

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

<i>Факультет</i>	<i>Кафедра</i>
вычислительной техники	МОиПЭВМ
Направление подготовки	09.03.02 «Информационные системы и технологии»
Направленность	Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

**Разработка персонального сайта-портфолио
для позиционирования в профессиональной
среде**

Студент	_____	Грошева Полина Александровна
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	Балашова И.Ю.
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролер	_____	Такташкин Д.В.
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 09.06.2025 г № 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, 2025

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
«20» декабря 2024 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Грошева Полина Александровна гр. 21ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Разработка персонального сайта-портфолио для позиционирования в профессиональной среде

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы к.т.н. доцент, доцент кафедры «МО и ПЭВМ» Балашова И.Ю.

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

1) *Назначение: разработка персонального сайта-портфолио для позиционирования в профессиональной среде.*

2) *Среда выполнения: Windows 11.*

Языковые средства разработки: JavaScript, CSS, HTML, SQL, UML.

Инструментальные средства: npm, Visual Studio Code, MySQL, Figma, Visual Paradigm 17.2.

Технологии разработки ООП, многослойная архитектура.

3) *Основные функции приложения:*

а) автоматизация;

б) управление перечнем цен и услуг;

в) расчет стоимости услуг в режиме онлайн;

г) оформление онлайн-заявки на выполнение услуг;

д) хранение и систематизация информации о заявках клиентов;

е) управление и редактирование портфолио работ.

5. Перечень подлежащих разработке вопросов:

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

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2025 – 14.04.2025	
2	Проектирование программных средств	20%	15.04.2025 – 25.04.2025	
3	Реализация программных средств	30%	26.04.2025 – 12.05.2025	
4	Тестирование программных средств	20%	13.05.2025 – 22.05.2025	
5	Оформление пояснительной записки	20%	23.05.2025 – 03.06.2025	

Дата выдачи задания «20» декабря 2024г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

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

Реферат

Пояснительная записка содержит 102 листа, 51 рисунок, 12 таблиц, 21 использованный источник и 6 приложений.

ПЕРСОНАЛЬНЫЙ САЙТ-ПОРТФОЛИО, ВЕБ-ПРИЛОЖЕНИЕ, МОНОЛИТНАЯ АРХИТЕКТУРА, REST API, UML, VISUAL STUDIO CODE, HTML, CSS, JS, БАЗА ДАННЫХ, MYSQL, REACT, NODE.

Объект исследования – веб-приложения, ориентированные на предоставление индивидуальных услуг в профессиональной среде.

Цель выпускной квалификационной работы – разработать персональный сайт-портфолио для позиционирования в профессиональной среде.

Технологии разработки – разработка клиентской части приложения проводилась на языке программирования JavaScript с использованием библиотеки ReactJS в среде разработки Visual Studio Code. Разработка серверной части приложения также производилась на языке программирования JavaScript в аналогичной среде разработки с использованием фреймворка NodeJS. Разработка базы данных проводилась в СУБД MySQL, разработка интерфейса и прототипа веб-приложения проводилась в онлайн-сервисе Figma.

Среда исполнения – Windows 11, веб-браузер.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная часть программных средств для персонального сайта-портфолио. Все требования к разработке приложения соблюдены.

					ПГУ 09.03.02 – 08BP211.09 ПЗ						
Изм.	Лист	№ докум.	Подпись	Дата	Разработка персонального сайта-портфолио для позиционирования в профессиональной среде. Пояснительная записка			Лит.	Лист	Листов	
Разраб.		Грошева П.А.									
Пров.		Балашова И.Ю.								4	102
Реценз.								Группа 21ВИ1			
Н. Контр.		Такташкин Д.В									
Утв.		Козлов А.Ю.									

Содержание

Введение.....	7
1 Анализ предметной области и постановка задачи на разработку персонального сайта-портфолио для позиционирования в профессиональной среде	8
1.1 Исследование предметной области	8
1.2 Обзор персональных сайтов-портфолио для позиционирования в профессиональной среде	9
1.2.1 Персональный сайт графического дизайнера Зябликовой Д.М.....	9
1.2.2 Персональный сайт графического дизайнера Замировского Р.	10
1.2.3 Персональный сайт графического, веб-дизайнера Видинеевой Н. ..	12
1.2.4 Сравнение персональных сайтов-портфолио для позиционирования в профессиональной среде	14
1.3 Анализ функциональных требований	16
1.4 Анализ нефункциональных требований	20
1.5 Планирование разработки и оценка бюджета	21
2 Проектирование программных средств персонального сайта-портфолио	25
2.1 Архитектура приложения.....	25
2.2 Проектирование структур данных	26
2.3 Проектирование пользовательского интерфейса.....	28
2.4 Проектирование классов программных средств для персонального сайта-портфолио.....	29
3 Реализация программных средств для персонального сайта-портфолио	32
3.1 Реализация базы данных	32
3.2 Реализация серверной части приложения	33
3.3 Реализация клиентской части приложения	34
3.4 Реализация интерфейса приложения	36
4 Контроль качества программного обеспечения.....	50
4.1 Обоснование выбора режима тестирования и метрик кода	50

4.2 Тестирование программных средств для персонального сайта	
портфолио	51
4.2.1 Функциональное тестирование	51
4.2.2 Кросс-браузерное тестирование	54
4.3 Оценка качества программного кода	55
4.3.1 Метрика Холстеда.....	55
4.3.2 Цикломатическая сложность	56
Заключение	59
Список использованных источников	61
Приложение А (справочное) Глоссарий	63
Приложение Б (рекомендуемое) Атрибуты сущностей базы данных	66
Приложение В (обязательное) Листинг серверной части приложения.....	70
Приложение Г (обязательное) Листинг клиентской части приложения	80
Приложение Д (рекомендуемое) Результаты функционального тестирования .	93
Приложение Е (рекомендуемое) Результаты кросс-браузерного тестирования	99

Введение

В современных условиях, когда информационно-коммуникационные технологии проникли во многие сферы деятельности, все большее значение для бизнеса приобретает эффективное взаимодействие с целевой аудиторией в онлайн-среде. В частности, многие специалисты-фрилансеры продвигают свои услуги и формируют клиентскую базу с помощью сети Интернет [1]. Особенно выгодно использование онлайн коммуникаций для творческих специалистов. Одним из наиболее востребованных инструментов является персональный сайт-портфолио, который не только демонстрирует творческие навыки и профессиональные достижения, но и служит платформой для взаимодействия с потенциальными клиентами и работодателями.

Разработка персонального сайта-портфолио для позиционирования в профессиональной среде является актуальной темой на сегодняшний день, потому что такие сайты помогают выделиться среди конкурентов, автоматизировать процессы приема заказов и расчета стоимости.

Целью выпускной квалификационной работы является разработка персонального сайта-портфолио для позиционирования в профессиональной среде, предназначенного для представления портфолио графического дизайнера, организации приема заказов, автоматизации расчета стоимости услуг и управления потоком заявок.

Исходя из цели данной работы, были поставлены следующие задачи:

- проанализировать предметную область и аналоги похожего программного обеспечения, поставить задачу на разработку;
- спроектировать программные средства, а также структуры данных разрабатываемого приложения;
- разработать модули программных средств;
- проконтролировать качество программного обеспечения.

Решение всех поставленных задач позволит создать эффективный инструмент для представления портфолио и взаимосвязи с клиентами.

1 Анализ предметной области и постановка задачи на разработку персонального сайта-портфолио для позиционирования в профессиональной среде

1.1 Исследование предметной области

Портфолио – это собрание достижений специалиста, подтверждающее его навыки и умения, например, коллекции фотографий, образцы иллюстраций художника, грамоты и сертификаты студента.

Портфолио может формироваться на разных носителях: бумажный вариант, электронный на флешке, сайт-портфолио. Наиболее эффективным носителем сведений о человеке является сайт. Основная цель сайта-портфолио – это привлечение внимания и продвижение услуг специалиста в определенной сфере деятельности. Сайт портфолио должен выглядеть не только выразительно, но и нести в себе определенный посыл, отражать всю важную информацию о специалисте как о профессионале своего дела. Грамотно созданный сайт помогает специалисту получать заказы на свои товары или услуги, завоевывать репутацию в определенной им нише и решать собственные поставленные задачи. Деятельность человека многообразна, поэтому удобнее всего рассматривать ее в контексте разных сфер жизни и направлений.

Персональные сайты-портфолио разрабатываются в основном для различных творческих специалистов, таких как дизайнеры, фотографы, иллюстраторы. Однако, в последнее время личные сайты хорошо развиваются и в других сферах: образование, психология, физическая культура, ведение менторства, репетиторства и наставничества. Эти платформы позволяют специалистам не только продемонстрировать свои навыки и достижения, расширить клиентскую базу, но и установить более тесный контакт с аудиторией. В данной же работе будет разработан персональный сайт-портфолио для графического дизайнера.

Данные веб-приложения обладают рядом особенностей, отличающих их от других видов интернет-ресурсов:

- насыщенное визуальное оформление;
- интерактивное взаимодействие с пользователем через фильтры категорий, анимации при наведении и прочее;
- простая навигация с логичной структурой сайта по разделам «Портфолио», «О себе», «Контакты» и т. д.;
- возможность связи через социальные сети, мессенджеры, формы обратной связи.

Обязательным условием при разработке сайтов-портфолио является предоставление владельцу сайта возможности самостоятельно, быстро и легко добавлять новые работы и редактировать существующие.

1.2 Обзор персональных сайтов-портфолио для позиционирования в профессиональной среде

1.2.1 Персональный сайт графического дизайнера Зябликовой Д.М.

Сайт дизайнера Дарьи Зябликовой по своей структуре является хорошим шаблоном, имеет достаточно понятный и интуитивный интерфейс. Раздел портфолио представляет собой несколько блоков страницы, каждый из которых демонстрирует одну из категорий работ, например, логотипы или сайты. В каждом блоке показано по три работы, для того чтобы увидеть больше работ из выбранной категории, можно нажать на кнопку «Показать еще». На рисунке 1 изображен блок «Логотипы и фирменные стили». У каждого проекта есть название и краткое описание выполненной работы, а также обложка проекта. Если пользователя заинтересовал конкретный проект, то более подробную информацию он может увидеть, нажав на кнопку «Посмотреть проект».

Среди достоинств данного веб-приложения можно выделить наличие четкой структуры и навигации, возможность оставить заявку на заказ услуг, просмотр достижений дизайнера, блок отзывов клиентов, новостной раздел.

ПОРТФОЛИО

ЛОГОТИПЫ И ФИРМЕННЫЕ СТИЛИ

БОЛЬШЕ ФОТО в каждом проекте



ENDEMICA | WOOD & FURNITURE

Нейминг и разработка логотипа

[ПОСМОТРЕТЬ ПРОЕКТ](#)



PAVLOVA DRESS | ПЛАТЬЯ НА ЗАКАЗ

Разработка логотипа и оформление Instagram

[ПОСМОТРЕТЬ ПРОЕКТ](#)



АНДРЕЙ МОРОЗОВ | МАССАЖИСТ

Разработка фирменного стиля

[ПОСМОТРЕТЬ ПРОЕКТ](#)



Рисунок 1 – Раздел портфолио на сайте Зябликовой Д.М.

Главным недостатком, с точки зрения пользователя, можно считать отсутствие информации о дизайнере и ее фотографий, что ухудшает восприятие, а также в разделе портфолио некоторые файлы загружены в плохом качестве, присутствуют нерабочие ссылки, что снижает функциональность сайта.

1.2.2 Персональный сайт графического дизайнера Замировского Р.

Персональный сайт Замировского Романа представляет собой яркий визуальный стиль, что повышает интерес и привлекает потенциальных клиентов. Первый блок главной страницы сайта представлен на рисунке 2.

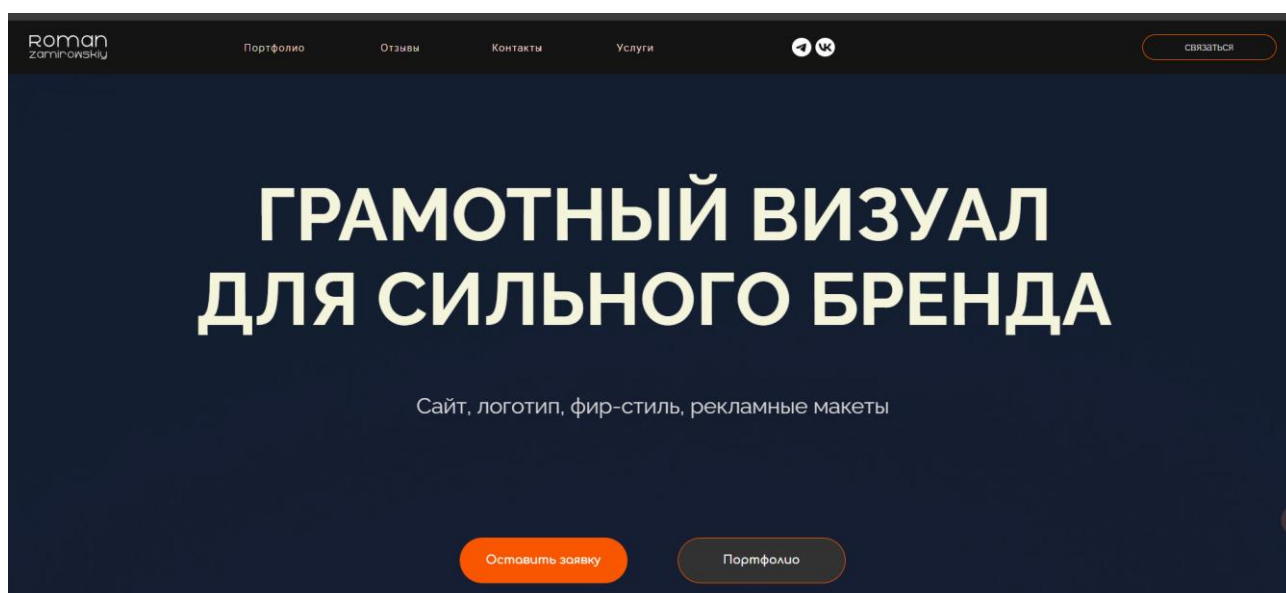


Рисунок 2 – Первый блок главной страницы сайта Замировского Р.

Среди достоинств этого веб-приложения можно выделить его интерактивность при взаимодействии с пользователем с помощью видео-контента, взаимодействия с блоком отзывов, составления слов из созданного дизайнером шрифта (рисунок 3, 4).

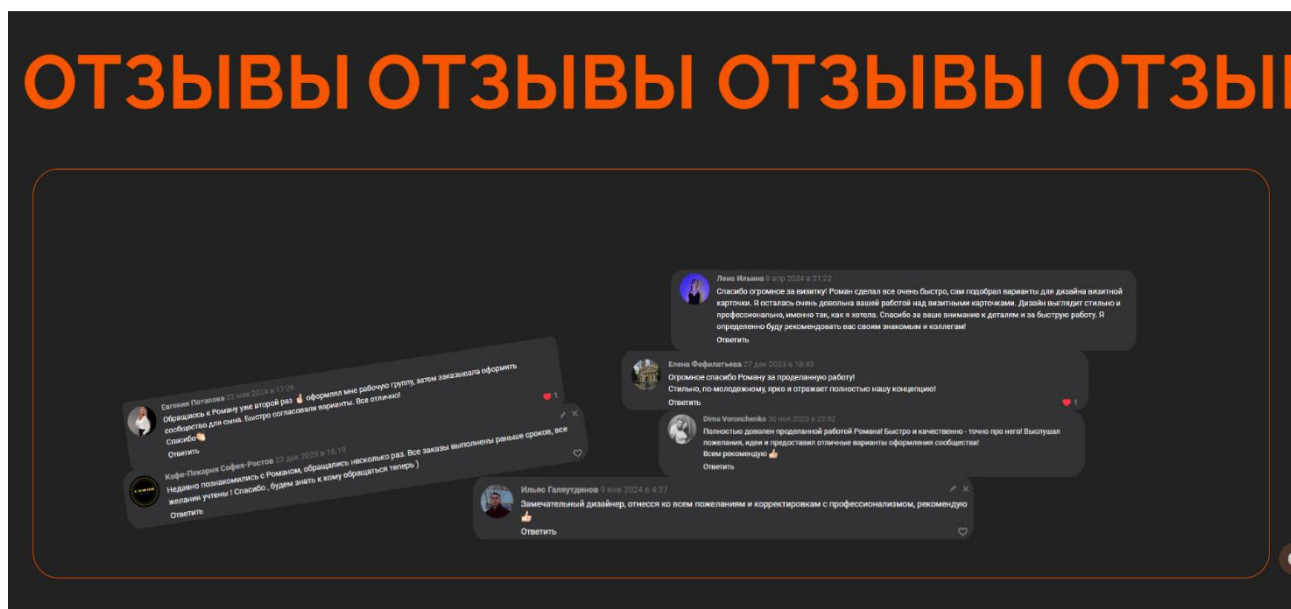


Рисунок 3 – Блок отзывов на сайте Замировского Р.

Сайт дизайнера предоставляет возможность оставить заявку на проект, имеет быстрый и удобный просмотр портфолио работ через меню сайта и отдельную кнопку на главной странице.

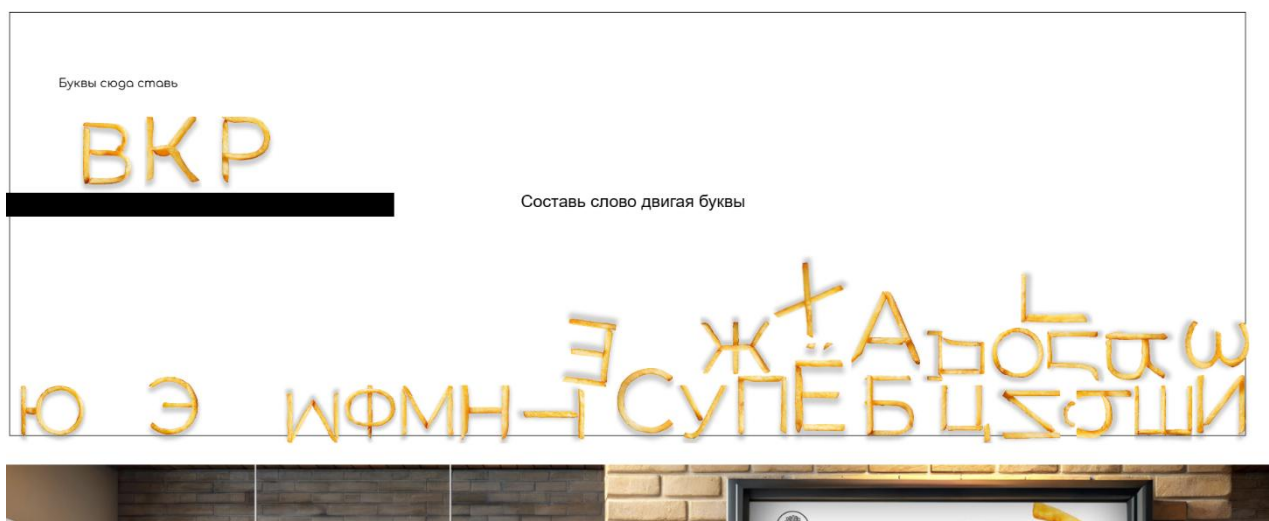


Рисунок 4 – Интерактивное взаимодействие с дизайнерским шрифтом

Недостатками этого веб-приложения являются отсутствие стоимости работ за предлагаемые услуги и отсутствие отдельного блока с контактной информацией, что затрудняет связь между дизайнером и клиентами.

1.2.3 Персональный сайт графического, веб-дизайнера Видинеевой Н.

Самым неординарным и функциональным является персональный сайт дизайнера Видинеевой Наталии, первый блок главной страницы которого представлен на рисунке 5.

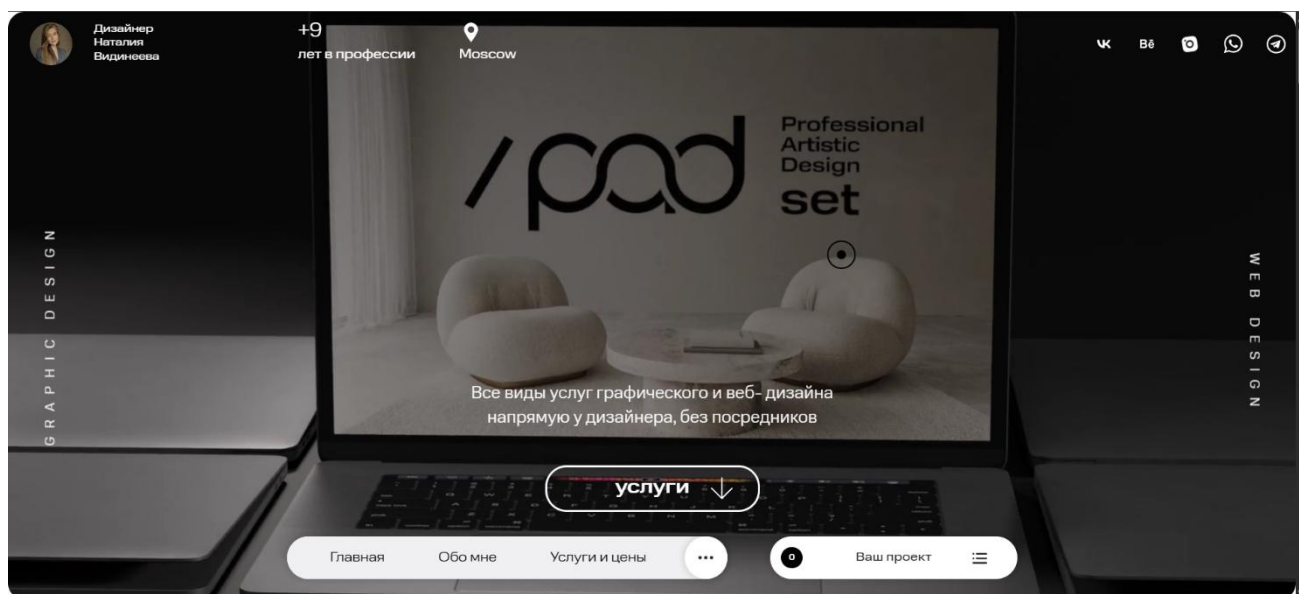


Рисунок 5 – Первый блок главной страницы сайта Видинеевой Н.

Главным достоинством веб-приложения являются функции добавления клиентом в корзину нужных услуг и расчета предварительной стоимости работ онлайн, отправляя заявку (рисунок 6).

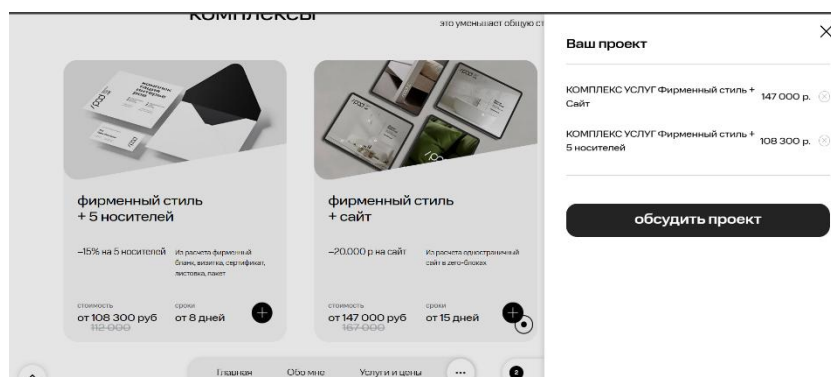


Рисунок 6 – Добавление услуг в корзину

Кроме того, на сайте реализовано всплывающее окно обучения использования данной функции (рисунок 7). Сайт использует файлы cookie, чтобы сохранять данные клиента. Есть функция смены раскладки языка с русского на английский и страница личного блога со статьями дизайнера.

