

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

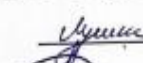
Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**СТАРТАП-ПРОЕКТ. КОНВЕЙЕР ОБРАБОТКИ
ВИДЕОФАЙЛОВ ДЛЯ ВЕБ-ПРИЛОЖЕНИЯ**

Студент

 31.05.24
(подпись, дата)

Лукин В.Д.

Руководитель

 31.05.24
(подпись, дата)

Федюнин Р.Н.

Нормоконтролёр

 31.05.24
(подпись, дата)

Бычков А.С.

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой


(подпись)

Митрохин М.А.

Работа защищена с отметкой отлично (протокол заседания ГЭК от

10.06.2024 № 15)

Секретарь ГЭК


(подпись)

Гурин Е.И.

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«19» апреля 2024 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Лукин Всеволод Дмитриевич гр. 20ВВП1 факультета Вычислительной техники.

2. Тема работы: «Стартап-проект. Конвейер обработки видеофайлов для веб-приложения»

Тема утверждена приказом ПГУ № 405/о от "19" апреля 2024 г.

3. Руководитель бакалаврской работы к.т.н., доцент, каф. ВТ ПГУ Федюнин Р.Н.

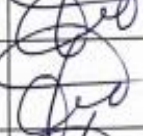
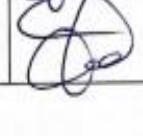
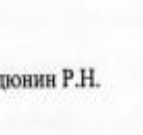
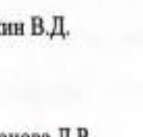
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать и реализовать конвейер обработки видеофайлов, использующий нейронные сети. Данное программное обеспечение должно в автоматическом режиме обрабатывать поступающие видеофайлы с параметрами улучшений и сохранять полученный результат. Повышение качества должно производиться при помощи изменяемого пула моделей нейронных сетей и набора библиотек для работы с мультимедиа за приемлемый коэффициент соотношения качества преобразования к временным затратам.

5. Перечень подлежащих разработке вопросов

Требуется провести анализ предметных областей: изучить существующие конвейеры или программное обеспечение, имеющее аналогичный принцип работы, выделить достоинства и недостатки данных систем. Выбрать язык программирования и другие составляющие разработки, подходящие для оптимальной реализации задачи. Определить архитектуру конвейера и описать внутреннюю организацию. Изучить функционал библиотек для работы с мультимедиа файлами и API для взаимодействия с пулом нейронных сетей.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	20	26.04.2024 – 27.04.2024	
2	Анализ требований	32	28.04.2024 – 30.04.2024	
3	Выбор пула технологий, применяемых при проектировании и разработке	30	01.05.2024 – 03.05.2024	
4	Разработка основной части	82	04.05.2024 – 14.05.2024	
5	Разработка интерфейса администрирования	16	15.05.2024 – 16.05.2024	
6	Разработка механизма взаимодействия с API нейронных сетей	40	16.05.2024 – 19.05.2024	
7	Отладка и тестирование	50	19.05.2024 – 23.05.2024	
8	Оформление пояснительной записки	54	24.05.2024 – 31.05.2024	

Дата выдачи задания "19" апреля 2024 г.

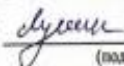
Руководитель бакалаврской работы



(подпись, дата)

Федонин Р.Н.

Задание к исполнению принял студент



(подпись, дата)

Лукин В.Д.

Работу к защите допустить

Декан факультета

 05.06.2024

(подпись, дата)

Фионова Л.Р.

Реферат

Бакалаврская работа содержит 150 страниц, 39 рисунков, 9 таблиц, 8 источников, 3 приложения, 1 формулу

КОНВЕЙЕР ОБРАБОТКИ ВИДЕОФАЙЛОВ ДЛЯ ВЕБ-ПРИЛОЖЕНИЯ, C++, C, BASH, FFMPEG, NVIDIA CUDA, VULKAN API, FAST C++, НЕЙРОННЫЕ СЕТИ, LINUX

Цель работы – разработать и реализовать программный комплекс, обрабатывающий видеофайлы в автоматическом режиме при помощи нейронных сетей по заранее созданному сценарию.

Объектом разработки является программный комплекс, позволяющий улучшать качество видеоконтента посредством использования заранее обученных моделей нейронных сетей. Комплекс должен обладать консольным интерфейсом администрирования, иметь возможность добавления новых моделей и их API в исходный код проекта без существенных затрат по времени, системой резервного сохранения и восстановления, возможностью установки настроек обработки видео.

Разработка проводилась в текстовом редакторе Visual Studio Code. В качестве языка программирования был выбран C++.

В результате выполненной работы разработан и реализован программный комплекс, удовлетворяющий требованиям по всем пунктам поставленной задачи, а именно: имеет информативный консольный интерфейс администрирования, возможность добавления новых моделей нейронных сетей и их API, встроенную систему резервного сохранения и восстановления, возможность установки предварительных настроек видео.

Использование, улучшение и поддержку созданного программного продукта планируется производить в рамках индивидуального стартап-проекта.

					ПГУ1.090301.15.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Стартап-проект. Конвейер обработки видеофайлов для веб- приложения. Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Лукин В. Д.		20.08.23					
Провер.		Федюкин Р. Н.		20.08.23			2	150	
Н. Контр.		Бочков А. С.		20.08.23		ФВТ, каф. ВТ гр. 20ВВП1			
Утверд.		Минералин М. А.		21.08.23					

Содержание

Перечень сокращений и обозначений	6
Введение	7
1 Конвейер обработки видеофайлов для веб-приложения	8
1.1 Актуальность	8
1.2 Пользователи и сценарии использования	10
1.3 Анализ аналогов	14
1.3.1 Topaz Video Enhance AI	15
1.3.2 Waifu2x-caffe	18
1.3.3 Bigjpg	21
1.3.4 Remini	24
1.4 Функциональное сравнение с аналогами	27
1.4.1 Постановка задачи	28
1.4.2 Требования к интерфейсу администрирования	28
1.4.3 Функциональные требования	29
1.4.4 Технические требования	31
1.4.5 Ограничения	32
1.5 Выбор инструментов разработки	33
1.5.1 Язык программирования	33
1.5.2 Среда разработки	34
1.5.3 Библиотека для работы с мультимедиа	36
1.6 Архитектура разрабатываемой системы	36
1.7 Разработка конвейера	41
1.7.1 Разработка и описание модели конвейера	42

1.7.2	Разработка классов.....	46
1.7.2.1	Класс Logger	46
1.7.2.2	Класс ThreadPool	48
1.7.2.3	Класс DaemonTask	49
1.7.2.4	Класс DaemonException	49
1.7.2.5	Класс DaemonBase	49
1.7.2.6	Класс BaseUpscaleSettings	51
1.7.2.7	Классы-наследники.....	52
1.8	Тестирование и отладка.....	52
2	Охрана труда.....	60
2.1	Описание негативных факторов, воздействующих на разработчика и способы их устранения.....	60
2.1.1	Оборудование рабочего места	61
2.1.2	Освещенность рабочего места	62
2.1.3	Электробезопасность	63
2.1.4	Пожарная безопасность	64
2.1.5	Режим труда и отдыха	65
3	Стартап проект	66
3.1	Сегменты ЦА	67
3.2	Ключевые процессы.....	72
3.3	Каналы взаимодействия	74
3.4	Источники дохода	75
3.5	Ключевые ресурсы	77
3.6	Риски проекта	79
	Заключение	80

Список использованных источников	81
Приложение А	82
Приложение А.1 – UML-диаграмма вариантов использования	83
Приложение А.2 – UML-диаграмма последовательностей	84
Приложение А.3 – UML-диаграмма классов.....	85
Приложение А.4 – UML-диаграмма развёртывания	86
Приложение А.5 – UML-диаграмма деятельности.....	87
Приложение Б	88
Приложение В.....	144

Перечень сокращений и обозначений

В данной работе применяются следующие сокращения и обозначения

ГОСТ – единый государственный стандарт, который чётко устанавливает требования к качеству определённого изделия или определённой деятельности.

Демон – компьютерная программа в UNIX-подобных системах, запускаемая самой системой и работающая в фоновом режиме без прямого взаимодействия с пользователем

ОС – операционная система

ООП – объектно-ориентированное программирование

ПК – персональный компьютер

Релиз – готовое приложение и одноимённая страница с его исходным кодом в веб-сервисе GitHub

Фреймворк – программная платформа, которая упрощает разработку программного продукта, предварительно определяет структуру проекта

AI (Artificial Intelligence) – искусственная интеллектуальная система

API (Application Programming Interface) – программный интерфейс приложения

FPS (Framerate Per Second) – количество сменяемых кадров за единицу времени

GitHub – крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки

IDE (Integrated Development Environment) – комплекс программных средств для разработки программного обеспечения

JVM (Java Virtual Machine) – виртуальная машина Java, основная часть исполняющей системы JRE (Java Runtime Environment)

JSON – текстовый формат обмена данными, основанный на JavaScript

.NET – модульная платформа для разработки программного обеспечения с открытым исходным кодом.

UML (Unified Modeling Language) – язык графического описания для объектного моделирования, системного проектирования и отображения организационных структур

					ПГУ1.090301.15.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

К нейронным сетям в настоящий момент времени приковано достаточно большое внимание не только программистов и учёных, но и обычных пользователей. Данная математическая модель показала себя с наилучшей стороны при анализе больших объёмов данных в различных областях. Например, распознавание объектов на изображениях или воссоздание полноценных речевых моделей. Однако широкий спектр применения нейронных сетей создаёт проблему использования для некоторого сегмента охватываемой аудитории. В некоторых случаях, пользователь может не знать, как пользоваться определённой предобученной моделью, что способно отпугнуть его от дальнейшего использования данной технологии.

Зачастую, вина такого исхода взаимодействия полностью лежит на разработчике. Из-за чрезвычайно быстрого развития области нейросетевых моделей реализация простого и доступного интерфейса, автоматизации работы, предварительного сохранения обработанных данных, предварительного просмотра результата обработки может быть второстепенной задачей или вовсе отсутствовать в планах разработчика. Необходимо решать данную проблему, так как упрощённая интеграция нейронных сетей позволит ускорить развитие областей, в которых используется рассматриваемая технология.

Первым шагом на пути упрощения интеграции нейросетей в определённые области является создание и популяризация инструментов упрощения для взаимодействия с нейронными сетями. Описанную проблему решает данная выпускная работа, направленная на популяризацию и упрощение взаимодействия с нейросетевыми технологиями.

					ПГУ1.090301.15.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

1 Конвейер обработки видеофайлов для веб-приложения

1.1 Актуальность

Рассматриваемая тема обладает довольно высоким уровнем актуальности из-за растущей популярности нейронных сетей и потребности в высококачественном видео в настоящий момент времени. В области анализа и повышения качества графической информации нейросети занимают доминирующее положение, что подтверждается количеством сервисов для предоставления данных услуг.

Конвейер обработки видеофайлов использует модели нейронных сетей для улучшения параметров исходного видеоматериала. Использование конвейера связано с низким уровнем качества, разрешения или другого параметра оригинала видеофайла, и направлено на улучшение характеристик и повышение качества. Сочетание нескольких инструментов для обработки графической информации и автоматизация данного процесса позиционирует рассматриваемый конвейер как универсальный автоматизированный инструмент для обработки видеоконтента.

Использование веб-приложения в качестве основной платформы для взаимодействия с конвейером оправдано следующими факторами:

- отсутствие привязки к определённому графическому интерфейсу. Веб-приложение, создаёт интерфейс взаимодействия под собственные требования и может не иметь строгих зависимостей от конвейера. Это позволяет довольно быстро менять или обновлять интерфейс без изменений в коде конвейера;
- удалённое использование и администрирование системы. Веб-приложение может обеспечить удалённый доступ к конвейеру без использования сторонних средств. Администрирование и использование при таком сценарии

может производиться без непосредственного контакта с сервером, на котором развёрнут конвейер;

– масштабируемость и балансировка нагрузки. Использование веб-приложения позволяет упростить процесс масштабирования системы и балансировку нагрузки между несколькими машинами. Преимущество связано с отсутствием ограничения на количество подключаемых конвейеров к веб-приложению (нет жёсткой привязки один конвейер – один интерфейс) и доступностью готовых реализаций балансировщиков нагрузки.

Попытки повышения качества видеоматериала при помощи стороннего программного обеспечения могут стать крайне трудоёмкими и неэффективными из-за отсутствия необходимых параметров и точек сохранения, высокого потребления долговременной и оперативной памяти, плохой оптимизации внутренних процессов и множества других причин. В свою очередь, разрабатываемый конвейер призван упростить процесс обработки, значительно уменьшить временные затраты, а также повысить качество получаемого результата.

Подобный программный продукт может быть использован во множестве различных областей:

– распознавание лиц. Позволит быстрее выявлять правонарушителей или производить поиск пропавших людей на видеозаписях. Имеет высокий потенциал для развития;

– улучшение качества старых кинофильмов и записей. Решит проблему сохранения и анализа архивных записей минувших лет. Поможет заинтересовать молодое поколение, отторгающее старые фильмы из-за низкого качества записи;

– повышение параметров видео на видеоплатформах. Способствует популяризации отечественных видеоплатформ и повышению их конкурентоспособности за счёт появления уникальных функций, отсутствующих у зарубежных аналогов;

– увеличение чёткости медицинских записей. Врачи смогут быстрее установить диагноз пациента по видеоматериалам обследований.

Кроме использования в областях, перечисленных выше, конвейер может послужить важным аспектом в области развития нейросетевых технологий. Это обусловлено тем, что данное программное обеспечение может быть использовано в качестве базовой платформы для разворачивания и тестирования различных моделей нейронных сетей. Модели могут иметь разный уровень обработки информации и затрат аппаратных ресурсов, что может быть обнаружено при помощи встроенных средств администрирования конвейера [1].

Разработка данного программного продукта также имеет большой практический интерес из-за пула затрагиваемых областей разработки и администрирования: нейронные сети, многопоточное программирование, разработка высоконагруженных систем, администрирование операционных систем на базе ядра Linux, работа с графическими ускорителями, настройка взаимодействия и использование служб (демонов). Данный стек технологий призван привлечь внимание потенциальных работодателей и увеличить шансы успешного трудоустройства на рынке труда.

Конвейер обработки видеофайлов для веб-приложения является не только актуальной темой в настоящее время. Он может быть востребованным в будущем с точки зрения улучшения работы и повышения эффективности бизнес – процессов в коммерческих проектах.

1.2 Пользователи и сценарии использования

Перед началом проектирования и разработки необходимо определить целевую аудиторию пользователей конвейера и проработать сценарии использования данного программного обеспечения. Стоит учитывать, что взаимодействие может проводиться как через внешнее веб-приложение, так и через встроенную консольную оболочку. Можно предположить, что основными пользователями продукта могут быть следующие группы:

					ПГУ1.090301.15.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

– исполнители проектов, использующие данный продукт для разворачивания и тестирования собственных моделей нейронных сетей, улучшающих характеристики графического контента. Использование преследует цель оценить показатель соотношения времени к качеству обработки;

– обычные пользователи, желающие развернуть данное программное обеспечение на локальной машине для последующего личного использования.

Определение целевой аудитории пользователей поможет точнее сформулировать сценарии использования разрабатываемого конвейера. Такой подход позволит избежать лишних трудовых затрат при разработке.

Сценарий использования для исполнителей проектов:

– загрузка репозитория проекта для последующей сборки. Сделать это можно на платформе GitHub, позволяющей публиковать собственные проекты для публичного использования;

– добавление локальных изменений в проект. Разработчик может изменить внутреннюю организацию кода под собственные требования или оставить изначальную организацию. Например, он может добавить дополнительный программный интерфейс, поддерживающий работу с определённой моделью нейронной сети;

– сборка проекта. Изменяется при добавлении новых целей для сборки по требованиям разработчика;

– установка конвейера. Производится при помощи скрипта установки, расположенного в основной директории проекта;

– предварительная настройка. Необходимо задать параметры работы программного комплекса: время работы компонентов, расположение исходных ресурсов и результатов обработки, сценарий поведения при ошибках. Можно использовать параметры, сгенерированные автоматически;

– запуск конвейера. Если параметры по умолчанию были изменены разработчиком, то каждый изменённый элемент конвейера необходимо

запустить вручную. После успешного запуска, последовательность будет запускаться автоматически;

- размещение графического контента в начальной точке обработки. Расположение данной точки указано в файлах настройки программного комплекса и может быть изменено при редактировании параметров программного комплекса;

- анализ скорости и качества преобразования. Замеры данных параметров можно производить при помощи встроенного анализатора. Разработчик, обладающий большим опытом, может использовать сторонний анализатор, если встроенный не обладает достаточной информативностью;

- использование веб-приложения для удалённого использования. Данный шаг обязателен только при реализации определённого проекта, инкапсулирующего конвейер. Добавление веб-приложения к данному программному обеспечению приведёт к формированию полноценного веб-сервиса.

Кроме исполнителей проектов, продуктом могут пользоваться и обычные пользователи. Учитывая возможность небольшого осведомления о механизмах работы и настройки программного комплекса рядового пользователя необходимо создать заранее настроенную версию для быстрой установки. Решением описанной выше проблемы станет создание релизов – версий проекта с предустановленными настройками и моделями нейростей. Это позволит упростить установку для обычного пользователя.

Сценарий использования для рядовых пользователей, желающих улучшить видеофайл:

- загрузка одного из релизов проекта. Загрузка релиза упрощает установку для пользователя из-за наличия предустановленных настроек и моделей нейронных сетей;

- установка конвейера. В основной директории проекта хранится файл с подробным описанием установки для разработчиков и коротким описанием для

1.3 Анализ аналогов

В рамках данной дипломной работы требуется провести сравнительный анализ различных аналогов описанного выше конвейера на основе списка доступных инструментов, удобства использования, сравнения показателя соотношения качества преобразования к временным затратам, стоимости и доступности для использования. Описанный шаг следует выполнить для последующего корректного формирования функциональных требований проекта. Подробное сравнение аналогов по критериям приведено в таблице 1.

Таблица 1 – Сравнение аналогов по критериям

Сервис, использующий конвейер	Функционал	Удобство использования	Соотношение качество / время	Стоимость использования
Topaz Video Enhance AI	Повышение качества графического контента	Нагруженный интерфейс, имеется большое количество настроек	Очень высокое качество / Большие затраты времени	От 30000р /год
Waifu2x-caffe	Повышение качества видео	Нагруженный интерфейс, небольшое количество функций	Высокое качество / Большие затраты времени	Бесплатно или от 700р /месяц
Bigjpg	Повышение качества изображений	Интерфейс понятен и удобен	Высокое качество / Низкие затраты времени	Бесплатно или от 300р /месяц
Remini	Повышение качества графического контента	Нагруженный интерфейс, большое количество функций	Среднее качество / Большие затраты времени	Бесплатно или от 600р /месяц

1.3.1 Topaz Video Enhance AI

Topaz Video Enhance AI – программное обеспечение, обладающее достаточно широким спектром инструментов для повышения качества графического контента при помощи встроенных нейронных сетей. Имеет большой диапазон настроек и поддерживает множество форматов. Обладает информативным интерфейсом пользователя, изображённым на рисунке 2.

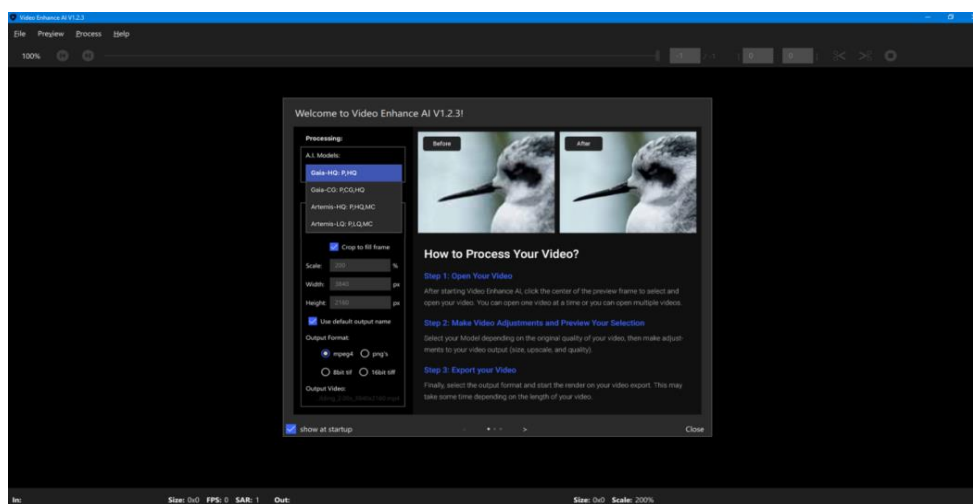


Рисунок 2 – Интерфейс Topaz Video Enhance AI

Однако данный сервис имеет некоторое количество недостатков, понижающих значимость описанных выше плюсов. Произведём анализ продукта по следующим параметрам:

– возможности сервиса. Topaz Video Enhance AI позволяет улучшить качество видеозаписей или изображений до достаточно высоких значений. Максимальные показатели параметров при улучшении:

- 1) разрешение видео: 4096x3072;
- 2) количество кадров: 60;
- 3) разрешение изображений: 7680x4320.

Поддерживается большая часть существующих форматов: mp4, mkv, avi, mov, m4b, png, jpg, bmp и т.д. Существует возможность обработки сразу нескольких файлов, что облегчает работу;

– удобство использования. Интерфейс Topaz Video Enhance AI достаточно нагружен из-за значительного количества настроек и может быть непонятным для рядового пользователя. Во время обработки данных, сервис создаёт отдельное окно для мониторинга времени обработки и затрачиваемых аппаратных ресурсов. Не менее приятной функцией является функция предварительного просмотра результата для видео, позволяющая оценить выбранный отрезок оригинала после применения улучшений;

– соотношение качества преобразования к временным затратам. Большое количество поддерживаемых функций, с одной стороны, является значительным плюсом при выборе данного программного обеспечения. С другой стороны, при использовании трудносовместимых параметров для одного обрабатываемого материала приводит к высокому потреблению аппаратных ресурсов, а иногда и вовсе невозможно. Однако, даже при выборе параметров, которые не конфликтуют друг с другом, временные затраты на обработку продолжают оставаться значительными;

Ниже приведена таблица 2 с временными затратами на определённые улучшения оригинального видеофайла.

Таблица 2 – Показатели при обработке видео в среде Topaz Video Enhance AI

Разрешение оригинала	Количество кадров оригинала	Длительность	Необходимое разрешение	Необходимое количество кадров	Время обработки и
800 x 600	30	2 мин 47 сек	1920 x 1080	30	2 ч 3 мин
800 x 600	30	2 мин 47 сек	1920 x 1080	60	3 ч 45 мин
800 x 600	30	2 мин 47 сек	3840 x 2160	30	3 ч 38 мин
800 x 600	30	2 мин 47 сек	3840 x 2160	60	4 ч 42 мин

– стоимость использования. Не предоставляется бесплатных пробных версий данного продукта. Использование возможно только после оформления годовой подписки за 30 000 рублей или полугодовой подписки за 17 000 рублей.

После подробного анализа данного продукта можно выделить следующие преимущества:

- самый большой диапазон параметров и функций среди конкурентов;
- поддержка большинства доступных форматов видео и изображений;
- один из самых удобных интерфейсов среди конкурентов;
- самое высокое поддерживаемое качество улучшения среди конкурентов;
- возможность предпросмотра фрагмента результата до начала конвертирования;
- возможность конвейерной обработки на машинах с большим количеством аппаратных ресурсов.

Но данное программное обеспечение имеет и существенные недостатки. В их числе:

- колоссальные временные издержки даже при работе с небольшими файлами и низкими настройками;
- ориентирован на использование рядовыми пользователями, но не поддерживается на машинах среднего и слабого диапазона мощности. В данных случаях, при использовании приводит к зависанию системы;
- не имеет встроенной системы предварительного сохранения результата. При прерывании процесса улучшения, промежуточный результат не будет сохранён. Ощутимая недоработка при больших временных затратах на конвертирование;
- не поддерживает использование сторонних нейронных сетей для повышения качества. Использует пул моделей, который заранее был определён разработчиками;

– расчёт стоимости использования является приблизительным, так как в данный момент времени не получится оформить подписку на территории Российской Федерации.

Подводя итог, Toraz Video Enhance AI является многофункциональным инструментом с большим пулом возможностей для улучшения графического контента. Данный продукт может быть хорошим выбором для разработчиков или пользователей машин с большим количеством аппаратных ресурсов. Однако рядовой пользователь, имеющий в своём распоряжении стандартную офисную машину, не сможет использовать данное программное обеспечение из-за высоких требований к ресурсам системы. Также существуют проблемы с получением лицензии, ограниченным количеством моделей нейронных сетей, сохранением промежуточного результата.

1.3.2 Waifu2x-caffe

Waifu2x-caffe – программное обеспечение для обработки видео контента с использованием соответствующей группы выборки нейронных сетей. Обладает информативным интерфейсом пользователя, изображённым на рисунке 3.

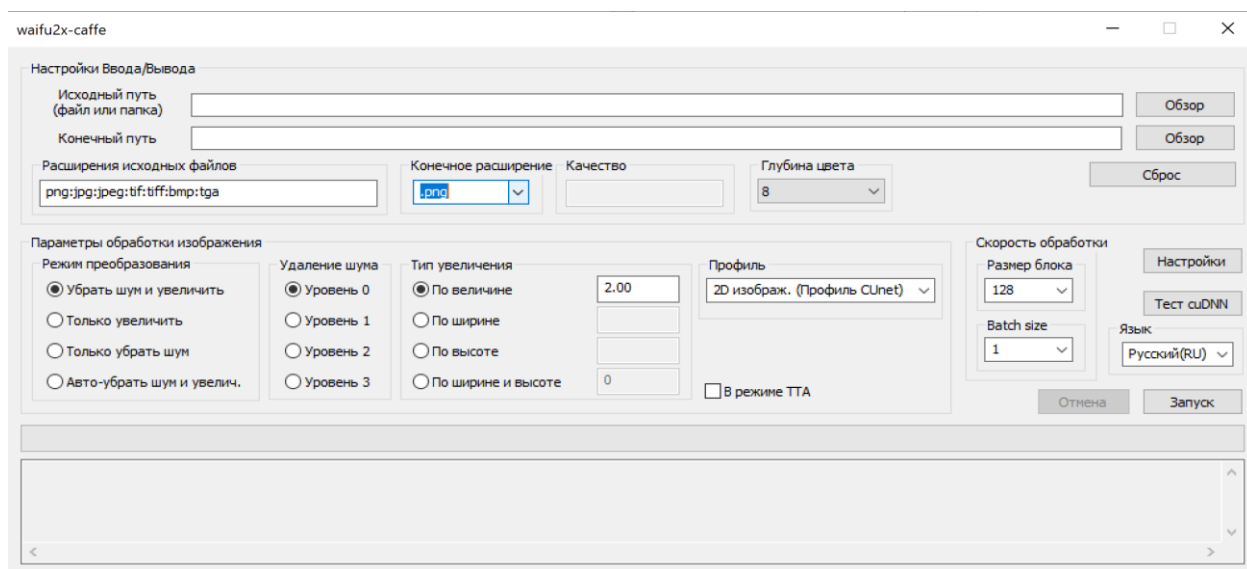


Рисунок 3 – Интерфейс Waifu2x-caffe

Произведём анализ продукта по следующим параметрам:

					ПГУ1.090301.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		18

– возможности сервиса. Предоставляет среднее количество настроек в сравнении с другими сервисами. Пользователю доступно повышение качества на пуле заранее определённых моделей, а также использование конвейера последовательной обработки. Программный продукт способен работать на слабых системах, но не имеет системы предварительного сохранения результата, что в некоторых ситуациях может привести к безвозвратной потере данных. Поддерживается только обработка видео и пять доступных форматов: mp4, mkv, avi, mov, m4b;

– удобство использования. Интерфейс Waifu2x-caffe немного запутан и может вызвать отторжение у новичков из-за большого количества различных вкладок с параметрами. Все настройки из вкладок можно было разместить в одном месте, чтобы пользователь мог выбрать всё необходимое без постоянного переключения по вкладкам. В интерфейсе отсутствует функция предпросмотра результата, однако есть отдельное окно для отслеживания прогресса улучшения видео. Не хватает справки об использовании интерфейса, что принуждает к просмотру дополнительных материалов о работе сервиса в интернете;

– соотношение качества преобразования к временным затратам. Отсутствие трудносовместимых параметров приводит к оптимизации скорости работы данного программного продукта, однако, временные затраты продолжают оставаться значительными при повышении качества. Ниже приведена таблица 3 с временными затратами на определённые улучшения оригинального видеофайла.

Таблица 3 – Показатели при обработке видео в среде Waifu2x-caffe

Разрешение оригинала	Количество кадров оригинала	Длительность	Необходимое разрешение	Необходимое количество кадров	Время обработки
800 x 600	30	2 мин 47 сек	1920 x 1080	30	3 ч 32 мин

800 x 600	30	2 мин 47 сек	1920 x 1080	60	5 ч 1 мин
800 x 600	30	2 мин 47 сек	2048 x 1080	30	4 ч 21 мин
800 x 600	30	2 мин 47 сек	2048 x 1080	60	6 ч 17 мин

– стоимость использования. В открытом доступе предоставлена бесплатная пробная версия данного программного обеспечения. Бесплатная версия имеет ограничения не позволяющие использовать данный продукт полноценно:

1) отсутствие многопоточного режима. Даже при наличии процессора с несколькими ядрами и потоками, продукт будет работать на одном потоке, что уменьшает предполагаемую скорость работы в разы;

2) отсутствие полноценной поддержки графических ускорителей. Бесплатная версия позволяет использовать лишь часть ресурсов, доступных системе, что также ограничивает предполагаемую скорость работы в разы.

Упомянутые выше ограничения можно убрать, оформив подписку за 700 рублей в месяц или 8400 рублей в год.

После подробного анализа данного продукта можно выделить следующие преимущества:

- существует бесплатная версия, но её функционал сильно ограничен;
- среднее соотношение качества преобразования к временным затратам среди конкурентов. Оптимальный выбор;
- хорошая оптимизация для слабых систем.

Но данное программное обеспечение имеет и некоторые недостатки. В их числе:

- достаточно запутанный интерфейс. Вкладки с настройками можно было бы разместить в одном разделе без потери понятности назначения;

– не поддерживается использование сторонних нейронных сетей для повышения качества. Используется пул моделей, который заранее был определён разработчиками;

– слишком сильно ограничен функционал бесплатной версии. Это не позволяет пользователям полноценно оценить функциональность и вызывает сомнения при трате средств на платную версию;

– не имеет встроенной системы предварительного сохранения результата. При прерывании процесса улучшения, промежуточный результат не будет сохранён. Ощутимая недоработка при больших временных затратах на конвертирование.

Исходя из сказанного выше, Waifu2x-caffe является хорошим бесплатным аналогом Topaz Video Enhance AI, который может быть использован для улучшения качества видеоматериалов на слабых системах. Данный продукт обладает средним размером пула параметров и функций повышения качества видео контента среди представленных конкурентов, но имеет достаточно запутанный интерфейс, что может оттолкнуть рядового пользователя.

1.3.3 Bigjpg

Bigjpg – сервис повышения параметров изображений, который использует искусственный интеллект для повышения разрешения файлов без видимой потери качества. Он предоставляет пользователям возможность улучшать детализацию и четкость изображений, что особенно полезно при работе с оригиналами в низком разрешении. Может использоваться как портативное приложение (рисунок 4) или веб-сервис в сети интернет (рисунок 5).

