Частное учреждение

профессиональная образовательная организация

ТЕХНИКУМ «БИЗНЕС И ПРАВО»

ЗАЩИЩЕНО С ОЦЕНКОЙ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и ФИО руководителя)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

**ОТЧЕТ О ВЫПОЛНЕНИИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

|  |  |
| --- | --- |
| по ПМ.09 | Проектирование и разработка веб-приложений |
| Выполнил | Сурин А. Ю. |
| Курс, группа | 4 курс, группа ИС-4 |
| Специальность | 09.02.07 Информационные системы и программирование |
| Руководитель практики | Мартыненко В.А. |

г. Белореченск

2023 г.

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| Введение | 3 |
| 1. ПРОЕКТИРОВАНИЕ ВЕБ ПРИЛОЖЕНИЯ | 5 |
| 1.1 Цели и задачи выполняемой системы | 5 |
| 1.2 Анализ конкурентов | 6 |
| 1.3 Техническое задание | 9 |
| 1.4 Дизайн веб-проекта | 19 |
| 2. РАЗРАБОТКА И ДОКУМЕНТИРОВАНИЕ ВЕБ ПРИЛОЖЕНИЯ | 29 |
| 2.1 Оптимизация веб-приложения | 29 |
| 2.2 Тестирование веб-приложения | 30 |
| 2.3 Руководства пользователя | 34 |
| 2.4 Руководство администратора | 37 |
| ЗАКЛЮЧЕНИЕ | 41 |
| СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ | 43 |
| ПРИЛОЖЕНИЕ |  |

ВВЕДЕНИЕ

Актуальность исследования – в современном цифровом мире социальные сети стали неотъемлемой частью повседневной жизни миллиардов людей по всему миру. Важность социальных медиа для бизнеса (как малого, так и большого), и людей в целом становится все более очевидной, поскольку они предоставляют возможность эффективного взаимодействия с аудиторией, продвижения бренда и увеличения онлайн-присутствия компании или бренда.

Однако, использование обычных методов СММ (социального медиа маркетинга), основанных на активности сотрудников компании или внешних специалистов, не всегда могут обеспечить достаточный уровень эффективности в продвижении в сети Интернета. Все больше бизнесов, особенно малых компаний и стартапов, сталкиваются с проблемой ограниченного бюджета и неспособности привлечь большое количество подписчиков и клиентов через традиционные методы.

В такой ситуации на помощь может прийти услуги накрутки в социальных сетях. Накрутка представляет собой стратегию, при которой компании, бизнесы и даже обычные пользователи платят за увеличение числа своих разных активностей на платформах социальных сетей. Этот метод может быть особенно полезен для малых бизнесов, которым необходимо быстро увеличить свою онлайн-присутствие и привлечь новых клиентов.

Важно понимать - компаниям, любому бизнесу или одному человеку должны учитывать то, что социальные сети могут иметь возможность дать негативный ответ за подобные продвижения, ведь накрутка в социальных сетях не является гуманным и поддерживаемым способом продвинуться в сети.

Например, накрутка подписчиков в социальных сетях может быть эффективным инструментом для создания впечатления о популярности бренда или личности, что может привести к увеличению доверия со стороны потенциальных клиентов и партнеров. Это может стимулировать дальнейший органический рост аудитории и увеличение вовлеченности пользователей.

Объект исследования – информационная система по платной накрутке социальных сетей.

Предмет исследования – предметом исследования являются разработка и оптимизация стратегий платного продвижения в социальных сетях, а также методы накрутки различных активностей для увеличения онлайн-присутствия бренда, или продвижения компании или бизнеса.

Цель исследования – рассмотреть и определить эффективные методы платного продвижения и накрутки различных активностей в социальных сетях, создав свою собственную удобную информационную систему с современными требованиями.

Методы исследования – в рамках исследования будут использованы методы анализа исходных данных о целевой аудитории, анализ действующих стратегий конкурентов, а также экспериментальные методы для оценки эффективности различных подходов к платному продвижению и накрутке в целом.

Задачи исследования:

* изучить особенности функционирования сервисов по накрутке различных активностей в социальных сетях;
* изучить законодательства и политики социальных сетей, с проведением анализа правил и политики использования социальных платформ;
* рассмотреть ряд существующих разработок конкурентов для решения поставленной цели;
* выбрать метод реализации работы автоматизированных процессов информационной системы;
* разработать информационную систему по накрутке социальных сетей;
* провести тестирование разработанной системы на практике и осуществить необходимые корректировки и адаптации в соответствии с полученными результатами;

1. ПРОЕКТИРОВАНИЕ ВЕБ ПРИЛОЖЕНИЯ

1.1 Цели и задачи выполняемой системы

Основными целями разрабатываемой информационной системы по накрутке социальных сетей является:

* Управление транзакциями для пользователей: разработка удобного интерфейса для пользователей, обеспечивающее легкое управление балансом в системе, давая при этом просмотр платежей за всё время;
* Управление транзакциями для администраторов: возможность просмотреть в удобном интерфейсе все транзакции сервиса, давая удобную возможность отменять операции, связанные с балансом пользователя;
* Управление сервисами социальных сетей и типами услуг сервиса: администратор имеет возможность работать с важными данными;
* Возможность оформления заказа пользователям: основным функционалом в приложении является оформление заказов по накрутке от зарегистрированных пользователей через интерфейс со всеми удобствами;
* Возможность администрировать задания (заказы) пользователей: администратор имеет возможность просмотреть все задания, созданные пользователями, а также имеет возможность отменять задания в случае какой-либо оплошности или ошибки пользователя;
* Интерфейс для пользователей: предоставление пользователю удобного и привлекательного интерфейса для взаимодействия с разделами магазина;

Задачами информационной системы являются:

* Разработка удобного интерфейса для администраторов, позволяющего легко работать с товарами, их категориями, пользователями и отзывами;
* Предоставление администратором возможности легко и эффективно управлять пользователями, обеспечивая возможность в добавлении, редактировании и удалении пользователе, а также специальных возможностей таких как сброс пароля;
* Обеспечение стабильной и бесперебойной работы сервиса, обеспечивая комфортное пользование приложения. Важно обеспечить страницы и функционал различными проверками от неподключенной базы данных или сервера;

1.2 Анализ конкурентов

Анализ конкурентов является важным этапом для понимания текущей ситуации на рынке и разработки эффективных стратегий. В данной работе будет проведён анализ конкурентов в сфере накрутки социальных сетей, с целью выявления их сильных и слабых сторон, а также выявления пробелов в их действиях.

Перед тем как перейти к анализу конкурентов, стоит отметить актуальные тренды в данной области. На сегодняшний день появилось множество сервисов, и они могут предлагать достаточно широкий и масштабный спектр услуг накрутки подписчиков, лайков, комментариев, а также интересные разделы из панели каждого проекта конкурента. Некоторые сервисы просто стали китами, которые живут более 3-4 лет. Однако, с ростом популярности социальных сетей их алгоритмы становятся все более сложными, что приводит некоторых конкурентов к краху, так как не принимают вовремя решений о пересмотре стратегий по действиям при накрутке социальных сетей.

Каждый из нижеописанных конкурентов обладает своими уникальными особенностями и недостатками. Потребуется определить правильно минусы и плюсы каждого из конкурентов, и внедрить решения в свой проект, тем самым имея меньше слабых сторон в проекте.

Сайт <https://topsmm.ru/> является первым из конкурентов. Из минусов – сам проект имеет дизайн интерфейс уровня 2016-2017 года без адаптивности под мобильные устройства. Также для того, чтобы найти определенную услугу для конкретной социальной сети, человеку требуется очень много листать и искать то, что нужно ему – отсутствует интуитивно-понятное деление по группам.

Из плюсов 1 конкурента – у сайта богатый перечень услуг по накрутке множества нужных социальных сетей. Также плюсом можно обозначить панель (личный кабинет) к этому сайту, сделанная на CMS WordPress. Данный конкурент владеет преимуществом в виде системы рефералов.

От данного конкурента была взята реализация на будущее – реализовать систему рефералов в сервисе, а также построить удобный интерфейс отображения сервисов, типы услуг к конкретному сервису.

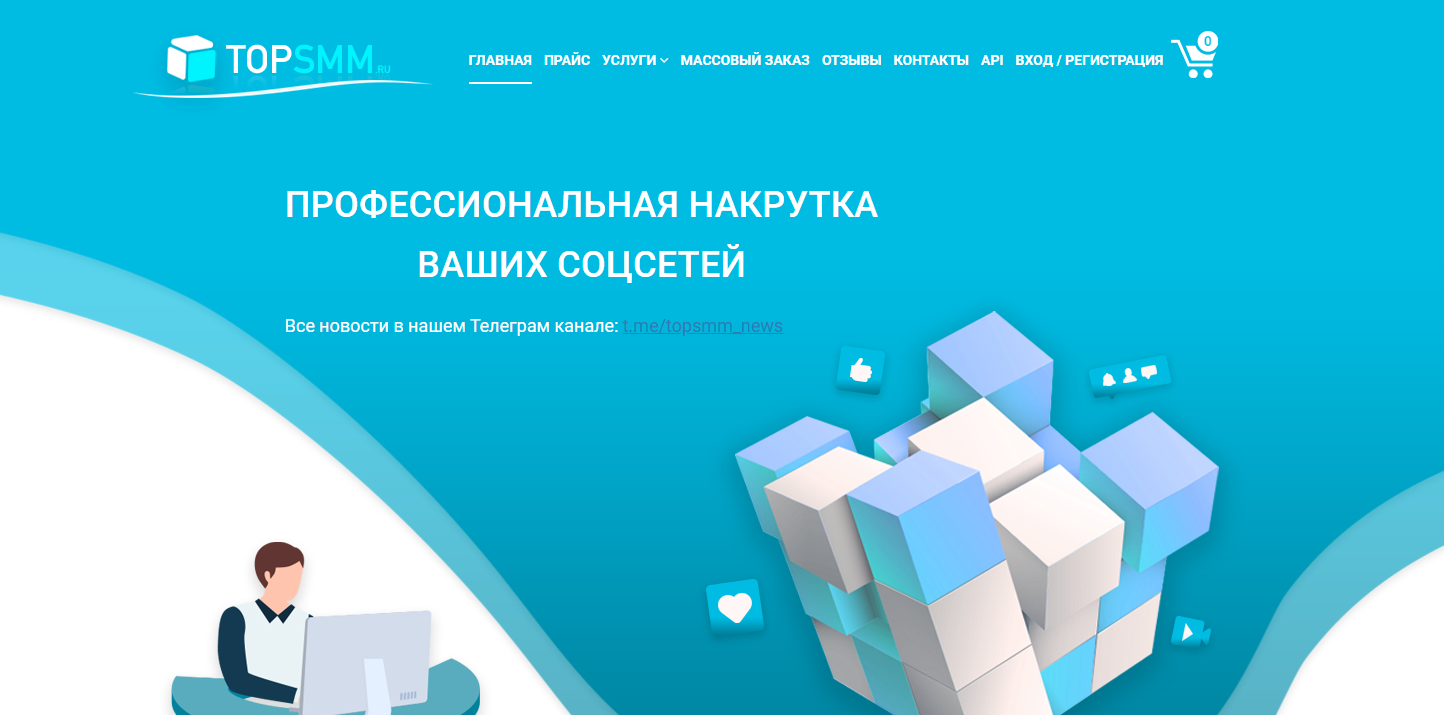


Рисунок 1 – Сайт конкурента <https://topsmm.ru/>

Сайт <https://senderin.ru/> является вторым конкурентом, который имеет более современный дизайн интерфейса, и более масштабный функционал внутри. Также стоит отметить то, что сайт имеет тип приложения PWA (прогрессивное веб-приложение) – это технология, которая визуально и функционально преобразует сайт в приложение. Его можно открыть в браузере или загрузить как обычное мобильное приложение на своём смартфоне. Удобством PWA является то, что переходы между разделами происходят без перезагрузок, и обновляется только часть контента на сайте, тратив меньше ресурсов устройства пользователя.