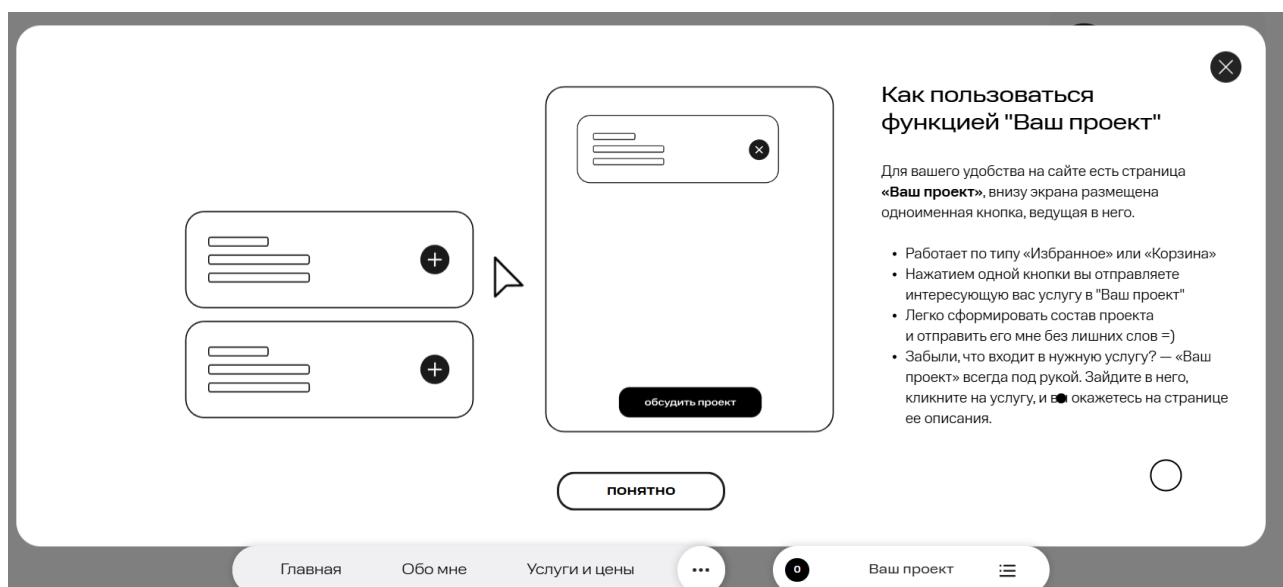


Рисунок 7 – Всплывающее окно обучения использования функции добавления услуг

Основным недостатком является отсутствие блока отзывов на сайте, также нестандартные меню и структура веб-приложения могут затруднять навигацию пользователей.

1.2.4 Сравнение персональных сайтов-портфолио для позиционирования в профессиональной среде

Проведем сравнительный анализ рассмотренных выше сайтов-портфолио Дарьи Зябликовой, Замировского Романа, Наталии Виденеевой, по ключевым функциям.

Для оценки веб-приложений будет использоваться трехбалльная шкала соответствия. В таблице 1 представлены характеристики оценки.

Таблица 1 – Шкала баллов

Балл	Характеристика оценки
0	функция (качество) полностью отсутствует
1	существует частичная реализация функции (качества)
2	функция (качество) полностью реализована

Результаты сравнения персональных сайтов-портфолио для позиционирования в профессиональной среде представлены в таблице 2.

Таблица 2 – Сравнение персональных сайтов аналогичного назначения

Критерий \ Приложение	Зябликова Д.	Замировский Р.	Виденеева Н.
1. Возможность оставлять заявку на услуги	1	1	2
2. Возможность связаться с дизайнером	2	2	2
3. Просмотр стоимости услуг дизайнера	1	0	2
4. Возможность интерактивного взаимодействия на сайте	1	2	1
5. Интуитивный интерфейс	2	2	2
6. Удобная навигация	2	1	1
7. Информация о дизайнере	0	1	2
8. Яркий современный визуальный стиль сайта	1	2	2
9. Быстрый и удобный просмотр портфолио работ	0	1	2

Продолжение таблицы 2

Критерий \ Приложение	Зябликова Д.	Замировский Р.	Видинеева Н.
10. Калькулятор стоимости услуг	0	0	2
ИТОГО	10	12	18

Для наглядности, полученные результаты отображены на диаграмме, приведенной на рисунке 8.

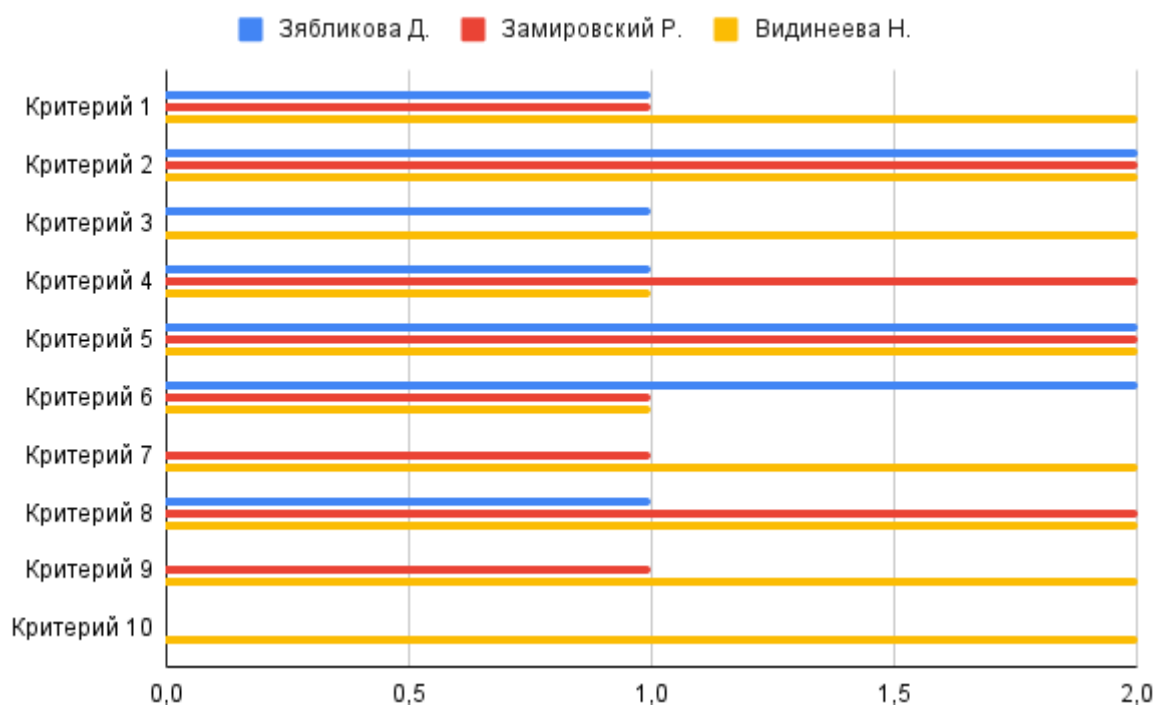


Рисунок 8 – Результаты сравнения приложений-аналогов

Таким образом, в результате обзора персональных сайтов-портфолио для позиционирования в профессиональной среде можно сделать вывод, что ни одно из рассмотренных решений не соответствует объявленным критериям в полном объеме. Среди всех приложений-аналогов сайт Видинеевой Наталии выделяется большим количеством реализованных функций, но данное приложение не обеспечивает в полной мере пользователям возможность интерактивного взаимодействия, а также не предоставляет в полном объеме удобную навигацию по сайту.

Как следствие данного обзора, было принято решение разработать собственный персональный сайт-портфолио, в котором будет реализована

простая и удобная навигация и интерактивный взаимодействием с пользователем, что сделает его более сбалансированным и комфортным веб-приложением для позиционирования в профессиональной среде.

1.3 Анализ функциональных требований

В рамках персонального сайта-портфолио были выделены два объекта пользования сайта – это администратор (конкретно в данной работе – графический дизайнер) и гость портала (потенциальный клиент).

Функциональные возможности приложения можно представить с помощью диаграммы вариантов использования UML [2].

На рисунке 9 представлено наследование прав доступа пользователями.

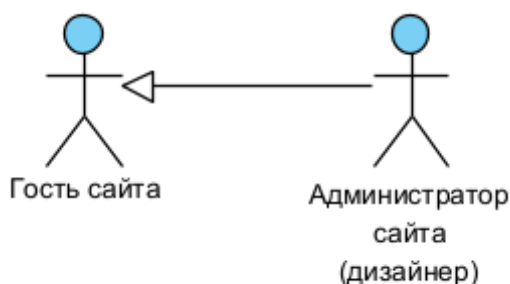


Рисунок 9 – Диаграмма вариантов использования пользователей системы

Администратор сайта обладает следующим набором функциональных требований:

- авторизация;
- редактирование раздела портфолио;
- редактирование работ в портфолио;
- удаление категории в разделе портфолио;
- добавление категории в разделе портфолио;
- удаление работы из портфолио;
- изменение данных о работе в портфолио;
- добавление работы в портфолио;
- просмотр заявок на оказание услуг;

- изменение статуса заявки на «не начато» / «в работе» / «готово»;
- фильтрация и сортировка заявок;
- удалить заявку;
- редактирование данных об услуге;
- удаление услуги.

На рисунке 10 представлена диаграмма вариантов использования для администратора сайта.

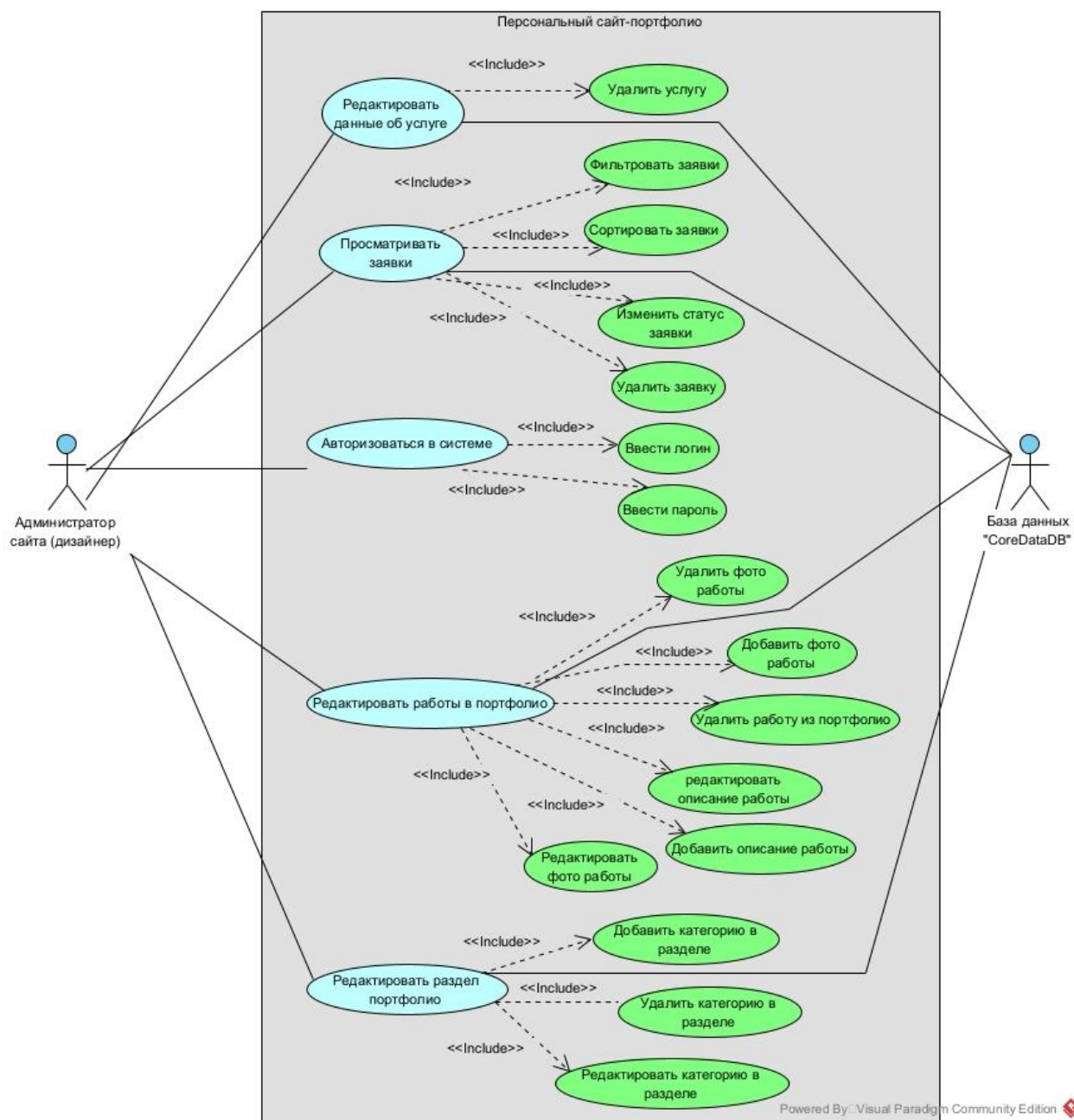


Рисунок 10 – Диаграмма вариантов использования администратора сайта

Гость сайта имеет достаточно простой функционал:

- просмотр страниц сайта;
- создание заявки на услуги;
- расчет стоимости нужных услуг;
- переход к профилям дизайнера в социальных сетях и мессенджерах.

Диаграмма вариантов использования для гостя сайта представлена на рисунке 11.

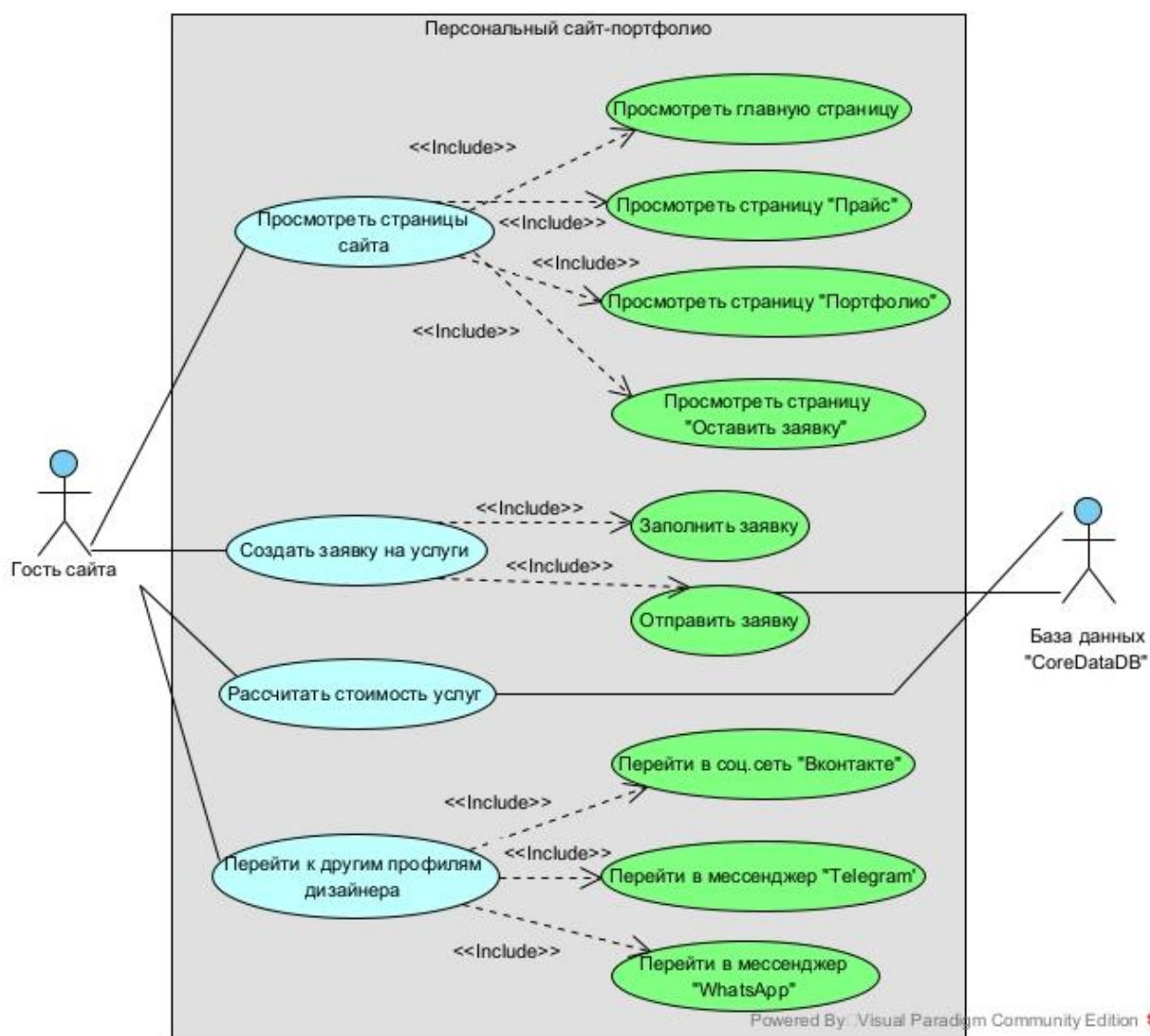


Рисунок 11 – Диаграмма вариантов использования гостя сайта

Ниже приведены спецификации наиболее важных бизнес-процессов.

Сценарий варианта использования «Авторизоваться в системе» представлен на рисунке 12.

Прецедент: Авторизоваться в системе
ID: 1
Краткое описание: Администратор авторизуется в приложении
Главные актеры: Администратор сайта
Второстепенные актеры: Нет
Предусловие: 1. Администратор открыл приложение 2. Администратор не был авторизован
Основной поток: 1. Администратор переходит в окно входа в приложение 2. Администратор вводит для входа существующий логин и пароль 3. Администратор получает доступ к функционалу приложения
Постусловия: 1. Администратор авторизовался в приложении 2. Администратору доступен функционал приложения

Рисунок 12 – Сценарий варианта использования «Авторизоваться в системе»

Сценарий варианта использования «Редактировать работы в портфолио» изображен на рисунке 13.

Прецедент: Редактировать работы в портфолио
ID: 2
Краткое описание: Администратор изменяет свои работы в портфолио
Главные актеры: Администратор сайта
Второстепенные актеры: База данных «CoreDataDB»
Предусловие: 1. Администратор авторизован 2. Администратор открыл нужную категорию в разделе портфолио
Основной поток: 1. Администратор добавляет новую работу в портфолио 2. Администратор добавляет фото в новую работу 3. Администратор добавляет описание работы Альтернативные потоки: 1.а Администратор редактирует фото в существующей работе 1.б Администратор редактирует описание в существующей работе 1.в Администратор удаляет фото в существующей работе 1.г Администратор удаляет описание в существующей работе 1.д Администратор удаляет работу из портфолио
Постусловия: 1. Администратор отредактировал работы в портфолио

Рисунок 13 – Сценарий варианта использования «Редактировать работы в

портфолио»

Сценарий варианта использования «Просматривать заявки» представлен на рисунке 14.

Прецедент: Просматривать заявки
ID: 3
Краткое описание: Администратор просматривает старые и недавно добавленные заявки клиентов
Главные актеры: Администратор сайта
Второстепенные актеры: База данных «CoreDataDB»
Предусловие: 1. Администратор открыл приложение 2. Администратор авторизован
Основной поток: 1. Администратор переходит на страницу заявок 2. Администратор просматривает заявки Альтернативные потоки: 2.а Администратор фильтрует заявки 2.б Администратор сортирует заявки 2.в Администратор меняет статус заявки 2.г Администратор удаляет заявку
Постусловия: 1. Администратор просмотрел заявки

Рисунок 14 – Сценарий варианта использования «Просматривать заявки»

В результате анализа функциональных требований были выделены необходимые варианты использования программы, для ее корректной работоспособности. Были представлены некоторые спецификации вариантов использования, позволяющие отобразить последовательность действий пользователя и системы.

1.4 Анализ нефункциональных требований

Разработка базы данных будет выполнена с применением MySQL – свободной реляционной СУБД.

Клиентская часть приложения будет написана на языке JS с использованием библиотеки ReactJS в среде разработки Visual Studio Code. Для сборки модулей клиентского приложения в единый файл будет использован сборщик Webpack.

Серверная часть приложения будет реализована с использованием монолитной архитектуры. Разработка архитектуры будет происходить по технологии REST API. Серверная часть приложения также будет написана на языке программирования JS и использован фреймворк Node.JS.

Выбор MySQL как базы данных обеспечивает согласованность и скорость выполнения SQL-запросов. Монолитная архитектура позволяет упростить интеграцию компонентов системы, облегчает ее обслуживание и целостность приложения в рамках единой кодовой базы. Взаимодействие серверной и клиентской части приложения осуществляется через REST API [4].

Диаграммы создаются с использованием языка UML 2.0 в среде Visual Paradigm – бесплатном онлайн и офлайн инструменте, для моделирования различных графических представлений, например, диаграмм классов, последовательности, вариантов использования, развертывания и других схем [5].

Разрабатываемое веб-приложение должно быть оптимизировано для работы в современных браузерах, таких как Google Chrome (версии 80 и выше), Mozilla Firefox (версии 75 и выше), Apple Safari (версии 13 и выше), Microsoft Edge (версии 80 и выше).

1.5 Планирование разработки и оценка бюджета

Очень важным этапом при разработке любого проекта является его планирование и оценка бюджета. Для управления проектом можно выделить два метода: гибкий «Agile» и классический «Waterfall». Исходя из метода начинается планирование разработки [6].

В рамках выпускной квалификационной работы был выбран классический метод управления, в котором работа ведется последовательно, т. е. только при завершении одного этапа начинается следующий. Для того чтобы отслеживать

все этапы разработки и определить финансовые и временные затраты, была выбрана программа Project Libre, бесплатный и близкий по интерфейсу аналог Microsoft Office Project 2007.

В начале работы был создан файл (новый проект), указаны его название, дата начала работы и данные менеджера по управлению проектом. После чего в проект вносятся все этапы разработки, заполняются необходимые ресурсы. Под ресурсами подразумевается все что участвует в создании проекта: сотрудники, оборудование, материалы и финансы [7].

При планировании разработки было выделено 4 этапа и 18 задач, представленных на рисунке 15. За дату начала работ взято 8 апреля 2025 года, а окончание работ по проекту запланировано на 4 июня 2025 года.

Название	Продолжитель...	Начало	Окончание
Начало проекта	0 дней	08.04.2025, 08:00	08.04.2025, 08:00
Анализ предметной области и постано	9 дней	09.04.2025, 08:00	21.04.2025, 17:00
Анализ предметной области	0,25 дней	09.04.2025, 08:00	09.04.2025, 10:00
Выбор приложений-аналогов	0,375 дней	14.04.2025, 08:00	14.04.2025, 11:00
Обзор приложений-аналогов	0,75 дней	15.04.2025, 08:00	15.04.2025, 15:00
Сравнительный анализ приложений-аналог	0,5 дней	17.04.2025, 08:00	17.04.2025, 13:00
Результаты сравнительного анализа	0,062 дней	17.04.2025, 08:00	17.04.2025, 08:30
Анализ функциональных требований	1,5 дней	18.04.2025, 08:00	21.04.2025, 13:00
Анализ нефункциональных требований	1,5 дней	21.04.2025, 08:00	21.04.2025, 17:00
Проектирование структур данных	7,167 дней	21.04.2025, 08:00	30.04.2025, 09:20
Проектирование архитектуры приложения	5 дней	21.04.2025, 08:00	25.04.2025, 17:00
Проектирование структур данных	2,667 дней	24.04.2025, 08:00	24.04.2025, 14:20
Проектирование пользовательского интер	1,5 дней	24.04.2025, 14:20	28.04.2025, 09:20
Проектирование классов программных сре	2 дней	28.04.2025, 09:20	30.04.2025, 09:20
Реализация веб-приложения	11,333 дней	03.05.2025, 08:00	20.05.2025, 10:40
Реализация базы данных	2 дней	03.05.2025, 08:00	06.05.2025, 17:00
Реализация серверной части приложения	5 дней	07.05.2025, 08:00	13.05.2025, 17:00
Реализация клиентской части приложения	2,667 дней	14.05.2025, 08:00	16.05.2025, 14:20
Реализация пользовательского интерфейса	1,667 дней	16.05.2025, 14:20	20.05.2025, 10:40
Контроль качества программного обес	7,375 дней	25.05.2025, 08:00	04.06.2025, 11:00
Определение режима тестирования	1 день	25.05.2025, 08:00	26.05.2025, 17:00
Тестирование программных средств прило	4,5 дней	27.05.2025, 08:00	02.06.2025, 13:00
Оценка качества программного кода	0,625 дней	03.06.2025, 15:00	04.06.2025, 11:00
Конец проекта	0 дней	04.06.2025, 11:00	04.06.2025, 11:00

Рисунок 15 – Сведения об этапах разработки приложения

Для наглядности работы была сгенерирована диаграмма Ганта, которая представлена на рисунке 16 [8].

