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»

Тема: «Применение n8n для реализации пайплайна обработки видео с использованием LLM моделей.»

Выполнила: Быкова Д.О.

Группа: 6232-010402D

Самара 2025

Задание на лабораторную работу

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.

Ход выполнения работы

В соответствии с пунктом 1 задания оркестровка процессов в рамках лабораторной работы была реализована в n8n. Схема Workflow представлена на рисунке 1.

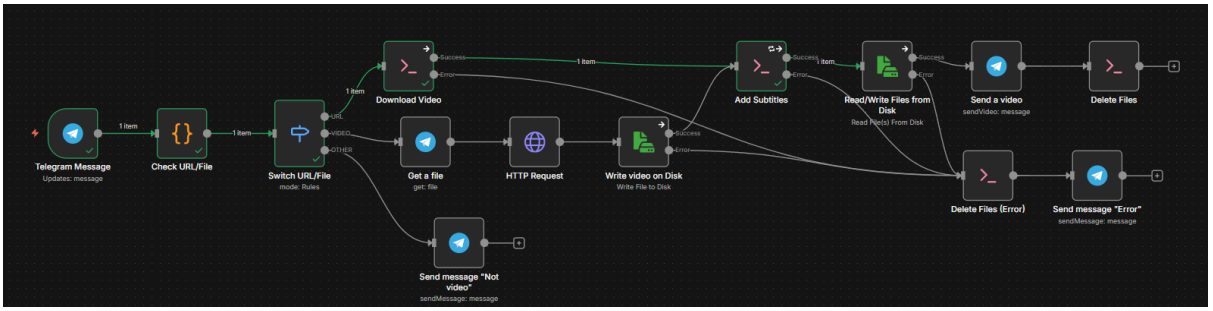

Рисунок 1 - Workflow

Первой нодой в представленном Workflow является Telegram Message - триггер на сообщение от пользователя.

Далее используется нода Code (использовала ее, потому что она показалась мне интересной). В ноде производится определение типа пришедшего сообщения: ссылка ютуб, видеофайл или другое. Код проверки представлен в файле tg_flow.json, строка 86.

После проверки идет нода Switch: если на предыдущем этапе было определено, что сообщение имеет ссылку, то идем по ветке URL, если пользователь прислал файл - VIDEO, если присланное пользователем сообщение не является ссылкой на видео или видеофайлом, пользователь получает сообщение, рисунок 2.

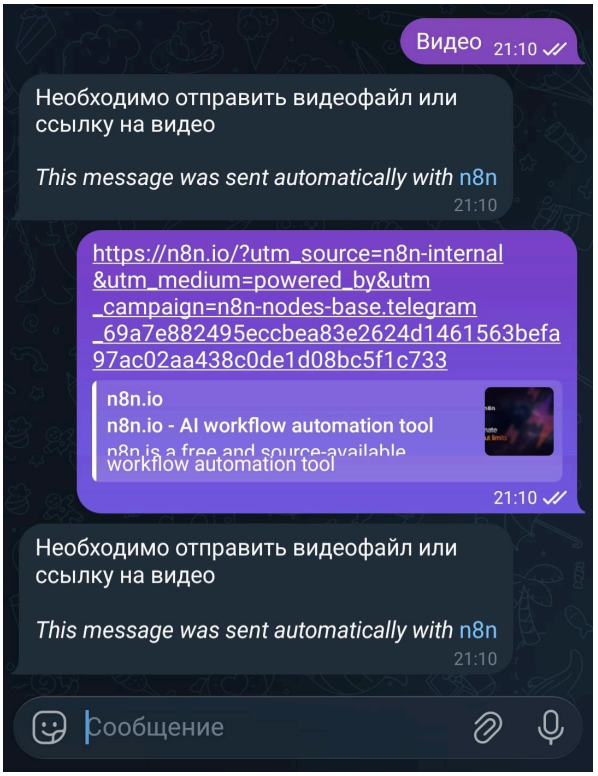

Рисунок 2 - Сообщение о необходимости отправить видеофайл или ссылку на него

Если был получен файл, видеофайл сохраняется для дальнейшего использования (ищем файл по File ID -> получаем его -> получаем сам файл с помощью ноды HTTP Request, метод GET запрос в файле tg_flow.json, строка 28 -> записываем файл на диск для дальнейшей работы).