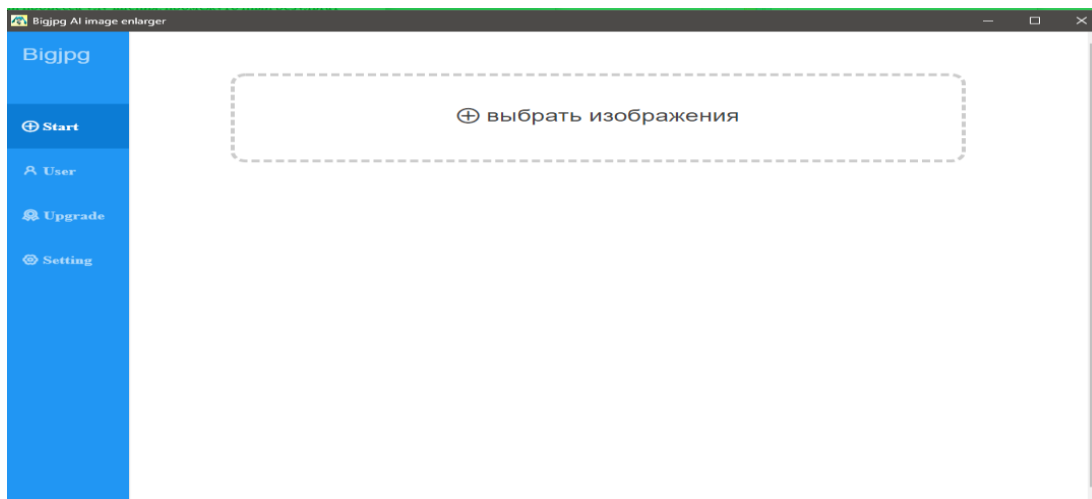


Рисунок 4 – Интерфейс Bigjpg

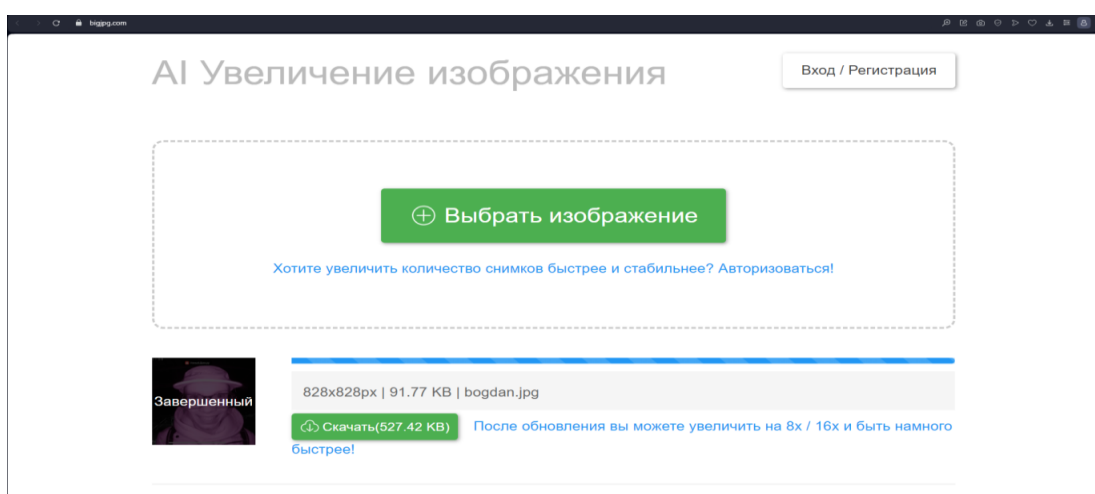


Рисунок 5 – Веб-интерфейс Bigjpg

Произведём анализ продукта по следующим параметрам:

– возможности сервиса. Bigjpg позволяет увеличить разрешение изображения в четыре раза от исходного размера с минимальными визуальными искажениями. Алгоритмы глубокого обучения анализируют и улучшают текстуру и границы объектов в изображении, увеличивают резкость и уменьшают эффекты пикселизации и размытости;

– удобство использования. Интерфейс Bigjpg интуитивно понятен. Пользователям нужно лишь загрузить изображение на сайт или через приложение, и выбрать желаемые параметры увеличения. Программа достаточно

резво обрабатывает фотографии, после чего предоставляет ссылку для скачивания улучшенного изображения из облачного хранилища;

– соотношение качества преобразования к временным затратам. Сервис выделяется хорошим сочетанием скорости обработки и качества получаемых изображений. Она может быстро увеличить изображения без существенных артефактов, что особенно важно для профессионалов, работающих с графическим контентом;

– стоимость использования. Bigjpg предлагает как бесплатные, так и платные опции. Бесплатный план имеет ограничения на количество и размер обрабатываемых изображений. Платная подписка расширяет эти лимиты и предоставляет более высокий приоритет обработки. Цены на платные подписки различаются, и пользователи могут выбрать наиболее подходящий для них план (рисунок 6);

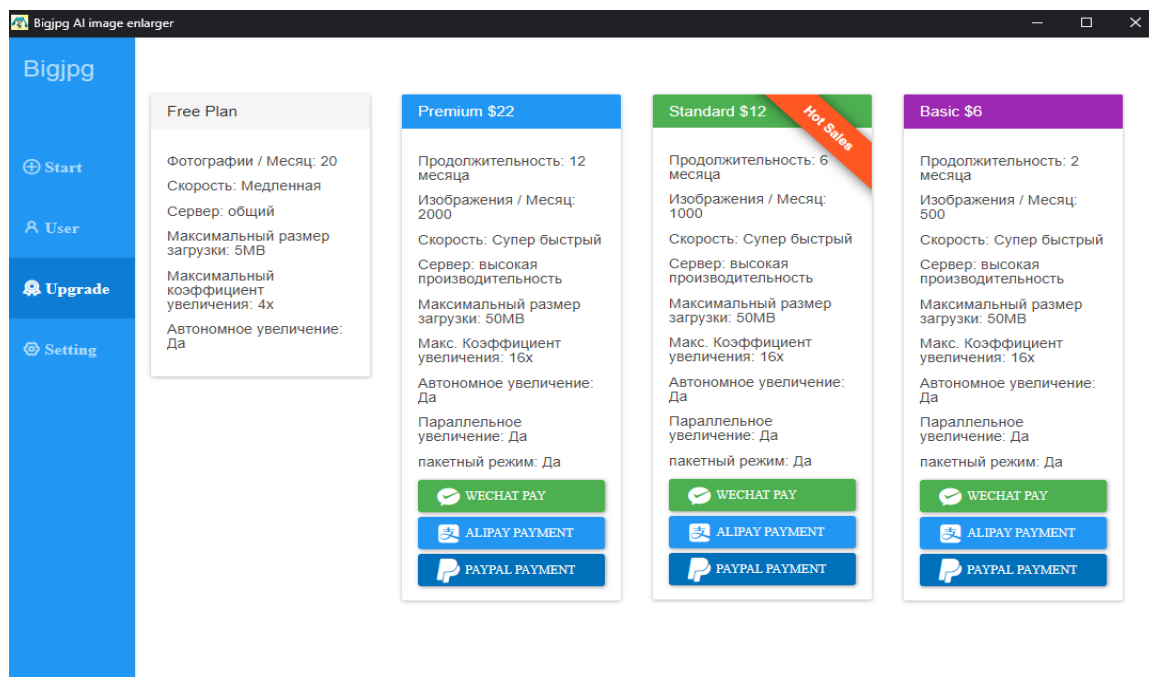


Рисунок 6 – Стоимость услуг Bigjpg

После подробного анализа данного продукта можно выделить некоторые преимущества. Bigjpg выделяется среди конкурентов благодаря сочетанию бесплатного доступа к увеличению изображений без потери качества и

доступных цен на платные планы. Его AI–алгоритмы способны генерировать изображения, которые выглядят естественно даже после значительного увеличения размера. Ещё одним конкурентным преимуществом является наличие десктопных и мобильных приложений, упрощающих процесс работы с изображениями.

Также стоит упомянуть недостатки данного программного обеспечения. Оно имеет ограниченное количество попыток бесплатных обработок, может не справляться с мелкими деталями и плохим качеством исходников, требует интернет-соединение и может вызвать опасения приватности из-за хранения данных на сервере. Платные планы необходимы для продвинутых функций, а время обработки может зависеть от нагрузки на сервер.

1.3.4 Remini

Remini – это приложение для обработки фотографий с использованием нейросетевых технологий, которое улучшает качество и повышает четкость изображений, восстанавливает старые или поврежденные снимки, устраняет размытость и повышает детализацию (рисунок 7).

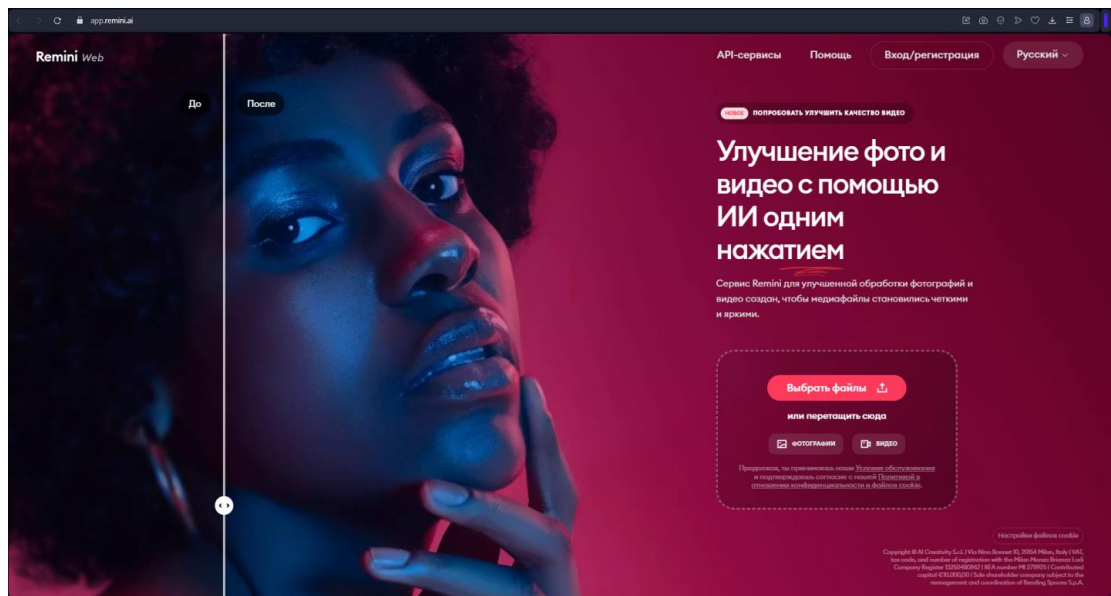


Рисунок 7 – Веб-интерфейс Remini

Произведём анализ продукта по следующим параметрам:

– возможности сервиса. Remini использует искусственный интеллект для улучшения качества изображений и видео. Приложение способно улучшать разрешение, восстанавливать старые и поврежденные фотографии, устранять размытость и увеличивать детализацию в изображениях;

– удобство использования. Сервис предлагает простой и интуитивно понятный интерфейс, что делает процесс улучшения изображений быстрым и легким даже для неопытных пользователей. Необходимо загрузить фотографию или видео, и Remini автоматически начнёт обработку, предложив предпросмотр улучшенного варианта (рисунок 8);

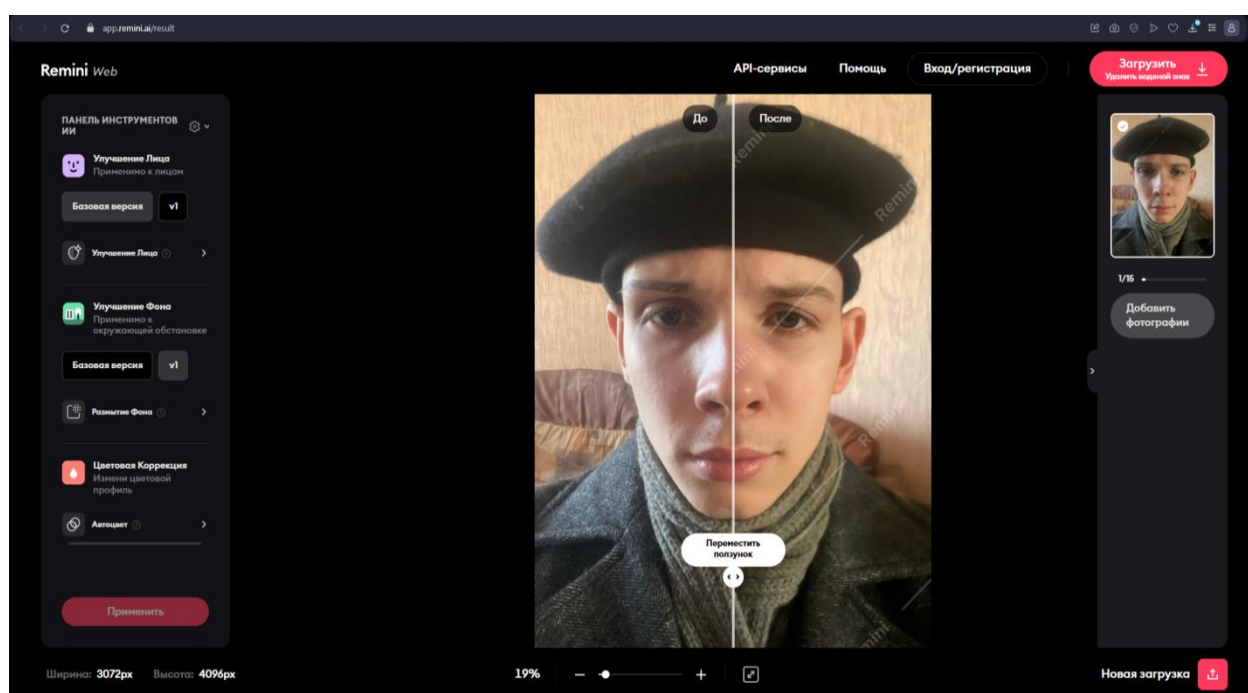


Рисунок 8 – Обработка изображений в Remini

– соотношение качества преобразования к временным затратам. Remini довольно быстро обрабатывает фотографии, хотя скорость может зависеть от размера файла и текущей нагрузки на сервер. Качество улучшенного изображения может варьироваться, но зачастую является средним. Лучше всего улучшения заметны на фотоснимках, сделанных на старые камеры или мобильные телефоны;

– стоимость использования. Приложение предлагает ограниченную бесплатную версию, где пользователи могут улучшить определенное количество изображений за день. Платная подписка обеспечивает безлимитный доступ к функциям и более доступную скорость обработки (рисунок 9);

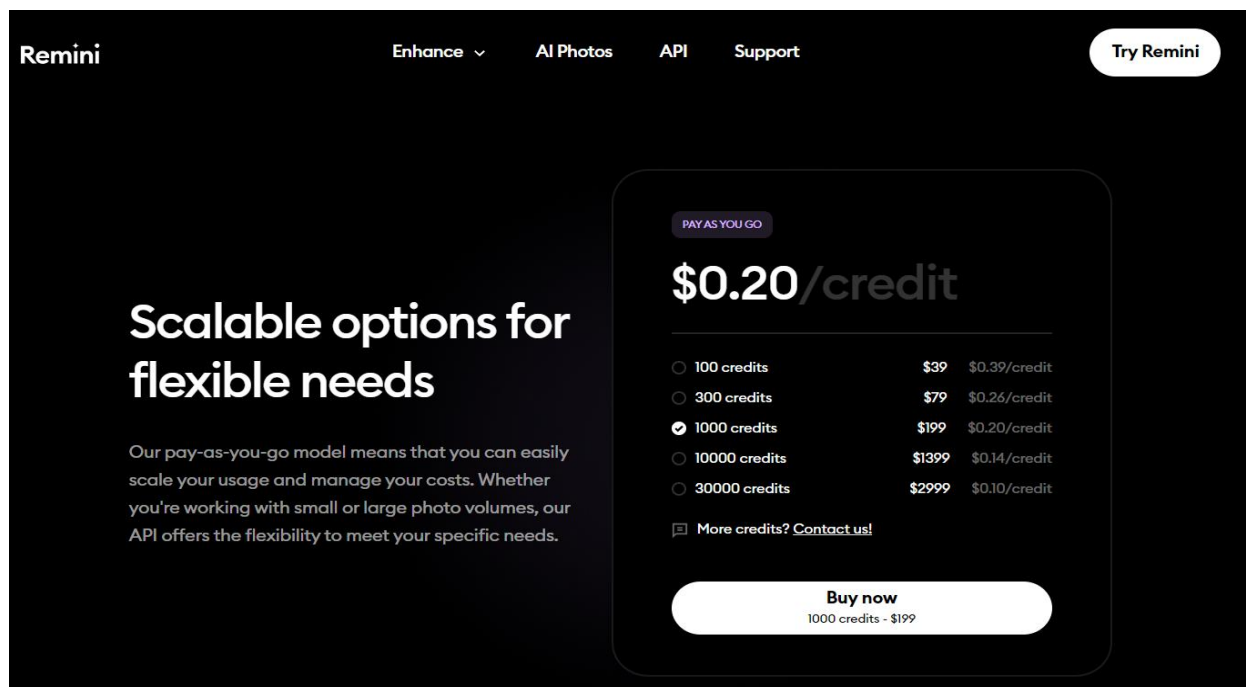


Рисунок 9 – Стоимость услуг Remini

Сервис имеет следующие преимущества:

- улучшение качества изображений низкого разрешения;
- интуитивно понятный дизайн и легкость использования;
- быстрый процесс обработки.

Однако преимущества могут быть нивелированы следующими недостатками:

- ограниченное количество бесплатных улучшений;
- некоторые результаты могут не соответствовать ожиданиям, особенно если исходное изображение сильно повреждено или имеет очень низкое качество;
- наличие платной подписки для полного доступа к функциям;

– возможные проблемы с конфиденциальностью и приватностью загруженных изображений.

1.4 Функциональное сравнение с аналогами

Разрабатываемый конвейер имеет некоторое количество сходств с аналогами. Улучшение реализации данных черт позволит выиграть конкуренцию:

- поддержка большинства доступных форматов видео;
- увеличение скорости обработки графического контента;
- наличие функции предварительного просмотра результатов;
- обработка данных на пуле заранее встроенных нейронных сетей;
- наличие бесплатной версии с доступным функционалом;
- поддержка работы на слабых системах.

Однако улучшения данных преимуществ может не хватить при борьбе за аудиторию. Для переманивания аудитории конкурентов на сторону данного проекта были придуманы следующие функциональные улучшения:

– предварительное сохранение результата. Даже при внеплановом завершении работы системы, конвейер сохранит данные и прогресс, что позволяет продолжить обработку без потери прогресса;

– поддержка индивидуального интерфейса нейронных сетей. Пользователь может использовать интегрировать собственные нейронные сети для обработки изображений и видео в данном конвейере. Это позволяет увеличить диапазон охватываемых областей применения;

– низкие затраты ресурсов системы для поддержки обвязки вокруг нейросети. Конвейер старается использовать минимальное количество системных ресурсов, чтобы модель могла использовать оставшуюся часть. Это позволит получить выигрыш по времени обработки.

– полностью бесплатная версия с полноценным функционалом. Любой пользователь может скачать релиз данного конвейера из интернета, либо

загрузить репозиторий Github для сборки вручную. Такой подход позволит максимально привлечь пользователя к использованию данного продукта. Разработчики программного обеспечения также будут привлечены открытым исходным кодом, который можно изменить под собственные требования.

1.4.1 Постановка задачи

Целью дипломной работы является разработка и реализация конвейера обработки видеофайлов, использующего нейронные сети для улучшения видеоконтента. Данное программное обеспечение должно в автоматическом режиме обрабатывать поступающие видеофайлы с параметрами улучшений и сохранять полученный результат. Повышение качества должно производиться при помощи изменяемого пула нейросетей и набора библиотек для работы с мультимедиа за приемлемый коэффициент соотношения качества преобразования к временным затратам.

На основе анализа конкурентов (пункт 1.3) были составлены определённые требования для разрабатываемого продукта, которые будут описаны далее.

1.4.2 Требования к интерфейсу администрирования

Конвейер должен быть обеспечен интерфейсом для взаимодействия и интерфейсом для администрирования. В качестве интерфейса для взаимодействия необходимо использовать веб-приложение, которое может быть спроектировано специально для данного проекта, либо использовать уже готовую реализацию. Использование веб-приложения в качестве интерфейса пользователя обусловлено следующими факторами:

- независимость графического интерфейса. Веб-приложение, привязанное к конвейеру, может создать интерфейс под собственные требования и может не иметь строгих зависимостей от обработчика. Это позволяет производить изменение интерфейса без затрагивания кода конвейера;

- удалённое использование. Веб-приложение способно обеспечить удалённый доступ к программному комплексу без использования сторонних

средств. При помощи удалённого доступа возможно производство администрирования без непосредственного контакта с сервером;

– масштабируемость. Использование жёстко привязанных к конвейеру интерфейсов взаимодействия в будущем приведёт к проблемам с масштабированием системы. Один интерфейс без дополнительных модификаций всегда будет привязан к единственному конвейеру. Использование веб-приложения решает данную проблему.

Интерфейс администрирования изначально реализуется в качестве консольной утилиты для отслеживания состояния работы системы. В данном случае консольный интерфейс является универсальным решением по нескольким причинам:

– отсутствие необходимости подключения и сопровождения дополнительных библиотек для графики. Позволяет уменьшить вес проекта и количество зависимостей внутри него;

– повышенная отказоустойчивость. Ошибки в графическом интерфейсе могут привести к остановке одного из компонентов программного комплекса [2];

– отсутствие совместимости с некоторыми операционными системами. Некоторые операционные системы, на которых планируется разворачивание конвейера, не поддерживают графический интерфейс пользователя (например, Ubuntu Server). При отклонении интерфейса в сторону графики, в любом случае, необходимо будет реализовывать консольный интерфейс для взаимодействия с ОС без графической оболочки.

1.4.3 Функциональные требования

Функциональные требования были составлены на основе анализа недостатков и преимуществ конкурентов (пункт 1.3). Выдвинуты следующие требования:

– поддержка обработки видео в популярных форматах. Под обработкой популярных форматов понимается взаимодействие с медиаконтейнерами типа:

mp4, mkv, avi, mov, m4b. Данные контейнеры являются компиляцией вариантов контейнеров, обрабатываемых конкурентами;

– возможность обработки видео различного разрешения и качества. Некоторые аналоги запрещают обработку видео разрешением ниже 800x600, что в некоторых ситуациях является критической недоработкой. Например, видеозапись может быть более низкого разрешения и качества из-за записи камерой с низким разрешением (видеонаблюдение, съёмка с дрона, экспериментирование с разрешением);

– обеспечение высококачественного результата улучшения. В редких случаях при тестировании приложений конкурентов можно было столкнуться с результатами некачественной обработки. Однако исправление данного дефекта могло заключаться только в повторной обработке. Повторная обработка является довольно дорогим процессом, когда речь идёт о нейронных сетях. Необходимо создать систему для отслеживания дефектов и использования ранее сгенерированных материалов для ликвидации возникших неполадок генерации;

– эффективное использование аппаратных ресурсов. Является одним из столпов функциональных требований. Данное требование включает в себя:

- 1) отказ от графического интерфейса;
- 2) использование операционных систем без графической оболочки;
- 3) грамотная организация архитектуры проекта;
- 4) использование специализированных средств для мониторинга;

– возможность масштабирования системы под возрастающую нагрузку. Необходимо создать архитектуру предполагающую возможность размещения компонентов конвейера на различных машинах. Данное решение позволит провести масштабирование системы без переписывания основной части кода при появлении дополнительных аппаратных ресурсов [3].

1.4.4 Технические требования

Для поддержки разрабатываемого программного продукта компьютер должен соответствовать установленным техническим требованиям. Данные требования были сформированы на основе функциональных требований проекта (пункт 1.4.3) и анализа программных продуктов конкурентов (пункт 1.3).

Минимальные технические требования для разворачивания:

- операционная система: Linux Ubuntu 18.04 или Linux Server 12.01;
- процессор: Intel Core i3-63xx;
- оперативная память: 6 Гб;
- графический ускоритель: Nvidia GeForce GT 705;
- свободное место: 500 Мбайт;
- устройства ввода: клавиатура;
- подключение к интернету.

Данные требования были получены и проверены при помощи развёртывания системы на виртуальных машинах. Подбирая отдельные аппаратные элементы машины был достигнут поддерживаемый технический минимум.

Рекомендуемые технические требования для разворачивания:

- операционная система: Linux Ubuntu 22.04 или Linux Server 14.01;
- процессор: Intel Core i5-93xx;
- оперативная память: 16 Гб;
- графический ускоритель: Nvidia GeForce GTX 1660 SUPER;
- свободное место: 500 Мбайт;
- устройства ввода: клавиатура;
- подключение к интернету.

Рекомендуемые требования были установлены по параметрам машины разработчика, на которой происходила основная часть разработки и тестирования. Однако использование машины с характеристиками,

превышающими рекомендованные, приведёт к повышению скорости работы и отклика системы [4].

1.4.5 Ограничения

Ограничения проекта необходимы из-за установки требований к работоспособности продукта. Некоторые элементы при превышении определённого лимита начинают работать некорректно или завершаются с критической ошибкой. Ориентирование на массовость использования обязывает предупреждать пользователя об ограничениях на определённые действия.

Выдвинуты следующие ограничения проекта:

– длительность видео или размер файла может ограничиваться в зависимости от системных ресурсов. Данный пункт является номинальным ограничением и служит напоминанием для разработчиков о том, что при создании собственной версии проекта необходимо установить ограничения на размерность и длительность обрабатываемого файла. Изначально, конвейер ограничен лишь свободным местом машины и может обрабатывать файлы любого размера. Но отсутствие ограничения может негативно сказаться на дальнейшей работоспособности;

– время обработки может варьироваться в зависимости от загрузки системы и сложности задачи. Ограничение является реальным из-за наличия в конвейере механизма предварительной установки доступных ресурсов. Каждому элементу программного комплекса можно задать доступное количество ресурсов. Достижение максимума данных ресурсов приводит к созданию очереди обрабатываемых задачи, что увеличивает потенциальное время обработки;

– поддержка и обновление моделей нейронных сетей. Ограничение является реальным. Из-за возможности встраивания различных моделей нейронных сетей в рассматриваемый конвейер, данные модели могут конфликтовать с продуктом при замене их на новую версию. Это может быть связано с изменением интерфейса модели или логики её поведения, а также с

множеством других факторов. Однако разработчики заранее помещают работоспособные модели в директорию установки и не могут нести ответственность за отказ приложения работать с другими нейронными сетями по старому интерфейсу.

1.5 Выбор инструментов разработки

Комплексный подход к разработке всегда подразумевает тщательный выбор инструментов для осуществления разработки. Необходимо выбрать оптимальные решения, либо ранее изученные, либо на изучение которых не уйдёт достаточно много времени при собственной высокой эффективности.

1.5.1 Язык программирования

Можно использовать достаточно большое количество языков программирования для разработки конвейера. Вот некоторые из них:

– Java. Один из самых распространённых языков программирования общего назначения, который может использоваться для backend-разработки в данном проекте. Имеет достаточно простой синтаксис, для запуска и разработки приложений требует только JVM и необходимые библиотеки в качестве зависимостей. Популярность данного языка способствовала появлению большого количества документации, библиотек и фреймворков из различных областей разработки.

Но данный язык имеет некоторые критичные недостатки для разработки высоконагруженного приложения, требующего минимального расхода ресурсов сторонними компонентами. Среди конкурентов (C, C#, C++) обладает самой долгой компиляцией и самым большим потреблением ресурсов системы. Из-за периодического обновления популярных фреймворков страдает от расхождения версий пакетов, поэтому поддерживать разрабатываемое приложение долго без устранения конфликтов зависимостей не получится;

– C++. Достаточно популярный язык программирования для разработки высоконагруженных backend-приложений. Имеет непростой синтаксис,

					ПГУ1.090301.15.001 ПЗ	Лист
						33
Изм.	Лист	№ докум.	Подпись	Дата		

компенсирующийся мощностью и функциональностью языка. Если сравнить его с соперниками, то он имеет самую высокую скорость компиляции приложений и большую мощность. Позволяет взаимодействовать с системой на достаточно низком уровне при помощи специализированных API и имеет большое количество документации. При большом количестве плюсов имеет неявный недостаток в необходимости использования нетривиальных конструкций для реализации функционала, присутствующего в других языках;

– C#. Выступает распространённым языком для разработки backend-приложений. Идеальный наследник C++ обладающий схожими преимуществами и недостатками. Улучшает идеи ООП, заложенные в C++, но уступает в скорости компиляции проектов. Обладает большим количеством документации и готовыми реализациями некоторых решений, отсутствующих в C++. Один из главных недостатков – сложность использования .NET Framework в среде Linux. Отсутствует возможность взаимодействия с системой на низком уровне, что выливается в понижение производительности приложения;

Каждый из перечисленных выше языков программирования имеет свои преимущества и недостатки, и выбор зависит не только от конкретных потребностей проекта, но и от компетенции разработчика в выбираемом языке. В данном случае, квалификация и требования сочетаются в языке программирования C++. Использование данного языка позволит написать высоконагруженную систему с возможностью оптимизации при помощи низкоуровневых API.

1.5.2 Среда разработки

Выбор среды разработки также важен, как и выбор языка программирования. Ранее, в качестве целевой операционной системы был выбран Linux, поэтому среда разработки будет выбрана из вариантов доступных для установки на данной ОС.

– Visual Studio Code. Относится больше к текстовым редакторам, нежели к полноценной среде разработки. Однако при помощи дополнений экземпляр

данного редактора может быть превращён в полноценную среду разработки. Данное программное обеспечение обладает приятным настраиваемым интерфейсом, большим набором дополнений для различных типов разработки и быстрой настройкой в качестве среды разработки (несколько шагов). Присутствует возможность использовать SSH, если в данный момент времени среду нельзя установить на Linux. Единственным значительным минусом данного продукта является необходимость периодического доступа в интернет для скачивания необходимых расширений;

– CodeBlocks. Полноценная среда разработки с встроенными инструментами для разработки на языках C и C++. Один из явных плюсов – отсутствие необходимости интернет соединения для установки основных пакетов и некоторых инструментов. Не обладает настолько же гибким и интуитивным интерфейсом как в Visual Studio Code, а также не поддерживает возможность разработки через SSH. При обновлении с небольшим шансом может создать конфликт пакетов в Linux, а также не поддерживает быстрый переход на редактирование других языков. Редактирование других языков может понадобиться при просмотре или настройке веб-приложения, выступающего в качестве интерфейса пользователя конвейера;

– QT Creator. Среда разработки с встроенными инструментами для создания графических приложений на языке C++. Обладает собственной графической библиотекой и редактором элементов, встроенными в среду. Как и CodeBlocks не поддерживает разработку через SSH и может создать конфликт пакетов в Linux. Встроенная графическая библиотека и редактор элементов не имеют практической ценности, так как не будут применяться в данном проекте. Поддерживает JavaScript, но не поддерживает сторонние фреймворки, что не позволяет полноценно редактировать определённые веб-приложения;

Повторно проанализировав требования, описанные ранее, выбор был сделан в пользу среды разработки Visual Studio Code.

1.5.3 Библиотека для работы с мультимедиа

Необходимо использовать данный компонент по ранее определённым требованиям проекта. При выборе стоит обратить внимание на удобство доступных API, скорость работы и количество документации.

– Ffmpeg. Одна из самых популярных библиотек для обработки мультимедиа файлов. Используется в большом числе популярных проектов из-за самой высокой скорости обработки данных в текущем сегменте. Имеет API для взаимодействия на множестве языков, в том числе и C, что позволяет довольно сильно оптимизировать базовый функционал под собственные нужды. Исходная библиотека постоянно обновляется и поддерживается, а широкий набор функций для взаимодействия с мультимедиа форматами закрывает все потребности текущего проекта. Из минусов можно отметить, что актуальность документации из-за постоянного обновления пула функций оставляет желать лучшего;

– OpenCV. Библиотека, инкапсулирующая алгоритмы компьютерного зрения, обработки изображений и численные алгоритмы общего назначения. Имеет обширный функционал, но он частично не покрывает задачи, затрагиваемые в проекте. Например, библиотека не может взаимодействовать с аудио потоками в видеофайле и это заставляет применять отдельные библиотеки для взаимодействия с аудио. Большое количество документации и практических примеров использования позиционируют библиотеку как часто используемую, однако по скорости и пулу функций данная библиотека уступает ffmpeg;

Выбор в данном случае не вызвал затруднений как было в предыдущих пунктах. Библиотека ffmpeg имеет наиболее универсальную модель организации и лучше всего подходит для выполнения задач, поставленных в проекте.

1.6 Архитектура разрабатываемой системы

Выбор правильной архитектуры системы является залогом стабильной работы и простоты поддержки в будущем. Каждая из рассматриваемых

организаций имеет свои преимущества и недостатки. В качестве возможных вариантов для проекта можно выбрать следующие подходы:

– монолитная архитектура системы. Традиционная модель программного обеспечения, которая представляет собой единый модуль, работающий автономно и независимо от других приложений [5]. Можно дать и другое определение – отдельная большая вычислительная сеть с единой кодовой базой, в которой объединены все задачи проекта. Данную организацию удобно использовать на начальных этапах проекта, однако подход содержит в себе и некоторое количество недостатков. Схема архитектуры изображена на рисунке 10.