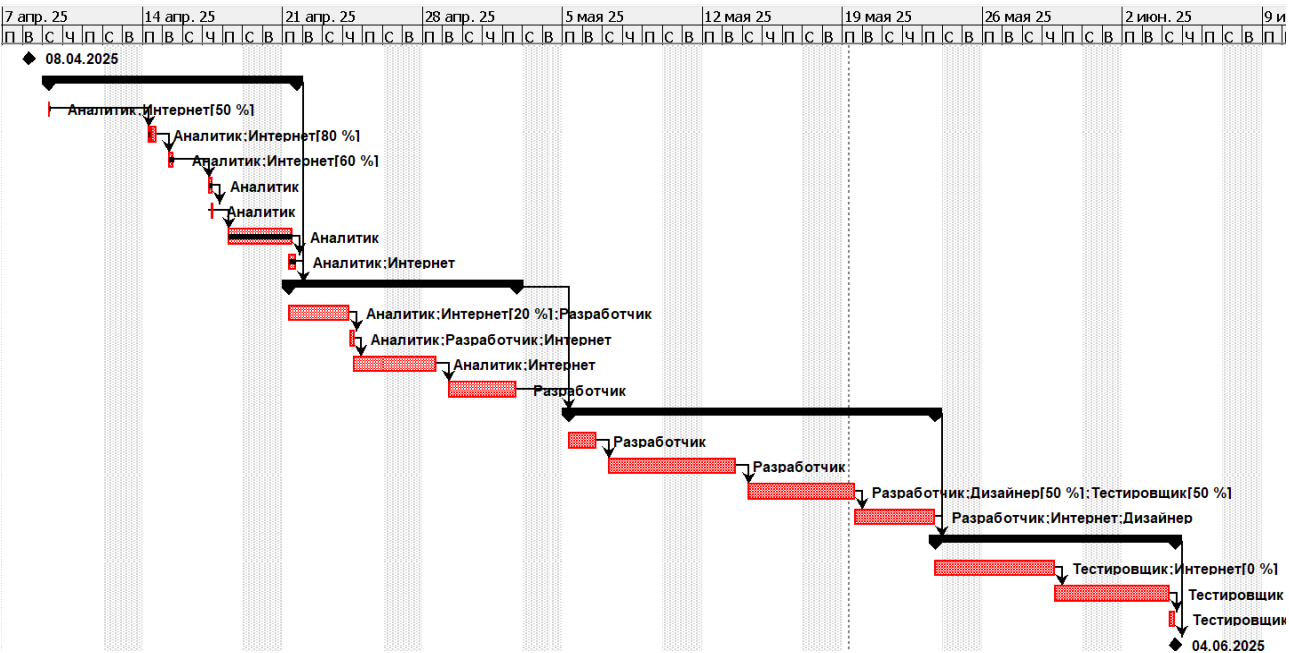


Рисунок 16 – Диаграмма Ганта по этапам разработки приложения

Для того чтобы оценить финансовые затраты на разработку, требуется определить ресурсы и назначить стоимость работ (рисунок 17).

ИД	Название	Тип	Стандартная ставка	Начислять	Основной календарь
1	Аналитик	Работа	400,00 Р/час	Пропорционально	Пятидневка
2	Разработчик	Работа	400,00 Р/час	Пропорционально	Пятидневка
3	Дизайнер	Работа	250,00 Р/час	Пропорционально	Пятидневка
4	Тестировщик	Работа	350,00 Р/час	Пропорционально	Пятидневка
5	Интернет	Работа	0,00 Р/час	Пропорционально	Пятидневка

Рисунок 17 – Трудовые ресурсы и их стоимость

На основании выше перечисленных данных можно подвести итог, что суммарное время трудовых ресурсов составляет 279 часов с учетом пятидневной рабочей недели.

Для каждого этапа работы рассчитывается стоимость, как часы работы одного специалиста умноженные на количество специалистов и почасовую ставку. Затем стоимость всех этапов складывается и получается итоговая стоимость по проекту. Оценка бюджета проекта представлена на рисунке 18.

Название	Стоимость
Анализ предметной области и постановка задачи	30200,00 Р
Анализ предметной области	800,00 Р
Выбор приложений-аналогов	1200,00 Р
Обзор приложений-аналогов	2400,00 Р
Сравнительный анализ приложений-аналогов	1600,00 Р
Результаты сравнительного анализа	200,00 Р
Анализ функциональных требований	4800,00 Р
Анализ нефункциональных требований	3200,00 Р
Проектирование структур данных	33333,33 Р
Проектирование архитектуры приложения	6400,00 Р
Проектирование структур данных	4266,67 Р
Проектирование пользовательского интерфейса	4800,00 Р
Проектирование классов программных средств	6400,00 Р
Реализация веб-приложения	43830,56 Р
Реализация базы данных	6400,00 Р
Реализация серверной части приложения	16000,00 Р
Реализация клиентской части приложения	10666,67 Р
Реализация пользовательского интерфейса	8666,67 Р
Контроль качества программного обеспечения	18200,00 Р
Определение режима тестирования	1050,00 Р
Тестирование программных средств приложения	12600,00 Р
Оценка качества программного кода	1750,00 Р
Конец проекта	0,00 Р
	Итого: 125 563,89 руб.

Рисунок 18 – Оценка бюджета

Таким образом, в результате оценки бюджета итоговая сумма по проекту составляет 125 563 рубля и 89 копеек.

2 Проектирование программных средств персонального сайта-портфолио

2.1 Архитектура приложения

Монолитная архитектура была выбрана исходя из требований к простоте развертывания и единому стеку технологий. В рамках выпускной квалификационной работы разработка единого приложения позволяет сократить накладные расходы на организацию взаимодействия между сервисами, уменьшить сложность конфигурации среды исполнения и ускорить процесс интеграции компонентов. Пользовательское приложение отображает информацию и выступает главным интерфейсом для ввода команд, которые затем передаются на сервер через REST API [9].

Для наглядного представления структуры приложения была построена UML-диаграмма компонентов, находящаяся на рисунке 19.

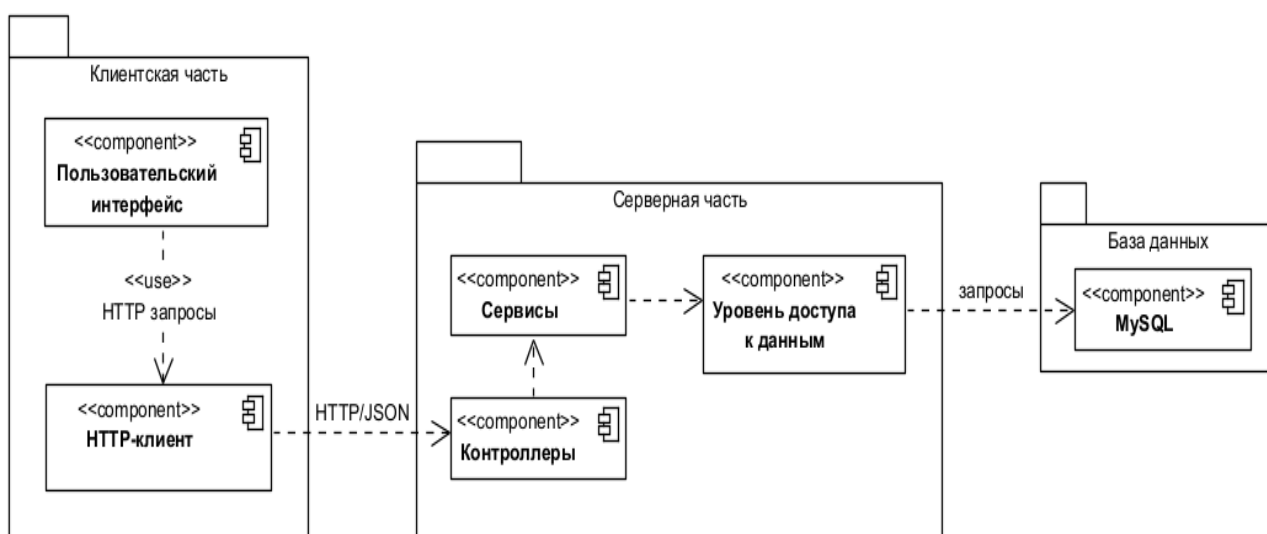


Рисунок 19 – Диаграмма компонентов структуры персонального сайта-портфолио

На рисунке 20 изображена диаграмма развертывания, которая показывает архитектуру персонального сайта-портфолио для позиционирования в профессиональной среде [10].

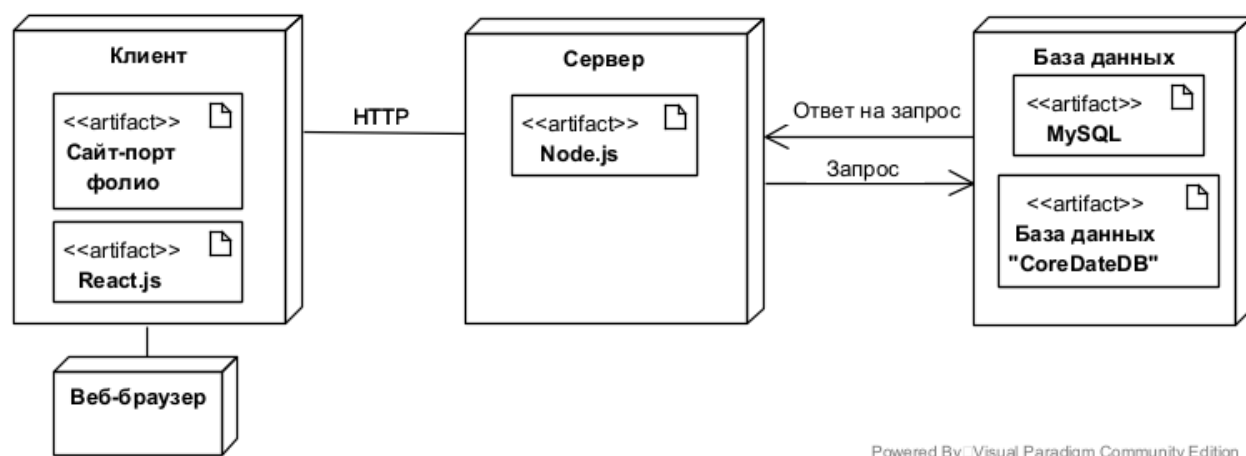


Рисунок 20 – Диаграмма развертывания архитектуры персонального сайта-портфолио

Серверная часть приложения является промежуточным звеном между запрашиваемыми ресурсами и базой данных.

2.2 Проектирование структур данных

Концептуальная модель – это модель, являющаяся абстрактным представлением данных. Она отражает основные сущности, их свойства и взаимосвязи без технических деталей [11].

Для того чтобы составить концептуальную модель данных, нужно определить все необходимые сущности базы данных. В данной работе было выделено четыре сущности. Описание этих сущностей предметной области представлено в таблице 3.

Таблица 3 – Сущности БД

Название сущности	Идентификатор сущности	Назначение сущности
Заявка	requests	Содержит сведения о заявке клиента на услуги
Услуга	services	Содержит сведения об предлагаемых услугах
Категория работы	categoryWorks	Содержит сведения о виде работы
Работа	works	Содержит информацию о конкретной работе из портфолио

Графическое представление концептуальной модели представлено на рисунке 21.

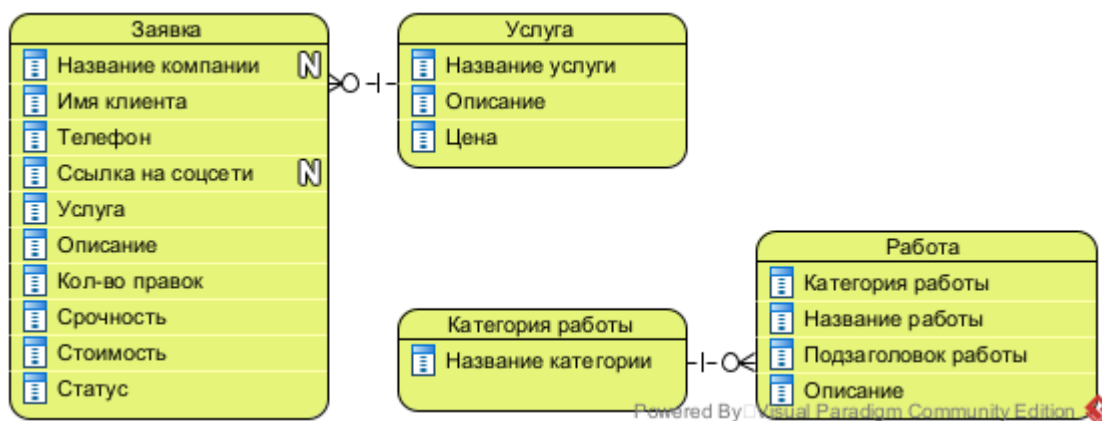


Рисунок 21 – Концептуальная модель базы данных предметной области
«Персональный сайт-портфолио»

Описание связей между сущностями представлено в таблице 4.

Таблица 4 – Описание связей между сущностями

Наименование связи	Описание	Мощность
Услуга-Заявка	В одной заявке может быть выбрано несколько услуг	1:M
Категория работы-Работа	В одной категории может быть представлено несколько работ	1:M

Наборы атрибутов для сущности базы данных предметной области и ее характеристики приведены в таблицах 1–4 приложения А.

На основе концептуальной модели, описания связей между сущностями и их атрибутами была спроектирована логическая модель данных, представленная ниже на рисунке 22.

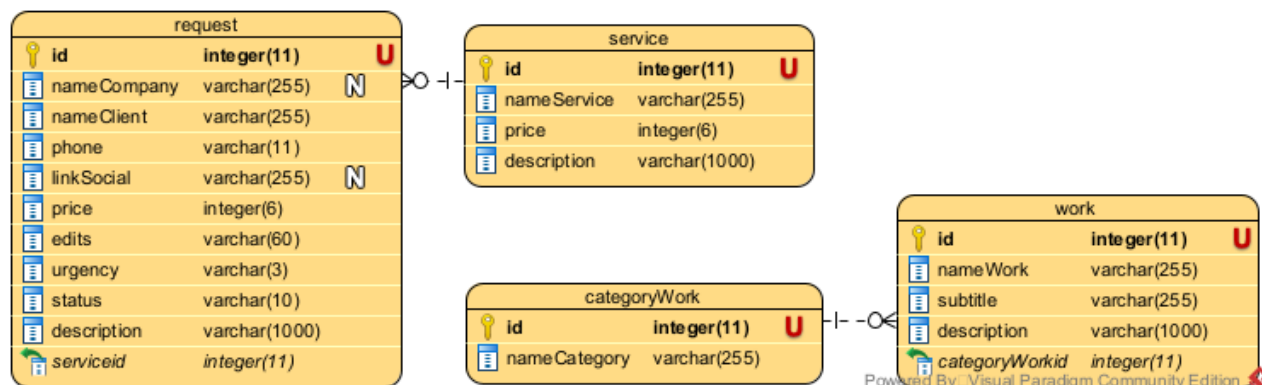


Рисунок 22 – Логическая модель базы данных предметной области
«Персональный сайт-портфолио»

Представление концептуальной и логической моделей данных было построено в среде Visual Paradigm 17.2 в нотации IDEF1X.

Таким образом, на основе логической модели будет построена физическая модель данных, которая нужна для реализации базы данных.

2.3 Проектирование пользовательского интерфейса

Цель проектирования пользовательского интерфейса – создать удобное и простое в навигации пользователю веб-приложение.

Проектируемый интерфейс должен содержать следующие страницы:

- «Авторизация»;
- «Главная»;
- «Портфолио»;
- «Прайс-лист услуг»;
- «Заявки»;
- «Оставить заявку».

Страница авторизации нужна для входа в систему администратору. Перейти на нее возможно, указав в поисковой строке следующий путь: <http://localhost:3000/login>.

На главной странице должны быть: раздел «Обо мне» с краткой информацией о владельце сайта и его навыками, раздел «Контакты» для возможности связи, раздел «Этапы работы» для ознакомления потенциальных клиентов, раздел с переходом на страницу «Портфолио». На разделы «Обо мне» и «Контакты» в меню сайта есть отдельные ссылки для быстрого перехода.

На странице «Портфолио» должны быть представлены вкладки с категориями работ, например, «Инфографика», «Полиграфия», «Фирменный стиль». Внутри каждой вкладки находятся примеры работ с текстовым описанием и изображениями.

Для прозрачности ценообразования должна быть создана страница «Прайс-лист услуг» с названиями, описаниями и ценами.

Страница «Заявки» доступна только администратору сайта, на ней отображены все созданные заявки клиентов на услуги, сортировка и фильтры для удобной навигации. В зависимости от стоимости работ в заявке для большей наглядности и привлечения внимания пользователя сами заявки имеют цветовую градацию, где зеленый – высокая прибыль, желтый – средняя прибыль, серый – низкая прибыль.

«Оставить заявку» – это страница представляет собой заполнение заявки с указанием необходимых данных заказчика и услугами, которые ему требуется оказать. Здесь же рассчитывается приблизительная стоимость работ.

Для администратора веб-приложения должен быть реализован функционал редактирования на страницах «Портфолио», «Прайс-лист услуг» и «Заявки», а также доступны все возможности гостя (потенциального клиента).

Каждый элемент интерфейса будет выполнен с помощью компонентов библиотеки ReactJS, которые управляют своим состоянием и возможностями. Использование компонентной архитектуры облегчает масштабирование приложения и повторное применение кода [12].

2.4 Проектирование классов программных средств для персонального сайта-портфолио

Диаграмма классов позволяет представить внутреннюю структуру данных программы. Все реальные объекты, с которыми должен работать программист, представляются в виде классов и их взаимосвязей. Эти объекты называются сущностями, каждая имеет свое уникальное имя. Внутри сущности могут быть различные атрибуты и методы [13].

В ходе проектирования серверной части приложения, была разработана диаграмма классов согласно шаблону проектирования «Model, View, Controller» (MVC), который позволяет изменять каждый компонент независимо друг от друга для простой разработки и поддержки веб-приложений. Диграмма представлена на рисунке 23.

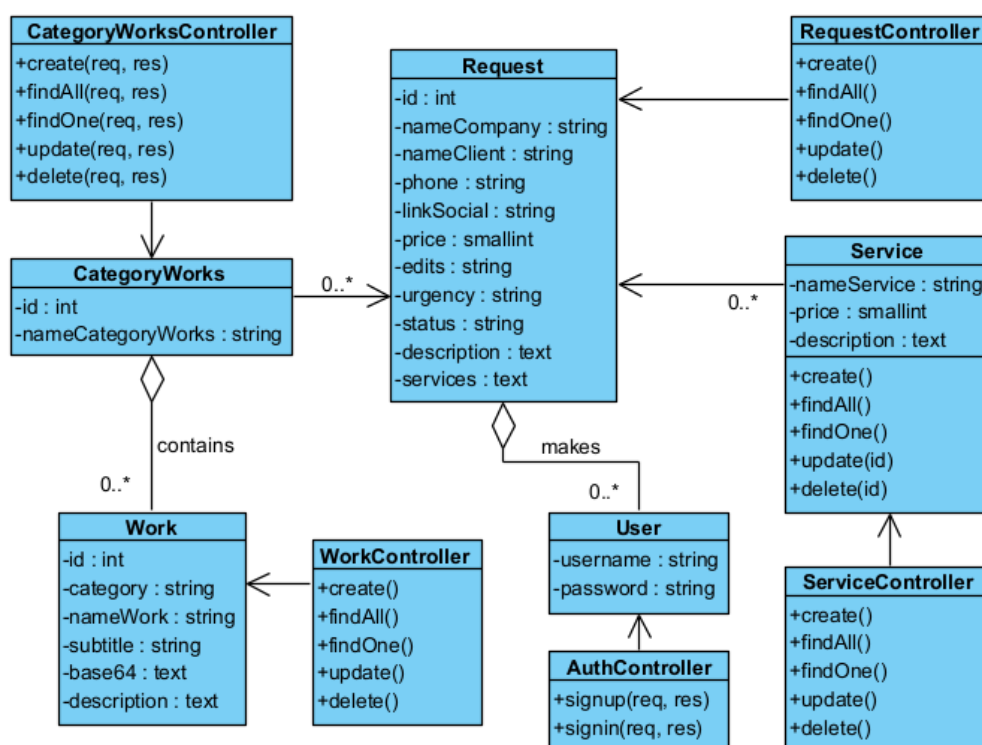


Рисунок 23 – Диаграмма классов программных средств персонального сайта-портфолио

Краткое описание функционального назначения классов приложения расположено в таблице 5.

Таблица 5 – Компоненты диаграммы классов приложения

Компонент	Назначение
CategoryWorks	Модель данных для категорий работ (хранит имя категории).
Work	Модель данных для портфолио-работ (хранит категорию, название, подзаголовок, изображение и описание).
User	Модель данных для пользователей (хранит логин и хеш пароля).
Request	Модель данных для заявок (хранит информацию о компании, клиенте, услугах, цене, статусе и комментариях).
Service	Модель данных для услуг (хранит название услуги, цену и описание).
AuthController	Контроллер, реализующий аутентификацию и авторизацию пользователей.
CategoryWorksController	Контроллер CRUD-операций для категорий работ.

Продолжение таблицы 5

Компонент	Назначение
WorkController	Контроллер CRUD-операций для портфолио-работ.
RequestController	Контроллер CRUD-операций для заявок.
ServiceController	Контроллер CRUD-операций для услуг.

Таким образом, были спроектированы классы программных средств для разрабатываемого веб-приложения, где пять основных моделей (CategoryWorks, Work, User, Request, Service) отражают сущности предметной области, а контроллеры (AuthController, CategoryWorksController и т. д.) реализуют над ними бизнес-логику и CRUD-операции. Таблица помогает быстро понять назначение каждого компонента: модели отвечают за структуру данных, контроллеры – за прием запросов и взаимодействие с данными.

3 Реализация программных средств для персонального сайта-портфолио

3.1 Реализация базы данных

Физическая модель базы данных – это модель данных, которая определяет, каким образом представляются данные, и содержит все детали, необходимые СУБД для создания базы данных [14].

Физическая модель базы данных «CoreDataDB» была построена в среде Visual Paradigm 17.2 в нотации IDEF1X, она представлена на рисунке 24.

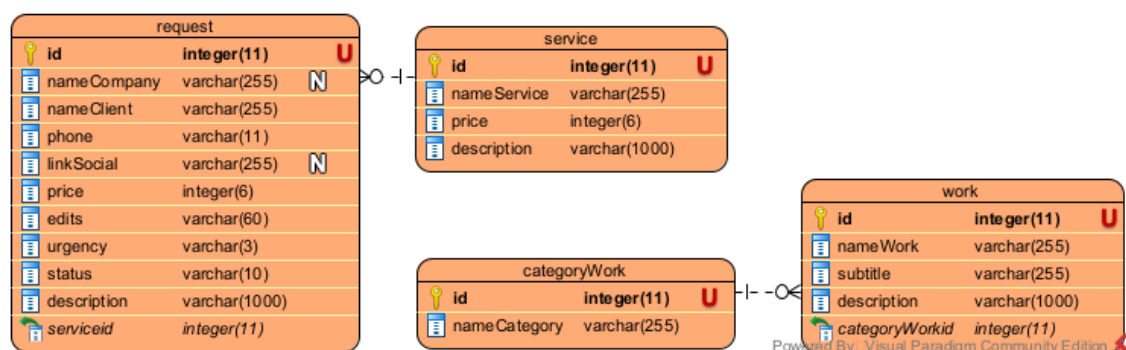


Рисунок 24 – Физическая модель данных предметной области «Персональный сайт-портфолио»

Для управления базой данных был выбран веб-инструмент «phpMyAdmin», имеющий удобный графический интерфейс для создания таблиц, управления данными и выполнения SQL-запросов.

Для тестирования работоспособности персонального сайта-портфолио были добавлены тестовые данные, наполнение которыми осуществлялось вручную.

Интеграция базы данных с серверной частью была реализована с помощью ORM Sequelize – специальной библиотеки для Node.js, которая дает возможность работать с данными на уровне JavaScript-объектов.

Данные из БД отображаются на страницах сайта, например, на странице «Портфолио» выводятся категории работ дизайнера. При добавлении новой

категории данные сохраняются в базу через функцию `CategoryWorks.create`, а при помощи функции `CategoryWorks.findAll` получаем список всех имеющихся категорий.

Пароль для входа на сайт с правами администратора перед записью в базу данных шифруется с помощью алгоритма `bcrypt`, затем для авторизации в веб-приложении пароль дешифруется и сверяется с введенным на странице.

3.2 Реализация серверной части приложения

Серверная часть программных средств персонального сайта-портфолио для позиционирования в профессиональной среде разрабатывалась на языке программирования JavaScript в среде программирования Visual Studio Code. Фрагменты кода серверной части приложения приведены в приложении В. С полным кодом можно ознакомиться на GitHub [15].

Серверная часть приложения реализована на Node.js с использованием быстрого и гибкого фреймворка Express.js, который обеспечивает высокую производительность, удобную работу с запросами [16].

Код серверной части организован по модульному принципу и разделен на следующие папки:

- «config», которая содержит подключение к БД и ключи для JWT-токенов;
- «models», определяющая структуру данных;
- «routes», которая задает маршруты API, связывая HTTP-запросы с контроллерами;
- «controllers», которая реализует обработку запросов и взаимодействие с моделями;
- «middleware», содержащая функции для аутентификации, валидации данных и обработки файлов.

Ниже в таблице 6 приведен перечень ключевых методов REST API для поддержки основных операций с данными.

