

Федеральное государственное бюджетное образовательное учреждение высшего образования

Факультет

Καφεδρα

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

на тему

Студент

31.05.2024
(подпись, дата)

Верховский М.В.

Руководитель

31.05.2024
(подпись, дата)

Федюнин Р.Н.

Нормоконтролёр

 31.05.2024
(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

10.06.2024г. № 16)

Секретарь ГЭК

(подпись)

Гурин Е.И.

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

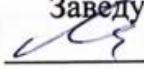
вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

 М.А. Митрохин

«19» апреля 2024 г.

ЗАДАНИЕ НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Верховский Михаил Владимирович гр. 20ВВП1 факультета Вычислительной техники.

2. Тема работы: «Стартап-проект. Веб-приложение улучшения качества видео»

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.

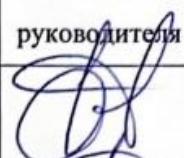
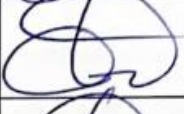
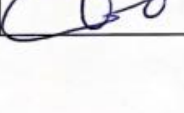
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать и реализовать программное обеспечение, предназначенное для выбора и загрузки видео/изображений в конвейер обработки, а также получения результата в виде нового видеофайла или изображения с улучшенными характеристиками. Нужно реализовать проект в виде веб-приложения, что включает в себя проектирование, а также создание клиентской и серверной части. При реализации требуется использовать базу данных для хранения и управления информацией.

5. Перечень подлежащих разработке вопросов

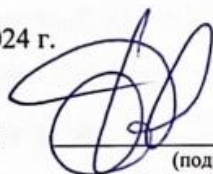
Требуется сделать анализ предметных областей, а именно: изучить существующие решения, которые имеют аналогичный принцип работы, выделить их достоинства и недостатки. Также нужно выбрать язык программирования и другие компоненты разработки, подходящие для оптимальной реализации задачи. Определить архитектуру веб-приложения и описать его внутреннюю организацию. Изучить имеющийся функционал и возможности инструментов и библиотек, необходимых для реализации веб-приложения.

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 инструментов, применяемых при проектировании и разработке	30	01.05.2024 – 03.05.2024	
4	Разработка архитектуры проекта	16	04.05.2024 – 5.05.2024	
5	Разработка клиентской части веб-приложения	82	6.05.2024 – 16.05.2024	
6	Разработка серверной части веб-приложения	40	16.05.2024 – 19.05.2024	
7	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)

Федюнин Р.Н.

Задание к исполнению принял студент



(подпись, дата)

Верховский М.В.

Работу к защите допустить
Декан факультета



05.06.2024

(подпись, дата)

Фионова Л.Р.

Реферат

Бакалаврская работа содержит 210 листов, 106 рисунков, 1 формула, 9 таблиц, 7 источников и 3 приложения

ВЕБ-ПРИЛОЖЕНИЕ, JAVA SCRIPT, HTML, CSS, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, FRONTEND, BACKEND, NODEJS, JSON, ЗАПРОСЫ, МАРШРУТЫ, БАЗА ДАННЫХ, MYSQL, VPN, VISUAL STUDIO CODE

Цель работы – разработать программное обеспечение, выполняющее роль посредника между клиентом и конвейером обработки графического контента.

Объектом разработки является веб-приложение, которое необходимо для выбора и загрузки видео и изображений в конвейер обработки, а также получения результата в виде нового видеофайла или изображения с улучшенными характеристиками. Итоговое веб-приложение должно содержать пользовательский интерфейс, должно быть спроектировано с применением клиент-серверной архитектуры и должно состоять из трёх связанных между собой частей: клиентская часть, серверная часть и база данных.

Разработка проводилась в среде программирования Visual Studio Code. В качестве языка программирования был выбран Java Script.

В итоге реализовано веб-приложение, соответствующее установленным требованиям: удобный и понятный пользовательский интерфейс, возможность загрузки видео и изображений на сервер для улучшения, а также получения результата. Клиентская и серверная часть взаимодействуют друг с другом с помощью REST API. Клиентская часть веб-приложения включает в себя страницу входа, страницу регистрации и главную страницу. Серверная часть реализована на основе Node.js, также использован фреймворк Express.js. База данных на основе MySQL используется для хранения информации и обладает инструментарием для работы по системе CRUD.

					ПГУ1.090301.04.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Верховский М.В.		31.05.24	Стартап-проект. Веб-приложение улучшения качества видео	Лит.	Лист	Листов
Провер.		Федюнин Р.Н.		31.05.24			2	210
Н. Контр		Бычков А.С.		31.05.24		ФВТ, каф. ВТ гр. 20ВВП1		
Утверд.		Митрохин М.А.		31.05.24	Пояснительная записка			

Содержание

Перечень сокращений и обозначений	6
Введение	8
1 Стартап-проект. Веб-приложение улучшения качества видео	10
1.1 Актуальность	10
1.2 Пользователи и сценарии использования	13
1.3 Анализ аналогов	15
1.3.1 Topaz Video Enhance AI	16
1.3.2 Waifu2x-caffe	20
1.3.3 Bigjpg	23
1.3.4 Remini	25
1.4 Постановка задачи	28
1.4.1 Требования к пользовательскому интерфейсу	29
1.4.2 Функциональные требования	30
1.4.3 Технические требования	31
1.5 Выбор инструментов разработки	35
1.5.1 Выбор языка программирования	35
1.5.2 Выбор инструментов для реализации графического интерфейса в веб-приложении	38
1.5.3 Выбор среды разработки	42
1.5.4 Выбор инструментов для реализации веб-сервера	43
1.5.5 Выбор СУБД	46
1.5.6 Выбор инструментов для работы с файлами	46
1.5.7 Выбор инструментов для аутентификации и безопасности в веб-приложении	50

1.5.8	Выбор VPN для создания локальной сети.....	51
1.6	Архитектура проекта	53
1.6.1	Выбор архитектуры веб-сервиса	53
1.6.2	Архитектура веб-приложения.....	58
1.7	Разработка веб-приложения.....	73
1.7.1	Настройка VPN.....	73
1.7.2	Разработка клиентской части веб-приложения.....	79
1.7.3	Разработка серверной части веб-приложения.....	87
1.7.4	Отношения между сущностями проекта	99
1.7.5	Разработка БД.....	100
1.8	Тестирование веб-приложения	109
2	Охрана труда.....	135
2.1	Анализ негативных факторов, воздействующих на оператора ПЭВМ и способы их устранения.....	135
2.1.1	Оборудование рабочего места	135
2.1.2	Освещенность рабочего места	137
2.1.3	Электробезопасность	137
2.1.4	Пожарная безопасность	138
2.1.5	Режим труда и отдыха	139
3	Стартап проект	141
3.1	Сегменты ЦА	142
3.2	Ключевые процессы.....	147
3.3	Каналы взаимодействия	149
3.4	Источники дохода	150
3.5	Ключевые ресурсы	152

3.6 Риски проекта	154
Заключение	155
Список использованных источников	156
Приложение А – UML-диаграммы.....	157
Приложение А.1 – UML-диаграмма вариантов использования	158
Приложение А.2 – UML-диаграмма последовательностей	159
Приложение А.3 – UML-диаграмма компонентов	160
Приложение А.4 – UML-диаграмма развёртывания	161
Приложение А.5 – UML-диаграмма деятельности.....	162
Приложение Б – Листинг программы	163
Приложение В – Презентация.....	201

Перечень сокращений и обозначений

В данной работе применяются следующие сокращения и обозначения

БД – база данных

ПК – персональный компьютер, предназначенный для индивидуального использования

СУБД – система управления базами данных

ЯП – язык программирования

API (Application Programming Interface) – интерфейс программирования приложений, определяющий способы взаимодействия компьютерных систем

CRUD (Create, Read, Update, Delete) – аббревиатура, которая описывает четыре основные операции над данными

IDE (Integrated Development Environment) – интегрированная среда разработки

JSON (JavaScript Object Notation) – текстовый формат для обмена данными, основанный на синтаксисе JavaScript

Landing Page – веб-страница, которая нужна для привлечения и конвертации посетителей в целевые действия

MVC (Model-View-Controller) – архитектурный паттерн, который разделяет приложение на три отдельных компонента: модель, представление и контроллер

MVP (Minimum Viable Product) – базовая версия приложения, содержащая только основные функции для быстрого тестирования и получения пользовательской обратной связи

Responsive Design – подход к веб-разработке, при котором веб-страницы адаптируются и оптимизируются для отображения на различных устройствах и экранах

REST API (Representational State Transfer Application Programming Interface) – стиль архитектуры программных интерфейсов, который основан на передаче репрезентативного состояния ресурса между клиентом и сервером через стандартные HTTP-методы

					ПГУ1.090301.04.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

SQL (Structured Query Language) – язык управления реляционными базами данных

UML (Unified Modeling Language) – язык визуального моделирования для проектирования программных систем

URL (Uniform Resource Locator) – адрес, используемый для идентификации и локализации ресурсов в Интернете

FPS (Frames Per Second) – единица измерения частоты кадров, например, в видео, которая показывает количество отображаемых кадров в секунду

					ПГУ1.090301.04.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

На данный момент в современном мире среди огромного числа людей широко распространена потребность и нужда в улучшении и усовершенствовании характеристик разнообразного графического контента, будь то видео или изображения. Это связано с глобальным развитием социальных сетей, онлайн-платформ и цифровой коммуникации, что к сегодняшнему моменту привело к значительному увеличению спроса на высококачественный визуальный контент. Также растущее применение мобильных устройств привело к увеличению объёма создаваемого визуального контента, который нуждается в обработке и улучшении. Люди хотят производить и потреблять приятные глазу изображения и видео, а плохое качество может негативно повлиять на их восприятие контента.

Довольно стремительный прогресс в области искусственного интеллекта, машинного обучения и компьютерного зрения дал ценную возможность улучшения качества изображений и видео с беспрецедентной эффективностью и впечатляющей детализацией.

Потребность в повышении качества графического и визуального контента особенно актуальна для многих профессиональных сфер деятельности, таких как: фотография, видеопроизводство, маркетинг, электронная коммерция, медицина, криминалистика и личные нужды. Это создаёт большой спрос на эффективные и доступные решения для улучшения качества изображений и видео.

Выбор формы реализации приложения по улучшению качества видео и изображений с помощью конвейера обработки, использующего нейросети, в виде веб-приложения объясняется его независимостью от аппаратной платформы, кроссплатформенной совместимостью, простотой использования, возможностью удалённого администрирования, эффективным расходом ресурсов, а также экономичностью, что делает его оптимальным решением для

					ПГУ1.090301.04.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

обработки медиафайлов пользователями из любой точки мира с любых устройств.

По перечисленным выше причинам «Стартап-проект. Веб-приложение улучшения качества видео» становится очень важным в данном контексте.

Целями выпускной работы является:

- исследование предметной области и анализ существующих решений;
- разработка веб-приложения для улучшения качества видео и изображений, что включает в себя проектирование и разработку удобного пользовательского интерфейса, связанных друг с другом клиентской и серверной частей, а также базу данных;
- тестирование веб-приложения;
- внедрение и развёртывание веб-приложения.

Веб-приложение улучшения качества видео является удобным и эффективным инструментом для улучшения качества графического контента с помощью конвейера обработки, использующего современные интеллектуальные технологии на основе нейронных сетей. Приложение позволит повысить следующие характеристики поступающего визуального контента: разрешение, количество кадров в секунду (для видеофайлов), чёткость, контрастность, а также качество цветовой палитры. Улучшение перечисленных характеристик позволит удовлетворить запрос пользователя в высоком уровне качества графического контента.

Проект поможет 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.

1 Стартап-проект. Веб-приложение улучшения качества видео

1.1 Актуальность

Исследование темы «Стартап-проект. Веб-приложение улучшения качества видео» имеет существенную важность и актуальность. Это связано с возрастающим спросом на качественный визуальный контент высокого уровня детализации, и, соответственно, на сервисы, которые удовлетворяют эти потребности. В основе таких сервисов лежат многофункциональные нейронные сети, что делает их доминирующими на современном повышении качества файлов разного формата. Количество сервисов для предоставления подобных услуг продолжает неуклонно расти, что создаёт потребность в новых качественных и эффективных решениях.

Задача улучшения качества видео и изображений является актуальной по разным причинам:

– растущий спрос на высококачественный визуальный контент. Глобальное развитие социальных сетей, онлайн-платформ и цифровой коммуникации к сегодняшнему моменту привело к значительному увеличению спроса на высококачественный визуальный контент;

– прогресс в технологиях. Стремительный прогресс в области искусственного интеллекта, машинного обучения, а также компьютерного зрения дал ценную возможность улучшения качества изображений и видео с беспрецедентной эффективностью и впечатляющей детализацией;

– распространение мобильных устройств. Растущее повсеместное применение мобильных устройств привело к увеличению объёма создаваемого визуального контента, который нуждается в обработке и улучшении, ведь довольно ограниченные возможности таких устройств часто приводят к низкому

					ПГУ1.090301.04.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

качеству производимых с их помощью фотографий и видеозаписей, которые требуют дальнейшей обработки и улучшения их характеристик;

– коммерческие и профессиональные нужды. Профессионалы в различных отраслях, таких как:

1) фотосъёмка и видеосъёмка. Нуждаются в улучшении своей работы, чтобы соответствовать ожиданиям клиентов и оставаться конкурентоспособными на рынке;

2) агентства по маркетингу и рекламе. Требуют высококачественного визуального контента, чтобы эффективно продвигать продукты и услуги;

3) интернет-магазины. Нуждаются в демонстрации продуктов в лучшем виде, чтобы увеличить продажи и удовлетворенность клиентов;

4) медицинские и научные учреждения. Могут нуждаться в улучшении медицинских изображений или видео для диагностических или исследовательских целей;

5) следственное дело. Нуждаются в получении достоверной доказательной базы путём улучшения материалов с камер видеонаблюдения;

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

Веб-приложение является важной частью веб-сервиса улучшения качества видео, т.к. выполняет роль посредника между клиентом и конвейером обработки графического контента. Оно предоставляет пользователю удобный интерфейс для работы. Веб-приложение позволяет выбирать видео и изображения, а также самостоятельно задавать параметры для обработки, которые представляются в формате JSON-файла, для дальнейшей загрузки в конвейер, который улучшает качество графического контента с помощью современных интеллектуальных технологий на основе нейронных сетей. Также оно позволяет отслеживать

состояние обработки на сервере. Для хранения различных данных используется БД.

Веб-сервис улучшения качества видео состоит из, собственно, веб-приложения, а также конвейера обработки видеопотока. В данной работе рассматривается разработка именно веб-приложения.

Предпочтительность разработки приложения в веб-формате обусловлена рядом факторов:

- независимость от аппаратной платформы. Одно из основных преимуществ веб-приложений заключается в том, что они могут использоваться на широком спектре устройств, от настольных компьютеров до мобильных телефонов, без необходимости в установке дополнительного программного обеспечения [1]. Чего нельзя сказать, например, о Java-интерфейсе, требующем JVM. Это позволяет запускать приложение с любого компьютера без предварительной установки специального программного обеспечения;

- кроссплатформенная совместимость. Веб-приложение не зависит от операционной системы, браузера или устройства, что гарантирует доступность для пользователей с различными конфигурациями;

- простота использования. Интуитивно понятный интерфейс веб-приложения позволяет пользователям легко загружать файлы, выбирать параметры обработки и скачивать улучшенный медиа-контент;

- удалённое администрирование. Возможность проведения обработки на отдельном компьютере. Интерфейс не зависит от конкретной машины. Выбор файла и настроек для обработки происходит в веб-интерфейсе;

- эффективное расходование ресурсов: в случае разработки приложения на Java или C++, значительные ресурсы системы будут использоваться на создание графического отображения. В свою очередь, веб-интерфейс является простой веб-страничкой, которую мы отправляем пользователю для отображения, т.е. не нужно тратить дополнительные ресурсы на её отрисовку;

– экономичность. Использование веб-приложения не предполагает покупку и установку программного обеспечения. Это повышает доступность сервиса.

Итак, веб-приложение улучшения качества видео представляет из себя эффективный инструмент, обеспечивающий удобное взаимодействие со сторонним конвейером, использующим современные интеллектуальные технологии на основе нейронных сетей для обработки графического контента, и получение результатов. Такое приложение позволит выбирать для улучшения следующие характеристики поступающего контента: разрешение, количество кадров в секунду (для видеофайлов), чёткость, контрастность и качество цветовой палитры. Работа над улучшением перечисленных характеристик позволяет удовлетворить запрос пользователя в высоком уровне качества визуального контента.

Веб-приложение позволяет пользователям использовать нейросеть без необходимости установки дополнительного программного обеспечения, тем самым обеспечивая доступность и удобство. Также оно включает в себя разнообразные элементы пользовательского интерфейса, делающие процесс работы с нейросетью интуитивно понятным.

Проект поможет 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.

1.2 Пользователи и сценарии использования

Пользователи веб-приложения улучшения качества видео могут быть разнообразными, включая как опытных веб-программистов, так и пользователей без специальных знаний по этой теме.

Далее будут представлены пользовательские сценарии.

Для исполнителей проектов:

					ПГУ1.090301.04.001 ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

– скачивание репозитория с кодом. Программист может скачать репозиторий с кодом веб-приложения обработки графического контента с помощью команды `git clone` или просто с помощью простого скачивания архива с файлами проекта;

– встраивание и настройка проекта. После скачивания репозитория программист должен настроить проект;

– запуск кода. После настройки проекта программист может запустить код веб-приложения, чтобы проверить его на работоспособность;

– тестирование и анализ работы кода. Можно провести тестирование кода, выполнив загрузку различных файлов и проанализировав результаты работы кода. Это позволяет сделать оценку качества работы веб-приложения, а также в последующем оптимизировать её;

– использование кода в собственных проектах. Программист может использовать код данного веб-приложения в своих собственных проектах, интегрировав его туда.

Кроме исполнителей проектов, данным веб-приложением могут пользоваться и обычные посетители веб-сайта.

Для обычных пользователей, которые хотят улучшить характеристики видео или изображения, взаимодействие с веб-приложением будет заключаться в следующем:

– открытие по адресу. Пользователь переходит на веб-сайт, на котором он может воспользоваться всем функционалом сервиса для улучшения качества изображений и видеофайлов;

– регистрация. Пользователь регистрируется на веб-сайте, где вводит своё имя, email, пароль;

– вход в аккаунт. Пользователь входит в свой аккаунт по email и паролю;

– выбор изображения или видеофайла для последующего улучшения.

Пользователь выбирает файл, который нужно улучшить;

– выбор параметров обработки видео/изображения. После выбора видео/изображения пользователь может самостоятельно выбрать нужные параметры для обработки, такие как разрешение, частота кадров (для видеофайлов), а также предобученная модель нейронной сети;

– отправка изображения или видеофайла для последующего улучшения. Пользователь загружает файл, который нужно улучшить, на сервер веб-сервиса;

– ожидание результата обработки. После загрузки видеофайла пользователь ожидает завершения обработки файла с помощью конвейера;

– загрузка полученного результата. После завершения обработки пользователь может загрузить улучшенный видеофайл для просмотра или дальнейшего использования;

– просмотр истории обработки в галерее на веб-странице. Пользователь может выставлять фильтры для просмотра, а также скачивать и удалять файлы из своей истории обработки.

Общие сценарии:

– пользователь хочет обработать некоторое количество видеофайлов за оптимальное время, используя определённую нейронную сеть;

– веб-приложение должно предоставлять функционал для обработки видеофайлов и изображений на конвейере обработки графического контента с использованием нейронных технологий в соответствии с UML-диаграммой вариантов использования, описанной в пункте 1.6.

1.3 Анализ аналогов

В рамках дипломной работы можно 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.

В таблице 1 приведено сравнение аналогов по критериям.

					ПГУ1.090301.04.001 ПЗ	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

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 нейронных сетей. Интерфейс Topaz Video Enhance AI показан на рисунке 1. Topaz Video Enhance AI имеет большой диапазон настроек и поддерживает множество форматов. Однако данный сервис имеет некоторое количество недостатков, понижающих значимость описанных выше плюсов.

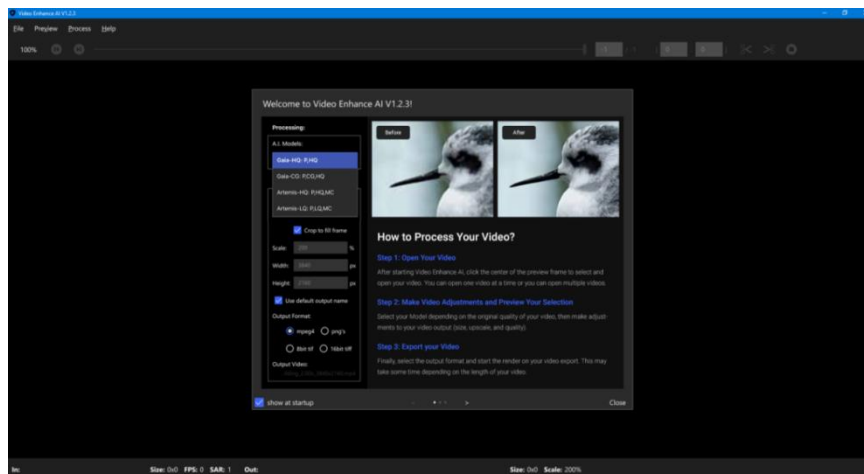


Рисунок 1 – Интерфейс Toraz Video Enhance AI

Далее представлен анализ продукта.

Функциональность сервиса. Toraz Video Enhance AI позволяет улучшить параметры видеозаписей или изображений до достаточно высоких значений. Максимальные показатели параметров при улучшении:

- разрешение видео: 4096x3072;
- количество кадров: 60;
- 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 Toraz Video Enhance AI достаточно нагружен из-за значительного количества настроек и может быть непонятным для рядового пользователя. Однако у сервиса есть функция поиска необходимых настроек и инструментов по ключевым словам. 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 труднсовместимых 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
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, можно сделать вывод, что Topaz 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 выборки нейронных сетей. Интерфейс Waifu2x-caffe показан на рисунке 2.

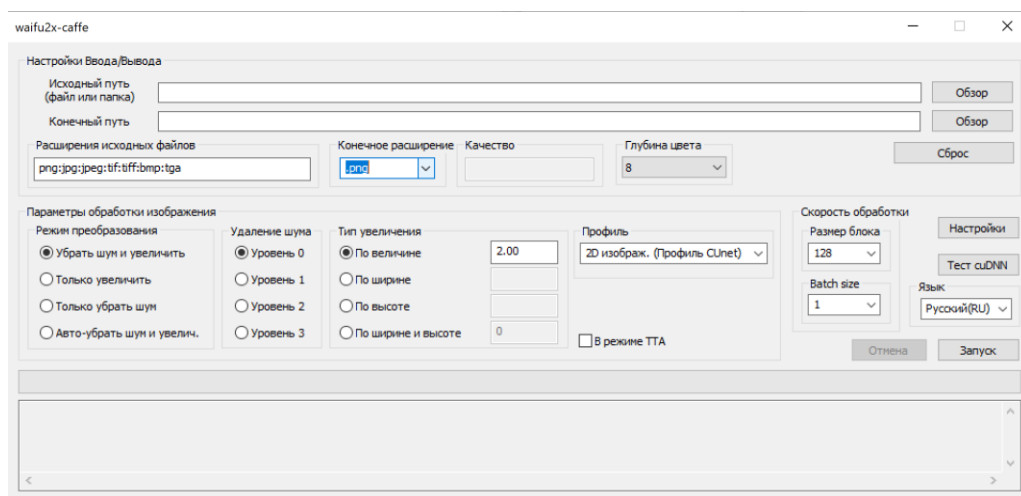


Рисунок 2 – Интерфейс Waifu2x-caffe

Далее представлен анализ продукта.

Функциональность 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. Поддерживаются 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 труднсовместимых параметров приводит к оптимизации скорости работы данного программного продукта, однако временные затраты продолжают оставаться значительными при повышении качества. Использование аппаратных ресурсов является умеренным, но продолжает оказывать влияние на показатель соотношения.

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. Данная версия имеет ограничения, не позволяющие использовать данный продукт полноценно:

– 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;

– 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, можно сделать вывод, что 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 пулом параметров и функций повышения качества видео контента, однако имеет достаточно запутанный интерфейс, что может оттолкнуть рядового пользователя.

1.3.3 Bigjpg

Bigjpg – это сервис улучшения изображений без потери качества при их увеличении, который в своей работе пользуется искусственным интеллектом. Он предоставляет пользователям возможность улучшать детализацию и чёткость изображений, что особенно полезно при работе с изображениями низкого разрешения. Интерфейс и веб-интерфейс Bigjpg показаны на рисунках 3, 4.

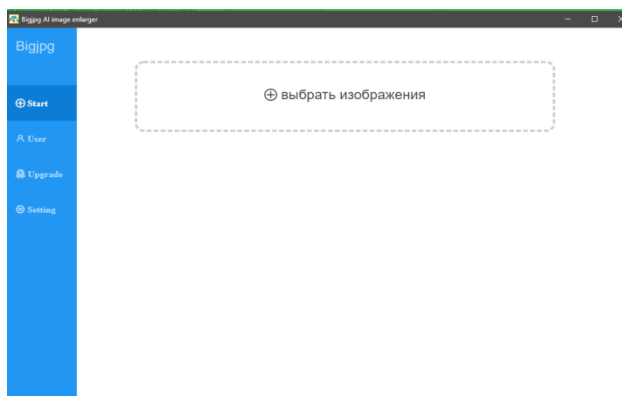


Рисунок 3 – Интерфейс Bigjpg

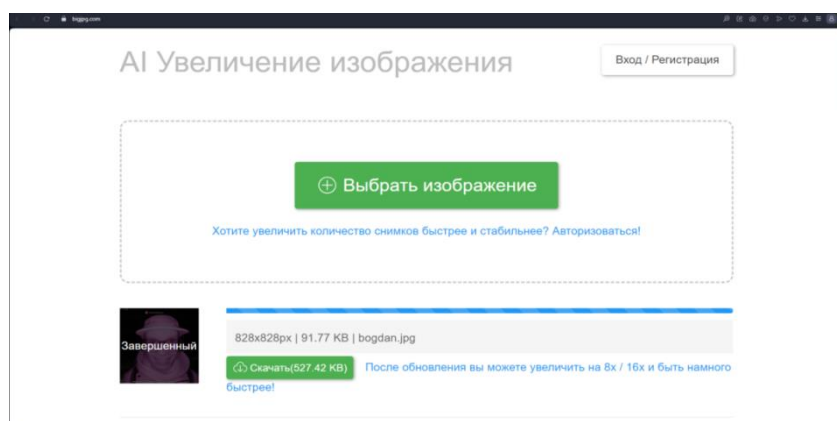


Рисунок 4 – Веб-интерфейс Bigjpg

Далее представлен анализ продукта.

Функциональность сервиса. Bigjpg позволяет увеличить размер картинки до 4 раз, не теряя при этом качества и не искажая исходник. Алгоритмы глубокого обучения, используемые этим приложением, занимаются анализом и улучшением текстур и границ объектов внутри изображения, убирают лесенки из пикселей, увеличивая резкость.

Удобство использования. Интерфейс пользователя сервиса удобен. От пользователя требуется лишь загрузить изображение и параметры увеличения через сайт или через приложение. Данная программа относительно быстро занимается обработкой, после чего появляется ссылка для скачивания результата.

Соотношение качества преобразования к временным затратам. Сервис выделяется хорошим соотношением скорости и качества. Программа может быстро увеличить изображения без существенных мелких деталей, что особенно важно для людей, работающих с графическим контентом.

Стоимость использования. Bigjpg предлагает как бесплатные, так и платные услуги. Пользуясь сервисом бесплатно, пользователь имеет ограничения на количество и масштаб изображений. За деньги приложение убирает некоторые ограничения и предоставляет более высокий приоритет обработки. Стоимость платных подписок разнится, так что пользователи могут выбрать наиболее подходящий для них план. Стоимость услуг Bigjpg показана на рисунке 5.

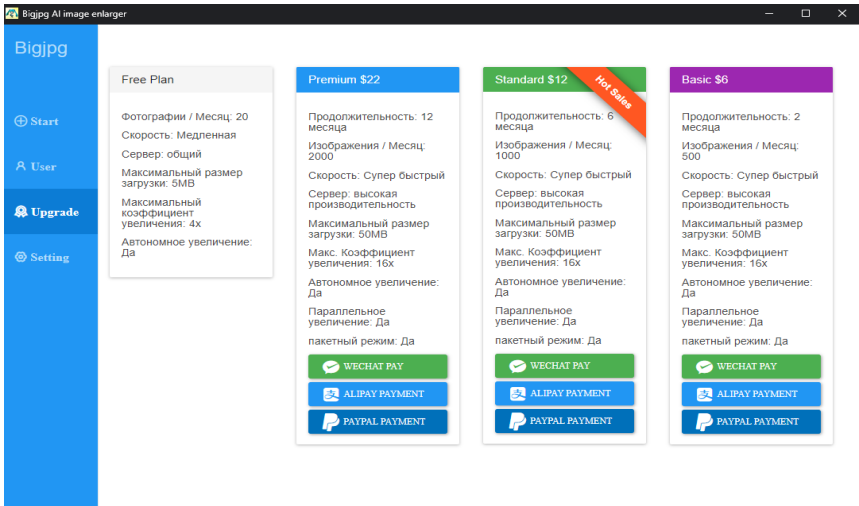


Рисунок 5 – Стоимость услуг Bigjpg

Проведя анализ сервиса Bigjpg, можно сказать, что он, конечно, имеет некоторые преимущества. Это бесплатный доступ к качественному AI-увеличению изображений и доступные цен на подписки, наличие десктопного

приложения и веб-приложения, что добавляет удобства в процессе работы с изображениями. Алгоритмы Bigjpg могут генерировать картинки, которые выглядят естественно даже после кратного увеличения их размера.

Недостатки тоже присутствуют. Всё-таки бесплатные обработки нельзя назвать полноценными. Они могут не справляться с важными мелкими деталями и плохим качеством исходников. Сам сервис требует интернет-соединение и из-за хранения данных на сервере может вызвать опасения приватности. Подписки, расширяющие функционал, нужны, скорее, для продвинутых задач. Долгое время обработки может зависеть от общей нагрузки на Bigjpg.

1.3.4 Remini

Remini – это приложение по обработке фотографий любого качества с помощью технологий искусственного интеллекта. Remini полезно для повышения чёткости, детализации изображений, восстановления старых или поврежденных фотографий. Веб-интерфейс Remini показан на рисунке 6.

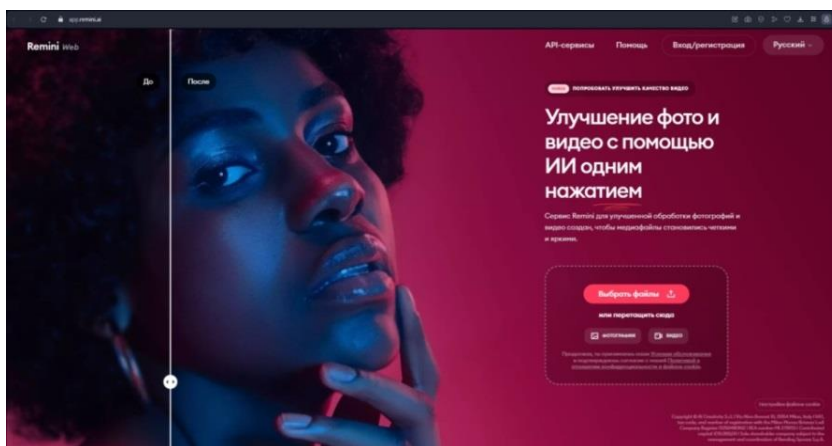


Рисунок 6 – Веб-интерфейс Remini

Далее представлен анализ продукта.

Функциональность сервиса. Remini для реализации своего функционала использует искусственный интеллект. Приложение улучшает характеристики фотографий, повышая их детализацию, резкость, разрешение и устраняя размытость.

Удобство использования. Сервис достаточно удобен, ведь обладает современным качественным дизайном сайта, что точно выделяет его среди конкурентов. Это даёт пользователям возможность быстро и легко ускорить процесс обработки. На сайте всего лишь нужно загрузить фотографию или видео, а Remini автоматически начнёт обработку, предоставив в конце улучшенный вариант. Обработка изображений в Remini показано на рисунке 7.

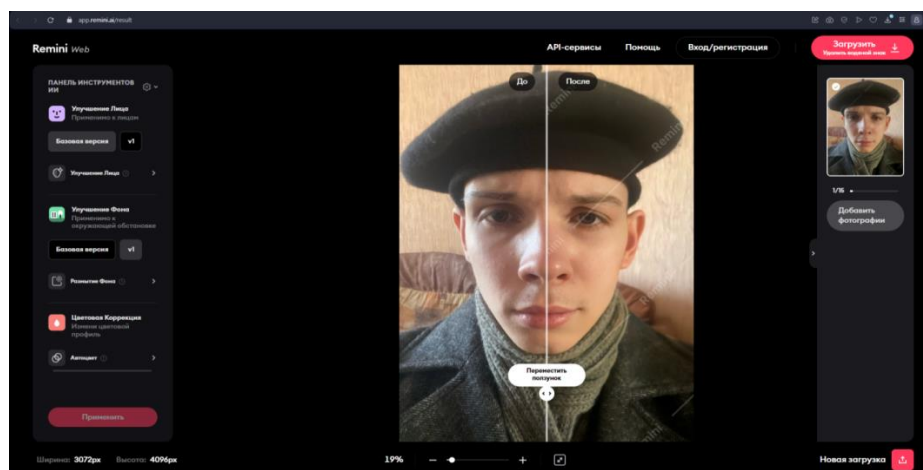


Рисунок 7 – Обработка изображений в Remini

Соотношение качества преобразования к временным затратам. Если проводить замеры, то Remini довольно быстро обрабатывает фотографии, но, конечно, скорость зависит и от размера файла, и от 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. Есть ограниченную бесплатная версия, которая даёт улучшить лишь небольшое количество изображений в день. Платная подписка обеспечивает безлимитный доступ ко всем функциям, недоступным в бесплатной версии, а также быстрее продвигает по очереди среди пользователей. Стоимость услуг Remini показано на рисунке 8.

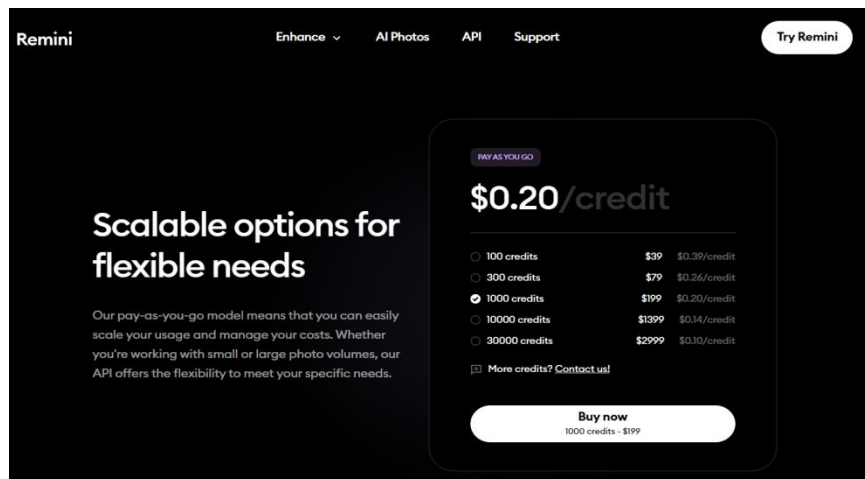


Рисунок 8 – Стоимость услуг Remini

Веб-сервис улучшения качества видео, предназначенный для улучшения качества видео через веб-интерфейс, имеет как сходства, так и преимущества над аналогами.

Веб-приложение имеет некоторое количество сходств, при сравнении с аналогами:

- интуитивно понятный веб-интерфейс. Пользователи могут легко взаимодействовать с нейросетью для улучшения видео без необходимости знания технических деталей процесса;

- имеет все перечисленные настройки из функционала контрагентов;

- имеет функцию предварительного просмотра результата.

Для того, чтобы проект был конкурентоспособен в условиях постоянного развития и появления подобных сервисов на рынке, был добавлен следующий уникальный функционал:

- интерфейс, который позволяет сравнить результат обработки видео или изображения с тем, что было изначально;

- поддержка большинства доступных форматов видео и изображений;

- возможность делать выбор предобученной модели нейронной сети для обработки определённых типов файлов;

- возможность просмотра на веб-странице галереи ранее обработанных пользователем изображений и видео;
- наличие бесплатной версии с доступным функционалом;
- поддержка работы на слабых системах.

Все приведённые выше функции и качества делают разрабатываемый веб-сервис улучшения качества видео и изображений конкурентоспособным и весьма эффективным инструментом для улучшения качества визуального контента и могут обеспечить ему успешное существование на рынке подобных решений.

1.4 Постановка задачи

Целью этого дипломного проекта является проектирование и разработка веб-приложения, которое выполняет роль посредника между клиентом и конвейером обработки графического контента. Главная задача этого приложения – предоставлять пользователям возможность загружать свои изображения и видео, а также самостоятельно задавать все необходимые параметры (разрешение, частота кадров (для видео) и выбор предобученной модели нейронной сети) для улучшения качества этих файлов. Выбранные параметры обработки будут сохраняться в виде JSON-файлов, которые в дальнейшем будут использоваться для улучшения контента с помощью конвейера обработки.

То есть данное приложение должно обеспечивать удобное взаимодействие с конвейером для обработки загруженных файлов на сервере и предоставлять возможность скачивания обработанных файлов. Для этого также должно быть реализовано отслеживание состояния обработки на сервере.

Проект нужно реализовать в виде веб-приложения. Это включает в себя проектирование и разработку пользовательского интерфейса, клиентской части, а также серверной части, при этом нужно использовать базу данных для хранения и записи информации. Взаимодействие между серверной и клиентской частью должно осуществляться с помощью REST API. База данных должна обладать инструментарием для работы по системе CRUD. Также в проекте должна быть

реализована система аутентификации и авторизации, чтобы каждый пользователь имел доступ к своей истории обработок, а также мог управлять своими уникальными файлами.

1.4.1 Требования к пользовательскому интерфейсу

Пользовательский интерфейс должен соответствовать некоторым требованиям, которые помогут конкретизировать большую часть моментов и вопросов при разработке частей веб-интерфейса. Данные требования скорректированы и выведены на основе анализа конкурентов (пункт 1.3), что позволит улучшить или добавить дополнительные аспекты при конкуренции с аналогами. Были сформулированы следующие пункты:

- наличие системы регистрации и входа в собственную учётную запись;
- возможность выбора параметров из определённого пула для улучшения качества выбранного файла;
- возможность сравнения исходного графического контента с полученным результатом;
- галерея обработанных видео и изображений. Данное нововведение позволит переманить аудиторию продуктов-конкурентов, которые не располагают данной функцией;
- возможность скачивания результата обработки;
- простой и понятный графический интерфейс пользователя. Интерфейс должен быстро помогать пользователю в навигации по сайту, что не отпугнёт его от использования;
- адаптивность и поддержка графического интерфейса пользователя. При анализе конкурентов был выявлен недостаток в поддержке работы сайта для разных типов устройств. Необходимо устранить это, чтобы пользователь мог успешно использовать веб-интерфейс, как на ПК, так и на смартфоне;
- удобная ориентация на многостраничном сайте, который имеет возможность масштабирования.

1.4.2 Функциональные требования

Функциональные требования также составлялись при анализе конкурентов (пункт 1.3). Это также, как и в случае с графическим интерфейсом, позволяет выявить недостатки продуктов-конкурентов, устранить их и добавить собственные уникальные черты. Сформулированы следующие функциональные требования:

- обеспечить простоту использования, чтобы пользователь смог легко понять функционал приложения;
- реализовать функционал, позволяющий пользователю выставить собственные настройки улучшения графического контента или выбирать из готовых пресетов по желанию;
- добавить отправку оригинала на сервер для обработки и приём результата пользователем;
- журнал готовности. Пользователь должен иметь перед глазами информацию, позволяющую понять, обработка контента завершена или нет;
- сравнение результата и оригинала. Одна из оригинальных черт проекта, позволяющая привлечь пользователей к использованию веб-приложения;
- иметь спроектированную архитектуру, включая веб-интерфейс для взаимодействия пользователей с системой, а также веб-сервер, который принимает HTTP-запросы от клиентов;
- обеспечить безопасность и конфиденциальность пользовательских данных;
- оповещение пользователя. Веб-приложение должно оповещать пользователя о происходящих процессах и доступности использования уникальных функций приложения;
- обеспечить поддержку различных форматов загружаемых файлов (например, mp4, mkv, avi, webm, png, jpg, webp, gif);

– галерея результатов. Связать приложение с БД для возможности вывода ранее обработанных файлов. В галерее должна присутствовать возможность использования фильтров, скачивания и удаления файлов;

– функционал аутентификации и авторизации пользователей;

– итоговое приложение должно взаимодействовать с БД, в которой хранятся данные для работы веб-сервиса.

1.4.3 Технические требования

Запуск веб-приложения, как и пуск другого программного обеспечения требует наличия определённых аппаратных мощностей на машинах клиента и сервера. Требования к серверу будут значительно выше, нежели к клиенту. Это связано с распределением ролей в организованной иерархии взаимодействия. Данная иерархия подразумевает высокий уровень нагрузки на серверную машину и относительно небольшой на машину клиента.

Определим более чёткие технические требования для успешного разворачивания проекта на серверной машине. Стоит упомянуть, что технический запрос аппаратных ресурсов формировался эмпирически, при помощи проверок на ПК разработчика.

Рекомендуемые системные требования к серверной машине:

– процессор. Обработка большого количества входящих запросов принуждает к использованию многоядерного процессора. Частота элемента должна составлять не менее 2,5 ГГц, а количество минимально необходимых ядер стартовать от 4 и выше;

– оперативная память. На сервере планируется использовать RAM-диск. Обозначенное решение позволит ускорить скорость передачи данных, а также уменьшить скорость износа основных дисков сервера. Из-за большого количества работ с файлами, описанный шаг будет довольно эффективной мерой оптимизации. Рекомендуемое количество оперативной памяти – 8 ГБ и выше;

– накопитель. В качестве накопителя можно использовать HDD или SSD диски, но рекомендуется установить оба типа дисков для сервера. Жёсткий диск

					ПГУ1.090301.04.001 ПЗ	Лист
						31
Изм.	Лист	№ докум.	Подпись	Дата		

(HDD) обладает повышенной отказоустойчивостью, но имеет небольшую скорость считывания / записи. В это же время твердотельный накопитель (SSD) может быстро производить операции чтения / записи, однако имеет сложности с восстановлением утерянной информации при поломке. Тандем работы двух типов дисков будет компенсировать недостатки работы друг друга. Рекомендуется установка HDD с объёмом памяти не менее 2 ТБ и SSD с ёмкостью более 1 ТБ. Твердотельный накопитель будет служить в качестве быстрой памяти сервера, а жёсткий диск должен хранить бэкапы системы и данных пользователей;

– операционная система. В качестве ОС для развёртывания была выбрана система Ubuntu Server. Это дистрибутив семейства Linux, обладающий превосходной производительностью и стабильностью работы. Также данный образ не имеет графической оболочки, что позволяет оптимизировать используемые ресурсы сервера. Запуск также возможен на операционных системах macOS и Windows, однако надёжность работы данных систем вызывает опасения при разворачивании высоконагруженных серверов;

– интернет-соединение. Для организации стабильного веб-сервера необходимо иметь высокую пропускную способность сети. Поток пользователей может увеличиться гиперболически в любой момент времени, что повлечёт за собой проблему ответов на запросы пользователей. Рекомендуемые характеристики используемой сети: скорость загрузки не менее 500 Мбит/с, скорость отдачи не менее 300 Мбит/с. Если приложение будет увеличивать объём передаваемых данных (например, загрузка / скачивание файлов большого размера), то требуется увеличить пропускную способность сети;

– программное обеспечение. Сервер должен иметь при запуске следующие программные продукты:

1) программная платформа Node.js. Необходима для поддержки и запуска приложений на языке JavaScript;

2) менеджер пакетов Npm. Позволяет устанавливать дополнительные пакеты в Node.js и автоматически настраивать внутренние зависимости проекта;

3) система контроля версий Git. Отвечает за откат проекта на более старые или обновления на актуальные версии;

4) СУБД MySQL. Хранит текущие данные пользователей и их бэкапы;

5) фреймворк express.js. Требуется для создания веб-приложений и их API;

6) драйвер mysql2. Программное дополнение для корректной работы упомянутой ранее СУБД;

7) библиотека body-parser. Отвечает за парсинг HTTP запросов;

8) библиотека express-session. Управляет сессиями пользователей;

9) библиотека bcryptjs. Хэширует пароли пользователей для более безопасного хранения данных;

10) библиотеки passport и passport-local. Позволяют реализовать компонент аутентификации пользователей;

11) библиотека multer. Позволяет управлять загрузкой файлов;

12) библиотеки fs и fs-extra. Реализуют управление файловой системой;

13) библиотека path. Отвечает за работу с расположениями в файловой системе;

14) библиотека dotenv. Требуется для взаимодействия с переменными окружения среды;

15) библиотека nodemon. Необходима для автоматического перезапуска сервера при изменении файлов.

Технические требования к клиентской машине, взаимодействующей с приложением через браузер, более простые, чем аналогичные пункты

требований для сервера. Это можно связать с простотой и доступностью взаимодействия ПК пользователя с сервером. основополагающим фактором возможности взаимодействия является наличие поддерживаемого браузера.

