МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ

ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»



**Курсовая работа**

на тему «Интеллектуальный телеграм-бот для автоматизации процесса создания конспектов»

Обучающегося 2 курса

очной формы обучения

направление подготовки:

09.03.01 Информатика и вычислительная техника

направленность (профиль):

Технологии разработки программного обеспечения

Волжанин Александр Павлович

Руководитель курсовой работы:

Доктор педагогических наук, профессор

Власова Елена Зотиковна

Санкт-Петербург

2024

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ

ТЕОРЕТИЧЕСКАЯ МОДЕЛЬ

ПРОЕКТИРОВАНИЕ ТЕЛЕГРАМ-БОТА  
РЕАЛИЗАЦИЯ ТЕЛЕГРАМ-БОТА  
ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

ЗАКЛЮЧЕНИЕ

ЛИТЕРАТУРА

**ВВЕДЕНИЕ**

#### Актуальность темы

В современном мире существует острая необходимость в автоматизации рутинных задач для повышения эффективности работы и экономии времени. Одной из таких задач является создание конспектов. Традиционные методы ведения конспектов часто требуют значительных временных затрат и усилий. С развитием технологий, в частности, мессенджеров и ботов, появилась возможность автоматизировать этот процесс, что делает тему создания телеграм-бота для конспектов актуальной и востребованной.

#### Цель и задачи исследования

Целью данной курсовой работы является разработка телеграм-бота для автоматизации процесса создания конспектов. Для достижения этой цели были поставлены следующие задачи:

* Исследование существующих решений для автоматизированного создания конспектов.
* Разработка требований к функционалу телеграм-бота.
* Проектирование и разработка телеграм-бота.
* Тестирование и анализ эффективности разработанного решения.

#### Методы исследования

В ходе работы использовались методы анализа и синтеза информации, а также методы программирования и тестирования программного обеспечения. Основным инструментом разработки стал язык программирования Python и его библиотеки для работы с Telegram API.

#### Структура работы

Работа состоит из введения, пяти глав основной части, заключения, списка литературы. Во введении обоснована актуальность темы, определены цель и задачи исследования. Основная часть включает теоретические основы разработки, описание процесса проектирования и реализации телеграм-бота, а также экспериментальные результаты. Заключение содержит основные выводы по проделанной работе.

### **Теоретические основы**

#### Основы работы с API Telegram

Telegram предоставляет обширный API, который позволяет создавать ботов для взаимодействия с пользователями. Боты могут выполнять различные задачи: от отправки уведомлений до сложных операций, таких как обработка текстов и медиа. API Telegram предоставляет доступ ко всем функциям мессенджера, что делает его мощным инструментом для разработки автоматизированных решений.

##### Основные компоненты Telegram API

* **Методы API**: Методы Telegram API позволяют выполнять действия, такие как отправка сообщений, редактирование сообщений, отправка файлов и т.д.
* **Webhook и long polling**: Эти два метода используются для получения обновлений от серверов Telegram. Webhook позволяет настроить URL-адрес, на который будут отправляться обновления, а long polling позволяет опрашивать серверы Telegram на наличие новых сообщений.
* **Типы обновлений**: Telegram поддерживает множество типов обновлений, включая текстовые сообщения, фотографии, видео, аудио и многое другое.

#### Язык программирования Python

Python — это высокоуровневый язык программирования с простой и читаемой синтаксисом. Он широко используется для разработки различных приложений, включая веб-приложения, скрипты для автоматизации и, конечно же, ботов. Python обладает обширной экосистемой библиотек, что делает его идеальным выбором для быстрой разработки.

##### Основные преимущества Python для разработки ботов

* **Простота и читабельность кода**: Python известен своей читаемостью, что облегчает разработку и поддержку кода.
* **Большое количество библиотек**: Существует множество библиотек, которые упрощают работу с API, обработку данных и другие задачи.
* **Сообщество и поддержка**: Большое сообщество разработчиков и множество учебных ресурсов.

#### Библиотека aiogram

aiogram — это современная и мощная библиотека для разработки Telegram ботов на языке Python. Она предоставляет удобный интерфейс для взаимодействия с API Telegram и упрощает разработку ботов благодаря своей асинхронной архитектуре.