Сайт имеет удобный функционал оформления заказа с селекторами в которых находится контент. В панели имеется листинг всех услуг, которые может предложить данный сервис. Каждый раздел на сайте сделан минималистично со всеми современными требованиями к дизайну.

Также из сильных сторон можно обозначить то, что в системе имеется система тикетов (общение между администрацией и пользователями для решения определённых проблем), партнёрская (реферальная) сеть и автопостинг для социальной сети VK и Telegram.

От данного конкурента была взята реализация дизайна интерфейса в проект учебной практики, а также удобная реализация (как технически, так и внешний вид) оформления заказа для обычного пользователя.

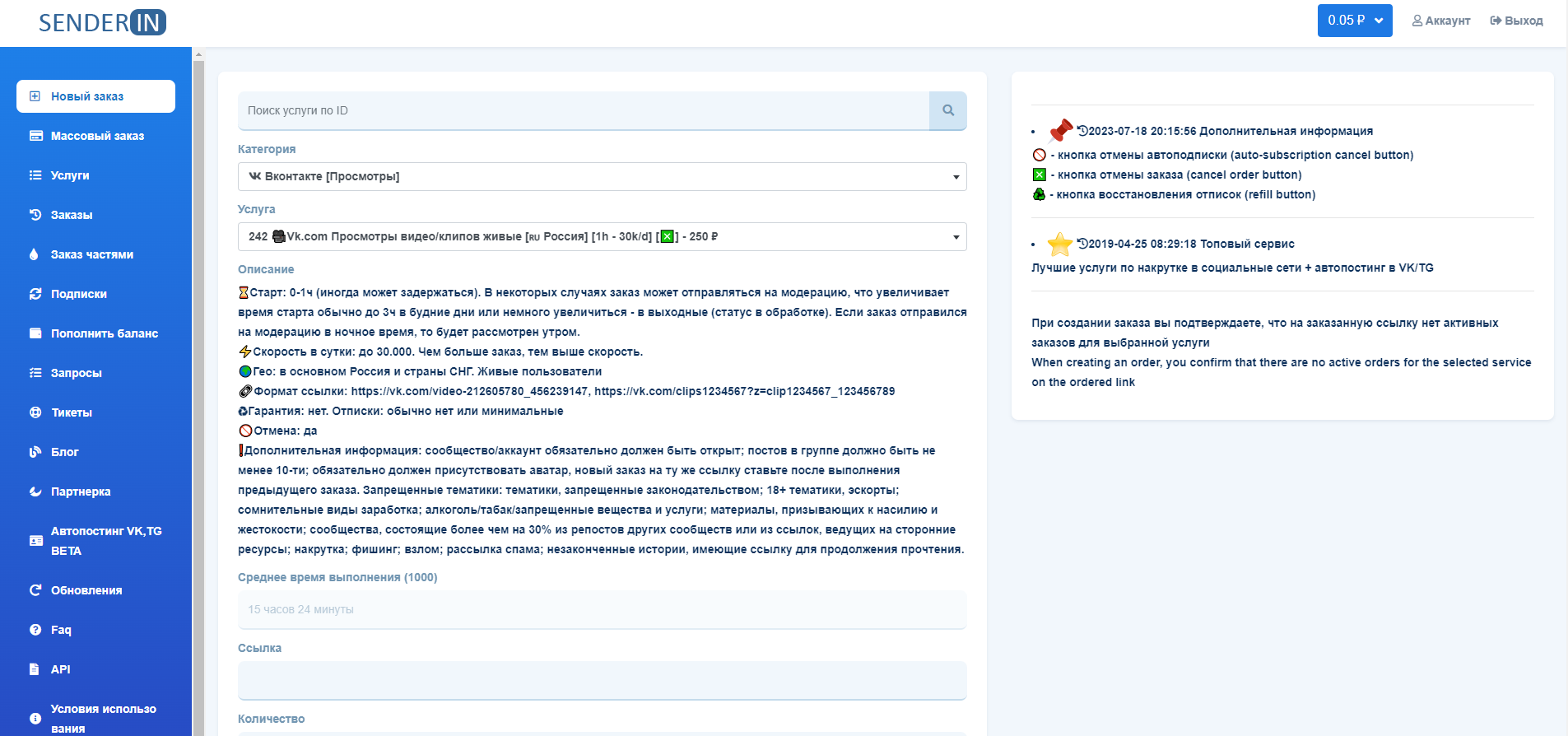


Рисунок 2 – Сайт конкурента <https://senderin.ru/>

Сайт <https://nakru-ti.com/> является последним конкурентом из всех тех которые нужно было проанализировать, выделив сильные и слабые стороны. Сайт является самым продвинутым по дизайну и интерфейсу. При заходе в интерфейс нас будет встречать сразу различные панели, где можно найти быстро те данные, которые потребуется для пользователя.

Сайт сделан на технологии PWA, как и второй конкурент, давая хороший скачок по скорости работы с данными и отображения интерфейса для пользователя. У сайта имеются разделы реферальной системы, API система с подробной документацией. Интерфейс оформления заказа идентичен как у 2 конкурента.

От данного конкурента можно взять множество реализаций по дизайну интерфейсов, а также различных технических аспектов, которые планируется внедрить в проект.

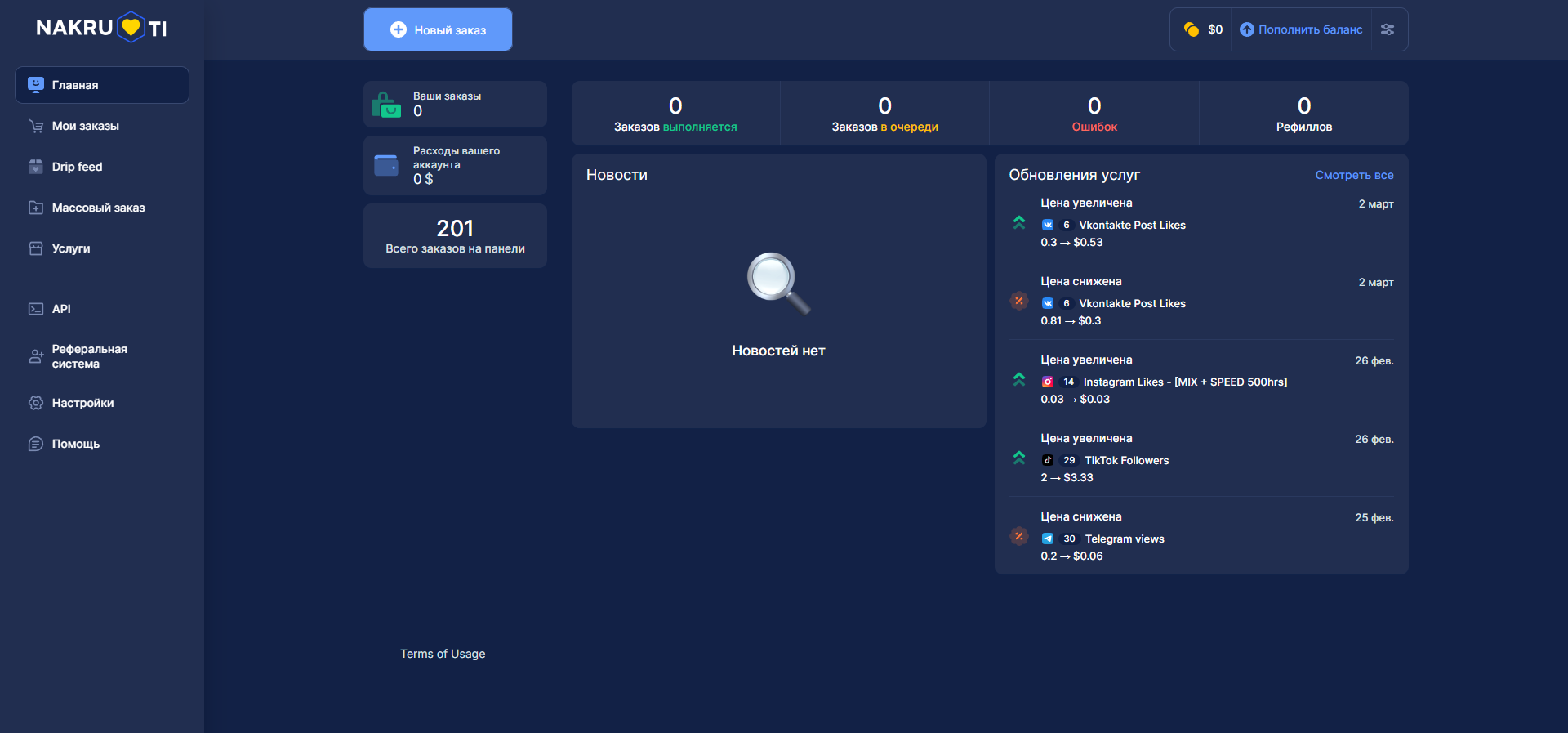


Рисунок 3 – Сайт конкурента <https://nakru-ti.com/>

С учетом анализа конкурентов мы стремимся объединить лучшие практики, предложенные указанными ресурсами, и создать уникальное пространство дизайна интерфейса у сервиса, которое сочетает в себе интуитивность, глубину знаний и работа с разными данными в пару кликов. Сервис рассчитан на широкую аудиторию, обеспечивая реальное платное продвижение в социальных сетях.

1.3 Техническое задание

В рамках технического задания были выделены ключевые аспекты для обеспечения функциональности и безопасности сайта.

Сервис по онлайн накрутке социальных сетей предназначен для предоставления клиентам возможности создавать свои задания для накрутки своей активности в социальных сетях. В разработку входит создание удобной и интуитивно понятной онлайн-платформы для клиентов и администраторов сервиса.

В первую очередь, предусмотрено использование капчи и другим мер против спама с целью защиты от атак ботов и предотвращения DDoS-атак, что повышает общую безопасность веб-ресурса. Для хорошей безопасности информационной системы должно быть приобретено домен, хостинг, и SSL-сертификат – это всё поможет обеспечить безопасное соединение и общение между клиентом и сервером.

В информационной системе должна быть предусмотрена отдельная административная часть. Данная часть системы будет доступна только авторизованному пользователю, который имеет больше привилегий, чем обычный пользователь.

Из функциональных требований исходят следующие пункты.

Управление сервисами и услугами для каждого сервиса:

* Создание, редактирование и удаление информации о сервисах (название, изображение);
* Создание, редактирование и удаление информации об услугах для конкретного сервиса (название, цены);

Управление транзакциями:

* Возможность пополнять средства на внутренний баланс системы;
* Просмотр своих транзакций, а также просмотр всех транзакций в системе (административная часть);
* Возможность отменять транзакции разных типов у пользователей;

Управление пользователями:

* Авторизация и аутентификация пользователей;
* Изменение информации о себе (текстовые данные, изображение);
* Возможность просмотра информации пользователей администраторами;
* Возможность добавлять, редактировать и удалять пользователей от имени администратора;

Интерфейс для пользователей:

* Просмотр посадочной страницы, своих транзакций, созданных заданий, информации о себе, а также страница формирования нового заказа (задания);

Дизайн системы должен быть интуитивно понятный и удобный для пользователей, соответствуя их потребностям и ожиданиям. Создание прототипов и макетов позволяет заказчику и разработчикам предварительно визуализировать функционал системы и согласовать детали дизайна.

