

Оглавление

Реферат	7
Abstract	8
Введение	9
1 Постановка задачи и аналитический обзор аналогичных решений	10
1.1 Постановка задачи.....	10
1.2 Аналитический обзор аналогов	10
1.2.1 Сайт «Зеленая карта».....	10
1.2.2 Сайт rsbor	11
1.2.3 Сайт target99	12
1.3 Выводы по разделу.....	13
2 Проектирование веб-приложения.....	14
2.1 Диаграмма вариантов использования	14
2.2 Логическая схема базы данных	15
2.3 Архитектура веб-приложения.....	20
2.4 Проектирование основных алгоритмов	21
2.4.1 Алгоритм добавления вторсырья	21
2.4.2 Алгоритм сдачи вторсырья	22
2.4.3 Алгоритм просмотра скидок и открытия промокодов	22
2.5 Выводы по разделу.....	23
3 Разработка веб-приложения	24
3.1 Обзор технологий реализации веб-приложения	24
3.2 Система управления базами данных MySQL	25
3.3 Взаимодействие с базой данных	25
3.4 Модели данных.....	26
3.5 Реализация серверной части приложения.....	27
3.6 Реализация функциональности	30
3.6.1 Регистрация.....	30
3.6.2 Авторизация.....	30
3.6.3 Просмотр пунктов приема	31
3.6.4 Отметка сдачи вторсырья.....	31
3.6.5 Обмен баллов.....	32
3.6.6 Добавление статьи	32
3.6.7 Добавление пункта приема	33
3.7 Функции промежуточной обработки	33
3.8 Реализация отправки писем.....	34
3.9 Реализация клиентской части приложения.....	36

					ДП 00.00.ПЗ			
		ФИО	Подпись	Дата	Оглавление			
Разраб.		Трубач Д.С.						
Пров.		Муцук А.Н.						
Н. контр.		Муцук А.Н.						
Утв.		Смелов В.В.			БГТУ 1-40 01 01, 2025			
					Лит.	Лист	Листов	
					У	1	3	

3.9.1	Реализация хранилища при помощи библиотеки Redux ToolKit..	37
3.9.2	Маршрутизация в клиентской части.....	38
3.10	Контейнеризация.....	39
3.11	Выводы по разделу.....	40
4	Тестирование веб-приложения.....	41
4.1	Негативное тестирование страницы входа.....	41
4.2	Негативное тестирование страницы регистрации.....	43
4.3	Негативное тестирование форм добавления и изменения вторсырья, скидок и пунктов приема.....	45
4.4	Негативное тестирование формы добавления статьи.....	47
4.4	Негативное тестирование формы проема вторсырья.....	48
4.5	Функциональное тестирование.....	49
4.6	Выводы по разделу.....	51
5	Руководство пользователя.....	52
5.1	Регистрация.....	52
5.2	Авторизация.....	53
5.3	Просмотр статей.....	54
5.4	Просмотр пунктов приема.....	55
5.5	Просмотр скидок.....	55
5.6	Обмен баллов на скидки.....	56
5.7	Добавление статьи.....	56
5.8	Изменение статьи.....	57
5.9	Удаление статьи.....	58
5.10	Добавление комментария.....	58
5.11	Удаление комментария.....	59
5.12	Добавление отметки «Нравится».....	59
5.13	Отметка сдачи вторсырья.....	60
5.14	Добавление ключей.....	61
5.15	Изменение ключей.....	61
5.16	Добавление пункта приема.....	62
5.17	Изменение времени работы пункта.....	62
5.18	Удаление пункта приема.....	63
5.19	Добавление вида вторсырья.....	63
5.20	Изменение вида вторсырья.....	64
5.21	Удаление вида вторсырья.....	65
5.22	Добавление проверки веса.....	66
5.23	Изменение проверки веса.....	67
5.24	Добавление скидки.....	67
5.25	Изменение скидки.....	68
5.26	Удаление скидки.....	69
5.27	Редактирование любых статей.....	69
5.28	Удаление любых статей и комментариев пользователей.....	70
5.30	Выводы по разделу.....	70
6	Технико-экономическое обоснование проекта.....	71

6.1 Общая характеристика разрабатываемого программного средства	71
6.2 Исходные данные для проведения расчетов	71
6.3 Обоснование цены программного средства	72
6.3.1 Расчет затрат рабочего времени на разработку программного средства	72
6.3.2 Расчет основной заработной платы.....	73
6.3.3 Расчет дополнительной заработной платы	74
6.3.4 Расчет отчислений в фонд социальной защиты населения и по обязательному страхованию	74
6.3.5 Расчет суммы прочих прямых затрат.....	75
6.3.6 Расчет суммы накладных расходов.....	76
6.3.7 Сумма расходов на разработку программного средства	76
6.3.8 Расходы на сопровождение и адаптацию	76
6.3.9 Расчет полной себестоимости.....	77
6.3.10 Определение цены, оценка эффективности	77
6.4 Выводы по разделу.....	79
Заключение	81
Список использованной литературы.....	82
Диаграмма вариантов использования ДП 01.00 ГЧ.....	84
Логическая схема базы данных ДП 02.00 ГЧ.....	85
Диаграмма развертывания ДП 03.00 ГЧ	86
Диаграмма последовательности добавления вторсырья ДП 04.00 ГЧ	87
Блок схема алгоритма сдачи вторсырья ДП 05.00 ГЧ.....	88
Блок схема алгоритма просмотра скидок и открытия промокодов	89
ДП 06.00 ГЧ	89
Таблица экономических показателей ДП 07.00 ГЧ.....	90
ПРИЛОЖЕНИЕ А	91
ПРИЛОЖЕНИЕ Б.....	95
ПРИЛОЖЕНИЕ В	102
ПРИЛОЖЕНИЕ Г.....	110

Реферат

Пояснительная записка содержит 90 страниц, 47 рисунков, 30 таблиц, 11 листингов, 28 использованных источников, 11 приложений.

ВЕБ-ПРИЛОЖЕНИЕ, NODE.JS, EXPRESS, REACT.JS, TAILWIND CSS, MYSQL, VISUAL STUDIO CODE, MYSQL WORKBEANCH, ВТОРСЫРЬЕ, ПРОМОКОДЫ.

Целью работы является разработать веб-приложение, которое повышает осведомленность и популяризацию сортировки бытовых отходов. Пояснительная записка дипломного проекта состоит из реферата, введения, оглавления, шести глав, заключения, списка использованной литературы, графической части и приложений.

Во введении обоснована актуальность темы, определена цель разработки веб-приложения для сортировки бытовых отходов, сформулированы основные задачи и указана целевая аудитория.

В первом разделе описаны аналоги веб-приложения, выделены плюсы и минусы каждого из них.

Во втором разделе представлены диаграммы вариантов использования, архитектуры приложения, развертывания, логическая схема базы данных и блок схемы алгоритмов.

В третьем разделе приведено описание разработки программного обеспечения веб-приложения. Приводятся листинги исходного кода.

В четвертом разделе приведены результаты тестирования веб-приложения.

В пятом разделе приведено руководство пользователя.

В шестом разделе приводится расчет экономических параметров.

В заключении приведены результаты проделанной работы, приводятся соображения насчет использования данного программного средства.

Объем графической части составляет семь схем, пять листов А3 и два листа А4.

					ДП 00.00.ПЗ		
		ФИО	Подпись	Дата	Реферат		
Разраб.		Трубач Д.С.					
Пров.		Муцук А.Н.					
Н. контр.		Муцук А.Н.					
Утв.		Смелов В.В.			БГТУ 1-40 01 01, 2025		
					Лит.	Лист	Листов
					У	1	1

Abstract

The explanatory note contains 90 pages, 47 drawings, 30 tables, 11 listings, 28 references, 11 applications.

WEB APP, NODE.JS, EXPRESS, REACT.JS, TAILWIND CSS, MYSQL, VISUAL STUDIO CODE, MYSQL WORKBEANCH, RECYCLING, PROMO-CODES.

The goal of the graduation project is to develop a web application that will help raise awareness and popularize the sorting of household waste. The explanatory report of the graduation project includes an abstract, an introduction, table of contents, six chapters, a conclusion, a list of references, a graphic part and appendices.

In the introduction, the relevance of the topic is substantiated, the purpose of developing a web application for sorting household waste is defined, the main tasks are formulated and the target audience is indicated.

The first section describes the analogues of the web application, highlights the pros and cons of each of them.

The second section provides diagrams of use cases, application architecture, deployment, database logic diagram, and algorithm flowcharts.

The third section describes the development of a web application software. The source code listings are provided.

The fourth chapter contains the results of the web application testing.

The fifth chapter provides a user manual for the software product.

The sixth chapter presents calculations of the economic parameters of the developed software.

In conclusion, the results of the work done are presented, and considerations regarding the use of this software tool are given.

The graphical part includes seven diagrams, five A3 sheets, and two A4 sheets.

					ДП 00.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Трубач Д.С.			Abstract	Лит.	Лист	Листов
Пров.		Мущук А.Н.				У	1	1
						БГТУ 1-40 01 01, 2025		
Н. контр.		Мущук А.Н.						
Утв.		Смелов В.В.						

Введение

Актуальность дипломного проекта обусловлена тем, что в условиях роста экологической сознательности и активного развития цифровых технологий особую значимость приобретают онлайн-системы, упрощающие организацию и мотивацию раздельного сбора вторсырья. Современные веб-приложения, направленные на вовлечение пользователей в экологические инициативы, позволяют автоматизировать ключевые процессы: регистрацию участников, отслеживание сданных отходов, начисление бонусов и визуализацию личного вклада в сохранение окружающей среды. Подобный функционал реализуется с помощью современных веб-технологий, где обработка данных распределяется между клиентской и серверной частью, а взаимодействие осуществляется через HTTP-протокол, обеспечивая стабильную и доступную работу платформы.

Целью дипломного проекта является разработка веб-приложения «WasteWise» – современной платформы для организации раздельного сбора бытовых отходов. Целевая аудитория веб-приложения включает широкий круг пользователей, заинтересованных в экологически ответственном образе жизни. Это могут быть как индивидуальные пользователи, так и организации, внедряющие раздельный сбор мусора.

Для достижения цели были сформулированы следующие задачи:

- провести анализ аналогичных решений для выявления их преимуществ и недостатков;
- спроектировать приложение (разработать функциональные требования, разработать архитектуру, структуру базы данных, и пользовательский интерфейс), обеспечивающее поддержку всех заявленных функций;
- реализовать клиентскую и серверную части с применением современных технологий;
- выполнить тестирование функциональности приложения;
- подготовить руководство пользователя;
- провести технико-экономическое обоснование проекта.

Актуальность проекта обусловлена следующими факторами:

Внедрение такого решения значительно сократит временные затраты на организацию экологических инициатив, минимизирует бумажный документооборот и сделает процесс участия максимально прозрачным и понятным для всех сторон. Это будет способствовать увеличению количества участников экодвижения и повышению эффективности программ по переработке отходов.

					ДП 00.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Трубац Д.С.			Введение			
Пров.		Муцук А.Н.						
Н. контр.		Муцук А.Н.						
Утв.		Смелов В.В.						
						Лит.	Лист	Листов
						У	1	1
						БГТУ 1-40 01 01, 2025		

1 Постановка задачи и аналитический обзор аналогичных решений

1.1 Постановка задачи

Задача заключается в разработке веб-приложения для отдельного сбора бытовых отходов, которое предоставит пользователям удобный и функциональный интерфейс для взаимодействия с системой утилизации мусора.

В системе предусмотрено две основные роли: пользователь и администратор. Для пользователя приложение должно реализовывать функции поиска и отображения ближайших пунктов сбора отходов, получения информации о сортировке мусора. Администратор должен иметь возможность управлять базой данных мест сбора, добавлять и удалять пункты, следить за системой выдачей промокодов.

1.2 Аналитический обзор аналогов

Были проанализированы цели и задачи, поставленные в данном проекте, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

1.2.1 Сайт «Зеленая карта»

Для анализа был изучен один из популярных экологических ресурсов, белорусский сайт «Зеленая карта» — greentap.by [1].

На рисунке 1.1 показана страница данного сайта.

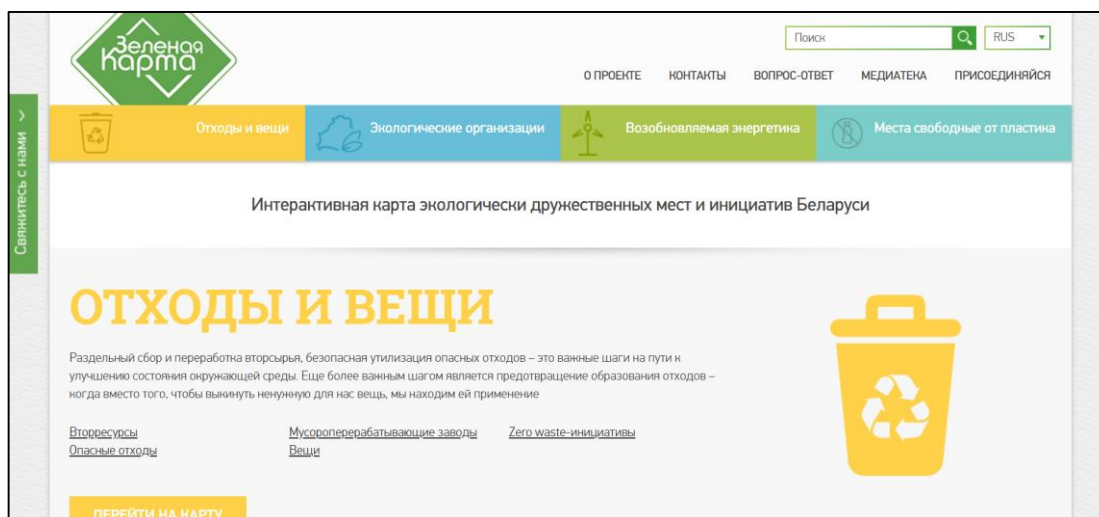


Рисунок 1.1 – Интерфейс главной страницы «Зеленая карта»

					ДП 01.00.ПЗ		
		ФИО	Подпись	Дата	1 Постановка задачи и аналитический обзор аналогичных решений		
Разраб.		Трубац Д.С.					
Пров.		Мушук А.Н.					
Н. контр.		Мушук А.Н.					
Утв.		Смелов В.В.					
					Лит.	Лист	Листов
					У	1	4
					БГТУ 1-40 01 01, 2025		

Кроме информации о раздельном сборе отходов, на сайте также можно найти информацию о различных экологических организациях, включая государственные организации, научные лаборатории и учебные заведения. Кроме того, на сайте есть отдельный пункт меню, который посвящен возобновляемой энергетике, и на котором можно найти информацию о солнечных панелях, солнечных водонагревателях, ветрогенераторах, биогазовых установках и гидроэлектростанциях. На сайте также есть пункт меню, где представлены места, свободные от пластика, такие как школы, церковные приходы, кафе и офисы.

В данном проекте не было найдено никаких недостатков, ресурс полностью соответствует своему названию «Зеленая карта», и все возможные места, связанные с экологическими объектами, расположены на карте в понятном и доступном виде. Этот ресурс является отличным источником информации для тех, кто заботится об окружающей среде и хочет внести свой вклад в экологическую инициативу. Благодаря «Зеленой карте» люди могут быть в курсе всех экологических объектов и ресурсов в своем городе, что помогает им принимать осознанные решения и принимать активное участие в раздельном сборе отходов, защите природы и пропаганде экологического образа жизни.

1.2.2 Сайт *rsbor*

Следующим результатом поиска аналогов стал сайт *rsbor* [2], главная страница которого представлена на рисунке 1.2.

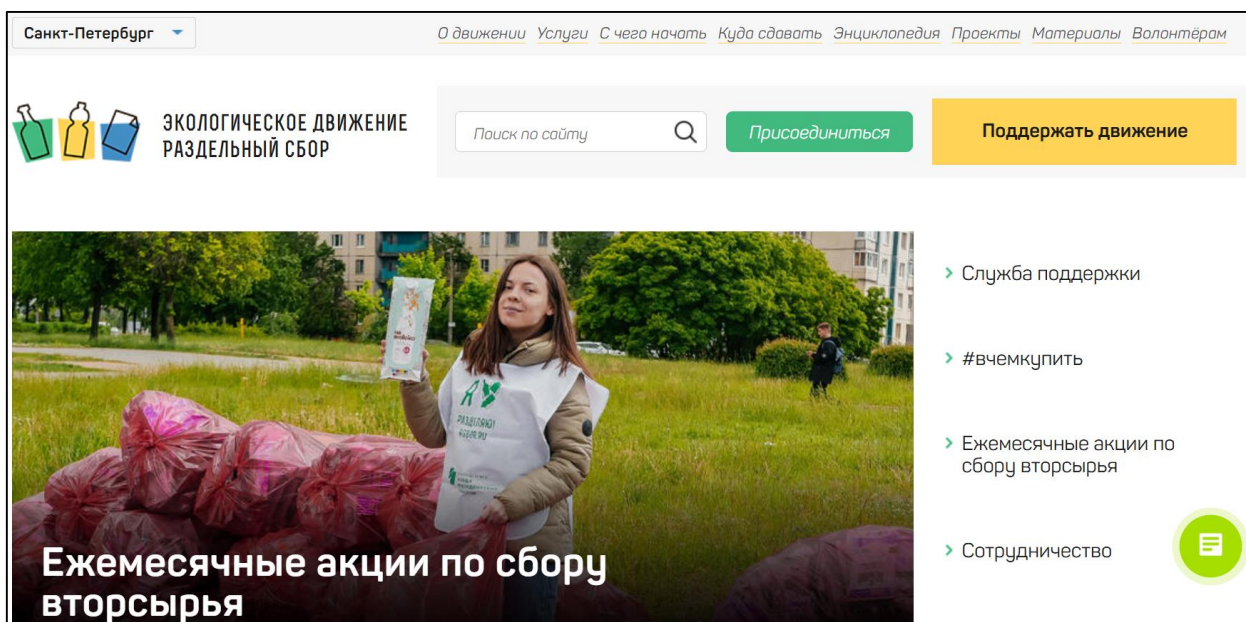


Рисунок 1.2 – Интерфейс главной страницы *rsbor*

Сайт *rsbor* представляет собой информационный ресурс, основной упор которого сделан на новостную составляющую. На главной странице пользователи могут ознакомиться с последними новостями, связанными со сбором, разделением и переработкой отходов и вторсырья в Санкт-Петербурге или выбранном городе из выпадающего списка. Кроме того, на других страницах

представлен зарубежный опыт в области переработки отходов и ресурсосбережения, позволяя пользователям расширить свои знания в этой сфере.

Одним из важных разделов сайта является раздел, посвященный проектам, проводимым Экологическим Движением «Раздельный Сбор». Здесь пользователи могут ознакомиться с текущими и предстоящими проектами, которые организует движение. Также на сайте предусмотрена возможность сделать пожертвования для поддержки и развития этого движения, позволяя желающим внести свой вклад.

Одной из удобных функций сайта является просмотр пунктов приема, где отходы могут быть сданы по видам вторсырья. Эта информация предоставляет удобный способ для пользователей найти ближайшие пункты приема и правильно сортировать свои отходы.

1.2.3 Сайт target99

После первых двух приложений обратилось внимание к белорусскому ресурсу государственного учреждения «Оператор вторичных материальных ресурсов» [3], уполномоченного министерством жилищно-коммунального хозяйства Республики Беларусь, для большего понимания какое развитие имеет ситуация с вторсырьем в нашей стране. Данный веб-ресурс представлен на рисунке 1.3.

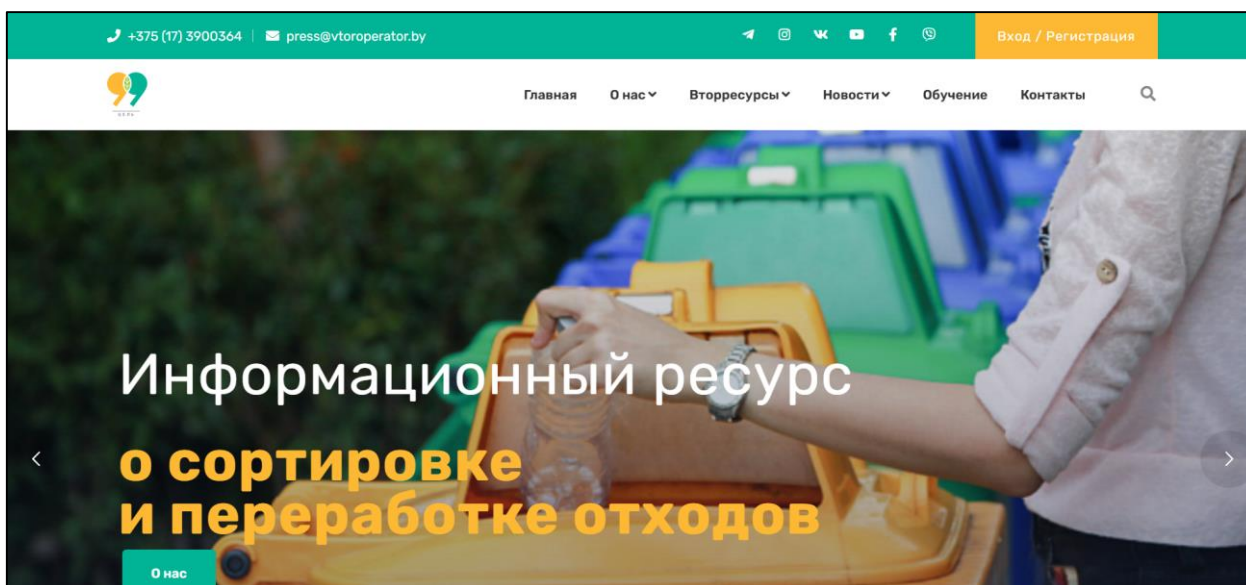


Рисунок 1.3 – Интерфейс главной страницы *target99*

Основная информация, представленная на главной странице сайта, сосредоточена на правилах сортировки вторичных ресурсов. Чтобы облегчить поиск нужной информации, правила разделены по категориям, что позволяет пользователям быстро найти необходимую информацию в соответствии с их потребностями и видами отходов.

Одним из важных аспектов развития раздельного сбора мусора является подробная карта пунктов приема вторичного сырья. В Беларуси сегодня

насчитывается более 1700 пунктов приема, и их число постоянно увеличивается. Эта карта предоставляет пользователю удобный инструмент для поиска ближайших пунктов приема и утилизации отходов, позволяя эффективно управлять своими ресурсами.

В целом, после рассмотрения всех указанных ресурсов на данном сайте не было выявлено никаких недостатков. Это говорит о том, что ресурс предоставляет полезную информацию и инструменты, необходимые для активного участия в раздельном сборе и переработке отходов, а также для продвижения экологичного образа жизни.

1.3 Выводы по разделу

Анализ аналогичных веб-ресурсов, таких как «Зеленая карта», *rsbor* и сайт государственного учреждения «Оператор вторичных материальных ресурсов», позволил выделить ключевые принципы, которые следует учитывать при разработке веб-приложения для управления отходами и экологии.

Во-первых, важна доступность информации и удобство навигации. «Зеленая карта» демонстрирует, как интерактивная карта с геолокацией точек сбора отходов повышает вовлеченность пользователей. Это позволяет быстро находить экологические объекты и точки раздельного сбора.

Во-вторых, новостная и образовательная составляющая играет значительную роль. Опыт *rsbor* показывает, что информирование пользователей о последних тенденциях, зарубежных практиках и локальных инициативах в сфере переработки отходов способствует осознанному подходу к экологии.

В-третьих, важна интеграция с государственными программами и официальными инициативами. Белорусский ресурс государственного учреждения «Оператор вторичных материальных ресурсов» демонстрирует, что предоставление актуальных правил сортировки отходов и возможности поиска пунктов приема делает сервис максимально полезным.

Таким образом, разрабатываемое веб-приложение должно включать удобный интерфейс для поиска пунктов приема отходов, актуальную новостную ленту о переработке отходов, а также интеграцию с государственными и частными экологическими инициативами. Это позволит создать полезный и востребованный ресурс для пользователей, заинтересованных в раздельном сборе и экологическом образе жизни.

2 Проектирование веб-приложения

Проектирование программного средства до его непосредственной реализации является критически важным этапом разработки, так как позволяет заблаговременно определить необходимые компоненты системы и механизмы их взаимодействия. Тщательно продуманный проект обеспечивает четкое понимание архитектуры и структуры приложения, что упрощает последующую реализацию и облегчает дальнейшее сопровождение и развитие программного продукта.

2.1 Диаграмма вариантов использования

Для описания основных сценариев взаимодействия пользователей с веб-приложением была разработана диаграмма вариантов использования. Диаграмма демонстрирует ключевые роли в системе и действия, доступные для каждой из них.

Описание ролей представлено в таблице 2.1.

Таблица 2.1 – Описание ролей

Роль	Описание
Гость	Имеет возможность произвести регистрацию и авторизацию, также может посмотреть имеющиеся статьи.
Пользователь	Может просматривать пункты приема вторсырья, обменивать накопленные баллы на скидки в различных сервисах, создавать, изменять и удалять свои статьи, оценивать статьи, а также получать информацию о том, сколько новой продукции будет произведено из сданных отходов.
Администратор	Обладает правами для администрирования системы: управление видами вторсырья, пунктами приема, управление промокодами.

В приложении имеются две основные роли: Пользователь и Администратор. Администратор обладает полным контролем над веб-приложением и выполняет следующие задачи: добавление, изменение и удаление собственных статей; удаление статей, опубликованных пользователями; добавление, изменение и удаление видов вторсырья; добавление, удаление и изменение скидок; добавление и удаление пунктов приема; изменение рабочего времени пунктов приема; добавление и изменение секретных ключей для пунктов приема; а также добавление проверки веса для видов вторсырья.

Пользователь имеет следующие возможности: чтение опубликованных статей; добавление, изменение и удаление собственных статей; комментирование и оценка статей; сдача вторсырья с получением баллов; обмен баллов на скидки; просмотр доступных видов вторсырья и пунктов приема.

					ДП 02.00.ПЗ		
		ФИО	Подпись	Дата	2 Проектирование веб-приложения		
Разраб.		Трубач Д.С.					
Пров.		Мушук А.Н.					
Н. контр.		Мушук А.Н.					
Утв.		Смелов В.В.					
					Лит.	Лист	Листов
					У	1	10
					БГТУ 1-40 01 01, 2025		

Также в приложении присутствует роль «Гость», который может читать статьи и комментарии к ним, а также зарегистрироваться в веб-приложении.

Результатом разработки является диаграмма вариантов использования, которая отображает функциональные возможности разрабатываемого программного средства. Эта диаграмма будет использоваться в дальнейшей разработке приложения для определения и описания функциональности, которую приложение должно предоставлять.

В ДП 01.00 ГЧ представлена полная диаграмма вариантов использования.

2.2 Логическая схема базы данных

Диаграмма базы данных таблиц (*Database Table Diagram*) – это визуальное представление структуры базы данных и отношений между таблицами, которые хранятся в этой базе данных.

Для веб-приложения была спроектирована база данных, включающая 12 таблиц, обеспечивающих хранение информации о пользователях, статьях, пунктах приема и других объектах системы. Ее логическая схема показана в ДП 02.00 ГЧ.

Таблица *users* содержит данные о пользователях приложения, включая их уникальные идентификаторы, учетные записи и другую важную информацию.