Рекомендуемые системные требования к клиентской машине:

– операционная система. Подходит любая операционная система, поддерживающая работу с браузерами. Из популярных можно выделить: Windows 10, Linux Ubuntu и macOS. Кроссплатформенность приложения должна обеспечивать одинаковый опыт использования на разных системах;

– браузер. Основной компонент взаимодействия пользователя с приложением. Компьютер клиента должен поддерживать работу в современных версиях популярных браузеров, таких как: Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, Opera. Требуется использование последней стабильной версии для обеспечения максимальной совместимости и производительности;

– оперативная память. Большинство браузеров требуют некоторое количество оперативной памяти для поддержания собственной работоспособности и вкладок на фоне. Данный фактор выдвигает требование о наличии необходимого количества оперативной памяти для поддержки работы браузера. Разные программные продукты могут предъявлять собственные минимальные требования, однако чаще всего нужно около 4-х ГБ оперативной памяти;

– процессор. Наличие мощного процессора ускоряет скорость работы в браузере. Данный компонент должен обладать частотой не ниже 1,5 ГГц. Желательно наличие нескольких ядер;

– интернет-соединение. Бесперебойное взаимодействие с приложением может обеспечить скоростное подключение к сети. Если рассматривать желаемый вариант, то скорость загрузки должна составлять 100 Мбит/с, а скорость отдачи – 50 Мбит/с. Повышение скорости сети позволит быстрее взаимодействовать с большими объёмами данных;

– дисковое пространство. Необходимо иметь 50 ГБ свободного места. Это оптимальный показатель, позволяющий без особых ограничений взаимодействовать с приложением на протяжении среднего количества времени. Увеличение данного параметра позволит взаимодействовать с приложением на порядок дольше.

Ограничения проекта:

- длительность видео и размер файла ограничиваются в зависимости от доступных системных ресурсов сервера;
- время обработки файла может варьироваться в зависимости от загрузки системы в текущий момент времени и сложности выполняемой задачи;
- необходимо обновлять и поддерживать встроенные модели машинного обучения.

1.5 Выбор инструментов разработки

1.5.1 Выбор языка программирования

1.5.1.1 Язык программирования Python

Python – это высокоуровневый интерпретируемый язык программирования, широко используемый для создания веб-приложений и сайтов. Простота изучения и последующая лёгкость использования делают данный язык одним из самых популярных как для новичков, так и для достаточно опытных разработчиков. Python позволяет решать разнообразный пул задач, устанавливаемый программистом. А гибкость и большая стандартная библиотека позволят сделать это ещё быстрее.

У данного языка присутствует чрезвычайно активная поддержка сообщества, широкий спектр доступных фреймворков (например, Django, Flask, Pyramid) и библиотек, распространяемых бесплатно, а также кроссплатформенная совместимость, позволяющая вести разработку на большинстве доступных систем.

Но Python имеет свои недостатки. Так, за большую стандартную библиотеку и понятность он расплатился скоростью выполнения написанных программ, что не позволяет использовать данный язык для веб-приложений реального времени. Некоторые конструкции языка могут вызвать вопросы у новичка, что также является проблемой.

Данный язык является одним из лучших выборов для веб-разработчика. Выбор в пользу данного средства может быть сделан ещё и по факту наличия механизма ООП и динамической типизации языка. Он достаточно хорошо подходит для создания веб-приложений и сайтов, разработки RESTful API и создания backend-логики для веб-приложений.

1.5.1.2 Язык программирования Ruby

Ruby – это динамический, интерпретируемый язык программирования, который известен своим простым синтаксисом, гибкостью и лёгкостью использования. Данный язык имеет большое количество готовых библиотек и фреймворков для веб-разработки. Одни из таких фреймворков – Ruby on Rails, упрощающий создание прототипов и разработку backend-логики для веб-приложений. Также Ruby обладает обширной поддержкой многопоточности, большим сообществом поддержки и собственной экосистемой gems.

Как и Python, рассматриваемый язык не лишён недостатков. Проблемы возникают при попытке масштабирования приложения на некоторое количество машин и оптимизации производительности (язык использует достаточно много аппаратных ресурсов для запуска приложения). Обучение с нуля данному языку гарантирует резкое повышение сложности из-за пула используемых технологий. Рассматриваемый язык не имеет встроенной поддержки для работы с графикой и мультимедиа, что сильно затруднит разработку веб-интерфейса. В конкретном случае он проигрывает сравнению с языком JavaScript.

1.5.1.3 Язык программирования PHP

PHP – скриптовый язык общего назначения, используемый для разработки веб-приложений. Хорошо интегрируется с HTML, что упрощает

создание динамических веб-сайтов. Он является одним из самых распространённых языков программирования для backend разработки веб-приложений.

Если говорить о преимуществах PHP, то можно отметить, что данный язык имеет относительно лёгкую схему изучения, огромное количество готовых библиотек, фреймворков и документации, а также может хорошо обеспечить высокую производительность приложения и его масштабируемость. Данный инструмент хорошо подходит для создания динамических веб-сайтов, которые работают с базами данных.

PHP имеет некоторые критичные недостатки, не позволяющие использовать его в данном проекте. Среди них можно отметить ограничение по поддержке многопоточности, сложность синтаксиса при создании некоторых логических конструкций, необходимость в дополнительных библиотеках для работы с графикой и мультимедиа. Частично не выдерживает конкуренцию с JavaScript из-за меньшего количества ресурсов и библиотек.

Хотя рассматриваемый язык не подходит при разработке данного проекта, он может успешно использоваться для создания динамических веб-сайтов, которые работают с базами данных, разработки backend-логики для веб-приложений, создания RESTful API.

1.5.1.4 Язык программирования JavaScript

JavaScript – это высокоуровневый, динамический, мультипарадигменный язык программирования, который используется для создания интерактивных веб-страниц и веб-приложений [2]. Является одним из самых популярных языков программирования для разработки веб-приложений.

Данный язык обеспечивает высокую производительность и масштабируемость разрабатываемых приложений, имеет большое количество готовых библиотек и фреймворков, а также доступных ресурсов для обучения, что поможет новичку ускорить изучение рассматриваемого инструмента. Из дополнительных плюсов JavaScript стоит упомянуть поддержку асинхронного

программирования и обработку событий и наличие большого сообщества программистов вокруг данного языка, что увеличивает количество доступной документации.

Несмотря на перечисленные преимущества, JavaScript имеет некоторое количество недостатков. Язык имеет достаточно сложный синтаксис, из-за чего некоторые логические конструкции могут быть довольно сложны в понимании. Требуется скачать и установить большое количество библиотек для работы с графикой и мультимедиа.

JavaScript идеально подходит для использования в данном проекте, а кроме этого, может использоваться при разработке frontend-логики для веб-приложений и создания RESTful API.

1.5.1.5 Выводы по выбору языка программирования

Все описанные выше языки программирования имеют свои достоинства, недостатки, а также коэффициент их соотношения. JavaScript имеет самый высокий коэффициент превалирования достоинств над недостатками из перечисленных языков. Он является оптимальным выбором для создания интерактивных веб-страниц и веб-приложений, которые работают с видео контентом и базами данных.

Если конкретизировать главные преимущества JavaScript над другими языками, то можно отметить, что он более производителен (PHP, Ruby), имеет широкую поддержку асинхронного программирования (Python), обладает большим количеством библиотек и фреймворков (PHP, Ruby), использует большое количество документации (Python, PHP, Ruby).

1.5.2 Выбор инструментов для реализации графического интерфейса в веб-приложении

1.5.2.1 HTML, CSS

HTML (HyperText Markup Language) – это язык разметки, который позволяет создавать структуру веб-страницы, добавлять заголовки, абзацы, таблицы, изображения и другие элементы. HTML является основой для создания

структуры и содержания веб-страниц, определяя их семантику и логическую организацию [3].

Затрагивая преимущества HTML, можно отметить, что он обладает простым и понятным синтаксисом, огромным количеством готовых шаблонов и фреймворков, достаточным объёмом документации, как для новичков, так и для опытных разработчиков. Рассматриваемый инструмент хорошо подходит для создания структуры веб-страниц и веб-приложений, а также обеспечивает высокую доступность и совместимость с различными браузерами и устройствами.

HTML может быть сложным для начинающих из-за большого количества тегов и атрибутов. Вдобавок данный язык не может полноценно использоваться без дополнительных технологий, таких как CSS и JavaScript, для создания интерактивных веб-страниц и веб-приложений. Чаще всего использование HTML подразумевает связку с языком стиля CSS.

CSS (Cascading Style Sheets) – это язык стиля. CSS предоставляет веб-разработчикам мощные средства для управления визуальным представлением веб-страниц, включая шрифты, цвета, размещение элементов и многое другое [3].

Синтаксис CSS лёгок и понятен, что позволяет довольно быстро использовать данный язык наравне с HTML. Также как и HTML, CSS обладает огромным количеством готовых шаблонов и фреймворков. С помощью описанного инструмента можно создать интерактивный и привлекательный дизайн веб-страниц и веб-приложений. Обладает высокой доступностью и совместимостью с различными браузерами и устройствами.

Из недостатков CSS можно отметить сложность использования при начале изучения из-за большого количества свойств и значений. У рассматриваемого языка есть обратная привязка к другим инструментам по типу HTML и JavaScript, что не позволяет использовать его самостоятельно.

CSS хорошо подходит для создания интерактивного и привлекательного дизайна веб-страниц и веб-приложений и разработки frontend-логики для веб-приложений.

1.5.2.2 EJS

EJS (Embedded JavaScript) – это шаблонный движок, позволяющий встраивать JavaScript-код в HTML-разметку. Он предоставляет простой и эффективный способ создания динамических веб-страниц.

Движок имеет интеграцию с Node.js и Express.js. EJS хорошо интегрируется с Node.js и фреймворком Express, что делает его распространённым выбором для серверной веб-разработки. Возможность повторного использования готовых модулей, а также повышенная производительность за счёт компиляции шаблонов в JavaScript-функции, позволяет увеличить скорость разработки веб-приложения.

EJS имеет недостаток в виде ограниченной логики использования. Из этого исходят проблемы с смешением логики и представления, отсутствием расширенной функциональности движка.

Описанный движок хорошо подходит для небольших и средних веб-приложений на Node.js/Express.js. EJS особенно хорошо подходит для создания динамических веб-страниц с относительно простой логикой, а также он идеален для быстрой разработки прототипов и MVP-приложений.

В целом EJS является хорошим выбором для средних веб-приложений, где важны простота, скорость разработки и интеграция с Node.js/Express.

1.5.2.3 Bootstrap

Bootstrap – это популярный фреймворк для создания адаптивных веб-сайтов и веб-приложений. Он включает в себя готовые CSS-стили, компоненты пользовательского интерфейса и JavaScript-плагины.

Адаптивность и отзывчивость Bootstrap позволяет легко создавать сайты, которые хорошо выглядят и работают на различных устройствах. Наличие готовых компонентов, таких как кнопки, формы, навигация, модальные окна и

многое другое позволяет максимально упростить разработку. Весомым плюсом также является кроссбраузерная совместимость, которая позволяет согласовать отображение и поведение на различных браузерах и устройствах.

Единственным большим недостатком служит стандартизация дизайна элементов. Это не позволяет использовать собственные графические элементы.

Bootstrap хорошо подходит для быстрой разработки MVP-приложений и приложений с простым и стандартным пользовательским интерфейсом.

1.5.2.4 Выводы по выбору инструментов для реализации графического интерфейса в веб-приложении

Максимально удобный графический интерфейс, который будет использоваться в данном проекте, можно реализовать при помощи перечисленных выше технологий. HTML, CSS, EJS, Bootstrap и JavaScript являются частью современного стека веб-разработки, что позволит комфортно использовать их в тандеме друг с другом. Если говорить более конкретно, то применение данных технологий обеспечит высокий уровень совместимости с различными браузерами и устройствами, а также повысит производительность интерфейса без увеличения аппаратных затрат.

HTML, а если точнее, семантические HTML-теги, будут основой графического интерфейса, формирующей структуру страниц. Оформление данных страниц будет производиться при помощи каскадной таблицы стилей CSS, что создаст более привлекательный для использования интерфейс. CSS-фреймворк Bootstrap ускорит эту задачу из-за наличия готовых шаблонных компонентов в собственном пуле (формы, кнопки, навигация, сетки и другие элементы).

При выборе использовались принципы адаптивного дизайна. Это позволит избежать больших проблем в будущем, которые обязательно проявятся с использованием на различных платформах. Одним из элементов решения данной проблемы служит использование подключаемых шрифтов (как пример, Inter-4.0), позволяющих обеспечить единообразный и стилизованный внешний вид.

Затрагивая клиентскую часть приложения, стоит упомянуть, что она будет реализована в файле `jscode.js` и будет использовать JavaScript для взаимодействия с пользовательским интерфейсом. Не стоит забывать и про манипуляции с DOM-элементами. Они позволяют отобразить выбранные файлы, обновить статус загрузки, управлять другими интерактивными компонентами.

Шаблонизатор EJS затронут в данном пуле из-за возможности динамического генерирования HTML-страниц. Технология может вставить данные, полученные с сервера, в соответствующие места на странице, обеспечивая модульность при разработке графического интерфейса.

Использование вышеописанного стека технологий позволит создать интерфейс пользователя с высоким уровнем кроссбраузерности, адаптивности, интерактивности и привлекательности, в отличие от тяжеловесных фреймворков, которые добавляют ненужной тяжести приложению.

1.5.3 Выбор среды разработки

В качестве среды разработки было решено выбрать Visual Studio Code, потому что именно она обладает множеством преимуществ для разработки как веб-клиента, так и веб-сервера по сравнению с другими IDE. Сюда можно отнести её лёгкость и гибкость, что делает её кроссплатформенной, широкие возможности для настройки и расширения функциональности с помощью плагинов.

Если подробно сравнивать Visual Studio Code с другими IDE для разработки веб-приложений, то можно сделать вывод, что она обладает более обширным рядом преимуществ. Это и бесплатное и открытое ПО, и высокая производительность, и гибкость из-за большого количества расширений. VS Code, в отличие от IntelliJ IDEA и WebStorm, предлагает более универсальные возможности, подходящие под разработку и на других различных языках. По сравнению с Atom и Sublime Text, выбранный VS Code предоставляет более совершенные встроенные инструменты для отладки, контроля версий (включая интеграцию с Git) и системами сборки. Visual Studio Code также имеет удобный

интерфейс для написания и редактирования кода с помощью автодополнения, подсветки, а также настройки форматирования.

Применительно к проекту веб-приложения улучшения качества видео Visual Studio Code выступает для разработки веб-клиента идеально подходящей средой из-за поддержки таких технологий, как HTML, CSS, JavaScript, а также TypeScript. Ещё есть встроенная поддержка Emmet, что позволит ускорить вёрстку HTML и CSS, а также автоматически генерировать код. И, конечно же, имеются расширения для популярных фреймворков, предназначенных специально для веб-разработки. Также для разработки веб-сервера Visual Studio Code тоже имеет все необходимые инструменты. Это и поддержка Node.js, Python, Ruby и PHP, и встроенная поддержка таких технологий, как Docker и Kubernetes, которые позволяют развёртывать и управлять веб-сервером в контейнере.

В целом Visual Studio Code является идеальной средой для разработки веб-приложений. Именно поэтому для создания данного проекта была выбрана эта IDE.

1.5.4 Выбор инструментов для реализации веб-сервера

1.5.4.1 Node.js

Node.js является программной платформой, которая служит для выполнения JavaScript на серверной стороне. Архитектуру Node.js использует событийно-ориентированную. Это позволяет ему эффективно обрабатывать множество одновременных соединений без создания новых потоков или процессов. С помощью Node.js можно создавать высокопроизводительные веб-приложения, обрабатывающие одновременно большое количество запросов. Также данная программная платформа позволяет разрабатывать масштабируемые веб-приложения, адаптирующиеся к изменяющимся нагрузкам.

Node.js предоставляет разработчикам возможность использовать JavaScript для создания серверных приложений, API и микросервисов [4]. Программная платформа обладает огромным количеством модулей и библиотек, которые

упрощают процесс разработки. Node.js является открытым исходным кодом, что позволяет разработчикам вносить вклад в его развитие. Node.js работает на различных операционных системах, включая Windows, macOS и Linux, что делает его привлекательным для разработчиков, работающих в разных средах.

Эта программная платформа очень распространена в разработке серверных API и микросервисов, веб-приложений с использованием фреймворков, таких как Express.js, реального времени приложений (чаты и игры), а также инструментов для разработки (сборщики и менеджеры пакетов).

Можно сделать вывод, что Node.js является важным инструментом для разработки высокопроизводительных веб-приложений. Этот инструмент обладает целым рядом удобств. К преимуществам Node.js можно отнести единый язык программирования для клиента и сервера, асинхронность и масштабируемость, высокую производительность, кроссплатформенность, простоту использования, обработку данных в реальном времени, обработку больших нагрузок (обработка большого количества асинхронных запросов), простоту развёртывания и обслуживания (благодаря своей портативности), а также его модульность и расширяемость (огромное количество модулей и библиотек). Все приведённые преимущества делают Node.js идеальным выбором для использования в разработке данного веб-приложения улучшения качества видео.

1.5.4.2 Express.js

Express.js является одним из самых популярных веб-фреймворков для Node.js, предоставляющим разработчикам широкий набор функций для создания веб-приложений и API [5].

При разработке веб-сервера для веб-приложения был выбран Express.js по многим причинам. Во-первых, Express.js даёт удобный и интуитивно понятный API для разработки веб-серверов на Node.js. Во-вторых, с помощью данного фреймворка можно опустить многие низкоуровневые детали HTTP-протокола, позволяя сосредоточиться на логике приложения. В-третьих, он предлагает

мощную систему маршрутизации, позволяющую быстро определять и обрабатывать различные HTTP-методы для разных URL-адресов. Это упрощает организацию логики приложения.

Также немаловажно, что Express.js интегрируется с шаблонизаторами, а это упрощает создание динамических HTML-страниц. Благодаря своей модульной архитектуре Express.js позволяет легко расширять функциональность приложения путем подключения дополнительных middleware-компонентов и библиотек. Как и Node.js, Express.js подразумевает кроссплатформенность, а это делает веб-разработку более гибкой и портативной.

Архитектура Express.js основана на концепции промежуточного программного обеспечения (middleware). Каждый запрос, поступающий на сервер, проходит через цепочку middleware-функций, которые могут модифицировать запрос, вызывать следующие middleware-функции или отправлять ответ клиенту.

Express.js используется при разработке одностраничных приложений, многостраничных веб-приложений, RESTful API-сервисов, микросервисных архитектур, а также реальных приложений. Так как Express.js построен на высокопроизводительной платформе Node.js, это обеспечивает высокую производительность веб-приложений.

В веб-приложении улучшения качества видео было решено использовать Express.js, потому что он является фреймворком, специально разработанным для использования с Node.js. Это обеспечивает качественную взаимосвязь между серверной логикой, написанной на Node.js, и веб-сервером, реализованным с помощью Express.js. Это упрощает разработку и поддержку приложения. Также Express.js предоставляет более гибкую и расширяемую архитектуру, по сравнению с традиционными веб-серверами, такими как Nginx. Также в отличие от Nginx, который в основном предназначен для обслуживания статического контента, Express.js лучше подходит для обработки динамического контента, генерируемого с помощью Node.js. Это особенно важно для данного веб-сервиса,

которое должно предоставлять пользовательский интерфейс и обрабатывать загрузку/отправку файлов. Как говорилось ранее, Express.js предоставляет встроенную поддержку шаблонов. Ещё Express.js предлагает мощную систему маршрутизации, которая позволяет легко определять и обрабатывать различные HTTP-методы (GET, POST, PUT, DELETE и др.) для разных URL-адресов. Это критично для реализации логики приложения, связанной с загрузкой и скачиванием файлов. Также, несмотря на то, что Nginx также может работать в локальной сети, созданной с помощью VPN-решений, таких как Hamachi, Express.js лучше подходит для этой задачи. Поскольку приложение использует динамический контент и обработку файлов, Express.js обеспечивает более простую и гибкую интеграцию с этой архитектурой.

Тем самым, можно сделать вывод, что использование Express.js в качестве веб-сервера для веб-приложения улучшения качества видео является оптимальным решением, потому что он обеспечивает весь необходимый функционал, описанный выше. Это делает его более подходящим решением по сравнению с традиционными веб-серверами, такими как Nginx.

1.5.5 Выбор СУБД

Разработка данного проекта сопряжена с использованием базы данных для хранения информации пользователей и других используемых данных. По этой причине был произведён анализ доступных к использованию СУБД, позволяющих без больших затрат ресурсов управлять ресурсами БД. Использованию могут подлежать следующие СУБД:

– Oracle Database. СУБД, разработанная компанией Oracle с целью создания объектно-реляционных баз данных. Обладает высоким уровнем надёжности и производительности. Может подвергаться масштабированию, предоставляет широкий спектр функций взаимодействия с данными. К сожалению, не имеет бесплатных версий для использования и, в данный момент времени, не может быть приобретена на территории Российской Федерации.

Также наличие строгих лицензионных ограничений с большой вероятностью повлияет на развитие данного веб-приложения, как стартап-проекта;

– MySQL. СУБД для создания реляционных баз данных. Является одной из самых популярных СУБД для использования в различных проектах. Открытая схема приложения, простота настройки и возможность бесплатного использования хорошо способствуют этому. Также MySQL обладает высокой скоростью работы, встроенной поддержкой языка SQL и других внешних языков программирования, что жизненно необходимо использовать в высоконагруженных проектах. К недостаткам СУБД можно отнести отсутствие широкого спектра функционала (за счёт этого достигается высокая скорость работы) и сложность при работе с большими объёмами данных. MySQL отлично подходит для реализации запланированного веб-приложения;

– PostgreSQL. СУБД с открытым исходным кодом для создания объектно-реляционных баз данных. Схожа по устройству с Oracle Database и MySQL. Достаточно хорошо масштабируется, поддерживает язык SQL и другие внешние языки программирования, обладает большим пулом функций. Однако имеет значительные требования к аппаратным ресурсам системы для собственной поддержки и более сложную схему настройки внутренней организации, что может стать проблемой для разработчика;

– SQL Server. СУБД, разработанная компанией Microsoft для создания реляционных баз данных. Поддерживается только на машинах с операционной системой Windows, что можно назвать минусом. Обладает самым большим пулом доступных функций для работы с данными в БД, а также самым эффективным и удобным управлением данными из перечисленных СУБД. На официальном сайте разработчика есть большое количество документации, позволяющей довольно быстро освоить работу с данным программным обеспечением. Кроме минуса в виде отсутствия кроссплатформенности можно выделить высокую стоимость использования. Если более корректно, то цена является самой высокой из всех перечисленных СУБД;

– MongoDB. СУБД с открытым исходным кодом, использующаяся в большом числе открытых проектов. Обладает большой возможностью горизонтального масштабирования, высокой скоростью работы на кластерах серверов, а также поддержкой работы с расширяемой памятью (например, на RAM-дисках, что позволяет увеличить скорость работы). В противовес перечисленным плюсам можно отметить низкую стабильность СУБД (из-за поддержки расширяемой памяти) и небольшое количество документации в открытом доступе;

После проведенного анализа было решено использовать MySQL как самую доступную и надежную СУБД из представленных. MySQL отличается высокой производительностью обработки транзакций, что делает её идеальным выбором для веб-приложений с интенсивной нагрузкой на базу данных [6].

1.5.6 Выбор инструментов для работы с файлами

В разрабатываемом проекте веб-приложения для работы с файлами в системе были выбраны следующие библиотеки:

– multer. Библиотека для обработки multipart / form-data запросов Node.js. Используется для Express.js при загрузке файлов, обеспечивая простой и эффективный способ обработки файлов. В текущем веб-приложении Multer используется для обработки загрузки видео и изображений, отправляемых пользователями. Он настраивает расположение сохранения получаемых файлов и предоставляет информацию о загруженных файлах в обработчиках маршрутов. Это упрощает процесс последующей обработки и интеграцию с другими частями приложения;

– fs. Встроенная библиотека Node.js, которая позволяет работать с файловой системой. Предоставляет интерфейс для выполнения различных операций с файлами и директориями, таких как чтение, запись, создание, удаление и т.д. В проекте библиотека применяется для создания папок для хранения загруженных файлов (видео, изображения), перемещения загруженных

файлов в соответствующие папки, чтения и записи файлов параметров (JSON-файлы), удаления файлов (при удалении записи из базы данных);

- fs-extra. Расширение для библиотеки fs, добавляющее расширенные функции для работы с файловой системой;

- path. Библиотека для работы с расположениями файлов и папок. Применяется для получения расширения обрабатываемого файла, построения полных путей к файлам и папкам, извлечения имени файла;

- mysql2. Расширение для взаимодействия с БД MySQL. Применяется для сохранения информации о загруженных файлах (название, путь, тип файла, описание) в базе данных. Это позволяет отображать информацию о загруженных файлах в галерее на главной странице. Интеграция с базой данных как раз обеспечивает возможность управления и отображения файлов, загруженных пользователями;

- express-fileupload. Библиотека Express.js, предназначенная для обработки загрузки файлов в приложениях. Использование вместе с multer обеспечивает более гибкие возможности для работы с загруженными файлами.

Описанные выше библиотеки и расширения обеспечивают полноценную функциональность, связанную с загрузкой, сохранением, обработкой и управлением файлами в данном веб-приложении. Каждая из них выполняет определённую роль и вносит свой вклад в реализацию этой важной части веб-приложения улучшения качества видео.

Кроме библиотек и расширений в проекте используется специализированный формат файлов – JSON.

JSON (JavaScript Object Notation) – это лёгкий в использовании формат для обмена данными. Он представляет данные в виде простых структур, таких как объекты, массивы, числа, строки, логические значения и null. Это делает его удобным и читаемым для людей, а также лёгким для анализа и генерации компьютерными программами.

В разрабатываемом веб-приложении JSON-формат используется для хранения параметров для обработки загруженных изображений/видео.

Когда пользователь загружает файл (видео или изображение), приложение создаёт JSON-файл, содержащий информацию о выбранных пользователем параметрах, таких как разрешение, нейронная сеть для обработки и FPS (для видео). Эти JSON-файлы с настройками сохраняются в той же папке на сервере, что и сам загруженный файл. Когда пользователь пытается скачать обработанный файл, приложение читает соответствующий JSON-файл с настройками и использует эту информацию для формирования описания файла, которое видно в галерее обработанных файлов.

1.5.7 Выбор инструментов для аутентификации и безопасности в веб-приложении

Для разработки функционала аутентификации в веб-приложении выбор пал на инструменты Passport, passport-local и bcryptjs. Было решено использовать именно их, ведь при взаимодействии с веб-сервисом пользователи ожидают, что их личные данные и информация будут надежно защищены от кражи. А эти фреймворки как раз помогают в разработке надежных систем аутентификации, которые будут обеспечивать высокий уровень безопасности.

Passport является фреймворком аутентификации, предоставляющим простой и гибкий способ аутентификации запросов, для проектов, созданных на Node.js. Это даёт возможность легко интегрировать стратегии аутентификации в свои приложения. Passport необходим для веб-сервиса, потому что он контролирует доступ к ресурсам и данным, тем самым обеспечивая высокий уровень безопасности веб-приложения. Фреймворк даёт выбор из различных способов входа в систему (например, через социальные сети). Также он разделяет логику аутентификации от логики приложения и упрощает поддержку и масштабирование системы.

Для Passport существует такая стратегия аутентификации, которая имеет название Passport-local. Она позволяет проводить аутентификацию с помощью

имени пользователя и пароля. Она необходима для веб-сервиса, потому что даёт пользователям привычный для них способ регистрироваться и входить в веб-приложение, обеспечивает безопасность этого веб-приложения приложения, контролируя доступ к ресурсам и данным, также не требует создания дополнительных аккаунтов.

И наконец, Bcryptjs. Эта библиотека использует алгоритм bcrypt для хеширования паролей. Она позволяет безопасно хранить пароли зарегистрированных пользователей с помощью защиты их от несанкционированного доступа, соответствует требованиям безопасности к хранению конфиденциальных данных и упрощает процесс хеширования паролей.

1.5.8 Выбор VPN для создания локальной сети

Виртуальная частная сеть (VPN) – это технология, которая создаёт безопасное и зашифрованное соединение между устройством пользователя и удаленным сервером, позволяя пользователям безопасно подключаться к другим компьютерам или сетям, даже если они находятся за брандмауэрами или за пределами локальной сети [7].

Для создания виртуальной локальной сети между группой машин было выбрано программное обеспечение Hamachi. Оно позволяет организовать взаимодействие машин в сети так, если бы они находились в одной локальной сети. Выбор данного программного продукта среди аналогов обоснован высокой рентабельностью при сравнении Hamachi с конкурентами. Выбранный сервис сравнивался со следующими аналогами:

– OpenVPN. Популярное и широко используемое программное обеспечение с открытым исходным кодом для создания VPN-соединений. Если сравнивать данный продукт с Hamachi, то его настройка более требовательна к знаниям пользователя, а отсутствие облачной инфраструктуры немного усложняет развёртывание и управление сетью;

– Zerotier. Программное обеспечение для создания виртуальных сетей. Имеет чуть более запутанный интерфейс и менее широкую базу активных

пользователей, образующих сообщество по поддержке. Бесплатная версия Zerotier предлагает меньше возможностей, чем аналогичная версия Hamachi;

– Radmin VPN. Программа для создания VPN-соединений, ориентированная на платформу Windows. Не имеет кроссплатформенности и предлагает менее широкий спектр услуг, в отличие от Hamachi. Также не обладает таким же большим количеством документации.

После аргументирования выбора инструмента Hamachi, стоит обозначить преимущества его использования. К явным плюсам данного ПО можно отнести кроссплатформенность, бесплатность и простоту при использовании. Безопасность также является достоинством данной программы, так как она использует шифрование при передаче данных через туннель.

Однако зависимость Hamachi от сервера компании производителя и понижение производительности на использующей машине являются конкретными недостатками, которые имеют хороший баланс, учитывая плюсы.

Hamachi часто используется небольшими группами разработчиков для локальных проектов. Также с его помощью может осуществляться тестирование и удалённая разработка программного обеспечения.

Для развёртывания веб-сервиса использование VPN (виртуальной частной сети) может быть предпочтительнее, по сравнению с использованием VPS (виртуального частного сервера). Это можно обосновать анализом следующих областей:

– безопасность и конфиденциальность. VPN обеспечивает более высокий уровень безопасности, шифруя весь трафик между клиентами и сервером. Это помогает защитить данные пользователей и предотвратить несанкционированный доступ к веб-сервису. VPN также позволяет скрыть реальные IP-адреса пользователей, что дополнительно повышает уровень конфиденциальности;

– контроль и управление. Используя VPN, можно более эффективно контролировать и управлять доступом к веб-приложению. Существует

возможность настраивать разрешения и ограничения для различных пользователей или групп, обеспечивая более гибкое управление;

– масштабируемость. VPN-решения более масштабируемы, чем VPS. Можно легко добавлять или удалять пользователей, не сталкиваясь с ограничениями, присущими VPS;

– экономичность. Развёртывание и поддержка VPN, как правило, более экономичны, чем аренда VPS. Можно избежать дополнительных расходов на инфраструктуру, связанных с VPS.

Учитывая, что аренда VPS может быть затратной, что для разработки веб-сервиса нужна собственная безопасная сеть, позволяющая контролировать доступ к веб-приложению, а также что пока что нет нужды в масштабировании проекта, использование Namachi стало логичным выбором. Он позволил быстро и легко создать безопасную виртуальную сеть между участниками команды, не требуя сложной настройки или дополнительных расходов на VPS. Это поможет сосредоточиться на разработке приложения, а не на управлении инфраструктурой. По мере роста и масштабирования, разумеется, потребуется использование VPS. VPS может быть более подходящим вариантом, если в будущем на веб-приложение будет высокая нагрузка или потребуется более гибкая настройка инфраструктуры. Тем не менее, для большинства веб-приложений, особенно на начальных этапах развития, использование VPN может быть более предпочтительным выбором.

1.6 Архитектура проекта

1.6.1 Выбор архитектуры веб-сервиса

Для общего проекта веб-сервиса улучшения качества видео, необходимо выбрать правильную архитектуру системы, ведь это является залогом стабильной работы и простоты поддержки в будущем. Каждая из рассматриваемых организаций имеет свои преимущества и недостатки. В

					ПГУ1.090301.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		53

качестве возможных вариантов для проекта можно выбрать монолитную архитектуру системы, а также микросервисную архитектуру.

Монолитная архитектура системы. Традиционная модель программного обеспечения, которая представляет собой единый модуль, работающий автономно и независимо от других приложений. Можно дать и другое определение – отдельная большая вычислительная сеть с единой кодовой базой, в которой объединены все бизнес-задачи проекта. Организация монолитной архитектуры показана на рисунке 9. Данную организацию удобно использовать на начальных этапах проекта, однако, подход содержит в себе и некоторое количество недостатков.

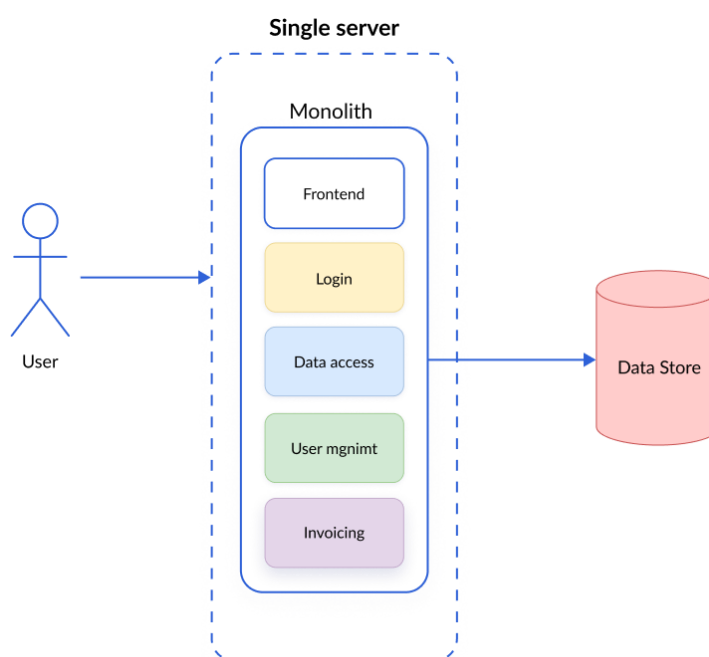


Рисунок 9 – Организация монолитной архитектуры

Преимущества монолитной архитектуры:

- 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;

- 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;

– 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;

– 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.

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

– 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;

– 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;

– 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;

– 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 бизнес-логикой и конкретной целью. Обновление, тестирование, развёртывание и масштабирование выполняется для каждой службы отдельно. Подобный подход не снижает сложности реализуемого проекта, но позволяет конкретизировать и устранить данные сложности в определённых сервисах. Подход независимого друг от друга функционирования является достаточно эффективным за счёт возможности разделённой разработки и часто используется в крупных проектах. Организация микросервисной архитектуры показана на рисунке 10.

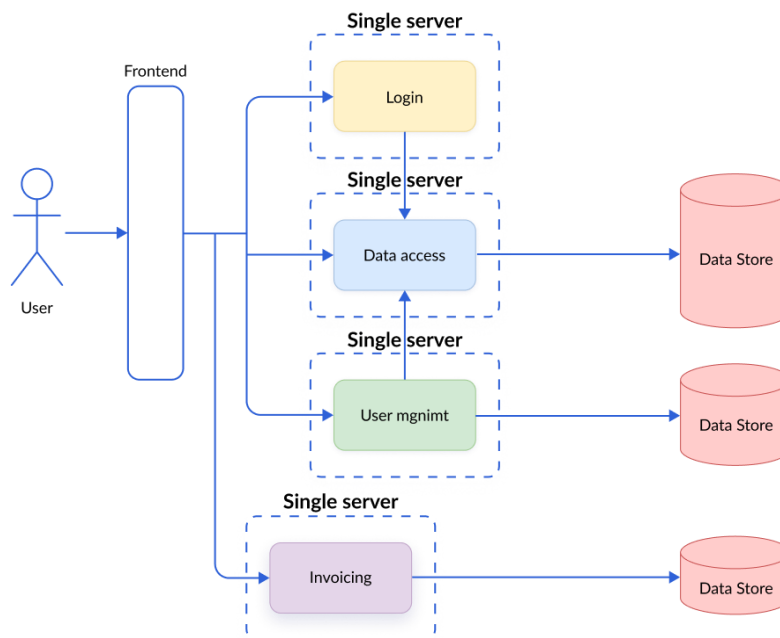


Рисунок 10 – Организация микросервисной архитектуры

Данная архитектура располагает следующими преимуществами:

– 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;

– 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;

– 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;

– 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:

– 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;

– 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;

– 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.

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.

1.6.2 Архитектура веб-приложения

1.6.2.1 Клиент-серверная архитектура

При проектировании веб-приложения было решено использовать клиент-серверную архитектуру, чтобы клиентом был веб-браузер, а сервером было приложение на Node.js. Клиент-серверная архитектура обеспечивает разделение ответственности между клиентской и серверной частями приложения. Взаимодействие между ними осуществляется с помощью REST API.

Клиентская часть содержит:

- файл style.css для стилизации клиентской части;
- папка images для изображений;

- папка `fonts`, которая содержит шрифты;
- `jscode.js` представляет из себя клиентский `js`-файл, который отвечает за динамику и взаимодействие с пользовательским интерфейсом на основной странице, где представлен основной функционал приложения;
- `homepage.ejs` – шаблон представления страницы, являющейся основной в проекте веб-приложения. Он отвечает за отображение в интерфейсе веб-страницы всех основных возможностей, доступных пользователям, которые авторизовались;
- `login.ejs` – для страницы, отвечающей за вход в веб-приложение. С его помощью происходит отображение всех элементов пользовательского интерфейса, необходимых для аутентификации;
- `register.ejs` – для страницы регистрации пользователя. Он отображает все элементы пользовательского интерфейса, которые нужны для регистрации.

Серверная часть включает в себя:

- `DBConnection.js`, содержащий в себе конфигурацию для подключения к базе данных. Установление соединения с базой данных MySQL и экспорт этого соединения для использования в других частях приложения. `viewEngine.js` – файл, который настраивает движок отображения (в этом проекте используется `ejs`) на сервере. По факту, это настройка системы отображения (`view engine`) для веб-приложения, которая включает в себя конфигурацию шаблонов, а также статических файлов;
- `homePageController.js` – он обрабатывает запросы на главную страницу, отвечает за получение списка видео и изображений, за удаление записей из галереи обработанных файлов, сохранение файлов, представленных в галерее, сохранение обработанного изображения/видео, также, например, за занесение информации о файле в БД от сервисного слоя и за создание представления. `loginController.js`, отвечающий за обработку запросов, связанных с авторизацией пользователя. Это вход в аккаунт, выход из аккаунта, а также проверка состояния аутентификации. `passportLocalController.js` – этот файл обрабатывает

аутентификацию пользователей, используя для этих целей passport.js. registerController.js отвечает за обработку связанных с регистрацией новых пользователей (валидация, создание новых учетных записей) запросов;

- web.js, который отвечает за определение маршрутов приложения и связывание их с соответствующими контроллерами;

- файл homePageService.js, отвечающий за реализацию бизнес-логики для управления видео и изображениями пользователей, то есть их их сохранение, удаление и получение информации об этих файлах. Ещё этот файл предоставляет данные о видео и изображениях, связанных с пользователем, для дальнейшего использования в контроллере. loginService.js – этот файл ответственный за реализацию бизнес-логики, связанной с аутентификацией пользователей (поиск зарегистрированных пользователей и сравнение введенных паролей). Файл registerService.js – отвечает за реализацию бизнес-логики, которая связана с регистрацией новых пользователей, что подразумевает проверку уникальности email-адреса и создание новых учётных записей;

- authValidation.js. Этот файл определяет правила валидации для процессов регистрации и входа в систему;

- server.js, который представляет из себя основной файл сервера. Он выполняет работу по настройке и запуску Express.js-приложения, а также обрабатывает входящие запросы от клиента.

Клиентская часть отвечает за представление данных и взаимодействие с пользователем, а серверная часть отвечает за обработку логики приложения, взаимодействие с базой данных и предоставление API для клиентской части.

Клиент шлёт HTTP-запрос, а сервер обрабатывает его в соответствующем контроллере, выполняет необходимые операции и формирует ответ. Ответ отправляется в виде HTML, JSON или статики. Клиент его получает и отображает пользователю.

На рисунке с общим представлением всех сущностей/файлов можно увидеть все основные файлы проекта, связанные друг с другом (рисунок 11).

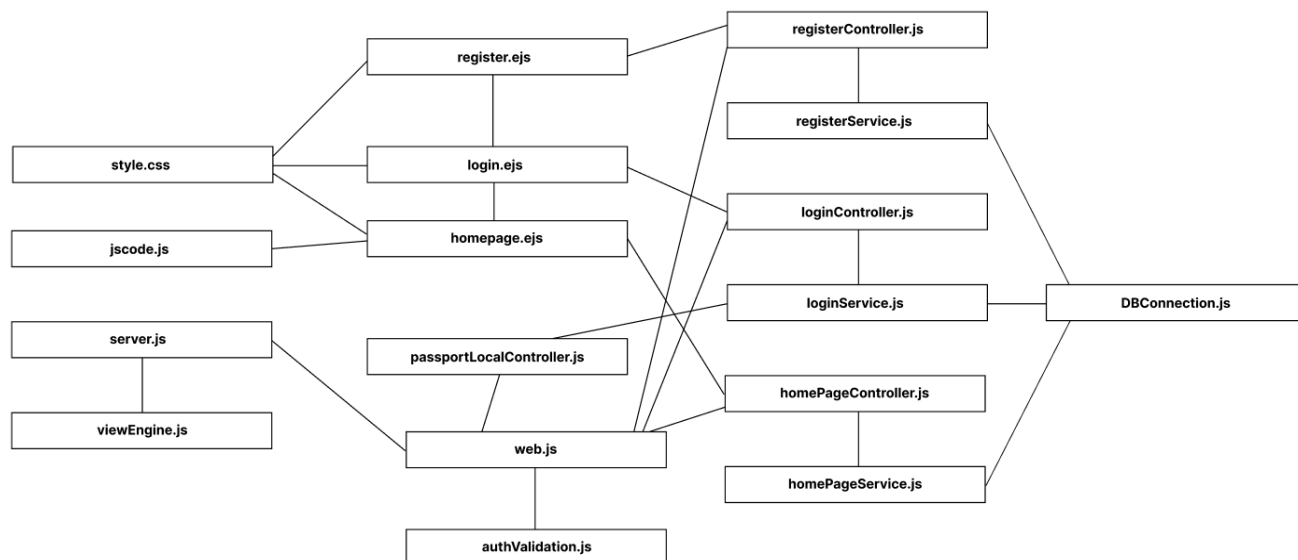


Рисунок 11 – Общее представление всех файлов

1.6.2.2 Паттерн проектирования MVC

Кроме клиент-серверной архитектуры в проекте используется паттерн проектирования MVC. Он делит проект на три связанных компонента:

- модели, что работают с данными из БД приложения;
- представления отвечают за представление разных данных пользователю, используют шаблонизатор EJS для генерации страниц;
- контроллеры связывают модель и представление, обрабатывая запросы пользователя.

Использование паттерна MVC имеет много достоинств. Например, каждый компонент отвечает за свою задачу, изменения в одном компоненте MVC не затрагивают другие компоненты, чёткое разделение ответственности делает его более структурированным.

Графическое представление паттерна можно увидеть на рисунке 12.

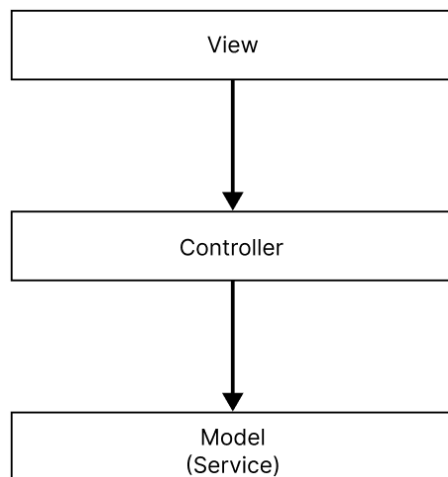


Рисунок 12 – Графическое представление паттерна MVC

Структура проекта разрабатываемого веб-приложения:

- node_modules – зависимости проекта;
- src – это главная корневая папка. Она включает в себя все основные файлы и папки проекта;
- содержание configs включает в себя файлы конфигурации;
- папка controllers содержит контроллеры для различных функциональных возможностей, таких как домашняя страница, авторизация, регистрация и т.д.;
- public является папкой, содержащей статические файлы, доступные для клиента. Это CSS, JavaScript, изображения;
- в папке routes присутствует определение маршрутов приложения, а именно в файле web.js;
- в services лежат модели;
- папка validation содержит файл authValidation.js, где описана логика валидации;
- views нужна для представлений для всех страниц веб-приложения;
- server.js запускает сервер;
- package.json содержит зависимости, скрипты и другую информацию.

С помощью такой структуры можно будет легко поддерживать приложение, редактировать и добавлять функционал. В этом будет помогать шаблон MVC, использованный в архитектуре проекта.

1.6.2.3 Архитектура и проектирование веб-приложения на примере UML-диаграмм

Для более точного описания архитектуры и проектирования веб-приложения по улучшению качества видео, далее будут представлены UML-диаграммы.

Например, диаграмма развёртывания, приведённая на рисунке 13, позволяет представить архитектуру проекта, наглядно показывая физическое размещение компонентов системы (на каких устройствах они расположены), а также их взаимодействие друг с другом на уровне аппаратного и программного обеспечения.

По представленной диаграмме можно увидеть, что веб-приложение состоит из VPN-сервера, а также frontend и backend частей. Сервером здесь выступает компьютер, который принимает HTTP-запросы от клиентов (веб-браузеров) и в ответах выдаёт HTML-страницы, файлы и др. А ещё сервер занимается сбором, обработкой и хранением информации, связанной с приложением. В качестве клиента тут выступает веб-браузер, который занимается отображением веб-страниц пользователю и выполняет JS-код. Клиент занимается отправкой запросов к серверу. Ответы получает в виде JSON, HTML, CSS, JavaScript.

Внутри сервера можно выделить Backend и Frontend.