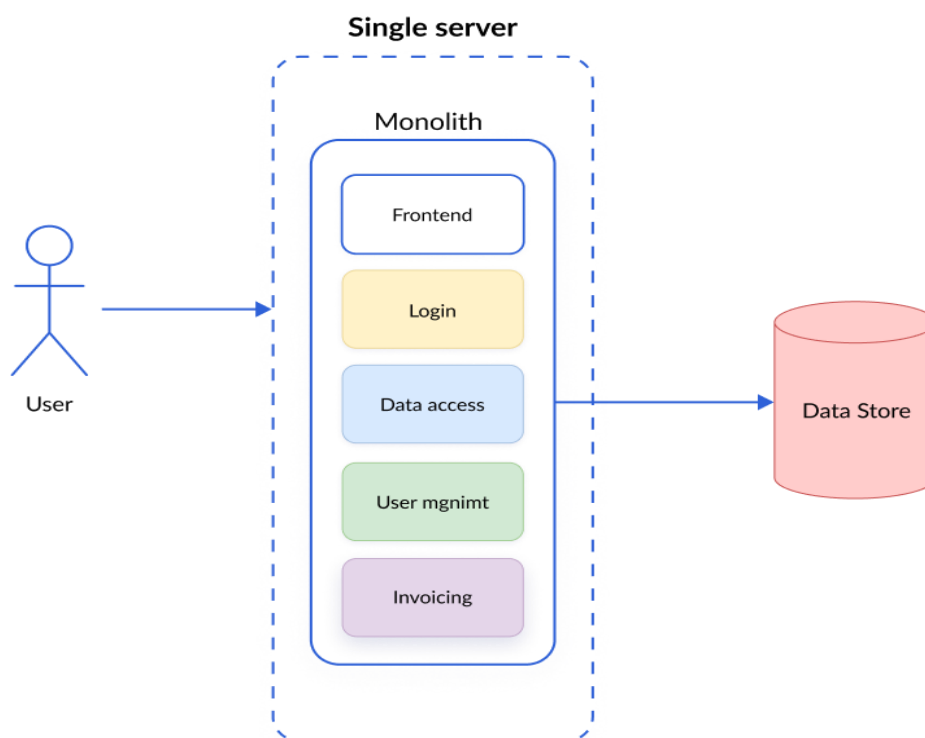


Рисунок 10 – Организация монолитной архитектуры

Преимущества монолитной архитектуры:

1) простота развёртывания. Из-за использования единого исполняемого файла развертывание системы происходит значительно проще, нежели в микросервисной архитектуре. Нет необходимости писать дополнительные скрипты, прописывающие требуемые зависимости;

2) простота контроля разработки. Если проект не имеет внушительной кодовой базы, то его курирование заметно упрощается. Однако, наращивание пула функций и количества кода может привести к затруднениям при изменениях в данном проекте;

3) улучшенная производительность. Из-за отсутствия внешних API для интерфейсов взаимодействия между службами не производится временных затрат на взаимодействие между разными модулями. Подразумевается наличие единой части кода, ответственной за весь пул функций проекта;

4) упрощенное тестирование и отладка. Так же, как и контроль разработки сильно зависит от количества кода и зависимостей в проекте. Написание тестов для небольшого монолита является простой задачей, но их дальнейшая поддержка с возможностью изменения внутренней организации может сильно усложнить данный процесс;

Не смотря на перечисленные выше плюсы монолитной организации, её использование в долгой перспективе может грозить следующими проблемами:

1) проблемы с масштабированием. Одна из главных проблем монолитной архитектуры. Появление дополнительных ресурсов в виде отдельных серверов, организация может предложить разворачивание системы только для индивидуальных машин. Данный факт не позволяет горизонтально расширять систему и приводит либо к переходу на микросервисную архитектуру, либо к дополнительной модификации на уровне выше для связывания нескольких систем в одну [6];

2) сомнительная надёжность. Если монолит получит ошибку в одной из своих частей, то это приведёт к остановке всего приложения. Поиск и ликвидация причины ошибки в данном случае также является довольно трудоёмким и запутанным процессом;

3) отсутствие гибкости. При попытке замены одной используемой технологии на другую возникнет проблема совмещения со

старыми технологиями. Отсутствие специализированных API между сегментами монолита приводит к сложностям при попытке модификации системы;

4) снижение скорости разработки. Поддержка старого кода при полной зависимости от него снижает общие темпы разработки из-за попыток не сломать старые зависимости;

При всех перечисленных преимуществах и недостатках, монолитная архитектура остаётся проверенной и надёжной организацией проекта, придающий быстрый старт при начале разработки.

– микросервисная архитектура системы. Представляет собой метод организации архитектуры, основанный на ряде независимо разрабатываемых служб (демонов). Данные службы обладают собственной логикой и конкретной целью работы. Обновление, тестирование, развёртывание и масштабирование выполняется для каждой службы отдельно. Подход независимого друг от друга функционирования является достаточно эффективным за счёт возможности разделённой разработки и часто используется в крупных проектах. Схема архитектуры изображена на рисунке 11.

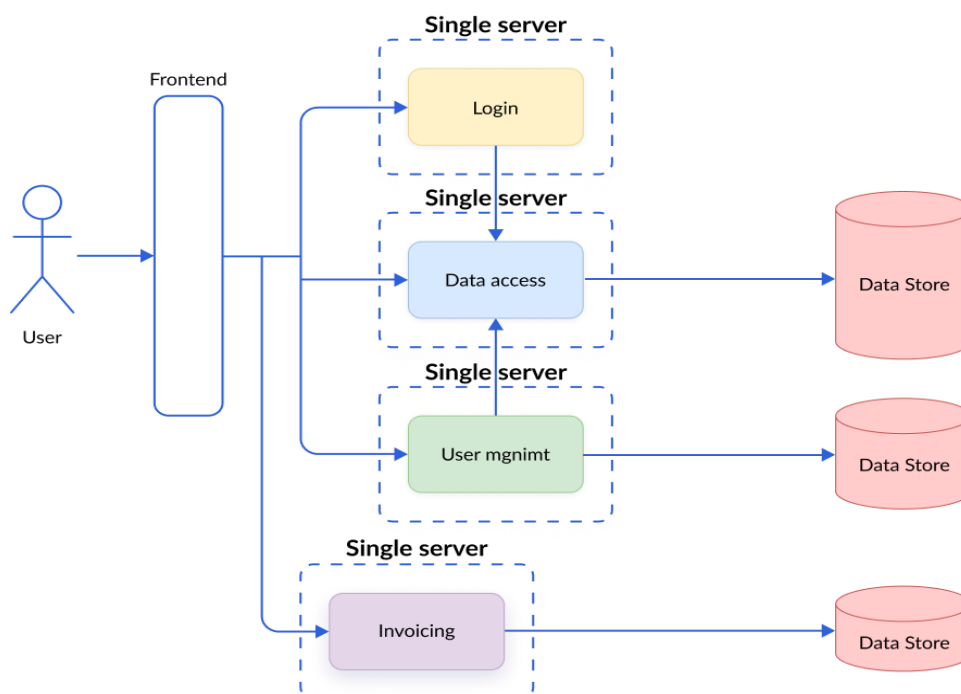


Рисунок 11 – Организация микросервисной архитектуры

Данная архитектура располагает следующими преимуществами:

1) непрерывное развёртывание. Подразумевает ускорение циклов релиза проекта. Если после утверждения изменений в монолите, разработчики модифицируют определённую часть с оглядкой на зависимости и из-за этого не могут опубликовать изменения долгое время, то в микросервисном подходе сразу после изменения определённой службы можно совершить публикацию. Это упрощает процесс разработки проекта;

2) возможность масштабирования. Если при рассмотрении монолитной организации одной из главных проблем являлась проблема интегрирования индивидуальных серверов, то в данном подходе горизонтальное расширение служит одним из главных преимуществ. Микросервисная архитектура создавалась специально для решения данной проблемы при проектировании разветвлённых систем. Поэтому необходимо в полной мере использовать данное преимущество;

3) высокий уровень надёжности. Сбой в одном из микросервисов не приводит к остановке работы всей системы. Если рассмотреть этот вопрос более подробно, то службы (демоны) перезапускаются в несколько раз быстрее монолитных систем и имеют специализированную систему контроля работоспособности. В Linux для данных целей используется подсистема инициализации и управления службами system;

4) лёгкость обслуживания и тестирования. Обеспечивается разделением команды программистов на обслуживание и тестирование разных микросервисов. Даже при общем большом объёме кода деление на модули позволяет разделить нагрузку между разработчиками и тестировщиками. Изменение тестов одной службы не приводит к изменению тестов другой;

Но описанные выше плюсы не избавляют данную архитектуру от противоположных закономерных минусов:

1) сложность старта процесса разработки. Запуск разработки проекта на данной архитектуре связан со сложностью начала разработки.

Необходимо, как минимум, два микросервиса для реализации подобной системы. Соответственно, дата первой итерации будет значительно позже аналогичной при использовании монолитной организации;

2) отсутствие стандартизации. Каждая из разрабатываемых служб не привязана к определённой схеме внутреннего устройства. Это может вносить путаницу в процессы поддержки и разработки, если каждый из микросервисов отличается друг от друга;

3) сложность поддержки при небольшой команде. Если команда разработчиков состоит из нескольких человек, то поддержка служб превращается в трудоёмкий процесс. Текущая архитектура направлена на использование в больших коллективах с разными задачами и стеками;

Микросервисная архитектура является антагонистом для монолитной архитектуры. Она способна закрыть недостатки монолита, но при этом лишается его преимуществ. Рассмотренная выше организация подойдёт для больших команд разработчиков с различными стеками разработки, но может использоваться и небольшими коллективами при точном понимании проекта.

Выбор архитектуры для использования осуществлялся на основе анализа функциональных требований (пункт 1.4.1) и постановки задачи (пункт 1.4). Для использования была выбрана микросервисная организация. Данное решение обосновано необходимостью поддержки расширения функционала без больших затрат, стабильности системы, горизонтального расширения в будущем, а также чёткого разделения задач между членами команды.

1.7 Разработка конвейера

После определения архитектуры проекта требуется построение конкретной модели конвейера. Использование веб-приложения в качестве интерфейса пользователя увеличит количество элементов используемых при составлении модели, однако позволит полностью описать внутреннюю организацию конечного продукта.

					ПГУ1.090301.15.001 ПЗ	Лист
						41
Изм.	Лист	№ докум.	Подпись	Дата		

1.7.1 Разработка и описание модели конвейера

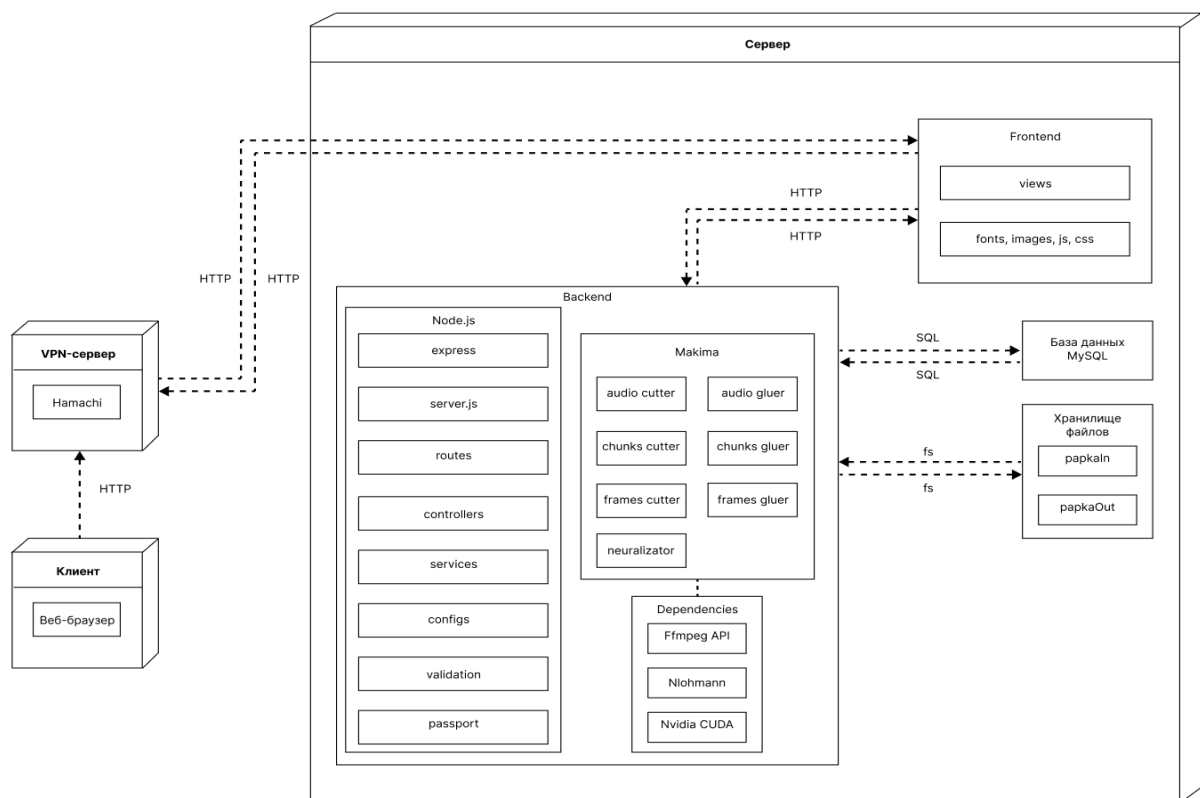


Рисунок 12 – UML-диаграмма развёртывания

Конвейер на UML-диаграмме развёртывания находится в разделе Backend и носит название Makima (рисунок 12). Внутри одноимённого блока изображены микросервисы, взаимодействующие друг с другом. Данные компоненты, исходя из описания микросервисной архитектуры, являются автономными службами (демонами), выполняющими конкретные функции. В конвейере для обработки видеофайлов используются следующие демоны:

- audio_cutter. Демон, отделяющий звуковую дорожку от получаемого видеофайла. Используется в данном проекте для сохранения оригинальной аудиодорожки, которая будет утеряна при дальнейшей обработке. В будущем может послужить плацдармом для внедрения нейронных сетей улучшающих качество звуковой дорожки;
- chunks_cutter. Компонент, отвечающий за деление файла оригинала на сегменты определённой длины (по умолчанию – 10 секунд). Будет крайне

востребован при обработке видео контента большой длительности. Разбиение одной большой задачи на небольшие подзадачи позволяет максимально использовать возможности индивидуального пула потоков демонов, который планируется использовать по умолчанию;

– frames_cutter. Служба для выделения последовательности кадров из входных видеоматериалов с последующим сохранением результата. В настоящий момент времени нейронные сети повышающие качество единичного изображения более распространены, нежели модели для аналогичной обработки видео. Данная реализация конвейера использует вариант с обработкой единственного изображения. Из-за этого требуется предварительное деление видео на последовательность кадров (фреймов), которая будет обработана в следующем модуле;

– neuralizator. Модуль-обертка, автоматизирующий работу с моделями нейронных сетей. Автоматизация позволяет организовать описанный ранее конвейер обработки отдельных изображений. Тип изображений не имеет значения – ими могут быть как обычные фотографии, так и фреймы, извлечённые из определённого видеофайла. Является одним из ключевых модулей в описанной архитектуре;

– frames_gluer. Объединяет папки содержащие фреймы в видеофайлы. Антагонист демона frames_cutter;

– chunks_gluer. Объединяет фрагменты преобразованного видео в итоговый видеофайл. Антагонист демона chunks_cutter;

– audio_gluer. Объединяет звуковую дорожку, извлечённую из оригинального файла, и итоговый видеофайл. Антагонист демона audio_cutter;

Перечисленные демоны имеют идентичную базовую схему работы основного цикла и, частично, функционирования потоков. Схемы представлены на рисунке 13 и рисунке 14, соответственно.

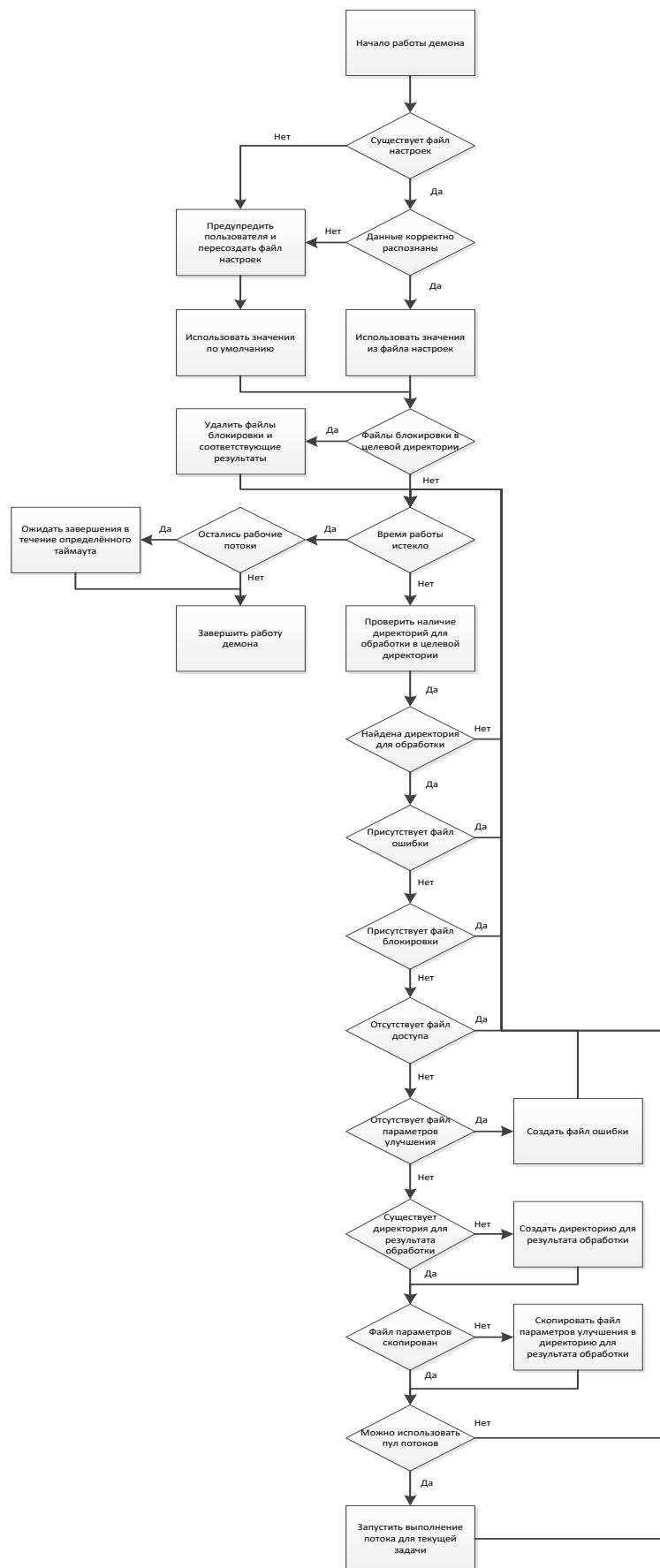


Рисунок 13 – Блок-схема работы основного цикла микросервисов



Рисунок 14 – Блок-схема работы потока микросервиса

Использование данных организаций обусловлено универсальной системой выявления ошибок, а также простым сценарием перезагрузки демона [7]. Универсальность агарантируется тем, что при наличии ошибки создаётся файл `makima.error`, который блокирует выполнение определённой задачи. Файл

ошибки вместе с контентом удаляется при перезагрузке демона. Также, при перезагрузке удаляются файлы блокировки с расширением .lock, сигнализирующие о текущем выполнении данной задачи.

1.7.2 Разработка классов



Рисунок 15 – UML-диаграмма классов

Показанная выше диаграмма классов (рисунок 15) была составлена до начала разработки основных демонов. Рассматриваемая организация имеет простую иерархию классов для упрощения будущего взаимодействия. Демон и вложенные в него объекты рассматриваются как отдельные составные элементы более крупных систем. Например, в основной класс DaemonBase входят объекты подклассов ThreadPool и Logger. И текущий случай не является единственным в рассматриваемой архитектуре.

1.7.2.1 Класс Logger

Отвечает за логирование событий внутри создаваемого демона. Используется в методах класса DaemonBase по прямому назначению. Имеет следующие уровни логирования:

– DEBUG. Логи отладки, интересующие только разработчика. Устанавливаются в местах сложной логики или местах, где необходимо зафиксировать результат без наблюдения со стороны пользователя;

– INFO. Логи информирования. Информировуют пользователя о свершившихся событиях. Используются как основные;

– WARNING. Логи предупреждения. Установлены в местах, где возможны не критичные сценарии продолжения работы. Предупреждают о некорректном поведении демона или параметрах, которые не мешают продолжать работу;

– ERROR. Логи ошибки. Возникла ошибка обработки, которая не нарушает общую работу, но не позволяет продолжить обработку конкретной задачи. В таких случаях в директории задачи также создаётся файл `makima.error`;

– CRITICAL. Логи критической ошибки. Возникают только при запуске демона. Их возникновение связано с недоступностью определённых компонентов, необходимых программе;

– UNKNOWN. Логи неизвестной ошибки. Возникают до момента инициализации пула потоков и имеют обратную связь с системными ошибками;

Данные уровни логирования позволяют в полной мере описать ситуации, возникающие в демоне. Запись лога может быть произведена при помощи вызова метода `Print`. При вызове данного метода необходимо указать отображаемое сообщение и уровень логирования. Все сообщения пишутся в системный журнал `journald`. Если демон был запущен вручную, логи дублируются в терминал запуска.

Кроме функции `Print`, логер обладает методами `SetLoggerName` и `SetLogLevel`. `SetLoggerName` принимает и устанавливает имя демона, которое необходимо отобразить во время записи сообщения. Вызывается в начале запуска метода `Launch`, класса `DaemonBase`. `SetLogLevel` устанавливает необходимый уровень логирования и отсекает сообщения, находящиеся в более низком уровне. Например, установка уровня логирования `INFO` и последующая

попытка вывода сообщения с уровнем DEBUG ни к чему не приведёт. Вызывается в методе Launch, класса DaemonBase.

Использование отдельного класса для описания данного элемента позволяет изменять внутреннюю логику без повреждения основных классов проекта. Это делает разработку более гибкой.

1.7.2.2 Класс ThreadPool

Описывает механизм упрощённого взаимодействия с потоками, а именно – пул потоков. Данный класс, как и Logger, является одним из ключевых классов для работы с DaemonBase. Реализация пула потоков позволяет упростить учёт создаваемых потоков и выполняемых ими задач, избежать утечек памяти, а также создавать синхронизированные элементы, не конфликтующие друг с другом, в автоматическом режиме. Эти плюсы позволяют использовать пул потоков в данном проекте с чрезвычайной выгодой.

Пул потоков имеет в наличии следующие методы:

- SetThreadCount. Установить максимальное значение доступных потоков. Это необходимо для контроля загрузки аппаратных ресурсов системы. Возможны случаи, когда будет использоваться не максимальное количество потоков, однако зарезервированные места будут созраться за ними;

- SetThreadState. Установить состояние потока. У элементов пула потоков может быть три состояния: FREE – свободен и готов к работе, EXECUTE – занят задачей и EXCEPTION – возникла ошибка во время выполнения задачи. Когда поток свободен, он захватывается при вызове LaunchThread, принимает задачу и устанавливает статус EXECUTE. При возникновении ошибки в потоке и установке статуса EXCEPTION, данный поток перестаёт функционировать до тех пор, пока исключение не будет устранено;

- LaunchThread. Запустить поток. Функция работает в неблокирующем режиме. Подразумевается поиск свободного потока или элемента с исключением, блокировка состоянием EXECUTE и последующее выполнение назначенной задачи. В приведённом контексте, задачей именуется указатель на

функцию с параметрами, которую требуется выполнить в отдельном потоке. Указанная функция передаётся в метод `LaunchThread`. Если не удалось найти подходящий поток – возвращается `false`, в иных случаях – `true`;

– `WaitAllThreads`. Ожидание завершения потоков. Функция остановки пула потоков с ожиданием завершения элементов. Если в пуле присутствуют активные потоки, то для них стартует таймер, по истечении которого, данные элементы будут завершены принудительно. Такой подход к завершению позволяет выполнить оставшиеся задачи за некричный период времени.

1.7.2.3 Класс `DaemonTask`

Затрагиваемый класс описывает задачу, поставленную демону. Исходя из специфики релизуемого конвейера, под задачей понимается определённая директория, в которой находится файл для обработки (далее, источник) и json-файл параметров обработки, описывающий требуемые преобразования над файлом. Расположение источника и тип файла (даёт возможность преобразовывать другие типы данных) обозначены как `source_` и `type_`. Однако, также требуется указать расположение результата после преобразования. Данный параметр хранится в поле `destination_`. Класс используется при поиске и описании соответствующих директорий в хранилище.

1.7.2.4 Класс `DaemonException`

Индивидуальное исключение, характерное только для внутренних процессов демона. Является перегрузкой исключения `std::exception` из стандартной библиотеки STL. Данная модификация необходима для универсализации всех исключений демона и возможности записи индивидуальных сообщений при возникновении некорректных ситуаций.

1.7.2.5 Класс `DaemonBase`

Является абстрактным родительским классом, содержащим фиксированные и виртуальные методы демона. Как и все абстрактные классы для использования в виде объекта требует обязательное наличие наследника с перегруженными методами. Такими наследниками выступают классы:

AudioCutter, ChunksCutter, FramesCutter, Neuralizator, FramesGluer, ChunksGluer, AudioGluer. Приемники перегружают виртуальные методы родителя, тем самым реализуют собственную логику работы в данных функциях. Это позволит универсализировать подход при использовании пула потоков ThreadPool.

Фиксированные методы:

– GetTaskList. Получить список задач для демона. При начальной настройке, программе, в обязательном порядке, задаются директории с предполагаемыми местами появления источников. Данный механизм позволяет отслеживать появление новых задач в определённом расположении, не расходуя попытки на успешное получение задачи извне. За получение списка задач из определённой директории и отвечает метод GetTaskList. Он принимает строку с положением поиска. После завершения поиска, возвращает вектор с элементами DaemonTask, упорядоченными по убыванию относительно даты. Указанная организация позволяет убить сразу двух зайцев: получить готовый список задач и распределить их по приоритету выполнения;

– SetDaemonName. Установить обозначение имени в логере;

– ParseSettings. Считать настройки для демона из json-файла. Файл настроек демона – json-файл, организованный специальным образом, содержащий необходимые параметры для работы программы, такие как: продолжительность работы, расположения директории источников и результатов, время ожидания завершения пула потоков, название демона. При провале проверки возникает критическое исключение, завершающее работу. Это позволяет не расходовать лишние ресурсы на поддержку нерабочего демона;

– Launch. Запуск основного цикла демона. Вызывается сразу после создания объекта дочернего класса. Вызывает единственный раз ParseSettings и SetDaemonName для выполнения первичной настройки, а также проверяет валидность полученных параметров. При успешном выполнении, вызывает функцию Lifeloop, отвечающую за цикл работы программы;

– Stop. Остановка основного цикла демона. Вызывается во время появления критической ошибки для аварийного прерывания работы демона или после истечения таймаута работы для запланированного перезапуска. Является посредником для вызова функции WaitAllThread.

Перегружаемые методы:

– Lifeloop. Основной цикл демона. Перегружается, так как для разных сценариев работы существуют разные способы организации основного цикла демона. Не имеет чёткого шаблона для использования в дочерних классах, однако рекомендуется использовать цикл с ограничением по времени. Таким образом, можно реализовать таймер работы основного цикла. Для использования пула потоков реализующего неблокирующий режим работы, необходимо вызвать LaunchThread с указателем на ранее параметризованную функцию VideoExecutor или ImageExecutor. Запуск в блокирующем режиме подразумевает прямой вызов функций VideoExecutor или ImageExecutor;

– VideoExecutor. Обработчик файлов типа видео. Реализует в себе необходимую логику обработки видеофайлов для данного демона, которая может быть использована напрямую (в блокирующем режиме) и через пул потоков (в неблокирующем режиме);

– ImageExecutor. Обработчик файлов типа изображение. Аналог VideoExecutor, но реализующий обработку для другого типа данных.

Подводя небольшой итог, стоит отметить, что DaemonBase служит основным классом для реализации классов отдельных демонов. Использование такого подхода позволяет не переписывать повторяющийся код по несколько раз. Также, это позволяет довольно быстро создавать новых потомков и поддерживать старых.

1.7.2.6 Класс BaseUpscaleSettings

Используется как описатель параметров для разных типов обрабатываемых данных. В описываемой реализации, могут обрабатываться изображения и видео, поэтому существуют потомки данного класса в лице VideoUpscaleSettings и

ImageUpscaleSettings, отличные друг от друга различными полями. Было принято решение скопировать универсальный подход, использованный в классе DaemonBase. BaseUpscaleSettings – абстрактный класс, возможность реализации которого заложена в наследниках. В частности, существуют виртуальные методы Parse и Print для парсинга и записи параметров изображения или видео. Данные функции перегружаются в зависимости от наследника и используют собственные проверки параметров для каждого типа.

Такой подход к настройкам изображений и видео можно также назвать универсальным, ведь можно продолжить создавать различные ветви потомков для разных сценариев работы демонов.

1.7.2.7 Классы-наследники

Под классами наследниками подразумеваются элементы: AudioCutter, ChunksCutter, FramesCutter, Neuralizator, FramesGluer, ChunksGluer, AudioGluer. Логика работы данных элементов представляет собой полную копию описания из архитектуры проекта (пункт 1.6). Взаимодействие между элементами также происходит по описанному ранее сценарию.

Наследники, реализуя собственные функции, перегружают функции Lifeloop, VideoExecutor и ImageExecutor. Это увеличивает гибкость системы и позволяет изменять вышеописанных демонов под любые нужды.

1.8 Тестирование и отладка

Тестирование конвейера стоит начать с подбора видеоматериалов и настроек для преобразования. Выбор был сделан в пользу короткого отрывка из мультика «Винни Пух» в формате mp4. Данное видео обладает следующими характеристиками:

- разрешение: 960x720;
- частота кадров: 30 кадров в секунду;
- продолжительность: 50 секунд.

В результате конвертирования необходимо проанализировать разницу в качестве оригинального и модифицированного контента, а также получить следующие параметры:

- разрешение: 3840x2160;
- частота кадров: 60 кадров в секунду;
- продолжительность: 50 секунд.

После формирования директории с обрабатываемым файлом (Vinipooh.mp4) и параметрами (upscale_settings.mp4) требуется провести копирование данной директории в расположение начальной точки работы конвейера (рисунки 16, 17, 18).

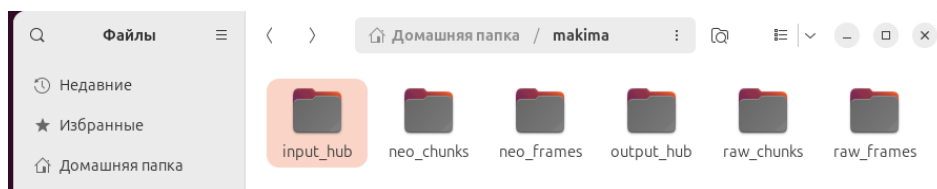


Рисунок 16 – Директория с расположениями точек отсчёта микросервисов



Рисунок 17 – Директория с обрабатываемым файлом и параметрами

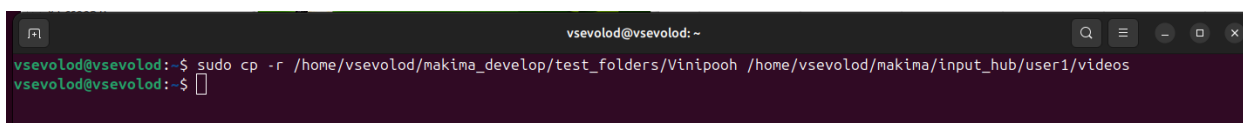


Рисунок 18 – Копирование директории обработки в начальную точку конвейера

После завершения копирования осуществляется запуск скрипта установки конвейера, написанного на языке Bash – install.sh. Скрипт выполняет роль установщика (загрузчика) демонов, а также утилиты обновления программного комплекса (рисунок 19).

```
21 мая 2024 05:30:03 +04 [ makima | SUCCESS ] Move daemon to daemon directory: /opt/makimamakina/frames_cutter
21 мая 2024 05:30:03 +04 [ makima | INFO ] Update daemon .service file: /opt/makimamakina/frames_cutter/frames_cutter.service
21 мая 2024 05:30:03 +04 [ makima | SUCCESS ] Daemon .service file was updated: /opt/makimamakina/frames_cutter/frames_cutter.service
21 мая 2024 05:30:03 +04 [ makima | INFO ] Create daemon symlink in systemd: /opt/makimamakina/frames_cutter/frames_cutter.service
21 мая 2024 05:30:03 +04 [ makima | SUCCESS ] Symlink /etc/systemd/system/frames_cutter.service exists and actual.
21 мая 2024 05:30:03 +04 [ makima | SUCCESS ] Daemon symlink was created.
21 мая 2024 05:30:03 +04 [ makima | INFO ] Daemon detected: /home/vsevolod/makima_develop/installer/daemons/frames_gluer
21 мая 2024 05:30:03 +04 [ makima | INFO ] Create daemon directory: /opt/makimamakina/frames_gluer
21 мая 2024 05:30:03 +04 [ makima | WARNING ] Directory /opt/makimamakina/frames_gluer existed.
21 мая 2024 05:30:03 +04 [ makima | INFO ] Move daemon to daemon directory: /opt/makimamakina/frames_gluer
sending incremental file list
frames_gluer
3,00M 100% 141,54MB/s 0:00:00 (xfr#1, to-chk=0/1)
sending incremental file list
frames_gluer.service
133 100% 0,00kB/s 0:00:00 (xfr#1, to-chk=0/1)
21 мая 2024 05:30:04 +04 [ makima | SUCCESS ] Move daemon to daemon directory: /opt/makimamakina/frames_gluer
21 мая 2024 05:30:04 +04 [ makima | INFO ] Update daemon .service file: /opt/makimamakina/frames_gluer/frames_gluer.service
21 мая 2024 05:30:04 +04 [ makima | SUCCESS ] Daemon .service file was updated: /opt/makimamakina/frames_gluer/frames_gluer.service
21 мая 2024 05:30:04 +04 [ makima | INFO ] Create daemon symlink in systemd: /opt/makimamakina/frames_gluer/frames_gluer.service
21 мая 2024 05:30:04 +04 [ makima | SUCCESS ] Symlink /etc/systemd/system/frames_gluer.service exists and actual.
21 мая 2024 05:30:04 +04 [ makima | SUCCESS ] Daemon symlink was created.
21 мая 2024 05:30:04 +04 [ makima | INFO ] Daemon detected: /home/vsevolod/makima_develop/installer/daemons/neuralizator
21 мая 2024 05:30:04 +04 [ makima | INFO ] Create daemon directory: /opt/makimamakina/neuralizator
21 мая 2024 05:30:04 +04 [ makima | WARNING ] Directory /opt/makimamakina/neuralizator existed.
21 мая 2024 05:30:04 +04 [ makima | INFO ] Move daemon to daemon directory: /opt/makimamakina/neuralizator
sending incremental file list
neuralizator
3,00M 100% 166,50MB/s 0:00:00 (xfr#1, to-chk=0/1)
sending incremental file list
neuralizator.service
133 100% 0,00kB/s 0:00:00 (xfr#1, to-chk=0/1)
21 мая 2024 05:30:04 +04 [ makima | SUCCESS ] Move daemon to daemon directory: /opt/makimamakina/neuralizator
21 мая 2024 05:30:04 +04 [ makima | INFO ] Update daemon .service file: /opt/makimamakina/neuralizator/neuralizator.service
21 мая 2024 05:30:04 +04 [ makima | SUCCESS ] Daemon .service file was updated: /opt/makimamakina/neuralizator/neuralizator.service
21 мая 2024 05:30:04 +04 [ makima | INFO ] Create daemon symlink in systemd: /opt/makimamakina/neuralizator/neuralizator.service
21 мая 2024 05:30:04 +04 [ makima | SUCCESS ] Symlink /etc/systemd/system/neuralizator.service exists and actual.
21 мая 2024 05:30:04 +04 [ makima | SUCCESS ] Daemon symlink was created.
```

Рисунок 19 – Фрагмент лога скрипта установки install.sh

По окончании установки (повторной загрузки) будет открыт специализированный консольный интерфейс администратора, отображающий активность установленных демонов. Рассматриваемый интерфейс устанавливается в систему автоматически и повторно доступен при вводе в терминал команды `makima` (рисунок 20).

```
21 мая 04:48:58 vsevolod audio_cutter[6849]: [ audio_cutter ] [ INFO ] Задача /home/vsevolod/makima/input_hub/user1/videos/Vlnlpooh/Vlnlpooh.mp4 была выполнена за 0 сек. / 42 миллисек.
```

Рисунок 20 – Интерфейс администратора конвейера

Можно заметить, что в интерфейсе уже появилась первая запись о работе демона `audio_cutter`. Появление записи формата: «Задача *** была выполнена за * сек. / * миллисек» означает выполнение определённой задачи одним из демонов конвейера. Дату, уровень события и наименование демона, оповестившего интерфейс можно увидеть в самом начале записи. Для однозначной фиксации результата перейдём в расположение результата выполненной задачи (рисунок 21).