Если была получена ссылка, то работает нода Execute Command, которая вызывает скачивание видео с ютуба.

В результате записи может произойти ошибка, тогда идем по ветке error к ноде Execute Command, которая запускает удаление файлов, затем с помощью ноды Telegram Send Message пользователю направляется сообщение, что произошла ошибка. Результат работы ветки представлен на Рисунке 3

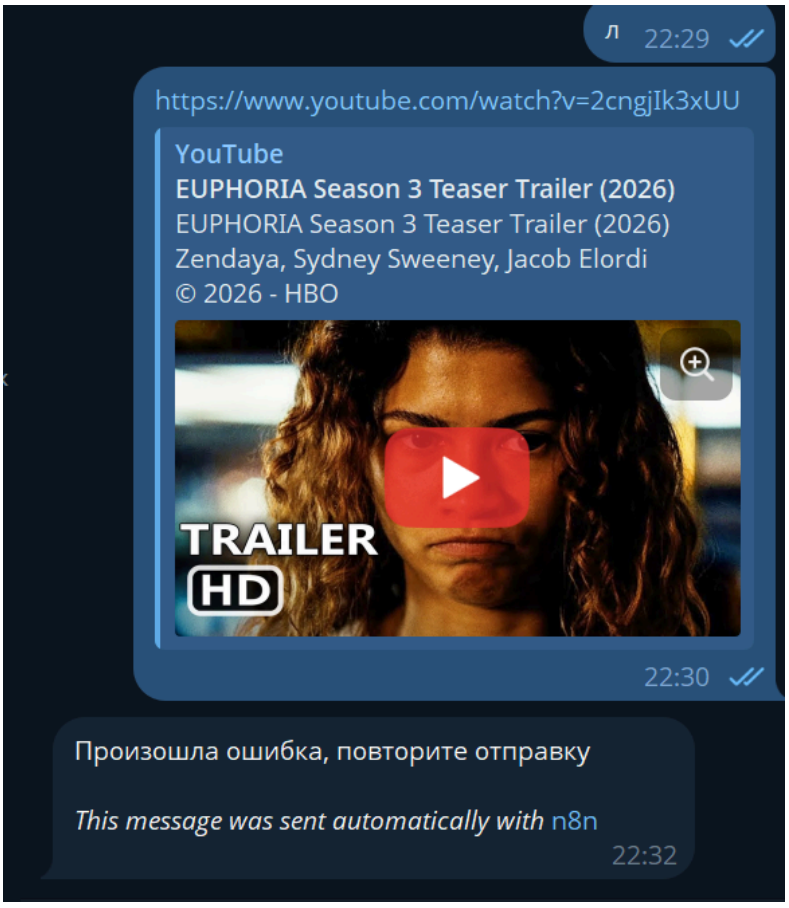

Рисунок 3 - Сообщение об ошибке

Если ошибки не произошло, переходим к ноде Execute Command, которая вызывает формирование субтитров в виде (выгрузка голоса из видео, формирование субтитров на английском, перевод субтитров, добавление субтитров в видео, формирование файла логов).

Если в процессе отработки ноды происходят ошибки, то повторяем попытку еще 2 раза с паузой в 5000 мс, настройки ноды представлены на рисунке 4. Если нода все еще

завершается с ошибкой, то переходим к удалению временных файлов и отправляем пользователю сообщение. Пользователь получает такое же сообщение, как на рисунке 3.