Таблица 6 – Методы API программных средств персонального сайта-портфолио

URL запроса	Http метод	Описание
/api/auth/signin	POST	Авторизация пользователя
/api/requests	POST	Создание новой заявки
/api/requests	GET	Получение списка всех заявок
/api/requests/:id	GET	Получение информации о заявке
/api/requests/:id	PUT	Обновление данных заявки
/api/requests/:id	DELETE	Удаление заявки
/api/services	POST	Создание новой услуги
/api/services	GET	Получение списка всех услуг
/api/services/:id	GET	Получение информации об услуге
/api/services/:id	PUT	Обновление данных услуги
/api/services/:id	DELETE	Удаление услуги
/api/works	POST	Создание новой работы
/api/works	GET	Получение списка всех работ
/api/works/:id	GET	Получение информации о работе
/api/works/:id	PUT	Обновление данных работы
/api/works/:id	DELETE	Удаление работы
/api/files/upload	POST	Загрузка изображения на сервер
/api/files	GET	Получение списка всех загруженных изображений
/api/files/:name	GET	Скачивание изображения
/api/files/:name	DELETE	Удаление изображения
/api/categoryWorks	POST	Создание новой категории работ
/api/categoryWorks	GET	Получение списка всех категорий работ
/api/categoryWorks/:id	GET	Получение информации о категории
/api/categoryWorks/:id	PUT	Обновление данных категории
/api/categoryWorks/:id	DELETE	Удаление категории

Эти API методы дают возможность для взаимодействия между клиентской частью приложения и серверной, тем самым обеспечивая удобство использования.

3.3 Реализация клиентской части приложения

Клиентская часть программных средств персонального сайта-портфолио для позиционирования в профессиональной среде также, как и серверная часть, создана в среде разработки Visual Studio Code с использованием языка программирования JavaScript с использованием библиотеки ReactJS. Фрагменты

кода клиентской части приложения представлены в приложении Г. С полным кодом можно ознакомиться на GitHub [15].

Для сборки этой части приложения был применен модульный сборщик Webpack, который упрощает работу с модулями и компонентами проекта.

Для клиентской части приложения была создана диаграмма компонентов, которая изображена на рисунке 25.

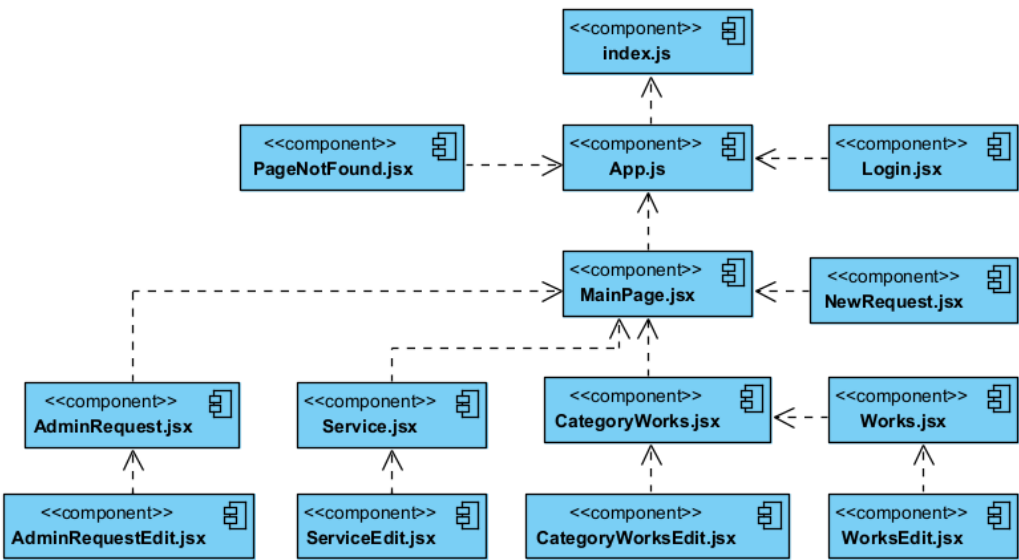


Рисунок 25 – Диаграмма компонентов клиентской части приложения

В таблице 7 приведено описание компонентов клиентской части приложения.

Таблица 7 – Компоненты клиентской части приложения

Компонент	Назначение
Index.js	Входная точка для React приложения
App.js	Корневой компонент, содержащий в себе маршрутизацию.
PageNotFound.jsx	Компонент, реализующий страницу «Страница не найдена 404»
Login.jsx	Компонент, реализующий функционал страницы авторизации.
MainPage.jsx	Компонент, реализующий функционал главной страницы, включающий в себя маршрутизацию по компонентам.
NewRequest.jsx	Компонент, реализующий функционал страницы создания заявки пользователем.

Продолжение таблицы 7

Компонент	Назначение
CategoryWorks.jsx	Компонент, реализующий отображение информации для любого пользователя и функционал страницы категорий работ при авторизации.
CategoryWorksEdit.jsx	Компонент, реализующий функционал страницы с редактированием категории работ, доступный для авторизованного пользователя.
Works.jsx	Компонент, реализующий функционал страницы с работами конкретной категории.
WorksEdit.jsx	Компонент, реализующий функционал редактирования работы в портфолио после авторизации пользователя.
Service.jsx	Компонент, реализующий отображение информации о стоимости и описании услуг для любого пользователя и функционал страницы при авторизации.
ServiceEdit.jsx	Компонент, реализующий функционал страницы об услугах для авторизованного пользователя.
AdminRequest.jsx	Компонент, реализующий функционал страницы заявок для авторизованного пользователя.
AdminRequestEdit.jsx	Компонент, реализующий функционал редактирования заявки для авторизованного пользователя.

Для обмена данными между клиентом и сервером применяется библиотека axios. Ее использование связано с удобной настройкой запросов и простой обработкой исключений.

3.4 Реализация интерфейса приложения

При реализации интерфейса веб-приложения были учтены требования из пункта 2.3 и созданы следующие страницы:

- страница авторизации;
- страница портфолио;
- страница создания заявки;
- страница со стоимостью и описанием услуг;
- страница просмотра заявок для администратора;
- главная страница сайта.

Все страницы будут представлены на функционале администратора. Для того чтобы зайти на сайт с его правами пользователь должен перейти на страницу авторизации, представленной на рисунке 26.

The image shows a web form for user authentication. At the top, the title 'Авторизация' is centered. Below it are three elements: a text input field for the username labeled 'Имя пользователя' with the value 'admin' and a number '1' pointing to it; a password input field labeled 'Пароль' with the value 'admin123' and a number '2' pointing to it; and a green button labeled 'Войти' with a number '3' pointing to it.

Рисунок 26 – Страница авторизации

Основные элементы страницы авторизации:

- 1 – поле ввода логина пользователя;
- 2 – поле ввода пароля пользователя;
- 3 – кнопка «Войти».

После ввода данных и нажатия на кнопку входа система проверяет корректность этих данных и существование соответствующего пользователя. Если обработка данных прошла успешно, пользователь переходит на главную страницу сайта, которая состоит из шести блоков (рисунок 27).

При входе в веб-приложение отображается блок с заголовком сайта и кратким описанием, какие виды услуг можно получить.

Основные элементы интерфейса первого блока страницы:

- 1 – ссылки на мессенджеры дизайнера;
- 2 – меню навигации сайта;
- 3 – кнопка «Оставить заявку».

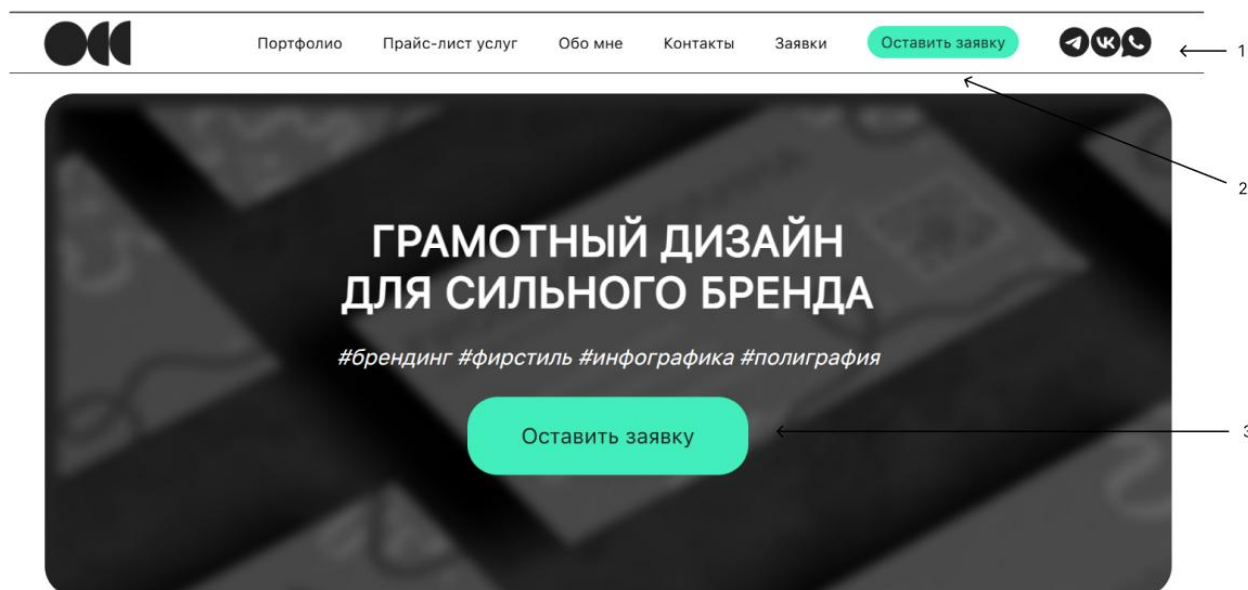


Рисунок 27 – Первый блок главной страницы

Далее начинается раздел «Обо мне» с краткой информацией о дизайнере, представленный на рисунках 28, 29.

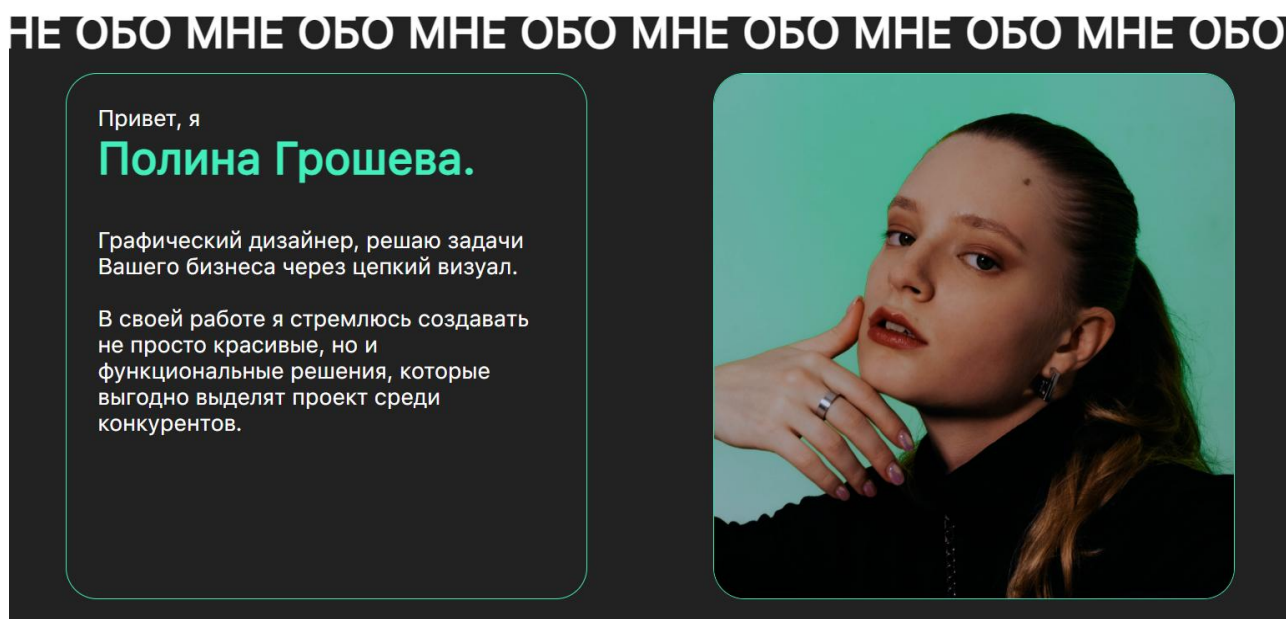


Рисунок 28 – Второй блок главной страницы

Второй блок переходит в третий, продолжая представлять дизайнера. В третьем блоке описаны технические навыки специалиста, наглядно показаны его уровни владения профессиональными программами, а также приведены его этапы обучения и имеющееся образование.

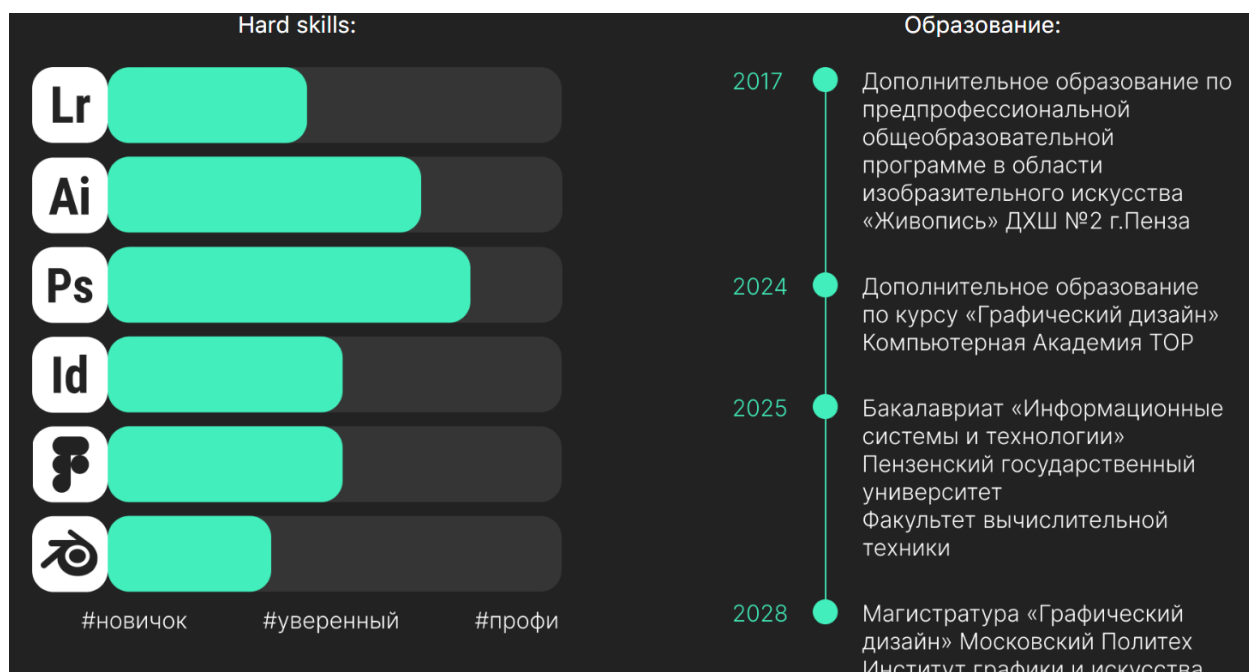


Рисунок 29 – Третий блок главной страницы

Следующий блок содержит информацию об этапах работы со специалистом, что упрощает потенциальному клиенту понимание процесса работы, повышает уровень доверия и помогает избежать недоразумений (рисунок 30).

РАБОТЫ ЭТАПЫ РАБОТЫ ЭТАПЫ РАБОТЫ ЭТАПЫ РАБОТЫ

- [1]. Обсуждение задачи/брифинг**
Бриф — неотъемлемая часть комфортного сотрудничества дизайнера с заказчиком. Это документ, который мы заполним вместе с Вами, устроим созвон в удобное время и подробно обсудим проект.
- [2]. Внесение предоплаты**
Внесение предоплаты в размере 30% от суммы гарантирует выполнение заказа и соблюдение сроков сдачи.
- [3]. Разработка и презентация**
Процесс разработки дизайна, постоянное взаимодействие с Вами по уточняющим вопросам, презентация проекта.
- [4]. Правки и согласование**
При необходимости внесение правок, полное согласование проекта, утверждение всех макетов.
- [5]. Оплата и передача**
Оплата оставшейся части и передача всех файлов в удобном Вам формате + памятка по использованию.

Рисунок 30 – Четвертый блок главной страницы

После описания этапов работы пользователю предлагается перейти на страницу с портфолио (рисунок 31).

ОЛИО ПОРТФОЛИО ПОРТФОЛИО ПОРТФОЛИО ПОРТФО

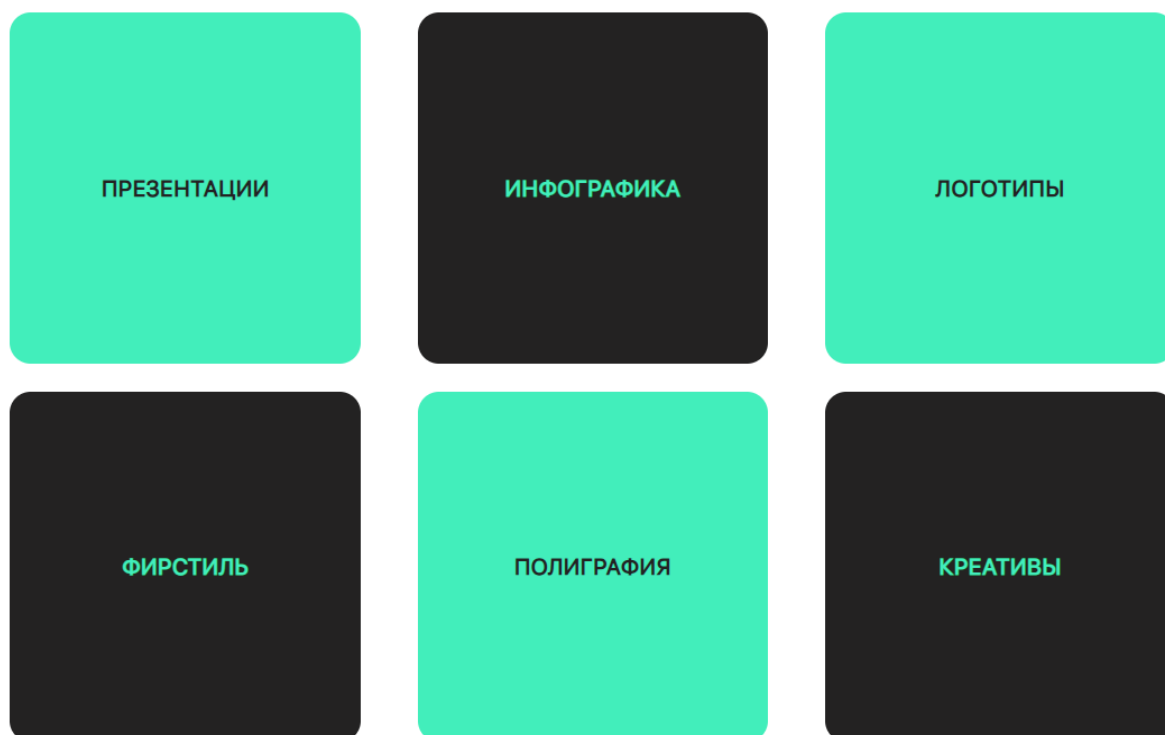
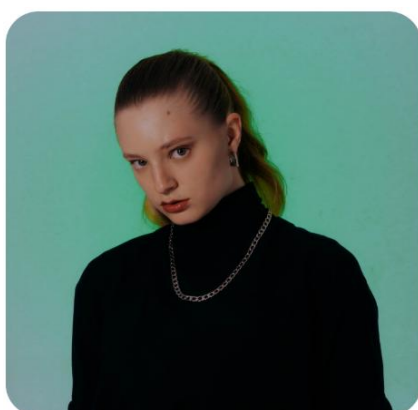


Рисунок 31 – Пятый блок главной страницы

Кликнув в любой области этого блока, пользователь попадет на страницу с выбором категорий работ, где в дальнейшем может ознакомиться с портфолио.

Последний блок главной страницы представляет раздел «Контакты», где продублированы ссылки для возможной связи с дизайнером, его адрес электронной почты и номер телефона (рисунок 32).



ОСТАЛИСЬ ВОПРОСЫ?

Свяжитесь со мной любым удобным для Вас способом.

Я в соцсетях:



polinagrosheva2018@yandex.ru

[+7 \(937\) 915-00-72](tel:+7(937)915-00-72)

Рисунок 32 – Шестой блок главной страницы

Каждая страница заканчивается подвалом сайта, который дублирует

панель меню для быстроты переходов и удобства навигации.

Страница «Прайс-лист услуг», на которую пользователь может перейти через панель меню, представлена на рисунке 33.

Ознакомьтесь со стоимостью работ
Не нашли что искали? Свяжитесь со мной в соцсетях

Услуга	Описание	Цена	Действия
Визитка, сертификат	Односторонний дизайн (двухсторонний стоимость x2)	2000Р	← 1
Флаер, листовка, банер	Односторонний дизайн (двухсторонний стоимость x2)	2000Р	
Презентация	За один слайд	450Р	
Инфографика	За один слайд	700Р	
Логотип	3 варианта, чб-варианты, презентация, мокапы	10000Р	
Шаблоны для постов/сторис	Визуальная раскладка на 9 постов	1500Р	
Креативы	За один рекламный продукт	1750Р	
Упаковка соцсетей	Шапка профиля, хайлайтс, меню, обложки	2500Р	

Наименование

Описание

Цена

← 2

Добавить

← 3

Рисунок 33 – Страница «Прайс-лист услуг»

Основные элементы страницы «Прайс-лист услуг»:

- 1 – иконки редактирования и удаления услуги;
- 2 – поля ввода наименования, описания и стоимости услуги;
- 3 – кнопка «Добавить» для создания новой услуги.

При нажатии на кнопку «Добавить» система проверяет наличие наименования услуги, если оно есть, появится всплывающее окно с уведомлением о добавлении новой услуги (рисунок 34).

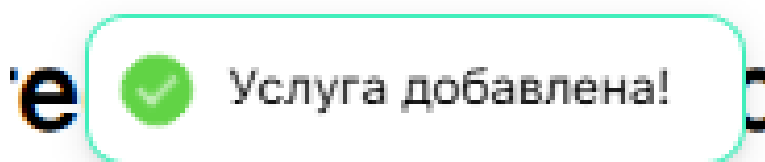


Рисунок 34 – Уведомление о добавлении услуги

Если наименование услуги отсутствует, система не даст добавить ее в перечень и выдаст сообщение об ошибке (рисунок 35).

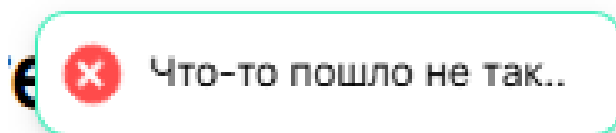


Рисунок 35 – Уведомление об ошибке при попытке добавления услуги

При нажатии на иконку редактирования услуги пользователь перейдет к соответствующему функционалу, представленном на рисунке 36.

Услуга: Упаковка соцсетей

 A form for editing a service. It consists of three input fields and two buttons at the bottom. The first field is labeled 'Услуга' (Service) and contains the text 'Упаковка соцсетей'. The second field is labeled 'Описание' (Description) and contains the text 'Шапка профиля, хайлайтс, меню, обложки'. The third field is labeled 'Цена' (Price) and contains the text '2500'. Below the fields are two buttons: 'Удалить' (Delete) and 'Обновить' (Update). Numbered callouts point to each element: 1 points to the service name field, 2 points to the description field, 3 points to the price field, 4 points to the 'Удалить' button, and 5 points to the 'Обновить' button.

Рисунок 36 – Функционал редактирования услуги

Основные элементы функционала редактирования:

- 1 – поле ввода наименования услуги;
- 2 – поле ввода описания услуги;
- 3 – поле ввода стоимости услуги;
- 4 – кнопка «Удалить» для удаления услуги;
- 5 – кнопка «Обновить» для обновления услуги.

После нажатия кнопки «Обновить» всплывает окно оповещения, представленное на рисунке 37.

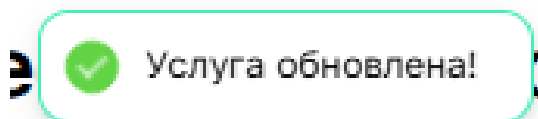


Рисунок 37 – Уведомление об обновлении данных по услуге

После нажатия кнопки «Удалить» или иконки корзины система оповещает пользователя об успешном удалении услуги (рисунок 38).

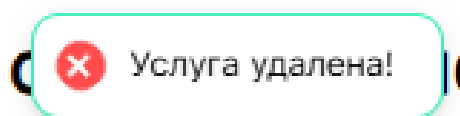


Рисунок 38 – Уведомление об удалении услуги

На странице «Оставить заявку» пользователь может заполнить все необходимые поля и рассчитать приблизительную стоимость работы, выбрав нужные ему услуги. На рисунке 39 изображена часть страницы.