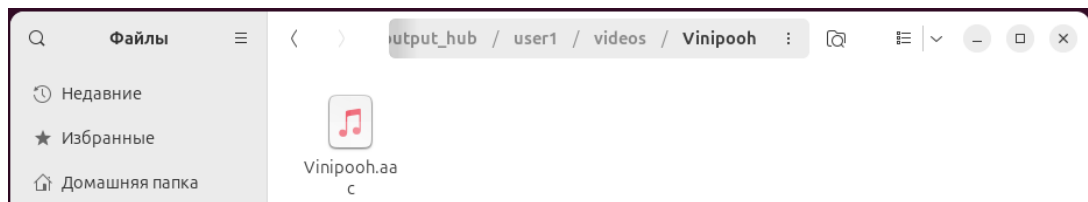


Рисунок 21 – Подтверждение результата работы audio_cutter

После подтверждения результата корректной работы audio_cutter продолжаем ожидать следующих сообщений в интерфейсе. Как и ожидалось при успешном сценарии, следующим интерфейс оповестил демон chunks_cutter (рисунок 22).

```

мая 21 04:48:50 vsevolod audio_cutter[6849]: [ audio_cutter ] [ INFO ] Задача /home/vsevolod/makina/input_hub/user1/videos/Vinipooh/Vinipooh.mp4 была выполнена за 0 сек. / 42 микросек.
мая 21 04:50:31 vsevolod systemd[1]: Started chunks_cutter.service - chunks_cutter.
мая 21 04:50:31 vsevolod chunks_cutter[7587]: [ unknown ] [ INFO ] Пул потоков инициализирован
мая 21 04:50:31 vsevolod chunks_cutter[7587]: [ chunks_cutter ] [ INFO ] Успешный парсинг файла настроек демона chunks_cutter по пути: /etc/makina/chunks_cutter.json
мая 21 04:50:31 vsevolod chunks_cutter[7587]: [ chunks_cutter ] [ INFO ] Демон запущен.
мая 21 04:50:31 vsevolod chunks_cutter[7587]: [ chunks_cutter ] [ INFO ] Создана директория для необработанных чанков /home/vsevolod/makina/raw_chunks/user1/Vinipooh
мая 21 04:50:31 vsevolod chunks_cutter[7587]: [ chunks_cutter ] [ INFO ] Файл настроек скопирован по расположению /home/vsevolod/makina/raw_chunks/user1/Vinipooh/upscale_settings.json
мая 21 04:50:31 vsevolod chunks_cutter[7587]: [ chunks_cutter ] [ INFO ] Задача /home/vsevolod/makina/input_hub/user1/videos/Vinipooh/Vinipooh.mp4 была выполнена за 0 сек. / 44 микросек.

```

Рисунок 22 – Оповещение интерфейса администратора конвейера службой chunks_cutter

В этот раз строчек лога чуть больше, чем было от audio_cutter. Это связано с тем, что при появлении задания на выполнение, все демоны выходят из неактивного режима, перезапускаясь и начиная работу заново. Переход в неактивный режим произойдёт по истечению таймаута работы демона, если в данный момент нет актуальных задач. chunks_cutter помечает задачу как выполненную, требуется фиксация (рисунок 23).



Рисунок 23 – Подтверждение результата работы chunks_cutter

Как и положено, кроме сегментов видео, в директории результата расположен файл подтверждения обработки (makima.access) и параметры

улучшения видео (upscale_settings.json). Файл подтверждения обработки используется в качестве подтверждения завершения работы в этой директории, а параметры улучшения видео необходимо передавать следующим демонам, которым требуется информация о характере производимых изменений.

Следующие записи не заставляют себя долго ждать. Демон frames_cutter начинает деление видео на последовательности изображений. Выделение и сохранение фреймов из видео является процессом, требующим некоторое количество времени. Это позволяет нам, в отличие от предыдущих демонов, наглядно проверить механизм работы блокирующих файлов (рисунки 24 – 27).

```
мая 21 04:48:59 vsevolod audio_cutter[6849]: [ audio_cutter ] [ INFO ] Задача /home/vsevolod/makima/input_hub/user1/videos/Vinipooh/Vinipooh.mp4 была выполнена за 0 сек. / 42 миллисек.
мая 21 04:50:31 vsevolod systemd[1]: Started chunks_cutter.service - chunks_cutter.
мая 21 04:50:31 vsevolod chunks_cutter[7587]: [ unknown ] [ INFO ] Пул потоков инициализирован
мая 21 04:50:31 vsevolod chunks_cutter[7587]: [ chunks_cutter ] [ INFO ] Успешный парсинг файла настроек демона chunks_cutter по пути: /etc/makima/chunks_cutter.json
мая 21 04:50:31 vsevolod chunks_cutter[7587]: [ chunks_cutter ] [ INFO ] Демон запущен.
мая 21 04:50:31 vsevolod chunks_cutter[7587]: [ chunks_cutter ] [ INFO ] Создана директория для необработанных чанков /home/vsevolod/makima/raw_chunks/user1/Vinipooh
мая 21 04:50:31 vsevolod chunks_cutter[7587]: [ chunks_cutter ] [ INFO ] Файл настроек скопирован по расположению /home/vsevolod/makima/raw_chunks/user1/Vinipooh/upscale_settings.json
мая 21 04:50:31 vsevolod chunks_cutter[7587]: [ chunks_cutter ] [ INFO ] Задача /home/vsevolod/makima/input_hub/user1/videos/Vinipooh/Vinipooh.mp4 была выполнена за 0 сек. / 44 миллисек.
мая 21 04:52:12 vsevolod systemd[1]: Started frames_cutter.service - frames_cutter.
мая 21 04:52:12 vsevolod frames_cutter[8428]: [ unknown ] [ INFO ] Пул потоков инициализирован
мая 21 04:52:12 vsevolod frames_cutter[8428]: [ frames_cutter ] [ INFO ] Успешный парсинг файла настроек демона frames_cutter по пути: /etc/makima/frames_cutter.json
мая 21 04:52:12 vsevolod frames_cutter[8428]: [ frames_cutter ] [ INFO ] Демон запущен.
мая 21 04:52:12 vsevolod frames_cutter[8428]: [ frames_cutter ] [ INFO ] Создана директория для необработанных фреймов /home/vsevolod/makima/raw_frames/user1/Vinipooh/chunk_2
мая 21 04:52:12 vsevolod frames_cutter[8428]: [ frames_cutter ] [ INFO ] Файл настроек скопирован по расположению /home/vsevolod/makima/raw_frames/user1/Vinipooh/upscale_settings.json
мая 21 04:52:12 vsevolod frames_cutter[8428]: [ frames_cutter ] [ INFO ] Создана директория для необработанных фреймов /home/vsevolod/makima/raw_frames/user1/Vinipooh/chunk_0
мая 21 04:52:13 vsevolod frames_cutter[8428]: [ frames_cutter ] [ INFO ] Создана директория для необработанных фреймов /home/vsevolod/makima/raw_frames/user1/Vinipooh/chunk_3
мая 21 04:52:13 vsevolod frames_cutter[8428]: [ frames_cutter ] [ INFO ] Создана директория для необработанных фреймов /home/vsevolod/makima/raw_frames/user1/Vinipooh/chunk_1
```

Рисунок 24 – Оповещение интерфейса администратора конвейера службой frames_cutter

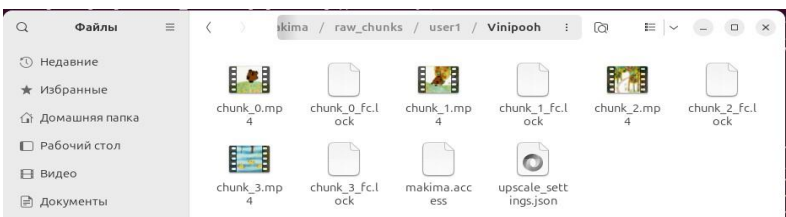


Рисунок 25 – Появление блокирующих файлов при работе frames_cutter в каталоге результатов chunks_cutter



Рисунок 26 – Подтверждение результата работы frames_cutter

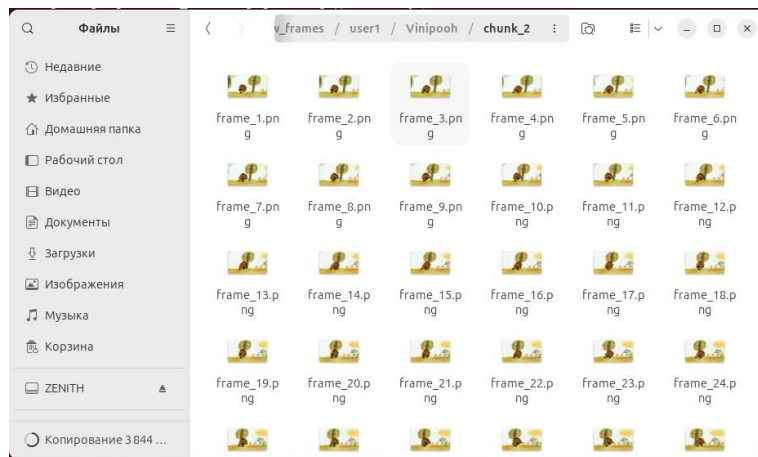


Рисунок 27 – Фреймы, расположенные в одной из папок результата работы frames_cutter

Работа демона frames_cutter полностью корректна. Уведомление произошло в штатном режиме, блокирующие файлы после реализации задачи были удалены, директории с фреймами без потерь присутствуют в конечной точке работы.

Соответствуя корректному сценарию последовательной работы, интерфейс администратора следующим оповещает neuralizator (рисунок 28). Отследить корректное поведение данного сервиса ещё проще, чем у frames_cutter. Можно проверить наличие результата и блокирующих файлов, а также оценить текущую нагрузку видеокарты при помощи сторонних приложений (рисунки 29, 30).

```

мая 21 04:52:53 vsevolod frames_cutter[8428]: [ frames_cutter ] [ INFO ] Задача /home/vsevolod/nakina/raw_chunks/user1/Vinipooh/chunk_2.mp4 была выполнена за 48 сек. / 48458 микросек.
мая 21 04:53:01 vsevolod frames_cutter[8428]: [ frames_cutter ] [ INFO ] Задача /home/vsevolod/nakina/raw_chunks/user1/Vinipooh/chunk_3.mp4 была выполнена за 48 сек. / 48883 микросек.
мая 21 04:55:08 vsevolod system[1]: Started neuralizator.service - neuralizator.
мая 21 04:55:08 vsevolod neuralizator[11653]: [ unknown ] [ INFO ] Путь потоков инициализирован
мая 21 04:55:08 vsevolod neuralizator[11653]: [ neuralizator ] [ INFO ] Успешный парсинг файла настроек демона neuralizator по пути: /etc/nakina/neuralizator.json
мая 21 04:55:08 vsevolod neuralizator[11653]: [ neuralizator ] [ INFO ] Демон запущен.
мая 21 04:55:08 vsevolod neuralizator[11653]: [ neuralizator ] [ INFO ] Создана директория для обработанных фреймов /home/vsevolod/nakina/neo_frames/user1/Vinipooh/chunk_1
мая 21 04:55:09 vsevolod neuralizator[11653]: [ neuralizator ] [ INFO ] Файл настроек скопирован по расположению /home/vsevolod/nakina/neo_frames/user1/Vinipooh/upscale_settings.json
мая 21 04:55:09 vsevolod neuralizator[11653]: [ neuralizator ] [ INFO ] Создана директория для обработанных фреймов /home/vsevolod/nakina/neo_frames/user1/Vinipooh/chunk_2
мая 21 04:55:09 vsevolod neuralizator[11653]: [ neuralizator ] [ INFO ] Создана директория для обработанных фреймов /home/vsevolod/nakina/neo_frames/user1/Vinipooh/chunk_3
мая 21 04:55:09 vsevolod neuralizator[11653]: [ neuralizator ] [ INFO ] Создана директория для обработанных фреймов /home/vsevolod/nakina/neo_frames/user1/Vinipooh/chunk_0
мая 21 05:06:39 vsevolod neuralizator[11653]: [ neuralizator ] [ INFO ] Задача /home/vsevolod/nakina/raw_frames/user1/Vinipooh/chunk_0 была выполнена за 689 сек. / 689617 микросек.
мая 21 05:06:40 vsevolod neuralizator[11653]: [ neuralizator ] [ INFO ] Задача /home/vsevolod/nakina/raw_frames/user1/Vinipooh/chunk_1 была выполнена за 691 сек. / 691145 микросек.
мая 21 05:06:40 vsevolod neuralizator[11653]: [ neuralizator ] [ INFO ] Задача /home/vsevolod/nakina/raw_frames/user1/Vinipooh/chunk_2 была выполнена за 691 сек. / 691401 микросек.
мая 21 05:10:43 vsevolod neuralizator[11653]: [ neuralizator ] [ INFO ] Задача /home/vsevolod/nakina/raw_frames/user1/Vinipooh/chunk_3 была выполнена за 933 сек. / 933692 микросек.

```

Рисунок 28 – Оповещение интерфейса администратора конвейера службой neuralizator



Рисунок 29 – Появление блокирующих файлов при работе neuralizator в каталоге результатов frames_cutter



Рисунок 30 – Подтверждение результата работы neuralizator

Для фиксации результатов проверки работоспособности демона neuralizator производится анализ оригинального и модифицированного фрейма. Данное сравнение можно производить в любом редакторе изображений с большой кратностью зума (рисунок 31). Разница может быть не видна на экранах с низким разрешением.

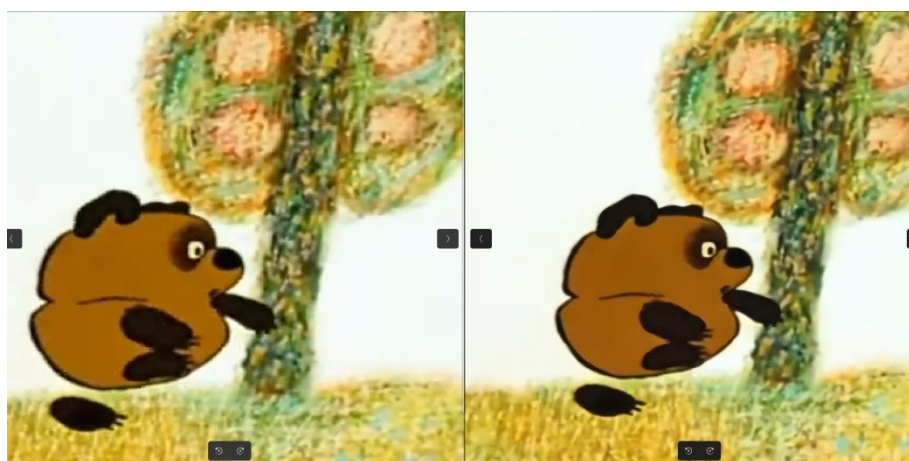


Рисунок 31 – Сравнение оригинального и полученного фрейма

Работа оставшихся трёх демонов идентична, поэтому они будут рассматриваться вместе. frames_gluer, chunks_gluer и audio_gluer являются микросервисами-антагонистами для frames_cutter, chunks_cutter, audio_cutter, соответственно. Они собирают полученные результаты обработки в полноценный итоговый файл (рисунки 32 – 37).

```
мая 21 05:14:15 vsevolod systemd[1]: Started frames_gluer.service - frames_gluer.
мая 21 05:14:15 vsevolod frames_gluer[24609]: [ unknown ] [ INFO ] Пул потоков инициализирован
мая 21 05:14:15 vsevolod frames_gluer[24609]: [ frames_gluer ] [ INFO ] Успешный парсинг файла настроек демона frames_gluer по пути: /etc/makina/frames_gluer.json
мая 21 05:14:15 vsevolod frames_gluer[24609]: [ frames_gluer ] [ INFO ] Демон запущен.
мая 21 05:14:15 vsevolod frames_gluer[24609]: [ frames_gluer ] [ INFO ] Создана директория для обработанных чанков /home/vsevolod/makina/neo_chunks/user1/Vinipooh
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: HEVC encoder version 3.5+1-f0c1022b6
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: build info [Linux][GCC 13.2.0][64 bit] 8bit+10bit+12bit
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: using cpu capabilities: MMX2 SSE2Fast LZCNT SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [warning]: halving the quality when psy-rd is enabled for 444 input. Setting cbQpOffset = 6 and crQpOffset = 6
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: Main 4:4:4 profile, Level-5 (Main tier)
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: Thread pool created using 12 threads
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: Slices : 1
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: frame threads / pool features : 3 / wpp(34 rows)
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: Coding QT: max CU size, min CU size : 64 / 8
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: Residual QT: max TU size, max depth : 32 / 1 inter / 1 intra
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: ME / range / subpel / merge : hex / 57 / 2 / 3
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: Keyframe min / max / scenecut / bias : 25 / 250 / 40 / 5.00
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: Cb/Cr QP Offset : 6 / 6
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: Lookahead / bframes / badapt : 20 / 4 / 2
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: b-pyramid / weightp / weightb : 1 / 1 / 0
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: References / ref-limit cu / depth : 3 / off / on
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: AQ: mode / str / qg-size / cu-tree : 2 / 1.0 / 32 / 1
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: Rate Control / qCompress : CRF-28.0 / 0.60
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: tools: rd=3 psy-rd=2.00 early-skip rskip mode=1 signhide tmvp
мая 21 05:14:16 vsevolod frames_gluer[24612]: x265 [info]: tools: b-intra strong-intra-smoothing lslices=8 deblock sao
```

Рисунок 32 – Оповещение интерфейса администратора конвейера службой frames_gluer

```
мая 21 05:25:41 vsevolod systemd[1]: Started chunks_gluer.service - chunks_gluer.
мая 21 05:25:41 vsevolod chunks_gluer[30218]: [ unknown ] [ INFO ] Пул потоков инициализирован
мая 21 05:25:41 vsevolod chunks_gluer[30218]: [ chunks_gluer ] [ INFO ] Успешный парсинг файла настроек демона chunks_gluer по пути: /etc/makina/chunks_gluer.json
мая 21 05:25:41 vsevolod chunks_gluer[30218]: [ chunks_gluer ] [ INFO ] Демон запущен.
мая 21 05:25:59 vsevolod chunks_gluer[30218]: [ chunks_gluer ] [ INFO ] Задача /home/vsevolod/makina/neo_chunks/user1/Vinipooh была выполнена за 1 сек. / 1804 миллисек.
```

Рисунок 33 – Оповещение интерфейса администратора конвейера службой chunks_gluer

```
мая 21 05:27:36 vsevolod systemd[1]: Started audio_gluer.service - audio_gluer.
мая 21 05:27:36 vsevolod audio_gluer[31060]: [ unknown ] [ INFO ] Пул потоков инициализирован
мая 21 05:27:36 vsevolod audio_gluer[31060]: [ audio_gluer ] [ INFO ] Успешный парсинг файла настроек демона audio_gluer по пути: /etc/makina/audio_gluer.json
мая 21 05:27:36 vsevolod audio_gluer[31060]: [ audio_gluer ] [ INFO ] Демон запущен.
мая 21 05:27:57 vsevolod audio_gluer[31060]: [ audio_gluer ] [ INFO ] Задача /home/vsevolod/makina/output_hub/user1/videos/Vinipooh/Vinipooh.mp4 была выполнена за 16 сек. / 16227 сек.
```

Рисунок 34 – Оповещение интерфейса администратора конвейера службой audio_gluer



Рисунок 35 – Подтверждение результата работы frames_gluer

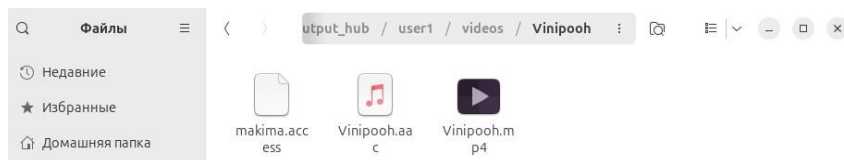


Рисунок 36 – Подтверждение результата работы chunks_gluer

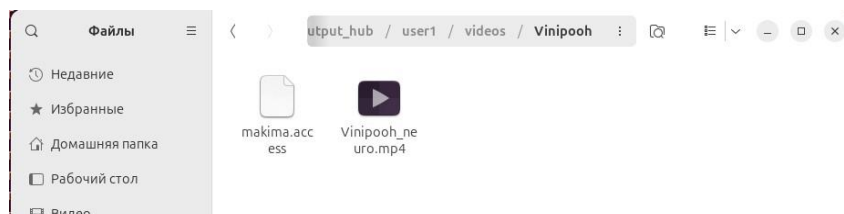


Рисунок 37 – Подтверждение результата работы audio_gluer

При благополучном сценарии работы (какой и получилось реализовать в данном разделе) будет создан файл Vinipooh_neuro.mp4 с соответствующими модификациями параметров. Далее, данный файл может быть использован веб-приложением или открыт на машине администратора для дальнейшего использования.

2 Охрана труда

2.1 Описание негативных факторов, воздействующих на разработчика и способы их устранения

Взаимодействие с компьютером происходит в условиях минимальной двигательной активности и длительного мышечного напряжения, что приводит к негативным последствиям, таким как:

- снижение работоспособности;
- ухудшение зрения;
- развитие гиподинамии;
- повышение психологического раздражения;
- усталость и переутомление.

Для уменьшения влияния перечисленных выше последствий на человека, необходимо придерживаться определённых правил работы за компьютером:

- соблюдать режим труда и отдыха;
- правильно организовать рабочее пространство;
- соблюдать правила обращения с электроприборами;
- не пренебрегать правилами пожарной безопасности.

2.1.1 Оборудование рабочего места

Существует несколько основных требований для организации рабочего места:

- расстояние от глаз до экрана должно быть не менее 500 мм;
- монитор должен располагаться ниже уровня глаз;
- угол между направлением при взгляде прямо и направлением на центр экрана должен составлять примерно 30 градусов;
- высота рабочего стола должна составлять от 620 до 820 мм;
- пространство для ног у рабочего стола должно иметь высоту – не менее 600 мм, глубину на уровне колен – не менее 450 мм, глубину на уровне вытянутых ног – не менее 650 мм.

На рисунке 38 изображена подходящая конфигурация рабочего места, которая отвечает указанным выше требованиям.

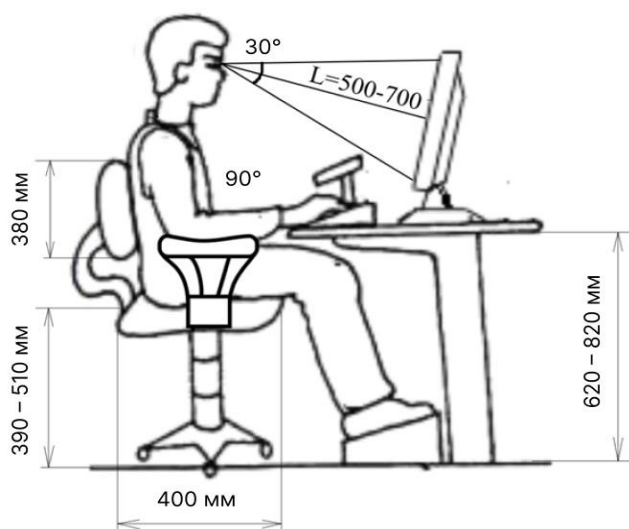


Рисунок 38 – Требуемая организация рабочего места

Рабочее место с дисплеем должно обеспечивать оператору возможность удобного выполнения работ в положении сидя и не создавать перегрузки костно-мышечной системы.

Когда рабочее место правильно организовано необходимо обеспечить корректную посадку пользователя ПК, изображённую на рисунке 39.

Удобное расположение должно обеспечивать выполнение следующих условий:

- запястья в прямом положении;
- плечи расслаблены;
- локти не должны свисать и располагаться перпендикулярно полу (угол в локтевых суставах должен составлять 90-100 градусов).

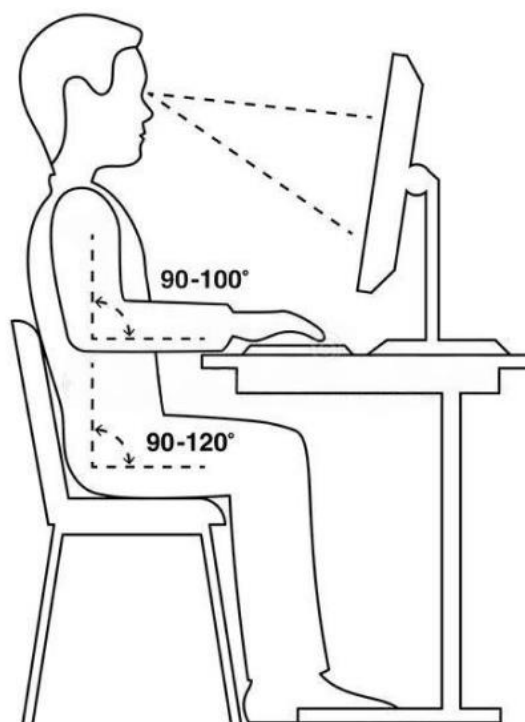


Рисунок 39 – Пример правильной посадки за персональный компьютер

2.1.2 Освещенность рабочего места

Чтобы обеспечить максимально комфортные условия работы на компьютере важно учитывать стандарты освещения, которые зависят от характера работы, расположения и поверхности рабочего места. Рекомендуется

достигать комбинированного освещения клавиатур и дисплеев не менее 400 люкс и общего освещения не менее 200 люкс. Одновременно необходимо обеспечивать равномерную яркость на рабочем месте, избегая ярких и блестящих предметов. Для этого часто используются потолочные или встроенные светильники с люминисцентными лампами, которые рекомендуется располагать сбоку от рабочего места, параллельно линии зрения пользователя, чтобы предотвратить засветку экранов.

2.1.3 Электробезопасность

Требования к данному пункту предъявляются по ГОСТ 12.1.019-2017 «ЭЛЕКТРОБЕЗОПАСНОСТЬ».

ПК – это комплекс устройств, работающий от сети переменного тока частотой 50 Гц с напряжением 230 В. Электрический ток при таком напряжении опасен для жизни. Поэтому при работе на ПК необходимо соблюдение следующих правил электробезопасности:

- все устройства должны питаться от одной фазы электрической сети;
- сетевое электропитание устройств ПК должно производиться только от розеток типа «Европа» с заземляющими контактами;
- все электрические розетки, предназначенные для подключения к ним устройств ПК, должны иметь маркировку по напряжению;
- значение номинального напряжения сети необходимо наносить яркой краской, крупными символами (высотой не менее 50 мм) на стене или щите, возле розетки.

При использовании компьютерной техники и периферийных устройств, требуется соблюдение мер безопасности и более внимательное отношение к электропроводке, приборам и оборудованию, так как нарушение правил безопасности может представлять угрозу для здоровья и жизни. Необходимо регулярно проверять состояние электропроводки, выключателей и розеток, которые используются для подключения оборудования к сети. Если обнаружена

неисправность, необходимо немедленно отключить электрооборудование. Продолжить работу возможно только после устранения всех неисправностей.

2.1.4 Пожарная безопасность

При эксплуатации персональных компьютеров наиболее частыми причинами пожаров, возникающих при использовании вычислительных машин, являются:

- короткое замыкание в проводке и электрических устройствах;
- воспламенение горючих материалов вблизи от электрических устройств;
- большие переходные сопротивления в местах контактных соединений.

Пожарные меры профилактики проводятся согласно федеральному закону от 21.12.1994 №69-ФЗ (ред. от 28.09.2010) "О пожарной безопасности". Для пожарной безопасности требуется выполнить ряд организационных мероприятий, направленных на обеспечение безопасности людей и предотвращение пожара с ограничением его распространения. А также создать условия для успешного тушения пожара с ограничением его распространения. Для повышения противопожарной безопасности помещения необходимы следующие мероприятия:

- периодическая проверка проходов в помещения и запасных выходов. Они не должны быть загромождены посторонними предметами, препятствующими эвакуации в чрезвычайных случаях;

- периодическая проверка средств пожаротушения и свободного доступа к ним;

- оборудование системы вентиляции устройством, обеспечивающим автоматическое отключение данной системы при пожаре, а также огнезадерживающим клапаном;

- установка и периодическая проверка автоматической пожарной сигнализации и средств пожаротушения. Данные блоки должны находиться в исправном состоянии.

					ПГУ1.090301.15.001 ПЗ	Лист
						64
Изм.	Лист	№ докум.	Подпись	Дата		

2.1.5 Режим труда и отдыха

Пользователь ПК обязан соблюдать режим труда и отдыха в зависимости от продолжительности, вида и категории трудовой деятельности.

Виды трудовой деятельности, связанные с ПК можно разделить на несколько групп:

- группа А – работы по считыванию информации с дисплея по предварительному запросу;
- группа Б – работы по вводу информации;
- группа В – творческая работа в режиме диалога с ПК.

При выполнении работ, относящихся к разным видам трудовой деятельности, за основную работу с персональным компьютером следует принимать тот вид деятельности, который занимает не менее 50% времени в течение всей рабочей смены или рабочего дня.

Если подробнее рассмотреть виды трудовой деятельности, то можно выделить три категории тяжести и напряжённости работы с ПК, которые описаны в таблице 4.

Таблица 4 – Время перерывов

Категория работы с ПЭВМ или ВДТ	Уровень нагрузки за рабочую смену при видах работы с ВДТ			Суммарное время регламентированных перерывов, мин	
	группа А, количество знаков	группа Б, количество знаков	группа В, количество часов	при 8- часовой смене	При 12- часовой смене
1	До 20000	До 15000	До 2	50	80
2	До 40000	До 30000	До 4	70	110
3	До 60000	До 40000	До 6	90	140

3 Стартап проект

Инициатор проекта планирует разработку веб-приложения улучшения качества видео и изображений с помощью специализированного конвейера обработки графического контента.

Разместить бизнес планируется в г. Пенза, в помещении, соответствующем всем нормам и требованиям РК.

Данный проект призван упростить рутинные задачи, расширить круг пользователей нейросетей, делая их более простыми и доступными для широкой аудитории, ведь задача улучшения качества видеофайлов и изображений является востребованной и особенно актуальной в наше время. Также деятельность проекта направлена на извлечение прибыли для улучшения материального благосостояния его участников.

Целевой группой планируемого предприятия будут являться люди из различных сфер профессиональной и коммерческой деятельности, т.е. профессионалы в различных отраслях, таких как:

- фотосъёмка и видеосъёмка. Нуждаются в улучшении своей работы, чтобы соответствовать ожиданиям клиентов и оставаться конкурентоспособными на рынке;
- агентства по маркетингу и рекламе. Требуют высококачественного визуального контента, чтобы эффективно продвигать продукты и услуги;
- интернет-магазины. Нуждаются в демонстрации продуктов в лучшем виде, чтобы увеличить продажи и удовлетворенность клиентов;
- медицинские и научные учреждения. Могут нуждаться в улучшении медицинских изображений или видео для диагностических или исследовательских целей;
- следственное дело. Нуждаются в получении достоверной доказательной базы путём улучшения материалов с камер видеонаблюдения;

– личные и социальные потребности. Люди могут хотеть улучшить свои личные фотографии и видео для социальных сетей, личных воспоминаний, творческого выражения.

3.1 Сегменты ЦА

Для более эффективного взаимодействия с каждым сегментом ЦА, необходимо учитывать их потребности и предпочтения.

Фотографы и видеографы:

– эта группа профессионалов в сфере визуального контента сталкивается с постоянно растущими требованиями клиентов к качеству фото- и видеоматериалов. Современные клиенты ожидают высокого разрешения, четкости, профессионального освещения и обработки;

– для фотографов и видеографов важно использовать передовые технологии и постоянно совершенствовать свои навыки, чтобы оставаться конкурентоспособными на рынке. Они должны быстро адаптироваться к меняющимся тенденциям и ожиданиям клиентов;

– наше предприятие сможет предложить фотографам и видеографам решения, которые значительно повысят качество и эффективность их работы.

