1МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Разработка программного обеспечения
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка веб-приложения с использованием технологии React.js

Обучающийся А. А. Землянская _____
(Инициалы Фамилия) (личная подпись)

Руководитель Н.Н. Рогова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант канд. филол. наук, доцент М.В. Дайнеко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Тема выпускной квалификационной работы «Разработка веб-приложения с использованием технологии React.js» Работа посвящена разработке веб-приложения для публикации и продажи цифрового искусства с использованием технологии React.js.

Целью данной работы является разработка веб-приложения с использованием технологии React.js, обеспечивающего высокую производительность, удобство взаимодействия с пользователем.

Работа состоит из введения, трёх разделов, заключения и списка литературы, включает в себя 57 страниц, 26 изображений, 11 таблиц.

Введение описывает цели и задачи работы, определяет объект, предмет и методы исследования, а также практическую ценность.

В первом разделе описывается актуальность темы, проводится анализ существующих программных решений и разрабатываются требования к будущему продукту.

Во втором разделе проводится проектирование разрабатываемого веб-приложения, выбор технологий и реализация функциональных требований. В рамках проектирования ПО создается ряд диаграмм в нотации UML: диаграммы классов, компонентов и деятельности, а также логическая модель данных, описываются реализованные алгоритмы и демонстрируется графический интерфейс приложения.

Третий раздел посвящён тестированию. Были показаны тестовые сценарии для применяемых видов тестирования и составлен отчет об обнаруженных ошибках.

В заключении подводится итог работы, перечисляются выполненные задачи и описывается ценность работы.

Abstract

The title of the graduation work is "Development of a web application using React.js technology" The work is devoted to the development of a web application for publishing and selling digital art using React.js technology.

The graduation work consists of an introduction, three chapters, 26 figures, 11 tables, a conclusion, and a list of 20 references including foreign sources.

The aim of this work is to develop a web application using React.js technology, providing high performance, ease of user interaction.

The object of the graduation work is the process of developing modern web applications using React.js technology stack.

The subject of the graduation work is methods for implementing digital art marketplace functionality (user authentication, artwork upload, payment integration).

The key issue of the graduation work is creating a user-friendly alternative to existing digital art platforms with excessive functionality and high commissions.

The graduation work may be divided into several logically connected parts which are:

- market analysis and requirements specification;
- application design and implementation;
- testing and validation.

The first part describes in details the analysis of existing platforms (ArtStation, DeviantArt), identifying their limitations (complex UI, high fees), and formulating technical requirements for the new solution. We examine React.js components architecture, Node.js backend solutions, and MongoDB database structure for optimal performance.

The second part outlines the results of UML diagram development (class, component, activity diagrams), implementation of key features (image feed, artwork upload system, shopping cart), and UI/UX design solutions.

The third part consists of comprehensive testing, including unit testing of React components, functional testing with positive and negative scenarios, testing of the database server's fault tolerance to attacks, and acceptance testing with the participation of the customer's team of experts.

In conclusion, it is emphasized that all key functions have been implemented: authentication, work upload system, main page with image feed, image search and sorting, image shopping cart. Final indicators: Stability - 94% successful test cases, Performance - 87% faster than ArtStation, Usability - visual defects need to be corrected.

Nevertheless, more experimental data are required to evaluate long-term performance under high user loads.

The work is of interest for web developers specializing in React.js and startup

Оглавление

Введение

Современные цифровые арт-платформы играют важную роль в популяризации искусства, предоставляя художникам возможность демонстрировать и продавать свои работы, а ценителям искусства — удобный способ находить и приобретать цифровые произведения. С развитием веб-технологий создание интерактивных и удобных платформ стало проще и доступнее. Популярность подобных сервисов подтверждает востребованность решений, объединяющих художников и покупателей в единой цифровой среде.

Существует множество платформ для размещения цифрового искусства (ArtStation, DeviantArt). Однако многие из них имеют сложный интерфейс, высокие комиссии, ориентированы только на профессиональных художников или вовсе неудобны для русскоязычного пользователя и рынка. Данная работа предлагает альтернативное решение — простую и удобную галерею с базовым функционалом, реализованную на современных веб-технологиях.

Цель работы - разработка веб-приложения с использованием технологии React.js, обеспечивающего высокую производительность, удобство взаимодействия с пользователем.

Задачи:

- выявить критерии производительности и юзабилити современных арт-платформ на основе сравнительного анализа их архитектурных решений и пользовательских метрик;
- определить требования к разрабатываемой системе, исходя из потребностей пользователей;
- разработать и апробировать оптимизированные алгоритмы загрузки и отображения цифровых произведений;
- оценить эффективность реализованного решения через юзабилити-тестирование с целевой аудиторией.

Объект исследования - процесс создания веб-приложений на базе React.js, включая архитектуру, компоненты и управление состоянием приложения.

Предмет исследования - методы и средства разработки веб-приложений с использованием технологии React.js, направленные на повышение производительности и удобства взаимодействия с пользователем.

Практическая значимость заключается в создании упрощенной, но функциональной отечественной платформы для художников и покупателей, в отличие от коммерческих решений с избыточным функционалом.

Работа состоит из введения, трех разделов, заключения и списка литературы:

- раздел 1 посвящен анализу существующих арт-платформ и выбору технологий;
- раздел 2 описывает проектирование и разработку веб-приложения;
- раздел 3 содержит тестирование и оценку результатов.

1 Постановка задачи на разработку программного обеспечения для предприятия

1.1 Актуальность разработки ПО для выбранной области

В современном мире цифровое искусство становится все более популярным, а художники все чаще используют интернет для продвижения своих работ, поиска клиентов и продажи произведений. Однако существующие платформы для публикации и продажи художественных работ имеют ряд ограничений:

- недостаток специализированных инструментов для демонстрации, продажи и продвижения цифрового искусства;
- ограниченная монетизация: художники сталкиваются с трудностями при попытке продать свои работы из-за отсутствия встроенных платежных систем или высоких комиссий комиссии (например, NFT-маркетплейсы взимают до 15% с продаж);
- отсутствие удобных механизмов для общения с потенциальными покупателями;
- сложный интерфейс существующих решений, затрудняющий использование новичками;
- ограниченные возможности для бесплатного размещения (многие сервисы ориентированы только на продажи).

Данные проблемы значительно влияют на заинтересованность потенциальной аудитории в продукте. Возможные их последствия отражены на рисунке 1.

Рисунок 1 – Диаграмма «Причина – следствие» недостатков существующих решений

Ежегодно художники в интернет-пространстве задаются вопросом, где им продавать своё творчество, и количество таких запросов не уменьшается, что подтверждает статистика интернет запросов (рисунок 2-3)

Рисунок 2 – Запросы за январь 2025

Рисунок 3 – Сравнение количества запросов за год

Эти проблемы делают разработку специализированного веб-приложения для художников актуальной и востребованной задачей.

Элементы улучшения в разработанном приложении:

– упрощенный интерфейс на React.js, адаптированный под начинающих художников и пользователей, которых отпугивает избыток функций;
– гибкая система публикации (бесплатно / платно);
– уменьшение комиссии за счет увеличения процента заинтересованной аудитории;
– упрощение процесса покупки изображения за счёт сокращения взаимодействия художника и покупателя.

Целевой аудиторией приложения являются:

– цифровые художники: профессионалы и любители, создающие цифровые иллюстрации, 3D-модели, анимацию и другие виды цифрового искусства.

– традиционные художники, работающие в традиционных техниках, но желающие оцифровать и продвигать свои работы.
– коллекционеры и покупатели: люди, заинтересованные в приобретении уникальных произведений искусства.

Наиболее распространенным решением для продвижения и продажи художественных работ является создание страниц или пабликов в социальных сетях. Однако данный метод имеет ряд существенных недостатков. Во-первых, для эффективной раскрутки требуется значительное количество времени и усилий, поскольку алгоритмы социальных сетей не обеспечивают достаточного охвата аудитории без ежедневной публикации контента. Во-вторых, данный подход предполагает необходимость прямого взаимодействия с покупателями, что может быть неудобным для художников в силу их занятости или отсутствия навыков коммуникации. Кроме того, социальные сети зачастую не предоставляют встроенных инструментов для автоматизированной оплаты, что усложняет процесс продажи и увеличивает риски недобросовестных сделок [1].

Вторым вариантом являются специализированные платформы, предназначенные для развития творческих проектов, включая художественные. Однако такие решения имеют существенные ограничения, связанные с их ориентацией на иностранную аудиторию. Основные проблемы заключаются в следующем: во-первых, интерфейс большинства платформ не локализован на русский язык, а в некоторых случаях даже на английский, что создает языковой барьер для русскоязычных пользователей. Во-вторых, возникают сложности с оплатой, поскольку далеко не все пользователи обладают банковскими картами, поддерживающими международные переводы, что делает процесс получения средств затруднительным.

Ввиду узкой направленности выбранной темы количество проектов с аналогичным функционалом среди выпускных работ других студентов ограничено. Однако анализ показал, что в большинстве случаев в качестве оптимального решения предлагается разработка веб-приложения. Данный

подход предпочтителен по сравнению с созданием мобильных или настольных приложений, а также интеграцией или реинжинирингом модулей существующих систем. Основными преимуществами веб-приложений являются:

- доступность: веб-приложения используются через браузер на любом устройстве с доступом в сеть, что позволяет пользователям легко получать доступ к приложению без необходимости установки его на устройства;
- централизованные обновления: все данные и функционал находятся на сервере, разработчики могут быстро вносить изменения и обновления, которые немедленно становятся доступны всем пользователям без необходимости их загрузки;
- низкие требования к ресурсам: веб приложения работают на менее мощных устройствах;
- адаптивный дизайн: интерфейс подстраивается под разные размеры экранов, от телефонов до планшетов и персональных компьютеров;
- интеграция с технологиями: специфика веб-приложения позволяет легко внедрять новые функции и интегрироваться с другими веб-сервисами.

Но главной причиной выбора решения по созданию веб-приложения с нуля является невозможность реинжиниринга существующих решений на данном этапе ввиду отсутствия любого доступа к ним, кроме пользовательского [12].

Далее представлен детальный анализ существующих платформ, предназначенных для публикации художественных работ. Для проведения сравнительного анализа, помимо базового функционала, были выбраны следующие критерии:

- удобство регистрации на платформе;
- удобство загрузки изображения;

– наличие удобного и понятного интерфейса;
– наличие локализации на русский язык;
– наличие возможности добавлять теги и описание к публикации;
– наличие сортировки публикаций по выбранным критериям;
– наличие встроенной платёжной системы;
– возможность оплаты Российской картой или через отечественные платежные системы;
– специфика заработка на платформе;
– возможность взаимодействовать с покупателем через переписку.

1.2 Обзор и анализ существующих программных решений

Для сравнительного анализа было выбрано несколько наиболее известных решений. Одним из таких сервисов является «DeviantArt».

После завершения регистрации платформа предоставляет возможность выбрать предпочитаемую стилистику публикаций, которые будут автоматически подбираться для ленты пользователя. Кроме того, доступна опция включения безопасного режима, при активации которого из ленты исключаются работы, содержащие контент для взрослых. По завершении настройки пользователь попадает на главную страницу, где отображается лента с работами других авторов.

Уже на этапе первичного взаимодействия с платформой можно выделить ряд преимуществ и недостатков:

Плюсы:

– удобная регистрация, предоставляющая в том числе и возможности для кастомизации страницы;
– наличие сортировки публикаций по средствам тегов, поиска и полоски сортировки по основным направлениям;

– при нажатии на понравившееся изображение происходит его увеличение, а снизу появляется поле для комментирования

Минусы:

– отсутствие русской локализации интерфейса. Эту проблему не решает даже авто-перевод страницы;
– обилие ярких цветов на тёмном фоне оформления и большое количество кликабельных элементов интерфейса мешают навигации по сайту

Плюсы формы публикации изображения:

– огромный инструментарий для добавления описания, тегов и уровней защиты работы;
– удобная загрузка изображения, сохранение черновика даже при отмене публикации;

Минусы:

– необходимость подключения платного плана для возможности выставления цен на работы, в который помимо фиксированной ежемесячной оплаты входит комиссия до 15% на заработок с каждой работы.

Одним из существенных недостатков платформы «DeviantArt» является отсутствие возможности проведения транзакций с использованием карт российских банков и популярных онлайн-кошельков. Это создает значительные трудности для художников из России, желающих монетизировать свои работы через данную платформу.

Следующим по популярности является японское веб-приложение «ArtStreet». Оно позволяет просматривать иллюстрации без регистрации, но запрашивает авторизацию при попытке добавить изображение. Форма регистрации не отличается уникальными особенностями и соответствует стандартным требованиям большинства аналогичных платформ.

Плюсы:

– удобная регистрация на платформе;
– интерфейс главной страницы интуитивно понятен и не перегружен, кнопка загрузки выделяется на фоне остальных;
– наличие сортировки по поиску и ключевым тегам;
– присутствует система рангов изображений на основе реакций пользователей, а также ежедневный и еженедельный рейтинг;
– возможность связаться с автором через комментарии или в личных сообщениях;
– комментарии и сообщения в чате, написанные не на английском языке, по нажатию кнопки переводятся на английский;
– наличие автоматической системы вотермарок, препятствующих краже изображения;

Минусы:

– отсутствие русской локализации: ресурс создан японской компанией Medibang и имеет локализацию только на английский язык, но это нивелируется авто-переводом страницы.
– отсутствие встроенной платёжной системы и прямого заработка на продаже изображений.

