МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Кафедра технической кибернетики

Отчет по лабораторной работе №2

Дисциплина: «Инженерия данных»

Выполнил: Осепян С.М.

Группа: 6233-010402D

Самара 2025

Цели работы

1. Освоить оркестровку в n8n: триггеры, ветвления, бинарные данные, обработка ошибок.
2. Интегрировать внешние инструменты: ffmpeg, открытые STT-модели (Whisper и аналоги), LLM для перевода.
3. Реализовать полный цикл: Telegram Bot → загрузка/скачивание видео → извлечение аудио → распознавание речи → перевод EN→RU → формирование субтитров → инкрустирование в видео → отправка ответа в бота.

Постановка задачи (пайплайн)

1. Telegram Bot принимает либо ссылку на видео, либо сам файл видео.
2. Если пришла ссылка — скачать видео; если пришёл файл — использовать его напрямую. В обоих случаях извлечь аудиодорожку с помощью ffmpeg.
3. Сгенерировать субтитры (EN), используя auto_subtitle.
4. Выполнить перевод EN→RU с помощью LLM из HuggingFace (допустим API; предпочтительно — собственный сервер vLLM/LLama).
5. Добавить полученные (RU) субтитры в исходное видео.
6. Отправить результат обратно пользователю в Telegram.

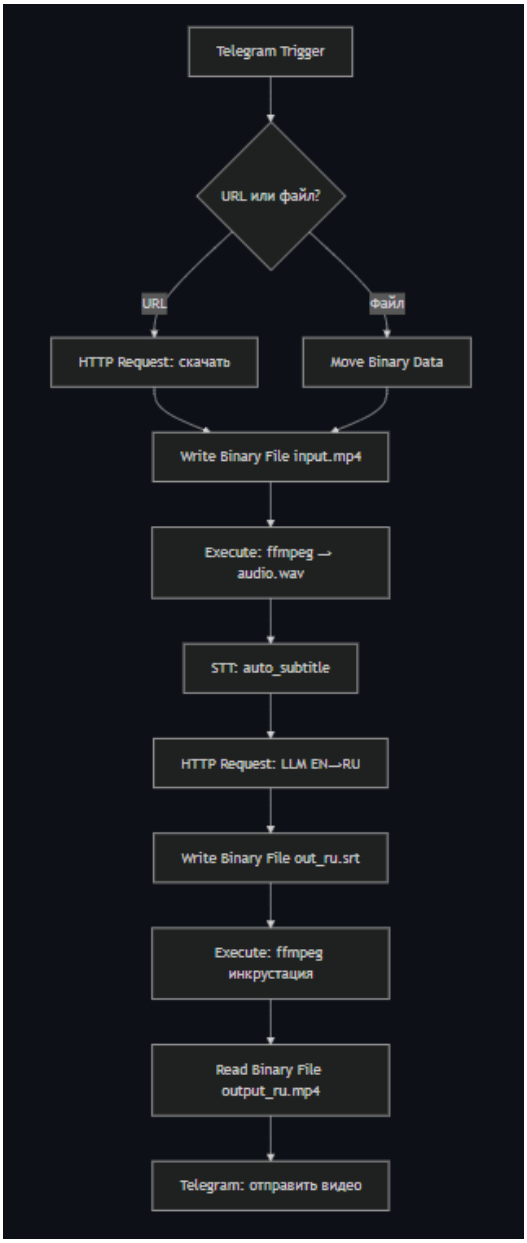

Рисунок 1 – схема работы пайплайна.

