МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА ЦИФРОВОГО АССИСТЕНТА ДЛЯ ОНЛАЙН-ВСТРЕЧ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Кожевникова Егора Дмитриевича

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванов

Заведующий кафедрой
доцент, к. ф.-м. н. _____ С. В. Миронов

Саратов 2025

ВВЕДЕНИЕ

В последние годы онлайн-встречи стали неотъемлемой частью деловой, образовательной и социальной коммуникации. Платформы для видеоконференций, такие как Google Meet, Zoom и Microsoft Teams, широко используются для проведения совещаний, переговоров, лекций и вебинаров. Вместе с ростом их популярности возрастает спрос на инструменты, позволяющие автоматизировать и упростить сопровождение таких мероприятий.

В рамках данной выпускной квалификационной работы, выполненной по коммерческому заказу от компании ТИМ ФОРС, предлагается разработка цифрового ассистента для онлайн-встреч.

Цель проекта — создание программного решения, способного подключаться к видеоконференциям, записывать и обрабатывать аудио- и видеопотоки, делать резюме встречи, а также обеспечивать последующее хранение и использование полученных данных. Цифровой ассистент должен обладать модульной архитектурой и эффективно взаимодействовать с платформами видеосвязи и обеспечивать надёжную работу при различных условиях эксплуатации.

Чтобы достичь поставленной цели, необходимо реализовать следующие задачи:

1. Проанализировать существующие решения и технологии, применяемые для автоматизации онлайн-встреч.
2. Разработать архитектуру цифрового ассистента с учётом требований к модульности, надёжности и расширяемости.
3. Реализовать сервисы для подключения к видеоконференции, записи аудио- и видеопотоков. Обеспечить синхронизацию аудио и видео, а также автоматическое сохранение и конвертацию файлов.
4. Настроить систему хранения, а также интерфейсы для взаимодействия с пользователем и внешними сервисами.

1 Анализ предметной области

1.1 Обзор существующих решений

На рынке представлено несколько зрелых решений для автоматизации онлайн-встреч: Otter.ai, Zoom AI Companion, Cisco Webex Assistant, Microsoft Teams Premium и Fireflies.ai. Они предлагают функции транскрипции, генерации заметок, выделения ключевых моментов и интеграции с другими сервисами. Otter.ai и Fireflies.ai позволяют даже генерировать структурированные отчёты с задачами и участниками встречи. Однако большинство решений англоязычны, ограничены в экосистемах или платны [1].

Что касается open-source решений, то полноценных аналогов практически нет. Есть отдельные библиотеки (DeepSpeech, Kaldi, Wav2Vec 2.0), но собрать их в единую систему сложно — требуется знание медиапотоков, сетевых протоколов и обработки данных. Это создаёт запрос на решение, которое будет более доступным, чем коммерческие SaaS-платформы, но при этом целостнее, чем набор open-source компонентов.

1.2 Постановка задачи

Компания ТИМ ФОРС поставила задачу разработать цифрового ассистента SmartMeetNote, который автоматически подключается к онлайн-встречам, записывает медиапотоки, синхронизирует их и сохраняет результат в удобном виде. Основные цели:

- Автоматизация участия в конференциях,
- Запись аудио и видео,
- Распознавание речи и генерация структурированного резюме,
- Хранение и поиск встреч через Telegram-бота.

Решение должно быть простым в использовании, масштабируемым и интегрируемым с внутренними сервисами компании.

2 Архитектура и реализация SmartMeetNote

SmartMeetNote представляет собой модульное приложение, разработанное с использованием контейнеризации (Docker), состоящее из двух основных частей: Python-модуля и TypeScript-модуля. Такое разделение обусловлено необходимостью решения различных задач: от взаимодействия с Telegram и обработки речи до записи видео и работы с медиапотоками. Контейнеризация обеспечивает изоляцию компонентов, переносимость системы и удобство развёртывания.