Продавать изображения можно только через личную переписку с покупателем и обмен платежными данными. Но, тем не менее, платформа финансово поддерживает художников через систему тематических конкурсов с денежными призами.

Плюсы формы загрузки:

– возможность загружать до 5 иллюстраций одновременно, в том числе и страниц комиксов;
– довольно обширный функционал для добавления описания и тегов.

Среди отечественных решений для публикации и продвижения художественных работ популярностью пользуется социальная сеть «ВКонтакте». Хотя данная платформа изначально не была ориентирована на

публикацию и продажу цифрового искусства, она предоставляет возможность ведения тематических пабликов, которые могут быть адаптированы под нужды художников.

Платформа «ВКонтакте» предлагает широкий набор инструментов для кастомизации внешнего вида паблика. Пользователи могут добавлять кликабельные кнопки-виджеты, которые позволяют структурировать страницу и упрощать навигацию для посетителей. Например, можно создать разделы для портфолио, информации о продажах, контактов и других важных элементов.

Плюсы:

- удобная и безопасная регистрация на платформе;
- структурированный и не перегруженный интерфейс;
- основной язык – русский;
- наличие встроенной платёжной системы;
- поддержка карт российских банков;
- возможность написать комментарий под работой и сообщение напрямую создателю;
- встроенная реклама, позволяющая получать пассивный доход

Минусы:

- отсутствие прямого функционала для продажи иллюстраций, только через переписку с покупателем;
- нет возможности сортировать изображения, их просмотр возможен только потоком в ленте или непосредственно в паблике создателя;
- необходимость регулярно публиковать посты чтобы оставаться в рекомендациях пользователей.

Плюсы формы загрузки изображения:

- удобная загрузка изображений перетаскиванием;
- возможность загрузки нескольких изображений и видео;
- возможность добавлять описание;

Минусы:

– необходимость проставлять теги вручную, отсутствие соответствующего автоматического функционала.

Для подведения итогов проведенного анализа результаты представлены в виде сравнительной таблицы 1.

Таблица 1 – Сравнение платформ

Критерии оценки	DeviantArt	Art Street	Вконтакте
Удобная форма загрузки изображения	+	+	+
Понятный интерфейс	-	+	+
Наличие русской локализации	-	-	+
Функционал добавления тегов	+	+	-
Наличие сортировки	+	+	-
Наличие встроенной платежной системы	+	-	+
Поддержка российских карт	-	-	+
Наличие прямой системы продажи иллюстраций	+	-	-
Возможность переписки между заказчиком и автором	+	+	+
Возможность непрямого заработка на платформе	-	+	+
Результат	6/10	6/10	7/10

Исходя из данных, представленных в таблице 1, можно сделать вывод, что все рассмотренные системы обладают как преимуществами, так и недостатками. Однако ни одна из них не соответствует в полной мере необходимым требованиям. Это свидетельствует о необходимости разработки нового веб-приложения, которое будет учитывать все указанные критерии и обеспечит оптимальное решение для поставленных задач.

1.3 Требования к функционалу и интерфейсу приложения

Для систематизации и структурирования требований к разрабатываемому веб-приложению для художников необходимо использовать модель FURPS+. Данная модель является общепризнанным

стандартом в области проектирования программного обеспечения и позволяет охватить все ключевые аспекты разработки, начиная от функциональности и заканчивая физическими требованиями [20].

Требования к будущему продукту по методологии FURPS+:

Functionality (функциональность)

Публикация работ:

- возможность загрузки изображений в форматах JPEG, PNG, PSD.
- добавление описания, тегов и категорий для каждой работы.

Монетизация:

- выставление цен на работы.

Поиск и фильтрация:

- поиск работ по тегам, категориям.
- фильтрация по цене, рейтингу, дате публикации.

Социальные функции:

- возможность оценки работ.

Usability (Удобство использования)

Требования к удобству интерфейса и взаимодействия с пользователем:

- Интуитивно понятный интерфейс:
- простота навигации по платформе
- минимальное количество шагов для загрузки работы или оформления покупки.

Reliability (Надежность)

Требования к стабильности и отказоустойчивости:

- минимальное количество ошибок и сбоев в работе приложения.
- автоматическое создание резервных копий данных.

Защита данных:

- шифрование данных при передаче и хранении.
- регулярное обновление системы безопасности.

Performance (Производительность)

Требования к скорости и эффективности работы:

- быстрая загрузка страниц: время загрузки страницы не должно превышать 2-3 секунды.

Supportability (Поддерживаемость)

Требования к легкости поддержки и развития системы:

- модульная архитектура: возможность добавления новых функций без изменения основной структуры приложения.
- документация: наличие подробной технической документации для разработчиков и пользователей.
- обновления: регулярное обновление приложения с исправлением ошибок и добавлением новых функций.

Design (Дизайн)

Требования к визуальному оформлению [18]:

- эстетичный дизайн: современный и минималистичный интерфейс.
- использование приятных цветовых схем и шрифтов (рисунок 4).

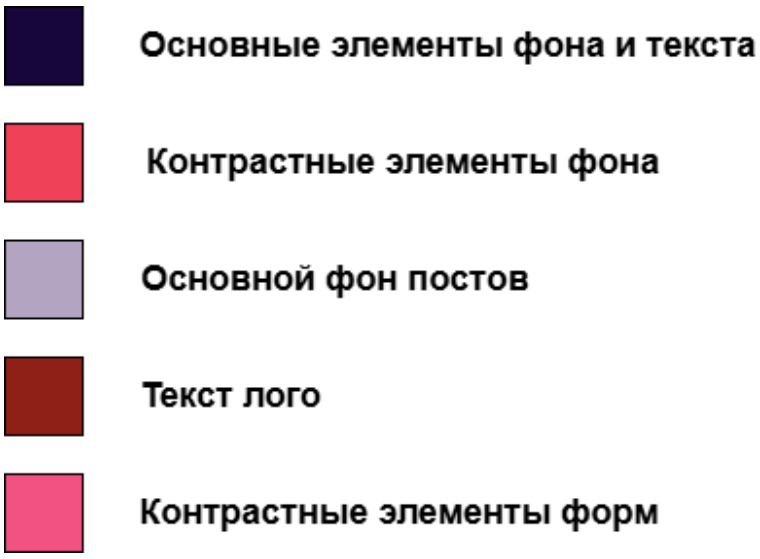

Рисунок 4 – Палитра приложения по желанию заказчика

Implementation (Реализация)

Требования к технологиям и реализации:

- Использование современных технологий:

- frontend: React.js
- backend: Node.js
- СУБД: MongoDB
- Облако для хранения изображений: Cloudinary

Physical (Физические требования)

Требования к аппаратному обеспечению:

- серверные требования: Поддержка облачных решений (Cloudinary).
- минимальные требования к серверному оборудованию для локального развертывания.
- клиентские требования: Поддержка современных браузеров (Chrome, Firefox, Yandex).

Вывод по первой главе

В первой главе выпускной квалификационной работы проведен анализ актуальности разработки специализированного веб-приложения для художников, а также исследованы существующие платформы для публикации и продажи цифрового искусства.

Сравнительный анализ платформ показал, что ни одно из решений не соответствует всем критериям удобства, функциональности и доступности для русскоязычной аудитории. Наивысший балл (7/10) получила платформа «ВКонтакте», но она не предоставляет встроенных механизмов для прямых продаж.

Сформулированы требования к разрабатываемому приложению по модели FURPS+, включая: функциональность (загрузка работ, гибкая монетизация, поиск по тегам), удобство (минималистичный интерфейс на React.js, адаптированный под новичков), надежность (шифрование данных, резервное копирование), технологии (стек MongoDB, Express, React, Node.js) + Cloudinary для хранения изображений.

2 Процесс разработки программного обеспечения

2.1 Описание и обоснование решения по системной архитектуре проекта

Существует обширный выбор системных архитектур, обладающих различными функциями, полезными для того или иного проектного решения. Наиболее распространённым выбором архитектуры для разработки веб-приложения является «Клиент-Сервер» (рисунок 5).

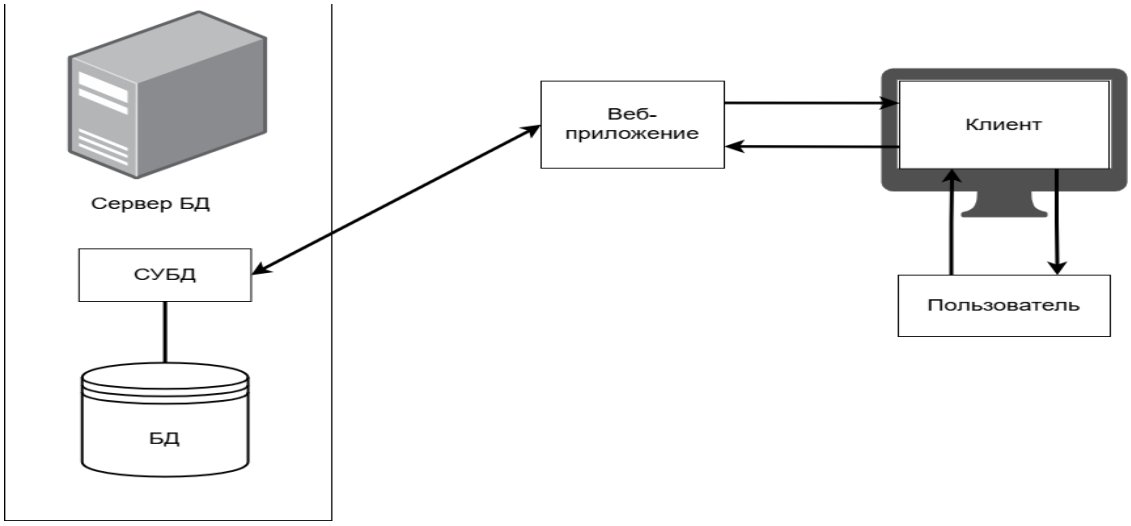

Рисунок 5 – Архитектура «клиент-сервер» веб-приложения

В архитектуре веб-приложения клиент, сервер, база данных и система управления базами данных (СУБД) играют ключевые роли и взаимодействуют друг с другом посредством запросов и ответов [14].

Взаимодействие между клиентом и сервером: клиент представляет собой интерфейс, с которым взаимодействует пользователь, выполняя различные действия в веб-приложении. Когда пользователь выполняет какое-либо действие, клиент отправляет HTTP-запрос на сервер, где он обрабатывается.

Сервер и СУБД: сервер, получив запрос от клиента, может потребовать информацию из базы данных. Для этого сервер формирует запрос и отправляет его в СУБД, которая управляет данными, хранящимися в базе данных. Когда сервер получает данные от СУБД, он преобразует их в формат, который может обработать клиент, и отправляет ему ответ на запрос [8].

Данная архитектура обладает высокой скоростью обмена данными, при этому допуская сравнительно низкие технические характеристики клиента, что и стало главным критерием для выбора именной этой системной архитектуры.

Для реализации проекта необходимо выполнить логическое моделирование программного обеспечения посредством построения UML диаграммы вариантов использования и диаграммы классов. (рисунок 6-7)

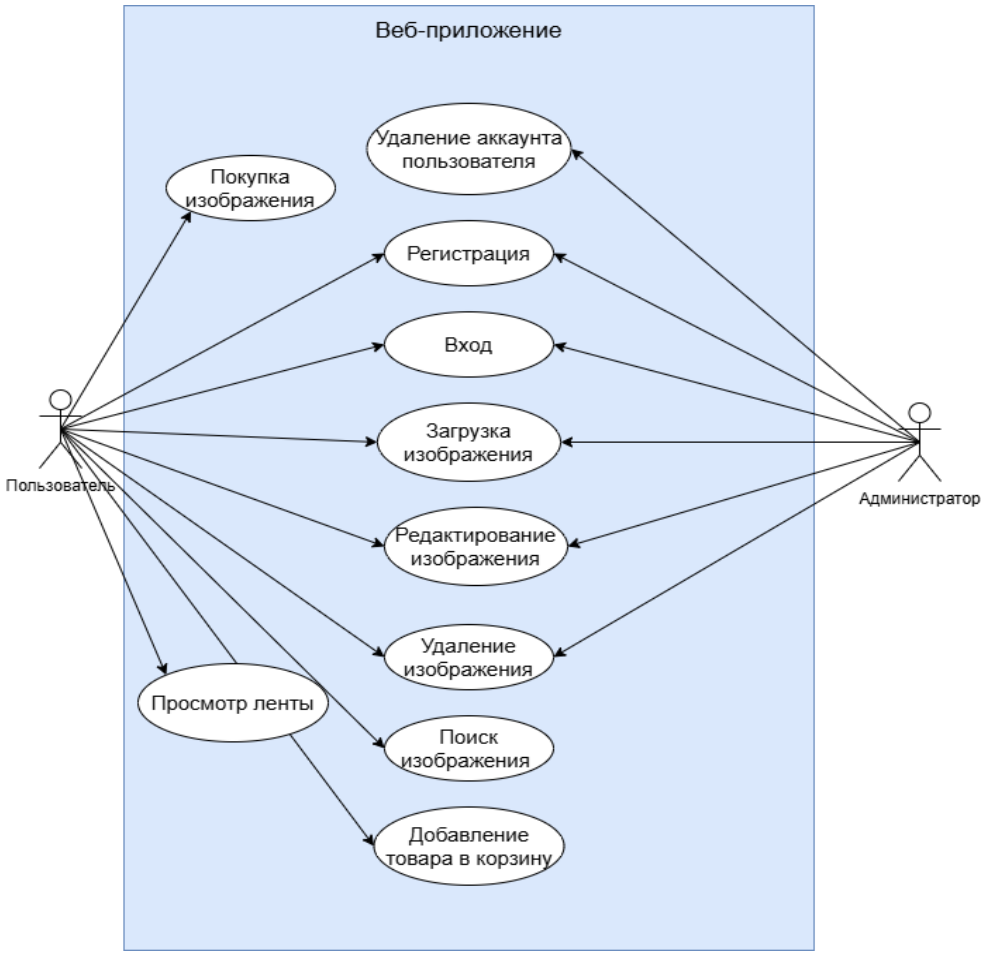

Рисунок 6 – Диаграмма вариантов использования

На рисунке 6 представлены два актора, совершающих действия в веб-приложении:

- пользователь – актор, который создает профиль в веб-приложении для возможности публиковать, редактировать и удалять свои посты с изображениями, просматривать и искать чужие, а также добавлять в корзину и покупать понравившиеся работы.
- администратор – пользователь веб-приложения, наделенный особыми правами, которые позволяют ему помимо базовых функций выполнять такие действия, как редактирование и удаление постов и аккаунтов других пользователей [3].