Окружение и сервисы

Обязательно:

- n8n (Docker)
- ffmpeg для работы с видео

Для STT:

- auto_subtitle как бинарь + Execute Command, или
- auto_subtitle в докер контейнер с реализованным API

Одно из (для перевода):

- vLLM/llama/llama.cpp сервер с Llama/Qwen/Mistral-Instruct (локально или удалённо), или
- HuggingFace Inference API (допустимо, но желательно локально)

1. Архитектура системы

Система состоит из нескольких изолированных контейнеров, взаимодействующих через HTTP-запросы:

1. Сервис обработки видео (video-processing):
    - Назначение: Извлекает аудио из видео (/extract-audio) и накладывает субтитры (/burn-subtitles).
    - Технологии: Образ на основе Dockerfile.video-processing (вероятно, включает ffmpeg и Python-скрипты для API).
2. Сервис генерации субтитров (auto-subtitle-api):
    - Назначение: Преобразует аудио в английские субтитры.
    - Технологии: Может быть контейнером с auto_subtitle (бинарник) или моделью Whisper, предоставляющим API эндпоинт /asr.
3. Сервис LLM для перевода:
    - Назначение: Переводит текст субтитров с английского на русский. В вашем workflow эту роль выполняет внешний API Mistral.

n8n выступает центральным оркестратором, который последовательно вызывает эти сервисы через HTTP, передавая бинарные данные (аудио, видео, файлы SRT) между ними.

В workflow данной лабораторной работы были реализованы следующие узлы:

1. Telegram Trigger (n8n-nodes-base.telegramTrigger)
    - Назначение: Триггер workflow. Ожидает новые сообщения или изменения сообщений в Telegram.
    - Параметры: Реагирует на message и edited_message.
    - Выходные данные: Данные о сообщении, включая message.text (текст) и message.video (видеофайл).
2. Check Input Type (n8n-nodes-base.switch)
    - Назначение: Определяет тип входных данных (видеофайл или ссылка).
    - Логика:
        - Ветка file: Активируется, если message.video не пусто. Передает управление узлу Get Telegram File.
        - Ветка link: Активируется, если message.text начинается с "http". Передает управление узлу Download Video URL.
3. Get Telegram File (n8n-nodes-base.telegram)

- Назначение (активируется для file): Получает бинарный файл видео из Telegram по file_id.
- Результат: Видеофайл в бинарном формате, доступный в поле data.

4. Download Video URL (n8n-nodes-base.httpRequest)
    - Назначение (активируется для link): Скачивает видеофайл по URL из текста сообщения.
    - Результат: Видеофайл в бинарном формате, доступный в поле data.
5. Extract Audio (n8n-nodes-base.httpRequest)
    - Назначение: Отправляет видеофайл на внешний сервис (video-processing:8100) для извлечения аудиодорожки. Действие: POST /extract-audio Параметры: Передает video_file как multipart/form-data.
    - Результат: Аудиофайл в бинарном формате, доступный в поле audio_file.
6. Extract Audio & Generate SRT (n8n-nodes-base.httpRequest)
    - Назначение: Отправляет аудиофайл на сервис субтитров (auto-subtitle-api:9000) для распознавания речи и генерации субтитров на английском.
    - Действие: POST /asr
    - Параметры: Передает audio_file, указывает задачу (transcribe), язык (en) и формат вывода (srt).
    - Результат: Файл субтитров .srt на английском в бинарном формате, доступный в поле srt_file.
7. Extract from File (n8n-nodes-base.extractFromFile)
    - Назначение: Читает содержимое бинарного SRT-файла и извлекает текст для последующего перевода.
    - Операция: text (извлечь текст из файла).
    - Результат: Текст английских субтитров в поле srt_file.
8. Translate via Mistral AI (n8n-nodes-base.httpRequest)
    - Назначение: Отправляет текст английских субтитров в модель Mistral AI для профессионального перевода на русский.
    - Действие: POST к https://api.mistral.ai/v1/chat/completions
    - Параметры: Использует модель mistral-small-latest. Системный промпт инструктирует модель сохранять структуру SRT. Переводит только текст, оставляя номера и таймкоды без изменений.

- Результат: JSON-ответ с переведенным текстом в поле choices[0].message.content.