Python-модуль отвечает за управление Telegram-ботом, работу с базой данных через ORM SQLAlchemy, распознавание речи с помощью OpenAI Whisper и генерацию структурированного резюме встречи с использованием Yandex GPT. Он реализован на Python 3.10 с применением библиотек aiogram для взаимодействия с Telegram API, Selenium для автоматизации Google Meet в headless-режиме, а также asyncio для асинхронной обработки задач.

TypeScript-модуль, напротив, фокусируется на видеозаписи через Playwright. Он запускает браузер в headed-режиме, подключается к встрече, записывает видеопоток и передаёт результат в Python-модуль для дальнейшей обработки. Использование TypeScript вместо Python обусловлено лучшей поддержкой типизации, упрощённой работой с DOM-элементами и более надёжным поведением движка Chromium при работе с медиапотоками.

Между модулями реализовано чёткое взаимодействие через файловый обмен и REST API. После завершения встречи медиафайлы обрезаются, конвертируются в формат .mp4 с помощью FFmpeg и объединяются по временным меткам начала и окончания. Полученный аудиофайл отправляется в OpenAI API для транскрипции через модель Whisper-1. Если файл превышает допустимый размер, он обрезается или разбивается на части. Транскрипция затем направляется в Yandex GPT для создания структурированного резюме встречи, включающего описание тем, список задач, участников и план следующих действий.

Для хранения данных используется SQLite через ORM SQLAlchemy. Для каждой встречи сохраняются chat_id пользователя, заголовок, временная метка, транскрипция, резюме и ключевые слова. Реализованы методы поиска по ключевым словам, фильтрации по дате и удалению старых записей. Поиск выполняется с использованием регулярных выражений и fuzzy-поиска, что позволяет

находить совпадения даже при неточном запросе.

Telegram-бот реализован на aiogram с поддержкой FSM (Finite State Machine) для управления состояниями пользователей. Он предоставляет команды /join и /join_with_video для начала записи, /search и /archive для поиска и просмотра истории, а также /summary для получения резюме загруженного файла. Бот работает в асинхронном режиме, поддерживает групповые чаты и корректно обрабатывает ошибки, информируя пользователя о состоянии встречи.

Проект полностью реализован в виде двух Docker-контейнеров, что позволило отделить логику обработки медиа от остальной системы. Это дало возможность масштабировать отдельные части независимо друг от друга, упростило тестирование и сделало систему переносимой. Файл docker-compose.yml определяет сетевые настройки, точки монтирования, переменные окружения и зависимости между контейнерами, что позволяет запускать всю систему одной командой.

Такая архитектура делает SmartMeetNote гибким решением, способным адаптироваться под различные сценарии использования, будь то одиночная встреча или одновременная запись нескольких конференций.

2.1 Подключение к видеоконференциям

Одной из ключевых задач при разработке цифрового ассистента SmartMeetNote стало обеспечение надёжного автоматического подключения к онлайн-встречам. Поскольку большинство популярных платформ видеосвязи, таких как Google Meet, не предоставляют полноценного API для программного присоединения к конференциям, была реализована эмуляция поведения пользователя с помощью инструментов автоматизации браузера.

Для этих целей были выбраны две технологии: Selenium — для Python-модуля, отвечающего за запись аудио, и Playwright — для TypeScript-модуля, необходимого при записи видео. Оба подхода позволяют открывать страницы, заполнять формы, кликать элементы интерфейса и взаимодействовать с медиапотоками так же, как это делает реальный пользователь.

Аудио-бот реализован на базе Selenium WebDriver и работает в headless-режиме, что позволяет ему оставаться незаметным для других участников встречи. При подключении он автоматически вводит своё имя — MeetNote-audio-bot, принимает условия использования сервиса и остаётся в конференции до её завершения. Микрофон активируется принудительно, чтобы обеспечить захват

звука, камера остается выключенной, что снижает нагрузку на систему и уменьшает объём передаваемых данных.