В таблице 2 отражена краткая характеристика прецедентов, которые были описаны в диаграмме вариантов использования.

Таблица 2 – Краткая характеристика прецедентов

Прецедент	Характеристика
Регистрация и вход	Регистрация с последующим входом в систему пользователя
Загрузка изображения	Администратор или пользователь загружает изображение, добавляет теги и описание. Вся информация автоматически добавляется в БД
Редактирование изображения	Изменение содержимого поста с изображением пользователем или администратором
Удаление изображения	Пользователь по желанию или администратор при несоблюдении контента правилам платформы удаляет пост с изображением
Поиск	Осуществление поиска изображения посредством поисковой строки или по тегам
Просмотр информации об изображении	Пользователь переходит на страницу поста, где более детально предоставлена информация об изображении
Добавление в корзину	Пользователь добавляет в корзину понравившееся изображение
Удаление аккаунта пользователя	Администратор принудительно удаляет аккаунт пользователя

Диаграмма классов UML представленная на рисунке 7, необходима для наглядного отображения структуры системы, её основных сущностей, их атрибутов, методов и взаимосвязей.

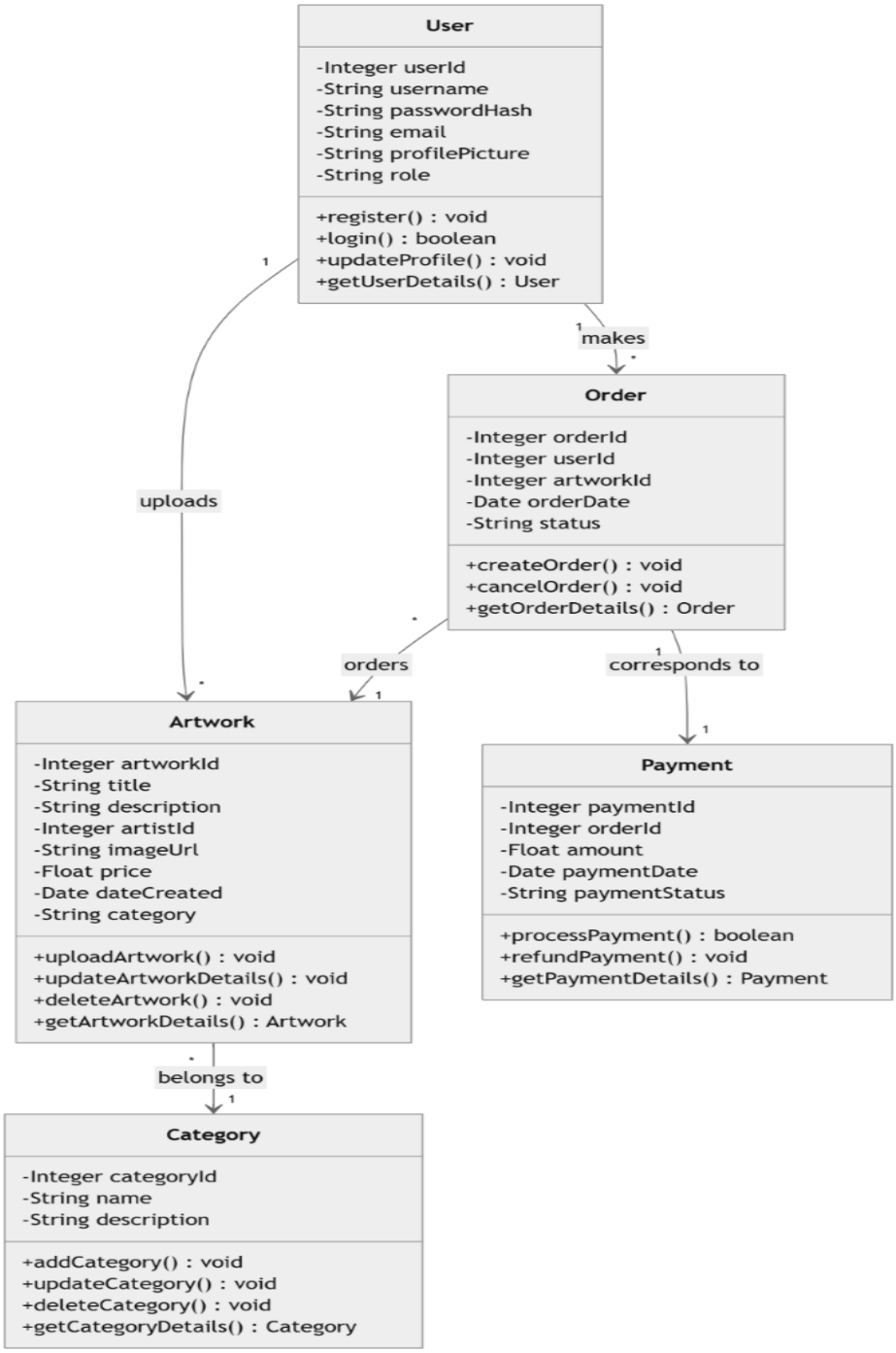

Рисунок 7 – Диаграмма классов веб-приложения

User – данные о пользователе, содержащие уникальный id, имя пользователя, хешированный пароль, email и роль (пользователь или администратор).

Order –данные о заказе, содержащая id заказа, пользователя, купившего изображение и id самого изображения, а также дату совершения сделки.

Artwork – данные об id изображения, его владельца, описание, теги и цена, а также информация о категории, к которой относится изображение.

Category – данные о категориях, на которые подразделяются изображения. Например, диджитал или традиционный арт.

Payment – информация о заказе, цене, дате и статусе.

Отношения классов:

- User(Пользователь) загружает Artwork(Изображение) и делает Order(покупку)
- Order(покупка) соответствует Payment(оплата)
- все Artwork(работы) принадлежат соответствующей Category(категории).

2.2 Выбор технологий разработки программного обеспечения

Перед началом разработки приложения необходимо выбрать подходящие технологии, такие как языки программирования, фреймворки, базы данных, инструменты и платформы, которые будут использоваться для создания будущего ПО. Для того, чтобы выбрать подходящие технологии, необходимо провести сравнительный анализ для каждого элемента. Начать стоит с выбора языка программирования. Наиболее популярные ЯП (язык программирования) для разработки веб-приложений это JavaScript, Python и Java.

Критерии оценки выбранных языков программирования:

- читаемость кода: насколько легко читать и понимать код другим программистом?

– простота синтаксиса: насколько простой и интуитивно понятный синтаксис языка?
– обучение: насколько легко новичку начать писать на языке?
– инструменты и библиотеки: обладает ли ЯП большим набором инструментов и библиотек для разработки?
– интеграция с веб-технологиями: есть ли возможность интеграции с популярными веб-фреймворками и технологиями и насколько легко это сделать.
– производительность: скорость выполнения кода и требования к ресурсам.
– совместимость: совместим ли ЯП с различными платформами и браузерами.
– масштабируемость: способен ли ЯП поддерживать большие и сложные проекты?

Результаты сравнения представлены в таблице 3.

Таблица 3 – Сравнение языков программирования

Критерий	JavaScript	Python	Java
Читаемость кода	-	+	+
Простота синтаксиса	+	+	+
Обучение	+	+	-
Наличие инструментов	+	+	+
Возможность интеграции	+	-	-
Производительность	+	-	+
Совместимость	+	+	+
Масштабируемость	+	-	+
Результат	7/8	5/8	6/8

По результатам сравнительной статистики, JavaScript – наиболее подходящий язык программирования для написания веб-приложения. JS поддерживается большинством современных браузеров и IDE редакторов кода.

React — это JavaScript-библиотека для создания пользовательских интерфейсов (UI), разработанная Facebook. React.js обладает самой обширной экосистемой инструментов и библиотек, а также наиболее активным сообществом разработчиков среди всех альтернатив [6]. Он пользуется популярностью за счёт модульного подхода к разработке, оптимизированного процесса обновления интерфейса, значительно повышающего производительность и встроенного прямо в JavaScript XML синтаксиса, позволяющего внедрять HTML прямо в код [4]. Поэтому именно данный фреймворк будет использован для разработки будущего веб-приложения.

Критерии сравнения СУБД:

- гибкость структуры данных: обладает ли база данных гибкой схемой данных?
- поддержка JSON: есть ли возможность хранить и обрабатывать данные в формате JSON?
- масштабируемость: поддерживает ли база данных масштабирование?
- производительность при больших объемах данных: способна ли база данных эффективно работать с большими объемами данных, обеспечивая высокую скорость выполнения запросов?
- поддержка полнотекстового поиска: есть ли возможность выполнять полнотекстовый поиск по данным?
- простота настройки и администрирования: насколько СУБД проста настройке и администрировании
- доступна ли база данных по бесплатной лицензии?
- интеграция с Node.j: есть ли возможность легко интегрировать базу данных с Node.js, используя драйверы или библиотеки?
- поддержка репликации: поддерживает ли СУБД репликацию данных?

Результаты сравнения приведены в таблице 4.

Таблица 4 – Сравнение СУБД

Критерии	MongoDB	PostgreSQL	MySQL
Гибкость структуры данных	+	-	-
Поддержка JSON	+	+	+
Масштабируемость	+	+	+
Производительность	+	+	-
Поддержка полнотекстного поиска	-	+	+
Настройка и администрирование	+	-	-
Бесплатная лицензия	+	+	+
Интеграция с Node.js	+	+	+
Поддержка репликации	+	+	+
Результат	8/9	7/9	7/9

Исходя из результатов сравнительной таблицы, MongoDB является оптимальным выбором для разрабатываемого веб-приложения для художников. Ее гибкая документоориентированная модель данных позволяет легко хранить и изменять изображения и метаданные без строгой схемы. Поддержка JSON упрощает интеграцию с современными веб-технологиями, а горизонтальное масштабирование обеспечивает высокую производительность даже при больших объемах данных. MongoDB легко интегрируется с Node.js через библиотеку Mongoose, что необходимо при разработке веб-приложений [2].

2.3 Проектирование базы данных

Построение ER-диаграммы, логической и физической модели базы данных — это ключевые этапы проектирования любой системы, работающей с данными.

ER-диаграмма позволяет визуализировать структуру данных на концептуальном уровне, выделяя сущности, их атрибуты и связи между ними.

Это помогает понять, какие данные будут храниться в системе, как они связаны друг с другом [13, с. 43].

Нотаций для создания ER-диаграмм довольно много, поэтому необходимо остановиться на одном варианте.

Основные виды нотаций:

– нотация Чена
– нотация Мартина (воронья лапка)
– IDEF1X
– нотация Баркера
– ORM

Нотации IDEF1X, Баркера и ORM не подходят из-за того, что ориентированы преимущественно на реляционные базы данных, в то время как в проекте используется документоориентированная NoSQL БД MongoDB.

Нотации Чена и Мартина отлично приспособлены для нереляционных БД, поэтому выбор будет производиться из них.

Нотация Чена использует простые и понятные элементы: прямоугольники для сущностей, ромбы для связей и овалы для атрибутов. Это делает диаграмму легко читаемой даже для неспециалистов, по сравнению с нотациями Мартина и UML, обладающими более компактной, но менее понятной для простого пользователя структурой [7].

Еще одним преимуществом нотации Чена является поддержка необязательных связей, которые обозначаются кругами на линиях. Это важно для моделей, где не все сущности обязаны участвовать в связях [5, с.51]. Например, пользователь может не совершать заказов, что было явно обозначено на диаграмме (рисунок 8).

