

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Онлайн-платформа для продвижения
произведений цифрового искусства**

Студент _____ Бочкарева Алёна Александровна
(подпись, дата) (ФИО полностью)

Руководитель _____ Гурьянов Л.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 03.06.2026г. № 12)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов «26»
декабря 2025 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Бочкарева Алёна Александровна гр. 22ВП2 факультета ВТ

<Фамилия, имя, отчество полностью>

2. Тема работы Онлайн-платформа для продвижения произведений цифрового искусства

Тема утверждена приказом ПГУ № 1583/о от "30" декабря 2025г.

2. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Гурьянов Л.В.

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

1) *Назначение: онлайн-платформа для продвижения произведений цифрового искусства с механизмом автоматической проверки контента на ИИ-генерацию*

2) *Среда выполнения: Windows 11, веб-браузер.*

Языковые средства разработки: Java, Python, TypeScript, SQL, UML.

Инструментальные средства: IntelliJ IDEA, VS Code, Docker, PostgreSQL, MinIO, Apache Kafka, Draw.io, Project Libre.

Технологии разработки: объектно-ориентированная парадигма, микросервисная архитектура, событийно-ориентированная архитектура.

3) *Основные функции приложения:*

а) *регистрация и авторизация;*

б) *публикация цифровых проектов;*

в) *автоматическая проверка загружаемых изображений на генерацию;*

г) *маркировка цифровых проектов, созданных с использованием генеративного ИИ;*

- д) фильтрация III-контента в пользовательской ленте;
- е) верификация аккаунтов пользователей;
- ж) просмотр ленты цифровых проектов и профилей цифровых креаторов;
- з) социальное взаимодействие: подписки на авторов, лайки и комментарии к цифровым проектам;
- и) настройка профиля пользователя.

4. Перечень подлежащих разработке вопросов:

- 1) Анализ предметной области и постановка задачи;
- 2) Проектирование;
- 3) Реализация;
- 4) Контроль качества программного обеспечения;
- 5) Оформление пояснительной записки.

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20%	14.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20%	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20%	22.05.2026 – 02.06.2026	

Дата выдачи задания «26» декабря 2025г.

Руководитель бакалаврской работы _____ 26.12.2025 г.
(подпись, дата)

Задание к исполнению принял студент _____ 26.12.2025 г.
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 110 листов, 51 рисунок, 31 таблица, 22 использованных источника и 9 приложений.

ЦИФРОВОЕ ИСКУССТВО, ЦИФРОВОЙ КРЕАТОР, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, МИКРОСЕРВИСНАЯ АРХИТЕКТУРА, РЕЛЯЦИОННАЯ БД, JAVA, SPRING BOOT, PYTHON, NEXT.JS.

Объект исследования – процесс продвижения произведений цифрового искусства в условиях популяризации генеративного искусственного интеллекта.

Цель выпускной квалификационной работы – проектирование и разработка онлайн-платформы для продвижения произведений цифрового искусства с механизмами обнаружения ИИ-контента и подтверждением авторства.

Технологии разработки – разработка серверной части выполнялась в среде IntelliJ IDEA на языке Java с использованием фреймворка Spring Boot на базе микросервисной архитектуры, с использованием брокера Apache Kafka, реляционной СУБД PostgreSQL и объектного хранилища MinIO. Сервис обнаружения ИИ-контента реализован на языке Python с использованием фреймворка FastAPI и модели машинного обучения. Серверная часть контейнеризирована при помощи Docker. Клиентская часть реализована на TypeScript и Next.js. Моделирование бизнес-процессов выполнено с использованием UML.

Среда исполнения – Windows 11, веб-браузер.

Результаты работы – были спроектирована и разработана онлайн-платформа для продвижения произведений цифрового искусства, включающая механику автоматической проверки изображений на предмет ИИ-генерации и верификации аккаунтов

					ПГУ 09.03.04 – 08BP222.03 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Бочкарева А.А.			Онлайн-платформа для продвижения произведений цифрового искусства. Пояснительная записка	Лит.	Лист	Листов
Пров.		Гурьянов Л.В.					4	110
Реценз.						Группа 22ВП2		
Н. Контр.		Такташкин Д.В.						
Утв.		Козлов А.Ю.						

Содержание

Введение.....	8
1 Анализ предметной области и планирование разработки онлайн-платформы для продвижения произведений цифрового искусства.....	11
1.1 Анализ предметной области разработки	11
1.1.1 Бизнес-процессы в сфере цифрового искусства.....	11
1.1.2 Предварительные требования к системе	13
1.2 Сравнительный анализ систем-аналогов.....	14
1.2.1 Обоснование сравнительного анализа.....	14
1.2.2 Обзор программных платформ для анализа.....	14
1.2.3 Результаты сравнительного анализа и выводы.....	17
1.3 Постановка задачи на разработку онлайн-платформы	19
1.3.1 Классы пользователей онлайн-платформы	19
1.3.2 Планирование и оценка бюджета процесса разработки онлайн-платформы.....	19
2 Проектирование онлайн-платформы для продвижения произведений цифрового искусства.....	23
2.1 Анализ функциональных требований	23
2.2 Анализ нефункциональных требований	27
2.3 Анализ требований к архитектуре программных средств.....	28
2.4 Анализ требований к данным	31
2.5 Анализ требований к пользовательскому интерфейсу	33
2.6 Анализ требований к аппаратно-программному обеспечению	34
2.6.1 Требования к аппаратному обеспечению.....	34
2.6.2 Требования к системному программному обеспечению	35
2.6.3 Требования к средствам разработки	35

	6
3 Разработка программного обеспечения онлайн-платформы.....	37
3.1 Развертывание физической схемы базы данных	37
3.2 Реализация серверного программного обеспечения	39
3.3 Реализация клиентского программного обеспечения	45
3.4 Реализация интерфейса приложения	47
4 Контроль качества программных средств	54
4.1 Планирование процесса контроля качества программных средств	54
4.1.1 Обоснование выбора модели тестирования.....	54
4.1.2 Обоснование выбора метрик программного кода	55
4.2 Тестирование программных средств	56
4.2.1 Функциональное тестирование	56
4.2.2 Модульное тестирование	58
4.3 Оценка качества программного кода	60
4.3.1 Метрика SLOC	60
4.3.2 Метрика Холстеда.....	60
4.3.3 Метрика Чепина	62
Заключение	64
Список использованных источников	66
Приложение А (справочное) Глоссарий	69
Приложение Б (рекомендуемое) Расчет бюджета проекта.....	73
Приложение В (рекомендуемое) Спецификация атрибутов таблиц БД	76
Приложение Г (рекомендуемое) SQL-скрипты создания БД.....	84
Приложение Д (обязательное) Листинг микросервиса «Медиафайлы»	88
Приложение Е (рекомендуемое) Программные компоненты серверной части приложения	97

Приложение Ж (рекомендуемое) Программные компоненты клиентской части приложения	102
Приложение И (рекомендуемое) Результаты тестирования.....	104
Приложение К (рекомендуемое) Листинг юнит-тестов.....	111

Введение

В современном мире всё больше произведений искусства создаются с использованием цифровых технологий. По данным Министерства экономического развития Российской Федерации на 2025 год средняя доля креативных индустрий в мировой экономике приближается к 6,6% [1]. Наряду с направлениями архитектуры, моды и книжного дела в креативный кластер также входит арт-индустрия. Арт-индустрия составляет пятую часть от данного кластера, что делает её важной составляющей не только рынка, но и повседневной жизни людей [2].

Во время локдауна из-за пандемии COVID-19 люди стали активнее пользоваться интернетом и интересоваться онлайн-профессиями, в том числе касающихся диджитал-искусства. По данным ИСИЭЗ НИУ ВШЭ, в 2024 году в креативной среде России было задействовано свыше 4,6 млн чел или 6,2% от общей численности занятых граждан страны [3].

В таких сферах, как дизайн, реклама и видеоигры традиционная форма искусства уступает цифровому. Происходит это из-за более удобной интеграции информационных технологий в производственный процесс, возможности неограниченного тиражирования конечного продукта и упрощения механизмов командной работы. При всех удобствах создания продуктов в диджитал формате, цифровые креаторы сталкиваются с рядом трудностей, касающихся защиты авторского права и продвижения своих работ на подходящих для этой цели платформах. В процессе продвижения своего творчества на площадках, пользователи могут сталкиваться не только с воровством и присваиванием авторства своих работ другими пользователями, но и с кражей творческого стиля человека. Также им приходится конкурировать на творческих платформах с нейрохудожниками, способными генерировать в разы больше контента ежедневно, вытесняя креаторов из ленты.

Большинство известных деятелей искусства имеют свой уникальный стиль, нравящийся их аудитории и способный привлечь потенциальных клиентов [4]. В то же время, нейросети обучаются на миллионах авторских работ.

Например, организация Meta официально заявила, что будет использовать посты на платформах, таких как Facebook и Instagram для обучения собственных ИИ-моделей. Это вызвало всплеск общественного недовольства, многие креаторы покинули площадки в пользу анти-ИИ платформы Cara, но подавляющее большинство осталось [5]. Набирать аудиторию, в особенности платёжеспособную, из-за возрастающей конкуренции с каждым годом становится все более сложной задачей, а основным заработком цифровых креаторов остаётся выполнение заказов.

Создатели приложения Glaze, разработанной для защиты работ художников от копирования стиля нейросетями, в своей статье пишут о том, насколько сильное негативное влияние генеративные модели ИИ оказывают на арт-индустрию. На сегодняшний день существуют модели, обучающиеся на онлайн-галереях работ, таких как ArtStation, Pinterest и DeviantArt, способные воссоздать чужой стиль по набору работ целевого художника [6].

Столь быстрое развитие генеративных ИИ также приносит изменения в опыт конечных пользователей творческих онлайн-платформ. От изобилия «машинного» контента многие пользователи разочаровываются в интернет-пространствах, потворствующих нейрокреаторам, а покупка цифрового продукта может обернуться неожиданными последствиями. Человек, плативший за чужой многолетний опыт и время, не хочет получить картинку, созданную нажатием одной кнопки за несколько минут.

Таким образом, существует необходимость в создании новой онлайн-платформы для цифровых креаторов, обеспечивающей продвижение работ, выполненных человеком, и препятствовать развитию аккаунтов нейрокреаторов. Она должна предоставлять защиту публикуемого авторского контента от обучения нейросетями и гарантировать подлинность самих креаторов.

Разрабатываемая в рамках данной дипломной работы платформа направлена на удовлетворение потребностей как внешней, так и внутренней аудитории арт-сообщества, предлагая инструментарий для верификации

аккаунтов, проверки изображений на генерацию и алгоритмы для маркировки контента.

Целью выпускной квалификационной работы является разработка программных средств для продвижения произведений цифрового искусства. Для достижения поставленной цели были поставлены следующие задачи:

- выполнить сравнительный анализ существующих онлайн-платформ для продвижения произведений цифрового искусства;
- осуществить анализ требований к разрабатываемому программному продукту;
- провести планирование бюджета на разработку программных средств;
- выполнить проектирование программных средств и структур данных разрабатываемого программного продукта;
- спроектировать базу данных для онлайн-платформы;
- разработать пользовательский интерфейс онлайн-платформы;
- реализовать серверную и клиентскую части онлайн-платформы;
- провести контроль качества разработанных программных средств.

Кроме того, в ходе выполнения выпускной квалификационной работы также требуется оформить пояснительную записку в соответствии с требованиями.

1 Анализ предметной области и планирование разработки онлайн-платформы для продвижения произведений цифрового искусства

1.1 Анализ предметной области разработки

1.1.1 Бизнес-процессы в сфере цифрового искусства

Цифровое искусство – это художественные произведения или практики, неотъемлемой частью которых является применение в той или иной степени информационных технологий [7]. В отличие от традиционной формы искусства, проявляющейся в живописи или скульптуре, в процессе создания диджитал-арта задействованы не холсты, краски или глина, а специализированное аппаратное и программное обеспечение. Термин «цифровое искусство» охватывает множество направлений: от 3D-моделирования и анимации до проектов с использованием технологий виртуальной реальности (Virtual Reality, VR) или дополненной реальности (Augmented Reality, AR).

Цифровой проект – это единица публикуемого контента на платформе, представляющая собой структурированную подборку медиа-материалов, объединённых общим творческим замыслом автора. Цифровой проект может включать одно или несколько изображений, а также название и сопроводительное текстовое описание.

Цифровые креаторы – это творческие люди, создающие произведения цифрового искусства и распространяющие свое творчество на специализированных платформах и социальных сетях для привлечения аудитории, в том числе потенциальных заказчиков. Они существуют в онлайн-среде и напрямую зависят от алгоритмов площадок, на которых они размещают свои работы.

Конечных потребителей контента на платформах для продвижения цифрового искусства можно разделить на две категории: платёжеспособную и неплатёжеспособную. В лице платёжеспособной аудитории выступают как частные лица, ищущие исполнителей индивидуальных заказов, так и бренды и компании, находящиеся в поиске новых сотрудников или работников на

аутсорсе. Неплатежеспособная аудитория вышеупомянутых платформ – ценители цифрового искусства, распространяющие творчество любимых авторов с помощью такого базового функционала социальных сетей, как лайки, подписки и комментарии. Представителей второй категории пользователей превалируют над первой, но именно их активность способствует росту аккаунтов креаторов, что, в свою очередь, привлекает внимание заказчиков.

Многие платформы предоставляют функционал усиления видимости проектов определённых аккаунтов посредством платного улучшения статуса профиля пользователя до «Про», что является одним из основных источников дохода подобных систем.

Генеративный искусственный интеллект (ГИИ) – ветвление искусственного интеллекта (ИИ), модели которого разрабатываются для генерации контента, основанного на датасетах для их обучения. На сегодняшний день результаты работы моделей ГИИ часто можно заметить в социальных сетях в виде сгенерированных изображений, текстов и музыки, часто трудноотличимых от рукотворных творений. Творческие платформы также не отстают от современных трендов и создают собственные модели ГИИ и не препятствуют распространению сгенерированного контента, чем разочаровывают арт-сообщество [8].

На сегодняшний день существует несколько решений по защите изображений от обучения нейросетями:

- отказ от включения в обучающие данные с помощью метаданных;
- отравление данных.

Метаданные можно отнести к ненадёжным средствам защиты, поскольку при изменении или передаче контента они могут быть утрачены, что повышает риск недобросовестного использования генерируемого контента. Способ отравления данных, в свою очередь, изменяет изображения на уровне пикселей, что затрудняет распознавание модификации со стороны человека. Данный метод наиболее эффективен при обработке высокодетализированных изображений, так как на минималистичных работах проще всего убрать подобные вмешательства

сглаживающими эффектами по типу размытия по Гауссу. Защита от ИИ-контента на арт-платформах чаще всего внедряется в их монетизируемую составляющую, в частности в разделы с пользовательскими товарами.

В правовом поле вопрос маркировки ИИ-контента на мировом уровне решается неравномерно. В рамках Закона об искусственном интеллекте ЕС (EU AI Act) провайдеры ИИ-моделей с ограниченным риском обязаны раскрывать пользователям факт генерации контента с помощью ИИ, а также принимать меры по предотвращению создания незаконного контента [9]. Наиболее жесткую систему внедрил Китай, обязав платформы маркировать любой ИИ-контент как визуально, так и в метаданных. В Российской Федерации соответствующий законопроект был предложен в 2026 году, однако отклонён в связи с недостаточностью финансирования разработки технологии [10].

Изложенное свидетельствует о том, что ответственность за реализацию механизмов маркировки и фильтрации ИИ-контента в настоящее время во многом возлагается непосредственно на платформы, где он публикуется.

1.1.2 Предварительные требования к системе

Проведя анализ аудитории и бизнес-процессов творческих онлайн-платформ, можно сделать вывод, что разрабатываемая система должна предоставлять функционал, удовлетворяющий потребностям как внешней части аудитории креативного сообщества, так и внутренней [11]. Результат работы должен охватывать следующие аспекты:

- верификацию пользовательских аккаунтов, включающую подтверждение авторства публикуемых работ;
- улучшение статуса аккаунта пользователя;
- проверку загружаемого на платформу контента на факт генерации;
- маркировку и отсутствие видимости в главной ленте ИИ-контента;
- функционал управления профилем и портфолио;
- функционал социальных сетей (лайки, комментарии, подписки, уведомления).

Стоит отметить, что разрабатываемый функционал должен быть реализован с учетом масштабируемости архитектуры, что позволит добавлять новые возможности взаимодействия пользователей по мере развития платформы.

1.2 Сравнительный анализ систем-аналогов

1.2.1 Обоснование сравнительного анализа

На рынке существует множество различных платформ для размещения своих работ творческими людьми: от любительских до профессиональных. Также их можно разделять по таким критериям, как язык интерфейса платформы, преобладающий вид публикуемых произведений искусства, отношение к ИИ-контенту и т. п.

Целью проведения сравнительного анализа платформ аналогичного назначения является нахождение сильных и слабых сторон существующих решений, определение незакрытых потребностей внутренней и внешней аудитории, актуальных на данный момент и выявление необходимости разработки собственного программного продукта.

Для проведения сравнительного анализа из множества арт-платформ были выбраны такие, как «ArtStation», «Cara» и «Иллюстраторс.ру». Они объединены ориентацией на цифровую форму искусства, возможностью размещения работ креаторами и наличием базового функционала социальных сетей.

1.2.2 Обзор программных платформ для анализа

1.2.2.1 Платформа «ArtStation»

Платформа «ArtStation» является одной из крупнейших платформ для цифровых креаторов и ценителей цифрового искусства. Она предназначена для публикации портфолио профессиональными цифровыми креаторами, работающими в игровой индустрии, кино, анимации и дизайне.

На рисунке 1 представлена домашняя страница платформы «ArtStation».

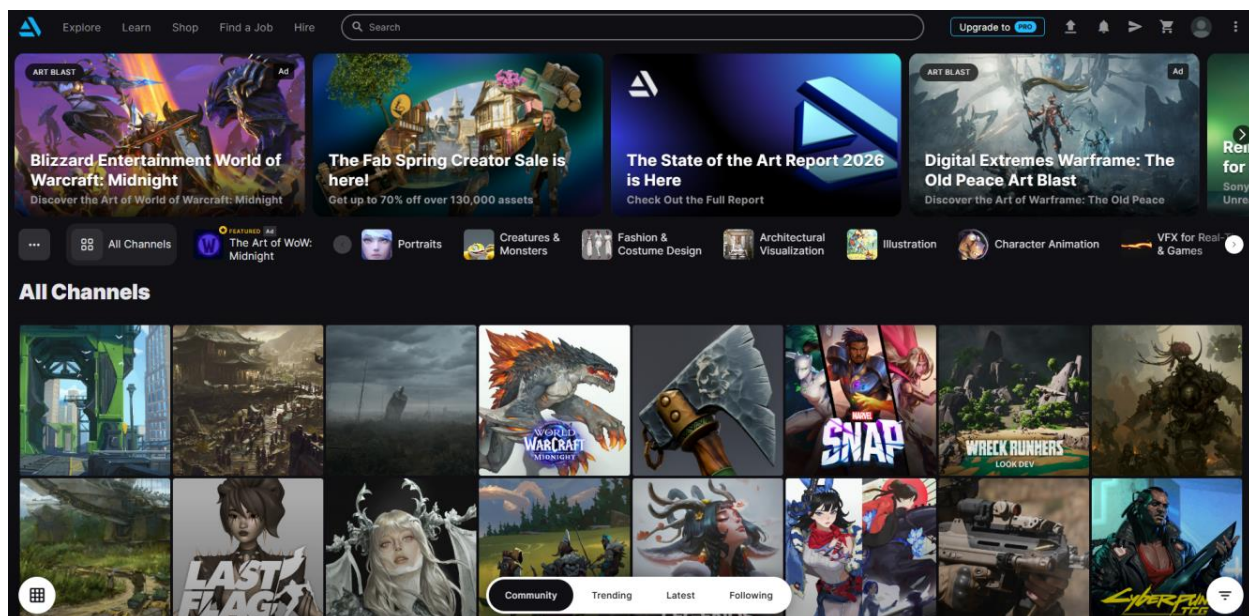


Рисунок 1 – Домашняя страница «ArtStation»

Платформа поддерживает такие функции социальных сетей, как лайки, подписки, реакции, комментарии, добавление в «избранное» понравившихся работ. Также она предоставляет гибкую настройку профиля с указанием личной информации и контактов креатора и чаты для личного взаимодействия с ними. На площадке есть свой маркетплейс с пользовательскими товарами и функционал как для поиска работы, так и для поиска сотрудников.

Несмотря на все преимущества и обширный функционал, «ArtStation» часто подвергается критике из-за отсутствия запрета ИИ-контента в публикациях, но присутствие на товарах магазина. В 2024 году при публикации контента появилась возможность добавить одноимённый HTML-метатегу тег «NoAI», запрещающий использование изображений ИИ-моделями [12].

1.2.2.2 Платформа «Cara»

«Cara» – это социальная сеть и платформа, открыто выступающая против сгенерированного контента и его создателей [13]. Она получила широкую известность в арт-сообществе, привлекая пользователей своей четкой анти-ИИ политикой, из-за чего стремительно набирает аудиторию начиная с запуска в 2023 году.

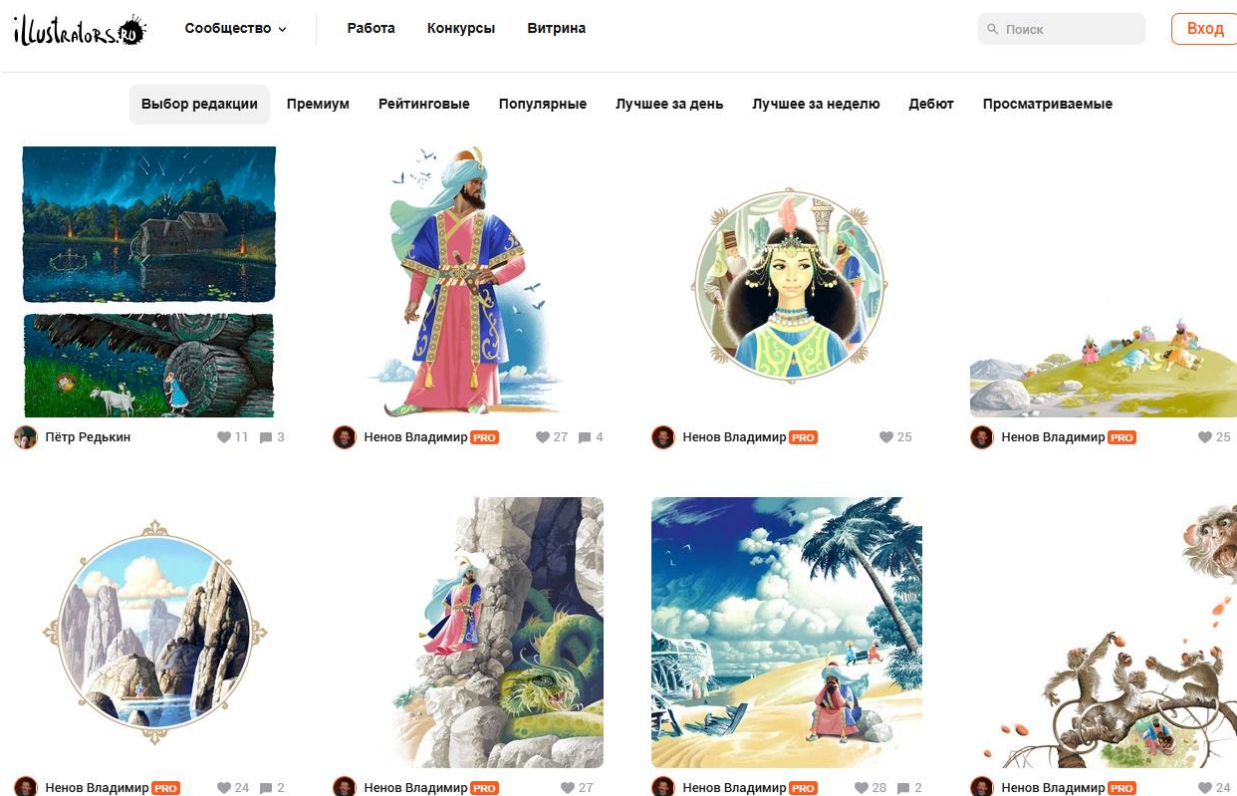


Рисунок 3 – Домашняя страница «Иллюстраторс.ру»

Платформа оснащена базовым функционалом социальных сетей, конкурсами иллюстраций, доской объявлений для поиска заказов и заказчиков, а также магазином, наполненным как предметами цифрового искусства, так и традиционного. Публикуемые работы помечаются как объекты авторского права.

К недостаткам «Иллюстраторс.ру» можно отнести отсутствие механизмов защиты от ИИ.

1.2.3 Результаты сравнительного анализа и выводы

В ходе выполнения работы был проведен сравнительный анализ рассмотренных выше программных средств. Для оценки приложений была использована трехбалльная шкала оценки, в которой 0 означал «полное отсутствие реализации», 1 – «частичную реализацию», а 2 – «полную реализацию» рассматриваемой функции. Результат сравнения программных средств для продвижения произведений цифрового искусства представлен в таблице 1.