Таблица *article* хранит опубликованные статьи, предоставляя пользователям доступ к актуальному контенту.

Таблица *ratings* содержит комментарии к статьям, позволяя пользователям оставлять обратную связь и участвовать в обсуждениях.

Таблица *likes* фиксирует информацию о лайках, которые пользователи поставили статьям. При повторном вызове функции лайка запись удаляется, что позволяет реализовать механику отмены лайка.

Таблица *points* содержит данные о пунктах приема вторсырья, помогая пользователям находить ближайшие места для сдачи отходов.

Таблица *marks* хранит сведения о видах вторсырья, что позволяет классифицировать принимаемые материалы.

Таблица *points_marks* реализует связь «многие-ко-многим» между пунктами приема вторсырья (*points*) и видами вторсырья (*marks*), обеспечивая гибкость в управлении информацией.

Таблица *check_weight* содержит коды для проверки веса сданного вторсырья и его типа, что позволяет автоматизировать процесс учета.

Таблица *s_keys* хранит секретные ключи для пунктов приема вторсырья, обеспечивая защиту данных и контроль доступа.

Таблица *discounts* содержит информацию о доступных скидках, предлагаемых пользователям.

Таблица *promo_codes* хранит промокоды, которые пользователи могут активировать для получения бонусов и специальных предложений.

Таблица *receptions* является статистической таблицей, фиксирующей данные о том, какой пользователь сдал какое вторсырье, что позволяет анализировать динамику и эффективность переработки.

В таблице 2.2 показано описание полей таблицы «*users*».

Таблица 2.2 – Описание структуры таблицы «*users*»

Поле	Тип данных	Ограничения	Описание
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор пользователя (первичный ключ)
<i>username</i>	<i>VARCHAR(20)</i>	<i>NOT NULL</i>	Имя пользователя
<i>email</i>	<i>VARCHAR(100)</i>	<i>NOT NULL, UNIQUE</i>	Почта
<i>password_hash</i>	<i>VARCHAR(200)</i>	<i>NULLABLE, UNIQUE</i>	Захешированный пароль
<i>points</i>	<i>INT</i>	<i>DEFAULT 0, NULL</i>	Количество накопленных баллов
<i>role</i>	<i>VARCHAR(5)</i>	<i>DEFAULT 'USER', NOT NULL, CHECK (role IN ('admin', 'user'))</i>	Роль пользователя
<i>is_activated</i>	<i>TINYINT(1)</i>	<i>DEFAULT 0, NULL</i>	Активирована почта или нет
<i>activation_link</i>	<i>VARCHAR(150)</i>	<i>NULL</i>	Ссылка активации, которая была отправлена на почту

В таблице 2.3 показано описание полей таблицы «*articles*».

Таблица 2.3 – Описание структуры таблицы «*articles*»

Поле	Тип данных	Ограничения	Описание
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор статьи (первичный ключ)
<i>title</i>	<i>VARCHAR(100)</i>	<i>NOT NULL, UNIQUE</i>	Название статьи
<i>text</i>	<i>TEXT</i>	<i>NOT NULL</i>	Текст статьи
<i>date_of_pub</i>	<i>DATE</i>	<i>NOT NULL</i>	Дата публикации
<i>image_url</i>	<i>VARCHAR(150)</i>	<i>NOT NULL</i>	Ссылка на изображение
<i>Author</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Автор публикации (внешний ключ)
<i>likes</i>	<i>INT</i>	<i>DEFAULT 0, NULL</i>	Количество лайков у данной статьи

В таблице 2.4 показано описание полей таблиц «*ratings*».

Таблица 2.4 – Описание структуры таблицы «*ratings*»

Поле	Тип данных	Ограничения	Описание
1	2	3	4
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор комментария (первичный ключ)
<i>article_id</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор статьи (внешний ключ)

Продолжение таблицы 2.4

1	2	3	4
<i>Commenta- tor</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор пользователя (внешний ключ)
<i>comment</i>	<i>VARCHAR(500)</i>	<i>NOT NULL</i>	Текст комментария
<i>date_of_add</i>	<i>DATETIME</i>	<i>DEFAULT CURRENT_TIMESTAMP</i>	Дата добавления комментария

В таблице 2.5 показано описание полей таблицы «likes».

Таблица 2.5 – Описание структуры таблицы «likes»

Поле	Тип данных	Ограничения	Описание
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор лайка (первич- ный ключ)
<i>user_id</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор пользователя (внешний ключ)
<i>article_id</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор статьи (внеш- ний ключ)
<i>date_of_add</i>	<i>DATETIME</i>	<i>DEFAULT CURRENT_TIMESTAMP</i>	Время добавления отметки
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор лайка (первич- ный ключ)

В таблице 2.6 представлена структура таблицы «s_keys».

Таблица 2.6 – Описание структуры таблицы «s_keys»

Поле	Тип данных	Ограничения	Описание
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор секретного ключа (первичный ключ)
<i>se- cret_key</i>	<i>VARCHAR(100)</i>	<i>NOT NULL</i>	Секретный ключ в хешированном виде
<i>is_used</i>	<i>INT</i>	<i>DEFAULT 0, NOT NULL, CHECK (is_used IN (1, 0))</i>	1 – если ключ использован и 0 – не использован

В таблице 2.7 представлена структура таблицы «points».

Таблицы 2.7 – Описание структуры таблицы «points»

Поле	Тип данных	Ограничения	Описание
1	2	3	4
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор пункта при- ема (первичный ключ)
<i>address</i>	<i>VARCHAR(100)</i>	<i>NOT NULL, UNIQUE</i>	Адрес пункта приема
<i>time_of_work</i>	<i>VARCHAR(100)</i>	<i>NOT NULL</i>	Время работы пункта приема
<i>key_id</i>	<i>INT</i>	<i>NOT NULL, UNIQUE, FOREIGN KEY</i>	Идентификатор на секретный ключ (внешний ключ)
<i>admin_id</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор администра- тора, который добавил этот пункт (внешний ключ)

Продолжение таблицы 2.7

1	2	3	4
<i>link_to_map</i>	<i>TEXT</i>	<i>NOT NULL</i>	Ссылка на Яндекс карту
<i>point_name</i>	<i>VARCHAR(100)</i>	<i>NOT NULL, UNIQUE</i>	Имя пункта приема

В таблице 2.8 представлена структура таблицы «*marks*».

Таблица 2.8 – Описание структуры таблицы «*marks*»

Поле	Тип данных	Ограничения	Описание
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор вида вторсырья (первичный ключ)
<i>rubbish</i>	<i>VARCHAR(50)</i>	<i>NOT NULL</i>	Вид вторсырья
<i>points_per_kg</i>	<i>INT</i>	<i>NOT NULL</i>	Начисляемые баллы за один килограмм
<i>new_from_kg</i>	<i>FLOAT</i>	<i>NOT NULL</i>	Сколько новой продукции будет произведено из 1 кг сданного вторсырья
<i>image_link</i>	<i>VARCHAR(255)</i>	<i>NULL</i>	Ссылка на картинку

В таблице 2.9 представлено описание структура таблицы «*points_marks*».

Таблица 2.9 – Описание структуры таблицы «*points_marks*»

Поле	Тип данных	Ограничения	Описание
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор (первичный ключ)
<i>points_id</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор пункта приема (внешний ключ)
<i>marks_id</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор вторсырья (внешний ключ)

В таблице 2.10 описание структуры таблицы «*check_weighth*».

Таблица 2.10 – Описание структуры таблицы «*check_weighth*»

Поле	Тип данных	Ограничения	Описание
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор (первичный ключ)
<i>rubbish_id</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор на вид вторсырья (внешний ключ)
<i>weight</i>	<i>INT</i>	<i>NOT NULL</i>	Вес
<i>key_of_weight</i>	<i>VAR-CHAR(100)</i>	<i>NOT NULL, UNIQUE</i>	Ключ для проверки в захешированном виде
<i>is_used</i>	<i>INT</i>	<i>DEFAULT 0, NOT NULL, CHECK (is_used IN (0, 1))</i>	Содержит значения 1 – если ключ использован и 0 – не использован
<i>oridinal_key</i>	<i>VAR-CHAR(100)</i>	<i>NOT NULL</i>	Оригинальный ключ

В таблице 2.11 представлено описание структуры таблицы «*receptions*».

Таблица 2.11 – Описание структуры таблицы «*receptions*»

Поле	Тип данных	Ограничения	Описание
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор приема (первичный ключ)
<i>user_id</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор пользователя (внешний ключ)
<i>weight</i>	<i>FLOAT</i>	<i>NOT NULL</i>	Вес сколько было сдано вторсырья
<i>accrued</i>	<i>INT</i>	<i>NULL</i>	Начисленные баллы
<i>new_kg</i>	<i>INT</i>	<i>NULL</i>	Вес новой продукции
<i>type_waste</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Вид вторсырья
<i>station_key</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор секретного ключа (внешний ключ)
<i>weight_key</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор ключа для проверки веса (внешний ключ)

В таблице 2.12 описание структуры таблицы «*discounts*».

Таблица 2.12 – Описание структуры таблицы «*discounts*»

Поле	Тип данных	Ограничения	Описание
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор скидки (первичный ключ)
<i>discounts</i>	<i>VARCHAR(50)</i>	<i>NOT NULL, UNIQUE</i>	Название скидки
<i>promo_code</i>	<i>VARCHAR(50)</i>	<i>NOT NULL, UNIQUE</i>	Промокод для применения скидки
<i>count_for_dnt</i>	<i>INT</i>	<i>NOT NULL</i>	Стоимость скидки в баллах

В таблице 2.13 представлено описание структуры таблицы «*promo_codes*».

Таблица 2.13 – Описание структуры таблицы «*promo_codes*»

Поле	Тип данных	Ограничения	Описание
<i>id</i>	<i>INT</i>	<i>PRIMARY KEY</i>	Идентификатор промокода (первичный ключ)
<i>promo_code</i>	<i>VARCHAR(50)</i>	<i>NOT NULL</i>	Текст промокода
<i>user_id</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор пользователя, который открыл скидку (внешний ключ)
<i>discount_id</i>	<i>INT</i>	<i>NOT NULL, FOREIGN KEY</i>	Идентификатор скидки (внешний ключ)
<i>date_of_add</i>	<i>DATE</i>	<i>NOT NULL, DEFAULT '2025-03-27'</i>	Дата, когда был открыт промокод

Один пользователь может быть автором множества статей, что определяется связью между таблицами *users* и *articles*. В свою очередь, одна статья может иметь множество комментариев, что устанавливает связь между *articles* и *ratings*.

Один пользователь может оставить множество лайков, что фиксируется связью между таблицами *users* и *likes*. Аналогично, одна статья может иметь множество лайков, что связывает *articles* и *likes*.

Один пользователь может быть администратором множества пунктов приема, что выражается в связи между *users* и *points*. Каждый пункт приема может принимать множество видов вторсырья, что фиксируется связью между *points* и *points_marks*.

Один пользователь может сдать множество сдач во множестве пунктов приема, что устанавливает связь между *users* и *receptions*, а также между *points* и *receptions*.

Один вид вторсырья может быть связан с множеством пунктов приема, что устанавливает связь между *marks* и *points_marks*.

Один вид вторсырья может иметь множество записей о проверке веса, что фиксируется связью между *marks* и *check_weight*.

Один вид вторсырья может быть сдан множество раз, что выражается в связи между *marks* и *receptions*.

Одна запись о проверке веса может быть связана с множеством сдач, что фиксируется связью между *check_weight* и *receptions*.

Одна скидка может быть связана с множеством промокодов, что устанавливает связь между *discounts* и *promo_codes*.

2.3 Архитектура веб-приложения

Основные компоненты системы включают серверную часть, клиентскую часть, базы данных и вспомогательные сервисы, которые взаимодействуют между собой посредством различных протоколов. Диаграмма развертывания предоставлена в ДП 03.00 ГЧ.

Описание компонент веб-приложения содержится в таблице 2.14.

Таблица 2.14 – Описание компонент веб-приложения

Компонента	Описание компоненты
<i>Frontend Server</i>	Предоставляет доступ к статическим ресурсам клиентской части веб-приложения
<i>Database Server</i>	Используется для хранения и предоставления доступа к данным, которые необходимы для работы веб-приложения.
<i>Backend Server</i>	Обрабатывает запросы пользователя, запрашивает данные из базы данных с помощью <i>ORM Sequelize</i> .
<i>Browser</i>	Отображает клиентскую часть веб-приложения, отправляет запросы пользователя, отображает ответы сервера.
<i>Cloudinary</i> [4]	Облачный сервис, предназначенный для хранения изображений.
<i>SMTP Yandex</i> [5]	Используется для отправки электронных писем, например, для уведомлений пользователей или восстановления пароля.
<i>Yandex Maps API</i> [6]	Предоставляет функциональность для работы с картами, геолокацией и маршрутами.
<i>Goodle API</i> [7]	Включает сервисы, такие как <i>Google Maps</i> , <i>OAuth</i> , <i>Drive</i> , и другие, используемые в веб-приложении.

Описание протоколов, используемых при работе веб-приложений, представлено в таблице 2.15.

Таблица 2.15 – Описание используемых протоколов

Протокол	Назначение
<i>HTTP</i> [8]	Обмен данными между <i>Client Browser</i> и <i>Frontend Server</i> , <i>Frontend Server</i> и <i>Backend Server</i> .
<i>HTTPS</i> [9]	Обмен данными между <i>Backend Server</i> и <i>Cloudinary</i> . Обеспечивает безопасную передачу данных путём использования криптографического протокола <i>TLS</i>
<i>SMTP</i> [10]	Протокол для отправки электронных писем, используемый для передачи сообщений между почтовыми серверами и клиентами.
<i>TCP</i> [11]	Обмен данными между <i>Database Server</i> и <i>Web API Server</i> .

Таким образом были рассмотрены все ключевые элементы архитектуры веб-приложения.

2.4 Проектирование основных алгоритмов

В данном подразделе будут описаны основные алгоритмы работы приложения, которые соответствуют целям дипломного проекта.

В ходе выполнения дипломного проекта были разработаны три ключевых алгоритма: два из них представлены классическими блок-схемами, демонстрирующими логическую структуру и последовательность операций, а третий алгоритм – диаграммой последовательности. Подробное описание каждого алгоритма с соответствующими схемами приводится в следующих подразделах.

2.4.1 Алгоритм добавления вторсырья

Чтобы добавить новый вид вторсырья в систему, администратор сначала взаимодействует с графическим интерфейсом (*GUI*), где вводит информацию о новом виде вторсырья и отправляет соответствующий запрос. После этого данные передаются в блок обработки запроса, который выполняет первичную валидацию и направляет информацию дальше, в блок сервиса по управлению вторсырьём. Этот сервис, в свою очередь, обращается к блоку базы данных, чтобы записать новый тип вторсырья в соответствующую таблицу. Если во время выполнения возникает ошибка, например, при подключении к базе данных или при нарушении уникальности записей, система формирует ответ с ошибкой, который возвращается из блока базы данных в сервисный блок, далее в блок обработки запроса, а затем обратно в графический интерфейс, где администратору отображается сообщение об ошибке. Если операция проходит успешно, база данных сохраняет новый вид вторсырья, сервис получает подтверждение и передаёт его обратно через цепочку: сначала в блок обработки запроса, затем в *GUI*, где администратор получает уведомление об успешном добавлении нового вида вторсырья в

систему. Диаграмма последовательности данного алгоритма приведена в ДП 04.00 ГЧ.

Данная схема наглядно отражает последовательность взаимодействия между компонентами системы при добавлении нового вида вторсырья, обеспечивая целостность и корректность внесения данных.

2.4.2 Алгоритм сдачи вторсырья

Чтобы отметить сданное вторсырье в приложении, пользователь должен сначала зарегистрироваться или войти в веб-приложение. Затем на странице приема вторсырья пользователь должен ввести два ключа, которые были выданы в пункте приема, в соответствующую форму. После ввода ключей начинаются проверки.

Вначале происходит проверка секретного ключа. Если ключ существует, проверка продолжается, в противном случае выводится ошибка, сообщающая о его отсутствии. Затем происходит проверка, связан ли данный ключ с конкретным пунктом приема. Если связь существует, проверка продолжается, в противном случае выводится сообщение о недействительности ключа. Далее происходит проверка второго ключа для подтверждения веса. Если ключ верен, работа алгоритма продолжается, в противном случае выводится сообщение о недействительности ключа. Исходя из введенного ключа находится информация о вторсырье, которое пользователь сдал, включая количество баллов, начисляемых за один килограмм, и количество новой продукции, произведенной из одного сданного килограмма вторсырья.

Затем проверяется количество баллов, имеющихся у пользователя. Рассчитываются новые баллы, вес и количество произведенной продукции. Новые баллы записываются пользователю, а информация о приеме добавляется в таблицу для статистики. Код для пункта приема обновляется, поскольку одноразовые ключи не могут быть повторно использованы. В конечном итоге пользователю выводится сообщение о его баллах, количество начисленных баллов и количество новой продукции, которая будет произведена из сданного пользователем вторсырья. Блок-схема данного алгоритма приведена в ДП 05.00 ГЧ.

2.4.3 Алгоритм просмотра скидок и открытия промокодов

Для того чтобы получить и активировать промокод в приложении, пользователь должен сначала авторизоваться. После успешной авторизации происходит получение данных с сервера, в том числе информации о текущей роли пользователя. Далее осуществляется проверка, обладает ли пользователь ролью «Пользователь». Если роль не соответствует, алгоритм прекращается. В случае, если роль подходит, происходит извлечение данных пользователя из базы данных.

Затем проверяется наличие баллов у пользователя. Если баллы отсутствуют, пользователю выводится сообщение «Скидок нет», и выполнение алгоритма завершается. В противном случае осуществляется вывод списка доступных скидок. Далее пользователь может открыть промокод, после чего с его аккаунта списываются баллы, соответствующие стоимости промокода. Затем происходит вывод информации об открытом промокоде, после чего алгоритм завершается. Блок-схема данного алгоритма приведена в ДП 06.00 ГЧ.

2.5 Выводы по разделу

В данной главе были определены функциональные возможности пользователей с ролями «Гость», «Пользователь» и «Администратор». Для каждой группы предусмотрен соответствующий набор прав и возможностей взаимодействия с платформой, что в совокупности составляет 29 реализованных функций. Эти функции охватывают весь спектр взаимодействия с системой - от процедуры авторизации до управления пунктами приема вторсырья и публикации контента. Взаимосвязь между ролями и функционалом наглядно отображена на диаграмме вариантов использования.

Структура базы данных включает 12 таблиц, охватывающие ключевые сущности: пользователи, вторсырье, пункты приема, статьи, скидки и промокоды.

Спроектирована трехзвенная архитектура приложения, включающая клиентский уровень на *React.js* для взаимодействия с пользователем, серверный уровень на *Node.js* с использованием *Express.js* для обработки бизнес-логики и *API*, а также уровень базы данных на *MySQL* для хранения и управления данными.

Разработанные функциональные возможности системы, продуманная структура базы данных и многоуровневая архитектура приложения обеспечивают стабильную и эффективную работу платформы. Реализованные решения полностью удовлетворяют потребности всех категорий пользователей – от гостей до администраторов, предоставляя каждому интуитивно понятный и удобный интерфейс для взаимодействия с веб-приложением.

3 Разработка веб-приложения

Важно, чтобы программа эффективно решала поставленные задачи и надежно выполняла свои функции в различных условиях. Ее характеристики должны включать надежность, обеспечение безопасности, высокую производительность и способность справляться с ростом нагрузки.

В ходе разработки были созданы два основных компонента – серверная и клиентская части, для реализации которых использовались подходящие технологии и библиотеки, необходимые для стабильной работы приложения.

3.1 Обзор технологий реализации веб-приложения

В качестве языка программирования для серверной части приложения был выбран *JavaScript* с использованием среды *Node.js*. *Node.js* [12] – это асинхронная, событийно-ориентированная среда выполнения, которая позволяет эффективно обрабатывать большое количество одновременных запросов. Благодаря своей неблокирующей архитектуре, *Node.js* отлично подходит для создания высоконагруженных веб-приложений и *API*.

Фреймворком для создания *API* был выбран *Express.js* [13] – минималистичный и гибкий веб-фреймворк для *Node.js*. *Express* предоставляет мощные инструменты для маршрутизации, обработки *HTTP*-запросов, *middleware*-функций и быстрого создания *RESTful API*. Он широко используется в индустрии благодаря своей простоте, расширяемости и активному сообществу.

Для клиентской части приложения был выбран *React* [14] – библиотека *JavaScript*, разработанная *Facebook* для создания пользовательских интерфейсов. *React* использует компонентный подход, позволяющий создавать повторно используемые *UI*-компоненты и эффективно управлять состоянием интерфейса. Благодаря виртуальному *DOM*, *React* обеспечивает высокую производительность и отзывчивость веб-приложений

Docker [16] – инструмент контейнеризации для развертывания приложений в изолированной среде. Обеспечивает стабильность и масштабируемость, но требует знаний и может иметь накладные расходы.

Docker Compose [17] – инструмент для управления многоконтейнерными приложениями. Упрощает настройку и запуск, но сложные конфигурации требуют опыта работы с *YAML*.

RTK Query [18] – библиотека для работы с *API* и состоянием в *React*. Упрощает запросы и кеширование, но требует освоения *Redux*-подхода.

Swagger [19] – инструменты и спецификация *OpenAPI* для проектирования и документации *REST API*. Позволяет визуализировать и тестировать *API*.

					ДП 03.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Трубач Д.С.			3 Разработка веб-приложения	Лит.	Лист	Листов
Пров.		Мушук А.Н.				У	1	17
						БГТУ 1-40 01 01, 2025		
Н. контр.		Мушук А.Н.						
Утв.		Смелов В.В.						

Axios [20] – библиотека для *HTTP*-запросов в *Node.js* и браузере. Удобна для взаимодействия с *REST API*.

Framer Motion [21] – библиотека анимаций для *React*. Обеспечивает плавные переходы и интерактивные элементы интерфейса.

React Toastify [22] – система уведомлений для *React*. Предоставляет готовые компоненты для отображения всплывающих сообщений.

Tailwind CSS [23] – утилитарный *CSS*-фреймворк для быстрой вёрстки интерфейсов. Позволяет создавать уникальные дизайны через комбинацию готовых классов, но требует изучения множества *utility*-правил.

TensorFlow.js [24] – библиотека машинного обучения для браузера. Позволяет запускать предобученные модели прямо на клиенте.

Для управления зависимостями и добавления дополнительных пакетов для *Node.js* в проект использовался менеджер пакетов *npm* [25]. *NPM* является стандартным менеджером пакетов, входящим в состав *Node.js*. Он упрощает процесс установки и обновления сторонних библиотек и инструментов в Visual Studio Code, а также управление зависимостями в приложении.

3.2 Система управления базами данных MySQL

В качестве базы данных используется *MySQL* – популярная реляционная СУБД с открытым исходным кодом. Она отлично подходит для структурированных данных и легко интегрируется с *Node.js* через различные драйверы и *ORM*-библиотеки. Скрипт создания базы данных приведён в приложении А.

Для взаимодействия с базой данных *MySQL* использовалась стандартная среда *MySQL Workbench* [26], предоставляющая удобный интерфейс для проектирования схем, выполнения запросов и администрирования БД.

3.3 Взаимодействие с базой данных

Sequelize [27] – это библиотека, которая обеспечивает взаимодействие приложения *Node.js* с базой данных *MySQL*.

Пример подключения базы данных представлен на листинге 3.1.

```
const dbConfig = {
  database: 'ecosort', username: 'root', password: '1111',
  host: isDocker ? 'host.docker.internal' : 'localhost',
  dialect: 'mysql',
}; global.sequelize = new Sequelize(dbConfig.database,
dbConfig.username, dbConfig.password, {
  host: dbConfig.host, dialect: dbConfig.dialect,});
```

Листинг 3.1 – Подключение базы данных

Данное подключение позволяет приложению безопасно и эффективно взаимодействовать с базой данных, выполняя основные операции чтения, записи, обновления и удаления данных.

3.4 Модели данных

Для работы с данными базы данных были созданы модели, которые соотносятся с соответствующими сущностями и хранятся в папке *config*. В таблице 3.1 перечислены модели данных, используемые в разработанном приложении.

Таблица 3.1 – Описание моделей данных приложения

Название модели	Описание модели
<i>Articles</i>	Содержит все необходимые свойства для создания, изменения и удаления статей
<i>Check_weights</i>	Содержит все необходимые свойства для создания кодов для проверки веса и вида вторсырья
<i>Discounts</i>	Содержит все необходимые свойства для создания, изменения и удаления скидок
<i>Keys</i>	Содержит все необходимые свойства для создания и изменения секретных ключей
<i>Likes</i>	Содержит все необходимые свойства для добавления лайков
<i>Marks</i>	Содержит все необходимые свойства для создания, изменения и удаления вторсырья
<i>Points</i>	Содержит все необходимые свойства для создания, изменения и удаления пунктов приема вторсырья
<i>Points_marks</i>	Содержит все необходимые свойства для реализации связи «многие-ко-многим» между таблицами <i>points</i> и <i>marks</i>
<i>Promo_codes</i>	Содержит все необходимые свойства для добавления промокодов
<i>Receptions</i>	Содержит все необходимые свойства для приема вторсырья
<i>Users</i>	Содержит свойства однозначно идентифицирующие зарегистрированного пользователя

Каждая модель данных соотносится с определенной сущностью базы данных, а ее свойства соответствуют полям этой сущности. В качестве примера в листинге 3.2 представлен исходный код модели *Marks*.

```
class Discounts extends Model{}
Discounts.init (
  {
    id: {type: Sequelize.INTEGER, primaryKey:true, unique:
true, autoIncrementIdentity: true, required: true},
    discount: {type: Sequelize.STRING, allowNull: false,
unique: true, required: true},
    count_for_dnt: {type: Sequelize.INTEGER, allowNull:
false, required: true},
    promo_code: {type: Sequelize.STRING, allowNull: false,
unique: true, required: true},
  }, { sequelize, modelName: 'Discounts', tableName:
'discounts', timestamps: false } ); module.exports =
{Discounts};
```

Листинг 3.2 – Модель *Marks*

После определения схемы *Sequelize* дает возможность создать модель, основанную на определенной схеме. Затем модель синхронизируется с документом *MySQL* с помощью определения схемы модели. В данном примере схема имеет пять свойств. Код реализованных моделей представлен в приложении Б.

3.5 Реализация серверной части приложения

Файлы серверной части организованы таким образом, чтобы они были сгруппированы в директории по их назначению. Ниже приведена таблица 3.2, описывающая основные папки серверной части.

Таблица 3.2 – Описание директорий серверной части веб-приложения

Директория	Описание
<i>config</i>	Содержит конфигурационные файлы и модели <i>Sequelize</i> для работы с базой данных. Включает модели для таких сущностей, как статьи, пользователи, рейтинги и др.
<i>controllers</i>	Реализует бизнес-логику приложения. Включает контроллеры, обрабатывающие запросы к <i>API</i> , и связанные с ними операции с данными.
<i>middleware</i>	Содержит промежуточные модули <i>Express</i> , такие как авторизация, проверка ролей, логирование и глобальная обработка ошибок.
<i>routes</i>	Содержит маршруты <i>Express</i> для обработки <i>HTTP</i> -запросов к различным сущностям, включая статьи, скидки, пользователей, приемки и пр.
<i>swagger</i>	Содержит конфигурацию для автоматической генерации <i>Swagger</i> -документации <i>API</i> .
<i>uploads</i>	Содержит статические изображения, загружаемые пользователями, а также иллюстрации для визуального представления объектов (например, бутылки, бумага и т.д.).
<i>validations</i>	Содержит схемы валидации данных на основе <i>Joi</i> . Используется для проверки корректности данных, поступающих в контроллеры.