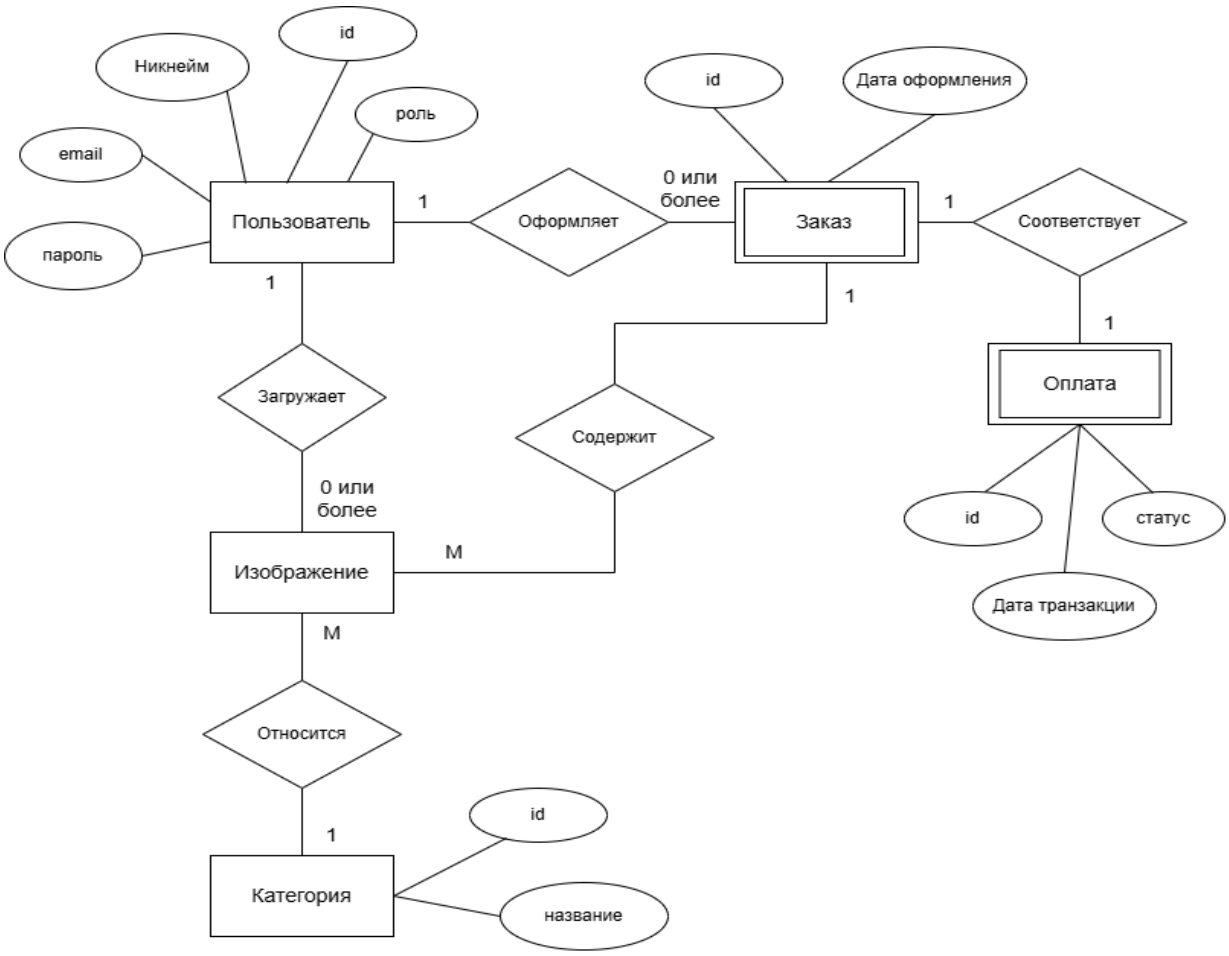

Рисунок 8 – ER-диаграмма нотации Чена

Модель данных углубляет описание структуры данных, добавляя ключи и более детализированные связи. Она служит мостом между концептуальным уровнем и физической реализацией, позволяя выявить потенциальные проблемы на ранних этапах проектирования. Одной из ключевых особенностей MongoDB является возможность хранения вложенных документов и массивов [2]. Это позволяет группировать связанные данные в одном документе, что упрощает чтение и запись. При построении логической модели важно решить, какие данные будут вложенными, а какие — ссылочными. Вложенные документы улучшают производительность для часто запрашиваемых данных, но могут привести к дублированию. Ссылки, напротив, нормализуют данные, но требуют дополнительных запросов для получения связанной информации (рисунок 9).

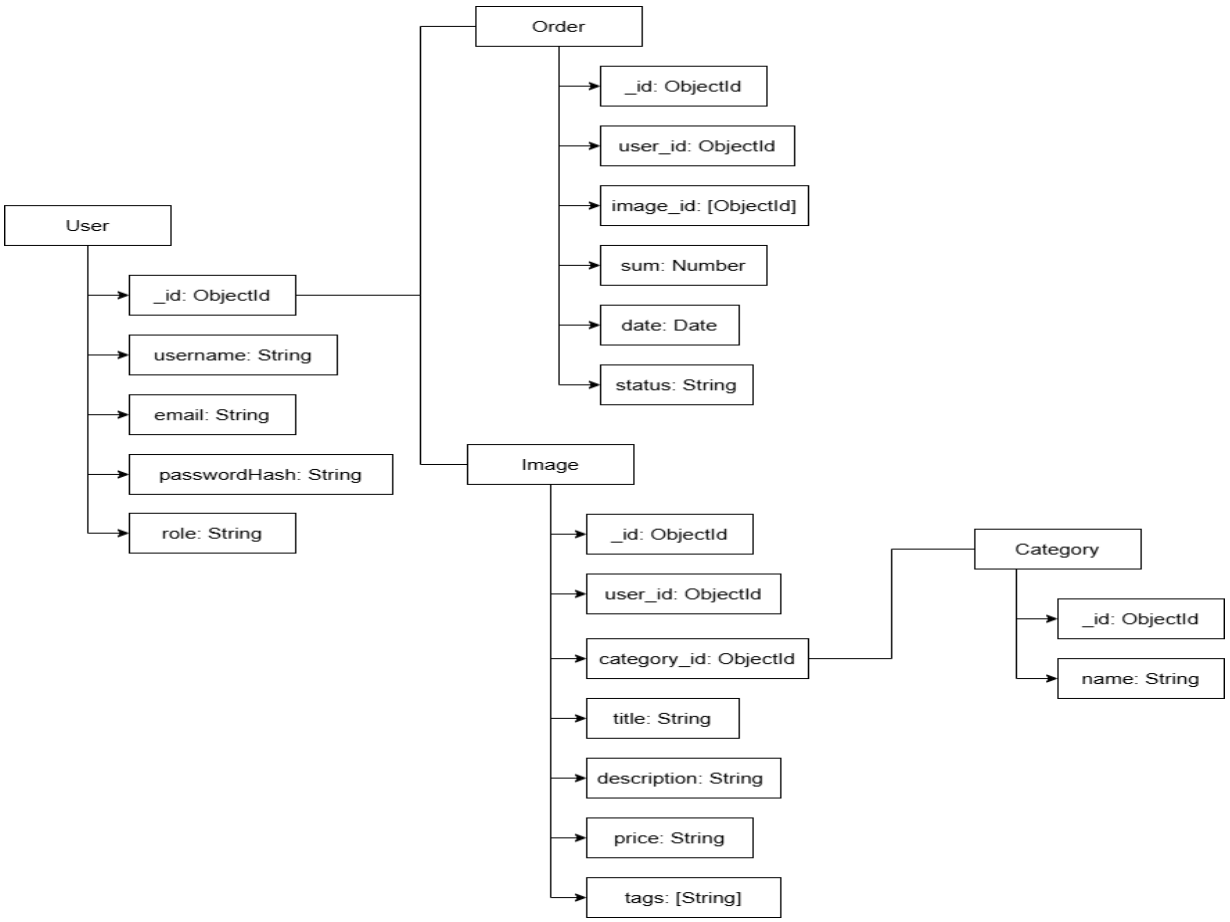

Рисунок 9 – Модель базы данных

Модель данных представляет 5 основных коллекций:

– коллекция «Пользователь» содержит данные пользователя, которые привязаны к его аккаунту, такие как никнейм, email и роль (пользователь или администратор);
– коллекция «Изображение» хранит в себе информацию о названии, описании (если оно есть), цене (если она есть) и тегах (если они есть), а также ссылается на коллекции «Пользователь» и «Категория» с помощью их ObjectId;
– коллекция «Категория» содержит информацию о категориях, к которым относятся все изображения;
– коллекция «Заказ» получает информацию из коллекций «Пользователь» и «Изображение» через ссылку, при чём id

изображения может содержать массив из нескольких ссылок, а также дату об оформлении заказа;

В MongoDB вместо используемых в реляционных базах данных типов данных VARCHAR и INTEGER используются такие типы данных, как ObjectID для уникальных ключей, String для строковых значений и Number для числовых.

Для реализации БД веб-приложения использована база данных MongoDB с подключением к СУБД MongoDB Compass. На рисунке 11 приведена примерная структура будущей базы данных.

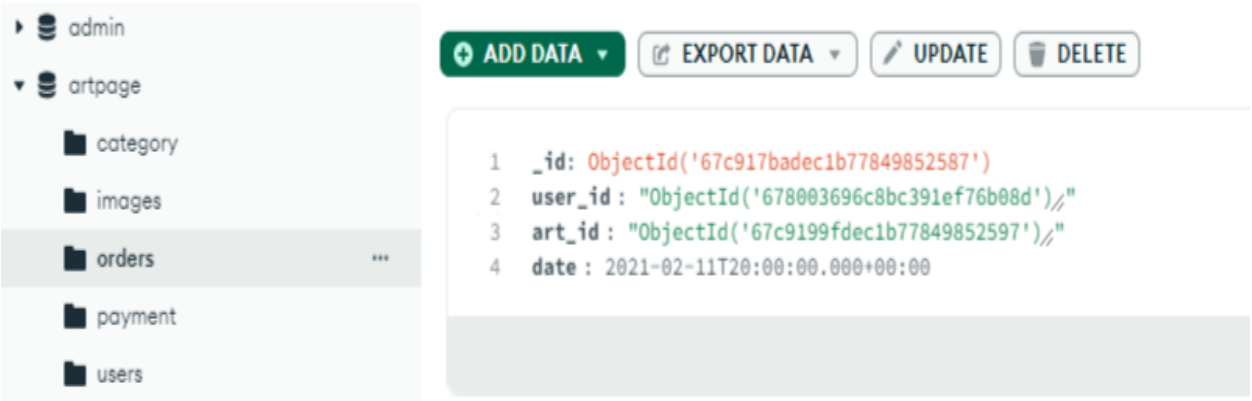

Рисунок 11 – База данных веб-приложения в Compass

На рисунке 12 изображены примеры записей данных пользователей коллекции users.

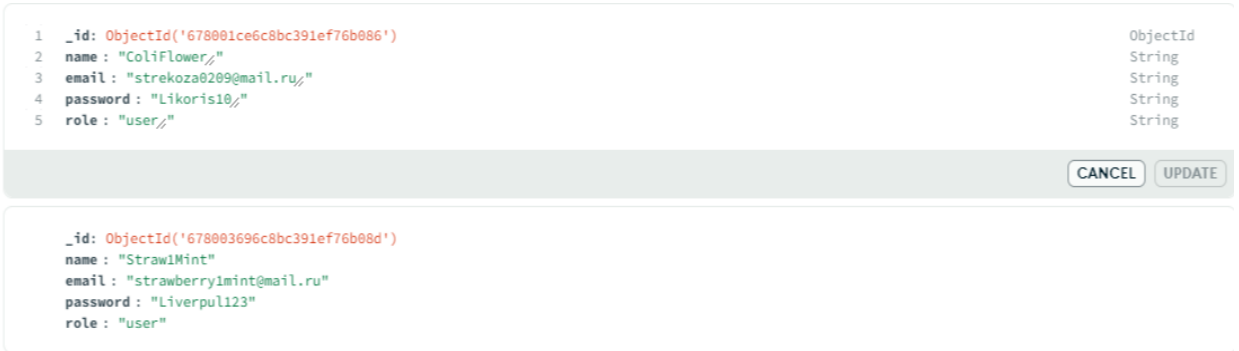

Рисунок 12 – Элементы коллекции users

Аналогом таблиц здесь являются коллекции (images, orders, users, payment, category), в которых хранятся документы – записи со всеми данными по каждому отдельному элементу коллекции.

2.4 Реализация программного обеспечения

Перед реализацией ПО, необходимо отобразить его архитектуру. Это позволит наглядно показать то, из каких компонентов состоит веб-приложение, и как они взаимодействуют друг с другом. Для таких целей создается архитектурная схема компонентов и диаграмма деятельности. (рисунок 13-15).

