратно, API от EDENAI предоставляет доступ к передовым нейронным сетям и алгоритмам, разработанным Google и OpenAI:

### 1.3.1 Google Text-to-Speech (TTS)

Google Text-to-Speech использует передовые нейронные сети для преобразования текста в натуральную, человеческую речь. Ключевые технологии включают:

* WaveNet: архитектура глубоких нейронных сетей, разработанная компанией DeepMind (дочерняя компания Google), способная генерировать высококачественную речь. WaveNet использует сети глубокого обучения для моделирования звуковых волн, что позволяет создавать более естественное и реалистичное озвучивание текста по сравнению с традиционными методами.
* Tacotron 2: система преобразования текста в речь, которая объединяет две нейронные сети: одна переводит текст в спектрограммы (визуальное представление аудиосигнала), а другая спектрограммы в аудиосигналы. Tacotron 2 отличается высокой точностью и естественностью синтезированной речи.

### 1.3.2 OpenAI Speech-to-Text (STT)

OpenAI разрабатывает модели, такие как Whisper, для преобразования речи в текст. Эти модели основаны на трансформерах и обучаются на огромных объемах данных, что позволяет им достигать высокой точности в различных условиях.

* Whisper: модель трансформера для распознавания речи, обученная на большом количестве различных языков и акцентов. Whisper использует архитектуру энкодер-декодер, где энкодер преобразует входной аудиосигнал в внутренние представления, а декодер генерирует текстовую последовательность на основе этих представлений. Whisper также может обрабатывать различные виды шума и искажений, обеспечивая высокую точность даже в сложных условиях.

Использование этих нейронных сетей и алгоритмов через API EDENAI обеспечивает разработчикам доступ к передовым технологиям в области обработки речи, что позволяет создавать высококачественные и эффективные решения для распознавания и синтеза речи.

## 1.4 Техническое задание

Данное Техническое задание (ТЗ) посвящено разработке telegram бота с функцией распознавания речи и последующим переводом её в текст с использованием готового API от EDENAI. Актуальность разработки данного бота обусловлена возрастанием потребности пользователей в удобных средствах общения, автоматизации задач и улучшении взаимодействия в мессенджерах.

Результатом выполнения данного Технического задания станет telegram бот, соответствующий заданным требованиям, который будет представлять собой эффективный инструмент для преобразования речи в текст и обратно.

### 1.4.1 Требования к функциональным характеристикам программы

Разрабатываемый telegram бот должен обладать следующими функциональными характеристиками:

* Распознавание речи: получение голосовых сообщений от пользователей, их обработка с использованием API от EDENAI и перевод аудио в текст.
* Синтез речи: преобразование текстовых сообщений пользователей в аудио и отправка голосового ответа.
* Управление настройками пользователя: возможность изменять язык распознавания речи и пол голоса для синтеза речи.
* Обработка текстовых команд: обработка команд /start, /help, /settings, /lang, /gender и предоставление соответствующих ответов.

### 1.4.2 Требования к техническим характеристикам программы

Разрабатываемый telegram бот должен соответствовать следующим техническим требованиям:

* Платформа и совместимость: бот должен быть совместим с API Telegram и корректно функционировать на серверах, поддерживающих Python 3.7 и выше.
* Производительность: время отклика бота не должно превышать 2 секунд при нормальных условиях использования API EDENAI.
* Пользовательский интерфейс: бот должен иметь интуитивно понятный и простой интерфейс, обеспечивающий удобный доступ ко всем функциям.
* Сетевые требования: бот должен корректно работать при наличии стабильного интернет-соединения, использовать защищенные методы передачи данных.

### 1.4.3 Требования к программным характеристикам программы

В программе используются следующие модули и библиотеки:

* Aiogram: библиотека для создания ботов на Python, обеспечивающая работу с API Telegram.
* Aiohttp: библиотека для выполнения асинхронных HTTP-запросов к API EDENAI.
* Python 3.8+: основной язык программирования.

### 1.4.4 Этапы разработки программы

Подход к разработке программы может быть следующим:

1. Планирование: определение функциональных и технических характеристик бота, подготовка технического задания.
2. Проектирование архитектуры: разработка архитектуры программы, определение структуры бота, компонентов, модулей и способов взаимодействия между ними.
3. Реализация функциональности: программирование и реализация функциональности бота.
4. Документирование: подготовка технической документации для разработчиков, создание пользовательского руководства.
5. Развертывание и поддержка: развертывание бота на сервере, обеспечение его стабильной работы и технической поддержки.

## 1.5 Технологии для разработки программы

Для обеспечения полноценного функционирования telegram бота с распознаванием речи необходимо использование следующих программных средств:

### 1.5.1 Среда разработки

Среда разработки: Visual Studio Code (VSCode) — мощная и удобная интегрированная среда разработки с поддержкой множества расширений и инструментов для работы с Python и Telegram Bot API.

### 1.5.2 Языки программирования и фреймворки

Основным языком программирования является Python версии 3.7 и выше. Для работы с Telegram API используется библиотека Aiogram, для выполнения асинхронных HTTP-запросов — библиотека Aiohttp, а для обработки данных — библиотека JSON.

### 1.5.3 Библиотеки

Основные библиотеки, используемые в программном коде:

* Aiogram: для создания telegram бота.
* Aiohttp: для выполнения асинхронных HTTP-запросов к API EDENAI.
* Pathlib: для работы с файловой системой.
* JSON: для обработки данных в формате JSON.

### 1.5.4 Инструменты для контроля версий и совместной работы

В качестве системы контроля версий, обеспечивающей управление изменениями в коде и совместную работу над проектом принято использовать Git.

Платформа для хостинга репозиториев, совместной работы над кодом и управления задачами проекта – GitHub.

## Вывод

В разработке telegram бота с распознаванием речи ключевыми аспектами являются анализ требований, выбор оптимальных технологий и этапы разработки. Ключевые этапы включают в себя анализ требований, проектирование, разработку, тестирование, документирование, развертывание и поддержку. Разработка и внедрение такого бота позволит улучшить качество взаимодействия пользователей с мессенджером и сделать общение более удобным и эффективным.

# 2 ПРОЕКТНАЯ ЧАСТЬ

В проектной части работы рассмотрены выбор алгоритма работы telegram бота. Кроме того, описан стандарт разработки для обеспечения единообразия и качества кода.

## 2.1 Алгоритм работы программы.

Алгоритм работы telegram бота начинается с команды /start, после которой пользователю отображается приветственное сообщение. Бот поддерживает несколько команд, таких как /help для получения справки, /settings для получения текущих настроек, /lang для изменения языка и /gender для изменения пола голосового ассистента. Основная функциональность включает обработку текстовых сообщений и голосовых сообщений, их преобразование в текст или в аудио с использованием API от EDENAI.

## 2.2. Проектирование структуры программы

В ходе проектирования структуры программы было выполнено разделение приложения на несколько пакетов для лучшей организованности и управляемости кода:

* bot.py: основной файл, содержащий логику работы бота.
* config.py: файл конфигурации, содержащий ключи для API EdenAI.
* resources.py: файл ресурсов, содержащий тексты для бота.
* speech\_to\_text.py: файл для распознавания речи.
* text\_to\_speech.py: файл для перевода текста в речь.

## 2.3. Блок-схемы

Блок-схемы для алгоритмов работы бота:

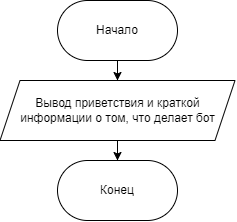


Рисунок 2.1 – Блок-схема работы команды /start



Рисунок 2.2 – Блок схема работы команды /help



Рисунок 2.3 – Блок-схема работы команды /settings

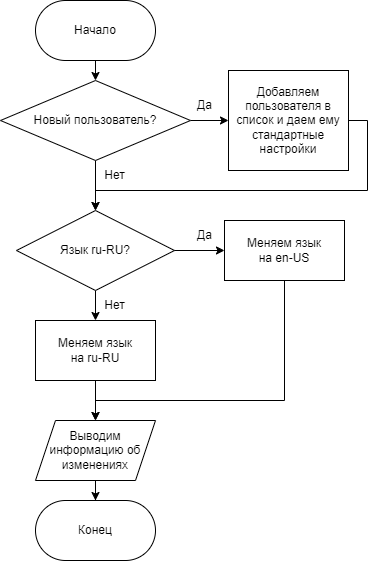


Рисунок 2.4 – Блок-схема работы команды /lang

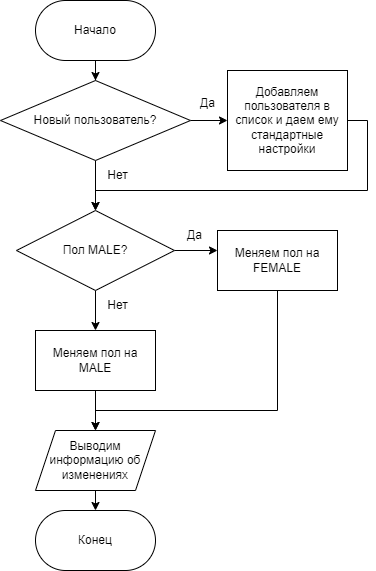


Рисунок 2.5 – Блок-схема работы команды /gender

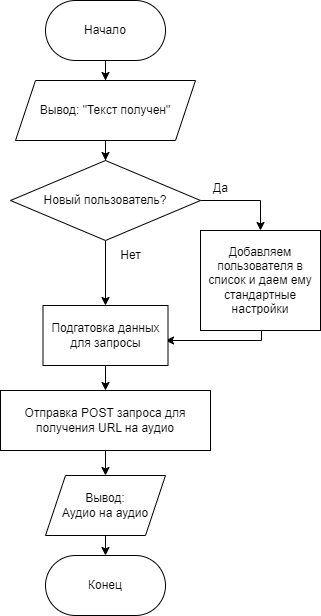


Рисунок 2.6 – Блок-схема выполняемой работы после отправки текста



Рисунок 2.7 – Блок-схема выполняемой работы после отправки голосового сообщения

## 2.4. UML-диаграмма классов

UML-диаграмма классов для программы:

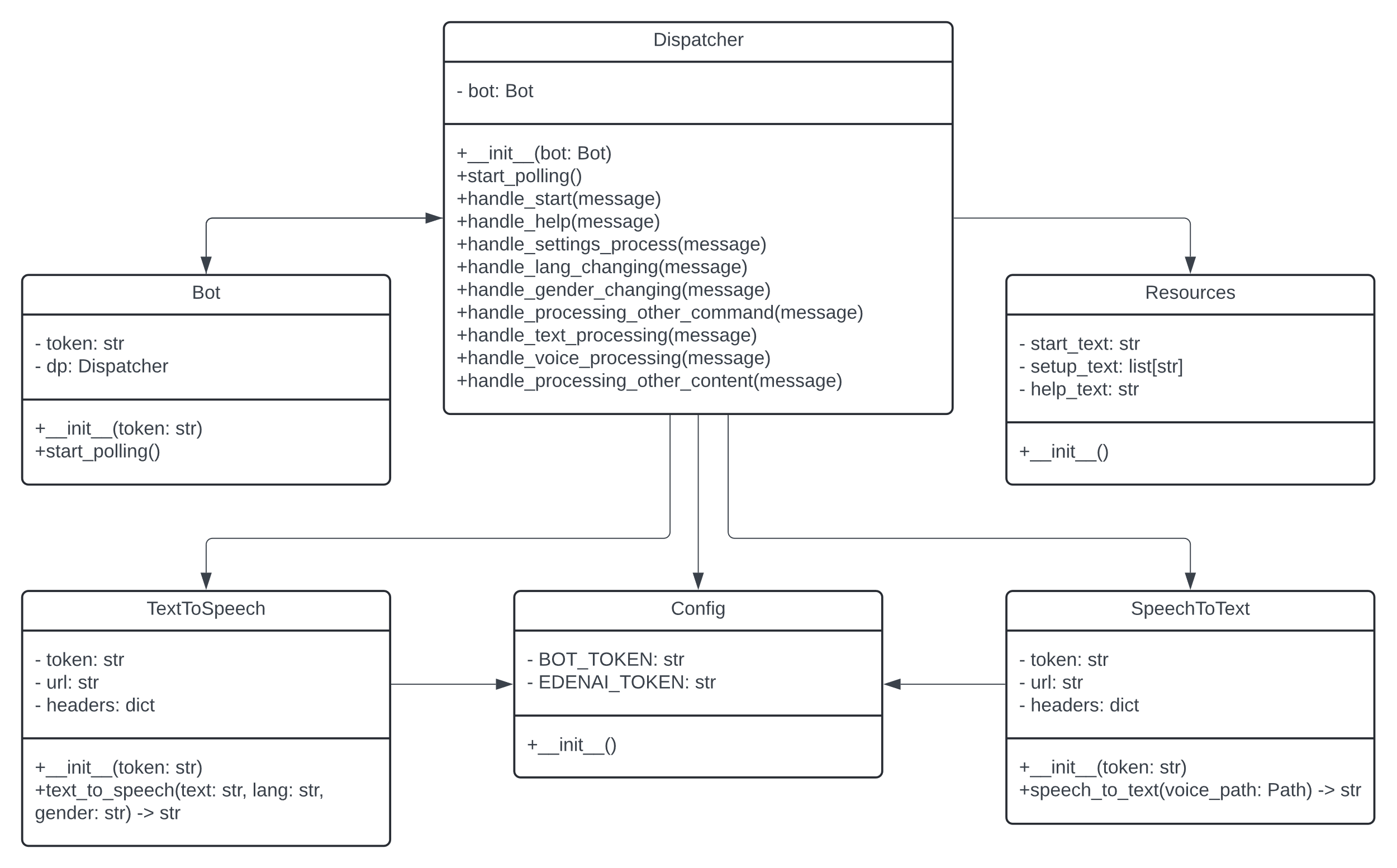


Рисунок 2.8 – UML-диаграмма классов

В этой диаграмме классов, Bot инициализируется с токеном и имеет связь с Dispatcher. Dispatcher имеет ссылку на Bot и использует конфигурацию (Config), а также функции для преобразования речи в текст (SpeechToText) и текста в речь (TextToSpeech). Resources содержит текстовые ресурсы для ответов бота. SpeechToText и TextToSpeech инициализируются с токеном и имеют ссылку на Config.

## 2.5. Диаграмма прецедентов

Диаграмма прецедентов для программы:

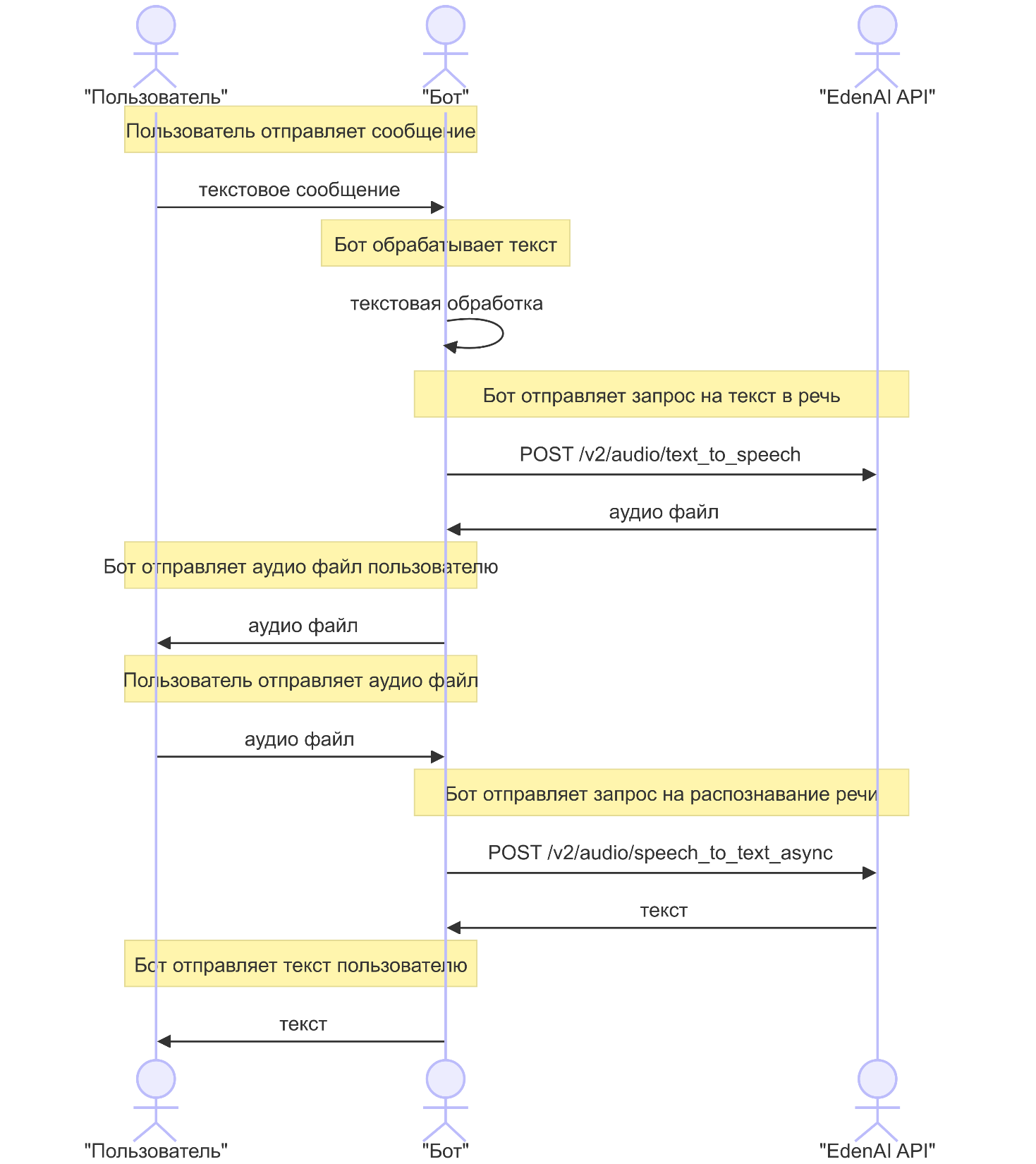


Рисунок 2.9 – Диаграмма прецедентов

В этой диаграмме прецедентов, пользователь отправляет текстовое сообщение боту, который обрабатывает текст и отправляет запрос на преобразование текста в речь в EdenAI API. EdenAI API преобразует текст в речь и отправляет аудио файл боту. Бот отправляет аудио файл пользователю. Затем пользователь отправляет аудио файл боту, который отправляет запрос на распознавание речи в EdenAI API. EdenAI API распознает речь и отправляет текст боту. Бот отправляет текст пользователю.