Это может включать в себя:

- 1) предоставление доступа к профессиональному оборудованию и программному обеспечению для съемки и обработки;
- 2) обучение и консультации по передовым техникам фото- и видеосъемки;
- 3) рекомендации по эффективным методам управления проектами и работы с клиентами;
- 4) возможность аутсорсинга отдельных этапов производства контента.

Агентства по маркетингу и рекламе:

– рекламные агентства нуждаются в высококачественном визуальном контенте для эффективного продвижения товаров и услуг своих клиентов. Привлекательные, запоминающиеся изображения и видео играют ключевую роль в современных маркетинговых кампаниях;

– зачастую рекламные агентства сталкиваются с необходимостью создания большого объема визуального контента в сжатые сроки. Это предъявляет высокие требования к скорости и эффективности производственных процессов;

– наше предприятие сможет предложить рекламным агентствам решения, которые позволят быстро и качественно создавать требуемый визуальный контент. Это может включать в себя:

- 1) предоставление доступа к обширной базе готовых изображений, видео и других медиаматериалов;
- 2) возможность быстрого заказа и получения индивидуальных фото- и видеосъемок;
- 3) услуги по профессиональной обработке и ретуши визуального контента;
- 4) консультации по выбору наиболее эффективных визуальных решений для маркетинговых кампаний.

Интернет-магазины:

– для интернет-магазинов крайне важно демонстрировать свои товары в наилучшем возможном виде. Качественные фотографии и видео помогают покупателям лучше оценить характеристики, внешний вид и детали товаров, что в значительной степени влияет на их решение о покупке;

– визуальный контент не только помогает продавать товары, но также играет важную роль в создании привлекательного и запоминающегося бренда интернет-магазина. Профессионально выполненные изображения и видео формируют благоприятное первое впечатление и повышают доверие клиентов;

– наше предприятие сможет предложить интернет-магазинам комплексные решения для улучшения презентации их товаров:

- 1) профессиональная фотосъемка и видеосъемка продукции с использованием передового оборудования;
- 2) обработка и ретушь изображений для достижения максимальной привлекательности;
- 3) создание стильных, брендированных визуальных материалов;
- 4) консультации по эффективным методам использования визуального контента на сайте и в маркетинговых кампаниях.

– такие решения помогут интернет-магазинам повысить конверсию, увеличить продажи и сформировать сильный, узнаваемый бренд.

Медицинские и научные учреждения:

– эта группа клиентов может нуждаться в улучшении качества медицинских изображений или видео для диагностических и исследовательских целей. Высокое качество визуальных материалов имеет критическое значение для точной постановки диагноза, проведения научных экспериментов и анализа результатов;

– например, медицинские учреждения могут использовать улучшенные рентгеновские снимки, микроскопические изображения или видео хирургических процедур для более эффективной диагностики и лечения. Научные организации, в свою очередь, могут нуждаться в высококачественных визуальных данных для документирования хода исследований и публикации результатов;

– наше предприятие сможет предложить медицинским и научным учреждениям решения, которые повысят качество и информативность их визуальных материалов:

- 1) использование профессионального оборудования и методик для получения четких, детализированных изображений;
- 2) обработка и улучшение визуальных данных с помощью специализированного ПО;

3) консультации экспертов по оптимальным методам визуализации для различных медицинских и научных задач.

– такие решения помогут медицинским и научным учреждениям повысить эффективность диагностики, исследований и публикаций, основанных на высококачественных визуальных данных.

Следственные органы:

– следственным органам крайне важно получать достоверную доказательную базу для успешного расследования различных происшествий и преступлений. Чёткость, детализация и достоверность визуальных доказательств, таких как видеозаписи с камер наблюдения, имеют критическое значение;

– зачастую качество видеозаписей с камер наблюдения оставляет желать лучшего – они могут быть размытыми, с плохим освещением, низким разрешением. Это серьёзно затрудняет использование таких материалов в качестве доказательств;

– наше предприятие сможет предложить следственным органам решения, которые позволят значительно повысить качество видеозаписей с камер наблюдения:

1) установка и настройка профессионального видеооборудования для получения чётких, детализированных записей;

2) применение современных методов обработки и улучшения видеоматериалов, включая повышение резкости, контраста, устранение артефактов;

3) консультации экспертов по использованию визуальных доказательств в следственных мероприятиях.

– такие решения позволят следственным органам усилить доказательную базу и повысить эффективность расследований.

Личные и социальные потребности:

					ПГУ1.090301.15.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

– помимо профессиональных сфер, наши услуги будут востребованы и среди обычных людей для удовлетворения их личных и социальных потребностей;

– многие люди хотят улучшить качество своих личных фотографий и видео для публикации в социальных сетях, создания личных воспоминаний, творческого самовыражения;

– современные смартфоны и камеры позволяют снимать высококачественный контент, но не всегда обеспечивают профессиональную обработку и ретушь. Наше предприятие сможет предложить решения для улучшения личных фото и видео, чтобы они выглядели более привлекательно и профессионально;

– это направление может быть особенно востребовано среди молодежи и активных пользователей социальных сетей, которые стремятся к созданию качественного визуального контента для личных нужд и самовыражения. Мы сможем помочь им создавать яркие, запоминающиеся изображения и видео для публикации в соцсетях;

– кроме того, наши услуги могут быть интересны людям, занимающимся творческой деятельностью – фотографами-любителями, видеоблогерами, художниками. Мы сможем предоставить им инструменты для улучшения и редактирования их работ.

Каждому сегменту ЦА будет предложено индивидуальное коммерческое предложение, учитывающее их потребности и предпочтения. Стоимость модификации стандартного продукта будет варьироваться в зависимости от объема и специфических требований каждого клиента.

Перед запуском проекта для каждого сегмента ЦА важно найти ключевых партнёров для ведения бизнеса. Партнёрами могут выступать поставщики технического оборудования и, собственно, представители от конкретной организации.

					ПГУ1.090301.15.001 ПЗ	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

Поставщиками технического оборудования могут стать компании, которые предоставляют компьютеры, серверы и другое необходимое оборудование для функционирования веб-сервиса. Выгода для партнёра заключается в получении денежных средств за предоставление качественного оборудования для разработки и разворачивания продукта, необходимого клиенту.

Представители и сотрудники конкретной организации станут основными клиентами продукта. Сотрудничество с ними позволит предоставить им удобное и эффективное решение для улучшения качества соответствующего определённой области графического контента. Выгода для партнёра заключается в получении продукта, который поможет им улучшить и упростить рабочие процессы, оптимизировать бизнес-процессы и сэкономить время, финансы и ресурсы.

3.2 Ключевые процессы

Ниже перечислены основные ключевые процессы при разработке стартап-проекта:

- создание ИП;
- поиск и создание команды;
- поиск средств;
- разработка заготовки ПО;
- поиск каналов сбыта и взаимодействие с целью продажи;
- модификация заготовки ПО под конкретные требования;
- поддержка созданного продукта.

Для начала процедуры открытия ИП нужно подготовить необходимые документы и данные. В первую очередь нужно найти ОКВЭДы для своего будущего бизнеса. Для этого нужно воспользоваться специальным справочником и подобрать наиболее подходящий код. В данном случае это будет – 62.01 «Разработка компьютерного программного обеспечения», а также 62.02 «Консультирование по компьютерным вопросам и связанная деятельность».

Вторым этапом станет выбор налогового режима. Выбор был сделан в пользу УСН, потому что этот вид предоставляет простоту в расчете и уплате налогов, а также предлагает низкие ставки налога. Можно самостоятельно рассчитывать и платить налоги раз в квартал, а ежегодно подавать декларацию. Доход будет облагаться налогом по ставке от 6% до 15%, в зависимости от региона. Регион размещения – Пензенская область, в нем установлена ставка 6%. Далее заполнить и подать документы, чтобы не оплачивать госпошлину принято решение подать их через МФЦ. Следующим этапом нужно получить справку из ЕГРИП, она придет на электронную почту, указанную в заявлении – обычно на это уходит 3 дня.

Для разработки такой системы нужно правильно подобрать команду, на начальном этапе в нее должны входить: архитектор, ведущие разработчики, разработчики поддержки, бухгалтер, менеджер по продажам. В нашем случае архитектор и ведущие разработчики найдены из числа единомышленников. Заниматься делами бухгалтерского учёта и менеджмента по продажам предложено бухгалтерам и менеджерам с бирж труда «hh.ru», а также «Profi.ru».

Поиск каналов сбыта осуществляется с помощью коммерческих предложений. Коммерческое предложение имеет несколько значимых достоинств, которые помогут привлечь покупателей и отличить этот проект от конкурентов.

Во-первых, аналоги подобного продукта в данный момент времени либо не доступны на территории Российской Федерации, либо не имеют предметного подхода при продаже собственного программного обеспечения.

Во-вторых, модификации и изменения индивидуального продукта могут быть выполнены без больших накладных расходов и затрат по времени.

В-третьих, для работы с предоставленным программным обеспечением не требуется установка специализированного клиента. Вся работа может производиться через приложение браузера на стандартных офисных машинах.

					ПГУ1.090301.15.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

Можно отметить, что использование передовых и популярных технологий, таких как нейросети, может повысить имидж компании и показать ее инновационный подход к бизнесу. Клиенты могут воспринимать компанию как современную, внимательную к новым тенденциям и готовую предложить им удобные и современные способы взаимодействия.

На российском рынке отсутствуют аналогичные программные продукты с таким же набором функций в оболочке веб-приложения. Набор функций предоставляет различные обработчики графического контента, основанные на различных нейросетевых моделях. В веб-интерфейсе пользователь может создать собственную учётную запись, что позволяет ему иметь в своём распоряжении историю ранее обработанных им файлов, а также создавать собственные заготовки настроек для повторного использования. Простота и понятность графического интерфейса позволяет использовать приложение без обращения к дополнительным источникам. Это позволяет занять уникальную позицию и быть востребованным проектом в данном сегменте рынка. Ниже в таблице 5 приведены услуги, предложенные в коммерческом предложении.

Таблица 5 – Наименование предложенных услуг

Наименование	Цена реализации
Разработка под организацию и внедрение	400 000р (обговаривается с заказчиком)
Поддержка	10 000р/мес (обговаривается с заказчиком)
Модификация существующего продукта	80 000р (обговаривается с заказчиком)

3.3 Каналы взаимодействия

Одним из способов привлечения клиентов станет веб-сервис, предоставляющий возможность ограниченно опробовать функционал стандартного продукта. Также веб-сервис предоставляет клиентам доступ к информации о компании и услугах в любое время. Потенциальные заказчики

смогут подробно ознакомиться ограниченным функционалом системы. Контактные формы и возможность обратной связи облегчат коммуникацию с будущими клиентами.

Для более эффективной демонстрации функционала и ценности веб-приложения нужно путем обзвона согласовать личные встречи с представителями коммерческих организаций. На таких встречах можно показать полный спектр возможностей продукта и ответить на вопросы потенциальных клиентов.

После заключения сделки, наша команда будет отвечать за поставку и внедрение индивидуального программного комплекса. Это включает в себя организацию отдельного серверного пространства, установку и настройку программного комплекса, а также последующую проверку доступа к обозначенному программному обеспечению, которое соответствует потребностям и предпочтениям клиента. Четкое планирование и организация этапов внедрения помогут сэкономить время и обеспечить плавный переход к полноценному использованию системы. Также нужно обеспечить поддержку клиентов в течение нескольких месяцев после покупки продукта. Это включает в себя решение технических проблем, обновление и доработку системы в соответствии с потребностями клиента, а также консультации и обучение пользователей использованию системы. Быстрый и качественный ответ на запросы клиентов поможет создать положительное впечатление от сотрудничества. Чтобы поддерживать долгосрочные отношения с клиентами, нужно стремиться регулярно предоставлять обновления и новости о продукте через электронные рассылки.

3.4 Источники дохода

Продукт будет иметь следующие источники дохода:

					ПГУ1.090301.15.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

– продажа и внедрение. Команда будет получать доход от разработки и внедрения программного обеспечения для каждой коммерческой организации. Стоимость описана в таблице 5;

– поддержка. Необходимо предоставлять услуги поддержки клиентам в течение нескольких месяцев после внедрения системы. Плата за поддержку составит около 10 000 рублей для каждой организации ежемесячно;

– модификация. Предусмотрена возможность модификаций индивидуальной системы. Предварительная цена модификации обговаривается с заказчиком.

Исходя из концепции распространения нашего продукта, нельзя точно посчитать количество всей полученной прибыли в перспективе. Если рассчитывать доход для конкретной выбранной сферы, например, медицины, то мы получим следующие результаты.

Согласно источнику 2ГИС, количество медицинских учреждений в городе Пенза составляет 1072. Если предположить, что около 30% из них готовы приобрести программный комплекс, то можно предположить, что примерно 321 учреждение может стать будущим клиентом.

Доход высчитывается по формуле 1.

$$Д = К * Ц \quad (1)$$

где Д – доход;

К – количество потребителей;

Ц – установленная цена;

Доход от разработки и внедрения будет высчитываться по формуле 1.

$$321 \text{ пр} * 400\,000 \text{ р} = 128\,400\,000 \text{ р}$$

Доход от поддержки будет высчитываться по формуле 1.

$$10\,000 \text{ р} * 321 \text{ пр} = 3\,210\,000 \text{ р/мес}$$

Таким образом, общий ожидаемый доход в сфере медицины в г. Пенза составит 128 400 000 рублей единовременно и 3 210 000 рублей ежемесячно.

					ПГУ1.090301.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		76

Для наглядности, в таблице 6 представлена прогнозируемая программа производства по месяцам для коммерческих образовательных учреждений по г. Пенза.

Таблица 6 – Прогнозируемая программа производства

Наименование услуги	1 мес.	2 мес.	3 мес.	4 мес.	5 мес.
Разработка и внедрение	8	6	10	9	11
Поддержка	8	14	24	33	44
Модификация	2	5	4	8	7

3.5 Ключевые ресурсы

Для запуска, функционирования и развития бизнеса потребуются некоторые ключевые ресурсы. В нашем случае это будут:

- архитектор и разработчики программного обеспечения, специализирующиеся на поддержке Backend- и Frontend-разработке, а также администрировании ОС Linux;
- менеджеры по продажам, обладающие навыками продажи и умением вести переговоры с потенциальными клиентами;
- компьютеры, серверы, программное обеспечение и другое техническое оборудование для разработки и обслуживания продуктов;
- ресурсы для финансирования разработки, маркетинга, продаж и операционных расходов бизнеса.

Финансирование проекта планируется осуществить как за счёт собственных средств инициатора проекта, так и за счет государственной поддержки «Гранты для молодых предпринимателей».

Ниже, в таблице 7 можно увидеть программу финансирования.

Таблица 7 – Источники финансирования

Источник финансирования	Сумма
Собственные средства	300 000р
Венчурное финансирование	400 000р
Гос. поддержка	500 000р
Всего	1 200 000р

Приведённые выше ключевые ресурсы можно разделить на финансовые и людские. Подробно направление использования средств указано на таблицах 8 и 9.

Таблица 8 – Финансовые ключевые ресурсы

Направление использования средств	Кол-во, шт	Цена, руб	Сумма, руб
Сборка сервера для разворачивания программного комплекса	2	120 000	240 000
Аренда помещения для серверов (6 мес)	1	25 000	25 000
Расходы на электроэнергию (6 мес)	1	25 000	25 000
Закупка другого оборудования	1	50 000	50 000

Таблица 9 – Людские ключевые ресурсы

Направление использования средств	Кол-во, шт	Цена, руб.	Сумма, руб
З/п архитектору (6 мес)	6	40 000	240 000
З/п разработчику (6 мес)	12	30 000	360 000

З/п бухгалтеру (6 мес)	6	20 000	120 000
З/п менеджеру (6 мес)	6	20 000	120 000

3.6 Риски проекта

Успешная реализация стартап-проекта сопряжена с различными рисками и форс-мажорными обстоятельствами, которые могут негативно повлиять на его исход. Некоторые из возможных рисков включают недостаточный спрос на предлагаемый продукт, изменения в законодательстве, регулирующем работу искусственного интеллекта, проблемы с финансированием проекта, технические сложности при разработке и внедрении системы, а также нестабильность экономической ситуации. Эти факторы могут создать значительные препятствия для реализации стартап-проекта и его дальнейшего успешного развития.

Успех стартапа во многом зависит от состава и компетенций команды. Важно сформировать сплоченную и высокопрофессиональную команду, обладающую необходимыми навыками и опытом.

Необходимо тщательно изучить и соблюдать все применимые законы, нормативы и требования, чтобы избежать правовых рисков.

Важно разработать эффективную маркетинговую стратегию, чтобы обеспечить осведомленность целевой аудитории о нашем продукте или услуге.

Форс-мажорные обстоятельства, такие как природные катастрофы, пандемии, также могут оказать существенное влияние на реализацию стартап-проекта.

Важно провести анализ и предусмотреть меры по минимизации возможных негативных последствий таких обстоятельств.

Заключение

Разработка конвейера обработки видеофайлов для веб-приложения была успешно завершена. Реализованы все запрашиваемые элементы, описанные в функциональных требованиях к конвейеру (пункт 1.4). Удачное тестирование программного комплекса также подтверждает работоспособность и состоятельность данного продукта. В процессе написания данной работы также было изучено множество механизмов оптимизации высоконагруженных приложений, изучено API библиотеки ffmpeg и API определённого пула нейронных сетей, что положительно сказалось на развитии навыков программирования.

Дальнейшее развитие данного проекта планируется в рамках стартапа. Материальные средства позволят использовать больше аппаратных ресурсов для разворачивания и масштабирования конвейера. Продолжение модернизации способствует увеличению коэффициента соотношения качества преобразования к временным затратам на одну выполняемую задачу. В свою очередь, это позволит предоставить разработанный продукт массовому пользователю через организацию полноценного веб-сервиса и поспособствует популяризации инструмента для простого и понятного разворачивания нейронных сетей.

Использование нейронных сетей должно быть доступно массовому пользователю без больших затрат времени и других ресурсов. Это позволит ускорить прогресс развития науки и человека, соответственно.

					ПГУ1.090301.15.001 ПЗ	Лист
						80
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Фостер Дэвид Ф81 Генеративное глубокое обучение. Творческий потенциал нейронных сетей. — СПб.: Питер, 2020. — 336 с.: ил. — (Серия «Бестселлеры O'Reilly»).
2. Шоттс У. Командная строка Linux. Полное руководство. 2-е межд. изд. — СПб.: Питер, 2020. — 544 с.: ил. — (Серия «Для профессионалов»).
3. Шрёдер Карла Linux. Книга рецептов. 2-е изд. — СПб.: Питер, 2022. — 592 с.: ил. — (Серия «Бестселлеры O'Reilly»).
4. Библия Linux. 10-е издание. / Негус Кристофер; перевод с английского языка Павлов А. — СПб.: Питер, 2022. — 928 с.: ил. — (Серия «Для профессионалов»).
5. Ньюмен С., Создание микросервисов. — СПб.: Питер, 2016. — 304 с.: ил. — (Серия «Бестселлеры O'Reilly»).
6. Ньюмен С., От монолита к микросервисам: Пер. с англ. — СПб.: БХВ-Петербург, 2021. — 272 с.: ил.
7. Мейерс, Скотт. М45 Эффективный и современный C++: 42 рекомендации по использованию C++ 11 и C++ 14.: Пер. с англ. - М. : ООО "ИЛ. Вильяме", 2016. - 304 с.: ил. – Пар.
8. Официальная документация Ffmpeg [Электронный ресурс] / Режим доступа <https://ffmpeg.org/documentation.html>, свободный (дата обращения 15.03.2024).

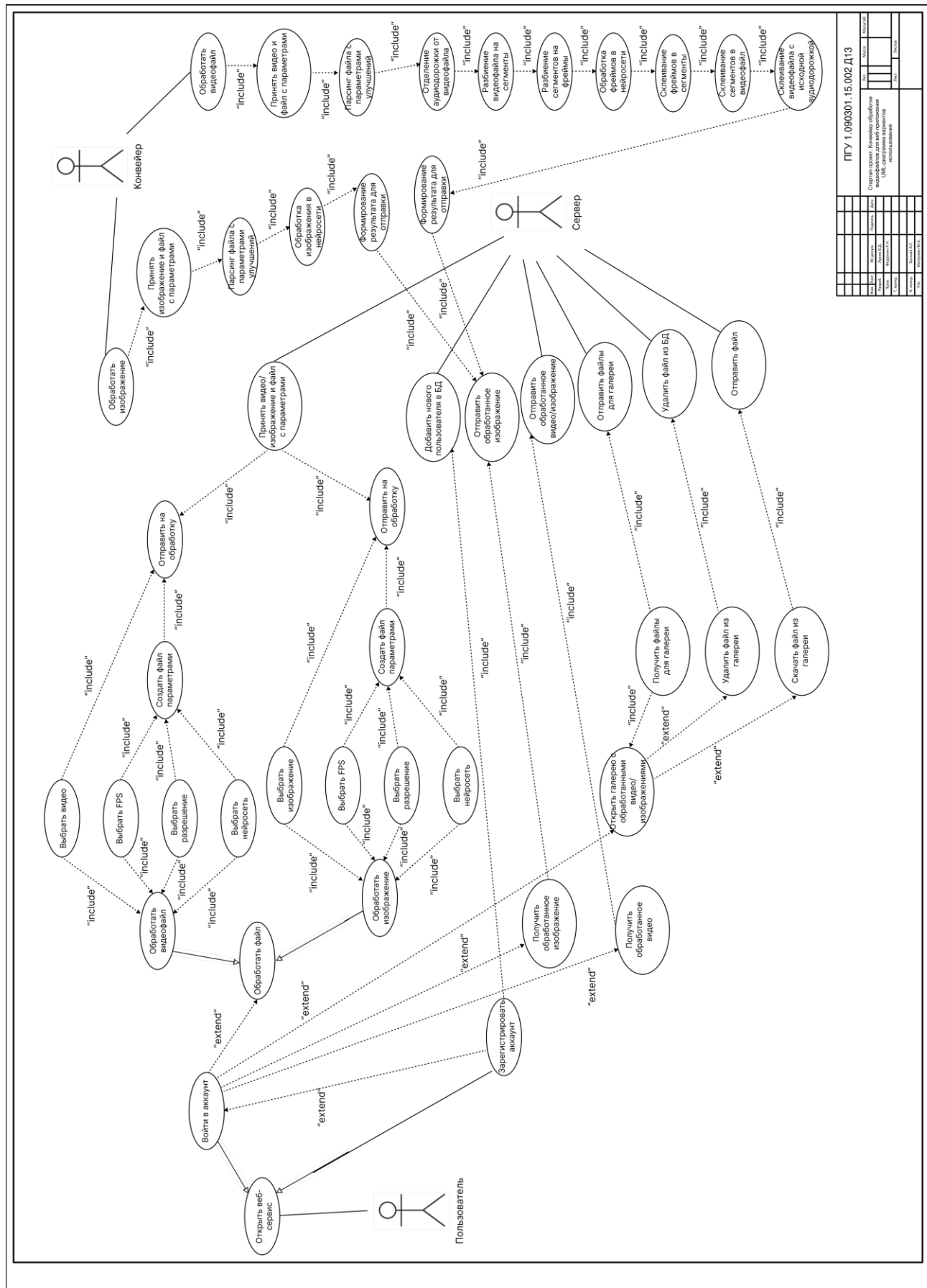
Приложение А

(Обязательное)

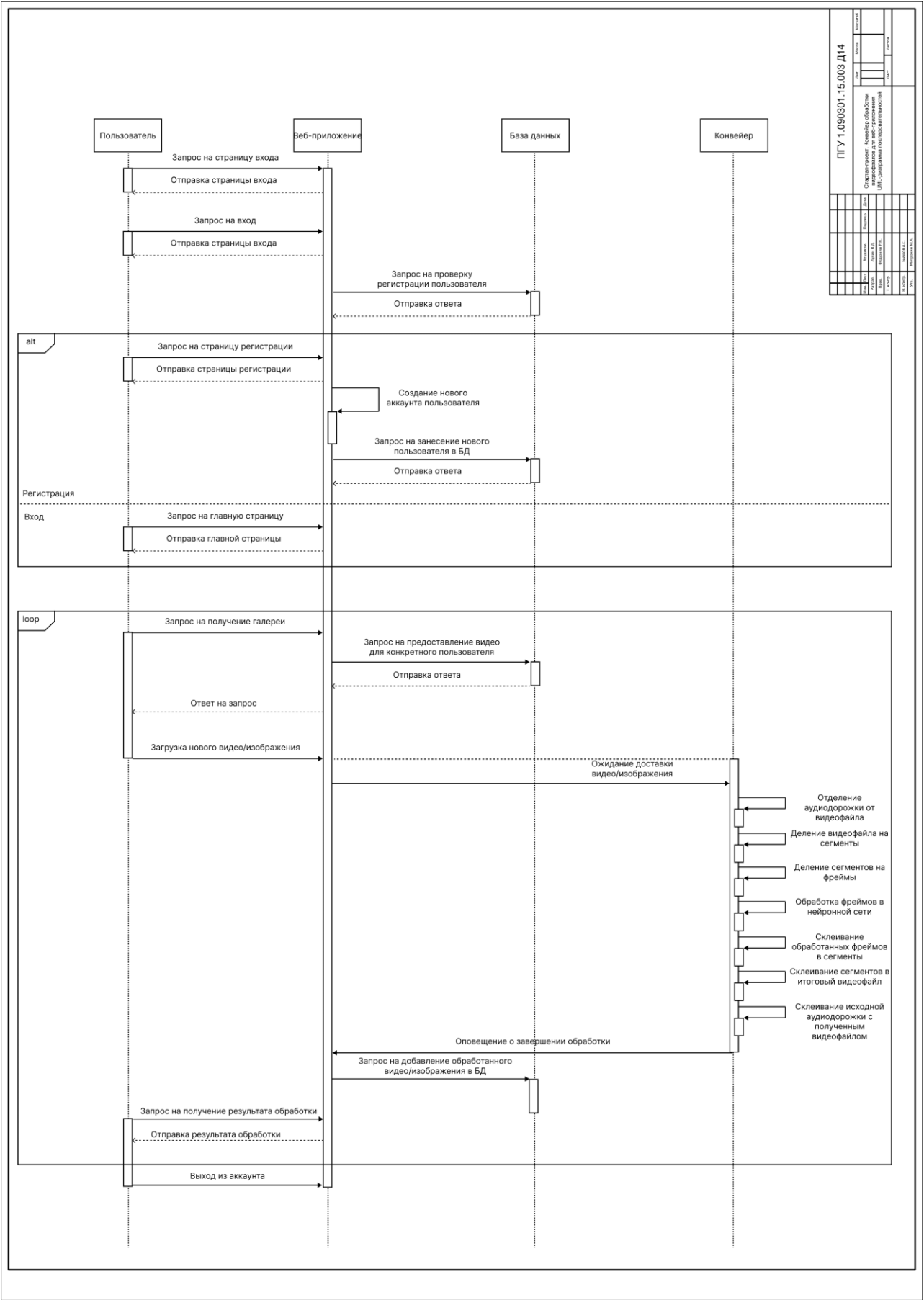
Стартап-проект. Конвейер обработки видеофайлов для веб-приложения
UML-диаграммы

					ПГУ1.090301.15.001 ПЗ	Лист
						82
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 – UML-диаграмма вариантов использования



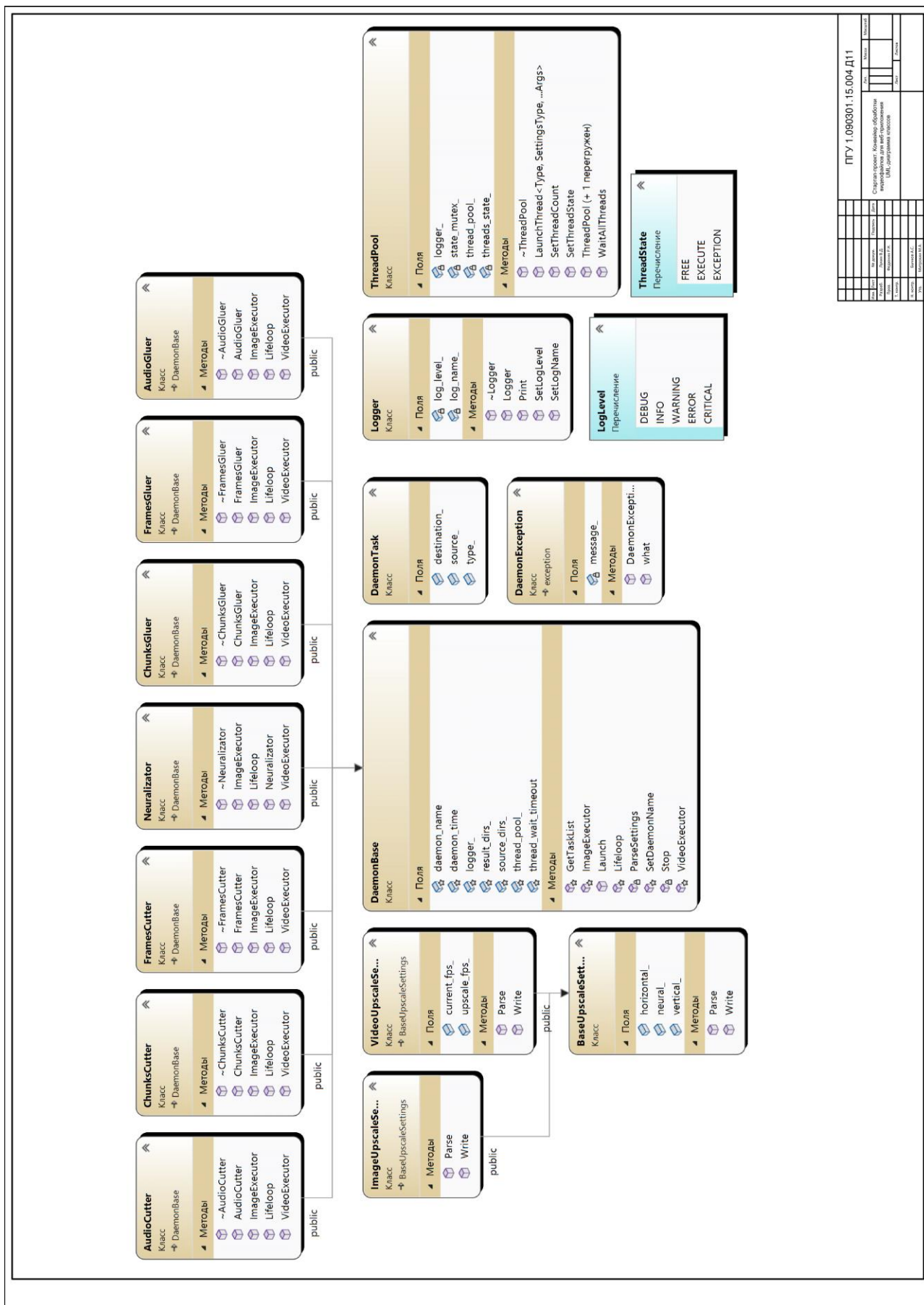
Приложение А.2 – UML-диаграмма последовательностей