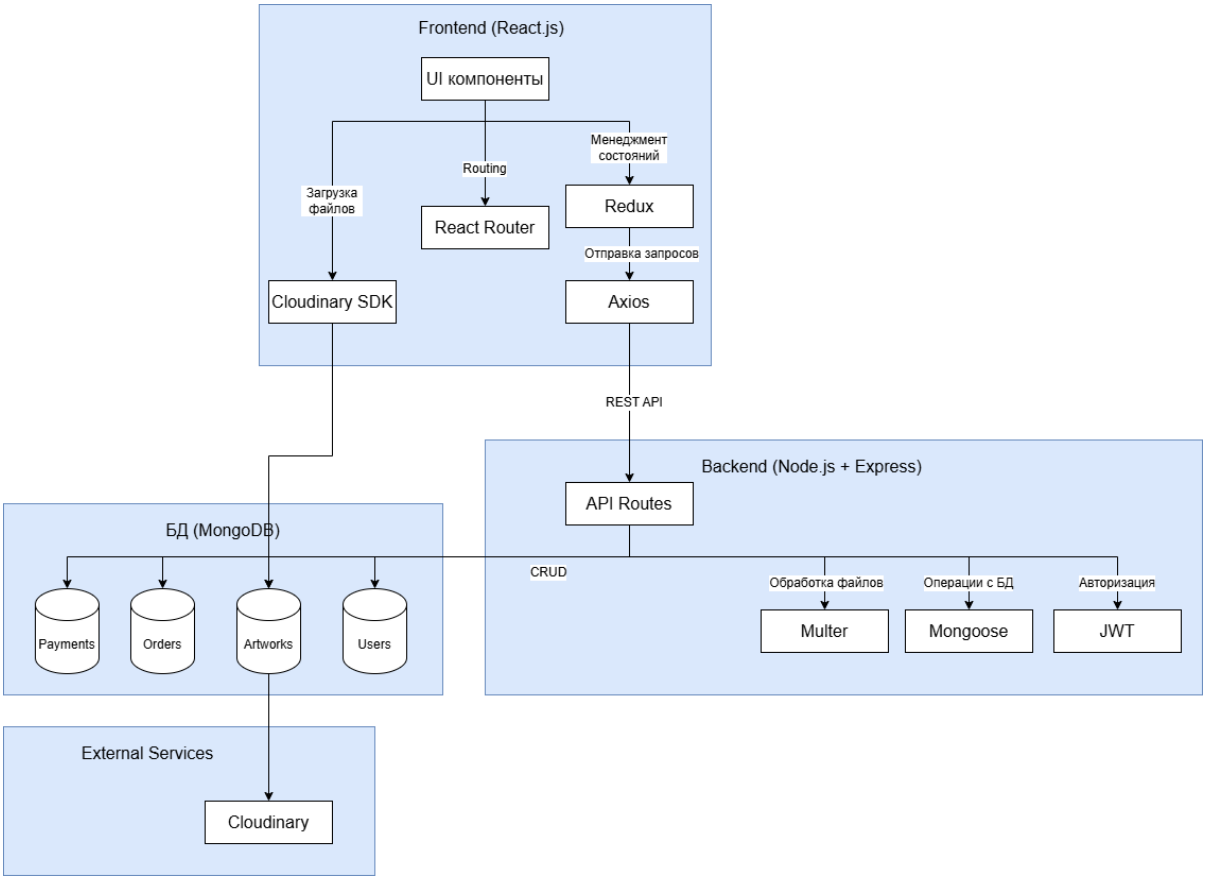

Рисунок 13 – Архитектурная схема компонентов с указанием технологий

Ниже приведены таблицы 5-8, которые объясняют ключевые аспекты архитектуры и работы веб-приложения подробнее.

Таблица 5 – Компоненты Frontend (React.js) [15]

Компонент	Технологии/Библиотеки	Назначение
UI Components	React, Material-UI	Кнопки, формы, карточки работ
Redux	Redux Toolkit	Хранение состояния (авторизация, корзина, список работ)
React Router	React Router v6	Навигация между страницами (галерея, профиль, форма загрузки)
Axios	Axios	Отправка HTTP-запросов к бэкенду
Cloudinary SDK	cloudinary-react	Загрузка изображений напрямую в Cloudinary

Таблица 6 – Backend (Node.js + Express)

Модуль	Технологии	Назначение
API Routes	Express.js	Маршруты для: /api/auth, /api/artworks, /api/orders
JWT	jsonwebtoken	Аутентификация пользователей
Mongoose	Mongoose	Взаимодействие с MongoDB (схемы для User, Artwork, Order)
Multer	multer	Обработка загружаемых файлов

Таблица 7 – Database (MongoDB)

Коллекция	Поля (пример)	Связи
Users	_id, email, passwordHash, role	Имеет много Artworks
Artworks	_id, title, artistId, price, category	Принадлежит User/Category
Orders	_id, artworkId, buyerId, date	Связан с Payment
Payments	_id, orderId, amount, status	Зависит от Order

Таблица 8 – External Services

Сервис	Назначение
Cloudinary	Хранение и оптимизация изображений (экономит место в вашей БД)

Ключевые взаимодействия:

– пользователь загружает работу: Frontend (React) → Multer (Node.js) → Cloudinary → URL сохраняется в MongoDB

– добавления товара в корзину: Frontend → POST /api/orders → Создается Order → Генерируется Payment

– аутентификация: Frontend → POST /api/auth → JWT токен → Сохраняется в Redux + LocalStorage

Ниже представлены два ключевых сценария взаимодействия пользователей с системой: загрузка изображения пользователем-художником

(рисунок 14) и покупка изображения пользователем-покупателем (рисунок 15).

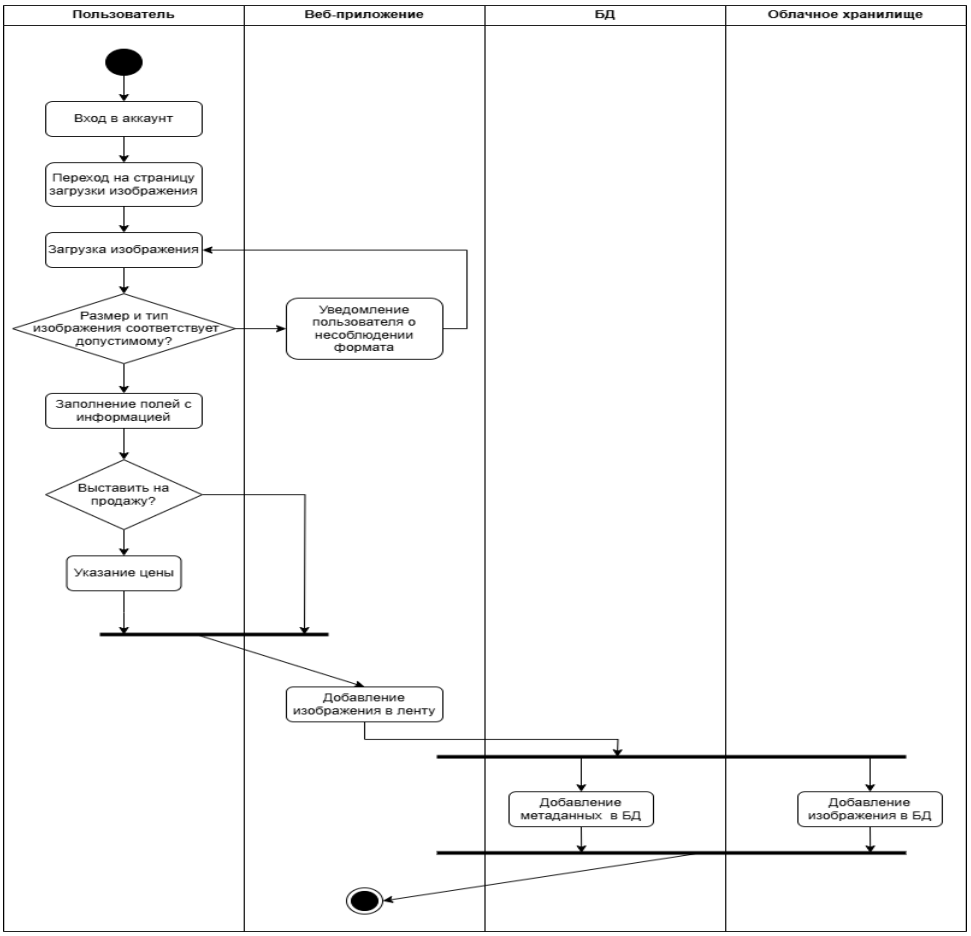

Рисунок 14 – Диаграмма действий сценария загрузки изображения

Процесс загрузки изображения начинается с аутентификации пользователя в веб-приложении. После успешной авторизации система перенаправляет пользователя на главную страницу интерфейса. Для инициирования процесса загрузки графического контента пользователю требуется активировать элемент интерфейса, обозначенный пиктограммой фотоаппарата.

После перехода на страницу загрузки пользователь должен:

– заполнить обязательные поля формы;
– определить статус изображения (платное/бесплатное распространение);

– загрузить файл графического содержимого.

Система автоматически выполняет валидацию загружаемых данных на соответствие установленным форматам. В случае обнаружения несоответствий генерируется соответствующее уведомление для пользователя.

При активации элемента интерфейса "Опубликовать" система последовательно выполняет следующие операции:

– публикация изображения в ленте новостей главной страницы;

– запись метаданных в базу данных;

– передача файла изображения в облачное хранилище.

Каждая операция сопровождается соответствующим протоколированием событий и проверкой статуса выполнения.

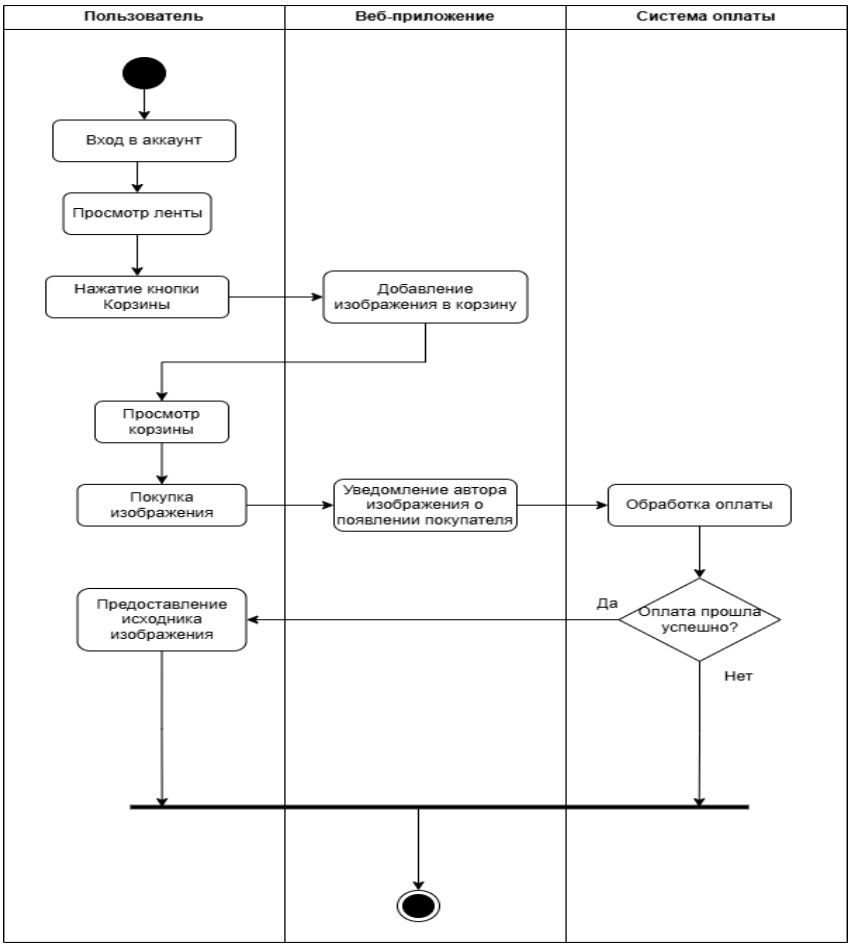

Рисунок 15 – Диаграмма действий сценария покупки изображения

Сценарий «Пользователь покупает изображение»:

Процедура добавления товара в корзину начинается с успешной аутентификации пользователя в системе. После авторизации пользователь получает доступ к основной ленте публикаций, содержащей доступные для приобретения графические материалы.

Для добавления товара в корзину пользователю необходимо:

- осуществить выбор требуемого изображения;
- активировать соответствующий элемент интерфейса («Добавить в корзину»).

Система предоставляет следующие варианты продолжения:

- немедленный переход к процедуре оформления заказа;
- отложенное оформление с возможностью последующего доступа к содержимому корзины через соответствующий раздел интерфейса.

При принятии решения о совершении покупки пользователь должен:

- перейти в раздел корзины;
- подтвердить состав заказа;
- активировать процесс перехода на страницу оплаты.

Описание реализованных модулей:

Модуль работы с базой данных. Для хранения данных о пользователях была подключена база данных MongoDB. В коллекции «users» создаются документы, содержащие информацию о зарегистрированных пользователях (рисунок 16). Каждый документ включает следующие поля:

- email (электронная почта);
- username (никнейм);
- password (хешированный пароль).

```
// Подключаемся к MongoDB
mongoose.connect('mongodb://localhost:27017/mydatabase', {
    useNewUrlParser: true,
    useUnifiedTopology: true,
})
.then(() => {
    console.log('Подключено к базе данных MongoDB');
})
.catch(err => {
    console.error('Ошибка подключения к базе данных:', err);
});

// Определяем модель пользователя
const UserSchema = new mongoose.Schema({
    username: String,
    email: String,
    password: String,
});

const User = mongoose.model('User', UserSchema);
```

Рисунок 16 – Листинг подключения к MongoDB

Модуль обработки данных регистрации [19]. Был реализован код обработчика регистрации, который выполняет следующие функции:

– получает данные, введенные пользователем на странице регистрации;
– проверяет корректность данных (валидация email, уникальность никнейма);
– в случае успешной проверки создает новый документ в коллекции users базы данных MongoDB;
– уведомляет пользователя об успешной регистрации или об ошибках, если таковые возникли (рисунок 17).

```
// Обработчик регистрации
app.post('/register', async (req, res) => {
    const { username, email, password } = req.body;

    const newUser = new User({ username, email, password });
    try {
        await newUser.save();
        res.status(201).json({ message: 'Пользователь успешно зарегистрирован!' });
    } catch (error) {
        res.status(500).json({ message: 'Ошибка регистрации пользователя', error });
    }
});

// Запускаем сервер
const PORT = 5000;
app.listen(PORT, () => {
    console.log(`Сервер запущен на http://localhost:${PORT}`);
});
```

Рисунок 17 – Листинг обработки данных регистрации

Главная страница веб-приложения является центральным элементом пользовательского интерфейса, обеспечивающим доступ ко всем основным функциям системы (рисунок 18). Она включает следующие ключевые компоненты:

– лента изображений;
– форма поиска;
– кнопка добавления изображения;
– кнопка перехода на страницу пользователя;
– кнопка выхода из системы.