Такое разделение обеспечивает модульность, расширяемость и удобство сопровождения серверной части веб-приложения.

Описание контроллеров приложения представлено в таблице 3.3.

Таблица 3.3 – Описание контроллеров приложения

Название	Описание
<i>ArticlesController</i>	Предназначен для работы со статьями
<i>AuthController</i>	Предназначен для работы с авторизацией и регистрацией пользователей
<i>Check_weightsController</i>	Предназначен для работы с проверкой веса
<i>DiscountsController</i>	Предназначен для работы со скидками
<i>KeysController</i>	Предназначен для работы с секретными ключами
<i>MarksController</i>	Предназначен для работы с видами вторсырья
<i>PointsController</i>	Предназначен для работы с пунктами приема вторсырья
<i>Promo_codesController</i>	Предназначен для работы с промокодами
<i>RatingsController</i>	Предназначен для работы с комментариями
<i>ReceptionsController</i>	Предназначен для работы с приемом вторсырья
<i>UsersController</i>	Предназначен для работы с информацией о пользователях

Маршрутизаторы (или роутеры) в *Node.js* играют важную роль в обработке входящих запросов с определенными *URL*-адресами в серверной части приложения. Они выполняют функцию маршрутизации запросов к соответствующим контроллерам или обработчикам.

Основная цель маршрутизатора заключается в определении, какой контроллер или обработчик должен быть вызван для обработки конкретного запроса и передачи ему необходимых параметров или данных из запроса.

Использование маршрутизаторов позволяет структурировать обработку запросов в приложении путем разделения их на отдельные маршруты с соответствующими *URL*-шаблонами. Это дает возможность эффективно управлять и обрабатывать различные типы запросов (*GET*, *POST*, *DELETE* и другие) для разных *URL*-адресов. Их описание предоставлено в таблице 3.4.

Таблица 3.4 – Описание маршрутизаторов приложения

Название	Описание
<i>ArticlesRouter</i>	Маршрутизатор для работы со статьями
<i>AuthRouter</i>	Маршрутизатор для работы с авторизацией и регистрацией
<i>Check_weightsRouter</i>	Маршрутизатор для работы с проверкой веса
<i>DiscountsRouter</i>	Маршрутизатор для работы со скидками
<i>KeysRouter</i>	Маршрутизатор для работы с секретными ключами
<i>MarksRouter</i>	Маршрутизатор для работы с вторсырьем
<i>PointsRouter</i>	Маршрутизатор для работы с пунктами приема вторсырья
<i>Promo_codesRouter</i>	Маршрутизатор для работы с промокодами
<i>RatingsRouter</i>	Маршрутизатор для работы с комментариями
<i>ReceptionsRouter</i>	Маршрутизатор для работы с приемом вторсырья
<i>UsersRouter</i>	Маршрутизатор для работы с информацией о пользователях

Веб-приложение построено на основе такой схемы коммуникации между роутером и контроллером. Когда клиент отправляет запрос на сервер, маршрутизатор перенаправляет этот запрос на соответствующий контроллер для его обработки. Контроллер выполняет необходимые действия, обрабатывает данные и формирует ответ. Затем этот ответ отправляется обратно клиенту. Таким образом, через взаимодействие между роутером и контроллером осуществляется обработка запросов и обеспечивается функциональность веб-приложения.

Таблица соответствия маршрутов контроллерам в исходном коде представлена в таблице 3.5.

Таблица 3.5 – Контроллеры и функции маршрутов

URL	Метод	Контроллер	Метод контроллера	Описание
1	2	3	4	5
/login	POST	AuthController	LoginUser	Авторизация пользователя
/register	POST	AuthController	RegisterUser	Регистрация пользователя
/articles	GET	ArticlesController	getArticles	Получение всех статей

Продолжение таблицы 3.5

1	2	3	4	5
/articles	POST	ArticlesController	addArticles	Добавление статьи
/articles/\${id}	DELETE	ArticlesController	deleteArticles	Удаление статьи
/articles/\${updatedArticles.id}	PUT	ArticlesController	updateArticles	Изменение статьи
/like/\${id}	PUT	ArticlesController	like	Лайк к статье
/weight	POST	Check_weightsController	addWeight	Добавление проверки веса
/editWeight	PUT	Check_weightsController	editWeight	Изменение проверки веса
/discounts/\${id}	DELETE	DiscountsController	deleteDiscounts	Удаление скидки
/discounts	POST	DiscountsController	addDiscounts	Добавление скидки
/discounts/\${updatedDiscount.id}	PUT	DiscountsController	editDiscounts	Изменение скидки
/user/viewDiscounts	GET	DiscountsController	viewUserDiscounts	Получение скидок доступных пользователю
/used/discounts/\${id}	PUT	DiscountsController	usedMyDiscounts	Использование скидки
/marks/\${id}	DELETE	MarksController	deleteMarks	Удаление вида вторсырья
/marks	POST	MarksController	addMarks	Добавления вида вторсырья
/marks/\${updatedMark.id}	PUT	MarksController	editMarks	Изменение вида вторсырья
/points	GET	PointsController	getPoints	Получение всех пунктов приема
/points/\${id}	DELETE	PointsController	deletePoints	Удаление пункта приема вторсырья
/points/\${updatesPoint.id}	PUT	PointsController	editPoints	Изменение существующего пункта приема вторсырья
/keys	POST	KeysController	addKeys	Добавление секретного ключа для пункта приема
/points/key/\${updatedPointK.id}	PUT	PointsController	editPointsKey	Изменение секретного ключа для пункта приема
/ratings/\${articles_id}	POST	RatingsController	addRatings	Добавление комментария
/ratings/\${id}	DELETE	RatingsController	deleteRatings	Удаление комментариев

Продолжение таблицы 3.5

1	2	3	4	5
/receptions	POST	ReceptionsController	Receptions	Сдача вторсырья

При передаче данных между клиентом и сервером используется формат *JSON (JavaScript Object Notation)*.

3.6 Реализация функциональности

В соответствии с диаграммой вариантов использования, приведенной в ДП 01.00.ГЧ, в коде были реализованы функции, доступные пользователям.

3.6.1 Регистрация

Функция *RegisterUser* предназначена для регистрации новых пользователей в системе. Это асинхронный метод, который обрабатывает *HTTP POST*-запросы. Его задача — принять данные, отправленные клиентом, обработать их, создать нового пользователя в базе данных, сгенерировать ссылку активации и отправить её на указанный пользователем email. Функция представлена в приложении В.

Если пользователь с таким *email* уже существует, возвращается сообщение об ошибке. Если проверка проходит успешно, создается новая запись пользователя с ролями и статусом неактивного аккаунта. Одновременно генерируется уникальная ссылка активации, которая отправляется на указанный *email*.

3.6.2 Авторизация

Функция *LoginUser* отвечает за процесс авторизации пользователя в системе. Функция представлена в листинге 3.3.

```

LoginUser: async (req, res) => {
  try {
    // ... валидация ...
  } else {
    const accessToken = jwt.sign({ id:
i_user.id, username: i_user.username, role: i_user.role },
accessKey, { expiresIn: 30 * 60 });
    const refreshToken = jwt.sign({ id:
i_user.id, username: i_user.username, role: i_user.role },
refreshKey, { expiresIn: 24 * 60 * 60 });
  }
});
}
} catch (err) { console.log(err);
res.json({ message: 'Не удалось авторизоваться' }); },

```

Листинг 3.3 – Фрагмент реализации метода авторизации

Если пользователь найден, осуществляется проверка правильности введенного пароля. В случае некорректного пароля возвращается сообщение об ошибке. При успешной проверке генерируются токены *Access Token* и *Refresh Token*, которые упаковываются в *cookies* для последующей работы клиента с системой.

3.6.3 Просмотр пунктов приема

Функция *getPoints* предназначена для извлечения пунктов приема из базы данных и формирования ответа в формате JSON. Функция представлена в листинге 3.3.

```
getPoints: async (req, res) => {
  try {
    const points = await db.models.Points.findAll({
      attributes: ["id", "address", "time_of_work",
"key_id", "admin_id", "link_to_map", "point_name"],
    })
    if (!points) {
      return res.json({ message: "Пунктов нет" })
    } else {
      res.json({ points })
    }
  } catch (error) {
    console.log(error);
    res.json({
      message: 'Не удалось найти пункт сдачи',
    });
  }
}
```

Листинг 3.3 – Реализация метода для извлечения пунктов приема

Если в результате выполнения запроса пункты приема не найдены, записывается предупреждение в лог о пустом списке пунктов, и возвращается сообщение «Пунктов нет».

3.6.4 Отметка сдачи вторсырья

Функция *Receptions* предназначена для начисления баллов пользователю за сдачу вторсырья. Она проверяет действительность ключей станции и веса, рассчитывает количество баллов на основе веса сдаваемого материала, обновляет баллы пользователя, а затем создает запись об операции в базе данных. Функция представлена в приложении В.

С помощью полученных данных рассчитывается количество баллов, которое будет начислено пользователю, исходя из веса сдаваемого материала. Затем обновляются баллы пользователя в базе данных, и ключ веса помечается как использованный.

3.6.5 Обмен баллов

Функция *usedMyDiscounts* предназначена для того, чтобы позволить пользователю использовать скидку, уменьшив количество баллов, связанных с этой скидкой, и применив соответствующий промокод. Функция представлена в приложении В.

Функция начинается с получения текущего количества баллов пользователя и данных о скидке, которую он пытается использовать, используя ID, переданный в параметрах запроса. Затем рассчитывается количество новых баллов пользователя, вычитаемых за скидку, и эти баллы обновляются в базе данных.

В завершение функция отправляет ответ с обновленными баллами пользователя и сообщением о том, что скидка была успешно использована.

3.6.6 Добавление статьи

Функция *addArticles* предназначена для добавления новой статьи в базу данных. Сначала проверяется наличие статьи с таким же заголовком в базе данных. Если заголовок уже существует, в логах фиксируется предупреждение, и клиенту возвращается сообщение об этом. Функция представлена в листинге 3.4.

```
addArticles: async (req, res) => {
  try {
    const v_check_title = await
db.models.Articles.findOne({
      where: { title: req.body.title },
    })
    const v_b_image_url = req.body.image_url;
    if (v_check_title == null) {
      logger.info(`Attempting to add article with
title: ${req.body.title}`);
      if (v_b_image_url != undefined) {
        const article = await
db.models.Articles.create({
          author: req.userId,
          title: req.body.title,
          text: req.body.text,
          image_url: req.body.image_url,
          date_of_pub: Date.now(),
        })
      }
    }
  } catch (error) {
    console.log(error);
    res.json({
      message: 'Не удалось добавить статью'
    });
  },
}
```

Листинг 3.4 – Реализация метода для добавления статьи

Если заголовок уникален, начинается процесс добавления статьи. Проверяется наличие *URL* изображения в запросе. Если изображение указано, статья создаётся с полем *image_url*, содержащим значение из запроса. Если изображение отсутствует, поле *image_url* заполняется пустой строкой.

3.6.7 Добавление пункта приема

Функция `addPoints` предназначена для добавления нового пункта приема отходов. Она проверяет наличие адреса и названия пункта на уникальность, создает запись для нового пункта в базе данных, связывает его с соответствующими типами отходов, а также помечает связанный ключ как использованный. Функция представлена в листинге 3.5.

```
addPoints: async (req, res) => {
  try {
    const rubbishTypes = req.body.rubbish.split(',');
    const freeKey = await db.models.Keys.findOne({ where: {
is_used: 0 } });
    const newPoint = await db.models.Points.create({
      address: req.body.address,
      time_of_work: req.body.time_of_work,
      key_id: freeKey.id,
      admin_id: req.userId,
      link_to_map: req.body.link_to_map,
      point_name: req.body.point_name,
    });
    await db.models.Keys.update({ is_used: 1 }, { where: {
id: freeKey.id } });
    for (const type of rubbishTypes) {
      const rubbish = await db.models.Marks.findOne({
where: { rubbish: type } });
      await db.models.Points_marks.create({
        points_id: newPoint.id,
        marks_id: rubbish.id,
      });
    }
  } catch (err) {
    res.json({ message: 'Не удалось добавить пункт сдачи
отходов' });
  },
},
```

Листинг 3.5 – Реализация метода для добавления пункта приема

После всех успешных операций функция возвращает сообщение об успешном добавлении пункта сдачи отходов.

3.7 Функции промежуточной обработки

Промежуточные обработчики являются функциями, разработанными для промежуточной обработки запросов в приложении. Они расположены в

папке *utils*. Функции промежуточной обработки имеют доступ к объекту запроса (*request*), объекту ответа (*response*) и следующей функции обработки в цепочке «запрос-ответ» в приложении.

Они выполняются перед выполнением основной функции обработки запроса и могут выполнять различные задачи, такие как проверка аутентификации, валидация данных, установка заголовков и другие операции, необходимые для промежуточной обработки запроса.

Промежуточные обработчики позволяют разбить обработку запросов на отдельные этапы и обеспечить модульность и гибкость в разработке приложений.

Описание промежуточных обработчиков представлено в таблице 3.6.

Таблица 3.6 – Описание промежуточных обработчиков приложения

Название	Описание
<i>checkAuth</i>	Содержит проверку авторизован пользователь или нет
<i>checkRole</i>	Содержит проверку на наличие у текущего пользователя роли «Администратора»
<i>HandleErrors</i>	Содержит вывод ошибок валидации

Для примера в листинге 3.6 приведен исходный код промежуточного обработчика *checkRole*. Данный обработчик обеспечивает проверку пользователя на наличие роли администратора.

```
const token = (req.headers.authorization || '')
  .replace(/Bearer\s?/, ''); if (!token) {
  return res.status(403).json({
    message: 'Нет доступа(поль)',
  }); } else {const decoded = jwt.verify(token, accessKey);
  req.role = decoded.role; const v_role = decoded.role ext();}
```

Листинг 3.6 – Промежуточный обработчик *checkRole*

Если текущая функция промежуточной обработки не завершает цикл «запрос-ответ», она должна вызвать функцию *next()* для передачи управления следующей функции обработки.

Вызов промежуточных обработчиков происходит из контроллеров, которые выполняют роль маршрутизатора, перехватывая запросы с определенными *URI*, вызывая промежуточные обработчики и передавая запрос дальше по цепочке к основным функциям обработки.

3.8 Реализация отправки писем

Для осуществления отправки электронных писем в приложении была использована библиотека *NodeMailer*. Эта популярная библиотека для *Node.js* предоставляет простой и понятный код для отправки электронных писем.

Для начала использования *NodeMailer* необходимо установить модуль *npm nodemailer*. Затем создается объект *transporter*, который содержит всю

конфигурацию *SMTP*. Для его создания используется метод *createTransport()*, принимающий объект с параметрами, включающими:

- *host*: адрес почтового сервера, который будет использоваться для отправки писем;
- *port*: порт почтового сервера (по умолчанию 25, 465 или 587);
- *secure*: логическое значение, установленное в *true*, если используется *SSL* (в таком случае значение порта должно быть 465);
- *auth*: объект с полями *user* и *pass*, содержащий логин и пароль почтового аккаунта.

Пример реализации метода *createTransport()* приведен на листинге 3.7.

```
const transporter = nodemailer.createTransport({
  host: "smtp.yandex.com",
  port: 465,
  secure: true,
  auth: { user: 'dimatruba2004@yandex.ru',
    pass: 'akllfknlkhqfbhtl'}});
```

Листинг 3.7 – Пример реализации метода *createTransport()*

В разрабатываемом проекте в качестве транспорта для отправки сообщений был использован сервис *Yandex*. В поле *auth* были указаны почтовый адрес и пароль созданного аккаунта на *yandex.ru*.

Для отправки письма используется метод *sendMail()*. Он принимает объект *message* с определенной структурой, которая включает:

- *to*: адрес получателя;
- *subject*: тема сообщения;
- *text*: текст сообщения;
- *html*: текст сообщения в формате *HTML*.

Пример формирования письма и сама отправка представлен на листинге 3.8.

```
const send_mail = req.body.email;
const send_link = 'http://localhost:8082/activate/' +
  v_activation_link;
const mailOptions = {
  from: dimatruba2004@yandex.ru, to: send_mail,
  subject: 'Активация аккаунта на http://localhost:8082/',
  html: `<div> <h1>Для активации перейдите по ссылке</h1>
    <a href="${send_link}">${send_link}</a></div>`
}
let info = await transporter.sendMail(mailOptions)
```

Листинг 3.8 – Пример формирования и отправки письма

Эта структура является основной, но не обязательной, и может быть изменена в зависимости от требований и предпочтений пользователей в отношении вида конечного письма.

3.9 Реализация клиентской части приложения

Фреймворк *React*, ориентированный на функциональные компоненты, оперирует двумя ключевыми понятиями — компонентами и хуками, формируя иерархическую структуру проекта за счет взаимного использования компонентов и применения хуков.

В таблице 3.7 представлено краткое описание директорий клиентской части веб-приложения.

Таблица 3.7 – Описание директорий клиентской части веб-приложения

Директория	Описание
<i>components</i>	Содержит компоненты приложения, такие как кнопки, элементы навигации, элементы управления, а также отдельные структурные элементы. Также включает элементы для отображения различных сущностей, таких как статьи или комментарии.
<i>style</i>	Содержит <i>CSS</i> -стили, используемые для стилизации отдельных компонентов.
<i>image</i>	Содержит статические изображения и <i>SVG</i> -иконки, используемые в интерфейсе.
<i>pages</i>	Содержит страницы приложения, соответствующие различным маршрутам, такие как главная страница, страницы для добавления, обновления и просмотра статей, скидок, пунктов приема и других сущностей. Эти страницы обрабатывают основные пользовательские действия.
<i>redux</i>	Реализует управление состоянием приложения. Включает директории для различных фиچ приложения, таких как управление статьями, комментариями, скидками, пунктами приема и другими сущностями. Содержит отдельные <i>slice</i> для обработки состояния и <i>store.js</i> для конфигурации хранилища.
<i>utils</i>	Содержит вспомогательные модули и утилиты.

Каждая директория выполняет строго определённую функцию: от визуального отображения элементов и страниц до управления состоянием приложения и обработки вспомогательной логики.

Компоненты в проекте играют центральную роль, предоставляя гибкие и переиспользуемые элементы пользовательского интерфейса. Они включают как небольшие элементы (например, кнопки), так и более сложные структуры (например, формы или меню). В таблица 3.7 представлено описание компонент.

Таблица 3.8 – Описание компонент

Компонента	Описание
1	2
AllDiscountItem	Отображает отдельный элемент скидки, предоставляя информацию о доступной скидке для пользователей.
ArticleItem	Отображает информацию о статье, включая её содержимое и автора. Используется в списке статей.
Button	Универсальная кнопка, стилизованная для использования в различных частях приложения.
CommentItem	Отображает отдельный комментарий, включающий текст и данные о пользователе, оставившем комментарий.

Продолжение таблицы 3.8

1	2
DiscountItem	Отображает данные о скидке, включая её название, описание и количество требуемых баллов.
Header	Реализует верхний колонтитул приложения, отображая название и основные навигационные элементы.
Layout	Компонент для управления общей структурой страниц, объединяющий шапку, меню и контент.
MarksItem	Отображает данные о виде вторсырья, включая название, категорию и другие детали.
MenuItem	Отображает отдельный пункт меню для навигации по страницам приложения.
MobileMenu	Реализует адаптивное меню для мобильной версии приложения.
Navbar	Реализует навигационную панель с основными ссылками и элементами управления.
NavItam	Компонент для создания отдельного элемента в навигации.
NavMenu	Отображает выпадающее меню с основными разделами приложения.
PointItem	Отображает данные о пункте приема, включая название, адрес и контактную информацию.
PointMarkItem	Реализует отображение данных о связи между пунктом приема и видом вторсырья.
PromoCodeItem	Отображает отдельный промокод с описанием и связанными скидками.
RecycleCamera	Реализует функциональность камеры для распознавания и отправки информации о вторсырье.
Wrapper	Обеспечивает структурирование компонентов, добавляя базовую разметку или стили для детей.

Разделение на мелкие и крупные компоненты позволяет эффективно управлять как отдельными элементами интерфейса, так и целыми разделами приложения. Благодаря такой архитектуре обеспечивается гибкость в изменении внешнего вида и логики без необходимости вносить изменения во всё приложение целиком.

3.9.1 Реализация хранилища при помощи библиотеки **Redux ToolKit**

В приложении данные хранятся и управляются с использованием библиотеки *Redux Toolkit (RTK)*. Конфигурация хранилища объединяет стандартные редьюсеры и редьюсеры *RTK Query*. Основной механизм хранения базируется на `configureStore`, который подключает все редьюсеры, *middleware* для обработки запросов и расширяет функциональность *RTK Query*.

Настроенные редьюсеры обрабатывают состояние и запросы следующим образом:

- стандартные редьюсеры (*authSlice*, *userSlice* и т.д.) управляют локальным состоянием приложения, связанным с конкретными сущностями;
- *RTK Query* редьюсеры (например, *receptionSlice*) обеспечивают автоматическое кэширование, инвалидацию данных и управление состоянием загрузки.

Описание всех редьюсеров представлено в таблице 3.9.

Таблица 3.9 – Описание редьюсеров

Редьюсер	Описание
<i>authSlice</i>	Управляет состоянием аутентификации пользователя.
<i>userSlice</i>	Сохраняет данные о пользователях.
<i>articleSlice</i>	Обрабатывает статьи.
<i>commentSlice</i>	Управляет состоянием комментариев.
<i>discountSlice</i>	Обеспечивает управление скидками.
<i>markSlice</i>	Управляет данными о видах вторсырья.
<i>pointSlice</i>	Обрабатывает пункты приема.
<i>point_markSlice</i>	Управляет связью между пунктами приема и видами вторсырья.
<i>promo_codeSlice</i>	Обеспечивает управление промокодами.
<i>receptionSlice</i>	Работает с данными о сдаче вторсырья.
<i>secretkeySlice</i>	Управляет секретными ключами для пунктов приема.
<i>weightSlice</i>	Обрабатывает записи о проверке веса видов вторсырья.

Использование *Redux Toolkit* и *RTK Query* позволяет минимизировать шаблонный код, упростить работу с асинхронными запросами и обеспечить высокую производительность за счёт встроенного кэширования и автоматической инвалидации данных. Такая архитектура облегчает сопровождение проекта, ускоряет разработку новых функций и улучшает пользовательский опыт благодаря эффективной и стабильной работе с данными.

3.9.2 Маршрутизация в клиентской части

Для обеспечения правильного получения данных в приложении используется *Axios API*. *Axios* предоставляет удобный *JavaScript* интерфейс для работы с *HTTP* запросами и ответами. Он позволяет легко и ясно выполнять асинхронные запросы к ресурсам по сети. *Axios* является современным и мощным инструментом, который поддерживается всеми современными браузерами.

Вот пример использования вызова *axios*, представленный в листинге 3.9.

```
const { data } = await axios.post('/register', {
  username,
  email,
  password,
})
```

Листинг 3.9 – Пример отправки запроса

В данном примере показан процесс регистрации пользователя: React-приложение отправляет запрос через *Axios* при нажатии кнопки, сервер обрабатывает запрос через соответствующий контроллер и возвращает ответ, который клиент использует для обновления интерфейса или сохранения данных. Листинг помещения данных в хранилище изображен в приложении Г.

Таким образом, взаимодействие между серверной и клиентской частями приложения происходит посредством отправки запросов с клиента на

сервер и обработки этих запросов на сервере с последующей передачей ответа обратно клиенту.

3.10 Контейнеризация

Docker – популярное кроссплатформенное решение для контейнеризации, позволяющее запускать приложения на любой ОС с установленным *Docker*. Оно поддерживает оркестрацию контейнеров, обеспечивая управление их жизненным циклом. Для работы приложения требуются конфигурационные файлы двух типов: *Dockerfile* для создания образов и *docker-compose.yml* для оркестрации контейнеров. Пример конфигурационного файла *Docker*, который нужен для создания контейнера клиентской части, приведен в листинге 3.10.

```
FROM node:16
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
EXPOSE 3000
CMD ["npm", "start"]
```

Листинг 3.10 – Пример конфигурационного файла *Docker*

Разработанное приложение использует *docker-compose.yml* для одновременного запуска клиентского и серверного контейнеров. Этот файл определяет зависимости, переменные окружения и проброс портов, обеспечивая согласованную работу компонентов и упрощая развертывание на любых платформах с *Docker*. На листинге 3.11 приведен пример запуска серверной части приложения.

```
backend:
  build: ./ecosort-backend
  ports:
    - "8082:8082"
  environment:
    - DOCKER_ENV=true
    - DB_HOST=host.docker.internal
    - DB_USER=root
    - DB_PASSWORD=1111
    - DB_NAME=ecosort
  volumes:
    - ./ecosort-backend:/app
    - /app/node_modules
  depends_on:
    - frontend
```

Листинг 3.11 – Пример части конфигурационного файла *docker compose*

Здесь указывается путь к исходному коду сервера, пробрасывается порт 8082, подключаются необходимые переменные окружения для доступа к базе данных, а также монтируются тома. Параметр *depends_on* означает, что серверная часть будет запускаться после клиентской.

Аналогично конфигурируется и *frontend*-сервис, который работает на порту 3000. Такой способ настройки позволяет упростить локальную разработку и обеспечить стабильность окружения.

Таким образом, можно легко и быстро разворачивать данное приложение на любой инфраструктуре и ОС, в которой есть поддержка контейнеризации.

3.11 Выводы по разделу

В процессе разработки веб-приложения были применены современные технологии и архитектурные принципы, обеспечивающие высокую производительность, гибкость и удобство использования системы. Серверная часть реализована на платформе *Node.js* с использованием фреймворка *Express.js*, что гарантирует эффективную обработку запросов и масштабируемость решения. Для хранения данных о пользователях, пунктах приема и видах вторсырья использована реляционная СУБД *MySQL*.

При проектировании системы учитывалась возможность интеграции со сторонними сервисами, включая *SMTP*-сервер для работы с электронной почтой и механизмы загрузки изображений. Архитектура приложения построена по модульному принципу с использованием актуальных библиотек как для клиентской, так и для серверной части.

Веб-приложение предоставляет полный набор функциональных возможностей для трех категорий пользователей: гостей, зарегистрированных пользователей и администраторов. Общее количество реализованных функций составило 29, охватывающих все аспекты работы системы. В рамках разработки были успешно реализованы как серверные компоненты (*HTTP*-сервер, работа с базой данных), так и клиентская часть на основе *React*-приложения, обеспечивающая удобный пользовательский интерфейс.

Для взаимодействия с серверным *API* используется библиотека *axios*, которая обеспечивает удобную отправку *HTTP*-запросов и обработку ответов.