9. Convert to File (n8n-nodes-base.convertToFile)
    - Назначение: Конвертирует переведенный текст (из предыдущего узла) обратно в бинарный SRT-файл.
    - Операция: toText
    - Результат: Бинарный файл русских субтитров в поле ru_srt_file.
10. Merge (n8n-nodes-base.merge)
    - Назначение: Объединяет два потока данных: Исходный видеофайл (из узлов 3 или 4), который был сохранен в потоке данных и дошел до этого узла через правую ветку. Новый русский SRT-файл (из узла 9).
    - Режим: combineAll (объединить все входные данные).
    - Результат: Единый набор данных, содержащий и видео, и файл субтитров.
11. Burn Subtitles (n8n-nodes-base.httpRequest)
    - Назначение: Отправляет видео и SRT-файл на сервис обработки видео для "вживления" (наложения) субтитров.
    - Действие: POST /burn-subtitles на video-processing:8100
    - Параметры: Передает video_file и srt_file как multipart/form-data.
    - Результат: Готовое видео с "вшитыми" русскими субтитрами в бинарном формате, доступное в поле result_video_file.
12. Send Result Video (n8n-nodes-base.telegram)
    - Назначение: Отправляет обработанное видео обратно в Telegram.
    - Операция: sendVideo
    - Параметры: Использует result_video_file как бинарные данные. Добавляет подпись с уведомлением об успешной обработке.
13. Cleanup Files (n8n-nodes-base.httpRequest)
    - Назначение: Отправляет запрос на сервис обработки видео для очистки временных файлов.
    - Действие: GET /clear на video-processing:8100
    - Цель: Управление дисковым пространством на сервере обработки.

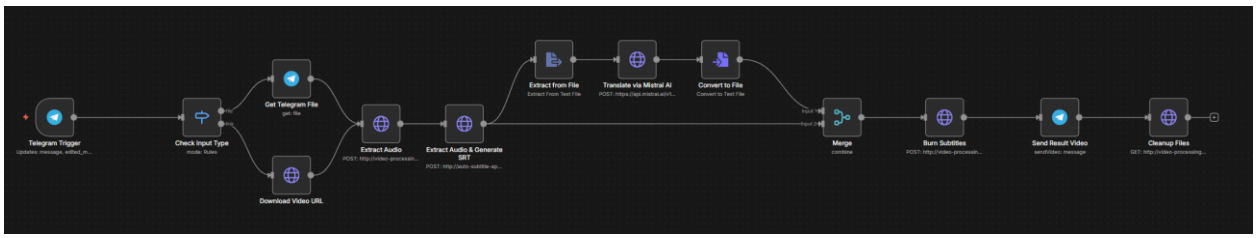

Рисунок 2 – Итоговый Пайплайн n8n

2. Развертывание системы

Был использован Telegram бот из предыдущей лабораторной работы, поэтому токен Telegram бота остался исходным.

Для развертывания архитектуры системы был использован Docker, при развертывании создаются следующие контейнеры:

1. Сервис n8n (n8n)
    - Роль: Главный оркестратор workflow. Принимает триггеры из Telegram, управляет потоком данных и вызывает другие сервисы.
    - Порт: 5678
    - Конфигурация:
        - Использует Dockerfile.n8n из репозитория для сборки.
        - Переменные окружения (API-ключи, URL сервисов) задаются через .env файл.
    - Особенность: Для работы вебхуков с Telegram (через триггер) хост с n8n должен быть доступен из интернета (используется ngrok, туннель или публичный IP).
2. Сервис обработки видео (video-processing)
    - Роль: Выполняет операции с видео: извлечение аудио и наложение (инкрустация) субтитров.
    - Порт: 8100
    - Основа: Собирается из Dockerfile.video-processing.
    - API-эндпоинты:
        - POST /extract-audio: Принимает видеофайл, возвращает аудио.
        - POST /burn-subtitles: Принимает видео и SRT-файл, возвращает видео с субтитрами.
        - GET /clear: Очистка временных файлов.
3. Сервис распознавания речи (STT) (auto-subtitle-api)
    - Роль: Преобразует аудио в английские текстовые субтитры (формат SRT).
    - Порт: 9000

- Реализация: Контейнер, предоставляющий API для утилиты auto_subtitle или модели Whisper.
- API-эндпоинт:
    - POST /asr: Принимает аудиофайл, возвращает .srt файл.