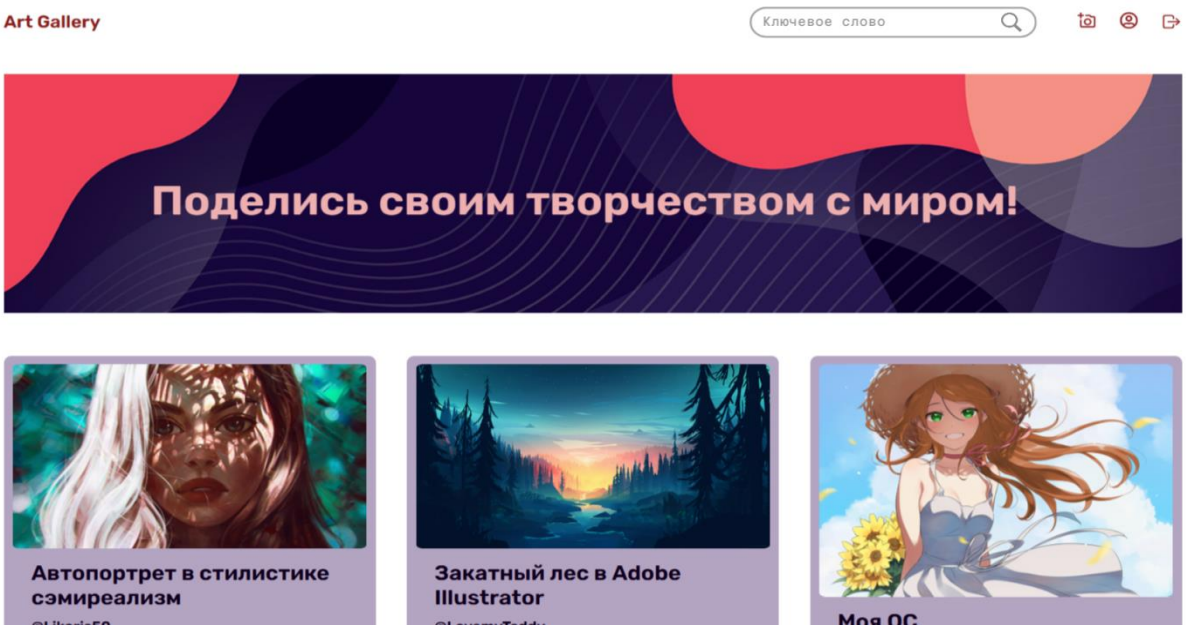

Рисунок 18 – Главная страница

Лента изображений представляет собой динамически обновляемый список загруженных пользователями изображений [17]. Каждое изображение отображается с краткой информацией, включающей:

– название изображения;
– имя автора (никнейм пользователя);
– миниатюру изображения;
– кнопку для перехода на страницу изображения с подробной информацией.

Кнопка добавления изображения предоставляет пользователю возможность загрузить новое изображение в систему. Данный функционал доступен только авторизованным пользователям.

Кнопка перехода на страницу пользователя позволяет авторизованным пользователям получить доступ к персональной странице, где отображается:

– информация о пользователе (никнейм, электронная почта);
– список загруженных изображений.

Кнопка выхода из системы позволяет пользователю завершить текущую сессию. При нажатии на кнопку:

– удаляются данные о текущей авторизации (токены, cookies);
– пользователь перенаправляется на главную страницу с ограниченным доступом (только просмотр ленты изображений).

Модуль загрузки изображения (рисунок 19):

Модуль загрузки изображения позволяет пользователям добавлять новые изображения в систему, которые затем отображаются в ленте на главной странице. Данный функционал доступен авторизованным пользователям. Процесс загрузки изображения включает несколько этапов, начиная с перехода на страницу загрузки и заканчивая публикацией изображения в ленте.

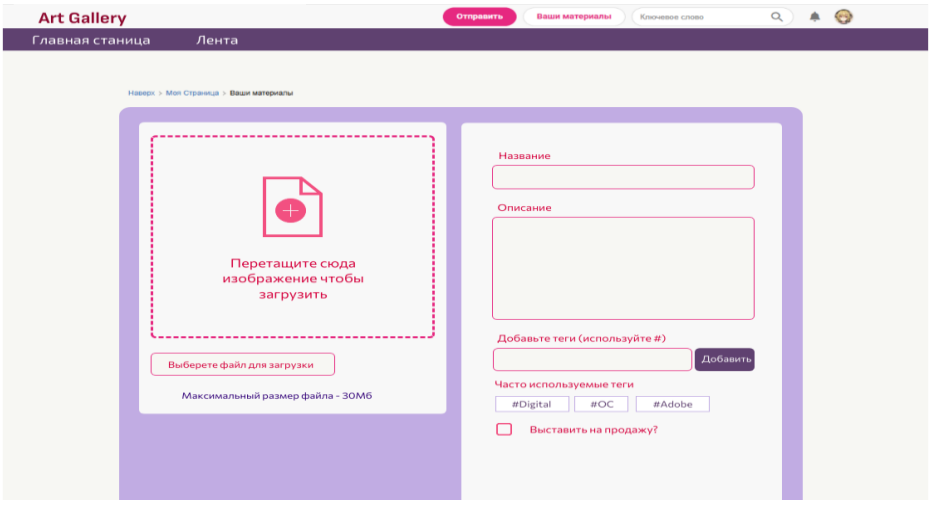

Рисунок 19 – Страница загрузки изображения.

На главной странице приложения, в правом верхнем углу, расположена кнопка с иконкой фотоаппарата. При нажатии на эту кнопку пользователь перенаправляется на страницу загрузки изображения.

Интерфейс страницы загрузки (рисунок 20):

Страница загрузки изображения разделена на две основные панели:

– левая панель: предназначена для загрузки файла изображения. Пользователь может выбрать файл через стандартный диалог выбора или перетащить его в указанную область;
– правая панель: содержит поля для ввода дополнительной информации об изображении:
  – название: поле для ввода названия изображения;
  – описание: поле для добавления описания изображения;
  – теги: поле для ввода тегов, которые добавляются через знак #.

Пользователю может вводить теги вручную или выбирать из списка часто используемых тегов, предлагаемых системой (рисунок 21).

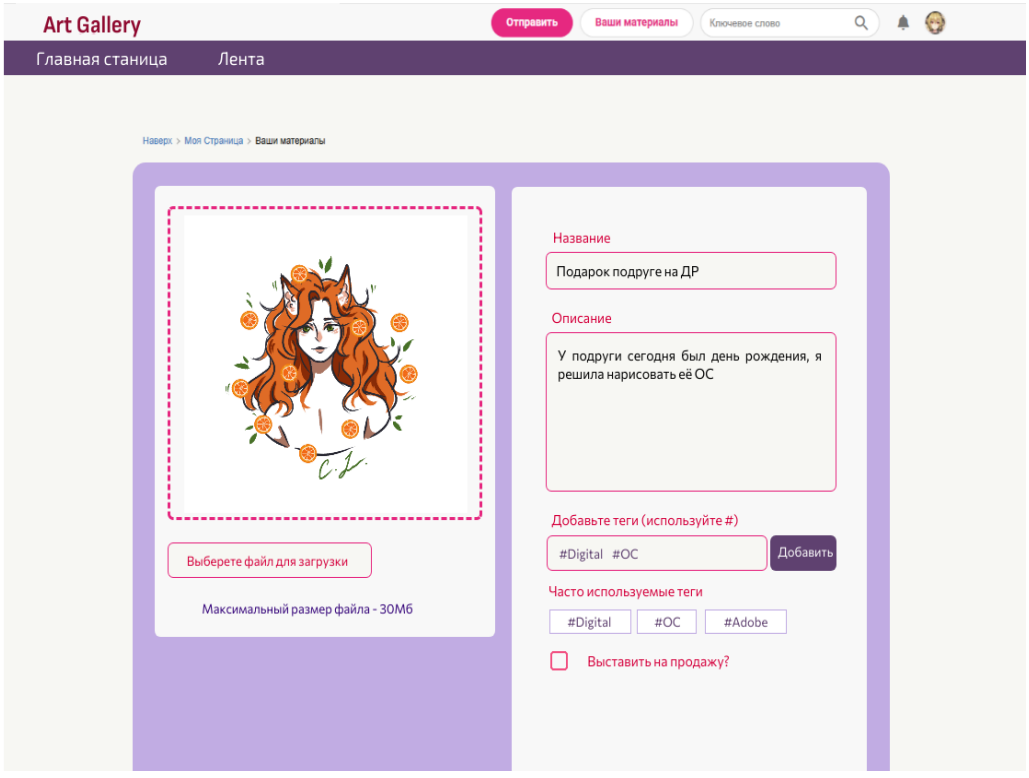

Рисунок 20 – Ввод данных в форму

Ниже полей ввода расположен чек-бокс, который позволяет пользователю указать, что изображение предназначено для продажи. При активации чек-бокса появляется дополнительное поле для ввода цены.

При нажатии на кнопку «Отправить»:

– изображение загружается в облачное хранилище Cloudinary;
– метаданные изображения (название, описание, теги, цена) сохраняются в базе данных MongoDB;
– изображение добавляется в ленту на главной странице (рисунок 21).

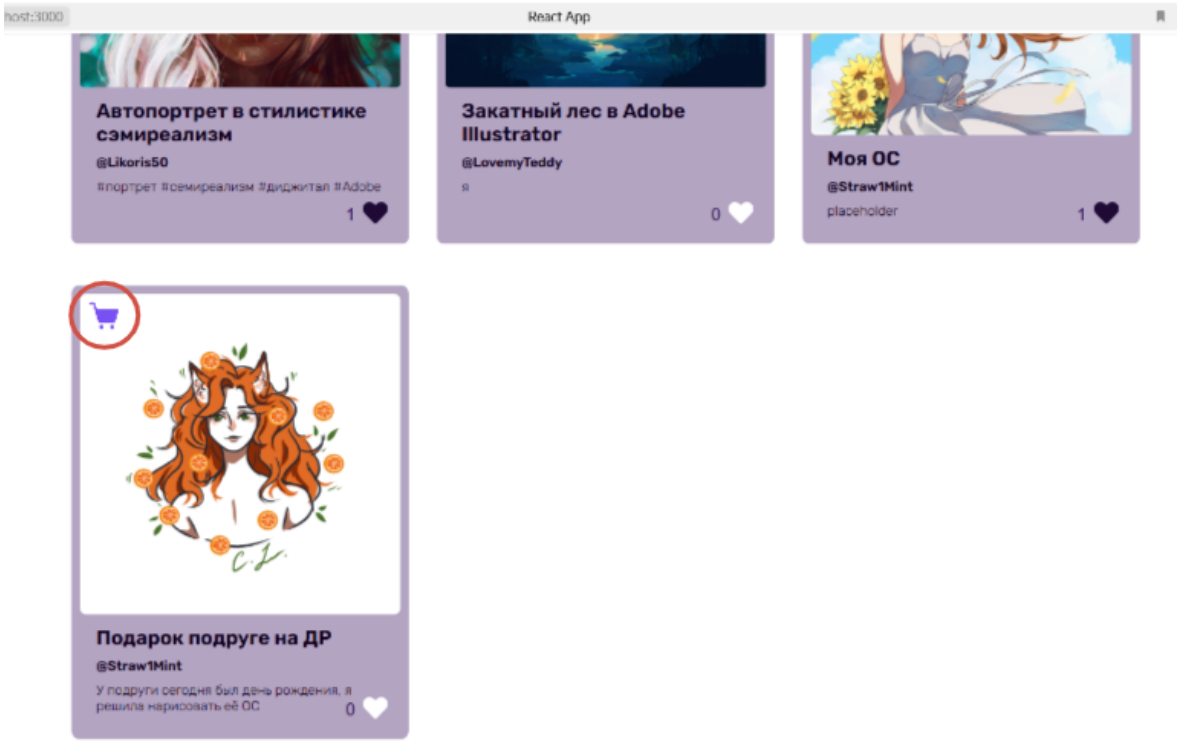

Рисунок 21 – Появление новой публикации в ленте

Отображение изображения в ленте. После успешной загрузки пользователь возвращается на главную страницу, где в ленте отображается новое изображение. Если изображение было выставлено на продажу, в левом углу его миниатюры появляется иконка тележки, указывающая на возможность покупки.