**Хотите поработать со мной?
Тогда давайте рассчитаем примерную стоимость!**

Заполните необходимые поля и выберите нужные вам услуги ниже

- 1 → Название компании (Необязательно)
Название компании
- 2 → ФИО представителя (Обязательно)
Грошева Полина Александровна
- 3 → Телефон (Обязательно)
+79379150072
- 4 → Ссылка на аккаунт в соц. сетях (VK, TG, Whatsapp) (Необязательно)
<https://m.vk.com/grosh06>
- 5 → Что вам нужно сделать?
- 6 →

☒ Визитка, сертификат	☐ Флаер, листовка, банер	☐ Презентация
Односторонний дизайн (двухсторонний стоимость x2)	Односторонний дизайн (двухсторонний стоимость x2)	За один слайд
☐ Инфографика	☒ Логотип	☐ Шаблоны для постов/сторис
За один слайд	3 варианта, чб-варианты, презентация, мокапы	Визуальная раскладка на 9 постов

Рисунок 39 – Страница «Оставить заявку» часть 1

Одни из основных элементов при заполнении заявки:

1 – поле ввода названия компании, если оно есть;

2 – поле ввода имени представителя или самого клиента;

- 3 – поле ввода номера телефона для связи с заказчиком;
- 4 – поле ввода ссылок на мессенджеры для связи с заказчиком;
- 5 – чекбокс для выбора нужных услуг заказчику;
- 6 – краткое описание услуги.

На рисунке 40 изображена вторая часть страницы, на которой заканчивается ее функционал.

☐ Креативы
За один рекламный продукт

☐ Упаковка соцсетей
Шапка профиля, хайлайтс, меню, обложки

☐ Фирменный стиль
2 варианта логотипа, подбор шрифтовой пары, цветовой палитры, разработка фирменной графики, 3 макета

1 Выберите подходящее предложение:

☒ 3 круга правок в рамках технического задания

☐ 5 кругов правок в рамках технического задания

2 Это срочный заказ?

☐ Да

☒ Нет

3 Ваша примерная стоимость заказа составит:
(Меняется в зависимости от Ваших индивидуальных требований и объема работ)

12000₽

4 Комментарий (Необязательно)

Комментарий

5 Оставить заявку

Рисунок 40 – Страница «Оставить заявку» часть 2

Одни из основных элементов при заполнении заявки:

- 1 – радиокнопка с выбором количества правок;
- 2 – радиокнопка с выбором срочности заказа;
- 3 – поле расчета стоимости заказа;
- 4 – поле ввода комментария для заказчика, если нужно что-то уточнить;
- 5 – кнопка «Оставить заявку».

При нажатии на кнопку «Оставить заявку» после корректного заполнения необходимых полей, пользователь увидит уведомление и сообщение об успешно созданной заявке (рисунок 41).

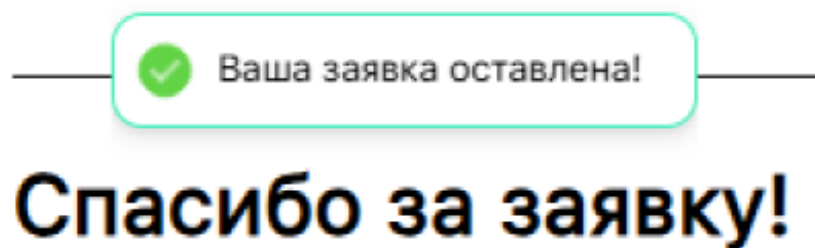


Рисунок 41 – Уведомление и сообщение об успешно созданной заявке

Администратор на странице «Заявки» может просмотреть, отфильтровать и редактировать все созданные заявки клиентов (рисунок 42).

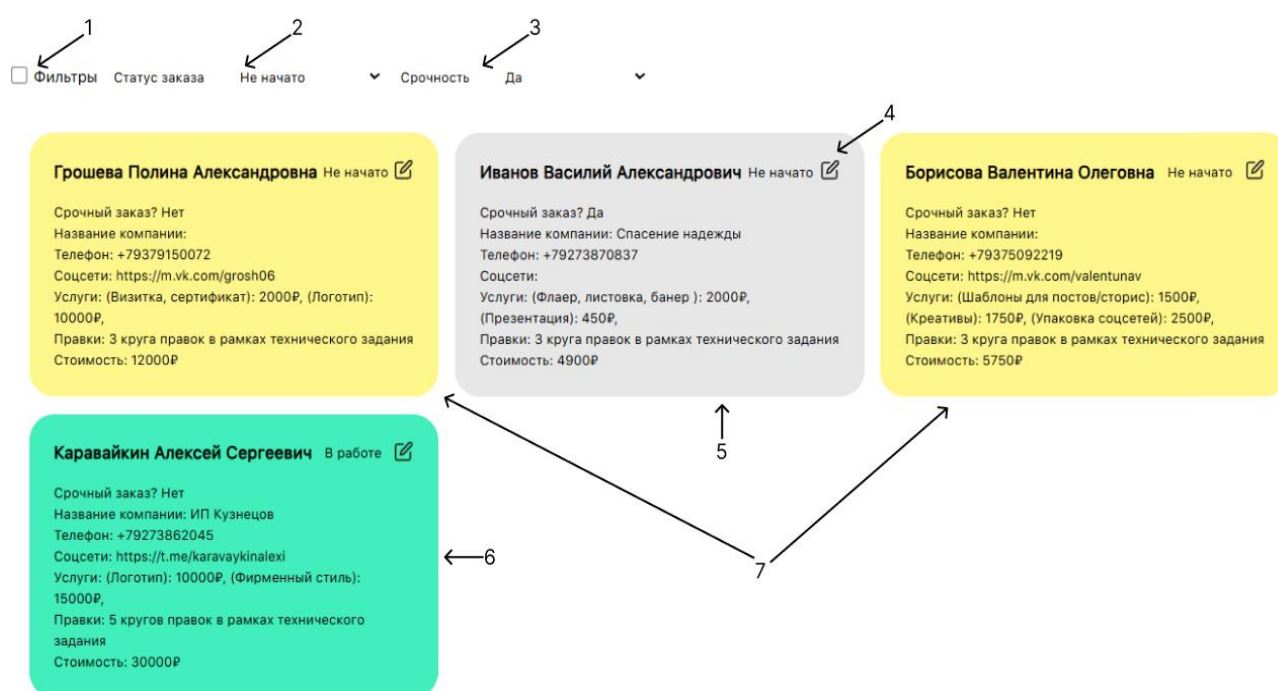


Рисунок 42 – Страница «Заявки»

Основные элементы страницы «Заявки»:

- 1 – чекбокс для применения фильтров;
- 2 – фильтр по статусу заказа с выпадающим меню «Не начато», «В работе», «Готово»;
- 3 – фильтр по срочности заказа с выпадающим меню «Да», «Нет»;
- 4 – иконка редактирования заявки;
- 5 – заявка с низкой прибылью;
- 6 – заявка с высокой прибылью;

7 – заявка со средней прибылью.

Интерфейс редактирования информации по заявке представлен на рисунке 43.

Заявка от Каравайкин Алексей Сергеевич, Цена: 30000₽

The screenshot shows a form for editing a request. At the top, it says 'Заявка от Каравайкин Алексей Сергеевич, Цена: 30000₽'. Below this are three input fields: 'Телефон' (Phone) with the value '+79273862045', 'Статус' (Status) with the value 'В работе' (In progress), and 'Цена' (Price) with the value '30000'. At the bottom are two buttons: 'Удалить' (Delete) and 'Обновить' (Update). Numbered callouts point to these elements: 1 points to the phone number field, 2 points to the status field, 3 points to the price field, 4 points to the 'Удалить' button, and 5 points to the 'Обновить' button.

Рисунок 43 – Функционал редактирования заявки

Основные элементы функционала редактирования заявки:

- 1 – поле ввода телефонного номера;
- 2 – поле изменения статуса заказа;
- 3 – поле ввода стоимости услуг после обсуждения работы;
- 4 – кнопка «Удалить» заявку;
- 5 –кнопка «Обновить» заявку.

При клике на кнопку для обновления система выдает уведомление об изменении данных, при нажатии на кнопку удаления появится аналогичное уведомление об удалении заявки. Эти уведомления выглядят идентично по дизайну с представленными на рисунках 37, 38.

На странице «Портфолио» сначала представлены категории услуг, перейдя в которые можно увидеть работы по выбранному направлению (рисунок 44).

Основные элементы функционала редактирования заявки:

- 1 – иконки редактирования и удаления категории работы;
- 2 – поле ввода наименования категории;
- 3 – кнопка «Добавить» для создания новой категории.

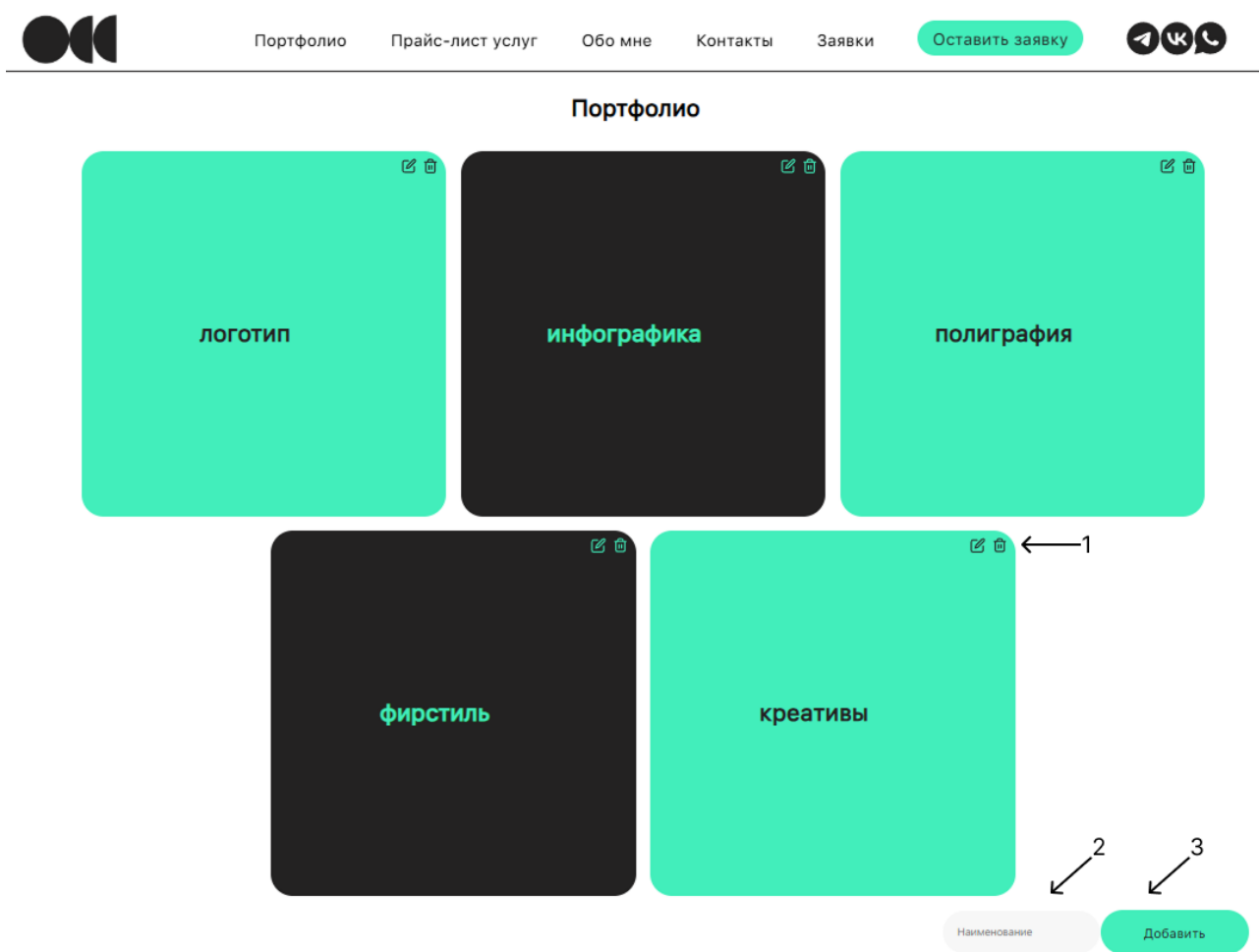


Рисунок 44 – Страница «Портфолио»

При клике по иконке редактирования откроется функционал для изменения названия категории, дизайн которого идентичен функционалу редактирования заявки. При удалении категории будут удалены и все находящиеся в ней работы из портфолио.

При нажатии на одну из категорий пользователь переходит к ее работам. По каждой работе представлено небольшое описание и подгружены изображения. На рисунке 45 показаны работы из категории «логотип».

Основные элементы страницы:

- 1 – заголовок страницы;
- 2 – изображения работ;
- 3 – заголовок работы;
- 4 – подзаголовок работы;

- 5 – описание работы;
- 6 – иконка редактирования работы;
- 7 – иконка удаления работы из портфолио.

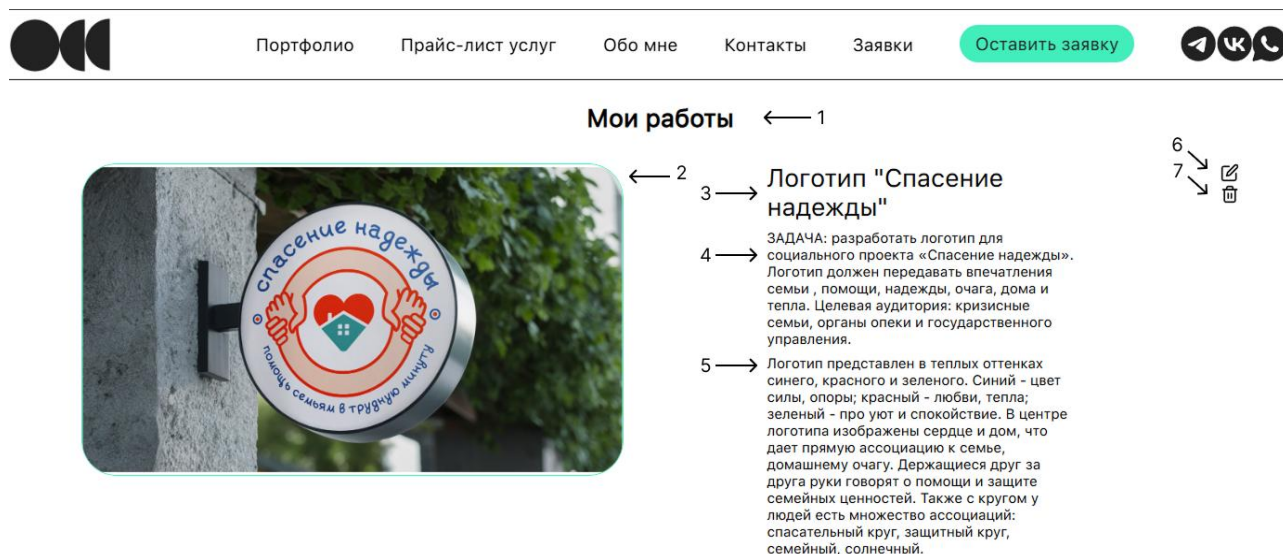


Рисунок 45 – Страница «Портфолио», категория «Логотип»

Прокрутка изображений в работе осуществляется через слайдер-контейнер за счет свойства `overflow: scroll`. Перелистывание изображений выполняется вручную, например, колесом прокрутки мыши.

Функционал добавления новой работы представлен на рисунке 46.

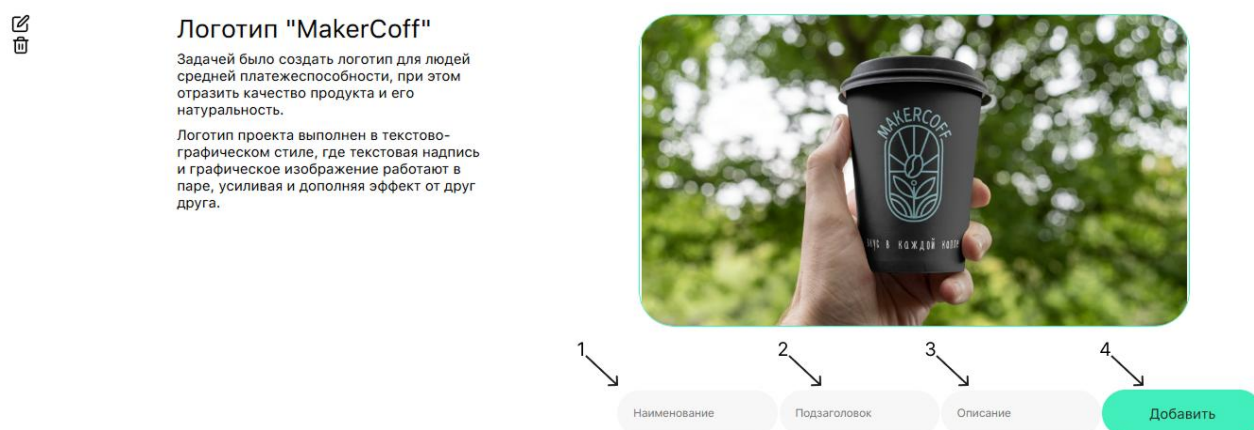


Рисунок 46 – Функционал добавления новой работы в портфолио

Основные элементы функционала:

- 1 – поле ввода наименования работы;
- 2 – поле ввода подзаголовка работы;

3 – поле ввода описания работы;

4 – кнопка «Добавить» для создания работы.

При нажатии на иконку редактирования откроется функционал с доступом к изменению данных работы и подгрузке изображений по проекту (рисунок 47).

Работа: Логотип "MakerCoff"

Имя работы

Логотип "MakerCoff"

Подзаголовок

Задачей было создать логотип для людей средней платежеспособности, при этом отразить качество продукта и его натуральность.

Описание

Логотип проекта выполнен в текстово-графическом стиле, где текстовая надпись и графическое изображение работают в паре, усиливая и дополняя эффект от друг друга.

Категория работ

ЛОГОТИПЫ

Изображение

Число файлов: 2



Рисунок 47 – Страница редактирования работы в портфолио

При наведении на любую кнопку интерфейса она меняет свой цвет с фирменного зеленого на черный и наоборот, что сделано для удобства пользователя, служит ему визуальным сигналом и помогает лучше ориентироваться на веб-страницах.

Создание пользовательского интерфейса проводилось с применением CSS-стилей и HTML-разметки. HTML отвечает за построение структуры веб-страниц, а CSS помогает при оформлении дизайна, включая настройку шрифтов, цветов и других элементов.

Все уведомления на сайте созданы с помощью специальной библиотеки react-hot-toast, которая позволяет показывать пользователю сообщения с минимальной настройкой, что существенно упрощает работу. Уведомления автоматически исчезают через заданное время [17].

4 Контроль качества программного обеспечения

4.1 Обоснование выбора режима тестирования и метрик кода

Тестирование ПО (программное обеспечение) – проверка соответствия между реальным и ожидаемым поведением программы, проводится на наборе тестов, который выбирается некоторым образом [18].

Тестирование различными методами программы нужно для того, чтобы убедиться в корректной работе приложения, удобстве использования, а также чтобы устранить ошибки, если таковые есть. Тестирование можно разделить на два типа: функциональное и нефункциональное.

Функциональное тестирование отвечает за проверку функциональности приложения. Оно позволяет убедиться в том, что приложение работает корректно и выполняет функции, соответствующие требованиям пользователей и заказчика [19].

К нефункциональному тестированию относят, например, тестирование производительности и нагрузочное тестирование, целью которых является проверка того, как быстро приложение сможет обработать большой объем данных и как много пользователей может одновременно использовать веб-сайт. Данные методы лучше подходят для тестирования систем, работающих с большим количеством запросов, таких как интернет-магазины и онлайн-кинотеатры. Персональный сайт-портфолио не предполагает такого потока пользователей, однако данное приложение должно корректно отображаться через любой браузер. Это важно, так как любые ошибки на сайте могут испортить впечатление потенциального клиента. Поэтому для проверки того, что веб-приложение работает одинаково стабильно и без ухудшения качества на разных браузерах, выбрано кросс-браузерное тестирование.

Метрики программного кода являются важным инструментом для оценки и анализа характеристик качества ПО. Они позволяют детально изучить технические характеристики продукта, оценить его сложность и

масштабируемость, спрогнозировать трудозатраты, проверить код на соответствие стандартам качества.

Для анализа качества программного кода веб-приложения персонального сайта-портфолио, были использованы широко применяемые метрики, такие как метрика Холстеда и цикломатическая сложность.

Метрика Холстеда основывается на подсчете операторов, операндов и строк кода, позволяя оценить сложность программы, объем работы и потенциал для расширения функциональности.

Цикломатическая сложность измеряет количество независимых путей выполнения в коде. Данная метрика покажет, насколько сложен код для тестирования и поддержки.

4.2 Тестирование программных средств для персонального сайта портфолио

4.2.1 Функциональное тестирование

Функциональное тестирование было проведено методом черного ящика, который не предполагает доступа к системе, а основывается исключительно на работе с внешним интерфейсом. Такой подход позволяет сосредоточиться на проверке правильной работы приложения с точки зрения пользователя, не вдаваясь в подробности внутренней реализации.

К основному функционалу данного приложения относится обратная связь с клиентом, который может оставить заявку на оказание услуг. Администратор, в свою очередь, может редактировать данные: обновлять статус каждой заявки, корректировать итоговую стоимость, а также удалять заявки. На рисунке 48 изображена диаграмма состояний и переходов заявки, которая наглядно демонстрирует все этапы жизненного цикла заявки и условия переходов между ними. У заявки есть три состояния: не существует, активна, удалена. После того как пользователь создал заявку, она переходит в состояние, когда ее можно редактировать или удалять.

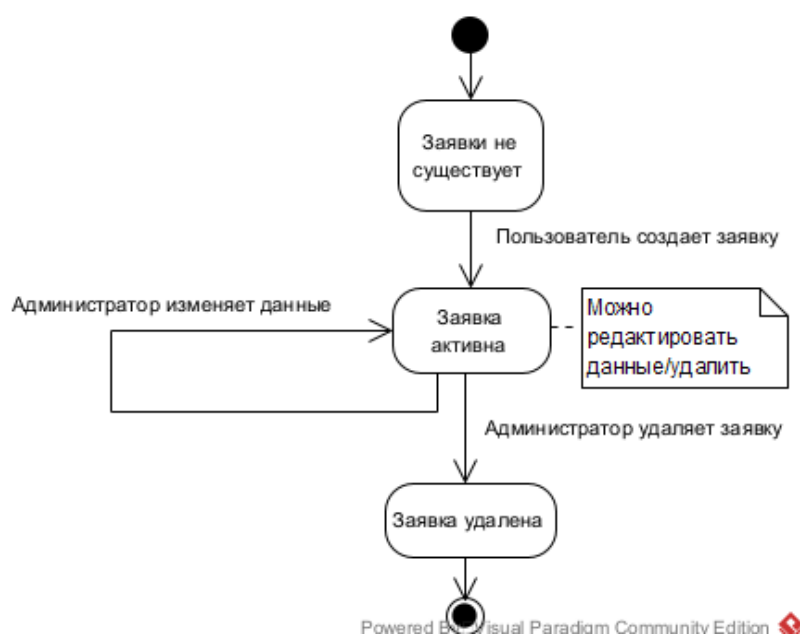


Рисунок 48 – Диаграмма состояний и переходов заявки

В таблице 8 представлен план функционального тестирования по заявкам.

Результаты функциональных тестов представлены в приложении Д.

Таблица 8 – План функционального тестирования заявок

Назначение теста	Описание теста	Результаты теста
Проверка отображения страниц	Пользователь переходит на страницы «Оставить заявку», «Заявки» на которых отображены все необходимые поля	Тест выполнен успешно (рисунок Д.1, рисунок Д.2)
Проверка создания заявки	Пользователь заполняет все обязательные поля и нажимает на кнопку «Оставить заявку», заявка успешно создана	Тест выполнен успешно (рисунок Д.3)
Проверка обновления данных заявки	Администратор на странице «Заявки» меняет контактный номер клиента	Тест выполнен успешно (рисунок Д.4)
Проверка обновления данных заявки	Администратор на странице «Заявки» меняет статус заявки	Тест выполнен успешно (рисунок Д.5)
Проверка фильтрации заявок	Администратор на странице «Заявки» фильтрует заявки по статусу и срочности заказа	Тест выполнен успешно (рисунок Д.6)
Проверка удаления заявки	Администратор на странице «Заявки» удаляет заявку	Тест выполнен успешно (рисунок Д.7)

Также к основному функционалу веб-приложения относится работа с портфолио специалиста. Для гостя сайта – это переход к странице портфолио и

просмотр работ. Для администратора – создание, редактирование и удаление. Диаграмма состояний и переход для работы портфолио показана на рисунке 49.