Для корректной работы Telegram Trigger необходимо создать туннель. Для этого была загружена утилита ngrok, которая предоставляет https ссылку. Она корректно не работает с российскими IP, для исправления этой ситуации включаем VPN. Далее запускаем туннель, с помощью команды ngrok http 5678, после чего в консоли появляется статус текущей сессии.

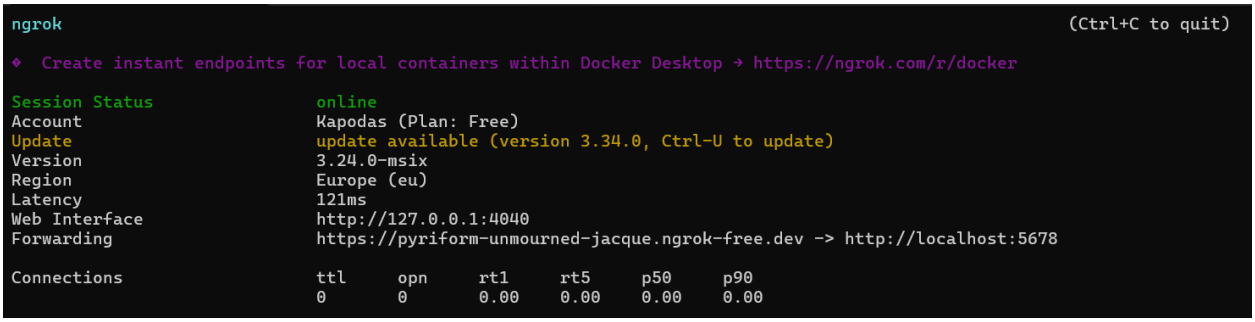

Рисунок 3 – Результат запуска ngrok по порту 5678

Здесь нам необходим https адрес указанный в Forwarding, его мы вставляем в WEBHOOK_URL для корректной работы триггера. Также, для отслеживания запросов, приходящих на сервер, заходим по ссылке указанном в Web Interface.

В нем можно увидеть, помимо статуса запроса также полностью его тело, данные пользователя, который отправил запрос, что было отправлено и тд.