4 Тестирование веб-приложения

4.1 Негативное тестирование страницы входа

Тестирование, которое использует сценарии, соответствующие непредвиденному поведению тестируемой системы, называется негативным тестированием. Примерами такого тестирования могут быть исключительные ситуации или неправильные данные. В данном случае, мы протестируем страницу входа для пользователя, так как некоторые функции приложения доступны только авторизованным пользователям.

На экране входа в приложение присутствуют поля для ввода электронной почты и пароля, а также кнопка входа в приложение, которая не появится пока оба поля пусты. Начнем тестирование, оставив поле пароля пустым, результат выполнения данного теста представлен на рисунке 4.1.

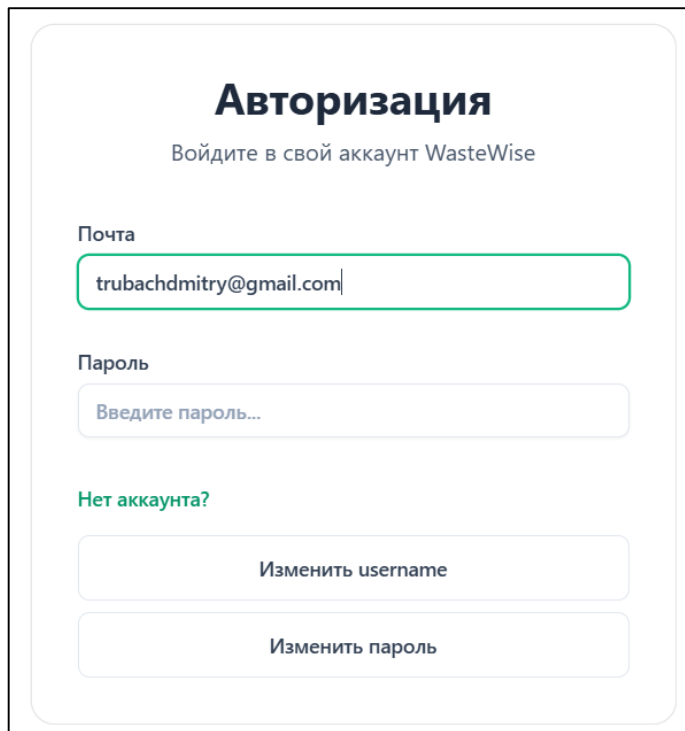


Рисунок 4.1 – Валидация формы входа

В соответствии с планом, кнопка «Войти» не будет отображаться, если хотя бы одно из полей «Почта» или «Пароль» остается пустым. Кроме того, если в поле «Почта» вводятся некорректные данные, такие как неправильный формат электронной почты или недостаточная длина строки. На рисунке 4.2 показан пример того, как будет выглядеть сообщение об ошибке валидации.

					ДП 04.00.ПЗ				
		ФИО	Подпись	Дата					
Разраб.		Трубач Д.С.			4 Тестирование веб-приложения		Лит.	Лист	Листов
Пров.		Мушук А.Н.					У	1	11
							БГТУ 1-40 01 01, 2025		
Н. контр.		Мушук А.Н.							
Утв.		Смелов В.В.							

The screenshot shows a login form titled "Авторизация" (Authorization) with the subtitle "Войдите в свой аккаунт WasteWise" (Log in to your WasteWise account). The form includes fields for "Почта" (Email) and "Пароль" (Password). The email field contains "trubachd@gmail.....com". Below the fields is a green "Войти" (Log in) button. Underneath the button is a link "Нет аккаунта?" (No account?). Below the link are two buttons: "Изменить username" (Change username) and "Изменить пароль" (Change password). A red error message box on the right side of the form reads "Неверный формат почты" (Incorrect email format).

Рисунок 4.2 – Валидация поля «Почта»

Также в форме присутствует вариант проверки на несуществующую почту. Результат выполнения этого теста показан на рисунке 4.3.

The screenshot shows the same login form as in Figure 4.2. The email field contains "trubachd@gmail.com". The "Войти" button is green. The "Нет аккаунта?" link is present. The "Изменить username" and "Изменить пароль" buttons are also present. A red error message box on the right side of the form reads "Неверная почта или пароль" (Incorrect email or password).

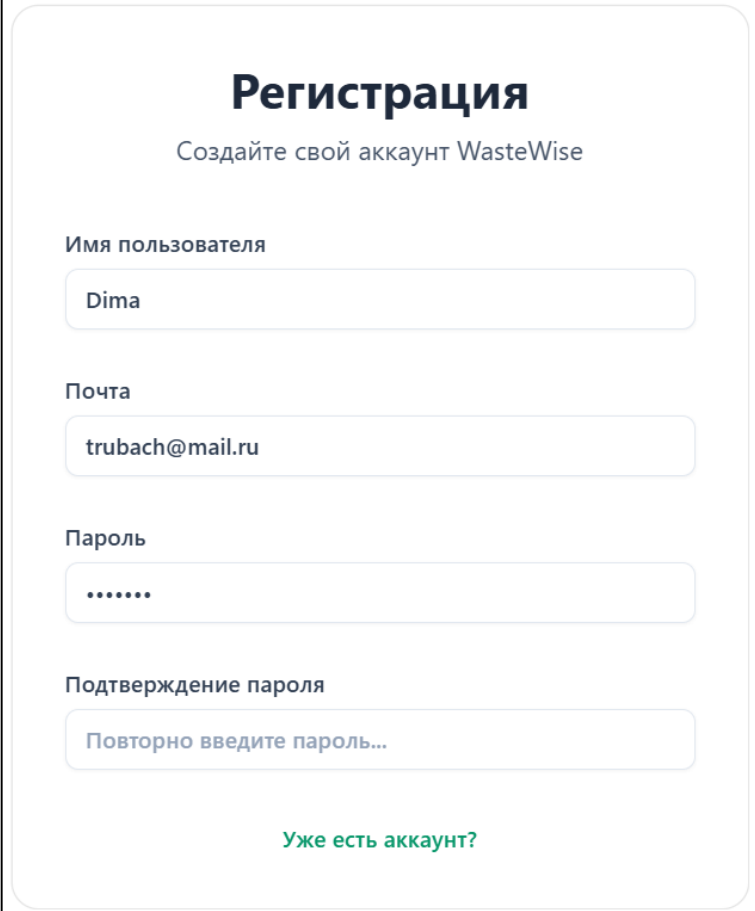
Рисунок 4.3 – Попытка входа с несуществующей почтой

Веб-приложение разработано с учетом обработки ошибочных ситуаций, связанных с вводом неправильных данных пользователем. Одним из примеров

таких ситуаций является неверная почта или пароль, введенные при авторизации. При вводе неверных данных, веб-приложение немедленно реагирует и информирует пользователя об ошибке. Такое уведомление является важным механизмом обратной связи, который помогает пользователю понять, что введенные им данные не прошли проверку и не соответствуют требованиям системы.

4.2 Негативное тестирование страницы регистрации

На экране регистрации присутствуют три обязательных поля: имя пользователя, почта и пароль. Проверка заполнения этих полей работает по тому же принципу, что и проверка в форме авторизации. Оставляем хотя бы одно из полей пустым, результат выполнения этого теста показан на рисунке 4.4.



The screenshot shows a registration form titled "Регистрация" (Registration) with the subtitle "Создайте свой аккаунт WasteWise" (Create your WasteWise account). The form contains four input fields: "Имя пользователя" (Username) with the value "Dima", "Почта" (Email) with the value "trubach@mail.ru", "Пароль" (Password) with masked characters ".....", and "Подтверждение пароля" (Confirm password) with the placeholder text "Повторно введите пароль..." (Re-enter password...). Below the fields is a green link that says "Уже есть аккаунт?" (Already have an account?). The form is enclosed in a light gray rounded rectangle.

Рисунок 4.4 – Валидация формы регистрации

Аналогично с формой авторизации, для формы регистрации предусмотрено валидационное правило ввода корректного *email*, который соответствует стандартам. Для этого в поле почта введем набор символов, которые нарушают одно или несколько ограничений валидации электронных адресов. Пример выполнения правил показан на рисунке 4.5.

The screenshot shows a registration form titled "Регистрация" (Registration) with the subtitle "Создайте свой аккаунт WasteWise". The form contains four input fields: "Имя пользователя" (Username) with the value "Dima", "Почта" (Email) with the value "trubach@mail.ru", "Пароль" (Password) with masked characters "*****", and "Подтверждение пароля" (Confirm password) with masked characters "*****". A green button labeled "Зарегистрироваться" (Register) is at the bottom. A red error message box on the right states: "Слишком маленький пароль, минимум 9 символов, Укажите имя пользователя" (Password is too short, minimum 9 characters, Specify username).

Рисунок 4.5 – Валидация одного из полей формы регистрации

В случае, если пользователь при регистрации укажет имя пользователя, которое уже используется другим пользователем выведется сообщение об ошибке «Имя пользователя занято, введите другое», аналогично для почты если она уже используется пользователь получит сообщение. Данное сообщение отображено на рисунке 4.6.

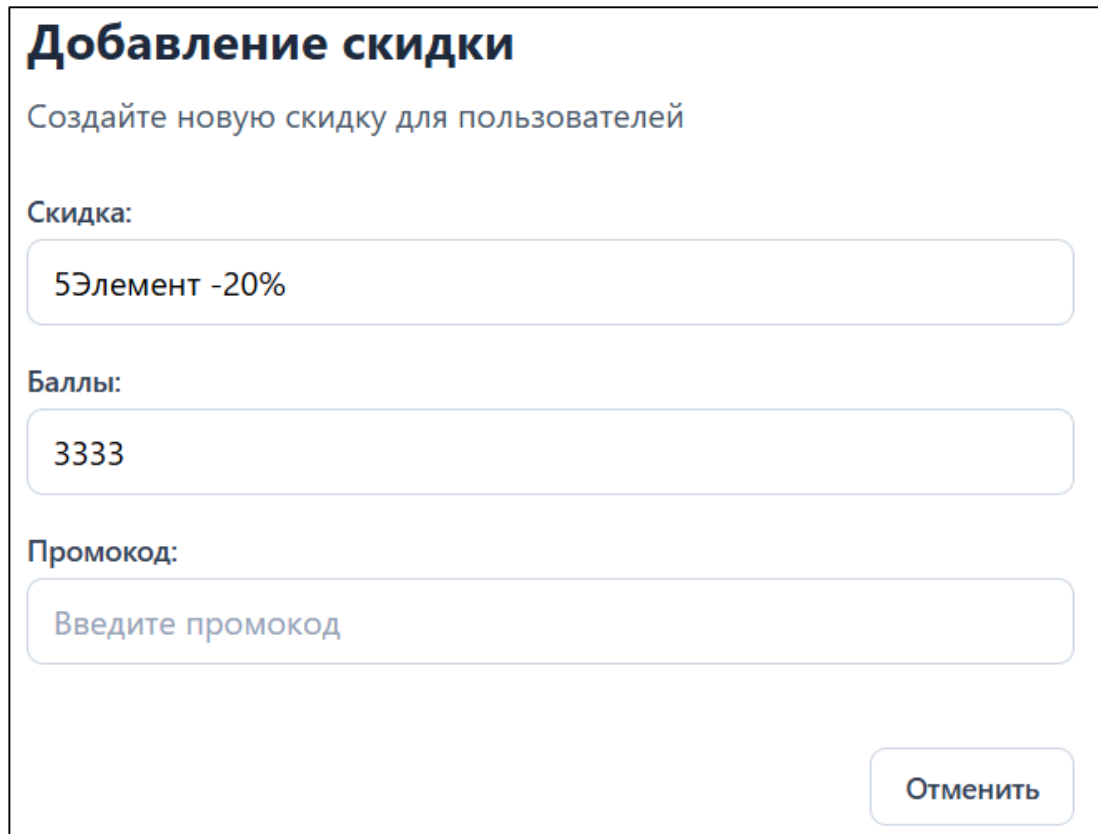
The screenshot shows the same registration form as Figure 4.5, but with different input values: "Имя пользователя" is "DimaTDS" and "Почта" is "trubachdmitry@gmail.com". The "Пароль" and "Подтверждение пароля" fields remain masked. The green "Зарегистрироваться" button is still present. A red error message box on the right states: "Почта занята, введите другую" (Email is taken, enter another).

Рисунок 4.6 – Регистрация на занятую почту

Таким образом организована валидация формы «Регистрация».

4.3 Негативное тестирование форм добавления и изменения вторсырья, скидок и пунктов приема

Как и в формах авторизации, регистрации в других формах необходимо заполнить все поля ввода, чтобы появились соответствующие кнопки «Добавить» или «Изменить», в зависимости от контекста страницы – добавления или изменения данных. Если хотя бы одно поле останется незаполненным, кнопки добавления или изменения не будут отображаться. Пример одной из таких форм изображен на рисунке 4.7.



Добавление скидки

Создайте новую скидку для пользователей

Скидка:

5Элемент -20%

Баллы:

3333

Промокод:

Введите промокод

Отменить

Рисунок 4.7 – Пример валидации формы добавления скидок

Также в форме добавления скидок присутствует валидация для каждого поля, для поля «Скидка» и «Промокод» – длина строки должна быть минимум 5 символов, а значение поля «Баллы» должно быть числом.

В форме добавления вторсырья также применяется валидация: поле «Вид вторсырья» требует ввода строки длиной не менее 3 символов, что исключает слишком общие или неполные названия, а поля «Новый вес» и «Баллы» проверяются на числовой формат, гарантируя ввод только допустимых числовых значений. Пример валидации поля на числовое значение можно увидеть на рисунке 4.8, где система выводит предупреждение при попытке ввода недопустимых символов. Таким образом, валидация в формах обеспечивает целостность данных, удобство взаимодействия и минимизацию ошибок при заполнении.

Добавление вторсырья

Прикрепить изображение:

Вторсырье:

Пластик

Баллы за 1 кг:

2500

Новая продукция из 1 кг:

не число

Добавить Отменить

! Вы ввели не число

Рисунок 4.8 – Пример валидации численного поля

В форме добавления проверки веса для вторсырья помимо валидации пустых, числовых полей и длины вводимого текста, есть еще проверка на существующий вид вторсырья, т.к. мы добавляем проверку веса только к тем видам вторсырья что есть в приложении. Пример данной проверки отображен на рисунке 4.9.

Добавление проверки веса

Вид отхода:

Камень

Вес:

15

Ключ:

rock

Добавить Отменить

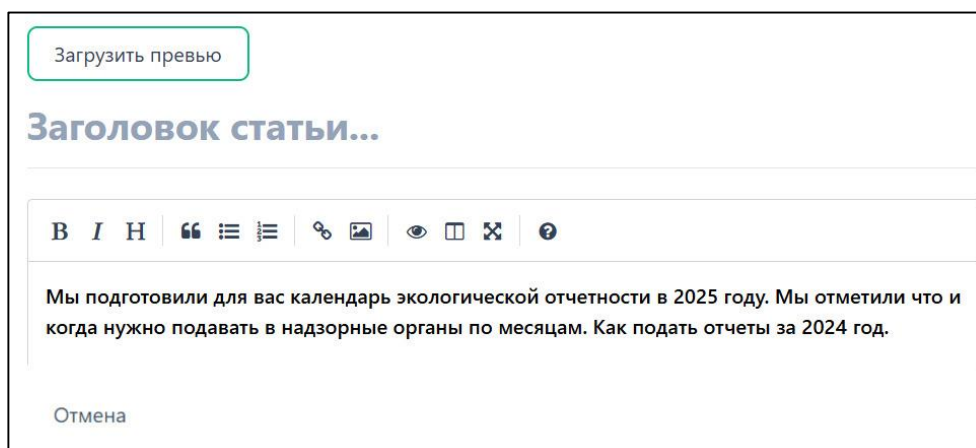
Такого вида вторсырья (Камень) нет, сначала добавьте его в список

Рисунок 4.9 – Проверка на существующий вид вторсырья

Формы добавления пункта приема, добавления секретного ключа и изменения вторсырья, пунктов приема и скидок также имеют аналогичные проверки.

4.4 Негативное тестирование формы добавления статьи

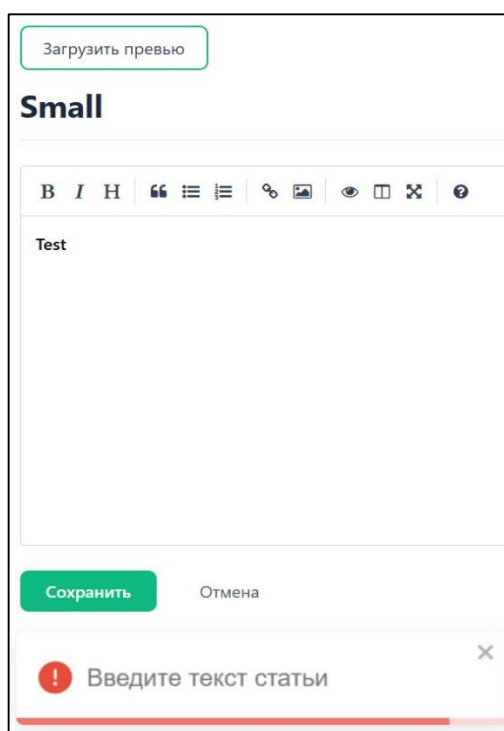
Формы добавления статьи имеет проверку на пустые поля, кнопка добавить не появится, пока поля «Заголовок» и «Текст» не будут заполнены. Пример данной проверки показан на рисунке 4.10.



The screenshot shows a form for adding an article. At the top, there is a button labeled "Загрузить превью" (Load preview). Below it, the title "Заголовок статьи..." (Article title...) is displayed. Under the title is a rich text editor with a toolbar containing icons for bold, italic, underline, list, link, image, eye, and other formatting options. The text area contains a paragraph: "Мы подготовили для вас календарь экологической отчетности в 2025 году. Мы отметили что и когда нужно подавать в надзорные органы по месяцам. Как подать отчеты за 2024 год." At the bottom left of the form, there is a button labeled "Отмена" (Cancel).

Рисунок 4.10 – Проверка на существующий вид вторсырья

Дополнительно реализованы правила валидации, которые определяют минимальную длину заголовка и текста статьи. Пример такой проверки представлен на рисунке 4.11.



The screenshot shows the same article addition form as in Figure 4.10, but with a validation error. The title field is labeled "Small". The text area contains the word "Test". At the bottom of the form, there are two buttons: "Сохранить" (Save) and "Отмена" (Cancel). Below the form, a red error message is displayed: "Введите текст статьи" (Enter article text), accompanied by a red exclamation mark icon and a close button (X).

Рисунок 4.11 – Валидация полей добавления статьи

Аналогичные правила валидации присутствуют в форме изменения статьи.

Кроме того, предусмотрено выполнение дополнительной проверки на сервере перед тем, как добавить новую статью. Она направлена на определение наличия уже существующей статьи с тем же заголовком, и, если такая статья обнаруживается, пользователь информируется об этом через всплывающее окно. Пример такой проверки проиллюстрирован на рисунке 4.12.

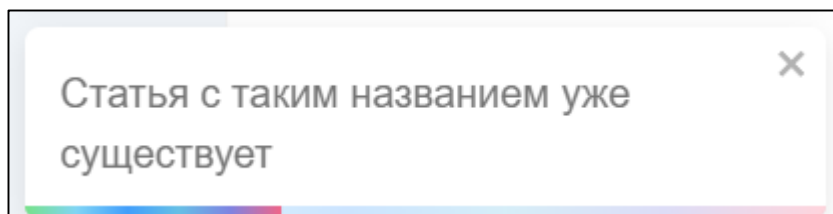


Рисунок 4.12 – Проверка на статью с таким же названием

Как и ожидалось сообщение о том, что статья с таким именем уже существует было выведено во всплывающее окно.

4.4 Негативное тестирование формы проема вторсырья

Форма приема, в которой пользователь вводит два секретных ключа для подтверждения сдачи вторсырья и дальнейшего начисления ему баллов имеет аналогичную проверку на не заполненные поля, кнопка «Добавить» будет неактивна пока пользователь не заполнит два необходимых поля, пример данной проверки отображен на рисунке 4.13.

Форма с заголовком «Прием отходов» и подсказкой «Введите ключи для подтверждения приема отходов». Она содержит два текстовых поля. Первое поле имеет метку «Секретный ключ пункта сдачи» и содержит текст «key1». Второе поле имеет метку «Секретный ключ для подтверждения веса» и подсказку «Введите проверку веса...». В нижнем правом углу расположены две кнопки: «Отменить» (серая) и «Добавить» (темно-синяя).

Рисунок 4.13 – Проверка на непустые поля

Также, если пользователь введет один из неверных ключей, он получит сообщение о том, что произошла ошибка. Пример ошибки отображен ниже на рисунке 4.14.

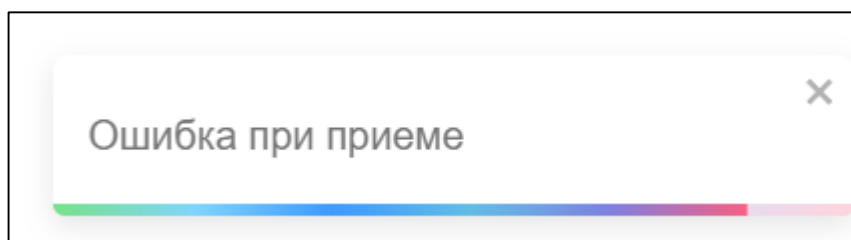


Рисунок 4.14 – Был введен неверный ключ

Таким образом выглядит валидация формы приема.

4.5 Функциональное тестирование

Функциональное тестирование направлено на проверку корректной работы функций приложения в различных ситуациях, включая стандартные и крайние случаи. Ниже приведена таблица 4.1, содержащая описание тестов, их ожидаемые результаты и статус выполнения.

Таблица 4.1 – Описание функциональных тестов

№	Функция	Описание теста	Ожидаемый результат	Статус теста
1	2	3	4	5
1	Регистрация с корректными данными	Действие: Ввести корректные данные (username, email, password) и отправить форму.	Данные пользователя заносятся в базу данных.	Пройден
2	Авторизация с корректными данными	Действие: Ввести корректные данные для входа (username, password) и нажать "Войти".	Пользователь успешно авторизуется и попадает на главную страницу.	Пройден
3	Просмотр статей	Действие: Перейти на страницу с категориями статей.	Статьи отображаются на странице.	Пройден
4	Просмотр пунктов приема	Действие: Открыть страницу с определенным видом вторсырья.	Все доступные пункты приёма для данного вида вторсырья отображаются на странице.	Пройден
5	Просмотр скидок	Действие: Открыть страницу с актуальными скидками.	Все скидки отображаются с актуальной информацией.	Пройден
6	Обмен баллов на скидки в сервисах	Действие: Выбрать подходящую скидку и нажать на кнопку "Промокод".	Баллы обменены на скидку, статус обновлён.	Пройден
7	Добавление статьи	Действие: Перейти в раздел создания статьи, ввести все необходимые данные и отправить.	Статья добавляется и отображается в списке статей.	Пройден

Продолжение таблицы 4.1

1	2	3	4	5
8	Изменение статьи	Действие: Перейти на страницу редактирования статьи, изменить её содержимое и сохранить изменения.	Статья изменяется, изменения отображаются на странице.	Пройден
9	Удаление статьи	Действие: Нажать кнопку "Удалить" рядом со статьёй и подтвердить действие.	Статья удаляется из списка и больше не отображается.	Пройден
10	Добавление комментария	Действие: Написать комментарий и отправить его под статьёй.	Комментарий добавляется и отображается на странице статьи.	Пройден
11	Удаление комментария	Действие: Нажать кнопку "Удалить" возле комментария.	Комментарий удаляется из статьи.	Пройден
12	Добавление отметки "Нравится"	Действие: Нажать кнопку "Нравится" под статьёй.	Отметка "Нравится" появляется, и количество лайков увеличивается.	Пройден
13	Отметка сдачи вторсырья	Действие: В разделе сдачи вторсырья отметить сдачу.	Результат сдачи вторсырья отображается на странице.	Пройден
14	Добавление пункта приёма	Действие: Перейти в форму добавления пункта приёма, заполнить поля и отправить.	Пункт приёма добавляется и отображается на карте/странице.	Пройден
15	Изменение времени работы	Действие: Перейти в раздел редактирования пункта приёма и изменить время работы.	Время работы пункта приёма изменяется.	Пройден
16	Удаление пункта приёма	Действие: Нажать кнопку "Удалить" на странице редактирования пункта приёма.	Пункт приёма удаляется из системы.	Пройден
17	Добавление ключей	Действие: Ввести ключи на странице добавления ключей и сохранить.	Ключи успешно добавляются в систему.	Пройден
18	Изменение ключей	Действие: Изменить данные ключей на странице редактирования и сохранить.	Изменённые ключи сохраняются и отображаются в списке.	Пройден
19	Добавление вторсырья	Действие: Ввести информацию о новом виде вторсырья и сохранить.	Новое вторсырье добавляется в систему.	Пройден
20	Изменение вторсырья	Действие: Изменить данные существующего вторсырья и сохранить.	Изменения вторсырья отображаются в системе.	Пройден
21	Удаление вторсырья	Действие: Удалить запись о вторсырье через интерфейс.	Вторсырье удаляется из базы данных.	Пройден
22	Добавление проверки веса	Действие: Добавить информацию о весе вторсырья и подтвердить.	Проверка веса сохраняется и отображается.	Пройден

Продолжение таблицы 4.1

1	2	3	4	5
23	Изменение проверки веса	Действие: Изменить введенный вес для вторсырья.	Измененный вес сохраняется в системе.	Пройден
24	Добавление скидки	Действие: Ввести параметры скидки и добавить её.	Скидка успешно добавляется и отображается в списке.	Пройден
25	Изменение скидки	Действие: Изменить параметры скидки и сохранить изменения.	Изменения скидки отображаются на странице.	Пройден
26	Удаление скидки	Действие: Удалить скидку через интерфейс управления.	Скидка удаляется из списка активных предложений.	Пройден
27	Редактирование любых статей	Действие: Перейти в раздел редактирования, изменить содержание статьи и сохранить.	Статья обновляется, и изменения отображаются на странице.	Пройден
28	Удаление любых статей	Действие: Нажать кнопку удаления на странице статьи.	Статья удаляется.	Пройден
29	Удаление комментариев пользователей	Действие: Удалить комментарий от пользователя через учетную запись администратора.	Комментарий удален.	Пройден

Было проведено 29 функциональных тестов, которые показали, что веб-приложение работает корректно.

4.6 Выводы по разделу

В ходе тестирования веб-приложения были успешно проверены все ключевые функции, включая регистрацию, авторизацию, работу с контентом и пунктами приема вторсырья. Проведенные 29 функциональных тестов подтвердили корректную работу системы: ошибки при авторизации обрабатываются должным образом, операции с контентом выполняются без сбоев, а функционал учета вторсырья и скидок полностью соответствует требованиям. Приложение стабильно работает во всех тестовых сценариях и корректно реагирует на ошибочные действия пользователей, выводя соответствующие уведомления. Результаты тестирования подтверждают готовность системы к эксплуатации.

Таким образом, все тесты прошли успешно, подтверждая правильность работы всех функций и корректную обработку ошибок.

5 Руководство пользователя

При запуске веб-приложения неавторизованный пользователь будет автоматически перенаправлен на главную страницу приложения. Главная страница является информационной точкой входа, где пользователь может получить краткую информацию о проекте.