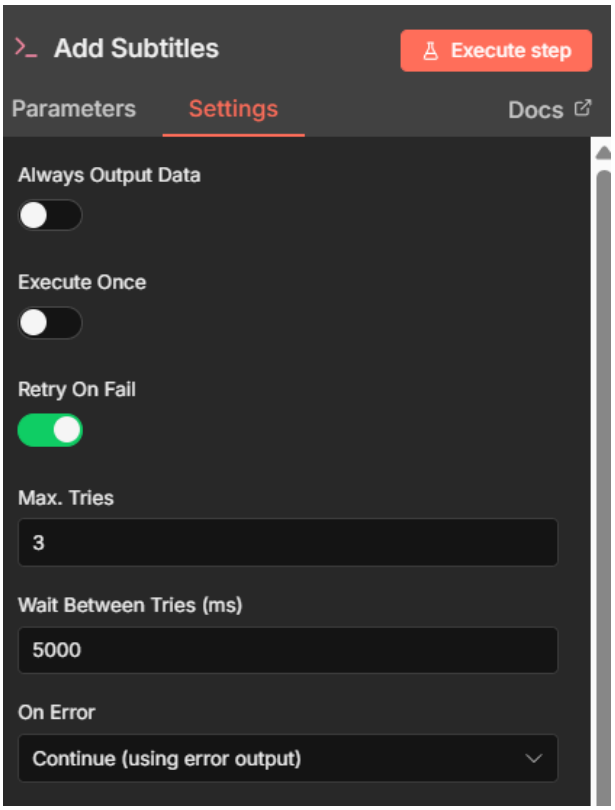

Рисунок 4 - Настройки ноды добавления субтитров в видео

Если добавление субтитров завершилось успешно, то забираем готовый файл с диска, отправляем его пользователю, удаляем промежуточные файлы.

На рисунке 5 представлены логи из n8n по всему пути. Исходное видео было получено из ссылки. Промежуточные файлы представлены на диске. В папке

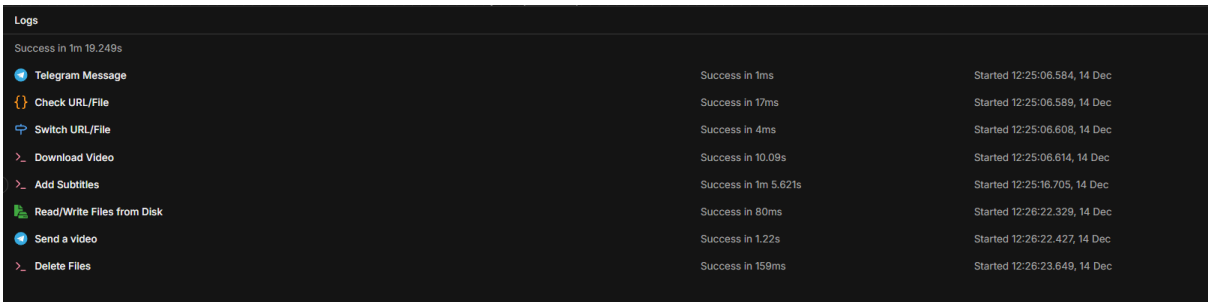

Рисунок 5 - Логи из n8n, весь путь

Входное видео имеет длительность 02:08, весит 7,7 мб. Можно отметить, что весь процесс добавления субтитров 2 минуты для этого видео.
Скачанное видео: video.mp4;
Выгруженный голос: voice.wav;

Субтитры: subtitles.str;
Перевод субтитров: subtitles_ru.str;
Готовое видео с субтитрами: video_subtitled.mp4.

На рисунке 6 представлены скриншоты из бота с отправленными и полученными видео.