## Вывод

В ходе работы были рассмотрены ключевые аспекты разработки telegram бота с распознаванием речи и переводом текста в речь, начиная с выбора языка программирования и среды разработки, подробного описания алгоритма работы приложения и стандартов разработки, и заканчивая дизайном и картой экранов, стилистиками приложения.

# 3. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 3.1 Реализация программы

Программа «Telegram Bot для распознавания речи и перевода в текст» представляет собой Telegram-бота, разработанного на языке Python. Бот предназначен для преобразования голосовых сообщений в текст и обратно с использованием нейронных сетей и внешних API сервисов. Основные возможности программы включают распознавание речи, перевод текста в речь и настройку параметров голосовых сообщений (язык и пол голоса).

### 3.1.1 Основные компоненты программы

Программа состоит из следующих основных компонентов:

1. Файл bot.py – основной файл, в котором реализован бот и его функциональность.
2. Модуль src/ – содержит вспомогательные скрипты для работы бота:
   * text\_to\_speech.py – модуль для преобразования текста в речь.
   * speech\_to\_text.py – модуль для распознавания речи и преобразования ее в текст.
   * resources.py – модуль, содержащий текстовые ресурсы для сообщений бота.
   * config.py – конфигурационный файл для хранения токенов и других настроек.

### 3.1.2 Основные функции программы

* Функция handle\_start отправляет приветственное сообщение при запуске бота.
* Функция handle\_help предоставляет пользователю справочную информацию по доступным командам.
* Функция handle\_settings\_process отображает текущие настройки пользователя (язык и пол голоса).
* Функции handle\_lang\_changing и handle\_gender\_changing позволяют пользователю изменять настройки языка и пола голоса.
* Функция handle\_text\_processing обрабатывает текстовые сообщения, преобразует их в речь и отправляет обратно пользователю в виде голосового сообщения.
* Функция handle\_voice\_processing обрабатывает голосовые сообщения, распознает речь и отправляет полученный текст обратно пользователю.
* Функции handle\_processing\_other\_command и handel\_processing\_other\_content обрабатывают неизвестные команды и контент, отправляя соответствующие сообщения пользователю.

### 3.1.3 Реализация базы данных

Для хранения данных в программе используется простая структура на основе словаря (dict), которая позволяет быстро и эффективно управлять настройками пользователей. Несмотря на свою простоту, такой подход подходит для небольших приложений или для прототипирования, предоставляя базовый функционал для сохранения и извлечения данных.

База данных представлена словарем user\_settings, где ключами являются идентификаторы пользователей (в виде строк), а значениями – вложенные словари, содержащие настройки каждого пользователя. Вложенные словари имеют следующие ключи:

* + - lang – язык, используемый для распознавания и синтеза речи.
    - gender – пол голоса, используемый для синтеза речи.

## 3.2 Описание готового продукта

Telegram-бот «Chat Assistant» с распознаванием речи и последующим переводом в текст представляет собой удобное и многофункциональное приложение, разработанное на языке Python. Этот бот предназначен для преобразования голосовых сообщений пользователей в текст и синтеза речи на основе текста, используя современные технологии нейронных сетей.

### 3.2.1 Пользовательский интерфейс

При первом запуске пользователя встречает приветственное сообщение, которое кратко описывает функционал бота и предлагает начать работу. Пример экрана начальной страницы представлен на рисунке ниже (рисунок 3.1).

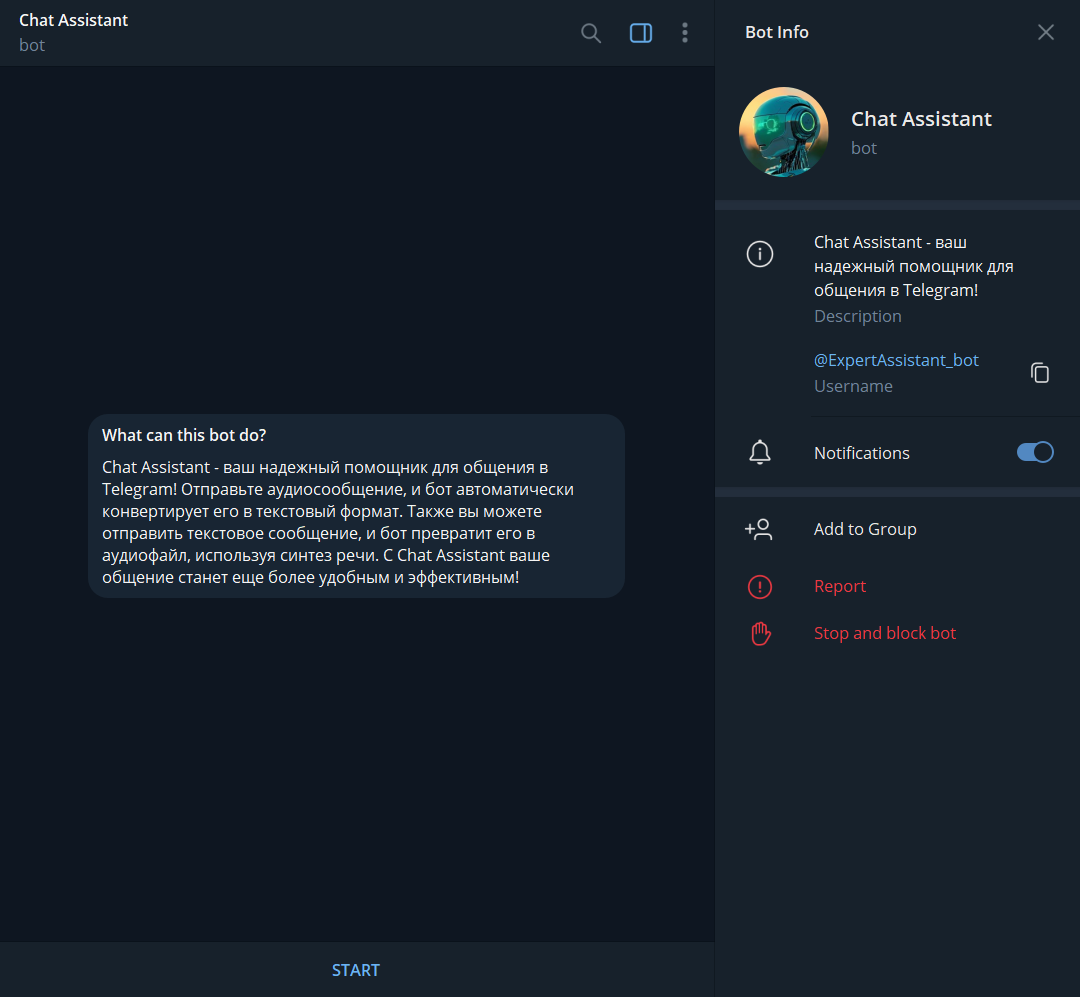


Рисунок 3.1 – Начальная страница бота

При первом запуске пользователь видит кнопки для начала работы, такие как «/start» для инициализации взаимодействия с ботом.

При нажатии кнопки «START», выполняется функция handle\_start() и пользователя встречает приветственное сообщение, которое кратко описывает функционал бота и предлагает начать работу. отображения результата выполнения команды «/start» представлен на рисунке ниже (рисунок 3.2).

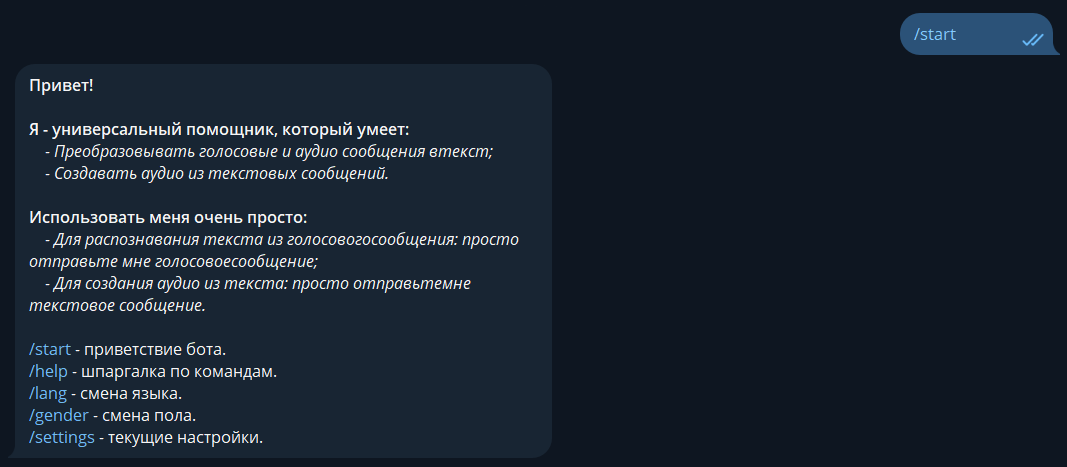


Рисунок 3.2 – Сообщения выводимое на команду «/start»

После нажатия «START» появляется строка для ввода сообщений. Таже появилась кнопка «Menu» при нажатии на который выводится список команд доступных пользователю для взаимодействия с ботом.

