

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

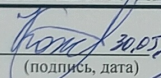
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

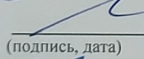
на тему

**СТАРТАП-ПРОЕКТ «CLView». «СЕРВИС ДЛЯ
УЛУЧШЕНИЯ КАЧЕСТВА ИЗОБРАЖЕНИЙ
С ИСПОЛЬЗОВАНИЕМ НЕЙРОННОЙ СЕТИ»**

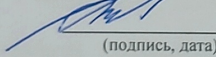
Студент

 30.05.24 Коженикова Алина Вадимовна
(подпись, дата) (ФИО полностью)

Руководитель

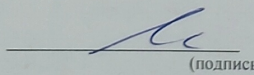
 30.05.24 М. А. Митрохин
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

 30.05.2024 А.С. Бычков
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.24 г. № 14)

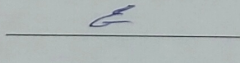
Заведующий кафедрой

 (подпись) М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

04.06.2024 № 11)

Секретарь ГЭК

 (подпись) Е.И. Гурин
(фамилия, инициалы)

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 » 04 20 24 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Кожевниковой Алины Вадимовны гр. 20ВВП2 факультета ВТ

2. Тема работы Стартап-проект «CLView». «Сервис для улучшения качества изображений с использованием нейронной сети»

Тема утверждена приказом ПГУ № 405/0 от " 19 " 04 20 24 г.

3. Руководитель бакалаврской работы Митрохин Максим Александрович

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать сервис для улучшения качества изображений с использованием нейронной сети. Составить бизнес-модель и финансовый план. Языки программирования python и flutter, среда разработки Android Studio и PyCharm. Веб-приложение должно обладать интуитивно понятным интерфейсом и использовать следующие технологии:

1. Flutter

2. PyTorch

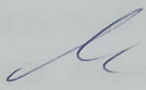
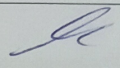
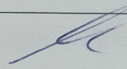
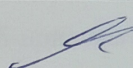
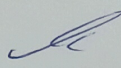
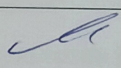
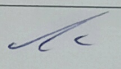
3. Асинхронное программирование

4. FastAPI

5. Перечень подлежащих разработке вопросов

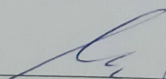
Требуется произвести анализ предметной области: изучить существующие модели нейронных сетей, выделить их достоинства и недостатки, а также изучить аналоги в области информационных систем, выделить их достоинства и недостатки. Разработать и обучить нейронные сети. Выбрать языки для UI и back-end частей, среду разработки, СУБД, облачное хранилище, а также облачный сервер. Определить архитектуру приложения и

6. Календарный график выполнения работы

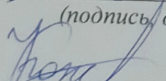
№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Исследование физических процессов снижения качества изображений при воздействии тумана, задымленности	46	19.04.2024 – 21.04.2024	
2	Создание наборов данных для обучения нейронных сетей	40	22.04.2024 – 25.04.2024	
3	Выбор критериев качества изображений	30	26.04.2024 – 27.04.2024	
4	Исследование архитектур нейронных сетей, используемых для повышения качества изображений	56	28.04.2024 – 04.05.2024	
5	Разработка и обучение архитектур нейронной сети для фильтрации задымлений и «замыленности», а также осветления тёмных участков изображений	80	05.05.2024 – 21.05.2024	
6	Разработка UI-дизайн веб-приложения	32	10.05.2024 – 20.05.2024	
7	Разработка back-end части для веб-приложения	40	22.05.2024 – 27.05.2024	

Дата выдачи задания "19" "04" 2024 г.

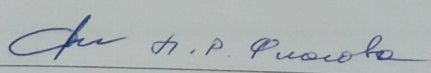
Руководитель бакалаврской работы


(подпись, дата)

Задание к исполнению принял студент


(подпись, дата)

Работу к защите допустить
Декан факультета


03.06.2024 (подпись, дата)

Реферат

Бакалаврская работа содержит 111 листов, 23 рисунка, 9 формул, 10 источников, 3 приложения, 7 таблиц.

ГЛУБОКИЕ НЕЙРОННЫЕ СЕТИ, ОСВЕТЛЕНИЕ ИЗОБРАЖЕНИЙ, ФИЛЬТРАЦИЯ ЗАТУМАНЕННЫХ ИЗОБРАЖЕНИЙ, УЛУЧШЕНИЕ КАЧЕСТВА ИЗОБРАЖЕНИЙ, ТУМАН, ЗАТЕМНЕНИЯ, PYTHON, FLUTTER, MINIO, БАЗА ДАННЫХ, НЕЙРОННЫЕ СЕТИ, АВТОЭНКОДЕР, ANDROID STUDIO

Цель работы – разработать информационный сервис, позволяющий осветлять тёмные и фильтровать туманные изображения с применением нейронной сети.

Один из путей достижения поставленной задачи является веб-приложение, выполняющее следующие функции: фильтрация затуманенных изображений, осветление тёмных изображений, авторизация пользователей, скачивание обработанных изображений.

В результате выполнения данной выпускной квалификационной работы, разработан сервис улучшения качества изображений с использованием нейронной сети, который представлен в виде веб-приложения. Сервис полностью соответствует поставленной цели. Для достижения результатов были использованы следующие технологии: python, fastapi, minIO, PostgreSQL, uvicorn, flutter.

					ПГУ1.090301.30.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата					
Разраб.		Кожеевникова А.В.		30.05.24	Стартап-проект CLView. Сервис для улучшения качества изображения с использованием нейронной сети	Лит.	Лист	Листов	
Провер.		Митрохин М.А.		30.05.24				3	111
Н. Контр		Бычков А.С.		30.05.24		ФВТ, каф. ВТ			
				30.05.24		гр. 20ВВП2			
Утверд.		Митрохин М.А.		30.05.24					

**МИНОБРНАУКИ РОССИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Кафедра "Вычислительная техника"

УТВЕРЖДАЮ
Заведующий кафедрой

Митрохин М.А.

(Подпись) (Фамилия, инициалы)
« 19 » 04 2024 г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ В ФОРМЕ СТАРТАПА

Студент Кожевникова Алина Вадимовна

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 «CLView». «Сервис для улучшения качества изображений с использованием нейронной сети», которая будет выполняться данным членом команды.

Тема одобрена экспертной комиссией и утверждена приказом ректора от « 19 » 04 2024 г. № 405/0.

2. Срок представления работы к защите « 4 » 06 2024 г.

3. Общая часть ВКР:

Подготовка компьютерной презентации бизнес-проекта (стартапа).

4. Индивидуальная часть ВКР:

4.1. Исходные данные:

Разработать сервис для улучшения качества изображений с использованием нейронной сети. Составить бизнес-модель и финансовый план. Языки программирования python и flutter, среда разработки Android Studio и PyCharm. Веб-приложение должно обладать интуитивно понятным интерфейсом и использовать следующие технологии:

1. Flutter

2. PyTorch

3. Асинхронное программирование

4. FastAPI

4.2. Содержание пояснительной записки ВКР (по разделам):

Анализ предметной области, обоснование актуальности разработки, обзор аналогов, анализ требований к разработке, постановка задачи, выбор инструментов разработки, архитектура приложения, проектирование и разработка, тестирование и отладка, БЖД, бизнес-план: финансовые и временные ограничения на проект (бюджет, сроки реализации, ожидаемые затраты и доходы и т.д.).

5. Перечень графического материала: UML-диаграммы (диаграмма классов, диаграмма развертывания, диаграмма деятельности, диаграмма вариантов использования, диаграмма последовательности)

Руководитель ВКР

Задание принял к исполнению

(подпись, дата)

(Ф.И.О.)

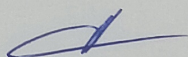
Экспертное заключение
о согласовании темы ВКР в форме стартапа

Экспертная комиссия в составе декан ФВТ д.т.н., профессор Л.Р. Фионова, доцент кафедры ВТ, к.э.н., И.В. Акифьев, ведущий системный архитектор ООО «Ай-теко ВЦ», к.т.н. М.П. Синев, рассмотрев бизнес-проект Стартап-проект "CLView". Сервис для улучшения качества изображений с использованием нейронной сети, Кожевниковой Алины Вадимовны подтверждает его актуальность, практическую значимость и возможность коммерциализации.

Решение экспертной комиссии: выпускная квалификационная работа на тему Стартап-проект "CLView". Сервис для улучшения качества изображений с использованием нейронной сети Кожевниковой Алины Вадимовны может выполняться в форме стартапа (бизнес-проекта).

Председатель ЭК

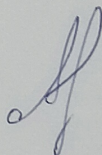
д.т.н., профессор
декан ФВТ



Л.Р. Фионова

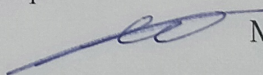
Члены ЭК

к.э.н., доцент кафедры ВТ



И.В. Акифьев

к.т.н., ведущий системный архитектор
ООО «Ай-теко ВЦ»



М.П. Синев

Содержание

Реферат	3
Перечень сокращений и обозначений.....	6
Введение.....	7
1 Стартап-проект сервис для улучшения качества изображений с использованием нейронной сети.	9
1.1 Анализ актуальности разработки	9
1.2 Создание базы данных изображений	10
1.3 Выбор метрики качества изображений.....	13
1.4 Исследование архитектур нейронных сетей, используемых для повышения качества изображений.....	14
1.4.1 Исследование архитектур нейронных сетей, фильтрующих “замыленные” и затуманенные изображения	14
1.4.2 Исследование архитектур нейронных сетей, осветляющих тёмные изображения	18
1.5 Разработка архитектуры нейронной сети для фильтрации задымлений и «замыленности», а также осветления тёмных участков изображений..	23
1.5.1 Разработка модели для фильтрации задымлений и «замыленности»	23
1.5.2 Разработка модели для осветления тёмных участков изображений	24
1.6 Обучение и тестирование двух моделей нейронной сети на наборе изображений	25
1.7 Анализ аналогов информационных средств	28
1.8 Разработка архитектуры информационного сервиса	28
1.9 Постановка задачи	30
1.10 Разработка архитектур нейронных сетей	30
1.10.1 Разработка модели для фильтрации задымлений и «замыленности»	30

1.10.2 Разработка модели для осветления тёмных участков изображений	32
1.11 Разработка веб-приложения.....	33
1.11.1 Работа с БД	33
1.11.2 Работа с системой хранения объектов.....	34
1.11.3 Описание back-end части веб-приложения	34
1.11.4 Описание функционала веб-приложения	37
2 Безопасность жизнедеятельности.....	47
2.1 Правила безопасности при использовании ПК.....	47
3 Стартап проект	51
3.1 Сегмент ЦА.....	51
3.2 Ключевые процессы.....	52
3.3 Каналы взаимодействия	54
3.4 Источники дохода	55
3.5 Ключевые ресурсы.....	56
3.6 Риски проекта	59
Заключение	61
Список использованных источников	63
Приложение А – UML диаграммы	65
Приложение Б – Листинг.....	72
Приложение В – Презентация	102

Перечень сокращений и обозначений

В настоящей выпускной квалификационной работе применяются следующие сокращения и обозначения

БД (база данных) — это организованная коллекция данных, обычно хранящихся в электронном формате на компьютере или другом устройстве. Она представляет собой структурированное хранилище информации, доступ к которой обеспечивается с помощью специальных программных средств, таких как системы управления базами данных

СУБД (система управления базами данных) — это программное обеспечение, предназначенное для создания, управления, и организации баз данных

НС (нейронная сеть) – это метод в искусственном интеллекте, который учит компьютеры обрабатывать данные таким же способом, как и человеческий мозг.

ПК (персональный компьютер) — это компьютер, предназначенный для использования одним человеком (или небольшой группой пользователей) в рамках личного, домашнего или офисного использования

API (Application Programming Inteface) – это интерфейс для взаимодействия между различными программными компонентами. Он определяет набор функций, процедур, методов или URL, которые один программный компонент предоставляет для использования другим компонентам

REST (Representational State Transfer) - это архитектурный подход, который устанавливает ограничения для API: как они должны быть устроены и какие функции поддерживать. Это позволяет стандартизировать работу программных интерфейсов, сделать их более удобными и производительными

IDE (Integrated development environment) - это программа, которой пользуются разработчики, когда пишут код. IDE похожа на редактор, но предоставляет больше функций, чем просто написание или изменение кода. IDE объединяют набор инструментов для редактирования, тестирования и отладки программного кода в одном интерфейсе, поэтому и называются интегрированными

Введение

Изображения всегда играли важную роль в жизни человека. Их создают с самыми разными целями: от развлекательных до мер предосторожности. Получить качественное изображение, которое будет наиболее близко передавать цвет и детализацию объектов сцены – одна из самых важных задач. Однако, существуют явления, мешающие созданию качественных фотографий. К таким можно отнести:

- плохое освещение - затемняет сцену и изображения, находящиеся в ней;
- осадки (дождь, снег, град) – капли, возникающие из-за данной группы явлений, могут искажать изображение вызывая смещение объектов и потерю фокуса;
- ветер – может вызвать размытие изображений из-за вибраций камеры;
- туман или дым – делают объекты менее чёткими, снижая контрастность и цветопередачу сцены.

Методы улучшения качества изображений делят на две группы:

- основанные на алгоритмах;
- основанные на технологиях глубоких нейронных сетей.

В последнее время методы, основанные на глубоком обучении, получили огромное распространение, поскольку было показано, что они превосходят классические алгоритмические способы обработки изображений. Нейронные сети позволяют решать задачи улучшения изображений более эффективно благодаря своей способности извлекать и анализировать сложные непрерывные закономерности в данных. В отличие от традиционных подходов, которые часто требуют явного определения правил и параметров обработки, нейронные сети способны автоматически извлекать признаки из изображений и адаптироваться к разнообразным структурам и шаблонам. Это позволяет им достигать высокой точности при обработке изображений, несмотря на их сложность и изменчивость.

Сервис для улучшения качества изображений с применением нейронной сети, который рассмотрен в данной квалификационной работе, представляет собой веб-приложение, способное отфильтровать затуманенные или замыленные изображения, а также их осветлить. Он направлен на устранение влияния вышеупомянутых факторов на фотографии, повышая общее качество изображений и удовлетворяя потребности пользователей в получении четких и выразительных фотографий даже в условиях неблагоприятной окружающей среды.

Цели выпускной квалификационной работы:

- исследовать физических процессов снижения качества изображений при воздействии тумана, задымленности;
- создать базы данных с «замыленными» и тёмными изображениями;
- выбрать критерии качества изображения;
- исследовать архитектуры нейронных сетей, используемых для повышения качества изображений;
- разработать архитектуры нейронной сети для фильтрации задымлений и «замыленности», а также осветления тёмных участков изображений;
- обучить и протестировать две модели нейронной сети на наборе изображений;
- разработать UI-дизайн веб-приложения, способного фильтровать задымления и «замыленности», а также осветлять тёмные участки на изображениях ,загруженных пользователем;
- разработать back-end часть для веб-приложения;
- протестировать программный модуль и устранить ошибки.

1 Стартап-проект сервис для улучшения качества изображений с использованием нейронной сети.

1.1 Анализ актуальности разработки

В современном мире использование изображений становится все более важным и распространенным в различных сферах человеческой деятельности. Однако часто изображения страдают от различных видов деградации, таких как туманы и затемнения, что значительно снижает их качество и визуальную привлекательность. Такие эффекты приводят к потере деталей, контраста и четкости изображений, что осложняет их анализ, обработку и интерпретацию человеком или автоматизированными системами.

С развитием технологий компьютерного зрения и глубокого обучения стало возможным эффективно бороться с такими негативными воздействиями на изображения. Применение нейронных сетей для улучшения качества изображений получило широкое распространение и активно развивается в современной науке и промышленности. Нейросетевые методы позволяют эффективно восстанавливать потерянные детали и улучшать контрастность, визуальную четкость и цветовую гамму изображений, несмотря на наличие атмосферных и других искажающих факторов.

Сервисы по улучшению качества изображений с использованием нейронных сетей представляют значимый научный и практический интерес. Они находят применение в различных областях человеческой деятельности и компьютерного зрения.

Безопасность и видеонаблюдение: В сфере видеонаблюдения, особенно при низком освещении или в условиях плохой видимости, сервисы по улучшению качества изображений позволяют более точно распознавать лица, транспортные средства и другие объекты на видеозаписях.

Автономное вождение: В разработке автономных транспортных средств качество обработки изображений из камер и датчиков играет решающую роль в обеспечении безопасности и эффективности дорожного движения.

Фотографии и видеомонтаж: Получение чёткой и яркой картинки является важной задачи для фотографов.

Все эти области жизни и сферы компьютерного зрения требуют эффективных методов обработки изображений для улучшения качества визуальной информации. Разработка сервиса улучшения изображений с применением нейронной сети является актуальной и перспективной задачей, способной значительно улучшить эффективность и точность анализа визуальных данных в различных областях человеческой деятельности.

1.2 Создание базы данных изображений

Сформированы две базы данных: одна для модели, осуществляющей фильтрацию “замыленных” изображений; вторая – для осветления.

База данных для модели, осуществляющей фильтрацию изображений создана в результате объединения нескольких датасетов:

- I-HAZE: a dehazing benchmark with real hazy and haze-free indoor images (35 пар изображений);
- O-HAZE: a dehazing benchmark with real hazy and haze-free outdoor images (45 пар изображений);
- Dense-Haze CVPR 2019 (130 пар изображений);
- SMOKE (Real Dense Non-Uniform Fog) (122 пары изображений)
- FogCityscape (4500 пар изображений)

Пары получены синтетическим способом, т.е. при создании затуманенных изображений была использована дым-машина. Датасет состоит из пар, содержащих туман разной плотности. Это сделано для увеличения

эффективности работы модели, поскольку туман и задымления бывают разной плотности.

Каждая фотография содержит цветовую диаграмму Макбета (Macbeth Color Checker) для проверки соответствия итоговых цветов сцены с входными. Поскольку размеры изображений различались, было решено привести их к одной ширине и высоте 512*512 пикселей.

На рисунке 1.1 представлены входные данные, поступающие на вход модели из полученного в рамках НИР датасета.



Рисунок 1.1 – Примерные пары изображений из базы данных

Для модели, осуществляющей осветление изображений датасет создан из представленных ниже баз данных:

- LOLDataset (500 пар изображений);
- LOLv2-real (800 пар изображений);
- LOLv2-synthetic (1000 пар изображений).

Также, на основе свободно распространяющих в интернете изображений, была создана и добавлена в датасет своя база данных, содержащая 1000 пар изображений. Пары были получены искусственно, т.е. на изначально светлые изображения были наложены затемняющие фильтры.

Поскольку каждая пара имело своё разрешение, было принято решение изменить размер изображений до 512*512 пикселей.

На рисунке 1.2 представлены входные данные, поступающие на вход модели из полученного в рамках НИР датасета.

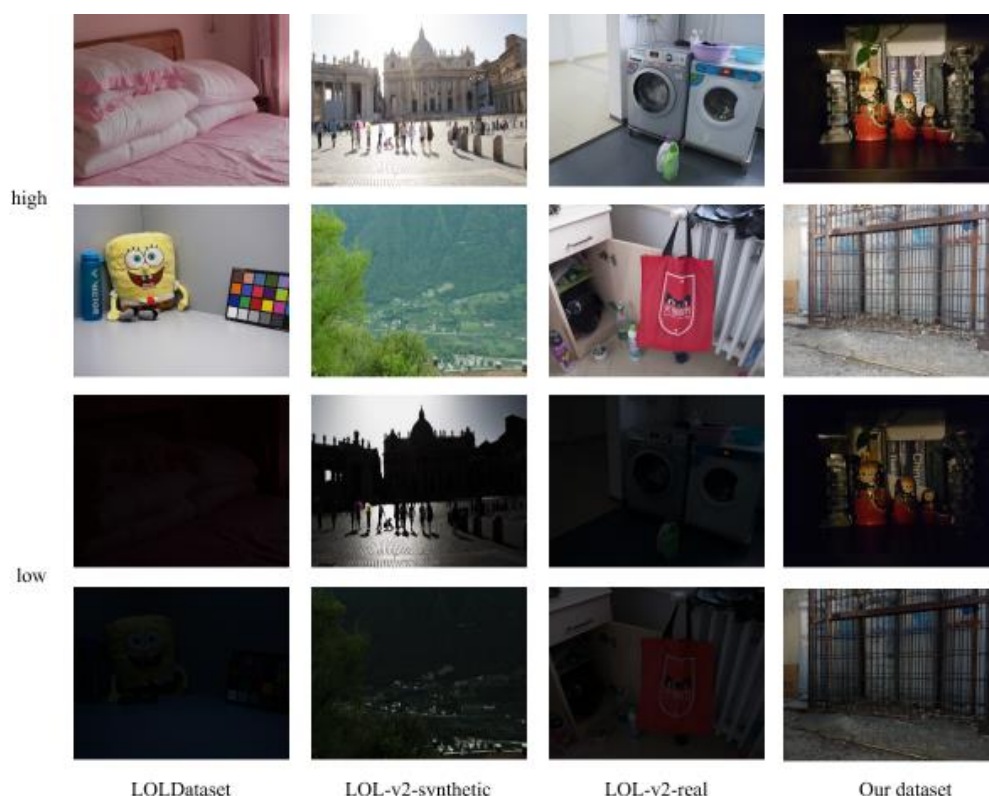


Рисунок 1.2 - Примерные пары изображений из базы данных

Разработанные базы данных могут быть использованы исследователями в качестве обучающей и тестовой выборки, необходимых для эффективного улучшения качества изображений.