Таблица 1 – Сравнение функционала арт-платформ

Критерий / Платформа	ArtStation	Cara	Иллюстраторс.ру
Публикация работ	2	2	2
Организация портфолио	2	1	1
Маркировка ИИ-сгенерированных работ	1	2	0
Фильтрация ИИ-контента в ленте	1	2	0
Защита работ от обучения ИИ	1	2	0
Система заказов	0	0	2
Социальные функции	2	2	2
Русскоязычный интерфейс	0	0	2
Ориентация на цифровое искусство	2	2	1
Настройка профиля	2	1	1
Связь с креатором	2	0	1
Наличие платного функционала	2	0	2
Итого	17	14	14

Подводя итог обзора программных средств для продвижения произведений цифрового искусства можно сделать вывод, что ни одно из рассмотренных решений не соответствует объявленным критериям в полном объеме.

«ArtStation» выделяется среди аналогов более продвинутой настройкой профиля креатора и встроенным в платформу чатом, однако, не смотря на огромную всемирную аудиторию цифровых креаторов, практически не предпринимает мер по контролю ИИ-контента. «Cara», напротив, является самым безопасным выбором платформы в контексте защиты от ИИ, но в данный момент не обладает обширным функционалом. Главным преимуществом «Иллюстраторс.ру» является ориентация на русскоязычное арт-сообщество, однако не обладает никакими средствами противодействия нейросетям.

Выявленные в результате данного обзора пробелы в реализации вышеперечисленных платформ подчеркивают необходимость разработки собственного программного средства. В новое программное средство для продвижения произведений цифрового искусства будут включены функции настройки портфолио и профиля, маркировка и фильтрация ИИ-контента, поддержка русскоязычного интерфейса и функционал социальных сетей.

1.3 Постановка задачи на разработку онлайн-платформы

1.3.1 Классы пользователей онлайн-платформы

Определение классов пользователей является важным этапом проектирования онлайн-платформы для продвижения произведений цифрового искусства. Оно позволяет определить различные сценарии использования системы и разграничить функционал в соответствии с выявленными группами пользователей.

Выявленные классы пользователей онлайн-платформы с их описанием и задачами представлены в таблице 2.

Таблица 2 – Классы пользователей онлайн-платформы

Класс пользователя	Описание	Задачи
Неавторизованный пользователь	Посетитель платформы, не прошедший процедуру регистрации или авторизации.	Просмотр общей ленты, профилей пользователей и цифровых проектов. Регистрация, авторизация на платформе.
Авторизованный пользователь	Зарегистрированный на платформе пользователь. Может иметь статус верифицированного пользователя или подписку «Про».	Публикация цифровых проектов, взаимодействие с контентом, подписка на авторов, настройка профиля, подача заявки на верификацию аккаунта, оформление подписки.

Таким образом, были определены основные классы пользователей, взаимодействующих с онлайн-платформой.

1.3.2 Планирование и оценка бюджета процесса разработки онлайн-платформы

Планирование процесса разработки программного средства позволяет определить объём работ, распределить ресурсы и установить временные рамки реализации проекта. Корректно выстроенный план разработки снижает риски срыва сроков и позволяет контролировать соответствие результата поставленным требованиям.

Жизненный цикл разработки программного обеспечения (ЖЦПО) – это последовательность этапов, через которые проходит программный продукт от момента возникновения идеи до его вывода из эксплуатации. Выбор модели жизненного цикла определяет порядок выполнения работ, характер взаимодействия участников команды и подход к управлению разработкой при изменении функциональных и нефункциональных требований.

Для разработки онлайн-платформы была выбрана итерационная модель жизненного цикла. В рамках данной модели разработка ведётся последовательными итерациями, каждая из которых представляет собой законченный цикл, состоящий из анализа, проектирования, реализации и тестирования. Завершением итерации является рабочая версия системы с расширенным относительно предыдущей итерации функционалом.

Выбор итерационной модели обусловлен рядом факторов, характерных для проекта, выполняемого в рамках данной выпускной квалификационной работы. Основным фактором является то, что разработка онлайн-платформы предполагает поэтапное наращивание функционала: от минимально жизнеспособного продукта до полноценного решения с инструментами монетизации. Итерационный подход позволяет на раннем этапе получить работающий продукт и при необходимости скорректировать требования к последующим версиям. Также, в условиях ограниченных ресурсов в виде одного разработчика данная модель обеспечивает управляемость процессом разработки и возможность приоритизации задач.

В рамках итерационной модели онлайн-платформа для продвижения произведений цифрового искусства была разбита на три версии, каждая из которых соответствует отдельной итерации. Планирование ресурсов, бюджета и временных затрат было осуществлено для первой версии платформы, также являющейся MVP.

Минимально жизнеспособный продукт (Minimum Viable Product, MVP) – это версия продукта, содержащая минимальный набор функций, достаточный для полноценного использования платформы и проверки основной гипотезы о

востребованности решения. Концепция MVP позволяет сосредоточить ресурсы на ключевом функционале и избежать траты времени и ресурсов на избыточные задачи.

Перечень функционала трех версий онлайн-платформы представлен в таблице 3.

Таблица 3 – Функционал версий онлайн-платформы

Версия	Функционал версии
v1.0	Регистрация, авторизация, публикация цифровых проектов, лента, лайки, комментарии, подписки, верификация аккаунта, автоматическая проверка на ИИ, маркировка ИИ-контента.
v2.0	Поиск пользователей и проектов, фильтрация по категориям, уведомления, статистика аккаунта, фильтрация ленты.
v3.0	Подписка «Про», пользовательские чаты, инструменты для заказчиков.

Для управления разработкой первой версии онлайн-платформы был выбран инструмент Project Libre, обеспечивающий возможность планирования этапов разработки, контроля выполнения задач и управления ресурсами.

На основе внесённых данных о стадиях разработки был составлен датированный план работы над проектом. Помимо этого, была рассчитана общая стоимость проекта, складывающаяся из затрат на ресурсы. План разработки представлен в виде диаграммы Ганта, отображающей ключевые этапы создания онлайн-платформы.

Диаграмма Ганта представляет собой проектный график, построенный в табличной форме с указанием этапов работ и ответственных исполнителей. Она включает две оси: вертикальную с перечнем задач и горизонтальную с временными сроками их выполнения [15]. Полученная диаграмма Ганта представлена в приложении Б (рисунок Б.1).

Начало разработки датируется 8 апреля 2026 года, завершение – 12 июня 2026 года в соответствии с построенной диаграммой.

Список поставленных задач представлен на рисунке Б.2 в приложении Б. Используемые ресурсы отражены на рисунке Б.3 в приложении Б.

По результатам планирования суммарные временные затраты на разработку платформы составили 384 часов, а общая стоимость проекта – 115200 рублей.

2 Проектирование онлайн-платформы для продвижения произведений цифрового искусства

2.1 Анализ функциональных требований

Проведение анализа функциональных требований является обязательным этапом разработки программного средства, поскольку позволяет сформировать чёткое понимание границ системы, определить сценарии взаимодействия пользователей с онлайн-платформой и исключить неоднозначность при реализации продукта.

После проведения сравнительного анализа существующих платформ для продвижения произведений цифрового искусства были выявлены функциональные требования к разрабатываемому программному средству. В соответствии с концепцией минимально жизнеспособного продукта (MVP) в рамках настоящей выпускной квалификационной работы реализуются следующие функциональные требования:

- регистрация и авторизация пользователей в системе;
- настройка профиля;
- публикация произведений цифрового искусства;
- автоматическая маркировка загружаемого контента как созданного с помощью генеративных ИИ-инструментов;
- верификация аккаунтов пользователей;
- лента для просмотра работ цифровых креаторов;
- фильтрация ИИ-контента в пользовательской ленте;
- подписка на креаторов;
- взаимодействие с контентом посредством комментариев и лайков;
- русскоязычный интерфейс платформы.

В соответствии с выявленными функциональными требованиями к платформе и определёнными классами пользователей были разработаны диаграммы вариантов использования UML для двух основных групп пользователей: неавторизованных и авторизованных.

Диаграмма вариантов использования онлайн-платформы для продвижения произведений цифрового искусства неавторизованным пользователем представлена на рисунке 4.



Рисунок 4 – Диаграмма вариантов использования неавторизованного пользователя

Все запросы пользователя, вне зависимости от рассматриваемого варианта использования, направляются через единую точку входа – шлюз API, который осуществляет маршрутизацию к соответствующим компонентам системы.

Стоит отметить, что авторизованный пользователь наследует весь функционал неавторизованного пользователя, а также получает доступ к расширенному набору возможностей платформы:

- публикация собственных цифровых проектов;
- кастомизация своего профиля и ленты;
- социальное взаимодействие между пользователями цифровыми проектами и цифровыми креаторами.

Таким образом, подобный функционал для пользователей платформы должен побуждать людей, только начавших изучать платформу, создавать собственные аккаунты, способствуя становлению полноценной частью арт-сообщества.

Диаграмма вариантов использования авторизованным пользователем представлена на рисунке 5.

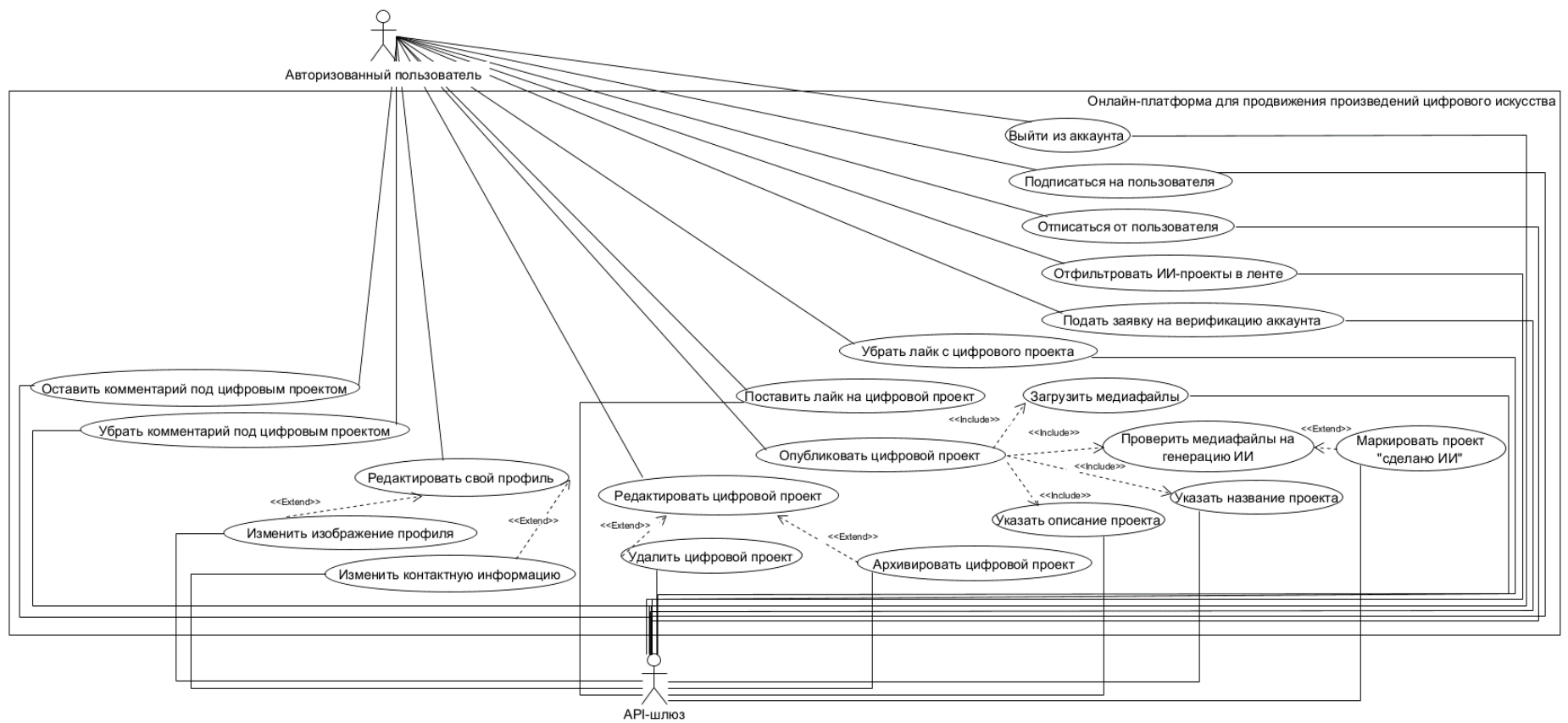


Рисунок 5 – Диаграмма вариантов использования авторизованного пользователя

Ниже приведены спецификации наиболее важных бизнес-процессов платформы.

Сценарий варианта использования «Авторизоваться» представлен на рисунке 6.

Прецедент	Авторизоваться
ID	8
Краткое описание	Зарегистрированный пользователь выполняет вход в аккаунт на платформе.
Главные актеры	Неавторизованный пользователь
Второстепенные актеры	Система (проверка учётных данных)
Предусловие	Пользователь не авторизован и имеет зарегистрированный аккаунт.
Основной поток	<ol style="list-style-type: none"> 1. Пользователь переходит на страницу авторизации. 2. Система отображает форму входа. 3. Пользователь вводит адрес электронной почты и пароль. 4. Пользователь нажимает кнопку «Войти». 5. Система проверяет соответствие введенных данных записям в базе. 6. Система создаёт сессию и перенаправляет пользователя на главную страницу.
Постусловие	Пользователь авторизован и получает доступ к полному функционалу платформы.
Альтернативные потоки	Введены неверные учётные данные: <ol style="list-style-type: none"> 1. Система выводит сообщение «Неверный email или пароль». 2. Пользователь повторяет попытку с корректными данными.

Рисунок 6 – Сценарий варианта использования «Авторизоваться»

Сценарий варианта использования «Отфильтровать ИИ-проекты в ленте» представлен на рисунке 7.

Прецедент	Отфильтровать ИИ-проекты в ленте
ID	3
Краткое описание	Авторизованный пользователь включает фильтрацию ленты для скрытия цифровых проектов, созданных с использованием ГИИ.
Главные актеры	Авторизованный пользователь
Второстепенные актеры	Система (применение фильтра к ленте)
Предусловие	Пользователь авторизован и находится на странице с лентой.
Основной поток	<ol style="list-style-type: none"> 1. Пользователь открывает страницу ленты. 2. Пользователь активирует переключатель «Скрыть ИИ-контент». 3. Система применяет фильтр и обновляет ленту, исключая проекты с маркировкой ИИ. 4. Система отображает обновлённую ленту без ИИ-проектов.
Постусловие	Лента отображается без цифровых проектов, созданных с использованием ГИИ. Настройка фильтра сохраняется в профиле пользователя.
Альтернативные потоки	Нет.

Рисунок 7 – Сценарий варианта использования «Отфильтровать ИИ-проекты в ленте»

Сценарий варианта использования «Подать заявку на верификацию аккаунта» представлен на рисунке 8.

Прецедент	Подать заявку на верификацию аккаунта
ID	6
Краткое описание	Авторизованный пользователь подаёт заявку на подтверждение авторства публикуемых работ для получения статуса верифицированного креатора.
Главные актеры	Авторизованный пользователь
Второстепенные актеры	Система (сохранение заявки и загруженных файлов)
Предусловие	Пользователь авторизован и не имеет статуса верифицированного креатора.
Основной поток	<ol style="list-style-type: none"> 1. Пользователь переходит в раздел верификации аккаунта. 2. Система отображает форму подачи заявки. 3. Пользователь загружает доказательства авторства: спидпейнты, скриншоты слоёв работ или иные материалы. 4. Пользователь нажимает кнопку «Отправить заявку». 5. Система валидирует загруженные файлы. 6. Система сохраняет заявку со статусом «На рассмотрении». 7. Система отображает сообщение об успешной отправке заявки.
Постусловие	Заявка на верификацию сохранена в системе и ожидает рассмотрения.
Альтернативные потоки	Нет.

Рисунок 8 – Сценарий варианта использования «Подать заявку на верификацию аккаунта»

В результате анализа функциональных требований были определены варианты использования платформы для авторизованных и неавторизованных пользователей, разработаны соответствующие диаграммы UML и составлены спецификации сценариев взаимодействия пользователей с системой.

2.2 Анализ нефункциональных требований

Нефункциональные требования определяют качественные характеристики разрабатываемой платформы, не связанные напрямую с конкретными функциями системы, но оказывающие непосредственное влияние на опыт использования, надёжность и безопасность продукта.

Для обеспечения максимальной доступности, систему следует реализовывать в виде веб-приложения, что освободит пользователей от необходимости установки дополнительного программного обеспечения на свои устройства.

Ввиду специфики платформы, функционал которой фокусируется на загружаемых на нее медиафайлах, необходимо обеспечивать их быструю загрузку. Пользовательская лента должна быть реализована с механизмом бесконечной прокрутки, что улучшает пользовательский опыт и, соответственно, время сессии.

В рамках работы платформы не должно допускаться физическое удаление данных. Скрытие данных от пользователя следует осуществлять механизмом «мягкое удаление» (англ. soft delete), что обеспечивает целостность данных. Это помогает как защититься от внутренних угроз, так и сохранять аудируемость данных на случаи расследований кибер-преступлений, проведения форензик-экспертиз [16].

Пароли от аккаунтов следует хранить в базе данных исключительно в хэшированном виде. Система не должна раскрывать информацию о существовании электронного адреса в базе данных при обработке запросов на регистрацию и восстановление пароля. Доступ к защищённым эндпоинтам осуществляется только при наличии валидного токена.

Интерфейс платформы должен быть реализован на русском языке. В процессе проектирования самого интерфейса и кода должен применяться принцип наименьшего удивления (Principle of Least Astonishment, POLA), согласно которому поведение системы должно соответствовать ожиданиям пользователя, а уровень подготовки, необходимый пользователю для достижения поставленных им целей в рамках платформы, должен быть минимальным.

2.3 Анализ требований к архитектуре программных средств

В рамках проведения анализа требований к архитектуре разрабатываемой платформы была выбрана микросервисная архитектура, которая представляет собой современный подход к разработке распределённых систем и считается одним из наиболее инновационных решений благодаря своей высокой гибкости.

Микросервисная архитектура является оптимальным выбором для разработки сложных и масштабируемых приложений [17].

Поскольку микросервисная архитектура хорошо интегрируется с предметно-ориентированным проектированием (Domain-Driven Design, DDD), для проектирования системы был выбран именно этот подход. Благодаря модульности сервисов и их функциональной независимости каждая предметная область эффективно реализуется в виде отдельного микросервиса. Каждый микросервис в системе охватывает конкретную бизнес-область и решает определённые задачи. Для хранения данных в каждом сервисе в приложении используется база данных PostgreSQL.

На рисунке 9 приведена диаграмма компонентов, из которых состоит система.

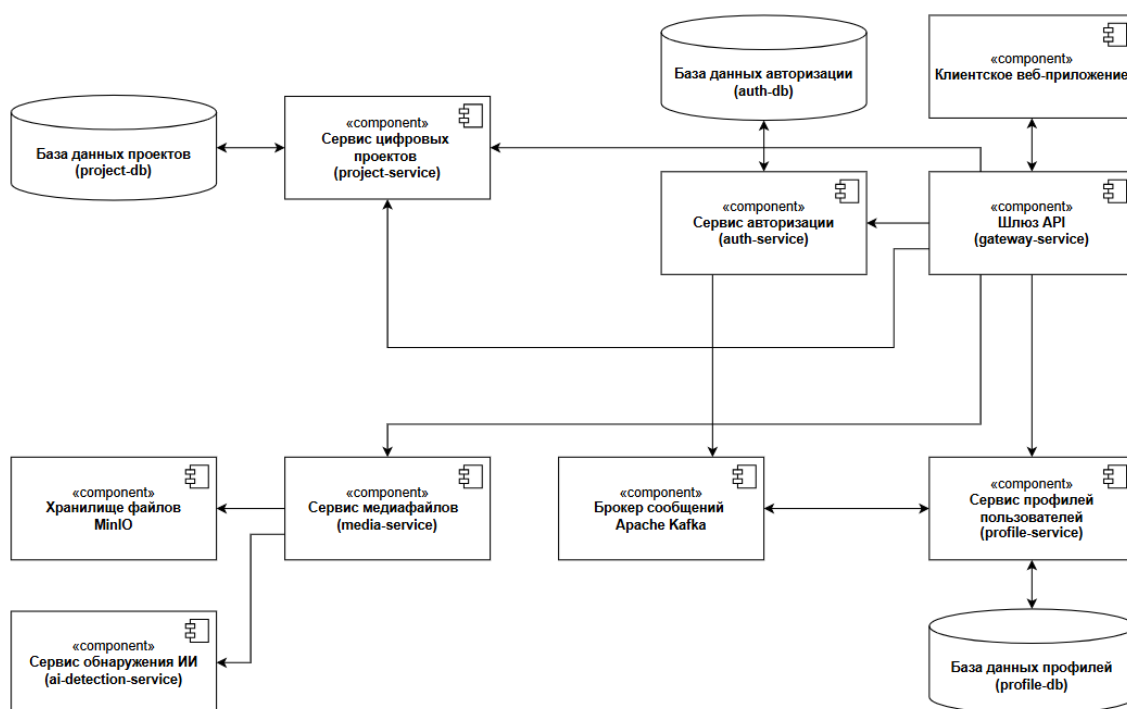


Рисунок 9 – Диаграмма компонентов

Каждый микросервис, представленный в разработанной архитектуре, отвечает за свою предметную область:

- шлюз API (gateway-service) – маршрутизация входящих запросов от клиентского веб-приложения к соответствующим микросервисам с проверкой авторизации;

- сервис авторизации (auth-service) – регистрация, авторизация, верификация аккаунта;
- сервис цифровых проектов (project-service) – работа с цифровыми проектами, формирование и фильтрация ленты, лайки, комментарии;
- сервис медиафайлов (media-service) – обработка и преобразование загружаемых на платформу медиафайлов;
- сервис профилей пользователей (profile-service) – управление данными профиля и подписками, верификация аккаунта;
- брокер сообщений Apache Kafka – асинхронная передача событий между микросервисами;
- сервис обнаружения ИИ (ai-detection-service) – автоматическая проверка загружаемых изображений на факт генерации;
- хранилище файлов MinIO – объектное хранилище для аватаров пользователей, медиафайлов цифровых проектов и доказательств авторства проектов.

В целях создания продуманной микросервисной архитектуры, подразумевающей высокую связность и низкую связанность компонентов системы, были задействованы распространённые архитектурные паттерны.

Все запросы от клиентского веб-приложения поступают через единую точку входа – шлюз API, который берёт на себя маршрутизацию запросов к соответствующим микросервисам.

Согласно паттерну «База данных на сервис» (Database per Service) каждый микросервис владеет собственной изолированной базой данных и является единственным компонентом системы, имеющим к ней прямой доступ. При необходимости сторонние микросервисы могут получить необходимые данные через события или API [18].

Взаимодействие между микросервисами организовано через асинхронный обмен событиями посредством брокера сообщений Apache Kafka. Сервис-производитель публикует событие, а сервисы-потребители обрабатывают его независимо [19].

В качестве основы для взаимодействия между клиентским приложением и микросервисами был выбран REST-ориентированный подход. Он обеспечивает унифицированный и масштабируемый способ обмена данными через стандартные HTTP-методы:

- GET – передача текущего представления целевого ресурса;
- POST – выполнение специфичной для ресурса обработки тела запроса;
- PUT – размещение или замена ресурса на сервере;
- PATCH – применение частичных изменений к целевому ресурсу;
- DELETE – удаление ресурса на сервере [20].

Внедрённые в проектирование шаблоны способствуют улучшению масштабируемости и обеспечению устойчивости системы к сбоям.

2.4 Анализ требований к данным

Концептуальная модель – это семантическая схема, иллюстрирующая сущности предметной области, их атрибуты и взаимосвязи, без учёта особенностей физического представления в информационных системах.

В процессе разработки концептуальной схемы базы данных был подготовлен набор сущностей, который представлен в таблице 4.

Таблица 4 – Первоначальный набор сущностей концептуальной модели базы данных

№ п/п	Наименование	Описание
1	Учетные данные	Данные для авторизации пользователя в системе.
2	Роль	Запись, определяющая уровень доступа пользователя в системе.
3	Токен сброса пароля	Временная запись для идентификации запроса на восстановление пароля с указанием срока действия и отметкой об использовании.
4	Заявка на верификацию	Запрос пользователя на подтверждение авторства публикуемых проектов со статусом рассмотрения и ссылкой на причину отказа.
5	Причина отказа	Справочная запись с описанием основания для отклонения заявки на верификацию.
6	Профиль пользователя	Публичная информация о пользователе.

Продолжение таблицы 4

№ п/п	Наименование	Описание
7	Подписка	Связь между двумя пользователями, отражающая факт отслеживания цифровых проектов одного пользователя другим.
8	Цифровой проект	Публикация цифрового креатора.
9	Медиафайл проекта	Медиафайл, прикрепленный к цифровому проекту с указанием порядка отображения. Максимальное количество изображений в одном проекте – 20.
10	Лайк	Отметка «нравится», которую можно поставить на цифровой проект.
11	Комментарий	Текстовый отклик пользователя на цифровой проект.

На основе первоначального набора сущностей была подготовлена диаграмма «сущность-связь» концептуальной модели данных.

Диаграмма «сущность-связь» концептуальной модели данных в полном атрибутивном представлении показана на рисунке 10.