Видео-бот, напротив, работает в headed-режиме через Playwright, поскольку Google Meet блокирует попытки записи в headless-браузере. Он действует как полноценный участник встречи: вводит уникальное имя MeetNote-video-bot-#<timestamp>, запрашивает доступ к камере и микрофону (эмулируя их), после чего начинает видеозапись. Этот режим позволяет более точно воспроизводить действия человека и минимизировать вероятность блокировки со стороны платформы.

Оба бота реализуют механизм обнаружения начала и окончания встречи. Система отслеживает изменение URL страницы, количество участников (если остаётся один — бот) и длительность бездействия. Эти параметры проверяются регулярно, с заданными интервалами, чтобы избежать преждевременного завершения записи или её бесконечного продолжения.

Кроме того, боты умеют обрабатывать модальные окна, которые могут появиться при входе в конференцию. Например, если отображается сообщение "You can't join this call система прекращает попытки подключения. Если появляется уведомление "Got it бот автоматически нажимает Enter, чтобы продолжить участие во встрече.

Такая реализация позволила создать универсальное решение, способное подключаться к различным платформам видеосвязи, сохраняя высокую степень автономности и надёжности. В дальнейшем планируется расширять поддержку новых платформ, таких как Яндекс Телемост, Zoom и Microsoft Teams, с учётом особенностей их интерфейсов и политик безопасности.

2.2 Запись медиапотоков

Для автоматизации онлайн-встреч необходимо записывать как аудио-, так и видеопотоки. SmartMeetNote реализует это с помощью двух модулей: Python-модуля для аудиозаписи и TypeScript-модуля для видеозаписи. Такое разделение позволяет эффективно обрабатывать разные типы данных и обеспечивает надёжность при работе с медиафайлами.

Аудиозапись осуществляется через JavaScript API `MediaRecorder`, внедрённый в страницу Google Meet. Это позволяет захватывать звуковой поток без установки дополнительного ПО. Браузер динамически добавляет новые элементы <audio> при подключении участников, поэтому система регулярно проверяет

наличие новых источников и добавляет их в список записываемых [2].

Видеозапись выполняется на уровне движка Chromium через функцию `recordVideo` из библиотеки Playwright. TypeScript-модуль запускает браузер в headed-режиме, так как headless-режим блокирует доступ к медиапотокам в Google Meet [3]. После завершения встречи видеофайлы сохраняются и передаются в Python-модуль для дальнейшей обработки .

Синхронизация аудио и видео происходит по временным меткам начала и окончания встречи. Для объединения файлов используется FFmpeg. Ниже приведён пример команды слияния, которая учитывает временные несоответствия между дорожками:

```
color=c=black:s=1280x720:d={gap_start}[black1]; \
[0:v]trim=duration={video_file_duration},setpts=PTS-STARTPTS[vcontent]; \
color=c=black:s=1280x720:d={gap_end}[black2]; \
[black1][vcontent][black2]concat=n=3:v=1:a=0[v]
```

Эта команда создаёт чёрные полосы в начале и конце видео, если аудио короче видео, тем самым сохраняя синхронизацию и визуальную целостность.

2.3 Распознавание речи и генерация резюме

После завершения записи аудиофайл отправляется на обработку. Для преобразования речи в текст используется модель Whisper-1 от OpenAI, вызываемая через API. Этот инструмент показывает высокую точность даже на зашумлённых записях и поддерживает несколько языков, включая русский [4].

Пример вызова модели:

```
transcript = self.client.audio.translations.create(
    file=audio_file,
    model="whisper-1"
)
```

Полученная транскрипция передаётся в Yandex GPT для создания структурированного резюме встречи. Модель анализирует текст, выделяет ключевые моменты, задачи, ответственных и сроки выполнения. Такой подход позволяет пользователю получать не просто запись разговора, а готовый документ с выводами и действиями.

Асинхронная реализация обеспечивает отзывчивость системы даже при длительной обработке больших объёмов данных. Результат сохраняется в базу и отправляется пользователю через Telegram-бота.