1.3 Выбор метрики качества изображений

Критерии, по которым оценивают изображения можно разделить на две группы: количественные и субъективные. Первая группа – оценивает с помощью использования математических методов, а вторая – на основе экспертной оценки. Несмотря на то, что экспертная оценка чаще всего наиболее точно оценивает качество восстановленных изображений, количественная может наиболее быстро справиться с оценкой большого количества данных, как в разработанной в проекте базе данных.

Среди количественных оценок для задач восстановления изображений используются метрики пикового отношения сигнала к шуму (*PSNR*) и индекса структурного сходства (*SSIM*).

PSNR определяет уровень искажений при сжатии и включает подсчет среднеквадратичной ошибки (*MSE*). Диапазон принимаемых значений от 0 до 100, измеряется в логарифмической шкале в децибелах и представляет собой числовое значение, которое количественно определяет сходство между входными и выходными изображениями.

Дано два изображения f и g , оба имеют разрешение $M \times N$, *PSNR* между f и g вычисляется по следующей формуле:

$$PSNR(f, g) = 10 \log_{10} \left(\frac{255^2}{MSE(f, g)} \right), \quad (1)$$

$$MSE(f, g) = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (f_{ij} - g_{ij})^2, \quad (2)$$

Чем выше значение *PSNR*, тем выше качество итогового изображения.

SSIM коррелирует с качеством и восприятием зрительной системы человека (цветовая модель *HVS*). Вместо использования традиционных методов суммирования ошибок *SSIM* моделирует искажение изображения как

комбинацию трех факторов: потеря корреляции, искажение яркости и искажение контрастности. Данная метрика вычисляется по формуле:

$$SSIM(f, g) = l(f, g)c(f, g)s(f, g), \quad (3)$$

$$\begin{cases} l(f, g) = \frac{2\mu_f\mu_g + c_1}{\mu_f^2 + \mu_g^2 + c_1} \\ c(f, g) = \frac{2\sigma_f\sigma_g + c_2}{\sigma_f^2 + \sigma_g^2 + c_2} \\ s(f, g) = \frac{\sigma_{fg} + c_3}{\sigma_f\sigma_g + c_3} \end{cases} \quad (4)$$

Первая часть формулы (4) – это функция, которая сравнивает яркость двух изображений. Она максимальна и равна 1 только когда $\mu_f = \mu_g$. Вторая часть сравнивает контрастность f и g . Измеряется отклонением σ_f и σ_g и принимает максимальное значение, когда $\sigma_f = \sigma_g$. Третья функция измеряет коэффициент корреляции между двумя изображениями.

В статьях [4] и [5] авторы, соотнеся объективные оценки со значениями $PSNR$ и $SSIM$ вывели оптимальные величины данных количественных показателей. Для того, чтобы восстановленные изображения были приятны человеческому глазу $PSNR$, должно быть больше 24, а $SSIM$ – больше 0.8.

1.4 Исследование архитектур нейронных сетей, используемых для повышения качества изображений

1.4.1 Исследование архитектур нейронных сетей, фильтрующих “замыленные” и затуманенные изображения

В настоящее время существуют множество методов поляризации изображений, на которых присутствуют туманы и задымления, достигшие значительных успехов в решении этой задачи. Они могут быть разделены на две группы: традиционные и основанные на технологиях глубоких нейронных

сетей (DNN). Традиционные методы фильтрации чаще всего основаны на эмпирических посылках и опираются на методы преобразования яркости изображений или применение пространственной и частотной фильтрации. Методы, основанные на DNN, чаще всего используют свёрточные нейронные сети или их комбинации с другими подходами.

Enhanced Pix2pix Dehazing Network.

Среди генеративно-сопоставительных моделей, созданных для решения задачи фильтрации затуманенных и задымлённых изображений, выделяют данную модель. Она представляет собой генеративно-сопоставительную модель. Модель состоит: из двух генераторов G1 и G2, которые включают слои свёртки; двух дискриминаторов D1 и D2 и блока усиления, извлекающего карту признаков. Во время работы данная архитектура не полагается на карту пропускания света и атмосферное освещение. На рисунке 1.3 представлена архитектура модели Enhanced Pix2pix Dehazing Network.

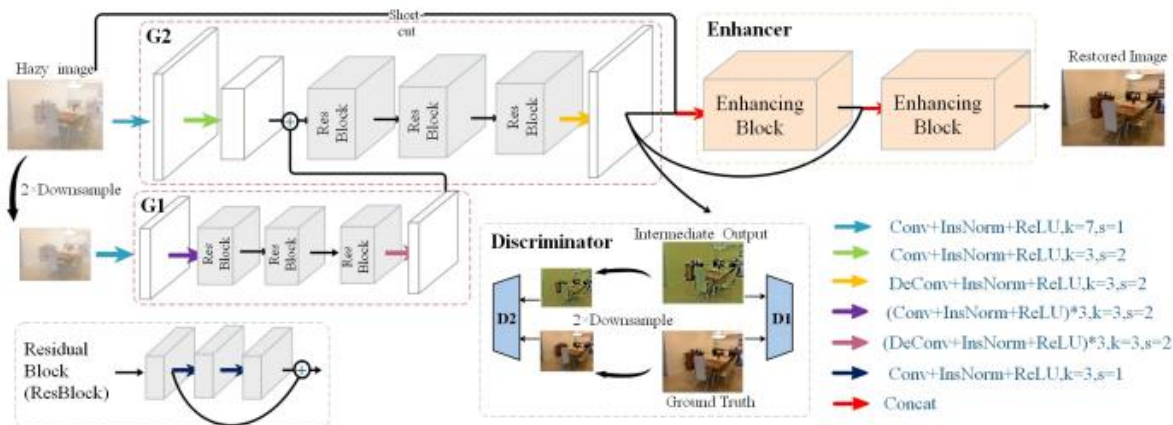


Рисунок 1.3 – Архитектура EDPN

В отличие от обычного GAN, данная модель использует подход conditional Generative Adversarial Network, который, основываясь на дополнительных параметрах, обеспечивает более гибкий контроль обучения. Недостатком модели является возможность появления некоторых артефактов и предметов, которых не было до этого.

На рисунке 1.4 представлены результаты работы модели EDPN.

Выходные
данные



Эталонные
данные



Рисунок 1.4— Результат работы EDPN

LCA-Net: Light Convolutional Autoencoder for Image Dehazing

Данная модель представляет собой свёрточный автоэнкодер. Состоит из трёх слоёв свёртки и активационного слоя ReLU. Denselayers добавили нелинейности и, с помощью них, была добавлена математическая функция, помогая эффективно извлекать важные признаки.

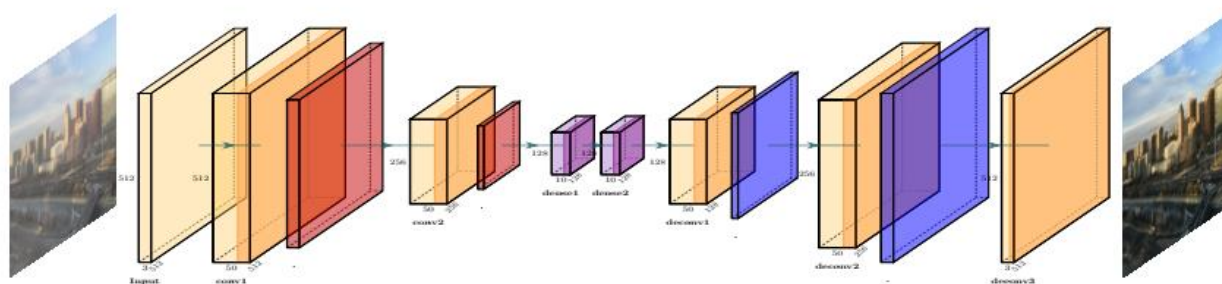


Рисунок 1.5 – Архитектура LCA-Net

Свёрточный автоэнкодер требует меньше памяти, по сравнению с прошлой моделью. Также выходные данные модели намного лучше, чем у прошлой.

Выходные
дающие



Эталонные
данные



Рисунок1.6 – Результат работы LCA-Net

ImageDehazingTransformer

Статья [6] рассматривает возможности нейронных сетей типа “трансформер” для решения задачи удаления дымки с изображения. В результате эффективной интеграции функций трансформера и сверточной нейронной сети авторы улучшают производительность операций удаления дымки.

В таблице 1 отображены количественных показатели метрик PSNR и SSIM трёх рассмотренных архитектур.

Таблица 1 - Количественные оценки результатов работы различных моделей

Модели	PSNR	SSIM
EDPN	23,34	0.336
LCA-Net	27,87	0.763
Image Dehazing Transformer	27,75	0,781

Несмотря на то, что и свёрточный автоэнкодер и трансформер показали хорошие результаты, авторы статьи [7] доказали, что в автоэнкодеры лучше в задачах восстановления, так как обучаются быстрее и лучше масштабируются

под длинные последовательности. Именно поэтому для разработки модели, фильтрующей затуманенные и задымлённые изображения будет использован свёрточный автоэнкодер.

1.4.2 Исследование архитектур нейронных сетей, осветляющих тёмные изображения

В последнее время методы, основанные на глубоком обучении, получили огромное распространение, поскольку было показано, что они превосходят другие современные инструменты машинного обучения во многих приложениях компьютерного зрения, включая распознавание объектов, понимание сцены и обнаружение ошибок. Большое количество моделей было предложено для решения проблемы осветления изображений. Их можно условно разделить на две группы: основанные на гистограммном методе и ретинекс (Retinex) методе [8].

Технология Ретинекс основана на устранении локальной неоднородности освещенности объектов наблюдаемой сцены и работает по принципу локального увеличения контраста. Слабо освещенные объекты при этом становятся лучше различимыми, восстанавливаются их детали и цвет. Эта технология предназначена в первую очередь для обработки изображений, содержащих малоконтрастные (темные либо светлые) зоны, в которых локальные детали обычно трудноразличимы [8].

Retinex-Net.

Данная нейронная сеть работает по теории Retinex о человеческом восприятии света.

Коэффициент отражения описывает внутреннее свойство объектов полученной сцены, которое считается неизменным при любых условиях освещенности. Степень освещённости отражает различную яркость объектов. На изображениях, полученных при слабом освещении, освещённость обычно страдает от темноты и несбалансированного распределения света.

Метод ретинекса разделяется на два подвида: одноканальный (SSR) и многоканальный (MSR). Первый можно представить как разницу между отдельно взятым пикселем изображения и среднее значение пикселей, окружающих данный пиксель. Для вычисления окружающего значения можно использовать любой фильтр высоких частот, например Гауссово распределение. На основе всего выше сказанного получим данное уравнение:

$$SSR_i(x, y) = \log(I_i(x, y)) - \log(G_\sigma * I_i)(x, y), \quad (5)$$

где $(G_\sigma * I_i)(x, y)$ – размытие по Гауссу.

Retinex-Net основан на SSR-методе и объединяет декомпозицию изображений и операции последовательного улучшения. Она состоит из двух подсетей: Decom-Net, которая используется для разделения наблюдаемого изображения на независимое от освещения отражение и плавное освещение с учетом структуры. Данная подсеть изучается с двумя ограничениями. Во-первых, изображения при слабом и нормальном освещении имеют один и тот же коэффициент отражения. Во-вторых, карта освещенности должна быть гладкой, но сохранять основные структуры, что достигается за счет полной потери вариации с учетом структуры. Затем, вторая подсеть Enhance-Net настраивает карту освещенности для поддержания согласованности в больших регионах, одновременно адаптируя локальные распределения путем многомасштабной конкатенации.

Поскольку шум часто более выражен в темных областях и даже усиливается в процессе уменьшения светлых областей, вводится снижение коэффициента отражения. Для обучения такой сети был создан набор данных из пар изображений при слабом и нормальном освещении на основе реальной фотографии и синтетических изображений [8]. Результат работы модели представлен на рисунке 1.7.

Эталонные
данные



Выходные
данные



Рисунок 1.7 – Результат работы Retinex-Net

cycleGAN.

Данная модель состоит из двух частей: дискриминатора и генератора. Генератор должен создавать изображения, максимально похожие на реальные данные из обучающей выборки. Цель дискриминатора – отличить созданное генератором изображение от реального. Данный тип архитектуры изначально не был ориентирован на улучшение изображений, однако этого можно достичь путём обучения на соответствующем наборе данных.

Результат работы модели после обучения представлены на рисунке 1.8.

The Low-Light Net (LLNet).

Многослойные шумоподавляющие автоэнкодеры (SSDA) – вид автоэнкодеров, способных удалять шум с входных данных. Ранние исследования показали, что, объединяя несколько автоэнкодеров с шумоподавлением (DA) для предварительного обучения, сеть способна найти лучшее пространство параметров во время обратного распространения ошибки.

Эталонные
данные



Выходные
данные



Рисунок 1.8 – Результат работы cycleGAN

Модель Low-LightNet или LLNet была основана на SSDA. При разработке были использованы возможности SSDA по снижению уровня шума и возможности комплексного моделирования глубоких сетей для изучения особенностей, лежащих в основе изображений, полученных при слабом освещении, и создания улучшенных версий с минимальным уровнем шума и улучшенной контрастностью.

При обучении модели, на вход подаётся две пары изображений затемнённые и чистые. Данные проходят сквозь три шумоподавляющих автоэнкодера. Затем участки размером 17×17 пикселей затемняются нелинейно для настройки гаммы-коррекции. Гамма-коррекция — это простой, но общий случай применения степенной формулы к изображениям

для улучшения по пикселям с

$$I_{out} = A * I_{in}^{\gamma}, \quad (6)$$

где A - константа, определяемая максимальной интенсивностью пикселя на изображении;

I_{in}^{γ} – входные данные;

I_{out} – полученные на выходе.

Изображение становится ярче, когда $\gamma < 1$, в то время как $\gamma = 1$ оставляет его неизменным. Следовательно, когда $\gamma > 1$, отображение взвешивается в сторону более низких (тёмных) значений интенсивности пикселей в оттенках серого [8]. Результат работы модели представлен на рисунке 1.9.

Эталонные
данные



Выходные
данные



Рисунок 1.9 – Результат работы LLNet

В таблице 2 отображены количественных показатели метрик PSNR и SSIM трёх рассмотренных архитектур.

Таблица 2 - Количественные оценки с различных

Модели	PSNR	SSIM
Retinex-Net	22,18	0.71
cycleGAN	16.21	0.59
LLNet	26,45	0,791

Наилучшие показатели были у LLNet, поэтому для разработки модели, осветляющей тёмные участки изображений, будет использован свёрточный автоэнкодер.

1.5 Разработка архитектуры нейронной сети для фильтрации задымлений и «замыленности», а также осветления тёмных участков изображений

1.5.1 Разработка модели для фильтрации задымлений и «замыленности»

Для решения проблемы фильтрации туманов и дыма предлагается использовать свёрточный автоэнкодер. По сравнению с классическими, свёрточные автоэнкодеры, больше подходят для обработки изображений. Именно данную архитектуру предлагается использовать в качестве объекта эксперимента.

Модель состоит из энкодера и декодера, каждый имеет четыре полносвязных свёрточных слоя с функцией активации ReLU. Для ускорения обучения и получения более чёткого изображения на выходе с минимальным количеством эпох, была использована пакетная нормализация как для энкодера, так и для декодера. Архитектура разработанной модели представлена на рисунке 1.10.

В качестве функции потерь была выбрана среднеквадратичная ошибка (mean squared error), формула которой представлена ниже:

$$\text{MSE}_{\text{AE}} = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y}_i)^2, \quad (7)$$

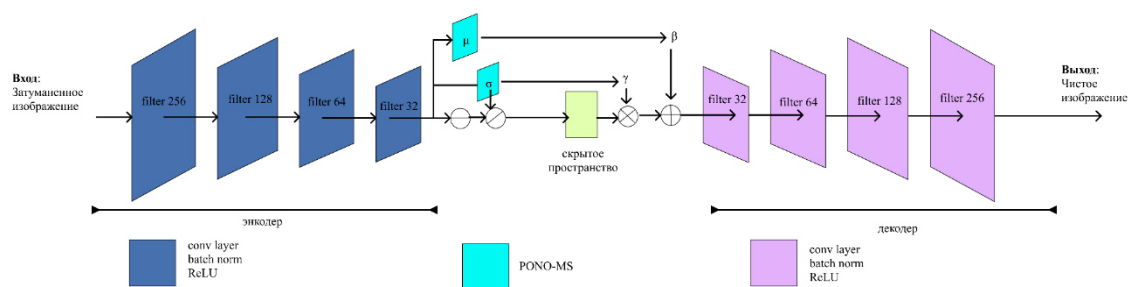


Рисунок 1.10 – Архитектура свёрточного автоэнкодера, фильтрующего изображения

Для повышения качества выходных изображений после свёрточных слоёв энкодера и до слоёв декодера, был добавлен PONO-MS [9]. Статистика PONO показывает, что среднее значение μ и стандартное отклонение σ действительно могут отражать структурную информацию изображения и извлеченных объектов, аналогично тому, как статистика, вычисляемая IN, имеет тенденцию отражать аспекты стиля входного изображения. PONO улавливает существенные структурные признаки входных изображений, благодаря которым можно использовать извлеченные моменты для передачи этой информации из источника в целевое изображение. Это открывает возможность выйти за рамки автоэнкодеров и использовать PONO-MS в настройках преобразования image-to-image.

PONO-MS позволил увеличить PSNR с 28.75 до 29.35 и SSIM с 0.833 до 0.914.

1.5.2 Разработка модели для осветления тёмных участков изображений

Для решения проблемы осветления тёмных участков изображений было принято взять свёрточный автоэнкодер.

Модель состоит из энкодера и декодера, каждый имеет четыре полносвязных свёрточных слоя с функцией активации ReLU. Для ускорения обучения и получения более чёткого изображения на выходе с минимальным

количеством эпох, была использована пакетная нормализация как для энкодера, так и для декодера. Архитектура разработанной модели представлена на рисунке 1.11.

Предсказанное автоэнкодером изображение подаётся на вход Multi-scale retinex with Color Restoration [10] представленной в виде свёрточного слоя с тремя цветовыми каналами на выходе.

В качестве функции потерь была выбрана среднеквадратичная ошибка (mean squared error)(7)

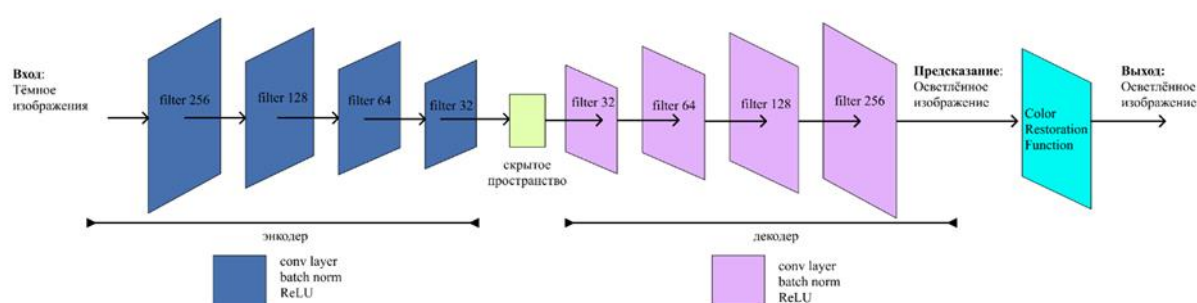


Рисунок 1.11 - Архитектура свёрточного автоэнкодера, осветляющего изображения

1.6 Обучение и тестирование двух моделей нейронной сети на наборе изображений

Для обучения и тестирования моделей две базы данных были разделены так, что 30% датасета тестовые и 70% тренировочные. Такое разделение необходимо для того, чтобы оценить, насколько хорошо работает разработанная модель.

Если данные не разделять, то произойдёт переобучение и модель не сможет выполнять свою работу наиболее эффективно на реальных данных.

Обучение проводилось на персональном компьютере с процессором Ryzen 9 5950x и двумя графическими картами Nvidia RTX3090. Для увеличения скорости обучения каждая модель параллельно обучалась на двух видеокартах. Для модели, осуществляющей осветление изображений в качестве алгоритма численной оптимизации, использовался Adam (adaptive moment estimation) с начальной скоростью сходимости (learning rate) 0.004. Размер батча составлял 8.

Для модели, созданной для фильтрации задымлённых и «замыленных» изображений скорость сходимости была равна 0.002. Размер батча составлял 8.

Тренировка обученных моделей проводилась на ноутбуке с процессором Intel Core i7 и с одной графической картой Nvidia GeForce GTX 1650 Max-Q.

На рисунках 1.12 и 1.13 отображены результаты работ обученных моделей на тренировочном наборе данных.