На данном этапе необходимо описать примерную архитектуру разрабатываемой системы. Для того, чтобы реализовать данную систему, будет использоваться клиент-серверная архитектура. В клиент-серверной архитектуре имеется три звена:

1. Представление данных — на стороне клиента;
2. Прикладной компонент — на выделенном сервере приложения, здесь происходит вся бизнес-логика, которая в дальнейшем будет взаимодействовать с клиентом;
3. Управление ресурсами — сервер БД (базу данных), который и представляет запрашиваемые данные;

Для данного проекта выбрана эта архитектура, так как у нее есть ряд преимуществ перед другими архитектурами:

1. Высокую степень гибкости и масштабируемости;
2. Высокую безопасность;
3. Высокую производительность (т. к. задачи распределены между Серверами);

MySQL — это мощная система управления базами данных, которая помогает хранить и организовывать информацию об сервисе. База данных будет содержать информацию о транзакциях пользователей, сервисах, услугах для сервиса и заданиях (заказы).

На рисунке 3 представлена структура клиент–серверной архитектуры:

Изображение выглядит как текст, диаграмма, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4 – Клиент–серверная архитектура

Требования к системе разработаны в соответствии со стандартом качества программного обеспечения ISO 9126:2001, описывающим многоуровневую модель характеристик качества и соответствующий им набор атрибутов.

Требования к информационной системе представлены в таблице 1:

Таблица 1 – Требования к системе

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | Метки | | Требования |
| Удобство | | | |
| Простота использования | Среднее время, необходимое  пользователю, чтобы найти  необходимый элемент в программе | | Поиск отдельного элемента интерфейса не должен занимать более 1 секунды |
| Привлекательность | Соответствие интерфейса  требованиям | | 99% элементов должны быть работоспособны для  пользователя |
| Обучаемость | Показатель, затрачиваемый  пользователями на обучение  работе | | Пользователь должен изучить работу ПО за первые 3-5 минут  использования |
| Производительность | | | |
| Временная эффективность | Время выполнения компонента программы | - Приложение должно  обслуживать любую функцию не дольше 1 сек  (не включая задержки в сети); | |
| Эффективность использования ресурсов | Объемы ресурсов требуемых для выполнения задач | - Постоянная память,  используемая программой – не более 120МБ;  - Приложение должно потреблять не более 3МБ  памяти на каждый неактивный сеанс с  пользователем; | |
| Удобство установки | Легкость установки | Необходима установка:  XAMPP, MySQL, PhpMyAdmin | |
| Переносимость | | | |
| Способность к сосуществованию | Способность ПО сосуществовать с другими  программами в общем окружении | Программа не должна  занимать большие объемы памяти (максимум 1ГБ) | |
| Удобство замены другого ПО данным | Совместимость структур данных | - Система должна быть  разработана по общим  принципам и не требовать навыков от пользователя;  - Должна поддерживать  внедрение новых версий;  - Система должна быть применима вместо других  программных систем для решения тех же задач в определенном окружении; | |

Требования к численности и квалификации персонала системы и режиму его работы:

1. В системе существуют 2 группы пользователей: пользователи и администраторы. Пользователи могут просматривать свои настройки профиля, редактировать профиль, просмотреть транзакции, пополнять баланс, просматривать свои созданные задания и создавать новые задания.
2. Администратор выполняет следующие функции: просмотра, занесение данных, редактирование, удаление данных, и отмена операций. Данный функционал относится к работе с пользователями, заданиями (заказам), транзакциями, сервисами и услугами сервиса.
3. Пользователи системы должны обладать базовыми навыками взаимодействия с компьютером и использования веб-ресурсов. Опыт работы с персональными компьютерами или устройствами, доступ в Интернет и уверенное понимание основных принципов работы веб-интерфейсов будет полезным для комфортного использования системы.

Кроме того, система предоставляет возможность быстрого и удобного доступа к актуальной информации о сервисах, услугах, транзакциях и заданий, обеспечивая удовлетворение потребностей как пользователей, так и администраторов.

В таблице 2 представлены требования к надежности.

Таблица 2 – Требования к надёжности

|  |  |  |
| --- | --- | --- |
| Атрибуты | Метрики | Требования |
| Надежность | | |
| Устойчивость к отказу | Коэффициент аварийных  отказов | - Не менее 90% ошибок  в программе должны  обрабатываться без  экстренного завершения;  - Обеспечение бесперебойного питание  активного сетевого  оборудования;  - Постоянная работа сервера; |
| Способность к восстановлению | Отношение количества числа успешного восстановление работы системы к ее аварийному завершению | - Восстановление  работоспособности  системы не должно превышать 5 минут; |
| Продолжение таблицы 2 | | |
| 1 | 2 | 3 |
| Уровень зрелости | Среднее время работы без сбоев | - Система должна  работать 24 часа в сутки;  - Система должна оповещать пользователя при некорректных действиях в системе;  - Данные, которые вводит пользователь должны сохраняться; |

Средняя доступность приложения должна составлять не менее 99%

Среднее время между сбоями — это среднее время, за которое компонент или модуль может выполнять свои функции без перерыва.

Измеряется от начала работы до момента следующего сбоя.

Среднее время работы без сбоев должно составлять не менее 1500 часов.

В таблице 3 представлен перечень и критерии отказов для каждой функции, по которой задаются требования по надежности.

Таблица 3 – Перечень и критерии отказов для каждой функции

|  |  |
| --- | --- |
| Функции | Критерии отказа |
| Отображение данных | Отсутствие соединения с БД |
| Взаимодействие с товарами | Отсутствие соединения с БД |
| Поиск | Отсутствие соединения с БД |
| Добавление | Отсутствие соединения с БД |
| Редактирование | Отсутствие соединения с БД |
| Удаление | Отсутствие соединения с БД |
| Отмена транзакций | Отсутствие соединения с БД |
| Авторизация и регистрация | Отсутствие соединения с БД |

Для нормального функционирования информационной системы на базе PHP Laravel требуется обеспечить соответствующую инфраструктуру:

* наличие сервера для хостинга приложения;
* персональные устройства для работы с приложением;
* надежное интернет-соединение для доступа к системе;
* программа XAMPP для локальной разработки и тестирования приложения;
* обязательное создание резервных копий информации для ее сохранности и возможности восстановления при необходимости;
* требование к наличию сервера базы данных MySQL для хранения данных приложения;
* мониторинг производительности и доступности системы для быстрого выявления и устранения проблемных ситуаций;
* настройка системы мониторинга безопасности для обнаружения и предотвращения угроз безопасности приложения;
* регулярное обновление программных компонентов и фреймворка для обеспечения безопасности и работоспособности приложения;

При анализе предметной области разрабатываемой системы были выделены следующие сущности, на основе которых будет разрабатываться база данных системы:

* Пользователи;
* Транзакции пользователей;
* Баланс пользователей;
* Сервисы (социальные сети);
* Услуги сервисов;
* Заказы (задания);

База данных информационной системы должна состоять из следующих связанных таблиц:

1. Users - содержащая в себе информацию о пользователях системы;
2. Balance Transactions - содержащая в себе информацию о транзакциях;
3. User Balances - содержащая в себе информацию о текущем балансе пользователя;
4. Services - содержащая в себе информацию о сервисах;
5. Service Items - содержащая в себе информацию о услугах для конкретного сервиса;
6. Orders – содержит в себе информацию об заказах (заданиях) пользователей;

Данные о пользователях представлены в таблице Users:

Таблица 4 – Данные о пользователях

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | BIGINT | 20 | Идентификатор пользователя | Первичный ключ |
| login | VARCHAR | 255 | Логин пользователя | NOT NULL |
| name | VARCHAR | 255 | Имя пользователя | NOT NULL |
| email | VARCHAR | 255 | Почта пользователя | NOT NULL |
| password | VARCHAR | 255 | Хешированный пароль пользователя | NOT NULL |
| avatar | VARCHAR | 255 | Уникальный файл аватар пользователя | - |
| role | ENUM | 255 | Роль пользователя | NOT NULL |

Данные о транзакциях системы будут размещаться в таблице BalanceTransactions

Таблица 5 – Данные о транзакциях системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | BIGINT | 20 | Идентификатор транзакции | Первичный ключ |
| user\_id | BIGINT | 20 | Идентификатор пользователя | Внешний ключ |
| amount | INT | 20 | Сумма транзакции | NOT NULL |
| type | VARCHAR | 255 | Тип транзакции | NOT NULL |
| Cancelled | BOOLEAN | - | Отмена транзакции | - |

Данные о текущем балансе пользователя будут хранится в таблице UserBalances

Таблица 6 – Данные о текущем балансе пользователей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | BIGINT | 20 | Идентификатор  баланса | Первичный ключ |
| user\_id | BIGINT | 20 | Идентификатор пользователя | NOT NULL |
| balance | INT | 20 | Баланс пользователя | NOT NULL |

Данные о сервисах хранятся в таблице Services

Таблица 7 – Данные о сервисах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | BIGINT | 20 | Идентификатор  сервиса | Первичный ключ |
| name | VARCHAR | 255 | Имя сервиса | NOT NULL |
| logo | VARCHAR | 255 | Уникальный файл логотипа сервиса | NOT NULL |

Данные о услугах сервиса будут находится в таблице ServiceItems

Таблица 8 – Данные о услугах сервиса

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | BIGINT | 20 | Идентификатор услуги | Первичный ключ |
| service\_id | BIGINT | 20 | Идентификатор сервиса | Внешний ключ |
| name | VARCHAR | 255 | Имя услуги | NOT NULL |
| price\_low | INT | 20 | Цена за низкое качество | NOT NULL |
| price\_medium | INT | 20 | Цена за среднее качество | NOT NULL |
| price\_high | INT | 20 | Цена за высокое качество | NOT NULL |

Данные о всех заказах (заданиях) пользователей должны располагаться в таблице Orders

Таблица 9 – Данные о заданиях пользователей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | BIGINT | 20 | Идентификатор заказа | Первичный ключ |
| user\_id | BIGINT | 20 | Идентификатор пользователя | Внешний ключ |
| service\_id | BIGINT | 20 | Идентификатор сервиса | Внешний ключ |
| service\_item\_id | BIGINT | 20 | Идентификатор услуги сервиса | Внешний ключ |
| source\_link | VARCHAR | 255 | Ссылка на ресурс | NOT NULL |
| quantity | VARCHAR | 255 | Количество элементов накрутки | NOT NULL |
| total\_price | INT | 20 | Итоговая стоимость | NOT NULL |
| status | VARCHAR | 255 | Статус заказа | NOT NULL |

Сайт должен быть структурирован на четко взаимосвязанные разделы, что сможет обеспечивать удобную навигацию и легкость в поиске информации для пользователей.

Навигационное меню, посадочная страница и панель управления разработаны с учетом удобства пользователей, обеспечивая легкость перехода между разделами и доступ к основной информации.

Нефункциональные требования играют важную роль в обеспечении эффективной работы и безопасности сервиса. Интерфейс должен быть интуитивно понятным для всех пользователей и адаптивным для корректного отображения на различных устройствах. Безопасность приложения несет важное значение и предполагает реализацию системы аутентификации и авторизации для администраторов и пользователей, а также защиту от уязвимостей веб-приложения, включая SQL-инъекции, XSS и CSRF атаки. Производительность включает оптимизацию загрузки изображений и данных для быстрого отображения страниц каталога, а также минимизацию времени ответа сервера на запросы клиентов.