##### Основные возможности aiogram

* **Асинхронная обработка запросов**: Позволяет обрабатывать большое количество запросов одновременно, что делает ботов более отзывчивыми и эффективными.
* **Удобный интерфейс для работы с API**: aiogram предоставляет интуитивно понятный интерфейс для работы с различными методами Telegram API.
* **Многофункциональный хэндлеры**: Хэндлеры aiogram позволяют легко управлять обработкой различных типов сообщений и событий.

##### Основные компоненты aiogram

* **Bot**: Основной класс, который используется для взаимодействия с Telegram API.
* **Dispatcher**: Компонент, отвечающий за маршрутизацию и обработку входящих обновлений.
* **Handler**: Функции или классы, которые обрабатывают различные типы сообщений или команд.

##### Пример использования aiogram

Для создания простого бота с использованием aiogram необходимо выполнить следующие шаги:

1. **Установить библиотеку**:

pip install aiogram

1. **Создать файл для бота**:

from aiogram import Bot, Dispatcher, executor, types

API\_TOKEN = 'YOUR\_TELEGRAM\_BOT\_TOKEN'

# Создание объектов бота и диспетчера

bot = Bot(token=API\_TOKEN)

dp = Dispatcher(bot)

# Обработчик команды /start

@dp.message\_handler(commands=['start'])

async def send\_welcome(message: types.Message):

await message.reply("Привет! Я бот для создания конспектов.")

if \_\_name\_\_ == '\_\_main\_\_':

executor.start\_polling(dp, skip\_updates=True)

**Запустить бота**:

python bot.py

#### Транскрибация с использованием AssemblyAI

AssemblyAI — это сервис для автоматической транскрибации аудио в текст. Он предоставляет API, с помощью которого можно загружать аудиофайлы и получать их текстовую расшифровку. AssemblyAI поддерживает множество языков и предоставляет высокую точность транскрибации благодаря использованию современных технологий машинного обучения.

##### Основные возможности AssemblyAI

* **Высокая точность транскрибации**: Использует модели машинного обучения для достижения высокой точности распознавания речи.
* **Поддержка множества форматов аудио**: Позволяет работать с различными форматами аудиофайлов.
* **Простота интеграции**: Легко интегрируется в различные приложения с помощью RESTful API.

##### Использование AssemblyAI для транскрибации

Для использования AssemblyAI необходимо выполнить следующие шаги:

1. **Создать учетную запись и получить API ключ**: Зарегистрируйтесь на сайте AssemblyAI и получите API ключ для доступа к сервису.
2. **Установить необходимую библиотеку**:

pip install assemblyai

1. **Написать код для транскрибации аудио**:

import assemblyai as aai

# Замените на свой API ключ

aai.settings.api\_key = "API ключ"

# Ссылка на файл в директории

FILE\_URL = "https://github.com/AssemblyAI-Examples/audio-examples/raw/main/20230607\_me\_canadian\_wildfires.mp3"

transcriber = aai.Transcriber()

transcript = transcriber.transcribe(FILE\_URL)

if transcript.status == aai.TranscriptStatus.error:

print(transcript.error)

else:

print(transcript.text)

##### Преимущества использования AssemblyAI

* **Быстрая интеграция**: Простота использования API позволяет быстро интегрировать транскрибацию в приложения.
* **Поддержка большого объема данных**: Сервис способен обрабатывать большие объемы аудио данных, что делает его подходящим для различных сценариев использования.
* **Адаптивность**: Возможность распознавания речи на разных языках и в различных акцентах.

Таким образом, использование Python с библиотекой aiogram и сервиса транскрибации AssemblyAI позволяет эффективно и быстро разрабатывать телеграм-ботов, способных выполнять сложные задачи, такие как создание конспектов на основе аудиозаписей.

Проектирование телеграм-бота

#### Определение функциональных требований

Для эффективного проектирования телеграм-бота важно детально определить функциональные требования, которые помогут структурировать процесс разработки.