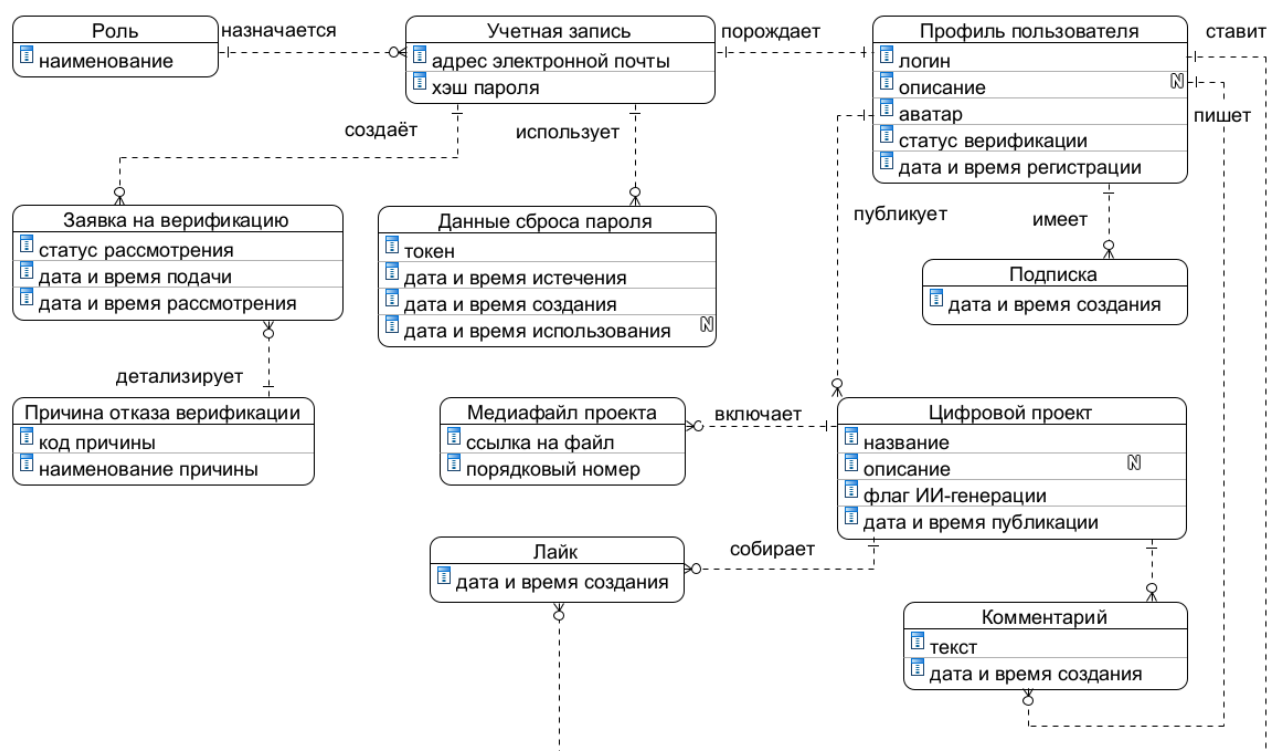


Рисунок 10 – Диаграмма «сущность-связь» концептуальной модели данных

Диаграмма демонстрирует, что все сущности тесно взаимосвязаны, без изолированных элементов, что свидетельствует о логической связности модели и её соответствии требованиям платформы для творческого сообщества.

2.5 Анализ требований к пользовательскому интерфейсу

Пользовательский интерфейс является ключевым элементом взаимодействия между пользователем и платформой. Проектируемый интерфейс должен позволять ранее выделенным группам пользователей выполнять все предусмотренные для них варианты использования разрабатываемой онлайн-платформы.

Страницы, доступные неавторизованному пользователю:

- страница входа / регистрации;
- страница восстановления пароля;
- лента цифровых проектов;
- страница цифрового проекта;
- страница профиля пользователя.

Стартовая страница для неавторизованных пользователей должна включать форму через почту и пароль, ссылку на страницу восстановления пароля и переключатель на форму регистрации. Форма регистрации состоит из полей для ввода почты и пароля.

В страницу для восстановления пароля от учетной записи следует включить форму с полем ввода почты для отправки ссылки для сброса пароля, а также форму ввода нового пароля, открываемую по ссылке из письма.

Главная страница платформы должна обладать бесконечной прокруткой. Лента публикаций отображает карточки цифровых проектов, то есть первый медиафайл публикации. Карточки проектов, не прошедших проверку на ИИ-генерацию, содержат соответствующую маркировку. Нажатие на карточку проекта должно перенаправлять пользователя на страницу цифрового проекта, содержащую карусель медиафайлов проекта, его название, описание, счетчик лайков, комментарии и аккаунт цифрового креатора.

Страница цифрового креатора: аватар, имя пользователя, биография, статус верификации, количество подписчиков и публикаций, сетка опубликованных цифровых проектов.

Для авторизованного пользователя предусмотрены следующие страницы:

- личный кабинет;
- редактирование профиля;
- публикация цифрового проекта;
- редактирование цифрового проекта;
- страница настройки ленты;
- страница верификации аккаунта;
- страница смены пароля.

Личный кабинет должен быть оснащен кнопками редактирования профиля, создания нового цифрового проекта и подачи заявки на верификацию (скрывается после верификации).

Форма изменения данных профиля: загрузка нового аватара, редактирование имени пользователя и биографии.

Форма публикации цифрового проекта: загрузка медиафайлов (до 20 файлов), поля ввода названия и описания. После загрузки медиафайлов автоматически запускается проверка на ИИ-генерацию.

В форму редактирования проекта, аналогичной странице его публикации, следует добавить кнопку удаления проекта.

Рядом с лентой цифровых проектов для авторизованных пользователей необходимо добавить переключатель фильтрации ИИ-контента.

Форма подачи заявки на верификацию должна содержать поле для загрузки доказательств авторства (спидпейнты, скриншоты слоёв работы). Возле поля следует поместить краткое описание требований к материалам и кнопку отправки заявки. Недоступна для уже верифицированных пользователей.

2.6 Анализ требований к аппаратно-программному обеспечению

2.6.1 Требования к аппаратному обеспечению

Для корректного развёртывания платформы были определены минимальные и рекомендуемые аппаратные требования к серверу (таблица 5).

Таблица 5 – Аппаратные требования к серверу

Характеристика	Минимальные требования	Рекомендуемые требования
Процессор	x64 с 2 ГГц, 4 ядра	x64 с 3 ГГц и более, 8 ядер и более
Оперативная память	8 Гб	16 Гб и более
Тип и объём жёсткого диска	SSD со 100 Гб	SSD с 500 Гб и более
Пропускная способность сети	100 Мбит/с	1 Гбит/с

Минимальные требования обусловлены микросервисной архитектурой платформы: одновременный запуск нескольких сервисов, брокера сообщений Apache Kafka и объектного хранилища MinIO предъявляет повышенные требования к аппаратному обеспечению по сравнению с монолитными приложениями. Рекомендуемые параметры направлены на обеспечение стабильной производительности при работе с медиафайлами и одновременными запросами пользователей.

2.6.2 Требования к системному программному обеспечению

Для развёртывания и эксплуатации платформы необходимо следующее системное программное обеспечение, представленное в таблице 6.

Таблица 6 – Требования к системному программному обеспечению

Программное обеспечение	Версия	Назначение
Docker	24.0 и выше	Контейнеризация микросервисов
Docker Compose	2.20 и выше	Оркестрация контейнеров
Confluent Kafka	7.0.1 и выше	Брокер сообщений
Apache ZooKeeper	7.0.1 и выше	Координация кластера Kafka
PostgreSQL	16 и выше	Реляционные базы данных микросервисов
MinIO	последняя	Объектное хранилище медиафайлов
Kafka UI	последняя	Мониторинг брокера сообщений

Корректная работа клиентского веб-приложения гарантируется в браузере Google Chrome, Microsoft Edge и Safari.

2.6.3 Требования к средствам разработки

В процессе разработки платформы используются программные средства, языки и фреймворки, представленные в таблице 7.

Таблица 7 – Средства разработки онлайн-платформы

Средство	Назначение
Java	Язык разработки серверной части платформы
Spring Boot	Фреймворк для разработки серверной части платформы
Spring Security	Аутентификация и авторизация
React	Фреймворк для разработки клиентского веб-приложения
TypeScript	Язык разработки клиентской части
Python	Язык разработки сервиса обнаружения ИИ
MinIO	Файловое хранилище
PostgreSQL	База данных
Docker Desktop	Локальный запуск контейнеров на Windows
Windows	Операционная система рабочей машины разработчика
IntelliJ IDEA	Среда разработки серверной части платформы
VS Code	Среда разработки клиентской веб-платформы
Figma	Платформа проектирования пользовательского интерфейса
Visual Paradigm	Разработка UML-диаграмм
Git	Система контроля версий
GitHub	Хранение и управление репозиторием
Project Libre	Планирование процесса разработки

Таким образом, выбранный стек технологий полностью покрывает функциональные и нефункциональные требования к разрабатываемой платформе, обеспечивает масштабируемость и высокую производительность системы.

3 Разработка программного обеспечения онлайн-платформы

3.1 Развертывание физической схемы базы данных

В качестве СУБД для хранения данных всех микросервисов платформы была выбрана реляционная система PostgreSQL. Данный выбор обусловлен поддержкой сложных запросов и соответствием задачам, решаемым в рамках платформы. Кроме того, данное решение хорошо интегрируется с экосистемой Spring Boot.

В соответствии с принципом Database per Service для каждого микросервиса разворачивается отдельный экземпляр базы данных: auth_db для сервиса авторизации, profile_db для сервиса профилей пользователей и project_db для сервиса цифровых проектов.

На рисунке 11 представлена физическая модель базы данных auth_db.

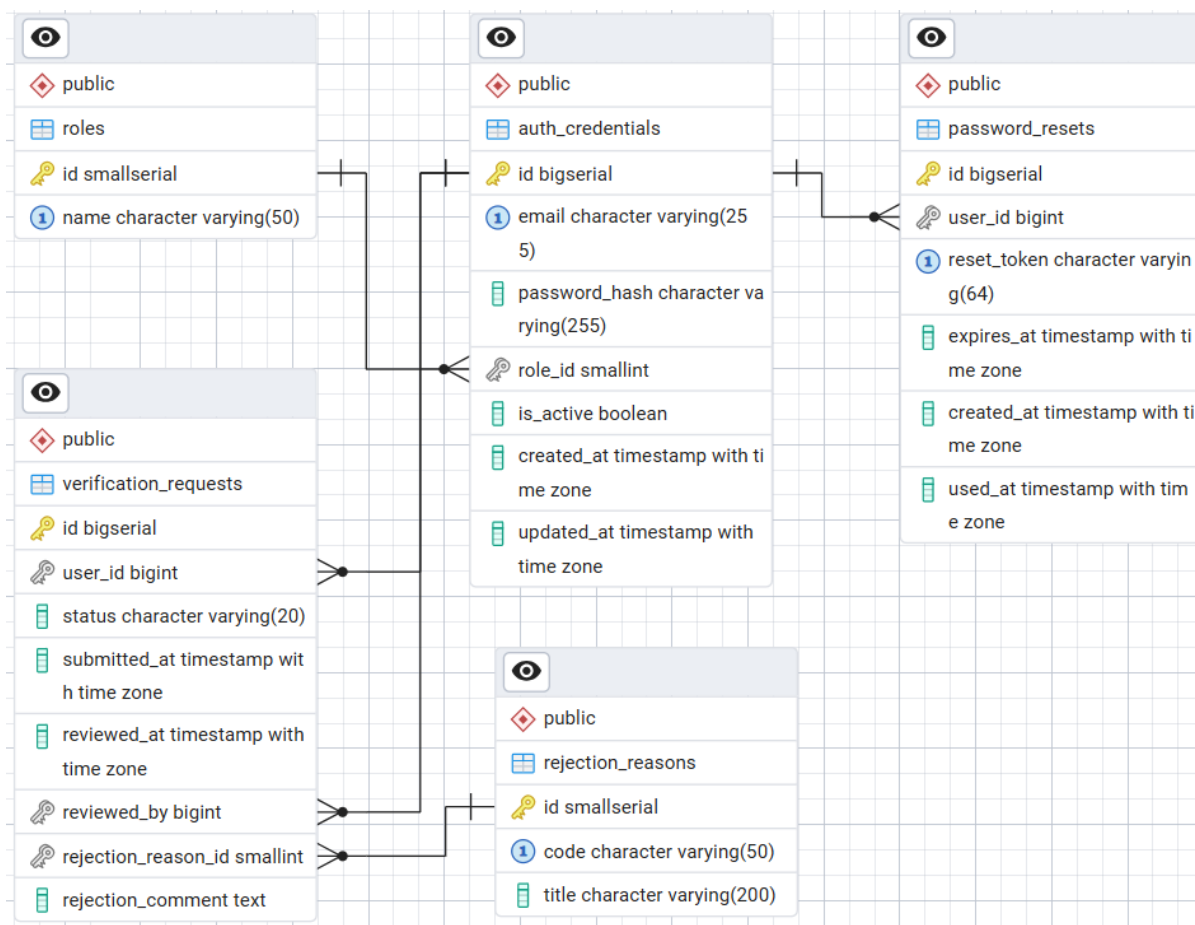
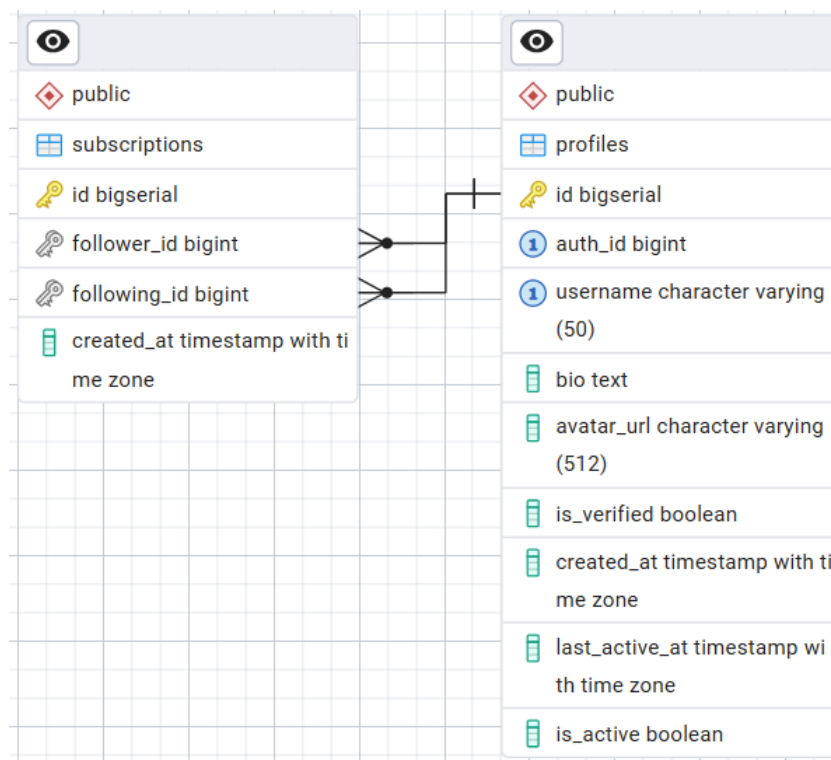
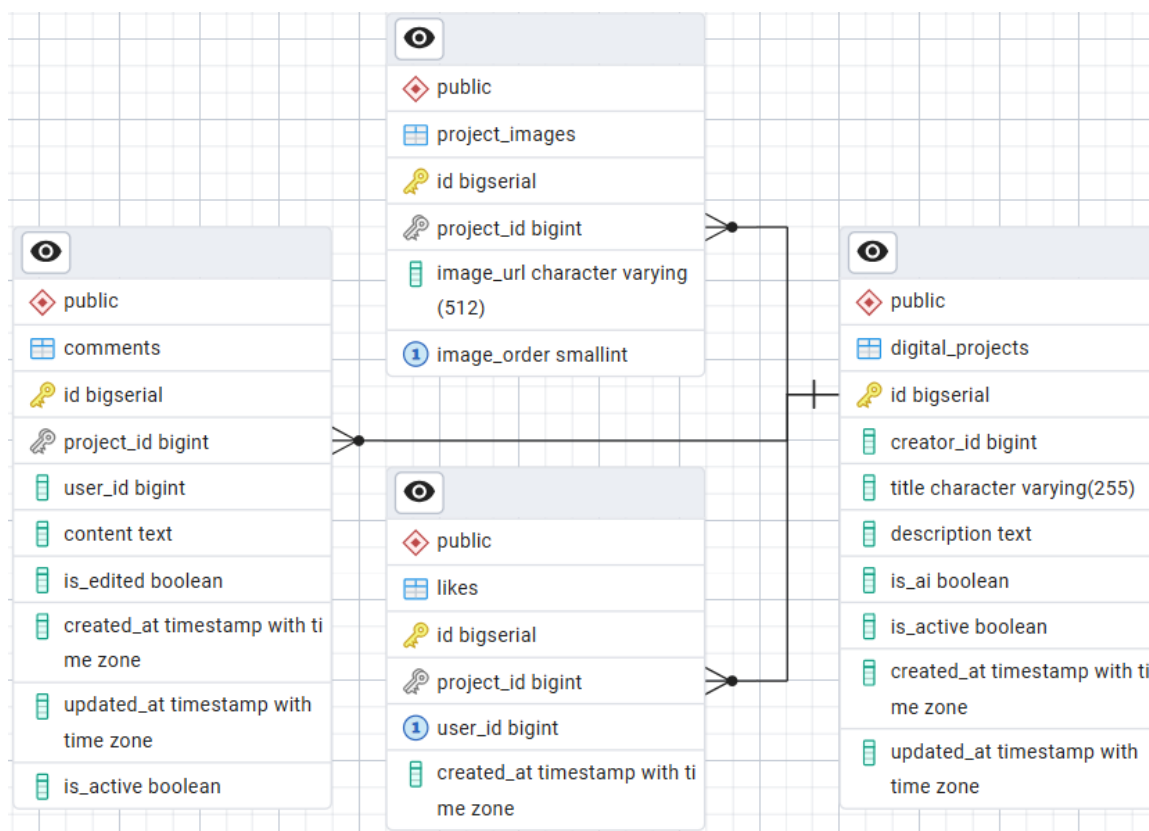


Рисунок 11 – Физическая модель БД auth_db

На рисунке 12 представлена физическая модель базы данных profile_db.

Рисунок 12 – Физическая модель БД `profile_db`

На рисунке 13 представлена физическая модель базы данных `project_db`.

Рисунок 13 – Физическая модель БД `project_db`

Спецификации атрибутов всех сущностей таблиц баз данных представлены в приложении В.

SQL-скрипты для создания таблиц представлены в приложении Г.

3.2 Реализация серверного программного обеспечения

В основе серверной части онлайн-платформы для продвижения цифровых арт-объектов лежит микросервисный подход к построению приложений. Ключевые компоненты системы написаны на Java с применением фреймворка Spring Boot.

Архитектурно каждый из микросервисов организован по многоуровневому принципу и соответствует канонам стиля REST API. Во всех микросервисах выдержана единая структура, включающая следующие функциональные уровни: представления, бизнес-логики, доступа к данным, доменный и уровень объектов передачи данных.

Ниже представлены таблицы с описанием методов API каждого микросервиса.

В таблице 8 представлены методы API сервиса авторизации.

Таблица 8 – Методы API сервиса авторизации

Метод	URL	Доступ	Описание
POST	/auth/register	Публичный	Регистрация пользователя.
POST	/auth/login	Публичный	Авторизация пользователя.
POST	/auth/verify	Авторизованный	Подача заявки на верификацию аккаунта.

В таблице 9 представлены методы API сервиса медиафайлов.

Таблица 9 – Методы API сервиса медиафайлов

Метод	URL	Доступ	Описание
POST	/media/projects/upload	Публичный	Загрузка изображений цифрового проекта.
POST	/media/avatars/upload	Публичный	Загрузка аватара пользователя.
DELETE	/media/projects	Публичный	Удаление изображения из проекта.

В таблице 10 представлены методы API сервиса цифровых проектов.

Таблица 10 – Методы API сервиса цифровых проектов

Метод	URL	Доступ	Описание
POST	/projects	Авторизованный	Создание цифрового проекта.
GET	/projects/all	Публичный	Получение всех активных цифровых проектов (лента).
GET	/projects/{id}	Публичный	Получение цифрового проекта по идентификатору.
GET	/projects/by-creator/{creatorId}	Публичный	Получение всех проектов автора.
PATCH	/projects/{id}	Авторизованный	Обновление названия и описания цифрового проекта.
DELETE	/projects/{id}	Авторизованный	Удаление цифрового проекта.
POST	/projects/{id}/like	Авторизованный	Постановка лайка на цифровой проект.
DELETE	/projects/{id}/like	Авторизованный	Снятие лайка с цифрового проекта.
POST	/projects/{id}/comments	Авторизованный	Добавление комментария к цифровому проекту.
GET	/projects/{id}/comments	Публичный	Получение списка комментариев к цифровому проекту.
DELETE	/projects/comments/{commentId}	Авторизованный	Удаление комментария у цифрового проекта.

В таблице 11 представлены методы API сервиса обнаружения ИИ-контента.

Таблица 11 – Методы API сервиса обнаружения ИИ-контента

Метод	URL	Доступ	Описание
GET	/health	Публичный	Проверка работоспособности сервиса.
POST	/detect	Внутренний	Проверка изображения на факт ИИ-генерации.

В таблице 12 представлены методы API сервиса профилей.

Таблица 12 – Методы API сервиса профилей

Метод	URL	Доступ	Описание
GET	/profiles/me	Авторизованный	Получение профиля текущего пользователя.
PUT	/profiles/me	Авторизованный	Обновление данных профиля.
POST	/profiles/me/avatar	Авторизованный	Загрузка нового аватара профиля.
POST	/profiles/subscribe/{followingId}	Авторизованный	Подписка на пользователя.
DELETE	/profiles/subscribe/{followingId}	Авторизованный	Отписка от пользователя.

Продолжение таблицы 12

Метод	URL	Доступ	Описание
GET	/profiles/by-username/{username}	Публичный	Получение профиля пользователя по имени.
GET	/profiles/by-auth-id/{authId}	Публичный	Получение профиля пользователя по идентификатору учётных данных.

Благодаря контейнеризации с использованием Docker каждый сервис может быть запущен отдельно для разработки, тестирования или отладки.

Для обеспечения единой точки входа для клиентских приложений был разработан API-шлюз. Он выполняет маршрутизацию запросов к соответствующим микросервисам и централизованную обработку аутентификации на основе JWT.

Для обеспечения слабой связности между микросервисами платформы асинхронное межсервисное взаимодействие реализовано посредством брокера сообщений Apache Kafka. При наступлении значимых событий в системе сервис-производитель публикует сообщение в соответствующий топик, а сервисы-потребители его обрабатывают.

В разрабатываемой онлайн-платформе используются два топика, представленные в таблице 13.

Таблица 13 – Топики в Apache Kafka

Топик	Производитель	Потребитель	Назначение
user-registered	auth-service	profile-service	Публикуется после успешной регистрации пользователя. Содержит идентификатор пользователя, email и роль. Profile-service создаёт профиль с автоматически сгенерированным именем пользователя.
account-verification-approved	auth-service	profile-service	Публикуется после одобрения заявки на верификацию аккаунта. Содержит идентификатор пользователя и email. Profile-service устанавливает флаг is_verified в значение true.

На рисунке 14 представлена диаграмма последовательности при создании цифрового проекта на платформе.

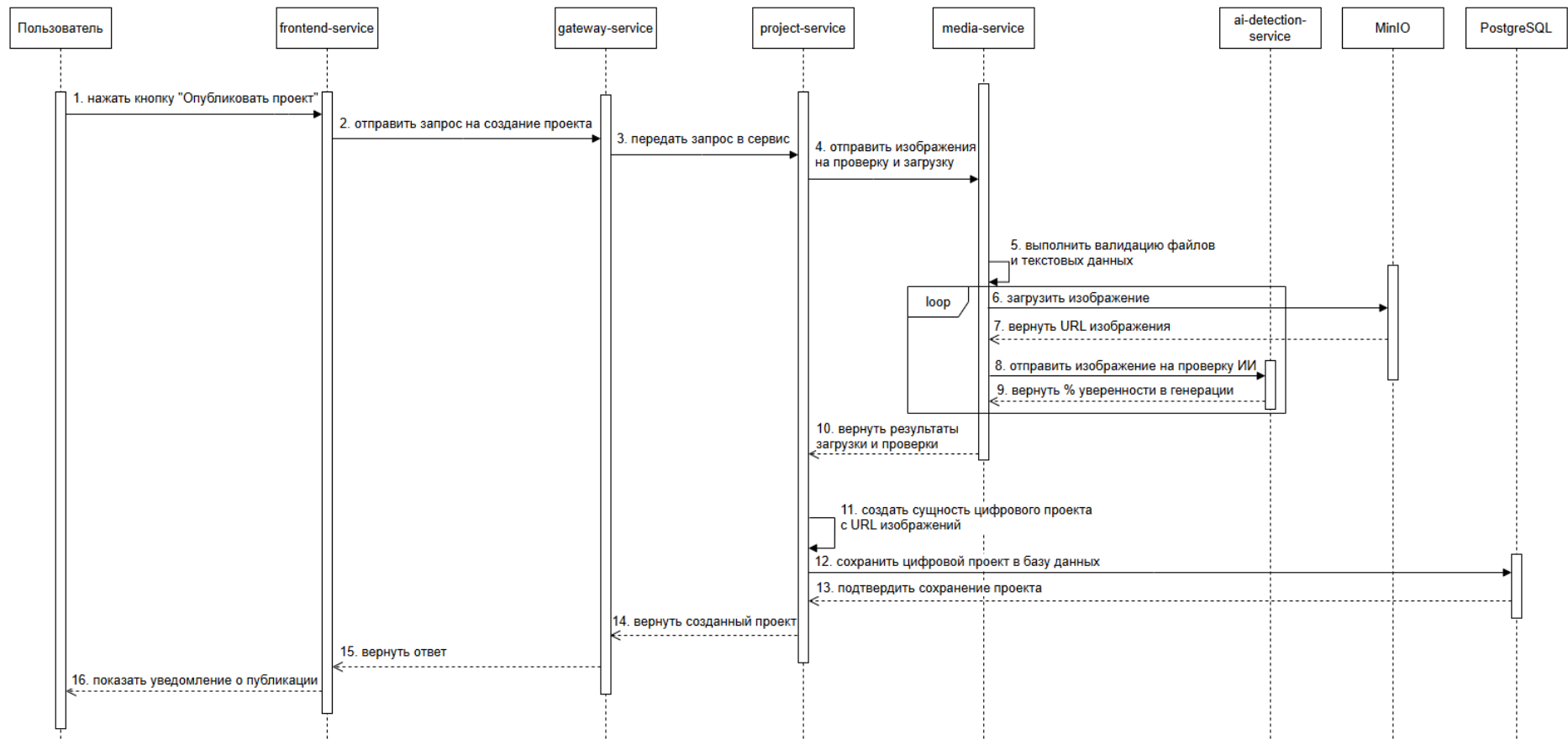


Рисунок 14 – Диаграмма последовательности создания цифрового проекта

Реализация функционала создания цифрового проекта разделена между тремя микросервисами: сервисом цифровых проектов, сервисом медиафайлов и сервисом обнаружения ИИ-контента.