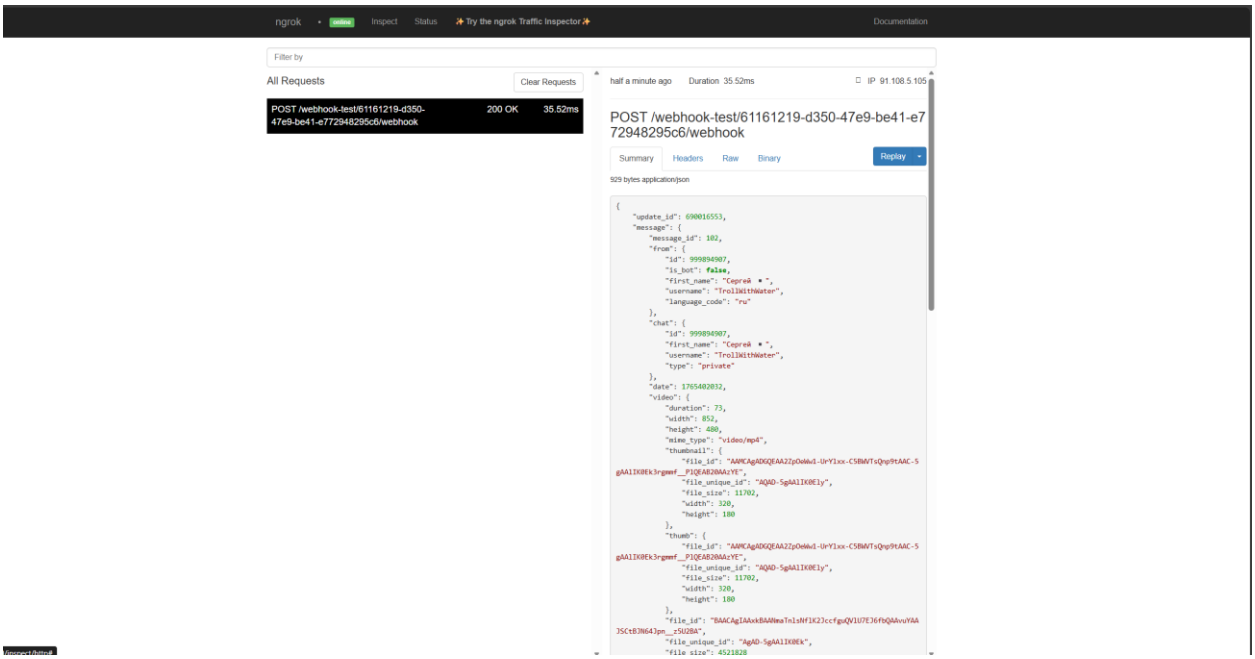

Рисунок 4 – Web Interface ngrok.

Запустить workflow можно двумя способами, 1 способ это запуск единичной отработки workflow, нажатием кнопки Execute workflow.

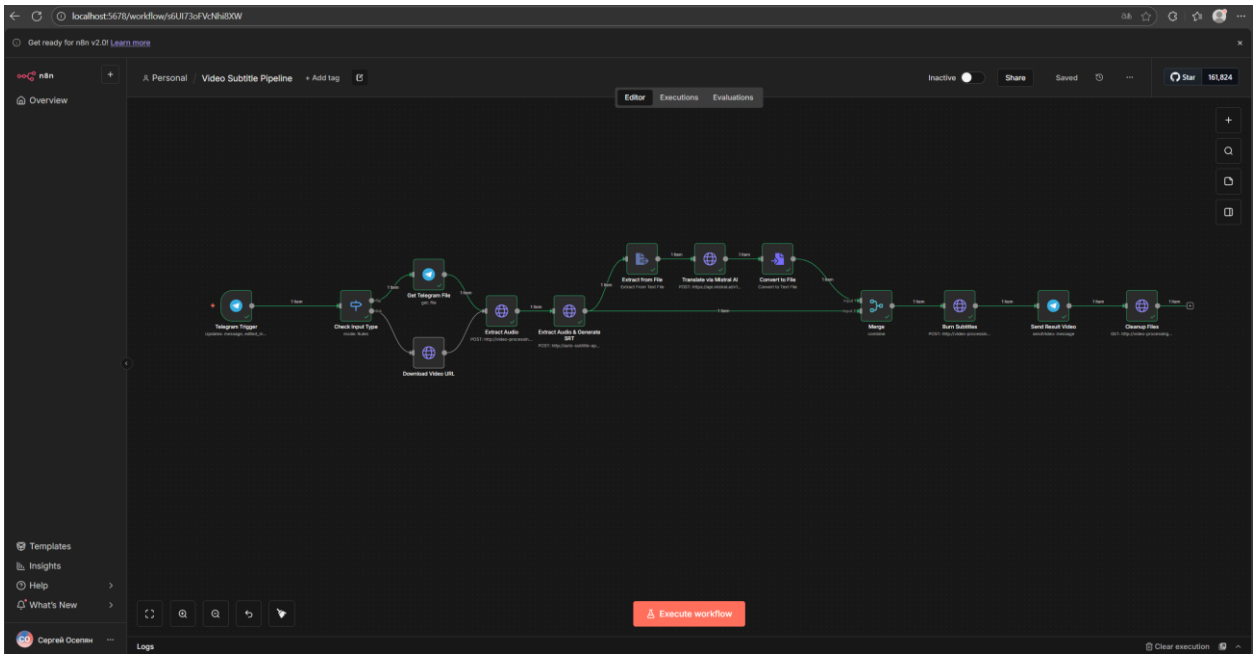

Рисунок 5 – Успешное выполнение workflow при единичной отработке workflow

Второй способ – это перевести workflow в статус Active, в таком случае он будет постоянно отслеживать чат в Telegram, ожидая события от которого срабатывает триггер (отправка видео или отправка URL ссылки на видео)