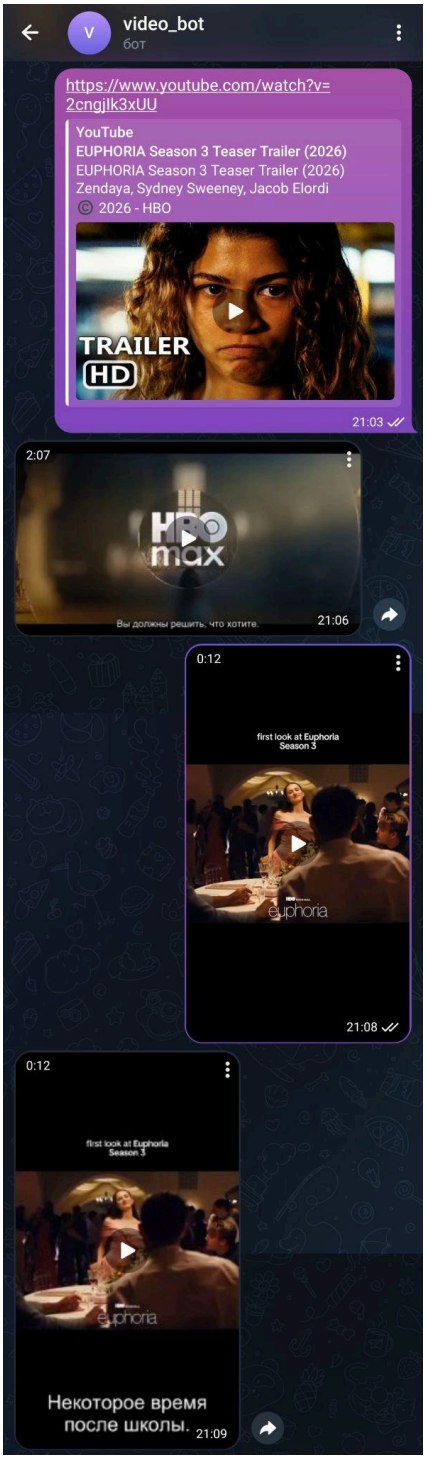

Рисунок 6 - Скриншот из бота

Более подробное описание использованных технологий

Основным инструментом в данной лабораторной работе являлся **Docker**.
Файл docker-compose.yml содержит информацию о запускаемых контейнерах их связь и порты/папки. Файл Dockerfile.video-processing “инструкция” по сборке контейнера для обработки видео (содержит скрипты, библиотеки). Файл Dockerfile.n8n “инструкция” по сборке контейнера с n8n (также содержит необходимые библиотеки).

N8n используется для оркестрации (по заданию используются триггеры, ветвления, бинарные данные, обработка ошибок). Экспорт Workflow представлен в файле tg_flow.json.

В файле get_video.py описан скрипт для скачивания видео с Ютуб с использованием **yt-dlp**.

В файле get_voice.py описывается скрипт для извлечение голоса из видео с использованием **ffmpeg**.

В файле subtitles.py описан скрипт для формирования субтитров (на английском языке) с использованием **whisper**.

В файле translate_subtitles.py описан скрипт для перевода субтитров на русский язык с помощью **ollama.**

В файле sub_into_video.py описан скрипт для добвыления переведенных субтитров в исходное видео с помощью **ffmpeg**.

В файле delete_files.py происходит удаление всех промежуточных файлов в конце всего процесса (после отправления видео пользователю).

Файл video_processor.py является управляющим. В данном файле последовательно запускаются скрипты get_voice.py -> subtitles.py -> translate_subtitles.p -> sub_into_video.py. Также в данном файле производится сохранение логов. В файле log_example/pipeline_error.log представлен пример лога, когда произошла ошибка. В файле log_example/pipeline_ok.log представлен пример лога, когда все прошло предсказуемо.

Логика обработки ошибок в video_processor.py:

1. Если шаг падает - он повторяется несколько раз
2. Перед повтором производится удаление файлов, которые могут создаться на этом шаге (например, если генерация субтитров завершается с ошибкой, то файл subtitles.str удаляется, если был создан)
3. Если после нескольких попыток шаг не завершился удачно, выполнение пайплайна останавливается (затем n8n перехватывает сбой и пользователь

получает сообщение с просьбой повторно отправить файл/ссылку, как уже было описано ранее)

В качестве дополнительной технологии был использован **ngrok** для проброса локального сервиса n8n во “внешний интернет”. Иначе n8n не работал корректно, так как локально входящие webhook-запросы не принимаются.

Выводы

При выполнении работы я пользовалась подсказками ChatGPT и ТГ чат Гаяне Маргарян. Также в файле docker-compose.yml строки 56-68 были взяты полность у нее (docker-compose.yml стр 22-34), так как у меня не получалось самостоятельно “подключить” модель.

Самым сложным для меня оказалось верно поднять контейнеры в Docker, в какой то момент спустя 15 часов работы над лабораторной, я поняла, что в самом начале неправильно все сконфигурировала и мне пришлось начинать все практически заново.

Также была проблема с использованием **ngrok**, если быть точным с его неиспользованием. Долго не могла понять, почему не получаю реакцию на сообщение в локальном n8n.

P.S. А вообще, эта лабораторная отобрала у меня всю радость, поэтому вот мой искренний вывод: как же замечательно, что я не работаю программистом.