Как видно из диаграммы, сервис цифровых проектов отвечает исключительно за управление сущностями проектов и их хранение в базе данных. Сервис медиафайлов выполняет валидацию, загрузку файлов в объектное хранилище MinIO и координирует проверку изображений. Сервис обнаружения ИИ-контента реализует логику классификации изображений с помощью модели машинного обучения.

Сервис обнаружения ИИ-контента был разработан на языке Python с использованием фреймворка FastAPI. Классификация изображений осуществляется с помощью специализированной модели Organika/sdxl-detector, размещенной на открытой платформе Hugging face [21]. Задачей данной модели является определение происхождения изображения: рукотворного или искусственного, в особенности сгенерированного современной моделью Stable Diffusion XL (SDXL), но также поддерживает детекцию контента, созданного более ранними генеративными алгоритмами. Результат обязательной проверки изображений в дальнейшем используется для информирования пользователей и обеспечения прозрачности публикуемого контента.

Листинг микросервиса медиафайлов приведен в приложении Д. Полный код серверной части проекта опубликован в публичном репозитории на платформе GitHub [22].

Диаграммы классов для сервисов авторизации, медиафайлов, профилей и цифровых проектов и представлены на рисунках Е.1, Е.2, Е.3 и Е.4 соответственно в приложении Е.

3.3 Реализация клиентского программного обеспечения

Для обеспечения удобного и современного интерфейса онлайн-платформы для продвижения произведений цифрового искусства клиентское приложение было разработано с использованием фреймворка Next.js 16 с использованием

React 19. Выбор данного набора технологий обусловлен требованиями к разработке клиентского компонента в контексте микросервисной архитектуры серверного, а Next.js предоставляет эффективные средства для создания и повторного использования независимых компонентов, организации кода по доменным областям и маршрутизации. Для стилизации пользовательского интерфейса применяется фреймворк Tailwind CSS. В качестве HTTP-клиента для обеспечения взаимодействия клиентской части с сервисами в качестве HTTP-клиента применяется библиотека Axios.

Диаграмма компонентов клиентской части онлайн-платформы приведена на рисунке Ж.1 в приложении Ж.

В таблице 14 приведено описание компонентов разработанного клиентского программного обеспечения.

Таблица 14 – Компоненты клиентского программного обеспечения

Компонент	Назначение
login.tsx	Страница авторизации пользователя
register.tsx	Страница регистрации пользователя
profile.tsx	Личный кабинет пользователя
profile-edit.tsx	Страница редактирования профиля пользователя
project-create.tsx	Страница создания проекта
project-edit.tsx	Страница редактирования проекта
feed.tsx	Главная страница с лентой цифровых проектов
project-detail.tsx	Страница отдельного проекта
user-profile.tsx	Публичная страница профиля пользователя
Navbar.tsx	Навигационная панель приложения
AuthGuard.tsx	Компонент, отвечающий за проверку авторизации
LogoutButton.tsx	Кнопка выхода из аккаунта
ProfileHeader.tsx	Блок информации о профиле
AvatarCropper.tsx	Компонент обрезки и загрузки аватара
ProjectCard.tsx	Предпросмотр проекта в ленте
ProjectCardSkeleton.tsx	Загрузочный заполнитель предпросмотра проекта
Carousel.tsx	Карусель изображений в проекте
LikeButton.tsx	Кнопка «Нравится»
CommentList.tsx	Список комментариев к проекту
authStore.ts (Zustand)	Хранилище состояния авторизации
AuthProvider.tsx	Провайдер контекста авторизации
QueryProvider.tsx	Провайдер React Query для управления серверными данными
api.ts	Компонент, отвечающий за централизованную настройку запросов
auth.ts	API-методы для работы с авторизацией
me.ts	API-методы для работы с данными текущего пользователя
projects.ts	API-методы для работы с проектами

Продолжение таблицы 14

Компонент	Назначение
profiles.ts	API-методы для работы с профилями пользователей
media.ts	API-методы для загрузки и работы с медиафайлами
useInfiniteScroll.ts	Компонент, отвечающий за бесконечную прокрутку проектов в ленте
index.ts	Общие интерфейсы приложения

Разработанная клиентская часть интегрирована в общую Docker-инфраструктуру проекта и запускается одновременно со всеми.

3.4 Реализация интерфейса приложения

В соответствии с требованиями к интерфейсу, описанными в пункте 2.5, пользователю доступны такие страницы, как главная, авторизации и регистрации, личного профиля и его кастомизации, а также страницы просмотра и создания цифрового проекта.

При переходе на онлайн-платформу неавторизованному пользователю открывается главная страница, представленная на рисунке 15.

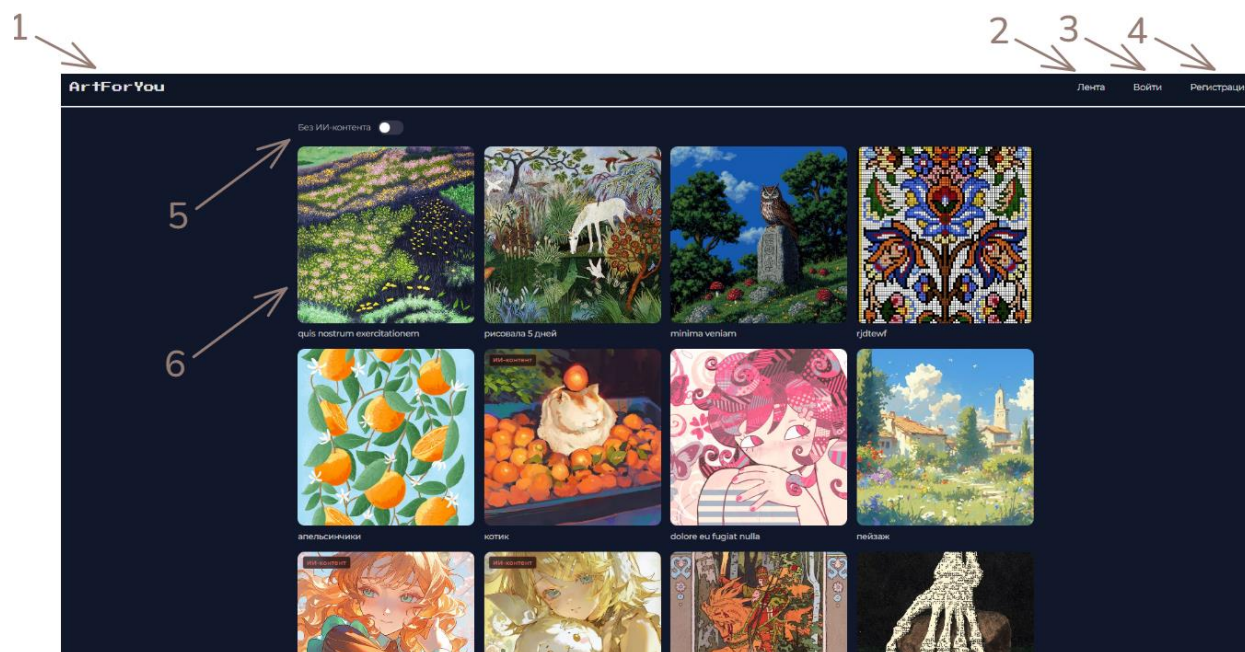


Рисунок 15 – Главная страница

Основные элементы главной страницы:

- 1 – логотип платформы;
- 2 – кнопка перехода на главную страницу;

- 3 – кнопка входа в аккаунт;
- 4 – кнопка регистрации на платформе;
- 5 – фильтр ИИ-контента;
- 6 – карточка (превью) цифрового проекта с названием.

Если пользователь авторизован, вместо кнопок входа и регистрации будет кнопка перехода в личный кабинет с миниатюрой пользовательского аватара и именем.

При нажатии на карточку проекта посетитель перейдёт на страницу выбранного проекта. Страница проекта представлена на рисунке 16.

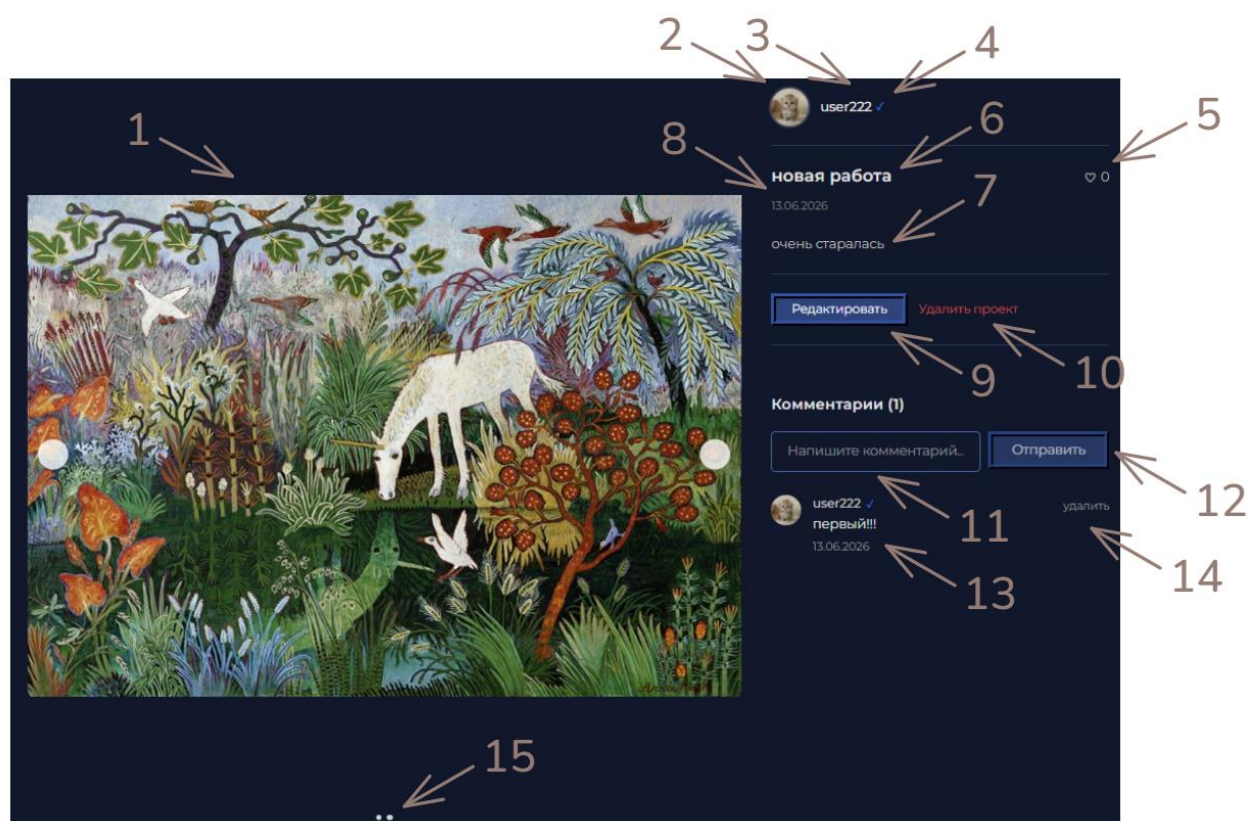


Рисунок 16 – Страница проекта

Основные элементы страницы проекта:

- 1 – изображение в проекте;
- 2 – аватар автора цифрового проекта;
- 3 – имя автора цифрового проекта;
- 4 – знак верификации аккаунта пользователя;
- 5 – количество лайков у цифрового проекта;

- 6 – название цифрового проекта;
- 7 – описание цифрового проекта;
- 8 – дата публикации цифрового проекта;
- 9 – кнопка редактирования проекта (только своего);
- 10 – кнопка удаление проекта (только своего);
- 11 – форма для написания комментария;
- 12 – кнопка публикации комментария;
- 13 – комментарий с никнеймом и аватаром его автора;
- 14 – кнопка удаления комментария (только своего);
- 15 – кнопка пагинации карусели.

Попытка неавторизованного пользователя поставить лайк, написать комментарий или подписаться на пользователя будет перенаправлять его на страницу входа в свой аккаунт. Форма входа представлена на рисунке 17.

The image shows a login form with a dark blue background. At the top, the title 'Вход' is centered. Below it are two input fields: 'Email' containing 'user222@test.com' and 'Пароль' (Password) with masked characters. A blue button labeled 'Войти' (Login) is positioned below the password field. At the bottom, there is a link that says 'Нет аккаунта? Зарегистрироваться' (No account? Register). Four numbered arrows point to specific elements: arrow 1 points to the Email field, arrow 2 points to the Password field, arrow 3 points to the 'Войти' button, and arrow 4 points to the registration link.

Рисунок 17 – Форма входа

Основные элементы страницы проекта:

- 1 – поле для ввода электронной почты;
- 2 – поле для ввода пароля;

- 3 – кнопка входа в аккаунт;
- 4 – ссылка на форму регистрации.

Если у неавторизованного пользователя нет своего аккаунта на платформе, он может создать его, заполнив форму регистрации (рисунок 18).

The image shows a registration form titled "Регистрация" (Registration) on a dark blue background. The form contains the following elements:

- 1**: Points to the "Email" input field, which contains the text "user222@test.com".
- 2**: Points to the "Пароль" (Password) input field, which contains a series of dots.
- 3**: Points to the "Повторите пароль" (Repeat password) input field, which is currently empty.
- 4**: Points to the "Зарегистрироваться" (Register) button.
- 5**: Points to the "Войти" (Login) link, which is preceded by the text "Уже есть аккаунт?" (Already have an account?).

Рисунок 18 – Форма регистрации

Основные элементы страницы проекта:

- 1 – поле для ввода электронной почты;
- 2 – поле для ввода пароля;
- 3 – поле для повторного ввода пароля;
- 4 – кнопка регистрации;
- 5 – ссылка на форму входа.

Авторизованный пользователь имеет доступ к своему личному кабинету. Интерфейс личного пространства структурирован таким образом, чтобы обеспечить быстрый переход к ключевым инструментам создания контента и настройкам учетной записи (рисунок 19).

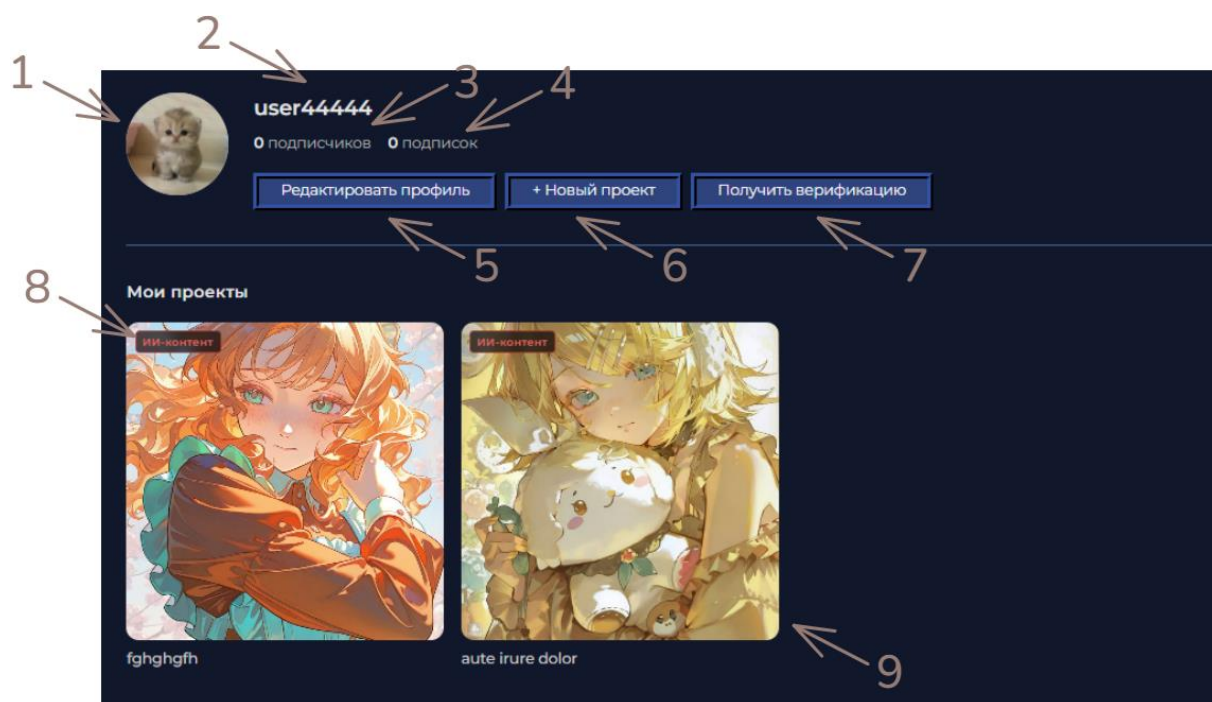


Рисунок 19 – Личный кабинет

Основные элементы страницы личного кабинета:

- 1 – аватар пользователя;
- 2 – имя пользователя;
- 3 – количество подписчиков пользователя;
- 4 – количество подписок пользователя;
- 5 – кнопка редактирования профиля;
- 6 – кнопка создания цифрового проекта;
- 7 – кнопка получения верификации;
- 8 – тэг «ИИ-контент»;
- 9 – карточка цифрового проекта.

Кнопка «+ Проект» при нажатии инициирует переход на страницу с формой создания проекта. После добавления медиафайлов через область загрузки становится доступной кнопка «Проверить файлы», запускающая их двухэтапную валидацию. Если у файлов некорректные параметры, по окончании первого этапа проверки она прервется, и в правом нижнем углу появится список ошибок. В противном случае в нижней части формы появится уведомление о завершении проверки на факт вмешательства ИИ. Если хотя бы одно

изображение было сгенерировано, то опубликованный цифровой проект будет отображаться с меткой «ИИ-контент». Форма создания цифрового проекта представлена на рисунке 20.

The image shows a digital form titled "Новый проект" (New Project) on a dark blue background. The form contains several sections and fields, with numbered arrows pointing to specific elements:

- 1** points to a dashed rectangular box for uploading project images, with the text "Изображения проекта (макс. 20)" and "Перетащите файлы или нажмите для выбора JPG, PNG - до 50 МБ".
- 2** points to two small square thumbnails of landscape paintings.
- 3** points to a blue button with a checkmark and the text "✓ Файлы проверены".
- 4** points to a text input field labeled "Название проекта *" containing the text "первый день весны".
- 5** points to a larger text input field labeled "Описание" containing the text "я посветил эту работу....".
- 6** points to a light green box containing a checkmark, the text "✓ Реальный контент (не ИИ)", and "Уверенность в ИИ: 0%".
- 7** points to a blue button at the bottom with the text "Опубликовать проект".

Рисунок 20 – Форма создания проекта

Основные элементы формы создания проекта:

- 1 – область загрузки медиафайлов;
- 2 – миниатюры загруженных медиафайлов по порядку отображения в итоговом проекте;
- 3 – кнопка проверки медиафайлов на генерацию (обязательная);
- 4 – поле ввода названия цифрового проекта;
- 5 – поле ввода описания цифрового проекта;

- 6 – результат проверки медиафайлов на генерацию;
- 7 – кнопка публикации цифрового проекта.

Пользователь также имеет возможность изменить данные своего аккаунта на странице редактирования профиля (рисунок 21).

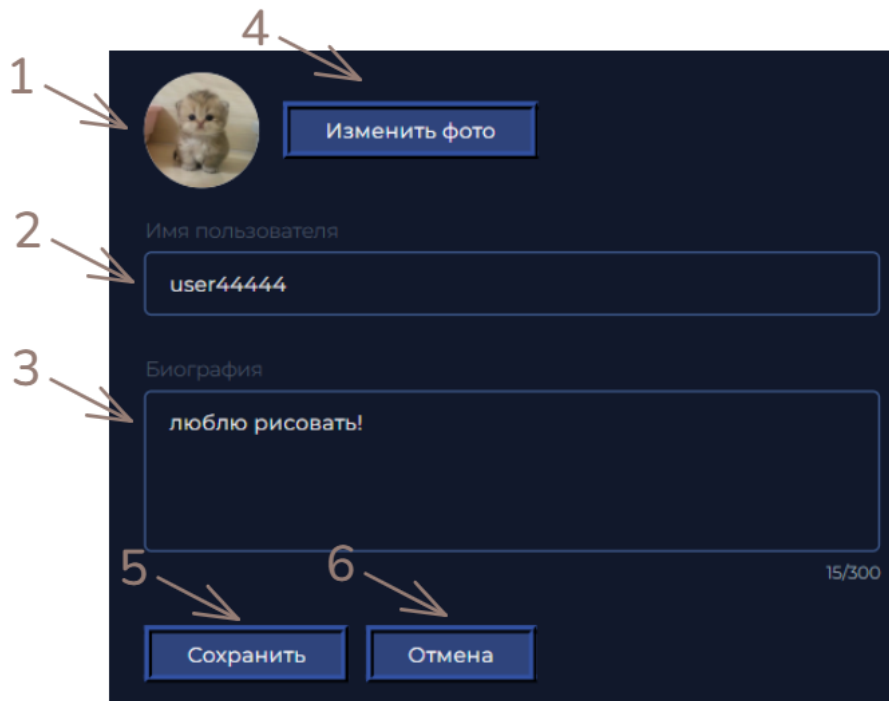


Рисунок 21 – Страница редактирования профиля

Основные элементы страницы редактирования профиля:

- 1 – аватар пользователя;
- 2 – поле ввода нового имени пользователя;
- 3 – поле ввода личной информации о пользователе;
- 4 – кнопка изменения аватара;
- 5 – кнопка сохранения изменений;
- 6 – кнопка отмены.

Таким образом, разработанный интерфейс предоставляет пользователям онлайн-платформы для продвижения произведений цифрового искусства полный доступ ко всем функциям и возможностям платформы.

4 Контроль качества программных средств

4.1 Планирование процесса контроля качества программных средств

4.1.1 Обоснование выбора модели тестирования

Практика разработки программных продуктов подтверждает, что ни одна реализация серверной части не обходится без тестирования, направленного на выявление дефектов и оценку соответствия системы заявленным характеристикам.

Наиболее распространенные модели тестирования включают:

- модульное тестирование – проверка работы отдельных модулей и компонентов в изолированной среде;
- интеграционное тестирование – контроль корректности взаимодействия между составными частями системы;
- функциональное тестирование – проверка соответствия реализованных функций заявленным требованиям;
- тестирование производительности – оценка быстродействия при обработке запросов, поведения системы под нагрузкой и её способности к горизонтальному или вертикальному масштабированию.

В данной выпускной квалификационной работе был сделан акцент на внедрении в разработку онлайн-платформы функционального и модульного видов тестирования.

Метод чёрного ящика – вид функционального тестирования, при котором тест-кейсы строятся строго по спецификациям, без знания внутренней реализации. В работе данный метод применён для проверки процесса загрузки изображений на платформу с использованием классов эквивалентности и граничных условий.

Модульное тестирование применялось для верификации сервиса проверки загружаемых на платформу медиафайлов. Остальные виды тестирования

запланированы к реализации на последующих итерациях разработки онлайн-платформы.

4.1.2 Обоснование выбора метрик программного кода

Одним из основных средств анализа качества ПО являются метрики программного кода. За счет их применения удастся получить развернутую оценку технических аспектов продукта, спрогнозировать требуемые ресурсы на разработку, измерить сложность и уровень надежности создаваемых систем.

Исходя из специфики предметной области, для всестороннего охвата характеристик был определен следующий набор оценочных параметров. Для проверки качества кода онлайн-платформы для продвижения цифрового искусства были выбраны такие метрики кода, как метрика SLOC, метрика Холстеда и метрика Чепина.

Количество строк кода (англ. Source Lines of Code, SLOC) – числовой показатель масштаба программного продукта, основанный на подсчёте строк в его исходном коде.

Метрика Холстеда строится на совокупном анализе операторов, операндов и строк программного кода. Она даёт возможность оценить вычислительную сложность программных модулей.

Инструментом оценивания качества внутренней логики модулей программы является метрика Чепина. Она оценивает количество и типы переменных, которые используются в модуле. В её основе лежит классификация переменных, задействованных в модуле, по количеству и характеру использования.