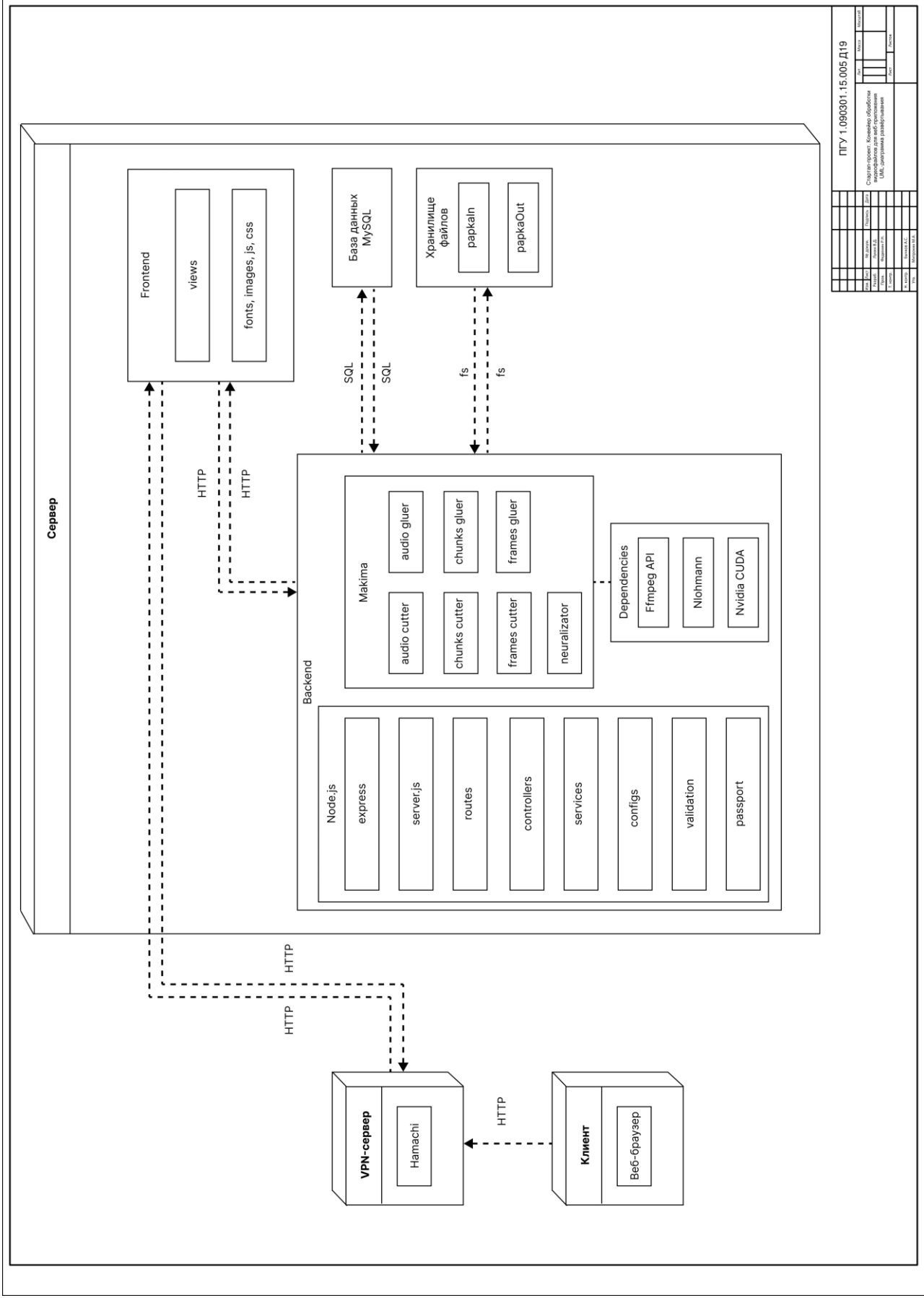
ПГУ 1.090301.15.003 Д14									
Старший проект: Конвейер обработки видеопотока для веб-приложения (UML-диаграмма последовательностей)									
Имя	Имя	Имя	Имя	Имя	Имя	Имя	Имя	Имя	Имя
В.А.	В.А.	В.А.	В.А.	В.А.	В.А.	В.А.	В.А.	В.А.	В.А.
И.И.	И.И.	И.И.	И.И.	И.И.	И.И.	И.И.	И.И.	И.И.	И.И.
С.С.	С.С.	С.С.	С.С.	С.С.	С.С.	С.С.	С.С.	С.С.	С.С.
М.М.	М.М.	М.М.	М.М.	М.М.	М.М.	М.М.	М.М.	М.М.	М.М.
Д.Д.	Д.Д.	Д.Д.	Д.Д.	Д.Д.	Д.Д.	Д.Д.	Д.Д.	Д.Д.	Д.Д.
В.В.	В.В.	В.В.	В.В.	В.В.	В.В.	В.В.	В.В.	В.В.	В.В.
А.А.	А.А.	А.А.	А.А.	А.А.	А.А.	А.А.	А.А.	А.А.	А.А.
К.К.	К.К.	К.К.	К.К.	К.К.	К.К.	К.К.	К.К.	К.К.	К.К.
Н.Н.	Н.Н.	Н.Н.	Н.Н.	Н.Н.	Н.Н.	Н.Н.	Н.Н.	Н.Н.	Н.Н.
Л.Л.	Л.Л.	Л.Л.	Л.Л.	Л.Л.	Л.Л.	Л.Л.	Л.Л.	Л.Л.	Л.Л.
З.З.	З.З.	З.З.	З.З.	З.З.	З.З.	З.З.	З.З.	З.З.	З.З.
С.С.	С.С.	С.С.	С.С.	С.С.	С.С.	С.С.	С.С.	С.С.	С.С.
М.М.	М.М.	М.М.	М.М.	М.М.	М.М.	М.М.	М.М.	М.М.	М.М.
И.И.	И.И.	И.И.	И.И.	И.И.	И.И.	И.И.	И.И.	И.И.	И.И.
О.О.	О.О.	О.О.	О.О.	О.О.	О.О.	О.О.	О.О.	О.О.	О.О.
Ф.Ф.	Ф.Ф.	Ф.Ф.	Ф.Ф.	Ф.Ф.	Ф.Ф.	Ф.Ф.	Ф.Ф.	Ф.Ф.	Ф.Ф.
Х.Х.	Х.Х.	Х.Х.	Х.Х.	Х.Х.	Х.Х.	Х.Х.	Х.Х.	Х.Х.	Х.Х.
Ц.Ц.	Ц.Ц.	Ц.Ц.	Ц.Ц.	Ц.Ц.	Ц.Ц.	Ц.Ц.	Ц.Ц.	Ц.Ц.	Ц.Ц.
Ч.Ч.	Ч.Ч.	Ч.Ч.	Ч.Ч.	Ч.Ч.	Ч.Ч.	Ч.Ч.	Ч.Ч.	Ч.Ч.	Ч.Ч.
Ш.Ш.	Ш.Ш.	Ш.Ш.	Ш.Ш.	Ш.Ш.	Ш.Ш.	Ш.Ш.	Ш.Ш.	Ш.Ш.	Ш.Ш.
Щ.Щ.	Щ.Щ.	Щ.Щ.	Щ.Щ.	Щ.Щ.	Щ.Щ.	Щ.Щ.	Щ.Щ.	Щ.Щ.	Щ.Щ.
Ъ.Ъ.	Ъ.Ъ.	Ъ.Ъ.	Ъ.Ъ.	Ъ.Ъ.	Ъ.Ъ.	Ъ.Ъ.	Ъ.Ъ.	Ъ.Ъ.	Ъ.Ъ.
Ы.Ы.	Ы.Ы.	Ы.Ы.	Ы.Ы.	Ы.Ы.	Ы.Ы.	Ы.Ы.	Ы.Ы.	Ы.Ы.	Ы.Ы.
Э.Э.	Э.Э.	Э.Э.	Э.Э.	Э.Э.	Э.Э.	Э.Э.	Э.Э.	Э.Э.	Э.Э.
Ю.Ю.	Ю.Ю.	Ю.Ю.	Ю.Ю.	Ю.Ю.	Ю.Ю.	Ю.Ю.	Ю.Ю.	Ю.Ю.	Ю.Ю.
Я.Я.	Я.Я.	Я.Я.	Я.Я.	Я.Я.	Я.Я.	Я.Я.	Я.Я.	Я.Я.	Я.Я.

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.3 – UML-диаграмма классов

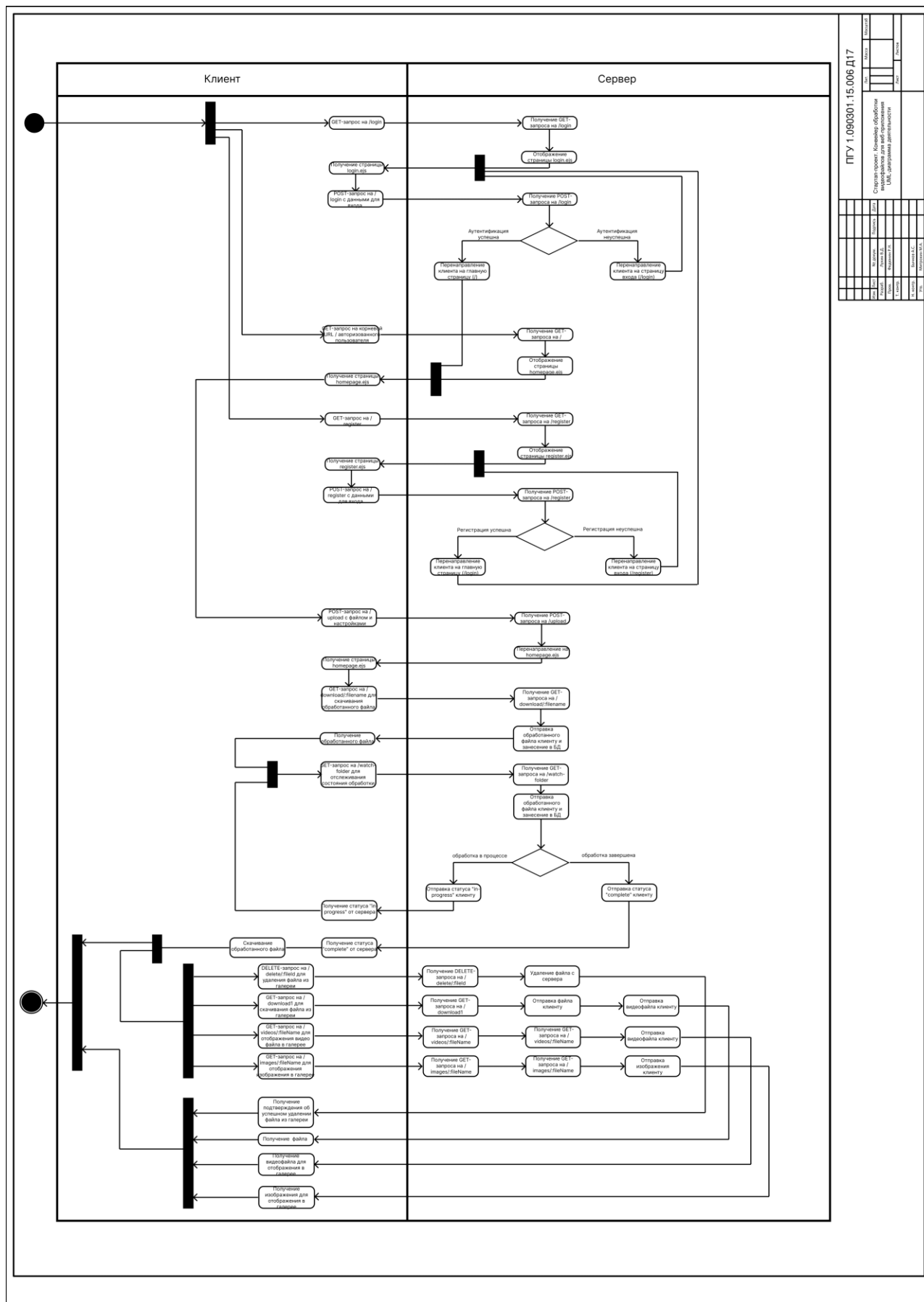


Приложение А.4 – UML-диаграмма развёртывания



ПГУ 1.090301.15.005 Д19									
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель

Приложение А.5 – UML-диаграмма деятельности



Приложение Б

(Обязательное)

Стартап-проект. Конвейер обработки видеофайлов для веб-приложения.

Листинг программы

Файл `daemon_interface.hpp`

```
#pragma once

// Подключение библиотек C++.
#include <iostream>
#include <filesystem>
#include <chrono>
#include <string>
#include <vector>
#include <fstream>
#include <cstdio>

// Подключение библиотек ffmpeg (С-формат).
extern "C"
{
#include <libavutil/imgutils.h>
#include <libavformat/avformat.h>
#include <libavcodec/avcodec.h>
#include <libswscale/swscale.h>
#include <libavutil/opt.h>
}

// Подключение логера и пула потоков.
#include "logger.hpp"
#include "thread_pool.hpp"

// Подключение библиотеки nlohmann-json.
#include "json.hpp"

// Подключение общих элементов.
#include "shares.hpp"

// Наименование файла ошибки по умолчанию.
const std::string error_file_name = "makima.error";
// Наименование файла разрешения по умолчанию.
const std::string access_file_name = "makima.access";
// Наименование файла с параметрами улучшения по умолчанию.
const std::string upscale_file_name = "upscale_settings.json";
// Наименование файла с параметрами демона по умолчанию.
const std::string daemon_settings_name = "daemon_settings.json";
// Наименование файла со списком чанков.
const std::string chunks_list_name = "concat_chunks.txt";
```

					ПГУ1.090301.15.001 ПЗ	Лист
						88
Изм.	Лист	№ докум.	Подпись	Дата		

```

// Формат обрабатываемых видео по умолчанию.
const std::string base_video_format = ".mp4";
// Формат обрабатываемых аудиодорожек по умолчанию.
const std::string base_audio_format = ".aac";
// Формат обрабатываемых фреймов по умолчанию.
const std::string base_frames_format = ".png";
// Вектор доступных нейронных сетей.
const std::vector<std::string> neurals{"realesrgan",
"realcugan", "waifu2x"};
// Коэффициент преобразования для нейронной сети.
const int neural_scale = 2;

/// @brief Найти все вложенные директории.
/// @param[in] path Расположение для поиска.
/// @return Список вложенных директорий.
inline std::vector<std::string> FindSubDirictories(const
std::filesystem::path& path)
{
    // Список директорий в расположении.
    std::vector<std::string> dirictories{};
    // Итератор для поиска директорий.
    std::filesystem::directory_iterator dir_iterator(path);
    while(dir_iterator != std::filesystem::end(dir_iterator))
    {
        // Если найдена директория - помещаем в список.
        if(std::filesystem::is_directory(dir_iterator-
>status()))
        {
            dirictories.push_back(dir_iterator-
>path().string());
        }
        dir_iterator++;
    }

    return dirictories;
}

/// @brief Найти все файлы с определённым расширением.
/// @param[in] path Расположение для поиска.
/// @param[in] extension Расширение для поиска.
/// @return Список найденных файлов.
inline std::vector<std::string> FindExtensionFiles(const
std::filesystem::path& path, const std::string& extension)
{
    // Список директорий в расположении.
    std::vector<std::string> files{};
    // Итератор для поиска директорий.
    std::filesystem::directory_iterator file_iterator(path);
    while(file_iterator != std::filesystem::end(file_iterator))
    {
        // Если найдена директория - помещаем в список.

```

```

        if(file_iterator->path().extension() == extension)
        {
            files.push_back(file_iterator->path().string());
        }
        file_iterator++;
    }

    return files;
}

/// @brief Проверить готовность файла к обработке. Проверка
/// осуществляется путём периодической проверки размера.
/// @param[in] file_path Расположение файла для проверки.
/// @param[in] timeout Таймаут ожидания между проверками.
/// @param[in] block_mode Включение / отключение блокирующего
/// режима.
/// @return Результат проверки готовности.
inline bool CheckFilePreparedness(const std::filesystem::path&
file_path, const std::chrono::milliseconds timeout, bool
block_mode = true)
{
    // Количество попыток проверки подряд.
    int retries_count = 3;
    // Текущее количество попыток проверки подряд.
    int current_retries = 0;
    // Размер файла на предыдущей итерации.
    int old_file_size = 0;
    // Текущий размер файла.
    int new_file_size = 0;

    // Если функция запущена в неблокирующем режиме.
    if(block_mode != true)
    {
        // Первое получение размера файла.
        old_file_size = std::filesystem::file_size(file_path);

        std::this_thread::sleep_for(timeout);

        // Второе получение размера файла.
        new_file_size = std::filesystem::file_size(file_path);

        // Сопоставление размеров.
        if(old_file_size == new_file_size)
        {
            return true;
        }

        return false;
    }

    // Проводим проверку до готовности файла.
    while(current_retries < retries_count)

```

```

{
    // Получение размера файла.
    new_file_size = std::filesystem::file_size(file_path);

    // Равенство размеров - успешная проверка. Иначе -
    сбрасываем счётчик проверки.
    if(old_file_size == new_file_size)
    {
        current_retries++;
    }
    else
    {
        current_retries = 0;
    }

    // Сохранение размера файла.
    old_file_size = new_file_size;
    std::this_thread::sleep_for(timeout);
}

return true;
};

/// @brief Исключения, отправляемые демоном.
class DaemonException : public std::exception
{
public:
    DaemonException(const std::string message) :
    message_(message) {};

    const char* what() const throw()
    {
        return message_.c_str();
    }

private:
    // Сообщение исключения.
    std::string message_{};
};

/// @brief Заготовка функций демона.
class DaemonBase
{
public:
    /// @brief Запуск демона.
    int Launch()
    {
        try
        {
            // Парсинг настроек демона.
            ParseSettings();
        }
    }
};

```

```

        logger_.Print(LogLevel::INFO, "Демон
инициализирован");
        logger_.Print(LogLevel::INFO, "Запуск
демона...");

        // Проверить существование расположение ресурсов
для обработки.

if(!std::filesystem::exists(std::filesystem::path(source_dir_)))
    throw DaemonException("Расположение " +
source_dir_ + " не существует!");

        // Проверить существование директории
результатов.

if(!std::filesystem::exists(std::filesystem::path(result_dir_)))
    throw DaemonException("Расположение " +
result_dir_ + " не существует!");

        logger_.Print(LogLevel::INFO, "Демон запущен.");
Lifeloop();
        logger_.Print(LogLevel::INFO, "Демон
остановлен.");
        Stop();
        return 0;
    }
    catch(const std::exception &ex)
    {
        logger_.Print(LogLevel::CRITICAL, ex.what());
        Stop();
        return -1;
    }
}

protected:
    /// @brief Парсинг параметров улучшения. Не проверяет
существование!
    /// @param[in] upscale_path Расположение файла
параметров улучшения.
    /// @param[out] settings Буфер для записи параметров.
    /// @return < 0 при ошибке, 0 при успехе.
    int ParseUpscale(const std::string upscale_path,
UpscaleSettings& settings)
    {
        try
        {
            // Открыть файл настроек.
            std::ifstream
settings_file(upscale_path.c_str());
            // Сохранить результаты парсинга.
            nlohmann::json data =
nlohmann::json::parse(settings_file);

```

```

        // Проверка поля FPS.
        if(data["FPS"] > 60)
        {
            logger_.Print(LogLevel::ERROR, "Некорректное
поле FPS (" + std::string(data["FPS"]) +") в" + upscale_path);
            return -1;
        }

        // Проверка поля Neural.
        bool checkout = false;
        for(auto &neural : neurals)
        {
            if(neural == std::string(data["Neural"]))
            {
                checkout = true;
                break;
            }
        }

        if(!checkout)
        {
            logger_.Print(LogLevel::ERROR, "Некорректное
поле Neural (" + std::string(data["Neural"]) +") в" +
upscale_path);

            return -1;
        }

        // Проверка поля Horizontal.
        if(data["Horizontal"] > 4096)
        {
            logger_.Print(LogLevel::ERROR, "Некорректное
поле Horizontal (" + std::string(data["Horizontal"]) +") в" +
upscale_path);

            return -1;
        }

        // Проверка поля Vertical.
        if(data["Vertical"] > 3072)
        {
            logger_.Print(LogLevel::ERROR, "Некорректное
поле Vertical (" + std::string(data["Vertical"]) +") в" +
upscale_path);

            return -1;
        }

        settings.fps_ = data["FPS"];
        settings.neural_ = data["Neural"];
        settings.horizontal_ = data["Horizontal"];
        settings.vertical_ = data["Vertical"];
    }
    catch(const std::exception &ex)

```

```

        {
            logger_.Print(LogLevel::ERROR, "При парсинге
параметров улучшения " + upscale_path + " возникло исключение: "
+ std::string(ex.what()));
            return -1;
        }

        return 0;
    }

    /// @brief Цикл работы демона. Вызывается в Start().
    virtual void Lifeloop() = 0;

    /// @brief Обработчик найденных ресурсов. Вызывается в
    Lifecycle().
    /// @param[in] thread_num Номер используемого потока.
    /// @param[in] settings Параметры улучшения качества.
    /// @param[in] source_file Расположение обрабатываемого
    ресурса.
    /// @param[in] result_file Расположение результата
    обработки.
    virtual void Executor(int thread_num, const
    UpscaleSettings& settings, std::string source_file, const
    std::string result_file) = 0;

    // Логер.
    Logger logger_{};
    // Пул потоков.
    ThreadPool thread_pool_{&logger_};
    // Время работы демона (в секундах).
    int daemon_time{};
    // Время ожидания завершения потоков демона при
    остановке.
    int thread_wait_timeout{};
    // Расположение ресурсов для обработки.
    std::string source_dir_{};
    // Расположение результатов обработки.
    std::string result_dir_{};

private:
    /// @brief Парсинг настроек демона.
    /// @exception DaemonException - при внутренних ошибках.
    void ParseSettings()
    {
        try
        {
            // Хранилище для результатов парсинга.
            nlohmann::json data{};

            // Если файл настроек не найден - создать с
            пустыми параметрами.

```

```

if(!std::filesystem::exists(std::filesystem::path(daemon_setting
s_name)))
{
    // Установка параметров по умолчанию.
    data["Source"] = "empty";
    data["Destination"] = "empty";
    data["Execute_Time"] = 180;
    data["Wait_Timeout"] = 20;

    // Запись установленных данных.
    std::ofstream
new_settings(daemon_settings_name);
    new_settings << data;
}

// Считать данные настроек.
std::ifstream
old_settings(daemon_settings_name);
data = nlohmann::json::parse(old_settings);

// Проверка поля Source.
if(data["Source"] == "empty" || data["Source"]
== "")
    throw DaemonException("Некорректное поле
Source (" + std::string(data["Source"]) + ") в " +
daemon_settings_name);

// Проверка поля Destination.
if(data["Destination"] == "empty" ||
data["Destination"] == "")
    throw DaemonException("Некорректное поле
Destination (" + std::string(data["Destination"]) + ") в " +
daemon_settings_name);

    source_dir_ = data["Source"];
    result_dir_ = data["Destination"];
    daemon_time = data["Execute_Time"];
    thread_wait_timeout = data["Wait_Timeout"];
}
catch(const DaemonException &de)
{
    throw de;
}
catch(const std::exception &ex)
{
    throw DaemonException("При парсинге параметров
демона возникло исключение: " + std::string(ex.what()));
}
}

/// @brief Остановка демона.

```

```

        void Stop()
        {
            logger_.Print(LogLevel::INFO, "Остановка
демона...");
            thread_pool_.WaitAllThreads(thread_wait_timeout);
            logger_.Print(LogLevel::INFO, "Демон остановлен");
        }
};

/// @brief Отделяет аудиопоток от видеофайлов.
class AudioCutter : public DaemonBase
{
public:
    AudioCutter()
    {
        logger_.SetLogName("audio_cutter");
        thread_pool_.SetThreadCount(5);
    }
    ~AudioCutter()
    {
        logger_.Print(LogLevel::INFO, "Демон
деинициализирован");
    }

    void Lifeloop()
    {
        // Временные засечки для контроля работы демона.
        auto launch_time = std::chrono::system_clock::now();
        auto rubicon = std::chrono::system_clock::now();

        // Тело демона. Пока не превысили время работы -
работаем.

        while(std::chrono::duration_cast<std::chrono::seconds>(rubicon -
launch_time).count() < daemon_time)
        {
            // Найти все вложенные директории
(пользователи).
            auto user_dirs =
FindSubDirictories(source_dir_);
            for(auto& user_dir : user_dirs)
            {
                // Найти все вложенные директории (задачи).
                auto task_dirs =
FindSubDirictories(user_dir);
                for(auto& task_dir : task_dirs)
                {
                    // Расположение файла ошибки.
                    std::string error_file_path = task_dir +
"/" + error_file_name;
                    // Расположение файла параметров
улучшения.

```

```

        std::string upscale_file_path = task_dir
+ "/" + upscale_file_name;

        // Если в каталоге присутствует файл
ошибки - пропускаем каталог.

if(std::filesystem::exists(error_file_path))
    continue;

        // Если в каталоге отсутствует файл
параметров улучшения - пропускаем каталог.

if(!std::filesystem::exists(upscale_file_path))
    continue;

        // Проверить готовность файла параметров
улучшения.

if(!CheckFilePreparedness(upscale_file_path, 150ms, false))
    continue;

        // Если парсинг параметров не удался -
создать файл ошибки.
UpscaleSettings upscale_settings{};
if(ParseUpscale(upscale_file_path,
upscale_settings) != 0)
{
    std::ofstream
error_file(error_file_path.c_str());
    error_file.close();
    continue;
}

        // Найти все вложенные директории
(видео).

auto video_files =
FindExtensionFiles(task_dir, base_video_format);

        // Отсутствуют файлы - ожидаем.
if(video_files.size() == 0)
    continue;

for(auto& video_file : video_files)
{
    // Наименование папки пользователя.
    std::string user_name =
std::filesystem::path(user_dir).stem();
    // Наименование папки с задачей.
    std::string task_name =
std::filesystem::path(task_dir).stem();

```

```

// Наименование обрабатываемого
ресурса.
std::string video_name =
std::filesystem::path(video_file).stem();
// Директория расположения
результата.
std::string result_dir_path =
result_dir_ + "/" + user_name + "/" + task_name + "/" +
video_name;
// Расположение результата.
std::string result_path =
result_dir_path + "/" + video_name + base_audio_format;
// Расположение файла блокировки.
std::string lock_path = source_dir_
+ "/" + user_name + "/" + task_name + "/" + video_name +
"_ac.lock";

// Если файл имеет файл блокировки -
не работаем.
if(std::filesystem::exists(lock_path))
    continue;

// Проверить готовность файла для
обработки.
if(!CheckFilePreparedness(upscale_file_path, 150ms, false))
    continue;

// Если не папка для результата не
существует - создать.
if(!std::filesystem::exists(result_dir_path))
{
if(!std::filesystem::create_directories(result_dir_path))
{
logger_.Print(LogLevel::ERROR, "Не удалось создать директорию
для результата " + result_dir_path);
continue;
}

logger_.Print(LogLevel::INFO,
"Создана директория для результата " + result_dir_path);
}

// Если нет файла блокировки -
создаём файл блокировки.
std::ofstream
lock_file(lock_path.c_str());
lock_file.close();

```

```

        // Если не удалось выделить поток
        под выполнение - удаляем файл блокировки.

        if(!thread_pool_.LaunchThread<AudioCutter>(&AudioCutter::Executo
        r, this, upscale_settings, video_file, result_path))
        {
            std::remove(lock_path.c_str());

        std::this_thread::sleep_for(20ms);
        }
    }

    }

    rubicon = std::chrono::system_clock::now();
    std::this_thread::sleep_for(20ms);
}

    logger_.Print(LogLevel::INFO, "Время работы демона
    истекло (" + std::to_string(daemon_time) + " сек). Плановый
    перезапуск.");
}

    /// @brief Отделяет аудиопоток от видеофайла.
    /// @param[in] thread_num Номер используемого потока.
    /// @param[in] settings Параметры улучшения качества.
    /// @param[in] source_file Расположение видеофайла с
    расширением.
    /// @param[in] result_file Расположение создаваемой
    аудиодорожки с расширением.
    /// @exception DaemonException - если произошла ошибка
    при выполнении функции.
    void Executor(int thread_num, const UpscaleSettings&
    settings, std::string source_file, const std::string
    result_file)
    {
        try
        {
            auto time_start =
            std::chrono::system_clock::now();

            AVFormatContext *input_format_context = nullptr;
            if (avformat_open_input(&input_format_context,
            source_file.c_str(), nullptr, nullptr) != 0)
                throw DaemonException("Невозможно открыть
            исходник: " + source_file);

            if
            (avformat_find_stream_info(input_format_context, nullptr) < 0)

```

```

        throw DaemonException("Невозможно найти
информацию о потках источника: " + source_file);

        int audio_stream_index = -1;
        for (int i = 0; i < input_format_context-
>nb_streams; i++) {
            if (input_format_context->streams[i]-
>codecpar->codec_type == AVMEDIA_TYPE_AUDIO) {
                audio_stream_index = i;
                break;
            }
        }

        if (-1 == audio_stream_index)
            throw DaemonException("Невозможно найти
аудио поток источника: " + source_file);

        AVFormatContext *output_format_context =
nullptr;

        AVOutputFormat *output_format =
const_cast<AVOutputFormat*>(av_guess_format("aac", nullptr,
nullptr));

        if(avformat_alloc_output_context2(&output_format_context,
output_format, nullptr, result_file.c_str()) < 0)
            throw DaemonException("Невозможно создать
результат: " + result_file);

        AVStream *audio_stream =
avformat_new_stream(output_format_context, nullptr);
        if (!audio_stream)
            throw DaemonException("Невозможно создать
аудио поток для результата: " + result_file);

        if (avcodec_parameters_copy(audio_stream-
>codecpar, input_format_context->streams[audio_stream_index]-
>codecpar) < 0)
            throw DaemonException("Невозможно
скопировать параметры кодека из источника: " + source_file);

        if(avio_open(&output_format_context->pb,
result_file.c_str(), AVIO_FLAG_WRITE) < 0)
            throw DaemonException("Невозможно открыть
результат: " + result_file);

        if(avformat_write_header(output_format_context,
NULL) < 0)
            throw DaemonException("Невозможно записать
заголовок результата: " + result_file);

        AVPacket packet;

```

```

        while (av_read_frame(input_format_context,
&packet) >= 0) {
            if (packet.stream_index ==
audio_stream_index) {
                packet.stream_index = 0;

av_interleaved_write_frame(output_format_context, &packet);
            }
            av_packet_unref(&packet);
        }

        av_write_trailer(output_format_context);

        avformat_close_input(&input_format_context);
        avio_close(output_format_context->pb);

        avformat_free_context(output_format_context);

        auto time_end =
std::chrono::system_clock::now();
        logger_.Print(LogLevel::INFO, "Задача " +
source_file + " была выполнена за " +

std::to_string(std::chrono::duration_cast<std::chrono::seconds>(
time_end - time_start).count()) + " сек. / " +

std::to_string(std::chrono::duration_cast<std::chrono::milliseco
nds>(time_end - time_start).count()) + " миллисек.");
        thread_pool_.SetThreadState(thread_num,
ThreadState::FREE);
    }
    catch(const std::exception &ex)
    {
        logger_.Print(LogLevel::ERROR, "Поток " +
std::to_string(thread_num) + " завершен с исключением: " +
ex.what());
        thread_pool_.SetThreadState(thread_num,
ThreadState::EXCEPTION);
    }
}

};

class ChunksCutter : public DaemonBase
{
public:
    ChunksCutter()
    {
        logger_.SetLogName("chunks_cutter");
        thread_pool_.SetThreadCount(5);
    }
    ~ChunksCutter()
    {

```

```

        logger_.Print(LogLevel::INFO, "Демон
деинициализирован");
    }

    void Lifeloop()
    {
        // Временные засечки для контроля работы демона.
        auto launch_time = std::chrono::system_clock::now();
        auto rubicon = std::chrono::system_clock::now();

        // Тело демона. Пока не превысили время работы -
        работаем.

        while(std::chrono::duration_cast<std::chrono::seconds>(rubicon -
launch_time).count() < daemon_time)
        {
            // Найти все вложенные директории
            (пользователи).
            auto user_dirs =
FindSubDirictories(source_dir_);
            for(auto& user_dir : user_dirs)
            {
                // Найти все вложенные директории (задачи).
                auto task_dirs =
FindSubDirictories(user_dir);
                for(auto& task_dir : task_dirs)
                {
                    // Расположение файла ошибки.
                    std::string error_file_path = task_dir +
"/" + error_file_name;
                    // Расположение файла параметров
улучшения.
                    std::string upscale_file_path = task_dir
+ "/" + upscale_file_name;

                    // Если в каталоге присутствует файл
ошибки - пропускаем каталог.

                    if(std::filesystem::exists(error_file_path))
                        continue;

                    // Если в каталоге отсутствует файл
параметров улучшения - пропускаем каталог.

                    if(!std::filesystem::exists(upscale_file_path))
                        continue;

                    // Проверить готовность файла параметров
улучшения.

                    if(!CheckFilePreparedness(upscale_file_path, 150ms, false))
                        continue;

```

```

        // Если парсинг параметров не удался -
создать файл ошибки.
        UpscaleSettings upscale_settings{};
        if(ParseUpscale(upscale_file_path,
upscale_settings) != 0)
        {
            std::ofstream
error_file(error_file_path.c_str());
            error_file.close();
            continue;
        }

        // Найти все вложенные директории
(видео) .

        auto video_files =
FindExtensionFiles(task_dir, base_video_format);

        // Отсутствуют файлы - ожидаем.
        if(video_files.size() == 0)
            continue;

        for(auto& video_file : video_files)
        {
            // Наименование папки пользователя.
            std::string user_name =
std::filesystem::path(user_dir).stem();
            // Наименование папки с задачей.
            std::string task_name =
std::filesystem::path(task_dir).stem();
            // Наименование обрабатываемого
ресурса.

            std::string video_name =
std::filesystem::path(video_file).stem();
            // Директория расположения
результата.

            std::string result_dir_path =
result_dir_ + "/" + user_name + "/" + task_name + "/" +
video_name;

            // Расположение файла блокировки.
            std::string lock_path = source_dir_
+ "/" + user_name + "/" + task_name + "/" + video_name +
"_cc.lock";

            // Если файл имеет файл блокировки -
не работаем.

            if(std::filesystem::exists(lock_path))
                continue;

            // Проверить готовность файла для
обработки.

```

```

if(!CheckFilePreparedness(upscale_file_path, 150ms, false))
    continue;

// Если не папка для результата не
существует - создать.

if(!std::filesystem::exists(result_dir_path))
{

if(!std::filesystem::create_directories(result_dir_path))
{

logger_.Print(LogLevel::ERROR, "Не удалось создать директорию
для необработанных чанков " + result_dir_path);
    continue;
    }

    logger_.Print(LogLevel::INFO,
"Создана директория для необработанных чанков " +
result_dir_path);
    }

// Копируем в папку результата
параметры для улучшения.
    std::string upscale_copy_path =
result_dir_ + "/" + user_name + "/" + task_name + "/" +
upscale_file_name;

if(!std::filesystem::exists(upscale_copy_path)){

if(!std::filesystem::copy_file(upscale_file_path,
upscale_copy_path))
    {

logger_.Print(LogLevel::ERROR, "Не удалось скопировать файл
настроек в " +
std::filesystem::path(upscale_file_path).parent_path().string())
;
        continue;
        }

        logger_.Print(LogLevel::INFO,
"Файл настроек скопирован по расположению " +
upscale_copy_path);
        }

// Если нет файла блокировки -
создаём файл блокировки.
        std::ofstream
lock_file(lock_path.c_str());

```

```

        lock_file.close();

        // Если не удалось выделить поток
        под выполнение - удаляем файл блокировки.

        if(!thread_pool_.LaunchThread<ChunksCutter>(&ChunksCutter::Execu
        tor, this, upscale_settings, video_file, result_dir_path))
        {
            std::remove(lock_path.c_str());

        std::this_thread::sleep_for(20ms);
        }
    }
}

        rubicon = std::chrono::system_clock::now();
        std::this_thread::sleep_for(10ms);
    }

    logger_.Print(LogLevel::INFO, "Время работы демона
    истекло (" + std::to_string(daemon_time) + " сек). Плановый
    перезапуск.");
}

    /// @brief Разбивает видеофайл на чанки.
    /// @param[in] thread_num Номер используемого потока.
    /// @param[in] settings Параметры улучшения качества.
    /// @param[in] source_file Расположение видеофайла с
    расширением.
    /// @param[in] result_file Расположение папки для
    отделяемых чанков.
    /// @exception DaemonException - если произошла ошибка
    при выполнении функции.
    void Executor(int thread_num, const UpscaleSettings&
    settings, std::string source_file, const std::string
    result_file)
    {
        try
        {
            auto time_start =
            std::chrono::system_clock::now();

            // Открыть исходник видео.
            AVFormatContext *input_context = nullptr;
            if (avformat_open_input(&input_context,
            source_file.c_str(), nullptr, nullptr) < 0)
                throw DaemonException("Невозможно открыть
            исходник: " + source_file);

            // Найти информационный поток в файле источника.