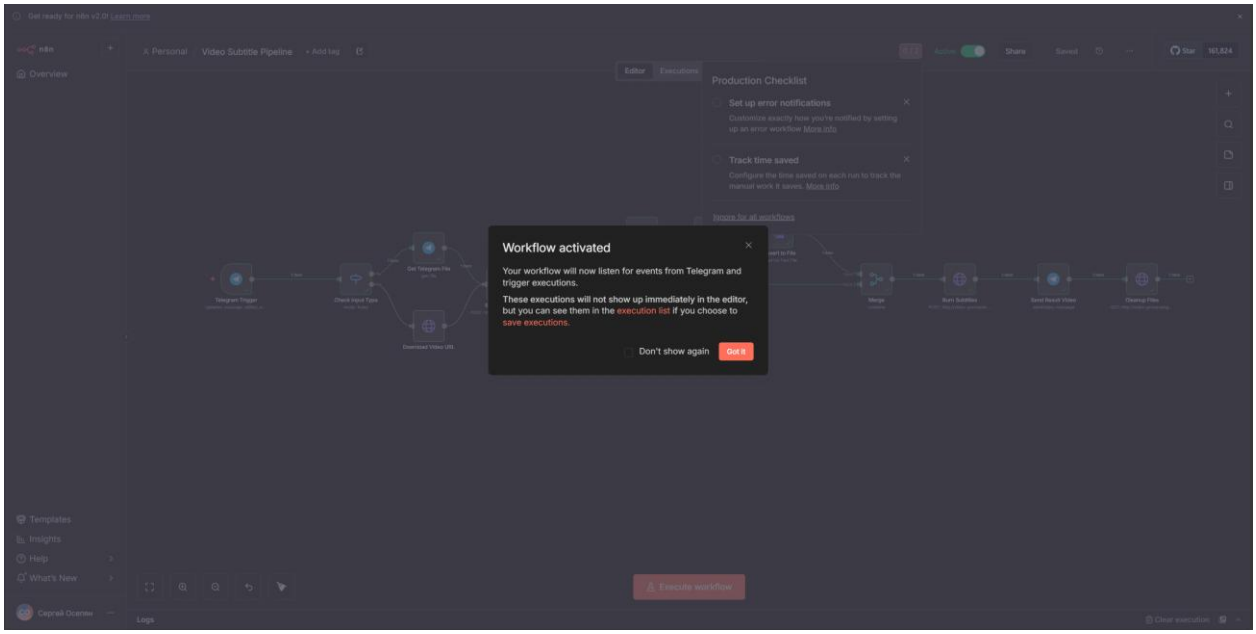

Рисунок 6 – Активация workflow.

3. Запуск и выполнение работы

Отправив видео, или ссылку на видео в чат с ботом Telegram, он начнет отработку workflow, в результате чего бот отправит видео со вставленной дорожкой с субтитрами. Результат работы бота представлены на изображениях 7 - 10