Таким образом, каждый из выбранных показателей отражает конкретный аспект оценки качества программного продукта. Их совместное использование при анализе онлайн-платформы позволяет выявить потенциальные проблемы, связанные с объемом, сложностью и структурой кода, а также определить направления его дальнейшей оптимизации.

4.2 Тестирование программных средств

4.2.1 Функциональное тестирование

Классы эквивалентности представляют собой совокупности входных значений, при подаче которых программа демонстрирует идентичное поведение или формирует одинаковый результат. Применение данного метода позволяет разделить область входных данных на классы и в дальнейшем сформировать минимально достаточный набор тестовых сценариев для проверки корректности работы программного продукта.

В рамках выполнения работы был рассмотрен параметр загружаемого изображения, являющегося ключевым для создания цифровых проектов. Классы эквивалентности в табличном виде представлены в таблице 15.

Таблица 15 – Классы эквивалентности поля изображения

	Правильные КЭ	Неправильные КЭ
Расширение файла	.png, (1) .jpeg (2)	Отсутствует формат (3) Другой формат (4)
Размер файла	.png $\geq$ 67 байт, (5) .jpeg $\geq$ 159 байт (6) $\leq$ 50 МБ (7)	.png < 67 байт (8) .jpeg < 159 байт (9) > 50 МБ (10)
Целостность файла	Файл не поврежден (11)	Файл поврежден (12)

Для данных классов были составлены соответствующие тест-кейсы, представленные в таблице 16.

Результаты тестирования кейсов для классов эквивалентности представлены в приложении И.

Таблица 16 – Тест-кейсы для классов эквивалентности

№	КЭ	Входные данные	Выходные данные	Фактический результат
1	1, 5, 7, 11	default-avatar.png (255 КБ)	Успех	Результат совпадает с ожидаемыми выходными данными (рис. И.1)
2	2, 6, 7, 11	default-avatar.jpeg (255 КБ)	Успех	Результат совпадает с ожидаемыми выходными данными (рис. И.2)

Продолжение таблицы 16

№	КЭ	Входные данные	Выходные данные	Фактический результат
3	3, 7, 11	default-avatar	Ошибка: неподдерживаемый формат	Результат совпадает с ожидаемыми выходными данными (рис. И.3)
4	4, 7, 11	default-avatar.txt	Ошибка: неподдерживаемый формат	Результат совпадает с ожидаемыми выходными данными (рис. И.4)
5	1, 8, 7, 11	default-avatar.png (30 байт)	Ошибка: слишком маленький размер файла (.png)	Результат совпадает с ожидаемыми выходными данными (рис. И.6)
6	2, 9, 7, 11	default-avatar.jpg (30 байт)	Ошибка: слишком маленький размер файла (.jpeg)	Результат совпадает с ожидаемыми выходными данными (рис. И.7)
7	1, 10, 11	default-avatar.png (60 МБ)	Ошибка: слишком большой размер файла (макс. 50 МБ)	Результат совпадает с ожидаемыми выходными данными (рис. И.8)
8	2, 7, 12	Повреждённый default-avatar.jpg	Ошибка: файл повреждён или имеет неверную структуру	Результат совпадает с ожидаемыми выходными данными (рис. И.9)

Каждый тест-кейс проверяет один неправильный класс эквивалентности при сохранении остальных параметров в правильных классах, что позволяет локализовать возможные ошибки.

В результате применения метода было выделено 7 правильных и 5 неправильных классов эквивалентности и составлено 9 тест-кейсов. Каждый тест-кейс проверяет один неправильный класс эквивалентности при сохранении остальных параметров в правильных классах, что позволяет локализовать возможные ошибки.

Граничные условия к полю изображений приведены в таблице 17.

Таблица 17 – Граничные условия

Границы	
Физические	Размер файла = 0 бит
Логические	Расширения файла не свойственное изображениям
Технологические	Размер файла меньше минимального (PNG < 67 байт, JPG < 159 байт) Поврежденный файл
Произвольные	Размер файла > 50 МБ

По выделенным граничным условиям были определены соответствующие тест-кейсы, представленные в таблице 18. Результаты тестирования тест-кейсов для граничных условий приведены в приложении И.

Таблица 18 – Тест-кейсы для граничных условий

№	Границы	Входные данные	Выходные данные	Фактический результат
1	Ф1	Файл размером 0 бит	Ошибка: некорректный размер файла	Результат совпадает с ожидаемыми выходными данными (рис. И.10)
2	Л1	image.txt	Ошибка: расширение файла не свойственное изображениям	Результат совпадает с ожидаемыми выходными данными (рис. И.11)
3	T1.1	image.png (66 байт)	Ошибка: слишком маленький размер файла .png	Результат совпадает с ожидаемыми выходными данными (рис. И.12)
4	T1.2	image.jpg (158 байт)	Ошибка: слишком маленький размер файла .jpg	Результат совпадает с ожидаемыми выходными данными (рис. И.13)
5	T2	Повреждённый image.png	Ошибка: файл повреждён или имеет неверную структуру	Результат совпадает с ожидаемыми выходными данными (рис. И.14)
6	П1	huge_image.jpg (размер: 60 МБ)	Ошибка: слишком большой размер файла (макс. 50 МБ)	Результат совпадает с ожидаемыми выходными данными (рис. И.15)
7	T1.1	image.png (67 байт)	Успех	Результат совпадает с ожидаемыми выходными данными (рис. И.16)
8	T1.2	image.jpg (159 байт)	Успех	Результат совпадает с ожидаемыми выходными данными (рис. И.17)
9	П1	image.jpg (50 МБ)	Успех	Результат совпадает с ожидаемыми выходными данными (рис. И.18)
10	П1	image.jpg (50 МБ + 1 байт)	Ошибка: слишком большой размер файла (макс. 50 МБ)	Результат совпадает с ожидаемыми выходными данными (рис. И.19)

Каждый тест-кейс охватывает граничные условия, подлежащие проверке на соответствие ожидаемым значениям, а также определяет входные и выходные данные для конкретного сценария.

4.2.2 Модульное тестирование

Для создания набора юнит-тестов использовался стек технологий, входящий в состав зависимости spring-boot-starter-test. Для проведения

тестирования был спроектирован класс `ProjectValidationService`, содержащий логику валидации файлов изображений, названия и описания цифрового проекта.

Были реализованы следующие группы тестов, покрывающие разработанный список причин и следствий: группа «Количество файлов», группа «Расширение файла», группа «Размер файла», группа «Целостность файла», группа «Название и описание».

Группа юнит-тестов, проверяющих количество изображений в цифровом проекте, приведена в листинге К.1 в приложении К.

Результат работы всех групп юнит-тестов представлен на рисунке 22.

✓	Группа 3: размер файла	1 sec 56 ms
✓	.jpeg 60 МБ → C5: размер превышает 50 МБ	1 sec 37 ms
✓	.jpeg ровно 50 МБ → успех (граница)	4 ms
✓	.jpeg ровно 159 байт → успех (граница)	4 ms
✓	.jpeg 30 байт → C4: слишком маленький	4 ms
✓	.png 30 байт → C4: слишком маленький	3 ms
✓	.png ровно 67 байт → успех (граница)	4 ms
✓	Группа 2: расширение файла	17 ms
✓	расширение .txt → C3: неподдерживаемое расшир	6 ms
✓	расширение .png → проходит проверку расширен	4 ms
✓	расширение .jpeg → проходит проверку расширен	3 ms
✓	файл без расширения → C3: нет расширения	4 ms
✓	Группа 4: целостность файла	15 ms
✓	неверная сигнатура JPEG → C6: файл повреждён	4 ms
✓	корректная PNG-сигнатура → не повреждён	3 ms
✓	корректная JPEG-сигнатура → не повреждён	4 ms
✓	неверная сигнатура PNG → C6: файл повреждён	4 ms
✓	Группа 5: название и описание	20 ms
✓	описание = null → успех	3 ms
✓	название = null → успех	4 ms
✓	название 201 символ → C7: название слишком дли	3 ms
✓	описание ровно 2000 символов → успех (граница)	3 ms
✓	название ровно 200 символов → успех (граница)	3 ms
✓	описание 2001 символ → C8: описание слишком д	4 ms
✓	Группа 1: количество файлов	12 ms
✓	1 корректный файл → успех	3 ms
✓	10 корректных файлов → успех	5 ms
✓	11 файлов → C2: максимум 10 изображений	2 ms
✓	files = null → C1: добавить хотя бы одно изображен	2 ms

Рисунок 22 – Результат работы всех групп юнит-тестов

Тестирование показало, что часть программы, ответственная за валидацию данных цифровых проектов функционирует согласно требованиям.

4.3 Оценка качества программного кода

4.3.1 Метрика SLOC

Для оценки качества программного кода платформы был выбран класс `ProjectValidationService` из сервиса медиафайлов, предназначенный для валидации данных из формы добавления цифрового проекта.

Для расчета SLOC был использован плагин `MetricsReloaded` для `IntelliJIDEA`. Параметр подсчета `Lines of code` дал результаты, представленные на рисунке 23.

Method metrics					
Class metrics					
Package metrics					
Module metrics					
File type metrics					
Project metrics					
method	^	CLOC	JLOC	LOC	NCLOC
com.tpo.lr2.service.ProjectValidationService.isValidImage(byte[], String)		2	0	17	15
com.tpo.lr2.service.ProjectValidationService.validateFile(MultipartFile, List)		3	0	27	24
com.tpo.lr2.service.ProjectValidationService.validate(List<MultipartFile>,		4	0	27	23
Total		9	0	71	62

Рисунок 23 – Результат работы `Lines of code`

Физическими строками считаются все непустые строки текстового файла.

В `ProjectValidationService` насчитывается:

- 71 физическая строка кода;
- 62 логические строки кода;
- 9 строк комментариев.

Данные значения являются нормальными для сервисного класса с тремя методами валидации.

4.3.2 Метрика Холстеда

В основе метрики Холстеда лежат четыре первичных измеримых характеристики, описанные в таблице 18.

Таблица 18 – Характеристики программы

Характеристика	Определение
NUOprtr (Number of Unique Operators)	Количество уникальных операторов программного модуля, образующих словарь операторов

Продолжение таблицы 18

Характеристика	Определение
NUOprnd (Number of Unique Operands)	Количество уникальных операндов программного модуля, составляющих словарь операндов
Noprtr (Number of Operators)	Суммарное число операторов в программном модуле
Noprnd (Number of Operands)	Суммарное число операндов в программном модуле

Подсчет операторов и операндов в классе ProjectValidationService приведен на рисунке 24.

Операторы	Число вхождений	Операнды	Число вхождений
if	10	files	4
else if	4	title	4
else	1	description	4
for	1	errors	9
return	5	file	3
=	6	filename	4
	3	size	7
&&	7	ext	8
!	4	bytes	8
>	4	MAX_SIZE	1
<	3	MIN_PNG_SIZE	1
>=	2	MIN_JPEG_SIZE	1
==	8	MAX_TITLE_LENGTH	1
.add()	8	MAX_DESC_LENGTH	1
.equals()	7	MAX_IMAGES	1
.isEmpty()	1	".png"	3
.size()	2	".jpeg"	3
.contains()	1	"."	2
.substring()	1	0x89	1
.lastIndexOf()	1	0x50	1
.toLowerCase()	1	0x4E	1
.getOriginalFilename()	1	0x47	1
.getSize()	1	0xFF	2
.getBytes()	1	0xD8	1
.length()	2	FALSE	1
new ArrayList<>()	1		
(byte)	7		
[]	7		
n1	N1	n2	N2
28	100	25	73

Рисунок 24 – Подсчет операторов и операндов в ProjectValidationService

Длина программы рассчитывается по формуле:

$$N=N_1+N_2=173.$$

Словарь программы рассчитывается по формуле:

$$n=n_1+n_2=53.$$

Объем программы рассчитывается по формуле:

$$HV=N*\log_2 n=990,9.$$

Сложность программы рассчитывается по формуле:

$$HN=\frac{n_1}{2}*\frac{N_2}{n_2}=40,9.$$

Полученное значение по методу Холстеда соответствует среднему уровню сложности анализируемого программного модуля. Подобный результат обусловлен большим числом используемых условных конструкций и операндов, проверяющих загружаемые на онлайн-платформу медиафайлы на корректность.

4.3.3 Метрика Чепина

Метрика Чепина оценивает связность программного модуля через анализ переменных ввода-вывода.

Набор переменных делится на функциональные группы, описанные в таблице 19.

Таблица 19 – Функциональные группы переменных по Чепину

Группа	Определение
P (Input)	Переменные, используемые в вычислениях и выводе результатов
M (Modified)	Переменные, создаваемые и изменяемые внутри модуля
C (Control)	Переменные, определяющие логику выполнения модуля
T (Tautological/Parasitic)	Неиспользованные переменные

Одна переменная может входить в несколько функциональных групп одновременно.

Метрика Чепина рассчитывается по следующей формуле:

$$Q=1*P+2*M+3*C+0,5*T.$$

Результат разбиения переменных сервиса на функциональные группы по Чепину показан на рисунке 25.

Переменная	Группа
files	P C
title	P C
description	P C
errors	M
file	P
filename	P C
size	P C
ext	P C
bytes	P C
MAX_SIZE	P C
MIN_PNG_SIZE	P C
MIN_JPEG_SIZE	P C
MAX_TITLE_LENGTH	P C
MAX_DESC_LENGTH	P C
MAX_IMAGES	P C

Рисунок 25 – Разбиение переменных на группы

В представленных функциях:

– $P = 14$;

– $M = 1$;

– $C = 13$;

– $T = 0$.

Итоговое значение метрики Чепина:

$$Q = 1 * 14 + 2 * 1 + 3 * 13 + 0,5 * 0 = 55.$$

Высокое значение Q обусловлено тем что большинство входных переменных одновременно участвуют в условных проверках, являясь и управляющими.

Заключение

В рамках выпускной квалификационной работы были разработаны программные средства для продвижения произведений цифрового искусства.

Был проведен анализ предметной области арт-платформ и бизнес-процессов в сфере диджитал-искусства. Сравнительный анализ популярных площадок аналогичного профиля, а именно ArtStation, «Cara» и «Иллюстраторс.ру», позволил выявить их преимущества и недостатки. Опираясь на результаты сравнения были сформулированы функциональные и нефункциональные требования к проектируемой платформе, определены категории пользователей, построены диаграммы в нотации UML.

С помощью инструмента Project Libre был спланирован процесс разработки и рассчитан бюджет.

На этапе проектирования была определена микросервисная архитектура платформы, разработана концептуальная модель базы данных системы, а также сформулированы требования к аппаратному и программному обеспечению и пользовательскому интерфейсу.

Серверная составляющая онлайн-платформы реализована с помощью фреймворка Spring Boot (Java) в виде пяти микросервисов: сервиса авторизации, сервиса цифровых проектов, сервиса медиафайлов, сервиса профилей пользователей и сервиса обнаружения ИИ-контента. Сервис обнаружения ИИ-контента был разработан на Python с использованием фреймворка FastAPI и модели машинного обучения Organika/sd-xl-detector. Асинхронная межсервисная коммуникация организована через брокер сообщений Apache Kafka. Пользовательский интерфейс реализован на основе фреймворка Next.js 16 и библиотеки React 19 (TypeScript). Развёртывание всех компонентов системы выполнено посредством контейнеризации в Docker.

Контроль качества разработанных программных средств осуществлялся с использованием методов функционального и модульного тестирования. Функциональное тестирование включало использование классов эквивалентности и граничных условий. Модульное тестирование сервиса

валидации медиафайлов было реализовано с использованием стека spring-boot-starter-test. Оценка качественных характеристик исходного кода проводилась с применением метрик SLOC, Холстеда и Чепина.

В результате выполнения выпускной квалификационной работы была разработана онлайн-платформа для продвижения произведений цифрового искусства, обеспечивающие публикацию портфолио цифровых проектов, автоматическую маркировку и фильтрацию ИИ-контента, верификацию аккаунтов пользователей с подтверждением авторства, а также базовый функционал социальных сетей. Разработанная платформа отвечает современным требованиям качества и может использоваться цифровыми креаторами для продвижения своих работ и привлечения потенциальных заказчиков.

Список использованных источников

1. Креативная экономика государств-членов ШОС // Министерство экономического развития Российской Федерации. [Электронный ресурс]. URL: https://www.economy.gov.ru/material/file/e2feb4687edd650532aafcd7cb47aee4/Kreativnaya_ekonomika_gosudarstv-chlenov_ShOS.pdf (Дата обращения: 16.04.2026)
2. На ярмарке «Арт Россия 2025» обсудили, как цифровые инструменты помогают художникам стать заметнее и влиятельнее // <https://rg.ru>. [Электронный ресурс]. URL: https://rg.ru/2025/04/08/probivaia-skorlupu-materialnogo.html?utm_referrer=https%3A%2F%2Fwww.google.com%2F (Дата обращения: 16.04.2026)
3. Креативный сектор России в цифрах: 2024 // Институт статистических исследований и экономики знаний НИУ ВШЭ. [Электронный ресурс]. URL: <https://issek.hse.ru/mirror/pubs/share/996745056.pdf> (Дата обращения: 16.04.2026)
4. Салита С. В., Гутько Е. Ю. Арт-маркетинг и специфика его организации в условиях развития виртуальной среды в художественной деятельности // Прикладные экономические исследования. 2023. №S1. URL: <https://cyberleninka.ru/article/n/art-marketing-i-spetsifika-ego-organizatsii-v-usloviyah-razvitiya-virtualnoy-sredy-v-hudozhestvennoy-deyatelnosti> (Дата обращения: 16.04.2026)
5. Tatum Hunter Artists are fleeing Instagram to keep their work out of Meta's AI // The Washington Post. [Электронный ресурс]. URL: <https://www.washingtonpost.com/technology/2024/06/06/instagram-meta-ai-training-cara/> (Дата обращения: 16.04.2026)
6. Shawn Shan, Jenna Cryan, Emily Wenger, Haitao Zheng, Rana Hanocka, Ben Y. Zhao Glaze: Protecting Artists from Style Mimicry by Text-to-Image Models // Department of Computer Science, University of Chicago. URL: <https://arxiv.org/pdf/2302.04222> (Дата обращения: 16.04.2026)
7. Wei, W. The Evolution of Digital Art: From Early Experiments to Contemporary Practices // International Journal of Literature and Art Innovation. –

2024. – Vol. 1. – с. 44-49. – DOI 10.70088/kqba1d92. – URL: https://www.researchgate.net/publication/386161452_The_Evolution_of_Digital_Art_From_Early_Experiments_to_Contemporary_Practices (Дата обращения: 18.04.2026)

8. Create AI-Generated Art Fairly with DreamUp // DeviantArt. [Электронный ресурс]. URL: <https://www.deviantart.com/team/journal/Create-AI-Generated-Art-Fairly-with-DreamUp-933537821> (Дата обращения: 18.04.2026)

9. European Parliament and Council of the European Union, Regulation (EU) 2024/1689 of 13 June 2024 Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) // Official Journal of the European Union. [Электронный ресурс]. URL: <https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng> (Дата обращения: 18.04.2026)

10. Маркировка ИИ-контента в России: всё, что нужно знать бизнесу и пользователям // <https://developers.sber.ru>. [Электронный ресурс]. URL: <https://developers.sber.ru/help/business-development/labeling-of-ai-content> (Дата обращения: 18.04.2026)

11. Klimenko V. Особенности исследования целевой аудитории креативных кластеров / V. Klimenko, E. Chudinova, I. Dydykin // Экономические науки. – 2024. – № 241. – с. 586-591. – DOI 10.14451/1.241.586. – URL: https://www.researchgate.net/publication/391484506_Osobennosti_issledovania_celevoj_auditorii_kreativnyh_klasterov (Дата обращения: 18.04.2026).

12. Use of AI Software on ArtStation // ArtStation Help Center. [Электронный ресурс]. URL: https://help.artstation.com/s/article/11451085663501-Use-of-AI-Software-on-ArtStation?language=en_US (Дата обращения: 20.04.2026)

13. About Cara // Cara Blog. [Электронный ресурс]. URL: <https://blog.cara.app/about> (Дата обращения: 20.04.2026)

14. Cara Glaze: About // Cara Blog. [Электронный ресурс]. URL: <https://cara.app/blog/cara-glaze-about> (Дата обращения: 20.04.2026)

15. Создаем диаграмму Ганта // UNISENDER. [Электронный ресурс]. URL: <https://www.unisender.com/ru/blog/chto-takoe-diagramma-ganta-i-kak-ee-sdelat/> (Дата обращения: 21.04.2026)
16. Суйц В. П., Анушевский И. И. Форензик-экспертиза: сущность и основные методы организации финансовых расследований в компаниях // Вестник Московского университета. Серия 6. Экономика. 2019. №3. URL: <https://cyberleninka.ru/article/n/forenzik-ekspertiza-suschnost-i-osnovnye-metody-organizatsii-finansovyh-rassledovaniy-v-kompaniyah> (Дата обращения: 16.05.2026)
17. Ричардс Марк, Форд Нил P56 Фундаментальный подход к программной архитектуре: паттерны, свойства, проверенные методы. – СПб.: Питер, 2023. – 448 с.: ил. – (Серия «Для профессионалов»). (Дата обращения: 21.04.2026)
18. Database-per-Service Pattern for Microservices // GeeksforGeeks. [Электронный ресурс]. URL: <https://www.geeksforgeeks.org/system-design/database-per-service-pattern-for-microservices/> (Дата обращения: 21.04.2026)
19. Технологии интеграции информационных систем. Часть 2. GraphQL, gRPC, WebSocket, webhook, брокеры сообщений // Habr. [Электронный ресурс]. URL: <https://habr.com/ru/articles/849852/> (Дата обращения: 21.04.2026)
20. HTTP-методы для начинающих: разбираем по букве спецификаций, а не пересказам // Habr. [Электронный ресурс]. URL: <https://habr.com/ru/articles/1010240/> (Дата обращения: 21.04.2026)
21. SDXL Detector // Hugging Face [Электронный ресурс]. URL: <https://huggingface.co/Organika/sd-xl-detector> (дата обращения 05.06.2026)
22. Бочкарева А. А. Онлайн-платформа для продвижения произведений цифрового искусства // GitHub [Электронный ресурс]. URL: <https://github.com/lensomni/art-platform> (дата обращения 05.06.2026)

Приложение А
(справочное)
Глоссарий

API (Application Programming Interface) – формализованный контракт, определяющий способы взаимодействия между программными компонентами или системами.

DDD (Domain-Driven Design, предметно-ориентированное проектирование) – подход к проектированию программного обеспечения, при котором структура и логика системы отражают суть предметной области.

ГИИ (генеративный искусственный интеллект) – отрасль искусственного интеллекта, модели которой разрабатываются для генерации контента на основе обучающих наборов данных.

ИИ (искусственный интеллект) – отрасль информатики, занимающаяся созданием программных систем, способных выполнять задачи, требующие интеллекта, подобного человеческому.

JWT (JSON Web Token) – открытый стандарт передачи данных между сторонами в виде объекта JSON, используемый для аутентификации и авторизации пользователей.

ЖЦПО (жизненный цикл разработки программного обеспечения) – последовательность этапов, которые проходит программный продукт с момента зарождения идеи до вывода из эксплуатации.

MinIO – система объектного хранения с открытым исходным кодом, предназначенная для хранения неструктурированных данных: изображений, видео и других медиафайлов.

MVP (Minimum Viable Product, минимально жизнеспособный продукт) – версия продукта, содержащая минимальный набор функций, достаточный для полноценного использования и для проверки гипотезы о рыночном спросе на решение.

POLA (Principle of Least Astonishment, принцип наименьшего удивления) – принцип проектирования, гласящий, что поведение системы должно соответствовать ожиданиям пользователя.

SDXL (Stable Diffusion XL) – современная генеративная модель для создания изображений на основе текстовых описаний.

UML (Unified Modeling Language) – графический язык для визуализации, проектирования и документирования программных систем.

Брокер сообщений – это программный компонент, который обеспечивает асинхронный обмен сообщениями между сервисами в распределенной системе.

Верификация учетной записи – процесс подтверждения того, что опубликованные цифровые проекты принадлежат конкретному пользователю платформы.

Диджитал-арт (цифровое искусство) – произведения искусства или художественные практики, в которых использование информационных технологий является неотъемлемой частью.

Контейнеризация – технология изоляции приложений в независимые программные контейнеры, обеспечивающая согласованность их выполнения в разных средах.

Маркировка ИИ-контента – визуальная метка или метаданные, указывающие, что произведение было создано с использованием генеративного искусственного интеллекта.

Микросервисная архитектура – принцип проектирования ПО, при котором функционал системы декомпозируется на самостоятельные слабосвязанные модули с четко очерченной зоной ответственности.

Мягкое удаление – механизм скрытия данных от пользователя без их физического удаления из базы данных, обеспечивающий целостность данных и возможность аудита.

Отравление данных – метод защиты изображений от использования при обучении нейронных сетей путем внесения изменений на уровне пикселей, незаметных для человеческого глаза.

Состязательные искажения (adversarial perturbations) – малозаметные изменения изображения, которые вызывают ошибки в работе моделей нейронных сетей при его анализе.

Цифровой проект – единица опубликованного контента на платформе, состоящая из структурированной коллекции медиафайлов, объединенных общим творческим замыслом автора.

Цифровые креаторы – творческие личности, которые создают цифровые произведения искусства и распространяют свои работы на специализированных платформах.