Backend-часть (серверная часть), написанная на Node.js, работает с БД, отвечает за логику приложения, функционал авторизации, а также работает. Предоставляет возможности, которые используются frontend-частью для получения данных и выполнения операций. В состав этой части входит routes (маршруты приложения), controllers, которые обрабатывают входящие запросы, взаимодействуют с сервисами и возвращают ответы, services (содержат бизнес-

логику приложения, такую как взаимодействие с базой данных, обработка файлов и т.д.), configs (настройка различных аспектов приложения, таких как подключение к базе данных, настройка шаблонизатора), validation (валидация входных данных, поступающих в приложение). Наконец, server.js – это основной файл веб-сервера, отвечающий за запуск сервера и инициализацию всех необходимых компонентов. Для создания серверной части приложения используется Express.js. Он нужен для обработки входящих HTTP-запросов на страницы, рендеринга HTML-страницы с помощью EJS, отдачи статики в ответ на соответствующие запросы. Кроме того, express запускает сервер и начинает прослушивать входящие подключения. Библиотека Passport.js нужна для аутентификации пользователей.

Frontend-часть (клиентская часть) отвечает за создание и отображение пользовательского интерфейса. Состоит из HTML, CSS и JavaScript, которые выполняются в веб-браузере пользователя. Отвечает за обработку взаимодействия пользователя, визуализацию данных и управление состоянием приложения. Сюда входит CSS (стили для веб-интерфейса), также шаблоны Views для генерации страниц, JS-скрипты для динамики и интерактивности страниц, а ещё Fonts и Images – статические ресурсы.

Внутри сервера ещё присутствует MySQL (реляционная БД), используемая для хранения данных, таких как информация о загруженных файлах. А также веб-приложение использует файловую систему для хранения загруженных пользователями файлов. Эти файлы сохраняются в директориях parkaIn и parkaOut, созданных на сервере.

					ПГУ1.090301.04.001 ПЗ	Лист
						64
Изм.	Лист	№ докум.	Подпись	Дата		

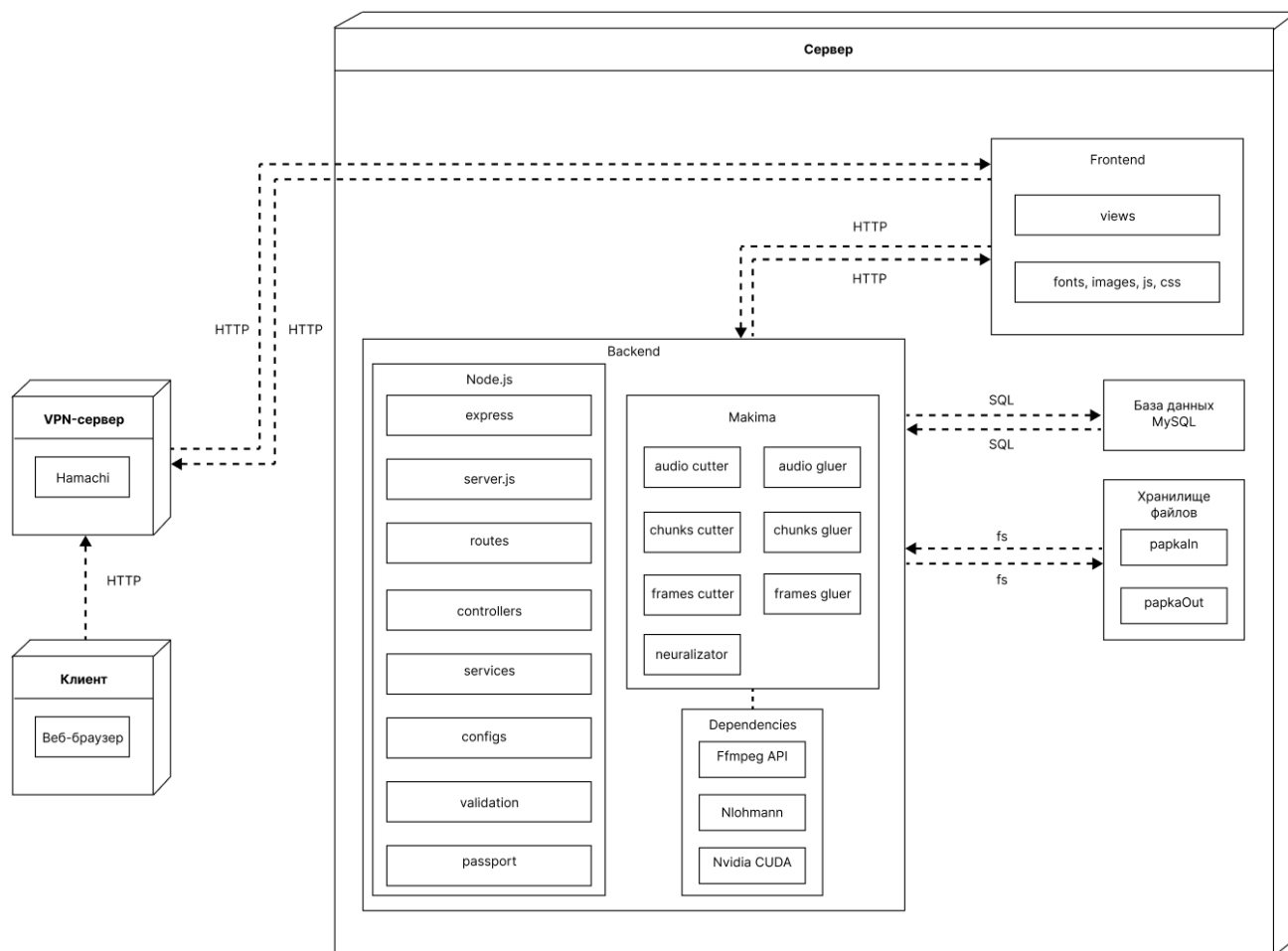


Рисунок 13 – UML-диаграмма развёртывания

На рисунке 14 можно увидеть различные запросы, их маршрутизацию, а также взаимодействие контроллеров и соответствующих им моделей (сервисов). Представленная UML-диаграмма компонентов используется для визуального представления структуры веб-приложения и взаимосвязей между его основными компонентами:

- views (Представления). Эти компоненты отвечают за отображение пользовательского интерфейса и взаимодействие с пользователем. Они получают данные от контроллеров и отображают их в виде, удобном для восприятия пользователем. Views в проекте представляют из себя EJS-шаблоны;
- запросы. Этот компонент представляет входящие запросы от пользователей, поступающие через веб-браузер. Пользовательские запросы

поступают в веб-приложение через HTTP-методы, такие как GET, POST, DELETE и т.д.;

– routes. Компоненты маршрутизации отвечают за связывание запросов пользователей с контроллерами. Они определяют, какой обработчик должен обрабатывать тот или иной запрос. Для этого им нужен URL-адрес, метод HTTP, а также некоторые дополнительные вводные. Внутри этих модулей содержатся конкретные пути запросов;

– controllers. Контроллеры являются связующим звеном между представлениями и моделями. Они получают запросы, обрабатывают их, взаимодействуют с сервисами для получения данных и передают эту информацию в представления;

– services – сервисы являются ключевым элементом, которые заняты бизнес-логикой приложения. В них заложены все необходимые функции для работы с данными. А контроллеры тут являются посредниками и с помощью сервисов выполняют задачи. Поэтому компонент «services» можно рассматривать как реализацию модели в MVC.

Эта UML-диаграмма последовательностей отражает основной поток управления и данных. Она помогает понять, как организовано взаимодействие между основными компонентами веб-приложения.

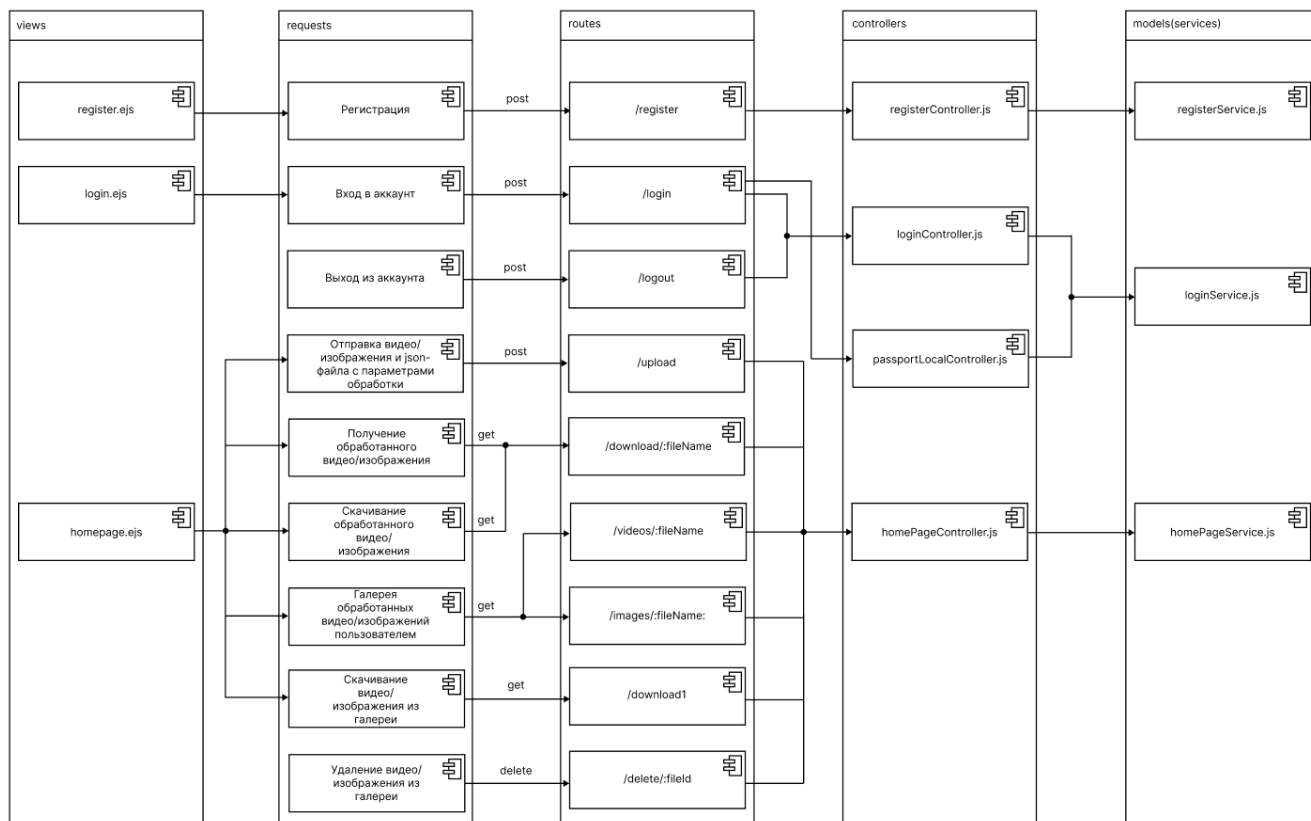


Рисунок 14 – UML-диаграмма компонентов

Для понимания взаимодействия пользователя с веб-приложением необходимо создать UML-диаграмму последовательностей.

Объектом на диаграмме последовательностей является пользователь – как внешний актер, инициирующий взаимодействие с системой, веб-приложение – как основной компонент системы, обрабатывающий запросы пользователя, база данных – как хранилище данных, к которому обращается веб-приложение, конвейер – как дополнительный компонент системы, отвечающий за обработку видео/изображений.

UML-диаграмма последовательностей, представленная на рисунке 15, будет показывать, как пользователь взаимодействует с веб-приложением, как веб-приложение обращается к базе данных для получения или сохранения данных, а также как веб-приложение задействует конвейер обработки видео при необходимости.

Такая диаграмма помогает наглядно отобразить динамику работы всей системы и взаимодействие её основных объектов.

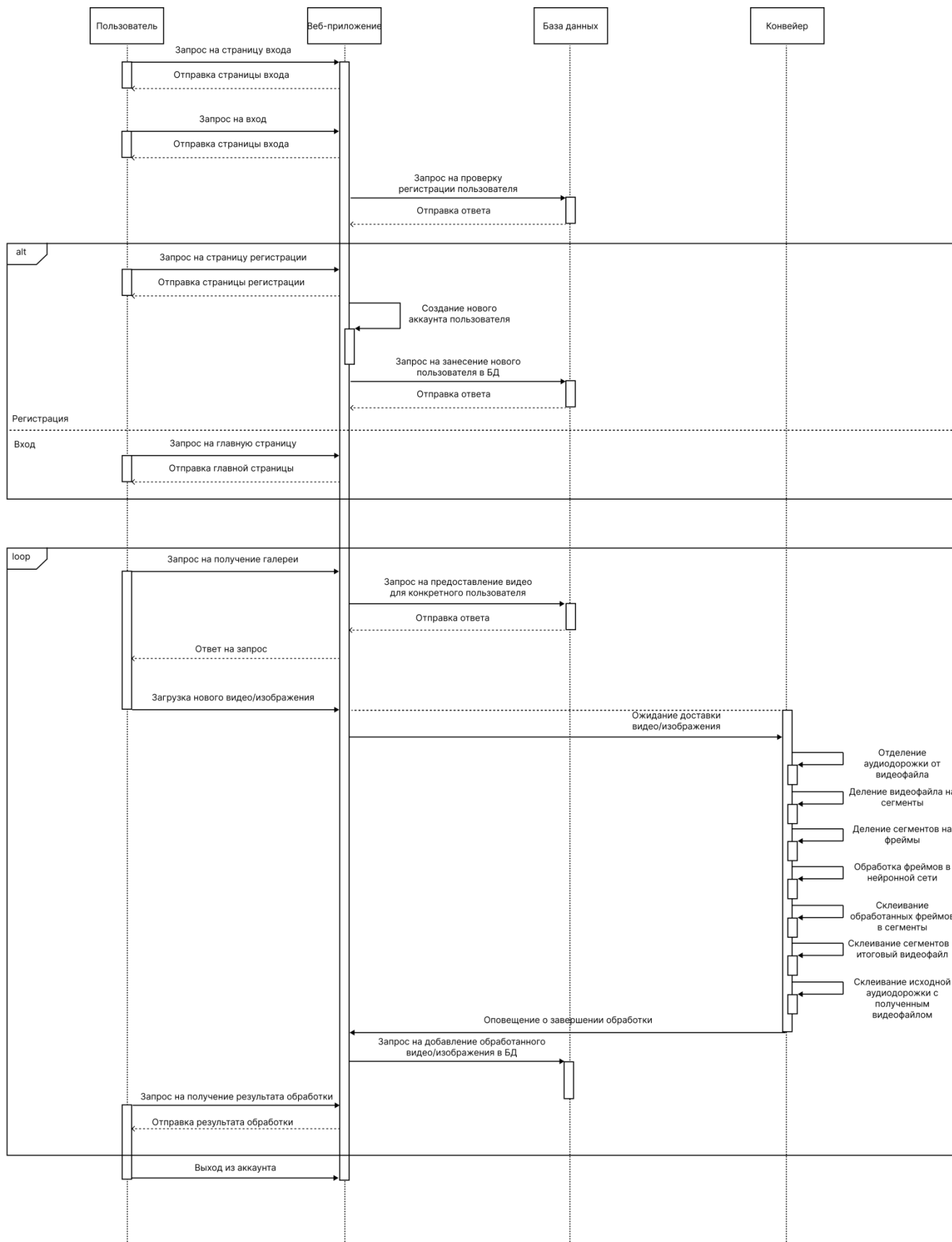


Рисунок 15 – UML-диаграмма последовательности

Функционал проекта можно описать, используя UML-диаграммы вариантов использования, а также деятельности.

Веб-приложение представлено в виде веб-сайта, который запускается в браузере пользователя. Оно позволяет регистрироваться пользователям, а далее под своей учётной записью выполнять различные действия на основной странице. Можно выбирать видео/изображения, выставлять различные пресеты параметров обработки, также параметры можно выбрать самостоятельно: FPS, нейросеть для обработки, разрешение – для видео, нейросеть для обработки, разрешение – для изображений. Далее можно отправить выбранное изображение или видео вместе с JSON-файлами выбранных параметров. После обработки изображения/видео пользователь может скачать файл на свой ПК, а наглядно увидеть результат работы сервиса можно увидеть внутри полей «до» и «после». Также пользователь может осуществлять просмотр галереи из ранее обработанных изображений или видео, выставлять фильтры по типу файлов, удалять и сохранять их из галереи. Выход из аккаунта тоже предусмотрен. Подробную диаграмму вариантов использования можно изучить на рисунке 16.

					ПГУ1.090301.04.001 ПЗ	Лист
						69
Изм.	Лист	№ докум.	Подпись	Дата		

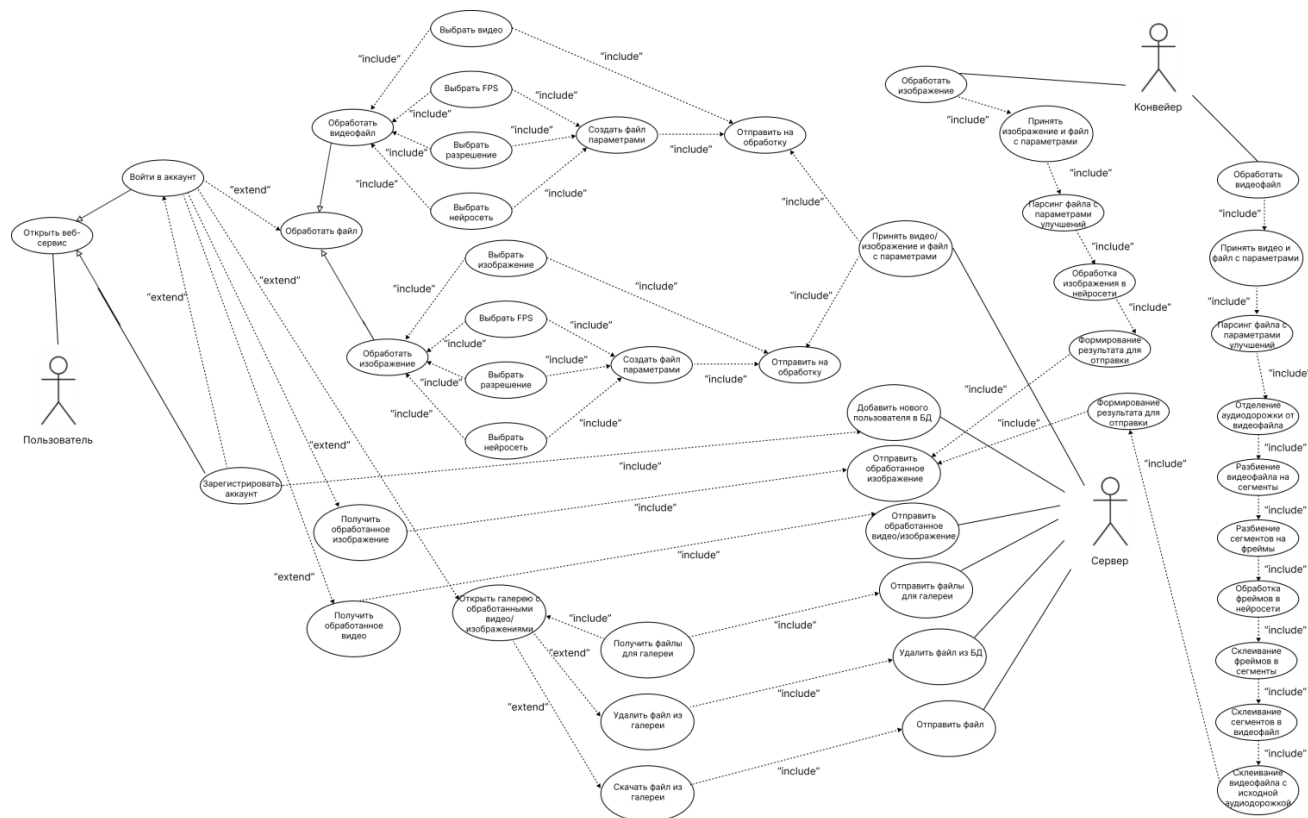


Рисунок 16 – UML-диаграмма вариантов использования

Для более точной демонстрации взаимодействия пользователя с веб-приложением составлена общая диаграмма деятельности. Диаграмма деятельности, приведённая на рисунке 17, наглядно показывает переход от одного действия к другому на примере запросов.

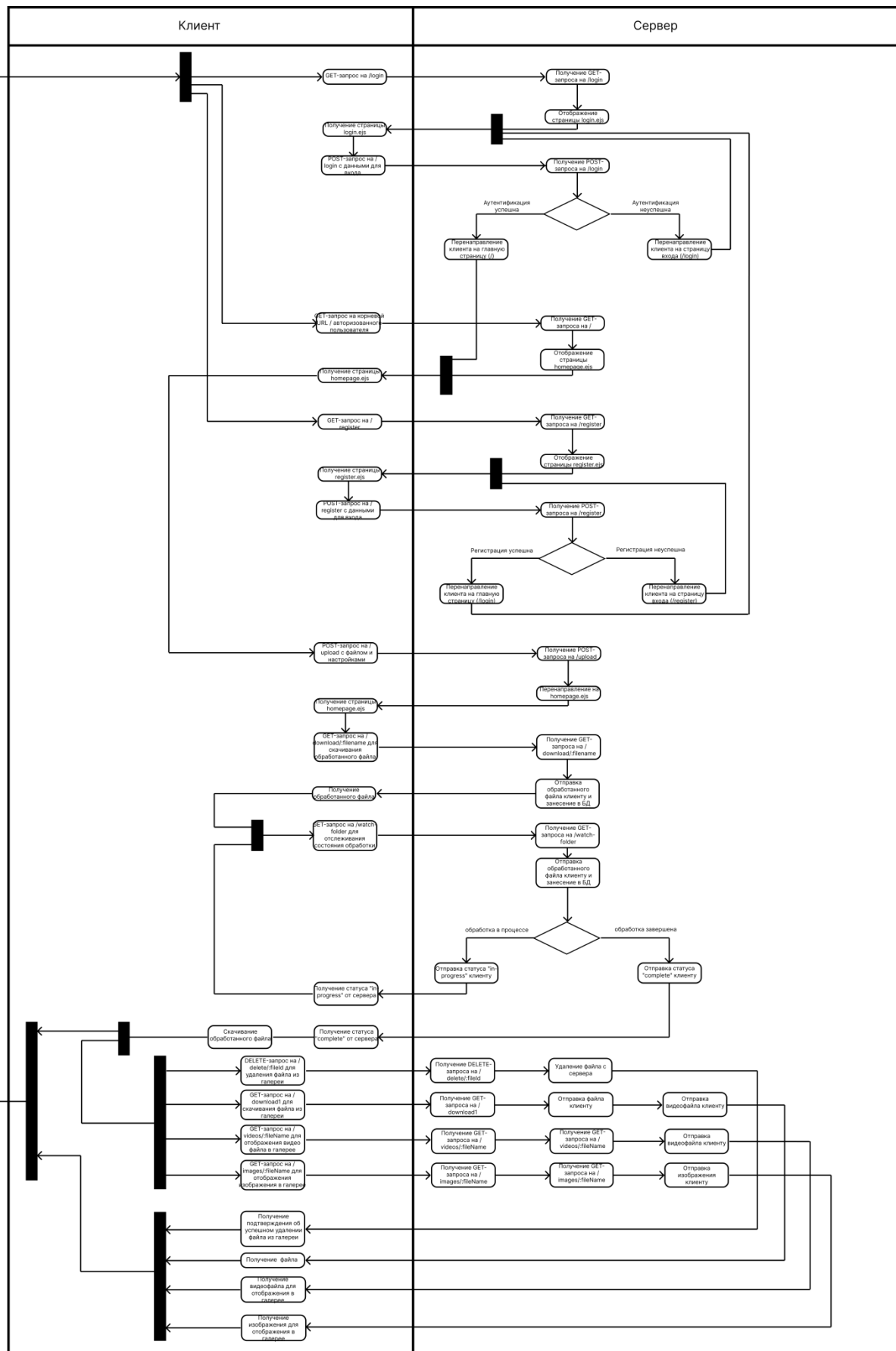


Рисунок 17 – UML-диаграмма деятельности

Взаимодействие клиента и сервера по каждому из запросов:

– GET /. Сначала клиент отправляет GET-запрос на корневой URL /. После этого его обрабатывает и отображает страницу homepage.ejs, которая содержит информацию для авторизованного пользователя;

– GET /login. Когда клиент хочет войти в систему, он отправляет GET-запрос на /login. Сервер отвечает, показывая страницу login.ejs, где нужно ввести свои данные: логин и пароль;

– POST /login. После отправки POST-запроса на /login с предоставленными учётными данными (email и пароль) сервер проверяет их достоверность. При успешной проверке, клиент перенаправляется на главную страницу (/);

– GET /register. Клиент отправляет GET-запрос на /register, чтобы открыть страницу регистрации. Сервер обрабатывает этот запрос и отображает страницу register.ejs, которая содержит форму для ввода данных нового пользователя;

– POST /register. После заполнения формы клиент отправляет POST-запрос на /register. Сервер проверяет введенные данные, создаёт нового пользователя в своей базе данных и перенаправляет на страницу входа (/login);

– POST /upload. Здесь клиент отправляет POST-запрос на /upload с файлом и настройками для обработки. Сервер обрабатывает этот запрос, сохраняет файл и настройки в соответствующих папках, и перенаправляет клиента на главную страницу (/);

– GET /download/:fileName. Клиент может скачать обработанный файл, отправив GET-запрос на адрес /download/:fileName. Сервер получает этот запрос, находит файл в соответствующей папке, использует настройки обработки из отдельного файла и отправляет обработанный файл клиенту;

– GET /watch-folder. Чтобы узнать о статусе обработки файла, клиент отправляет запрос GET на /watch-folder. Сервер проверяет, завершена ли обработка, и возвращает статус «in-progress» или «complete» клиенту;

- DELETE /delete/:fileId. При отправке DELETE-запроса на /delete/:fileId клиент инициирует удаление файла из галереи ранее обработанных файлов. Сервер находит файл, удаляет его и отправляет клиенту подтверждение об этом;
- GET /download1/. В случае, если клиент отправляет GET-запрос на /download1/, чтобы скачать обработанный файл из галереи, сервер обрабатывает этот запрос, находит файл в соответствующей папке и отправляет файл клиенту;
- GET /videos/:fileName. Клиент отправляет GET-запрос на /videos/:fileName, чтобы получить видео файл для отображения в галерее. Сервер обрабатывает этот запрос, ищет видео папке D:/papkaOut/{user.fullname}/videos и отправляет нужный файл клиенту;
- GET /images/:fileName. Также клиент отправляет GET-запрос на /images/:fileName, чтобы получить уже изображение для отображения в галерее. Сервер делает обработку этого запроса, находит изображение в папке D:/papkaOut/{user.fullname}/images и отправляет его клиенту.

1.7 Разработка веб-приложения

1.7.1 Настройка VPN

Предварительная настройка VPN (Virtual Private Network) необходима перед выполнением дипломного проекта для проверки взаимодействия связанных элементов через сетевой туннель. Сетевое туннелирование позволяет создать логическое соединение между двумя логическими точками посредством инкапсуляции различных протоколов. В рассматриваемом проекте конечными точками являются машины клиента и сервера.

Ранее тестирование работоспособности разрабатываемого программного продукта производилось путём разворачивания решения на серверной машине и использования данной машины в качестве клиентской. Однако требование архитектуры об использовании двух разных компьютеров в качестве клиента и сервера привело к использованию продукта LogMeIn Hamachi.

					ПГУ1.090301.04.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

Hamachi – программное обеспечение, предназначенное для построения VPN. Данное решение позволяет создать локальную сеть поверх сети Интернет для соединения рабочих станций с серым IP.

1.7.1.1 Установка Hamachi

Установка данной программы возможна с официального сайта LogMeIn Hamachi (рисунок 18). Необходимо две версии для операционных систем Linux и Windows.

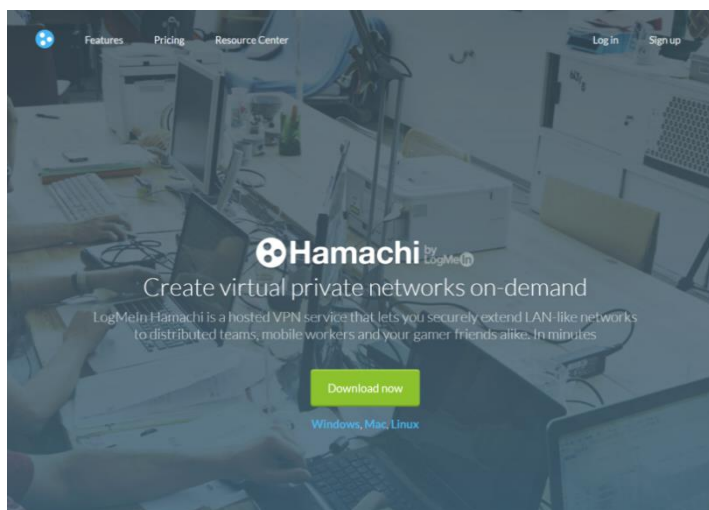


Рисунок 18 – Официальный сайт LogMeIn Hamachi

После скачивания и установки .exe файла (для Windows) и deb-пакета (для Linux) можно приступить к созданию собственного VPN.

1.7.1.1 Создание VPN

Создание VPN для Windows и Linux не имеет отличий, поэтому будет рассмотрена единая настройка для данных операционных систем. После запуска приложения Hamachi появляется основное окно с предложением создать новую сеть (рисунок 19).

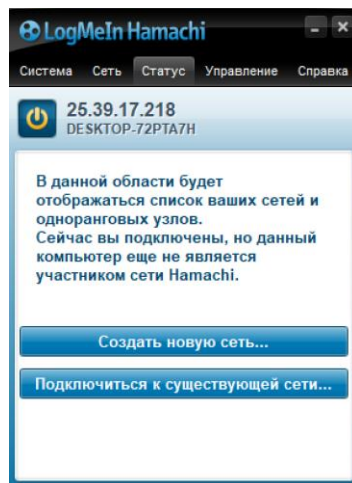


Рисунок 19 – Основное окно Hamachi

Создадим новую сеть. После нажатия на раздел «Создать новую сеть...» рядом с основным окном отобразится форма с данными для создания нового VPN (рисунок 20).

Рисунок 20 – Форма с параметрами нового VPN

В качестве имён для VPN могут использоваться наименования частей проекта: Німено (веб-приложение) и Makіma (конвейер). Заполним указанные поля и создадим сеть, нажав на одноимённую кнопку (рисунок 21).

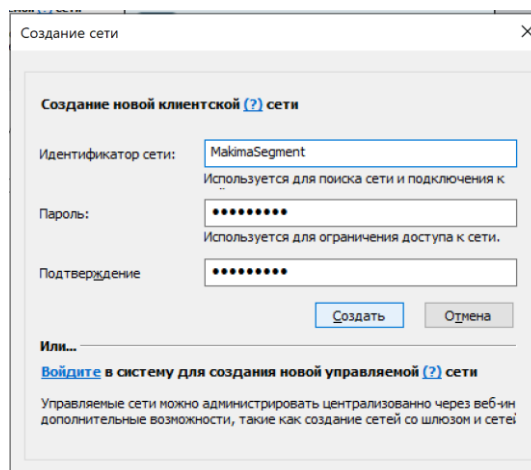


Рисунок 21 – Форма с заполненными параметрами нового VPN

После создания VPN, соответствующий элемент должен отобразиться в основном окне приложения (рисунок 22). Зелёная индикация обозначает отсутствие проблем в текущей сети, жёлтая – существуют незначительные проблемы, красная – сеть не работает.

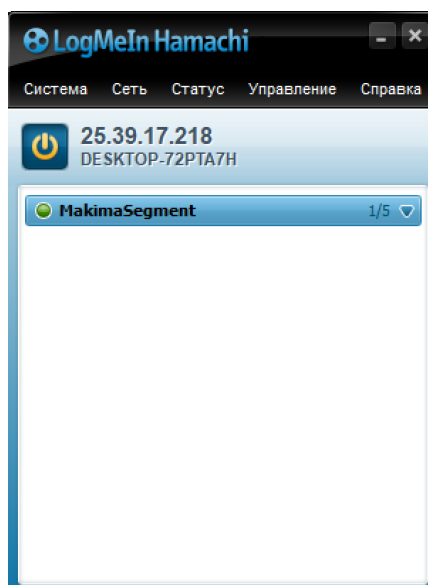


Рисунок 22 – Отображение созданного VPN

1.7.1.3 Добавление клиента в VPN

Для добавления компьютера, осуществляющего роль клиента, в созданный VPN и последующей активации туннеля требуется установить аналогичное программное обеспечение.

Завершив установку Hamachi на клиенте и открыв программу, следует активировать пункт «Подключиться к существующей сети...». Активация пункта приведёт к появлению формы для поиска соответствующего VPN. Используем данные, аналогичные вводимым при создании сети на серверной машине (рисунок 23).

Рисунок 23 – Заполненная форма для подключения к VPN

Успешность подключения определяется появлением клиентского компьютера в числе доступных при отображении основного окна (рисунок 24).

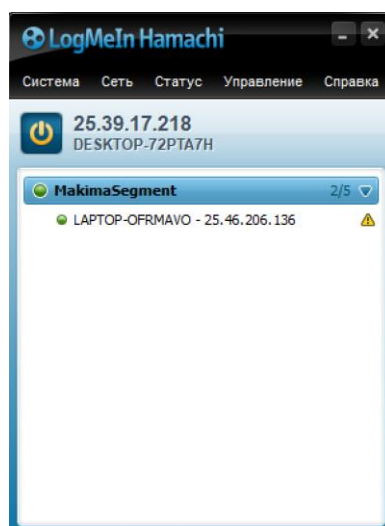


Рисунок 24 – Появление клиентской машины

Знак восклицания указывает на неполадки при построении туннеля между двумя машинами. Для понимания проблемы можно проверить доступность другой машины при помощи соответствующего раздела в выпадающем меню (рисунки 25 и 26).

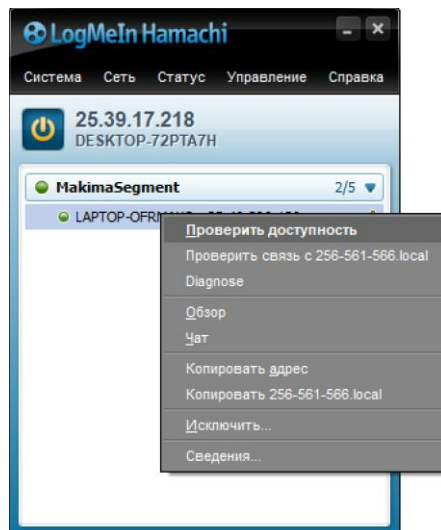


Рисунок 25 – Активация пункта выпадающего меню «Проверить доступность»

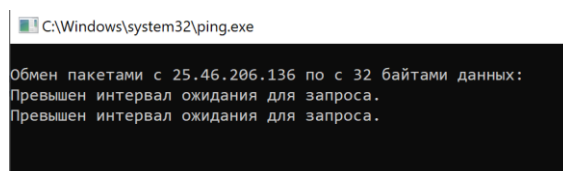


Рисунок 26 – Результат проверки доступности машины

Полученный результат проверки на доступность указывает на некорректную настройку сети. Для исправления затронутой проблемы требуется перейти в раздел «Система», подраздел «Параметры». Далее следует отключить шифрование и сжатие (рисунок 27).

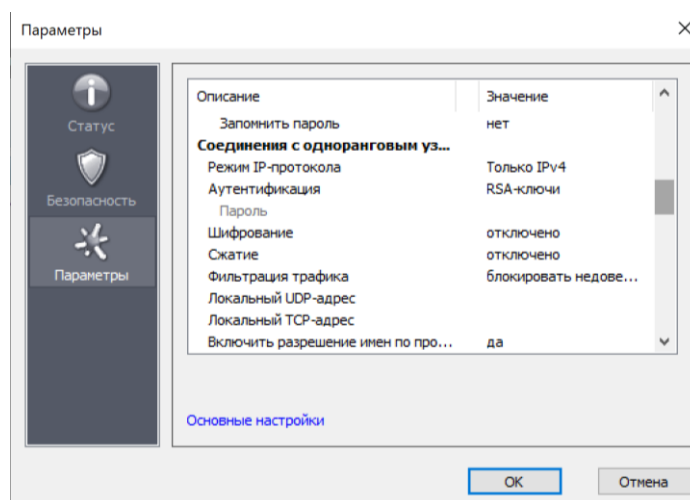


Рисунок 27 – Изменения в разделе «Параметры»

Повторное построение туннеля должно привести к успешной проверке на доступность.

1.7.2 Разработка клиентской части веб-приложения

В клиентской части веб-приложения графический интерфейс (представление) отделён от логики приложения (модель и контроллер). В проекте графический интерфейс пользователя реализован с использованием шаблонизатора EJS. Стили для оформления страниц вынесены в отдельный файл style.css. JS-скрипты добавляют динамичность графическому интерфейсу, работая с событиями и элементами DOM. Также используются статические ресурсы.

Страницы.

Сначала рассмотрим файл homepage.ejs. Он является основным шаблоном представления главной страницы веб-приложения и отвечает за отображение всех основных функциональных возможностей, доступных авторизованным пользователям.

Если рассмотреть код, то сначала идёт подключение CSS-файлов и библиотек JS.

Далее идёт кнопка «Выбрать файл для обработки», и поле выбора файла:

```
<input type="file" id="videoInput" name="video" style="display: none" />
<button id="selectVideoBtn" type="button"
class="button_design">Выбрать файл для обработки</button>
```

Тут селектор выбора пресетов с параметрами обработки (стандартный, максимальный, кастомный):

```
<div class="preset-checkboxes">
  <label><input type="radio" name="preset" value="standard"
class="preset-radio" checked>Стандартный</label>
</div>
```

Всплывающие меню для выбора параметров обработки, таких как FPS, нейронная сеть и разрешение находятся ниже пресетов.

Кнопка «Отправить на обработку» для отправки файла:

```
<button type="submit" id="sendAllBtn"
class="button_design">Отправить на обработку</button>
```

Кнопка «Скачать обработанный файл», которая изначально скрыта, нужна для скачивания обработанных файлов:

```
<button id="downloadButton" class="button_design"
style="display: none">
    Скачать обработанный файл
</button>
```

Ещё снизу страницы есть галерея ранее обработанных файлов. Здесь идёт цикл для перебора списка файлов, а также отображения этих файлов вместе с их описанием из БД:

```
<div class="gallery-container">
  <% if (videos.length > 0) { %>
    <% videos.forEach(video => { %>
      <div class="gallery-item">
        <!-- Отображение видео, изображений или ссылок на файлы
-->
        <div class="gallery-caption">
          <h3><%= video.title %></h3>
          <p><%= video.description %></p>
          <div class="action-container">
            <span class="delete-link" data-file-id="<%=
video.id %>">Удалить</span>
            <span class="download-link" data-file-path="<%=
video.file_path %>">Скачать</span>
          </div>
        </div>
      </div>
    <% }) %>
  <% } else { %>
    <p>Нет доступных файлов.</p>
  <% } %>
</div>
```

Также подключается файл `jscode.js`, который подключает код, ответственный за подключаемые скрипты для страницы.

```
<script src="/js/jscode.js"></script>
```

Следующий файл `login.ejs` уже относится к странице входа. Он отображает всё, что нужно для аутентификации пользователя. Страница использует Bootstrap для стилизации элементов. На ней есть фоновое изображение, а также форму

входа. Если при входе возникли ошибки, они отображаются в виде сообщений об ошибках, которые расположены над формой входа. Ошибки могут быть связаны с неверным email-адресом или паролем. Страница содержит поля для ввода email-адреса и пароля, а также кнопку «Войти». При отправке формы, данные передаются на сервер по адресу /login методом POST. В нижней части страницы расположены ссылки для навигации по веб-приложению. Можно перейти на страницу регистрации, если ещё нет аккаунта, или вернуться на главную страницу приложения.

Следующий шаблон представления – это register.ejs, который определяет структуру и внешний вид страницы регистрации в веб-приложении. Файл отвечает за отображение всех элементов графического интерфейса, необходимых для регистрации нового пользователя. Эта страница тоже использует Bootstrap, но уже содержит форму регистрации. Ошибки отображаются в виде предупреждающих сообщений, расположенных выше формы регистрации. Форма регистрации содержит поля для ввода имени, email-адреса, пароля и подтверждения введённого пароля. Поля, помеченные звёздочкой, обязательны для заполнения. При отправке формы данные передаются на сервер по адресу /register с помощью метода POST, где они обрабатываются и создаётся новая учётная запись. В нижней части страницы расположены навигационные по веб-приложению ссылки. Есть возможность перейти на страницу входа, если аккаунт уже есть, а также вернуться на главную страницу приложения.

Статическими файлами являются, например, изображения, такие как background.jpeg.

Стили (CSS) для оформления графического интерфейса представлены в файле style.css.

Например, дизайн кнопок в веб-приложении выглядит следующим образом:

```
.button_design {  
  display: inline-block;  
  text-decoration: none;
```

```

background-color: #ffffff;
color: #000000;
border: 1px solid #000000;
border-radius: 0px;
font-size: 16px;
padding: 8px 22px;
transition: all 0.2s ease;
}

.button_design:hover {
text-decoration: none;
background-color: #000000;
color: #ffffff;
border-color: #000000;
}

```

Скрипты.

Представлены файлом jscode.js, который является клиентским и отвечает за динамическое поведение и взаимодействие с пользовательским интерфейсом на основной странице веб-приложения.

Далее рассмотрим код.

Нажатие на кнопку «Выбрать файл для обработки» сохраняет файл в переменную selectedVideo для дальнейшей работы с ним.

```

const selectVideoBtn =
document.getElementById("selectVideoBtn");
const videoInput = document.getElementById("videoInput");

selectVideoBtn.addEventListener("click", () => {
  videoInput.click();
});

videoInput.addEventListener("change", () => {
  selectedVideo = videoInput.files[0];
});

```

В следующем фрагменте обрабатываются изменения выбора пресета настроек (стандартный, максимальный, кастомный) для обработки файла. В зависимости от выбранного пресета настроек код показывает или скрывает на веб-странице возможность выбора некоторых параметров, такие как разрешение, нейронная сеть и частота кадров, чтобы определённым типам выбранных файлов соответствовал определённый выбор из параметров обработки. Также за каждым

пресетом в зависимости от его названия закрепляются заранее выставленные параметры:

```
const presetRadios = document.querySelectorAll(".preset-  
radio");  
  
presetRadios.forEach((radio) => {  
  radio.addEventListener("change", (event) => {  
    const selectedPreset = event.target.value;  
    switch (selectedPreset) {  
      case "standard":  
        // Показываем или скрываем выпадающие меню с выбором  
        параметров в зависимости от выбранного пресета параметров  
        break;  
    }  
  });  
});
```

В этом фрагменте файл отправляется на сервер с помощью AJAX-запроса. Отправляется FormData с файлом и параметрами для его обработки в виде JSON-файла:

```
const uploadForm = document.getElementById("uploadForm");  
  
uploadForm.addEventListener("submit", (event) => {  
  event.preventDefault();  
  
  const formData = new FormData();  
  formData.append("video", selectedVideo);  
  formData.append("settings",  
JSON.stringify(videoSettingsJSON));  
  
  const xhr = new XMLHttpRequest();  
  xhr.open("POST", `/upload`, true);  
  xhr.send(formData);  
});
```

Далее описано, что будет при нажатии на кнопку «Скачать обработанный файл». Отправится AJAX-запрос на сервер, чтобы скачать обработанный файл:

```
const downloadButton =  
document.getElementById("downloadButton");  
  
downloadButton.addEventListener("click", () => {  
  const xhr = new XMLHttpRequest();  
  const filePath = `/download/${selectedVideoForDwnload}`;
```

```

const downloadUrl = `${window.location.origin}${filePath}`;

xhr.open("GET", downloadUrl, true);
xhr.responseType = "blob";
xhr.send();
});

```

Далее расписаны фильтры для галереи с изображениями и видео:

```

const filterImagesBtn =
document.getElementById("filterImages");
const filterVideosBtn =
document.getElementById("filterVideos");
const showAllBtn = document.getElementById("showAll");
const galleryItems = document.querySelectorAll(".gallery-
item");

filterImagesBtn.addEventListener("click", () => {
  galleryItems.forEach((item) => {
    if (item.querySelector(".image-container")) {
      item.style.display = "block";
    } else {
      item.style.display = "none";
    }
  });
});

filterVideosBtn.addEventListener("click", () => {
  galleryItems.forEach((item) => {
    if (item.querySelector(".video-container")) {
      item.style.display = "block";
    } else {
      item.style.display = "none";
    }
  });
});

showAllBtn.addEventListener("click", () => {
  galleryItems.forEach((item) => {
    item.style.display = "block";
  });
});

```

Удаление файлов из галереи нажатием на кнопку «Удалить» около снизу её элемента с помощью AJAX-запроса осуществляется здесь:

```

const deleteButtons = document.querySelectorAll(".delete-
link");

deleteButtons.forEach((button) => {

```

```

button.addEventListener("click", async (event) => {
    const fileId = event.target.dataset.fileId;

    try {
        const response = await fetch(`/delete/${fileId}`, {
            method: "DELETE",
        });

        if (response.ok) {
            event.target.closest(".gallery-item").remove();
            console.log("Файл успешно удален");
        } else {
            console.error("Ошибка при удалении файла");
        }
    } catch (error) {
        console.error("Ошибка при удалении файла:", error);
    }
});
});
});

```

Скачивание файлов из галереи нажатием на кнопку «Скачать» около снизу её элемента с помощью AJAX-запроса осуществляется здесь:

```

const downloadButtons = document.querySelectorAll(".download-link");

downloadButtons.forEach((button) => {
    button.addEventListener("click", async (event) => {
        event.preventDefault();
        const filePath = event.target.dataset.filePath;

        try {
            const response = await
fetch(`/download?fileName=${filePath}`, {
                method: "GET",
            });

            if (response.ok) {
                const fileBlob = await response.blob();
                const downloadLink = document.createElement("a");
                downloadLink.href = URL.createObjectURL(fileBlob);
                downloadLink.download = filePath.split("/").pop();
                document.body.appendChild(downloadLink);
                downloadLink.click();
                document.body.removeChild(downloadLink);
            } else {
                const errorMessage = await response.text();
                console.error("Ошибка при скачивании файла:",
errorMessage);
                alert(`Ошибка при скачивании файла: ${errorMessage}`);
            }
        }
    });
});

```

```
}
```

watchFolder() делает AJAX-запрос на сервер для получения статуса обработки. Кнопка «Скачать обработанный файл» также становится видимой, когда статус «complete». Ещё появляеся сообщение «Обработка завершена»:

```
function watchFolder() {
  $.ajax({
    url: "/watch-folder",
    type: "GET",
    success: function (data) {
      console.log("Server response:", data);
      if (data.status === "complete") {
        document.getElementById("upload-status").innerHTML =
          "Обработка завершена";
        clearInterval(watchIntervalId); // Останавливаем
        проверку папки
        document.getElementById("downloadButton").style.display
        = "inline";
      } else {
      }
    },
    error: function (xhr, status, error) {
      console.error("Error watching folder:", error);
    },
  });
}

// каждые 3 секунды
const watchIntervalId = setInterval(watchFolder, 3000);
```

Также есть события выбора файла, изменения пресета, отправки формы загрузки, скачивания файлов и фильтрация галереи:

```
selectVideoBtn.addEventListener("click", () => {
  videoInput.click();
});

videoInput.addEventListener("change", () => {
});

presetRadios.forEach((radio) => {
});

uploadForm.addEventListener("submit", (event) => {
});

downloadButton.addEventListener("click", () => {
});
```

```
filterImagesBtn.addEventListener("click", () => {
});
```

1.7.3 Разработка серверной части веб-приложения

На серверной части реализована маршрутизация, которая отвечает за правильное направление запросов (URL сопоставлены с контроллерами), исходящих от клиента на сервер. Также представлены контроллеры (взаимодействуют с моделями и передают данные в представления, организуя связь). Они занимаются обработкой запросов от клиента. Модели нужны для работы с данными из БД и для инкапсуляции логики приложения. Также есть файлы, которые отвечают за настройку и конфигурацию веб-приложения. В одном файле проекта есть код сравнения (валидации) данных для авторизации. И самый главный файл сервера – это server.js. Он отвечает за запуск сервера.