Нефункциональные требования выполняются для обеспечения безопасности, производительности и удобства использования электронного каталога цветочного магазина:

Интерфейс:

* Интуитивно понятный интерфейс для всех категорий пользователей;
* Адаптивный дизайн для корректного отображения на различных устройствах;

Безопасность:

* Реализация системы аутентификации и авторизации для администраторов и пользователей;
* Защита от уязвимостей веб-приложения, таких как SQL-инъекции, CSRF-атак и XSS-атаки;

Производительность:

* Оптимизация загрузки изображений и данных для быстрого отображения страниц каталога;
* Минимизация времени ответа сервера на запросы клиентов;

Инновационные технологии, безупречная функциональность и надежная безопасность — вот то, что делает этот электронный каталог цветочного магазина неповторимым. Позвольте погрузиться в детали технических требований, определяющих основу его функционирования.

1.3 Дизайн веб-проекта

Для обеспечения максимального комфорта пользователей в процессе работы с информационной системой, был разработан доступный и интуитивно понятный интерфейс.

В разработке логотипе было принято решение сделать его простым вместо нескольких трудных ребрендингов и раздумий, назвав данный сервис Glorify Boost. Так как эта тематика относится к сфере накрутки социальных сетей, то была сделана иконка в виде лайка, так как данная иконка является самой популярной в нашей жизни. В этом же логотипе были выбраны 2 акцентных цвета, которые будут участвовать в проекте.

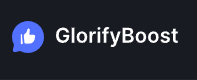


Рисунок 5 – Логотип информационной системы «GlorifyBoost»

Выбор цветовой палитры был основан как минимум 2 цветов из логотипа, что должно привести к общему стилю и узнаваемости приложения. Сайт содержит в себе лишь тёмную тему без возможностей переключаться на светлую.

Вся цветовая палитра была составлена на рисунке 6. Данная палитра является отличным решением, когда цвета будут сочетаться между собой легко и спокойно для каждого пользователя.



Рисунок 6 – Вся цветовая палитра сайта

Типографика – это искусство, обладающее художественными, эстетическими и техническими ценностями, поэтому она заслуживает большего внимания, чем обычно ему уделяет широкая публика. Мы настолько привыкли видеть текст повсюду, что почти никогда не останавливаемся, чтобы взглянуть на буквы, на то, как они созданы и расположены, и, что особенно важно, как они вписываются в дизайн и взаимодействуют с графическим контекстом, в который они помещены.

Одни шрифты выражают энергию и оптимизм, другие придают дизайну элегантность или изысканность, третьи романтичны, и, конечно же, существует множество профессиональных и довольно строгих гарнитур. Правильный выбор шрифта обеспечивает согласованность дизайна, а также чрезвычайно важен с точки зрения брендинга и создания эстетической индивидуальности.

В разработке дизайн был выбран шрифт Inter. Inter – это шрифт без засечек. Так же, как и предыдущие шрифты, довольно нейтральный по настроению, что делает его универсальным.

Этот шрифт хорошо подходит для применения в веб-дизайне благодаря своему чистому и современному виду. Его можно использовать как для основного текста, так и в заголовках. Одним из главных преимуществ Inter является его удобочитаемость. Шрифт имеет большую x-высоту букв, что позволяет легко читать его в мелком размере.

В дальнейшем после с определением цветовой палитрой, разработкой логотипа и выбором шрифта – можно приступать к разработке веб-интерфейса, чтобы приступить к следующему этапу по созданию информационной системы.

В данной информационной системе было принято выбрать один из трендов в дизайне – дизайн, ориентированный на UX. Это одна из неизменных тенденций в веб-дизайне, которая будет актуальна и в следующем году. Дизайнеры по-прежнему стремятся завладеть вниманием посетителей сайта.

Веб-дизайн, ориентированный на пользователя, может сочетать в себе различные техники и приемы. В качестве результата заказчики хотят получить увеличение трафика, повышение конверсии. Достигнуть этих целей можно именно благодаря лояльности посетителей. Каждое взаимодействие или точка соприкосновения на сайте прорабатывается с учётом потребностей пользователя.

На рисунке 7 предоставлен интерфейс главного экрана посадочной страницы информационной системы.

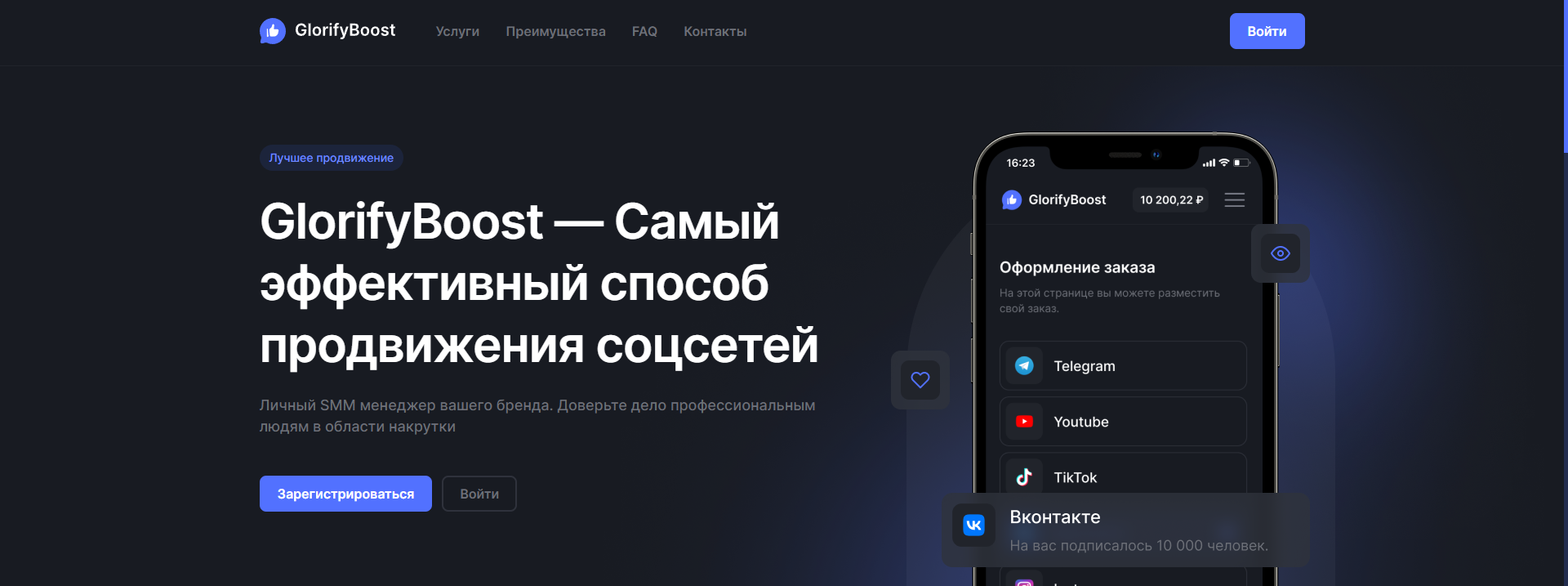


Рисунок 7 – Главный экран посадочной страницы

В данной информационной системе сделано интересное решение в виде авторизации и регистрации на этой же самой главной странице с помощью модальных окон. Данное действие позволило сократить время ожидания и убрать переход на другую страницу для авторизации или регистрации.

На рисунке 8 предоставлен интерфейс модального окна авторизации.

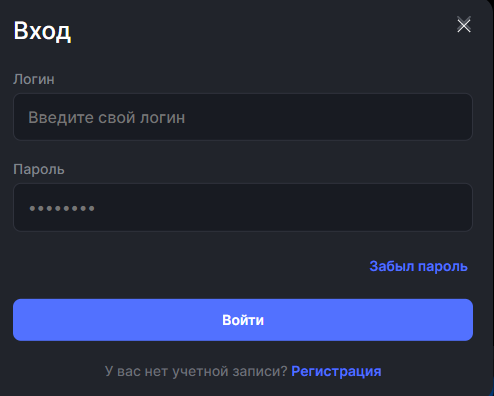


Рисунок 8 – Интерфейс авторизации в системе

На рисунке 9 предоставлен интерфейс для регистрации аккаунта в системе.

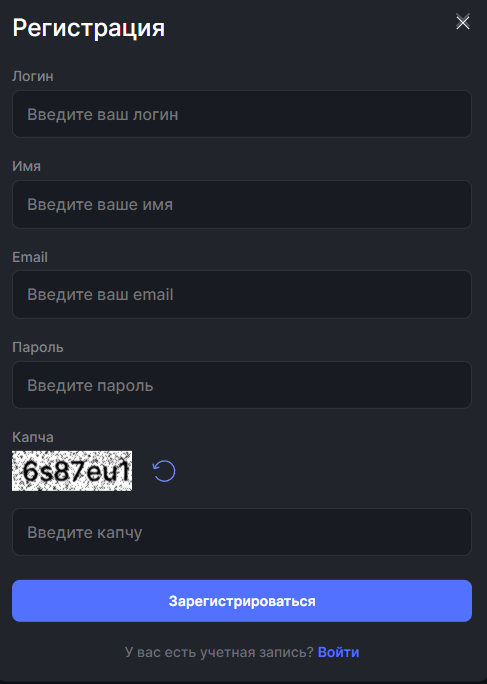


Рисунок 9 – Интерфейс регистрации в системы

В дальнейшем пошла разработка интерфейса, который доступен только для авторизованных пользователей и администраторам. На рисунке 10 предоставлен интерфейс раздела «Настройки».

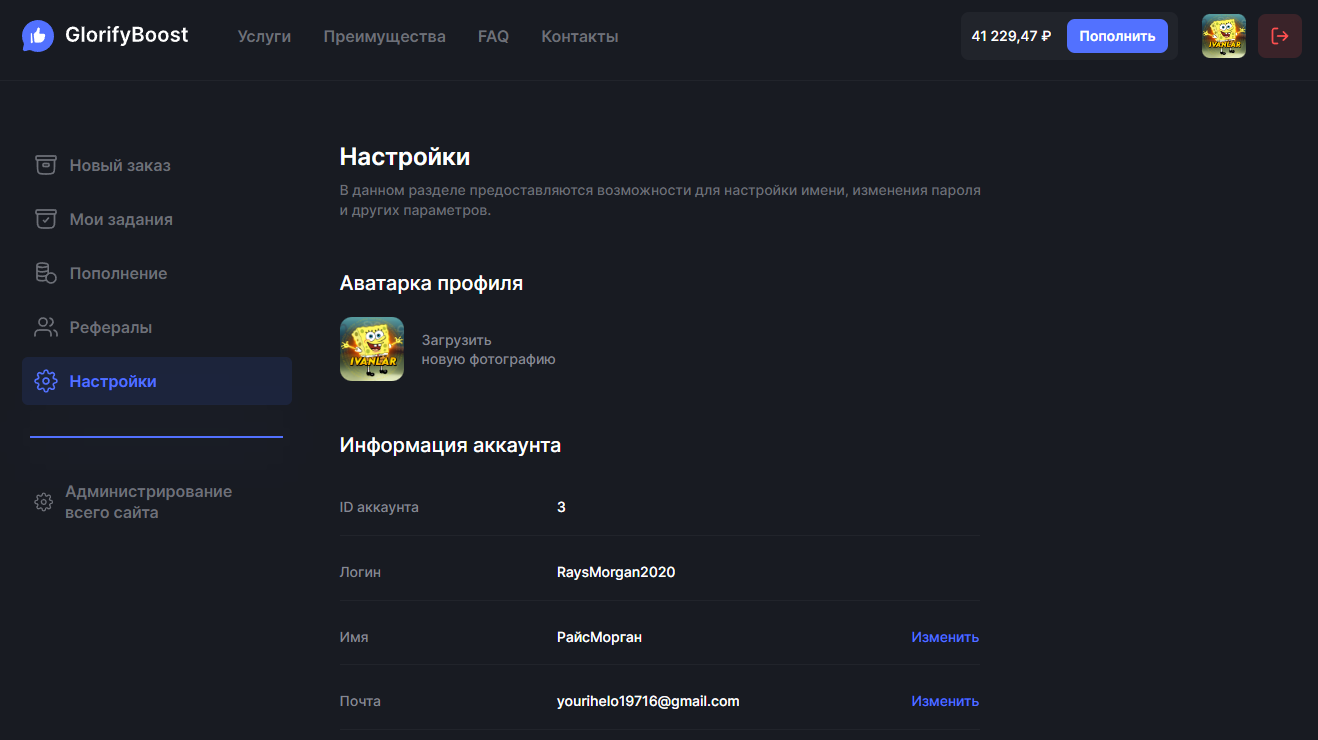


Рисунок 10 – Раздел «Настройки» информационной системы

На рисунке 11 предоставлен интерфейс раздела «Рефералы».