##### Основные функции бота

1. **Создание конспектов**:
   * Пользователь должен иметь возможность загрузить аудиофайл для его транскрибации в текст.
2. **Создание конспектов в формате** .docx:
   * Конспекты должны быть автоматически сохраняться в формате .docx и быть доступны для скачивания пользователями.

##### Необходимые команды и интерактивные элементы

* /start — Начало работы с ботом, отображение приветственного сообщения.

#### Разработка архитектуры бота

Архитектура должна быть гибкой, модульной и поддерживать удобное расширение функционала. Рассмотрим основные компоненты архитектуры и их взаимодействие.

##### Основные компоненты архитектуры

1. **Пользовательский интерфейс**:
   * Взаимодействие с пользователем через Telegram, прием и отправка сообщений.
2. **Слой обработки и логики**:
   * Управление созданием и редактированием конспектов, транскрибация аудиофайлов, обработка и экспорт в формат .docx.
3. **Внешние сервисы**:
   * Сервис транскрибации (например, AssemblyAI) для преобразования аудио в текст.
   * Chat gpt для обработки голого текста в полноценный конспект

##### Архитектурная диаграмма

**Пользователь** ↔ **Telegram API** ↔ **Телеграм-бот (aiogram)** ↔ **Сервис транскрибации (AssemblyAI) ↔ Сервис для обработки текста(Chat gpt)**

##### Описание взаимодействия компонентов

* **Пользователь** отправляет команды и сообщения через интерфейс Telegram.
* **Телеграм-бот** (используя aiogram) принимает и обрабатывает команды, обращается к **API Telegram**.
* При загрузке аудиофайла, бот отправляет его на обработку в **сервис транскрибации** (например, AssemblyAI).
* Полученный текст **отправляется в chat gpt по API**
* После чего пользователь получает файл в расширении .docx с помощью **библиотеки python-docx**.
* Бот отправляет файл обратно пользователю через **Telegram API**.

#### Описание используемых технологий и инструментов

Для успешной реализации телеграм-бота мы используем следующие технологии и инструменты:

##### Язык программирования Python

Python выбран благодаря своей простоте и мощным библиотекам, которые позволяют легко разрабатывать и поддерживать ботов.

##### Библиотека aiogram

aiogram предоставляет асинхронные возможности для работы с API Telegram, что делает бота более отзывчивым и производительным.

##### Сервис AssemblyAI

AssemblyAI используется для транскрибации аудиофайлов в текст, обеспечивая высокую точность и простоту интеграции.

##### Библиотека python-docx

python-docx используется для создания и работы с документами в формате .docx. Она позволяет легко генерировать документы из текстовых данных.

##### Сервис Chat gpt

Chat gpt используется для обработки текста, полученного с помощью AssemblyAI.

#### Пример реализации основных функций

##### Создание нового конспекта вручную

from aiogram.types import Message, FSInputFile

from aiogram.filters import CommandStart

from aiogram import F, Router

import os

from cmd\_message import start\_message

import keyboards as kb

from main\_processing import main\_processing

from conversion\_txt\_to\_docx import txt\_to\_docx

from loguru import logger

from cmd\_message import error

router = Router()

# Обработчик команды /start

@logger.catch

@router.message(CommandStart())

async def cmd\_start(message: Message):

await message.answer(start\_message, reply\_markup=kb.main)

# Обработчик аудиосообщений

@router.message(F.voice)

async def handle\_voice\_message(message: Message):

await message.answer("""

🎧 Ваше аудиосообщение принято и обрабатывается. ⏳ Пожалуйста, подождите 5-15 минут. Спасибо за ваше терпение!