Конфигурация.

Подключение к БД настраивает DBConnection.js:

```
require("dotenv").config();
import mysql from "mysql2";

let connection = mysql.createConnection({
  host: process.env.DB_HOST,
  user: process.env.DB_USERNAME,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME,
});

connection.connect(function (err) {
  if (err) throw err;
  console.log("БД подключена!");
});

module.exports = connection;
```

Для создания соединения с БД DBConnection.js использует функционал библиотеки mysql2. Необходимые для подключения параметры (host, user, password, database) берутся из файла .env. Метод mysql.createConnection() создаёт объект соединения с базой данных, а connection.connect() устанавливает соединение с БД. Также здесь экспортируется

					ПГУ1.090301.04.001 ПЗ	Лист
						87
Изм.	Лист	№ докум.	Подпись	Дата		

объект connection, который может быть использован в других файлах проекта для взаимодействия с БД.

Внутри файла .env находятся конфиденциальные данные, в частности конфигурационные переменные окружения для реализуемого веб-приложения. Так, APP_HOST (localhost) нужна, чтобы определять хост, на котором будет запущено веб-приложение, PORT (3000) – для определения порта, на котором будет запущено веб-приложение, а также NODE_ENV, которая определяет среду, в которой работает веб-приложение. Кроме того, есть переменная DB_CONNECTION (mysql), определяющая тип БД, используемой приложением, переменная DB_HOST, которая определяет хост, на котором размещена база данных, в проекте это по-прежнему localhost, переменная DB_PORT (3306), которая нужна для определения порта, на котором будет доступна БД, DB_NAME – для определения названия базы данных, используемой приложением, в данном веб-приложении это mydatabase, DB_USERNAME – имя пользователя для подключения к БД, DB_PASSWORD – пароль пользователя для подключения к базе данных.

Все переменные из этого файла загружаются в DBConnection.js с помощью dotenv, что делает их доступными к использованию в веб-приложении.

```
APP_HOST=localhost
PORT=3000
NODE_ENV=developer

DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
DB_NAME=mydatabase
DB_USERNAME=root
DB_PASSWORD=34rtM!876
```

Файл viewEngine.js отвечает за настройку движка отображения (view engine) для веб-приложения.

Модуль express нужен для настроек веб-приложения:

```
import express from "express";
```

Функция конфигурации выполняет настройку движка отображения.

Аргумент app – экземпляр express.js:

```
let configViewEngine = (app) => {  
};
```

Доступ к статическим файлам:

```
app.use(express.static("./src/public"));
```

Установка движка отображения EJS:

```
app.set("view engine", "ejs");
```

Конфигурация экспортируется для использования в других местах:

```
module.exports = configViewEngine;
```

Контроллеры (controllers).

Первый контроллер, который мы рассмотрим, – это homePageController.js. Он отвечает за обработку запросов, связанных с домашней страницей веб-приложения.

Этот сервис содержит бизнес-логику для управления видеофайлами и изображениями пользователей, включая их добавление, удаление и получение информации о них:

```
const homePageService = require("../services/homePageService");
```

Обработчик запроса на домашнюю страницу:

```
const handleHelloWorld = async (req, res) => {  
};
```

Тут мы получаем список файлов с помощью функции getVideos из сервиса homePageService.js:

```
const videos = await homePageService.getVideos(req.user.id);
```

Статус загрузки файла берём из сессии пользователя, а потом очищаем переменную:

					ПГУ1.090301.04.001 ПЗ	Лист
						89
Изм.	Лист	№ докум.	Подпись	Дата		

```
const uploadStatus = req.session.uploadStatus;
req.session.uploadStatus = null;
```

Вызывается эта функция, чтобы сделать рендеринг страницы. Передаётся информация пользователя, список из файлов, статус загрузки и path:

```
res.render("homepage.ejs", {
  user: req.user,
  videos: videos,
  uploadStatus: uploadStatus,
  path: require("path"), // Передаем модуль 'path' в шаблон
});
```

Чтобы использовать в других частях кода:

```
module.exports = {
  handleHelloWorld,
};
```

Контроллер loginController.js отвечает за обработку запросов, связанных с авторизацией пользователей в веб-приложении. В самом начале кода импортируется функция validationResult для обработки результатов сравнения данных. А ещё loginService, содержащий бизнес-логику, связанную с авторизацией. Далее по коду идёт функция getPageLogin. Эта функция обрабатывает запросы на отображение страницы входа в систему. А handleLogin занимается обработкой запросы на вход. После идёт сравнение введённых данных. Если валидация успешна, срабатывает функция loginService.handleLogin() для аутентификации пользователя. После этого код перенаправляет на основную веб-страницу, иначе обратно на страницу входа. Функции checkLoggedIn и checkLoggedOut являются промежуточными функциями, которые проверяют авторизацию пользователя и перенаправляют его при необходимости. postLogout отвечает за выход из веб-приложения, перенаправляя на страницу входа.

Следующий контроллер – это passportLocalController.js, который обрабатывает аутентификацию пользователей в веб-приложении, отвечая за её логику. Сначала импортируется passport-local, Passport.js и loginService (бизнес-логика, связанная с авторизацией пользователей). По коду функция

initPassportLocal настраивает Passport.js с помощью LocalStrategy из модуля passport-local для определения параметров метода аутентификации (email и пароль). Далее при попытке аутентификации обработчик использует loginService для поиска пользователя по его email, потом сравнивает введённый пароль с паролем, который хранится в БД в таблице с пользователями. Эта функция возвращает пользователя. serializeUser сохраняет идентификатор пользователя в сессии. deserializeUser восстанавливает полные данные пользователя из БД по идентификатору. Благодаря этому данные пользователя будут правильно сохраняться и восстанавливаться в сессии.

А последний контроллер registerController.js отвечает за обработку запросов, связанных с регистрацией новых пользователей в веб-приложении. В нём импортируется registerService, функция validationResult, которая обрабатывает результаты сравнения данных. Запросы на отображение страницы регистрации обрабатывает getPageRegister. А вот самая главная функция createNewUser обрабатывает запросы на создание нового пользователя. Она проверяет поля с данным для регистрации, соответствуют ли они установленным требованиям. Если есть недочёты, то опять открывается страница регистрации. Иначе – создаётся новый объект пользователя с его введёнными данными. Потом этот объект поступает в сервис для сохранения в БД в таблицу с зарегистрированными пользователями. Если регистрация состоялась успешно, то нас перекидывает на страницу входа.

Маршрутизация.

Файл web.js отвечает за настройку маршрутизации (routing) веб-приложения.

Представленные маршруты в файле:

- / – для перехода к основной странице;
- /login – на страницу входа;
- /register – к странице регистрации;
- /logout – для запроса выхода из основной страницы на страницу входа.

В самом начале импортируются контроллеры для всех страниц, а также функция `initPassportLocal` для настройки аутентификации:

```
import express from "express";
import homePageController from
"../controllers/homePageController";
import registerController from
"../controllers/registerController";
import loginController from "../controllers/loginController";
import initPassportLocal from
"../controllers/passportLocalController";
```

Сначала вызывается функция `initPassportLocal`, настраивающая `Passport.js` для аутентификации:

```
initPassportLocal();
```

Далее идёт определение всех маршрутов (`initWebRoutes`):

```
let initWebRoutes = (app) => {
};
```

Для основной страницы. Происходит отображение страницы, проверка авторизации пользователя и обработка запроса на эту страницу:

```
router.get(
  "/",
  loginController.checkLoggedIn,
  homePageController.handleHelloWorld
);
```

Страница входа. Тут отображение страницы, проверка неавторизованности пользователя и обработка запроса на эту страницу входа:

```
router.get(
  "/login",
  loginController.checkLoggedOut,
  loginController.getPageLogin
);
```

Вход в веб-приложение. POST-запрос для входа, аутентификация пользователя, а также перенаправление на основную страницу, иначе — на страницу входа в веб-приложение:

					ПГУ1.090301.04.001 ПЗ	Лист
						92
Изм.	Лист	№ докум.	Подпись	Дата		

```

router.post(
  "/login",
  passport.authenticate("local", {
    successRedirect: "/",
    failureRedirect: "/login",
    successFlash: true,
    failureFlash: true,
  })
);

```

Регистрация. Отображение регистрационной страницы и обработка запроса для регистрации нового пользователя:

```

router.get("/register", registerController.getPageRegister);
router.post(
  "/register",
  auth.validateRegister,
  registerController.createNewUser
);

```

Выход из основной страницы веб-приложения:

```

router.post("/logout", loginController.postLogOut);

```

initWebRoutes экспортируем для использования в коде:

```

module.exports = initWebRoutes;

```

Модели (сервисы).

Среди сервисов есть loginService.js. Он содержит бизнес-логику, связанную с авторизацией пользователей в веб-приложении. Тут можно увидеть, что импортируется объект подключения к БД, а также импортируется библиотека bcrypt (для хэширования паролей). В этом сервисе handleLogin отвечает за обработку входа. Сначала по email-адресу ищется пользователь (findUserByEmail), потом пароль проходит проверку с тем, что находится в БД. Далее с помощью функции findUserByEmail происходит вычисление пользователя по его email-адресу в БД. Делается это с помощью SQL-запроса. Сравнением введенного пароля пользователя с паролем в БД занимается функция comparePassword. Все функции сервисов далее используются в контроллерах.

Следующий сервис – это `homePageService.js`, связанный с управлением видеофайлами и изображениями пользователей. Относится к главной странице, где всё это представлено.

После импорта объекта `connection` подключения к БД по коду описана функция `getVideos(userId)`, которая помогает получить файлы, с которыми взаимодействовал конкретный пользователь. Для этого выполняется SQL-запрос к БД:

```
const getVideos = async (userId) => {
  try {
    const [rows] = await connection.promise().query("SELECT *
FROM video_files WHERE user_id = ?", [userId]);
    return rows;
  } catch (err) {
    throw err;
  }
};
```

Следующая функция `saveFileToDatabase(req, fileName, filePath, fileType, description)` сохраняет информацию (название, расположение, тип, описание) об изображении или видео в БД:

```
const saveFileToDatabase = async (req, fileName, filePath,
fileType, description) => {
  const query = `
    INSERT INTO video_files (user_id, title, file_path,
file_type, description)
    VALUES (?, ?, ?, ?, ?)
  `;
  const values = [req.user.id, fileName, filePath, fileType,
description];
  try {
    await connection.promise().query(query, values);
    console.log("Видео/изображение сохранено в БД");
  } catch (err) {
    console.error("Ошибка сохранения видео/изображения в БД:",
err);
  }
};
```

За удаление изображения или видео из БД отвечает `deleteFileFromDatabase(fileId)`:

```
const deleteFileFromDatabase = async (fileId) => {
  try {
    await connection.promise().query("DELETE FROM video_files
WHERE id = ?", [fileId]);
  } catch (error) {
    console.error("Ошибка удаления файла из БД:", error);
    throw error;
  }
};
```

В свою очередь, `getFileInfoFromDatabase(fileName)` ответственна за получение видео или изображения из базы данных:

```
const getFileInfoFromDatabase = async (fileName) => {
  try {
    const [rows] = await connection.promise().query("SELECT *
FROM video_files WHERE file_path = ?", [fileName]);
    return rows;
  } catch (err) {
    throw err;
  }
};
module.exports = {
  getVideos,
  saveFileToDatabase,
  deleteFileFromDatabase,
  getFileInfoFromDatabase,
};
```

Файл `registerService.js` содержит бизнес-логику, связанную с регистрацией новых пользователей в веб-приложении, для обработки данных.

Точно так же импортируется объект подключения к БД, а потом `bcrypt` для хэширования паролей. В этом сервисе `createNewUser` создаёт нового пользователя. Она проверяет, существует ли уже пользователь с указанным email-адресом, используя функцию `checkExistEmail`. Пароль хешируется, если введённый email-адрес не существует среди зарегистрированных, потом создаётся новый объект пользователя. Функция `checkExistEmail` делает проверку существования зарегистрированного email-адреса в единственном виде, чтобы не было повторов.

Валидация.

В проекте это единственный файл `authValidation.js`, он содержит правила валидации для введенных данных в случае входа или в случае регистрации. `check` импортирована для определения правил валидации. Правила валидации для регистрации записаны в переменной `validateRegister` (с email-адресом, правильной ли длины пароль, совпали ли пароли). А правила, касающиеся входа, записаны в `validateLogin` (правильный ли email-адрес, введен пароль или нет).

`Server.js`.

Файл `server.js` является основным файлом сервера, который запускает приложение (сервер) и обрабатывает входящие запросы. Он отвечает за настройку и запуск `Express.js`-приложения.

Загружаются переменные окружения из `.env`:

```
require("dotenv").config();
```

Экземпляр `express`:

```
import express from "express";  
let app = express();
```

Настройка представлений:

```
configViewEngine(app);
```

Инициализация маршрутов:

```
initWebRoutes(app);
```

Далее по коду идёт маршрут `/upload` для обработки отправки файлов. На сервере создаются все нужные папки для файлов. Затем отправленный файл сохраняется в папку `inFolderPath`, а также сохраняются выбранные параметры обработки загруженного файла в `JSON`-файле.

В приведённом ниже коде приведены маршруты для отображения в галерее видео и изображений. Они дают возможность получать обработанные видео и изображения, чтобы потом отобразить их в галерее:

```
app.get("/videos/:fileName", (req, res) => {  
  const fileName = req.params.fileName;
```

```

    const fileNameWithoutExtension = path.basename(fileName,
path.extname(fileName));

    const filePath = path.join("D:", "papkaOut",
req.user.fullname, "videos", `${fileNameWithoutExtension}`,
fileName);

    fs.access(filePath, fs.constants.R_OK, (err) => {
        if (err) {
            console.error("Error accessing file:", err);
            res.status(404).send("File not found");
            return;
        }

        res.sendFile(filePath);
    });
});

app.get("/images/:fileName", (req, res) => {
    });
});

```

Снизу показан код маршрута, в котором обрабатываются запросы проверки состояния обработки, завершена она или нет. Делается это так, что маршрут определяет нужную папку с обработанным файлом, ждёт, когда там появится файл `makima.access`, это будет обозначать, что файл полностью загружен в папку и готов для скачивания. Если файл `makima.access` найден, то ответ «complete», иначе будет «in-progress»:

```

app.get("/watch-folder", (req, res) => {
    const fileName = req.session.uploadedFileName;
    const fileNameWithoutExtension = path.basename(fileName,
path.extname(fileName));

    const fileExtension =
path.extname(fileName).slice(1).toLowerCase();
    const folderPath = isFileExtensionAllowed(fileExtension,
["mp4", "mkv", "avi", "webm"])
        ? path.join("D:", "papkaOut", req.user.fullname, "videos",
fileNameWithoutExtension)
        : isFileExtensionAllowed(fileExtension, ["png", "jpg",
"webp", "gif"])
        ? path.join("D:", "papkaOut", req.user.fullname, "images",
fileNameWithoutExtension)
        : path.join("D:", "papkaOut", req.user.fullname,
fileNameWithoutExtension);

    // Проверяем, существует ли папка

```

```

    if (!fs.existsSync(folderPath)) {
        console.log("Folder not found:", folderPath);
        res.json({ status: "in-progress" });
        return;
    }

    console.log("Watching folder:", folderPath);

    // Проверяем наличие файла 'makima.access' в папке
    const checkFile = () => {
        fs.access(path.join(folderPath, "makima.access"),
fs.constants.F_OK, (err) => {
            if (!err) {
                console.log("makima.access найден в папке");
                req.session.uploadStatus = "Обработка завершена";
                res.json({ status: "complete" });
            } else {
                console.log("makima.access не найден в папке");
                res.json({ status: "in-progress" });
            }
        });
    };

    // Вызываем функцию проверки каждые 3 секунды
    const intervalId = setInterval(checkFile, 3000);

    // Очищаем интервал при завершении запроса
    res.on("close", () => {
        clearInterval(intervalId);
        console.log("Остановка просмотра папки:", folderPath);
    });

    checkFile(); // Первая проверка
});

```

За скачивание обработанных файлов отвечает маршрут `/download/:fileName`. Он определяет изображение это или видео, получает описание файла из JSON-файла с параметрами обработки, а также сохраняет информацию о файле в БД. После этого он позволяет скачать файл.

За удаление файлов из галереи отвечает маршрут `/delete/:fileId`. Он удаляет файлы из БД с помощью функции `deleteFileFromDatabase` из `homePageService`, и они пропадают из галереи:

```

app.delete("/delete/:fileId", async (req, res) => {
    const fileId = req.params.fileId;

    try {

```

```

        await homePageService.deleteFileFromDatabase(fileId);
        res.sendStatus(200);
    } catch (error) {
        console.error("Error deleting file:", error);
        res.sendStatus(500);
    }
});

```

Также есть маршрут `/download1` для скачивания файлов из галереи, который проверяет доступность файла и позволяет скачивать изображения или видео из БД.

В итоге определяется порт, сервер запускается и начинает слушать приходящие запросы:

```

let port = process.env.PORT || 8000;
app.listen(port, () =>
    console.log(`Веб-приложение запущено на порту ${port}!`)
);

```

1.7.4 Отношения между сущностями проекта

В проекте присутствует множество элементов, входящие в его структуру. Естественно, они связаны друг с другом. Так, контроллеры связаны с сервисами. Контроллеры обрабатывают запросы передачей логики в сервисы. Сервисы отвечают за бизнес-логику, такую как работа с пользователями и файлами, и взаимодействуют с базой данных. Например, если точнее, то `loginController` использует функции сервиса `loginService`, чтобы выполнились операции, относящиеся к функционалу входа в приложение, а `registerService` используется контроллером `registerController` для создания новых пользователей и занесения в БД информации о них. С базой данных в веб-приложении в основном связаны сервисы. Для этого используется файл `DBConnection` (соединение с БД).

Следующим уровнем связи является взаимосвязь между контроллерами и маршрутизатором (`web.js`). Все контроллеры экспортируют свои функции, чтобы их использовал маршрутизатор. `initWebRoutes` из файла маршрутизатора соотносит маршруты с функциями контроллеров для выполнения различных операций в веб-приложении. Привести пример можно на маршруте `/login`.

Данный маршрут использует `loginController.getPageLogin` и `loginController.handleLogin`.

Далее по логике организации проекта идут контроллеры, связанные с представлениями. Шаблоны демонстрируют в пользовательский интерфейс данные, которые им передали контроллеры. Контроллер `homePageController` по такому принципу отдаёт имя пользователя в шаблон `homepage.ejs`. То есть используются некоторые подготовленные данные для представления динамики в интерфейсе.

Некоторые контроллеры связаны с `jscode.js` (интерактивность на стороне клиента), который занят отправкой AJAX-запросов на сервер, чтобы выполнить разные действия, связанные с работой с видео, изображениями. Контроллеры, представленные на сервере, обрабатывают эти запросы, берут функции из сервисов и выполняют нужные действия. Для примера рассмотрим, как сделана загрузка файлов на сервер: `jscode.js` отправляет POST-запрос на маршрут `/upload`, а он в свою очередь использует функционал из сервиса `homePageService`, чтобы добавить информацию об этих файлах в БД.

`viewEngine.js` занят настройкой EJS и описывает пути для статики в проекте. `authValidation.js` определяет правила валидации для регистрации и авторизации пользователей. `passportLocalController` настраивает стратегию аутентификации `Passport.js`, которая используется для проверки учётных данных пользователей.

Каждый элемент делает свои задачи, но для полной картины он связан с другими элементами проекта.

1.7.5 Разработка БД

1.7.5.1 Взаимодействие с MySQL Workbench

Чтобы создать базу данных MySQL можно использовать приложение MySQL Workbench 8.0 CE. Выбор в сторону данной программы сделан из-за высокой популярности данного программного решения среди подобных инструментов. Также в число факторов входят:

					ПГУ1.090301.04.001 ПЗ	Лист
						100
Изм.	Лист	№ докум.	Подпись	Дата		

- наличие кроссплатформенности. MySQL Workbench доступен для операционных систем Linux, Windows и macOS. Это позволяет продолжить разработку в случае порчи образа основной системы на резервной машине;
- свободное распространение. При последующей реализации веб-приложения как стратап-проекта, используемая база данных не создаёт обратной зависимости от собственной лицензии;
- удобный интерфейс взаимодействия. В частности, можно отметить наличие возможности представить модель БД в графическом виде. Графическое представление упрощает понимание общей модели;
- наличие дополнительных функций. Например, наличие функции Reverse Engineering позволяет восстановить структуру таблиц и связей из той, что ранее была реализована и хранится на сервере БД;
- редактор SQL-запросов. Позволяет просмотреть ответ, полученный от сервера в табличном виде.

1.7.5.2 Установка MySQL Workbench

Для установки MySQL Workbench нужно перейти на официальный сайт MySQL (<https://www.mysql.com/products/workbench/>) и скачать последнюю версию MySQL Workbench для операционной системы Windows (ОС на машине разработчика). Данный процесс описан на рисунках 28 и 29.

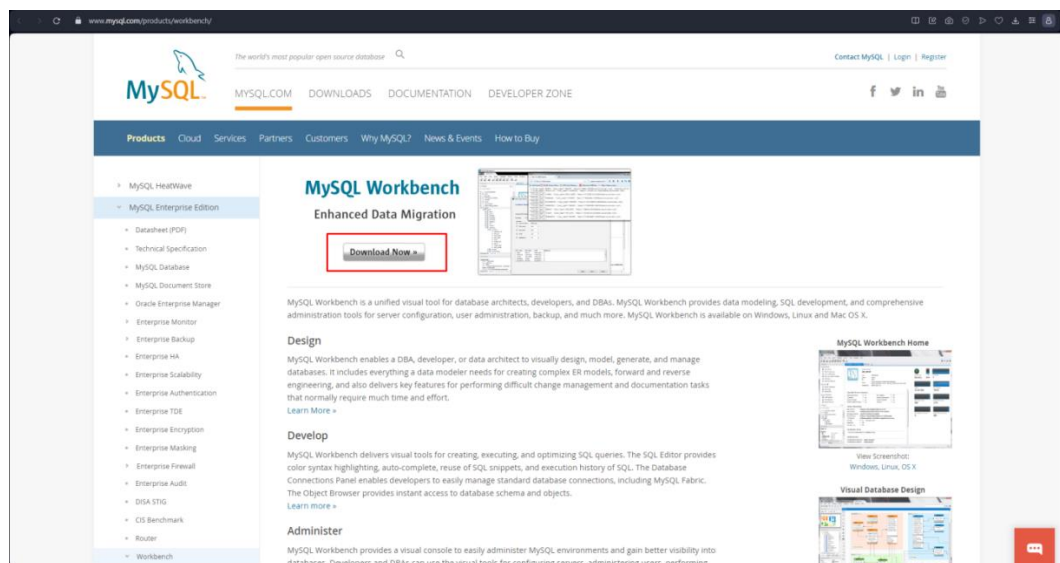


Рисунок 28 – Официальный сайт MySQL Workbench

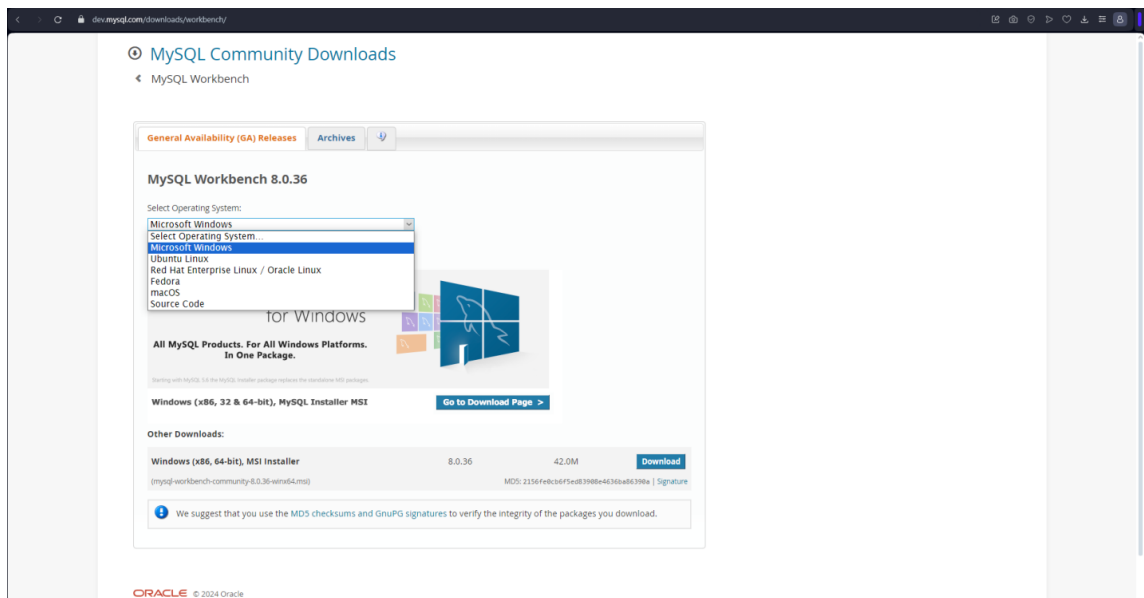


Рисунок 29 – Загрузка MySQL Workbench для Windows

После загрузки необходимо запустить процесс установки. Для этого нужно выделить и запустить скачанный файл, выбрать пункт «Установить» и принять лицензионное соглашение. После установки для запуска приложения требуется перейти в меню «Пуск» и найти ярлык «MySQL Workbench».

При первом запуске приложение может предложить установить дополнительные компоненты, такие как MySQL Server и MySQL Router. Можно продолжить установку, либо отказаться от неё в угоду скорости разработки. Далее следует подключиться к серверу MySQL. Необходимо в MySQL Workbench нужно перейти во вкладку «Connection Management», нажать на элемент «+» для создания нового подключения к серверу MySQL и заполнить необходимые параметры (хост, порт, имя пользователя и пароль). Пример заполнения параметров показан на рисунке 30.

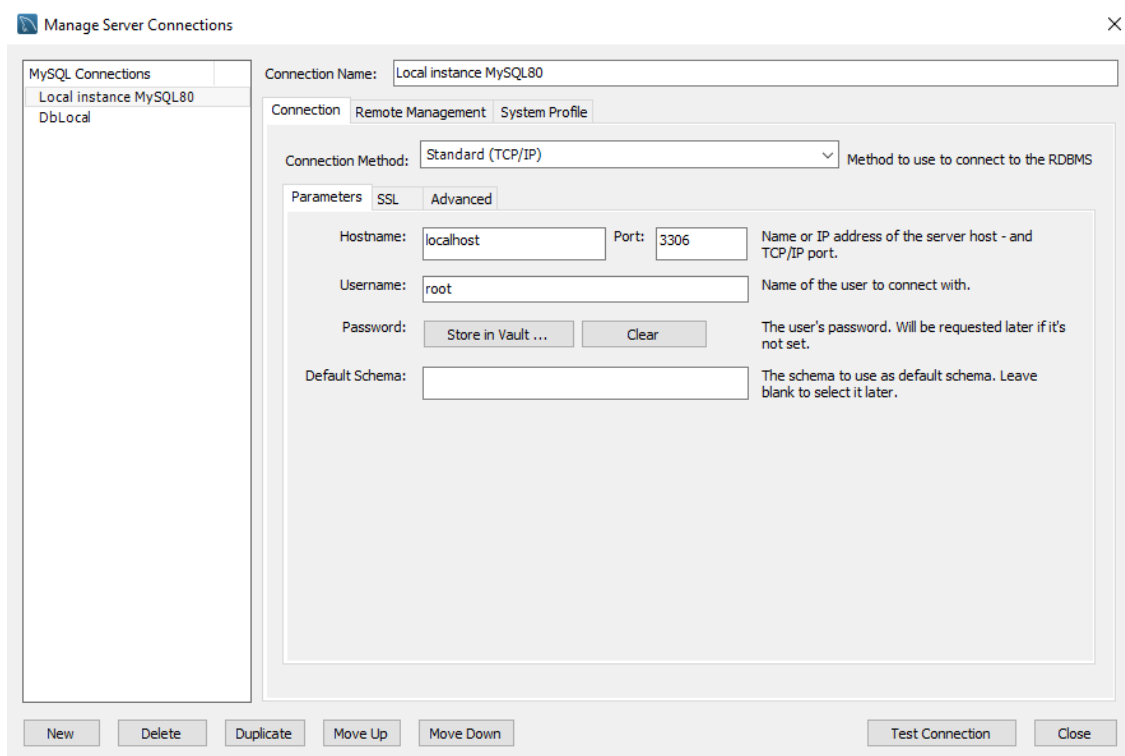


Рисунок 30 – Manage Server Connections

После завершения этих шагов можно продолжать работу с MySQL Workbench 8.0 CE и управлять созданными базами данных MySQL.

1.7.4.3 Создание БД

Создание БД начинается с подключения к серверу MySQL. Для этого был выбран пункт Local instance MySQL80 и введён пароль (рисунок 31).

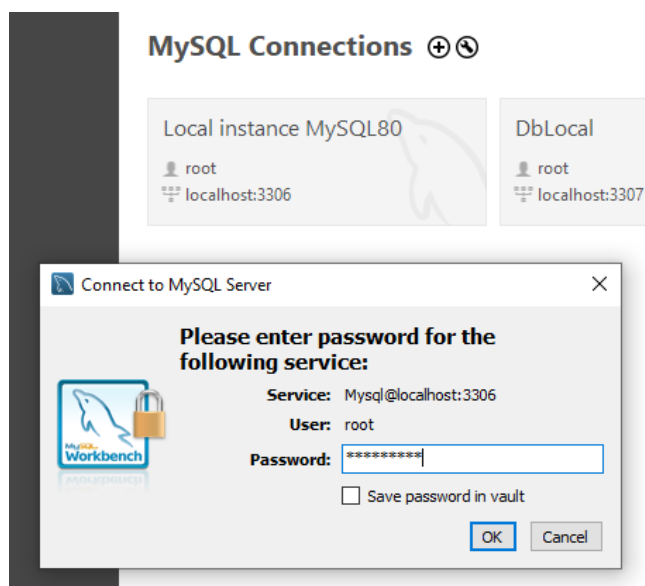


Рисунок 31 – Подключение к серверу MySQL

Изм.	Лист	№ докум.	Подпись	Дата

Далее создаём БД mydatabase (рисунок 32).

id	user_id	title	file_path	file_type	description
98	4	trap-nation.mp4	D:\papakOut\shrek\videos\trap-nation\trap-nation.mp4	video	FPS: 120, Neural: realesrgan, Resolution: 3840x2160
131	4	logo.png	D:\papakOut\shrek\images\logo\logo.png	image	Neural: realesrgan, Resolution: 3840x2160
134	4	Test_Mouse (7).mp4	D:\papakOut\shrek\videos\Test_Mouse (7)\Test_Mouse (7).mp4	video	FPS: 30, Neural: waifu2x, Resolution: 800x600
135	4	Test_Mouse.mp4	D:\papakOut\shrek\videos\Test_Mouse\Test_Mouse.mp4	video	FPS: 120, Neural: realesrgan, Resolution: 3840x2160
138	4	Test_Mouse (2).mp4	D:\papakOut\shrek\videos\Test_Mouse (2)\Test_Mouse (2).mp4	video	FPS: 120, Neural: realesrgan, Resolution: 3840x2160
140	4	Test_Mouse (14)....	D:\papakOut\shrek\videos\Test_Mouse (14)\Test_Mouse (14).mp4	video	FPS: 60, Neural: realcugan, Resolution: 2560x1600
141	4	logo (12).png	D:\papakOut\shrek\images\logo (12)\logo (12).png	image	Neural: realcugan, Resolution: 2560x1440
142	4	logo (11).png	D:\papakOut\shrek\images\logo (11)\logo (11).png	image	Neural: realcugan, Resolution: 1440x1080
143	4	logo (11).png	D:\papakOut\shrek\images\logo (11)\logo (11).png	image	Neural: realcugan, Resolution: 1440x1080
144	4	maddysen-ob-tpedia...	D:\papakOut\shrek\videos\maddysen-ob-tpedia (2)\maddysen-ob-tpedia (2).mp4	video	FPS: 120, Neural: realcugan, Resolution: 2048x1536
145	4	maddysen-ob-tpedia...	D:\papakOut\shrek\videos\maddysen-ob-tpedia (2) (1)\maddysen-ob-tpedia (2) (1).mp4	video	FPS: 30, Neural: realesrgan, Resolution: 3200x1800
147	4	Test_Mouse (2) (2)...	D:\papakOut\shrek\videos\Test_Mouse (2) (2) (1)\Test_Mouse (2) (2) (1).mp4	video	FPS: 60, Neural: realcugan, Resolution: 2560x1600
148	7	VinipooH.mp4	D:\papakOut\shrek\videos\VinipooH\VinipooH.mp4	video	FPS: 60, Neural: realcugan, Resolution: 3840x2160
149	7	logo.png	D:\papakOut\shrek\images\logo\logo.png	image	Neural: waifu2x, Resolution: 1600x1200
150	8	VinipooH.mp4	D:\papakOut\shrek\videos\VinipooH\VinipooH.mp4	video	FPS: 60, Neural: realcugan, Resolution: 3840x2160
151	9	VinipooH.mp4	D:\papakOut\shrek\videos\VinipooH\VinipooH.mp4	video	FPS: 60, Neural: realcugan, Resolution: 3840x2160

Рисунок 32 – БД mydatabase

Далее создаём таблицу video_files в этой базе. Нужно нажать на кнопку «Создать таблицу» рядом с названием базы данных (рисунок 33).

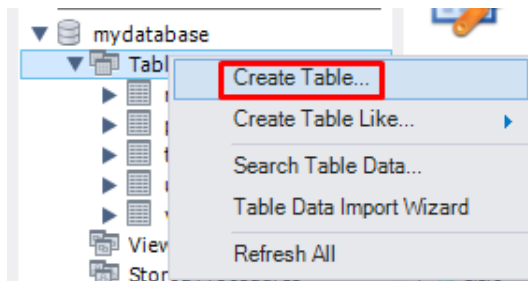


Рисунок 33 – Создание таблицы

Это откроет редактор таблиц, где можно настроить структуру новой таблицы (рисунок 34).

Table Name: Schema: mydatabase

Charset/Collation: Default Charset Default Collation Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	☒	☒	☐	☐	☐	☐	☐	☐	
user_id	INT	☐	☐	☐	☐	☐	☐	☐	☐	
title	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	
file_path	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	
file_type	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	
description	LONGTEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Column Name: Data Type:

Charset/Collation: Default:

Comments:

Storage: ☐ Virtual ☐ Stored ☐ Primary Key ☐ Not Null ☐ Unique ☐ Binary ☐ Unsigned ☐ Zero Fill ☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

Рисунок 34 – Настройка структуры новой таблицы video_files

После установки параметров нужно нажать на кнопку «Подтвердить» (рисунок 35).

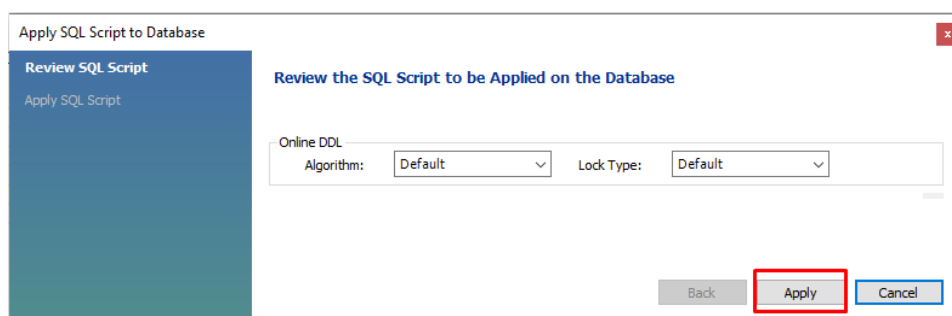


Рисунок 35 – Подтверждение создания таблицы video_files

Далее нажимаем на кнопку «Финиш», завершая создание таблицы (рисунок 36).

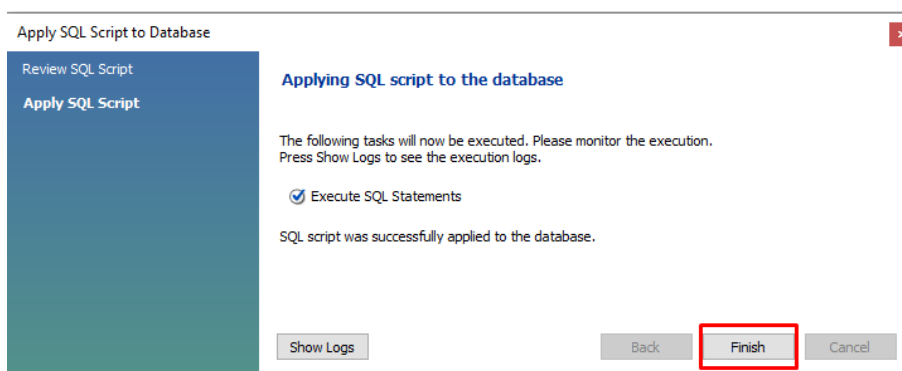


Рисунок 36 – Окончание создания таблицы video_files

Получилась таблица video_files для хранения информации об обработанных видеофайлах и изображениях (рисунки 37, 38):

- id – уникальный идентификатор файла (первичный ключ);
- user_id – идентификатор пользователя, который получит этот файл (внешний ключ, связан с таблицей users);
- title – название файла;
- file_path – путь к обработанному файлу на сервере;
- file_type – тип файла (image или video);
- description – описание файла, включающее в себя параметры обработки.

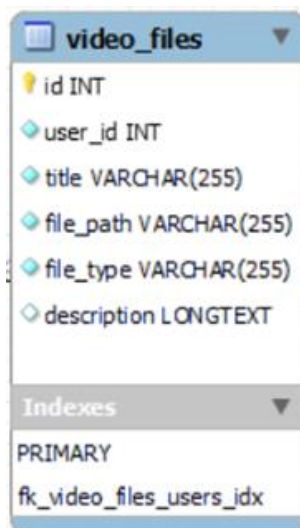


Рисунок 37 – Таблица video_files

id	user_id	title	file_path	file_type	description
98	4	trap-nation-fart.mp4	D:\pafkaOut\shrek\videos\trap-nation-fart\trap-nation-fart.mp4	video	FPS: 120, Neural: realesrgan, Resolution: 3840x2160
131	4	logo.png	D:\pafkaOut\shrek\images\logo\logo.png	image	Neural: realesrgan, Resolution: 3840x2160
134	4	Test_Mouse (7).mp4	D:\pafkaOut\shrek\videos\Test_Mouse (7)\Test_Mouse (7).mp4	video	FPS: 30, Neural: waifu2x, Resolution: 800x600
135	4	Test_Mouse.mp4	D:\pafkaOut\shrek\videos\Test_Mouse\Test_Mouse.mp4	video	FPS: 120, Neural: realesrgan, Resolution: 3840x2160
138	4	Test_Mouse (2).mp4	D:\pafkaOut\shrek\videos\Test_Mouse (2)\Test_Mouse (2).mp4	video	FPS: 120, Neural: realcugan, Resolution: 2560x1440
140	4	Test_Mouse (14)....	D:\pafkaOut\shrek\videos\Test_Mouse (14)\Test_Mouse (14).mp4	video	FPS: 60, Neural: realcugan, Resolution: 2560x1600
141	4	logo (12).png	D:\pafkaOut\shrek\images\logo (12)\logo (12).png	image	Neural: realcugan, Resolution: 2560x1440
142	4	logo (11).png	D:\pafkaOut\shrek\images\logo (11)\logo (11).png	image	Neural: realcugan, Resolution: 1440x1080
143	4	logo (11).png	D:\pafkaOut\shrek\images\logo (11)\logo (11).png	image	Neural: realcugan, Resolution: 1440x1080
144	4	maddysen-ob-tpedia (2).mp4	D:\pafkaOut\shrek\videos\maddysen-ob-tpedia (2)\maddysen-ob-tpedia (2).mp4	video	FPS: 120, Neural: realcugan, Resolution: 2048x1536
145	4	maddysen-ob-tpedia (2) (1).mp4	D:\pafkaOut\shrek\videos\maddysen-ob-tpedia (2) (1)\maddysen-ob-tpedia (2) (1).mp4	video	FPS: 30, Neural: realesrgan, Resolution: 3200x1800
147	4	Test_Mouse (2) (2) (1).mp4	D:\pafkaOut\shrek\videos\Test_Mouse (2) (2) (1)\Test_Mouse (2) (2) (1).mp4	video	FPS: 60, Neural: realcugan, Resolution: 2560x1600
148	7	Vinipooh.mp4	D:\pafkaOut\Misha\images\Vinipooh\Vinipooh.mp4	video	FPS: 60, Neural: realcugan, Resolution: 3840x2160
149	7	logo.png	D:\pafkaOut\Misha\images\logo\logo.png	image	Neural: waifu2x, Resolution: 1600x1200
150	8	Vinipooh.mp4	D:\pafkaOut\user2\videos\Vinipooh\Vinipooh.mp4	video	FPS: 60, Neural: realcugan, Resolution: 3840x2160
151	9	Vinipooh.mp4	D:\pafkaOut\user1\videos\Vinipooh\Vinipooh.mp4	video	FPS: 60, Neural: realcugan, Resolution: 3840x2160

Рисунок 38 – Заполненная таблица video_files

Теперь база данных mydatabase с таблицей video_files успешно существует в MySQL Workbench. В созданную БД можно добавлять множество других связанных друг с другом таблиц. Такая структура данных обеспечивает гибкость и масштабируемость системы, позволяя легко расширять ее функциональность в будущем. За настройку подключения к базе данных отвечает DBConnection.js.

Для веб-приложения была разработана БД с определёнными типами таблиц.

users – таблица для хранения информации о пользователях:

- id – уникальный идентификатор пользователя (первичный ключ);
- fullname – полное имя пользователя;
- email – адрес электронной почты пользователя;

- password – хэшированный пароль пользователя.
- video_files – таблица для хранения информации об обработанных файлах:
 - id – уникальный идентификатор файла (первичный ключ);
 - user_id – идентификатор пользователя, который получит этот файл (внешний ключ, связан с таблицей users);
 - title – название файла;
 - file_path – путь к обработанному файлу на сервере;
 - file_type – тип файла (image или video);
 - description – описание файла, включающее в себя параметры обработки.
- presets – таблица для хранения информации о пользовательских пресетах:
 - id – уникальный идентификатор пресета (первичный ключ);
 - user_id – идентификатор пользователя, который создал этот пресет (внешний ключ, связан с таблицей users);
 - preset – выбранные параметры;
 - preset_type – тип пресета (например, «video», «image»).
- neuro – таблица для хранения информации о нейронных сетях:
 - id – уникальный идентификатор нейронной сети (первичный ключ);
 - name – название нейронной сети;
 - difficult – сложность нейронной сети;
 - availability – доступность нейронной сети.
- video_files_raw – таблица для хранения информации о необработанных загруженных файлах:
 - id – уникальный идентификатор необработанного видеофайла (первичный ключ);
 - user_id – идентификатор пользователя, который отправил этот видеофайл (внешний ключ, связан с таблицей users);
 - title – название необработанного видеофайла;
 - file_path – путь к необработанному файлу на сервере;
 - file_type – тип необработанного файла (image или video);

- description – описание файла, включающее в себя параметры обработки.
- servers – таблица для хранения информации о серверах:
- id – уникальный идентификатор сервера (первичный ключ);
- ip – IP-адрес сервера;
- public_key – публичный ключ сервера;
- priority – приоритет сервера;
- neuro_id – идентификатор нейронной сети, используемой на сервере (внешний ключ, связан с таблицей neuro).

Таблица типа video_files имеет связь с другой таблицей типа users через элемент (внешний ключ) user_id. Каждый файл после обработки подлежит доставке определённому пользователю, который ранее отправлял оригинал данного файла. Тип таблицы presets связан с типом таблицы users через поле user_id. Каждый из пресетов принадлежит установленному пользователю, являющимся создателем пресета.