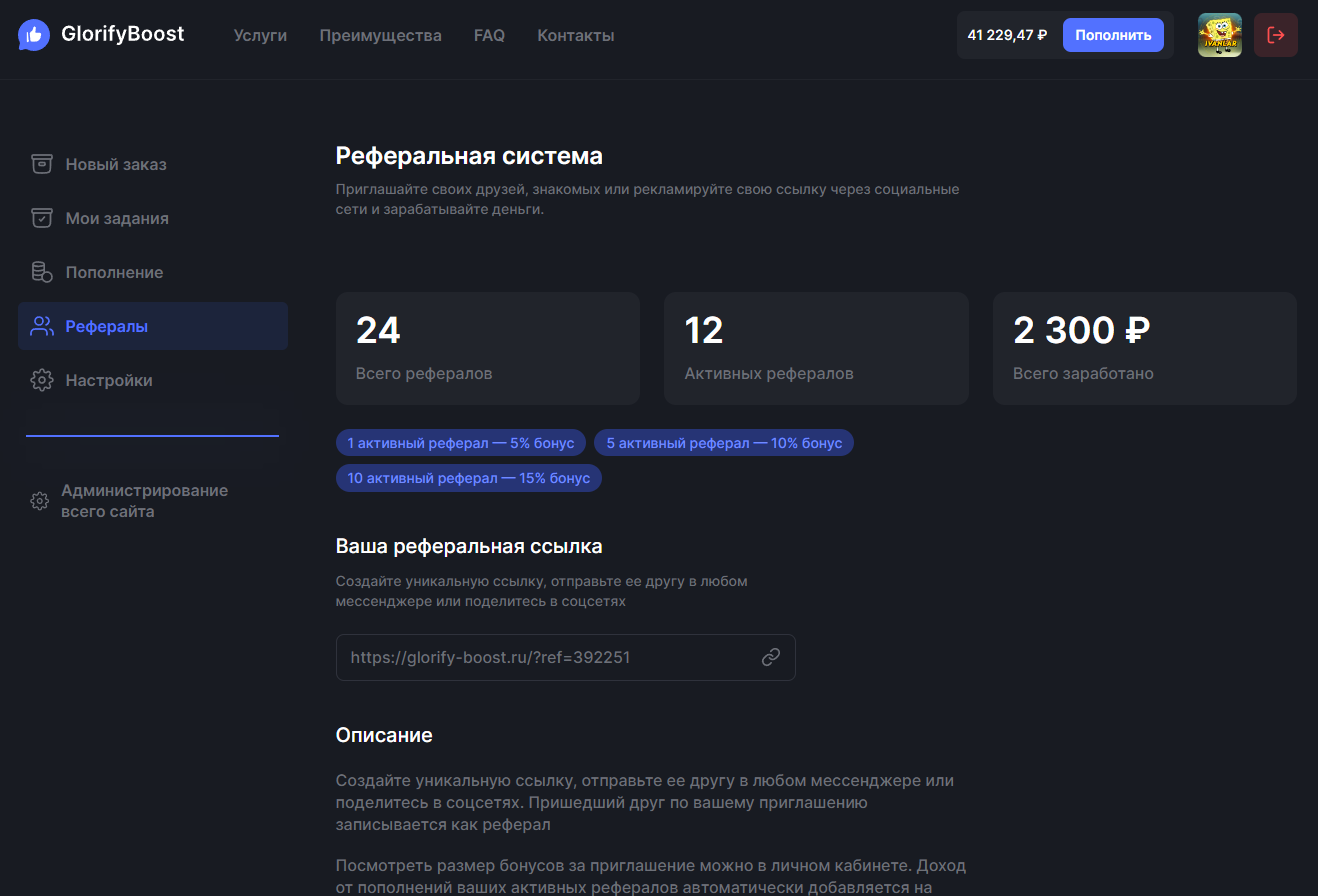


Рисунок 11 – Раздел «Рефералы» информационной системы

На рисунке 12 предоставлен интерфейс раздела «Пополнение».

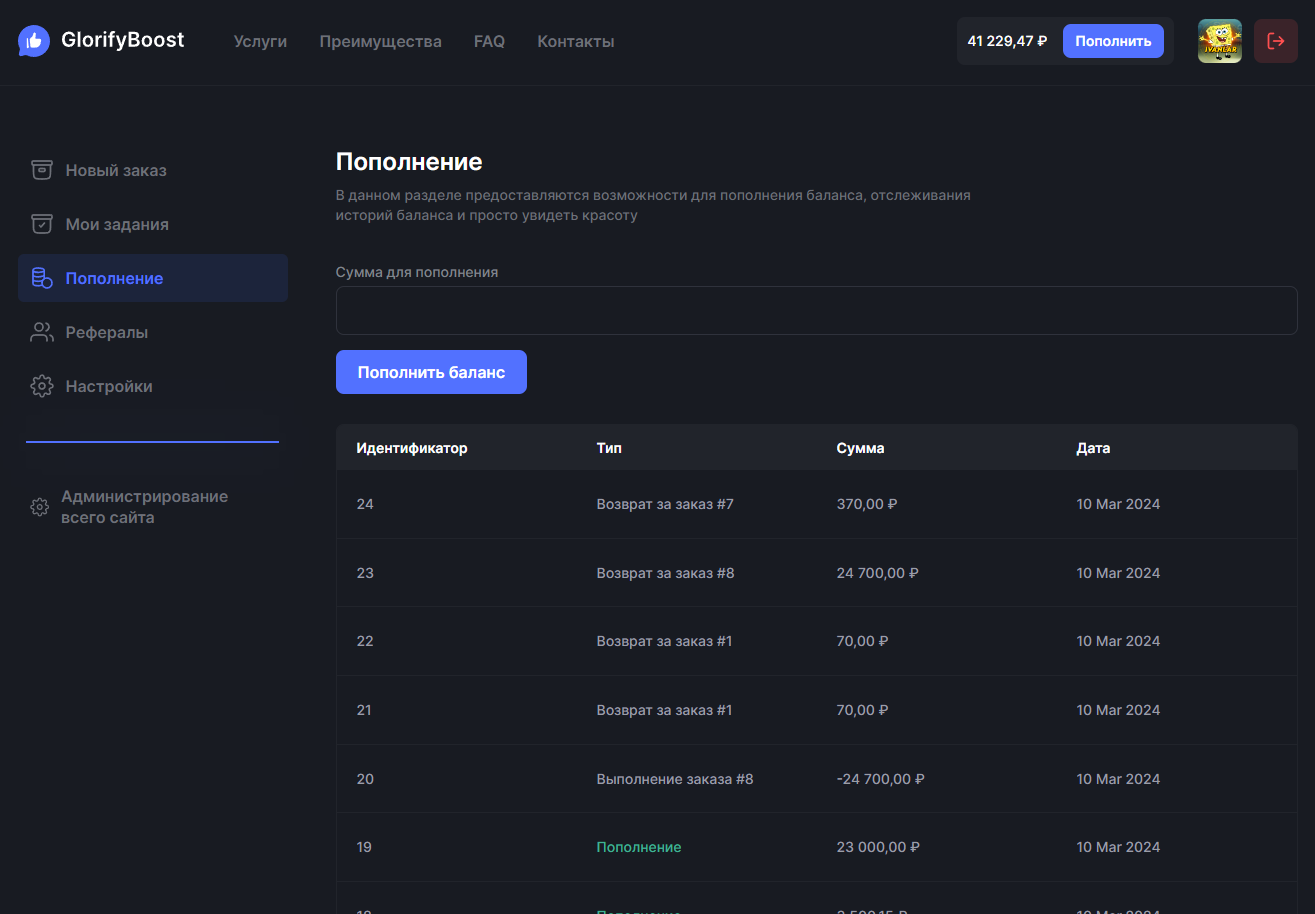


Рисунок 12 – Раздел «Пополнение» информационной системы

На рисунке 13 предоставлен интерфейс раздела «Мои задания».

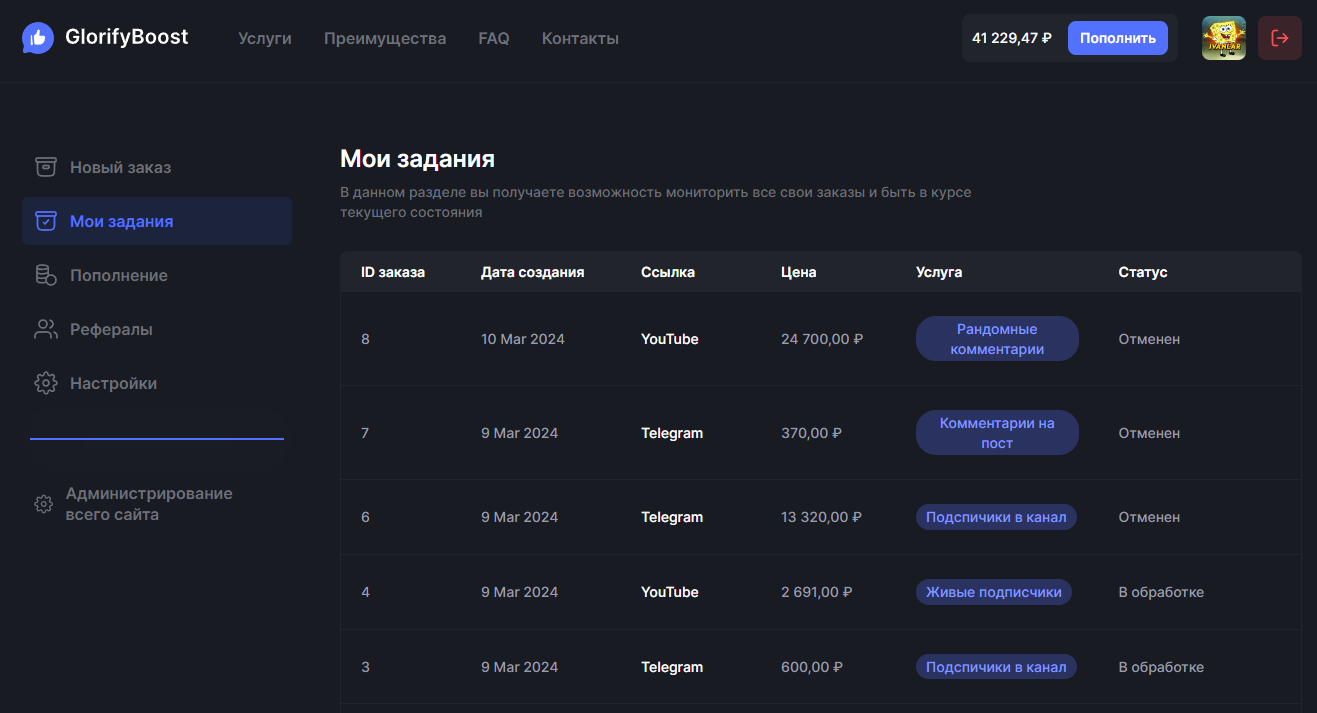


Рисунок 13 – Раздел «Мои задания» информационной системы

На рисунке 14 предоставлен интерфейс страницы «Новый заказ».