""")

voice = message.voice

file\_id = voice.file\_id

# Получаем объект File по file\_id

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

# Скачиваем файл

file\_path = file.file\_path

download\_path = "/home/alexandervolzhanin/pet-project/CONSPECTIUS/app/audio/audio\_message.ogg"

await message.bot.download\_file(file\_path, download\_path)

logger.info(f"Аудиосообщение сохранено")

conspect = main\_processing()

logger.info(f"Конспект получен")

# Сохранение файла в .docx

try:

txt\_to\_docx(conspect)

except Exception as err:

await message.answer(error)

logger.error(f"Ошибка при конвертировании конспекта: {err}")

try:

# Путь к вашему файлу

file\_path = '/home/alexandervolzhanin/pet-project/CONSPECTIUS/app/received\_txt/example.docx'

# Создаем объект InputFile

input\_file = FSInputFile(file\_path)

# Отправляем файл

await message.reply\_document(input\_file, caption="Ваш конспект: ")

# Удаляем файлы

os.remove(file\_path)

os.remove(download\_path)

logger.info("Файл скинут")

except Exception as err:

await message.answer(error)

logger.error(f"Ошибка при пересылке файла: {err}")

##### Загрузка аудиофайла и транскрибация

import assemblyai as aai

import time

from loguru import logger

import os

from dotenv import load\_dotenv

@logger.catch

def transcribing\_aai() -> str:

load\_dotenv()

try:

# API key

aai.settings.api\_key = os.getenv("ASSEMBLY\_AI\_API")

logger.info("Обработка ASSEMBLY\_AI\_API прошла успешно")

except Exception as err:

logger.error(f"Ошибка при обработке ASSEMBLY\_AI\_API: {err}")

transcriber = aai.Transcriber()

audio\_url = (

"/home/alexandervolzhanin/pet-project/CONSPECTIUS/app/audio/audio\_message.ogg"

)

config = aai.TranscriptionConfig(language\_code="ru")

try:

logger.info("Начало транскрибации")

start\_time = time.perf\_counter()

transcript = transcriber.transcribe(audio\_url, config)

end\_time = time.perf\_counter()

logger.info("Конец транскрибации")

logger.info(f"Время выполения: {end\_time - start\_time}")

except Exception as err:

logger.error(f"Ошибка при транскрибации: {err}")

return transcript.text

##### Экспорт конспекта в формат .docx

from docx import Document

import os

def txt\_to\_docx(text: str) -> None:

# Создаем объект документа

doc = Document()

# Добавляем содержимое переменной в документ

doc.add\_paragraph(text)

# Определяем путь к папке

folder\_path = '/home/alexandervolzhanin/pet-project/CONSPECTIUS/app/received\_txt'

# Полный путь к файлу

file\_path = os.path.join(folder\_path, 'example.docx')

# Сохраняем документ в файл с расширением .docx

doc.save(file\_path)

Экспериментальная часть

#### 1. Подготовка окружения

##### 1.1. Установка Python и необходимых инструментов

Для реализации телеграм-бота потребуются следующие инструменты и библиотеки:

* **Python**: Язык программирования для написания кода.
* **Poetry**: Инструмент для управления зависимостями и виртуальным окружением.
* **Aiogram**: Библиотека для взаимодействия с API Telegram.
* **Python-docx**: Библиотека для работы с документами формата .docx.
* **OpenAI**: Для использования сервиса транскрибации и обработки текста.
* **AssemblyAI**: Для транскрибации аудиофайлов.

###### Шаги по установке:

1. **Установка Python**: Убедитесь, что Python установлен на вашем компьютере. Его можно скачать с [официального сайта](https://www.python.org/downloads/).
2. **Установка Poetry**: Установите Poetry, следуя инструкциям на официальном сайте.

curl -sSL https://install.python-poetry.org | python3 -

1. **Создание и активация виртуального окружения**:

poetry new telegram-bot

cd telegram-bot

poetry shell

1. **Добавление необходимых зависимостей**:

poetry add aiogram python-docx requests openai assemblyai

##### 1.2. Получение API ключей

* **Telegram API Token**: Зарегистрируйте своего бота в Telegram через @BotFather и получите API токен. Он потребуется для взаимодействия с API Telegram.
* **API ключ AssemblyAI**: Зарегистрируйтесь на сайте AssemblyAI и получите API ключ для транскрибации аудио.
* **API ключ OpenAI**: Получите API ключ на сайте OpenAI для использования его возможностей по обработке текста.