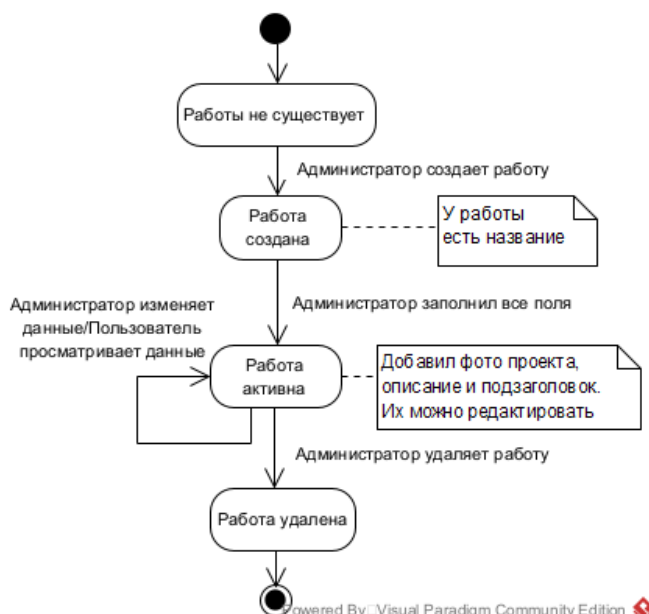


Рисунок 49 – Диаграмма состояний и переходов работы портфолио

Диаграмма состояний и переходов работы портфолио была использована как основа, на которой построен план функционального тестирования портфолио работ, представленного в таблице 9.

Таблица 9 – План функционального тестирования портфолио работ

Назначение теста	Описание теста	Результаты теста
Проверка отображения страницы	Пользователь переходит на страницы «Портфолио», на которой отображены все категории работ	Тест выполнен успешно (рисунок Д.8)
Проверка отображения работ в категории	Пользователь переходит в одну из категорий портфолио и смотрит работы	Тест выполнен успешно (рисунок Д.9)
Проверка просмотра изображений работ	Пользователь может перелистывать изображения для просмотра работы	Тест выполнен успешно (рисунок Д.10)
Проверка создания работы	Администратор может создать работу в портфолио	Тест выполнен успешно (рисунок Д.11)
Проверка редактирования работы	Администратор может добавить дополнительную информацию о работе и прикрепить изображения	Тест выполнен успешно (рисунок Д.12, рисунок Д.13)
Проверка удаления работы	Администратор может удалить работу из портфолио	Тест выполнен успешно (рисунок Д.14)

Полученные результаты соответствуют ожидаемым, что свидетельствует о корректной работе программы.

4.2.2 Кросс-браузерное тестирование

Самые важные страницы персонального сайта-портфолио с точки зрения конечного пользователя – это главная страница и страница с работами портфолио специалиста. На этих страницах находятся контакты для связи, информация о дизайнере и об этапах работы, а также примеры работ. Потенциальный клиент и гость сайта может пользоваться различными веб-браузерами, например, Google Chrome, Mozilla Firefox, Safari, Microsoft Edge и Яндекс.Браузер. Каждый браузер может интерпретировать HTML, CSS и JavaScript по-разному, что может привести к ошибкам в отображении и функциональности приложения [20]. Такие ошибки могут возникать в связи с тем, что разные браузеры написаны на разных движках:

- Google Chrome, Microsoft Edge, Opera и Яндекс.Браузер написаны на Blink;
- Safari написан на WebKit;
- Mozilla Firefox написан на Gecko.

Поэтому для проверки отображения страниц достаточно взять три браузера, написанных на разных движках.

В таблице 10 представлен план кросс-браузерного тестирования. Результаты тестов приведены в приложении Е.

Таблица 10 – План кросс-браузерного тестирования

Страница веб-сайта	Браузер	Разрешение
Главная страница	Google Chrome	1024 px (рисунок Е.1)
Главная страница	Mozilla Firefox	1024 px (рисунок Е.2)
Главная страница	Safari	1024 px (рисунок Е.3)
Портфолио работ	Google Chrome	1024 px (рисунок Е.4)
Портфолио работ	Mozilla Firefox	1024 px (рисунок Е.5)
Портфолио работ	Safari	1024 px (рисунок Е.6)

По результатам кросс-браузерного тестирования можно сделать вывод, что приложения корректно открывается в всех популярных браузерах. Пользовательский интерфейс сайта отлично оптимизирован для необходимых разрешений при просмотре приложения на компьютере и ноутбуке.

4.3 Оценка качества программного кода

4.3.1 Метрика Холстеда

Для расчета метрики Холстеда были выбраны функции `onChangeService` и `onChangeSumm`, представленные в приложении Г в листинге Г.4.

Операторы, операнды и их число представлены в таблице 11.

Таблица 11 – Операторы и операнды функций

Операторы	Число операторов	Операнды	Число операндов
<code>let</code>	2	<code>index</code>	2
<code>=</code>	5	<code>e</code>	1
<code>[</code>	3	<code>strService</code>	3
<code>]</code>	3	<code>intServiceSumm</code>	3
<code>.</code>	10	<code>selectServices</code>	4
<code>!</code>	1	<code>this.state.listServices</code>	1
<code>map</code>	2	<code>slice</code>	1
<code>(</code>	5	<code>checked</code>	2
<code>)</code>	5	<code>item</code>	2
<code>+</code>	4	<code>selectService</code>	2
<code>,</code>	2	<code>nameService</code>	1
<code>if</code>	2	<code>price</code>	2
<code>!=</code>	1	<code>"("</code>	1
<code>undefined</code>	1	<code>"";."</code>	1
<code>replace</code>	1	<code>"P"</code>	1
<code>filter</code>	2	<code>","</code>	1
<code>Summ</code>	2	<code>undefined</code>	1
<code>this</code>	4	<code>"</code>	1
<code>setState</code>	1	<code>Summ</code>	2

Продолжение таблицы 11

Операторы	Число операторов	Операнды	Число операндов
console.log	1	urgency	2
		"Да"	1
		"Нет"	1
		fullSumm	2
		this.state.priceSumm	1

Таким образом, общее число вхождений операторов составило 56, операндов – 38. На основе этих данных можно рассчитать основные характеристики и проанализировать сложность программного кода.

Результаты вычислений представлены в таблице 12.

Таблица 12 – Результаты вычислений метрики Холстеда

Значение	Результат
Словарь программы (n)	44
Длина программы (N)	94
Объем программы (V)	512.24
Сложность программы (D)	4.17

Метрика Холстеда показывает, что данные функции имеют среднюю сложность, все показатели находятся в пределах нормы. Длина реализации кода меньше длинны программы, а значит, несовершенства в программе отсутствуют, можно подвести итог: код читаем, поддерживаем и не перегружен.

4.3.2 Цикломатическая сложность

Для проверки цикломатической сложности кода была выбрана функция deleteImagesWorks, осуществляющая удаление изображения из портфолио работы. Данная функция представлена в приложении Г листинга Г.7.

Для расчета метрики была построена блок-схема для создания графа потока управления функции, изображенный на рисунке 50. Блок-схема была построена с помощью графической среды Draw.io [21].

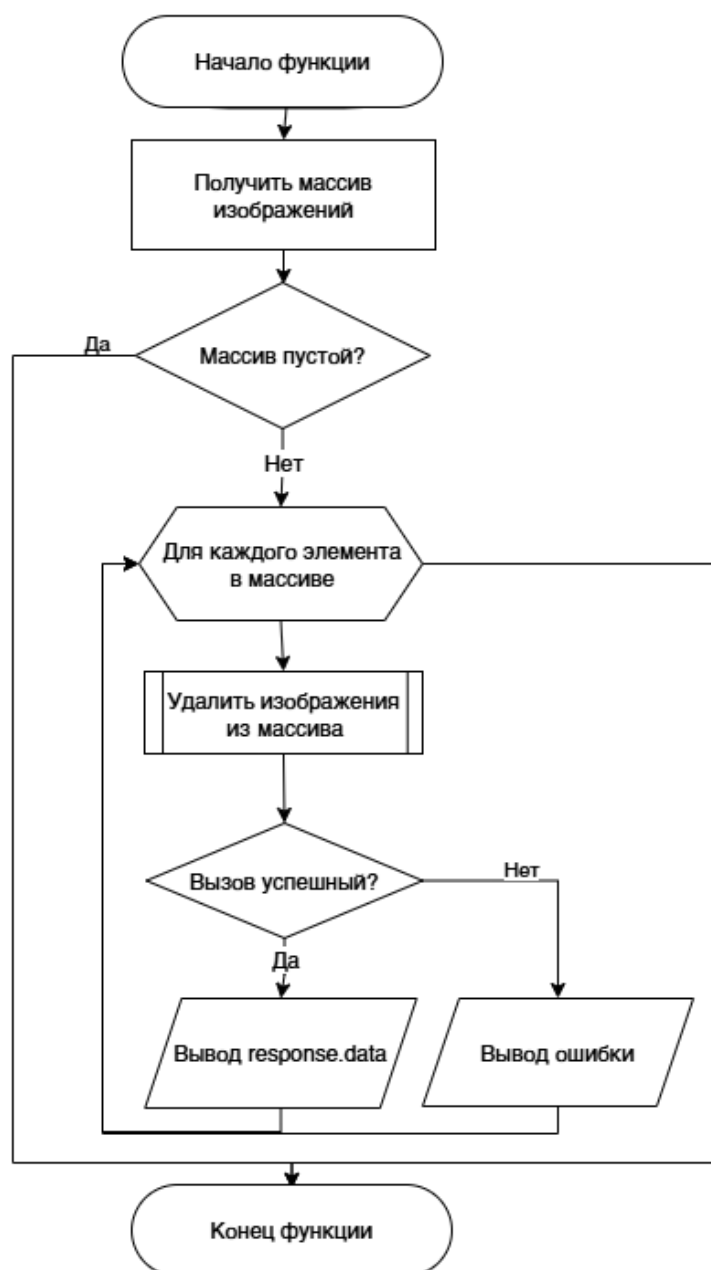


Рисунок 50 – Блок-схема функции deleteImagesWorks

На основе этой схемы можно построить граф потока управлений, который состоит из узлов, названными латинскими строчными буквами, и ребер, которые представляют собой связи между узлами и изображаются как стрелки на рисунке 51.

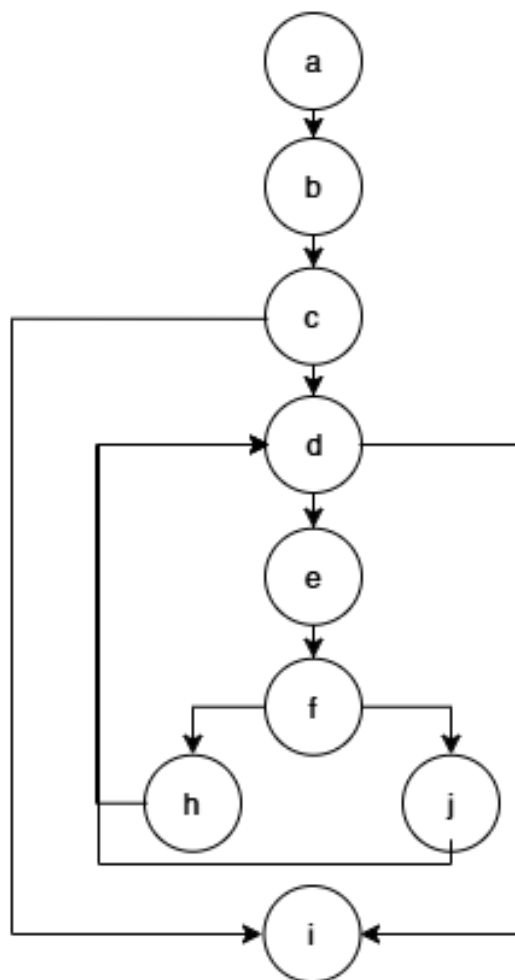


Рисунок 51 – Граф потока управления функции

Цикломатическая сложность рассчитывается как разность между количеством ребер и узлов сложенной с удвоенный произведением количества компонент связности. Количество узлов в графе равно девяти, а количество ребер – одиннадцати. Функция в расчет берется одна, значит и количество компонент равно единице. Таким образом, цикломатическая сложность данной функции равна четырем, что говорит о низкой сложности. Функция является относительно простой для понимания и поддержки, код программы работает корректно и в пределах нормы.

Заключение

В рамках выпускной квалификационной работы проведен анализ предметной области и поставлена задача на разработку персонального сайта-портфолио. Проведен обзор приложений аналогичного назначения и выполнен сравнительный анализ. Выявлены и функциональные и нефункциональные требования к разработке программного обеспечения. Созданы UML-диаграммы вариантов использования и спецификации наиболее значимых бизнес-процессов.

Проведено планирование разработки и оценка бюджета. Затраты на проект составили 125623,89 рублей. Начало работ определено как 8 апреля, а окончание работ по проекту запланировано на 4 июня текущего года. Затраты трудовых ресурсов составили 279 часов.

На стадии проектирования разработана архитектура программного обеспечения, выделены классы разрабатываемого приложения, а также выявлены требования к интерфейсу, разработаны концептуальная и логические модели базы данных и созданы две UML-диаграммы: компонентов структуры приложения и развертывания архитектуры.

Разработка программных средств персонального сайта-портфолио для позиционирования в профессиональной среде включала в себя реализацию базы данных в MySQL с использованием инструмента phpMyAdmin, создание серверной части и одного клиентского приложения в среде разработки Visual Studio Code с использованием Node, сборщика Webpack и библиотеки React.

Проведен контроль качества разработанного приложения с помощью функционального и кросс-браузерного тестирования. Выполнен подсчет метрик Холстеда и цикломатической сложности.

Таким образом, в результате выполнения выпускной квалификационной работы разработан персональный сайт-портфолио для позиционирования в профессиональной среде, обеспечивающий взаимодействие между клиентом и специалистом, позволяющий собрать структурированный и эстетичный

портфель работ, отражающих навыки специалиста. Разработанное ПО соответствует современным стандартам качества, отвечает актуальным требованиям в сфере программирования.

Список использованных источников

1. Шарипова, И.И. Создание и продвижение персонального бренда на региональном рынке практической психологии / И.И. Шарипова // Вестник Омского государственного педагогического университета. Серия: Психологические науки – 2020 – №4(32). – С. 140 – 146.
2. Диаграмма вариантов использования: учебно-методическое пособие по дисциплине «Современные информационные технологии». – Пенза: Изд-во Пензенского государственного университета. – 14 с.
3. MySQL: что это, как устроена, чем хороша и как начать с ней работать // SKILLBOX [Электронный ресурс]. URL: <https://skillbox.ru/media/code/mysql-cto-eto-kak-ustroena-chem-khorosha-i-kak-nachat-s-ney-rabotat/> (дата обращения: 14.05.2025).
4. REST API // Skillfactory Media. [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/rest-api/> (дата обращения: 14.05.2025).
5. Построение диаграмм // VISUAL-PARADIGM [Электронный ресурс]. URL: <https://online.visual-paradigm.com/ru/diagrams/> (дата обращения: 15.05.2025).
6. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов. – Рн/Д: Феникс, 2018. –102 с.
7. Катышева, М.А., Кирюхин, Ю.Г., Маслов, В.А. Основы проектной деятельности: Лабораторный практикум / М.А. Катышева, Ю.Г. Кирюхин, В.А. Маслов. – Пенза: Изд-во Пензенского государственного университета, 2022. – 21 с.
8. Создаем диаграмму Ганта // UNISENDER [Электронный ресурс] URL: <https://www.unisender.com/ru/blog/cto-takoe-diagramma-ganta-i-kak-ee-sdelat/> (Дата обращения: 15.05.2025).
9. REST API // SKILLFACTORY [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/rest-api/> (дата обращения: 15.05.2025).

10. Урок по диаграммам развертывания // LUCIDCHART [Электронный ресурс]. URL: <https://www.lucidchart.com/pages/ru/uml-deployment-diagram> (дата обращения: 16.05.2025).
11. Рудь В. Концептуальное, логическое и физическое моделирование данных // MARCUS-AURELIUS [Электронный ресурс]. URL: https://marcus-aurelius.ru/articles/Data_modeling.html (дата обращения 16.05.2025).
12. Дарков, А.В. Информационные технологии: Учебное пособие / А.В. Дарков, Н.Н. Шапошников. – СПб.: Лань, 2016. – 448 с.
13. Использование диаграммы классов UML при проектировании и документировании программного обеспечения. // ХАБР [Электронный ресурс]. URL: <https://habr.com/ru/articles/572234/> (дата обращения: 20.05.2025).
14. Физическая модель данных // STUDFILE [Электронный ресурс]. URL: <https://studfile.net/preview/6337934/> (дата обращения 29.05.2025).
15. Репозиторий кода Portfolio // GITHUB. [Электронный ресурс]. URL: <https://github.com/Poli00602/Portfolio.git> (дата обращения: 05.06.2025).
16. Что такое фреймворк Express.js и как его использовать // SKYPRO [Электронный ресурс] URL: <https://sky.pro/media/chto-takoe-express-js-i-kak-ego-ispolzovat/> (дата обращения: 29.05.2025).
17. The Best Toast in Town [Электронный ресурс]. URL: <https://react-hot-toast.com/> (дата обращения 04.06.2025)
18. Теория тестирования ПО просто и понятно // ХАБР [Электронный ресурс] URL: <https://habr.com/ru/articles/587620/> (дата обращения 04.06.2025)
19. Обзорная статья о видах тестирования IT-продуктов // ХАБР [Электронный ресурс] URL: <https://habr.com/ru/companies/otus/articles/720664/> (дата обращения 04.06.2025)
20. Что такое кроссбраузерное тестирование // SKYPRO [Электронный ресурс] URL: <https://sky.pro/wiki/javascript/chto-takoe-krossbrauzernoe-testirovanie/> (дата обращения: 04.06.2025).
21. Draw.io // SKILLFACTORY MEDIA [Электронный ресурс]. URL: <https://blog.skillfactory.ru/draw-io/> (дата обращения: 04.06.2025).

Приложение А
(справочное)
Глоссарий

БД (База данных) – это имеющая название совокупность данных, которая отражает состояние объектов и их отношений в рассматриваемой предметной области.

СУБД (Система управления базами данных) – это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

REST (Representational State Transfer) – это архитектурная концепция, определяющая набор ограничений для построения масштабируемых распределенных систем. Основными принципами REST является ресурсно-ориентированная архитектура, использование стандартных HTTP-методов для манипуляции ресурсами, передача данных без сохранения состояния на сервере, унифицированный интерфейс доступа.

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

MySQL – это свободная, мощная реляционная система управления данными, основанная на объектно-ориентированной парадигме, с открытым исходным кодом, работающая на языке SQL.

SQL – это декларативный язык программирования, применяющийся для работы с данными в контексте реляционных баз данных.

Мессенджер – программа для быстрого обмена сообщениями между пользователями, чаще всего через сеть интернет.

Фреймворк – программная платформа, задающая архитектурные принципы и общие правила разработки приложений в определенной предметной области.

Веб-браузер – прикладное программное обеспечение для поиска и просмотра web-страниц, а также управления web-приложениями;

Веб-приложение — это клиент-серверное программное решение, доступ к которому осуществляется через сетевой браузер посредством передачи данных по протоколу HTTP, размещается на удаленном сервере

JSON – JavaScript Object Notation, легковесный текстовый формат обмена данными, полностью независимый от языка реализации.

MVC – это архитектурный паттерн, предписывающий разделение программного кода на три взаимосвязанных компонента: модель данных, представление для визуализации информации и контроллер для обработки пользовательских запросов.

CRUD-операции – это сокращение от четырех основных операций, используемых при работе с данными: Create (создать), Read (прочитать), Update (обновить), Delete (удалить).

Приложение Б
(рекомендуемое)
Атрибуты сущностей базы данных

Таблица Б.1 – Атрибуты сущности «request»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
id	Уникальный идентификатор заявки	int	11	00000000001 - 99999999999	нет	РК	1234
nameCompany	Название компании клиента	varchar	255	Нет ограничений	да		ИП «Кремлевский дворик»
nameClient	Имя фамилия представителя клиента	varchar	255	a-Я a-Z	нет		Иван Данилов
phone	Телефонный номер	varchar	11	0 - 9	нет		7 999 999 99 99
linkSocial	Ссылки на социальные сети и/или мессенджеры клиента	varchar	255	a-Z 0 - 9	да		https://vk.com/grosh06
servicesid	Идентификатор услуги, которая выбрана в заказе	varchar	600	a-Я a-Z	нет	FK	Логотип
price	Приблизительная стоимость работы дизайнера за выбранные услуги	int	6	0 - 9	нет		15000
edits	Выбор количества правок	varchar	60	a-Я 0 - 9	нет		3 круга правок в рамках технического задания
urgency	Срочность заказа	varchar	3	a-Я	нет		Да
status	Статус заявки «Не начато» / «В работе» / «Готово»	varchar	10	a-Я	нет		Готово
description	Описание услуги	varchar	1000	a-Я a-Z 0 - 9 ,;.:/()[]	нет		Рекламные материалы (от 2 дней)

Таблица Б.2 – Атрибуты сущности «service»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
id	Уникальный идентификатор услуги	int	11	000000000001 - 999999999999	нет	РК	1234
nameService	Название услуги	varchar	255	а-Я а-Z ,	нет		Логотип
price	Цена предоставляемой услуги	int	6	0 - 9	нет		10000
description	Описание услуги	varchar	1000	а-Я а-Z 0 - 9 ,:;./()[]	нет		3 варианта, чб-варианты, презентация, мокапы

Таблица Б.3 – Атрибуты сущности «work»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
id	Уникальный идентификатор работы	int	11	000000000001 - 999999999999	нет	РК	1234
nameWork	Название работы	varchar	255	а-Я а-Z ,/	нет		Логотип «Спасение надежды»
subtitle	Подзаголовок	varchar	255	а-Я а-Z ,/	нет		Для компании волонтеров
description	Описание работы	varchar	1000	Нет ограничений	нет		Задачи: требовалось составить эмблемный логотип с фирменными цветами компании.
categoryWorkid	Идентификатор категории работы	varchar	600	а-Я а-Z	нет	FK	Логотип

Таблица Б.4 – Атрибуты сущности «categoryWork»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
id	Уникальный идентификатор категории работ	int	11	000000000001 - 999999999999	нет	РК	1234
nameCategory	Название категории	varchar	255	а-Я а-Z ,/	нет		Полиграфия

Приложение В
(обязательное)
Листинг серверной части приложения

Листинг В.1 – Файл auth.controller.js

```
// импортируем модели баз данных, ключ для генерации jwt токена
const db = require("../models");
const config = require("../config/auth.config");
const User = db.user;

// jwt - токен авторизации
var jwt = require("jsonwebtoken");
// шифровальщик для пароля в таблице users
var bcrypt = require("bcryptjs");

// авторизация
exports.signin = (req, res) => {
  // находим пользователя по его имени, чтоб проверить его наличие
  // в БД
  User.findOne({
    where: {
      username: req.body.username
    }
  })
  .then(user => {
    if (!user) {
      return res.status(404).send({ message: "Пользователь не
найден." });
    }
    // дешифруем пароль и проверяем его с введенным
    var passwordIsValid = bcrypt.compareSync(
      req.body.password,
      user.password
    );

    if (!passwordIsValid) {
      return res.status(401).send({
        accessToken: null,
        message: "Неверный пароль!"
      });
    }
    // создаем токен jwt для сессии пользователя
    const token = jwt.sign({ id: user.id },
      config.secret,
      {
        algorithm: 'HS256',
        allowInsecureKeySizes: true,
        expiresIn: 86400, // через 24 часа выйдет из аккаунта
      });
    // успех Код 200
    res.status(200).send({
      id: user.id,
      username: user.username,
      accessToken: token
    });
  });
};
```

```

    })
    .catch(err => {
      res.status(500).send({ message: err.message });
    });
  };
  // фрагмент

```

Листинг В.2 – Файл categoryWorks.controller.js

```

// импорт модели бд, бд Категорий работ, функцию библиотеки для
// работы с БД
const db = require("../models");
const CategoryWorks = db.categoryWorks;
const Op = db.Sequelize.Op;

// создаем категорию работ
exports.create = (req, res) => {
  // проверяем что ввели имя категории работ на клиенте, чтобы
  // значение не было пустым
  if (!req.body.nameCategoryWorks) {
    res.status(400).send({
      message: " Содержимое не может быть пустым!"
    });
    return;
  }

  // получим значения имени категории работ с клиента
  const work = {
    nameCategoryWorks: req.body.nameCategoryWorks,
  };

  // сохраняем категорию работ в БД
  CategoryWorks.create(work)
    .then(data => {
      res.send(data);
    })
    .catch(err => {
      res.status(500).send({
        message:
          err.message || "Произошла ошибка при создании
CategoryWorks."
      });
    });
};

// получим все категории работ, по конкретной работе
exports.findAll = (req, res) => {
  const nameCategoryWorks = req.query.nameCategoryWorks;
  var condition = nameCategoryWorks ? { nameCategoryWorks: {
    [Op.like]: `%${nameCategoryWorks}%` } } : null;

  CategoryWorks.findAll({ where: condition })

```

```

    .then(data => {
      res.send(data);
    })
    .catch(err => {
      res.status(500).send({
        message:
          err.message || "Произошла ошибка при извлечении
CategoryWorks."
      });
    });
  });
};
// фрагмент