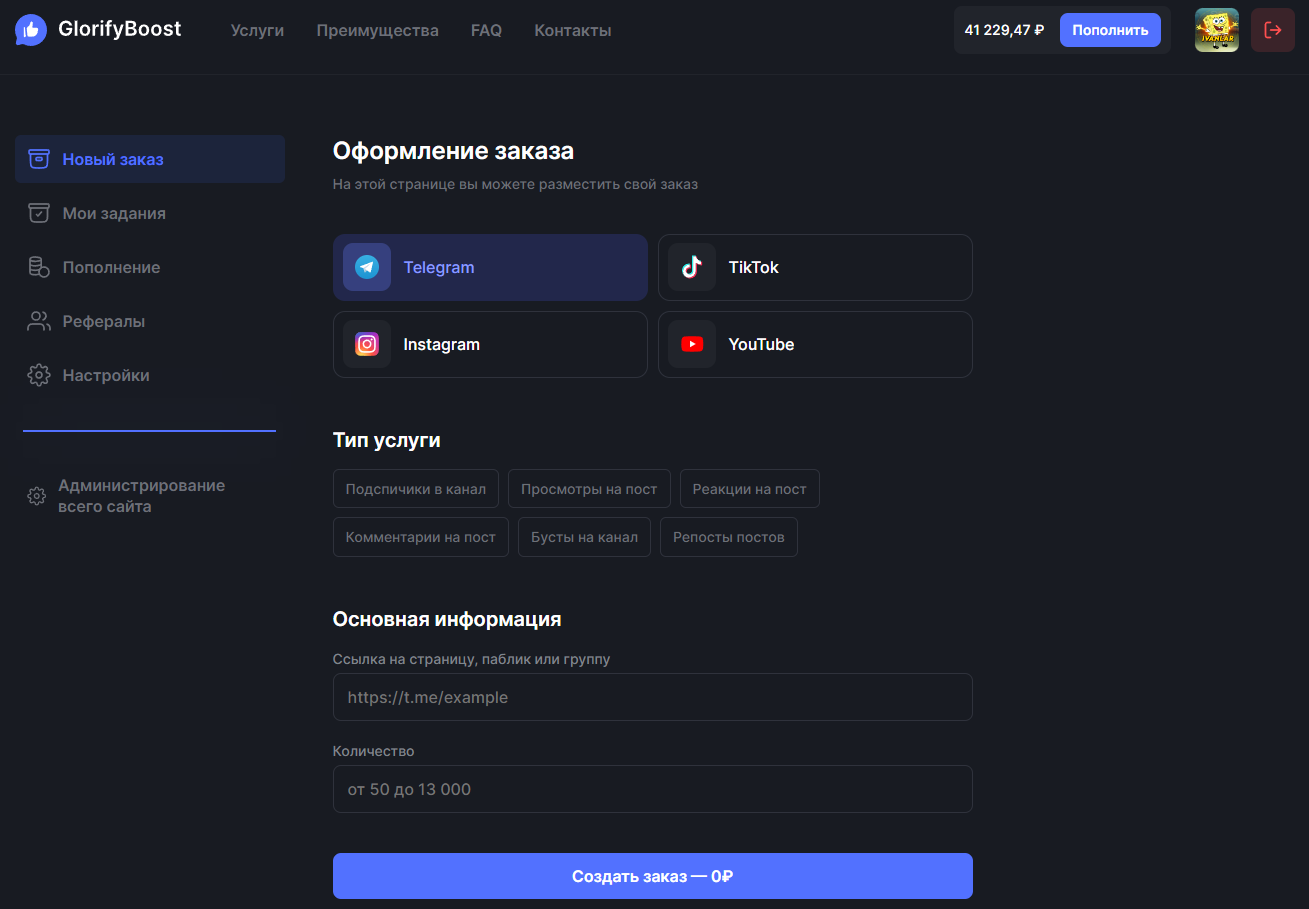


Рисунок 14 – Раздел «Новый заказ» информационной системы

В дальнейшем шла разработка административной части сайта, дав удобный и быстрый доступ администраторам управление контентом. Данная ссылка была везде отображена («Администрирование всего сайта»), что без проблем слилось с обычной панелью пользователя.

На рисунке 15 предоставлен интерфейс раздела «Управление пользователями».

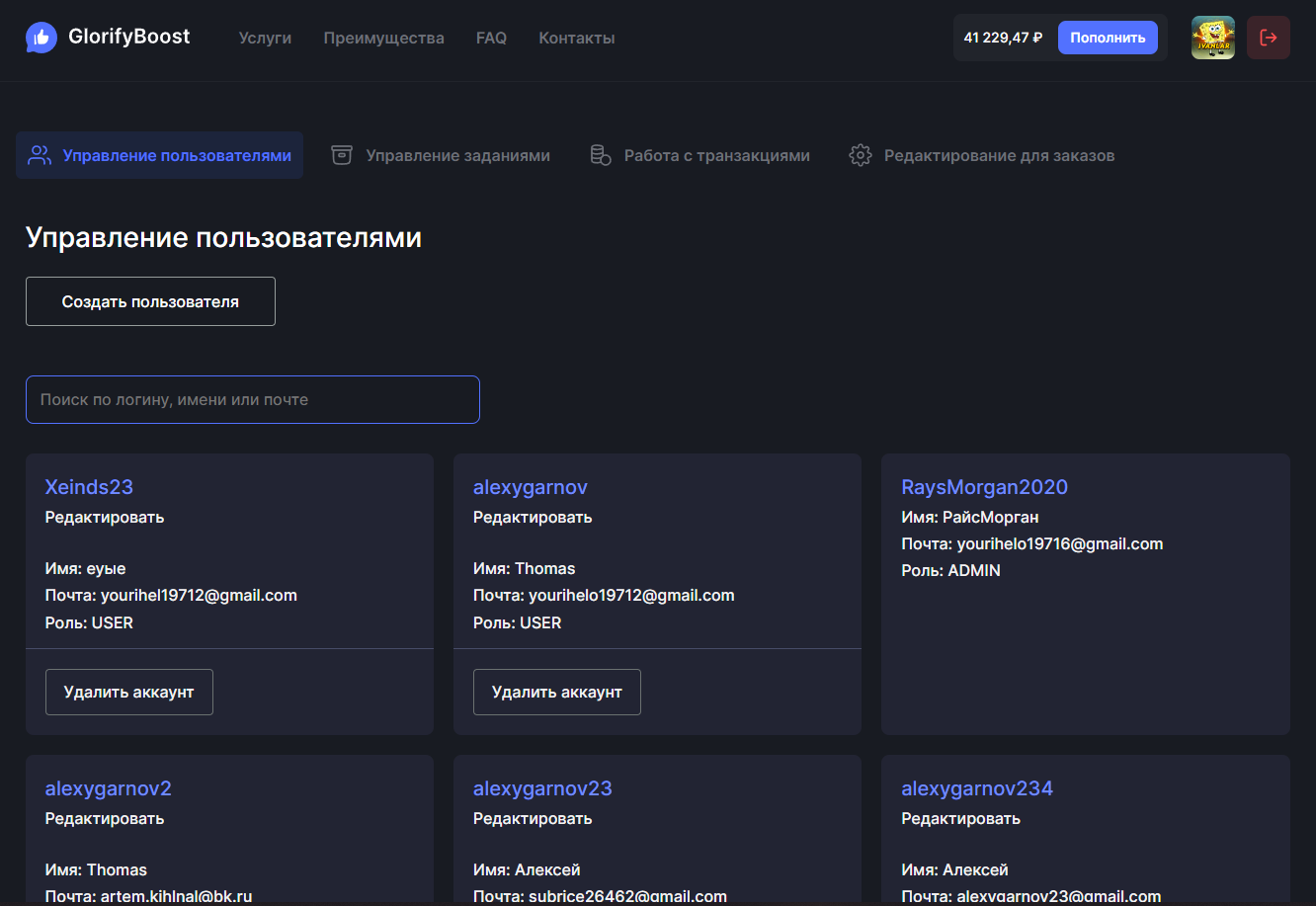


Рисунок 15 – Раздел «Управление пользователями» информационной системы

На рисунке 16 предоставлен интерфейс для удаления пользователя

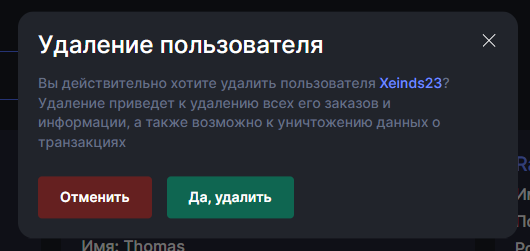


Рисунок 16 – Интерфейс для удаления пользователя

На рисунке 17 предоставлен интерфейс раздела «Работа с сервисами»

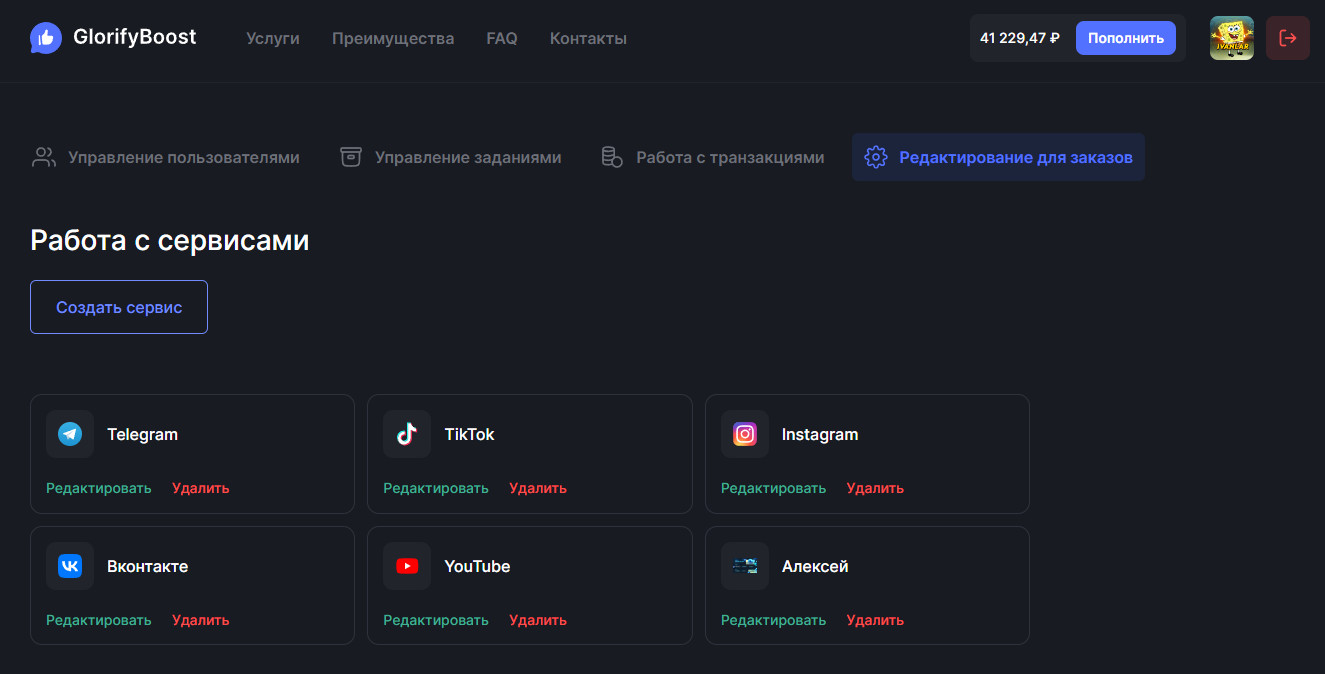


Рисунок 17 – Раздел «Работа с сервисами» информационной системы

На рисунке 18 предоставлен интерфейс для работы с существующим сервисом о социальной сети.