Приложение Б
(рекомендуемое)
Расчет бюджета проекта

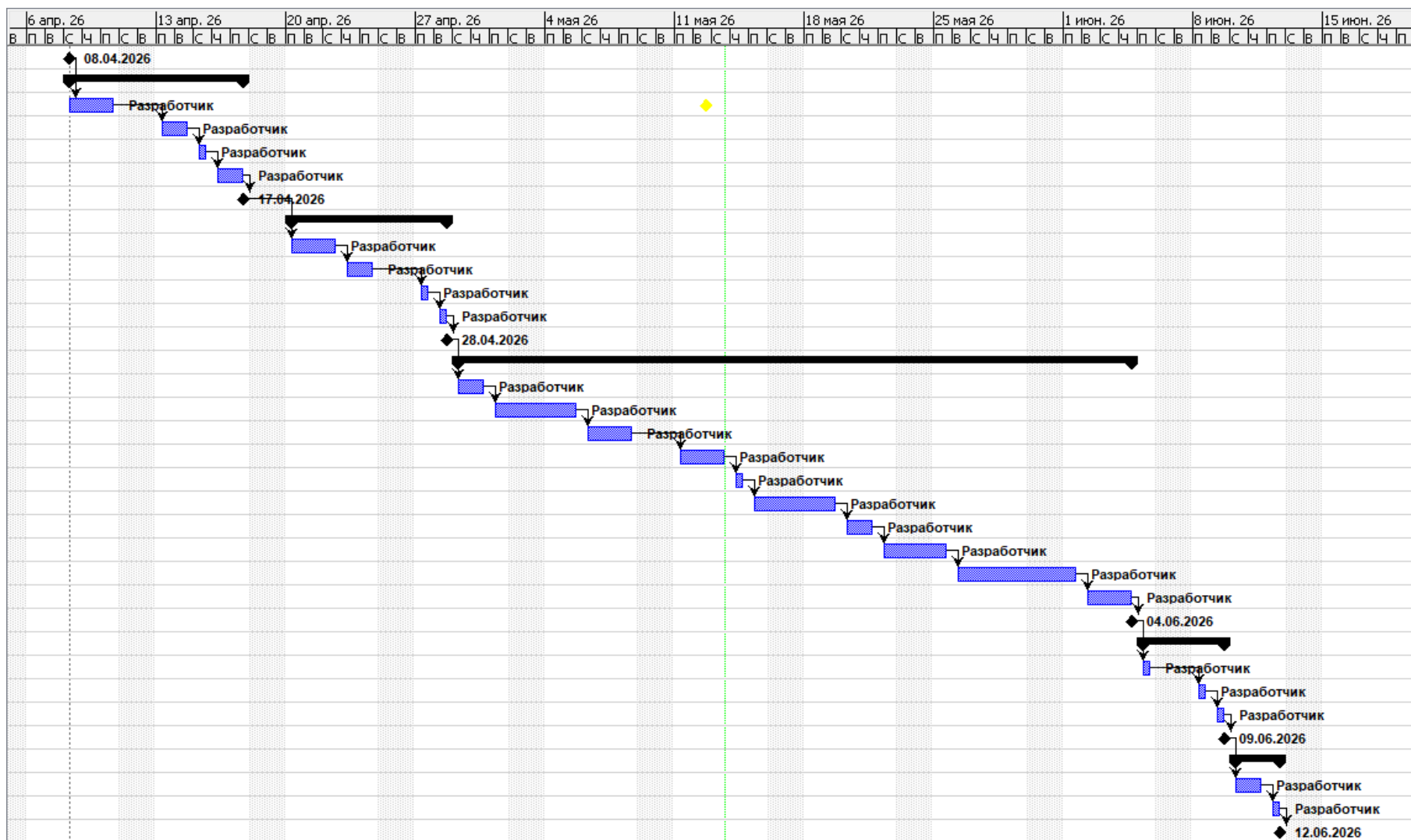


Рисунок Б.1 – Диаграмма Ганта

	⊗	Название	Продолжи...	Начало	Окончание	Предшествующие	Название ресурса
1		Начало проекта	0 дней	08.04.2026, 08:00	08.04.2026, 08:00		
2		☐ Анализ предметной области и постановка задачи	8 дней	08.04.2026, 08:00	17.04.2026, 17:00		
3		Изучение платформ-аналогов	3 дней	08.04.2026, 08:00	10.04.2026, 17:00	1	Разработчик
4		Определение функциональных требований	2 дней	13.04.2026, 08:00	14.04.2026, 17:00	3	Разработчик
5		Определение классов пользователей	1 день	15.04.2026, 08:00	15.04.2026, 17:00	4	Разработчик
6		Планирование процесса разработки	2 дней	16.04.2026, 08:00	17.04.2026, 17:00	5	Разработчик
7		Анализ предметной области завершён	0 дней	17.04.2026, 17:00	17.04.2026, 17:00	6	
8		☐ Проектирование программных средств	7 дней	20.04.2026, 08:00	28.04.2026, 17:00		
9		Проектирование архитектуры микросервисов	3 дней	20.04.2026, 08:00	22.04.2026, 17:00	7	Разработчик
10		Проектирование схем баз данных	2 дней	23.04.2026, 08:00	24.04.2026, 17:00	9	Разработчик
11		Разработка диаграммы вариантов использования	1 день	27.04.2026, 08:00	27.04.2026, 17:00	10	Разработчик
12		Разработка диаграммы компонентов	1 день	28.04.2026, 08:00	28.04.2026, 17:00	11	Разработчик
13		Проектирование завершено	0 дней	28.04.2026, 17:00	28.04.2026, 17:00	12	
14		☐ Разработка программных средств	27 дней	29.04.2026, 08:00	04.06.2026, 17:00		
15		Разработка шлюза API (gateway-service)	2 дней	29.04.2026, 08:00	30.04.2026, 17:00	13	Разработчик
16		Разработка сервиса авторизации (auth-service)	3 дней	01.05.2026, 08:00	05.05.2026, 17:00	15	Разработчик
17		Разработка сервиса профилей (user-profile-service)	3 дней	06.05.2026, 08:00	08.05.2026, 17:00	16	Разработчик
18		Разработка сервиса медиафайлов (media-service)	3 дней	11.05.2026, 08:00	13.05.2026, 17:00	17	Разработчик
19		Интеграция сервиса обнаружения ИИ	1 день	14.05.2026, 08:00	14.05.2026, 17:00	18	Разработчик
20		Разработка сервиса цифровых проектов (project-)	3 дней	15.05.2026, 08:00	19.05.2026, 17:00	19	Разработчик
21		Разработка сервиса комментариев (comment-ser-	2 дней	20.05.2026, 08:00	21.05.2026, 17:00	20	Разработчик
22		Разработка сервиса электронной почты (email-ser	2 дней	22.05.2026, 08:00	25.05.2026, 17:00	21	Разработчик
23		Разработка клиентского веб-приложения	5 дней	26.05.2026, 08:00	01.06.2026, 17:00	22	Разработчик
24		Интеграция клиентского приложения с backend	3 дней	02.06.2026, 08:00	04.06.2026, 17:00	23	Разработчик
25		Разработка программных средств завершена	0 дней	04.06.2026, 17:00	04.06.2026, 17:00	24	
26		☐ Контроль качества	3 дней	05.06.2026, 08:00	09.06.2026, 17:00		
27		Планирование процесса тестирования	1 день	05.06.2026, 08:00	05.06.2026, 17:00	25	Разработчик
28		Тестирование программных средств	1 день	08.06.2026, 08:00	08.06.2026, 17:00	27	Разработчик
29		Оценка качества программного кода	1 день	09.06.2026, 08:00	09.06.2026, 17:00	28	Разработчик
30		Контроль качества завершён	0 дней	09.06.2026, 17:00	09.06.2026, 17:00	29	
31		☐ Завершение работы над проектом	3 дней	10.06.2026, 08:00	12.06.2026, 17:00		
32		Оформление документации	2 дней	10.06.2026, 08:00	11.06.2026, 17:00	30	Разработчик
33		Сдача проекта	1 день	12.06.2026, 08:00	12.06.2026, 17:00	32	Разработчик
34		Проект завершён	0 дней	12.06.2026, 17:00	12.06.2026, 17:00	33	

Рисунок Б.2 – Список задач проекта

⊗	Название	Тип	Стандартная ставка	Ставка сверхурочных	Начислять	Основной календарь
👤	Разработчик	Работа	300,00 Р/час	400,00 Р/час	Пропорционально	Пятидневка

Рисунок Б.3 – Список ресурсов проекта

Приложение В
(рекомендуемое)
Спецификация атрибутов таблиц БД

Таблица В.1 – Спецификация атрибутов таблицы roles

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
1	id	Идентификатор роли	SMALLSERIAL	-	PK	Да	-	-
2	name	Наименование роли	VARCHAR	50	-	Да	UNIQUE	-

Таблица В.2 – Спецификация атрибутов таблицы auth_credentials

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
1	id	Идентификатор учётных данных	BIGSERIAL	-	PK	Да	-	-
2	email	Адрес электронной почты	VARCHAR	255	-	Да	UNIQUE	-
3	password_hash	Хэш пароля	VARCHAR	255	-	Да	-	-
4	role_id	Идентификатор роли	SMALLINT	-	FK	Да	REFERENCES roles(id)	1
5	is_active	Признак активности аккаунта	BOOLEAN	-	-	Да	-	TRUE
6	created_at	Дата и время создания	TIMESTAMPTZ	-	-	Нет	-	CURRENT_TIMESTAMP
7	updated_at	Дата и время последнего обновления	TIMESTAMPTZ	-	-	Нет	-	CURRENT_TIMESTAMP

Таблица В.3 – Спецификация атрибутов таблицы password_resets

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
1	id	Идентификатор записи	BIGSERIAL	-	PK	Да	-	-

Продолжение таблицы В.3

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
2	user_id	Идентификатор пользователя	BIGINT	-	FK	Да	REFERENCES auth_credentials(id) ON DELETE CASCADE	-
3	reset_token	Токен для сброса пароля	VARCHAR	64	-	Да	UNIQUE	-
4	expires_at	Дата и время истечения токена	TIMESTAMPTZ	-	-	Да	-	-
5	created_at	Дата создания запроса	TIMESTAMPTZ	-	-	Нет	-	CURRENT_TIMESTAMP
6	used_at	Дата использования токена	TIMESTAMPTZ	-	-	Нет	-	NULL

Таблица В.4 – Спецификация атрибутов таблицы rejection_reasons

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
1	id	Идентификатор причины отказа	SMALLSERIAL	-	PK	Да	-	-
2	code	Код причины отказа	VARCHAR	50	-	Да	UNIQUE	-
3	title	Наименование причины отказа	VARCHAR	200	-	Да	-	-

Таблица В.5 – Спецификация атрибутов таблицы verification_requests

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
1	id	Идентификатор заявки	BIGSERIAL	-	PK	Да	-	-
2	user_id	Идентификатор пользователя	BIGINT	-	FK	Да	REFERENCES auth_credentials(id) ON DELETE CASCADE	-
3	status	Статус заявки	VARCHAR	20	-	Да	CHECK (status IN ('PENDING', 'APPROVED', 'REJECTED'))	'APPROVED'
4	submitted_at	Дата подачи заявки	TIMESTAMPTZ	-	-	Нет	-	CURRENT_TIMESTAMP
5	reviewed_at	Дата рассмотрения заявки	TIMESTAMPTZ	-	-	Нет	-	NULL
6	reviewed_by	Идентификатор администратора	BIGINT	-	FK	Нет	REFERENCES auth_credentials(id)	NULL
7	rejection_reason_id	Идентификатор причины отказа	SMALLINT	-	FK	Нет	REFERENCES rejection_reasons(id) ON DELETE SET NULL	NULL
8	rejection_comment	Комментарий при отказе	TEXT	-	-	Нет	-	NULL

Таблица В.6 – Спецификация атрибутов таблицы profiles

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
1	id	Идентификатор профиля	BIGSERIAL	-	PK	Да	-	-
2	auth_id	Идентификатор учётных данных	BIGINT	-	-	Да	UNIQUE NOT NULL	-

Продолжение таблицы В.6

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
3	username	Имя пользователя (никнейм)	VARCHAR	50	-	Да	UNIQUE NOT NULL	-
4	bio	Биография пользователя	TEXT	-	-	Нет	-	NULL
5	avatar_url	Ссылка на аватар	VARCHAR	512	-	Да	-	'http://minio:9000/avatars/default.png'
6	is_verified	Признак верификации профиля	BOOLEAN	-	-	Да	-	FALSE
7	created_at	Дата создания профиля	TIMESTAMP PTZ	-	-	Нет	-	CURRENT_TIMESTAMP
8	last_active_at	Дата последней активности	TIMESTAMP PTZ	-	-	Нет	-	CURRENT_TIMESTAMP
9	is_active	Признак активности профиля	BOOLEAN	-	-	Да	-	TRUE

Таблица В.7 – Спецификация атрибутов таблицы subscriptions

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
1	id	Идентификатор подписки	BIGSERIAL	-	PK	Да	-	-
2	follower_id	Идентификатор подписчика	BIGINT	-	FK	Да	REFERENCES profiles(id) ON DELETE CASCADE	-
3	following_id	Идентификатор автора	BIGINT	-	FK	Да	REFERENCES profiles(id) ON DELETE CASCADE	-
4	created_at	Дата подписки	TIMESTAMP Z	-	-	Нет	-	CURRENT_TIMESTAMP

Таблица В.8 – Спецификация атрибутов таблицы digital_projects

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
1	id	Идентификатор проекта	BIGSERIAL	-	PK	Да	-	-
2	creator_id	Идентификатор создателя проекта	BIGINT	-	-	Да	NOT NULL	-
3	title	Название проекта	VARCHAR	255	-	Да	NOT NULL	-
4	description	Описание проекта	TEXT	-	-	Нет	-	NULL
5	is_ai	Признак ИИ-генерации	BOOLEAN	-	-	Да	-	FALSE
6	is_active	Признак активности проекта	BOOLEAN	-	-	Да	-	TRUE
7	created_at	Дата создания проекта	TIMESTAMPTZ	-	-	Нет	-	CURRENT_TIMESTAMP
8	updated_at	Дата последнего обновления	TIMESTAMPTZ	-	-	Нет	-	CURRENT_TIMESTAMP

Таблица В.9 – Спецификация атрибутов таблицы project_images

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
1	id	Идентификатор изображения	BIGSERIAL	-	PK	Да	-	-
2	project_id	Идентификатор проекта	BIGINT	-	FK	Да	REFERENCES digital_projects(id) ON DELETE CASCADE	-
3	image_url	Ссылка на изображение	VARCHAR	512	-	Да	NOT NULL	-
4	image_order	Порядок отображения	SMALLINT	-	-	Да	CHECK (image_order BETWEEN 0 AND 19)	-

Таблица В.10 – Спецификация атрибутов таблицы comments

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
1	id	Идентификатор комментария	BIGSERIAL	-	PK	Да	-	-
2	project_id	Идентификатор проекта	BIGINT	-	FK	Да	REFERENCES digital_projects(id) ON DELETE CASCADE	-
3	user_id	Идентификатор автора комментария	BIGINT	-	-	Да	NOT NULL	-
4	content	Текст комментария	TEXT	-	-	Да	NOT NULL	-
5	is_edited	Признак редактирования	BOOLEAN	-	-	Да	-	FALSE
6	created_at	Дата создания комментария	TIMESTAMPTZ	-	-	Нет	-	CURRENT_TIMESTAMP
7	updated_at	Дата последнего редактирования	TIMESTAMPTZ	-	-	Нет	-	NULL
8	is_active	Признак активности комментария	BOOLEAN	-	-	Да	-	TRUE

Таблица В.11 – Спецификация атрибутов таблицы likes

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
1	id	Идентификатор лайка	BIGSERIAL	-	PK	Да	-	-
2	project_id	Идентификатор проекта	BIGINT	-	FK	Да	REFERENCES digital_projects(id) ON DELETE CASCADE	-

Продолжение таблицы В.11

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения	Значение по умолчанию
3	user_id	Идентификатор пользователя	BIGINT	-	-	Да	NOT NULL	-
4	created_at	Дата постановки лайка	TIMESTAMPTZ	-	-	Нет	-	CURRENT_TIMESTAMP

Приложение Г
(рекомендуемое)
SQL-скрипты создания БД

Листинг Г.1 – Скрипт создания БД auth_db

```
-- Роли пользователей
CREATE TABLE roles (
    id SMALLSERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL UNIQUE
);

-- Учётные данные
CREATE TABLE auth_credentials (
    id BIGSERIAL PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role_id SMALLINT NOT NULL REFERENCES roles(id) DEFAULT
1,
    is_active BOOLEAN NOT NULL DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_auth_email ON auth_credentials(email);
CREATE INDEX idx_auth_role_id ON auth_credentials(role_id);

INSERT INTO rejection_reasons (code, title) VALUES
    ('invalid_personal_info', 'Ошибки в предоставлении личной
информации'),
    ('poor_document_quality', 'Недостаточное качество
документов'),
    ('invalid_or_incomplete_docs', 'Недействительные или неполные
документы'),
    ('suspicious_activity', 'Подозрительная активность');
```

Листинг Г.2 – Скрипт создания БД profile_db

```
CREATE TABLE profiles (
    id BIGSERIAL PRIMARY KEY,
    auth_id BIGINT UNIQUE NOT NULL,
    username VARCHAR(50) UNIQUE NOT NULL,
    bio TEXT,
    avatar_url VARCHAR(512) NOT NULL DEFAULT
'http://minio:9000/avatars/default.png',
    is_verified BOOLEAN NOT NULL DEFAULT FALSE,
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    last_active_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    is_active BOOLEAN NOT NULL DEFAULT TRUE
);

CREATE INDEX idx_profiles_username ON profiles(username);
CREATE INDEX idx_profiles_auth_id ON profiles(auth_id);
CREATE INDEX idx_profiles_verified ON profiles(is_verified);

-- Подписки
CREATE TABLE subscriptions (
```

```

        id          BIGSERIAL PRIMARY KEY,
        follower_id BIGINT NOT NULL REFERENCES profiles(id) ON DELETE
CASCADE,
        following_id BIGINT NOT NULL REFERENCES profiles(id) ON DELETE
CASCADE,
        created_at   TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
        UNIQUE (follower_id, following_id),
        CHECK (follower_id <> following_id)
);

CREATE INDEX idx_subscriptions_follower ON
subscriptions(follower_id);
CREATE INDEX idx_subscriptions_following ON
subscriptions(following_id);

```

Листинг Г.3 – Скрипт создания БД project_db

```

CREATE TABLE digital_projects (
    id          BIGSERIAL PRIMARY KEY,
    creator_id  BIGINT NOT NULL,
    title       VARCHAR(255) NOT NULL,
    description TEXT,
    is_ai       BOOLEAN NOT NULL DEFAULT FALSE,
    is_active   BOOLEAN NOT NULL DEFAULT TRUE,
    created_at  TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    updated_at  TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_projects_creator_id ON
digital_projects(creator_id);
CREATE INDEX idx_projects_is_ai      ON digital_projects(is_ai);
CREATE INDEX idx_projects_active     ON
digital_projects(is_active);
CREATE INDEX idx_projects_created_at ON
digital_projects(created_at DESC);

-- Изображения проектов
CREATE TABLE project_images (
    id          BIGSERIAL PRIMARY KEY,
    project_id  BIGINT NOT NULL REFERENCES digital_projects(id) ON
DELETE CASCADE,
    image_url   VARCHAR(512) NOT NULL,
    image_order SMALLINT NOT NULL CHECK (image_order BETWEEN 0 AND
19),
    UNIQUE (project_id, image_order)
);

CREATE INDEX idx_project_images_project_id ON
project_images(project_id);

-- Лайки
CREATE TABLE likes (
    id          BIGSERIAL PRIMARY KEY,

```

```
    project_id BIGINT NOT NULL REFERENCES digital_projects(id) ON
DELETE CASCADE,
    user_id     BIGINT NOT NULL,
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    UNIQUE (project_id, user_id)
);

CREATE INDEX idx_likes_project_id ON likes(project_id);
CREATE INDEX idx_likes_user_id    ON likes(user_id);
```

Приложение Д
(обязательное)
Листинг микросервиса «Медиафайлы»

Листинг Д.1 – Файл S3Config.java

```
package com.lensomni.media_service.config;

import ...

@Configuration
public class S3Config {

    @Value("${minio.endpoint}")
    private String endpoint;

    @Value("${minio.region}")
    private String region;

    @Bean
    public S3Client s3Client(
        @Value("${minio.access-key}") String accessKey,
        @Value("${minio.secret-key}") String secretKey) {
        return S3Client.builder()
            .region(Region.of(region))

        .credentialsProvider(StaticCredentialsProvider.create(
            AwsBasicCredentials.create(accessKey,
secretKey)))
            .endpointOverride(URI.create(endpoint))
            .forcePathStyle(true)
            .build();
    }
}
```

Листинг Д.2 – Файл MediaController.java

```
package com.lensomni.media_service.controller;

import ...

@RestController
@RequestMapping("/media")
@RequiredArgsConstructor
public class MediaController {

    private final StorageService storageService;
    private final AiDetectionService aiDetectionService;
    private final ProjectValidationService validationService;

    @PostMapping("/projects/upload")
    public ResponseEntity<MediaUploadResponse>
uploadProjectImages(
    @RequestParam("images") List<MultipartFile> images,
    @RequestParam String title,
    @RequestParam(required = false) String description)
throws IOException {
```

```

validationService.validateFiles(images);
validationService.validateProjectText(title, description);

List<String> urls = new ArrayList<>();
boolean isAi = false;
double maxConfidence = 0.0;

for (MultipartFile image : images) {
    String url = storageService.uploadProjectImage(image);
    urls.add(url);

    AiDetectionResult result =
aiDetectionService.detect(image);
    if (result.isAi()) isAi = true;
    if (result.getConfidence() > maxConfidence)
maxConfidence = result.getConfidence();
}

return ResponseEntity.ok(MediaUploadResponse.builder()
    .imageUrls(urls)
    .isAi(isAi)
    .confidence(maxConfidence)
    .build());
}
}

```

Листинг Д.3 – Файл AiDetectionResult.java

```

package com.lensomni.media_service.dto;

import ...

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class AiDetectionResult {

    @JsonProperty("is_ai")
    private boolean isAi;

    private double confidence;
}

```

Листинг Д.4 – Файл MediaUploadResponse.java

```

package com.lensomni.media_service.dto;
import ...

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class MediaUploadResponse {

```

```

        private List<String> imageUrls;
        private boolean isAi;
        private double confidence;
    }

```

Листинг Д.5 – Файл GlobalExceptionHandler.java

```

package com.lensomni.media_service.exception;

import ...

@RestControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(ValidationException.class)
    public ResponseEntity<Map<String, Object>>
    handleValidation(ValidationException ex) {
        return ResponseEntity.badRequest().body(Map.of("errors",
ex.getErrors()));
    }

    @ExceptionHandler(IllegalArgumentException.class)
    public ResponseEntity<Map<String, Object>>
    handleIllegal(IllegalArgumentException ex) {
        return ResponseEntity.badRequest().body(Map.of("error",
ex.getMessage()));
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<Map<String, Object>>
    handleGeneral(Exception ex) {
        return ResponseEntity.internalServerError()
            .body(Map.of("error", "Внутренняя ошибка
сервера"));
    }
}

```

Листинг Д.6 – Файл ValidationException.java

```

package com.lensomni.media_service.exception;

import ...

@Getter
public class ValidationException extends RuntimeException {
    private final List<String> errors;

    public ValidationException(List<String> errors) {
        super(String.join("; ", errors));
        this.errors = errors;
    }
}

```

Листинг Д.7 – Файл AiDetectionService.java

```

package com.lensomni.media_service.service;

import ...

@Service
@Slf4j
public class AiDetectionService {

    @Value("${ai-detection.enabled:false}")
    private boolean enabled;

    @Value("${ai-detection.url:http://localhost:8000}")
    private String detectionServiceUrl;

    private final RestTemplate restTemplate = new RestTemplate();

    public AiDetectionResult detect(MultipartFile file) {
        if (!enabled) {
            log.info("AI detection stub. File: {}",
file.getOriginalFilename());
            return AiDetectionResult.builder()
                .isAi(false)
                .confidence(0.0)
                .build();
        }

        try {
            byte[] bytes = file.getBytes();

            ResponseEntity<AiDetectionResult> response =
restTemplate.postForEntity(
                detectionServiceUrl + "/detect",
                requestEntity,
                AiDetectionResult.class
            );

            AiDetectionResult result = response.getBody();
            if (result == null) {
                log.warn("AI detection вернул пустой ответ для
файла: {}",
                    file.getOriginalFilename());
                return fallback();
            }

            log.info("AI detection: файл={}, isAi={},
confidence={} ",
                file.getOriginalFilename(), result.isAi(),
result.getConfidence());

            return result;
        } catch (IOException e) {

```

```

        log.error("Ошибка чтения файла {}: {}",
file.getOriginalFilename(), e.getMessage());
        return fallback();
    } catch (Exception e) {
        log.error("Ошибка вызова ai-detection-service для
файла {}: {}",
            file.getOriginalFilename(), e.getMessage());
        return fallback();
    }
}

private AiDetectionResult fallback() {
    return AiDetectionResult.builder()
        .isAi(false)
        .confidence(0.0)
        .build();
}
}