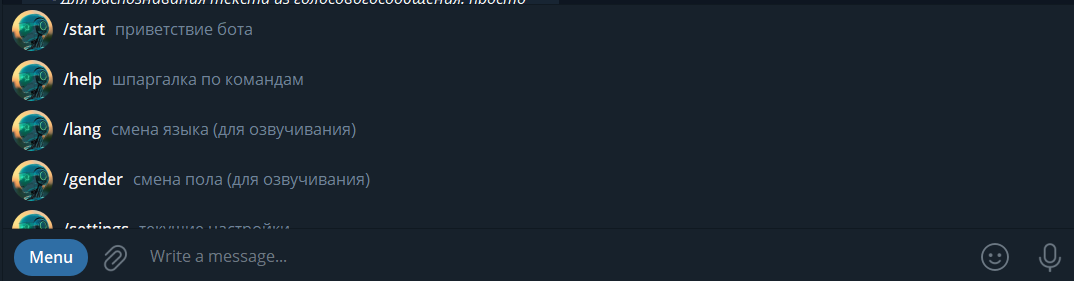


Рисунок 3.3 – Список команд

Рядом с каждой командой есть краткое описание.

Мы уже знаем, что выводит команда «/start». Теперь рассмотри команду «/help». Когда пользователь вводит команду «/help», выполняется функция handle\_help(), бот предоставляет справочную информацию по основным командам, которые можно использовать для взаимодействия с ним. Пример отображения результата выполнения команды «/help» представлен на рисунке ниже (рисунок 3.4).

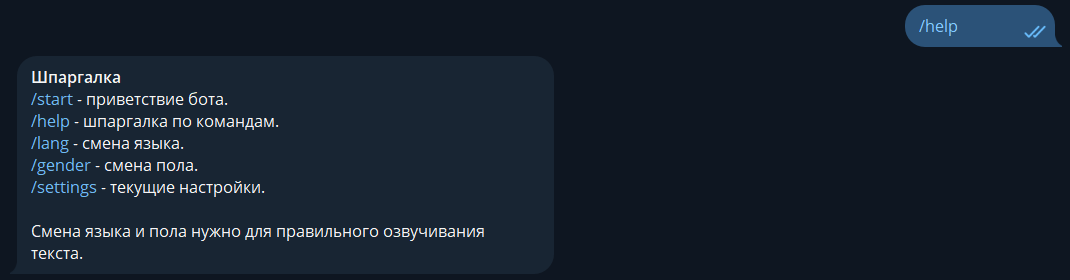


Рисунок 3.4 – Сообщения выводимое на команду «/help»

Когда пользователь вводит команду «/settings», выполняется функция handle\_settings\_process(). Бот предоставляет информацию о текущих настройках, которые включают выбранный язык и пол для озвучивания текстов. Если пользователь ранее редактировал настройки, то они будут искаться в словаре, иначе применяются стандартные. Пример отображения результата выполнения команды «/settings» представлен на рисунке ниже (рисунок 3.5).

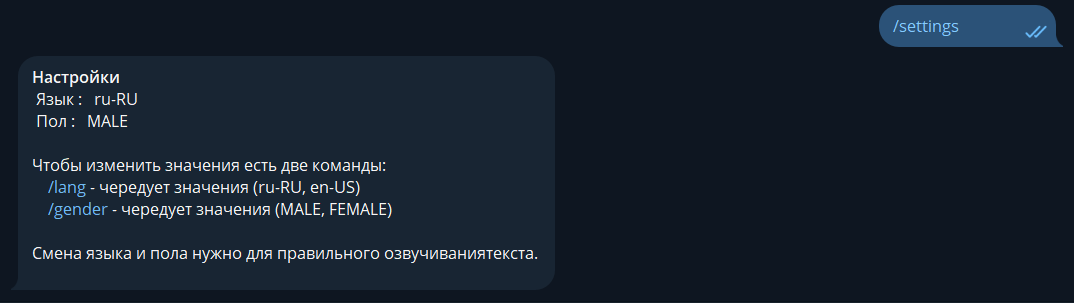


Рисунок 3.5 – Сообщения выводимое на команду «/settings»

Когда пользователь вводит команду «/lang», выполняется функция handle\_lang\_changing(), бот изменяет текущий язык и предоставляет информацию о новом значении. Пример отображения результата выполнения команды «/lang» представлен на рисунке ниже (рисунок 3.6).

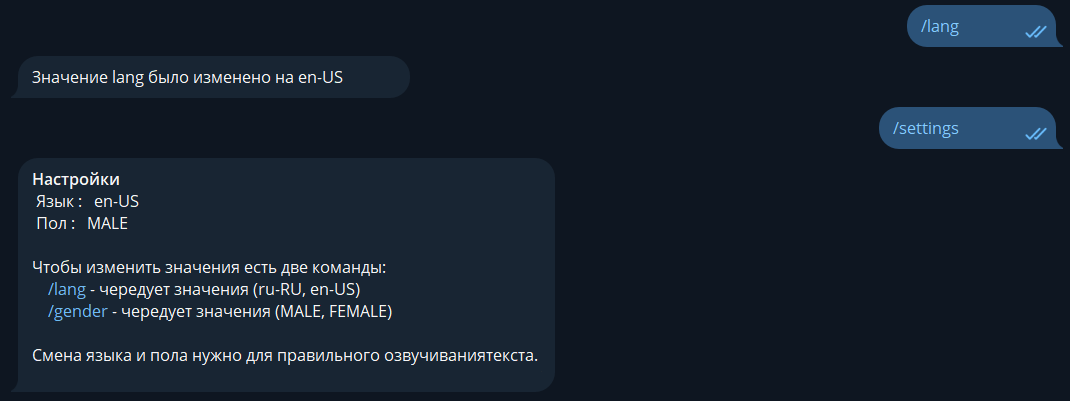


Рисунок 3.6 – Сообщения выводимое на команду «/lang»

Когда пользователь вводит команду «/gender», выполняется функция handle\_gender\_changing(), бот изменяет текущий пол для озвучивания текстов и предоставляет информацию о новом значении. Пример отображения результата выполнения команды «/gender» представлен на рисунке ниже (рисунок 3.7).

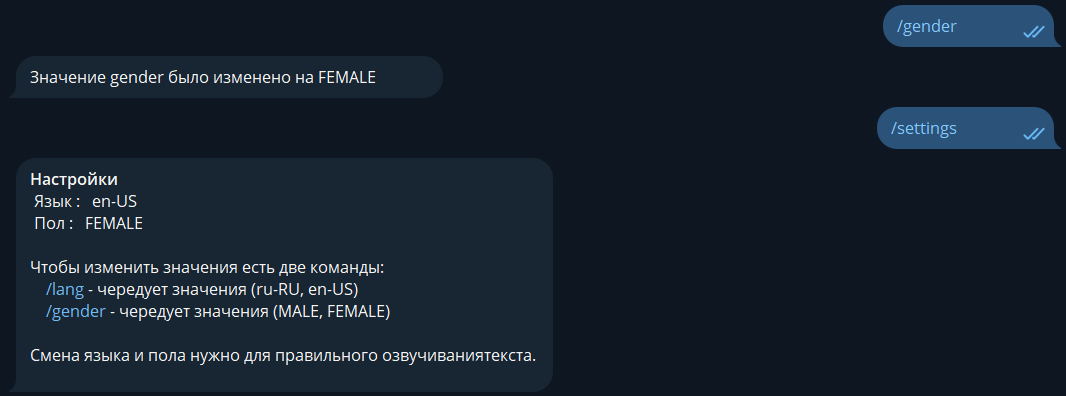


Рисунок 3.7 – Сообщения выводимое на команду «/gender»

При редактирования какой-либо настройки пользователя, изменения добавляются в словарь для последующего обращения к ним.

Когда пользователь отправляет боту текстовое сообщение, выполняется функция handle\_text\_processing(), которая преобразует текст в голосовое сообщение и возвращает его боту. Затем бот отправляет это голосовое сообщение пользователю. Таким образом, текст пользователя озвучивается и предоставляется в аудиоформате. Пример отображения результата на отправку текстового сообщения представлен на рисунке ниже (рисунок 3.8).

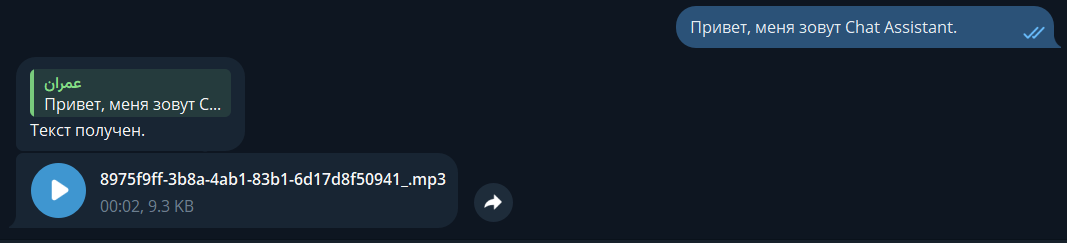


Рисунок 3.8 – Озвучивание текста.