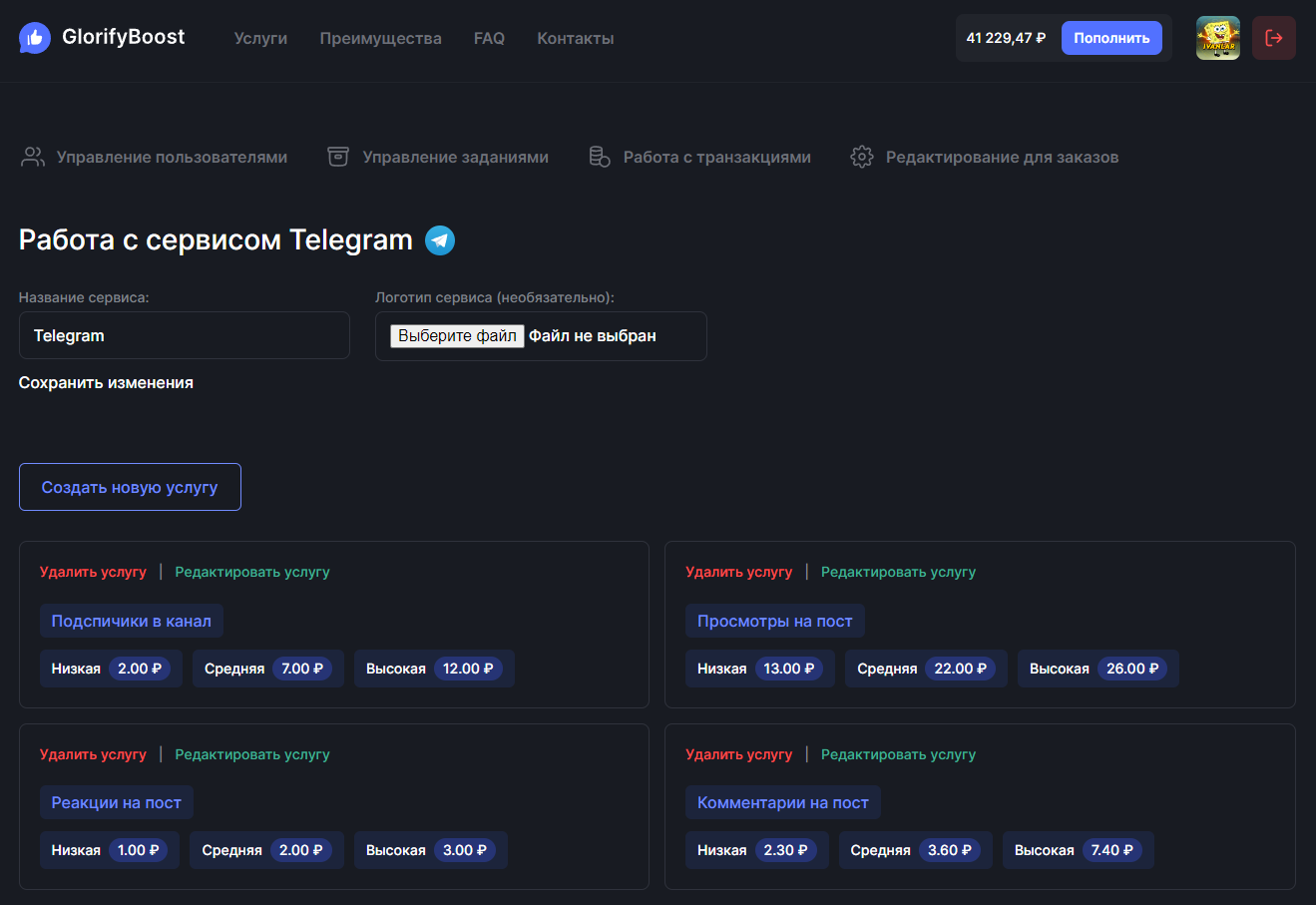


Рисунок 18 – Интерфейс для редактирования сервиса

На рисунке 19 предоставлен интерфейс для создания услуги в сервисе.

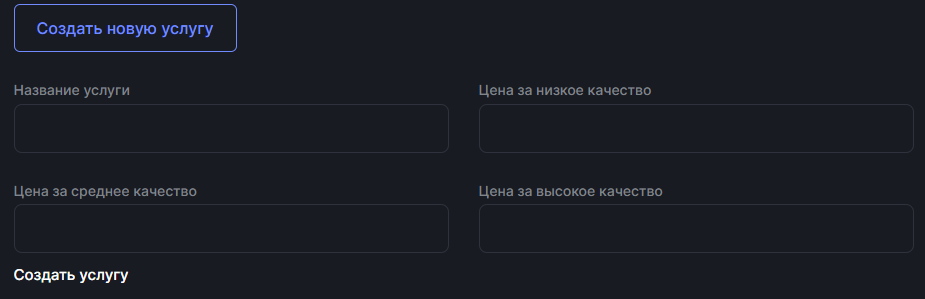


Рисунок 19 – Интерфейс для создания услуги в сервисе

На рисунке 20 предоставлен интерфейс раздела «Работа с транзакциями».

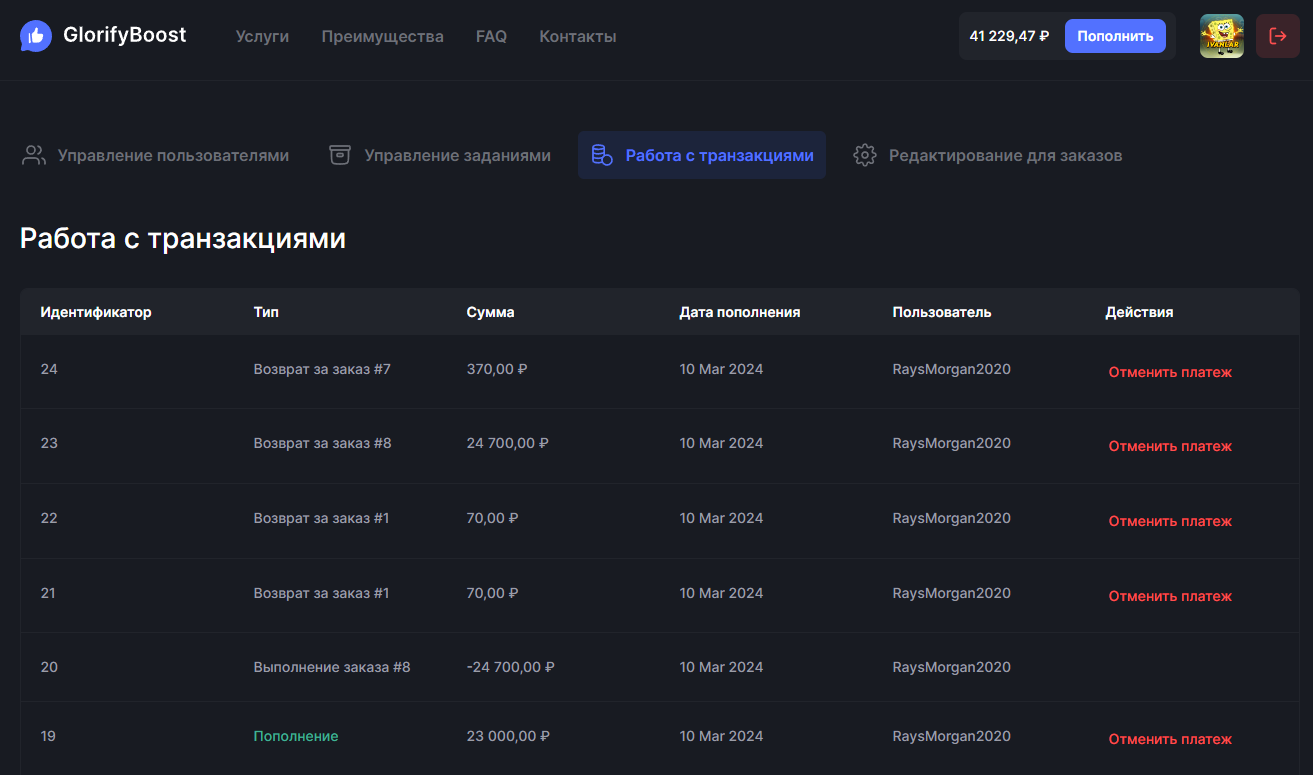


Рисунок 20 – Раздел «Работа с транзакциями» в информационной системе

На рисунке 21 предоставлен интерфейс для отмены платежа транзакции.

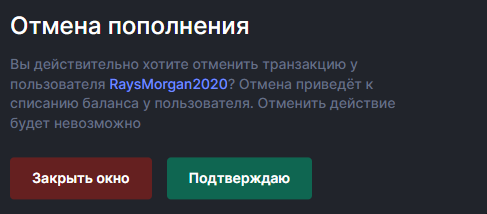


Рисунок 21 – Интерфейс для отмены транзакции

На рисунке 22 предоставлен интерфейс раздела «Управление заданиями».