2.4 Хранение данных и пользовательский интерфейс

Для хранения информации о встречах используется SQLite через ORM SQLAlchemy. Такой выбор обусловлен простотой развёртывания, отсутствием необходимости внешнего сервера и гибкостью при переходе на более мощные СУБД в будущем [5]. Каждая запись содержит:

- chat_id — для привязки к пользователю,
- title — название встречи,
- timestamp — дату и время,
- transcript — текст транскрипции,
- summary — резюме встречи,
- keywords — ключевые слова для поиска.

Все данные сохраняются автоматически после завершения встречи. Пользователь может искать встречи по ключевым словам или диапазону дат, просматривать историю, удалять или переименовывать записи. Реализовано это через асинхронный метод search(), который позволяет эффективно фильтровать данные:

```
async def search(self, query: str, chat_id: str, start_date: Optional[datetime]
= None, end_date: Optional[datetime]
= None) -> List[MeetingEntry]:
    ...
    if query:
        query_lower = query.lower().strip()
        for entry in entries:
            print(entry.timestamp)
            title_match = check_partial_matches(
                query_lower, entry.title.lower())
            summary_match = check_partial_matches(
                query_lower, entry.summary.lower())
            transcript_match = check_partial_matches(query_lower,
                                                     entry.transcript.lower())
            if any((title_match, summary_match, transcript_match)):
                matched_entries.append(entry)
        entries = matched_entries
    return entries
```

Пользовательский интерфейс реализован через Telegram-бота, что делает его доступным без установки дополнительного ПО. Основные команды:

— /join — начать запись встречи,
— /search — найти встречу по ключевым словам,
— /archive — просмотреть историю встреч,
— /summary — получить результат в виде .docx.

Каждая команда вызывает соответствующий обработчик, использующий FSM для управления состояниями и callback-запросы для подтверждения действий. Это делает взаимодействие с ботом интуитивно понятным и надёжным даже при множественных одновременных запросах [6].

3 Пример использования

Для демонстрации работы цифрового ассистента SmartMeetNote рассмотрим типичный сценарий его использования при организации онлайн-встречи. Предположим, что необходимо автоматически зафиксировать обсуждение, чтобы позже вернуться к нему или передать коллегам.

Пользователь открывает Telegram и отправляет команду `/join`, указав ссылку на конференцию (рис. 1).

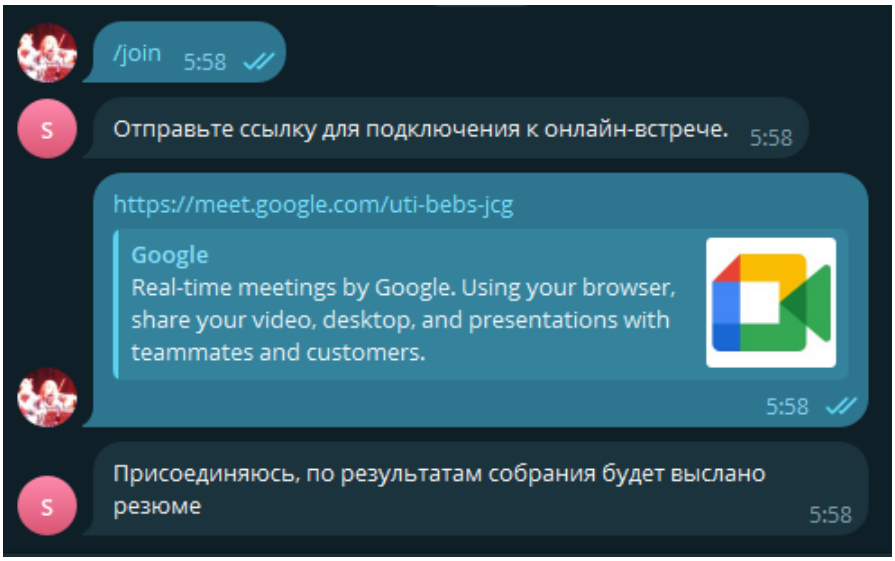

Рисунок 1 – Команда для присоединения ко встрече

Python-модуль запускает процесс подключения к указанной ссылке через Selenium. Бот эмулирует действия обычного участника: вводит уникальное имя, принимает условия использования сервиса и присоединяется к встрече. Микрофон активируется принудительно, камера остаётся выключенной, чтобы минимизировать нагрузку.