Пользователь может ознакомиться с описанием проекта, его значимостью и преимуществами использования данного приложения. Для наглядного представления главной страницы веб-приложения, на рисунке 5.1 отображается внешний вид и расположение информационных элементов на странице.

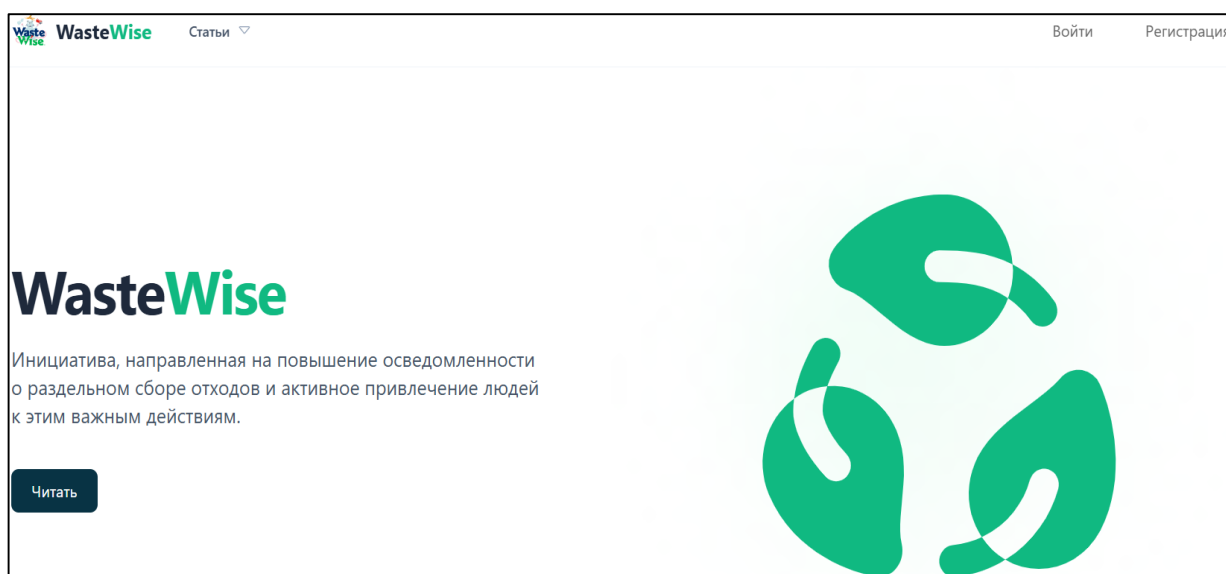


Рисунок 5.1 – Главная страница

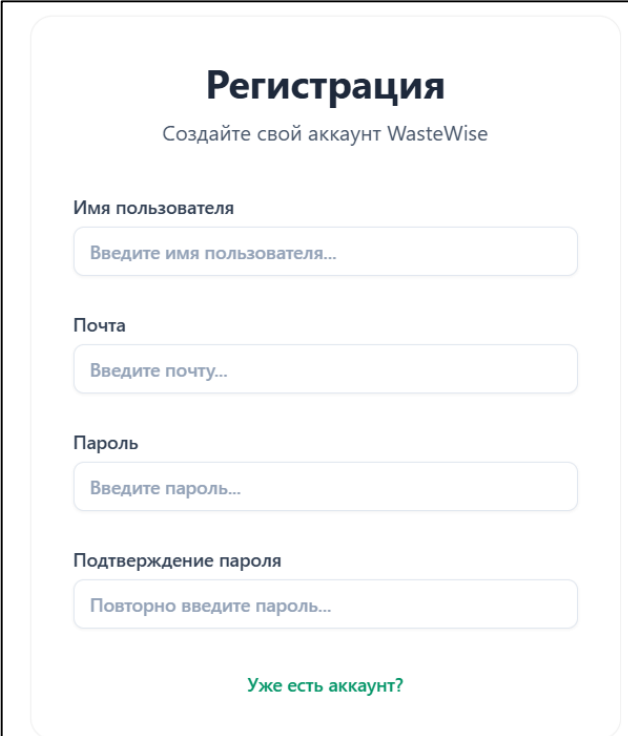
5.1 Регистрация

Если гость хочет получить более широкий функционал веб-приложения, ему необходимо пройти процедуру регистрации. Для этого следует нажать на кнопку «Регистрация», расположенную в навигационном меню интерфейса.

После перехода на страницу регистрации перед пользователем открывается интерактивная форма, требующая ввода персональных данных. Форма включает несколько обязательных полей: имя пользователя, действующий адрес электронной почты, который в дальнейшем будет использоваться в качестве логина для входа в систему, а также поле для создания надежного пароля с возможностью его подтверждения.

Форма регистрации представлена на рисунке 5.2.

					ДП 05.00.ПЗ				
		ФИО	Подпись	Дата					
Разраб.		Трубач Д.С.			5 Руководство пользователя	Лит.	Лист	Листов	
Пров.		Мушук А.Н.				У	1	19	
						БГТУ 1-40 01 01, 2025			
Н. контр.		Мушук А.Н.							
Утв.		Смелов В.В.							

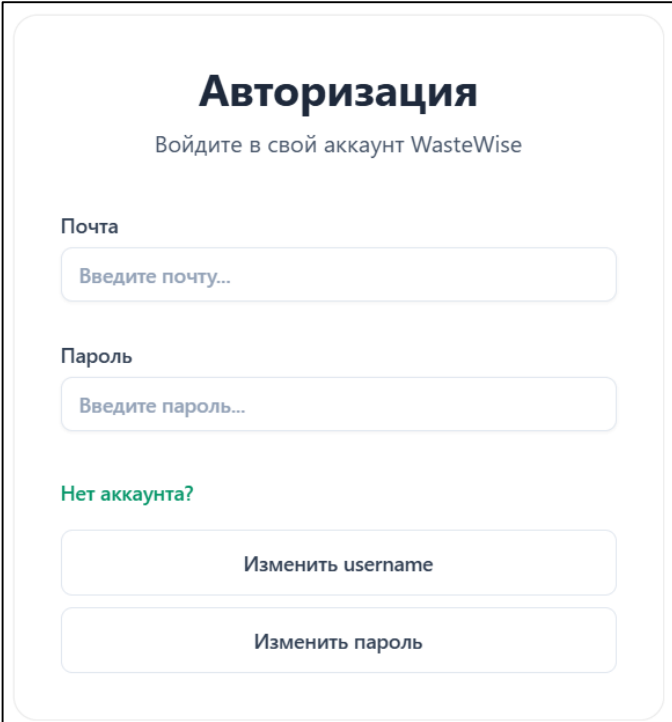


The registration form is titled "Регистрация" (Registration) with the subtitle "Создайте свой аккаунт WasteWise" (Create your WasteWise account). It contains four input fields: "Имя пользователя" (Username) with placeholder "Введите имя пользователя...", "Почта" (Email) with placeholder "Введите почту...", "Пароль" (Password) with placeholder "Введите пароль...", and "Подтверждение пароля" (Confirm password) with placeholder "Повторно введите пароль...". At the bottom, there is a green link "Уже есть аккаунт?" (Already have an account?).

Рисунок 5.2 – Форма регистрации

5.2 Авторизация

После успешной регистрации требуется выполнить вход на сайт, необходимо корректно заполнить данные email и пароль. Форма входа в аккаунт представлена на рисунке 5.3.



The authorization form is titled "Авторизация" (Authorization) with the subtitle "Войдите в свой аккаунт WasteWise" (Log in to your WasteWise account). It contains two input fields: "Почта" (Email) with placeholder "Введите почту..." and "Пароль" (Password) with placeholder "Введите пароль...". Below these fields is a green link "Нет аккаунта?" (No account?). At the bottom, there are two buttons: "Изменить username" (Change username) and "Изменить пароль" (Change password).

Рисунок 5.3 – Форма авторизации

5.3 Просмотр статей

После успешного входа на сайт пользователь может перейти к просмотру статей. Для этого необходимо выбрать соответствующий раздел в меню или на главной странице. Страница с просмотром статей представлена на рисунке 5.4.



Рисунок 5.4 – Просмотр списка статей

Пользователь может выбрать интересующую его статью из списка доступных материалов. На рисунке 5.5 видно, где после выбора статьи откроется страница с полным текстом статьи, где можно ознакомиться с содержанием.



Рисунок 5.5 – Просмотр статьи

На странице просмотра статьи пользователь может увидеть заголовок статьи, дату публикации, автора и полный текст статьи.

5.4 Просмотр пунктов приема

Пользователь также может просматривать пункты приема вторсырья. Для этого необходимо выбрать соответствующий вид вторсырья, и он увидит все пункты, которые относятся к данному виду. Страница с просмотром пунктов приема представлена на рисунке 5.6.

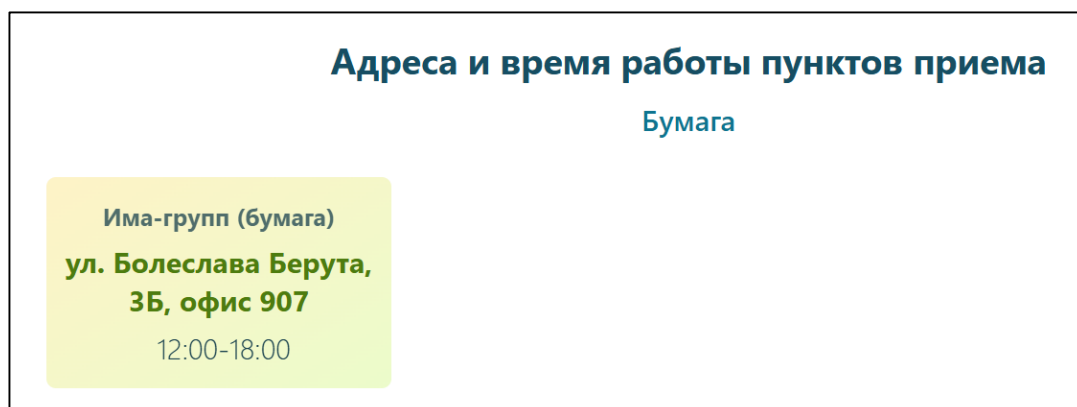


Рисунок 5.6 – Просмотр пунктов приема

На странице просмотра пунктов приема пользователь может увидеть список доступных пунктов приема вторсырья. Каждый пункт приема отображается с указанием его названия, адреса, времени работы и видов принимаемого вторсырья.

5.5 Просмотр скидок

Пользователь также может просматривать доступные скидки. Страница с просмотром скидок представлена на рисунке 5.7.

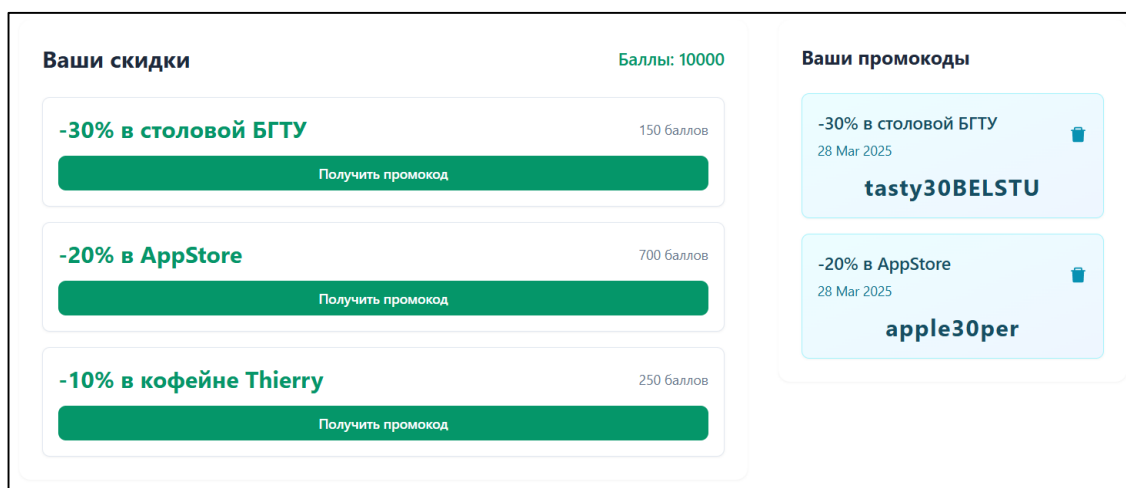


Рисунок 5.7 – Просмотр скидок

На странице просмотра скидок пользователь может увидеть список доступных скидок и акций.

5.6 Обмен баллов на скидки

На рисунке 5.7 показан интерфейс обмена накопленных баллов на скидки – пользователю необходимо выбрать нужную скидку и нажать кнопку «Получить промокод».

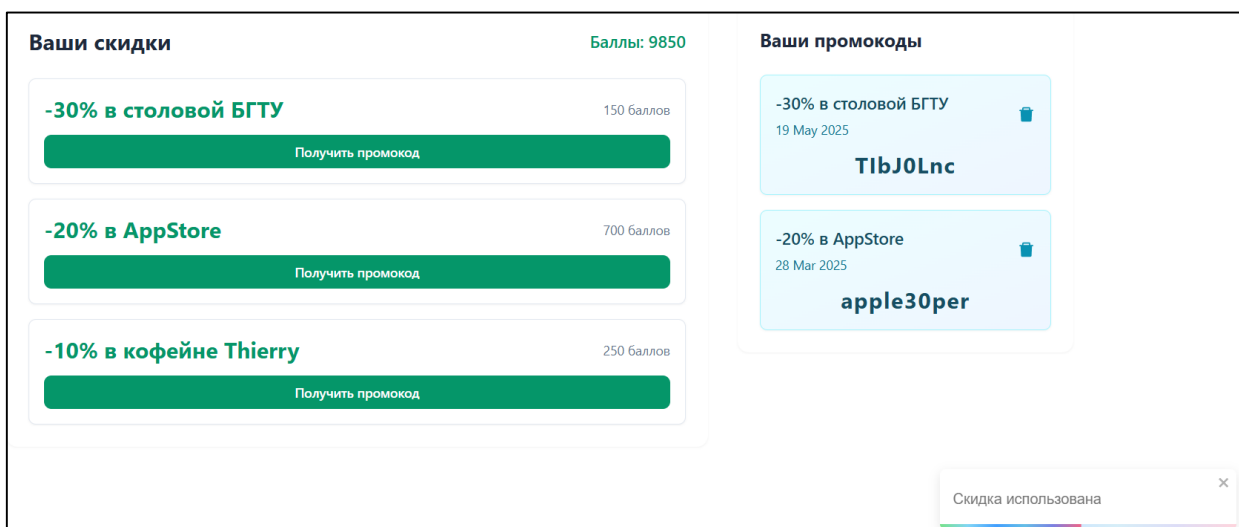


Рисунок 5.7 – Обмен баллов

Как можно видеть, обмен баллов произошел успешно и промокод отображается в правой части интерфейса.

5.7 Добавление статьи

Пользователь может добавлять новые статьи через специальный интерфейс. Для этого необходимо выбрать соответствующий раздел в меню или на главной странице.

Интерфейс добавления статьи представлен на рисунке 5.8.

На странице добавления статьи пользователь может загрузить превью статьи, ввести заголовок и текст статьи. Превью статьи можно загрузить, нажав на кнопку «Загрузить превью» и выбрав файл изображения. После загрузки превью отображается на странице, и пользователь может удалить его, нажав на кнопку «Удалить».

Пользователь также может ввести заголовок статьи в соответствующее поле и текст статьи в редактор. Редактор поддерживает форматирование текста, такие как жирный шрифт, курсив, списки и ссылки. После заполнения всех полей пользователь может сохранить статью, нажав на кнопку «Сохранить». Если все поля заполнены корректно, статья будет добавлена, и пользователь будет перенаправлен на страницу со списком статей.

Рисунок 5.8 – Интерфейс добавления статьи

Если пользователь хочет отменить добавление статьи, он может нажать на кнопку "Отмена", чтобы вернуться на главную страницу.

5.8 Изменение статьи

Пользователь может редактировать существующие статьи через специальный интерфейс. Интерфейс изменения статьи представлен на рисунке 5.9.

Рисунок 5.9 – Интерфейс изменения статьи

На странице изменения статьи пользователь может загрузить новое превью статьи, изменить заголовок и текст статьи. Превью статьи можно загрузить, нажав на кнопку «Загрузить превью» и выбрав файл изображения. После загрузки превью отображается на странице, и пользователь может удалить его, нажав на кнопку «Удалить».

Пользователь также может изменить заголовок статьи в соответствующем поле и текст статьи в редакторе. Редактор поддерживает форматирование текста, такие как жирный шрифт, курсив, списки и ссылки. После внесения всех изменений пользователь может сохранить статью, нажав на кнопку «Сохранить». Если все поля заполнены корректно, статья будет обновлена, и пользователь будет перенаправлен на страницу со списком статей.

Если пользователь хочет отменить изменения, он может нажать на кнопку «Отмена», чтобы вернуться на страницу со списком статей.

5.9 Удаление статьи

Рисунок 5.10 демонстрирует механизм удаления статей, реализованный для авторов и администраторов - соответствующая кнопка отображается на странице статьи и при наведении будет подсвечиваться красным цветом.

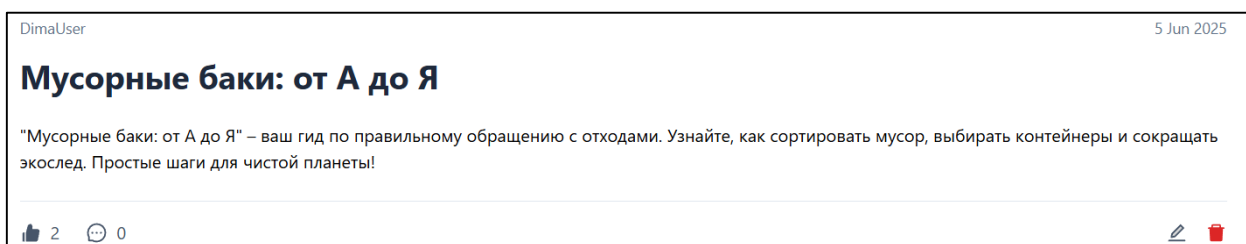


Рисунок 5.10 – Удаление статьи

5.10 Добавление комментария

Пользователь может добавлять комментарии к статьям, чтобы выразить свое мнение или задать вопросы. Для этого необходимо перейти на страницу статьи. На рисунке 5.11 отображена форма для добавления комментария.

 The image shows a form titled 'Добавить комментарий' (Add comment). At the top, there is a rich text editor with various icons for text formatting: bold (B), italic (I), underline (U), quote (”), list (bulleted and numbered), link (chain), image (picture), eye (visibility), and a speech bubble icon. Below the editor is a large text input area with the placeholder text 'Введите комментарий...'. At the bottom right of the form, there are two buttons: a light gray button labeled 'Отмена' (Cancel) and a green button labeled 'Сохранить' (Save).

Рисунок 5.11 – Добавление комментария

После добавления комментария пользователь увидит уведомление о том, что комментарий был успешно добавлен.

5.11 Удаление комментария

Пользователь может удалить свои комментарии или комментарии других пользователей, если он является автором комментария или имеет роль администратора. Это позволяет поддерживать чистоту и актуальность обсуждений под статьями. Для удаления комментария необходимо нажать на соответствующую иконку удаления, которая изображена на рисунке 5.12.

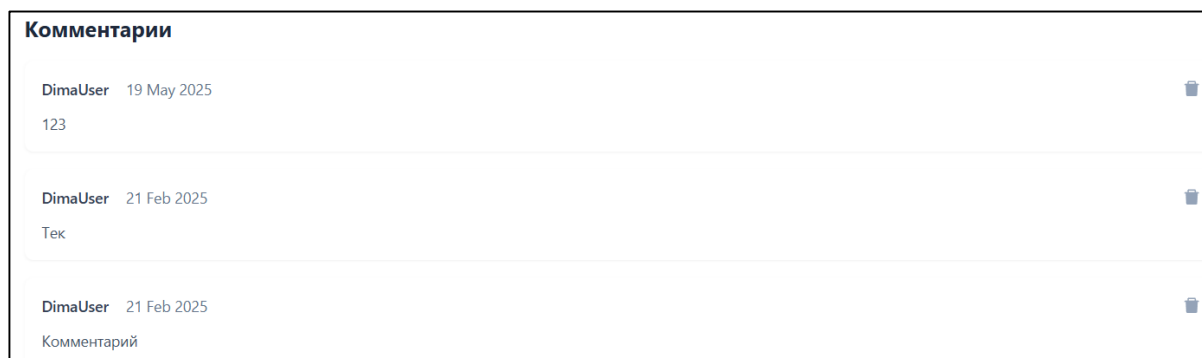


Рисунок 5.12 – Удаление комментария

После подтверждения удаления комментариев будет удален из системы, и пользователь увидит уведомление о том, что комментарий был успешно удален.

5.12 Добавление отметки «Нравится»

Пользователь может выразить свое одобрение статье, добавив отметку «Нравится». Чтобы добавить отметку «Нравится», пользователь должен нажать на кнопку с иконкой лайка (рисунок 5.13).

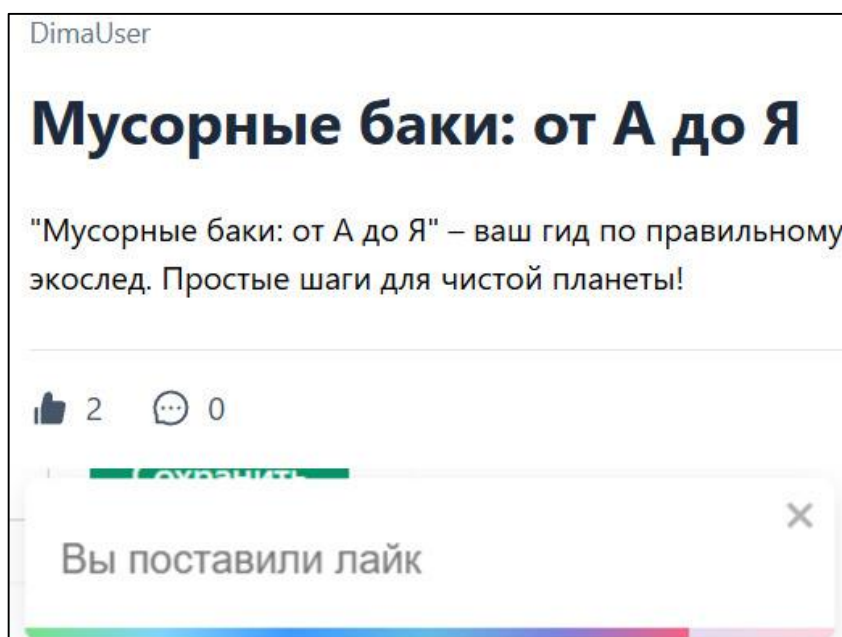


Рисунок 5.13 – Добавление отметки «Нравится»

Если пользователь не авторизован, он увидит уведомление с просьбой авторизоваться для оценки статьи.

После нажатия на кнопку отметка «Нравится» будет добавлена к статье, и количество лайков увеличится.

5.13 Отметка сдачи вторсырья

Пользователь может отметить сдачу вторсырья на пункте приема и получить баллы за это. Для этого необходимо выбрать соответствующий раздел в меню или на главной странице и перейти к форме сдачи вторсырья. Форма сдачи вторсырья представлена на рисунке 5.14.

Прием отходов

Введите ключи для подтверждения приема отходов

Секретный ключ пункта сдачи

Введите секретный ключ...

Секретный ключ для подтверждения веса

Введите проверку веса...

Отменить Добавить

Рисунок 5.14 – Форма сдачи вторсырья

На странице сдачи вторсырья пользователь может ввести ключ пункта приема и ключ веса. Если ключи действительны и соответствуют пункту приема, система проверяет вид вторсырья и его вес.

Результат сдачи вторсырья отображается на рисунке 5.15.

Прием отходов

Введите ключи для подтверждения приема отходов

10 кг новой продукции будет произведено

3000 балл(а/ов) вам начислено

12850 ваши баллы

Секретный ключ пункта сдачи

Введите секретный ключ...

Секретный ключ для подтверждения веса

Введите проверку веса...

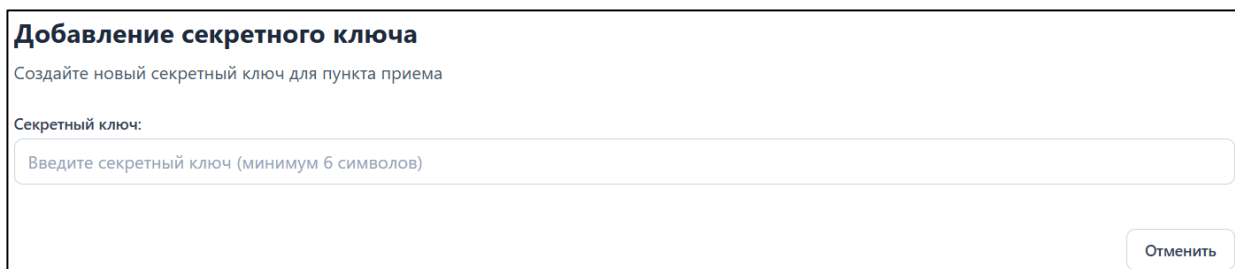
Отменить Добавить

Рисунок 5.15 – Результат сдачи

На странице результата сдачи пользователь увидит количество начисленных баллов, количество килограммов нового вторсырья и общее количество баллов на его счету.

5.14 Добавление ключей

Администратор может добавлять новые секретные ключи через специальный интерфейс для добавления пункта приема. Форма добавления секретного ключа представлена на рисунке 5.16.



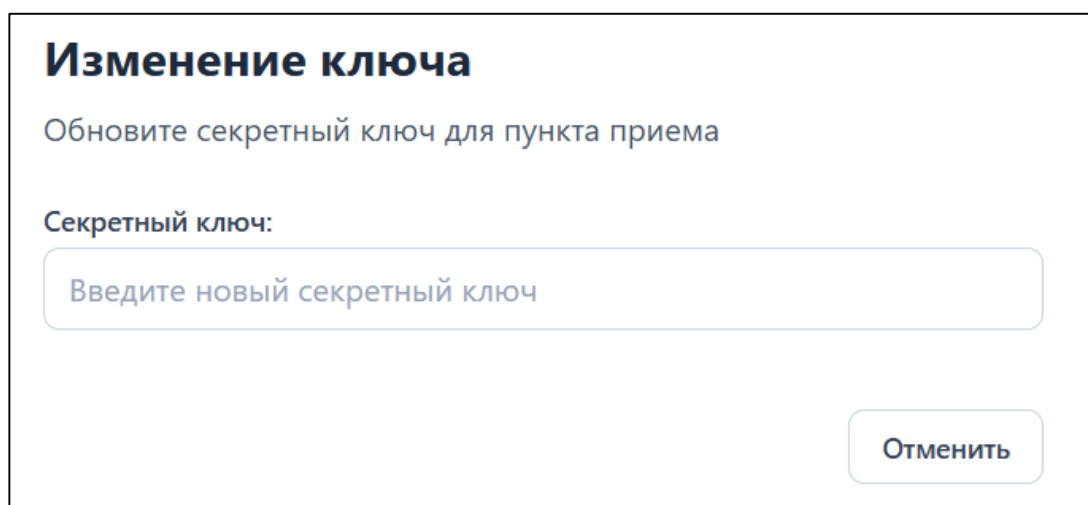
The screenshot shows a web form titled "Добавление секретного ключа" (Adding a secret key). Below the title is a subtitle "Создайте новый секретный ключ для пункта приема" (Create a new secret key for the reception point). The form contains a label "Секретный ключ:" followed by a text input field with the placeholder text "Введите секретный ключ (минимум 6 символов)". At the bottom right of the form is a button labeled "Отменить".