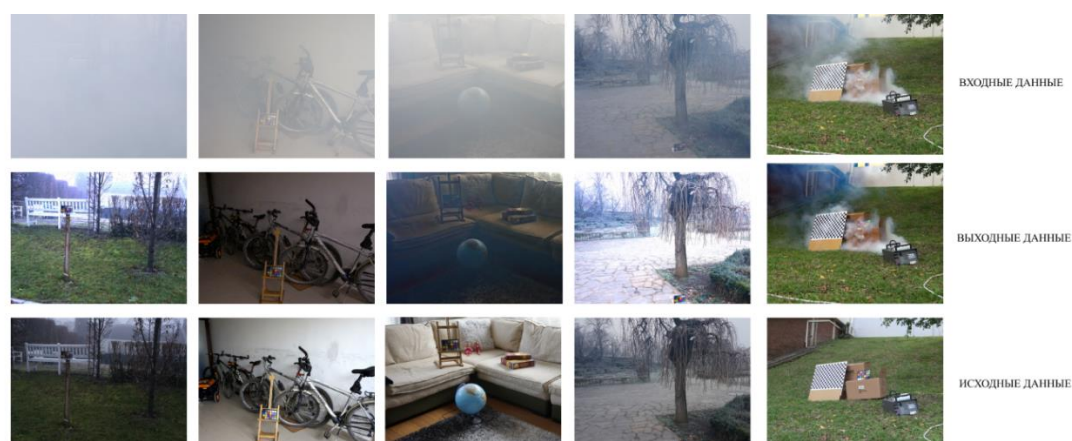


Рисунок 1.12 – Результат работы фильтрующей модели на тренировочной выборке

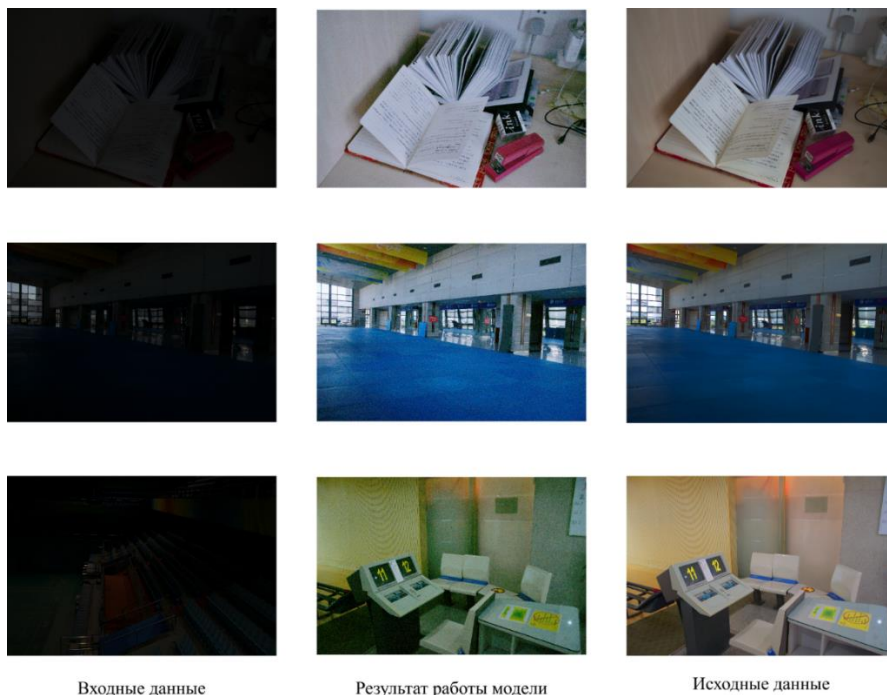


Рисунок 1.13 – Результат работы осветляющей модели на тренировочной выборке

Для того чтобы оценить, насколько хорошо нейронные сети справились со своими задачами, с помощью выбранных ранее метрик было проведено сравнение исходных изображений («чистых» или «светлых») с восстановленными. В таблице 3 приведены результаты количественных оценок на тренировочном наборе данных.

Таблица 3 – Количественная оценка восстановленных изображений

Модель	PSNR	SSIM
Фильтрация туманов и «задымлений»	29.35	0.914
Освещение тёмных участков	24.03	0.891

1.7 Анализ аналогов информационных средств

Среди растровых графических редакторов было выявлено два крупных аналога:

- Adobe Photoshop, несомненно, является ведущим редактором в области растровой графики. С помощью комбинации некоторых настроек может позволить осветлять тёмные и фильтровать “замыленные” участки с низкой плотностью;

- Affinity Photo, с другой стороны, тоже представляет собой мощный инструмент для работы с изображениями. В отличие от Photoshop, он обладает способностью фильтровать замыленные области и осветлять тёмные участки, однако при этом изменяет цветовые характеристики при удалении мутной дымки с участков высокой плотности.

1.8 Разработка архитектуры информационного сервиса

Архитектура информационного сервиса состоит из трёх основных компонентов: клиентской части, серверной части и базы данных.

Веб-приложение имеет две части: интуитивно понятный пользовательский интерфейс и бэкенд часть. Приложение даёт возможность осветлить и “отфильтровать” “замыленные” или затемнённые изображения с помощью одной из двух нейронных сетей. Подробную диаграмму вариантов пользования можно изучить на рисунке 1.14.

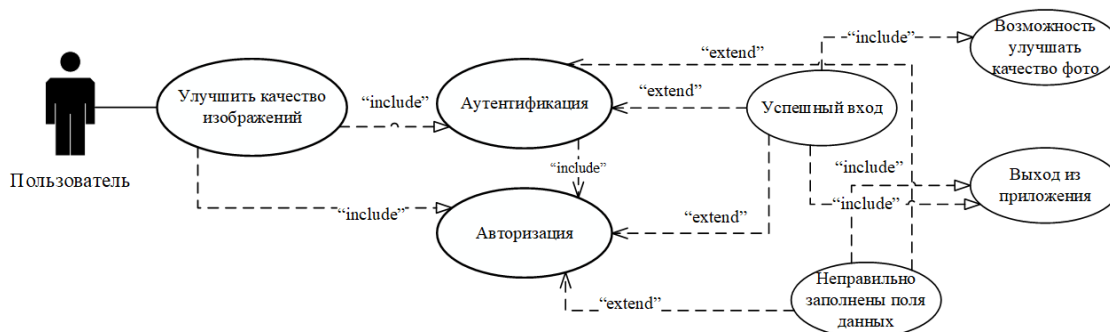


Рисунок 1.14 – UML-диаграмма вариантов пользования

Для чёткого понимания взаимодействия компонентов веб-приложения, необходимо создать UML-диаграмму последовательности. На рисунке 1.15 представлена диаграмма последовательности.

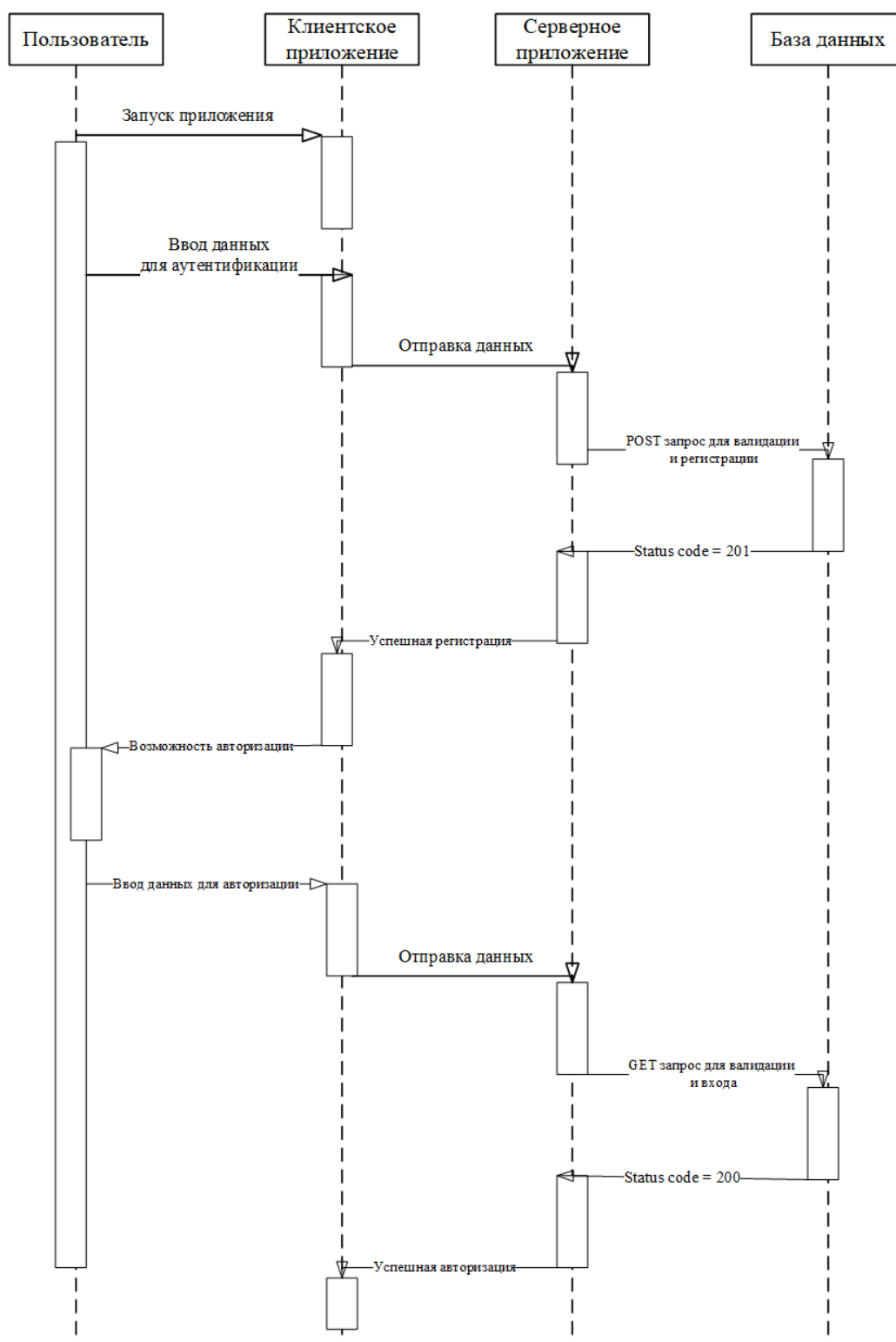


Рисунок 1.15 – UML-диаграмма последовательности

1.9 Постановка задачи

Необходимо разработать информационные сервис улучшения качества изображений с применением нейронной сети. Веб-приложение должно содержать клиентскую и серверную части, а также базу данных.

Серверная часть должна быть реализована на языке программирования Python и включать в себя две модели искусственных нейронных сетей. База данных - свободная объектно-реляционная система управления базами данных PostgreSQL.

Клиентская часть веб-приложения будет реализована с использованием фреймворка Flutter, который используется как для создания мобильных приложений под Android и iOS, веб-приложений, так и для приложений под Windows, macOS и Linux с использованием языка программирования Dart. Для использования веб-приложения пользователь должен будет выполнить вход в аккаунт или регистрацию в системе. Затем он должен будет выбрать изображение из файловой системы своего устройства и загрузить его в приложение.

Программный модуль должен представлять собой предварительно обученную модель нейронной сети. Он должен быть написан на языке программирования Python и библиотек OpenCV для алгоритмов компьютерного зрения, PyTorch для глубокого обучения, а также NumPy, используемого для обработки массивов.

1.10 Разработка архитектур нейронных сетей

1.10.1 Разработка модели для фильтрации задымлений и

«замыленности»

Для решения проблемы фильтрации туманов и дыма предлагается использовать свёрточный автоэнкодер. По сравнению с классическими,

свёрточные автоэнкодеры, больше подходят для обработки изображений. Именно данную архитектуру предлагается использовать в качестве объекта эксперимента.

Модель состоит из энкодера и декодера, каждый имеет четыре полносвязных свёрточных слоя с функцией активации ReLU. Для ускорения обучения и получения более чёткого изображения на выходе с минимальным количеством эпох, была использована пакетная нормализация как для энкодера, так и для декодера. Архитектура разработанной модели представлена на рисунке 1.16.

В качестве функции потерь была выбрана среднеквадратичная ошибка (mean squared error), формула которой представлена ниже:

$$\text{MSEAE} = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y}_i)^2, \quad (8)$$

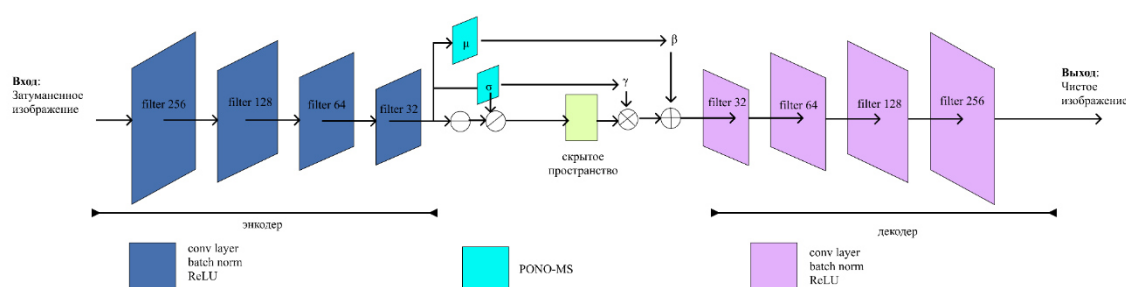


Рисунок 1.16 – Архитектура свёрточного автоэнкодера, фильтрующего изображения

Для повышения качества выходных изображений после свёрточных слоёв энкодера и до слоёв декодера, был добавлен PONO-MS [9]. Статистика PONO показывает, что среднее значение μ и стандартное отклонение σ действительно могут отражать структурную информацию изображения и извлеченных объектов, аналогично тому, как статистика, вычисляемая IN, имеет тенденцию отражать аспекты стиля входного изображения. PONO улавливает существенные структурные признаки входных изображений, благодаря

которым можно использовать извлеченные моменты для передачи этой информации из источника в целевое изображение. Это открывает возможность выйти за рамки автоэнкодеров и использовать PONO-MS в настройках преобразования image-to-image.

PONO-MS позволил увеличить PSNR с 28.75 до 29.35 и SSIM с 0.833 до 0.914.

1.10.2 Разработка модели для осветления тёмных участков изображений

Для решения проблемы осветления тёмных участков изображений было принято взять свёрточный автоэнкодер.

Модель состоит из энкодера и декодера, каждый имеет четыре полносвязных свёрточных слоя с функцией активации ReLU. Для ускорения обучения и получения более чёткого изображения на выходе с минимальным количеством эпох, была использована пакетная нормализация как для энкодера, так и для декодера. Архитектура разработанной модели представлена на рисунке 5.

Предсказанное автоэнкодером изображение подаётся на вход Multi-scale retinex with Color Restoration [10] представленной в виде свёрточного слоя с тремя цветовыми каналами на выходе.

В качестве функции потерь была выбрана среднеквадратичная ошибка (mean squared error)(7). На рисунке 1.17 представлена архитектура разработанной модели.

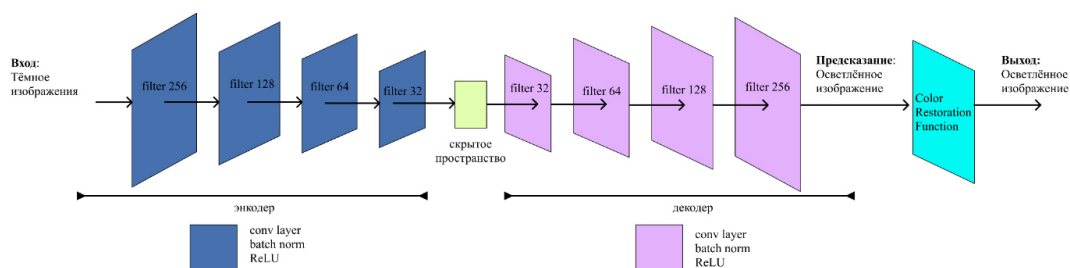


Рисунок 1.17 - Архитектура свёрточного автоэнкодера, осветляющего изображения

1.11 Разработка веб-приложения

1.11.1 Работа с БД

Для хранения информации была реализована база данных со содержащая следующие таблицы:

- user_data – таблица, содержащая информацию о зарегистрированных пользователях;
- image – таблица, содержащая информацию об обработанных нейронными сетями изображениях, которые загрузил авторизованный пользователь;
- image_class – таблица, содержащая названия классов (осветлённое или отфильтрованное) изображений;
- codes – таблица, содержащая сгенерированные токены для сброса пароля авторизованным пользователем.

Для наглядного понимания архитектуры БД, на рисунке 1.18 представлена схема физической модели базы данных.

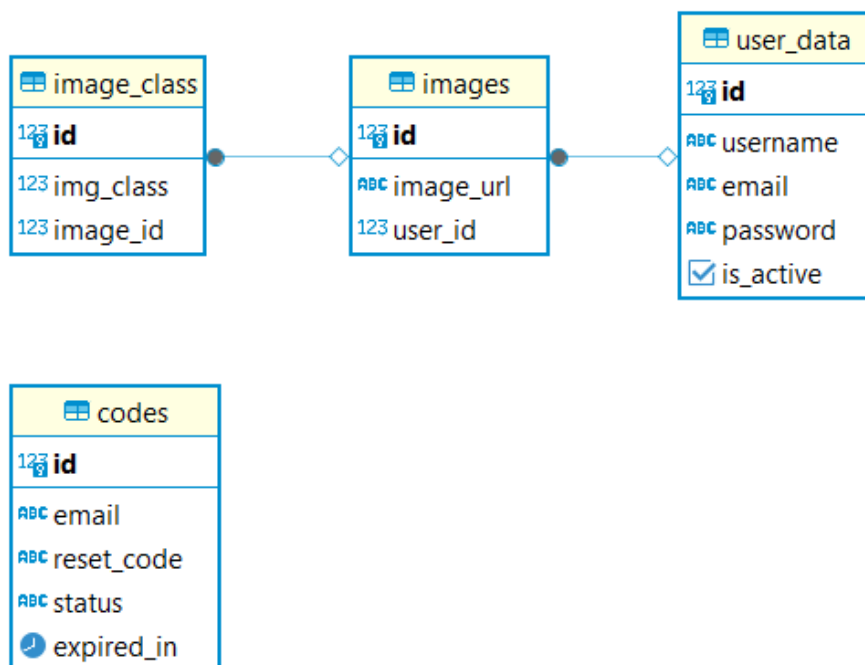


Рисунок 1.18 – Физическая модель базы данных

1.11.2 Работа с системой хранения объектов

Для хранения и обработки пользовательских изображений была выбрана система хранения объектов MinIO.

Она была выбрана по нескольким причинам, во-первых, данная система полностью совместима с Amazon S3 API, что позволяет легко интегрировать её с существующими инструментами и библиотеками, упрощая процесс работы с объектным хранилищем.

Кроме того, MinIO обеспечивает высокую производительность, позволяя хранить и обрабатывать большие объёмы изображений, обеспечивая быстрый доступ к данным.

Развертывание и управление MinIO также являются довольно простыми задачами. MinIO прост в развёртывании на локальных и удалённых серверах, таких как Docker и Kubernetes.

Кроме того, MinIO позволяет обезопасить данных. Данная система поддерживает шифрование данных как на уровне хранилища, так и при их передаче. Это обеспечивает надёжную защиту данных, что особенно важно при работе с персональными данными.

Было создано два бакета:

- clview-orig-bucket – бакет для хранения исходных изображений авторизованного пользователя;
- clview-rendered-bucket – бакет для хранения обработанных нейронными сетями изображений авторизованного пользователя.

1.11.3 Описание back-end части веб-приложения

Диаграмма развертывания, приведенная на рисунке 1.19, используется для визуализации физического размещения компонентов системы и их взаимодействия на уровне аппаратного и программного обеспечения. Она позволяет представить архитектуру системы, указывая, на каких устройствах

или серверах расположены компоненты приложения и как они связаны друг с другом.

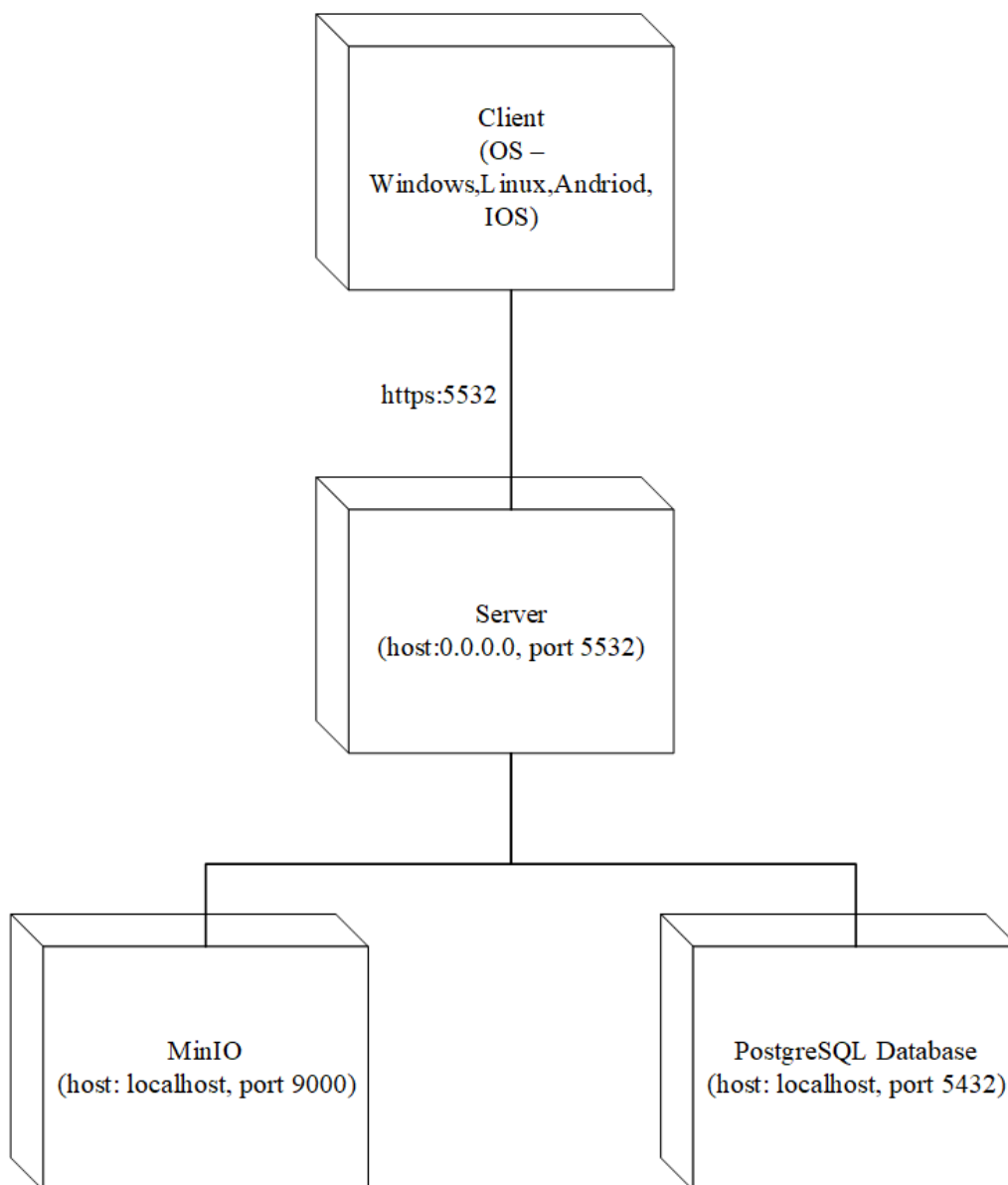


Рисунок 1.19 – UML-диаграмма развёртывания

Файл *.env* – содержит параметры базы данных и MinIO.