#### 2. Реализация телеграм-бота

##### 2.1. Структура проекта

Создайте основной файл для бота main.py в корневой директории проекта.

##### 2.2. Основные компоненты кода

Создаем файл main.py и импортируем необходимые модули:

from aiogram import Bot, Dispatcher

import asyncio

from loguru import logger

import os

from dotenv import load\_dotenv

from handlers import router

from logger import file\_logger

async def main():

load\_dotenv()

file\_logger()

bot = Bot(token=os.getenv("BOT\_TOKEN"))

dp = Dispatcher()

dp.include\_router(router)

await dp.start\_polling(bot)

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

try:

logger.info("Бот запущен")

asyncio.run(main())

except KeyboardInterrupt:

logger.error("Бот выключен")

##### 2.3. Реализация команд

###### Команда /start

Эта команда приветствует пользователя и предоставляет краткую информацию о возможностях бота.

from aiogram.types import Message, FSInputFile

from aiogram.filters import CommandStart

from aiogram import F, Router

import os

from cmd\_message import start\_message

import keyboards as kb

from main\_processing import main\_processing

from conversion\_txt\_to\_docx import txt\_to\_docx

from loguru import logger

from cmd\_message import error

router = Router()

# Обработчик команды /start

@logger.catch

@router.message(CommandStart())

async def cmd\_start(message: Message):

await message.answer(start\_message, reply\_markup=kb.main)

##### 2.5. Загрузка и обработка аудиофайлов

# Обработчик аудиосообщений

@router.message(F.voice)

async def handle\_voice\_message(message: Message):

await message.answer("""

🎧 Ваше аудиосообщение принято и обрабатывается. ⏳ Пожалуйста, подождите 5-15 минут. Спасибо за ваше терпение!

""")

voice = message.voice

file\_id = voice.file\_id

# Получаем объект File по file\_id

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

# Скачиваем файл

file\_path = file.file\_path

download\_path = "/home/alexandervolzhanin/pet-project/CONSPECTIUS/app/audio/audio\_message.ogg"

await message.bot.download\_file(file\_path, download\_path)

logger.info(f"Аудиосообщение сохранено")

conspect = main\_processing()

logger.info(f"Конспект получен")

# Сохранение файла в .docx

try:

txt\_to\_docx(conspect)

except Exception as err:

await message.answer(error)

logger.error(f"Ошибка при конвертировании конспекта: {err}")

try:

# Путь к вашему файлу

file\_path = '/home/alexandervolzhanin/pet-project/CONSPECTIUS/app/received\_txt/example.docx'

# Создаем объект InputFile

input\_file = FSInputFile(file\_path)

# Отправляем файл

await message.reply\_document(input\_file, caption="Ваш конспект: ")

# Удаляем файлы

os.remove(file\_path)

os.remove(download\_path)

logger.info("Файл скинут")

except Exception as err:

await message.answer(error)

logger.error(f"Ошибка при пересылке файла: {err}")

##### 2.6. Экспорт конспекта в формат .docx

from docx import Document

import os

def txt\_to\_docx(text: str) -> None:

# Создаем объект документа

doc = Document()

# Добавляем содержимое переменной в документ

doc.add\_paragraph(text)

# Определяем путь к папке

folder\_path = '/home/alexandervolzhanin/pet-project/CONSPECTIUS/app/received\_txt'

# Полный путь к файлу

file\_path = os.path.join(folder\_path, 'example.docx')

# Сохраняем документ в файл с расширением .docx

doc.save(file\_path)

##### 2.7. Запуск бота

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

try:

logger.info("Бот запущен")

asyncio.run(main())

except KeyboardInterrupt:

logger.error("Бот выключен")

#### 3. Тестирование и результаты

##### 3.1. Тестирование функциональности

Проведем тестирование основных функций бота:

1. **Загрузка аудиофайла и транскрибация**:
   * Пользователь загружает аудиофайл.
   * Бот отправляет аудиофайл на транскрибацию в AssemblyAI.
   * После файл с текстом попадает в Chat gpt
   * Полученный текст сохраняется как конспект.
   * Бот создает документ .docx и отправляет его пользователю.