Рисунок 5.16 – Добавление секретного ключа

После нажатия на кнопку «Добавить» система проверяет, существует ли уже такой ключ в базе данных. Если ключ уникален, он будет добавлен в систему, и пользователь увидит уведомление о том, что ключ был успешно добавлен. Если ключ уже существует, пользователь увидит сообщение об ошибке и сможет ввести другой ключ.

5.15 Изменение ключей

Администратор может изменять секретные ключи, связанные с пунктами приема вторсырья. Форма изменения секретного ключа представлена на рисунке 5.17.



The screenshot shows a web form titled "Изменение ключа" (Changing a key). Below the title is a subtitle "Обновите секретный ключ для пункта приема" (Update the secret key for the reception point). The form contains a label "Секретный ключ:" followed by a text input field with the placeholder text "Введите новый секретный ключ". At the bottom right of the form is a button labeled "Отменить".

Рисунок 5.17 – Изменение секретного ключа

На странице изменения секретного ключа пользователь может ввести новый ключ для выбранного пункта приема. После ввода нового ключа пользователь может нажать на кнопку «Сохранить», чтобы обновить ключ в системе.

Если ключ уникален, он будет обновлен в системе, и пользователь увидит уведомление о том, что ключ был успешно обновлен. Если ключ уже используется, пользователь увидит сообщение об ошибке и сможет ввести другой ключ.

5.16 Добавление пункта приема

Администратор может добавлять новые пункты приема вторсырья через специальный интерфейс. Форма добавления пункта приема представлена на рисунке 5.18.

Добавление пункта приема
Создайте новый пункт приема отходов

Имя:
Введите имя пункта приема

Адрес:
Введите адрес пункта приема

Виды вторсырья:
Введите виды принимаемого вторсырья

Время работы:
--:-- --:--

Ссылка на карту:
Введите ссылку на карту

Отменить

Рисунок 5.18 – Форма добавления пункта приема

На странице добавления пункта приема администратор может ввести необходимую информацию о новом пункте приема. Форма включает несколько полей для заполнения: имя пункта приема, адрес, виды принимаемого вторсырья, время работы и ссылка на карту. Пользователь может заполнить эти поля, чтобы создать новый пункт приема.

Если администратор хочет отменить добавление пункта приема, он может нажать на кнопку «Отменить», чтобы очистить форму и вернуться к предыдущему состоянию.

5.17 Изменение времени работы пункта

Администратор может изменять время работы пунктов приема вторсырья. На странице с пунктами приема будет доступна кнопка для изменения времени работы. После нажатия на кнопку редактирования пользователь будет перенаправлен на страницу изменения времени работы пункта (рисунок 5.19).

Рисунок 5.19 – Изменение времени работы пункта

На этой странице пользователь может ввести новое время работы пункта и сохранить изменения. Если все поля заполнены корректно, время работы пункта будет обновлено, и пользователь увидит уведомление о успешном изменении.

5.18 Удаление пункта приема

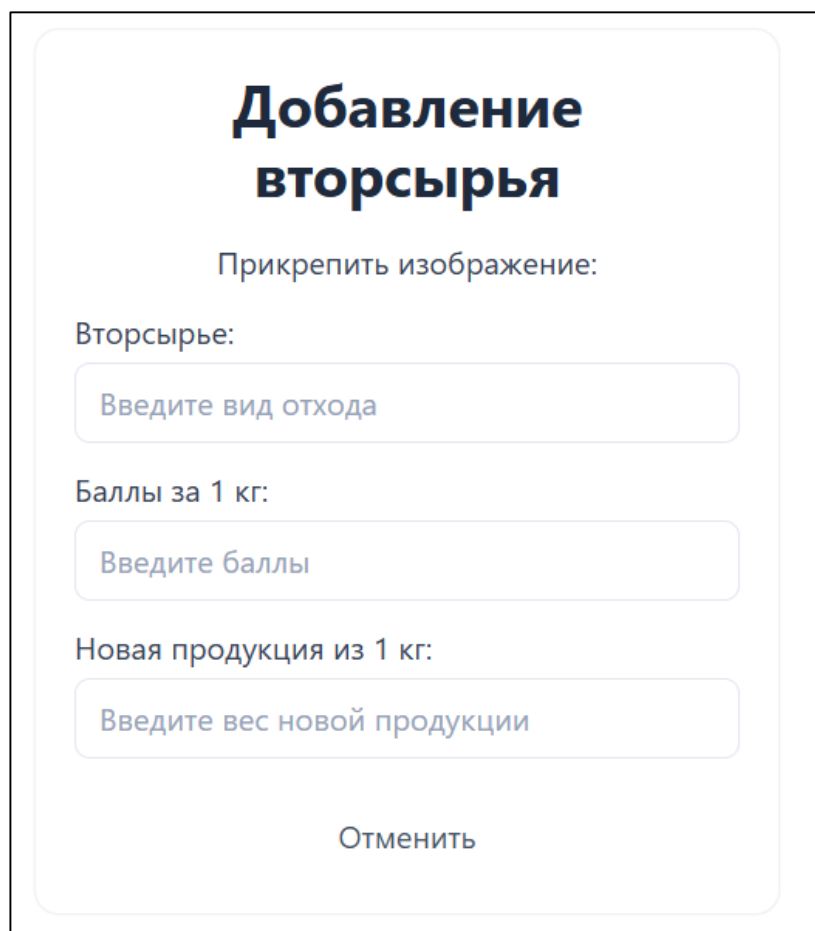
Администратор может удалять пункты приема вторсырья. На странице с пунктами приема будет доступна кнопка для удаления пункта (рисунок 5.20).

Рисунок 5.20 – Удаление пункта приема

После нажатия на кнопку удаления пункт приема будет удален из системы, и пользователь увидит уведомление о том, что пункт был успешно удален. Это позволяет администратору легко управлять списком пунктов приема и поддерживать его актуальность.

5.19 Добавление вида вторсырья

Администратор может добавлять новые виды вторсырья через специальный интерфейс. Форма добавления вида вторсырья представлена на рисунке 5.21.



**Добавление
вторсырья**

Прикрепить изображение:

Вторсырье:

Введите вид отхода

Баллы за 1 кг:

Введите баллы

Новая продукция из 1 кг:

Введите вес новой продукции

Отменить

Рисунок 5.21 – Форма добавления вида вторсырья

На странице добавления вида вторсырья пользователь может ввести информацию о новом виде вторсырья. Форма включает несколько полей для заполнения: название вторсырья, количество баллов за килограмм, количество новой продукции из килограмма и ссылка на изображение. Пользователь может заполнить эти поля, чтобы создать новый вид вторсырья.

После заполнения всех полей пользователь может нажать на кнопку «Добавить», чтобы сохранить информацию о новом виде вторсырья. Если все поля заполнены корректно, вид вторсырья будет добавлен в систему, и пользователь увидит уведомление о том, что вид вторсырья был успешно добавлен. Если такой вид вторсырья уже существует, пользователь увидит сообщение об ошибке и сможет ввести другой вид вторсырья.

Если пользователь хочет отменить добавление вида вторсырья, он может нажать на кнопку «Отменить», чтобы очистить форму и вернуться к предыдущему состоянию.


5.20 Изменение вида вторсырья

Администратор может изменять информацию о существующих видах вторсырья через специальный интерфейс. Форма изменения вида вторсырья представлена на рисунке 5.22.


Изменение вторсырья

Обновите информацию о виде отходов

Прикрепить изображение:



Загрузить файл



Удалить

Вторсырье:

Пластик

Баллы начисляемые за 1 кг:

150

Новая продукция из 1 кг:

0.9

Отменить

Изменить

Рисунок 5.22 – Форма изменения вида вторсырья

После внесения всех изменений пользователь может нажать на кнопку «Сохранить», чтобы обновить информацию о виде вторсырья в системе. Если все поля заполнены корректно, вид вторсырья будет обновлен, и пользователь увидит уведомление о том, что вид вторсырья был успешно обновлен.

Если возникнут ошибки валидации, пользователь увидит сообщения об ошибках и сможет исправить их.

5.21 Удаление вида вторсырья

Пользователь может удалять существующие виды вторсырья через специальный интерфейс. Для этого необходимо выбрать соответствующий раздел в меню или на главной странице и перейти к списку видов вторсырья.

Форма удаления вида вторсырья представлена на рисунке 5.23.



Рисунок 5.23 – Удаления вида вторсырья

На странице списка видов вторсырья каждый вид отображается с указанием его названия, количества баллов за килограмм, количества новой продукции из килограмма и ссылки на изображение. Если пользователь имеет права администратора, рядом с каждым видом вторсырья будет отображаться кнопка для удаления.

5.22 Добавление проверки веса

Пользователь может добавлять новые ключи для проверки веса вторсырья через специальный интерфейс. Форма добавления проверки веса представлена на рисунке 5.24.

A screenshot of a web form titled 'Добавление проверки веса' (Adding weight check). The form contains three input fields: 'Вид отхода:' (Waste type) with placeholder text 'Введите вид отхода', 'Вес:' (Weight) with placeholder text 'Введите вес', and 'Ключ:' (Key) with placeholder text 'Введите ключ'. At the bottom of the form is a button labeled 'Отменить' (Cancel).

Рисунок 5.24 – Добавление проверки веса

Если все поля заполнены корректно, ключ для проверки веса будет добавлен в систему, и администратор увидит уведомление о том, что ключ был успешно добавлен. Если такой ключ уже существует, администратор увидит сообщение об ошибке и сможет ввести новый ключ.

5.23 Изменение проверки веса

Пользователь может изменять существующие записи проверки веса вторсырья через специальный интерфейс. Форма изменения проверки веса представлена на рисунке 5.25.

ID	Вид вторсырья	Вес	Ключ	Действия
1	Бумага	10	paper1	Редактировать

Рисунок 5.25 – Добавление проверки вторсырья

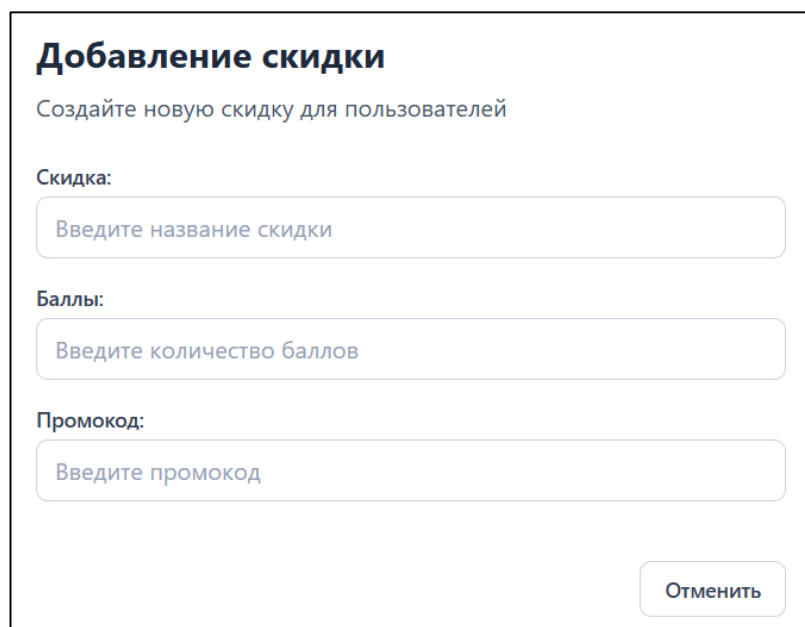
Для изменения записи проверки веса пользователь должен нажать на кнопку «Редактировать». После этого откроется модальное окно, где пользователь может ввести новые данные для вида вторсырья, веса и ключа. После внесения всех изменений пользователь может нажать на кнопку «Сохранить», чтобы обновить запись в системе. В случае ошибок система выведет соответствующее уведомление, и запись не будет обновлена до исправления всех несоответствий. Если все поля заполнены корректно, запись будет обновлена.

5.24 Добавление скидки

Пользователь может добавлять новые скидки через специальный интерфейс. Для этого необходимо выбрать соответствующий раздел в меню или на главной странице.

Форма добавления скидки представлена на рисунке 5.26.

На странице добавления скидки администратор может ввести информацию о новой скидке. Форма включает несколько полей для заполнения: размер скидки, количество баллов, необходимых для получения скидки, и промокод.



Добавление скидки
Создайте новую скидку для пользователей

Скидка:

Баллы:

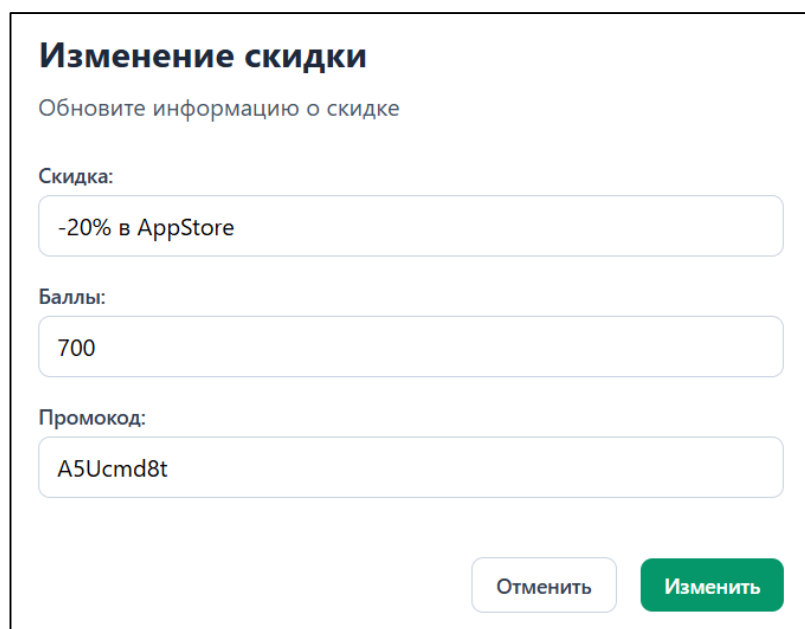
Промокод:

Рисунок 5.26 – Форма добавления скидки

После заполнения всех полей администратор может нажать на кнопку «Добавить», чтобы сохранить информацию о новой скидке.

5.25 Изменение скидки

Пользователь может изменять существующие скидки через специальный интерфейс. Форма изменения скидки представлена на рисунке 5.27.



Изменение скидки
Обновите информацию о скидке

Скидка:

Баллы:

Промокод:

Рисунок 5.27 – Форма добавления скидки

После внесения всех изменений пользователь может нажать на кнопку «Сохранить», чтобы обновить информацию о скидке в системе.

5.26 Удаление скидки

На рисунке 5.28 показан интерфейс, через который администратор может удалять существующие скидки, выбрав соответствующий раздел в меню управления.

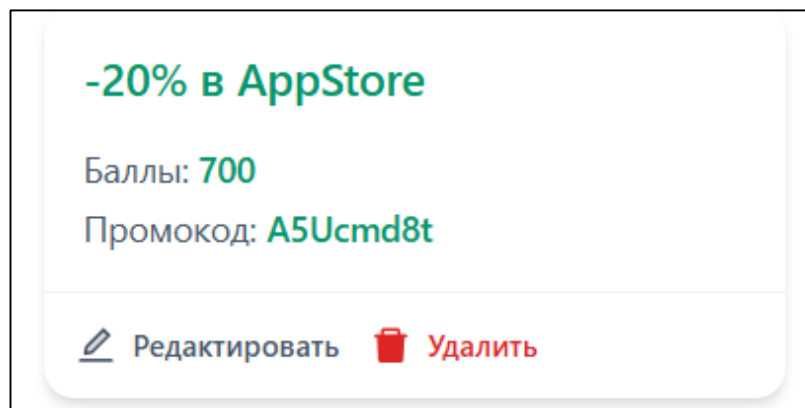


Рисунок 5.28 – Удаление скидки

Для удаления скидки пользователь должен нажать на кнопку удаления. После подтверждения скидка будет удалена из системы, и администратор увидит уведомление о том, что скидка была успешно удалена.

5.27 Редактирование любых статей

Администраторы имеют возможность редактировать любые статьи на платформе, независимо от того, кто является автором статьи (рисунок 5.29).

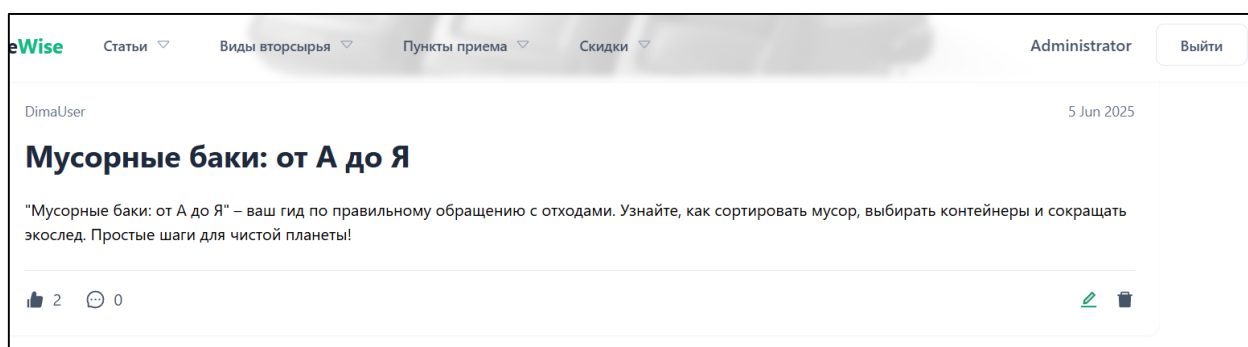


Рисунок 5.29 – Редактирование статьи пользователя

На странице статьи, если пользователь имеет роль администратора, рядом с каждой статьей будет отображаться кнопка для редактирования. При нажатии на эту кнопку система перенаправляет администратора в форму редактирования, где можно откорректировать заголовок, текст статьи, изображения и другие связанные данные. После внесения необходимых правок администратор сохраняет изменения, и обновленная версия статьи сразу становится доступной для просмотра всем пользователям.

5.28 Удаление любых статей и комментариев пользователей

Администраторы имеют возможность удалять любые статьи и комментарии на платформе, независимо от того, кто является автором статьи или комментария (рисунок 5.30).

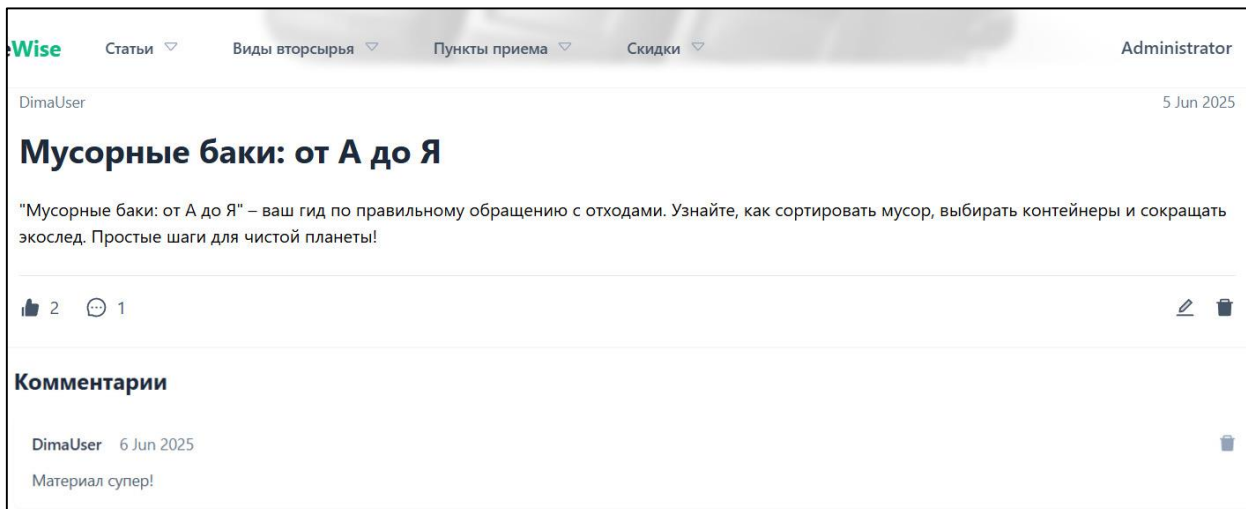


Рисунок 5.30 – Удаление статьи пользователя

На странице статьи, если пользователь имеет роль администратора, рядом с каждой статьей будет отображаться кнопка для удаления. Администратор может нажать на эту кнопку, чтобы удалить статью.

На странице статьи, где отображаются комментарии, если пользователь имеет роль администратора, рядом с каждым комментарием будет отображаться кнопка для удаления. Администратор может нажать на эту кнопку, чтобы удалить комментарий.

5.30 Выводы по разделу

В данном разделе было разработано руководство, описывающее действия пользователей системы: «Гость», «Пользователь», и «Администратор», с учетом уникальных функций каждой роли описанных в диаграмме вариантов использования. Каждое действие подробно описано и подкреплено скриншотами элементов интерфейса для эффективной работы с веб-приложением.

Таким образом рассмотрели, как взаимодействовать с системой от имени клиента и от имени администратора.

6 Технико-экономическое обоснование проекта

6.1 Общая характеристика разрабатываемого программного средства

Основной целью экономического раздела является обоснование экономической целесообразности разработки и последующей продажи веб-приложения, разработанного в рамках выпускной квалификационной работы. В данном разделе производится расчёт затрат на всех этапах разработки программного продукта, а также прогнозируется его рыночная стоимость и ожидаемая прибыль от реализации.

При выполнении данного проекта было разработано веб-приложение, предназначенное для привлечения пользователей к практике раздельного сбора бытовых отходов и их последующей сдачи в соответствующие пункты приема вторсырья.

Прежде всего, оно позволяет пользователям отмечать сданное вторсырье и за это получать баллы, которые впоследствии могут быть обменены на скидки в сервисах-партнерах приложения. Помимо этого, в приложении пользователи могут получать информацию о различных видах вторсырья и узнать, в каких пунктах приема вторсырья они могут сдать свои отходы.

В процессе создания приложения были применены *Node.js* для разработки серверной части и *React.js* для разработки клиентской части веб-приложения. Кроме того, для стилизации клиентской части была использована гибкая и эффективная библиотека *Tailwind CSS*. Основная цель разработки данного веб-приложения заключается в его дальнейшем коммерческом использовании.

Продажа разработанного веб-приложения и смежных прав на него заказчику является способом монетизации, который позволяет получить прибыль от интеллектуальной собственности и труда, вложенных в создание приложения.

6.2 Исходные данные для проведения расчетов

Источниками исходных данных для расчетов выступают действующие законы и нормативно-правовые акты.

Исходные данные для расчета стоимости приведены в таблице 6.1.

Таблица 6.1 – Исходные данные для расчетов

Наименование показателя	Условные обозначения	Норматив
1	2	3
Норматив дополнительной заработной платы, %	$H_{дз}$	10
Ставка отчислений в Фонд социальной защиты населения, %	$H_{фсзн}$	34

					ДП 06.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Трубач Д.С.			6 Технико-экономическое обоснование проекта	Лит.	Лист	Листов
Пров.		Муцук А.Н.				У	1	10
Консульт.		Познякова Л.С.				БГТУ 1-40 01 01, 2025		
Н. контр.		Муцук А.Н.						
Утв.		Смелов В.В.						

Продолжение таблицы 6.1

1	2	3
Ставка отчислений по обязательному страхованию в БРУСП «Белгосстрах», %	$H_{бгс}$	0,6
Норматив прочих прямых затрат, %	$H_{пз}$	20
Норматив накладных расходов, %	$H_{обп, обх}$	55
Норматив расходов на сопровождение и адаптацию, %	$H_{рса}$	15
Ставка НДС, %	$H_{ндс}$	20
Налог на прибыль, %	$H_{п}$	20

В ходе маркетингового анализа была определена стоимость разработки аналогичных веб-приложений для покупки жилой недвижимости при помощи калькулятора разработки веб-приложений «Majento». Результаты анализа представлены в таблице 6.2.

Таблица 6.2 – Маркетинговый анализ аналогов

Продукт-аналог	Источник	Стоимость, руб	Примечание
Зеленая карта	http://greenmap.by/	116 000	Включает функции интерактивной карты и экосервисов, требует интеграции с геолокацией и базами данных.
<i>RSbor</i>	https://rsbor.ru/	25 500	Имеет широкий функционал, включая регистрацию пользователей и приём заявок, что увеличивает сложность реализации.
<i>Target99</i>	https://target99.by/	56 000	Содержит аналитические инструменты и визуализацию данных, что требует дополнительной проработки интерфейса и логики.

Средняя стоимость разработки подобных решений варьируется от 25 500 руб. до 116 000 руб. Учитывая функциональные возможности разрабатываемого веб-приложения для раздельного сбора бытовых отходов, рыночная стоимость проекта оценивается в 70 750 руб.

6.3 Обоснование цены программного средства

6.3.1 Расчет затрат рабочего времени на разработку программного средства

Для начала необходимо посчитать время, которое было затрачено на разработку данного программного средства. Время на разработку в основном состоит из этапа анализа и проектирование, этапа самой разработки приложения, а также финального этапа в виде тестирования приложения. Все эти этапы

необходимо учитывать в равной пропорции, так как без них невозможно разработать качественное и конкурентоспособное приложение. В таблице 6.3 представлены затраты рабочего времени на разработку программного средства.

Таблица 6.3 – Затраты рабочего времени на разработку веб-приложения

Содержание работ	Исполнитель	Трудозатраты, часов
Изучение темы разрабатываемого продукта	Бизнес-аналитик	24
	Менеджер проекта	16
Анализ аналогичных решений (Зеленая карта, <i>RSbor</i> , <i>Target99</i>)	Бизнес-аналитик	32
	Менеджер проекта	24
Формирование требований к системе	Менеджер проекта	20
Разработка <i>UI/UX</i> дизайна и адаптивных макетов	<i>UI/UX</i> -дизайнер	80
Проектирование и реализация БД	Специалист по базам данных	40
Разработка серверной части (<i>Node.js</i> , <i>Express.js</i>)	Бэкенд-разработчик	160
Разработка клиентской части	Фронтенд-разработчик	150
Функциональное и нагрузочное тестирование	Тестировщик	70
Развёртывание и настройка инфраструктуры (<i>Docker</i>)	<i>DevOps</i> -инженер	40
Настройка системы и запуск в <i>production</i>	Системный администратор	32
	<i>DevOps</i> -инженер	32
Итого		720

На основании представленных данных общие трудозатраты на разработку веб-приложения составляют 720 часов, при этом наибольший объём работ приходится на этапы бэкенд и фронтенд разработки. Указанные трудозатраты формируют базис для расчёта себестоимости программного продукта.

6.3.2 Расчет основной заработной платы

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в сфере разработки и определение их часовых ставок. Источником данных служили открытые веб-порталы, различные форумы, официальная отчётность, а также общий средний уровень заработка в сфере информационных технологий в Республике Беларусь.