Модуль *database.py* – файл, в котором описывается подключение к базе данных PostgreSQL.

Модуль *crud.py* – содержит функции, которые позволяют добавлять или изменять информацию о пользователях.

Модуль crud_model.py – содержит функции, которые позволяют добавлять, обрабатывать и получать данные об изображениях.

Модуль models.py – файл, в котором прописываются поля таблиц в БД.

```
from sqlalchemy import Boolean, Column, ForeignKey, Integer, String, DateTime
```

```
from sqlalchemy.orm import relationship
from CLView.database import Base
```

```
class User(Base):
    __tablename__ = "user_data"
    #__table_args__ = {'extend_existing': True}
```

```
id = Column(Integer, primary_key=True, index=True)
username = Column(String(64), unique=True)
email = Column(String(64), unique=True)
password = Column(String(64))
is_active = Column(Boolean, default=True)
```

```
images = relationship("Images", back_populates="owner")
```

```
class Codes(Base):
    __tablename__ = "codes"
    #__table_args__ = {'extend_existing': True}
```

```
id = Column(Integer, primary_key=True, index=True)
email = Column(String(64))
reset_code = Column(String(64), unique=True)
status = Column(String(1))
expired_in = Column(DateTime)
```

```
class Images(Base):
    __tablename__ = "images"
    #__table_args__ = {'extend_existing': True}
```

```
id = Column(Integer, primary_key=True, index=True)
image_url = Column(String(100))
user_id = Column(Integer, ForeignKey("user_data.id"))
```

```
owner = relationship("User", back_populates="images")
img_class = relationship("ImageClass",
back_populates="model")
```

```
class ImageClass(Base):
    __tablename__ = "image_class"
```

```
id = Column(Integer, primary_key=True, index=True)
```

```
img_class = Column(Integer)

image_id = Column(Integer, ForeignKey("images.id"))

model = relationship("Images", back_populates="img_class")
```

Модуль `model_lightning.py` – файл, который содержит код обученной модели нейронной сети для осветления изображений.

Модуль `test_model.py` – файл, который содержит функцию осветления изображения.

Модуль `model_dehaze.py` – файл, который содержит код обученной модели нейронной сети для фильтрации затуманенных изображений.

Модуль `test_model_dehaze.py` – файл, который содержит функцию фильтрации затуманенных изображений.

Модуль `email_send.py` – файл, который содержит функцию отправления почты по указанной почте.

Модуль `image_tensor.py` – файл, который содержат функции перевода изображений с тензора или байтов.

Модуль `main.py` – файл, который отправляет REST запросы на фронт-энд часть.

1.11.4 Описание функционала веб-приложения

Главная форма состоит из двух кнопок с надписями: “Dehaze image”, “Enlight image”, расположенных в центре. На рисунке 1.20 представлен интерфейс главного экрана.

Ниже представлен листинг дизайна и работы главного экрана.

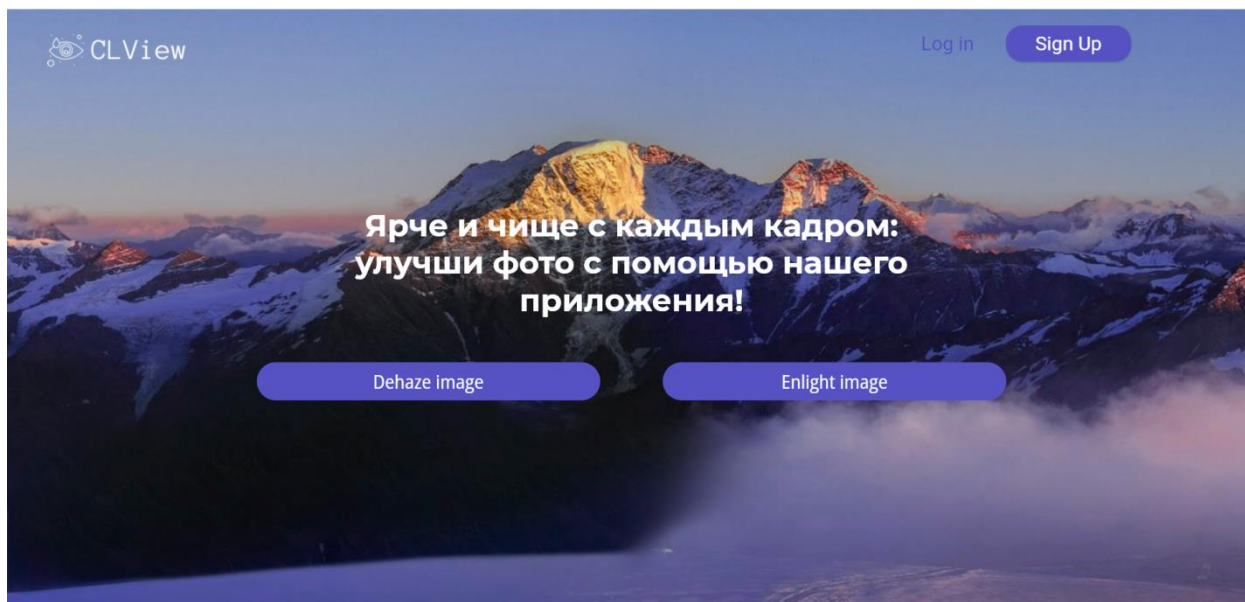


Рисунок 1.20 – Интерфейс главного экрана

```
import 'package:file_picker/file_picker.dart';
import 'package:flutter/material.dart';
import
'package:b_im_q/desktop_screens/WelcomePage/background.dart';
import 'package:b_im_q/components/RoundedButton.dart';
import 'package:auto_size_text/auto_size_text.dart';
import 'package:open_file/open_file.dart';

class Body extends StatefulWidget {
  const Body({super.key});

  @override
  State<Body>createState() => _BodyState();
}

class _BodyState extends State<Body> {
  void _pickFile() async {

    // opens storage to pick files and the picked file or files
    // are assigned into result and if no file is chosen result is
    null.
    // you can also toggle "allowMultiple" true or false depending
    on your need
    FilePickerResult? result = await
    FilePicker.platform.pickFiles(
    type: FileType.custom,
    allowedExtensions: ['jpg', 'png'],
    );
  }
}
```



```

        // if no file is picked
        if (result == null) return;

        // we get the file from result object
        final file = result.files.first;

        _openFile(file);
    }

    void _openFile(PlatformFile file) {
        OpenFile.open(file.path);
    }

    @override
    Widget build(BuildContext context) {
        //Future<Users?>? users;
        Size size = MediaQuery.of(context).size;
        //This size provide us total height and width of screen
        return Background(
            child: SingleChildScrollView(
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: <Widget>[
                        SizedBox(
                            width: 300,
                            height: 140,
                            child: AutoSizeText("Remove haze from your images or videos",
                                style: TextStyle(
                                    fontFamily: 'Montserrat',
                                    color: Colors.white,
                                    fontWeight: FontWeight.bold,
                                    fontSize: 36,
                                ),
                            ),
                        ),
                        SizedBox(height: size.height * 0.05),
                        RoundedButtonClear(
                            text: "Clear desk",
                            press: () {
                                _pickFile();
                            },
                        ),
                        RoundedButtonClear(
                            text: "Clear video",
                            press: () {
                                //
                            },
                        ),
                    ],
                ),
            ),
        ),
    ),

```

```
);
}
}
```

По нажатию пользователем на иконку, происходит переход на страницу регистрации, содержащую три текстовых поля для ввода логина, пароля и электронной почты, одну кнопку и ссылку на страницу авторизации, если пользователь уже регистрировался в системе. На рисунке 1.21 представлен интерфейс страницы регистрации.

CLView

Create Account

Login

Email

Password

Sign up

Already have an account? Log in

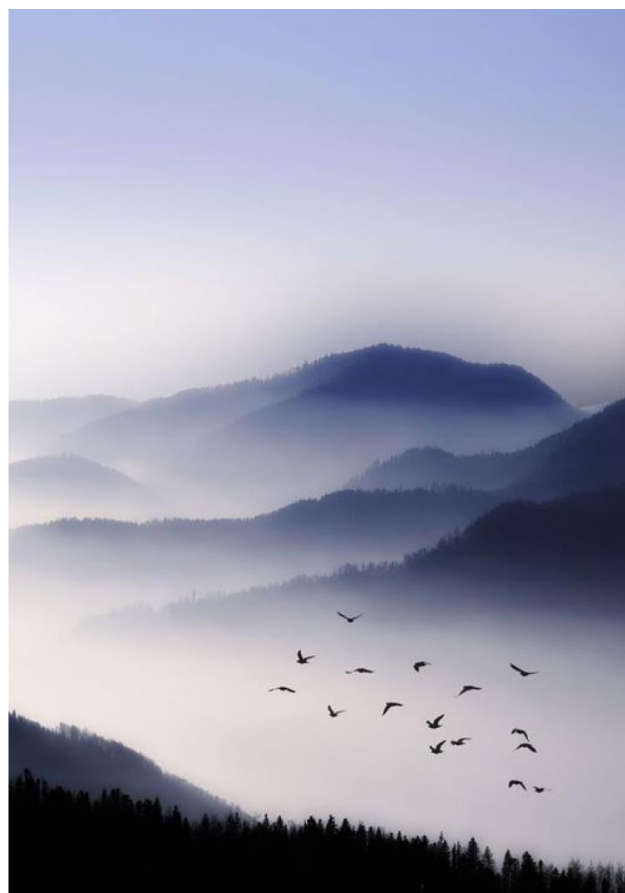


Рисунок 1.21 – Интерфейс страницы регистрации

Реализация интерфейса страницы регистрации:

```
import 'package:b_im_q/backend/CRUD.dart';
import 'package:b_im_q/components/RoundedButton.dart';
import 'package:b_im_q/components/have_an_account_check.dart';
import 'package:b_im_q/components/rounded_input_field.dart';
import 'package:b_im_q/components/rounded_password_field.dart';
```

```

import
'package:b_im_q/desktop_screens/Login/login_screen_desktop.dart';
import
'package:b_im_q/desktop_screens/signUp/components/backgroundSign.d
art';
import 'package:flutter/foundation.dart';

import 'package:flutter/material.dart';
import '../../constants.dart';
import '../../backend/users.dart';
import 'package:email_validator/email_validator.dart';

class Body extends StatefulWidget {
  const Body({
    Key? key,
  }): super(key : key);

  @override
  State<Body>createState() => _BodyState();
}

class _BodyState extends State<Body> {
  TextEditingController loginController = TextEditingController();
  TextEditingController pwdController = TextEditingController();
  TextEditingController emailController = TextEditingController();
  String _errorMessage = '';

  @override
  Widget build(BuildContext context) {
    Size size = MediaQuery.of(context).size;
    Future<Users> users;
    final DioClient _client = DioClient();

    return BackgroundSignUp(child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        SizedBox(height: size.height * 0.1),
        Text("Create Account",
          style: TextStyle(
            fontFamily: 'Montserrat',
            color: kPrimaryColor,
            fontWeight: FontWeight.bold,
            fontSize: 36,
          ),
        ),
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Text(_errorMessage, style: TextStyle(color: Colors.red)),
        ),
      ],
    ),
  )

```

```

RoundedInputField(
  hintText: "Your login",
  icon : Icons.person_2_outlined,
  controller: loginController,
),
RoundedInputField(
  hintText: "Your email",

  icon: Icons.email_outlined,

  controller: emailController),
RoundedPasswordField(controller: pwdController,),
RoundedButtonLogSign(text: "Sign up", press: () async{
  var username = loginController.text;
  var email = emailController.text;
  var password = pwdController.text;
  validateEmail(email);
  await _client.signUpUser(username: username, email: email,
  password: password).then((value){
  Navigator.push(context, MaterialPageRoute(
  builder: (context){return LoginDesktop();}
  ));
  }).catchError((err) {_errorMessage = err;});

  if(kDebugMode){
  print("username: "+ username + "\temail: "+ email + "\tpassword:
  "+ password);
  }
  }),
  HaveAnAccountCheck(login: false, press: (){
  Navigator.push(context, MaterialPageRoute(
  builder: (context){return LoginDesktop();}
  ));
  })
  ],
  ),,);
}
String validateEmail(String val) {
if(val.isEmpty){
  _errorMessage = "Email can not be empty";
}else if(!EmailValidator.validate(val, true)){
  _errorMessage = "Invalid Email Address";
}else{
  _errorMessage = "";
}
return _errorMessage;
}
}

```


При удачной регистрации или, если пользователь уже имеет аккаунт и хочет войти в него, открывается окно авторизации. Оно содержит два текстовых поля для ввода логина и пароля, кнопку для авторизации и ссылку на страницу аутентификации. На рисунке 1.22 изображён интерфейс странице регистрации

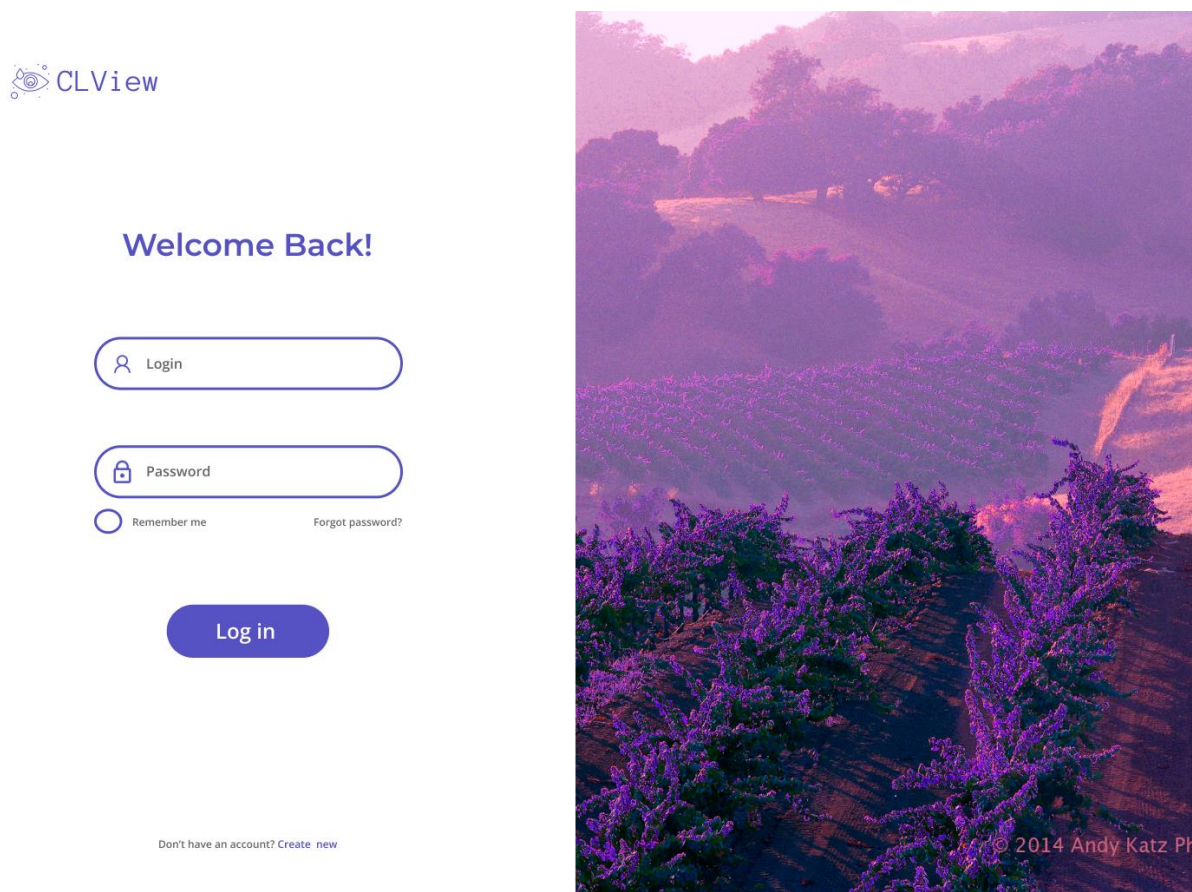


Рисунок 1.22 – Интерфейс страницы авторизации

Реализация страницы авторизации:

```
import 'package:b_im_q/backend/users.dart';
import 'package:b_im_q/components/RoundedButton.dart';
import 'package:b_im_q/components/rounded_input_field.dart';

import 'package:b_im_q/components/text_filed_container.dart';
import 'package:b_im_q/constants.dart';
import
'package:b_im_q/desktop_screens/Login/components/background.dart';
import 'package:b_im_q/desktop_screens/signUp/signUp_screen.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
```

```

import '../.../backend/CRUD.dart';
import '../.../components/RememberMeCheck.dart';
import '../.../components/have_an_account_check.dart';
import '../.../components/rounded_password_field.dart';
import '../.../ResetPassword/ResetPwd_screen_desktop.dart';
import '../.../main_screen_desktop.dart';

class body extends StatefulWidget {

  const body({
    Key? key,
  }): super(key : key);

  @override
  State<body> createState() => _bodyState();
}

class _bodyState extends State<body> {
  TextEditingController loginController = TextEditingController();
  TextEditingController pwdController = TextEditingController();
  bool isLoggedIn = false;

  @override
  Widget build(BuildContext context) {
    Size size = MediaQuery.of(context).size;

    Users dataUser;
    final DioClient _client = DioClient();

    return Background(child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Text("Welcome desk!",
          style: TextStyle(
            fontFamily: 'Montserrat',
            color: kPrimaryColor,
            fontWeight: FontWeight.bold,
            fontSize: 36,
          ),
        ),
        SizedBox(height: size.height * 0.07),
        RoundedInputField(

          hintText: "Your login",
          icon: Icons.person_2_outlined,
          controller: loginController,
        ),
        RoundedPasswordField(controller: pwdController,),
      ],
    ),
  )

```

```

Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    GestureDetector(
      onTap: () {Navigator.push(
        context, MaterialPageRoute(
          builder: (context){
            return ResetPwdDesktop();
          },),
        );},
      child: Text(
        "Forgot password?",
        style: TextStyle(color: kPrimaryColor, fontWeight:
          FontWeight.bold),
      ),
    ],
  ),
  RoundedButtonLogSign(text: "Log in", press: () async {
    var username = loginController.text;
    var pwd = pwdController.text;
    isLoggedIn = await _client.dioLogin(username: username, password:
    pwd);
    if(isLoggedIn == true){
      //сделать переброс на главную страницу
      String? result = await _client.DioGetUserData();
      if(kDebugMode){
        print(result);
      }
      Navigator.push(
        context, MaterialPageRoute(
          builder: (BuildContext context){
            return mainScreenDesktop();
          },),
      );
    }
  },
  ),
  SizedBox(height: size.height * 0.03),
  HaveAnAccountCheck(press: () {
    Navigator.push(
      context, MaterialPageRoute(
        builder: (context){
          return SignUpScreenDesktop();
        },),
    );
  },)
],
));
}

```

}

После входа в аккаунт пользователь можно воспользоваться основными функциями данного проекта, а именно осветления тёмных и фильтрация затуманенных изображений. По нажатии на одну из кнопок “Dehaze image”, “Enlight image” открывается проводник, в котором пользователь должен выбрать изображение. На рисунке 1.23 представлен случай выбора изображения для одного из видов обработки.