Когда пользователь отправляет голосовое сообщение, выполняется функция handle\_voice\_processing(), которая преобразует голосовое сообщение в текст и возвращает боту. Затем бот отправляет это текстовое сообщение пользователю. Таким образом, голосовое сообщение пользователя представляется в виде текста. Пример отображения результата на отправку текстового сообщения представлен на рисунке ниже (рисунок 3.9).



Рисунок 3.9 – Обработка речи

### 3.2.2 Методы, функции и файлы.

Структура проекта состоит из нескольких файлов, которые включают основной файл запуска бота, модули для преобразования текста в речь и обратно, конфигурационные файлы и файлы с ресурсами.

Файл bot.py (Приложение A1) является основным компонентом системы, содержащим логику работы бота и обработку пользовательских запросов. Он включает в себя создание экземпляра бота с использованием библиотеки aiogram и настройку диспетчера для обработки команд и сообщений.

В начале файла импортируются необходимые библиотеки и модули для работы с Telegram API, а также другие вспомогательные модули и конфигурации (рисунок 3.10).

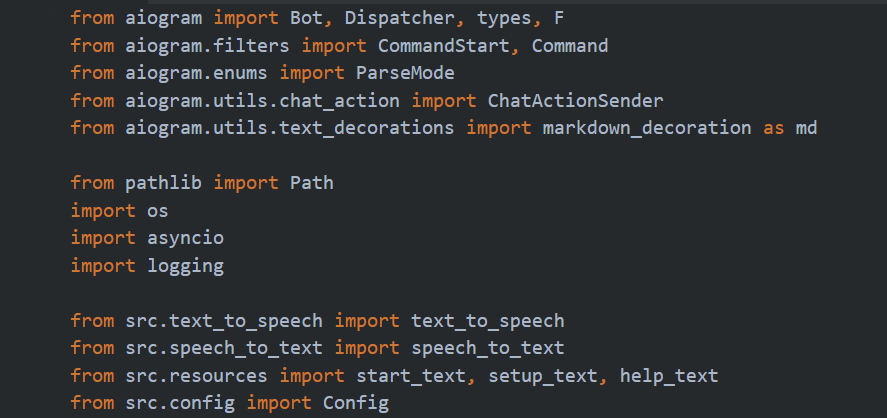


Рисунок 3.10 – Импорты в файле bot.py

Создание экземпляров бота и диспетчера с использованием токена, указанного в конфигурационном файле .env (рисунок 3.11).

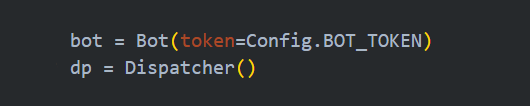


Рисунок 3.11 – Экземпляры бота и диспетчера

Словарь user\_settings используется для хранения настроек пользователей, таких как язык и пол. По умолчанию задаются значения для языка (ru-RU) и пола (MALE) (рисунок 3.12).

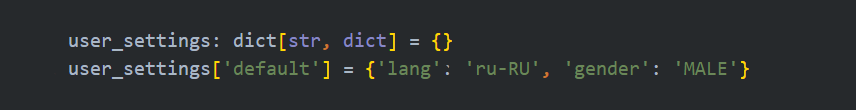


Рисунок 3.12 – Словарь настроек

Функция handle\_start() обрабатывает команду /start и отправляет приветственное сообщение пользователю (рисунок 3.13).

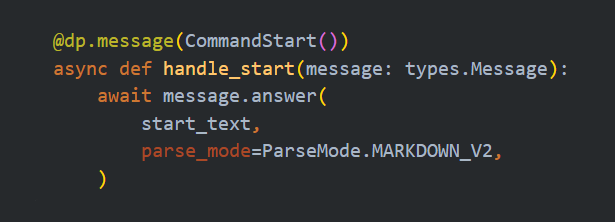


Рисунок 3.13 – Функция handle\_start()

Функция handle\_help() обрабатывает команду /help и отправляет сообщение с информацией о доступных командах и функционале бота (рисунок 3.14).

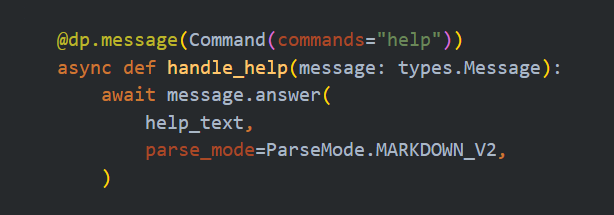


Рисунок 3.14 – Функция handle\_help()

Функция handle\_settings\_process() обрабатывает команду /settings и отправляет пользователю текущие настройки, такие как язык и пол. Если у пользователя еще нет настроек, используются значения по умолчанию (рисунок 3.15).

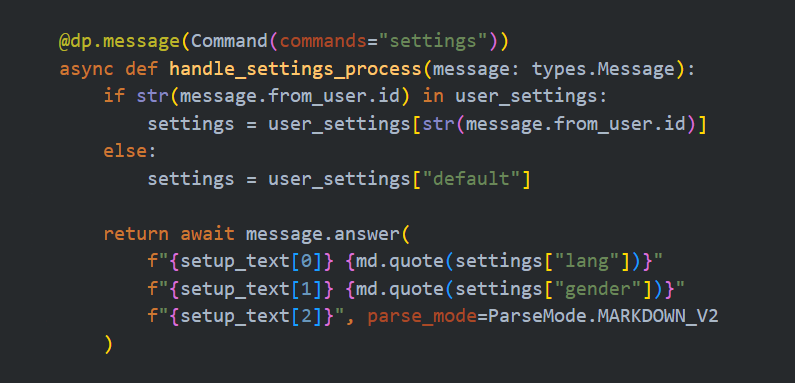


Рисунок 3.15 – Функция handle\_settings\_process()

Функция handle\_lang\_changing() обрабатывает команду /lang и переключает язык пользователя между ru-RU и en-US (рисунок 3.16).

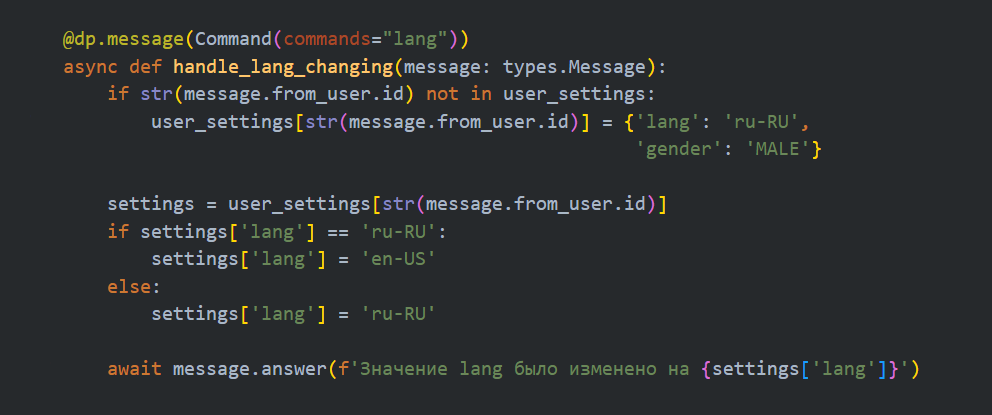


Рисунок 3.16 – Функция handle\_lang\_changing()

Функция handle\_gender\_changing() обрабатывает команду /gender и переключает пол пользователя между MALE и FEMALE (рисунок 3.17).

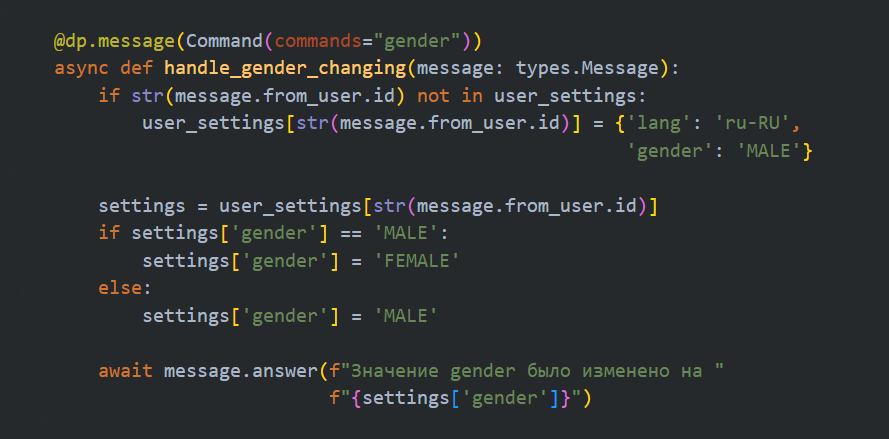


Рисунок 3.17 – Функция handle\_gender\_changing()

Функция handle\_processing\_other\_command обрабатывает неизвестные команды, отправляя пользователю сообщение о том, что команда не распознана (рисунок 3.18).

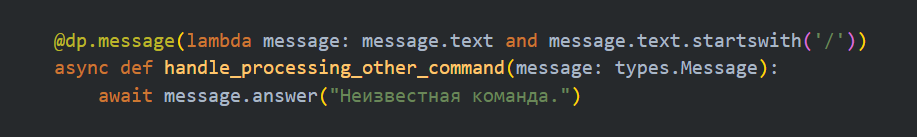


Рисунок 3.18 – Функция handle\_processing\_other\_command

Функция handle\_text\_processing() обрабатывает текстовые сообщения, отправляя подтверждение о получении текста и затем преобразует его в голосовое сообщение с помощью функции text\_to\_speech() (рисунок 3.19).