```

Листинг В.3 – Файл request.controller.js

```

const db = require("../models");
const Request = db.requests;
const Op = db.Sequelize.Op;

// создадим заявку
exports.create = (req, res) => {
  // проверим что заполнено имя заказчика (идентификатор заявки в
будущем)
  if (!req.body.nameClient) {
    res.status(400).send({
      message: "Содержимое не может быть пустым!"
    });
    return;
  }

  // получим значения заявки с клиента
  const request = {
    nameCompany: req.body.nameCompany,
    nameClient: req.body.nameClient,
    phone: req.body.phone,
    linkSocial: req.body.linkSocial,
    services: req.body.services,
    price: req.body.price,
    edits: req.body.edits,
    urgency: req.body.urgency,
    status: req.body.status,
    description: req.body.description
  };

  // создадим заявку и запишем в БД
  Request.create(request)
    .then(data => {
      res.send(data);
    })
    .catch(err => {
      res.status(500).send({
        message:

```

```

        err.message || "Произошла ошибка при создании запроса."
    });
});
};

// найдем все зайвки по имени заказчика
exports.findAll = (req, res) => {
    const nameClient = req.query.nameClient;
    var condition = nameClient ? { nameClient: { [Op.like]:
`%${nameClient}%` } } : null;

    Request.findAll({ where: condition })
        .then(data => {
            res.send(data);
        })
        .catch(err => {
            res.status(500).send({
                message:
                    err.message || "Произошла ошибка при получении
запросов."
            });
        });
};

// найдем заявку по id в БД
exports.findOne = (req, res) => {
    const id = req.params.id;

    Request.findByPk(id)
        .then(data => {
            if (data) {
                res.send(data);
            } else {
                res.status(404).send({
                    message: `Невозможно найти заявку с id=${id}.`
                });
            }
        })
        .catch(err => {
            res.status(500).send({
                message: "Ошибка при получении заявки с id=" + id
            });
        });
};

// обновление текущей редактируемой заявки в БД
exports.update = (req, res) => {
    const id = req.params.id;

    Request.update(req.body, {
        where: { id: id }
    })

```

```

    })
    .then(num => {
      if (num == 1) {
        res.send({
          message: "Заявка успешно обновлена."
        });
      }
    });
// фрагмент

```

Листинг В.4 – Файл service.controller.js

```

const db = require("../models");
const Service = db.services;
const Op = db.Sequelize.Op;

// создадим новую услугу
exports.create = (req, res) => {
  // проверим что введено имя услуги
  if (!req.body.nameService) {
    res.status(400).send({
      message: "Имя не может быть пустым!"
    });
    return;
  }

  // создадим услугу
  const service = {
    nameService: req.body.nameService,
    price: req.body.price,
    description: req.body.description
  };

  // создадим услугу и запишем в БД
  Service.create(service)
    .then(data => {
      res.send(data);
    })
    .catch(err => {
      res.status(500).send({
        message:
          err.message || "Произошла ошибка при создании услуги."
      });
    });
};

// получим все услуги
exports.findAll = (req, res) => {
  const nameService = req.query.nameService;
  var condition = nameService ? { nameService: { [Op.like]:
    `%${nameService}%` } } : null;

  Service.findAll({ where: condition })
    .then(data => {

```

```

        res.send(data);
    })
    .catch(err => {
        res.status(500).send({
            message:
                err.message || "Произошла ошибка при получении услуг."
        });
    });
};

// найдем услугу по id
exports.findOne = (req, res) => {
    const id = req.params.id;

    Service.findByPk(id)
        .then(data => {
            if (data) {
                res.send(data);
            } else {
                res.status(404).send({
                    message: `Невозможно найти услуг с id=${id}.`
                });
            }
        })
        .catch(err => {
// фрагмент

```

Листинг В.5 – Файл work.controller.js

```

const db = require("../models");
const Work = db.works;
const Op = db.Sequelize.Op;

// Создание и сохранение новой работы
exports.create = (req, res) => {
    // Validate work
    if (!req.body.nameWork) {
        res.status(400).send({
            message: "Имя не может быть пустым!"
        });
        return;
    }

    // Создание
    const work = {
        category: req.body.category,
        nameWork: req.body.nameWork,
        subtitle: req.body.subtitle,
        base64: req.body.base64,
        description: req.body.description
    };

```

```

// Сохранение в бд
Work.create(work)
  .then(data => {
    res.send(data);
  })
  .catch(err => {
    res.status(500).send({
      message:
        err.message || "Произошла ошибка при создании работы."
    });
  });
};

// Получение всех работ из БД
exports.findAll = (req, res) => {
  const nameWork = req.query.nameWork;
  var condition = nameWork ? { nameWork: { [Op.like]:
    `%${nameWork}%` } } : null;

  Work.findAll({ where: condition })
    .then(data => {
      res.send(data);
    })
    .catch(err => {
      res.status(500).send({
        message:
          err.message || "Произошла ошибка при получении работ."
      });
    });
};

// Поиск одной работы по id
exports.findOne = (req, res) => {
  const id = req.params.id;

  Work.findByPk(id)
    .then(data => {
      if (data) {
        res.send(data);
      } else {
        res.status(404).send({
          message: `Невозможно найти работы с id=${id}.`
        });
      }
    })
    .catch(err => {
      res.status(500).send({
        message: "Ошибка получения работы с with id=" + id
      });
    });
};
}; // фрагмент

```

Листинг В.6 – Файл workImages.controller.js

```

const uploadFile = require("../middleware/upload");
const fs = require("fs");
// ссылка для api файлов
const baseUrl = "http://localhost:8081/api/files/";

// загрузка изображений
exports.upload = async (req, res) => {
  try {
    await uploadFile(req, res);

    if (req.file == undefined) {
      return res.status(400).send({ message: "Пожалуйста загрузите файл!" });
    }

    res.status(200).send({
      message: "Файл успешно загружен: " + req.file.originalname,
    });
  } catch (err) {
    console.log(err);
    // ошибка, размер файла больше допустимого
    if (err.code == "LIMIT_FILE_SIZE") {
      return res.status(500).send({
        message: "Размер файла превышает 10MB!",
      });
    }

    res.status(500).send({
      message: `Не удалось загрузить файл: ${req.file.originalname}. ${err}`,
    });
  }
};

// получить все файлы
exports.getListFiles = (req, res) => {
  const directoryPath = __basedir +
    "/resources/static/assets/uploads/";

  fs.readdir(directoryPath, function (err, files) {
    if (err) {
      res.status(500).send({
        message: "Unable to scan files!",
      });
    }

    let fileInfos = [];

    files.forEach((file) => {
      fileInfos.push({

```

```

        name: file,
        url: baseUrl + file,
    });
});

res.status(200).send(fileInfos);
});
};

// загрузить изображения
exports.download = (req, res) => {
    const fileName = req.params.name;
    const directoryPath = __basedir +
"/resources/static/assets/uploads/";

    res.download(directoryPath + fileName, fileName, (err) => {
        if (err) {
            res.status(500).send({
                message: "Не удалось загрузить файл. " + err,
            });
        }
    });
};

// фрагмент

```

Приложение Г
(обязательное)
Листинг клиентской части приложения

Листинг Г.1 – Файл AdminRequest.jsx

```
// страница заявок для администратора
// импортируем необходимые зависимости/компоненты
import React, { Component } from "react";
import { Link } from 'react-router-dom';
import NavBar from '../components/NavBar'
// сервис для заявок
import RequestDataService from "../services/request.service.js";
import '../styles/Root.css';
import '../styles/AdminRequests.css';

export default class AddRequest extends Component {
  constructor(props) {
    super(props);
    // инициализация функций
    this.setSelectedStatus = this.setSelectedStatus.bind(this);
    this.setSelectedUrgency = this.setSelectedUrgency.bind(this);
    this.retrieveRequest = this.retrieveRequest.bind(this);
    // переменные
    this.state = {
      checked: false,
      requests: [],
      selectedStatus: "Не начато",
      selectedUrgency: "Да"
    };
  }

  // при загрузке страницы
  componentDidMount() {
    // получим все заявки
    this.retrieveRequest();
  }

  // фильтр установить статус
  setSelectedStatus(e) {
    this.setState({
      selectedStatus: e.target.value
    });
  }

  // фильтр установить срочность
  setSelectedUrgency(e) {
    this.setState({
      selectedUrgency: e.target.value
    });
  }

  // при изменении фильтров
  onChangeFilters = (e) => {
    const { checked } = e.target
```

```

    // если флаг активен, то применим фильтры
    this.setState({
      checked: checked
    })
    // перезагрузим заявки
    this.retrieveRequest();
  }

  // получить все заявки (сервис)
  retrieveRequest() {
    RequestDataService.getAll()
      .then(response => {
        this.setState({
          requests: response.data
        });
      });
  }
// фрагмент

```

Листинг Г.2 – Файл Login.jsx

```

import React, { Component } from "react";
import { withRouter } from '../common/with-router';
import toast from 'react-hot-toast';
// для валидации формы
import Form from "react-validation/build/form";
// поля ввода для валидации и проверок
import Input from "react-validation/build/input";
// проверочная кнопка
import CheckButton from "react-validation/build/button";
// сервис для авторизации
import AuthService from '../services/auth.service';

// обязательное поле проверка
const required = value => {
  if (!value) {
    return (
      <div className="alert alert-danger" role="alert">
        Обязательное поле
      </div>
    )
  }
}

class Login extends Component {
  constructor(props) {
    super(props);
    // инициализация функций
    this.handleLogin = this.handleLogin.bind(this);
    this.onChangeUsername = this.onChangeUsername.bind(this);
    this.onChangePassword = this.onChangePassword.bind(this);
    // переменные
    this.state = {

```

```

        username: "",
        password: "",
        loading: false,
        message: ""
    }
}

// при изменении имени
onChangeUsername(e) {
    this.setState({
        username: e.target.value
    })
}

// при изменении пароля
onChangePassword(e) {
    this.setState({
        password: e.target.value
    })
}

// при нажатии авторизация
handleLogin(e) {
    e.preventDefault();
    this.setState({
        message: "",
        loading: true
    })

    // валидируем имя и пароль
    this.form.validateAll();

    // если ошибок нет, то с помощью сервиса авторизируем
    пользователя
    if (this.checkBtn.context._errors.length === 0) {
        AuthService.login(this.state.username,
        this.state.password).then(
            () => {
                this.props.router.navigate('/requests');
                window.location.reload();
            }
        )
    }
}
// фрагмент

```

Листинг Г.3 – Файл MainPage.jsx

```

import React, { Component } from "react";
import CategoryWorksData from
"../services/categoryWorks.service.js";
import NavBar from '../components/NavBar'
import Footer from '../components/Footer'
import { Link } from 'react-router-dom';
import '../styles/Root.css';

```

```

import '../styles/MainPage.css';

// изображения на странице
import PhotoShe1 from '../assets/img/block_about/me.png';
import PhotoShe2 from '../assets/img/block_questions/Me.png';
import Education from '../assets/img/block_about/edication.png';
import Skills from '../assets/img/block_about/skills.png';

export default class AddRequest extends Component {
  constructor(props) {
    super(props);

    this.state = {
      categoryWorks: [],

      submitted: false
    };
  }

  componentDidMount() {
    // получим массив всех категорий
    this.retrieveCategoryWorks();
  }

  retrieveCategoryWorks() {
    CategoryWorksData.getAll()
      .then(response => {
        this.setState({
          categoryWorks: response.data
        });
        console.log(response.data);
      })
      .catch(e => {
        console.log(e);
      });
  }

  render() {
    const { categoryWorks } = this.state;

    return (
      <div className="container">
        <NavBar />
        <main className="page">
          <section className="header__section__container">
            <div className="header__block">
              <div className="header__body">
                <div className="header-body__text-container">
                  <h1>
                    ГРАМОТНЫЙ ДИЗАЙН <br />

```

// фрагмент

Листинг Г.4 – Файл NewRequest.jsx

```
// страница новой заявки
import React, { Component } from "react";
import toast from 'react-hot-toast';
import Form from "react-validation/build/form";
import Input from "react-validation/build/input";
import CheckButton from "react-validation/build/button";
import RequestDataService from "../services/request.service.js";
import ServicesDataService from "../services/services.service.js";
import NavBar from '../components/NavBar'

// styles
import '../styles/Root.css';
import '../styles/Request.css';

const required = value => {
  if (!value) {
    return (
      <div className="alert alert-danger" role="alert">
        Обязательное поле
      </div>
    )
  }
}

const vnameclient = value => {
  const rusRegex = /^[a-яё]/; // регулярное выражение на русс.
  буквы
  if (!rusRegex.test(value) || value.length < 5 || value.length >
30) {
    return (
      <div className="alert alert-danger" role="alert">
        Только русские буквы (от 5 до 30 символов)
      </div>
    )
  }
}

const vphone = value => {
  const phoneRegex = /^\+7\d{10}$/; // регулярное выражение для
10-значного номера телефона
  if (!phoneRegex.test(value)) {
    return (
      <div className="alert alert-danger" role="alert">
        Формат номера: +7XXXXXXXXXX
      </div>
    )
  }
}
```

```

var Summ = 0
var fullSumm = 0

export default class AddRequest extends Component {
  constructor(props) {
    super(props);
    // инициализация функций
    this.handleNewRequest = this.handleNewRequest.bind(this);
    this.onChangeNameCompany =
this.onChangeNameCompany.bind(this);
    this.onChangeNameClient = this.onChangeNameClient.bind(this);
    this.onChangePhone = this.onChangePhone.bind(this);
    this.onChangeLinkSocial = this.onChangeLinkSocial.bind(this);
    this.onChangeEdits = this.onChangeEdits.bind(this);
    this.onChangeUrgency = this.onChangeUrgency.bind(this);
    this.onChangeDescription =
this.onChangeDescription.bind(this);
    // this.saveRequest = this.saveRequest.bind(this);
    this.newRequest = this.newRequest.bind(this);

    // начальные значения
    this.state = {
      listServices: [],
      selectServices: "",
      priceSumm: 0,
      edits1: "3 круга правок в рамках технического задания",
      edits2: "5 кругов правок в рамках технического задания",
      urgency1: "Да",
      urgency2: "Нет",
      id: null,
      nameCompany: "",
      nameClient: "",
      phone: "",
      linkSocial: "",
      services: "",
      price: "",
      edits: "",
      urgency: "",
      status: "Не начато",
      description: "",

      submitted: false
    };
  }

  // заполняем строку отмеченными услугами
  onChangeService(index, e) {
    let strService;
    let intServiceSumm = 0;
    let selectServices = this.state.listServices.slice();
    selectServices[index].checked = !selectServices[index].checked
  }
}

```

```

        selectServices.filter(item =>
item.checked).map((selectService, index) => (
    strService = strService + " (" + selectService.nameService +
"): " + selectService.price + "₽" + ", "
    ))
    if (strService !== undefined) {
        strService = strService.replace('undefined', '');
    }
    this.state.selectServices = strService
    selectServices.filter(item =>
item.checked).map((selectService, index) => (
    intServiceSumm = intServiceSumm + selectService.price
    ))
    Summ = intServiceSumm
    this.onChangeSumm(this.state.urgency)
}

onChangeSumm(urgency) {
    if (urgency == "Да") {
        fullSumm = Summ * 2
    }
    if (urgency == "Нет") {
        fullSumm = Summ
    }
    this.setState({
        priceSumm: fullSumm
    });
    console.log(fullSumm)
}
// фрагмент

```

Листинг Г.5 – Файл Service.jsx

```

import React, { Component } from "react";
import toast from 'react-hot-toast';
import ServiceDataService from "../services/services.service.js";
import NavBar from '../components/NavBar'
import Footer from '../components/Footer'
import { Link } from 'react-router-dom';
import '../styles/Root.css';
import '../styles/Service.css';
import { withRouter } from '../common/with-router';
// импортируем компонент для авторизованного пользователя
import AuthService from '../services/auth.service';

class Service extends Component {
    constructor(props) {
        super(props);
        this.onChangeNameService =
this.onChangeNameService.bind(this);

```

```

    this.onChangeDescription =
this.onChangeDescription.bind(this);
    this.onChangePrice = this.onChangePrice.bind(this);
    this.saveService = this.saveService.bind(this);
    this.newService = this.newService.bind(this);
    this.getAdminUser = this.getAdminUser.bind(this);

    this.state = {
      apiData: false,
      services: [],
      id: null,
      nameService: "",
      description: "",
      price: ""
    };
  }

  componentDidMount() {
    // получаем права администратора, чтобы скрывать/показывать
элементы
    this.getAdminUser(AuthService.getCurrentUser());
    // получим массив всех услуг
    this.retrieveServices();
  }

  getAdminUser(admin) {
    if (admin) {
      this.setState({
        apiData: true
      });
    }
  }

  retrieveServices() {
    ServiceDataService.getAll()
      .then(response => {
        this.setState({
          services: response.data
        });
        console.log(response.data);
      })
      .catch(e => {
        console.log(e);
      });
  }

  onChangeNameService(e) {
    this.setState({
      nameService: e.target.value
    });
  }

```

```

onChangeDescription(e) {
  this.setState({
    description: e.target.value
  });
}
onChangePrice(e) {
  this.setState({
    price: e.target.value
  });
}

deleteService(id) {
  ServiceDataService.delete(id)
    .then(response => {
      console.log(response.data);
      this.retrieveServices();
      toast.error("Услуга удалена!")
    })
}
// фрагмент

```

Листинг Г.6 – Файл Works.jsx

```

import React, { Component } from "react";
import toast from 'react-hot-toast';
import WorksData from "../services/works.service";
import workImagesData from "../services/workImages.service";
import NavBar from '../components/NavBar'
import Footer from '../components/Footer'
import { Link } from 'react-router-dom';
import '../styles/Root.css';
import '../styles/Works.css';
import { withRouter } from '../common/with-router';
import AuthService from '../services/auth.service';

class Works extends Component {
  constructor(props) {
    super(props);
    this.onChangeNameWorks = this.onChangeNameWorks.bind(this);
    this.onChangeSubtitle = this.onChangeSubtitle.bind(this);
    this.onChangeDescription =
this.onChangeDescription.bind(this);
    this.saveWorks = this.saveWorks.bind(this);
    this.newWorks = this.newWorks.bind(this);
    this.getAdminUser = this.getAdminUser.bind(this);
    this.deleteImagesWorks = this.deleteImagesWorks.bind(this);

    this.state = {
      apiData: false,
      works: [],
      selectedFiles: undefined,
      imagePreviews: [],

```

```

        imageInfos: [],
        nameCategoryWorks: "",

        id: null,
        nameWork: "",
        subtitle: "",
        category: "",
        description: ""
    };
}

componentDidMount() {
    // получаем права администратора, чтобы скрывать/показывать
элементы
    this.getAdminUser(AuthService.getCurrentUser());
    // получим массив всех работ по категории
    this.retrieveWorks();
    // получим категорию работ
    this.setState({
        nameCategoryWorks:
this.props.router.params.nameCategoryWorks
    });
    // получим все картинки работы с каталога в папке сервера
    workImagesData.GetFiles().then((response) => {
        this.setState({
            imageInfos: response.data
        });
    });
}

getAdminUser(admin) {
    if (admin) {
// фрагмент

```

Листинг Г.7 – Файл WorksEdit.jsx

```

import React, { Component } from "react";
import toast from 'react-hot-toast';
import WorksData from "../services/works.service.js";
import NavBar from '../components/NavBar'
import WorkImagesData from "../services/workImages.service";
import '../styles/Root.css';
import '../styles/Works.css';
import { withRouter } from '../common/with-router';

class Works extends Component {
    constructor(props) {
        super(props);
        this.onChangeNameWork = this.onChangeNameWork.bind(this);
        this.onChangeDescription =
this.onChangeDescription.bind(this);

```

```

this.onChangeCategory = this.onChangeCategory.bind(this);
this.onChangeSubtitle = this.onChangeSubtitle.bind(this);
this.getWorks = this.getWorks.bind(this);
this.updateWorks = this.updateWorks.bind(this);
this.deleteWorks = this.deleteWorks.bind(this);
this.fileToDataUri = this.fileToDataUri.bind(this);
this.selectFiles = this.selectFiles.bind(this);
this.uploadImages = this.uploadImages.bind(this);
this.upload = this.upload.bind(this);
this.deleteImagesWorks = this.deleteImagesWorks.bind(this);

this.state = {
  selectedFiles: undefined,
  imagePreviews: [],
  imageInfos: [],
  currentWorks: {

    id: null,
    nameWork: "",
    subtitle: "",
    category: "",
    description: ""
  },
  message: ""
};
}

onChangeNameWork(e) {
  const nameWork = e.target.value;
  this.setState(function (prevState) {
    return {
      currentWorks: {
        ...prevState.currentWorks,
        nameWork: nameWork
      }
    };
  });
}

onChangeSubtitle(e) {
  const subtitle = e.target.value;
  this.setState(prevState => ({
    currentWorks: {
      ...prevState.currentWorks,
      subtitle: subtitle
    }
  }));
}

onChangeDescription(e) {
  const description = e.target.value;
  this.setState(prevState => ({

```

```

        currentWorks: {
            ...prevState.currentWorks,
            description: description
        }
    }));
}
onChangeCategory(e) {
    const category = e.target.value;

    this.setState(prevState => ({
        currentWorks: {
            ...prevState.currentWorks,
            category: category
        }
    }));
}

onChangeImage(file) {
    if (!file) {
        this.state.selectedImage = ''
        return
    }

    deleteWorks() {
        WorksData.delete(this.state.currentWorks.id)
            .then(response => {
                this.deleteImagesWorks(this.state.currentWorks.id)
                console.log(response.data);
                toast.error("Работа удалена!")
            })
        this.props.router.navigate(`/${this.state.currentWorks.category}/works`);
    }
    .catch(e => {
        toast.error("Что-то пошло не так..")
        console.log(e);
    });
}

deleteImagesWorks(id) {
    this.state.imageInfos.filter(item =>
    item.name.includes("work_ID" + id))
        .map((img, index) => (
            WorkImagesData.deleteFilesWork(img.name)
                .then(response => {
                    console.log(response.data);
                })
                .catch(e => {
                    console.log(e);
                })
        ))
}
} // фрагмент

```

Приложение Д
(рекомендуемое)
Результаты функционального тестирования

[Портфолио](#)
[Прайс-лист услуг](#)
[Обо мне](#)
[Контакты](#)
[Заявки](#)
[Оставить](#)

Хотите поработать со мной? Тогда давайте рассчитаем примерную стоимость!

Заполните необходимые поля и выберите нужные вам услуги ниже

Название компании (Необязательно)

Название компании

ФИО представителя (Обязательно)

ФИО представителя

Рисунок Д.1 – Отображение страницы «Оставить заявку»

☐ Фильтры Статус заказа Не начато ☐ Срочность Да ☐

Грошева Полина Александровна Не начато

Срочный заказ? Нет
Название компании:
Телефон: +79379150072
Соцсети: <https://m.vk.com/grosh06>
Услуги: (Визитка, сертификат): 2000Р, (Логотип): 10000Р,
Правки: 3 круга правок в рамках технического задания
Стоимость: 12000Р

Иванов Василий Александрович Не начато

Срочный заказ? Да
Название компании: Спасение надежды
Телефон: +79273870837
Соцсети:
Услуги: (Флаер, листовка, банер): 2000Р, (Презентация): 450Р,
Правки: 3 круга правок в рамках технического задания
Стоимость: 4900Р

Борисова Валентина Олеговна Не начато

Срочный заказ? Нет
Название компании:
Телефон: +79375092219
Соцсети: <https://m.vk.com/valentunav>
Услуги: (Шаблоны для постов/сторис): 1500Р, (Креативы): 1750Р, (Упаковка соцсетей): 2500Р,
Правки: 3 круга правок в рамках технического задания
Стоимость: 5750Р

Полина В работе

Срочный заказ? Да
Название компании:
Телефон: +79379150072
Соцсети:
Услуги: (Флаер, листовка, банер): 2000Р, (Фирменный стиль): 15000Р,
Правки: 3 круга правок в рамках технического задания
Стоимость: 34000Р

Рисунок Д.2 – Отображение страницы «Заявки»



Ваша заявка оставлена!

Спасибо за заявку!