Если требуется запись видео, используется команда `/join_with_video`. В этом случае запускается TypeScript-модуль, который управляет браузером в headed-режиме, обеспечивая корректную работу медиапотоков.

Запись встречи продолжается до тех пор, пока система не определит её завершение. Для этого отслеживаются: изменение URL страницы, количество оставшихся участников (если остаётся только один) и длительность бездействия.

Результат сохраняется в базе данных. Каждая запись содержит заголовок, временную метку, транскрипцию, резюме и ключевые слова. Это позволяет быстро находить нужные встречи и организовывать их хранение.

После обработки пользователь получает уведомление в Telegram с кратким содержанием встречи и прикреплённым .docx-файлом, содержащим полный текст транскрипции и резюме. При необходимости можно найти встречу позже через команду `/search`, указав ключевые слова или диапазон дат.

Для просмотра всех записей за определённый период используется команда `/archive`. Система выводит список всех доступных записей, где можно выбрать конкретную встречу, скачать файл, переименовать или удалить запись. Выбор осуществляется через inline-кнопки, что делает интерфейс удобным и интуитивно понятным (рис. 2).

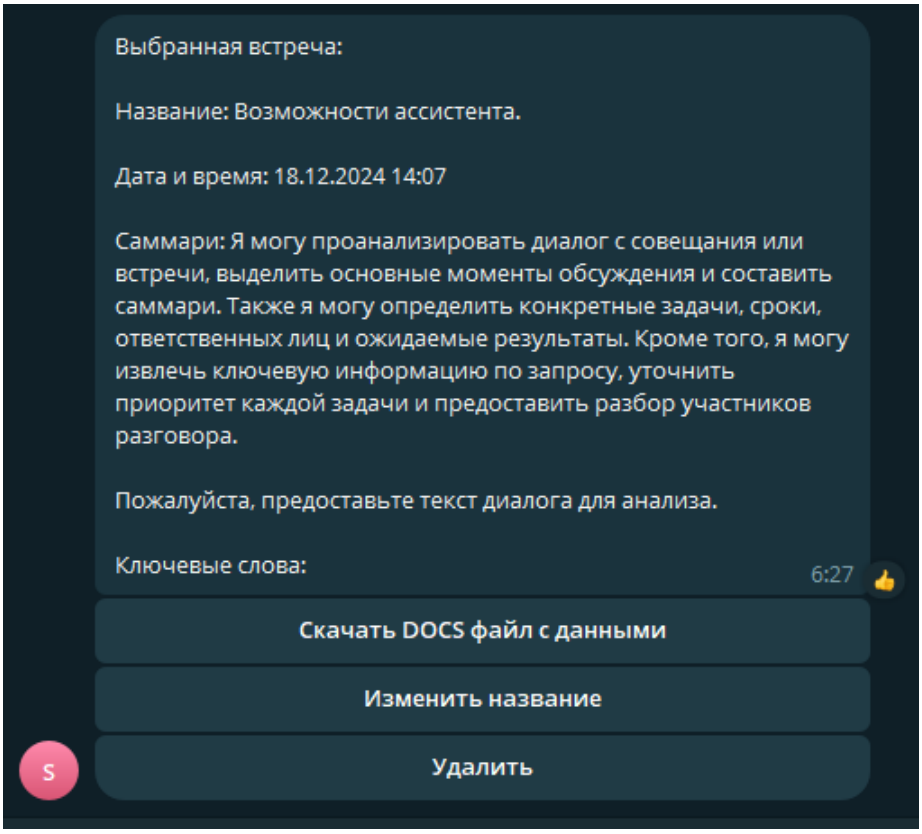

Рисунок 2 – Просмотр записи

Кроме прямой записи встреч, SmartMeetNote позволяет работать с уже имеющимися аудио- и видеофайлами. Например, можно отправить голосовое сообщение или прикрепить файл, добавив команду `/summary`, чтобы получить текстовое резюме.

ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы был разработан цифровой ассистент SmartMeetNote — программное решение для автоматического сопровождения онлайн-встреч. Система позволяет подключаться к видеоконференциям Google Meet, записывать аудио- и видеопотоки, распознавать речь и формировать текстовое представление встречи, генерировать структурированное резюме, а также хранить данные и предоставлять пользователю удобный способ поиска и управления записями. Разработка была выполнена по коммерческому заказу от компании ТИМ ФОРС, что позволило учесть реальные потребности корпоративного пользователя и создать инструмент, максимально соответствующий внутренним процессам. Ассистент реализован в виде двух модулей: Python-части, отвечающей за взаимодействие с пользователем через Telegram и обработку аудиозаписи, и TypeScript-модуля, обеспечивающего запись видео. Такое разделение позволило достичь модульности, расширяемости и независимости компонентов друг от друга.

Для реализации использовались современные технологии: Selenium и Playwright для автоматизации браузера и эмуляции поведения участника встречи, Whisper-1 для распознавания речи, Yandex GPT для генерации резюме, FFmpeg для обработки и объединения медиапотоков, aiogram для построения интерфейса взаимодействия с пользователем. Особое внимание было уделено вопросам надёжности — реализованы механизмы отслеживания завершения встречи, обработки ошибок, логирования действий, отправки уведомлений и сохранения данных в базе SQLite, что делает систему устойчивой к внештатным ситуациям и готовой к использованию в ежедневной работе команды.

Актуальность проекта обусловлена всё более широким распространением онлайн-встреч в бизнесе, образовании и других сферах, где участники конференций вынуждены тратить время на фиксацию важных моментов, переписывание решений и планирование дальнейших действий. SmartMeetNote берёт на себя эти задачи, позволяя сосредоточиться на содержательной части беседы.

Реализация цифрового ассистента SmartMeetNote стала успешным примером использования технологий искусственного интеллекта, автоматизации браузера и контейнеризации для решения реальной задачи бизнеса. Разработанная система показала высокую эффективность при тестировании и может быть рекомендована к внедрению в ежедневную работу распределённых команд.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Fireflies features [Электронный ресурс]. — 2025. — URL: https://fireflies.ai/product/features (Дата обращения: 12.03.2025). Загл. с экрана. Яз. англ.

2 Захват и запись аудио в браузере: Использование MediaStream и MediaRecorder API [Электронный ресурс]. — 2025. — URL: https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder_API (Дата обращения: 15.10.2024). Загл. с экрана. Яз. рус.

3 Playwright Documentation: Modern Web Testing Library. — 2025. — URL: https://playwright.dev/docs/intro (Дата обращения: 15.10.2024). Загл. с экрана. Яз. англ.

4 *Radford, A.* Robust Speech Recognition via Large-Scale Weak Supervision / A. Radford, J. W. Wu, D. Amodei et al. — 2022. — URL: https://cdn.openai.com/papers/whisper.pdf (Дата обращения: 15.10.2024). Загл. с экрана. Яз. англ.

5 SQLite Database Engine: Official Documentation [Электронный ресурс]. — 2025. — URL: https://www.sqlite.org/docs.html (Дата обращения: 15.10.2024). Загл. с экрана. Яз. англ.

6 Aiogram Documentation [Электронный ресурс]. — 2023. — URL: https://docs.aiogram.dev/ (Дата обращения: 5.02.2025). Загл. с экрана. Яз. англ.