Таблица video_files_raw соотносится с таблицей users через элемент user_id, где каждый необработанный видеофайл принадлежит определённому пользователю, добавившему данный файл на обработку. Если рассмотреть таблицу servers, то можно заметить связь с таблицей neuro через поле neuro_id. Это обозначение того, что на каждом сервере используется конкретная нейронная сеть.

Описанная выше структура БД гарантирует удобство и гибкость в управлении основными сущностями:

- пользователями. Их учётными данными и загруженными файлами;
- пресетами параметров. Используемые пользователями для выбора уровней обработки видео;
- информацией о нейронных сетях;
- информацией о серверах. В частности, об установленных на них моделях нейронных сетей.

Физическая модель БД приведена на рисунке 39.

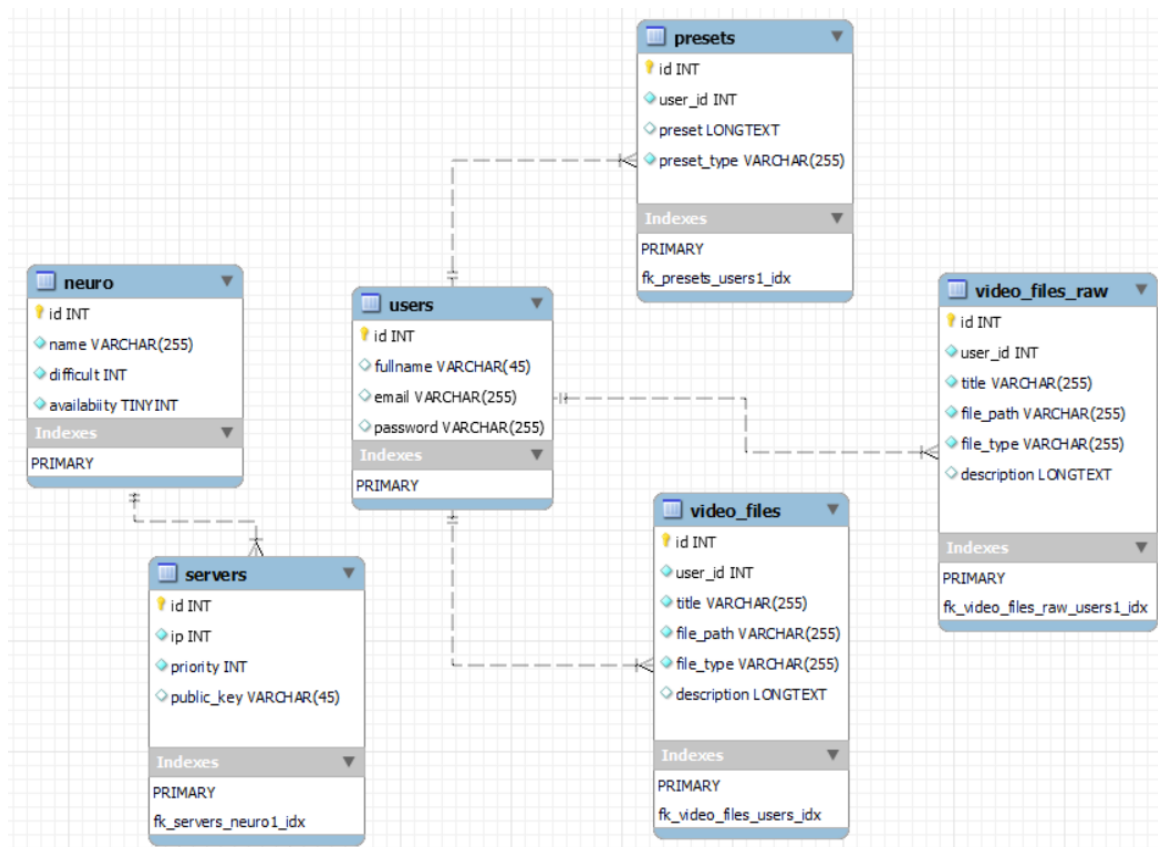


Рисунок 39 – Физическая модель базы данных

1.8 Тестирование веб-приложения

Протестируем веб-приложение на ОС Windows. Для этого нужно зайти на сайт веб-приложения.

Пользователя встречает страница входа с полями ввода данных email-адреса и пароля для авторизации (рисунок 40).

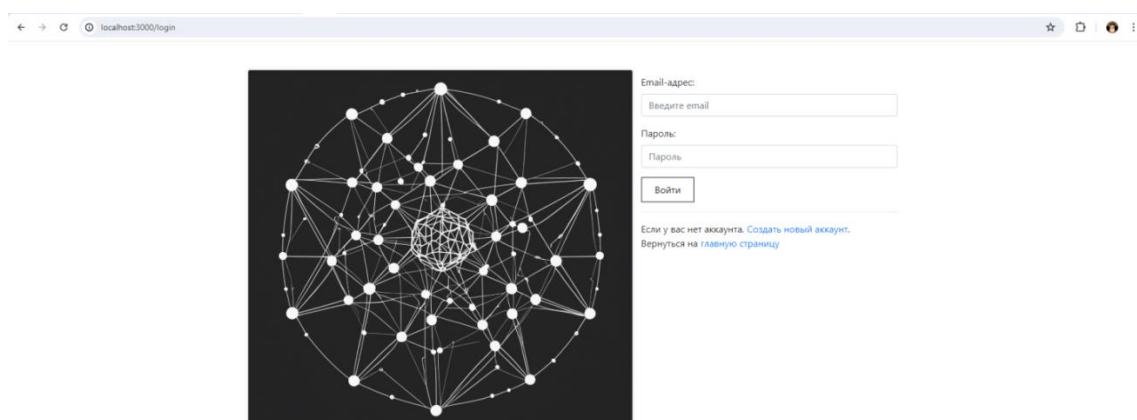


Рисунок 40 – Страница авторизации в веб-приложении

Пробуем ввести незарегистрированного пользователя для проверки работоспособности вывода сообщений об ошибках (рисунок 41).

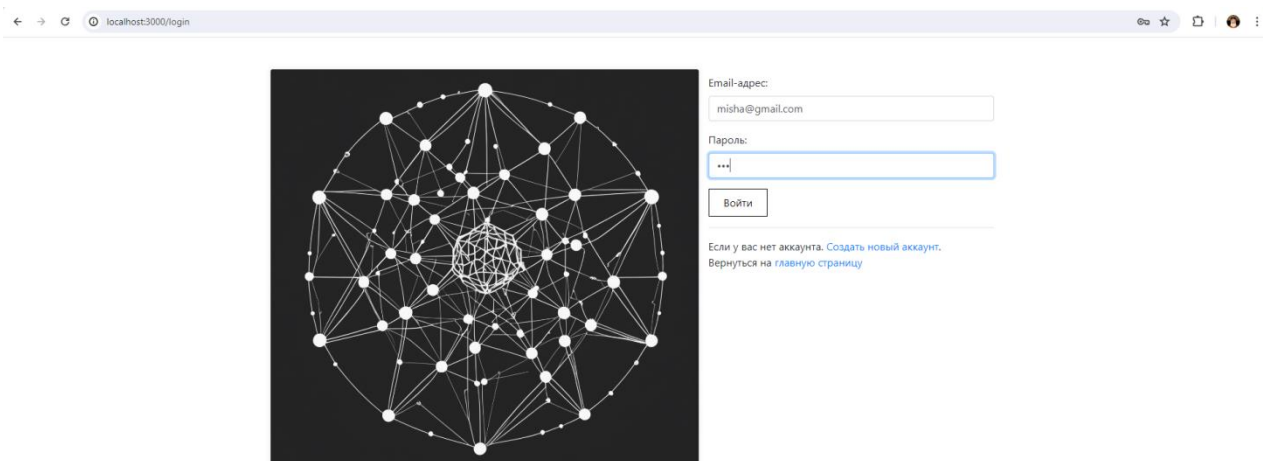


Рисунок 41 – Ввод некорректных данных

Можно увидеть, что появилось предупреждающее сообщение о том, что пользователя с такими данными не существует (рисунок 42).

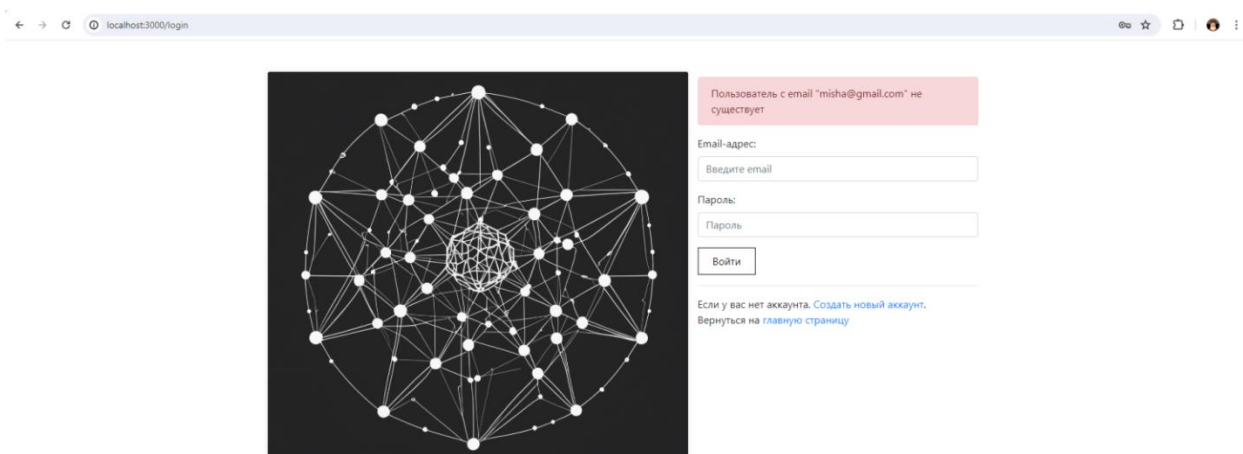


Рисунок 42 – Вывод сообщения о том, что такого пользователя не существует

Если пользователь хочет зарегистрироваться в веб-приложении, то ему надо перейти на страницу регистрации, нажав на ссылку «Создать новый аккаунт» снизу полей ввода персональных данных, где можно увидеть поля для ввода имени, email-адреса, пароля и повторения ранее введённого пароля (рисунок 43).

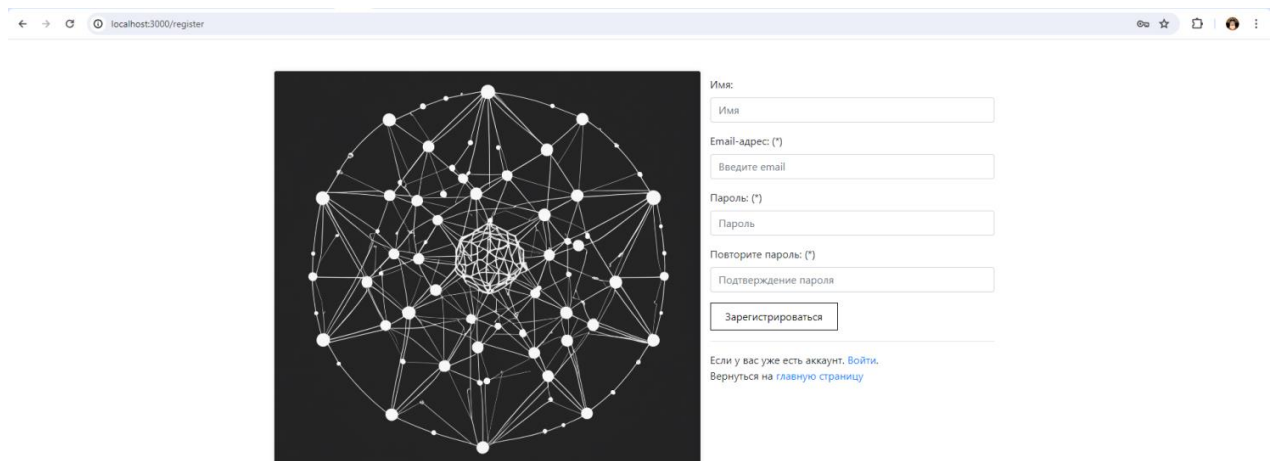


Рисунок 43 – Страница регистрации пользователя в веб-приложении

Пробуем ввести несовпадающие пароли в процессе заполнения полей регистрации для проверки работоспособности вывода сообщений об ошибках (рисунок 44).

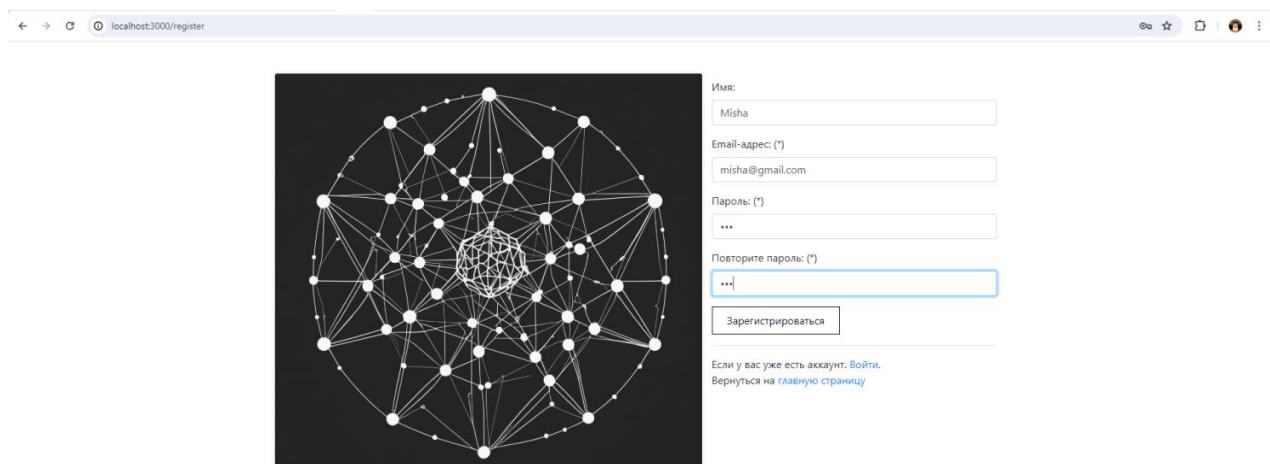


Рисунок 44 – Страница регистрации пользователя в веб-приложении

Можно увидеть, что появилось предупреждающее сообщение о том, что введенные в поля регистрации пароли не совпадают (рисунок 45).

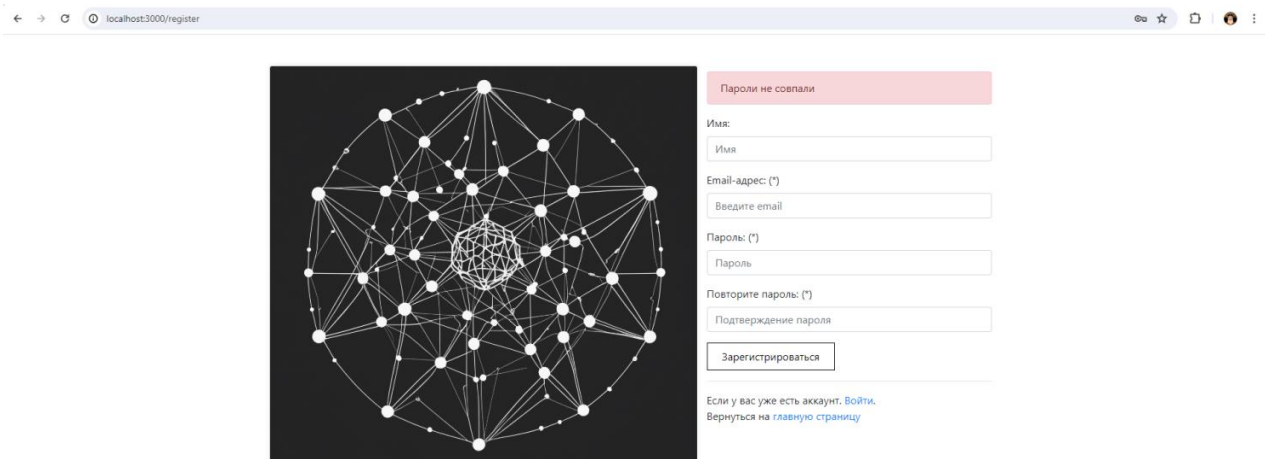


Рисунок 45 – Вывод сообщения о том, что введенные пароли не совпадают

Пробуем ввести email-адрес, уже использованный другим пользователем при регистрации, в процессе заполнения полей регистрации. Можно увидеть, что появилось предупреждающее сообщение о том, что пользователь с таким email-адресом уже существует (рисунок 46).

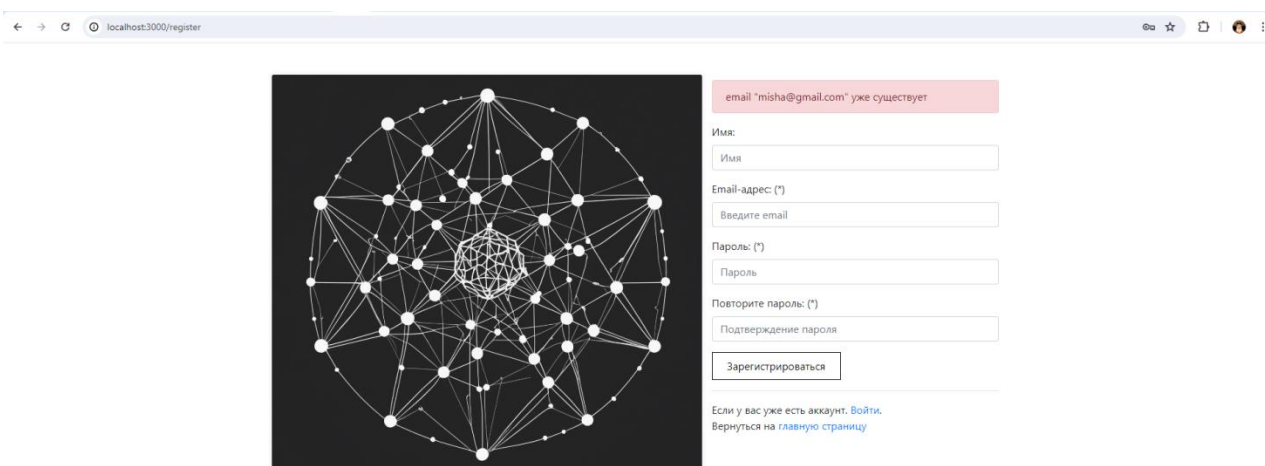


Рисунок 46 – Вывод сообщения о том, что такой email-адрес уже был использован ранее при регистрации в веб-приложении

При успешной регистрации пользователя перенаправляет обратно на страницу авторизации. Также в БД таблицу «users» записываются данные о новом зарегистрированном пользователе: id, имя, email и хешированный пароль (рисунок 47).

	id	fullname	email	password
▶	1	misha	misha956683.40@gmail.com	\$2a\$10\$dbYqiljD4Q4b9tPHI7FMOuqq.xwIm4Xh...
	3	misha v	mishaverxovskiy01@gmail.com	\$2a\$10\$GSlmCiQ2VeBtWLnBfaTb2u2ONIfsJiws...
	4	shrek	shrek@gmail.com	\$2a\$10\$Fytu0t5I.r17X7ch7Rzehu79J1xWxVf6...
	6	Kersh	Kersh@gmail.com	\$2a\$10\$mgOq7l5Y0uAiCTe7WlXIhegXIevZwlb7...
	7	Misha	misha@gmail.com	\$2a\$10\$NZxTPKsCgoII/o1JpgOBV.3.9U6Tvl1Q...
*	NULL	NULL	NULL	NULL

Рисунок 47 – Появление записи в таблице «users» о новом зарегистрированном пользователе

Пробуем ввести неверный пароль в процессе заполнения полей на странице авторизации. Можно увидеть, что появилось предупреждающее сообщение о том, что введён неверный пароль (рисунок 48).

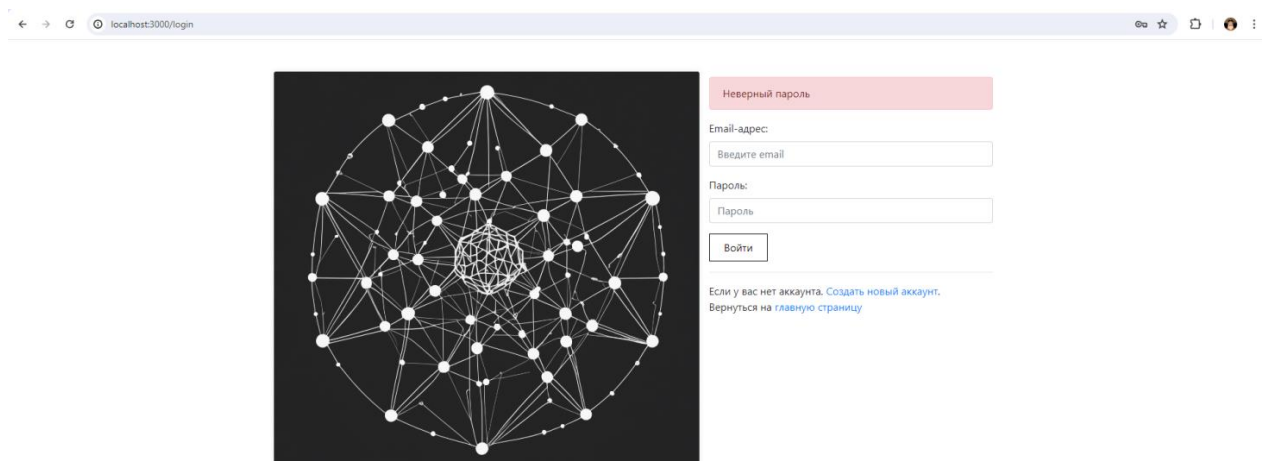


Рисунок 48 – Вывод сообщения о том, что введённый пароль неверный, на странице авторизации пользователя

Пробуем ввести правильные данные пользователя Misha в полях ввода пользователя Misha для авторизации в веб-приложении (рисунок 49).

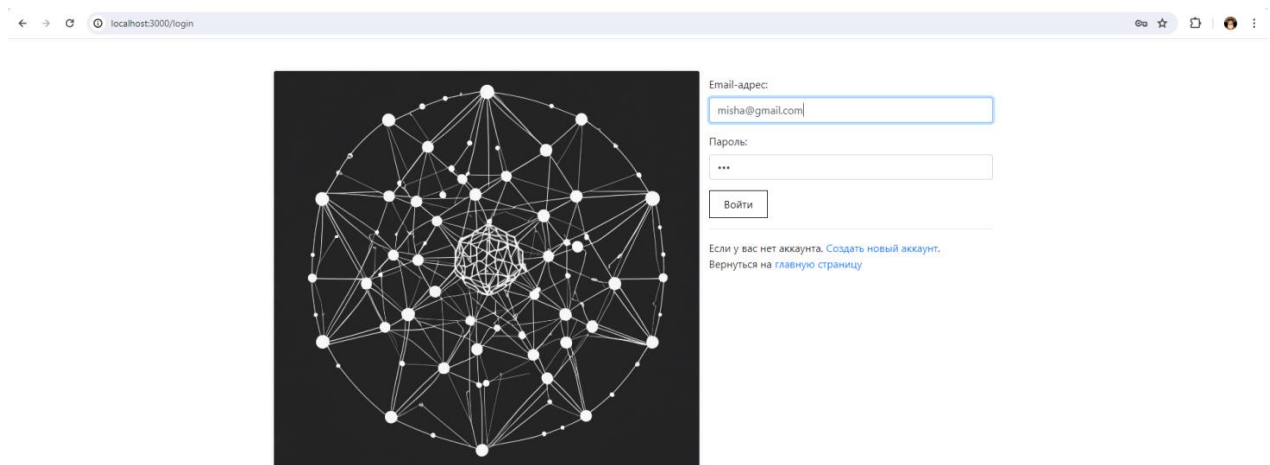


Рисунок 49 – Ввод правильных данных для авторизации пользователя Misha

При успешной авторизации происходит переход на главную страницу сайта, где сразу можно увидеть поле для работы с настройками и управлением, а также поля «До» и «После», где позже можно будет увидеть результат работы веб-сервиса по улучшению качества видео (рисунок 50).



Рисунок 51 – Главная страница сайта

При нажатии на кнопку «Выберите видео/изображение» в поле слева открывается окно выбора файла для обработки (рисунок 52).

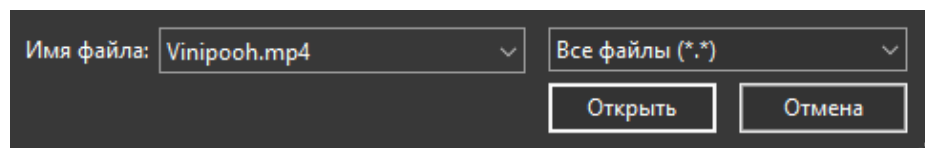


Рисунок 52 – Окно выбора файла для обработки

Для первого тестирования работы веб-приложения выбираем видеофайл. Выбранный видеофайл сразу же появился в поле «До» (рисунок 53).

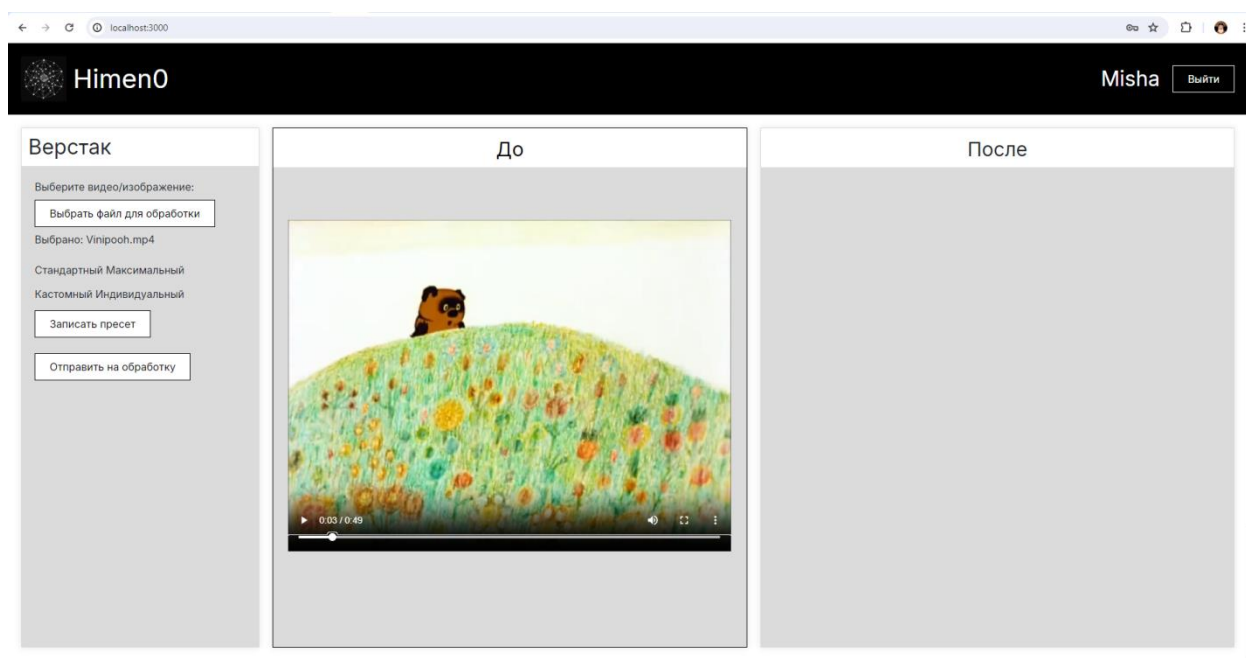


Рисунок 53 – Выбор видеофайла

Далее пользователь может выбрать из пресетов параметров обработки, доступных для видеофайлов (рисунки 54, 55).

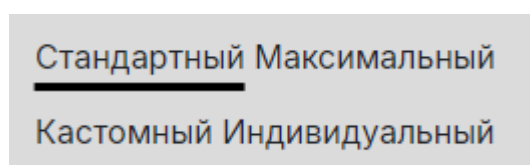


Рисунок 54 – Стандартный пресет

«Стандартный» пресет содержит следующие параметры:

```
{"Current_fps":0,"Upscale_fps":30,"Neural":"waifu2x","Horizontal":800,"Vertical":600}
```

Стандартный Максимальный
Кастомный Индивидуальный

Рисунок 55 – Максимальный пресет

«Максимальный» пресет содержит следующие параметры:

`{"Current_fps":0,"Upscale_fps":120,"Neural":"realesrgan","Horizontal":3840,"Vertical":2160}`

Также можно выбрать «Индивидуальный» пресет параметров, который можно постоянно перезаписывать (рисунок 56).

Стандартный Максимальный
Кастомный Индивидуальный

Записать пресет

Рисунок 56 – Индивидуальный пресет

Перезаписывается пресет «Индивидуальный» для каждого пользователя заново, в зависимости от того, изображение выбрано или видео, в БД таблице «presets» (рисунок 57).

	id	user_id	preset	preset_type
	1	7	<code>{"Current_fps":0,"Upscale_fps":60,"Neural":"realcugan","Horizontal":3840,"Vertical":2160}</code>	video
	NULL	NULL	NULL	NULL

Рисунок 57 – Запись в таблицу «presets» информации о новом пресете параметров

Также есть «Кастомный» пресет параметров, с помощью которого можно выбирать из предоставленных параметров для обработки видео (рисунок 58).

Стандартный Максимальный
Кастомный Индивидуальный

Рисунок 58 – Кастомный пресет

Выбор параметров обработки показан на рисунке 59.

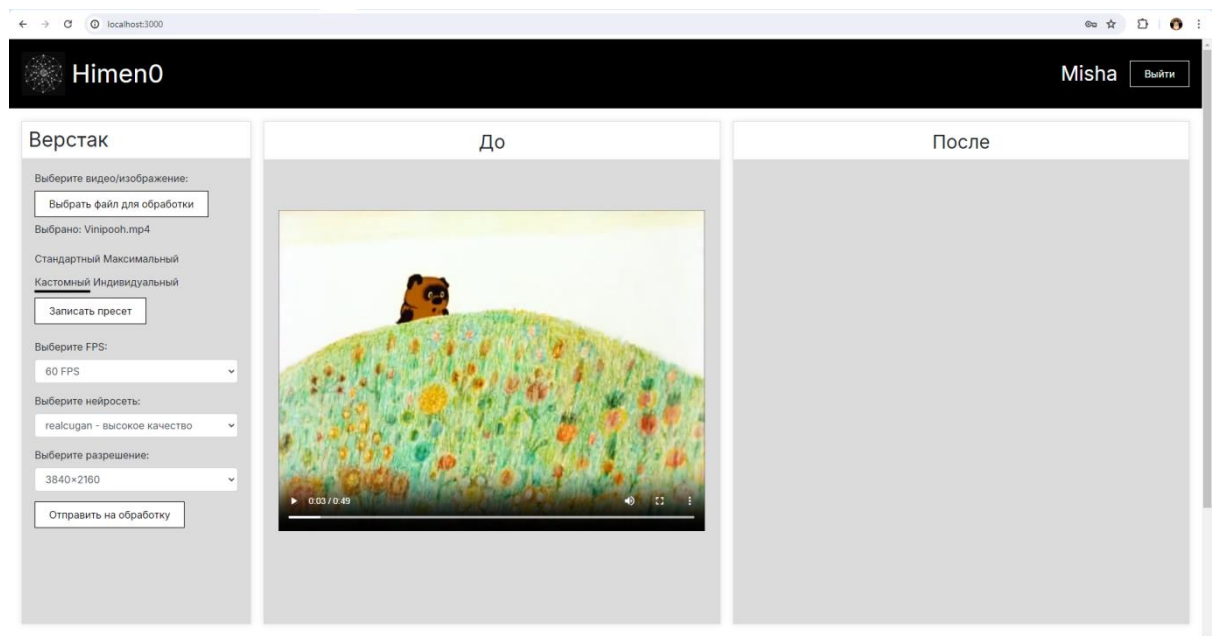


Рисунок 59 – Выбор параметров обработки

Выбираем FPS, нейросеть для обработки и разрешение (рисунки 60, 61, 62).

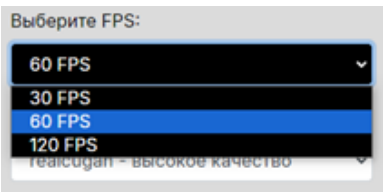


Рисунок 60 – Выбор FPS

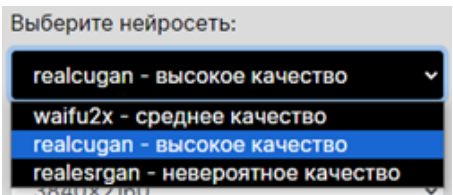


Рисунок 61 – Выбор нейросети

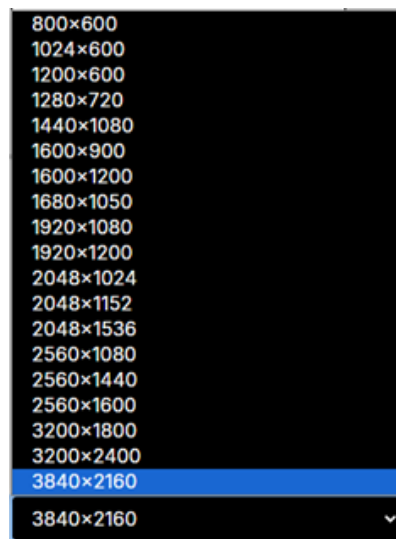


Рисунок 62 – Выбор разрешения

Далее нажимаем на кнопку «Отправить на обработку», чтобы отправить видео и JSON-файл с параметрами на сервер (рисунок 63). При успешной отправке файла на сервер появится надпись об успешной отправке.

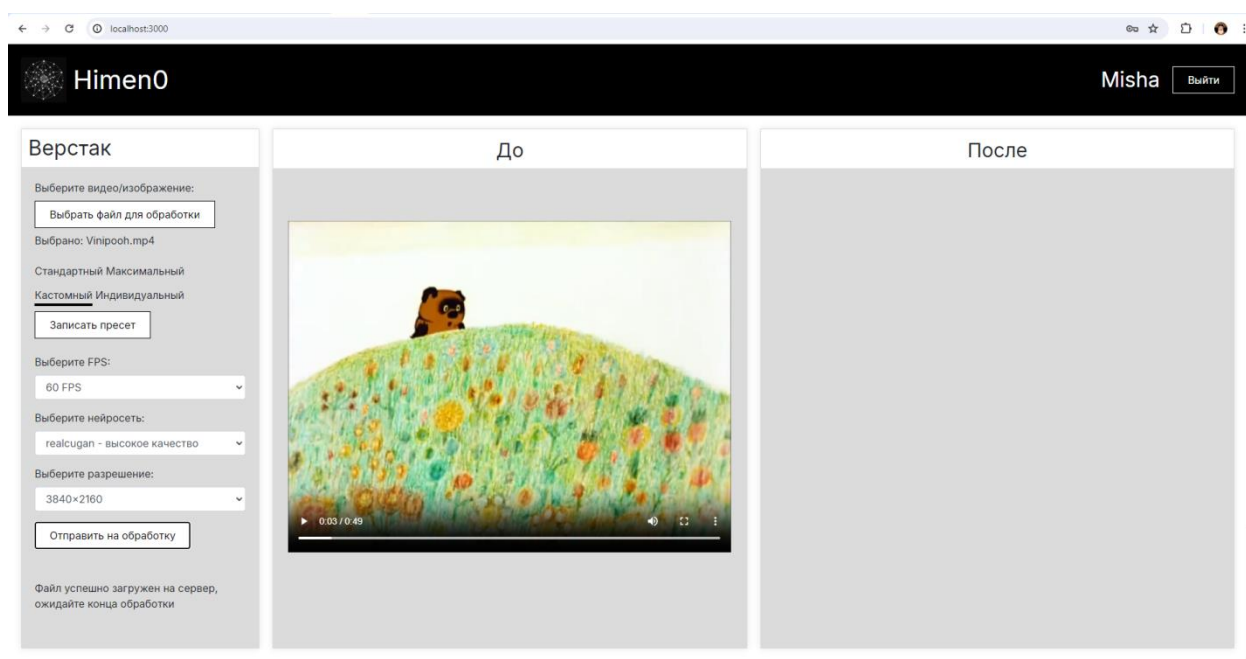


Рисунок 63 – Успешная отправка видеофайла на обработку

Также в БД таблицу «video_files_raw» записываются данные об отправленном видеофайле (рисунок 64).

id	user_id	title	file_path	file_type	description
131	4	logo.png	D:\papakIn\shrek\images\logo\logo.png	image	Neural: realesrgan, Resolution: 3840x2160
134	4	Test_Mouse (7).mp4	D:\papakIn\shrek\videos\Test_Mouse (7)\Test_Mouse (7).mp4	video	FPS: 30, Neural: waifu2x, Resolution: 800x600
135	4	Test_Mouse.mp4	D:\papakIn\shrek\videos\Test_Mouse\Test_Mouse.mp4	video	FPS: 120, Neural: realesrgan, Resolution: 3840x2160
138	4	Test_Mouse (2).mp4	D:\papakIn\shrek\videos\Test_Mouse (2)\Test_Mouse (2).mp4	video	FPS: 120, Neural: realcugan, Resolution: 2560x1440
140	4	Test_Mouse (14).mp4	D:\papakIn\shrek\videos\Test_Mouse (14)\Test_Mouse (14).mp4	video	FPS: 60, Neural: realcugan, Resolution: 2560x1600
141	4	logo (12).png	D:\papakIn\shrek\images\logo (12)\logo (12).png	image	Neural: realcugan, Resolution: 2560x1440
148	7	Vinipooh.mp4	D:\papakIn\Misha\videos\Vinipooh\Vinipooh.mp4	video	FPS: 60, Neural: realcugan, Resolution: 3840x2160

Рисунок 64 – Запись в таблицу «video_files_raw» информации о загруженном видео

При попытке отправки на обработку файла с неверным форматом, выведется предупреждение (рисунок 65).

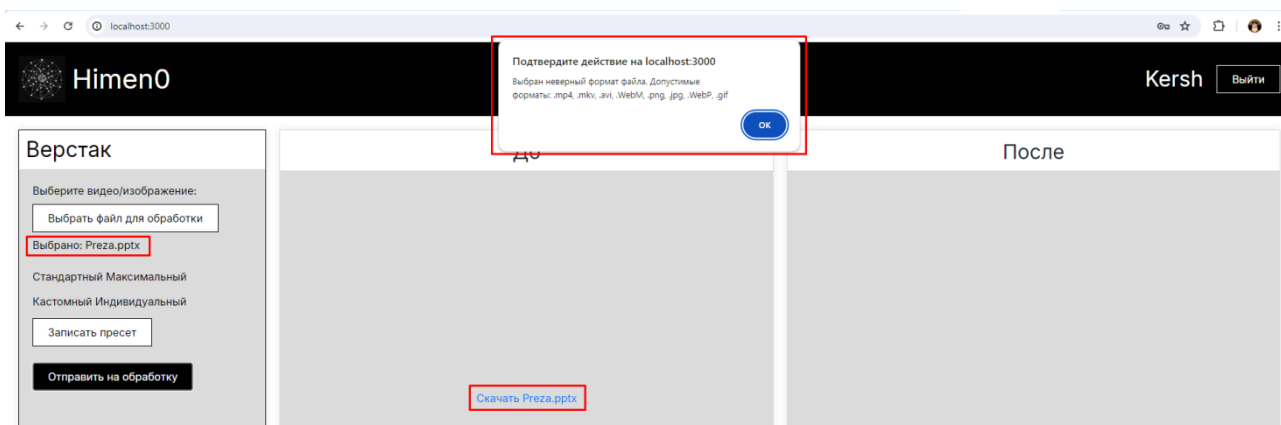


Рисунок 65 – Вывод предупреждения о том, что выбран неверный формат файла

Можно увидеть, что на сервере в папке papkaLn создана индивидуальная папка для пользователя Misha для принимаемых от клиента файлов (рисунок 66).

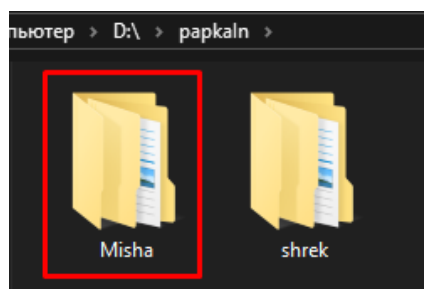


Рисунок 66 – Папка пользователя Misha для принимаемых файлов

Можно увидеть, что на сервере в папке papkaLn/Misha создана папка videos для принимаемых от клиента видеофайлов (рисунок 67).

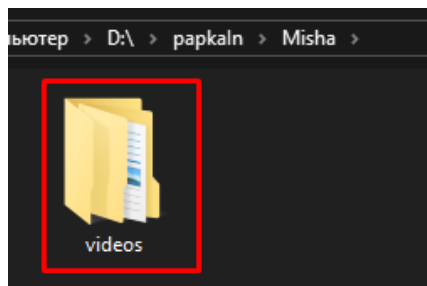


Рисунок 67 – Папка videos

Можно увидеть, что на сервере в папке parkaOut создается индивидуальная папка для пользователя Misha для отсылаемых клиенту файлов (рисунок 68).

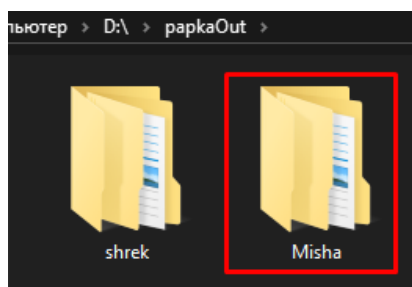


Рисунок 68 – Папка пользователя Misha для отсылаемых файлов

Можно увидеть, что на сервере в папке parkaln/Misha/videos создается папка Vinipooh для видеофайла, который пользователь Misha отправил на сервер для обработки (рисунок 69)

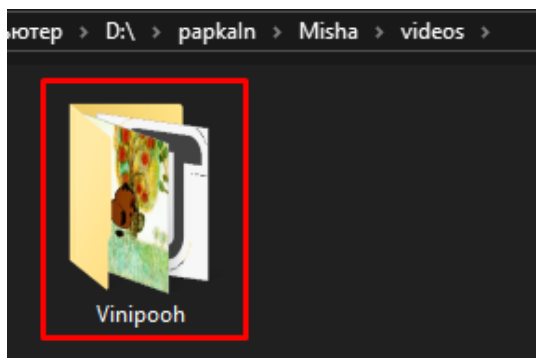


Рисунок 69 – Папка parkaln/Misha/videos/Vinipooh

Можно увидеть, что в папке Vinipooh для видеофайла, который пользователь Misha отправил на сервер для обработки, лежит видеофайл

Vinipooh.mp4 и JSON-файл Vinipooh-settings.json, который содержит параметры для обработки видеофайла (рисунок 70).

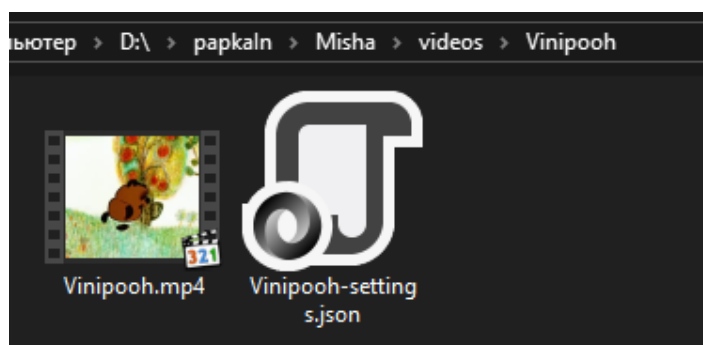


Рисунок 70 – Содержимое папки parkaLn/Misha/videos/Vinipooh

Содержимое JSON-файла Vinipooh-settings.json показано на рисунке 71.

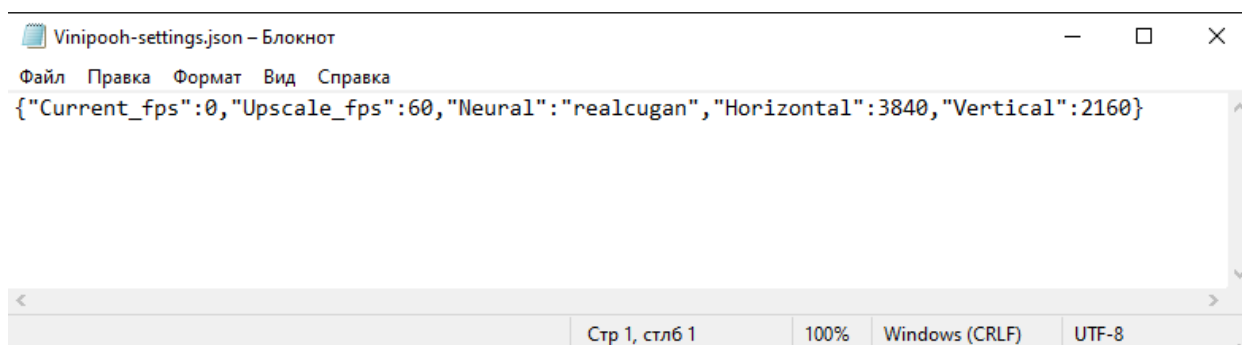


Рисунок 71 – Содержимое JSON-файла Vinipooh-settings.json

Можно увидеть, что на сервере в папке parkaOut/Misha создана папка videos для отсылаемых клиенту видеофайлов (рисунок 72).

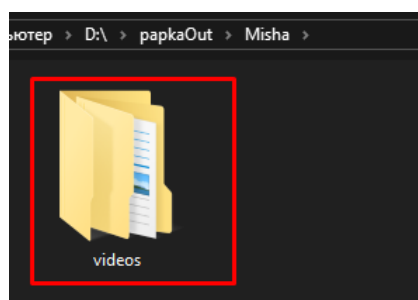


Рисунок 72 – Папка videos для отсылаемых файлов

Можно увидеть, что на сервере в папке parkaOut/Misha/videos создана папка Vinipooh для отсылаемых клиенту видеофайлов (рисунок 73).