Модуль корзины (рисунок 22) позволяет пользователям просматривать и управлять списком товаров (изображений), которые они планируют приобрести. Интерфейс страницы корзины разделен на две основные части: список товаров и панель подсчета общей стоимости. Данный модуль обеспечивает удобство выбора товаров и подготовки к оплате [16].

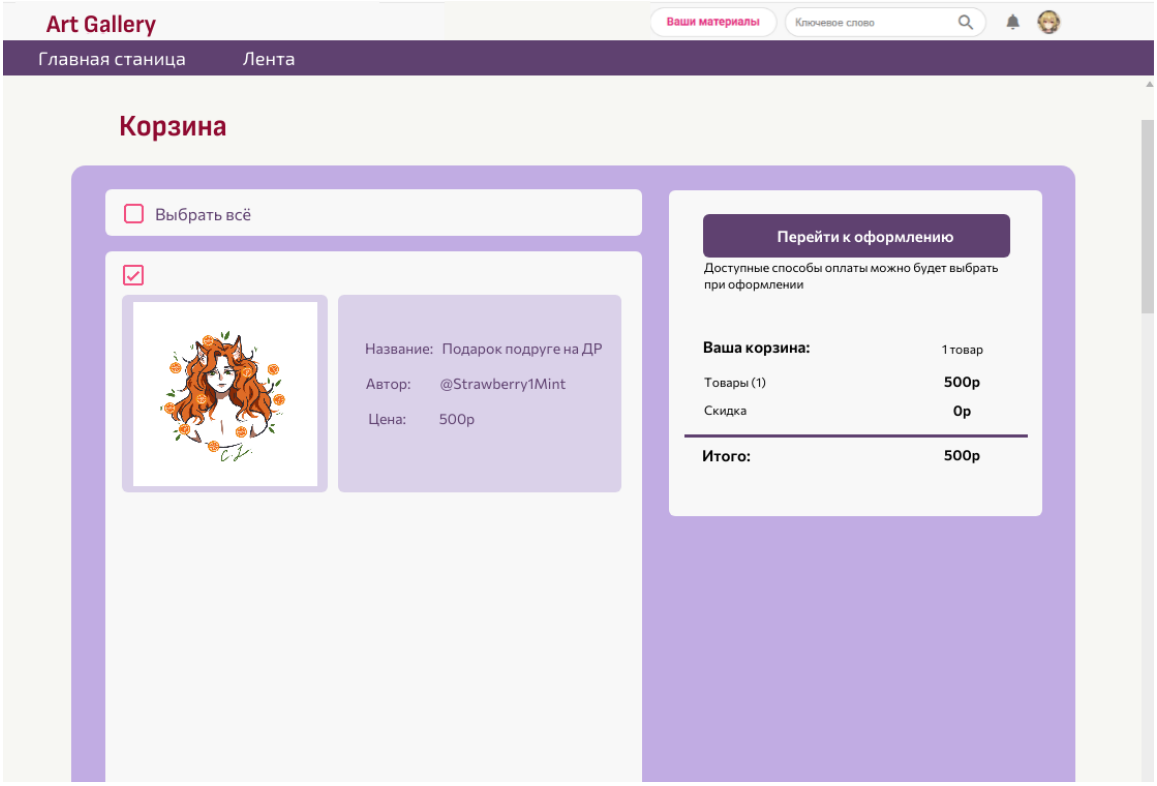

Рисунок 22 – Страница корзины

Список товаров:

В левой части интерфейса отображается список всех товаров, добавленных в корзину. Для каждого товара указывается следующая информация:

– название изображения;
– автор (никнейм пользователя);
– цена изображения.

Над списком товаров расположена кнопка «Выделить все», которая позволяет пользователю быстро выбрать все элементы для дальнейших действий (например, удаление или оплата).

В правой части интерфейса находится панель, которая отображает итоговую информацию о заказе:

- количество товаров: общее число изображений в корзине;
- общая стоимость: сумма цен всех товаров в корзине;
- итоговая стоимость: общая стоимость товаров с учетом скидки (если она будет применена).

В верхней части интерфейса расположена кнопка, предназначенная для перехода на страницу выбора платежных средств и обработки платежей. Реализация данной функциональности выходит за рамки текущего этапа разработки. В будущем планируется интеграция с платежными системами для обеспечения полноценного процесса оплаты.

Вывод по второй главе

Во второй главе ВКР был реализован ключевой этап разработки веб-приложения для цифровых художников. Основой проекта стал стек технологий MERN (MongoDB, Express, React, Node.js), обеспечивающий высокую производительность и масштабируемость. Особое внимание уделено проектированию гибкой документоориентированной базы данных в MongoDB с интеграцией облачного хранилища Cloudinary для медиафайлов.

На фронтенде реализован интерфейс на React.js с системой загрузки и просмотра работ, а на бэкенде - REST API на Node.js с JWT-аутентификацией. Проработаны основные сценарии взаимодействия пользователей. Полученные результаты подтвердили правильность выбранных решений и готовность системы к дальнейшему развитию, включая расширение платежного функционала и системы модерации.

3 Оценка эффективности разработанного программного обеспечения

3.1 Тестирование веб-приложения

Тестирование валидации публикации изображений:

Был вручную написан комплекс тестов, проверяющих, можно ли создать нестандартную запись при добавлении информации для публикации изображения (рисунок 23).

Тест проверяет возможность:

– добавление описания без заголовка;

– добавление описания с «» заголовком;

– добавление описания с заголовком, состоящим только из пробелов;

– добавление описания с заголовком, превышающим максимальную длину;

– добавление описания с заголовком, начинающимся или заканчивающимся пробелом [11, с. 128].

```
describe('Валидация заголовка', () => {
    test.each([
      [null, 'Field is required'],
      [' ', 'Не может состоять из пробелов'],
      [' Начинается с пробела', 'Не может начинаться с пробела'],
      ['Корректный', undefined] // undefined = ошибок нет
    ])('Заголовок "%s" → %s', (title, error) => {
      const artwork = new Artwork({ title });
      if (error) {
        expect(() => artwork.validate()).toThrow(error);
      } else {
        expect(artwork.validate()).resolves.toBeTruthy();
      }
    });
  });
```

Рисунок 23 – Тест-кейсы на Jest

Пройденные тесты:

Параметризованный тест для случаев:

- «X»*150 (слишком длинный заголовок)
  Ожидаемо получили ValidationError (превышение длины)
- «Нормальный заголовок» (валидный кейс)
  Успешное сохранение без ошибок
- «A» (минимально допустимый заголовок) - Успешное сохранение без ошибок
- «С пробелом внутри» (валидный кейс)
  Успешное сохранение без ошибок
- «Несколько слов внутри» (валидный кейс)(проверка на возможность добавления нескольких пробелов между словами)
  Успешное сохранение без ошибок;
- None (полное отсутствие title)
  ValidationError, ошибка типа «Field is required»;
- "" (пустая строка)
  ValidationError, ошибка типа «String value is too short»;
- " " (пробел)
  ValidationError, ошибка типа «Заголовок не может состоять только из пробелов»»
- «Начинается с пробела» (пробел в начале) - ValidationError, ошибка типа «Заголовок не может начинаться с пробела»;
- «Заканчивается пробелом» (пробел в конце)
  ValidationError, ошибка типа «Заголовок не может заканчиваться пробелом»;

На рисунке 24 показан результат запуска тестов в консоли.

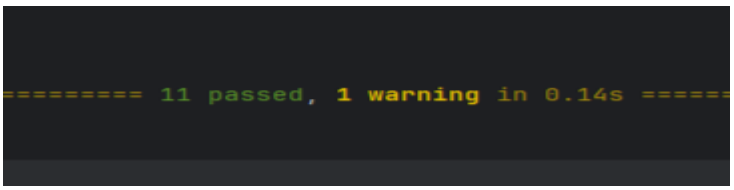

Рисунок 24 – Результаты запуска тестов в Jest

Тестирование уязвимости форм ввода данных в БД. Для тестирования форм на предмет уязвимости БД был выбран инструмент OWASP ZAP. Получая адрес сайта на вход, он запускает симуляцию атаки, которая будет отправлять множественные запросы серверу БД в попытках добавить несоответствующую стандартам вредоносную запись, попутно создавая перегрузку системы запросами, от которой может произойти временный shut-down сервера. Таким образом проводится и инъекционное, и нагрузочное тестирование [9].

По результатам запуска системы сервер не отключился, что говорит о достаточной устойчивости системы, новых несанкционированных записей в БД добавлено не было, а по результатам отчета OWASP ZAP было обнаружено всего пара незначительных недочетов системы, не влияющих на безопасность (рисунок 25).

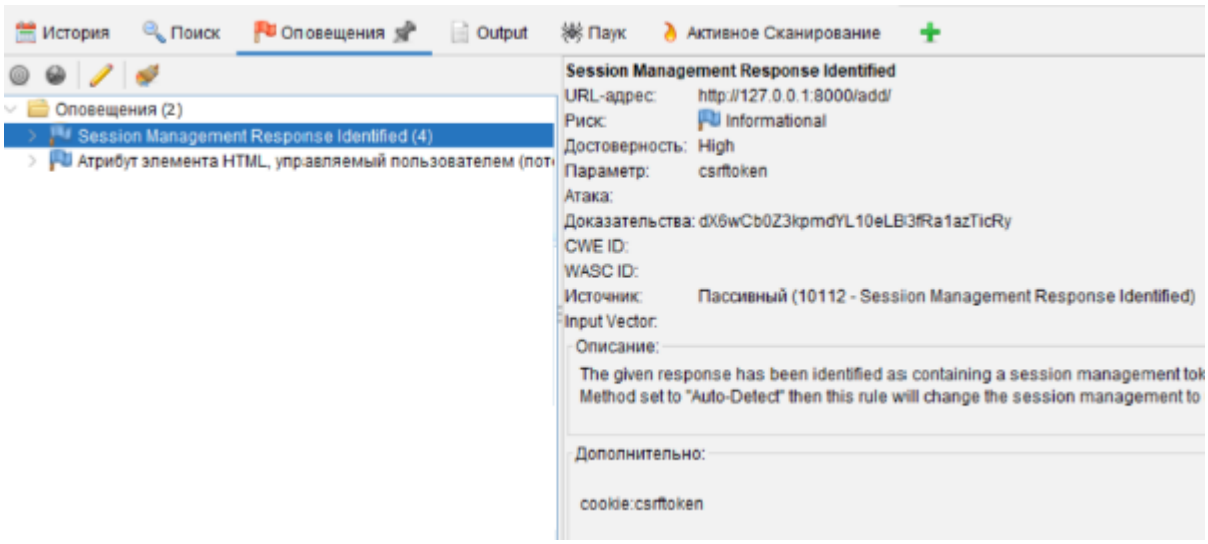

Рисунок 25 – Отчет OWASP ZAP после проведения атаки

Функциональное тестирование. Тестирование положительных сценариев позволяет убедиться, что система работает так, как ожидается, в нормальных условиях (таблица 9).

Таблица 9 – Тестирование положительных сценариев

Тест	Ожидаемый результат	Фактический результат
Регистрация нового пользователя		
Ввод корректных данных (email, пароль, никнейм) и нажатие кнопки "Зарегистрироваться"	Пользователь успешно зарегистрирован, данные сохранены в БД, переход на главную страницу.	+
Проверка уникальности email и никнейма	Система разрешает регистрацию, если email и никнейм уникальны.	+
Авторизация пользователя		
Ввод корректных данных (email и пароль) и нажатие кнопки "Войти"	Пользователь успешно авторизован, переход на главную страницу.	+
Проверка доступа к защищенным страницам после авторизации	Пользователь получает доступ к личному кабинету	+
Загрузка изображения		
Загрузка изображения через выбор файла	Изображение успешно загружено, отображается в ленте, данные сохранены в БД и облаке	+
Загрузка изображения через перетаскивание.	Изображение успешно загружено, отображается в ленте, данные сохранены в БД и облаке	+
Заполнение всех полей (название, описание, теги).	Данные корректно сохранены в БД, отображаются на странице изображения.	+
Заполнение не всех полей	Данные корректно сохранены в БД, незаполненные строки остаются пустыми	+
Проставление галочки в чек-бокс выставления на продажу	Появляется поле для ввода цены, введённые данные корректно сохранены в БД	+
Работа с корзиной		
Добавление изображения в корзину	Изображение отображается в корзине, общая стоимость обновляется	+
Удаление изображения из корзины	Изображение удалено из корзины, общая стоимость обновляется	+
Подсчет общей стоимости товаров в корзине	Общая стоимость корректно рассчитана и отображена	+
Проставление галочки «Выделить всё»	Выделяются все товары из списка, обновляется общая стоимость	+

Продолжение таблицы 9

Тест	Ожидаемый результат	Фактический результат
Отображение ленты изображений		
Загрузка ленты изображений при открытии главной страницы.	Лента изображений отображается корректно, все элементы загружены	+
Отображение возможности покупки (иконка тележки).	Иконка тележки отображается для изображений, выставленных на продажу.	+
Выйти из системы		
Нажатие кнопки "Выйти"	Пользователь успешно выходит из системы, доступ к защищенным страницам ограничен.	+

Отрицательные сценарии помогают выявить, как система справляется с ошибками, и насколько она устойчива к непредвиденным ситуациям. Вместе эти подходы обеспечивают всестороннюю проверку функциональности системы [10] (таблица 10).