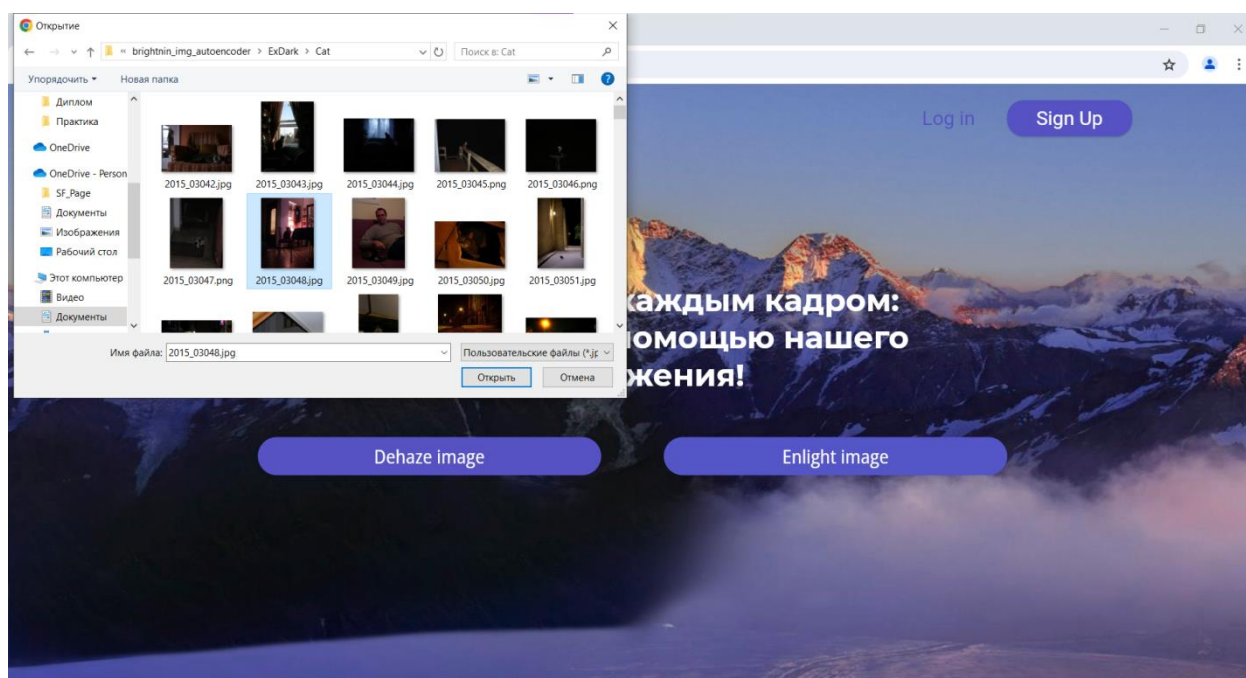


Рисунок 1.23 – Выбор изображения для обработки

Время обработки изображения одной из моделей зависит от множества факторов, однако, после обработки результат скачивается на устройство пользователя.

2 Безопасность жизнедеятельности

2.1 Правила безопасности при использовании ПК

В данном разделе необходимо учесть риски, перечисленные в ГОСТ 12.0.003-2015 “Система стандартов безопасности труда. Электронно-вычислительная техника. Общие требования безопасности”. Ниже приведён перечень рисков, которым следует придать особое значение:

- риск повреждения зрения и утомления глаз:
 - 1) длительное использование ПК без перерывов и отдыха глаз;
 - 2) плохая эргономика рабочего места, включая неправильное расположение монитора, отражение света на экране и недостаточное освещение;
- риск травм и заболеваний опорно-двигательной системы:
 - 1) неправильное положение тела при работе с ПК, например, неправильная осанка, неправильная поза или неправильное расположение рук и ног;
 - 2) отсутствие эргономической мебели и принадлежностей, таких как стулья с подлокотниками и подставки для рук;
- риск воздействия электромагнитных полей:
 - 1) использование устаревшего или неисправного оборудования, которое может создавать сильные электромагнитные поля;
 - 2) неправильное расположение оборудования, например, слишком близко к рабочему месту или другим электронным устройствам;
- риск пожара и электрического удара:
 - 1) неправильное подключение и использование электрооборудования, включая перегрузку розеток и использование повреждённых кабелей;

- 2) нарушение правил пожарной безопасности, включая использование запрещённых электроприборов вблизи ПК и наличие горючих;
- 3) материалов рядом с оборудованием;
- риск нарушения информационной безопасности:
 - 1) небезопасное использование паролей и учётных записей, включая слабые пароли или их совместное использование;
 - 2) несанкционированный доступ к компьютерной информации, включая утечку конфиденциальных данных;
- риск воздействия шума:
 - 1) использование шумных компьютерных компонентов или работа в шумной среде.

2.2. Правила безопасности при использовании смартфона

В этом разделе следует уделить внимание рисками, связанным с безопасностью при использовании данного устройства. А также разработать свод правил, чтобы гарантировать предотвратить следующие риски:

- риск зависимости и негативного воздействия на психологическое и физическое здоровье. Чрезмерное использование смартфона, включая зависимость от социальных сетей, игр и постоянного онлайн-присутствия, что может негативно сказываться на здоровье и социальной активности.
- риск отрицательного влияния на сон и здоровье:
 - 1) использование смартфона перед сном, из-за светового излучения экрана, может нарушать естественный сон и вызывать бессонницу;
 - 2) чрезмерное использование смартфона в тёмное время суток может снижать уровень мелатонина, гормона сна;
- риск потери или кражи смартфона:

1) несанкционированное получение доступа к устройству, возможность потери или кражи смартфона, что может привести к утечке личных данных или использованию их в недобросовестных целях;

2) нарушение конфиденциальности и безопасности данных в случае потери или кражи смартфона;

– риск воздействия радиочастотного излучения:

1) возможные воздействия радиочастотного излучения на организм при длительном использовании смартфона, хотя точные последствия такого воздействия все еще находятся в стадии исследования;

2) при работе в плохо покрытых районах мобильной связи смартфон может выделять больше энергии для подключения к сети, что может привести к повышенному радиочастотному излучению;

– риск отвлечения водителя. Использование смартфона во время вождения может отвлекать водителя и повышать риск дорожно-транспортных происшествий. Это особенно актуально в случае чтения сообщений, отправки сообщений или проведения телефонных разговоров во время движения.

С учетом указанных рисков, следует разработать и рекомендовать следующие правила безопасности при использовании смартфона:

– ограничьте время использования смартфона, особенно перед сном;

– соблюдайте эргономику и правильное положение тела при работе со смартфоном;

– используйте оригинальные зарядные устройства и кабели, избегайте перегрева и короткого замыкания;

– устанавливайте надежные пароли и используйте функции блокировки экрана для защиты данных;

– храните смартфон в безопасном месте и будьте бдительны в общественных местах;

– регулярно делайте резервные копии данных и используйте антивирусное программное обеспечение для защиты от вредоносных программ;

– ограничьте использование смартфона во время вождения и соблюдайте правила безопасности на дороге.

Правильное соблюдение этих правил поможет минимизировать риски и обеспечить безопасное использование смартфона.

					ПГУ1.090301.30.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

3 Стартап проект

Руководитель проекта планирует разработку сервиса улучшения качества изображения с использованием нейронной сети.

Разместить бизнес на начальном этапе планируется в помещении, в городе Пенза, которое должно соответствовать нормам и требованиям РК.

Цель деятельности предприятия - получение прибыли для улучшения материального благосостояния его участников, а также помощи физическим лицам и компаниям в улучшении качества изображений. Нужно предоставить интернет-пользователям приложение, способное улучшить качество изображений, тем самым позволяя сократить время на изменение параметров вручную. Основная целевая аудитория - интернет-пользователи, а в перспективе - компании, разрабатывающие системы видеонаблюдения и автопилоты.

3.1 Сегмент ЦА

Целевая аудитория состоит из физических лиц и при последующем масштабировании, компаний, разрабатывающих системы видеонаблюдения и автопилоты. Для более эффективного взаимодействия с каждым сегментом ЦА, необходимо учитывать их потребности и предпочтения.

Для каждого сегмента целевой аудитории будет подготовлено индивидуальное коммерческое предложение, учитывающее их потребности и предпочтения. Стоимость подписки на веб-приложение и модуль нейронных сетей будет варьироваться от типа задачи и количества обрабатываемых изображений в месяц.

Перед началом проекта необходимо определить ключевых партнеров для ведения бизнеса. В их число должны входить фотографы, видеомонтажеры, а также компании, специализирующиеся на системах видеонаблюдения и автопилотах.

3.2 Ключевые процессы

Ниже приведены ключевые процессы необходимые во время реализации стартап-проекта:

- создание ИП;
- поиск и создание команды высококвалифицированных специалистов;
- поиск денежных средств или спонсоров;
- разработка модуля;
- поиск каналов взаимодействия и сбыта с целью продажи продукта;
- поддержка данного продукта.

Прежде чем начать процедуру регистрации индивидуального предпринимателя, необходимо подготовить соответствующие документы и информацию. Первым шагом будет определение соответствующих кодов ОКВЭД для предстоящего бизнеса. Для этого необходимо воспользоваться специальным справочником и найти подходящий код. Для данного проекта был выбран 62.09 «Деятельность, связанная с использованием вычислительной техники и информационных технологий, прочая» и 72.19.9 «Научные исследования и разработки в области естественных и технических наук прочие, не включенные в другие группировки».

Вторым шагом будет определение налогового режима. Решение было принято в пользу упрощенной системы налогообложения (УСН), поскольку она обеспечивает легкость в расчете и уплате налогов, а также предлагает низкие ставки. В рамках этого режима можно самостоятельно рассчитывать и уплачивать ежеквартальные налоги, а также ежегодно подавать декларацию. Налог на доходы варьируется от 6% до 15%, в зависимости от региона.

На начальном этапе регионом размещения станет Пензенская область, ставка в которой составляет 6%. Для минимизации расходов на госпошлину решено подать документы через МФЦ.

Следующим этапом будет получение справки из ЕГРИП, которая будет отправлена на указанный в заявлении адрес, обычно это занимает 3 дня.

Формирование команды для разработки системы включает в себя подбор высококвалифицированных специалистов, таких как разработчики, бухгалтера и менеджеры по продажам. Разработчики будут выбраны из числа единомышленников. Бухгалтерский учет будет поручен специалистам из компании "Мое дело". Менеджер по продажам будет найден на веб-платформе для поиска работы.

Этапы разработки информационного сервиса описаны в пункте 1.

Поиск каналов сбыта осуществляется через предложения с коммерческим уклоном, которые выделяются своими преимуществами и помогают выделить проект среди конкурентов.

Также на отечественном рынке отсутствуют аналогичные информационные сервисы, которые также качественно освещают тёмные и фильтруют затуманенные изображения. Продукт может быть представлен в виде двух продуктов: веб-приложения и встраиваемого модуля нейронных сетей. Это позволяет занять уникальную позицию и быть востребованным проектом на рынке.

В таблице 4 приведена стоимость подписки на информационный сервис.

Таблица 4 – Стоимость и виды подписок на продукт

Назначение	Лимит	Абонентская плата	Сверх лимита
Для компаний, производящих фото- и видео- редакторы	Нет	8000 руб./год	Нет
Для компаний, производящих системы автопилотирования	946 080 000 (изображений)	20000 руб./год	10000 руб./год
Для компаний, производящих системы видеонаблюдения	946 080 000 (изображений)	20000 руб./год	10000 руб./год
Для физических лиц	77 760 000 (изображений)	100 руб./мес.	нет

3.3 Каналы взаимодействия

Одним из способов привлечения клиентов станет реклама сервиса в различных социальных сетях и на других сайтах. Данный метод позволит увеличить аудиторию и повысить узнаваемость и авторитет продукта.

Для демонстрации продукта компаниям, занимающим видеорегистраторами, камерами видеонаблюдения и автопилотов, планируется путём обзвона или переписки, согласовать встречи с директорами или их заместителями. На данных встречах можно показать возможности модели и ответить на вопросы потенциальных клиентов, а также обозначить вектор дальнейшего развития.

После заключения договора, наша команда будет отвечать за поставку и внедрения модуля. При необходимости разработчик нашей команды будет осуществлять внедрение обученной модели с определёнными параметрами на сервер или напрямую в устройство клиента.

Четкое планирование и структурирование этапов внедрения способствуют экономии времени и обеспечивают плавный переход к полноценному использованию сервиса и модуля. Поддержка клиентов в течение нескольких месяцев после приобретения продукта не менее важна. Это

включает в себя разрешение технических вопросов, адаптацию и улучшение системы в соответствии с потребностями клиента, а также предоставление консультаций и обучение пользователей. Быстрый и качественный отклик на запросы клиентов способствует формированию положительного опыта взаимодействия. Проведение периодических встреч или вебинаров для сбора обратной связи и обсуждения возможностей модуля также является необходимым.

3.4 Источники дохода

Информационный сервис будет иметь представленные ниже источники доходов:

- продажа. Команда будет получать доход от продажи подписки физическим лицам или компаниями, занимающимися продаже систем видеонаблюдения, автопилотирования, а также видеорегистраторов. Стоимость подписки представлена в таблице 4;

- сверх лимита. Нужно будет предоставить возможность компаниям-клиентам увеличения количества обрабатываемых изображений. Плата за увеличения лимита составит 10000 рублей в год;

- реклама в веб-приложении. После того, как веб-приложение станет популярным, предлагается интегрировать ненавязчивую рекламу компаний, находящихся в том же рынке.

На момент 2024 года насчитывается: около 1 миллиарда физических лиц во всём мире, пользующихся фото- и видео- редакторами; более 80 крупных компаний, занимающихся разработкой разработкой данных редакторов; более 500 крупных компаний в мире, разрабатывающих и производством системы автопилотирования, а также более 800 компаний, изготавливающих системы видео наблюдений. Предположив, что около 40% всех компаний приобретут наш продукт, тогда получаем 552 клиента.

Предположив, что приблизительно 20% физических лиц оформят подписку, получаем 20 000 000 потенциальных клиентов.

Для вычисления дохода воспользуемся следующей формулой:

$$Д = К * Ц, \quad (9)$$

где Д – доход;

К – количество потребителей;

Ц – установленная цена;

Доход от подписки на веб-приложение будет определяться:

$$20\,000\,000 \text{ пр} * 100 \text{ р} = 2\,000\,000\,000 \text{ р в месяц}$$

Доход от абонентской платы на модуль будет определяться:

$$552 \text{ пр} * 8000 \text{ р} = 4\,416\,000 \text{ р в год}$$

Доход от рекламы высчитывается:

$$552 \text{ пр} * 1000 \text{ р} = 552\,000 \text{ р в месяц}$$

Таким образом, общий доход в месяц составит 2 000 368 000.

3.5 Ключевые ресурсы

Для эффективного ведения и дальнейшего развития бизнеса необходимо обеспечить определённые ключевые ресурсы. В нашем конкретном случае они представлены следующим образом:

- разработчики программного обеспечения, специализирующиеся на создании веб-приложения;
- специалисты по машинному обучению и глубоким нейронным сетям для дообучения моделей;
- менеджеры по продажам, обладающие навыками продажи и умением вести переговоры с потенциальными клиентами;
- персональные компьютеры, серверы с мощными графическими картами, программное обеспечение и другое техническое оборудование для разработки и обслуживания информационного сервиса;

- офис для работы команды и проведения встреч с клиентами;
- финансовые ресурсы для финансирования разработки, маркетинга, продаж и операционных расходов бизнес.

Предполагается, что финансирование проекта будет реализовано путем использования как личных средств руководителя и членов команды, так и путем получения государственной финансовой поддержки «Фонд содействия развитию малых форм предприятий в научно-технической сфере». В таблице 5 представлена программа финансирования.

Таблица 5 – Источники финансирования

Источник финансирования	Сумма
Собственные средства	500 000руб.
Поддержка государством	1 000 000руб.
Итого	1 500 000руб.

Более подробно направления использования средств указано в таблицах 6 и 7.

Таблица 6– Ключевые финансовые ресурсы

Направление использования средств	Кол-во., шт.	Цена, руб.	Сумма, руб.
Аренда облачного сервера (в месяц)	3	62 830	188 490
Аренда помещения (в месяц)	3	8000	24 000
Закупка персональных компьютеров	4	150 000	600 000
Закупка предметов канцелярии (наборы)	4	2000	8000
Коммунальные услуги (в месяц)	3	5000	15000

Таблица 7 – Ключевые людские ресурсы

Направление использования средств	Кол-во., шт.	Цена, руб.	Сумма, руб.
З/П разработчику (за 3 месяца)	12	70 000	840 000
З/П бухгалтеру (за 3 месяца)	3	20 000	60 000
З/П менеджеру по продажам (за 3 месяца)	3	30 000	90 000

3.6 Риски проекта

Реализация стартап-проекта может быть сопряжена с различными рисками и форс-мажорными обстоятельствами, которые способны негативно повлиять на его монетизацию.

К возможным рискам можно отнести:

- Технические риски. Существует риск того, что команда разработчиков не сможет реализовать задачу в срок или не будет достаточно квалифицирована для выполнения проекта;
- Конкурентные и интеллектуальные риски. Рынок улучшения изображений перенасыщен различными конкурентами, и стартап может столкнуться с сильной конкуренцией, а также с нарушением интеллектуальной собственностью стартапа;
- Финансовые риски. Развитие и масштабирование стартапа требует значительных инвестиций, и существует риск того, что компания не

сможет привлечь достаточно финансирования для реализации своих планов развития;

– Риски безопасности данных. С увеличением объема обрабатываемых данных стартап становится уязвимым для кибератак и утечек конфиденциальной информации.

Именно поэтому важно провести качественный анализ и предусмотреть меры по минимизации вероятных негативных последствий.

					ПГУ1.090301.30.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

Результат выполнения данной выпускной квалификационной работы можно считать успешным, поскольку был реализован информационный сервис для улучшения качества изображений с использованием нейронной сети в соответствии с заданием.

В результате выполнения данной работы были выполнены следующие задачи:

- исследование существующих моделей нейронных сетей, улучшающих качество изображений;
- разработка баз данных изображений. Было создано две базы данных, объединяющих 4832 пары затуманенных и “чистых” изображений, а также 3300 пар тёмных и светлых фотографий;
- выбор критерия качества изображений. Были выбраны метрики пикового отношения сигнала к шуму (PSNR) и индекса структурного сходства (SSIM) как наиболее подходящие для задач восстановления изображений;
- выбор СУБД. Выбор PostgreSQL был обусловлен следующим: данная система обладает рядом преимуществ, включая мощные функциональные возможности, широкий спектр типов данных и поддержку сложных запросов. Кроме того, он известен своей надежностью, масштабируемостью и открытым исходным кодом, что делает его популярным выбором для различных проектов и приложений;
- разработка функционала информационного сервиса. Кроме основных функций осветления тёмных и фильтрации затуманенных изображений, были реализованы возможности регистрации и авторизации пользователя.

В результате данной выпускной квалификационной работы был создан сервис улучшения качества изображений, который обеспечивает качественную

обработку изображений, а также предоставляет интуитивно понятный интерфейс.

Дальнейшее развитие проекта включает, добавление функций оплаты подписки, восстановления пароля, отображения все обработанных пользовательских изображений и оптимизацию процессов.

					ПГУ1.090301.30.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Явления, связанные с поглощением и рассеянием света. URL:<https://forkettle.ru/vidioteka/estestvoznaniye/fizika-dlya-chajnikov/779-optika/opticheskie-yavleniya-v-prirode/8622-yavleniya-svyazannye-s-pogloshcheniem-i-rasseyaniem-sveta> (дата обращения 19.04.2024)
2. Рассеяние света и оптические явления в атмосфере URL:<https://poznayka.org/s25638t2.html> (дата обращения 21.04.2024)
3. Рассеяние света. URL: <https://physics.spbstu.ru/userfiles/files/opt3-9.pdf> (дата обращения 15.05.2024)
4. Zhou, W., A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. // IEEE Transactions on Image Processing. Vol. 13, Issue 4, April 2004, pp. 600–612 (дата обращения 15.05.2024)
5. Zhenzhen Y., Zhen Y., Guan G. A Convex Constraint Variational Method for Restoring Blurred Images in the Presence of Alpha-Stable Noises // Sensors, 18(4):1175, 2018 (дата обращения 19.05.2024)
6. C. Guo, Q. Yan, S. Anwar, R. Cong, W. Ren and C. Li. // Image Dehazing Transformer with Transmission-Aware 3D Position Embedding, 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 2022, pp. 5802-5810, doi: 10.1109/CVPR52688.2022.00572 (дата обращения 21.05.2024)
7. Yi T., Mostafa D., Jai G., Vamsi A., Dara B., Zhen Q. Are Pre-trained Convolutions Better than Pre-trained Transformers? // Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 4349–4359. 2021 (дата обращения 23.05.2024)
8. Кожевникова А.В. Освещение участков изображений с использованием свёрточной нейронной сети // Новые информационные

<https://elibrary.ru/item.asp?edn=mnlnmtf>

9. Boyi Li1, Felix Wu, Kilian Q. Weinberger, Serge Belongie. // Positional normalization. 33rd Conference on Neural Information Processing Systems ,2017 (дата обращения 23.05.2024)

10. Liang S., Zihan Y., Fan F., Quan C., Shihao L., and Jie M.// MSR-net:Low-light Image Enhancement Using Deep Convolutional Network.arXiv:1711.02488v1, 2017 (дата обращения 24.05.2024)

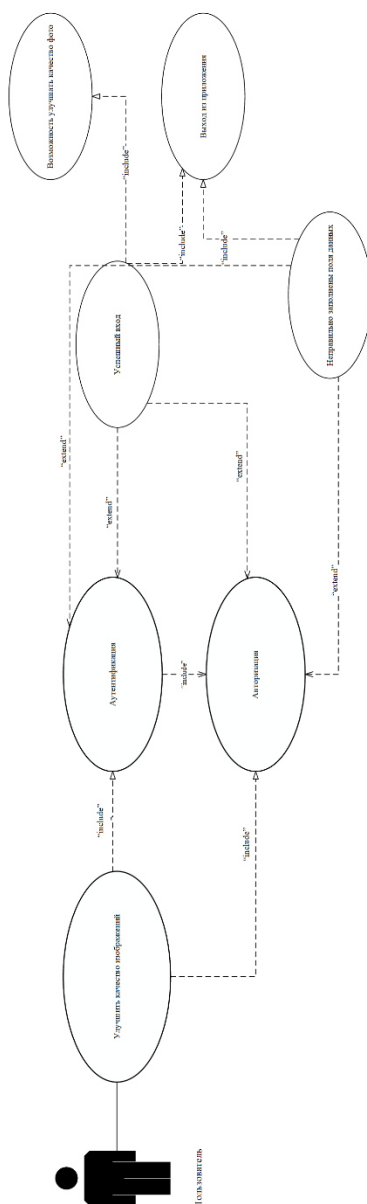
Приложение А

(Обязательное)