Рисунок 73 – Папка папкаOut/Misha/videos/Vinipooh

Когда конвейер полностью обработает видеофайл Vinipooh.mp4, то в папке папкаOut/Misha/videos/Vinipooh появится файл makima.access, что будет означать о завершении обработки (рисунок 74).

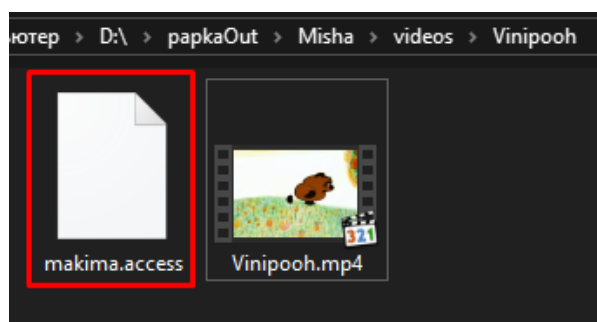


Рисунок 74 – Папка папкаOut/Misha/videos/Vinipooh. Появление файла makima.access

Можно увидеть, что на главной странице появилась кнопка «Скачать обработанный файл», а также надпись о том, что обработка завершена (рисунок 75).

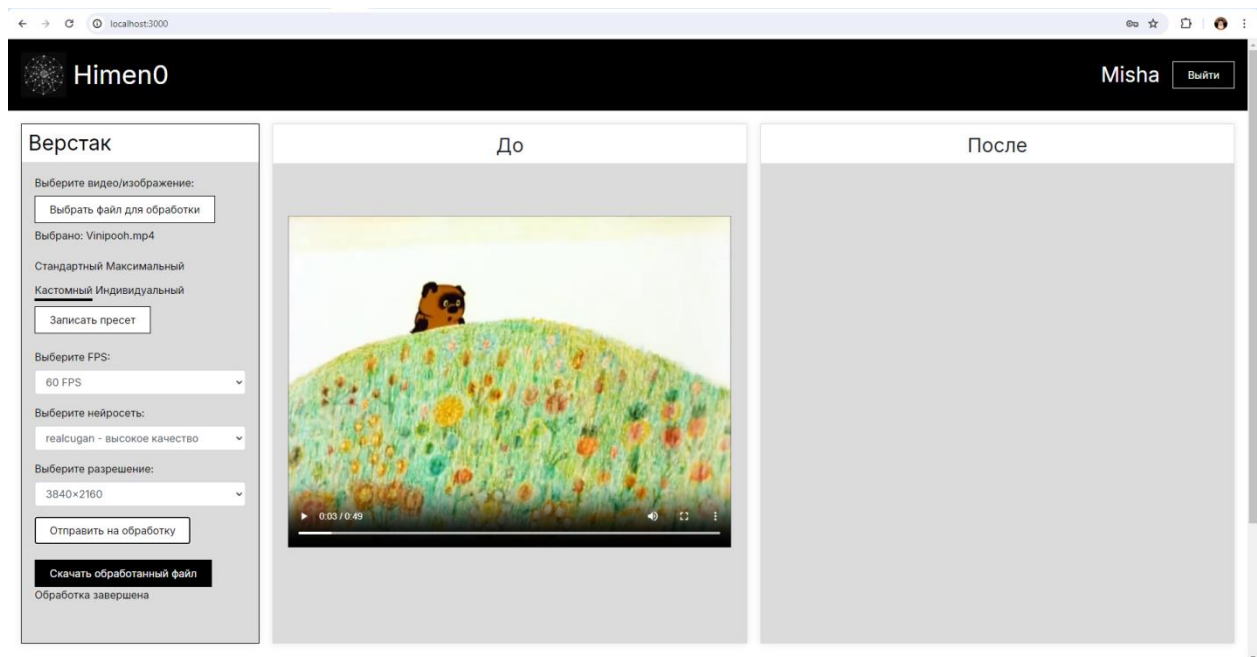


Рисунок 75 – Завершение обработки

При нажатии на кнопку «Скачать обработанный файл» обработанный файл скачивается на компьютер пользователя, а также обработанный файл появляется в поле «После», чтобы можно было сравнить два видеофайла (рисунок 76).

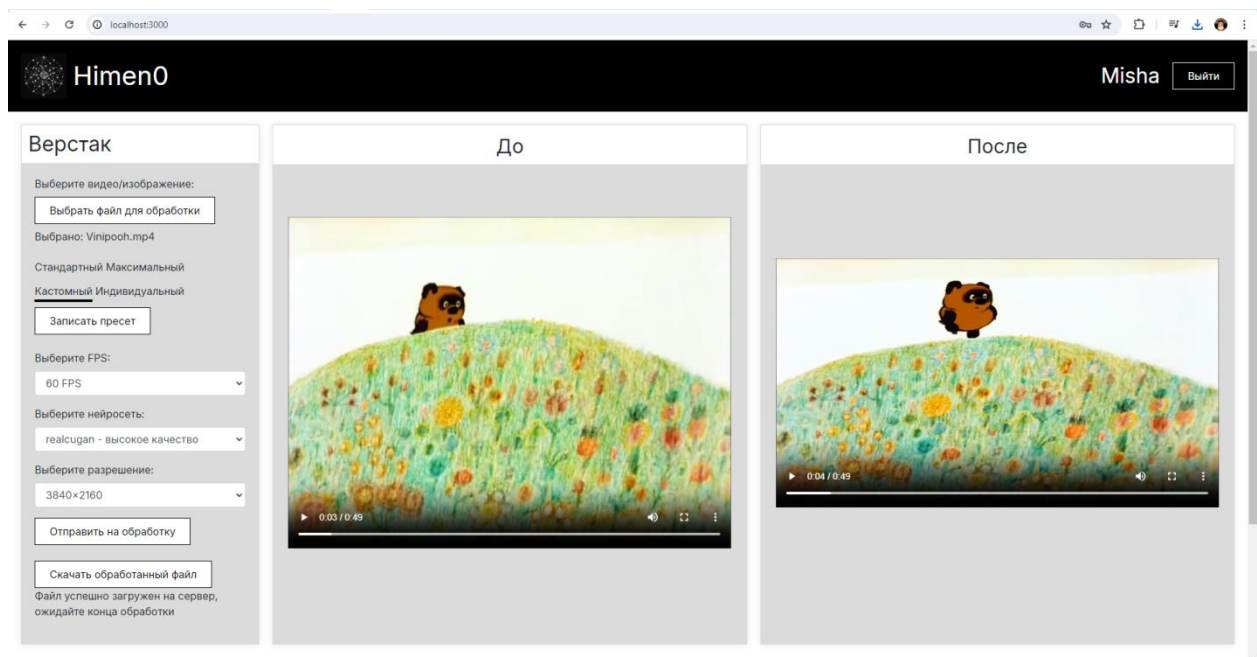


Рисунок 76 – Сравнение видеофайла «до» и «после» обработки

Также информация об обработанном видеофайле записалась в БД таблицу «video_files» (рисунок 77).

id	user_id	title	file_path	file_type	description
131	4	logo.png	D:\papkOut\shrek\images\logo\logo.png	image	Neural: realesrgan, Resolution: 3840x2160
134	4	Test_Mouse (7).mp4	D:\papkOut\shrek\videos\Test_Mouse (7)\Test_Mouse (7).mp4	video	FPS: 30, Neural: waifu2x, Resolution: 800x600
135	4	Test_Mouse.mp4	D:\papkOut\shrek\videos\Test_Mouse\Test_Mouse.mp4	video	FPS: 120, Neural: realesrgan, Resolution: 3840x2160
138	4	Test_Mouse (2).mp4	D:\papkOut\shrek\videos\Test_Mouse (2)\Test_Mouse (2).mp4	video	FPS: 120, Neural: realcugan, Resolution: 2560x1440
140	4	Test_Mouse (14)....	D:\papkOut\shrek\videos\Test_Mouse (14)\Test_Mouse (14).mp4	video	FPS: 60, Neural: realcugan, Resolution: 2560x1600
141	4	logo (12).png	D:\papkOut\shrek\images\logo (12)\logo (12).png	image	Neural: realcugan, Resolution: 2560x1440
148	7	Vinipooh.mp4	D:\papkOut\Misha\videos\Vinipooh\Vinipooh.mp4	video	FPS: 60, Neural: realcugan, Resolution: 3840x2160

Рисунок 77 – Запись в таблицу «video_files» информации об обработанном видео

Также на главной странице есть галерея из ранее обработанных изображений и видеофайлов пользователем. Там можно увидеть обработанный файл Vinipooh.mp4 (рисунок 78).

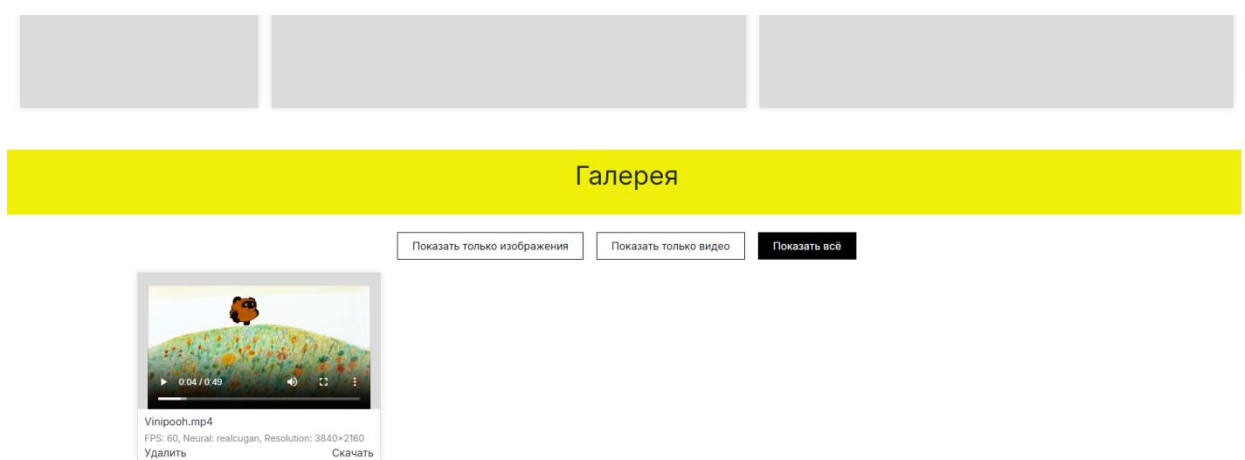


Рисунок 78 – Галерея обработанных изображений и видео

Файлы из галереи можно скачивать при нажатии на кнопку «Скачать» (рисунки 79, 80).

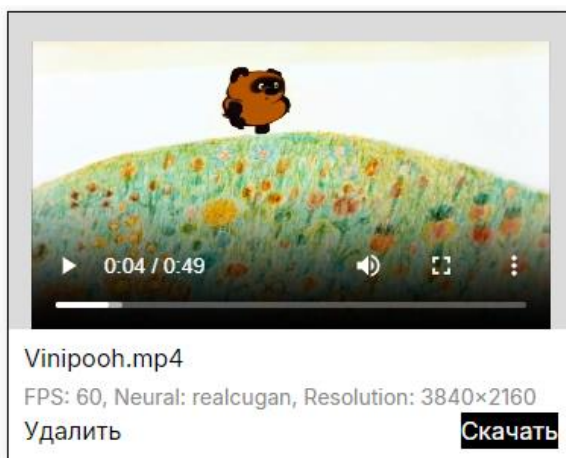


Рисунок 79 – Кнопка «Скачать»

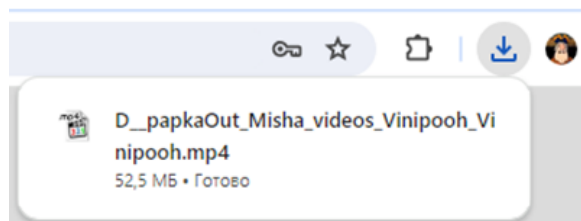


Рисунок 80 – Файл загружен

Файлы из галереи можно удалять при нажатии на кнопку «Удалить» (рисунки 81, 82).

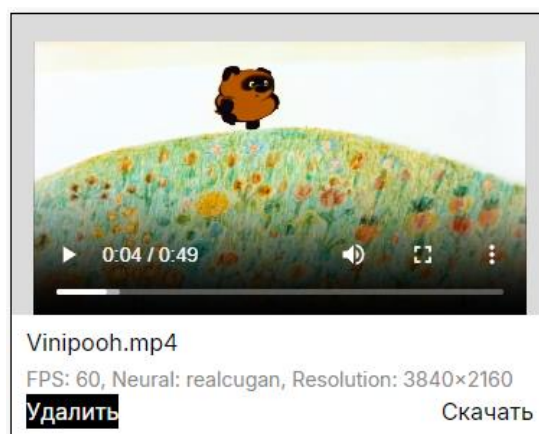


Рисунок 81 – Кнопка «Удалить»

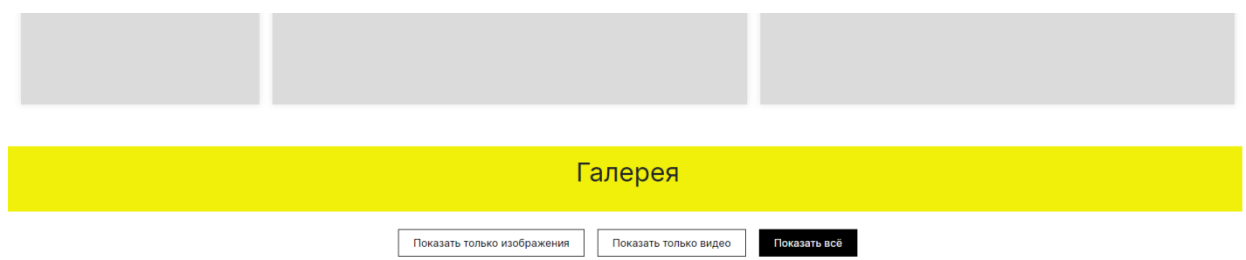


Рисунок 82 – Файл удалён из галереи

Для второго тестирования работы веб-приложения выбираем изображение. При нажатии на кнопку «Выберите видео/изображение» в поле слева открывается окно выбора файла для обработки (рисунок 83).

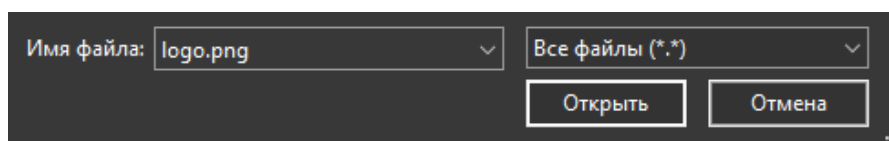


Рисунок 83 – Окно выбора файла для обработки

Выбранное изображение сразу же появилось в поле «До» (рисунок 84).

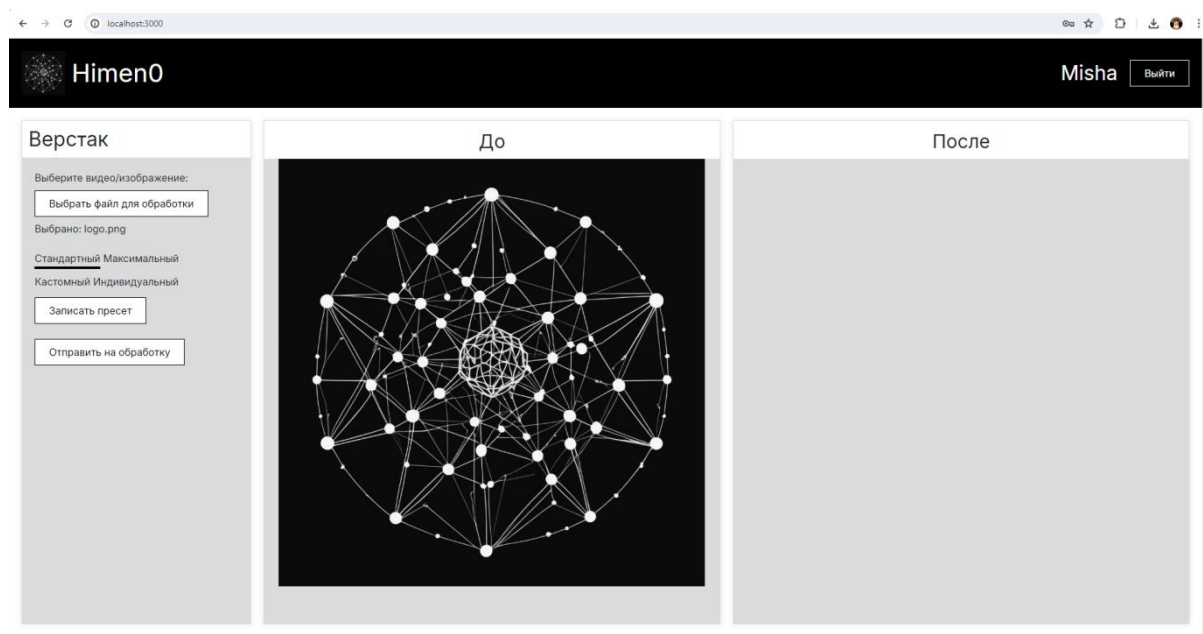


Рисунок 84 – Выбор изображения

Для изображений выбор пресетов настроек такой же, как и у видеофайлов, только во всех них отсутствует FPS.

Выбираем «Кастомный» пресет настроек.

Выбираем нейросеть для обработки и разрешение (рисунок 85, 86).

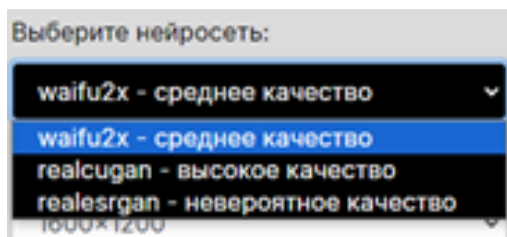


Рисунок 85 – Выбор нейросети

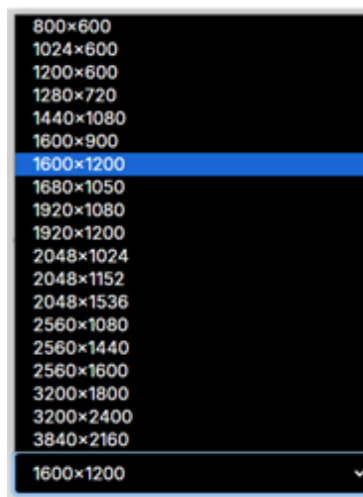


Рисунок 86 – Выбор разрешения

Далее нажимаем на кнопку «Отправить на обработку», чтобы отправить изображение и JSON-файл с параметрами на сервер (рисунок 87). При успешной отправке файла на сервер появится надпись об успешной отправке.

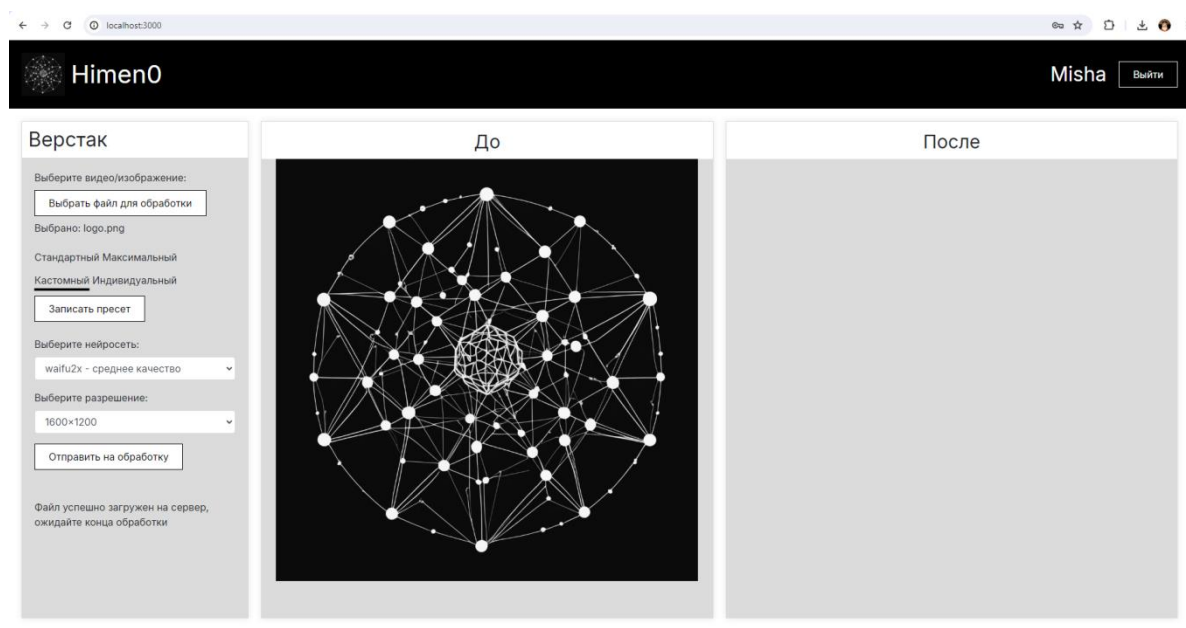


Рисунок 87 – Успешная отправка изображения на обработку

Также в БД таблицу «video_files_raw» записываются данные об отправленном изображении (рисунок 88).

	id	user_id	title	file_path	file_type	description
	131	4	logo.png	D:\parkaIn\shrek\images\logo\logo.png	image	Neural: realesrgan, Resolution: 3840x2160
	134	4	Test_Mouse (7).mp4	D:\parkaIn\shrek\videos\Test_Mouse (7)\Test_Mouse (7).mp4	video	FPS: 30, Neural: waifu2x, Resolution: 800x600
	135	4	Test_Mouse.mp4	D:\parkaIn\shrek\videos\Test_Mouse\Test_Mouse.mp4	video	FPS: 120, Neural: realesrgan, Resolution: 3840x2160
	138	4	Test_Mouse (2).mp4	D:\parkaIn\shrek\videos\Test_Mouse (2)\Test_Mouse (2).mp4	video	FPS: 120, Neural: realcugan, Resolution: 2560x1440
	140	4	Test_Mouse (14).mp4	D:\parkaIn\shrek\videos\Test_Mouse (14)\Test_Mouse (14).mp4	video	FPS: 60, Neural: realcugan, Resolution: 2560x1600
	141	4	logo (12).png	D:\parkaIn\shrek\images\logo (12)\logo (12).png	image	Neural: realcugan, Resolution: 2560x1440
	148	7	Vinipooh.mp4	D:\parkaIn\Misha\videos\Vinipooh\Vinipooh.mp4	video	FPS: 60, Neural: realcugan, Resolution: 3840x2160
▶	149	7	logo.png	D:\parkaIn\Misha\images\logo\logo.png	image	Neural: waifu2x, Resolution: 1600x1200

Рисунок 88 – Запись в таблицу «video_files_raw» информации о загруженном изображении

Можно увидеть, что на сервере в папке parkaLn/Misha создалась папка images для принимаемых от клиента изображений (рисунок 89).

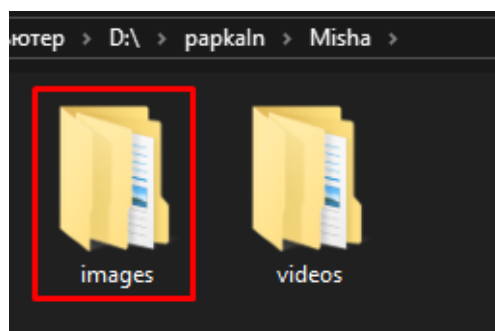


Рисунок 89 – Папка images

Можно увидеть, что на сервере в папке parkaLn/Misha/videos создалась папка logo для изображения, которое пользователь Misha отправил на сервер для обработки (рисунок 90).

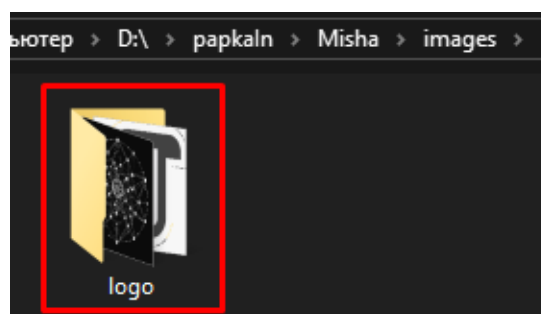


Рисунок 90 – Папка parkaLn/Misha/images/logo

Можно увидеть, что в папке logo для видеофайла, который пользователь Misha отправил на сервер для обработки, лежит изображение logo.png и JSON-

файл logo-settings.json, который содержит параметры для обработки изображения (рисунок 91).



Рисунок 91 – Папка logo

Содержимое JSON-файла logo-settings.json показано на рисунке 92.

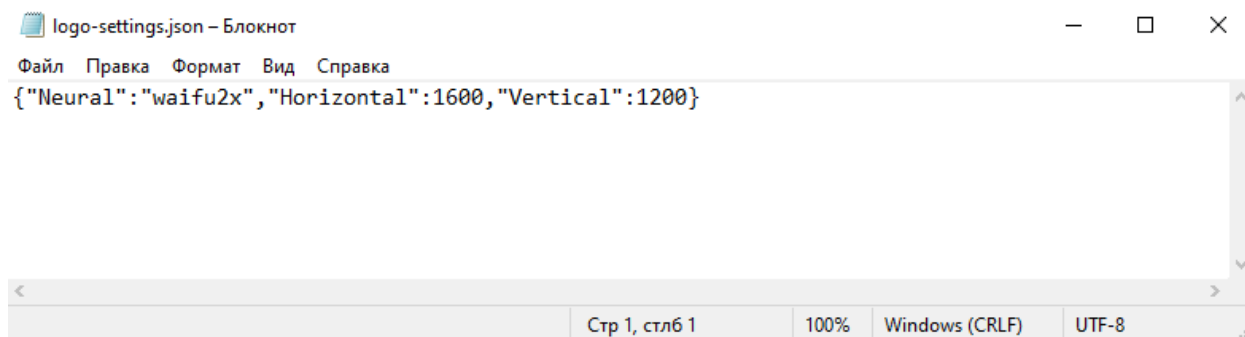


Рисунок 92 – Содержимое JSON-файла logo-settings.json

Можно увидеть, что на сервере в папке parkaOut/Misha создавалась папка images для пользователя Misha для отсылаемых клиенту изображений (рисунок 93).

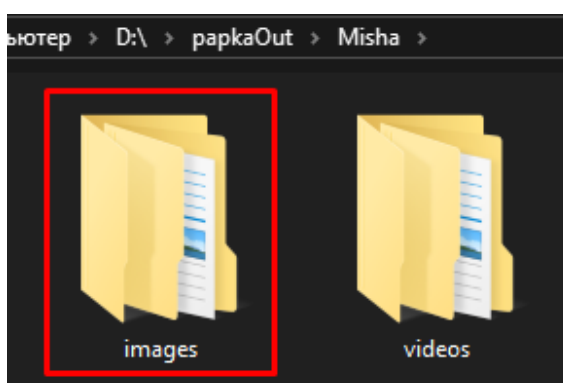


Рисунок 93 – Папка images для отсылаемых файлов

Можно увидеть, что на сервере в папке `parkaOut/Misha/images` создалась папка `logo` для отсылаемых клиенту изображений (рисунок 94).



Рисунок 94 – Папка `parkaOut/Misha/images/logo`

Когда конвейер полностью обработает изображение `logo.png`, то в папке `parkaOut/Misha/videos/logo` появится файл `makima.access`, что будет означать о завершении обработки (рисунок 95).

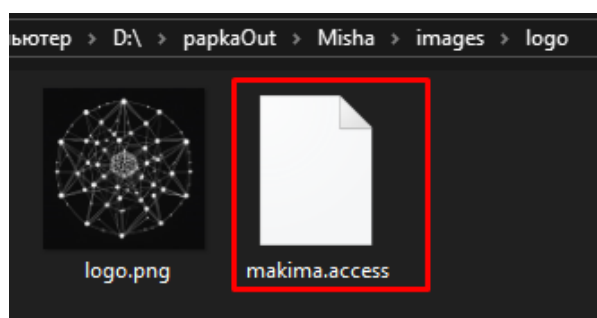


Рисунок 95 – Папка `parkaOut/Misha/videos/logo`. Появление файла `makima.access`

Можно увидеть, что на главной странице появилась кнопка «Скачать обработанный файл», а также надпись о том, что обработка завершена (рисунок 96).

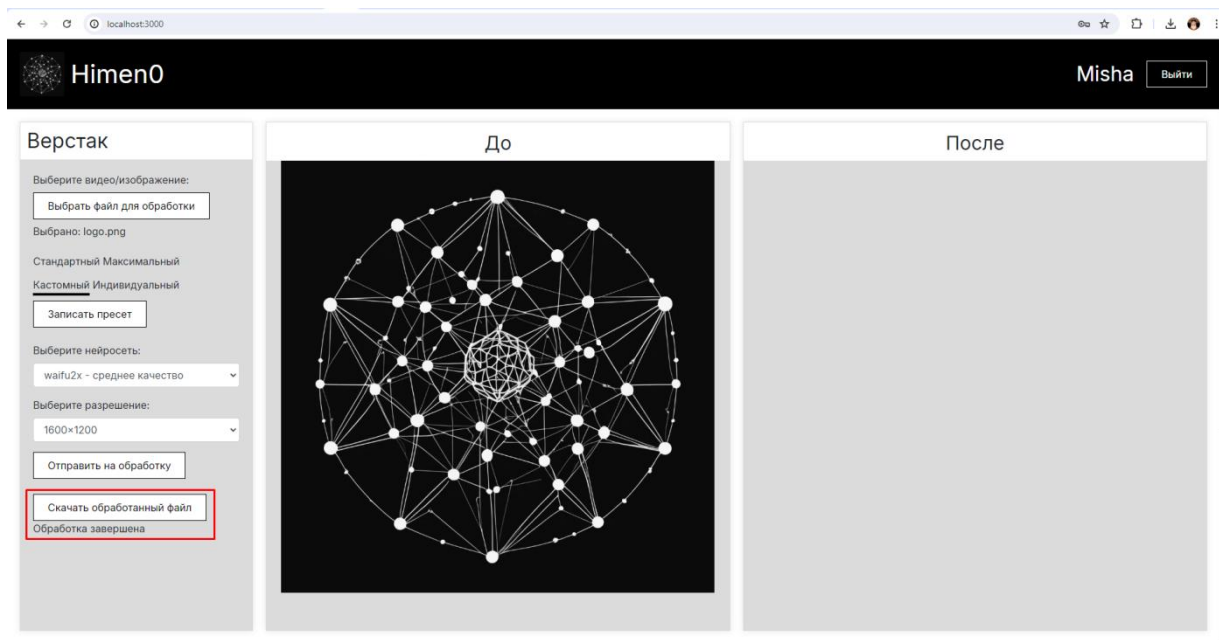


Рисунок 96 – Завершение обработки

При нажатии на кнопку «Скачать обработанный файл» обработанный файл скачивается на компьютер пользователя, а также обработанный файл появляется в поле «После», чтобы можно было сравнить два изображения (рисунок 97).

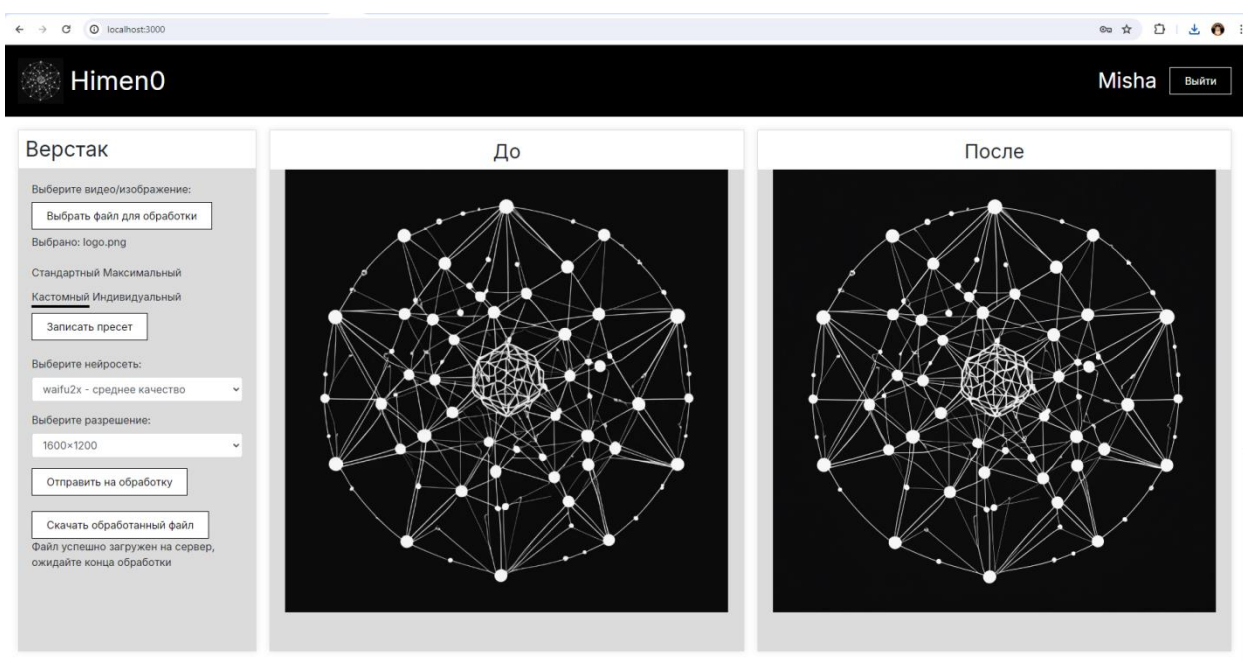


Рисунок 97 – Сравнение изображения «до» и «после» обработки

Также информация об обработанном изображении записалась в БД таблицу video_files (рисунок 98).

id	user_id	title	file_path	file_type	description
131	4	logo.png	D:\pafkaOut\shrek\images\logo\logo.png	image	Neural: realesrgan, Resolution: 3840x2160
134	4	Test_Mouse (7).mp4	D:\pafkaOut\shrek\videos\Test_Mouse (7)\Test_Mouse (7).mp4	video	FPS: 30, Neural: waifu2x, Resolution: 800x600
135	4	Test_Mouse.mp4	D:\pafkaOut\shrek\videos\Test_Mouse\Test_Mouse.mp4	video	FPS: 120, Neural: realesrgan, Resolution: 3840x2160
138	4	Test_Mouse (2).mp4	D:\pafkaOut\shrek\videos\Test_Mouse (2)\Test_Mouse (2).mp4	video	FPS: 120, Neural: realcugan, Resolution: 2560x1440
140	4	Test_Mouse (14)....	D:\pafkaOut\shrek\videos\Test_Mouse (14)\Test_Mouse (14).mp4	video	FPS: 60, Neural: realcugan, Resolution: 2560x1600
141	4	logo (12).png	D:\pafkaOut\shrek\images\logo (12)\logo (12).png	image	Neural: realcugan, Resolution: 2560x1440
148	7	Vinipooh.mp4	D:\pafkaOut\Misha\videos\Vinipooh\Vinipooh.mp4	video	FPS: 60, Neural: realcugan, Resolution: 3840x2160
149	7	logo.png	D:\pafkaOut\Misha\images\logo\logo.png	image	Neural: waifu2x, Resolution: 1600x1200

Рисунок 98 – Запись в таблицу «video_files» информации об обработанном видео

Также на главной странице есть галерея из ранее обработанных изображений и видеофайлов пользователем. Там можно увидеть обработанный файл logo.png (рисунок 99).



Рисунок 99 – Галерея обработанных изображений и видео

Файлы из галереи можно скачивать при нажатии на кнопку «Скачать» (рисунки 100, 101).

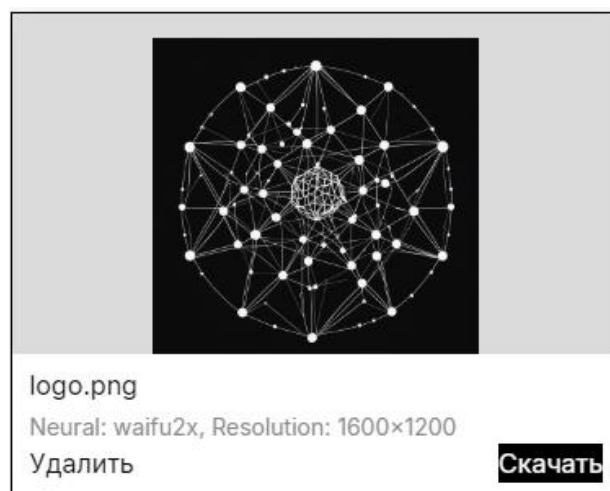


Рисунок 10 – Кнопка «Скачать»

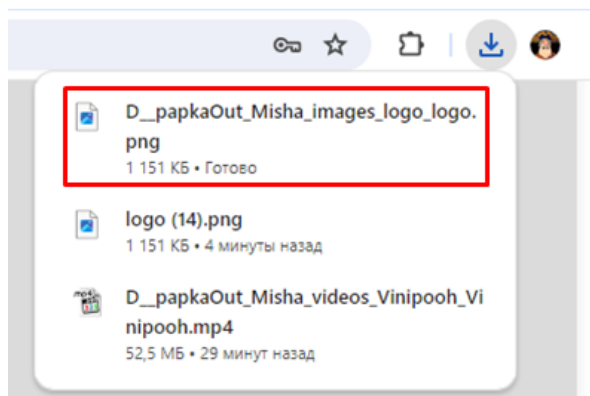


Рисунок 101 – Файл загружен

Файлы из галереи можно удалять при нажатии на кнопку «Удалить» (рисунок 102).

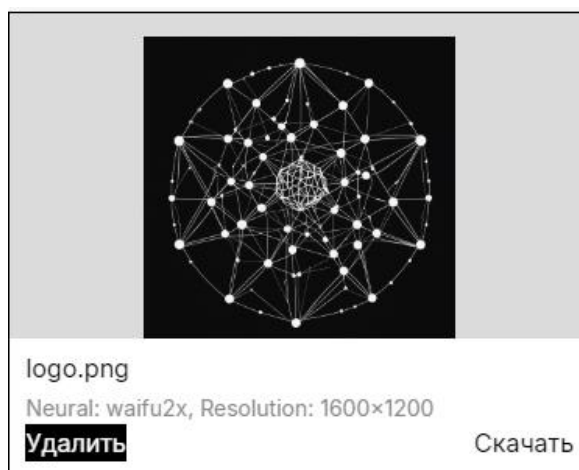


Рисунок 102 – Кнопка «Удалить»

Также при выборе конкретного фильтра можно менять вид галереи.

При выборе фильтра «Показать только видео» в галерее отображаются только видеофайлы (рисунок 103).

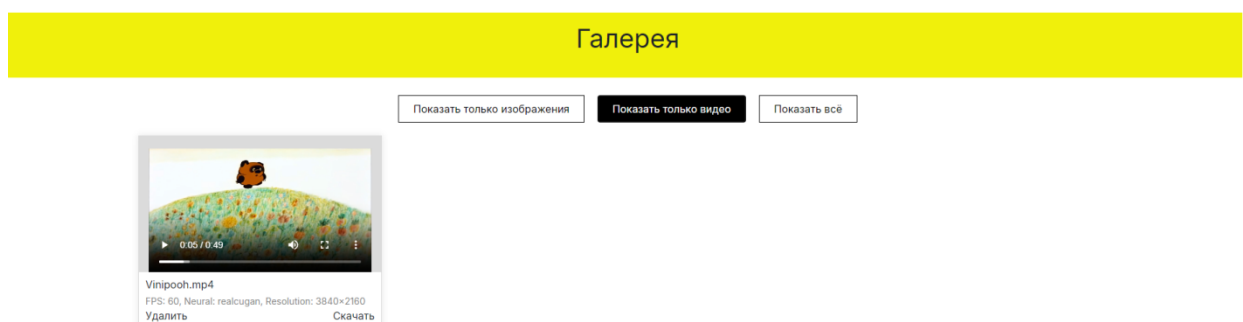


Рисунок 103 – Фильтр «Показать только видео»

При выборе фильтра «Показать только изображения» в галерее отображаются только изображения (рисунок 104).

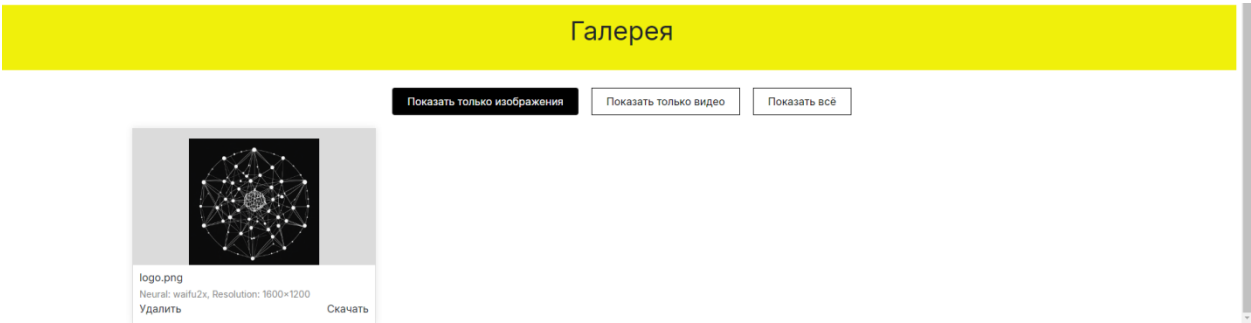


Рисунок 104 – Фильтр «Показать только изображения»

Тестирование веб-приложения успешно проведено.

2 Охрана труда

2.1 Анализ негативных факторов, воздействующих на оператора ПЭВМ и способы их устранения

Взаимодействие с компьютерной техникой происходит в условиях минимальной подвижности 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

Рабочее место должно удовлетворять следующим основным требованиям:

- 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 105 изображена подходящая конфигурация рабочего места, которая отвечает указанным выше требованиям.

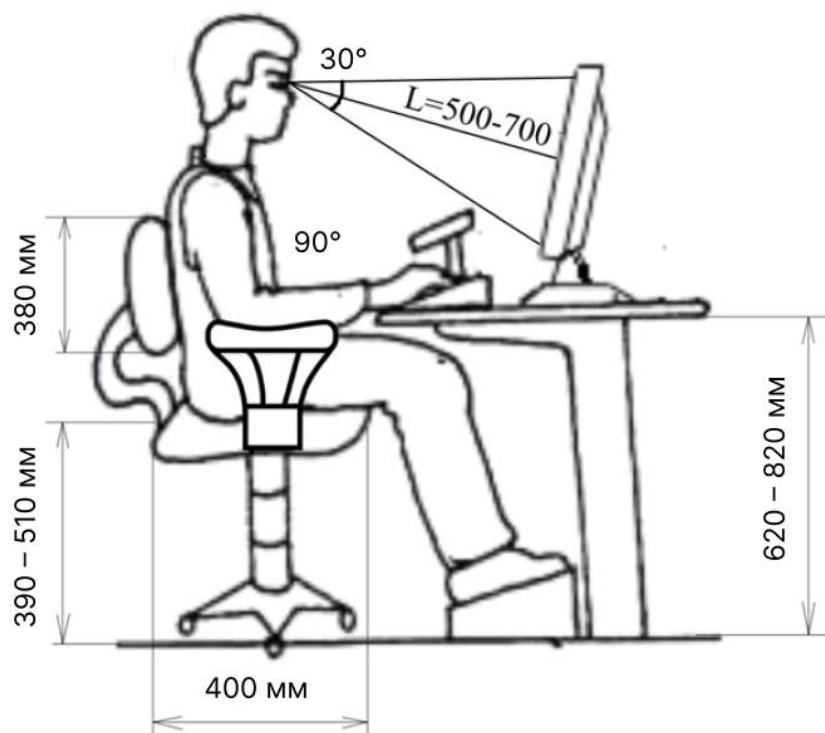


Рисунок 105 – Требуемая организация рабочего места

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 106.

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).

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.04.001 ПЗ

Лист
136

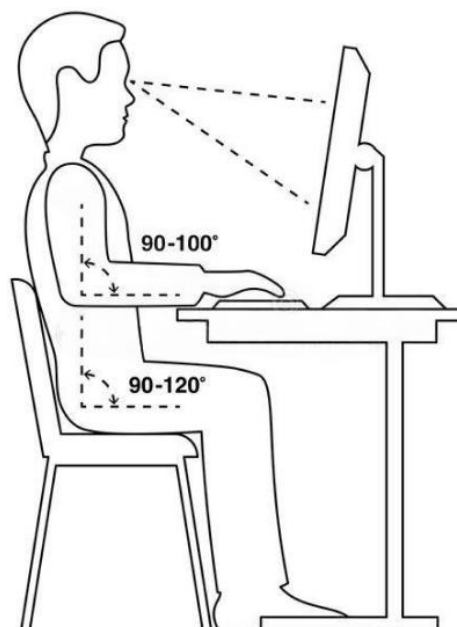


Рисунок 106 – Пример правильной посадки за персональный компьютер

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.

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 определяются следующим образом:

– для группы А – по суммарному числу считываемых знаков за рабочую смену (не более 60000 знаков за смену);

– для группы Б – по суммарному числу считываемых или вводимых знаков за рабочую смену (не более 40000 знаков за смену);

– для группы В – по суммарному времени непосредственной работы с компьютером за рабочую смену (не более 6 часов за смену).

Суммарное время перерывов для каждой категории и видов работы указаны 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, т.к. 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:

– 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.

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 госпошину 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 было предложено бухгалтерам и менеджерам с бирж труда «hh.ru», а также «Profi.ru».

Этапы разработки заготовки ПО описаны в пункте 1.

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.

					ПГУ1.090301.04.001 ПЗ	Лист
						148
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 данное пространство, 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 по 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:

					ПГУ1.090301.04.001 ПЗ	Лист
						150
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.