Основная заработная плата $C_{оз}$ будет рассчитываться по формуле 6.1.

$$C_{оз} = T_{раз} \cdot C_{зп}, \quad (6.1)$$

где $T_{раз}$ – трудоемкость, чел./час;

$C_{зп}$ – средняя часовая ставка, руб.

Результаты подсчетов представлены в таблице 6.4.

Таблица 6.4 – Расчет основной заработной платы специалистов

Исполнитель	Затраты рабочего времени, чел-часов	Средняя часовая ставка, руб/час	Основная заработная плата, руб.
Бизнес-аналитик	56	14	784
Менеджер проекта	60	17	1 020
UI/UX-дизайнер	80	12	960
Специалист по базам данных	40	12	480
Бэкенд-разработчик	160	16	2 560
Фронтенд-разработчик	150	15	2 250
Тестировщик	70	11	770
DevOps-инженер	72	13	936
Системный администратор	32	11	352
Всего	720		10 112

Таким образом, при разработке программного средства основная заработная плата всех специалистов составит 10 112 руб.

6.3.3 Расчет дополнительной заработной платы

Дополнительная заработная плата на конкретное ПС включает выплаты, предусмотренные законодательством о труде и определяется по нормативу в процентах к основной заработной плате. Таким образом дополнительная заработная плата $C_{дз}$ рассчитывается по формуле 6.2.

$$C_{дз} = \frac{C_{оз} \cdot N_{дз}}{100}, \quad (6.2)$$

где $C_{оз}$ – основная заработная плата, руб.;

$N_{дз}$ – норматив дополнительной заработной платы, %.

$$C_{дз} = 10\,112 \cdot 10 / 100 = 1\,011,20 \text{ руб.}$$

Дополнительная заработная плата составила 1 011,20 руб.

6.3.4 Расчет отчислений в фонд социальной защиты населения и по обязательному страхованию

Взносы в Фонд социальной защиты населения (ФСЗН) и в Белорусское республиканское унитарное страховое предприятие «Белгосстрах» рассчитываются согласно действующим законодательным нормам, исходя из установленного процента от фонда основной и дополнительной заработной платы исполнителей. Отчисления в ФСЗН $C_{фсзн}$ вычисляются по формуле 6.3. Аналогичный

принцип применяется при расчете взносов в «Белгосстрах». Процентные ставки регулярно пересматриваются и утверждаются соответствующими нормативными документами.

$$C_{\text{фсзн}} = \frac{(C_{\text{оз}} + C_{\text{дз}}) \cdot N_{\text{фсзн}}}{100}, \quad (6.3)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$C_{\text{дз}}$ – дополнительная заработная плата на конкретное ПС, руб.;

$N_{\text{фсзн}}$ – норматив отчислений в Фонд социальной защиты населения, %.

$$C_{\text{фсзн}} = (10\,112 + 1\,011,20) \cdot 34 / 100 = 3\,781,89 \text{ руб.}$$

Отчисления в БРУСП «Белгосстрах» $C_{\text{бгс}}$ вычисляются по формуле 6.4.

$$C_{\text{бгс}} = \frac{(C_{\text{оз}} + C_{\text{дз}}) \cdot N_{\text{бгс}}}{100}, \quad (6.4)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$C_{\text{дз}}$ – дополнительная заработная плата, руб.;

$N_{\text{бгс}}$ – норматив отчислений в БРУСП «Белгосстрах», %.

$$C_{\text{бгс}} = (10\,112 + 1\,011,20) \cdot 0,6 / 100 = 66,74 \text{ руб.}$$

Таким образом, отчисления в фонд социальной защиты населения и в БРУСП «Белгосстрах» составляют 3 781,89 руб. и 66,74 руб. соответственно.

6.3.5 Расчет суммы прочих прямых затрат

Расходы на конкретное программное средство $C_{\text{пз}}$ включают расходы на приобретение и подготовку специальной технической информации, платных сервисов тестирования и прочие операционные издержки, прямо относимые на проект и рассчитываются по формуле 6.5.

$$C_{\text{пз}} = \frac{C_{\text{оз}} \cdot N_{\text{пз}}}{100}, \quad (6.5)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$N_{\text{пз}}$ – норматив прочих затрат, %.

$$C_{\text{пз}} = 10\,112 \cdot 20 / 100 = 2\,022,40 \text{ руб.}$$

Таким образом, сумма прочих прямых затрат при разработке веб-приложения составила 2 022,40 руб.

6.3.6 Расчет суммы накладных расходов

Сумма накладных расходов $C_{\text{нр}}$ – произведение основной заработной платы исполнителей на конкретное программное средство на норматив накладных расходов в целом по организации, по формуле 6.6.

$$C_{\text{нр}} = \frac{C_{\text{оз}} \cdot N_{\text{обп, обх}}}{100}, \quad (6.6)$$

где $C_{\text{оз}}$ – основная заработная плата, руб;

$N_{\text{обп, обх}}$ – норматив накладных расходов, %.

$$C_{\text{нр}} = 10\,112 \cdot 55 / 100 = 5\,561,60 \text{ руб.}$$

Таким образом, сумма накладных расходов составила 5 561,60 руб.

6.3.7 Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства C_p определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, суммы прочих затрат и суммы накладных расходов, по формуле 6.7.

$$C_p = C_{\text{оз}} + C_{\text{дз}} + C_{\text{фсзн}} + C_{\text{бгс}} + C_{\text{пз}} + C_{\text{нр}}, \quad (6.7)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$C_{\text{дз}}$ – дополнительная заработная плата, руб.;

$C_{\text{фсзн}}$ – сумма отчислений в Фонд социальной защиты населения, руб.;

$C_{\text{бгс}}$ – сумма отчислений в БРУСП «Белгосстрах», руб.;

$C_{\text{пз}}$ – сумма прочих затрат, руб.;

$C_{\text{нр}}$ – сумма накладных расходов, руб.

$$\begin{aligned} C_p &= 10\,112 + 1\,011,20 + 3\,781,89 + 66,74 + 2\,022,40 + 5\,561,60 \\ &= 22\,555,83 \text{ руб.} \end{aligned}$$

Сумма расходов на разработку программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе, и составила 22 555,83 руб.

6.3.8 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию программного средства $C_{\text{рса}}$ определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию, по формуле 6.8.

$$C_{\text{рса}} = \frac{C_{\text{р}} \cdot H_{\text{рса}}}{100}, \quad (6.8)$$

где $C_{\text{р}}$ – общая сумма расходов на разработку программного средства, руб.;
 $H_{\text{рса}}$ – норматив расходов на реализацию, %.

$$C_{\text{рса}} = 22\,555,83 \cdot 10 / 100 = 2\,255,58 \text{ руб.}$$

Получим, что сумма расходов на сопровождение и адаптацию программного средства, составляет 2 255,58 руб.

6.3.9 Расчет полной себестоимости

Полная себестоимость $C_{\text{п}}$ определяется как сумма двух элементов: суммы расходов на разработку и суммы расходов на сопровождение и адаптацию.

Полная себестоимость вычисляется по формуле 6.9

$$C_{\text{п}} = C_{\text{р}} + C_{\text{рса}}, \quad (6.9)$$

где $C_{\text{р}}$ – сумма расходов, руб.;

$C_{\text{рса}}$ – сумма расходов на сопровождение и адаптацию, руб.

$$C_{\text{п}} = 22\,555,83 + 2\,255,58 = 24\,811,41 \text{ руб.}$$

Получим, что полная себестоимость веб-приложения равна 24 811,41 руб.

6.3.10 Определение цены, оценка эффективности

Отпускная цена веб-приложения зависит от цены разработчика, которая формируется на основе рентабельности продукции. Рентабельность и прибыль от создания веб-приложения определяются комплексным анализом рыночных условий, включая переговоры с потенциальным заказчиком (потребителем) и учет налога на добавленную стоимость.

Так как даже приблизительные цены аналогов не выкладываются в публичный доступ, а в основном остаются в тайне, то для расчета отпускной цены воспользуемся вариантом на основе желаемого уровня рентабельности.

При маркетинговом анализе рынка существующих аналогичных программных средств, предназначенных для организации и поддержки процессов раздельного сбора вторсырья, можно сделать вывод, что рынок содержит ряд решений с подобным функционалом. Поскольку данное программное средство разрабатывается под конкретные потребности системы раздельного сбора и учитывает основные недостатки приложений-аналогов, уровень его рентабельности можно обозначить на уровне 20%.

Прибыль от реализации $\Pi_{\text{пс}}$ вычисляется по формуле 6.10.

$$П_{\text{пс}} = \frac{С_{\text{п}} \cdot У_{\text{рент}}}{100}, \quad (6.10)$$

где $С_{\text{п}}$ – полная себестоимость программного средства, руб.;

$У_{\text{рент}}$ – уровень рентабельности, %.

Чистая прибыль от реализации $П_{\text{чист}}$ вычисляется по формуле 6.11.

$$П_{\text{чист}} = П_{\text{пс}} \cdot \left(1 - \frac{Н_{\text{п}}}{100}\right), \quad (6.11)$$

где $П_{\text{пс}}$ – прибыль от реализации, руб.;

$Н_{\text{п}}$ – налог на прибыль, %.

Цена разработки программного средства без налогов $Ц_{\text{р}}$ находится по формуле 6.12.

$$Ц_{\text{р}} = С_{\text{п}} + П_{\text{пс}} \quad (6.12)$$

Сумма налога на добавленную стоимость рассчитывается из соотношения 6.13.

$$\text{НДС} = \frac{Ц_{\text{р}} \cdot Н_{\text{ндс}}}{100}, \quad (6.13)$$

где $Ц_{\text{р}}$ – цена разработки программного средства, руб.;

$Н_{\text{ндс}}$ – ставка НДС, %.

Планируемая отпускная цена с НДС $Ц_{\text{сНДС}}$ вычисляется по формуле 6.14.

$$Ц_{\text{сНДС}} = Ц_{\text{р}} + \text{НДС} \quad (6.14)$$

Рассчитаем рентабельность проекта $R_{\text{а}}$, как отношение чистой прибыли к себестоимости по формуле 6.15.

$$R_{\text{а}} = П_{\text{ч}} / С_{\text{п}}. \quad (6.15)$$

Исходя из вышеописанных данных рассчитаем прибыль от реализации программного средства, чистую прибыль на основе прибыли от реализации, цену разработчика без налогов, сумму налогов на добавленную стоимость, планируемую отпускную цену с учетом НДС, а также рентабельность проекта.

$$П_{\text{пс}} = 24\,811,41 \cdot 20 / 100 = 4\,962,28 \text{ руб.}$$

$$П_{\text{чист}} = 4\,962,28 \cdot 0,8 = 3\,969,82 \text{ руб.}$$

$$Ц_p = 24\,811,41 + 4\,962,28 = 29\,773,69 \text{ руб.}$$

$$\text{НДС} = 29\,773,69 \cdot 20 / 100 = 5\,954,74 \text{ руб.}$$

$$Ц_{\text{сНДС}} = 29\,773,69 + 5\,954,74 = 35\,728,43 \text{ руб.}$$

$$R_a = 3\,969,82 / 24\,811,41 \cdot 100\% \approx 16,00\%$$

Как итог, планируемая отпускная цена продукта с учётом НДС составляет 35 728,43 руб. Стоимость веб-приложения для отдельного сбора отходов остаётся ниже, чем у большинства типовых решений на рынке. При этом система предлагает ключевые преимущества: удобный поиск пунктов сбора, гибкое управление базой данных отходов и встроенные инструменты информирования пользователей. Эти возможности делают приложение особенно полезным для экологических организаций, управляющих компаний и муниципальных структур, заинтересованных в популяризации отдельного сбора мусора. За счёт ориентации на конкретные потребности пользователей и оптимизации процессов разработки достигается высокая эффективность системы при снижении общих затрат на её внедрение и эксплуатацию.

6.4 Выводы по разделу

В рамках данного раздела были проведены экономические расчёты, на основе которых была определена себестоимость разрабатываемого программного средства, а также прогнозируемая отпускная цена всего продукта путём продажи его заказчику и передаче прав на владение продуктом. Анализ такого вида позволяет определить целесообразность разработки приложения, издержки при разработке приложения и определить итоговую прибыль от продажи программного средства.

В таблице 6.5 и в ДП 07.00 ГЧ представлены результаты расчётов для основных показателей данной главы в краткой форме.

Таблица 6.5 – Результаты расчётов

Наименование показателя	Значение
1	2
Время разработки, ч.	720
Основная заработная плата, руб.	10 112
Дополнительная заработная плата, руб.	1 011,20
Отчисления в Фонд социальной защиты населения, руб.	3 781,89
Отчисления в БРУСП «Белгосстрах», руб.	66,74
Прочие прямые затраты, руб.	2 022,40
Накладные расходы, руб.	5 561,60
Себестоимость разработки программного средства, руб.	22 555,83

Продолжение таблицы 6.5

1	2
Расходы на реализацию, руб	2 255,58
Полная себестоимость, руб.	24 811,41
Прибыль от реализации программного средства, руб.	4 962,28
Чистая прибыль, руб.	3 969,82
Цена разработки без налогов, руб.	29 773,69
НДС	5 954,74
Планируемая отпускная цена с НДС, руб.	35 728,43
Рентабельность разработки, %	16,00

Проведенные экономические расчеты подтвердили целесообразность разработки веб-приложения для организации и проведения проектных конкурсов. По итогам вычислений полная себестоимость проекта составила 24 811,41 руб. при сроке разработки 720 часов. Планируемая отпускная цена в размере 35 728,43 руб. с НДС обеспечивает рентабельность на уровне 20% и чистую прибыль 3 969,82 руб., что свидетельствует о финансовой устойчивости проекта.

Исходя из этих данных, можно сделать вывод о целесообразности разработки программного средства, так как проект принесет чистую прибыль в умеренном размере.

Необходимость разработки приложения для отдельного сбора вторсырья обусловлена растущим спросом на удобный инструмент, упрощающий процесс ответственного обращения с отходами. Оно объединяет данные о пунктах приёма, правилах сортировки и экологических инициативах в единую платформу, экономя время и снижая барьеры для вовлечения в раздельный сбор. Приложение централизует управление экологическими привычками, делая сортировку проще и повышая осознанность пользователей.

Социальный эффект выражается в упрощении поиска пунктов приёма, получении актуальной информации о сортировке и участии в экологических акциях. Это делает практику раздельного сбора более доступной и популяризирует экологическую ответственность.

Таким образом, приложение не только упрощает процесс раздельного сбора, но и вносит вклад в формирование более осознанного и экологичного общества, где переработка отходов становится частью повседневной жизни.

Заключение

В результате выполнения дипломного проекта на тему «Веб-приложение для раздельного сбора бытовых отходов» было разработано веб-приложение, предназначенное для популяризации процесса раздельного сбора вторсырья. Веб-приложение поддерживает три роли: гость, пользователь и администратор. Каждая роль имеет уникальный набор прав и возможностей, что позволяет обеспечить персонализированный подход к взаимодействию с программным средством.

Были проанализированы аналогичные программные средства, выбраны основные технологии и средства для реализации дипломного проекта. Клиентская часть приложения написана на языке *JavaScript* с использованием фреймворка *React.js*, а серверная часть реализована на платформе *Node.js* с использованием фреймворка *Express.js*.

Спроектирована база данных, состоящая из 12 таблиц, а также клиентская и серверная части приложения. Описаны принципы взаимодействия между компонентами приложения и архитектура частей приложения. Были описаны подходы к взаимодействию серверной части приложения с базой данных.

Были разработаны клиентская и серверная компоненты веб-приложения, реализован весь необходимый для работы функционал.

В ходе тестирования были проверены основные функции веб-приложения, было написано 29 тестов, покрытие кода тестами составило 100%.

Для веб-приложения разработано исчерпывающее руководство использования, описывающее основные сценарии работы с системой, что позволяет конечным пользователям легко ориентироваться в приложении и эффективно работать с ним.

Рассчитаны затраты на разработку представленного в дипломном проекте программного средства. На основании полученных данных можно сделать вывод, что разработанный проект является экономически выгодным.

Общий объем написанного программного кода веб-приложения составил примерно 11 000 авторских строк. Этот показатель наглядно демонстрирует масштабность и высокую степень комплексности реализованного проекта, отражая значительный объем выполненной разработки и проделанной работы по созданию функционального и надежного программного продукта.

Приложение является завершенным программным продуктом, реализующим все заявленные возможности в полной мере. Архитектура предусматривает возможности быстрого и безопасного расширения функционала.

					ДП 00.00.ПЗ		
		ФИО	Подпись	Дата	Заключение		
Разраб.		Трубач Д.С.					
Пров.		Мушук А.Н.					
Н. контр.		Мушук А.Н.					
Утв.		Смелов В.В.			БГТУ 1-40 01 01, 2025		
		Лит.	Лист	Листов			
		У	1	1			

Список использованной литературы

1. Зеленая карта : сайт по раздельному сбору вторсырья : [сайт]. – [s.l.], 2024. – URL: <http://greenmap.by/> (дата обращения: 05.10.2024).
2. Rsbtor : сайт по раздельному сбору вторсырья : [сайт]. – [s.l.], 2024. – URL: <https://rsbor.ru/> (дата обращения: 05.10.2024).
3. Target99 : сайт по раздельному сбору вторсырья : [сайт]. – [s.l.], 2024. – URL: <https://target99.by/> (дата обращения: 09.10.2024).
4. Cloundinary Official Documentation : [сайт]. – [s.l.], 2025. – URL: <https://cloudinary.com/> (дата обращения: 24.03.2025).
5. Yandex SMTP Documentation : [сайт]. – [s.l.], 2025. – URL: <https://yandex.ru/support/mail/mail-clients/smtp.html> (дата обращения: 24.03.2025).
6. Yandex Maps API Documentation : [сайт]. – [s.l.], 2025. – URL: <https://yandex.ru/dev/maps/> (дата обращения: 24.03.2025).
7. Google APIs Official Documentation : [сайт]. – [s.l.], 2025. – URL: <https://developers.google.com/apis> (дата обращения: 24.03.2025).
8. HTTP Protocol Specification : [сайт]. – [s.l.], 2024. – URL: <https://tools.ietf.org/html/rfc2616> (дата обращения: 24.03.2025).
9. HTTPS Protocol Specification : [сайт]. – [s.l.], 2024. – URL: <https://tools.ietf.org/html/rfc2818> (дата обращения: 24.03.2025).
10. SMTP Protocol Specification : [сайт]. – [s.l.], 2024. – URL: <https://tools.ietf.org/html/rfc5321> (дата обращения: 24.03.2025).
11. TCP Protocol Specification : [сайт]. – [s.l.], 2024. – URL: <https://tools.ietf.org/html/rfc793> (дата обращения: 24.03.2025).
12. Node.js Official Documentation : [сайт]. – [s.l.], 2025. – URL: <https://nodejs.org/> (дата обращения: 24.03.2025).
13. Express.js Official Documentation : [сайт]. – [s.l.], 2025. – URL: <https://expressjs.com/> (дата обращения: 24.03.2025).
14. React Official Documentation : [сайт]. – [s.l.], 2025. – URL: <https://reactjs.org/> (дата обращения: 24.03.2025).
15. MySQL Official Documentation : [сайт]. – [s.l.], 2025. – URL: <https://www.mysql.com/> (дата обращения: 24.03.2025).
16. Docker Official Documentation : [сайт]. – [s.l.], 2024. – URL: <https://docs.docker.com> (дата обращения: 04.12.2024).
17. Docker Compose Official Documentation : [сайт]. – [s.l.], 2025. – URL: <https://docs.docker.com/compose/> (дата обращения: 11.05.2025).

					ДП 00.00.ПЗ			
		ФИО	Подпись	Дата	Список использованной литературы			
Разраб.		Трубач Д.С.						
Пров.		Мушук А.Н.						
Н. контр.		Мушук А.Н.						
Утв.		Смелов В.В.			БГТУ 1-40 01 01, 2025			
					Лит. Лист Листов			
					У 1 2			

18. Redux Toolkit Query Documentation : [сайт]. – [s.l.], 2024. – URL: <https://redux-toolkit.js.org/rtk-query/overview> (дата обращения: 24.03.2025)
19. Swagger OpenAPI Documentation : [сайт]. – [s.l.], 2024. – URL: <https://swagger.io/specification/> (дата обращения: 24.03.2025).^у
20. Axios GitHub Repository : [сайт]. – [s.l.], 2024. – URL: <https://github.com/axios/axios> (дата обращения: 24.03.2025).
21. Framer Motion Documentation : [сайт]. – [s.l.], 2025. – URL: <https://www.framer.com/motion/> (дата обращения: 24.03.2025).
22. React Toastify Documentation : [сайт]. – [s.l.], 2025. – URL: <https://fkhadra.github.io/react-toastify/> (дата обращения: 24.03.2025).
23. TailwindCSS Documentation : [сайт]. – [s.l.], 2025. – URL: <https://tailwindcss.com/docs/> (дата обращения: 24.03.2025).
24. TensorFlow.js Official Documentation : [сайт]. – [s.l.], 2025. – URL: <https://www.tensorflow.org/js> (дата обращения: 24.03.2025).
25. npm Official Documentation : [сайт]. – [s.l.], 2025. – URL: <https://docs.npmjs.com/> (дата обращения: 24.03.2025).
26. MySQL Workbench Documentation : [сайт]. – [s.l.], 2025. – URL: <https://dev.mysql.com/doc/workbench/en/> (дата обращения: 24.03.2025).
27. Sequelize Documentation : [сайт]. – [s.l.], 2025. – URL: <https://sequelize.org/docs/v6/> (дата обращения: 24.03.2025).
28. NGINX Documentation : [сайт]. – [s.l.], 2025. – URL: <https://nginx.org/en/docs/> (дата обращения: 24.03.2025).

Диаграмма вариантов использования ДП 01.00 ГЧ

Логическая схема базы данных ДП 02.00 ГЧ

Диаграмма развертывания ДП 03.00 ГЧ

Диаграмма последовательности добавления вторсырья ДП 04.00 ГЧ

Блок схема алгоритма сдачи вторсырья ДП 05.00 ГЧ

Блок схема алгоритма просмотра скидок и открытия промокодов
ДП 06.00 ГЧ

Таблица экономических показателей ДП 07.00 ГЧ

ПРИЛОЖЕНИЕ А

Скрипт создания базы данных

```
show databases;
create database ecosort;
drop database ecosort;
```

```
UPDATE ecosort.users
SET points = 10000
WHERE id = 2;
```

```
select * from ecosort.users;           -- пользователи
select * from ecosort.articles;        -- статьи
select * from ecosort.ratings;         -- комментарии
select * from ecosort.likes;           -- лайки
select * from ecosort.s_keys;          -- ключи от пунктов приема
select * from ecosort.points;          -- пункты приема
select * from ecosort.receptions;      -- переписано
select * from ecosort.points_marks;    -- переписано
select * from ecosort.marks;           -- виды вторсырья
select * from ecosort.check_weight;    -- подтверждение сдачи
select * from ecosort.points_marks;    -- используется для связи
«многие-ко-многим» таблиц points и marks
select * from ecosort.discounts;       -- скидки
select * from ecosort.promo_codes;     -- промокоды
```

-- Пользователь

```
CREATE TABLE IF NOT EXISTS ecosort.users (
    id int auto_increment,
    username      varchar(20)          not null,
    email         varchar(100)         not null,
    password_hash varchar(200)         not null,
    points        int                  default 0    null,
    role          varchar(5) default 'user' not null,
    is_activated  tinyint(1) default 0    null,
    activation_link varchar(150)        null,
    constraint users_ck CHECK (role IN ('admin', 'user')),
    constraint email_un unique (email),
    constraint password_hash_un unique (password_hash),
    constraint users_pk primary key (id));
```

-- Статьи --

```
CREATE TABLE IF NOT EXISTS ecosort.articles (
    id int auto_increment,
    title      varchar(100) not null,
    text       text        not null,
    date_of_pub date        not null,
    image_url  varchar(150) not null,
    author     int          not null,
    likes      int default 0 null,
    constraint articles_pk primary key (id),
    constraint title_un unique (title),
```



```

        constraint articles_fk_users foreign key (author)
references ecosort.users(id) on delete cascade);
-- КОММЕНТЫ --
CREATE TABLE IF NOT EXISTS ecosort.ratings (
    id            int auto_increment,
    article_id    int            not null,
    commentator   int            not null,
    comment       varchar(500) not null,
    date_of_add   DATETIME DEFAULT CURRENT_TIMESTAMP,
    constraint ratings_pk primary key (id),
    constraint ratings_fk_users foreign key (commentator)
references ecosort.users(id) on delete cascade,
    constraint ratings_fk_articles foreign key (article_id)
references ecosort.articles(id) on delete cascade);
-- Лайки --
CREATE TABLE IF NOT EXISTS ecosort.likes(
    id            INT AUTO_INCREMENT,
    user_id       INT NOT NULL,
    article_id    INT NOT NULL,
    date_of_add   DATETIME DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT likes_pk PRIMARY KEY (id),
    CONSTRAINT likes_fk_users FOREIGN KEY (user_id) REFERENCES
ecosort.users(id) ON DELETE CASCADE,
    CONSTRAINT likes_fk_articles FOREIGN KEY (article_id)
REFERENCES ecosort.articles(id) ON DELETE CASCADE
);
-- Секретный ключ для ПП--
create table IF NOT EXISTS ecosort.s_keys(
    id int auto_increment,
    secret_key varchar(100) not null,
    is_used    int default 0 not null,
    constraint s_keys_ck CHECK (is_used IN (1,0)),
    constraint s_keys_pk primary key (id));
-- Пункт сдачи --
CREATE TABLE IF NOT EXISTS ecosort.points(
    id            int auto_increment,
    address       varchar(100) not null,
    time_of_work  varchar(100) not null,
    key_id        int            not null,
    admin_id      int            not null,
    link_to_map   text           not null,
    point_name    varchar(100) not null,
    constraint address_un unique (address),
    constraint key_id_un unique (key_id),
    constraint point_name_un unique (point_name),
    constraint points_pk primary key (id),
    constraint points_fk_users foreign key (admin_id)
references ecosort.users(id) on delete cascade,
    constraint points_fk_s_keys foreign key (key_id) references
ecosort.s_keys(id) on delete cascade);
-- Отходы --
CREATE TABLE IF NOT EXISTS ecosort.marks(

```

```

        id int auto_increment,
        rubbish varchar(50) not null,
        points_per_kg int not null,
        new_from_kg float not null,
        image_link varchar(255) null,
        constraint marks_pk primary key (id));
-- Точки сбора с отходами --
CREATE TABLE IF NOT EXISTS ecosort.points_marks(
    id int auto_increment,
    points_id int not null,
    marks_id int not null,
    constraint points_marks_pk primary key (id),
    constraint points_marks_fk_points foreign key (points_id)
references ecosort.points(id) on delete cascade,
    constraint points_marks_fk_marks foreign key (marks_id)
references ecosort.marks(id) on delete cascade);
-- Проверка веса --
CREATE TABLE IF NOT EXISTS ecosort.check_weight(
    id int auto_increment,
    rubbish_id int not null,
    weight int not null,
    key_of_weight varchar(100) not null,
    is_used INT DEFAULT 0 NOT NULL,
    original_key VARCHAR(100) NOT NULL,
    CONSTRAINT check_weight_is_used_ck CHECK (is_used IN (0,
1)),
    constraint key_of_weight_un unique (key_of_weight),
    constraint check_weight_pk primary key (id),
    constraint check_weight_fk_marks foreign key (rubbish_id)
references ecosort.marks(id) on delete cascade);
-- Сдача --
CREATE TABLE IF NOT EXISTS ecosort.receptions(
    id int auto_increment,
    user_id int not null,
    weight float not null,
    accrued int null,
    new_kg int null,
    type_waste int not null,
    station_key int not null,
    weight_key int not null,
    constraint receptions_pk primary key (id),
    constraint receptions_fk_users foreign key (user_id)
references ecosort.users(id) on delete cascade,
    constraint receptions_fk_s_keys foreign key (station_key)
references ecosort.s_keys(id) on delete cascade,
    constraint receptions_fk_marks foreign key (type_waste)
references ecosort.marks(id) on delete cascade,
    constraint receptions_fk_check_weight foreign key
(weight_key) references ecosort.check_weight(id) on delete
cascade);
-- Скидки --
CREATE TABLE IF NOT EXISTS ecosort.discounts(