Рисунок 3.19 – Функция handle\_text\_processing()

Файл text\_to\_speech.py (Приложение A2) содержит логику преобразования текста в речь с использованием сервиса Eden AI (рисунок 3.20).

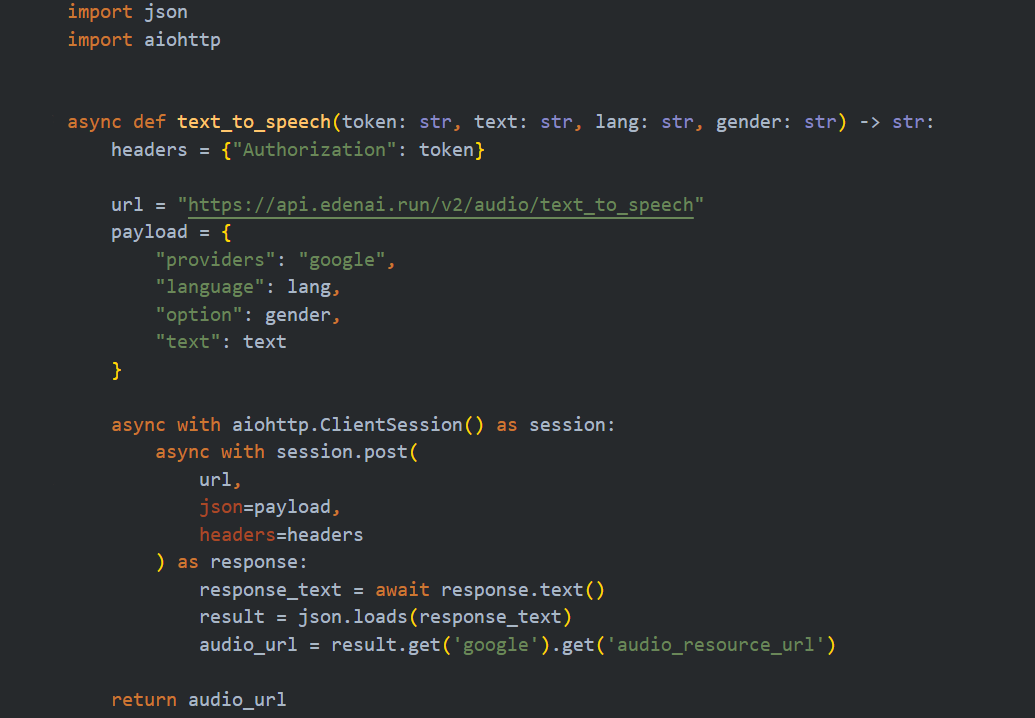


Рисунок 3.20 – Модуль text\_to\_speech.py

В этом модуле настраиваются заголовки и данные запроса. Заголовок Authorization содержит токен для аутентификации на сервере Eden AI. В data указывается поставщик услуг, язык, пол диктора и текст для преобразования в речь. Далее выполняется POST-запрос к API Eden AI для преобразования текста в речь. Если запрос был успешным (код состояния 200), извлекается URL аудиофайла из ответа и выполняется запрос для его загрузки. Если запрос на загрузку аудиофайла успешен (код состояния 200), файл сохраняется на диск. В случае ошибки выбрасывается исключение с соответствующим сообщением.

Функция handle\_voice\_processing() обрабатывает голосовые сообщения, аудио и документы, извлекая файл, загружая его на диск и преобразуя его в текст с помощью функции speech\_to\_text() (рисунок 3.21).



Рисунок 3.21 – Функция handle\_voice\_processing()

Файл speech\_to\_text.py (Приложение A3) содержит функцию speech\_to\_text() для преобразования аудиофайлов в текст, используя API сервиса EdenAI (рисунок 3.22).

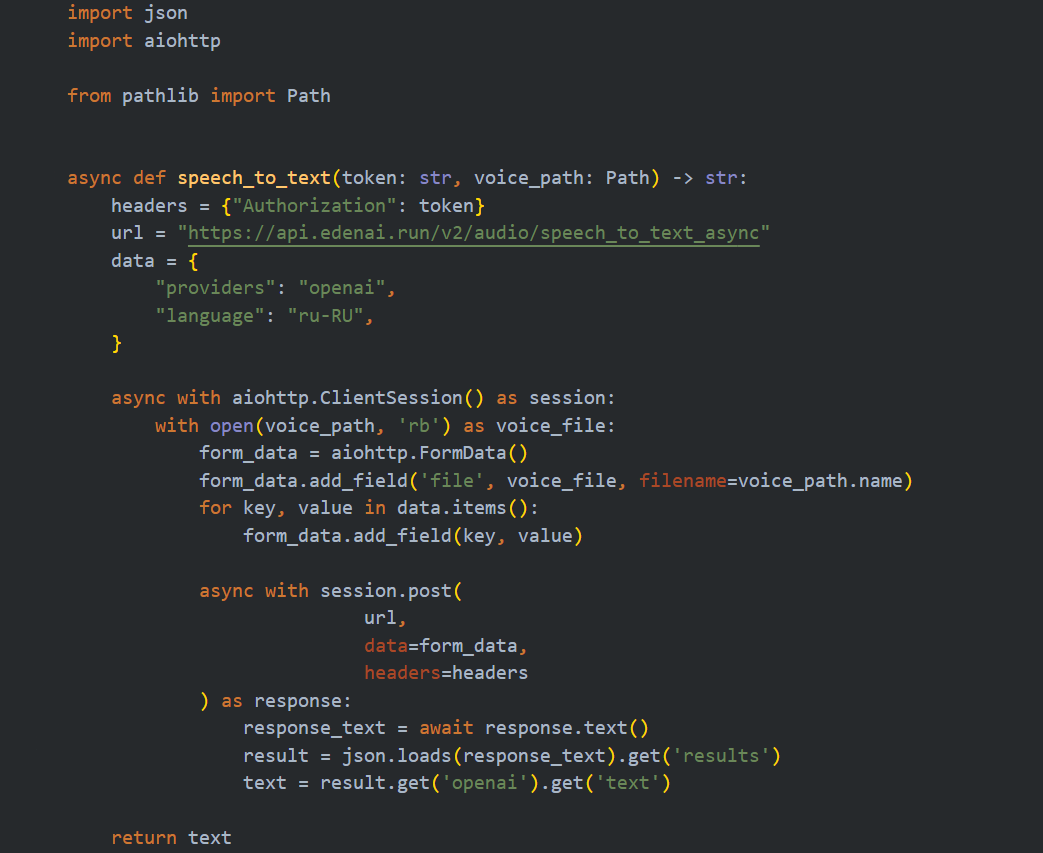


Рисунок 3.22 – Функция speech\_to\_text()

Этот модуль открывает аудиофайл в бинарном режиме и читает его содержимое. Далее настраиваются заголовки, файлы и данные запроса. Заголовок Authorization содержит токен для аутентификации на сервере Eden AI. В files указывается аудиофайл, который будет отправлен на сервер, а в data указываются поставщик услуг и язык распознавания. Потом выполняется POST-запрос к API Eden AI для преобразования аудио в текст. Если запрос был успешным (код состояния 200), извлекается транскрибированный текст из ответа и возвращается. Если текст не был распознан, возвращается соответствующее сообщение. В случае ошибки возвращается код состояния и текст ошибки.

Функция handel\_processing\_other\_content() обрабатывает сообщения с типом контента, который бот не поддерживает, отправляя пользователю соответствующее сообщение (рисунок 3.23).

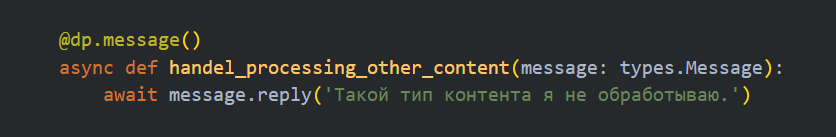


Рисунок 3.23 – Функция handel\_processing\_other\_content()

Функция main() настраивает логирование и запускает процесс опроса для получения обновлений и обработки сообщений. Бот запускается, если скрипт выполняется как основная программа (рисунок 3.24).

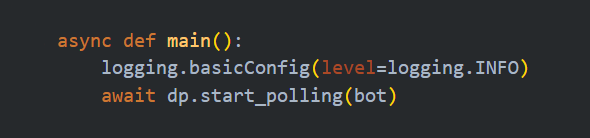


Рисунок 3.24 – Функция main()

Также есть модуль recources.py (Приложение A4) в котором хранятся все тексты для сообщений. Они были перемещены для того, чтобы не занимали много места в основном файле (рисунок 3.25).