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 таблице 8, 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.

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

В ходе выполнения дипломной работы были получены обширные практические навыки в разработке современных веб-приложений. Были освоены такие технологии, как JavaScript, HTML, CSS, Node.js, Express.js, MySQL, а также принципы клиент-серверного программирования и паттерн проектирования MVC.

В результате проведенного исследования предметной области и анализа существующих решений было разработано веб-приложение для улучшения качества видео и изображений. Приложение позволяет пользователям загружать файлы, выбирать различные параметры обработки (разрешение, нейронная сеть, FPS), а затем скачивать обработанные файлы.

Были спроектированы и реализованы основные страницы веб-приложения, включая страницы регистрации, входа, главную страницу с загрузкой и обработкой файлов, а также галерею ранее обработанных изображений и видео. Была спроектирована и разработана база данных для хранения информации.

Получившееся веб-приложение было протестировано в локальной сети с использованием VPN Hamachi, что подтвердило его работоспособность и готовность к дальнейшему развитию и масштабированию.

Выполнение дипломной работы позволило значительно улучшить навыки разработки клиент-серверных веб-приложений, работы с базами данных, проектирования программных систем и взаимодействия с пользовательским интерфейсом. Полученные знания и опыт будут полезны в дальнейшей профессиональной деятельности.

Список использованных источников

1. Дакетт Дж. HTML и CSS. Разработка и дизайн веб-сайтов, Москва: Эксмо, 2023. – 480 с.
2. Официальная документация JavaScript на MDN Web Docs. [Электронный ресурс] / Режим доступа: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web, свободный (дата обращения 25.03.2024).
3. Сойер Д. HTML и CSS. Разработка и дизайн веб-сайтов. Москва : Эксмо, 2018. – 384 с.
4. Официальная документация Node.js. [Электронный ресурс] / Режим доступа: <https://nodejs.org/en/docs/>, свободный (дата обращения 27.02.2024).
5. Официальная документация Express.js. [Электронный ресурс] / Режим доступа: <https://expressjs.com/en/starter/installing.html>, свободный (дата обращения 15.03.2024).
6. Грофф Дж. Р., Вайнберг П.Н., Оппель Э. Дж. SQL: полное руководство. – 2-е изд. Москва : Вильямс, 2015. – 960 с.
7. Официальный сайт LogMeIn Hamachi. [Электронный ресурс] / Режим доступа: <https://www.vpn.net/>, свободный (дата обращения 20.04.2024).

Приложение А

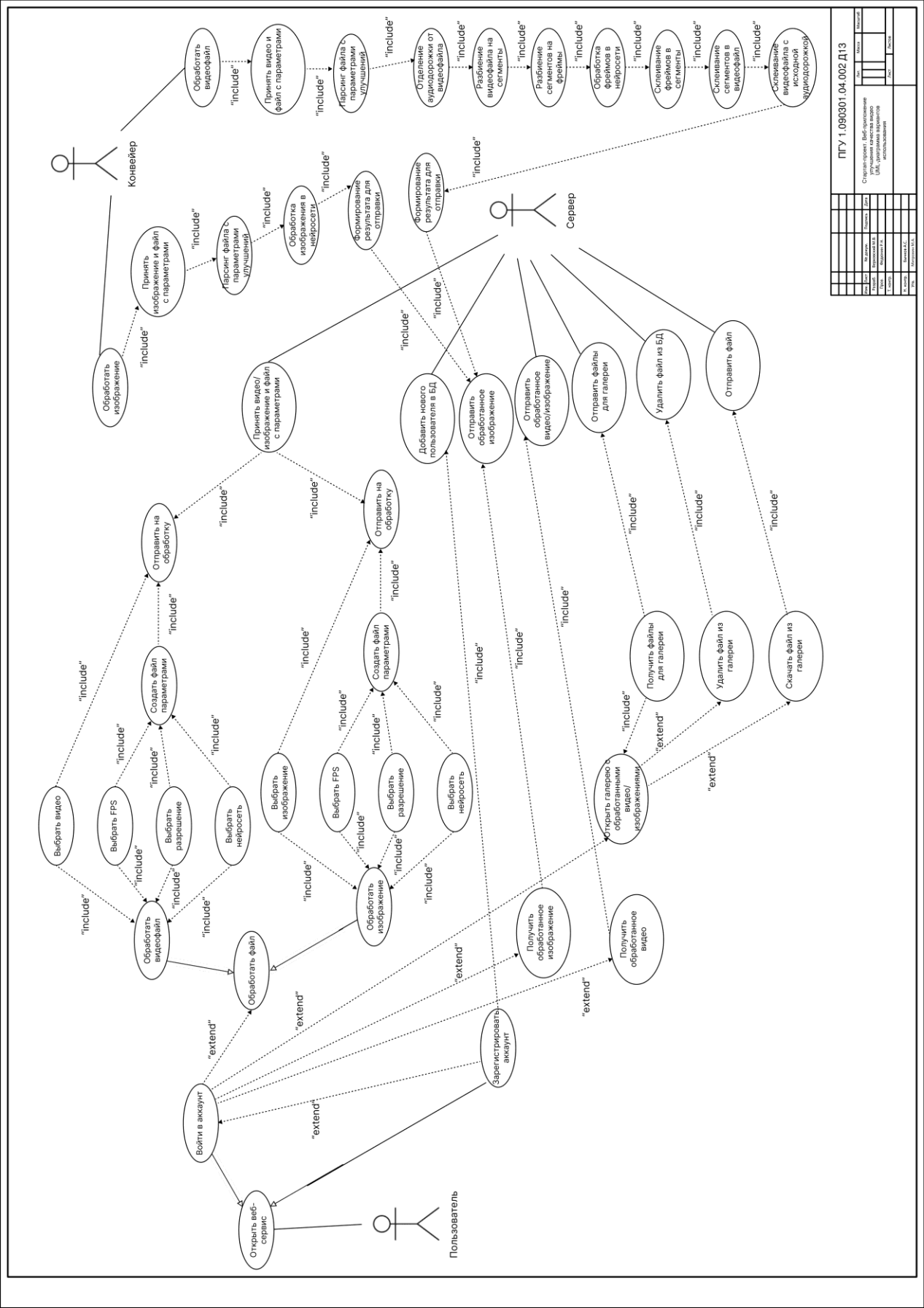
(Обязательное)

Стартап-проект. Веб-приложение улучшения качества видео

UML-диаграммы

					ПГУ1.090301.04.001 ПЗ	Лист
						157
Изм.	Лист	№ докум.	Подпись	Дата		

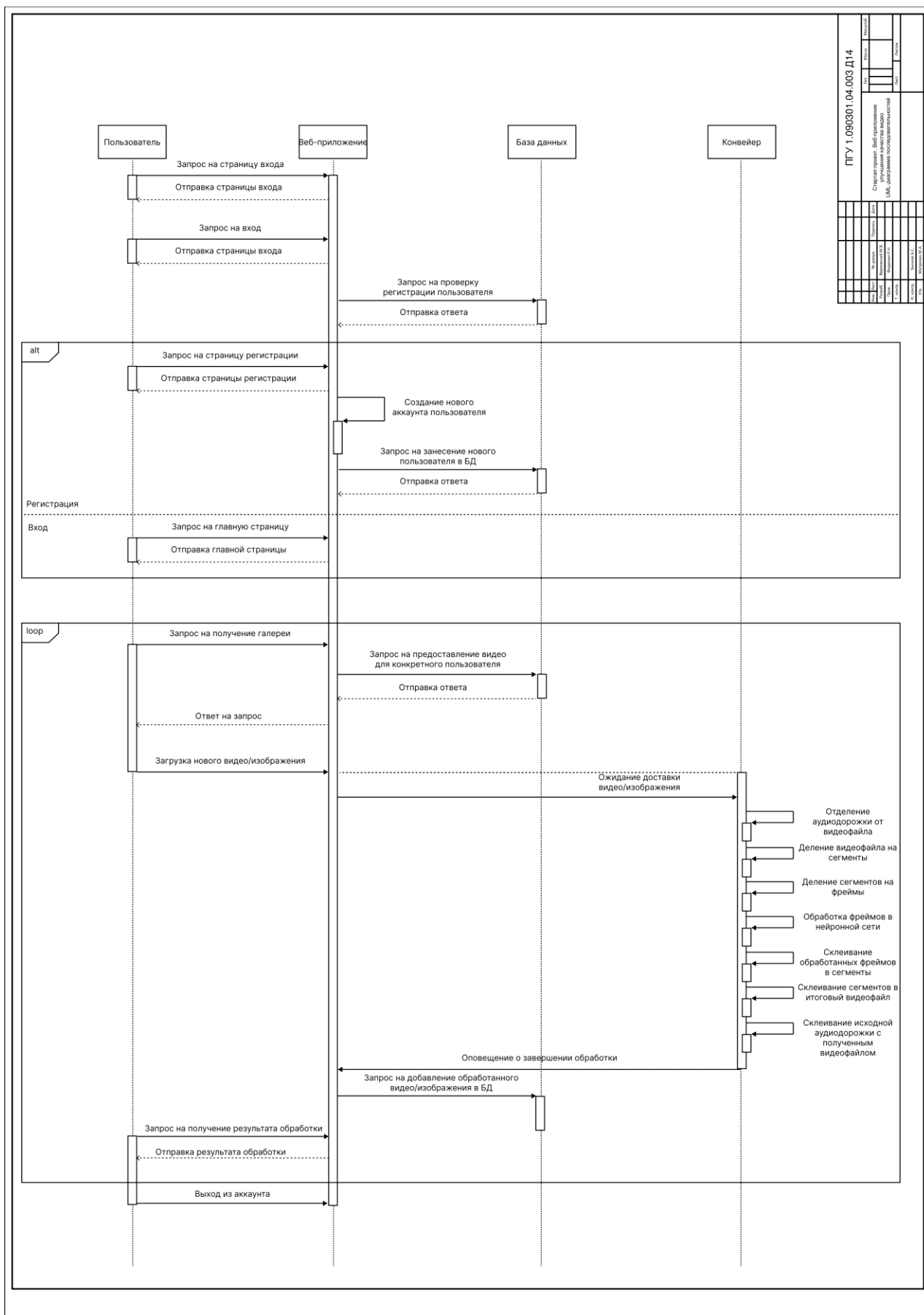
Приложение А.1 – UML-диаграмма вариантов использования



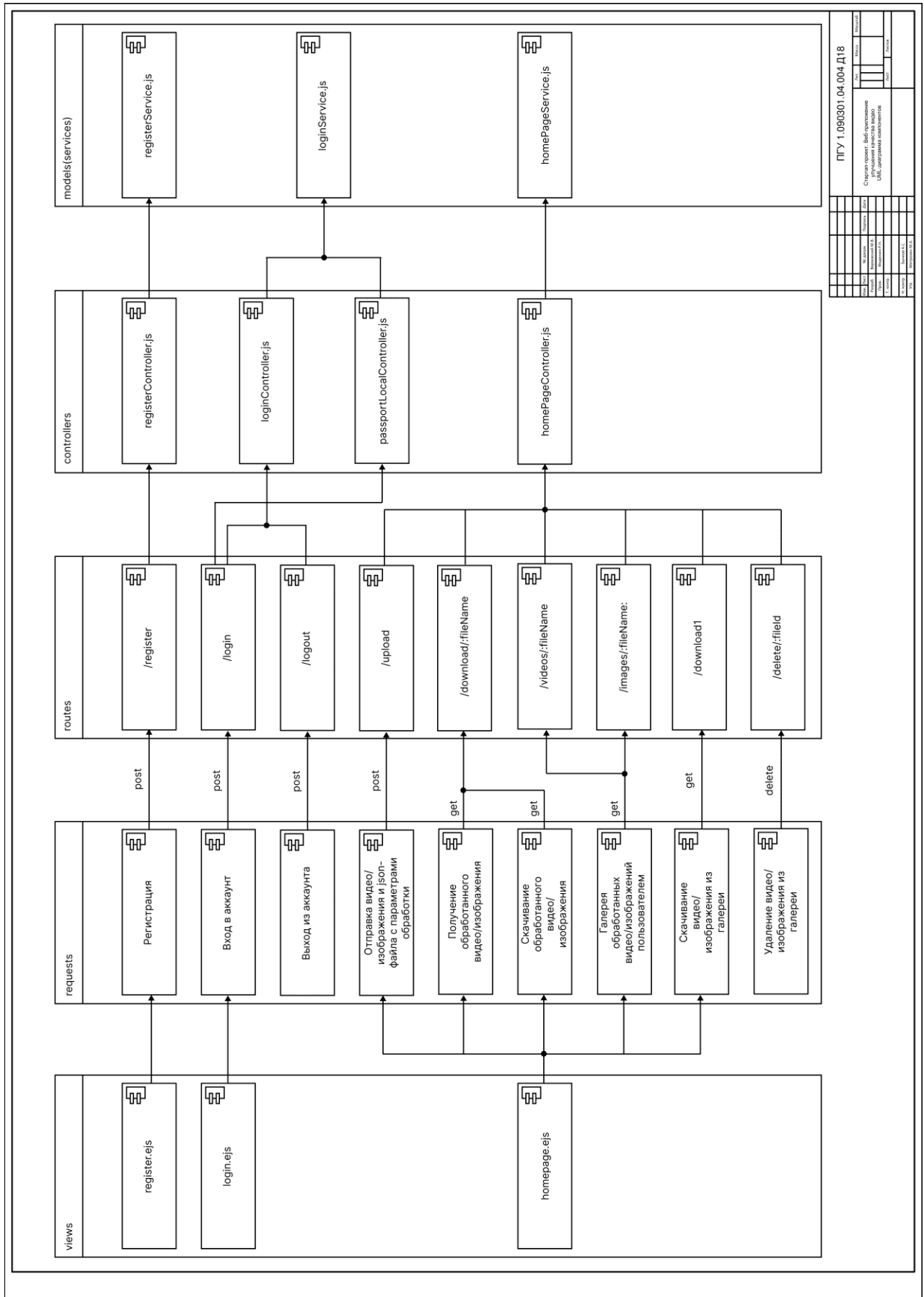
ПГУ 1.090301.04.002 Д13									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия

Изм.	Лист	№ докум.	Подпись	Дата

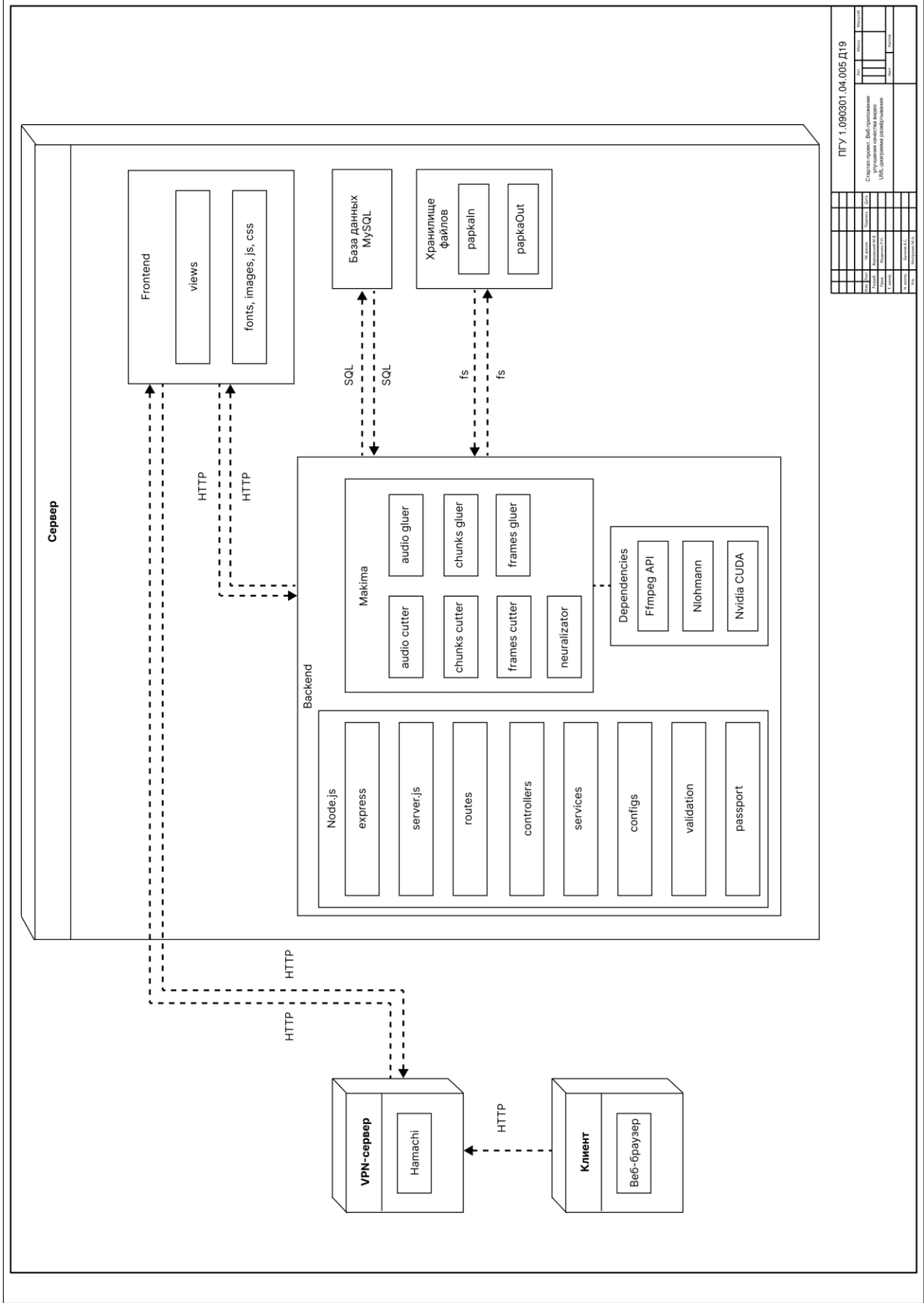
Приложение А.2 – UML-диаграмма последовательностей



Приложение А.3 – UML-диаграмма компонентов



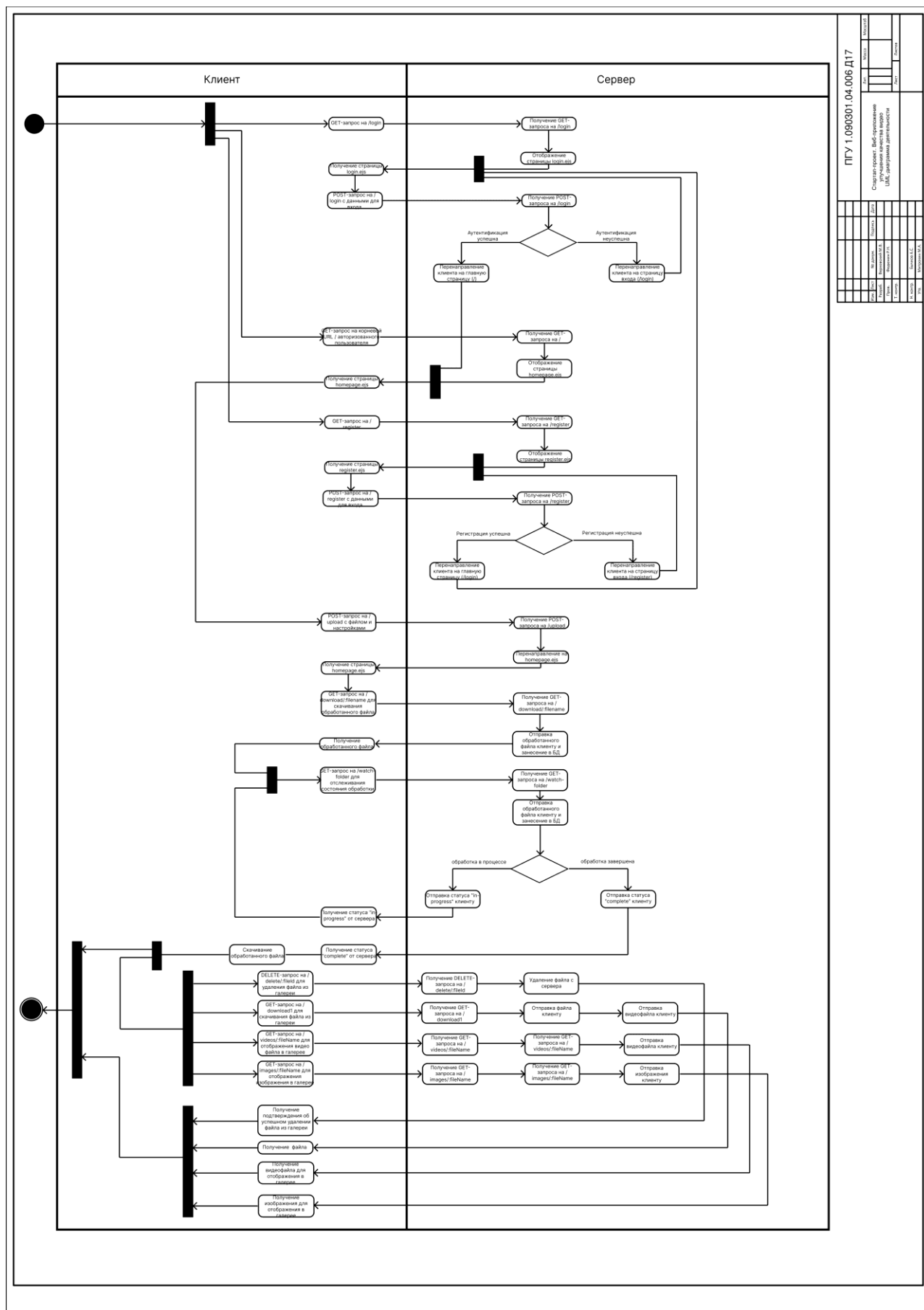
Приложение А.4 – UML-диаграмма развёртывания



ПГУ 1.090301.04.005 Д19									
Страница: 1 из 1									
Содержание: Видеонаблюдение									
Исполнитель: И.И.И.									
Исполнитель: И.И.И.									
Исполнитель: И.И.И.									
Исполнитель: И.И.И.									
Исполнитель: И.И.И.									
Исполнитель: И.И.И.									
Исполнитель: И.И.И.									

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.5 – UML-диаграмма деятельности



Приложение Б

(Обязательное)

Стартап-проект. Веб-приложение улучшения качества видео

Листинг программы

					ПГУ1.090301.04.001 ПЗ	Лист
						163
Изм.	Лист	№ докум.	Подпись	Дата		

Файл .env

```
APP_HOST=localhost
PORT=3000
NODE_ENV=developer

DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
DB_NAME=mydatabase
DB_USERNAME=root
DB_PASSWORD=34rtM!876
```

Файл DBConnection.js

```
require("dotenv").config();
import mysql from "mysql2";

let connection = mysql.createConnection({
  host: process.env.DB_HOST,
  user: process.env.DB_USERNAME,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME,
});

connection.connect(function (err) {
  if (err) throw err;
  console.log("БД подключена");
});

module.exports = connection;
```

Файл viewEngine.js

```
import express from "express";

let configViewEngine = (app) => {
  app.use(express.static("./src/public"));
  app.set("view engine", "ejs");
  app.set("views", "./src/views");
};

module.exports = configViewEngine;

Файл homePageController.js:
const homePageService = require("../services/homePageService");

const handleHelloWorld = async (req, res) => {
  try {
    const videos = await homePageService.getVideos(req.user.id);
    const uploadStatus = req.session.uploadStatus;
```

```

    req.session.uploadStatus = null; // очистить переменную
сеанса
    res.render("homepage.ejs", {
        user: req.user,
        videos: videos,
        uploadStatus: uploadStatus,
        path: require("path"), // передать модуль 'path' в шаблон
    });
} catch (err) {
    console.error("Error fetching videos:", err);
    const uploadStatus = req.session.uploadStatus;
    req.session.uploadStatus = null; // очистить переменную
сеанса
    res.render("homepage.ejs", {
        user: req.user,
        videos: [],
        uploadStatus: uploadStatus,
        path: require("path"), // передать модуль 'path' в шаблон
    });
}
};

module.exports = {
    handleHelloWorld,
};

```

Файл loginController.js

```

import { validationResult } from "express-validator";
import loginService from "../services/loginService";

let getPageLogin = (req, res) => {
    return res.render("login.ejs", {
        errors: req.flash("errors"),
    });
};

let handleLogin = async (req, res) => {
    let errorsArr = [];
    let validationErrors = validationResult(req);
    if (!validationErrors.isEmpty()) {
        let errors = Object.values(validationErrors.mapped());
        errors.forEach((item) => {
            errorsArr.push(item.msg);
        });
        req.flash("errors", errorsArr);
        return res.redirect("/login");
    }

    try {
        await loginService.handleLogin(req.body.email, req.body.password);
    }
};

```

					ПГУ1.090301.04.001 ПЗ	Лист
						165
Изм.	Лист	№ докум.	Подпись	Дата		

```

    return res.redirect("/");
  } catch (err) {
    req.flash("errors", err);
    return res.redirect("/login");
  }
};

let checkLoggedIn = (req, res, next) => {
  if (!req.isAuthenticated()) {
    return res.redirect("/login");
  }
  next();
};

let checkLoggedOut = (req, res, next) => {
  if (req.isAuthenticated()) {
    return res.redirect("/");
  }
  next();
};

let postLogOut = (req, res) => {
  req.session.destroy(function (err) {
    return res.redirect("/login");
  });
};

module.exports = {
  getPageLogin: getPageLogin,
  handleLogin: handleLogin,
  checkLoggedIn: checkLoggedIn,
  checkLoggedOut: checkLoggedOut,
  postLogOut: postLogOut,
};

```

Файл passportLocalController.js

```

import passportLocal from "passport-local";
import passport from "passport";
import loginService from "../services/loginService";

let LocalStrategy = passportLocal.Strategy;

let initPassportLocal = () => {
  passport.use(new LocalStrategy({
    usernameField: 'email',
    passwordField: 'password',
    passReqToCallback: true
  },
    async (req, email, password, done) => {
      try {

```

```

        await
loginService.findUserByEmail(email).then(async (user) => {
            if (!user) {
                return done(null, false,
req.flash("errors", `Пользователь с email "${email}" не
существует`));
            }
            if (user) {
                let match = await
loginService.comparePassword(password, user);
                if (match === true) {
                    return done(null, user, null)
                } else {
                    return done(null, false,
req.flash("errors", match)
                )
            }
        }
    });
} catch (err) {
    console.log(err);
    return done(null, false, { message: err });
}
}));

};

passport.serializeUser((user, done) => {
    done(null, user.id);
});

passport.deserializeUser((id, done) => {
    loginService.findUserById(id).then((user) => {
        return done(null, user);
    }).catch(error => {
        return done(error, null)
    });
});

module.exports = initPassportLocal;

```

Файл registerController.js

```

import registerService from "../services/registerService";
import { validationResult } from "express-validator";

let getPageRegister = (req, res) => {
    return res.render("register.ejs", {
        errors: req.flash("errors")
    });
};

```

```

let createNewUser = async (req, res) => {
  //validate required fields
  let errorsArr = [];
  let validationErrors = validationResult(req);
  if (!validationErrors.isEmpty()) {
    let errors = Object.values(validationErrors.mapped());
    errors.forEach((item) => {
      errorsArr.push(item.msg);
    });
    req.flash("errors", errorsArr);
    return res.redirect("/register");
  }

  //create a new user
  let newUser = {
    fullname: req.body.fullName,
    email: req.body.email,
    password: req.body.password
  };
  try {
    await registerService.createNewUser(newUser);
    return res.redirect("/login");
  } catch (err) {
    req.flash("errors", err);
    return res.redirect("/register");
  }
};

module.exports = {
  getPageRegister: getPageRegister,
  createNewUser: createNewUser
};

```

Файл jscode.js

```

let selectedVideo = null;
let settingsJSON = null;
let selectedVideoForDwnload = null;
let Current_fps = 0;

const selectVideoBtn =
document.getElementById("selectVideoBtn");
const ipInput = document.getElementById("ipInput");
const fpsSelect = document.getElementById("fpsSelect");
const sendAllBtn = document.getElementById("sendAllBtn");
const uploadForm = document.getElementById("uploadForm");
const videoInput = document.getElementById("videoInput");
const uploadStatusDiv = document.getElementById("upload-
status");
const finishedText = document.getElementById("finished-text");

const presetRadios = document.querySelectorAll(".preset-radio");

```

```

const ipInputContainer = document.getElementById("ipInput-
container");
const fpsSelectContainer = document.getElementById("fpsSelect-
container");
const neuralSelectContainer =
document.getElementById("neuralSelect-container");
const resolutionSelectContainer = document.getElementById(
    "resolutionSelect-container"
);

// Открыть диалог выбора файла
selectVideoBtn.addEventListener("click", () => {
    videoInput.click();
});

// Обработка выбора видеофайла
videoInput.addEventListener("change", () => {
    selectedVideo = videoInput.files[0];
    const fileExtension =
selectedVideo.name.split(".").pop().toLowerCase();
    document.getElementById(
        "selectedVideoName"
    ).textContent = `Выбрано: ${selectedVideo.name}`;
    sendAllBtn.style.display = "inline";

    selectedVideoForDwnload = selectedVideo.name;

// Отображение выбранного видео на странице
const selectedVideoFileDiv =
document.getElementById("selectedVideoFile");
selectedVideoFileDiv.innerHTML = "";

    if (isFileExtensionAllowed(fileExtension, ["mp4", "mkv",
"avi", "webm"])) {
        // Отображение видео
        const video = document.createElement("video");
        video.src = URL.createObjectURL(selectedVideo);
        video.controls = true;
        video.style.maxWidth = "680px";
        video.style.maxHeight = "680px";
        selectedVideoFileDiv.appendChild(video);

        // Проверяем, какой пресет выбран
        const selectedPreset = document.querySelector(
            'input[name="preset"]:checked'
        ).value;
        if (selectedPreset === "custom") {
            fpsSelectContainer.style.display = "block"; // Показываем
поле FPS для пользовательского пресета
        } else {
            fpsSelectContainer.style.display = "none"; // Скрываем
поле FPS для стандартного и максимального пресетов

```

```

    }
    } else if (
        isFileExtensionAllowed(fileExtension, ["png", "jpg", "webp",
"gif"]))
    ) {
// Отображение изображения
    const img = document.createElement("img");
    img.src = URL.createObjectURL(selectedVideo);
    img.alt = selectedVideo.name;
    img.style.maxWidth = "680px";
    img.style.maxHeight = "680px";
    selectedVideoFileDiv.appendChild(img);
    fpsSelectContainer.style.display = "none"; // Скрываем поле
FPS для изображений
    } else {
// Отображение ссылки для скачивания
    const link = document.createElement("a");
    link.href = URL.createObjectURL(selectedVideo);
    link.download = selectedVideo.name;
    link.textContent = `Скачать ${selectedVideo.name}`;
    selectedVideoFileDiv.appendChild(link);
    fpsSelectContainer.style.display = "none"; // Скрываем поле
FPS для других типов файлов
    }
});

presetRadios.forEach((radio) => {
    radio.addEventListener("change", (event) => {
// Удалить подчеркивание из ранее выбранного пресета
        presetRadios.forEach((r) => {
            r.parentElement.classList.remove("selected");
        });

// Добавляем подчеркивание к выбранному в данный момент пресету
        event.target.parentElement.classList.add("selected");

        const selectedPreset = event.target.value;
        switch (selectedPreset) {
            case "standard":
                ipInputContainer.style.display = "none";
                if (
                    isFileExtensionAllowed(
                        selectedVideo?.name?.split(".").pop().toLowerCase(),
                        ["mp4", "mkv", "avi", "webm"]
                    )
                ) {
                    fpsSelectContainer.style.display = "none"; // Скрываем
поле FPS для стандартного пресета и видео
                } else {
                    fpsSelectContainer.style.display = "none"; // Скрываем
поле FPS для стандартного пресета и изображений
                }
            }
        }
    });
});

```

```

        neuralSelectContainer.style.display = "none";
        resolutionSelectContainer.style.display = "none";
        fpsSelect.value = "30";
        neuralSelect.value = "waifu2x";
        resolutionSelect.value = "1920x1080";
        break;
    case "max":
        ipInputContainer.style.display = "none";
        if (
            isFileExtensionAllowed(
                selectedVideo?.name?.split(".").pop().toLowerCase(),
                ["mp4", "mkv", "avi", "webm"]
            )
        ) {
            fpsSelectContainer.style.display = "none"; // Скрываем
поле FPS для максимального пресета и видео
        } else {
            fpsSelectContainer.style.display = "none"; // Скрываем
поле FPS для максимального пресета и изображений
        }
        neuralSelectContainer.style.display = "none";
        resolutionSelectContainer.style.display = "none";
        fpsSelect.value = "120";
        neuralSelect.value = "realesrgan";
        resolutionSelect.value = "3840x2160";
        break;
    case "custom":
        ipInputContainer.style.display = "none";
        if (
            isFileExtensionAllowed(
                selectedVideo?.name?.split(".").pop().toLowerCase(),
                ["mp4", "mkv", "avi", "webm"]
            )
        ) {
            fpsSelectContainer.style.display = "block"; //
Показываем поле FPS для пользовательского пресета и видео
        } else {
            fpsSelectContainer.style.display = "none"; // Скрываем
поле FPS для пользовательского пресета и изображений
        }
        neuralSelectContainer.style.display = "block";
        resolutionSelectContainer.style.display = "block";
        break;
    default:
        break;
    }
});
});

function isFileExtensionAllowed(fileExtension,
allowedExtensions) {

```

```

    return
    allowedExtensions.includes(fileExtension.toLowerCase());
}

uploadForm.addEventListener("submit", (event) => {
    event.preventDefault();

    const fileExtension =
selectedVideo.name.split(".").pop().toLowerCase();
    if (
        !isFileExtensionAllowed(fileExtension, [
            "mp4",
            "mkv",
            "avi",
            "webm",
            "png",
            "jpg",
            "webp",
            "gif",
        ])
    ) {
        alert(
            "Выбран неверный формат файла. Допустимые форматы: .mp4,
.mkv, .avi, .WebM, .png, .jpg, .WebP, .gif"
        );
        return;
    }

    const formData = new FormData();
    formData.append("video", selectedVideo);
    const selectedPreset =
document.querySelector('input[name="preset"]:checked')
        .value;

    // Создаем JSON-объект с выбранными настройками
    const settingsJSON = {
        Neural: document.getElementById("neuralSelect").value,
        Horizontal: parseInt(
            document.getElementById("resolutionSelect").value.split("x
") [0]
        ),
        Vertical: parseInt(
            document.getElementById("resolutionSelect").value.split("x
") [1]
        ),
    };

    if (isFileExtensionAllowed(fileExtension, ["mp4", "mkv",
"avi", "webm"])) {
        settingsJSON.Current_fps = 0;
        settingsJSON.Upscale_fps = parseInt(fpsSelect.value);
    } else {

```

```

        delete settingsJSON.Upscale_fps; // Удаляем FPS из настроек,
        если выбран не видео файл
    }

    // Меняем порядок свойств в JSON-объекте
    const videoSettingsJSON = {
        Current_fps: settingsJSON.Current_fps, // Сохраняем порядок
        свойств
        Upscale_fps: settingsJSON.Upscale_fps,
        Neural: settingsJSON.Neural,
        Horizontal: settingsJSON.Horizontal,
        Vertical: settingsJSON.Vertical,
    };

    formData.append("settings",
    JSON.stringify(videoSettingsJSON));

    const xhr = new XMLHttpRequest();
    xhr.open("POST", `/upload`, true);
    xhr.upload.addEventListener("progress", (e) => {
    // обрабатываем ход загрузки, если необходимо
    });
    xhr.addEventListener("load", () => {
        if (xhr.status === 200) {
            console.log("Files successfully uploaded and saved");
            uploadStatusDiv.innerHTML =
                "Файл успешно загружен на сервер, ожидайте конца
обработки";
        } else {
            console.error("Error uploading files");
        }
    });
    xhr.send(formData);
});

// Обработчик события для кнопки "Скачать обработанный файл"
document.getElementById("downloadButton").addEventListener("click", () => {
    const fileName = selectedVideoForDwnload;

    const xhr = new XMLHttpRequest();
    const filePath = `/download/${fileName}`;
    const downloadUrl = `${window.location.origin}${filePath}`;

    xhr.open("GET", downloadUrl, true);
    xhr.responseType = "blob";
    xhr.onload = () => {
        if (xhr.status === 200) {
            selectedDownloadFile = xhr.response;
            const downloadLink = document.createElement("a");
            downloadLink.href =
                URL.createObjectURL(selectedDownloadFile);

```

```

downloadLink.download = fileName;

// Отображаем скачиваемый файл на странице
const downloadedFileDiv =
document.getElementById("downloadedFile");
downloadedFileDiv.innerHTML = "";

if (selectedDownloadFile.type.startsWith("image/")) {
    // Отображаем изображение
    const img = document.createElement("img");
    img.src = URL.createObjectURL(selectedDownloadFile);
    img.alt = fileName;
    img.style.maxWidth = "680px";
    img.style.maxHeight = "680px";
    downloadedFileDiv.appendChild(img);
} else if (selectedDownloadFile.type.startsWith("video/"))
{
    // Отображаем видео
    const video = document.createElement("video");
    video.src = URL.createObjectURL(selectedDownloadFile);
    video.controls = true;
    video.style.maxWidth = "680px";
    video.style.maxHeight = "680px";
    downloadedFileDiv.appendChild(video);
} else {
    // Отображаем ссылку на скачивание файла
    downloadLink.textContent = `Скачать ${fileName}`;
    downloadedFileDiv.appendChild(downloadLink);
}

downloadLink.click();
} else {
    console.error(`Failed to download file: ${fileName}`);
}
};
xhr.onerror = () => {
    console.error(`Failed to download file: ${fileName}`);
};
xhr.send();
});

function watchFolder() {
    $.ajax({
        url: "/watch-folder",
        type: "GET",
        success: function (data) {
            console.log("Server response:", data);
            if (data.status === "complete") {
                document.getElementById("upload-status").innerHTML =
                    "Обработка завершена";
                clearInterval(watchIntervalId); // Останавливаем
проверку папки

```

```

        document.getElementById("downloadButton").style.display
= "inline";
    } else {
        // Продолжаем проверять папку
    }
},
error: function (xhr, status, error) {
    console.error("Error watching folder:", error);
},
});
}

// Начинаем проверять папку каждые 3 секунды
const watchIntervalId = setInterval(watchFolder, 3000);

const deleteButtons = document.querySelectorAll(".delete-link");

// Добавляем обработчик события клика на каждую кнопку
deleteButtons.forEach((button) => {
    button.addEventListener("click", async (event) => {
        const fileId = event.target.dataset.fileId;

        try {
            // Отправляем запрос на сервер для удаления файла
            const response = await fetch(`/delete/${fileId}`, {
                method: "DELETE",
            });

            if (response.ok) {
                // Удаляем элемент галереи из DOM
                event.target.closest(".gallery-item").remove();
                console.log("Файл успешно удален");
            } else {
                console.error("Ошибка при удалении файла");
            }
        } catch (error) {
            console.error("Ошибка при удалении файла:", error);
        }
    });
});

// Получаем все кнопки скачивания
const downloadButtons = document.querySelectorAll(".download-
link");

// Добавляем обработчик события клика на каждую кнопку
скачивания
downloadButtons.forEach((button) => {
    button.addEventListener("click", async (event) => {
        event.preventDefault(); // Предотвращаем стандартное
поведение ссылки

```

```

const filePath = event.target.dataset.filePath;
console.log("Downloading file:", filePath);

try {
  // Отправляем запрос на сервер для скачивания файла
  const response = await
fetch(`/download1?fileName=${filePath}`, {
  method: "GET",
});

  if (response.ok) {
    // Создаем ссылку для скачивания и кликаем по ней
    const fileBlob = await response.blob();
    const downloadLink = document.createElement("a");
    downloadLink.href = URL.createObjectURL(fileBlob);
    downloadLink.download = filePath.split("/").pop();
    document.body.appendChild(downloadLink);
    downloadLink.click();
    document.body.removeChild(downloadLink);
  } else {
    const errorMessage = await response.text();
    console.error("Ошибка при скачивании файла:",
errorMessage);
    alert(`Ошибка при скачивании файла: ${errorMessage}`);
  }
} catch (error) {
  console.error("Ошибка при скачивании файла:", error);
  alert(`Ошибка при скачивании файла: ${error.message}`);
}
});

const filterImagesBtn = document.getElementById("filterImages");
const filterVideosBtn = document.getElementById("filterVideos");
const showAllBtn = document.getElementById("showAll");
const galleryItems = document.querySelectorAll(".gallery-item");

let activeFilterButton = showAllBtn;
activeFilterButton.style.backgroundColor = "black";
activeFilterButton.style.color = "white";

filterImagesBtn.addEventListener("click", () => {
  activeFilterButton.style.backgroundColor = "";
  activeFilterButton.style.color = "";
  activeFilterButton = filterImagesBtn;
  activeFilterButton.style.backgroundColor = "black";
  activeFilterButton.style.color = "white";

  galleryItems.forEach((item) => {
    if (item.querySelector(".image-container")) {
      item.style.display = "block";
    } else {

```

```

        item.style.display = "none";
    }
    });
});

filterVideosBtn.addEventListener("click", () => {
    activeFilterButton.style.backgroundColor = "";
    activeFilterButton.style.color = "";
    activeFilterButton = filterVideosBtn;
    activeFilterButton.style.backgroundColor = "black";
    activeFilterButton.style.color = "white";

    galleryItems.forEach((item) => {
        if (item.querySelector(".video-container")) {
            item.style.display = "block";
        } else {
            item.style.display = "none";
        }
    });
});

showAllBtn.addEventListener("click", () => {
    activeFilterButton.style.backgroundColor = "";
    activeFilterButton.style.color = "";
    activeFilterButton = showAllBtn;
    activeFilterButton.style.backgroundColor = "black";
    activeFilterButton.style.color = "white";

    galleryItems.forEach((item) => {
        item.style.display = "block";
    });
});

```

Файл web.js

```

import express from "express";
import homePageController from
"../controllers/homePageController";
import registerController from
"../controllers/registerController";
import loginController from "../controllers/loginController";
import auth from "../validation/authValidation";
import passport from "passport";
import initPassportLocal from
"../controllers/passportLocalController";

// Инициализируем весь passport
initPassportLocal();

let router = express.Router();

let initWebRoutes = (app) => {

```

```

router.get(
  "/",
  loginController.checkLoggedIn,
  homePageController.handleHelloWorld
);
router.get(
  "/login",
  loginController.checkLoggedOut,
  loginController.getPageLogin
);
router.post(
  "/login",
  passport.authenticate("local", {
    successRedirect: "/",
    failureRedirect: "/login",
    successFlash: true,
    failureFlash: true,
  })
);

router.get("/register", registerController.getPageRegister);
router.post(
  "/register",
  auth.validateRegister,
  registerController.createNewUser
);
router.post("/logout", loginController.postLogOut);
return app.use("/", router);
};
module.exports = initWebRoutes;

```

Файл homePageService.js

```

const connection = require("../configs/DBConnection");

const getVideos = async (userId) => {
  try {
    const [
      rows,
      fields,
    ] = await connection
      .promise()
      .query("SELECT * FROM video_files WHERE user_id = ?",
[userId]);
    return rows;
  } catch (err) {
    throw err;
  }
};

const saveFileToDatabase = async (req, fileName, filePath,
fileType, description) => {

```

					ПГУ1.090301.04.001 ПЗ	Лист
						178
Изм.	Лист	№ докум.	Подпись	Дата		

```

    const query = `
        INSERT INTO video_files (user_id, title, file_path,
file_type, description)
        VALUES (?, ?, ?, ?, ?)
    `;
    const values = [req.user.id, fileName, filePath, fileType,
description];
    try {
        await connection.promise().query(query, values);
        console.log("Video/Image file saved to database");
    } catch (err) {
        console.error("Ошибка сохранения видео/изображения в БД:",
err);
    }
};

const deleteFileFromDatabase = async (fileId) => {
    try {
        await connection.promise().query("DELETE FROM video_files
WHERE id = ?", [fileId]);
    } catch (error) {
        console.error("Ошибка удаления файла из базы данных:",
error);
        throw error;
    }
};

const getFileInfoFromDatabase = async (fileName) => {
    try {
        const [rows] = await connection.promise().query("SELECT *
FROM video_files WHERE file_path = ?", [fileName]);
        return rows;
    } catch (err) {
        throw err;
    }
};

module.exports = {
    getVideos,
    saveFileToDatabase,
    deleteFileFromDatabase,
    getFileInfoFromDatabase,
};

```

Файл loginService.js

```

import DBConnection from "../configs/DBConnection";
import bcrypt from "bcryptjs";

let handleLogin = (email, password) => {
    return new Promise(async (resolve, reject) => {
        // Проверяем существует или нет email

```

					ПГУ1.090301.04.001 ПЗ	Лист
						179
Изм.	Лист	№ докум.	Подпись	Дата		

```

    let user = await findUserByEmail(email);
    if (user) {
        // Сравниваем пароли
        await bcrypt.compare(password,
user.password).then((isMatch) => {
            if (isMatch) {
                resolve(true);
            } else {
                reject(`Пароль неверный`);
            }
        });
    } else {
        reject(`Пользователь с email "${email}" не существует`);
    }
});
};

let findUserByEmail = (email) => {
    return new Promise((resolve, reject) => {
        try {
            DBConnection.query(
                " SELECT * FROM `users` WHERE `email` = ? ",
                email,
                function (err, rows) {
                    if (err) {
                        reject(err);
                    }
                    let user = rows[0];
                    resolve(user);
                }
            );
        } catch (err) {
            reject(err);
        }
    });
};

let findUserById = (id) => {
    return new Promise((resolve, reject) => {
        try {
            DBConnection.query(
                " SELECT * FROM `users` WHERE `id` = ? ",
                id,
                function (err, rows) {
                    if (err) {
                        reject(err);
                    }
                    let user = rows[0];
                    resolve(user);
                }
            );
        } catch (err) {