##### 3.2. Результаты тестирования

В ходе тестирования были проверены следующие аспекты:

* **Точность транскрибации**:
  + Транскрибация аудиофайлов выполняется корректно, текст совпадает с содержанием аудио.
  + В случае низкого качества аудио, точность транскрибации может снизиться.
* **Создание и сохранение конспектов**:
  + Все введенные и транскрибированные тексты сохраняются корректно.
  + Конспекты успешно экспортируются в формат .docx.
* **Удобство использования**:
  + Интерфейс бота прост и интуитивно понятен.
* **Использование ресурсов**:
  + Бот эффективно использует API Telegram и AssemblyAI.
  + Время отклика на команды и транскрибацию аудиофайлов находится в приемлемых пределах.

Вывод

В ходе выполнения курсовой работы был разработан телеграм-бот для создания конспектов, который позволяет пользователям легко и быстро создавать текстовые заметки и сохранять их в формате .docx.

Разработка включала в себя следующие этапы:

1) Анализ и выбор инструментов:

1.1 Для разработки был выбран язык программирования Python из-за его богатого набора библиотек и простоты использования.

1.2 Использовались библиотеки aiogram для взаимодействия с API Telegram, python-docx для создания документов и requests для работы с внешними API.

1.3 Сервисы OpenAI и AssemblyAI были использованы для обработки текста и транскрибации аудиофайлов соответственно.

2) Проектирование и реализация:

1.1 Была разработана архитектура бота, включающая основные функции, такие как создание конспектов, транскрибация аудиофайлов, обработка текста и экспорт конспектов в формат .docx.

1.2 Бот был реализован с использованием асинхронных возможностей Python, что обеспечивает быструю обработку запросов и повышает отзывчивость системы.

3) Тестирование и анализ:

1.1 Бот был протестирован на предмет корректности выполнения основных функций.

1.2 Тестирование показало, что все функции работают стабильно, текстовые конспекты создаются и сохраняются корректно, а экспорт в формат .docx выполняется без ошибок.

4) Результаты и выводы:

1.1 Телеграм-бот успешно выполняет поставленные задачи и является удобным инструментом для создания и хранения конспектов.

1.2 В ходе эксперимента были выявлены возможности для улучшения, такие как оптимизация обработки аудиофайлов и улучшение интерфейса пользователя.

1.3 В целом, разработанный телеграм-бот демонстрирует потенциал использования современных технологий и сервисов для автоматизации создания конспектов и может быть успешно применен в различных образовательных и профессиональных сферах.

Список литературы

1. Python Software Foundation. Python Language Reference, version 3.9 [Электронный ресурс]. URL: https://www.python.org (дата обращения: 14.06.2024).

2. Aiogram Official Documentation [Электронный ресурс]. URL: https://docs.aiogram.dev/en/latest/ (дата обращения: 14.06.2024).

3. Python-Docx Documentation [Электронный ресурс]. URL: https://python-docx.readthedocs.io/en/latest/ (дата обращения: 14.06.2024).

4. Poetry Documentation [Электронный ресурс]. URL: https://python-poetry.org/docs/ (дата обращения: 14.06.2024).

5. OpenAI API Documentation [Электронный ресурс]. URL: https://beta.openai.com/docs/ (дата обращения: 14.06.2024).

6. AssemblyAI API Documentation [Электронный ресурс]. URL: https://docs.assemblyai.com/ (дата обращения: 14.06.2024).

7. Telegram Bot API [Электронный ресурс]. URL: https://core.telegram.org/bots/api (дата обращения: 14.06.2024).

8. Van Rossum G., Warsaw B., Coghlan N. PEP 8 – Style Guide for Python Code [Электронный ресурс]. URL: https://peps.python.org/pep-0008/ (дата обращения: 14.06.2024).

9. Real Python. Python Asyncio: The Complete Guide [Электронный ресурс]. URL: https://realpython.com/async-io-python/ (дата обращения: 14.06.2024).

10. Telegram BotFather [Электронный ресурс]. URL: https://core.telegram.org/bots#botfather (дата обращения: 14.06.2024).