```

```

        if (avformat_find_stream_info(input_context,
        nullptr) < 0)
            throw DaemonException("Невозможно найти
информацию о потоках источника: " + source_file);

        // Длительность видеофайла.
        int64_t video_len = input_context->duration;
        // Начальная точка отсчёта чанка.
        int64_t start_len{};
        // Конечная точка отсчёта чанка.
        int64_t end_len{};
        // Номер обрабатываемого чанка.
        int current_chunk = -1;
        // Размер деления чанков в секундах.
        const int chunk_size = 10;
        // Префикс чанков.
        const std::string chunk_prefix = "chunk_";

        // До тех пор, пока не будет обработан весь
объём видео
        while(end_len < video_len)
        {
            int stream_index = 0;
            int stream_mapping[input_context->nb_streams];
            int
stream_rescaled_start_seconds[input_context->nb_streams];
            int
stream_rescaled_end_seconds[input_context->nb_streams];

            // Если осталось меньше одного чанка на
обработку.
            if(video_len < end_len + chunk_size *
AV_TIME_BASE)
            {
                start_len = current_chunk * chunk_size *
AV_TIME_BASE;
                end_len = video_len;

                // Если сегмент меньше одной секунды -
не интересуется!
                if(end_len - start_len < AV_TIME_BASE)
                    break;
            }
            else
            {
                current_chunk++;
                start_len = current_chunk * chunk_size *
AV_TIME_BASE;
                end_len += chunk_size * AV_TIME_BASE;
            }
        }
    
```

```

        // Создать выходной контекст чанка.
        AVFormatContext *output_context = nullptr;
        std::string output_file_name = result_file +
"/" + chunk_prefix + std::to_string(current_chunk) +
base_video_format;

avformat_alloc_output_context2(&output_context, NULL, NULL,
output_file_name.c_str());
        if(!output_context)
            throw DaemonException("Невозможно
создать выходной контекст: " + output_file_name);

        // Скопировать потоки из входного файла в
выходной файл.
        for (int i = 0; i < input_context-
>nb_streams; i++) {

            AVStream* inStream = input_context-
>streams[i];

            stream_rescaled_start_seconds[i] =
av_rescale_q(start_len, AV_TIME_BASE_Q, inStream->time_base);
            stream_rescaled_end_seconds[i] =
av_rescale_q(end_len, AV_TIME_BASE_Q, inStream->time_base);

            if (inStream->codecpar->codec_type !=
AVMEDIA_TYPE_AUDIO &&
inStream->codecpar->codec_type !=
AVMEDIA_TYPE_VIDEO &&
inStream->codecpar->codec_type !=
AVMEDIA_TYPE_SUBTITLE) {
                stream_mapping[i] = -1;
                continue;
            }

            stream_mapping[i] = stream_index++;

            AVStream* outStream =
avformat_new_stream(output_context, NULL);
            if (!output_context)
                throw DaemonException("Ошибка
создания выходного потока: " + output_file_name);

            if(avcodec_parameters_copy(outStream-
>codecpar, inStream->codecpar) < 0)
                throw DaemonException("Не удалось
скопировать параметры в поток: " + output_file_name);

            outStream->codecpar->codec_tag = 0;
        }

        if (!(output_context->oformat->flags &
AVFMT_NOFILE)) {

```

```

        if(avio_open(&output_context->pb,
output_file_name.c_str(), AVIO_FLAG_WRITE) < 0)
            throw DaemonException("Невозможно
открыть файл результата: " + output_file_name);
        }

        if(avformat_write_header(output_context,
NULL) < 0)
            throw DaemonException("Ошибка при
открытии файла результата: " + output_file_name);

        if(avformat_seek_file(input_context, -1,
INT64_MIN, start_len, start_len, 0) < 0)
            throw DaemonException("Ошибка смещения:
" + output_file_name);

        AVPacket *packet = av_packet_alloc();
        while (true) {

            if(av_read_frame(input_context, packet)
< 0)
                break;

            // Пропускать пакеты из неизвестных
потоков и пакетов после конечной метки.
            if (packet->stream_index >=
input_context->nb_streams ||
                stream_mapping[packet->stream_index] < 0
|| packet->pts > stream_rescaled_end_seconds[packet-
>stream_index])
            {
                av_packet_unref(packet);
                continue;
            }

            packet->stream_index =
stream_mapping[packet->stream_index];

            // Ужать пакет до новой позиции.
            packet->pts -=
stream_rescaled_start_seconds[packet->stream_index];
            packet->dts -=
stream_rescaled_start_seconds[packet->stream_index];

            av_packet_rescale_ts(packet,
input_context->streams[packet->stream_index]->time_base,
output_context->streams[packet->stream_index]->time_base);
            packet->pos = -1;

            if(av_interleaved_write_frame(output_context, packet) < 0)

```

```

        throw DaemonException("Ошибка сборки
пакета: " + output_file_name);
    }

    av_write_trailer(output_context);
    av_packet_free(&packet);

    if(output_context && !(output_context->
oformat->flags & AVFMT_NOFILE))
        avio_closep(&output_context->pb);
    avformat_free_context(output_context);
}

avformat_close_input(&input_context);

// Создаём файл разрешения доступа.
std::ofstream access_file(result_file + "/" +
access_file_name);
access_file.close();

    auto time_end =
std::chrono::system_clock::now();
    logger_.Print(LogLevel::INFO, "Задача " +
source_file + " была выполнена за " +

std::to_string(std::chrono::duration_cast<std::chrono::seconds>(
time_end - time_start).count()) + " сек. / " +

std::to_string(std::chrono::duration_cast<std::chrono::milliseco
nds>(time_end - time_start).count()) + " милисек.");
    thread_pool_.SetThreadState(thread_num,
ThreadState::FREE);
    }
    catch(const std::exception &ex)
    {
        logger_.Print(LogLevel::ERROR, "Поток " +
std::to_string(thread_num) + " завершен с исключением: " +
ex.what());
        thread_pool_.SetThreadState(thread_num,
ThreadState::EXCEPTION);
    }
}

};

class FramesCutter : public DaemonBase
{
public:
    FramesCutter()
    {
        logger_.SetLogName("frames_cutter");
        thread_pool_.SetThreadCount(3);
    }
}

```

```

~FramesCutter()
{
    logger_.Print(LogLevel::INFO, "Демон
деинициализирован");
}

void Lifeloop()
{
    // Временные засечки для контроля работы демона.
    auto launch_time = std::chrono::system_clock::now();
    auto rubicon = std::chrono::system_clock::now();

    // Тело демона. Пока не превысили время работы -
работаем.

while(std::chrono::duration_cast<std::chrono::seconds>(rubicon -
launch_time).count() < daemon_time)
    {
        // Найти все вложенные директории
(пользователи).
        auto user_dirs =
FindSubDirictories(source_dir_);
        for(auto& user_dir : user_dirs)
        {
            // Найти все вложенные директории (задачи).
            auto task_dirs =
FindSubDirictories(user_dir);
            for(auto& task_dir : task_dirs)
            {
                // Расположение файла ошибки.
                std::string error_file_path = task_dir +
"/" + error_file_name;
                // Расположение файла параметров
улучшения.
                std::string upscale_file_path = task_dir
+ "/" + upscale_file_name;

                // Если в каталоге присутствует файл
ошибки - пропускаем каталог.

                if(std::filesystem::exists(error_file_path))
                    continue;

                // Если в каталоге отсутствует файл
параметров улучшения - пропускаем каталог.

                if(!std::filesystem::exists(upscale_file_path))
                    continue;

                // Проверить готовность файла параметров
улучшения.

```

					ПГУ1.090301.15.001 ПЗ	Лист
						110
Изм.	Лист	№ докум.	Подпись	Дата		

```

if(!CheckFilePreparedness(upscale_file_path, 150ms, false))
    continue;

// Если парсинг параметров не удался -
создать файл ошибки.
UpscaleSettings upscale_settings{};
if(ParseUpscale(upscale_file_path,
upscale_settings) != 0)
{
    std::ofstream
error_file(error_file_path.c_str());
    error_file.close();
    continue;
}
// Найти все вложенные директории
(видео) .
    auto video_dirs =
FindSubDirictories(task_dir);
    for(auto& video_dir : video_dirs)
    {
        // Расположение файла доступа.
        std::string access_file_path =
video_dir + "/" + access_file_name;

        // Если в каталоге отсутствует файл
доступа - пропускаем каталог.

if(!std::filesystem::exists(access_file_path))
    continue;

        // Найти все вложенные файлы
(чанки) .
        auto chunk_files =
FindExtensionFiles(video_dir, base_video_format);

        // Отсутствуют файлы - ошибка.
        if(chunk_files.size() == 0)
        {
            std::ofstream
error_file(error_file_path.c_str());
            error_file.close();
            logger_.Print(LogLevel::ERROR,
"Не найдены ресурсы для обработки в каталоге: " + video_dir);
            continue;
        }

        for(auto& chunk_file : chunk_files)
        {
            // Наименование папки
пользователя.

```

```

std::string user_name =
std::filesystem::path(user_dir).stem();
// Наименование папки с задачей.
std::string task_name =
std::filesystem::path(task_dir).stem();
// Наименование папки с видео.
std::string video_name =
std::filesystem::path(video_dir).stem();
// Наименование чанка видео.
std::string chunk_name =
std::filesystem::path(chunk_file).stem();
// Директория расположения
результата.
std::string result_dir_path =
result_dir_ + "/" + user_name + "/" + task_name + "/" +
video_name + "/" + chunk_name;
// Расположение файла
блокировки.
std::string lock_path =
source_dir_ + "/" + user_name + "/" + task_name + "/" +
video_name + "/" + chunk_name + "_fc.lock";

// Если файл имеет файл
блокировки - не работаем.
if(std::filesystem::exists(lock_path))
    continue;

// Если не папка для результата
не существует - создать.
if(!std::filesystem::exists(result_dir_path))
{
if(!std::filesystem::create_directories(result_dir_path))
{
logger_.Print(LogLevel::ERROR, "Не удалось создать директорию
для необработанных фреймов " + result_dir_path);
continue;
}

logger_.Print(LogLevel::INFO, "Создана директория для
необработанных фреймов " + result_dir_path);
}

// Копируем в папку результата
параметры для улучшения.
std::string upscale_copy_path =
result_dir_ + "/" + user_name + "/" + task_name + "/" +
upscale_file_name;

```

```

if(!std::filesystem::exists(upscale_copy_path)){

if(!std::filesystem::copy_file(upscale_file_path,
upscale_copy_path))

{

logger_.Print(LogLevel::ERROR, "Не удалось скопировать файл
настроек в " +
std::filesystem::path(upscale_file_path).parent_path().string())
;

continue;

}

logger_.Print(LogLevel::INFO, "Файл настроек скопирован по
расположению " + upscale_copy_path);
}

// Если нет файла блокировки -
создаём файл блокировки.
std::ofstream
lock_file(lock_path.c_str());
lock_file.close();

// Если не удалось выделить
поток под выполнение - удаляем файл блокировки.

if(!thread_pool_.LaunchThread<FramesCutter>(&FramesCutter::Execu
tor, this, upscale_settings, chunk_file, result_dir_path))
{

std::remove(lock_path.c_str());

std::this_thread::sleep_for(20ms);
}

}

}

}

rubicon = std::chrono::system_clock::now();
std::this_thread::sleep_for(20ms);
}

logger_.Print(LogLevel::INFO, "Время работы демона
истекло (" + std::to_string(daemon_time) + " сек). Плановый
перезапуск.");
}

/// @brief Разбивает фидеофайл на фреймы.

```

```

        /// @param[in] thread_num Номер используемого потока.
        /// @param[in] settings Параметры улучшения качества.
        /// @param[in] source_file Расположение видеофайла с
расширением.
        /// @param[in] result_file Расположение папки для
отделяемых фреймов.
        /// @exception DaemonException - если произошла ошибка
при выполнении функции.
        void Executor(int thread_num, const UpscaleSettings&
settings, std::string source_file, const std::string
result_file)
        {
            try
            {
                auto time_start =
std::chrono::system_clock::now();

                // Префикс фреймов.
                const std::string frame_prefix = "frame_";

                // Получение кадров с необходимой частотой.
(ЗАГ)
                std::string command = "ffmpeg -i " + source_file
+ " -r " + std::to_string(settings.fps_) +
                " -vf scale=" +
std::to_string(settings.horizontal_/neural_scale) + ":" +
std::to_string(settings.vertical_/neural_scale) + " " +
                result_file + "/" +
frame_prefix + "%d" + base_frames_format;
                std::system(command.c_str());

                // Проверка остановки генерации файлов. (ЗАГ)
                std::vector<std::string> control{};
                int retries = 3;
                int new_size = 0;
                int old_size = 0;
                int success_retries = 0;

                // Цикл ожидания завершения генерации. (ЗАГ)
                while(success_retries < retries)
                {
                    control = FindExtensionFiles(result_file,
base_frames_format);
                    new_size = control.size();

                    if(new_size != old_size)
                    {
                        success_retries = 0;
                    }
                    else
                    {
                        success_retries++;
                    }
                }
            }
        }
    }

```

```

    }

    old_size = new_size;
    std::this_thread::sleep_for(1500ms);
}

// Создаём файл разрешения доступа.
std::ofstream access_file(result_file + "/" +
access_file_name);
    access_file.close();

    auto time_end =
std::chrono::system_clock::now();
    logger_.Print(LogLevel::INFO, "Задача " +
source_file + " была выполнена за " +

std::to_string(std::chrono::duration_cast<std::chrono::seconds>(
time_end - time_start).count()) + " сек. / " +

std::to_string(std::chrono::duration_cast<std::chrono::milliseco
nds>(time_end - time_start).count()) + " миллисек.");
    thread_pool_.SetThreadState(thread_num,
ThreadState::FREE);
    }
    catch(const std::exception &ex)
    {
        logger_.Print(LogLevel::ERROR, "Поток " +
std::to_string(thread_num) + " завершен с исключением: " +
ex.what());
        thread_pool_.SetThreadState(thread_num,
ThreadState::EXCEPTION);
    }
}

};

class Neuralizator : public DaemonBase
{
public:
    Neuralizator()
    {
        logger_.SetLogName("neuralizator");
        // Компонент для взаимодействия с нейронной сетью.
Меньше потоков.
        thread_pool_.SetThreadCount(5);
    }
    ~Neuralizator()
    {
        logger_.Print(LogLevel::INFO, "Демон
деинициализирован");
    }

    void Lifeloop()

```

```

{
    // Временные засечки для контроля работы демона.
    auto launch_time = std::chrono::system_clock::now();
    auto rubicon = std::chrono::system_clock::now();

    // Тело демона. Пока не превысили время работы -
    работаем.

    while(std::chrono::duration_cast<std::chrono::seconds>(rubicon -
    launch_time).count() < daemon_time)
    {
        // Найти все вложенные директории
        (пользователи).
        auto user_dirs =
        FindSubDirictories(source_dir_);
        for(auto& user_dir : user_dirs)
        {
            // Найти все вложенные директории (задачи).
            auto task_dirs =
            FindSubDirictories(user_dir);
            for(auto& task_dir : task_dirs)
            {
                // Расположение файла ошибки.
                std::string error_file_path = task_dir +
                "/" + error_file_name;
                // Расположение файла параметров
                улучшения.
                std::string upscale_file_path = task_dir
                + "/" + upscale_file_name;

                // Если в каталоге присутствует файл
                ошибки - пропускаем каталог.

                if(std::filesystem::exists(error_file_path))
                    continue;

                // Если в каталоге отсутствует файл
                параметров улучшения - пропускаем каталог.

                if(!std::filesystem::exists(upscale_file_path))
                    continue;

                // Проверить готовность файла параметров
                улучшения.

                if(!CheckFilePreparedness(upscale_file_path, 150ms, false))
                    continue;

                // Если парсинг параметров не удался -
                создать файл ошибки.
                UpscaleSettings upscale_settings{};

```

```

        if (ParseUpscale(upscale_file_path,
upscale_settings) != 0)
        {
            std::ofstream
error_file(error_file_path.c_str());
            error_file.close();
            continue;
        }
        // Найти все вложенные директории
(видео).

        auto video_dirs =
FindSubDirictories(task_dir);
        for(auto& video_dir : video_dirs)
        {
            // Расположение файла ошибки.
            std::string internal_error_file_path
= video_dir + "/" + error_file_name;

            // Если в каталоге присутствует файл
ошибки - пропускаем каталог.

            if(std::filesystem::exists(internal_error_file_path))
                continue;

            // Найти все вложенные директории
(фреймы).

            auto frame_dirs =
FindSubDirictories(video_dir);
            if(frame_dirs.size() == 0)
            {
                std::ofstream
error_file(internal_error_file_path.c_str());
                error_file.close();
                logger_.Print(LogLevel::ERROR,
"Не найдены ресурсы для обработки в каталоге: " + video_dir);
                continue;
            }

            for(auto& frame_dir : frame_dirs)
            {
                // Наименование основной
директории.

                std::string user_name =
std::filesystem::path(user_dir).stem();
                // Наименование директории
задачи.

                std::string task_name =
std::filesystem::path(task_dir).stem();
                // Наименование директории
видео.

                std::string video_name =
std::filesystem::path(video_dir).stem();

```

```

// Наименование директории
фреймов.
std::string frames_name =
std::filesystem::path(frame_dir).stem();
// Общая часть пути.
std::string shared_path =
user_name + "/" + task_name + "/" + video_name + "/" +
frames_name;
// Директория расположения
результата.
std::string result_dir_path =
result_dir_ + "/" + shared_path;
// Расположение файла
блокировки.
std::string lock_path =
source_dir_ + "/" + shared_path + "/" + frames_name +
"_neural.lock";
// Расположение файла доступа.
std::string access_file_path =
frame_dir + "/" + access_file_name;
// Расположение файла ошибки.
std::string
frame_error_file_path = frame_dir + "/" + error_file_name;
// Если в каталоге присутствует
файл ошибки - пропускаем каталог.
if(std::filesystem::exists(frame_error_file_path))
    continue;
// Если в каталоге отсутствует
файл доступа - пропускаем каталог.
if(!std::filesystem::exists(access_file_path))
    continue;
// Если файл имеет файл
блокировки - не работаем.
if(std::filesystem::exists(lock_path))
    continue;
// Если нет файла блокировки -
создаём файл блокировки.
std::ofstream
lock_file(lock_path.c_str());
lock_file.close();
// Если не папка для результата
не существует - создать.
if(!std::filesystem::exists(result_dir_path))

```

```

        {

if(!std::filesystem::create_directories(result_dir_path))
        {

std::remove(lock_path.c_str());

logger_.Print(LogLevel::ERROR, "Не удалось создать директорию
для обработанных фреймов " + result_dir_path);
                continue;
        }

logger_.Print(LogLevel::INFO, "Создана директория для
обработанных фреймов " + result_dir_path);
        }

        // Если в папке для результата
нет параметров для улучшения - копируем.
                std::string upscale_file_path =
result_dir_ + "/" + user_name + "/" + task_name + "/" +
upscale_file_name;

if(!std::filesystem::exists(upscale_file_path))
        {
                // Копируем в папку
результата параметры для улучшения.

if(!std::filesystem::copy_file(task_dir + "/" +
upscale_file_name, upscale_file_path))
                {

std::remove(lock_path.c_str());

logger_.Print(LogLevel::ERROR, "Не удалось скопировать файл
настроек в " +
std::filesystem::path(upscale_file_path).parent_path().string())
;

                continue;
        }

logger_.Print(LogLevel::INFO, "Файл настроек скопирован в " +
std::filesystem::path(upscale_file_path).parent_path().string())
;

        }

        // Если не удалось выделить
поток под выполнение - удаляем файл блокировки.

if(!thread_pool_.LaunchThread<Neuralizator>(&Neuralizator::Execu
tor, this, upscale_settings, frame_dir, result_dir_path))

```

```

        {

std::remove(lock_path.c_str());

std::this_thread::sleep_for(20ms);
        }
    }
}

rubicon = std::chrono::system_clock::now();
std::this_thread::sleep_for(20ms);
}

logger_.Print(LogLevel::INFO, "Время работы демона
истекло (" + std::to_string(daemon_time) + " сек). Плановый
перезапуск.");
}

/// @brief Разбивает видеофайл на фреймы.
/// @param[in] thread_num Номер используемого потока.
/// @param[in] settings Параметры улучшения качества.
/// @param[in] source_file Расположение видеофайла с
расширением.
/// @param[in] result_file Расположение папки для
отделяемых фреймов.
/// @exception DaemonException - если произошла ошибка
при выполнении функции.
void Executor(int thread_num, const UpscaleSettings&
settings, std::string source_file, const std::string
result_file)
{
    try
    {
        auto time_start =
std::chrono::system_clock::now();

        // Формирование команды для нейронной сети.
        std::string command = "./neurals/" +
settings.neural_ + "/" + settings.neural_ + " -i " + source_file
+
        " -s " +
std::to_string(neural_scale) + " -o " + result_file;
        std::system(command.c_str());

        // Пока-что, будет проверка на сопоставление
кол-ва кадров. (ЗАГ)
        auto control = FindExtensionFiles(source_file,
base_frames_format);
        int control_size = control.size();

```

```

        int current_size = 0;

        while(current_size < control_size)
        {
            control = FindExtensionFiles(source_file,
base_frames_format);
            current_size = control.size();

            std::this_thread::sleep_for(1500ms);
        }

        // Создаём файл разрешения доступа.
        std::ofstream access_file(result_file + "/" +
access_file_name);
        access_file.close();

        auto time_end =
std::chrono::system_clock::now();
        logger_.Print(LogLevel::INFO, "Задача " +
source_file + " была выполнена за " +

std::to_string(std::chrono::duration_cast<std::chrono::seconds>(
time_end - time_start).count()) + " сек. / " +

std::to_string(std::chrono::duration_cast<std::chrono::milliseco
nds>(time_end - time_start).count()) + " миллисек.");
        thread_pool_.SetThreadState(thread_num,
ThreadState::FREE);
    }
    catch(const std::exception &ex)
    {
        logger_.Print(LogLevel::ERROR, "Поток " +
std::to_string(thread_num) + " завершен с исключением: " +
ex.what());
        thread_pool_.SetThreadState(thread_num,
ThreadState::EXCEPTION);
    }
}

};

class FramesGluer : public DaemonBase
{
public:
    FramesGluer()
    {
        logger_.SetLogName("frames_gluer");
        thread_pool_.SetThreadCount(5);
    }
    ~FramesGluer()
    {
        logger_.Print(LogLevel::INFO, "Демон
деинициализирован");
    }
};

```

```

    }

    void Lifeloop()
    {
        // Временные засечки для контроля работы демона.
        auto launch_time = std::chrono::system_clock::now();
        auto rubicon = std::chrono::system_clock::now();

        // Тело демона. Пока не превысили время работы -
        работаем.

        while(std::chrono::duration_cast<std::chrono::seconds>(rubicon -
        launch_time).count() < daemon_time)
        {
            // Найти все вложенные директории
            (пользователи).
            auto user_dirs =
            FindSubDirictories(source_dir_);
            for(auto& user_dir : user_dirs)
            {
                // Найти все вложенные директории (задачи).
                auto task_dirs =
                FindSubDirictories(user_dir);
                for(auto& task_dir : task_dirs)
                {
                    // Расположение файла ошибки.
                    std::string error_file_path = task_dir +
                    "/" + error_file_name;
                    // Расположение файла параметров
                    улучшения.
                    std::string upscale_file_path = task_dir
                    + "/" + upscale_file_name;

                    // Если в каталоге присутствует файл
                    ошибки - пропускаем каталог.

                    if(std::filesystem::exists(error_file_path))
                        continue;

                    // Если в каталоге отсутствует файл
                    параметров улучшения - пропускаем каталог.

                    if(!std::filesystem::exists(upscale_file_path))
                        continue;

                    // Проверить готовность файла параметров
                    улучшения.

                    if(!CheckFilePreparedness(upscale_file_path, 150ms, false))
                        continue;
                }
            }
        }
    }

```

```

        // Если парсинг параметров не удался -
        создать файл ошибки.
        UpscaleSettings upscale_settings{};
        if(ParseUpscale(upscale_file_path,
upscale_settings) != 0)
        {
            std::ofstream
error_file(error_file_path.c_str());
            error_file.close();
            continue;
        }

        // Найти все вложенные директории
        (видео) .
        auto video_dirs =
FindSubDirictories(task_dir);
        for(auto& video_dir : video_dirs)
        {
            // Расположение файла ошибки.
            std::string internal_error_file_path
= video_dir + "/" + error_file_name;

            // Если в каталоге присутствует файл
            ошибки - пропускаем каталог.

            if(std::filesystem::exists(internal_error_file_path))
                continue;

            // Найти все вложенные директории
            (чанки) .
            auto chunk_dirs =
FindSubDirictories(video_dir);

            // Отсутствуют файлы - ошибка.
            if(chunk_dirs.size() == 0)
            {
                std::ofstream
error_file(error_file_path.c_str());
                error_file.close();
                logger_.Print(LogLevel::ERROR,
"Не найдены ресурсы для обработки в каталоге: " + video_dir);
                continue;
            }

            for(auto& chunk_dir : chunk_dirs)
            {
                // Наименование папки
                пользователя.

                std::string user_name =
std::filesystem::path(user_dir).stem();
                // Наименование папки с задачей.

```

```

std::string task_name =
std::filesystem::path(task_dir).stem();
// Наименование папки с видео.
std::string video_name =
std::filesystem::path(video_dir).stem();
// Наименование чанка видео.
std::string chunk_name =
std::filesystem::path(chunk_dir).stem();
// Директория расположения
результата.

std::string result_dir_path =
result_dir_ + "/" + user_name + "/" + task_name + "/" +
video_name + "/" + chunk_name + base_video_format;
// Расположение файла
блокировки.

std::string lock_path =
source_dir_ + "/" + user_name + "/" + task_name + "/" +
video_name + "/" + chunk_name + "_fg.lock";
// Расположение файла доступа.
std::string access_file_path =
chunk_dir + "/" + access_file_name;
// Расположение файла ошибки.
std::string
frame_error_file_path = chunk_dir + "/" + error_file_name;

// Если в каталоге присутствует
файл ошибки - пропускаем каталог.

if(std::filesystem::exists(frame_error_file_path))
    continue;

// Если в каталоге отсутствует
файл доступа - пропускаем каталог.

if(!std::filesystem::exists(access_file_path))
    continue;

// Если файл имеет файл
блокировки - пропускаем каталог.

if(std::filesystem::exists(lock_path))
    continue;

// Если не папка для результата
не существует - создать.

if(!std::filesystem::exists(std::filesystem::path(result_dir_pat
h).parent_path().string()))
{

if(!std::filesystem::create_directories(std::filesystem::path(re
sult_dir_path).parent_path().string()))

```

```

        {

logger_.Print(LogLevel::ERROR, "Не удалось создать директорию
для обработанных чанков " + result_dir_path);
        continue;
        }

logger_.Print(LogLevel::INFO, "Создана директория для
обработанных чанков " + result_dir_path);
        }

        // Если нет файла блокировки -
создаём файл блокировки.
        std::ofstream
lock_file(lock_path.c_str());
        lock_file.close();

        // Если не удалось выделить
поток под выполнение - удаляем файл блокировки.

if(!thread_pool_.LaunchThread<FramesGluer>(&FramesGluer::Executo
r, this, upscale_settings, chunk_dir, result_dir_path))
        {

std::remove(lock_path.c_str());

std::this_thread::sleep_for(20ms);
        }
        }
        }
        }

        rubicon = std::chrono::system_clock::now();
std::this_thread::sleep_for(20ms);
        }

        logger_.Print(LogLevel::INFO, "Время работы демона
истекло (" + std::to_string(daemon_time) + " сек). Плановый
перезапуск.");
        }

        /// @brief Разбивает фидеофайл на фреймы.
        /// @param[in] thread_num Номер используемого потока.
        /// @param[in] settings Параметры улучшения качества.
        /// @param[in] source_file Расположение видеофайла с
расширением.
        /// @param[in] result_file Расположение папки для
отделяемых фреймов.
        /// @exception DaemonException - если произошла ошибка
при выполнении функции.