Рисунок Д.3 – Отображение создания заявки

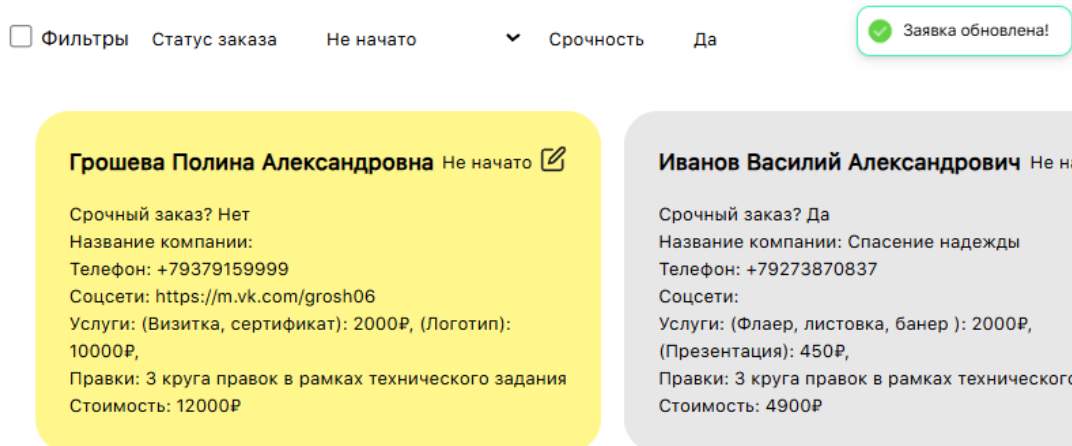


Рисунок Д.4 – Отображение обновления контактного номера в заявке

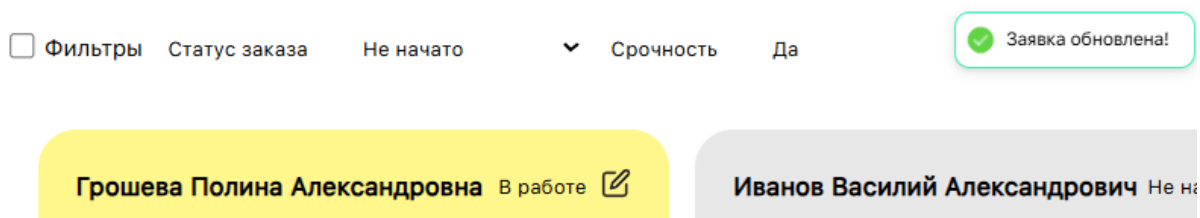


Рисунок Д.5 – Отображение обновления статуса заявки

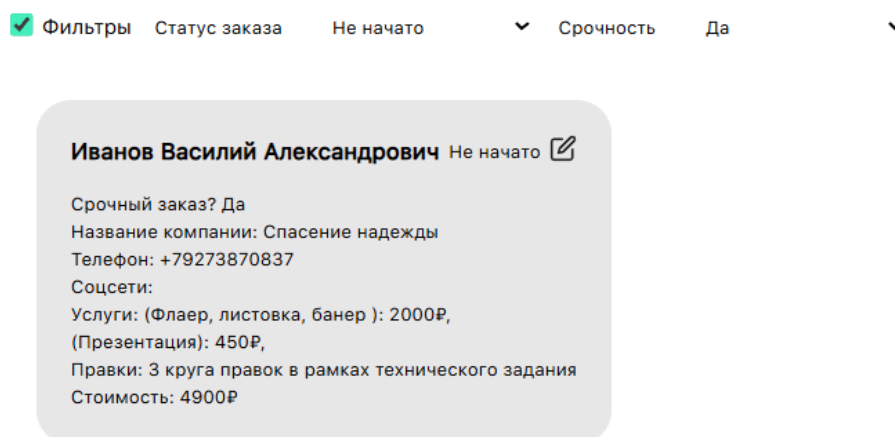


Рисунок Д.6 – Фильтрация заявок по статусу и срочности заказа

☐ Фильтры

Статус заказа

Не начато

▼ Срочность

Да

▼

Грошева Полина Александровна В работе

Срочный заказ? Нет
Название компании:
Телефон: +79379159999
Соцсети: <https://m.vk.com/grosh06>
Услуги: (Визитка, сертификат): 2000Р, (Логотип): 10000Р,
Правки: 3 круга правок в рамках технического задания
Стоимость: 12000Р

Иванов Василий Александрович Не начато

Срочный заказ? Да
Название компании: Спасение надежды
Телефон: +79273870837
Соцсети:
Услуги: (Флаер, листовка, банер): 2000Р,
(Презентация): 450Р,
Правки: 3 круга правок в рамках технического задания
Стоимость: 4900Р

Борисова Валентина

Срочный заказ? Нет
Название компании:
Телефон: +793750922
Соцсети: <https://m.vk.com/grosh06>
Услуги: (Шаблоны для соцсетей): 1750Р, (Услуги копирайтера): 1750Р,
Правки: 3 круга правок в рамках технического задания
Стоимость: 5750Р

Рисунок Д.7 – Удаление заявок

Портфолио

Прайс-лист услуг

Обо мне

Контакты

Заявки

Оставить заявку

Портфолио

логотип

инфографика

полиграфия

фирстиль

креативы

Наименование

Добавить

Рисунок Д.8 – Отображение страницы «Портфолио»

Мои работы



Логотип "Спасение надежды"

ЗАДАЧА: разработать логотип для социального проекта «Спасение надежды». Логотип должен передавать впечатления семьи, помощи, надежды, очага, дома и тепла. Целевая аудитория: кризисные семьи, органы опеки и государственного управления.

Логотип представлен в теплых оттенках синего, красного и зеленого. Синий - цвет силы, опоры; красный - любви, тепла; зеленый - про уют и спокойствие. В центре логотипа изображены сердце и дом, что дает прямую ассоциацию к семье, домашнему очагу. Держащиеся друг за друга руки говорят о помощи и защите семейных ценностей. Также с кругом у людей есть множество ассоциаций: спасательный круг, защитный круг, семейный, солнечный.

Рисунок Д.9 – Отображение работы в категории «Логотипы»



Рисунок Д.10 – Перелистывание изображений работы

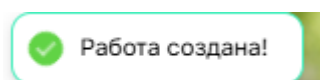


Рисунок Д.11 – Создание работы в портфолио

Работа: Логотип "ССС ФВТ"

Имя работы

Логотип "ССС ФВТ"

Подзаголовок

Создан совместно командой активистов факультета вычислительной техники

Описание

Логотип отражает четкой и сдержанной, присущей техническому факультету, должен быть краток и лаконичен. Будет изображаться на полиграфической продукции, а также использоваться в сети интернет.

Категория работ

ЛОГОТИПЫ

Изображение

[Выбрать файлы](#) Снимок экрана 2025-06-09 035938.png

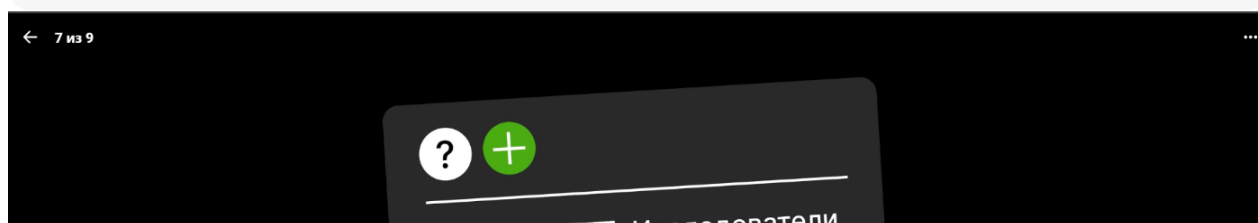


Рисунок Д.12 – Редактирование работы в портфолио 1

Логотип "ССС ФВТ"

Создан совместно командой активистов факультета вычислительной техники

Логотип отражает четкой и сдержанной, присущей техническому факультету, должен быть краток и лаконичен. Будет изображаться на полиграфической продукции, а также использоваться в сети интернет.

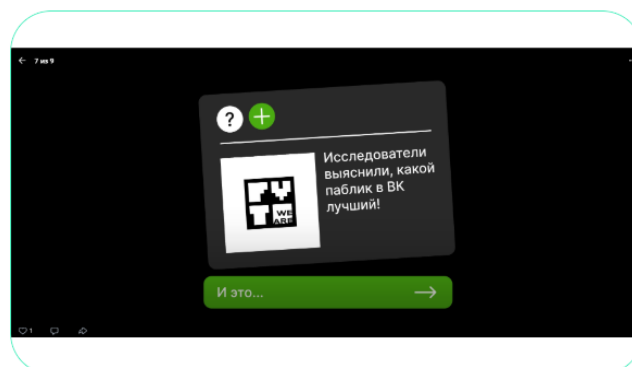


Рисунок Д.13 – Редактирование работы в портфолио 2

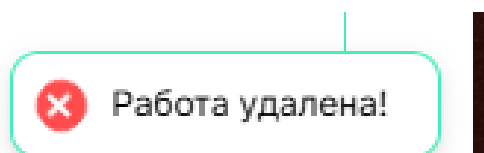


Рисунок Д.14 – Удаление работы из портфолио

Приложение Е
(рекомендуемое)

Результаты кросс-браузерного тестирования

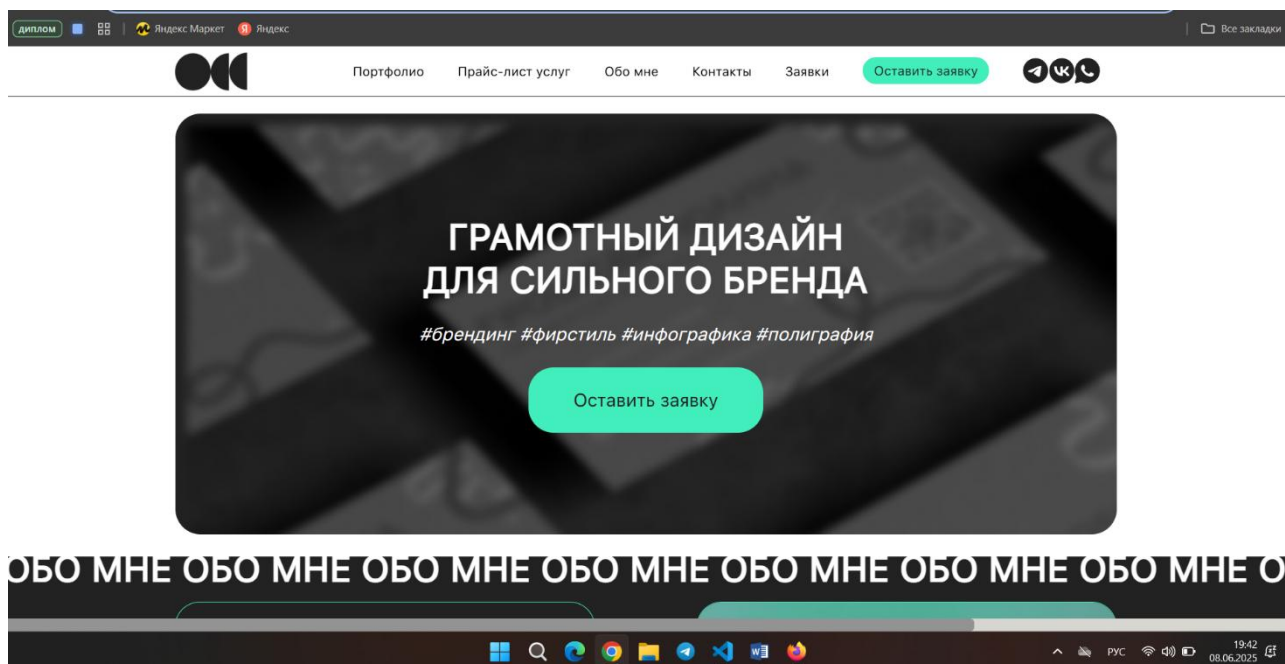


Рисунок Е.1 – Главная страница в Google Chrome

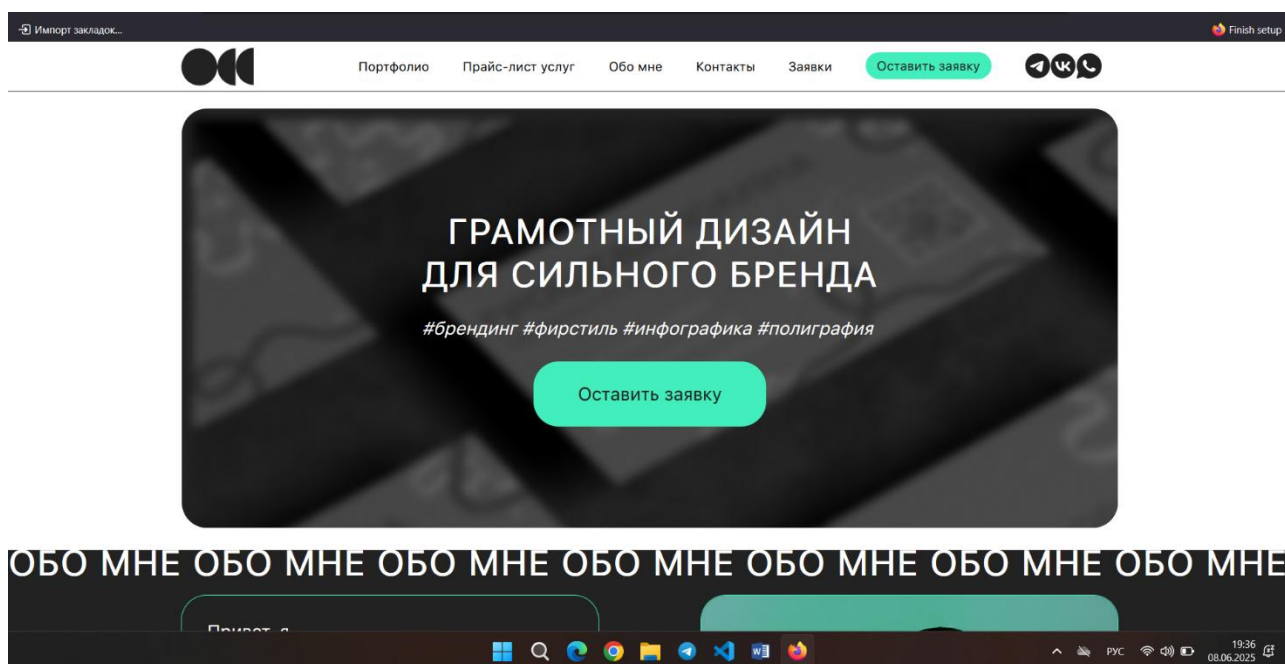


Рисунок Е.2 – Главная страница в Mozilla FireFox

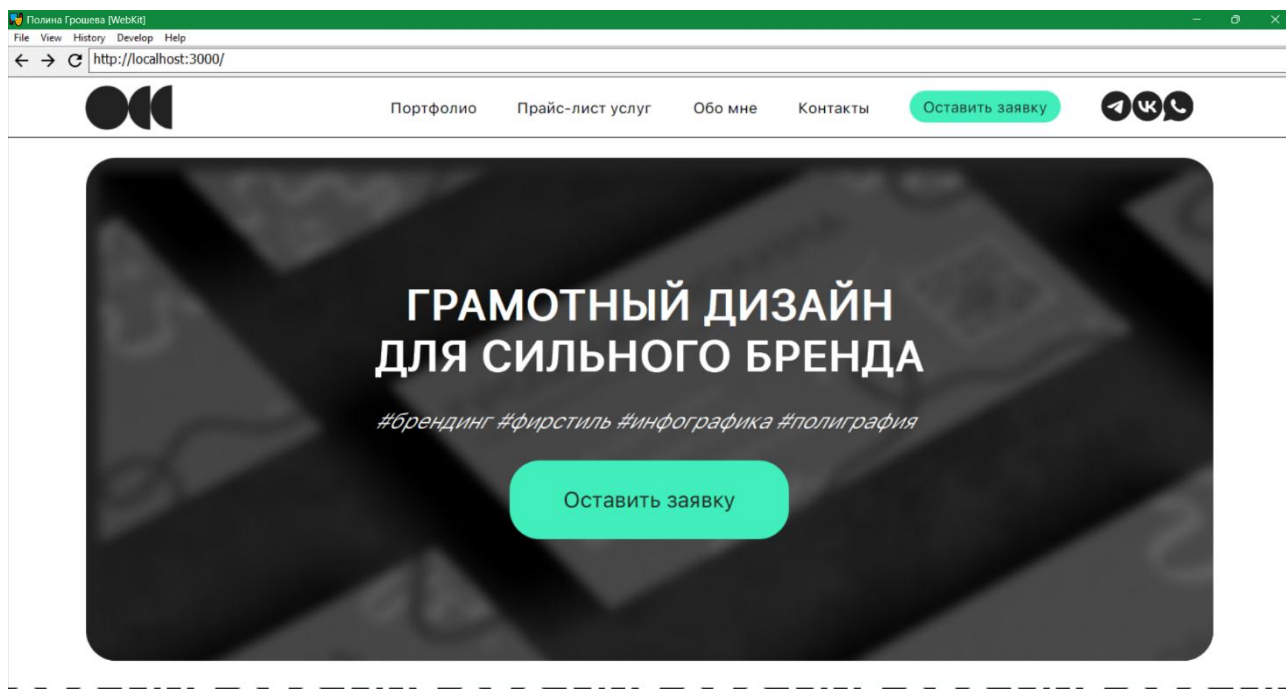


Рисунок Е.3 – Главная страница на движке WebKit

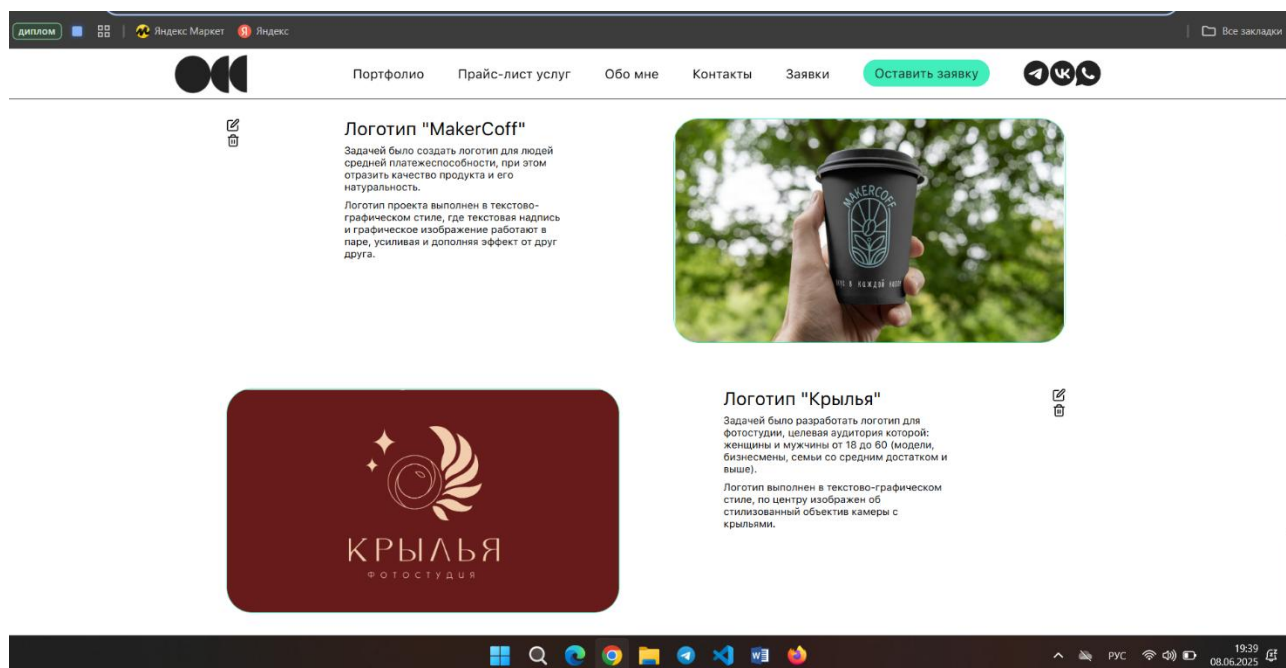


Рисунок Е.4 – Страница портфолио работ в Google Chrome

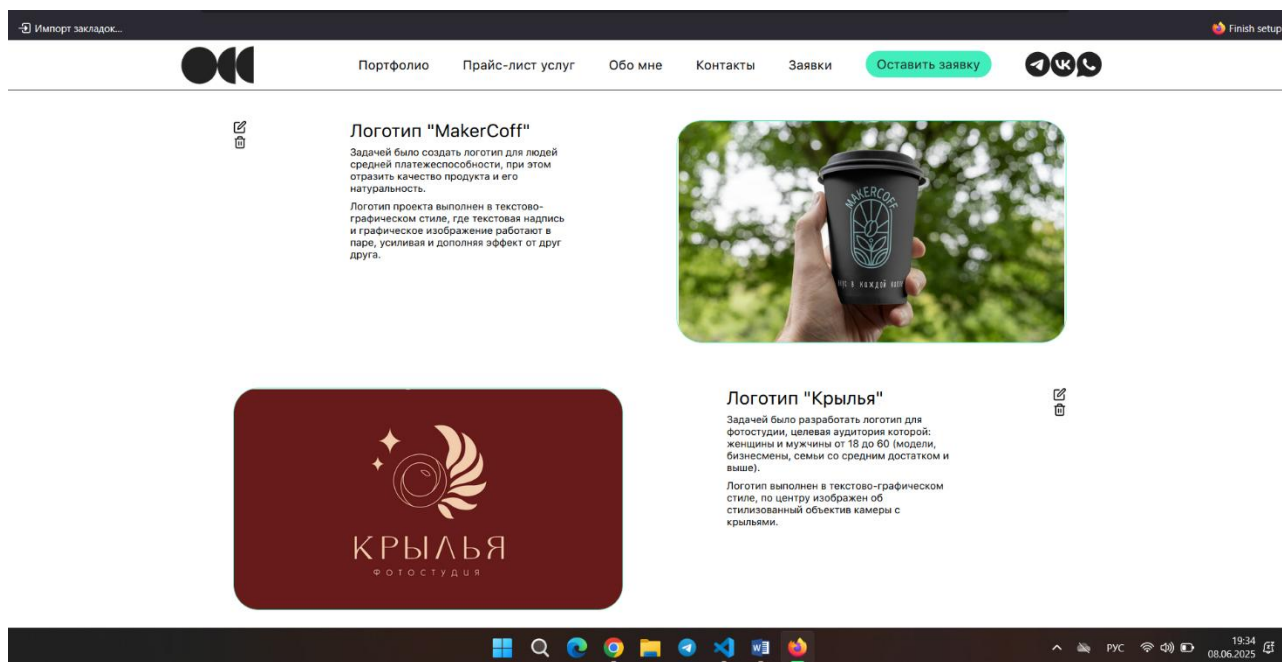


Рисунок Е.5 – Страница портфолио работ в Mozilla FireFox

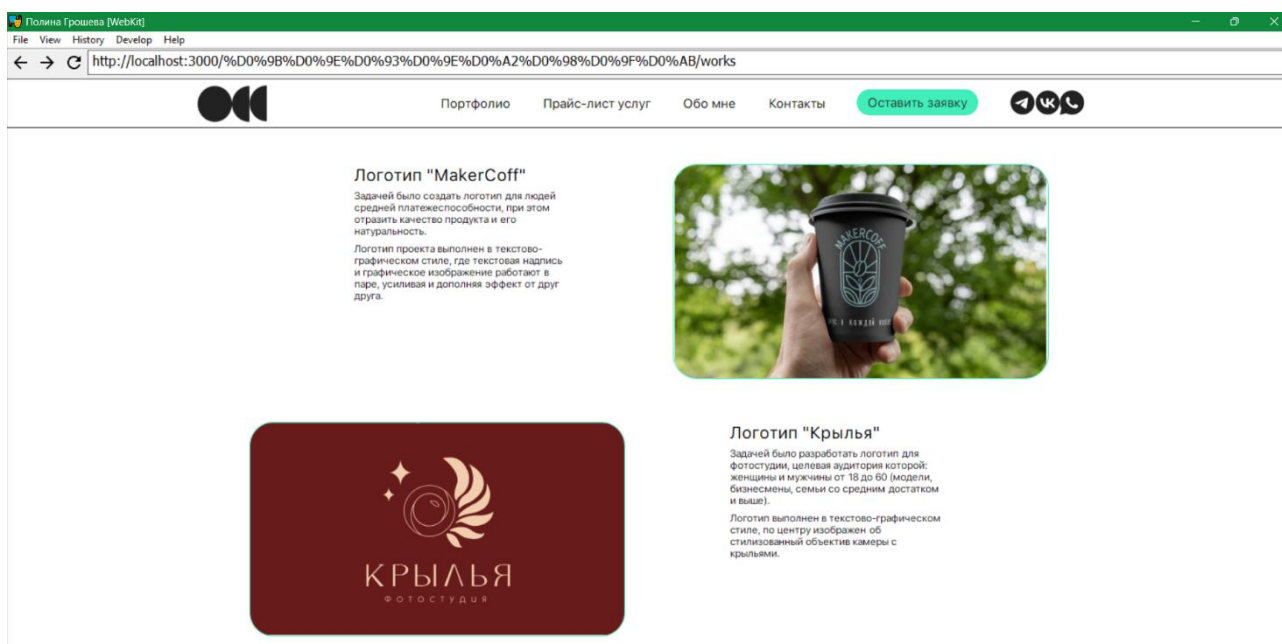


Рисунок Е.6 – Страница портфолио работ на движке WebKit