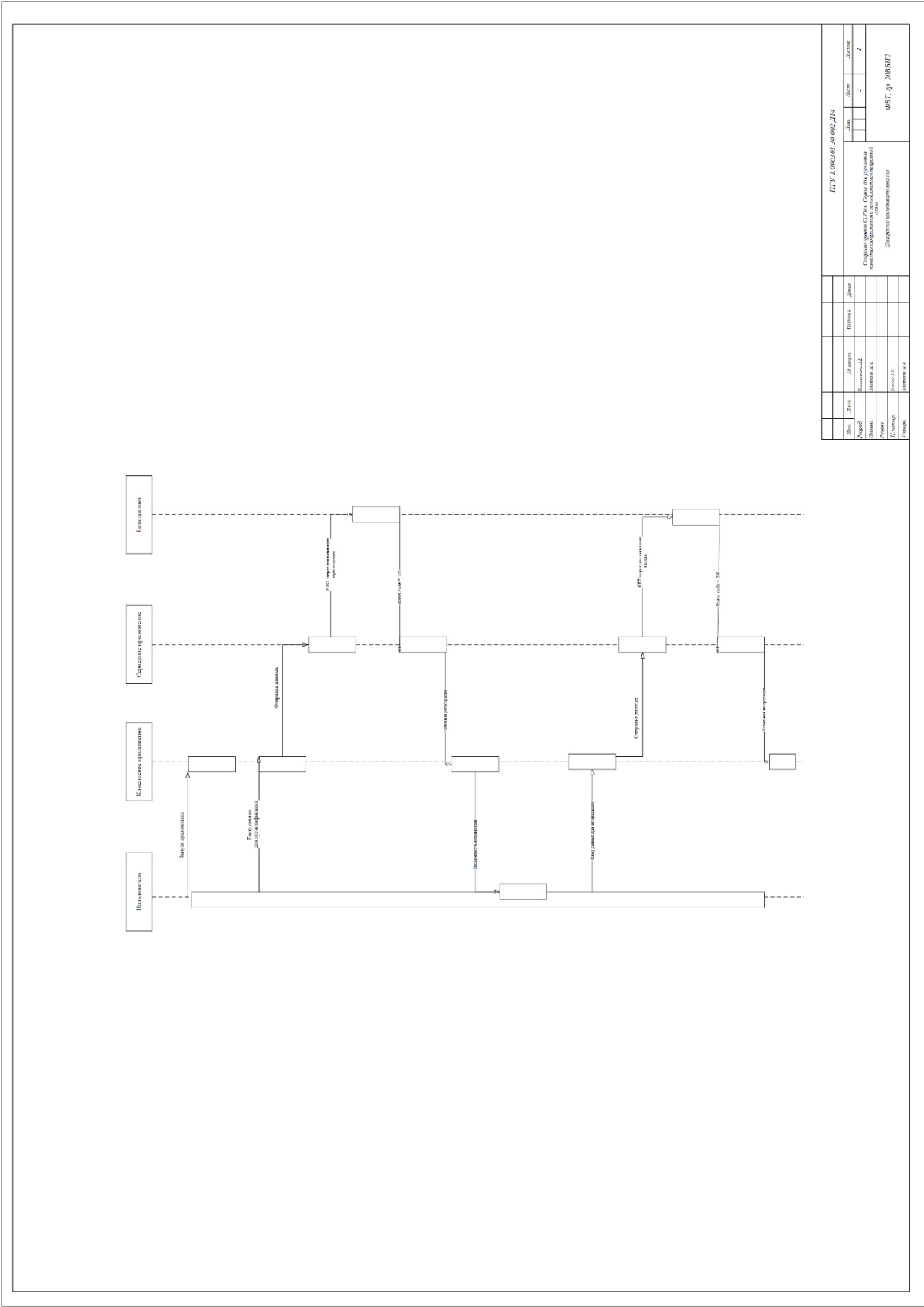
Стартап-проект CLView. Сервис для улучшения качества изображений с
использованием нейронной сети
UML-диаграммы

					ПГУ1.090301.30.001 ПЗ	Лист
						65
Изм.	Лист	№ докум.	Подпись	Дата		

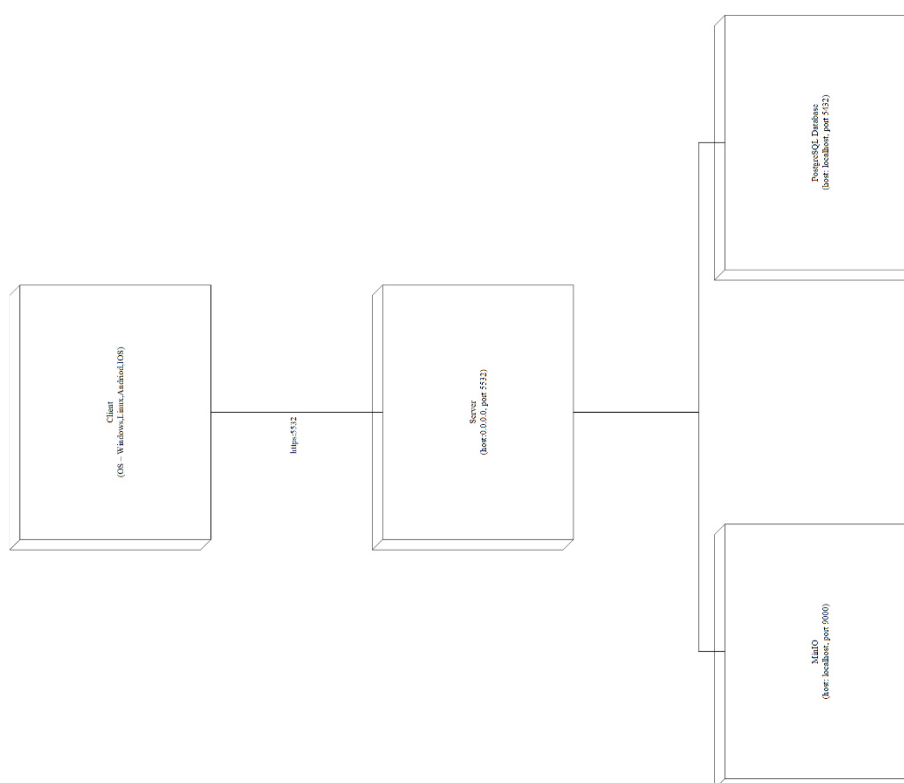
Приложение А.1 – UML – диаграмма вариантов пользования

[illegible]

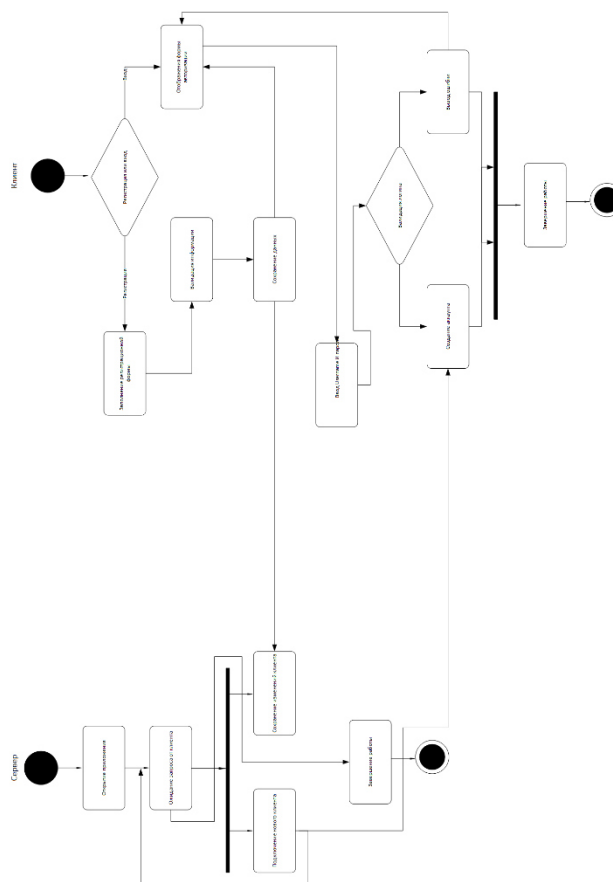
Приложение А.2 – UML диаграмма последовательности



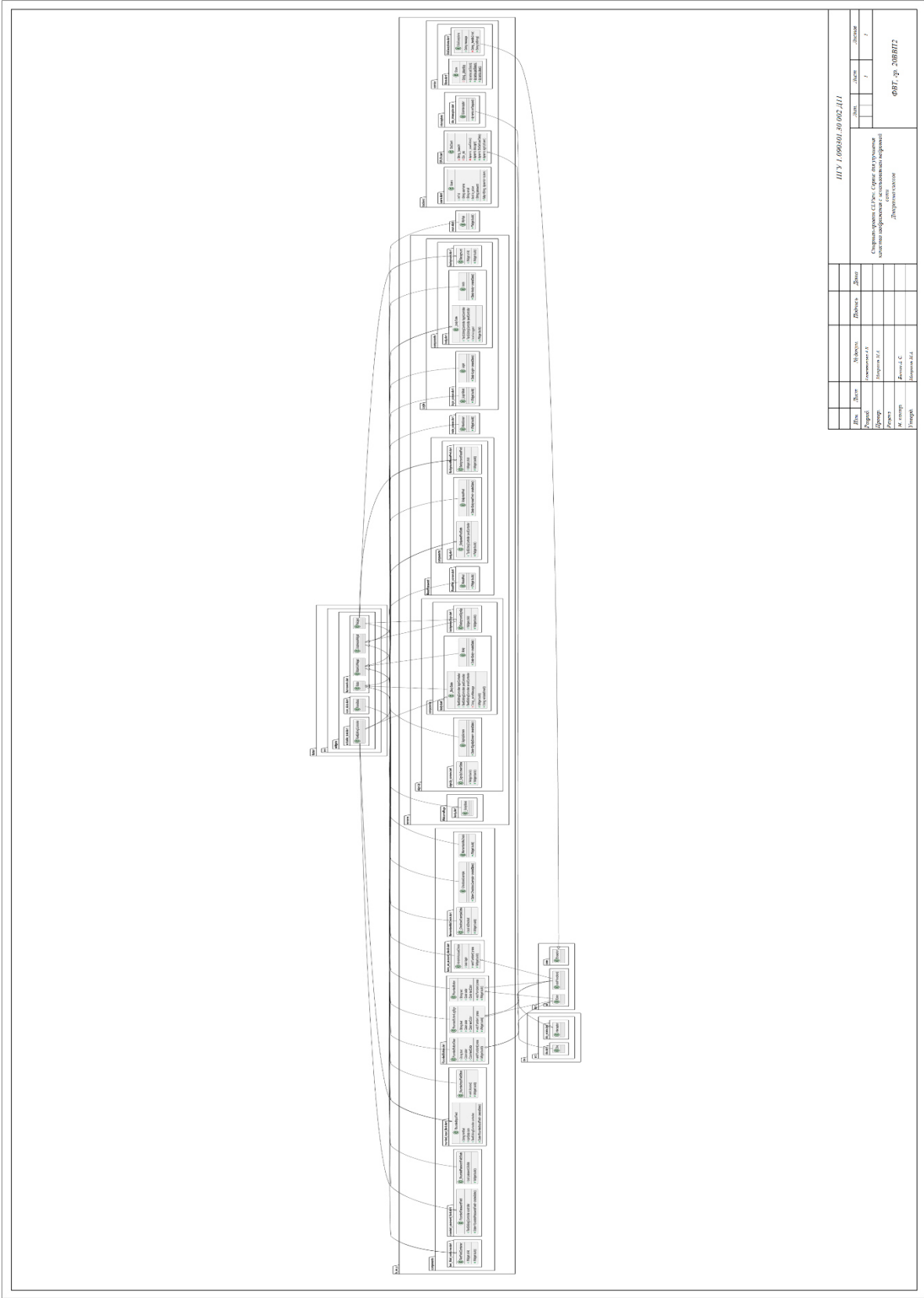
Приложение А.3 – UML диаграмма развёртывания

[illegible]

Приложение А.4 – UML диаграмма деятельности

[illegible]

Приложение А.5 – UML диаграмма классов



ИД 1.090301.30.001.ПЗ									
Имя	Иван	Михайлов	Павлов	Жуков	Иванов	Петров	Сидоров	Смирнов	Тихонов
Фамилия	Иванов	Иванов	Иванов	Иванов	Иванов	Иванов	Иванов	Иванов	Иванов
Роль	Администратор	Администратор	Администратор	Администратор	Администратор	Администратор	Администратор	Администратор	Администратор
Ранг	Младший	Младший	Младший	Младший	Младший	Младший	Младший	Младший	Младший
М. номер	1	2	3	4	5	6	7	8	9
Подпись	[Подпись]								
Дата	[Дата]								

Изм.	Лист	№ докум.	Подпись	Дата

Приложение Б

(Обязательное)

Стартап-проект CLView. Сервис для улучшения качества изображений с
использованием нейронной сети

Листинг программы

					ПГУ1.090301.30.001 ПЗ	Лист
						72
Изм.	Лист	№ докум.	Подпись	Дата		

Файл «main.py»

```

import uvicorn
from fastapi import FastAPI, HTTPException, Depends, Security,
status, UploadFile, File
from fastapi.security import OAuth2PasswordRequestForm
from typing import Annotated
import models
from database import engine
from sqlalchemy.orm import Session
from datetime import timedelta
import schemas_user, crud, crud_model, schemas_image
import uuid
import utils.email_send

models.Base.metadata.create_all(bind=engine)
app = FastAPI()

@app.get("/users/{user_id}", status_code=status.HTTP_200_OK)
async def read_user(user_id: int, db: crud.db_dependency):
    user = db.query(models.User).filter(models.User.id ==
user_id).first()
    if user is None:
        raise HTTPException(status_code=404, detail='User not found')
    return user

@app.post("/users/", response_model=schemas_user.UserCreate)
def create_user(user: schemas_user.UserCreate, db: Session =
Depends(crud.get_db)):
    db_user = crud.get_user_by_email_query(db=db,
email=user.email)
    if db_user:
        raise HTTPException(status_code=400, detail="Email already
registered")
    return crud.create_user_query(db=db, user=user)

@app.post("/token", response_model=schemas_user.Token)
async def login_for_access_token(form_data:
OAuth2PasswordRequestForm = Depends(), db: Session=
Depends(crud.get_db)):
    user = crud.authenticate_user(db, form_data.username,
form_data.password)
    if not user:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Incorrect username or password",
            headers={"WWW-Authenticate": "Bearer"},
        )

```

					ПГУ1.090301.30.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		73

```

        access_token_expires =
timedelta(minutes=crud.ACCESS_TOKEN_EXPIRE_MINUTES)
        access_token = crud.create_access_token(
data={"sub": user.username}, expires_delta=access_token_expires
        )
return {"access_token": access_token, "token_type": "bearer"}

@app.get("/users/me/", response_model=schemas_user.UserBase)
async def read_users_me(
    current_user: Annotated[schemas_user.UserBase,
Depends(crud.get_current_active_user)]
):
return current_user

@app.patch("/users/forgot-password/")
async def forgot_password(forgot_pwd:
schemas_user.ForgotPassword,
db: Session = Depends(crud.get_db)):
    db_user = crud.get_user_by_email_query(db=db,
email=forgot_pwd.email)
    if not db_user:
raise HTTPException(status_code=404, detail="User not found")

        #create reset code and save it database
        reset_code = str(uuid.uuid1())
        reset_code_expires =
timedelta(minutes=crud.RESET_CODE_EXPIRE_MINUTES)

        crud.create_reset_code(db=db, email=forgot_pwd.email,
reset_code=reset_code, expires_delta=reset_code_expires)

        #sending email
subject = "CLView user: You want to reset your password"
recipient = forgot_pwd.email
        message = f"""
<!DOCTYPE html>
<html>
<title>Сброс пароля</title>
<body>
<div style="width:100; font-family:monospace;">
<h1>Приветствуем тебя</h1>
<pstyle="font-size=18px;">К нам пришло уведомление о твоём
желании сбросить пароль. Если это желание твоего сердца,то мы не
смеем мешать. Для сброса пароля нажми на волшебную кнопку
ниже</p>
<a href="http://192.168.208.137:5532/users/forgot-
password?reset_password_token={reset_code}" style="box-
size:border-box; border-color:#1fgfeb;text-
decoration:none;background-color:#1fgfeb;color:#ffffff;font-

```

```

size:18px;font-weight:bold;margin:0;padding:12px
24px;display:inline-block;"></a>
<pstyle="font-size=18px;">Пароль не сброситься пока ты не
нажмёшь на ссылку и помни,что время её действия ограничено!</p>
</div>
</body>
</html>
"""

#чтобы функция работала надо завести no-reply почту
utils.email_send.send_mail(subject, recipient, message)

return {
    "code": 200,
    "message": "Мы отправили письмо для сброса пароля на Ваш
email."
}

@app.post("/users/{user_id}/dehaze/",status_code=status.HTTP_201
_CREATED)
async def enhancement_image(current_user:
Annotated[schemas_user.User,
Security(crud.get_current_active_user)], class_name: int,
image_url: UploadFile = File(...), db: Session =
Depends(crud.get_db)):
return crud_model.create_user_image(db=db,
user_id=current_user.id, class_name=class_name,
imageName=image_url)

if __name__ == "__main__":
uvicorn.run("main:app", host="HOST", port=PORT)

```

Файл «crud.py»

```

from fastapi.security import OAuth2PasswordBearer
from jose import JWTError, jwt
from passlib.context import CryptContext
from datetime import datetime, timedelta, timezone
import CLView.models
from fastapi import HTTPException, Depends, status
from sqlalchemy.orm import Session

from CLView.schemas_user import *
from CLView.database import *
from pydantic import EmailStr
from typing import Annotated

```

```

SECRET_KEY =
"1cc638fc401f85c408a3eddf0a2a35b761e79c5cfb09445a53db69cb5c87079
6"
ALGORITHM = "HS256"

# 30 minutes
ACCESS_TOKEN_EXPIRE_MINUTES = 30

#2 minutes
RESET_CODE_EXPIRE_MINUTES = 2

pwd_context = CryptContext(schemes=["bcrypt"],
deprecatd="auto")

oauth2_scheme = OAuth2PasswordBearer(tokenUrl="token")

def get_db():
db = SessionLocal()
try:
yield db
finally:
db.close()

db_dependency = Annotated[Session, Depends(get_db)]

def verify_password(plain_password, hashed_password):
return pwd_context.verify(plain_password, hashed_password)

def get_password_hash(password):
return pwd_context.hash(password)

def get_user(username: str, db: Session):
return
db.query(CLView.models.User).filter(CLView.models.User.username
== username).first()

#username: sunny, password: 12345
#username: admin, password: admin
#username: belix, password: 147258
def authenticate_user(db: Session, username: str, password: str):
user = get_user(username, db=db)
if not user:
return False
if not verify_password(password, user.password):
return False
return user

```



```

def create_access_token(data: dict, expires_delta: timedelta |
None = None):
    to_encode = data.copy()
    if expires_delta:
        expire = datetime.now(timezone.utc) + expires_delta
    else:
        expire = datetime.now(timezone.utc) + timedelta(minutes=15)
    to_encode.update({"exp": expire})
    encoded_jwt = jwt.encode(to_encode, SECRET_KEY,
algorithm=ALGORITHM)
    return encoded_jwt

async def get_current_user(token: Annotated[str,
Depends(oauth2_scheme)], db: Session = Depends(get_db)):
    credentials_exception = HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="Could not validate credentials",
        headers={"WWW-Authenticate": "Bearer"},
    )
    try:
        payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
        username: str = payload.get("sub")
        if username is None:
            raise credentials_exception
        token_data = TokenData(username=username)
    except JWTError:
        raise credentials_exception
    user = get_user(username=token_data.username, db=db)
    if user is None:
        raise credentials_exception
    return user

async def get_current_active_user(
    current_user: Annotated[UserBase, Depends(get_current_user)]
):
    if not current_user.is_active:
        raise HTTPException(status_code=400, detail="Inactive user")
    return current_user

def create_user_query(db: Session, user:
CLView.schemas_user.UserCreate):
    """create user by email and password"""
    hashed_password = get_password_hash(user.password)
    db_user = CLView.models.User(username=user.username,
email=user.email, password=hashed_password)
    db.add(db_user)
    db.commit()
    db.refresh(db_user)
    return db_user

```

```

def get_user_by_email_query(db: Session, email: EmailStr):
    """get user by email"""
    return
    db.query(CLView.models.User).filter(CLView.models.User.email ==
    email).first()

def create_reset_code(db:Session, email:str, reset_code: str,
expires_delta: timedelta | None = None):
    if expires_delta:
        expire = datetime.now(timezone.utc) + expires_delta
    else:
        expire = datetime.now(timezone.utc) + timedelta(minutes=2)

    db_reset = CLView.models.Codes(email=email,
reset_code=reset_code, status='1', expired_in=expire)
    db.add(db_reset)
    db.commit()
    db.refresh(db_reset)
    return db_reset

```

Файл «crud_model.py»

```

import torchvision
from sqlalchemy.orm import Session
from fastapi import UploadFile, HTTPException, status, File
import CLView.models
import requests
from io import BytesIO
from loguru import logger
from dotenv import load_dotenv
from minio import Minio
from CLView.utils.dehaze_light_image import *
import uuid
import os

def create_user_image(imageName: UploadFile = File(...)):
    # loading variables from .env file
    load_dotenv()
    if imageName.content_type not in ["image/jpeg", "image/png"]:
        raise
    HTTPException(status_code=status.HTTP_415_UNSUPPORTED_MEDIA_TYPE
, detail="Not an image format")

    #рассмотреть вариант когда картинка слишком большая

    logger.debug('Uploading to S3')

    try:
        file_id = str(uuid.uuid4())
        file_name = file_id + ".png"