```

```

        void Executor(int thread_num, const UpscaleSettings&
settings, std::string source_file, const std::string
result_file)
        {
            try
            {
                auto time_start =
std::chrono::system_clock::now();

                // Префикс фреймов.
                const std::string frame_prefix = "frame_";

                int old_file_size = 0;
                int new_file_size = 0;
                int retries = 3;
                int current_retries = 0;

                // Формирование команды для склеивания фреймов.
                std::string command = "ffmpeg -r " +
std::to_string(settings.fps_) +
                " -i " + source_file + "/" + frame_prefix +
"%d.png -c:v libx264 " + result_file;
                std::system(command.c_str());

                while(current_retries < retries)
                {
                    new_file_size =
std::filesystem::file_size(result_file);
                    if(old_file_size == new_file_size)
                    {
                        current_retries++;
                    }
                    else
                    {
                        current_retries = 0;
                    }

                    old_file_size = new_file_size;
                    std::this_thread::sleep_for(200ms);
                }

                // Создаём файл разрешения доступа.
                //std::ofstream
                access_file(std::filesystem::path(result_file).parent_path().str
ing() + "/" + access_file_name);
                //access_file.close();

                auto time_end =
std::chrono::system_clock::now();
                logger_.Print(LogLevel::INFO, "Задача " +
source_file + " была выполнена за " +

```

```

std::to_string(std::chrono::duration_cast<std::chrono::seconds>(
time_end - time_start).count()) + " сек. / " +

std::to_string(std::chrono::duration_cast<std::chrono::milliseco
nds>(time_end - time_start).count()) + " миллисек.");
        thread_pool_.SetThreadState(thread_num,
ThreadState::FREE);
    }
    catch(const std::exception &ex)
    {
        logger_.Print(LogLevel::ERROR, "Поток " +
std::to_string(thread_num) + " завершен с исключением: " +
ex.what());
        thread_pool_.SetThreadState(thread_num,
ThreadState::EXCEPTION);
    }
}

};

class ChunksGluer : public DaemonBase
{
public:
    ChunksGluer()
    {
        logger_.SetLogName("chunks_gluer");
    }
    ~ChunksGluer()
    {
        logger_.Print(LogLevel::INFO, "Демон
деинициализирован");
    }

    void Lifeloop()
    {
        // Временные засечки для контроля работы демона.
        auto launch_time = std::chrono::system_clock::now();
        auto rubicon = std::chrono::system_clock::now();

        UpscaleSettings upscale_settings{};

        // Тело демона. Пока не превысили время работы -
работаем.

        while(std::chrono::duration_cast<std::chrono::seconds>(rubicon -
launch_time).count() < daemon_time)
        {
            // Найти все вложенные директории
(пользователи).
            auto user_dirs =
FindSubDirictories(source_dir_);
            for(auto& user_dir : user_dirs)

```

```

        {
            // Найти все вложенные директории (задачи).
            auto task_dirs =
FindSubDirictories(user_dir);
            for(auto& task_dir : task_dirs)
            {
                // Расположение файла ошибки.
                std::string error_file_path = task_dir +
"/" + error_file_name;

                // Если в каталоге присутствует файл
ошибки - пропускаем каталог.

if(std::filesystem::exists(error_file_path))
                continue;

                // Найти все вложенные директории
(видео) .

                auto video_dirs =
FindSubDirictories(task_dir);
                for(auto& video_dir : video_dirs)
                {
                    // Расположение файла ошибки.
                    std::string internal_error_file_path
= video_dir + "/" + error_file_name;

                    // Если в каталоге присутствует файл
ошибки - пропускаем каталог.

if(std::filesystem::exists(internal_error_file_path))
                    continue;

                    // Найти файлы для склеивания
(чанки) .

                    auto chunk_files =
FindExtensionFiles(video_dir, base_video_format);

                    // Наименование основной директории.
                    std::string user_name =
std::filesystem::path(user_dir).stem();
                    // Наименование директории задачи.
                    std::string task_name =
std::filesystem::path(task_dir).stem();
                    // Наименование директории видео.
                    std::string video_name =
std::filesystem::path(video_dir).stem();

                    // Проверить целостность файлов и
количество. (Заглушка!)

                    auto control_size =
FindExtensionFiles("/home/user/makima/raw_chunks/" + user_name +
"/" + task_name + "/" + video_name, base_video_format);

```

					ПГУ1.090301.15.001 ПЗ	Лист
						128
Изм.	Лист	№ докум.	Подпись	Дата		

```

// Если количество чанков
соответствует изначальному.
if(control_size.size() !=
chunk_files.size())

    continue;

// Проверка на изменяемость.
bool test_result = false;
for(auto& chunk_file : chunk_files)
{

if(!CheckFilePreparedness(chunk_file, 4000ms, false))
    {
        test_result = true;
        break;
    }

if(test_result)
    continue;

std::string base_dir =
std::filesystem::path(chunk_files[0]).parent_path().string();

// Создать файл со списком чанков.
std::ofstream
chunks_list_file(video_dir + "/" + chunks_list_name);
for(int i = 0; i <
chunk_files.size(); i++)
{
    chunks_list_file << "file '" <<
base_dir << "/chunk_" << std::to_string(i) << base_video_format
<< ".\n";
}
chunks_list_file.close();

// Общая часть пути.
std::string shared_path = user_name
+ "/" + task_name + "/" + video_name;
// Расположения результата.
std::string result_path =
result_dir_ + "/" + shared_path + "/" + video_name +
base_video_format;

// Расположение файла блокировки.
std::string lock_path = source_dir_
+ "/" + shared_path + "/" + video_name + "_neural.lock";

// Если файл имеет файл блокировки -
не работаем.

if(std::filesystem::exists(lock_path))
    continue;

```

```

        // Если нет файла блокировки -
создаём файл блокировки.
        std::ofstream
lock_file(lock_path.c_str());
        lock_file.close();

        // Если не папка для результата не
существует - создать.

if(!std::filesystem::exists(result_dir_ + "/" + shared_path))
{

if(!std::filesystem::create_directories(result_dir_ + "/" +
shared_path))

{

std::remove(lock_path.c_str());

logger_.Print(LogLevel::ERROR, "Не удалось создать директорию
для обработанного видео " + result_dir_ + "/" + shared_path);
        continue;
    }

        logger_.Print(LogLevel::INFO,
"Создана директория для обработанного видео " + result_dir_ +
"/" + shared_path);
    }

        // Если не удалось выделить поток
под выполнение - удаляем файл блокировки.

if(!thread_pool_.LaunchThread<ChunksGluer>(&ChunksGluer::Executo
r, this, upscale_settings, video_dir, result_path))
{
        std::remove(lock_path.c_str());

std::this_thread::sleep_for(20ms);
    }
}

        rubicon = std::chrono::system_clock::now();
std::this_thread::sleep_for(20ms);
    }

        logger_.Print(LogLevel::INFO, "Время работы демона
истекло (" + std::to_string(daemon_time) + " сек). Плановый
перезапуск.");
    }

```

```

        /// @brief Разбивает видеофайл на фреймы.
        /// @param[in] thread_num Номер используемого потока.
        /// @param[in] settings Параметры улучшения качества.
        /// @param[in] source_file Расположение видеофайла с
расширением.
        /// @param[in] result_file Расположение папки для
отделяемых фреймов.
        /// @exception DaemonException - если произошла ошибка
при выполнении функции.
        void Executor(int thread_num, const UpscaleSettings&
settings, std::string source_file, const std::string
result_file)
        {
            try
            {
                auto time_start =
std::chrono::system_clock::now();

                int old_file_size = 0;
                int new_file_size = 0;
                int retries = 3;
                int current_retries = 0;

                // Расположение файла с чанками.
                std::string chunks_list_path = source_file + "/"
+ chunks_list_name;

                // Проверка наличия файла.
                if(!std::filesystem::exists(chunks_list_path))
                    throw DaemonException("Отсутствует файл-
перечисление для сливания чанков: " + chunks_list_path);

                // Формирование команды для склеивания чанков.
                std::string command = "ffmpeg -f concat -safe 0
-i " + chunks_list_path + " -c copy " + result_file;
                std::system(command.c_str());

                // Пока файл изменяет размер - продолжаем ждать.
                while(current_retries < retries)
                {
                    new_file_size =
std::filesystem::file_size(result_file);
                    if(old_file_size == new_file_size)
                    {
                        current_retries++;
                    }
                    else
                    {
                        current_retries = 0;
                    }

                    old_file_size = new_file_size;

```

```

        std::this_thread::sleep_for(300ms);
    }

    // Создаём файл разрешения доступа.
    std::ofstream
    access_file(std::filesystem::path(result_file).parent_path().str
    ing() + "/" + access_file_name);
    access_file.close();

    auto time_end =
    std::chrono::system_clock::now();
    logger_.Print(LogLevel::INFO, "Задача " +
    source_file + " была выполнена за " +

    std::to_string(std::chrono::duration_cast<std::chrono::seconds>(
    time_end - time_start).count()) + " сек. / " +

    std::to_string(std::chrono::duration_cast<std::chrono::milliseco
    nds>(time_end - time_start).count()) + " миллисек.");
    thread_pool_.SetThreadState(thread_num,
    ThreadState::FREE);
    }
    catch(const std::exception &ex)
    {
        logger_.Print(LogLevel::ERROR, "Поток " +
    std::to_string(thread_num) + " завершен с исключением: " +
    ex.what());
        thread_pool_.SetThreadState(thread_num,
    ThreadState::EXCEPTION);
    }
    }
};

class AudioGluer : public DaemonBase
{
public:
    AudioGluer()
    {
        logger_.SetLogName("audio_gluer");
    }
    ~AudioGluer()
    {
        logger_.Print(LogLevel::INFO, "Демон
    деинициализирован");
    }

    void Lifeloop()
    {
        // Временные засечки для контроля работы демона.
        auto launch_time = std::chrono::system_clock::now();
        auto rubicon = std::chrono::system_clock::now();
    }
};

```

```

UpscaleSettings upscale_settings{};

// Тело демона. Пока не превысили время работы -
работаем.

while(std::chrono::duration_cast<std::chrono::seconds>(rubicon -
launch_time).count() < daemon_time)
{
    // Найти все вложенные директории
    (пользователи).
    auto user_dirs =
FindSubDirictories(source_dir_);
    for(auto& user_dir : user_dirs)
    {
        // Найти все вложенные директории (задачи).
        auto task_dirs =
FindSubDirictories(user_dir);
        for(auto& task_dir : task_dirs)
        {
            // Найти все вложенные директории
            (видео).
            auto video_dirs =
FindSubDirictories(task_dir);
            for(auto& video_dir : video_dirs)
            {
                // Расположение файла ошибки.
                std::string error_file_path =
video_dir + "/" + error_file_name;

                // Если в каталоге присутствует файл
                ошибки - пропускаем каталог.

                if(std::filesystem::exists(error_file_path))
                    continue;

                // Найти все вложенные видео.
                auto video_files =
FindExtensionFiles(video_dir, base_video_format);
                for(auto& video_file : video_files)
                {

                    // Проверка соответствия размеров
                    видео.

                    if(!CheckFilePreparedness(video_file, 4000ms, false))
                        continue;

                    // Путь к аудиофайлу.
                    const std::string audio_path =
std::filesystem::path(video_file).parent_path().string() +

```

```

        "/" +
std::filesystem::path(video_file).stem().string() +
base_audio_format;

        // Если аудиофала не существует
- пропускаем.

if(!std::filesystem::exists(audio_path))
    continue;

        // Расположение файла
блокировки.
        const std::string lock_path =
std::filesystem::path(video_file).parent_path().string() + "/" +
std::filesystem::path(video_file).stem().string() + "_ag.lock";

        // Если файл имеет файл
блокировки - не работаем.

if(std::filesystem::exists(lock_path))
    continue;

        // Если нет файла блокировки -
создаём файл блокировки.
        std::ofstream
lock_file(lock_path.c_str());
        lock_file.close();

        // Если не удалось выделить
поток под выполнение - удаляем файл блокировки.

if(!thread_pool_.LaunchThread<AudioGluer>(&AudioGluer::Executor,
this, upscale_settings, video_file, video_dir))
{

std::remove(lock_path.c_str());

std::this_thread::sleep_for(20ms);
    }
}
}

        rubicon = std::chrono::system_clock::now();
std::this_thread::sleep_for(20ms);
    }

        logger_.Print(LogLevel::INFO, "Время работы демона
истекло (" + std::to_string(daemon_time) + " сек). Плановый
перезапуск.");
    }

```

```

        /// @brief Разбивает видеофайл на фреймы.
        /// @param[in] thread_num Номер используемого потока.
        /// @param[in] settings Параметры улучшения качества.
        /// @param[in] source_file Расположение видеофайла с
расширением.
        /// @param[in] result_file Расположение папки для
отделяемых фреймов.
        /// @exception DaemonException - если произошла ошибка
при выполнении функции.
        void Executor(int thread_num, const UpscaleSettings&
settings, std::string source_file, const std::string
result_file)
        {
            try
            {
                auto time_start =
std::chrono::system_clock::now();

                // Путь до оригинала видео.
                const std::filesystem::path
file_path(source_file);
                // Наименование нового файла.
                const std::string new_file_path =
file_path.parent_path().string() + "/" +
file_path.stem().string() + "_neuro" + base_video_format;
                // Наименование аудио файла.
                const std::string audio_path =
file_path.parent_path().string() + "/" +
file_path.stem().string() + base_audio_format;
                // Наименование файла блокировки.
                const std::string lock_file =
file_path.parent_path().string() + "/" +
file_path.stem().string() + "_ag.lock";

                int old_file_size = 0;
                int new_file_size = 0;
                int retries = 3;
                int current_retries = 0;

                // Формирование команды для склеивания чанков.
                std::string command = "ffmpeg -i " +
source_file + " -i " + audio_path + " -c:v copy -c:a copy " +
new_file_path;

                std::system(command.c_str());

                // Пока файл изменяет размер - продолжаем ждать.
                while(current_retries < retries)
                {
                    new_file_size =
std::filesystem::file_size(new_file_path);
                    if(old_file_size == new_file_size)

```

```

        {
            current_retries++;
        }
        else
        {
            current_retries = 0;
        }

        old_file_size = new_file_size;
        std::this_thread::sleep_for(4000ms);
    }

    // Предварительное удаление.
    std::filesystem::remove(source_file);
    std::filesystem::remove(audio_path);
    std::filesystem::remove(lock_file);

    // Создаём файл разрешения доступа.
    std::ofstream access_file(result_file + "/" +
access_file_name);
    access_file.close();

    auto time_end =
std::chrono::system_clock::now();
    logger_.Print(LogLevel::INFO, "Задача " +
source_file + " была выполнена за " +

std::to_string(std::chrono::duration_cast<std::chrono::seconds>(
time_end - time_start).count()) + " сек. / " +

std::to_string(std::chrono::duration_cast<std::chrono::milliseco
nds>(time_end - time_start).count()) + " миллисек.");
    thread_pool_.SetThreadState(thread_num,
ThreadState::FREE);
    }
    catch(const std::exception &ex)
    {
        logger_.Print(LogLevel::ERROR, "Поток " +
std::to_string(thread_num) + " завершен с исключением: " +
ex.what());
        thread_pool_.SetThreadState(thread_num,
ThreadState::EXCEPTION);
    }
}

};

```

Файл thread_pool.hpp

```
#pragma once
```

```
// Подключение библиотек C++.
#include <memory>
```

					ПГУ1.090301.15.001 ПЗ	Лист
						136
Изм.	Лист	№ докум.	Подпись	Дата		

```

#include <chrono>
#include <mutex>
#include <thread>
#include <vector>

// Подключение логера.
#include "logger.hpp"

// Подключение общих элементов.
#include "shares.hpp"

using namespace std::chrono_literals;

/// @brief Состояния потока.
enum ThreadState
{
    FREE = 0,
    EXECUTE = 1,
    EXCEPTION = 2
};

/// @brief Пул потоков для распараллеливания нагрузки.
class ThreadPool
{
public:
    ThreadPool() = delete;
    ThreadPool(Logger* logger, int threads_count = 5) :
logger_(logger)
    {
        SetThreadCount(threads_count);
        logger_>Print(LogLevel::INFO, "Пул потоков
инициализирован");
    };
    ~ThreadPool()
    {
        logger_>Print(LogLevel::INFO, "Пул потоков
деинициализирован");
    };

    /// @brief Установить необходимое кол-во потоков
(принудительно).
    /// @param[in] threads_count Новое кол-во потоков.
    void SetThreadCount(int threads_count = 10)
    {
        std::unique_lock lock(state_mutex_);
        int threads_tendence = threads_count -
thread_pool_.size();

        // Если потоков меньше, чем требуется - добавляем
новые.
        if(threads_tendence > 0)
        {

```

```

        for(int t = 0; t < threads_tendence; t++)
        {
thread_pool_.push_back(std::shared_ptr<std::thread>());
            threads_state_.push_back(ThreadState::FREE);
        }
    }
    // Если потоков больше, чем требуется, то
освобождаем. (Проверить!!!)
    else if(threads_tendence < 0)
    {
        do
        {
            for(int th = 0; th < threads_state_.size();
th++)
            {
                if(ThreadState::FREE ==
threads_state_[th])
                {
thread_pool_.erase(thread_pool_.begin() + th);
threads_state_.erase(threads_state_.begin() + th);
                    th--;
                    threads_tendence++;

                    if(threads_tendence <= 0)
                    {
                        // <- Добавить комментарий о
сбросе потоков.
                        break;
                    }
                }
            }
        }while(threads_tendence < 0);
    }
    // Идентичное кол-во потоков - ничего не делаем.
    else
    {
        return;
    }

    logger_->Print(LogLevel::DEBUG, "Установлено новое
кол-во потоков: " + std::to_string(threads_count));
    logger_->Print(LogLevel::DEBUG, "Фактическое кол-во
потоков: " + std::to_string(thread_pool_.size()));
}

/// @brief Устанавливает состояние потока с порядковым
номером.
/// @param[in] thread_num Порядковый номер потока.
/// @param[in] new_state Новое состояние потока.

```

```

void SetThreadState(int thread_num, ThreadState
new_state)
{
    std::unique_lock lock(state_mutex_);
    threads_state_[thread_num] = new_state;

    switch(new_state)
    {
        case ThreadState::FREE:
            logger_>Print(LogLevel::DEBUG, "Поток " +
std::to_string(thread_num) + " освобождён");
            break;

        case ThreadState::EXECUTE:
            logger_>Print(LogLevel::DEBUG, "Поток " +
std::to_string(thread_num) + " заблокирован");
            break;

        case ThreadState::EXCEPTION:
            logger_>Print(LogLevel::DEBUG, "В потоке "
+ std::to_string(thread_num) + " возникло исключение");
            break;
    }
}

/// @brief Запустить переданную функцию в потоке.
/// @tparam Type Тип рассматриваемого класса.
/// @tparam ...Args Аргументы функции класса.
/// @param[in] func Вызываемая в потоке функция класса.
/// @param[in] type Объект от которого происходит вызов
функции класса.
/// @param[in] settings Параметры улушений, применяемые
к файлу.
/// @param[in] source_file Путь к обрабатываемому файлу.
/// @param[in] result_file Путь к итоговому файлу.
/// @return true - задача запущена, false - все потоки
заняты.
template<typename Type, typename... Args>
bool LaunchThread(void (Type::*func)(Args...), Type*
type, const UpscaleSettings& settings, const std::string
source_path, const std::string result_path)
{
    std::unique_lock lock(state_mutex_);
    for(size_t t = 0; t < threads_state_.size(); t++)
    {
        if(ThreadState::FREE == threads_state_[t] ||
ThreadState::EXCEPTION == threads_state_[t])
        {
            logger_>Print(LogLevel::DEBUG, "Поток " +
std::to_string(t) + " запущен с задачей " + source_path);
            threads_state_[t] = ThreadState::EXECUTE;

```

```

        thread_pool_[t] =
std::make_shared<std::thread>(func, type, t, settings,
source_path, result_path);
        thread_pool_[t]->detach();
        return true;
    }
}
return false;
}

/// @brief Ожидать завершение всех потоков в пуле.
/// @param[in] wait_sec Время ожидания в секундах.
void WaitAllThreads(int wait_sec)
{
    bool complete = false;
    auto launch_time = std::chrono::system_clock::now();
    auto current_time =
std::chrono::system_clock::now();
    auto wait_time =
std::chrono::duration_cast<std::chrono::seconds>(current_time -
launch_time).count();
    logger_->Print(LogLevel::INFO, "Ожидание завершения
потоков...");

    // Ожидаем до тех пор, пока все потоки не
освободятся или не истечёт таймаут ожидания.
    do{
        std::unique_lock lock(state_mutex_);
        complete = true;

        // Поиск исполняемых потоков.
        for(auto t = 0; t < thread_pool_.size(); t++)
        {
            if(ThreadState::EXECUTE ==
threads_state_[t])
            {
                complete = false;
                break;
            }
        }
        lock.unlock();
        std::this_thread::sleep_for(100ms);

        current_time = std::chrono::system_clock::now();
        wait_time =
std::chrono::duration_cast<std::chrono::seconds>(current_time -
launch_time).count();

        // Если превышено время ожидания - выходим в
любом случае.
        if(wait_time > wait_sec)
        {

```

```

        logger_>Print(LogLevel::WARNING, "Истёк
таймаут ожидания завершения потоков (" +
std::to_string(wait_sec) + " сек).");
        return;
    }

    } while(!complete);

    logger_>Print(LogLevel::INFO, "Все потоки были
завершены");
}

private:
    // Пул потоков.
    std::vector<std::shared_ptr<std::thread>>
thread_pool_{};
    // Вектор состояний потоков.
    std::vector<ThreadState> threads_state_{};
    // Объект синхронизации для вектора состояний потоков.
    std::mutex state_mutex_{};
    // Указатель на используемый логгер.
    Logger* logger_{};
};

```

Файл logger.hpp

```

// Реализация универсального логера для демонов.
#pragma once

// Подключение библиотек C++.
#include <iostream>
#include <string>

/// @brief Уровни логирования.
enum LogLevel
{
    DEBUG = 0,
    INFO = 1,
    WARNING = 2,
    ERROR = 3,
    CRITICAL = 4
};

/// @brief Логер для индексирования событий.
class Logger
{
public:
    Logger(const std::string log_name = "unknown", LogLevel
log_level = LogLevel::INFO) : log_name_(log_name),
log_level_(log_level) {};
    ~Logger() {};

```

```

    /// @brief Установить уровень логирования.
    /// @param[in] log_level Уровень логирования.
    void SetLogLevel(LogLevel log_level) { log_level_ =
log_level; }

    /// @brief Установить имя компонента, использующего логер.
    /// @param log_name Имя компонента
    void SetLogName(const std::string log_name) { log_name_ =
log_name; }

    /// @brief Вывод лога.
    /// @param[in] log_level Уровень лога.
    /// @param[in] log_text Текст лога.
    void Print(LogLevel log_level, const std::string log_text)
    {
        // Если уровень не соответствует - не обрабатываем.
        if(log_level < log_level_) return;

        std::string str_log_level{};
        switch(log_level)
        {
            case LogLevel::DEBUG:
                str_log_level = "DEBUG";
                break;

            case LogLevel::INFO:
                str_log_level = "INFO";
                break;

            case LogLevel::WARNING:
                str_log_level = "WARNING";
                break;

            case LogLevel::ERROR:
                str_log_level = "ERROR";
                break;

            case LogLevel::CRITICAL:
                str_log_level = "CRITICAL";
                break;

            default:
                str_log_level = "UNKNOWN";
                break;

        }

        // TO DO: добавить "сборщик" строки и цвет вывода.
        std::cout << "[ " << log_name_ << " ] " << "[ " <<
str_log_level << " ] " << log_text << std::endl;
    }

```

Изм.	Лист	№ докум.	Подпись	Дата

```
private:
    // Имя компонента, использующего логер (компонента.
    std::string log_name_{};
    // Выводимый уровень логирования.
    LogLevel log_level_{};
};
```

Файл shares.hpp

```
// Общие константы, библиотеки, классы, используемые элементами.
#pragma once

/// @brief Параметры улучшения видео.
class UpscaleSettings
{
public:
    // Количество кадров в секунду.
    int fps_{};
    // Наименование используемой нейронной сети.
    std::string neural_{};
    // Размер видеофайла по горизонтали.
    int horizontal_{};
    // Размер видеофайла по вертикали.
    int vertical_{};
};
```

Файл CMakeLists.txt

```
cmake_minimum_required(VERSION 3.17)
project(makima)

add_subdirectory(audio_cutter)
add_subdirectory(chunks_cutter)
add_subdirectory(frames_cutter)
add_subdirectory(neuralizator)
add_subdirectory(frames_gluer)
add_subdirectory(chunks_gluer)
add_subdirectory(audio_gluer)

add_subdirectory(neural_test)
```

Файл main.cpp

```
#include "../daemon_interface.hpp"

int main()
{
    Neuralizator daemon{};
    return daemon.Launch();
}
```

Приложение В

(Обязательное)

Стартап-проект. Конвейер обработки видеофайлов для веб-приложения

Презентация

					ПГУ1.090301.15.001 ПЗ	Лист
						144
Изм.	Лист	№ докум.	Подпись	Дата		

Стартап-проект. Конвейер обработки видеофайлов для веб-приложения

Автор работы:
студент гр. 20ВВП1 Лукин Всеволод Дмитриевич
Научный руководитель:
к.т.н., доцент Федюнин Роман Николаевич

Пенза, 2024

Цель работы и объекты

Цель работы – разработать и реализовать программный комплекс, обрабатывающий видеофайлы в автоматическом режиме при помощи нейронных сетей по заранее созданному сценарию.

Объектом разработки является программный комплекс, позволяющий улучшать качество видеоконтента посредством использования заранее обученных моделей нейронных сетей. Комплекс должен обладать консольным интерфейсом администрирования, иметь возможность добавления новых моделей и их API в исходный код проекта без существенных затрат по времени, системой резервного сохранения и восстановления, возможностью установки настроек обработки видео.

Актуальность

Задача улучшения качества видеофайлов и изображений является актуальной по разным причинам:

- растущий спрос на высококачественный визуальный контент;
- распространение мобильных устройств и приложений для просмотра визуального контента;
- профессиональные и коммерческие нужды. Необходимость улучшения качества материалов для профессионалов в различных отраслях, таких как
 - а) фотосъёмка и видеосъёмка;
 - б) агентства по маркетингу и рекламе;
 - в) интернет-магазины;
 - г) медицинские и научные учреждения;
 - д) следственное дело;
 - е) личные и социальные потребности.

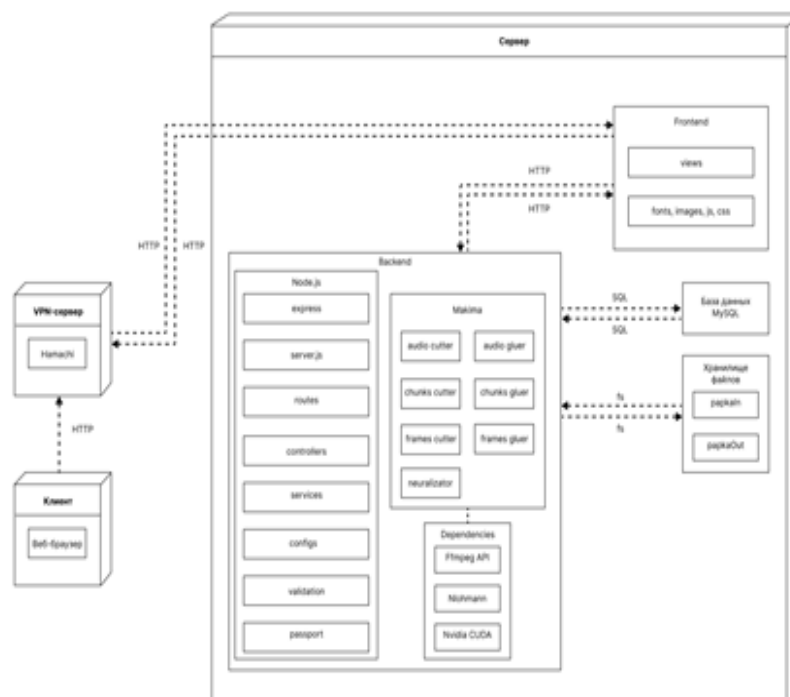
Сравнение аналогов

Сервис, использующий конвейер	Функционал	Удобство использования	Соотношение качество / время	Стоимость использования
Topaz Video Enhance AI	Повышение качества графического контента	Нагруженный интерфейс, имеется большое количество настроек.	Очень высокое качество / Большие затраты времени	От 30000р /год
Waifu2x-caffe	Повышение качества видео	Нагруженный интерфейс, небольшое количество функций	Высокое качество / Большие затраты времени	Бесплатно или от 700р /месяц
Bigjpg	Повышение качества изображений	Интерфейс понятен и удобен.	Высокое качество / Низкие затраты времени	Бесплатно или от 300р /месяц
Remini	Повышение качества графического контента	Нагруженный интерфейс, большое количество функций	Среднее качество / Большие затраты времени	Бесплатно или от 600р /месяц

Использованные технологии



Архитектура проекта



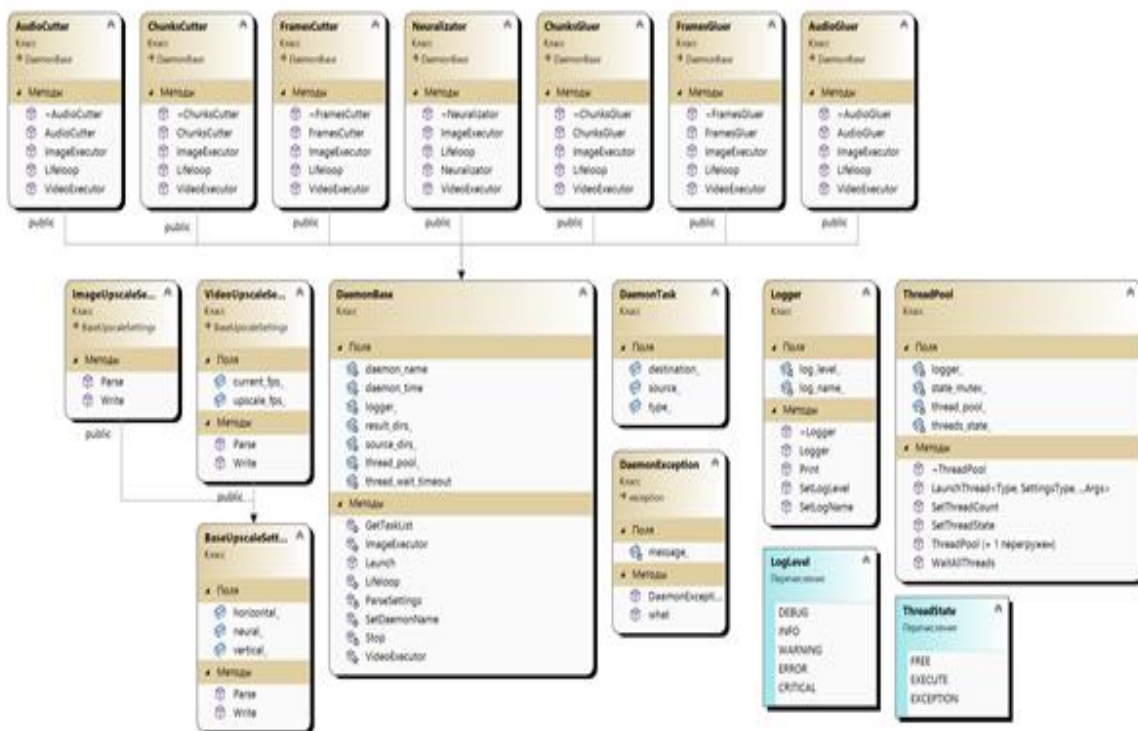
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.15.001 ПЗ

Лист

147

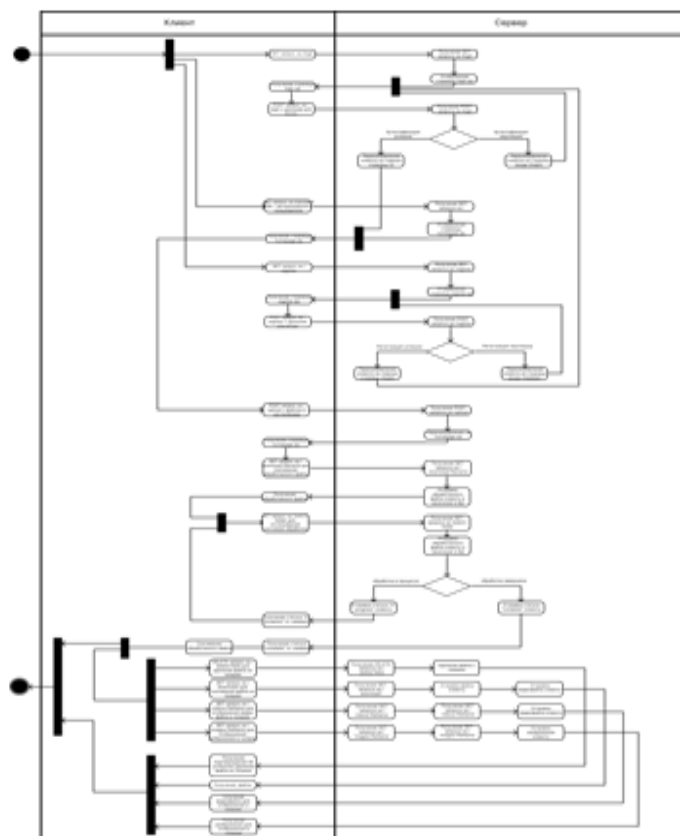
UML-диаграмма классов



UML-диаграмма вариантов использования



UML – диаграмма последовательности



UML – диаграмма последовательности



Тестирование программного комплекса



Результаты работы

Разработка конвейера обработки видеофайлов для веб-приложения была успешно завершена. Реализованы все запрашиваемые элементы, описанные в функциональных требованиях к конвейеру. Удачное тестирование программного комплекса также подтверждает работоспособность и состоятельность данного продукта. В процессе написания данной работы также было изучено множество механизмов оптимизации высоконагруженных приложений, изучено API библиотеки ffmpeg и API определённого пула нейронных сетей, что положительно сказалось на развитии навыков программирования.

Дальнейшее развитие данного проекта планируется в рамках стартапа. Материальные средства позволят использовать больше аппаратных ресурсов для разворачивания и масштабирования конвейера. Продолжение модернизации способствует увеличению коэффициента соотношения качества преобразования к временным затратам на одну выполняемую задачу. В свою очередь, это позволит предоставить разработанный продукт массовому пользователю через организацию полноценного веб-сервиса и поспособствует популяризации инструмента для простого и понятного разворачивания нейронных сетей.

					ПГУ1.090301.15.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		151

МИНОБРАЗОВАНИЯ РОССИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра "Вычислительная техника"

УТВЕРЖДАЮ

Заведующий кафедрой

Митрохин М.А.

(Подпись)

(Фамилия, инициалы)

«19» апреля

2024 г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ В ФОРМЕ СТАРТАПА

Студент Дукин Всеволод Дмитриевич

Направление подготовки 09.03.01 Информатика и вычислительная техника

Профиль Программное обеспечение средств вычислительной техники и автоматизированных систем

Форма обучения Очная

1. Тема: Стартап-проект «Конвейер обработки видеофайлов для веб-приложения». Конвейер обработки видеофайлов для веб-приложения является программным комплексом, обеспечивающим непрерывную обработку поступающего графического контента.

Тема одобрена экспертной комиссией и утверждена приказом ректора от «19» апреля 2024 г. № 405/О.

2. Срок представления работы к защите «10» июня 2024 г.

3. Общая часть ВКР:

Подготовка компьютерной презентации бизнес-проекта (стартапа).

4. Индивидуальная часть ВКР:

4.1. Исходные данные:

Разработать конвейер обработки видеофайлов для веб-приложения. Составить бизнес-модель и финансовый план. Язык программирования C, C++, Bash, среда разработки Visual Studio. Программа должна использовать следующие технологии:

1. C, C++, Bash
2. Ffmpeg API Neural Networks API
3. Nlohmann JSON
4. Многопоточное программирование, ООП
5. Docker
6. CUDA Nvidia, Vulkan

4.2. Содержание пояснительной записки ВКР (по разделам):

Анализ предметной области, обоснование актуальности разработки, обзор аналогов, анализ требований к разработке, постановка задачи, выбор инструментов разработки, архитектура приложения, проектирование и разработка, тестирование и отладка, БЖД, бизнес-план: финансовые и временные ограничения на проект (бюджет, сроки реализации, ожидаемые затраты и доходы и т.д.).

5. Перечень графического материала: UML-диаграммы (диаграмма классов, диаграмма развертывания, диаграмма деятельности, диаграмма вариантов использования, диаграмма последовательности)

Руководитель ВКР

Задание принял к исполнению

(подпись, дата)

(Ф.И.О.)

Экспертное заключение
о согласовании темы ВКР в форме стартапа

Экспертная комиссия в составе декан ФВТ д.т.н., профессор Л.Р. Фионова, доцент кафедры ВТ, к.э.н., И.В. Акифьев, главный разработчик ООО «Ай-теко ВЦ», к.т.н. М.П. Синев, рассмотрев бизнес-проект, стартап-проект “Конвейер обработки видеофайлов для веб-приложения” Лукина Всеволода Дмитриевича подтверждает его актуальность, практическую значимость и возможность коммерциализации.

Решение экспертной комиссии: выпускная квалификационная работа на тему “Стартап-проект. Конвейер обработки видеофайлов для веб-приложения.” Лукина Всеволода Дмитриевича может выполняться в форме стартапа (бизнес-проекта).

Председатель ЭК

д.т.н., профессор
декан ФВТ



Л.Р. Фионова

Члены ЭК

к.э.н., доцент кафедры ВТ



И.В. Акифьев

к.т.н., главный разработчик
ООО «Ай-теко ВЦ»



М.П. Синев

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.15.001 ПЗ

Лист

153