Таблица 10 – Тестирование негативных сценариев

Тест	Ожидаемый результат	Фактический результат
Регистрация нового пользователя		
Ввод некорректного email без знака «@»	Система выводит сообщение об ошибке, регистрация не завершена	+
Ввод некорректного email без знака «.»	Система выводит сообщение об ошибке, регистрация не завершена	+
Ввод короткого пароля (менее 6 символов)	Система выводит сообщение об ошибке, регистрация не завершена	+
Ввод уже существующего email или никнейма	Система выводит сообщение о том, что email или никнейм уже заняты	+
Авторизация пользователя		
Ввод некорректного email или пароля	Система выводит сообщение об ошибке, вход не выполнен	+
Попытка входа без заполнения полей	Система выводит сообщение о необходимости заполнить все поля	+

Продолжение таблицы 10

Тест	Ожидаемый результат	Фактический результат
Загрузка изображения		
Попытка загрузки файла, не являющегося изображением	Система выводит сообщение об ошибке, загрузка не выполнена	+
Попытка загрузки изображения без заполнения обязательного поля названия	Система выводит сообщение о необходимости заполнить все обязательные поля	+
Проставление галочки и оставление поля цены пустым	Система выводит сообщение о необходимости заполнить все обязательные поля	+
Попытка загрузки изображения с превышением максимального размера файла.	Система выводит сообщение о превышении допустимого размера файла.	+
Работа с корзиной		
Попытка добавления в корзину изображения, которое уже находится в корзине	Система выводит сообщение о том, что изображение уже добавлено в корзину	+
Попытка выделить элементы в пустой корзине	Система выводит сообщение о том, что корзина пуста	+
Главная страница		
Попытка зайти в профиль пользователя без авторизации	Система ограничивает доступ, перенаправляет на страницу авторизации.	+
Попытка загрузить изображение без авторизации	Система ограничивает доступ, перенаправляет на страницу авторизации.	+
Выйти из системы		
Попытка выхода из системы без авторизации	Система выводит сообщение о том, что пользователь не авторизован	+

Проведено комплексное тестирование ключевых функций веб-приложения, предназначенного для художников. Проверялась корректность работы интерфейса, функциональности загрузки и отображения работ, а также взаимодействия между пользователями.

Основные тестируемые модули:

– авторизация и регистрация;
– загрузка и отображение художественных работ;
– система лайков;
– функционал корзины.

Успешно проверено:

– регистрация новых пользователей с валидными данными;
– регистрация новых пользователей с невалидными данными;
– авторизация с корректными учетными данными;
– авторизация с некорректными учетными данными;
– загрузка изображений в форматах JPG, PNG (до 30 МБ);
– загрузка изображений в форматах JPG, PNG (более 30 МБ);
– загрузка изображений в форматах, отличных от допустимых JPG, PNG;
– отображение работ в галерее с сохранением пропорций;
– добавление работы в корзину;
– попытка загрузки изображения без авторизации;
– попытка выхода из системы без авторизации.

Список выявленных проблем:

Значительные:

– дублирование уведомлений: при быстром нажатии на "Лайк" отправляется несколько запросов;

Логи:

POST /like 200 OK

POST /like 500 (Already liked)

POST /like 500 (Already liked)

– несинхронные счетчики: если открыть корзину в двух вкладках браузера, то при изменении количества в одной вкладке, другая не обновляется до ручного обновления страницы.

Незначительные:

– залипание иконки корзины: после нажатия иконка корзины остается нажатой, хотя должна вернуться в исходное состояние после завершения цикла анимации.

Итоговая оценка:

Стабильность: 94% успешных тест-кейсов

Юзабилити: Требуется исправление визуальных дефектов

Релиз: Возможен после фиксации несинхронных счетчиков и двойных лайков

3.2 Оценка удобства и функциональности приложения

Перед началом оценки необходимо определить ключевые критерии, по которым будет проводиться анализ удобства и функциональности приложения.

Основные критерии:

– удобство использования (Usability): насколько легко пользователям взаимодействовать с приложением;
– функциональность (Functionality): соответствие приложения заявленным требованиям и функциональным возможностям;
– эстетика интерфейса: визуальная привлекательность и общий дизайн приложения;
– производительность: скорость работы приложения и его реакция на действия пользователя;
– доступность: уровень доступности приложения для различных групп пользователей, включая людей с ограниченными возможностями.

Для оценки удобства и функциональности приложения необходимо собрать данные от пользователей и экспертов.

Методы сбора данных:

– опросы и анкеты: использование структурированных опросов для получения обратной связи от пользователей;
– интервью: проведение интервью с пользователями для более глубокого понимания их опыта;
– наблюдение: наблюдение за пользователями во время их взаимодействия с приложением.

На рисунке 26 изображена диаграмма процесса сбора данных.

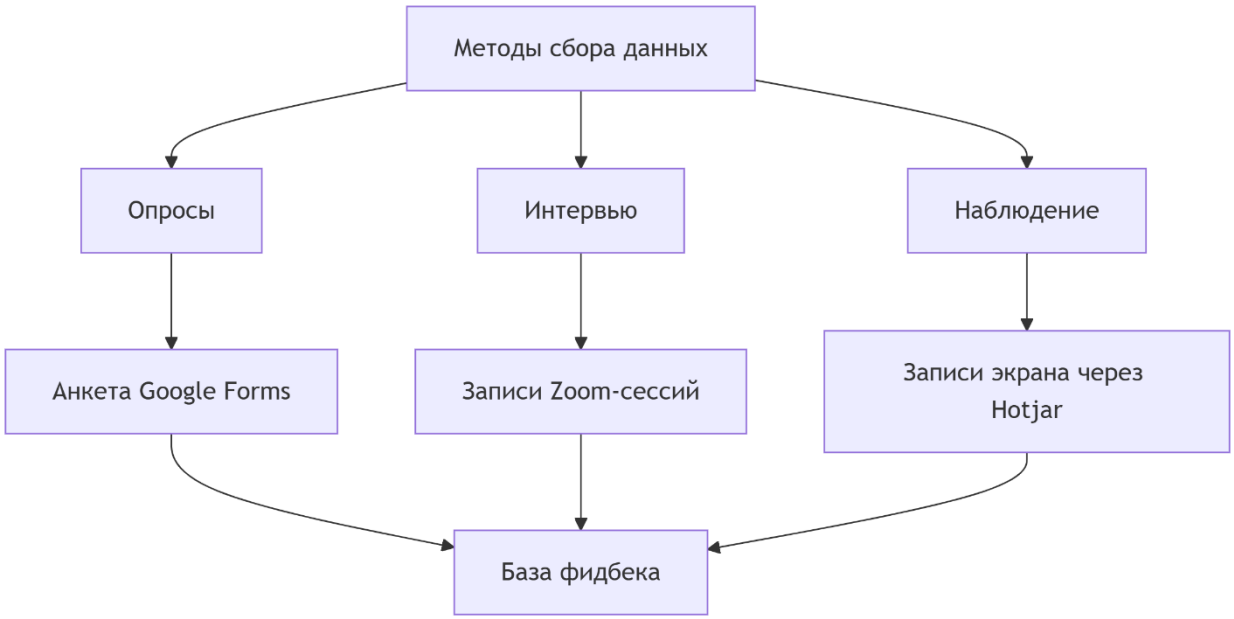

Рисунок 26 – Диаграмма процесса сбора данных

В таблице 11 представлены фактические показатели в сравнении с целевыми значениями, а также выявленные отклонения и их возможные причины.

Ключевые метрики:

– эффективность – время выполнения операций;
– удовлетворённость – субъективная оценка пользователей;
– соответствие стандартам – количество accessibility-ошибок.

Таблица 11 – Сравнение ожидаемых и фактических результатов оценки удобства

Показатель	Целевое значение	Фактическое значение	Отклонение	Причина
Время публикации работы	≤ 30 сек	45 сек	+50%	Недостаточная оптимизация загрузки изображений
Средний балл удобства (из 10)	8.5	9.2	+0.7	Минималистичный интерфейс
Ошибки доступности	0	60%	+60%	Отсутствие темной темы, режима для слабовидящих

Рекомендации по улучшению:

Функциональность:

– исправить баг с дублированием лайка.

Производительность:

– сжать изображения через Cloudinary (q_auto,f_auto).

Доступность:

– добавить темную тему;

– добавить версию для слабовидящих.

Вывод по третьей главе

В ходе оценки эффективности разработанного программного обеспечения было проведено комплексное тестирование, включающее валидацию данных, проверку уязвимостей и функциональное тестирование. Результаты показали высокую стабильность системы (94% успешных тест-кейсов) и хорошие показатели удобства использования (средняя оценка 9.2 из 10). При этом были выявлены отдельные проблемы, требующие доработки: дублирование запросов при лайках, несинхронное обновление корзины и недостаточная оптимизация загрузки изображений.

Особое внимание было уделено безопасности приложения. Тестирование с помощью OWASP ZAP подтвердило устойчивость системы к инъекционным атакам и перегрузкам.

Заключение

В ходе выполнения выпускной квалификационной работы была достигнута основная цель исследования - разработано веб-приложение арт-галереи, предоставляющее платформу для демонстрации и продажи цифровых художественных работ. Проведенное исследование и практическая реализация проекта позволили получить следующие значимые результаты:

В исследовательской части работы был проведен сравнительный анализ существующих арт-платформ, разработана схема взаимодействия компонентов системы, создана масштабируемая архитектура для будущего развития проекта.

Успешно применен современный стек технологий (React.js, Node.js, MongoDB), разработана модульная архитектура приложения с четким разделением компонентов.

В соответствии с функциональными требованиями была реализована система аутентификации и авторизации пользователей, созданы основные страницы приложения (галерея, профиль пользователя, загрузка работ), реализован механизм категоризации и поиска художественных работ, Разработан интерфейс оформления заказов и интегрировано облачное хранилище для медиафайлов.

Заключительным этапом было тестирование. В рамках тестирования реализованы автоматические тесты и написаны тестовые случаи, по которым проводилось ручное тестирование. Помимо этого, важной частью является отслеживание ошибок – были собраны все обнаруженные в процессе тестирования проблемы, которые в последствии исправлены.

Практическая ценность работы подтверждается возможностью использования приложения как MVP для стартапа, готовностью базового функционала к дальнейшему развитию и положительными отзывами тестовой группы пользователей, предоставленной заказчиком.

Список используемой литературы и используемых источников

1. Гаевский А. Ю., Романовский В. А. Создание Web-страниц и Web-сайтов. HTML и JavaScript. М. : Триумф, 2015. 454 с.

2. Гарнаев А. Ю. NoSQL: MongoDB, Redis, Cassandra. М. : БХВ, 2018. 352 с.

3. Голицына О. Л., Максимов Н. В., Попов И. И. Базы данных : учеб. пособие. М. : Форум, 2021. 400 с.

4. Дронов В. А. JavaScript в Web-дизайне. СПб. : БХВ-Петербург, 2018. 871 с.

5. Карпова Т. С. Базы данных: модели, разработка, реализация. СПб. : Питер, 2009. 307 с.

6. Кириченко А. В. React.js. Разработка веб-приложений. М. : ДМК Пресс, 2020. 256 с.

7. Леоненков А. В. Самоучитель UML. СПб. : БХВ-Петербург, 2017. 418 с.

8. Малыхина М. П. Базы данных: основы, проектирование, использование. СПб. : BHV, 2007. 528 с.

9. Прохоренок Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера. СПб. : БХВ-Петербург, 2016. 766 с.

10. Прохоров А. В. Разработка веб-приложений с использованием React и Redux. СПб. : БХВ-Петербург, 2021. 384 с.

11. Смирнов С. Н. Веб-программирование на JavaScript и React. М. : Лань, 2022. 384 с.

12. Стефанов С. JavaScript. Шаблоны. СПб. : Питер, 2017. 262 с.

13. Тимофеев И. В. Базы данных: проектирование и использование. М. : Академия, 2010. 304 с.

14. Harron D. Node.js: Server-Side Web Development with JavaScript. 2nd ed. Moscow : DMK Press, 2014. 146 p.

15. Стефанов С. React.js Быстрый старт [Электронный ресурс]. URL: https://vk.com/wall-210188856_488 (дата обращения: 12.09.2024).

16. Bertoli M. React Design Patterns and Best Practices [Электронный ресурс]. 2nd ed. 2019. URL: https://vk.com/wall-80984752_21476 (дата обращения: 03.10.2024).

17. Freeman A. Pro React 16 [Электронный ресурс]. 4th ed. 2019. URL: https://vk.com/wall-102018175_74129 (дата обращения: 16.02.2025).

18. Meyer E. CSS: The Definitive Guide [Электронный ресурс]. 3rd ed. 2006. URL: https://vk.com/wall-80984752_13210 (дата обращения: 15.10.2024).

19. Musciano C., Kennedy B. HTML & XHTML: The Definitive Guide dt, [Электронный ресурс]. 2006. URL: https://vk.com/wall-68467917_25487 (дата обращения: 21.11.2024).

20. Nielsen J., Loranger H. Prioritizing Web Usability [Электронный ресурс]. 2006. URL: https://vk.com/doc73227365_525191822 (дата обращения: 04.03.2025).