```

```

client = Minio(
    endpoint=os.getenv('S3_ENDPOINT_URL'),
        access_key=os.getenv('S3_ACCESS_KEY_ID'),
        secret_key=os.getenv('S3_SECRET_ACCESS_KEY'),
    secure=False,
)
print(client.list_buckets())
if not client.bucket_exists(os.getenv('S3_BUCKET')):
    client.make_bucket(os.getenv('S3_BUCKET'))

    client.put_object(os.getenv('S3_BUCKET'),
object_name=file_name, data=imageName.file,
length=os.fstat(imageName.file.fileno()).st_size,
content_type="image/png")

    upload_file_url =
f"http://127.0.0.1:9000/{os.getenv('S3_BUCKET')}/{file_name}"
except Exception as e:
    logger.error(e)
    raise
HTTPException(status_code=status.HTTP_422_UNPROCESSABLE_ENTITY,
detail="Bad Zip")

logger.info("S3 attachment upload task done.")

return upload_file_url

def enhance_image(db: Session, user_id: int, class_name: int,
imageName: UploadFile = File(...)):
    img_url = create_user_image(imageName=imageName)

response = requests.get(img_url)
img = Image.open(BytesIO(response.content))

transform = torchvision.transforms.Compose([
    torchvision.transforms.ToTensor(),
        #transforms.ConvertImageDtype(torch.float32),
    ])
    img_tensor = transform(img)

try:
    enhance_image_tensor = lightning_image(img_tensor)
except Exception as e:
    logger.error(e)
    raise
HTTPException(status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
detail="Can't be process by model")

logger.debug('Uploading to S3')

```

```

try:
    image = tensor_to_image(enhance_image_tensor)
    img_bytes = BytesIO()
    image.save(img_bytes, "PNG")
    #torch.save(image, img_bytes)
    img_bytes.seek(0)

    file_id = str(uuid.uuid4())
    file_name = file_id + ".png"

    client = Minio(
        endpoint=os.getenv('S3_ENDPOINT_URL'),
        access_key=os.getenv('S3_ACCESS_KEY_ID'),
        secret_key=os.getenv('S3_SECRET_ACCESS_KEY'),
        secure=False,
    )
    print(client.list_buckets())
    if not client.bucket_exists(os.getenv('S3_NEW_BUCKET')):
        client.make_bucket(os.getenv('S3_NEW_BUCKET'))

        client.put_object(os.getenv('S3_NEW_BUCKET'),
            object_name=file_name, data=img_bytes,
            length=img_bytes.getbuffer().nbytes, content_type="image/png")

        upload_file_url =
f"http://127.0.0.1:9000/{os.getenv('S3_NEW_BUCKET')}/{file_name}"
    except Exception as e:
        logger.error(e)
        raise
    HTTPException(status_code=status.HTTP_422_UNPROCESSABLE_ENTITY,
        detail="Bad Zip")

    logger.info("S3 attachment upload task done.")

    db_image = CLView.models.Images(image_url=upload_file_url,
        user_id=user_id)
    db.add(db_image)
    db.commit()
    db.refresh(db_image)

    db_class = CLView.models.ImageClass(img_class=class_name,
        image_id=db_image.id)
    db.add(db_class)
    db.commit()
    db.refresh(db_class)

    def dehaze_image(db: Session, user_id: int, class_name: int,
        imageName: UploadFile = File(...)):

```

```

img_url = create_user_image(imageName=imageName)

response = requests.get(img_url)
img = Image.open(BytesIO(response.content))

img_transform = torchvision.transforms.Compose([
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize(mean=[0.64, 0.6,
0.58],std=[0.14,0.15, 0.152]),
]) (img) [None, ::]
    #img_tensor = transform(img)

try:
    dehaze_image_tensor = single_dehaze_test(img_transform)
except Exception as e:
    logger.error(e)
    raise
HTTPException(status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
detail="Can't be process by model")

logger.debug('Uploading to S3')

try:
image = tensor_to_image(dehaze_image_tensor)
    img_bytes = BytesIO()
image.save(img_bytes, "PNG")
    #torch.save(image, img_bytes)
    img_bytes.seek(0)

    file_id = str(uuid.uuid4())
    file_name = file_id + ".png"

client = Minio(
endpoint=os.getenv('S3_ENDPOINT_URL'),
    access_key=os.getenv('S3_ACCESS_KEY_ID'),
    secret_key=os.getenv('S3_SECRET_ACCESS_KEY'),
secure=False,
)
print(client.list_buckets())
if not client.bucket_exists(os.getenv('S3_NEW_BUCKET')):
    client.make_bucket(os.getenv('S3_NEW_BUCKET'))

    client.put_object(os.getenv('S3_NEW_BUCKET'),
object_name=file_name, data=img_bytes,
length=img_bytes.getbuffer().nbytes, content_type="image/png")

    upload_file_url =
f"http://127.0.0.1:9000/{os.getenv('S3_NEW_BUCKET')}/{file_name}"
except Exception as e:

```

```

logger.error(e)
raise
HTTPException(status_code=status.HTTP_422_UNPROCESSABLE_ENTITY,
detail="Bad Zip")

logger.info("S3 attachment upload task done.")

db_image = CLView.models.Images(image_url=upload_file_url,
user_id=user_id)
db.add(db_image)
db.commit()
db.refresh(db_image)

db_class = CLView.models.ImageClass(img_class=class_name,
image_id=db_image.id)
db.add(db_class)
db.commit()
db.refresh(db_class)

```

Файл «models.py»

```

from sqlalchemy import Boolean, Column, ForeignKey, Integer,
String, DateTime
from sqlalchemy.orm import relationship
from CLView.database import Base

class User(Base):
    __tablename__ = "user_data"
    #__table_args__ = {'extend_existing': True}

    id = Column(Integer, primary_key=True, index=True)
    username = Column(String(64), unique=True)
    email = Column(String(64), unique=True)
    password = Column(String(64))
    is_active = Column(Boolean, default=True)

    images = relationship("Images", back_populates="owner")

class Codes(Base):
    __tablename__ = "codes"
    #__table_args__ = {'extend_existing': True}

    id = Column(Integer, primary_key=True, index=True)
    email = Column(String(64))
    reset_code = Column(String(64), unique=True)
    status = Column(String(1))
    expired_in = Column(DateTime)

class Images(Base):
    __tablename__ = "images"
    #__table_args__ = {'extend_existing': True}

```

```

id = Column(Integer, primary_key=True, index=True)
image_url = Column(String(100))
user_id = Column(Integer, ForeignKey("user_data.id"))

owner = relationship("User", back_populates="images")
img_class = relationship("ImageClass",
back_populates="model")

class ImageClass(Base):
    __tablename__ = "image_class"
    #__table_args__ = {'extend_existing': True}

id = Column(Integer, primary_key=True, index=True)
img_class = Column(Integer)
image_id = Column(Integer, ForeignKey("images.id"))

model = relationship("Images", back_populates="img_class")

```

Файл «schemas_image.py»

```

from pydantic import BaseModel, Field

# улучшение изображения клиента
class ImageBase(BaseModel):
    image_url: str = Field(...)

class ImageCreate(ImageBase):
    pass

class Image(ImageBase):
    id: int
    user_id: int

class Config:
    from_attributes = True

class ImageClass(BaseModel):
    img_class: int
    image_id: int

class Config:
    from_attributes = True

```

Файл «schemas_user.py»

```

from pydantic import BaseModel, EmailStr, Field
import CLView.schemas_image

class Token(BaseModel):

```



```

        access_token: str
        token_type: str

class TokenData(BaseModel):
    username: str

class UserBase(BaseModel):
    username: str = Field(...)
    email: EmailStr = Field(...)
    is_active: bool

class UserCreate(UserBase):
    password: str = Field(...)

class User(UserBase):
    id: int
    images: list[CLView.schemas_image.Image] = []

class Config:
    from_attributes = True

class ForgotPassword(BaseModel):
    email: str = Field(...)

```

Файл «database.py»

```

from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy.ext.declarative import declarative_base

SQLALCHEMY_DATABASE_URL =
"postgresql://postgres:dtv.irf@localhost:5532/db_clview"
engine = create_engine(SQLALCHEMY_DATABASE_URL)

SessionLocal = sessionmaker(autocommit=False, autoflush=False,
bind=engine)

Base = declarative_base()

```

Файл «email_send.py»

```

import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import os
from dotenv import load_dotenv
from typing import List
# loading variables from .env file
load_dotenv()

def send_mail(subject: str, recipient: str, template: str):
    # create email

```

```

msg = MIMEMultipart()
msg['Subject'] = subject
msg['From'] = os.getenv("MAIL_USERNAME")
msg['To'] = recipient # type Email

msg.attach(MIMEText(template, 'html'))

# send email
s = smtplib.SMTP(os.getenv("MAIL_SERVER"),

os.getenv("MAIL_PORT"))

try:
    # start TLS for security
s.starttls()
    # Authentication
s.login(msg['From'], os.getenv("MAIL_PASSWORD"))
    # sending the mail
s.sendmail(msg['From'], msg['To'], msg.as_string())
except Exception as e:
print(e)
finally:
s.quit()

```

Файл «dehaze_light_image.py»

```

import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import os
from dotenv import load_dotenv
from typing import List
# loading variables from .env file
load_dotenv()

def send_mail(subject: str, recipient: str, template: str):
    # create email
    msg = MIMEMultipart()
    msg['Subject'] = subject
    msg['From'] = os.getenv("MAIL_USERNAME")
    msg['To'] = recipient # type Email

    msg.attach(MIMEText(template, 'html'))

    # send email
    s = smtplib.SMTP(os.getenv("MAIL_SERVER"),
os.getenv("MAIL_PORT"))

    try:
        # start TLS for security
s.starttls()

```

```

        # Authentication
s.login(msg['From'], os.getenv("MAIL_PASSWORD"))
        # sending the mail
s.sendmail(msg['From'], msg['To'], msg.as_string())
except Exception as e:
print(e)
finally:
s.quit()

```

Файл «main.dart»

```

import 'package:b_im_q/constants.dart';
import 'package:b_im_q/mobile_screens/main_screen.dart';
import

'package:b_im_q/desktop_screens/main_screen_desktop.dart';
import 'package:b_im_q/tablet_screens/main_screen_tablet.dart';
import 'package:flutter/material.dart';
import 'package:b_im_q/responsive_layout.dart';
import 'package:dcdg/dcdg.dart';

void main() {
runApp(const MyApp());
}

class MyApp extends StatelessWidget {
const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
return MaterialApp(
debugShowCheckedModeBanner: false,
title: 'Make better image quality',
theme: ThemeData(
primaryColor: kPrimaryColor,
scaffoldBackgroundColor: Colors.white,
),
home: ResponsiveLayout(
mobileBody: const mainScreenMobile(),
tabletBody: const mainScreenTablet(),
desktopBody: const mainScreenDesktop(),
),
);
}
}

```

Файл «main_screen.dart»

```

import
'package:b_im_q/desktop_screens/WelcomePage/body_desktop.dart';
import 'package:flutter/material.dart';

```

```

class mainScreenDesktop extends StatelessWidget {
  const mainScreenDesktop({super.key});

  @override
  Widget build(BuildContext context) {
    return const Scaffold(body: Body());
  }
}

```

Файл «welcome.dart»

```

import 'package:file_picker/file_picker.dart';
import 'package:flutter/material.dart';
import
'package:b_im_q/desktop_screens/WelcomePage/background.dart';
import 'package:b_im_q/components/RoundedButton.dart';
import 'package:auto_size_text/auto_size_text.dart';

import 'package:open_file/open_file.dart';

class Body extends StatefulWidget {
  const Body({super.key});

  @override
  State<Body> createState() => _BodyState();
}

class _BodyState extends State<Body> {
  void _pickFile() async {

    // opens storage to pick files and the picked file or files
    // are assigned into result and if no file is chosen result
    is null.
    // you can also toggle "allowMultiple" true or false
    depending on your need
    FilePickerResult? result = await
    FilePicker.platform.pickFiles(
    type: FileType.custom,
    allowedExtensions: ['jpg', 'png'],
    );

    // if no file is picked
    if (result == null) return;

    // we get the file from result object
    final file = result.files.first;

    _openFile(file);
  }
}

```

```

void _openFile(PlatformFile file) {
  OpenFile.open(file.path);
}

@override
Widget build(BuildContext context) {
  //Future<Users?>? users;
  Size size = MediaQuery.of(context).size;
  //This size provide us total height and width of screen
  return Background(
    child: SingleChildScrollView(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          SizedBox(
            width: 300,
            height: 140,
            child: AutoSizeText("Remove haze from your images or videos",
              style: TextStyle(
                fontFamily: 'Montserrat',
                color: Colors.white,
                fontWeight: FontWeight.bold,
                fontSize: 36,
              ),
            ),
          ),
          SizedBox(height: size.height * 0.05),
          RoundedButtonClear(
            text: "Clear desk",
            press: () {
              _pickFile();
            },
          ),
          RoundedButtonClear(
            text: "Clear video",
            press: () {
              //
            },
          ),
        ],
      ),
    );
}

```

Файл «login_screen.dart»

```
import 'package:b_im_q/constants.dart';
```

					ПГУ1.090301.30.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		88

```

import
'package:b_im_q/desktop_screens/Login/components/body_desktop.da
rt';
import 'package:flutter/material.dart';

class LoginDesktop extends StatefulWidget {
  @override
  State<LoginDesktop> createState() => _LoginState();
}

class _LoginState extends State<LoginDesktop> {
  @override
  Widget build(BuildContext context) {
    Size size = MediaQuery.of(context).size;
    return Scaffold(body: body(),
      );
  }
}

```

Файл «body_login.dart»

```

import 'package:b_im_q/backend/users.dart';
import 'package:b_im_q/components/RoundedButton.dart';
import 'package:b_im_q/components/rounded_input_field.dart';

import 'package:b_im_q/components/text_filed_container.dart';
import 'package:b_im_q/constants.dart';
import
'package:b_im_q/desktop_screens/Login/components/background.dart
';
import
'package:b_im_q/desktop_screens/signUp/signUp_screen.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';

import '../.../backend/CRUD.dart';
import '../.../components/RememberMeCheck.dart';
import '../.../components/have_an_account_check.dart';
import '../.../components/rounded_password_field.dart';
import '../.../ResetPassword/ResetPwd_screen_desktop.dart';
import '../.../main_screen_desktop.dart';

class body extends StatefulWidget {
  const body({
    Key? key,
  }): super(key : key);

  @override
  State<body> createState() => _bodyState();
}

```

```

class _bodyState extends State<body> {
  TextEditingController loginController =
TextEditingController();
  TextEditingController pwdController = TextEditingController();
  bool isLoggedIn = false;

  @override
  Widget build(BuildContext context) {
    Size size = MediaQuery.of(context).size;

Users dataUser;
final DioClient _client = DioClient();

return Background(child: Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    Text("Welcome desk!",
      style: TextStyle(
        fontFamily: 'Montserrat',
        color: kPrimaryColor,
        fontWeight: FontWeight.bold,
        fontSize: 36,
      ),
    ),
    SizedBox(height: size.height * 0.07),
    RoundedInputField(

      hintText: "Your login",
      icon: Icons.person_2_outlined,
      controller: loginController,
    ),
    RoundedPasswordField(controller: pwdController,),
    Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        GestureDetector(
          onTap: () {Navigator.push(
            context, MaterialPageRoute(
              builder: (context){
                return ResetPwdDesktop();
              },),
            );},
        child: Text(
          "Forgot password?",
          style: TextStyle(color: kPrimaryColor, fontWeight:
            FontWeight.bold),
        ),
      ],
    ),
  ],
)

```



```

    ),
    RoundedButtonLogSign(text: "Log in", press: () async {
    var username = loginController.text;
    var pwd = pwdController.text;
    isLoggedIn = await _client.dioLogin(username: username, password:
    pwd);
    if(isLoggedIn == true){
        //сделать переброс на главную страницу
    String? result = await _client.DioGetUserData();
    if(kDebugMode){
    print(result);
    }
    Navigator.push(
    context, MaterialPageRoute(
    builder: (BuildContext context){
    return mainScreenDesktop();
    },),
    );
    }
    },
    ),
    SizedBox(height: size.height * 0.03),
    HaveAnAccountCheck(press: (){
    Navigator.push(
    context, MaterialPageRoute(
    builder: (context){
    return signUpScreenDesktop();
    },),
    );
    },)
    ],
    ));
    }
}

```

Файл «welcome_page.dart»

```

import 'package:file_picker/file_picker.dart';
import 'package:flutter/material.dart';
import
'package:b_im_q/desktop_screens/WelcomePage/background.dart';
import 'package:b_im_q/components/RoundedButton.dart';
import 'package:auto_size_text/auto_size_text.dart';
import 'package:open_file/open_file.dart';

class Body extends StatefulWidget {
  const Body({super.key});

  @override
  State<Body>createState() => _BodyState();
}

```

}

ПГУ1.090301.30.001 ПЗ

```

    ),
    SizedBox(height: size.height * 0.05),
    RoundedButtonClear(
      text: "Clear desk",
      press: () {
        _pickFile();
      },
    ),
    RoundedButtonClear(
      text: "Clear video",
      press: () {
        _pickFile();
      },
    ),
  ],
),
),
);
}
}

```

Файл «signUp_screen.dart»

```

import 'package:b_im_q/backend/CRUD.dart';
import 'package:b_im_q/components/RoundedButton.dart';
import 'package:b_im_q/components/have_an_account_check.dart';
import 'package:b_im_q/components/rounded_input_field.dart';
import 'package:b_im_q/components/rounded_password_field.dart';
import
'package:b_im_q/desktop_screens/Login/login_screen_desktop.dart'
;
import
'package:b_im_q/desktop_screens/signUp/components/backgroundSign
.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import '../../constants.dart';
import '../../backend/users.dart';
import 'package:email_validator/email_validator.dart';

class Body extends StatefulWidget {
  const Body({
    Key? key,
  }): super(key : key);

  @override
  State<Body>createState() => _BodyState();
}

class _BodyState extends State<Body> {

```

```

    TextEditingController loginController =
    TextEditingController();
    TextEditingController pwdController = TextEditingController();
    TextEditingController emailController =
    TextEditingController();
    String _errorMessage = '';

    @override
    Widget build(BuildContext context) {
        Size size = MediaQuery.of(context).size;
        Future<Users> users;
        final DioClient _client = DioClient();

        return BackgroundSignUp(child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
                SizedBox(height: size.height * 0.1),
                Text("Create Account",
                    style: TextStyle(
                        fontFamily: 'Montserrat',
                        color: kPrimaryColor,
                        fontWeight: FontWeight.bold,
                        fontSize: 36,
                    ),
                ),
                Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: Text(_errorMessage, style: TextStyle(color: Colors.red)),
                ),
                RoundedInputField(
                    hintText: "Your login",
                    icon : Icons.person_2_outlined,
                    controller: loginController,
                ),
                RoundedInputField(
                    hintText: "Your email",
                    icon: Icons.email_outlined,
                    controller: emailController),
                RoundedPasswordField(controller: pwdController,),
                RoundedButtonLogSign(text: "Sign up", press: () async{
                    var username = loginController.text;
                    var email = emailController.text;
                    var password = pwdController.text;
                    validateEmail(email);
                    await _client.signUpUser(username: username, email: email,
                        password: password).then((value){
                        Navigator.push(context, MaterialPageRoute(
                            builder: (context){return LoginDesktop();}
                        ));
                    }).catchError((err) {_errorMessage = err;});
                })
            ],
        ));
    }

```

```

if(kDebugMode){
print("username: "+ username + "\temail: "+ email + "\tpassword:
"+ password);
    }
    }),
HaveAnAccountCheck(login: false, press: (){
Navigator.push(context, MaterialPageRoute(
builder: (context){return LoginDesktop();}
    ));
    })
    ],
    ),);
}
String validateEmail(String val) {
if(val.isEmpty){
    _errorMessage = "Email can not be empty";
}else if(!EmailValidator.validate(val, true)){
    _errorMessage = "Invalid Email Address";
}else{
    _errorMessage = "";
    }
return _errorMessage;
}
}

```

Файл «CRUD.dart»

```

import
'package:b_im_q/backend/interceptors/dio_interceptor.dart';
import 'package:b_im_q/backend/service/DioExceptions.dart';
import 'package:b_im_q/backend/service/Store.dart';
import 'package:b_im_q/backend/users.dart';
import 'package:flutter/foundation.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'package:dio/dio.dart';

class DioClient {
    String _baseUrl = "http://192.168.1.77:10012/";
    late final Dio _dio;

    DioClient(){
        _dio = Dio(
        BaseOptions(
        connectTimeout: const Duration(seconds:10), //10 sec
        receiveTimeout: const Duration(seconds: 5), //5 sec
        // headers: {
        //     'Accept': 'application/json',
        // },
        ),
    ),