```

Листинг Д.8 – Файл ProjectValidationService.java

```

package com.lensomni.media_service.service;

import ...

@Service
public class ProjectValidationService {

    private static final long MAX_SIZE          = 30L * 1024 *
1024;
    private static final long MIN_PNG_SIZE      = 67;
    private static final long MIN_JPEG_SIZE     = 159;
    private static final int  MAX_IMAGES        = 20;
    private static final int  MAX_TITLE_LENGTH  = 255;
    private static final int  MAX_DESC_LENGTH   = 2000;

    public void validateFiles(List<MultipartFile> files) throws
IOException {
        List<String> errors = new ArrayList<>();

        if (files == null || files.isEmpty()) {
            errors.add("В проект нужно добавить хотя бы одно
изображение");
        } else if (files.size() > MAX_IMAGES) {
            errors.add("В проекте может быть максимум " +
MAX_IMAGES + " изображений");
        } else {
            for (MultipartFile file : files) {
                validateFile(file, errors);
            }
        }

        if (!errors.isEmpty()) throw new
ValidationException(errors);
    }
}

```

```

    public void validateProjectText(String title, String
description) {
        List<String> errors = new ArrayList<>();

        if (title == null || title.isBlank()) {
            errors.add("Название проекта не может быть пустым");
        } else if (title.length() > MAX_TITLE_LENGTH) {
            errors.add("Название не должно превышать " +
MAX_TITLE_LENGTH + " символов");
        }

        if (description != null && description.length() >
MAX_DESC_LENGTH) {
            errors.add("Описание не должно превышать " +
MAX_DESC_LENGTH + " символов");
        }

        if (!errors.isEmpty()) throw new
ValidationException(errors);
    }

    private void validateFile(MultipartFile file, List<String>
errors) throws IOException {
        String filename = file.getOriginalFilename();
        long size = file.getSize();

        if (filename == null || !filename.contains(".")) {
            errors.add("Файл \"" + filename + "\": недопустимое
расширение");
            return;
        }

        String ext =
filename.substring(filename.lastIndexOf(".")).toLowerCase();

        if (!ext.equals(".png") && !ext.equals(".jpeg") &&
!ext.equals(".jpg")) {
            errors.add("Файл \"" + filename + "\": разрешены
только .png, .jpg и .jpeg");
            return;
        }

        if (size > MAX_SIZE) {
            errors.add("Файл \"" + filename + "\": превышает 50
МБ");
        } else if (ext.equals(".png") && size < MIN_PNG_SIZE) {
            errors.add("Файл \"" + filename + "\": слишком
маленький размер");
        } else if ((ext.equals(".jpeg") || ext.equals(".jpg")) &&
size < MIN_JPEG_SIZE) {
            errors.add("Файл \"" + filename + "\": слишком
маленький размер");
        }
    }

```

```

    }

    byte[] bytes = file.getBytes();
    if (!isValidImage(bytes, ext)) {
        errors.add("Файл \"" + filename + "\": повреждён или
не является изображением");
    }
}

private boolean isValidImage(byte[] bytes, String ext) {
    if (bytes.length < 4) return false;
    return switch (ext) {
        case ".png" -> bytes[0] == (byte) 0x89 && bytes[1] ==
(byte) 0x50
                        && bytes[2] == (byte) 0x4E && bytes[3] ==
(byte) 0x47;
        case ".jpeg", ".jpg" -> bytes[0] == (byte) 0xFF &&
bytes[1] == (byte) 0xD8
                        && bytes[2] == (byte) 0xFF;
        default -> false;
    };
}
}

```

Листинг Д.9 – Файл StorageService.java

```

package com.lensomni.media_service.service;

import ...

@Service
@RequiredArgsConstructor
public class StorageService {

    private final S3Client s3Client;

    @Value("${minio.bucket.projects}")
    private String projectsBucket;

    @Value("${minio.bucket.avatars}")
    private String avatarsBucket;

    @Value("${minio.endpoint}")
    private String endpoint;

    public void deleteAvatar(String fileUrl) {
        deleteFile(fileUrl, avatarsBucket);
    }

    private String upload(MultipartFile file, String bucket)
throws IOException {
        String fileName = UUID.randomUUID() + "_" +
file.getOriginalFilename();
        s3Client.putObject(

```

```

        PutObjectRequest.builder()
            .bucket(bucket)
            .key(fileName)
            .contentType(file.getContentType())
            .build(),
        RequestBody.fromBytes(file.getBytes())
    );
    String url = endpoint + "/" + bucket + "/" + fileName;
    return url.replace("minio:9000", "localhost:9000");
}

private void deleteFile(String fileUrl, String bucket) {
    String key = fileUrl.substring(fileUrl.lastIndexOf("/") +
1);
    s3Client.deleteObject(DeleteObjectRequest.builder()
        .bucket(bucket)
        .key(key)
        .build());
}
}

```

Приложение Е
(рекомендуемое)

Программные компоненты серверной части приложения

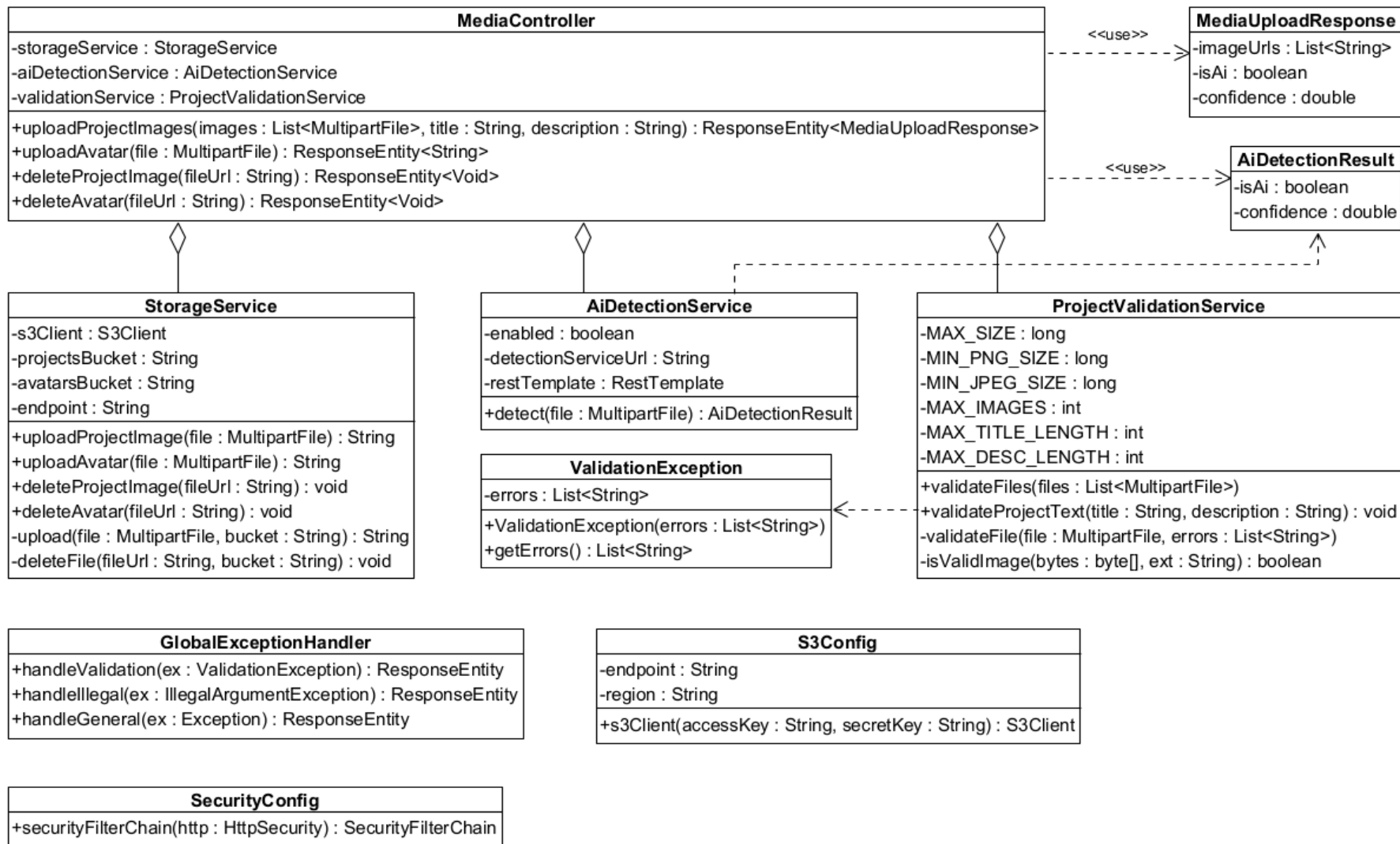


Рисунок Е.2 – Диаграмма классов микросервиса «Медиафайлы»

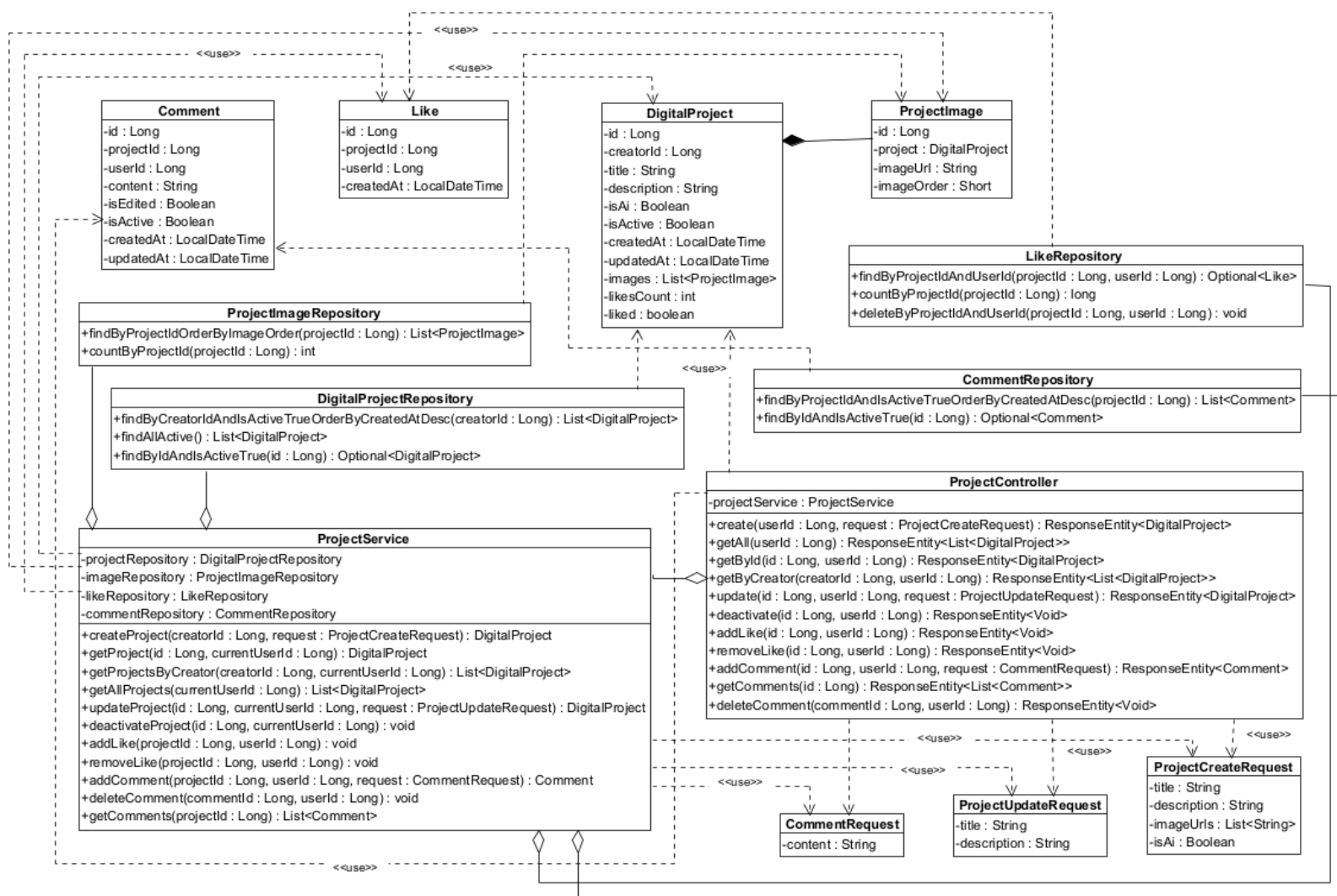
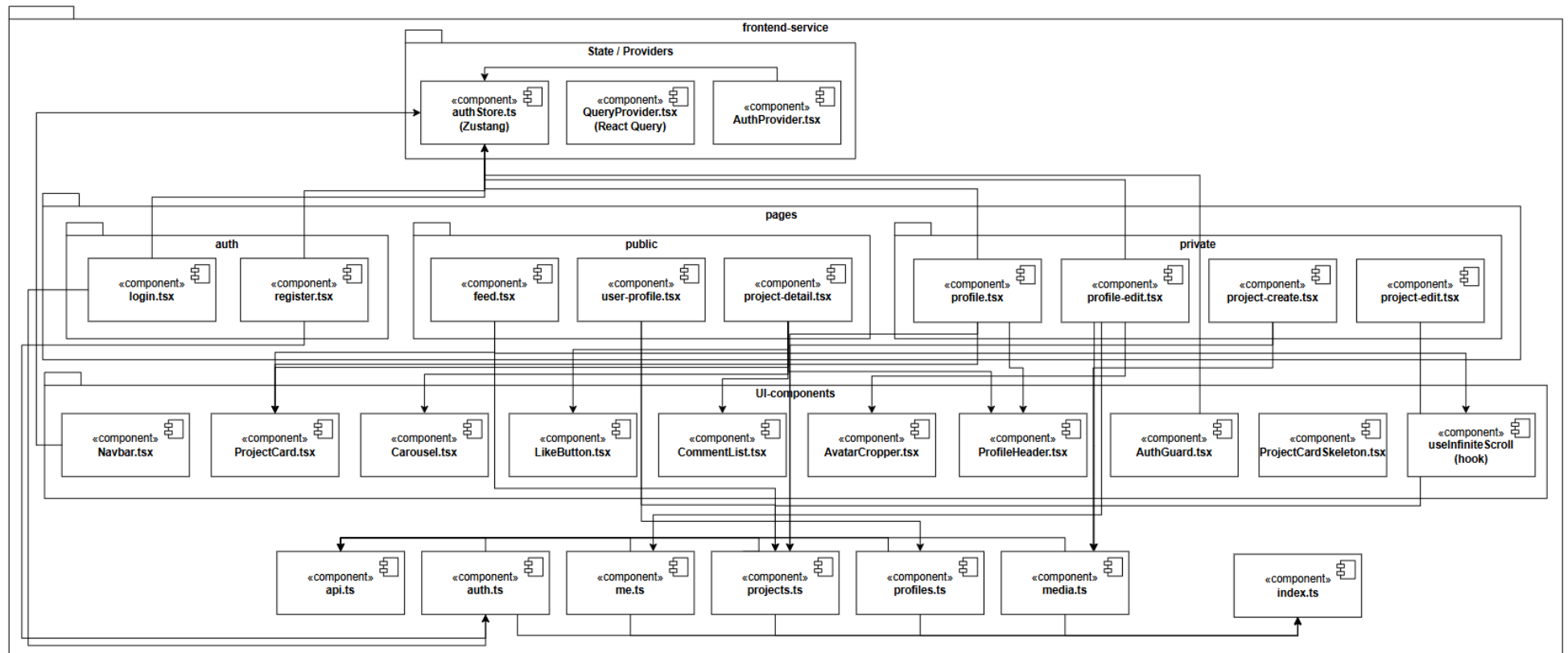


Рисунок Е.4 – Диаграмма классов микросервиса «Цифровые проекты»

Приложение Ж
(рекомендуемое)

Программные компоненты клиентской части приложения



Приложение И
(рекомендуемое)
Результаты тестирования

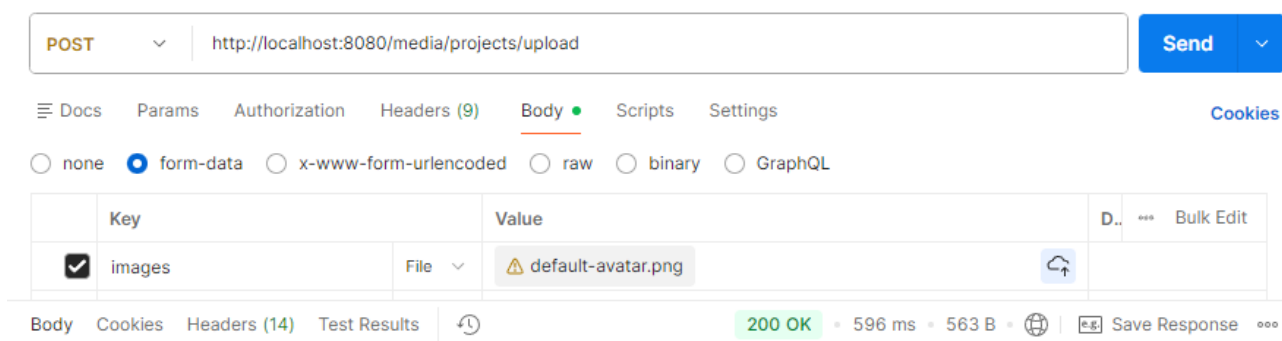


Рисунок И.1 – Результат загрузки default-avatar.png (255 КБ)

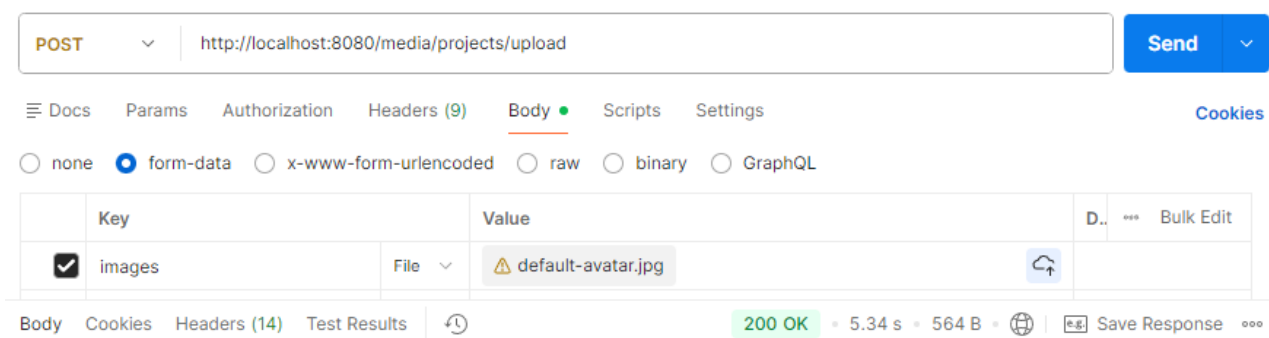


Рисунок И.2 – Результат загрузки default-avatar.jpg (255 КБ)

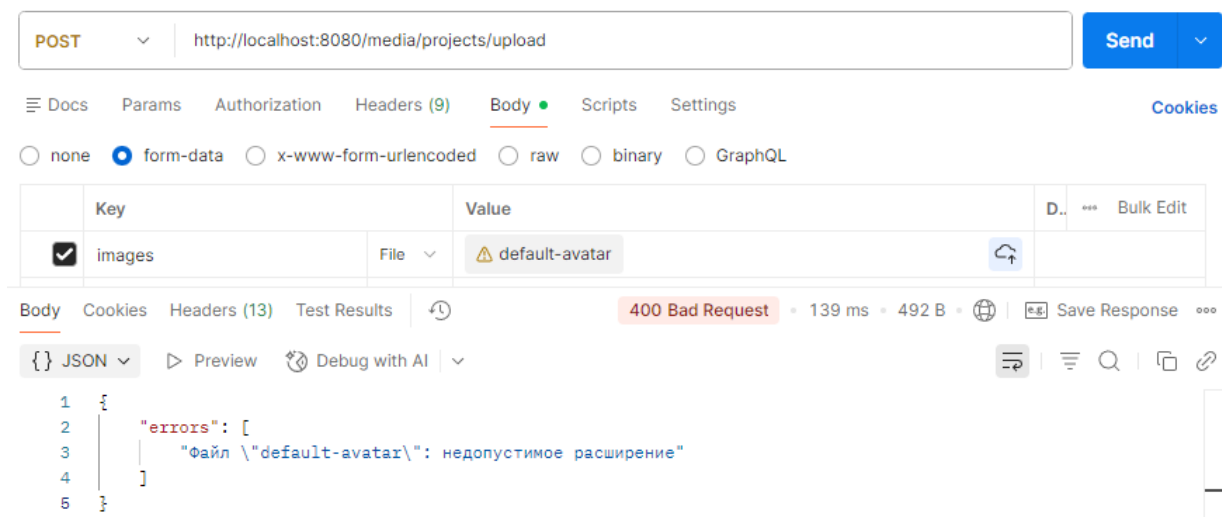


Рисунок И.3 – Результат загрузки default-avatar

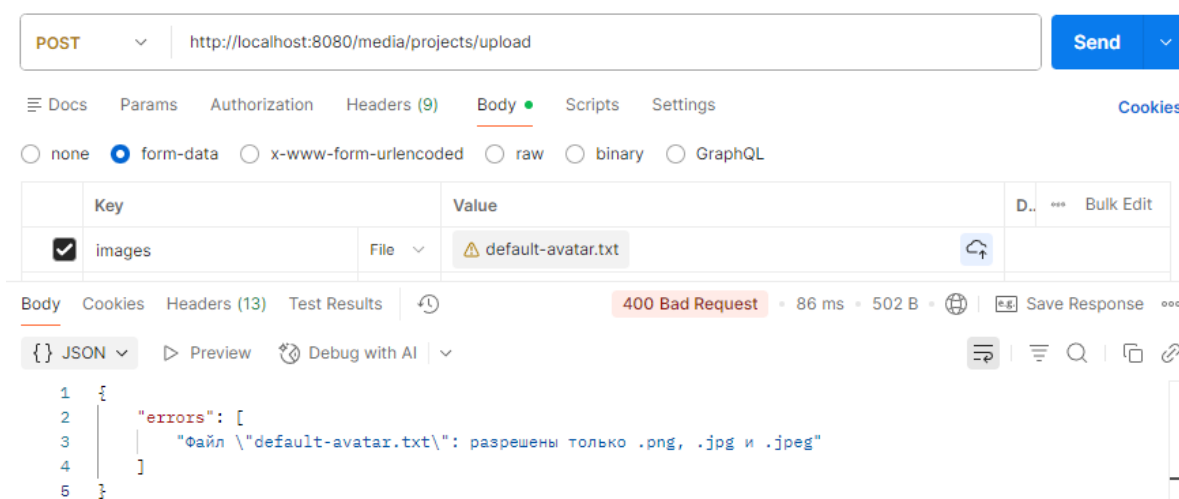


Рисунок И.4 – Результат загрузки default-avatar.txt

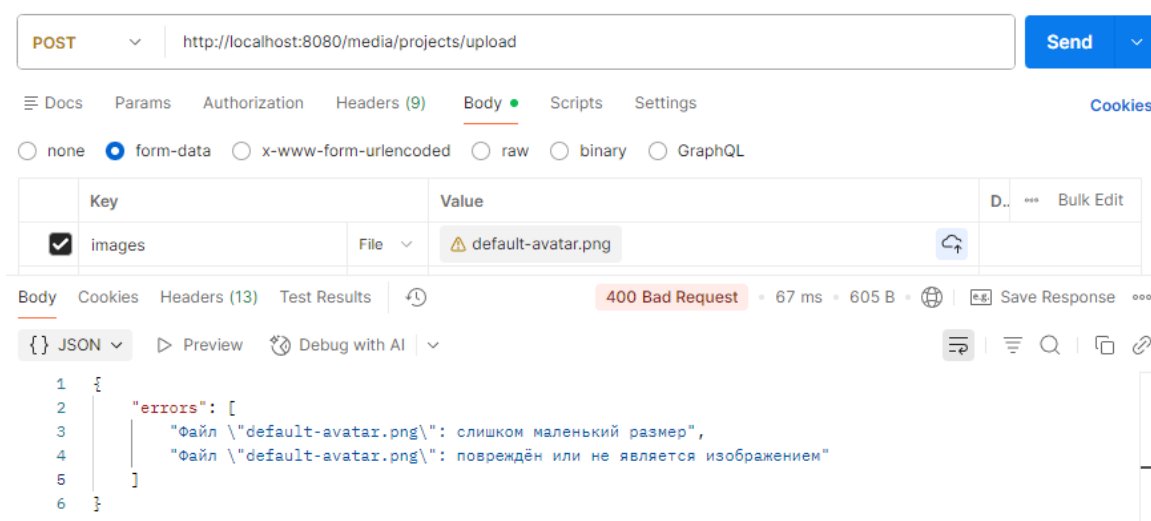


Рисунок И.5 – Результат загрузки default-avatar.png (30 байт)