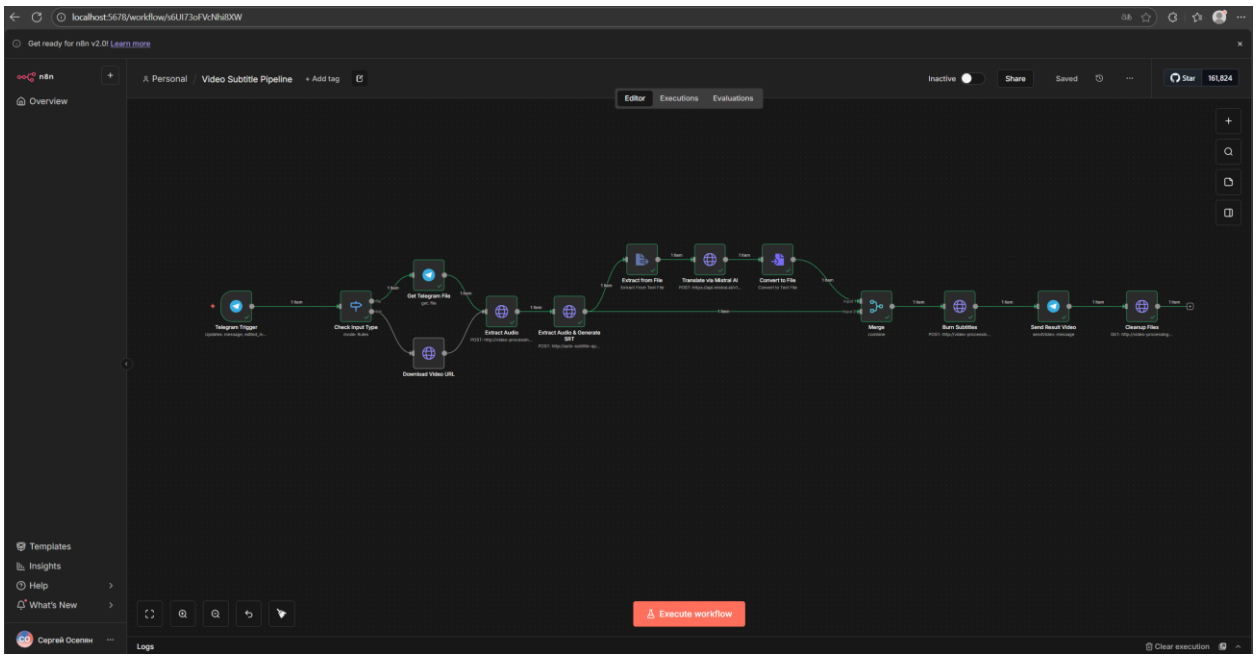

Рисунок 7 – отработка workflow при отправке видеофайла

Рисунок 8 – результат работы бота при отправке видеофайла

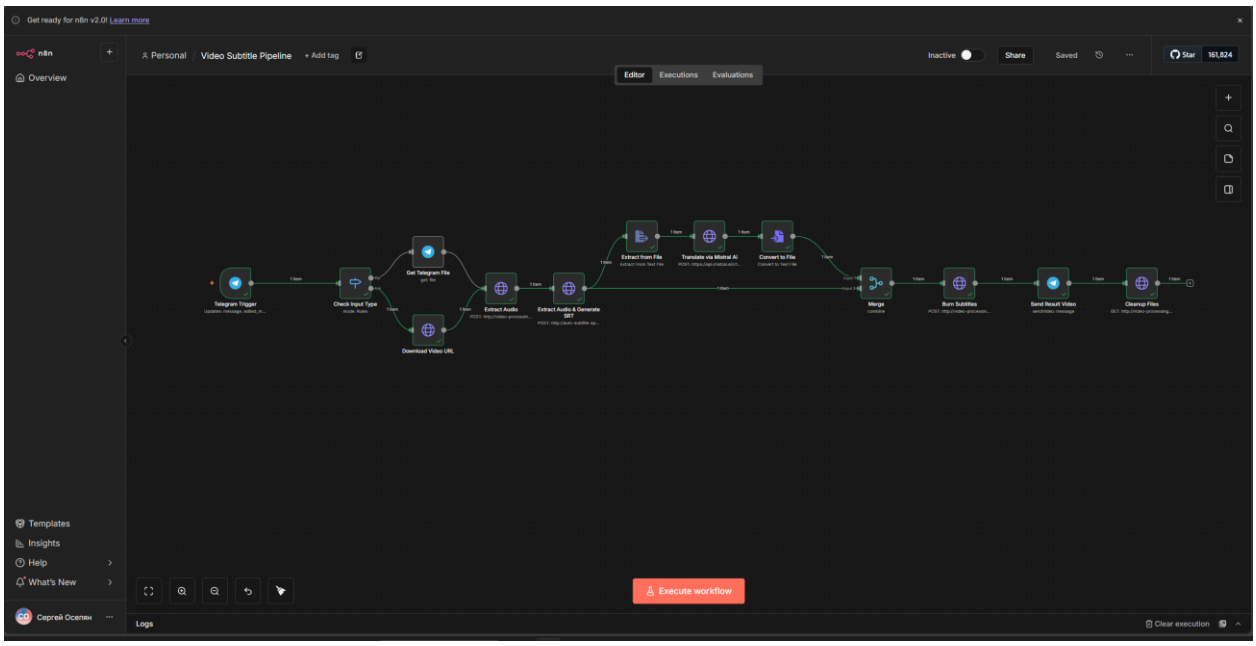

Рисунок 9 – отработка workflow при отправке ссылки на видео

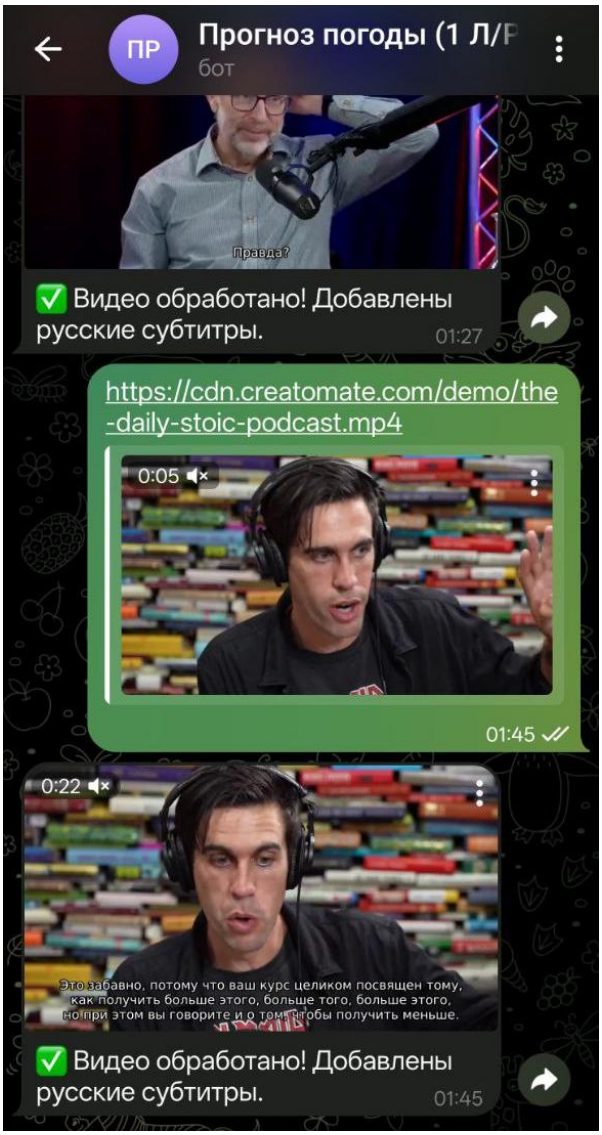

Рисунок 10 – результат работы бота при отправке ссылки на видео

ЗАКЛЮЧЕНИЕ

В рамках данной работы была успешно спроектирована, реализована и развернута распределенная система для автоматической обработки видео с целью извлечения аудио, распознавания речи, перевода субтитров и их интеграции обратно в видеофайл. Система построена по модульному принципу с использованием контейнеризации (Docker) и оркестрации workflow через n8n.

Были достигнуты поставленные цели:

1. Освоена оркестровка в n8n: В ходе работы были применены ключевые концепции n8n, включая работу с триггерами, ветвлением логики для обработки разных типов входных данных, передачей и преобразованием бинарных данных, а также реализовано базовое управление задачами и очистка временных ресурсов.
2. Интегрированы внешние инструменты и сервисы: Архитектура системы основана на взаимодействии специализированных микросервисов:
    - FFmpeg (в контейнере video-processing) для операций с мультимедиа.
    - STT-модель (Whisper) через сервис auto-subtitle-api для генерации английских субтитров.
    - Внешняя LLM (Mistral AI API) для качественного перевода текста с английского на русский с сохранением структуры временных меток SRT.
3. Реализован полный автоматизированный пайплайн: Разработанный workflow в n8n полностью автоматизирует заявленный цикл обработки:
    - Прием входных данных через Telegram-бота.
    - Маршрутизация и загрузка видео.
    - Последовательное извлечение аудио, распознавание речи, перевод текста.
    - Наложение переведенных субтитров на видео.
    - Возврат готового результата пользователю в Telegram-чат.