```

```

        id int auto_increment,
        discount varchar(50) not null,
        promo_code varchar(50) not null,
        count_for_dnt int not null,
        constraint discount_un unique (discount),
        constraint promo_code_un unique (promo_code),
        constraint discounts_pk primary key (id));
-- Промокоды --
CREATE TABLE IF NOT EXISTS ecosort.promo_codes(
    id int auto_increment,
    promo_code varchar(50) not null,
    user_id int not null,
    discount_id int not null,
    date_of_add DATE not null DEFAULT '2025-03-27',
    constraint discounts_pk primary key (id),
    constraint user_discount_ids_un unique (user_id,
discount_id),
    constraint promo_codes_fk_users foreign key (user_id)
references ecosort.users (id) on delete cascade,
    constraint promo_codes_fk_discounts foreign key
(discount_id) references ecosort.discounts (id) on delete
cascade
);

ALTER TABLE ecosort.check_weight
ADD is_used INT DEFAULT 0 NOT NULL,
ADD CONSTRAINT check_weight_is_used_ck CHECK (is_used IN (0,
1));

```

ПРИЛОЖЕНИЕ Б

Модели данных

```
const Sequelize = require('sequelize')
const Model = Sequelize.Model;

class Articles extends Model { }
const { Users } = require('../config/Users')
Articles.init(
  {
    id: { type: Sequelize.INTEGER, primaryKey: true,
unique: true, autoIncrementIdentity: true, required: true },
    title: { type: Sequelize.STRING, allowNull: false,
unique: true, required: true },
    text: { type: Sequelize.TEXT, allowNull: false,
required: true },
    date_of_pub: { type: Sequelize.DATE, allowNull:
false, required: true },
    image_url: { type: Sequelize.STRING },
    author: { type: Sequelize.INTEGER, allowNull:
false, required: true },
    likes: { type: Sequelize.INTEGER, required: true },
  },
  { sequelize, modelName: 'Articles', tableName: 'articles',
timestamps: false }
);
Users.hasMany(Articles, { foreignKey: 'author' });
Articles.belongsTo(Users, { foreignKey: 'author' });
module.exports = { Articles };

class Check_weights extends Model{}
Check_weights.init(
  {
    id: { type: Sequelize.INTEGER, primaryKey: true,
unique: true, autoIncrement: true, required: true },
    rubbish_id: { type: Sequelize.INTEGER, allowNull:
false, required: true },
    weight: { type: Sequelize.INTEGER, allowNull: false,
required: true },
    key_of_weight: { type: Sequelize.STRING, allowNull:
false, required: true },
    original_key: { type: Sequelize.STRING, allowNull:
false }, // Новое поле для оригинального ключа
    is_used: {
      type: Sequelize.INTEGER,
      defaultValue: 0,
      allowNull: false,
    },
  },
  { sequelize, modelName: 'Check_weights', tableName:
'check_weight', timestamps: false }
```

```

);

Marks.hasMany(Check_weights, {foreignKey:'rubbish_id'});
Check_weights.belongsTo(Marks, {foreignKey: 'rubbish_id'});

module.exports = {Check_weights};

class Discounts extends Model{}

Discounts.init (
  {
    id: {type: Sequelize.INTEGER, primaryKey:true, unique:
true, autoIncrementIdentity: true, required: true},
    discount: {type: Sequelize.STRING, allowNull: false,
unique: true, required: true},
    count_for_dnt: {type: Sequelize.INTEGER, allowNull:
false, required: true},
    promo_code: {type: Sequelize.STRING, allowNull: false,
unique: true, required: true},
  },
  {sequelize, modelName: 'Discounts', tableName:
'discounts', timestamps: false }
);

module.exports = {Discounts};

class Keys extends Model{}

Keys.init({
  id: {type: Sequelize.INTEGER, primaryKey:true, unique:
true, autoIncrementIdentity: true, required: true},
  secret_key: {type: Sequelize.STRING, allowNull: false,
unique: true, required: true},
  is_used: {type: Sequelize.INTEGER, allowNull: false,
defaultValue:0 } ,
},
  {sequelize, modelName: 'Keys', tableName: 's_keys',
timestamps: false}
);

module.exports = {Keys};

class Likes extends Model{}

const {Users} = require('../config/Users')
const {Articles} = require('../config/Articles')

Likes.init (
  {
    id: {type: Sequelize.INTEGER, primaryKey:true, unique:
true, autoIncrementIdentity: true, required: true},

```

```

        article_id: {type: Sequelize.INTEGER, allowNull: false,
required: true},
        user_id: {type: Sequelize.INTEGER, allowNull: false,
required: true},
        date_of_add: {
            type: Sequelize.DATE,
            defaultValue: Sequelize.NOW
        },
    },
    { sequelize, modelName: 'Likes', tableName: 'likes',
timestamps: false }
);

```

```

Users.hasMany(Likes, {foreignKey: 'user_id'});
Likes.belongsTo(Users, {foreignKey: 'user_id'});
Articles.hasMany(Likes, {foreignKey: 'article_id'});
Likes.belongsTo(Articles, {foreignKey: 'article_id'});

```

```
module.exports = {Likes};
```

```
class Marks extends Model{}
```

```

Marks.init (
    {
        id: {type: Sequelize.INTEGER, primaryKey:true, unique:
true, autoIncrementIdentity: true, required: true},
        rubbish: {type: Sequelize.STRING, allowNull: false,
required: true},
        points_per_kg: {type: Sequelize.INTEGER, allowNull:
false, required: true},
        new_from_kg: {type: Sequelize.FLOAT, allowNull: false,
required: true},
        image_link: { type: Sequelize.STRING },
    },
    { sequelize, modelName: 'Marks', tableName: 'marks',
timestamps: false }
);

```

```
module.exports = {Marks};
```

```

class Points_marks extends Model{}
const {Marks} = require('../config/Marks')
const {Points} = require('../config/Points')

```

```

Points_marks.init(
    {
        id: {type: Sequelize.INTEGER, primaryKey:true, unique:
true, autoIncrementIdentity: true, required: true},
        points_id: {type: Sequelize.INTEGER, allowNull: false,
required: true},
        marks_id: {type: Sequelize.INTEGER, allowNull: false,
required: true},
    }
);

```

```

    },
    {sequelize, modelName: 'Points_marks', tableName:
'points_marks', timestamps: false}
);

Points.hasMany(Points_marks, {foreignKey:'points_id'});
Points_marks.belongsTo(Points, {foreignKey: 'points_id'});

Marks.hasMany(Points_marks, {foreignKey:'marks_id'});
Points_marks.belongsTo(Marks, {foreignKey: 'marks_id'});

module.exports = {Points_marks};

class Points extends Model{}
const {Users} = require('../config/Users')
const {Keys} = require('../Keys')

Points.init (
  {
    id: {type: Sequelize.INTEGER, primaryKey:true,
unique: true, autoIncrementIdentity: true, required: true},
    address: {type: Sequelize.STRING, allowNull: false,
unique: true, required: true},
    point_name: {type: Sequelize.STRING, allowNull:
false, unique: true, required: true},
    time_of_work: {type: Sequelize.STRING, allowNull:
false, required: true},
    key_id: {type: Sequelize.INTEGER, allowNull: false,
unique: true, required: true},
    admin_id: {type: Sequelize.INTEGER, allowNull:
true, required: true },
    link_to_map: {type: Sequelize.TEXT, required:true},
  },
  {sequelize, modelName: 'Points', tableName: 'points',
timestamps: false}
);

Users.hasMany(Points, {foreignKey:'admin_id'});
Points.belongsTo(Users, {foreignKey: 'admin_id'});
Keys.hasMany(Points, {foreignKey:'key_id'});
Points.belongsTo(Keys, {foreignKey: 'key_id'});

module.exports = {Points};

class Promo_codes extends Model{}

const {Users} = require('../config/Users')
const {Discounts} = require('../config/Discounts')

Promo_codes.init(
  {

```

```

      id: {type: Sequelize.INTEGER, primaryKey:true, unique:
true, autoIncrementIdentity: true, required: true},
      promo_code: {type: Sequelize.STRING, allowNull: false,
required: true},
      user_id: {type: Sequelize.INTEGER, allowNull:
false, required: true},
      discount_id: {type: Sequelize.INTEGER, allowNull:
false, required: true},
      date_of_add: {type: Sequelize.DATE, allowNull: false,
required: true}
    },
    { sequelize, modelName: 'Promo_codes', tableName:
'promo_codes', timestamps: false }
  );

Users.hasMany(Promo_codes, { foreignKey: 'user_id' });
Promo_codes.belongsTo(Users, { foreignKey: 'user_id' });
Discounts.hasMany(Promo_codes, { foreignKey: 'discount_id' });
Promo_codes.belongsTo(Discounts, { foreignKey: 'discount_id'
});

module.exports = {Promo_codes};

class Ratings extends Model{}

const {Users} = require('../config/Users')
const {Articles} = require('../config/Articles')

Ratings.init (
  {
    id: {type: Sequelize.INTEGER, primaryKey:true, unique:
true, autoIncrementIdentity: true, required: true},
    article_id: {type: Sequelize.INTEGER, allowNull: false,
required: true},
    commentator: {type: Sequelize.INTEGER, allowNull:
false, required: true},
    comment: {type: Sequelize.STRING, required: true},
    date_of_add: {type: Sequelize.DATE, allowNull: false,
required: true}
  },
  { sequelize, modelName: 'Ratings', tableName: 'ratings',
timestamps: false }
);

Users.hasMany(Ratings, {foreignKey: 'commentator'});
Ratings.belongsTo(Users, {foreignKey: 'commentator'});
Articles.hasMany(Ratings, {foreignKey: 'article_id', onDelete:
'cascade'});
Ratings.belongsTo(Articles, {foreignKey: 'article_id'});

module.exports = {Ratings};

```



```

class Receptions extends Model{}

const {Users} = require('./Users')
const {Marks} =require('./Marks')
const {Keys} = require('./Keys')
const {Check_weights} = require('./Check_weights')

Receptions.init (
  {
    id: {type: Sequelize.INTEGER, primaryKey:true,
unique: true, autoIncrementIdentity: true, required: true},
    user_id: {type: Sequelize.INTEGER, allowNull:
false, required: true},
    accrued: {type: Sequelize.INTEGER, required: true},
    new_kg: {type: Sequelize.INTEGER, required: true},
    weight: {type: Sequelize.FLOAT, allowNull: false,
required: true},
    type_waste: {type: Sequelize.INTEGER, allowNull:
false, required: true},
    station_key: {type: Sequelize.INTEGER, allowNull:
false, required: true},
    weight_key: {type: Sequelize.INTEGER, allowNull:
false, required: true},
  },
  { sequelize, modelName: 'Receptions', tableName:
'receptions', timestamps: false }
);

Users.hasMany(Receptions, {foreignKey: 'user_id'});
Receptions.belongsTo(Users, {foreignKey: 'user_id'});

Marks.hasMany(Receptions, {foreignKey: 'type_waste'});
Receptions.belongsTo(Marks, {foreignKey: 'type_waste'});

Keys.hasMany(Receptions, {foreignKey: 'station_key'});
Receptions.belongsTo(Keys, {foreignKey: 'station_key'});

Check_weights.hasMany(Receptions, {foreignKey: 'weight_key'});
Receptions.belongsTo(Check_weights, {foreignKey:
'weight_key'});

module.exports = {Receptions};

class Users extends Model{}

Users.init (
  {
    id: {type: Sequelize.INTEGER, primaryKey:true, unique:
true, autoIncrementIdentity: true, required: true},
    username:{type: Sequelize.STRING,  allowNull: false,
unique: true, required: true},

```

```

        email: {type: Sequelize.STRING, allowNull: false,
unique: true, required: true},
        password_hash: {type: Sequelize.STRING, allowNull:
false, required: true},
        points: {type: Sequelize.INTEGER},
        role: {type: Sequelize.STRING, validate:
{isIn:[['user', 'admin']] } },
        is_activated:{type: Sequelize.BOOLEAN, default: false},
        activation_link: {type: Sequelize.STRING}
    },
    { sequelize, modelName:'Users', tableName:'users',
timestamps: false}
);

module.exports = {Users};

```

ПРИЛОЖЕНИЕ В

Функция *RegisterUser*

```
RegisterUser: async (req, res) => {
  try {
    const i_password = req.body.password;
    const salt = '$2b$10$qNuSSupDD53DkQfO8wqpF.';
    const o_password = await bcrypt.hash(i_password,
salt);

    const v_check_user = await
db.models.Users.findOne({
      where: { username: req.body.username }
    })

    const v_check_email = await
db.models.Users.findOne({
      where: { email: req.body.email }
    })

    if (v_check_user == null) {
      if (v_check_email == null) {

        const v_activation_link = uuid.v4();

        const candidate = await
db.models.Users.create({
          username: req.body.username,
          email: req.body.email,
          password_hash: o_password,
          role: 'user',
          is_activated: false,
          activation_link: v_activation_link
        })

        const send_mail = req.body.email;
        const send_link =
'http://localhost:8082/activate/' + v_activation_link;

        const transporter =
nodemailer.createTransport({
          host: "smtp.yandex.ru",
          port: 465,
          secure: true,
          auth: {
            user: 'dimatruba2004@yandex.ru',
            pass: 'akllfknlkhqfbhtl'
          }
        });

        const mailOptions = {
```

```

        from: 'dimatruba2004@yandex.ru',
        to: send_mail,
        subject: 'Подтверждение регистрации на
сайте WasteWise',
        html: `
            <div style="font-family: Arial,
sans-serif; line-height: 1.6; color: #333;">
                <h2 style="color:
#4CAF50;">Здравствуйтесь, ${req.body.username}</h2>
                <p>
                    Благодарим за регистрацию
на платформе <strong>WasteWise</strong> – месте, где мы вместе
заботимся о нашем будущем через переработку отходов.
                </p>
                <p>
                    Для завершения регистрации
и активации вашего аккаунта, пожалуйста, перейдите по ссылке
ниже:
                    </p>
                    <a
                        href="${send_link}"
                        style="display: inline-
block; margin: 10px 0; padding: 10px 20px; color: #fff;
background-color: #4CAF50; text-decoration: none; border-
radius: 5px;"
                    >
                        Активировать аккаунт
                    </a>
                <p>
                    Или вы можете скопировать
ссылку в браузер:
                </p>
                <p style="font-size: 14px;
color: #555;">${send_link}</p>
                <hr style="border: none;
border-top: 1px solid #ddd;">
                <p style="font-size: 12px;
color: #777;">
                    Если вы не регистрировались
на сайте WasteWise, проигнорируйте это сообщение.
                </p>
                <p>
                    С уважением,<br>Команда
WasteWise
                </p>
            </div>
        `
    }

    let info = await
transporter.sendMail(mailOptions)

```

```

        });
    }
    else {
        res.json({
            message: 'Почта занята, введите другую'
        });
    }
}
else {
    res.json({
        message: 'Имя пользователя занято, введите
другое'
    });
}
}
catch (err) {
    console.log(err);
    res.json({
        message: 'Не удалось зарегистрироваться'
    });
}
},

```

Функция *Receptions*

```

Receptions: async (req, res) => {
    try {
        // СЧИТЫВАЕМ КЛЮЧ СТАНЦИИ
        const i_station_key = req.body.station_key;
        const salt = '$2b$10$qNuSSupDD53DkQf08wqpF.';
        const o_station_key = await
bcrypt.hash(i_station_key, salt);

        // Находим запись по ключу станции
        const v_find_key = await db.models.Keys.findOne({
            where: { secret_key: o_station_key }
        });

        if (!v_find_key || v_find_key.is_used !== 1) {
            return res.status(400).json({
                message: 'Ключ станции недействителен'
            });
        }

        // Находим пункт приема, связан ли с ключом станции
        const v_find_key_point = await
db.models.Points.findOne({
            where: { key_id: v_find_key.id }
        });
    }
}

```

```

        if (!v_find_key_point) {
            return res.status(400).json({
                message: 'Станция с таким ключом не
найдена'
            });
        }

        // Находим виды отходов, которые принимает этот
пункт сдачи
        const pointWasteTypes = await
db.models.Points_marks.findAll({
            where: { points_id: v_find_key_point.id },
            attributes: ['marks_id'],
        });

        const pointWasteIds = pointWasteTypes.map(item =>
item.marks_id);

        // Считываем ключ для проверки веса
        const i_key_of_weight = req.body.key_of_weight;
        const o_key_of_weight = await
bcrypt.hash(i_key_of_weight, salt);

        // Находим запись по ключу веса
        const v_check_key_w = await
db.models.Check_weights.findOne({
            attributes: ["id", "rubbish_id", "weight",
"is_used"],
            where: { key_of_weight: o_key_of_weight }
        });

        if (!v_check_key_w) {
            return res.status(400).json({
                message: 'Ключ веса недействителен'
            });
        }

        // Проверка, был ли уже использован ключ веса
        if (v_check_key_w.is_used === 1) {
            return res.status(400).json({
                message: 'Ключ веса уже использован'
            });
        }

        // Проверяем, соответствует ли вид отхода пункту
сдачи
        if
(!pointWasteIds.includes(v_check_key_w.rubbish_id)) {
            return res.status(400).json({
                message: 'Этот вид отходов не принимается
на данный пункт сдачи'
            });
        }

```

```

    }

    // Получаем данные о вторсырье
    const v_rubbish = await db.models.Marks.findOne({
      attributes: ["id", "rubbish", "points_per_kg",
"new_from_kg"],
      where: { id: v_check_key_w.rubbish_id }
    });

    if (!v_rubbish) {
      return res.status(400).json({
        message: 'Вид вторсырья не найден'
      });
    }

    // Рассчитываем начисление баллов
    const v_weight = v_check_key_w.weight;
    const o_new_points = v_weight *
v_rubbish.points_per_kg;
    const o_new_kg = v_weight * v_rubbish.new_from_kg;

    // Находим текущие баллы пользователя
    const i_user_points = await
db.models.Users.findOne({
      attributes: ["points"],
      where: { id: req.userId }
    });

    if (!i_user_points) {
      return res.status(400).json({
        message: 'Пользователь не найден'
      });
    }

    // Обновляем баллы пользователя
    const o_new_points_user = o_new_points +
i_user_points.points;
    await db.models.Users.update({
      points: o_new_points_user
    }, {
      where: { id: req.userId }
    });

    // Отметим ключ веса как использованный
    await db.models.Check_weights.update({
      is_used: 1
    }, {
      where: { id: v_check_key_w.id }
    });

    // Создаем запись в receptions
    await db.models.Receptions.create({

```

```

        user_id: req.userId,
        accrued: o_new_points,
        new_kg: o_new_kg,
        weight: v_weight,
        type_waste: v_rubbish.id,
        station_key: v_find_key.id,
        weight_key: v_check_key_w.id
    });

    // Отправляем успешный ответ
    res.json({
        o_new_kg,
        o_new_points,
        o_new_points_user,
        message: 'Ваши баллы успешно начислены',
    });
} catch (error) {
    console.log(error);
    res.status(500).json({
        message: 'Не удалось начислить баллы.
Попробуйте ещё раз.'
    });
}
}

```

Функция *usedMyDiscounts*

```

usedMyDiscounts: async (req, res) => {
    try {
        const v_user = await db.models.Users.findOne({
            attributes: ["points"],
            where: {id: req.userId}
        })

        // console.log(v_user.points);

        const i_discounts = await
db.models.Discounts.findOne({
            attributes: ["id", "discount", "count_for_dnt",
"promo_code"],
            where: {id: req.params.id}
        })

        // console.log(i_discounts.count_for_dnt);

        const o_new_points = v_user.points -
i_discounts.count_for_dnt

        // console.log(o_new_points);

        await db.models.Users.update({
            points: o_new_points,

```



```

    }, {where: {id: req.userId}})

    const v_check_promo_codes =
await    db.models.Promo_codes.findOne({
        where: {
            [Op.and]: [{ user_id: req.userId }, {
discount_id: i_discounts.id }],
        }
    })

    console.log(v_check_promo_codes)
    if (v_check_promo_codes!=null) {
        await db.models.Promo_codes.update({
            promo_code: i_discounts.promo_code,
            date_of_add: Date.now(),
            {where: {
                [Op.and]: [{ user_id: req.userId },
{ discount_id: i_discounts.id }],
            }
        })
    }
    else{
        await db.models.Promo_codes.create({
            promo_code: i_discounts.promo_code,
            user_id: req.userId,
            discount_id: i_discounts.id,
            date_of_add: Date.now(),
        })
    }

    function generateString(length) {
        let result = '';
        const characters =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
';

        for (let i = 0; i < length; i++) {
            const randomIndex =
Math.floor(Math.random() * characters.length);
            result += characters.charAt(randomIndex);
        }
        return result;
    }

    const generatedString = generateString(8);
    // console.log(generatedString);

    await db.models.Discounts.update({
        promo_code: generatedString,
        {where: {id: req.params.id}
    })

```

```
        res.json({
            o_new_points,
            message: 'Скидка использована'
        });
    }
    catch (error) {
        console.log(error);
    }
},
```

ПРИЛОЖЕНИЕ Г

Помещение данных в *state*

```
import { createSlice, createAsyncThunk } from
 '@reduxjs/toolkit'
import axios from '../../utils/axios'

const initialState = {
  alldiscounts: [],
  loading: false,
  status: null,
  valid: null,
}

export const getAllDiscounts = createAsyncThunk(
  'discount/getAllDiscounts',
  async () => {
    try {
      const { data } = await axios.get('/Discounts')
      return data
    } catch (error) {
      console.log(error)
      throw error
    }
  }
)

export const removeAllDiscount = createAsyncThunk(
  'discount/removeAllDiscount',
  async (id) => {
    try {
      const { data } = await
axios.delete(`/Discounts/${id}`)
      return data
    } catch (error) {
      console.log(error)
      throw error
    }
  }
)

export const addDiscount = createAsyncThunk(
  'discount/addDiscount',
  async ({ discount, count_for_dnt, promo_code }) => {
    try {
      const { data } = await axios.post('/Discounts', {
discount, count_for_dnt, promo_code })
      return data
    } catch (error) {
      console.log(error)
      throw error
    }
  }
)
```

```

    }
  }
)

export const updateDiscount = createAsyncThunk(
  'discount/updateDiscount',
  async (updatedDiscount) => {
    try {
      const { data } = await
axios.put(`/discounts/${updatedDiscount.id}`, updatedDiscount)
      return data
    } catch (error) {
      console.log(error)
      throw error
    }
  }
)

export const myDiscount = createAsyncThunk(
  'discount/myDiscount',
  async () => {
    try {
      const { data } = await axios.get('/used/discounts')
      return data
    } catch (error) {
      console.log(error)
      throw error
    }
  }
)

export const myUsedDiscount = createAsyncThunk(
  'discount/UseDiscount',
  async (id) => {
    try {
      const { data } = await
axios.put(`/used/discounts/${id}`, id)
      return data
    } catch (error) {
      console.log(error)
      throw error
    }
  }
)

export const alldiscountSlice = createSlice({
  name: 'alldiscount',
  initialState,
  reducers: {
    clearStatus: (state) => {
      state.status = null;
    },
  },
})

```

```

    },
    extraReducers: (builder) => {
      builder
        // Получить все скидки
        .addCase(getAllDiscounts.pending, (state) => {
          state.loading = true
          state.status = null
        })
        .addCase(getAllDiscounts.fulfilled, (state, action)
=> {
          state.loading = false
          state.alldiscounts =
action.payload.alldiscounts
        })
        .addCase(getAllDiscounts.rejected, (state, action)
=> {
          state.loading = false
          state.status = action.error.message || 'Ошибка
при получении скидок'
        })

        // Удаление скидки
        .addCase(removeAllDiscount.pending, (state) => {
          state.loading = true
          state.status = null
        })
        .addCase(removeAllDiscount.fulfilled, (state,
action) => {
          state.loading = false
          state.alldiscounts = state.alldiscounts.filter(
            (discount) => discount.id !==
action.payload.id
          )
          state.status = action.payload.message ||
'Sкидка удалена'
        })
        .addCase(removeAllDiscount.rejected, (state,
action) => {
          state.loading = false
          state.status = action.error.message || 'Ошибка
при удалении скидки'
        })

        // Добавление скидки
        .addCase(addDiscount.pending, (state) => {
          state.loading = true
          state.status = null
        })
        .addCase(addDiscount.fulfilled, (state, action) =>
{
          state.loading = false
          state.alldiscounts.push(action.payload)

```

```

        //state.status = action.payload.message ||
'Sкидка добавлена'
        state.valid = action.payload.msg ||
'Dействительная скидка'
    })
    .addCase(addDiscount.rejected, (state, action) => {
        state.loading = false
        state.status = action.error.message || 'Ошибка
при добавлении скидки'
        state.valid = action.payload?.msg ||
'Недействительная скидка'
    })

    // Обновление скидки
    .addCase(updateDiscount.pending, (state) => {
        state.loading = true
        state.status = null
    })
    .addCase(updateDiscount.fulfilled, (state, action)
=> {
        state.loading = false
        const index = state.alldiscounts.findIndex(
            (discount) => discount.id ===
action.payload.id
        )
        state.alldiscounts[index] = action.payload
        state.status = action.payload.message ||
'Sкидка обновлена'
    })
    .addCase(updateDiscount.rejected, (state, action)
=> {
        state.loading = false
        state.status = action.error.message || 'Ошибка
при обновлении скидки'
    })

    // Получить мои скидки
    .addCase(myDiscount.pending, (state) => {
        state.loading = true
        state.status = null
    })
    .addCase(myDiscount.fulfilled, (state, action) => {
        state.loading = false
        state.alldiscounts =
action.payload.alldiscounts
    })
    .addCase(myDiscount.rejected, (state, action) => {
        state.loading = false
        state.status = action.error.message || 'Ошибка
при получении моих скидок'
    })

```

```

        // Использовать скидку
        .addCase(myUsedDiscount.pending, (state) => {
            state.loading = true
            state.status = null
        })
        .addCase(myUsedDiscount.fulfilled, (state, action)
=> {
            state.loading = false
            state.alldiscounts = action.payload
            state.status = action.payload.message ||
'Скидка использована'
        })
        .addCase(myUsedDiscount.rejected, (state, action)
=> {
            state.loading = false
            state.status = action.error.message || 'Ошибка
при использовании скидки'
        })
    })
})

export const { clearStatus } = alldiscountSlice.actions;

export default alldiscountSlice.reducer

```