```

```

    );
    _dio.interceptors.add(DioInterceptor());
}

//GET Api Request
Future<void> _saveToken(Map<String, dynamic> data) async{
final access_token = data['access_token'];
await Store.setToken(access_token);
}

Future<bool>dioLogin({required String username,required String
password}) async{

final response = await _dio.post('${_baseUrl}token',
data: FormData.fromMap({
    'username': username,
    'password': password,
    // Add any other data you want to send in the body
}),
);
if(response.statusCode == 200){
await _saveToken(response.data);
return true;
}
return false;
}

Future<String?>DioGetUserData() async{
    //_dio.interceptors.add(DioInterceptor());
try{
final response = await _dio.get('${_baseUrl}users/me/');
if(response.statusCode == 200){
if (kDebugMode) {
print(response.data);
}
return response.data['username'];
}
} on DioError catch (e){
final errorMessage = DioExceptions.fromDioError(e).toString();
throw errorMessage;
}
}

Future<Users?>signUpUser({required String username,required
String email, required String password}) async{
Users retrieveUser;
final bool is_active = true;
try {
    Response response = await _dio.post("${_baseUrl}users/",
data: jsonEncode(<String, dynamic>{

```

```

        'username': username,
        'email': email,
        'is_active': is_active,
        'password': password,
        // Add any other data you want to send in the body
    )),
    );
if (response.statusCode == 200) {
    // Successful POST request, handle the response here
    print('User created: ${response.data}');
    retrieveUser = Users.fromJson(response.data);
    return retrieveUser;
}
}on DioError catch (e) {
    final errorMessage = DioExceptions.fromDioError(e).toString();
    throw errorMessage;
}
}
}

// POST Api Request
Future<String?>createUser(String username, String email, String
password) async {
    try {
        final response = await http.post(
        Uri.parse('http://192.168.1.77:10012/users/'),
        body: jsonEncode(<String, dynamic>{
            'username': username,
            'email': email,
            'password': password,
            // Add any other data you want to send in the body
        })),
        );
if (response.statusCode == 200) {
    // Successful POST request, handle the response here
    final responseData = jsonDecode(response.body);
    return 'ID: ${responseData['id']}\nUsername:
    ${responseData['username']}\nEmail:
    ${responseData['email']}\nPassword:
    ${responseData['password']}'
    }
    } catch (e) {
        throw('Error: $e');
    }
}

//Update Api Users
Future<Users>updateUsers(String username, String email, String
password) async{
    Map<String, dynamic> request = {

```

```

        'username' : username,
        'email' : email,
        'password' : password,
        'id' : "111"
    };

    final uri =
    Uri.parse("https://jsonplaceholder.typicode.com/users/1");
    final response = await http.put(uri, body: request);

    if (response.statusCode == 200){
    return Users.fromJson(json.decode(response.body));
    }
    else{
    throw Exception('Failed to load user data');
    }
}

```

Файл «users.dart»

```

import 'package:json_annotation/json_annotation.dart';

@JsonSerializable()
class Users{
    int? id;
    final String username;
    final String email;
    final bool is_active;
    final String password;

    Users({
        this.id,
        required this.username,
        required this.email,
        required this.is_active,
        required this.password,
    });

    factory Users.fromJson(Map<String,dynamic> json) => Users(
    id: json['id'],
    username: json['username'],
    email: json['email'],
        is_active: json['is_active'],
    password: json['password'],
    );

    Map<String, dynamic>toJson() =>{
        "id": id,
        "username": username,
        "email": email,
    }
}

```



```

        "is_active": is_active,
        "password": password,
    };
}

```

Файл «Store.dart»

```

import 'package:shared_preferences/shared_preferences.dart';

class Store{
  const Store._();

  static const String _tokenKey = "access_token";

  static Future<void> setToken(String access_token) async{
    final preferences = await SharedPreferences.getInstance();

    await preferences.setString(_tokenKey, access_token);
  }

  static Future<String?> getToken() async{
    final preferences = await SharedPreferences.getInstance();

    return preferences.getString(_tokenKey);
  }

  static Future<void> clear() async{
    final preferences = await SharedPreferences.getInstance();
    await preferences.clear();
  }
}

```

Файл «DioException.dart»

```

import 'package:dio/dio.dart';

class DioExceptions implements Exception {
  late String message;

  DioExceptions.fromDioError(DioException dioError) {
    switch (dioError.type) {
      case DioExceptionType.cancel:
        message = "Request to API server was cancelled";
        break;
      case DioExceptionType.connectionTimeout:
        message = "Connection timeout with API server";
        break;
      case DioExceptionType.receiveTimeout:
        message = "Receive timeout in connection with API server";
        break;
      case DioExceptionType.badResponse:

```

```

message = _handleError(
dioError.response?.statusCode,
dioError.response?.data,
    );
break;
case DioExceptionType.sendTimeout:
message = "Send timeout in connection with API server";
break;
case DioExceptionType.unknown:
if (dioError.message!.contains("SocketException")) {
message = 'No Internet';
break;
    }
message = "Unexpected error occurred";
break;
default:
message = "Something went wrong";
break;
    }
}

String _handleError(int? statusCode, dynamic error) {
switch (statusCode) {
case 400:
return 'Email already registered';
case 401:
return 'Unauthorized';
case 403:
return 'Forbidden';
case 404:
return 'User not found';
case 500:
return 'Internal server error';
case 502:
return 'Bad gateway';
default:
return 'Oops something went wrong';
    }
}

@override
String toString() => message;
}

```

Файл «dio_interceptor.dart»

```

import 'package:dio/dio.dart';
import 'package:b_im_q/backend/service/Store.dart';

class DioInterceptor extends Interceptor{
  @override

```

```

    Future<void>onRequest(RequestOptions options,
RequestInterceptorHandler handler) async{

    final access_token = await Store.getToken();
    if(access_token!= null && access_token.isNotEmpty){
    options.headers['Authorization'] = 'Bearer $access_token';
    }
    options.headers['Content-Type'] = 'application/json';

    super.onRequest(options, handler);
    }
}

```

					ПГУ1.090301.30.001 ПЗ	Лист
						101
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(Обязательное)

Стартап-проект CLView. Сервис для улучшения качества изображений с
использованием нейронной сети

Презентация

					ПГУ1.090301.30.001 ПЗ	Лист
						102
Изм.	Лист	№ докум.	Подпись	Дата		

CLView. Сервис для улучшения качества изображений с использованием нейронной сети

Докладчик:
студентка 20ВВП2
Кожевникова Алина

2

Актуальность разработки

Сервисы по улучшению качества изображений с использованием нейронных сетей представляют значимый научный и практический интерес. Они находят применение в различных областях человеческой деятельности и компьютерного зрения.



01

Безопасность и
видеонаблюдение

02

Автономное
вождение

03

Фотографии и
видеомонтаж

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.30.001 ПЗ

Лист

103

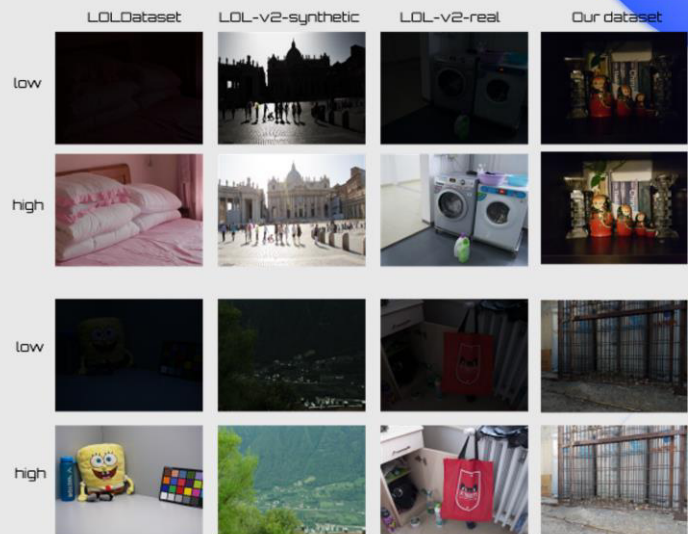


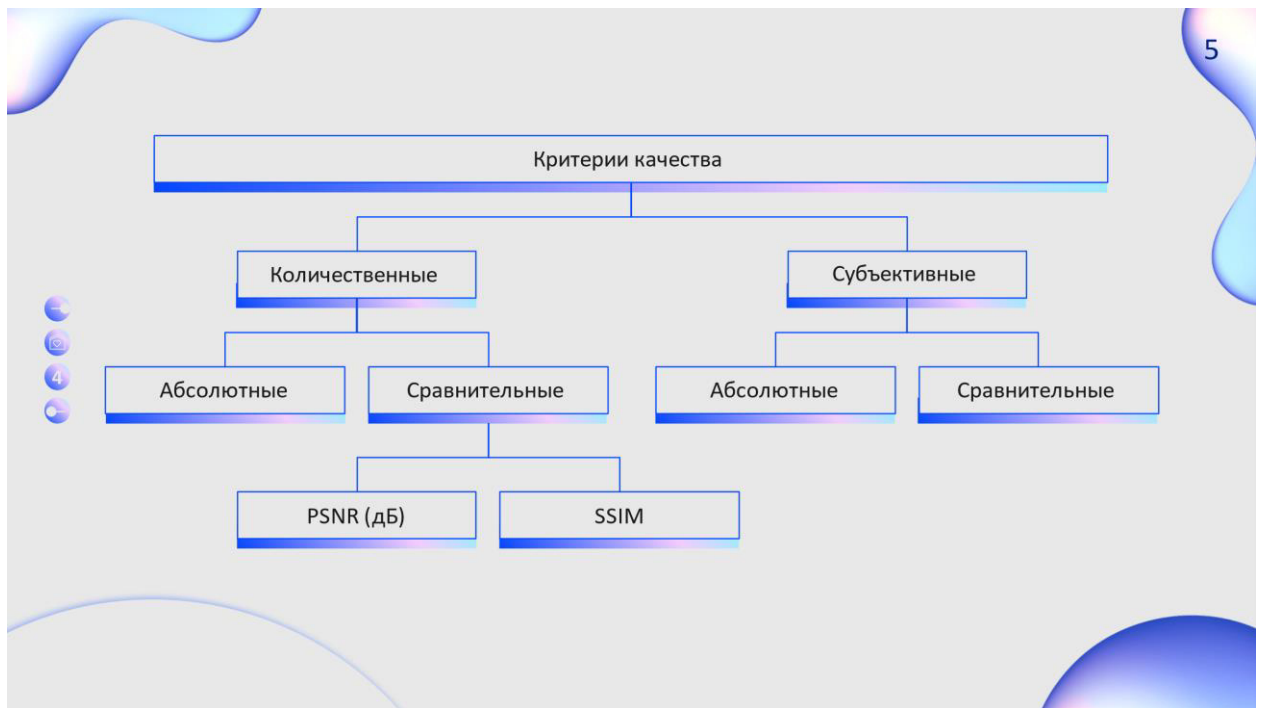
База данных для модели, осуществляющей фильтрацию изображений

- I-HAZE: 35 пар изображений
- O-HAZE: 45 пар изображений
- Dense-Haze CVPR: 130 пар изображений
- SMOKE: 122 пары изображений
- Fog Cityscape: 4500 пар изображений

База данных для модели, осуществляющей осветление изображений

- LOLDataset : 500 пар изображений
- LOLv2-real : 800 пар изображений
- LOLv2-synthetic : 1000 пар изображений





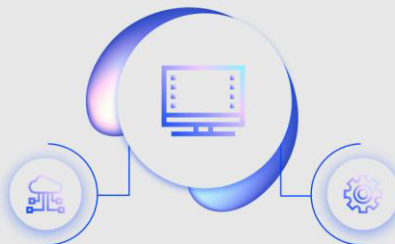
Исследование архитектур нейронных сетей, используемых для повышения качества изображений!

Методы поляризации изображений, на которых присутствуют туманы и задымления



DNN

Чаще всего используют свёрточные нейронные сети или их комбинации с другими подходами



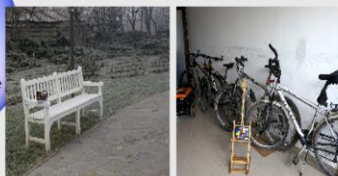
Традиционные

Чаще всего основаны на эмпирических посылках и опираются на методы преобразования яркости изображений или применение пространственной и частотной фильтрации

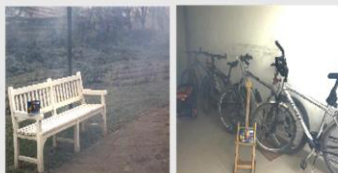
Enhanced Pix2pix Dehazing Network

LCA-Net

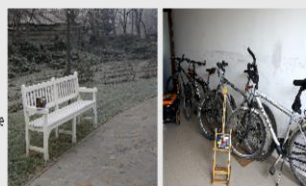
Эталонные
данные



Выходные
данные



Эталонные
данные

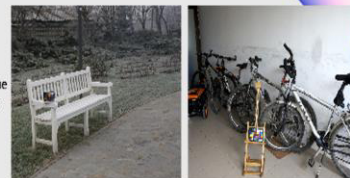


Выходные
данные

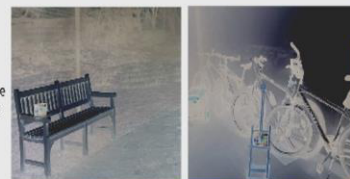


Image Dehazing Transformer

Эталонные
данные



Выходные
данные



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.30.001 ПЗ

Лист

106

Исследование архитектур нейронных сетей, осветляющих изображения



Основана на устранении локальной неоднородности освещенности объектов наблюдаемой сцены и работает по принципу локального увеличения контраста. Слабо освещенные объекты при этом становятся лучше различимыми, восстанавливаются их детали и цвет. Эта технология предназначена в первую очередь для обработки изображений, содержащих малоконтрастные (тёмные либо светлые) зоны, в которых локальные детали обычно трудноразличимы

Enhanced Pix2pix Dehazing Network

LCA-Net



Эталонные
данные

Выходные
данные

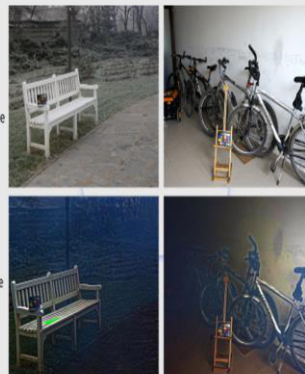
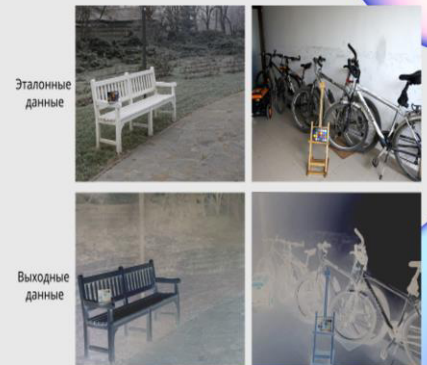


Image Dehazing Transformer



Эталонные
данные

Выходные
данные

Изм.	Лист	№ докум.	Подпись	Дата

Исследование архитектур нейронных сетей, осветляющих изображения

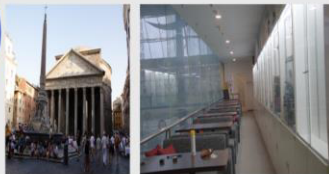


Основана на устранении локальной неоднородности освещенности объектов наблюдаемой сцены и работает по принципу локального увеличения контраста. Слабо освещенные объекты при этом становятся лучше различимыми, восстанавливаются их детали и цвет. Эта технология предназначена в первую очередь для обработки изображений, содержащих малоконтрастные (тёмные либо светлые) зоны, в которых локальные детали обычно трудноразличимы

Retinex-Net

cycleGAN

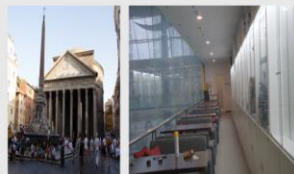
Эталонные
данные



Выходные
данные



Эталонные
данные

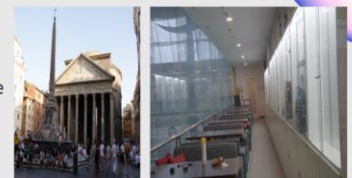


Выходные
данные

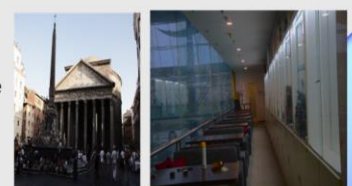


The Low-Light Net (LLNet)

Эталонные
данные



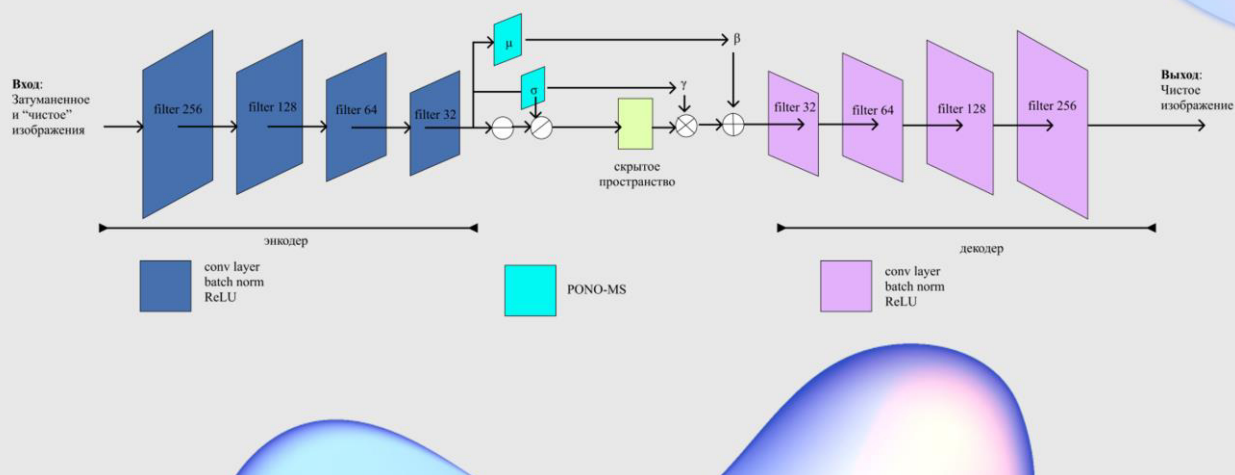
Выходные
данные



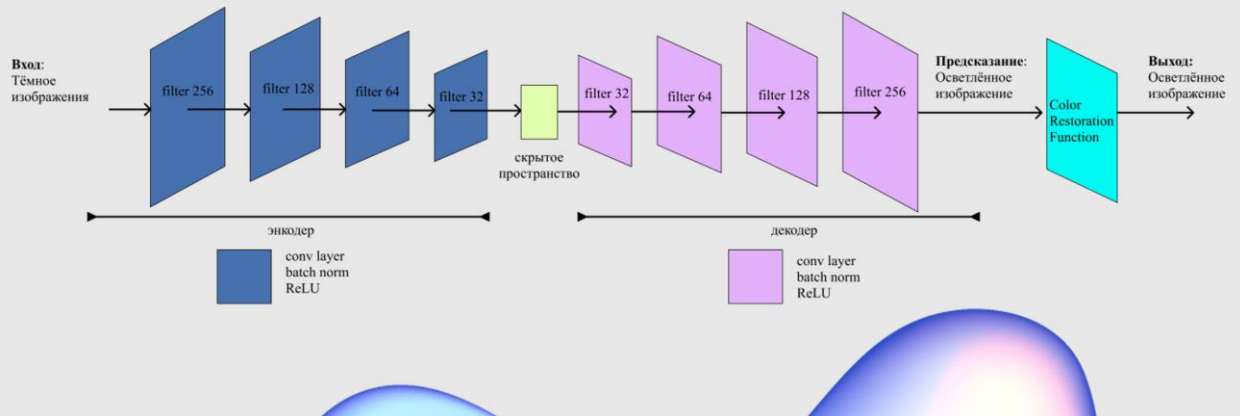
Изм.	Лист	№ докум.	Подпись	Дата

РАЗРАБОТКА АРХИТЕКТУРЫ НЕЙРОННОЙ СЕТИ ДЛЯ ФИЛЬТРАЦИИ ЗАДЫМЛЕНИЙ И «ЗАМЫЛЕННОСТИ», А ТАКЖЕ ОСВЕТЛЕНИЯ ТЁМНЫХ УЧАСТКОВ ИЗОБРАЖЕНИЙ

Архитектура модели для фильтрации задымлений и «замыленности»



Разработка модели для осветления тёмных участков изображений



Бизнес - план

Изм.	Лист	№ докум.	Подпись	Дата

Потенциальная аудитория

1

Компании, занимающиеся разработкой фото- и видео- редакторов.

Более 70 крупных компаний в мире.

2

Компании, занимающиеся разработкой и производством систем автопилотирования.

Более 500 крупных компаний в мире.

3

Компании, занимающиеся разработкой и изготовлением систем видеонаблюдения.

Более 800 компаний в мире.

4

Физические лица.

Более 1 млрд. физических лиц – пользователи фото и видео редакторов.

Перспективы коммерциализации

Реализация по двум направлениям

Программный модуль с закрытым исходным кодом. Функционал адаптирован под готовые коммерческие решения, доступ к технологии будет платным.

Ежегодный рост рынка систем машинного зрения (автопилотирования и видеонаблюдения)

55,7%

Веб-приложение для обработки единичных или пакетных изображений.

По прогнозам, объём мирового рынка машинного зрения к 2028 году достигнет

95,563 млрд. \$

Примерная тарификация услуг

Назначение	Лимит	Абон.плата	Сверх лимита
Для компаний, производящих фото- и видео- редакторы	Нет	8000 ₽/год	Нет
Для компаний, производящих системы автопилотирования	946 080 000 изображений	20000 ₽/год	10000 ₽/год
Для компаний, производящих системы видеонаблюдения	946 080 000 изображений	20000 ₽/год	10000 ₽/год
Для физических лиц	77 760 000 изображений	100 ₽/мес.	Нет

Спасибо за
внимание

Готова ответить на Ваши вопросы