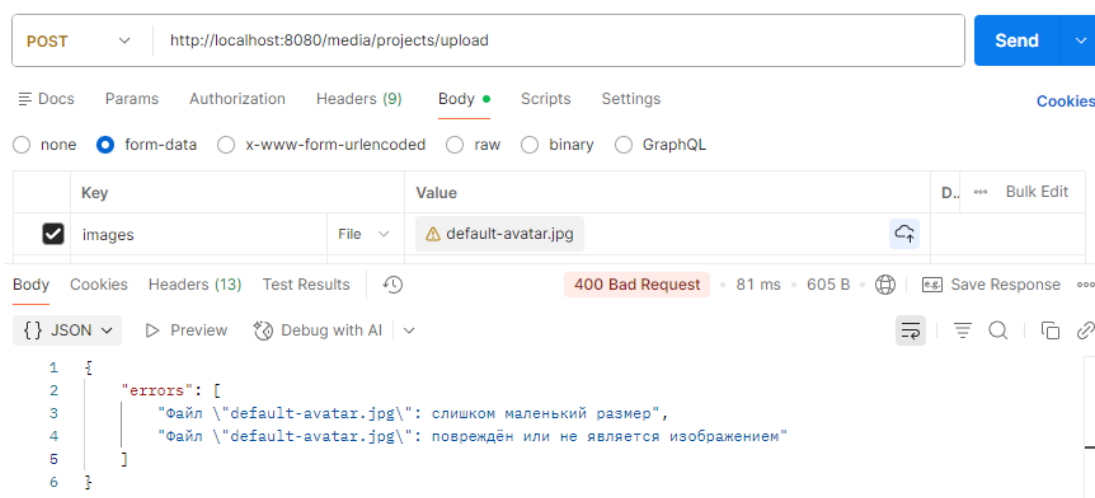


Рисунок И.6 – Результат загрузки default-avatar.jpg (30 байт)

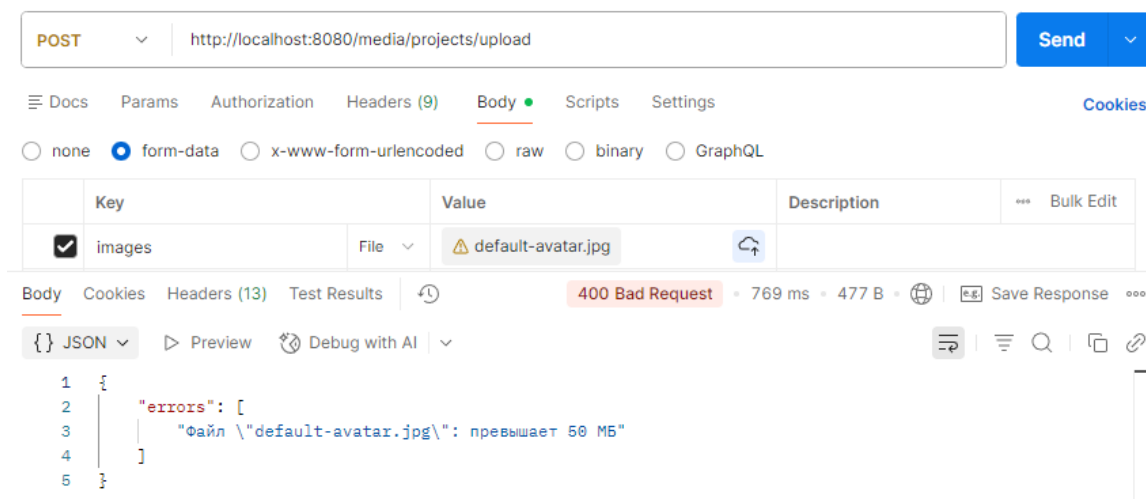


Рисунок И.7 – Результат загрузки `default-avatar.png` (52 МБ)

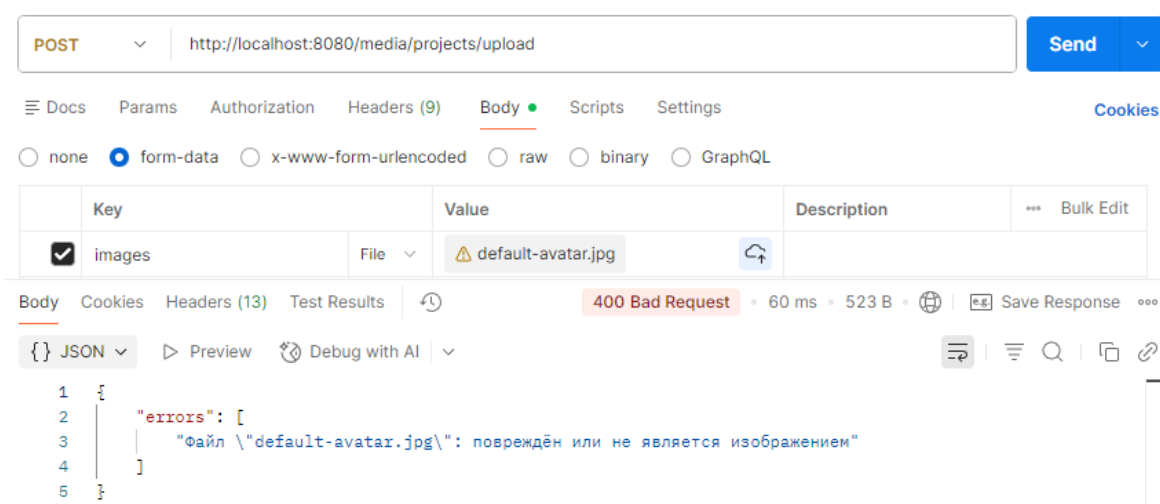


Рисунок И.8 – Результат загрузки повреждённого `default-avatar.jpg`

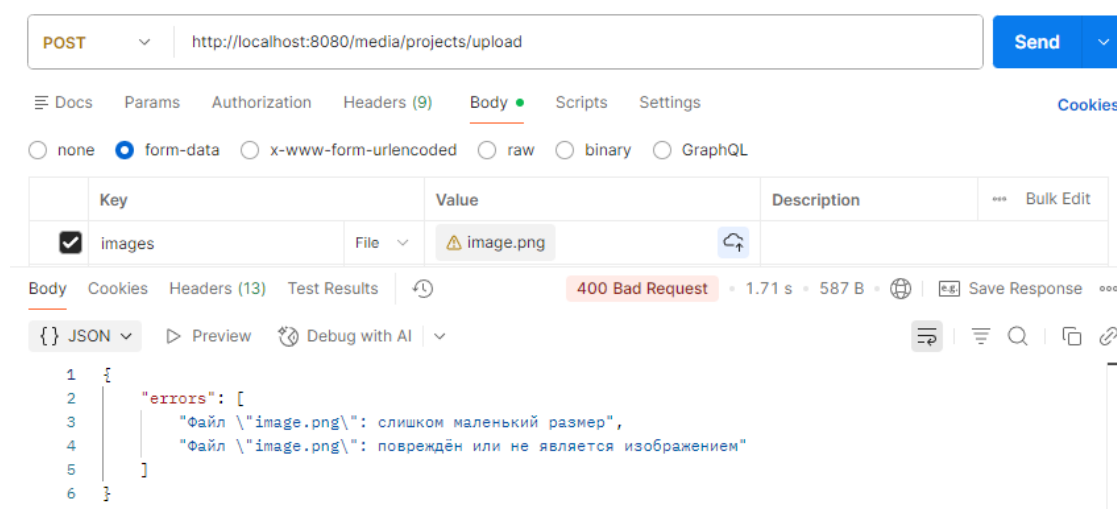


Рисунок И.9 – Результат загрузки файла размером 0 бит

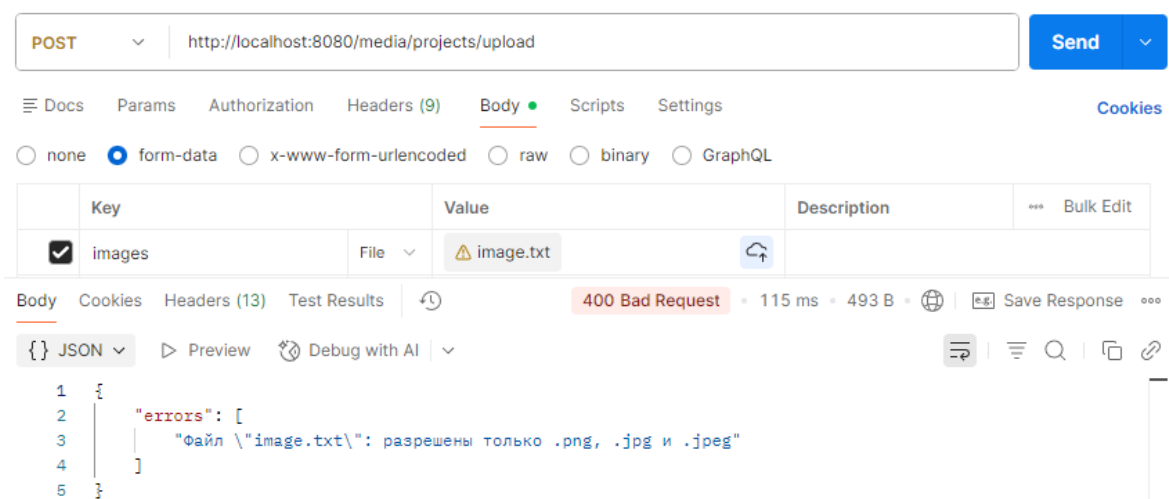


Рисунок И.10 – Результат загрузки image.txt

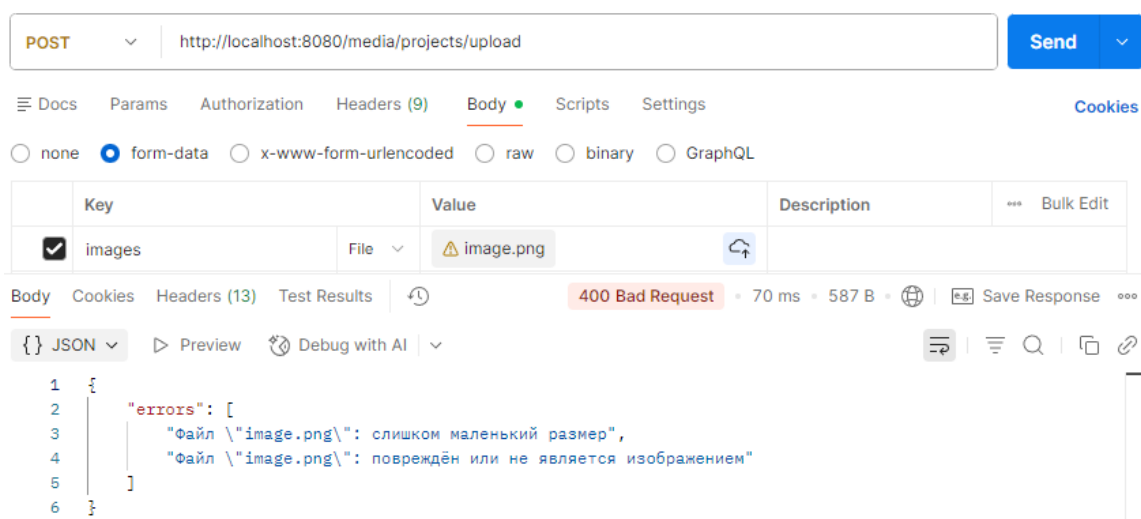


Рисунок И.11 – Результат загрузки image.png (66 байт)

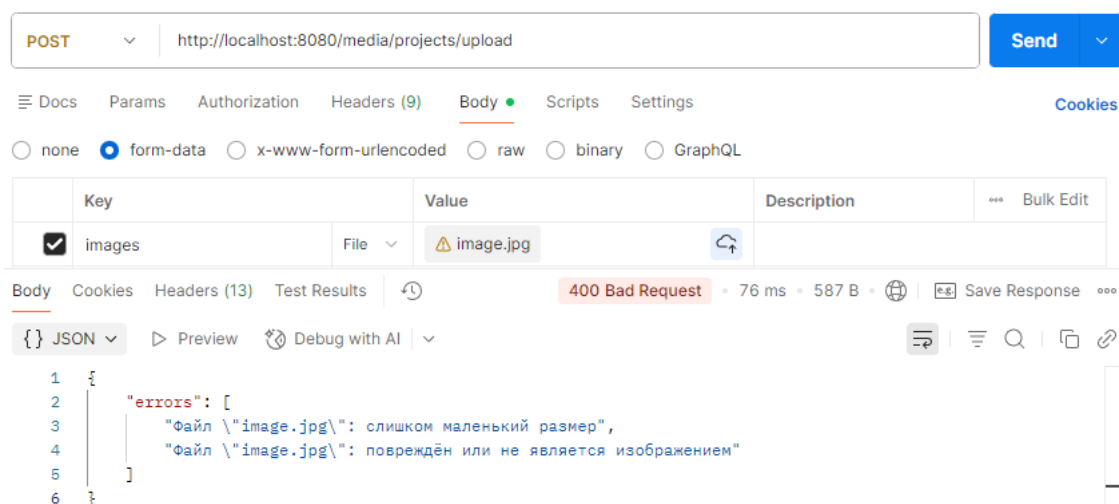


Рисунок И.12 – Результат загрузки image.jpg (158 байт)

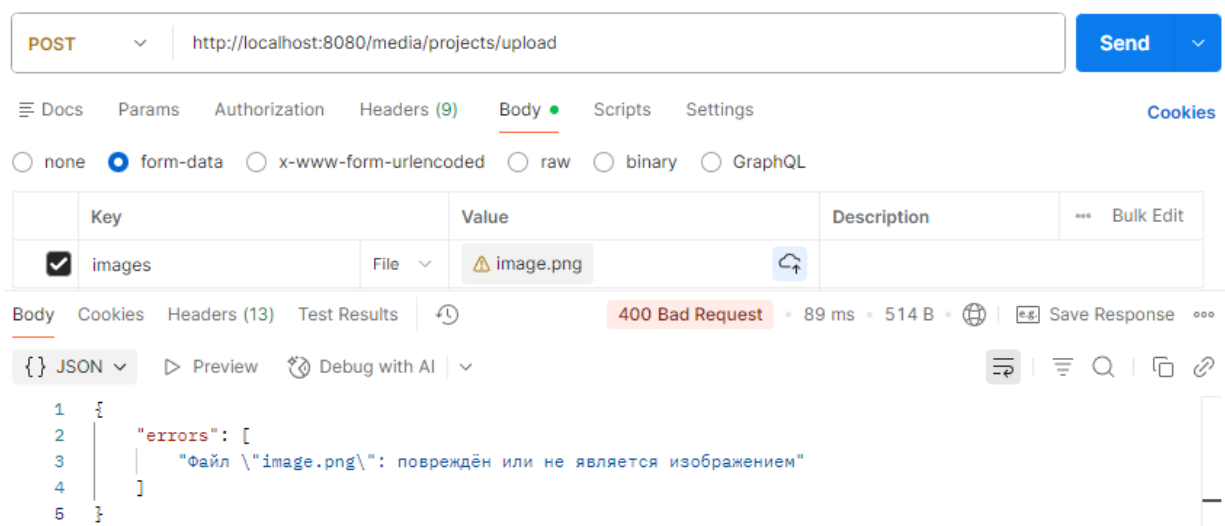


Рисунок И.13 – Результат загрузки повреждённого image.png

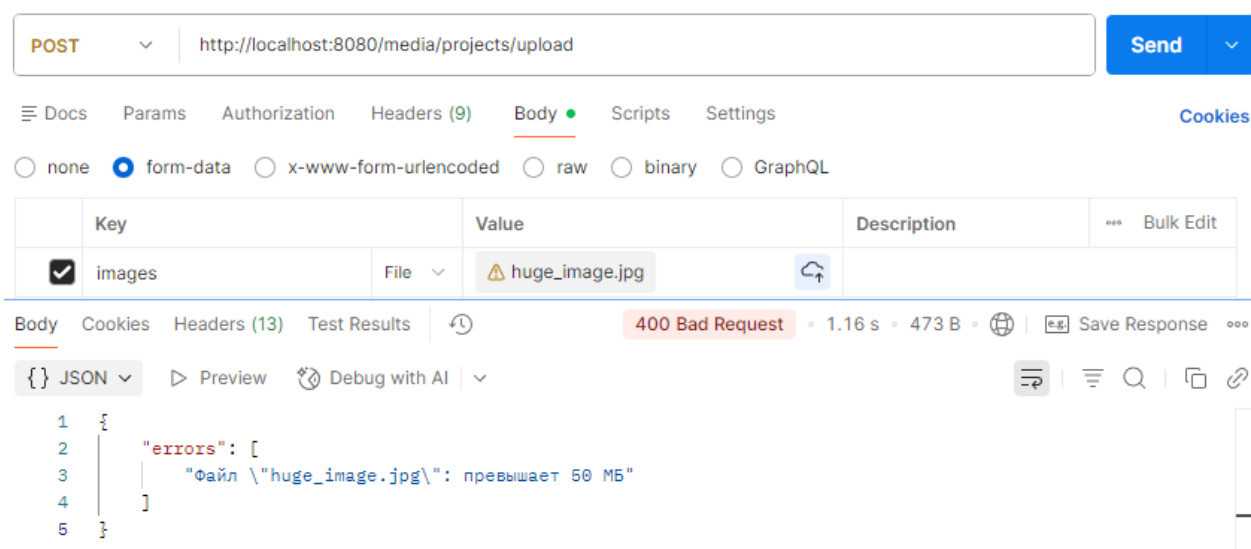


Рисунок И.14 – Результат загрузки huge_image.jpg (60 МБ)

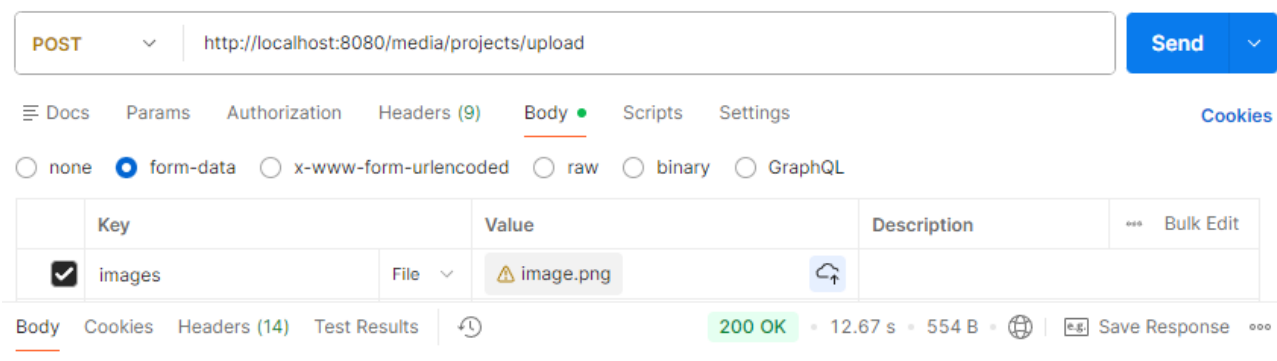


Рисунок И.15 – Результат загрузки image.png (67 байт)

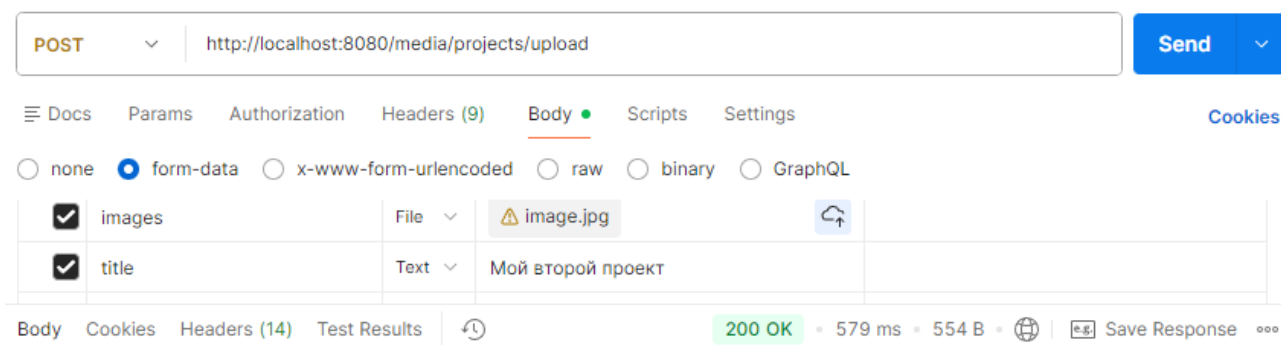


Рисунок И.16 – Результат загрузки image.jpg (159 байт)

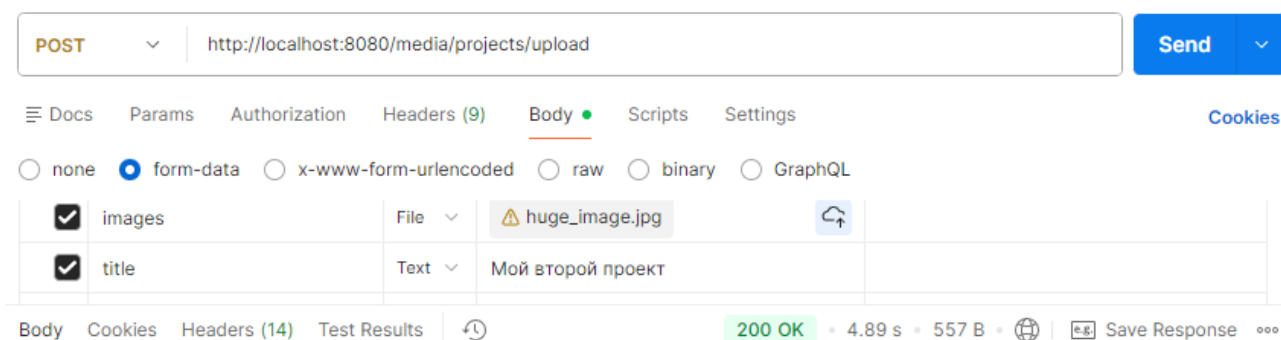


Рисунок И.17 – Результат загрузки image.jpg (50 МБ)

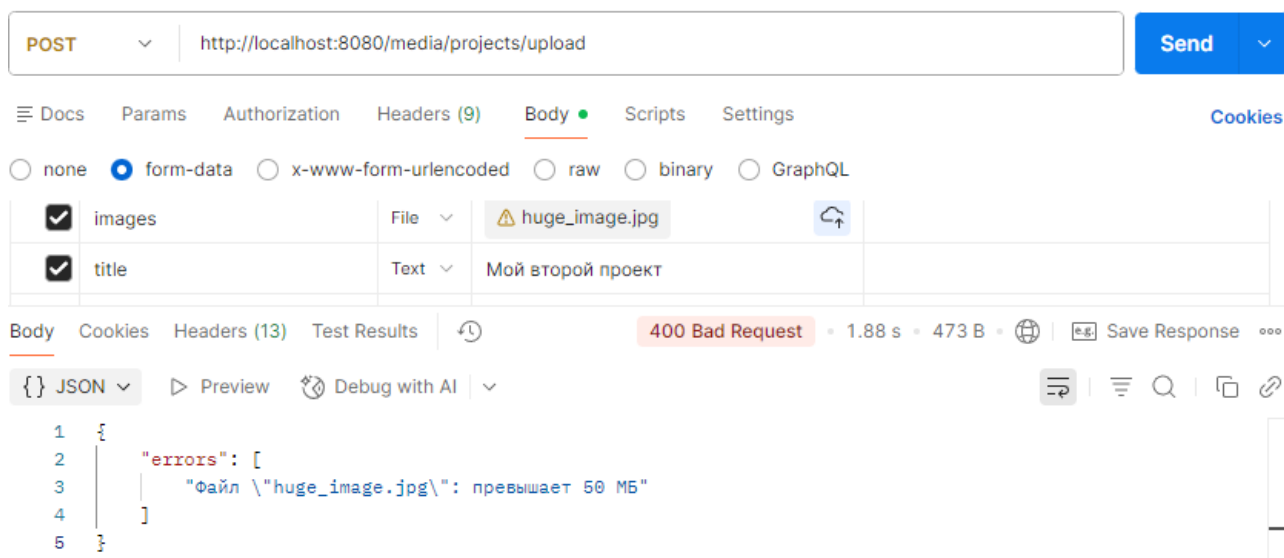


Рисунок И.18 – Результат загрузки image.jpg (50 МБ + 1 байт)

Приложение К
(рекомендуемое)
Листинг юнит-тестов

```

@Nested
@DisplayName("Группа 1: количество файлов")
class FileCountTests {
    @Test
    @DisplayName("files = null - C1: добавить хотя бы одно
изображение")
    void shouldReturnC1WhenFilesIsNull() throws IOException {
        List<String> errors = service.validate(null, VALID_TITLE,
VALID_DESC);

        assertThat(errors).containsExactly(
            "В проект нужно добавить хотя бы одно изображение"
        );
    }
    @Test
    @DisplayName("11 файлов - C2: максимум 10 изображений")
    void shouldReturnC2WhenMoreThan10Files() throws IOException {
        MultipartFile simpleMock =
org.mockito.Mockito.mock(MultipartFile.class);
        List<MultipartFile> files = Collections.nCopies(11,
simpleMock);

        List<String> errors = service.validate(files, VALID_TITLE,
VALID_DESC);

        assertThat(errors).containsExactly(
            "В проекте может быть максимум 10 изображений"
        );
    }
    @Test
    @DisplayName("1 корректный файл - успех")
    void shouldPassWith1ValidFile() throws IOException {
        List<MultipartFile> files = List.of(mockValidJpeg());

        List<String> errors = service.validate(files, VALID_TITLE,
VALID_DESC);

        assertThat(errors).isEmpty();
    }
    @Test
    @DisplayName("10 корректных файлов - успех")
    void shouldPassWith10ValidFiles() throws IOException {
        List<MultipartFile> files = Collections.nCopies(10,
mockValidJpeg());

        List<String> errors = service.validate(files, VALID_TITLE,
VALID_DESC);

        assertThat(errors).isEmpty();
    }
}

```