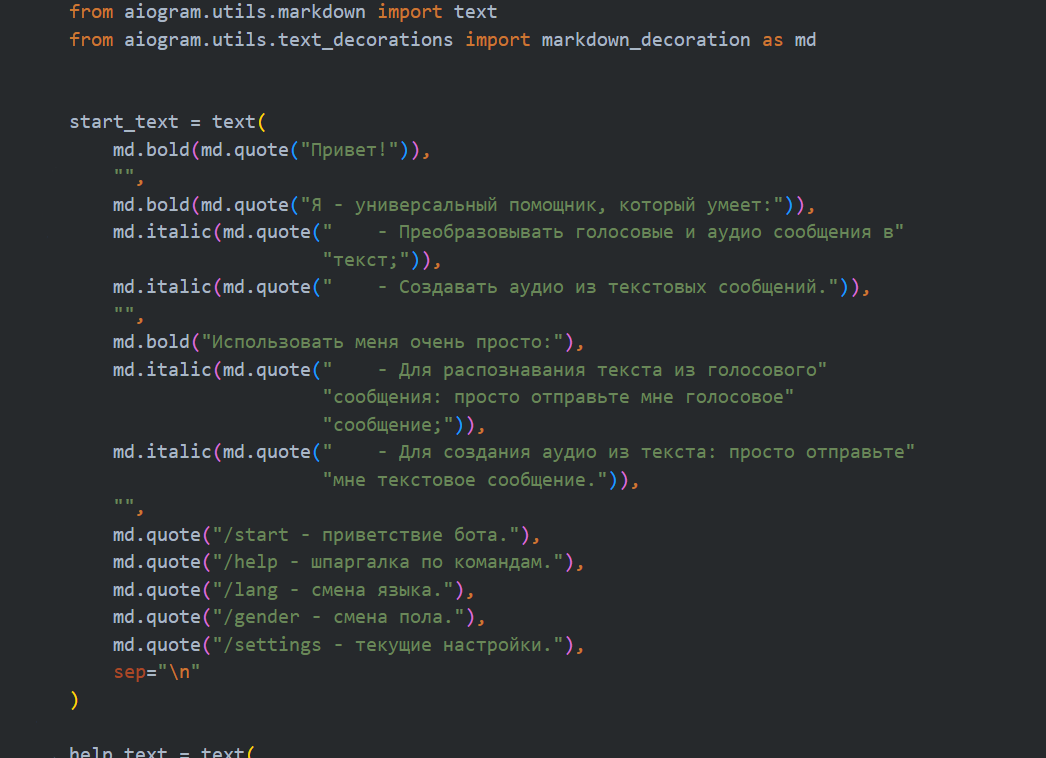


Рисунок 3.25 – Файл recources.py

Еще один конфигурационный файл config.py (Приложение A5), который импортирует ключи из виртуального окружения (рисунок 3.26).

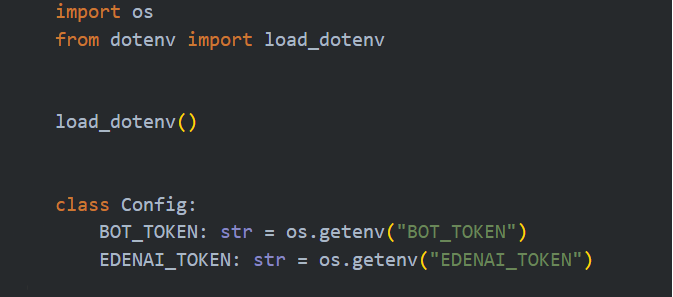


Рисунок 3.26 – Файл конфигурации

## 3.3 Руководство для пользователя

Для начала с клонируйте репозиторий с исходным кодом проекта на ваш локальный компьютер (рисунок 3.27):

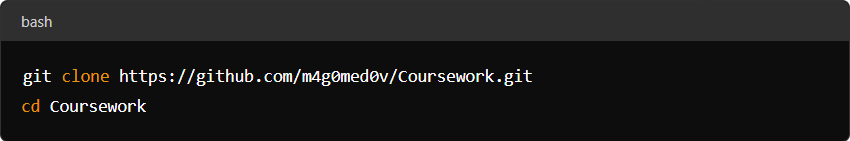


Рисунок 3.27 – Команда клонирования проекта

Создаем и активируем виртуальное окружение (рисунок 3.28):

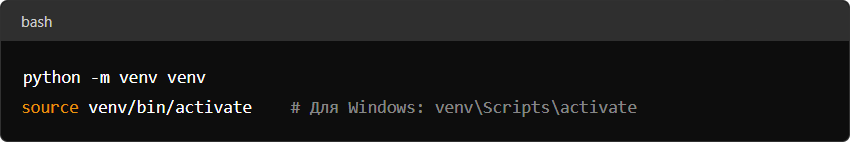


Рисунок 3.28 – Создание и активация виртуального окружения

Устанавливаем зависимости python (рисунок 3.29):

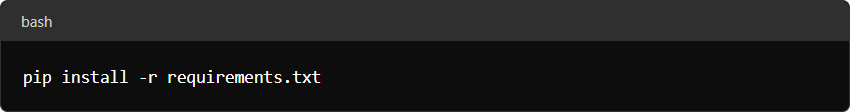


Рисунок 3.29 – Команда для установки зависимостей

Создайте файл .env в корневой папке проекта. Добавьте свои API-ключи для EdenAI и Telegram в файл .env (рисунок 3.30):

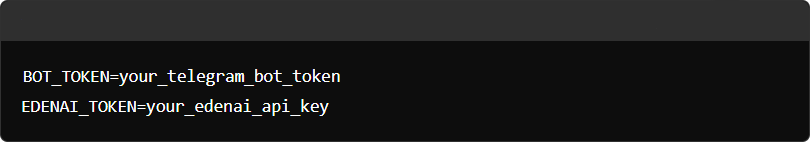


Рисунок 3.30 – Содержание файла .env

Запустите бота с помощью команды (рисунок 3.31):

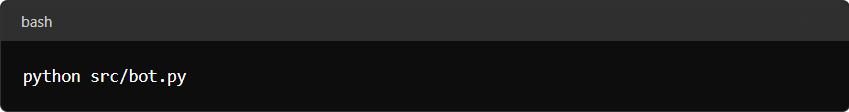


Рисунок 3.31 – Запуск бота

Для получения дополнительных сведений о настройке и использовании бота, ознакомьтесь с файлом README.md в репозитории проекта.

Ссылка на GitHub: <https://github.com/m4g0med0v/Coursework.git>.

## Вывод

Был разработан telegram-бот, который обеспечивает преобразование голосовых сообщений в текст и обратно. Приложение обладает интуитивно понятным интерфейсом и высокой производительностью. Для разработки бота использовались современные технологии и подходы, что позволило создать функциональное и надежное программное решение. Пользователи могут легко взаимодействовать с ботом, отправляя и получая сообщения в различных форматах, что расширяет возможности для использования голосовых интерфейсов в повседневной жизни.

# ЗАКЛЮЧЕНИЕ

В рамках данного курсового проекта был разработан telegram-бот, предназначенный для преобразования голосовых сообщений в текст и обратно. В ходе работы были достигнуты следующие результаты.

Разработанное приложение обладает интуитивно понятным интерфейсом и высокой производительностью, что позволяет пользователям легко взаимодействовать с ботом и использовать его в повседневной жизни. Проект включал в себя анализ требований, выбор оптимальных технологий, разработку и тестирование функциональных модулей.

Таким образом, данный telegram-бот представляет собой важный инструмент для цифровизации голосового взаимодействия, открывая новые возможности для пользователей.

# СПИСОК ЛИТЕРАТУРЫ

1. Р.А. Гудфеллоу, Й.И. Гудфеллоу, М.И. Курпи, "Глубокое обучение", Москва, ДМК Пресс, 2019, 1072 с.
2. Франсуа Шолле, "Глубокое обучение на Python", Санкт-Петербург, БХВ-Петербург, 2018, 404 с.
3. Ян Гудфеллоу, Аарон Курвилль, Иошуа Бенджио, "Глубокое обучение", Москва, ДМК Пресс, 2019, 798 с.
4. Себастьян Рашка, "Python и машинное обучение", Санкт-Петербург, Питер, 2016, 392 с.
5. Аурелиен Жерон, "Погружение в машинное обучение", Москва, ДМК Пресс, 2019, 384 с.
6. Николенко Сергей, Кадурин Андрей, Архангельская Татьяна, "Глубокое обучение. Погружение в мир нейронных сетей", Москва, ДМК Пресс, 2018, 480 с.