```

```

        reject(err);
    }
    });
};

let comparePassword = (password, userObject) => {
    return new Promise(async (resolve, reject) => {
        try {
            await bcrypt.compare(password,
userObject.password).then((isMatch) => {
                if (isMatch) {
                    resolve(true);
                } else {
                    resolve(`Пароль неверный`);
                }
            });
        } catch (e) {
            reject(e);
        }
    });
};

module.exports = {
    handleLogin: handleLogin,
    findUserByEmail: findUserByEmail,
    findUserById: findUserById,
    comparePassword: comparePassword,
};

```

Файл registerService.js

```

import DBConnection from "../configs/DBConnection";
import bcrypt from "bcryptjs";

let createNewUser = (data) => {
    return new Promise(async (resolve, reject) => {
        // check email is exist or not
        let isEmailExist = await checkExistEmail(data.email);
        if (isEmailExist) {
            reject(
                `Пользователь с email "${data.email}" уже существует.
Пожалуйста, выберите другой email`
            );
        } else {
            // хешируем пароль
            let salt = bcrypt.genSaltSync(10);
            let userItem = {
                fullname: data.fullname,
                email: data.email,
                password: bcrypt.hashSync(data.password, salt),
            };

```

```

        //создаём новый аккаунт
        DBConnection.query(
            " INSERT INTO users set ? ",
            userItem,
            function (err, rows) {
                if (err) {
                    reject(false);
                }
                resolve("Создание нового пользователя прошло
успешно");
            }
        );
    });
};

let checkExistEmail = (email) => {
    return new Promise((resolve, reject) => {
        try {
            DBConnection.query(
                " SELECT * FROM `users` WHERE `email` = ? ",
                email,
                function (err, rows) {
                    if (err) {
                        reject(err);
                    }
                    if (rows.length > 0) {
                        resolve(true);
                    } else {
                        resolve(false);
                    }
                }
            );
        } catch (err) {
            reject(err);
        }
    });
};

module.exports = {
    createNewUser: createNewUser,
};

```

Файл authValidation.js

```

import { check } from "express-validator";

let validateRegister = [
    check("email", "Неправильный email").isEmail().trim(),

    check(
        "password",
        "Неправильный пароль. Пароль должен быть длиннее 2 символов"
    )
];

```

```

    ).isLength({ min: 2 })),

    check("passwordConfirmation", "Пароли не совпали").custom(
      (value, { req }) => {
        return value === req.body.password;
      }
    ),
  ];

let validateLogin = [
  check("email", "Неправильный email").isEmail().trim(),

  check("password", "Неправильный пароль").not().isEmpty(),
];

module.exports = {
  validateRegister: validateRegister,
  validateLogin: validateLogin,
};

```

Файл server.js

```

require("dotenv").config();
import express from "express";
import configViewEngine from "../configs/viewEngine";
import initWebRoutes from "../routes/web";
import bodyParser from "body-parser";
import cookieParser from "cookie-parser";
import session from "express-session";
import connectFlash from "connect-flash";
import passport from "passport";
import multer from "multer";
import fs from "fs";
const path = require("path");
const homePageService = require("../services/homePageService");

let app = express();

// Используем парсер cookie
app.use(cookieParser("secret"));

// Настраиваем сеанс
app.use(
  session({
    secret: "secret",
    resave: true,
    saveUninitialized: false,
    cookie: {
      maxAge: 1000 * 60 * 60 * 24, // 1 день
    },
  })
);

```

```

// Enable body parser post data
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

// Настраиваем view engine
configViewEngine(app);

// Enable flash message
app.use(connectFlash());

// Настраиваем passport middleware
app.use(passport.initialize());
app.use(passport.session());

// Создаём папку papkaIn на диске D, если она не существует
const videosFolderPath = "D:/papkaIn";
if (!fs.existsSync(videosFolderPath)) {
    fs.mkdirSync(videosFolderPath);
}

// Настраиваем multer storage
const storage = multer.diskStorage({
    destination: function (req, file, cb) {
        cb(null, videosFolderPath);
    },
    filename: function (req, file, cb) {
        cb(null, file.originalname);
    },
});

// multer экземпляр
const upload = multer({ storage: storage });

app.post("/upload", upload.single("video"), async (req, res) =>
{
    const { ipAddress, fps, settings } = req.body;

    if (!req.file) {
        req.session.uploadStatus = "Ни один видеофайл не был
загружен.";
        res.redirect("/");
        return;
    }

    const videoFile = req.file;
    const settingsObject = JSON.parse(settings);

    // Создаем папку с именем пользователя
    const userFolderPath = `D:/papkaIn/${req.user.fullname}`;
    const userFolderPath2 = `D:/papkaOut/${req.user.fullname}`;

```

```

// Определяем папки назначения
const fileExtension = path
  .extname(videoFile.originalname)
  .slice(1)
  .toLowerCase();
const inFolderPath = isFileExtensionAllowed(fileExtension, [
  "mp4",
  "mkv",
  "avi",
  "webm",
])
? `${userFolderPath}/videos`
: isFileExtensionAllowed(fileExtension, ["png", "jpg",
"webp", "gif"])
? `${userFolderPath}/images`
: userFolderPath;

const outFolderPath = isFileExtensionAllowed(fileExtension, [
  "mp4",
  "mkv",
  "avi",
  "webm",
])
? `${userFolderPath2}/videos`
: isFileExtensionAllowed(fileExtension, ["png", "jpg",
"webp", "gif"])
? `${userFolderPath2}/images`
: userFolderPath2;

// Создаем папки, если они не существуют
if (!fs.existsSync(userFolderPath)) {
  fs.mkdirSync(userFolderPath);
}
if (!fs.existsSync(userFolderPath2)) {
  fs.mkdirSync(userFolderPath2);
}
if (!fs.existsSync(inFolderPath)) {
  fs.mkdirSync(inFolderPath, { recursive: true });
}
if (!fs.existsSync(outFolderPath)) {
  fs.mkdirSync(outFolderPath, { recursive: true });
}

const fileName = path.parse(videoFile.originalname).name;
const videoFolderPath = `${inFolderPath}/${fileName}`;
const videoFolderPath2 = `${outFolderPath}/${fileName}`;

if (!fs.existsSync(videoFolderPath)) {
  fs.mkdirSync(videoFolderPath);
}

if (!fs.existsSync(videoFolderPath2)) {

```

```

    fs.mkdirSync(videoFolderPath2);
  }

  // Сохраняем видеофайл и настройки в папке
  const videoFilePath =
    `${videoFolderPath}/${videoFile.originalname}`;
  fs.renameSync(videoFile.path, videoFilePath);
  fs.writeFileSync(
    `${videoFolderPath}/${fileName}-settings.json`,
    JSON.stringify(settingsObject)
  );

  const fileExtension1 = path.extname(videoFile.originalname);
  req.session.uploadedFileName = `${fileName}${fileExtension1}`;
  req.session.uploadStatus = `Файл успешно загружен на сервер`;
  res.redirect("/");
});

// инициализируем все веб-маршруты
initWebRoutes(app);

app.get("/videos/:fileName", (req, res) => {
  const fileName = req.params.fileName;
  const fileNameWithoutExtension = path.basename(
    fileName,
    path.extname(fileName)
  );

  const filePath = path.join(
    "D:",
    "papkaOut",
    req.user.fullname,
    "videos",
    `${fileNameWithoutExtension}`,
    fileName
  );

  fs.access(filePath, fs.constants.R_OK, (err) => {
    if (err) {
      console.error("Ошибка доступа к файлу:", err);
      res.status(404).send("Файл не найден");
      return;
    }

    res.sendFile(filePath);
  });
});

app.get("/images/:fileName", (req, res) => {
  const fileName = req.params.fileName;
  const fileNameWithoutExtension = path.basename(
    fileName,

```

```

    path.extname(fileName)
  );

  const filePath = path.join(
    "D:",
    "papkaOut",
    req.user.fullname,
    "images",
    `${fileNameWithoutExtension}`,
    fileName
  );

  fs.access(filePath, fs.constants.R_OK, (err) => {
    if (err) {
      console.error("Ошибка доступа к файлу:", err);
      res.status(404).send("Файл не найден");
      return;
    }

    res.sendFile(filePath);
  });
});

// Обработчик для маршрута по прослеживанию появления файла
'makima.access' в папке
app.get("/watch-folder", (req, res) => {
  const fileName = req.session.uploadedFileName;
  const fileNameWithoutExtension = path.basename(
    fileName,
    path.extname(fileName)
  );

  const fileExtension =
    path.extname(fileName).slice(1).toLowerCase();
  const folderPath = isFileExtensionAllowed(fileExtension, [
    "mp4",
    "mkv",
    "avi",
    "webm",
  ])
    ? path.join(
        "D:",
        "papkaOut",
        req.user.fullname,
        "videos",
        fileNameWithoutExtension
      )
    : isFileExtensionAllowed(fileExtension, ["png", "jpg",
"webp", "gif"]);
  ? path.join(
    "D:",
    "papkaOut",

```

```

        req.user.fullname,
        "images",
        fileNameWithoutExtension
    )
    : path.join("D:", "papkaOut", req.user.fullname,
fileNameWithoutExtension);

// Проверяем, существует ли папка
if (!fs.existsSync(folderPath)) {
    console.log("Папка не найдена:", folderPath);
    res.json({ status: "in-progress" });
    return;
}

console.log("Watching folder:", folderPath);

// Проверяем наличие файла 'makima.access' в папке
const checkFile = () => {
    fs.access(
        path.join(folderPath, "makima.access"),
        fs.constants.F_OK,
        (err) => {
            if (!err) {
                console.log("makima.access найден в папке");
                req.session.uploadStatus = "Обработка завершена";
                res.json({ status: "complete" });
            } else {
                console.log("makima.access не найден в папке");
                res.json({ status: "in-progress" });
            }
        }
    );
};

// Вызываем функцию проверки каждые 3 секунды
const intervalId = setInterval(checkFile, 3000);

// Очищаем интервал при завершении запроса
res.on("close", () => {
    clearInterval(intervalId);
    console.log("Остановка просмотра папки:", folderPath);
});

checkFile(); // Первая проверка
});

app.get("/download/:fileName", async (req, res) => {
    const fileName = req.params.fileName;
    const fileNameWithoutExtension = path.basename(
        fileName,
        path.extname(fileName)
    );
};

```

```

    const fileExtension =
path.extname(fileName).slice(1).toLowerCase();
    const folderPath = isFileExtensionAllowed(fileExtension, [
        "mp4",
        "mkv",
        "avi",
        "webm",
    ])
    ? path.join(
        "D:",
        "papkaOut",
        req.user.fullname,
        "videos",
        fileNameWithoutExtension
    )
    : isFileExtensionAllowed(fileExtension, ["png", "jpg",
"webp", "gif"])
    ? path.join(
        "D:",
        "papkaOut",
        req.user.fullname,
        "images",
        fileNameWithoutExtension
    )
    : path.join("D:", "papkaOut", req.user.fullname,
fileNameWithoutExtension);

    const filePath = path.join(folderPath, fileName);

    fs.access(filePath, fs.constants.R_OK, (err) => {
        if (err) {
            console.error("Ошибка доступа к файлу:", err);
            res.status(404).send("Файл не найден");
            return;
        }

        // Определяем тип файла (видео или изображение)
        const fileType = isFileExtensionAllowed(fileExtension, [
            "mp4",
            "mkv",
            "avi",
            "webm",
        ])
        ? "video"
        : isFileExtensionAllowed(fileExtension, ["png", "jpg",
"webp", "gif"])
        ? "image"
        : "unknown";

        // Получаем содержимое json-файла
        const settingsFilePath = path.join(

```

```

        "D:",
        "папкаIn",
        req.user.fullname,
        fileType === "video" ? "videos" : "images",
        fileNameWithoutExtension,
        `${fileNameWithoutExtension}-settings.json`
    );
    fs.access(settingsFilePath, fs.constants.R_OK, (err) => {
        if (err) {
            console.error("Ошибка доступа к файлу параметров:",
err);
            // Если файл настроек не найден, используйте пустую
строку в качестве описания
            const description = "";
            homePageService.saveFileToDatabase(req, fileName,
filePath, fileType, description);
            res.sendFile(filePath);
        } else {
            fs.readFile(settingsFilePath, "utf8", (err, data) => {
                if (err) {
                    console.error("Ошибка чтения файла параметров:",
err);
                    // Если не удалось прочитать файл настроек,
используйте пустую строку в качестве описания
                    const description = "";
                    homePageService.saveFileToDatabase(req, fileName,
filePath, fileType, description);
                    res.sendFile(filePath);
                } else {
                    // Сохраняем содержимое json-файла в качестве
описания
                    const settingsObject = JSON.parse(data);
                    let description;
                    if (fileType === "video") {
                        description = `FPS: ${settingsObject.Upscale_fps},
Neural: ${settingsObject.Neural}, Resolution:
${settingsObject.Horizontal}x${settingsObject.Vertical}`;
                    } else if (fileType === "image") {
                        description = `Neural: ${settingsObject.Neural},
Resolution:
${settingsObject.Horizontal}x${settingsObject.Vertical}`;
                    } else {
                        description = "";
                    }
                    homePageService.saveFileToDatabase(req, fileName,
filePath, fileType, description);
                    res.sendFile(filePath);
                }
            });
        }
    });
});
});

```

```

});

function isFileExtensionAllowed(fileExtension,
allowedExtensions) {
    return
allowedExtensions.includes(fileExtension.toLowerCase());
}

app.delete("/delete/:fileId", async (req, res) => {
    const fileId = req.params.fileId;

    try {
        await homePageService.deleteFileFromDatabase(fileId);
        res.sendStatus(200);
    } catch (error) {
        console.error("Ошибка удаления файла:", error);
        res.sendStatus(500);
    }
});

app.get("/download1", async (req, res) => {
    const fileName = req.query.fileName;
    console.log("Запрошенный файл:", fileName);

    try {
        const rows = await
homePageService.getFileInfoFromDatabase(fileName);
        const file = rows[0];

        if (!file) {
            res.status(404).send("Файл не найден");
            return;
        }

        const filePath = file.file_path;
        console.log("Созданный путь к файлу:", filePath);

        fs.access(filePath, fs.constants.R_OK, (err) => {
            if (err) {
                console.error("Ошибка доступа к файлу:", err);
                res.status(500).send("Ошибка доступа к файлу");
                return;
            }

            console.log("Отправка файла:", filePath);
            res.download(filePath, file.title, (err) => {
                if (err) {
                    console.error("Ошибка загрузки файла:", err);
                    res.status(500).send("Ошибка загрузки файла");
                }
            });
        });
    }
});

```

```

    } catch (err) {
      console.error("Ошибка получения файла из БД:", err);
      res.status(500).send("Ошибка при получении файла");
    }
  });

let port = process.env.PORT || 8000;
app.listen(port, () =>
  console.log(`Веб-приложение запущено на порту ${port}!`)
);

```

Файл homepage.ejs

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Home page</title>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <link rel="stylesheet" href="/css/style.css" />
    <link rel="preload" href="https://htmlacademy.ru/fonts/robot
o.woff2" as="font">
    <link
      rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/
bootstrap.min.css"
      integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
"
      crossorigin="anonymous"
    />
  </head>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6
.0/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/
2.9.3/umd/popper.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/
4.6.0/js/bootstrap.min.js"></script>

  <header>
    <div class="header-content">
      <div class="logo-container">
        
        <h1>Himen0</h1>
      </div>
      <div class="user-logout-container">
        <h2><%= user.fullname %></h2>
        <div class="logout-container">
          <form method="POST" action="/logout">

```

```

        <button type="submit" class="button_design_inv">Вы
йти</button>
    </form>
</div>
</div>
</div>
</header>

<div class="container1">
    <div class="main-content">
        <h2>Берстак</h2>
        <form
            id="uploadForm"
            enctype="multipart/form-data"
            method="POST"
            action="/upload"
        >
            <div class="form-group">
                <label for="videoInput">Выберите видео/изображение:<
/label>

                <div class="input-group">
                    <input
                        type="file"
                        id="videoInput"
                        name="video"
                        style="display: none"
                    />
                    <button
                        id="selectVideoBtn"
                        type="button"
                        class="button_design"
                    >
                        Выбрать файл для обработки
                    </button>
                    <p id="selectedVideoName" class="form-control-
plaintext"></p>
                </div>
            </div>

            <div class="preset-checkboxes">
                <label>
                    <input type="radio" name="preset" value="standard"
class="preset-radio" checked>
                        Стандартный
                    </label>

                <label>
                    <input type="radio" name="preset" value="max" clas
s="preset-radio">
                        Максимальный
                    </label>
                <br>

```

```

        <label>
            <input type="radio" name="preset" value="custom" c
lass="preset-radio">
            Кастомный
        </label>
        <label>
            <input type="radio" name="preset" value="individua
l" class="preset-radio">
            Индивидуальный
        </label>
        <br>
        <button id="savenewPresetBtn" class="button_design">
Записать пресет</button>
        </div>
        <br>
        <div class="form-group" id="ipInput-
container" style="display: none;">
            <label for="ipInput">Введите IP-адрес:</label>
            <input
                type="text"
                id="ipInput"
                name="ipAddress"
                placeholder="Введите IP-адрес"
                class="form-control"
            />
        </div>

        <div class="form-group" id="fpsSelect-
container" style="display: none;">
            <label for="fpsSelect">Выберите FPS:</label>
            <select id="fpsSelect" name="fps" class="form-
control">
                <option value="30">30 FPS</option>
                <option value="60">60 FPS</option>
                <option value="120">120 FPS</option>
            </select>
        </div>

        <div class="form-group" id="neuralSelect-
container" style="display: none;">
            <label for="neuralSelect">Выберите нейросеть:</label
>
            <select id="neuralSelect" name="neural" class="form-
control">
                <option value="waifu2x">waifu2x -
среднее качество</option>
                <option value="realcugan">realcugan -
высокое качество</option>
                <option value="realesrgan">realesrgan -
невероятное качество</option>
            </select>
        </div>

```

```

        <div class="form-group" id="resolutionSelect-
container" style="display: none;">
            <label for="resolutionSelect">Выберите разрешение:</
label>
            <select id="resolutionSelect" name="resolution" clas
s="form-control">
                <option value="800x600">800x600</option>
                <option value="1024x600">1024x600</option>
                <option value="1200x600">1200x600</option>
                <option value="1280x720">1280x720</option>
                <option value="1440x1080">1440x1080</option>
                <option value="1600x900">1600x900</option>
                <option value="1600x1200">1600x1200</option>
                <option value="1680x1050">1680x1050</option>
                <option value="1920x1080">1920x1080</option>
                <option value="1920x1200">1920x1200</option>
                <option value="2048x1024">2048x1024</option>
                <option value="2048x1152">2048x1152</option>
                <option value="2048x1536">2048x1536</option>
                <option value="2560x1080">2560x1080</option>
                <option value="2560x1440">2560x1440</option>
                <option value="2560x1600">2560x1600</option>
                <option value="3200x1800">3200x1800</option>
                <option value="3200x2400">3200x2400</option>
                <option value="3840x2160">3840x2160</option>
            </select>
        </div>

        <button
            type="submit"
            id="sendAllBtn"
            class="button_design"
        >
            Отправить на обработку
        </button>
    </br>
    </br>
        <button id="downloadButton" class="button_design" styl
e="display: none">
            Скачать обработанный файл
        </button>
    </br>
        <div id="upload-status">
            <% if (uploadStatus) { %> <%= uploadStatus %> <% } %
>

        </div>

    </div>

    <div class="main-content3">
        <div class="after-label">До</div>

```

```

        <div id="selectedVideoFile" class="selected-
file"></div>
    </div>
    <div class="main-content4">
        <div class="after-label">После</div>
        <div id="downloadedFile" class="selected-file"></div>
    </div>
</form>
</div>
</br>
<div class="video-gallery">
    <h1>Галерея</h1>
    <div class="gallery-filters">
        <button id="filterImages" class="button_design">Показать тол
ько изображения</button>
        <button id="filterVideos" class="button_design">Показать тол
ько видео</button>
        <button id="showAll" class="button_design">Показать всё</but
ton>
    </div>

    <div class="gallery-container">
        <% if (videos.length > 0) { %>
            <% videos.forEach(video => { %>
                <div class="gallery-item">
                    <% if (video.file_type === 'video') { %>
                        <div class="video-container">
                            <video controls>
                                <source src="/videos/<%= path.parse(video.file_p
ath).base %>" type="video/mp4">
                                    Ваш браузер не поддерживает тег видео.
                                </video>
                            </div>
                        <% } else if (video.file_type === 'image') { %>
                            <div class="image-container">
                                ">
                            </div>
                        <% } else { %>
                            <a href="/files/<%= path.parse(video.file_path).base
%>" download><%= video.title %></a>
                        <% } %>
                        <div class="gallery-caption">
                            <h3><%= video.title %></h3>
                            <p><%= video.description %></p>
                            <div class="action-container">
                                <span class="delete-link" data-file-
id="<%= video.id %>">Удалить</span>
                                <span class="download-link" data-file-
path="<%= video.file_path %>">Скачать</span>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

        </div>
        <% }) %>
        <% } else { %>
        <p>Нет доступных файлов.</p>
        <% } %>
    </div>
</div>
    <script src="/js/jrcode.js"></script>
</body>
</html>

```

Файл login.ejs

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Страница входа</title>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <link rel="stylesheet" href="/css/style.css" />
    <link
      rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/
bootstrap.min.css"
      integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
"
      crossorigin="anonymous"
    />
  </head>
  <body>
    <div class="container">
      <div class="container">
        <div class="row mt-5">
          <div class="col-7 login-left"></div>
          <div class="col-5 mt-2">
            <div>
              <% if(errors != "") { %> <% errors.forEach((item)
=> { %>
                <div class="alert alert-danger"><%- item %></div>
                <% }); %> <% } %>
            </div>
            <form method="POST" action="/login">
              <div class="form-group">
                <label for="email">Email-адрес:</label>
                <input
                  type="email"
                  class="form-control"
                  id="email"
                  name="email"

```

```

        placeholder="Введите email"
        required
    />
</div>
<div class="form-group">
    <label for="password">Пароль:</label>
    <input
        type="password"
        class="form-control"
        id="password"
        name="password"
        placeholder="Пароль"
        required
    />
</div>
<button type="submit" class="button_design">Войти<
/button>

</form>

<hr />
<div>
    Если у вас нет аккаунта.
    <a href="/register">Создать новый аккаунт</a>.
</div>
<div>Вернуться на <a href="/">главную страницу</a></
div>

    <br /><br />
</div>
</div>
</div>
</div>
</body>
</html>

```

Файл register.ejs

```

<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Страница регистрации</title>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
        <meta http-equiv="X-UA-Compatible" content="ie=edge" />
        <link rel="stylesheet" href="/css/style.css" />
        <link
            rel="stylesheet"
            href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/
bootstrap.min.css"
            integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
"

```

```

        crossorigin="anonymous"
    />
</head>
<body>
    <div class="container">
        <div class="container">
            <div class="row mt-5">
                <div class="col-7 login-left"></div>
                <div class="col-5 mt-2">
                    <div>
                        <% if(errors != "") { %> <% errors.forEach((item)
=> { %>

                            <div class="alert alert-danger"><%- item %></div>
                            <% }); %> <% } %>
                        </div>
                        <form method="POST" action="/register">
                            <div class="form-group">
                                <label for="fullName">Имя:</label>
                                <input
                                    type="text"
                                    class="form-control"
                                    name="fullName"
                                    id="fullName"
                                    placeholder="Имя"
                                />
                            </div>
                            <div class="form-group">
                                <label for="email">Email-адрес: (*)</label>
                                <input
                                    type="email"
                                    name="email"
                                    class="form-control"
                                    id="email"
                                    placeholder="Введите email"
                                    required
                                />
                            </div>
                            <div class="form-group">
                                <label for="password">Пароль: (*)</label>
                                <input
                                    type="password"
                                    name="password"
                                    class="form-control"
                                    id="password"
                                    placeholder="Пароль"
                                    required
                                />
                            </div>
                            <div class="form-group">
                                <label for="passwordConfirmation">Повторите паро
ль: (*)</label>
                                <input

```

```

        type="password"
        class="form-control"
        name="passwordConfirmation"
        id="passwordConfirmation"
        placeholder="Подтверждение пароля"
        required
    />
</div>
<button type="submit" class="button_design">
    Зарегистрироваться
</button>
</form>

<hr />
<div>Если у вас уже есть аккаунт. <a href="/login">В
ойти</a>.</div>
<div>Вернуться на <a href="/">главную страницу</a></
div>
    <br /><br />
</div>
</div>
</div>
</div>
</body>
</html>

```

					ПГУ1.090301.04.001 ПЗ	Лист
						200
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(Обязательное)

Стартап-проект. Веб-приложение улучшения качества видео

Презентация

					ПГУ1.090301.04.001 ПЗ	Лист
						201
Изм.	Лист	№ докум.	Подпись	Дата		

Стартап-проект. Веб-приложение улучшения качества видео

Автор работы:

студент гр. 20ВВП1 Верховский Михаил Владимирович

Научный руководитель:

к.т.н., доцент Федюнин Роман Николаевич

Пенза, 2024

Цель работы и объекты

Цель работы – разработать программное обеспечение, выполняющее роль посредника между клиентом и конвейером обработки графического контента.

Объект разработки – веб-приложение, предназначенное для выбора и загрузки видео/изображений в конвейер обработки, а также получения результата в виде нового видеофайла или изображения с улучшенными характеристиками. Веб-приложение обладает пользовательским интерфейсом, спроектировано с применением клиент-серверной архитектуры и состоит из трёх связанных между собой частей: клиентская часть, серверная часть и база данных.

					ПГУ1.090301.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		202

Требования к проекту

Главная задача этого веб-приложения – предоставлять пользователям возможность загружать свои изображения и видео, а также самостоятельно задавать все необходимые параметры (разрешение, частота кадров (для видео) и выбор модели нейронной сети) для улучшения качества этих файлов. Выбранные параметры обработки должны сохраняться в виде JSON-файлов, которые в дальнейшем должны использоваться для улучшения контента с помощью конвейера обработки. Должна быть возможность скачивания обработанных файлов. Для этого также должно быть реализовано отслеживание состояния обработки на сервере. Также веб-приложение должно:

- иметь галерею обработанных ранее изображений/видео пользователем. В галерее должны быть фильтры, возможность скачивания и удаления файла;
- иметь связь с БД, в которой хранятся данные для работы приложения;
- иметь функционал, позволяющий пользователю сравнить видео/изображение до и после обработки;
- иметь функционал аутентификации и авторизации пользователей.

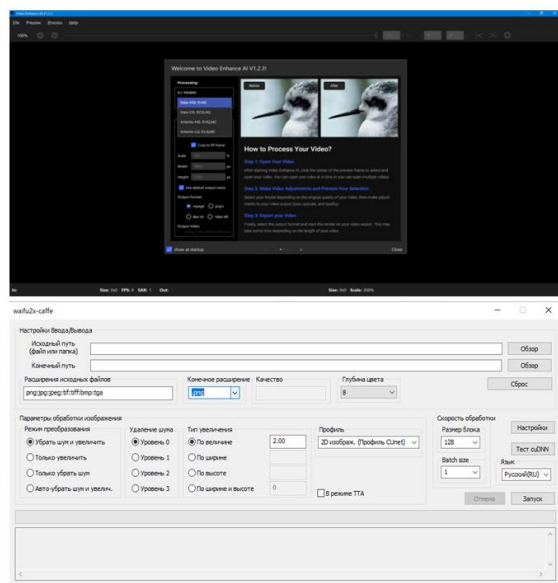
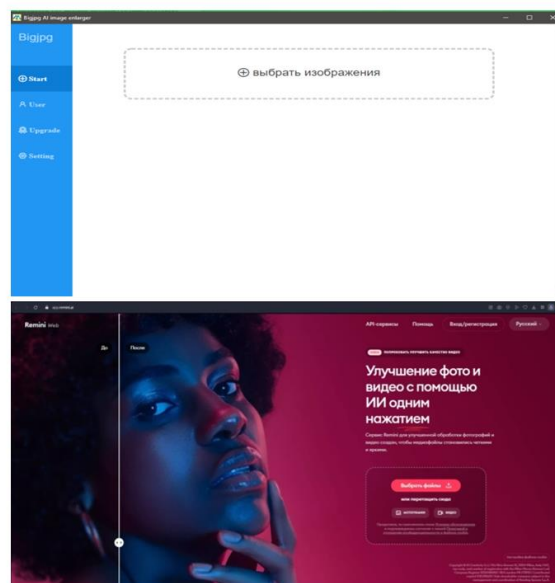
Актуальность

Задача улучшения качества видеофайлов и изображений является актуальной по разным причинам:

- растущий спрос на высококачественный визуальный контент;
- прогресс в технологиях;
- распространение мобильных устройств;
- профессиональные и коммерческие нужды. Профессионалы в различных отраслях, таких как:

- 1) фотосъёмка и видеосъёмка;
- 2) агентства по маркетингу и рекламе;
- 3) интернет-магазины;
- 4) медицинские и научные учреждения;
- 5) следственное дело;
- 6) личные и социальные потребности.

Аналоги



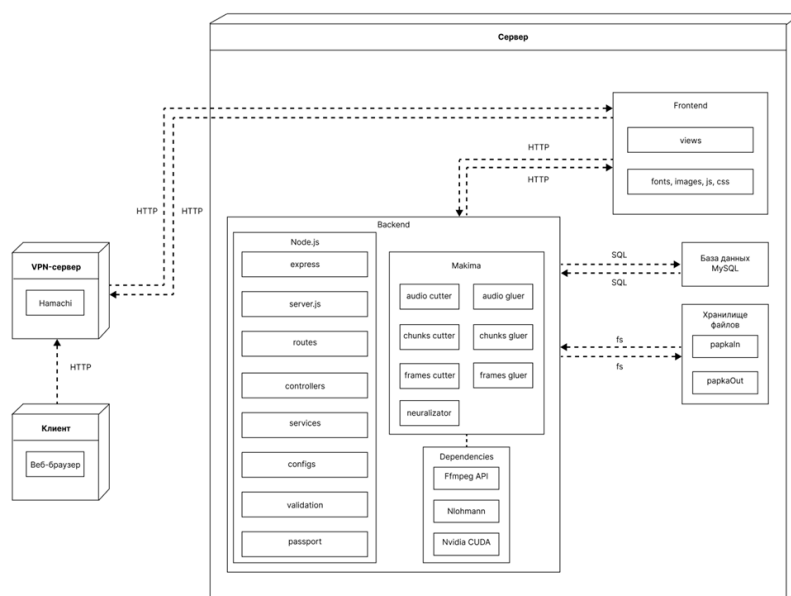
Сравнение аналогов

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

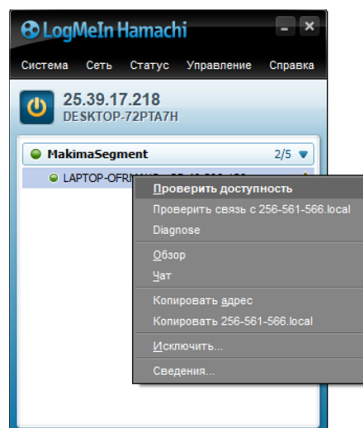
Использованные технологии



Архитектура проекта



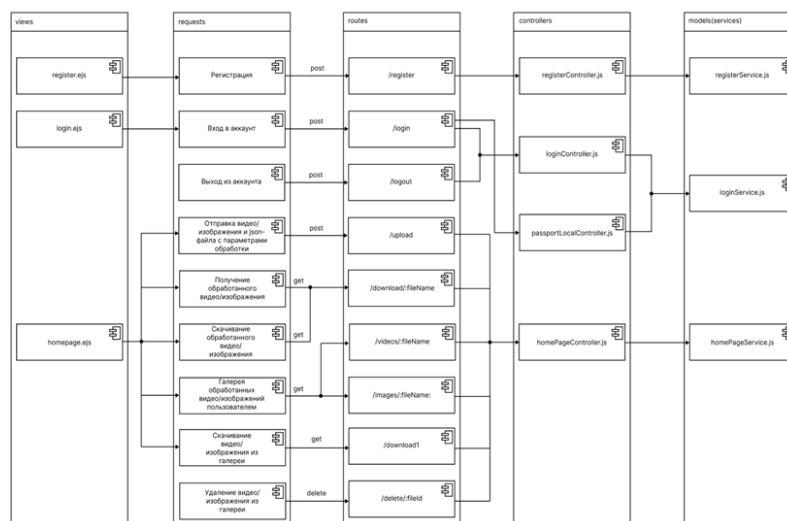
VPN и веб-сервер



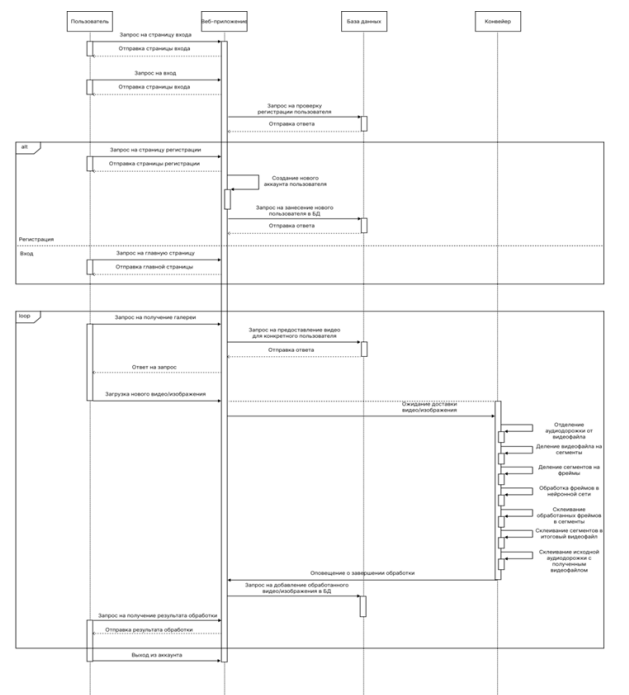
Express JS



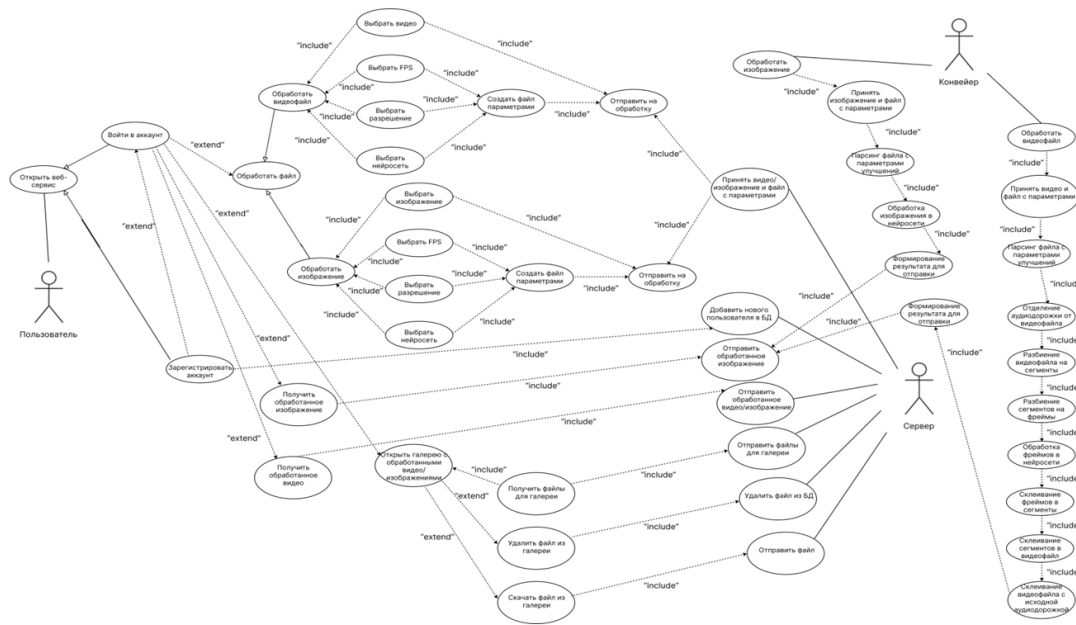
UML-диаграмма компонентов



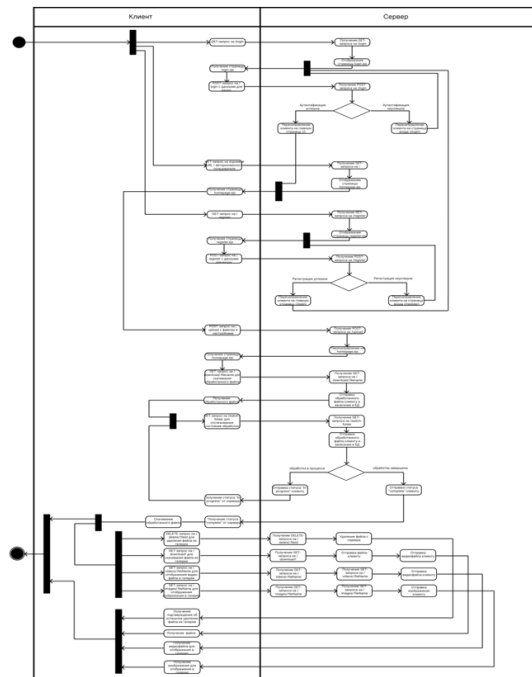
UML-диаграмма последовательности



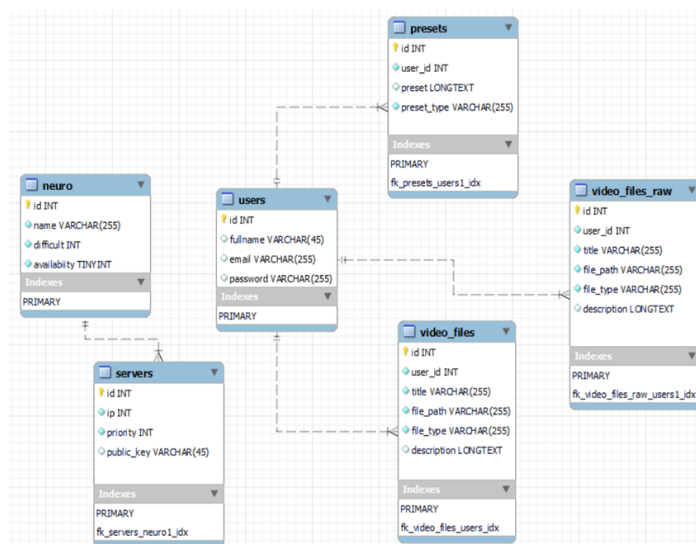
UML-диаграмма вариантов использования



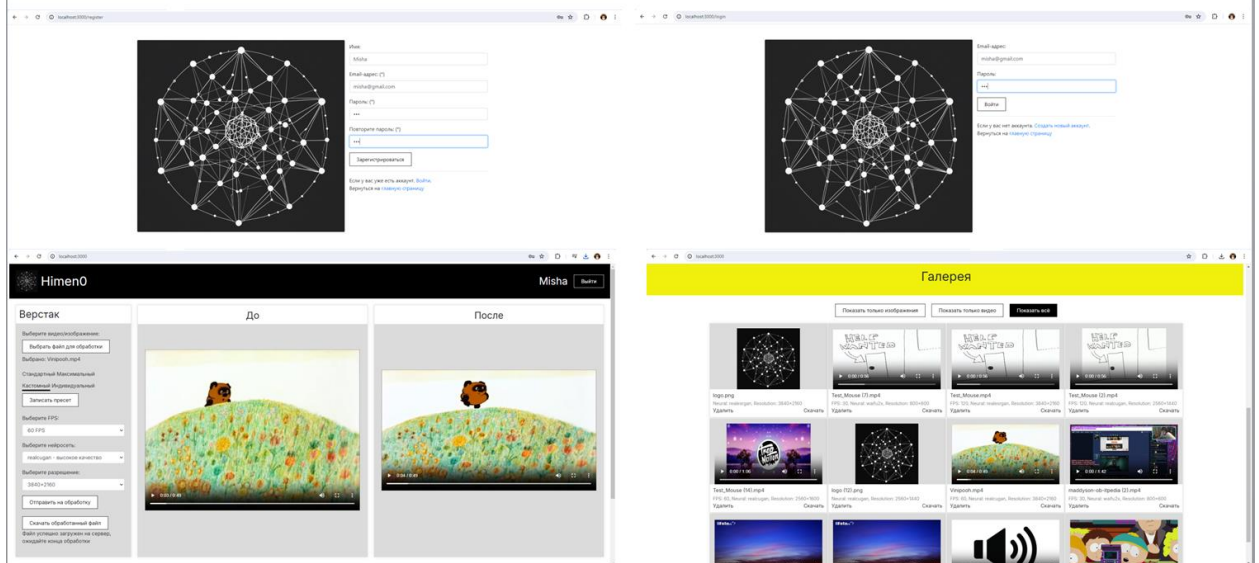
UML-диаграмма деятельности



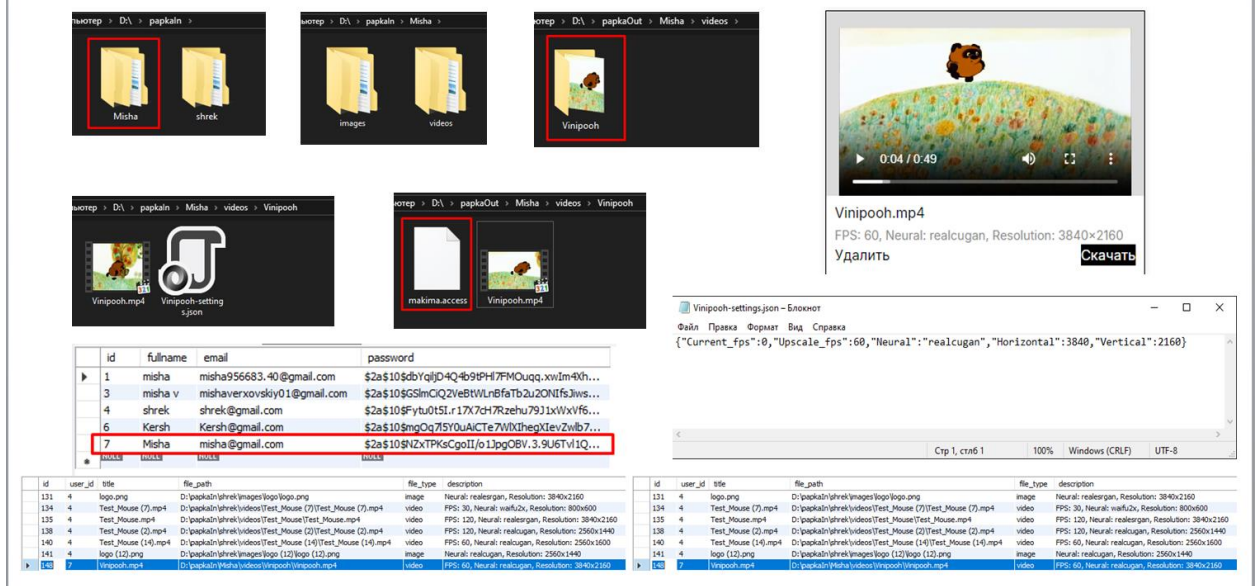
Структура базы данных



Тестирование веб-приложения



Тестирование веб-приложения



Результаты работы

В результате проведенного исследования предметной области и анализа существующих решений было разработано веб-приложение для улучшения качества видео и изображений. Приложение позволяет пользователям загружать файлы, выбирать различные параметры обработки (разрешение, нейронная сеть, FPS), а затем скачивать обработанные файлы.

Были спроектированы и реализованы основные страницы веб-приложения, включая страницы регистрации, входа, главную страницу с загрузкой и обработкой файлов, а также галерею ранее обработанных изображений и видео. Была спроектирована и разработана база данных для хранения информации.

Получившееся веб-приложение было протестировано в локальной сети с использованием VPN [Hamachi](#), что подтвердило его работоспособность и готовность к дальнейшему развитию и масштабированию.

Спасибо за внимание!

					ПГУ1.090301.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		210

МИНОБРНАУКИ 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 Верховский Михаил Владимирович

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 «Веб-приложение улучшения качества видео». Веб-приложение улучшения качества видео является веб-приложением, позволяющим взаимодействовать с конвейером нейросетевой обработки, повышающим качество графического контента.

Тема одобрена экспертной комиссией и утверждена приказом ректора от «19» апреля 2024 г. № 405/0.

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 веб-приложение улучшения качества видео. Составить бизнес-модель и финансовый план. Язык программирования JavaScript, среда разработки Visual Studio Code. Программа должна обладать интуитивно понятным интерфейсом и использовать следующие технологии:

1. JavaScript, HTML, CSS

2. Node.js

3. Express.js

4. MySQL

5. Frontend, Backend

6. JSON

7. Сетевое программирование

4.2. Содержание пояснительной записки ВКР (по разделам):

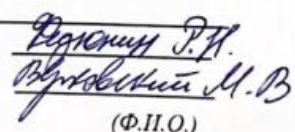
Анализ предметной области, обоснование актуальности разработки, обзор аналогов, анализ требований к разработке, постановка задачи, выбор инструментов разработки, архитектура приложения, проектирование и разработка, тестирование и отладка, БЖД, бизнес-план: финансовые и временные ограничения на проект (бюджет, сроки реализации, ожидаемые затраты и доходы и т.д.).

5. Перечень графического материала: UML-диаграммы (диаграмма компонентов, диаграмма развертывания, диаграмма деятельности, диаграмма вариантов использования, диаграмма последовательности)

Руководитель ВКР

Задание принял к исполнению


(подпись, дата)


(Ф.И.О.)

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.04.001 ПЗ

Лист

211

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. Веб-приложение улучшения качества видео” Верховского Михаила Владимировича подтверждает его актуальность, практическую значимость и возможность коммерциализации.

Решение экспертной комиссии: выпускная квалификационная работа на тему “Стартап-проект. Веб-приложение улучшения качества видео” Верховского Михаила Владимировича 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

ПГУ1.090301.04.001 ПЗ

Лист

212