# ПРИЛОЖЕНИЕ

## Приложение A

Листинг A1 – Файл bot.py

# Импорт необходимых библиотек и модулей

from aiogram import Bot, Dispatcher, types, F

from aiogram.filters import CommandStart, Command

from aiogram.enums import ParseMode

from aiogram.utils.chat\_action import ChatActionSender

from aiogram.utils.text\_decorations import markdown\_decoration as md

from pathlib import Path

import os

import asyncio

import logging

# Импорт пользовательских модулей для преобразования текста в речь и речи в

# текст, а также текстов сообщений и конфигурации

from src.text\_to\_speech import text\_to\_speech

from src.speech\_to\_text import speech\_to\_text

from src.resources import start\_text, setup\_text, help\_text

from src.config import Config

# Инициализация бота и диспетчера

bot = Bot(token=Config.BOT\_TOKEN)

dp = Dispatcher()

# Словарь для хранения настроек пользователей

user\_settings: dict[str, dict] = {}

user\_settings['default'] = {'lang': 'ru-RU', 'gender': 'MALE'}

# Обработчик команды /start

@dp.message(CommandStart())

async def handle\_start(message: types.Message):

await message.answer(

start\_text,

parse\_mode=ParseMode.MARKDOWN\_V2,

)

# Обработчик команды /help

@dp.message(Command(commands="help"))

async def handle\_help(message: types.Message):

await message.answer(

help\_text,

parse\_mode=ParseMode.MARKDOWN\_V2,

)

# Обработчик команды /settings

@dp.message(Command(commands="settings"))

async def handle\_settings\_process(message: types.Message):

# Получение настроек пользователя, если они существуют, иначе

# использование настроек по умолчанию

if str(message.from\_user.id) in user\_settings:

settings = user\_settings[str(message.from\_user.id)]

else:

settings = user\_settings["default"]

# Ответ с текущими настройками пользователя

return await message.answer(

f"{setup\_text[0]} {md.quote(settings["lang"])}"

f"{setup\_text[1]} {md.quote(settings["gender"])}"

f"{setup\_text[2]}", parse\_mode=ParseMode.MARKDOWN\_V2

)

# Обработчик команды /lang для изменения языка

@dp.message(Command(commands="lang"))

async def handle\_lang\_changing(message: types.Message):

# Создание настроек пользователя, если их нет

if str(message.from\_user.id) not in user\_settings:

user\_settings[str(message.from\_user.id)] = {'lang': 'ru-RU',

'gender': 'MALE'}

# Изменение значения языка

settings = user\_settings[str(message.from\_user.id)]

if settings['lang'] == 'ru-RU':

settings['lang'] = 'en-US'

else:

settings['lang'] = 'ru-RU'

# Ответ с новым значением языка

await message.answer(f'Значение lang было изменено на {settings['lang']}')

# Обработчик команды /gender для изменения пола голоса

@dp.message(Command(commands="gender"))

async def handle\_gender\_changing(message: types.Message):

# Создание настроек пользователя, если их нет

if str(message.from\_user.id) not in user\_settings:

user\_settings[str(message.from\_user.id)] = {'lang': 'ru-RU',

'gender': 'MALE'}

# Изменение значения пола

settings = user\_settings[str(message.from\_user.id)]

if settings['gender'] == 'MALE':

settings['gender'] = 'FEMALE'

else:

settings['gender'] = 'MALE'

# Ответ с новым значением пола

await message.answer(f"Значение gender было изменено на "

f"{settings['gender']}")

# Обработчик неизвестных команд

@dp.message(lambda message: message.text and message.text.startswith('/'))

async def handle\_processing\_other\_command(message: types.Message):

await message.answer("Неизвестная команда.")

# Обработчик текстовых сообщений

@dp.message(F.text)

async def handle\_text\_processing(message: types.Message):

await message.reply("Текст получен.")

# Отправка голосового сообщения, созданного из текста

async with ChatActionSender.upload\_voice(

bot=bot,

chat\_id=message.chat.id,

):

if str(message.from\_user.id) not in user\_settings:

user\_settings[str(message.from\_user.id)] = {'lang': 'ru-RU',

'gender': 'MALE'}

voice\_url = await text\_to\_speech(

Config.EDENAI\_TOKEN, message.text,

str(user\_settings[str(message.from\_user.id)]['lang']),

str(user\_settings[str(message.from\_user.id)]['gender'])

)

await bot.send\_voice(message.from\_user.id, voice=voice\_url)

# Обработчик голосовых, аудио и документов

@dp.message(F.voice | F.audio | F.document)

async def handle\_voice\_processing(message: types.Message):

if message.voice:

file\_id = message.voice.file\_id

elif message.audio:

file\_id = message.audio.file\_id

elif message.document:

file\_id = message.document.file\_id

try:

# Скачивание и сохранение файла на диск

file = await bot.get\_file(file\_id)

file\_path = file.file\_path

file\_on\_disk = Path("", f"{file\_id}.mp3")

await bot.download\_file(file\_path, destination=file\_on\_disk)

await message.reply("Аудио получено")

# Преобразование аудио в текст и отправка результата

async with ChatActionSender.typing(

bot=bot,

chat\_id=message.chat.id,

):

text = await speech\_to\_text(Config.EDENAI\_TOKEN, file\_on\_disk)

if not text:

text = "Не удалось обработать данный файл."

await message.answer(text)

finally:

os.remove(file\_on\_disk)

# Обработчик других типов контента

@dp.message()

async def handel\_processing\_other\_content(message: types.Message):

await message.reply('Такой тип контента я не обработываю.')

# Основная функция для запуска бота

async def main():

logging.basicConfig(level=logging.INFO)

await dp.start\_polling(bot)

# Запуск основного события

if \_\_name\_\_ == "\_\_main\_\_":

asyncio.run(main())

Листинг A2 – Модуль text\_to\_speech.py

import json

import aiohttp

# Асинхронная функция для преобразования текста в речь с использованием

# API EdenAI

async def text\_to\_speech(token: str, text: str, lang: str, gender: str) -> str:

# Заголовки запроса, включающие авторизационный токен

headers = {"Authorization": token}

# URL для API текст-в-речь

url = "https://api.edenai.run/v2/audio/text\_to\_speech"

# Параметры запроса: провайдер, язык, опции (пол голоса) и сам текст

payload = {

"providers": "google",

"language": lang,

"option": gender,

"text": text

}

# Создание асинхронной сессии HTTP-клиента

async with aiohttp.ClientSession() as session:

# Отправка POST-запроса к API

async with session.post(

url,

json=payload,

headers=headers

) as response:

# Чтение текста ответа

response\_text = await response.text()

# Парсинг JSON-ответа

result = json.loads(response\_text)

# Извлечение URL аудио ресурса из ответа

audio\_url = result.get('google').get('audio\_resource\_url')

# Возвращение URL аудио ресурса

return audio\_url

Листинг A3 – Модуль speech\_to\_text.py

import json

import aiohttp

from pathlib import Path

# Асинхронная функция для преобразования речи в текст с использованием

# API EdenAI

async def speech\_to\_text(token: str, voice\_path: Path) -> str:

# Заголовки запроса, включающие авторизационный токен

headers = {"Authorization": token}

# URL для API распознавания речи

url = "https://api.edenai.run/v2/audio/speech\_to\_text\_async"

# Параметры запроса: провайдер и язык

data = {

"providers": "openai",

"language": "ru-RU",

}

# Создание асинхронной сессии HTTP-клиента

async with aiohttp.ClientSession() as session:

# Открытие аудиофайла для чтения в бинарном режиме

with open(voice\_path, 'rb') as voice\_file:

# Создание формы данных для отправки файла и параметров запроса

form\_data = aiohttp.FormData()

form\_data.add\_field('file', voice\_file, filename=voice\_path.name)

for key, value in data.items():

form\_data.add\_field(key, value)

# Отправка POST-запроса к API

async with session.post(

url,

data=form\_data,

headers=headers

) as response:

# Чтение текста ответа

response\_text = await response.text()

# Парсинг JSON-ответа и извлечение результата

result = json.loads(response\_text).get('results')

# Получение распознанного текста из ответа

text = result.get('openai').get('text')

# Возвращение распознанного текста

return text

Листинг A4 – Модуль resources.py

from aiogram.utils.markdown import text

from aiogram.utils.text\_decorations import markdown\_decoration as md

start\_text = text(

md.bold(md.quote("Привет!")),

"",

md.bold(md.quote("Я - универсальный помощник, который умеет:")),

md.italic(md.quote(" - Преобразовывать голосовые и аудио сообщения в"

"текст;")),

md.italic(md.quote(" - Создавать аудио из текстовых сообщений.")),

"",

md.bold("Использовать меня очень просто:"),

md.italic(md.quote(" - Для распознавания текста из голосового"

"сообщения: просто отправьте мне голосовое"

"сообщение;")),

md.italic(md.quote(" - Для создания аудио из текста: просто отправьте"

"мне текстовое сообщение.")),

"",

md.quote("/start - приветствие бота."),

md.quote("/help - шпаргалка по командам."),

md.quote("/lang - смена языка."),

md.quote("/gender - смена пола."),

md.quote("/settings - текущие настройки."),

sep="\n"

)

help\_text = text(

md.bold("Шпаргалка"),

md.quote("/start - приветствие бота."),

md.quote("/help - шпаргалка по командам."),

md.quote("/lang - смена языка."),

md.quote("/gender - смена пола."),

md.quote("/settings - текущие настройки."),

"",

md.quote("Смена языка и пола нужно для правильного озвучивания текста."),

sep="\n"

)

setup\_text = [

text(

md.bold("Настройки"),

"\n",

"Язык : "

),

text(

"\n",

"Пол : "

),

text(

"",

"",

md.quote("Чтобы изменить значения есть две команды:"),

md.quote(" /lang - чередует значения (ru-RU, en-US)"),

md.quote(" /gender - чередует значения (MALE, FEMALE)"),

"",

md.quote("Смена языка и пола нужно для правильного озвучивания"

"текста."),

sep="\n"

)

]

Листинг A5 – Модуль config.py

import os

from dotenv import load\_dotenv

load\_dotenv()

class Config:

BOT\_TOKEN: str = os.getenv("BOT\_TOKEN")

EDENAI\_TOKEN: str = os.getenv("EDENAI\_TOKEN")