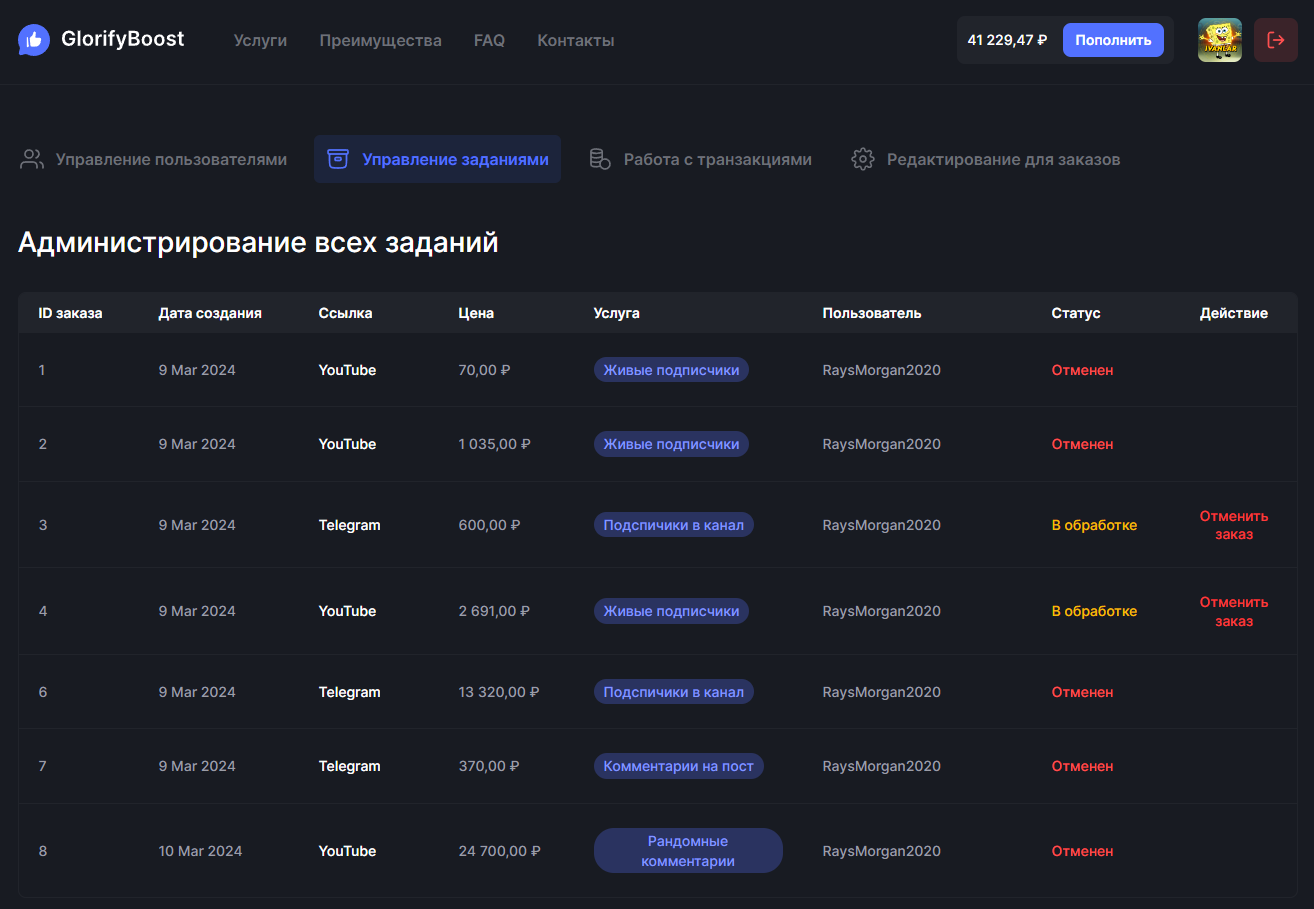


Рисунок 22 – Раздел «Управление заданиями» в информационной системе

На рисунке 23 предоставлен интерфейс для отмены заказа пользователя.

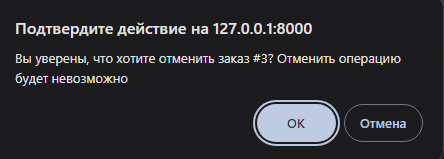


Рисунок 23 – Интерфейс для отмены заказа пользователя

2. ПРОЕКТНАЯ ЧАСТЬ

2.1 Оптимизация веб-приложения

Для максимальной оптимизации функционирования веб-приложения были предприняты несколько важных шагов, направленные на улучшение производительности и удобства пользователей.

В первую очередь, для обеспечения быстрой и стабильной работы сайта, является выбор хостинга. Был выбран хостинг высокого уровня доступности, доступный по адресу <https://www.reg.ru/hosting/>. Reg.ru обеспечивает быстрый доступ к ресурсам и минимизацию времени отклика веб-приложения. Под капотом данного хостинга лежит много интересных инструментов, а главное — это панель управления Ispmanager для лёгкого и быстрого администрирования.

Следующий важный аспект при оптимизации веб-приложения – это изображения в проекте. Лучшим вариантом для современных приложений является хранить изображения в WEBP формате заместо JPG, PNG, GIF. Также для использования векторных изображений должен использоваться формат SVG. Данное действие позволяет выдать быструю загрузку приложения для пользователей с различной скоростью интернета.

Также была проведена большая проверка по наличию повторяющихся или неработающих ссылок в приложении. Данный аспект предотвращает задержки ответов от сервера, улучшает производительность веб-приложения.

Все стили приложения были написаны на препроцессоре SCSS, давая на выходе обычный CSS-файл, в котором лежат свойства со всеми требующемся параметрами под различные браузеры (Safari, Microsoft Edge, Firefox и т.д.). Это позволяет без опасений отображать один и тот же контент в разных браузерах с своей спецификой движка. Также была проделана работа с пользовательским опытом, где проработаны различные ховер и фокус эффекты, и адаптивность под различные устройства пользователей.

Дополнительно были прописаны файлы robots.txt и sitemap.xml для более ясного и эффективного индексирования сайта поисковыми роботами. Эти меры содействуют повышению видимости веб-ресурса в поисковых системах и способствуют успешному продвижению в поисковых результатах.

В файле robots.txt содержится информация о том, какие разделы и папки нужно индексировать поисковым роботам, а также была указана ссылка на sitemap.xml. Это позволяет профессионально обрабатывать информацию поисковым роботам, соответствую текущим требованиям по продвижению для сайтов.

Были созданы свои уникальные страницы для 403, 404 и 503 ошибки в случае различных конфликтов у пользователей, случающиеся при работе с интерфейсом приложения.

Дополнительно, была проведена работа по разделению ключевых слов на группы в соответствии с их значением, интересами потенциальных клиентов и схожестью поисковой выдачи по запросам. Это стратегическое подход к организации ключевых слов содействует более точному и эффективному продвижению сайта в поисковых системах.

Заголовок и описание на каждой странице приложения были подробно описаны. Это оптимизирует отображение в поисковых результатах, делая информацию более привлекательной и информативной для пользователей.

Проведена проверка контента, направленная на формирование и увеличение доверия со стороны потенциального клиента. Это включает в себя анализ текстов, изображений и других элементов контента с целью обеспечения информативности и релевантности.

2.2 Тестирование веб-приложения

После завершения разработки кода, дизайн, и интеграции всех компонентов приложения, наступает этап тестирования, на котором проверяется работоспособность, корректность и соответствие функционала веб-приложения заявленным требованиям и ожиданиям пользователей.

Тестирование веб-приложения включает в себя различные виды тестирования, такие как функциональное тестирование, тестирование пользовательского интерфейса, тестирование производительности, тестирование безопасности и другие. Каждый вид тестирования направлен на проверку определенных аспектов функциональности и качества веб-приложения.

В случае учебной практики будет использоваться функциональное тестирование и тестирование пользовательского интерфейса (UI). Будет применено 35 тест-кейсов, которые проверят работоспособность информационной системы. Суть данного тестирования заключается в проверке работы всех обязательных функций сайта и его корректное отображение во всех актуальных браузерах.

Таблица 10 - Тест-кейсы, пройденные в ходе тестирования системы

|  |  |  |  |
| --- | --- | --- | --- |
| № Тест-кейса | Название проверки | Результат | Замечание и рекомендации |
| 1 | 2 | 3 | 4 |
| Кроссбраузерное тестирование | | | |
| 1 | Приложение корректно отображается и функционирует во всех актуальных браузерах (Brave, Microsoft Edge, Chrome, Firefox, Safari, Opera, Яндекс Браузер) | Положительно |  |
| 2 | Приложение корректно отображается и функционирует во всех актуальных версиях браузеров | Положительно |  |
| 3 | Корректная обработка тайм-аутов при работе с интерфейсом | Положительно |  |
| 4 | Работоспособность при удалении cookies во время использования приложения | Положительно |  |
| 5 | Работоспособность при удалении cookies после использования приложения | Положительно |  |
| Работа с формами | | | |
| 6 | Обязательные поля должны быть заполнены до подтверждения ввода данных. | Положительно |  |
| 7 | Обязательные поля должны быть явно обозначены. | Положительно |  |
| 8 | Если пользователь нажал кнопку подтверждения ввода данных с незаполненными обязательными полями, эти поля должны быть подсвечены и отображено сообщение об ошибке | Положительно |  |
| Продолжение таблицы 10 | | | |
| 1 | 2 | 3 | 4 |
| 9 | Сообщение об ошибке не отображается, если не заполнены необязательные поля | Положительно |  |
| 10 | Текстовые поля допускают ввод специальных символов | Положительно |  |
| 11 | При вводе количества знаков меньше минимального отображается сообщение | Положительно |  |
| 12 | При вводе количества знаков больше максимального отображается сообщение об ошибке | Положительно |  |
| 13 | Если обязательное поле заполнено не полностью, отображается сообщение об ошибке | Отрицательно | Доработать валидацию для полей |
| 14 | Сообщение об ошибке отображается при попытке ввести недопустимые символы (например, ввод текста в поле, принимающее только числа) | Положительно |  |
| 15 | Нельзя отменить повторно отменённую операцию | Положительно |  |
| 16 | Пополнять баланс нельзя на отрицательную сумму | Положительно |  |
| Авторизация и работа с профилем | | | |
| 17 | Создайте новый аккаунт | Положительно |  |
| 18 | Авторизуйтесь с существующим аккаунтом | Положительно |  |
| 19 | При вводе неверного пароля отображается сообщение об ошибке и ссылка на страницу восстановления пароля | Отрицательно | Дать возможность пользователю восстановить пароль |
| 20 | Отредактируйте профиль пользователя | Положительно |  |
| 21 | Загрузите в профиль аватар/картинку | Положительно |  |
| 22 | Отредактируйте аватар/картинку | Положительно |  |
| 23 | При выполнении действий (например, выход из аккаунта) и манипуляциях с данными (например, удаление файла) отображается сообщение с подтверждением | Отрицательно | Дополнить регистрацию аккаунта подтверждением по почте |
| 24 | При смене пароля пользователю приходит подтверждение по электронной почте | Отрицательно | Дополнить безопасность смены пароли на серверной части |
| Навигация | | | |
| 25 | Корректность работы кнопок и ссылок в меню навигации | Положительно |  |
| Продолжение таблицы 10 | | | |
| 1 | 2 | 3 | 4 |
| 26 | Отображение страницы с ошибкой 404, если пользователь указал неверный путь к файлу или URL | Положительно |  |
| 27 | Отображение страницы с ошибкой 403, если пользователь указал зашёл на запрещенный ему ресурс | Положительно |  |
| 28 | Отображение страницы с ошибкой 503, если ответ от сервера клиенту не пришёл (истекло время ожидания) | Положительно |  |
| 29 | Числовые поля не допускают ввода текста, отображается сообщение об ошибке | Положительно |  |
| 30 | При попытке ввода отрицательных чисел, отображается сообщение об ошибке, если предусмотрен ввод строго положительных чисел | Положительно |  |
| 31 | Поля специальных форматов (сумма денег, даты) отображаются корректно | Положительно |  |
| 32 | На странице для пользователей с ролью ADMIN отображается панель администрирования | Положительно |  |
| 33 | На странице для пользователей с ролью USER не отображаются элементы администрирования сайта | Положительно |  |

В данном разделе представлены выводы по результатам испытаний, а также общая статистика по дефектам, которые были выявлены в результате тестирования информационной системы. В период испытаний был выявлено 4 дефекта из 33 тест-кейсов.

Рисунок N – Диаграмма результатов выполнения тест-кейсов

2.3 Руководство пользователя

В приложении имеется отличный функционал для пользователя – это возможность управлять своими транзакциями (просмотр, пополнение баланса) и управлять своими заданиями (просмотр всех заданий и добавление нового задания). Функционал может иметь только авторизованный пользователь, а также пользователь с ролью USER или ADMIN.

Чтобы пользователь мог работать с своим балансом, то нужно перейти из навигации панели сайта в раздел «Пополнение», где будет предоставлен следующий интерфейс как на рисунке N.

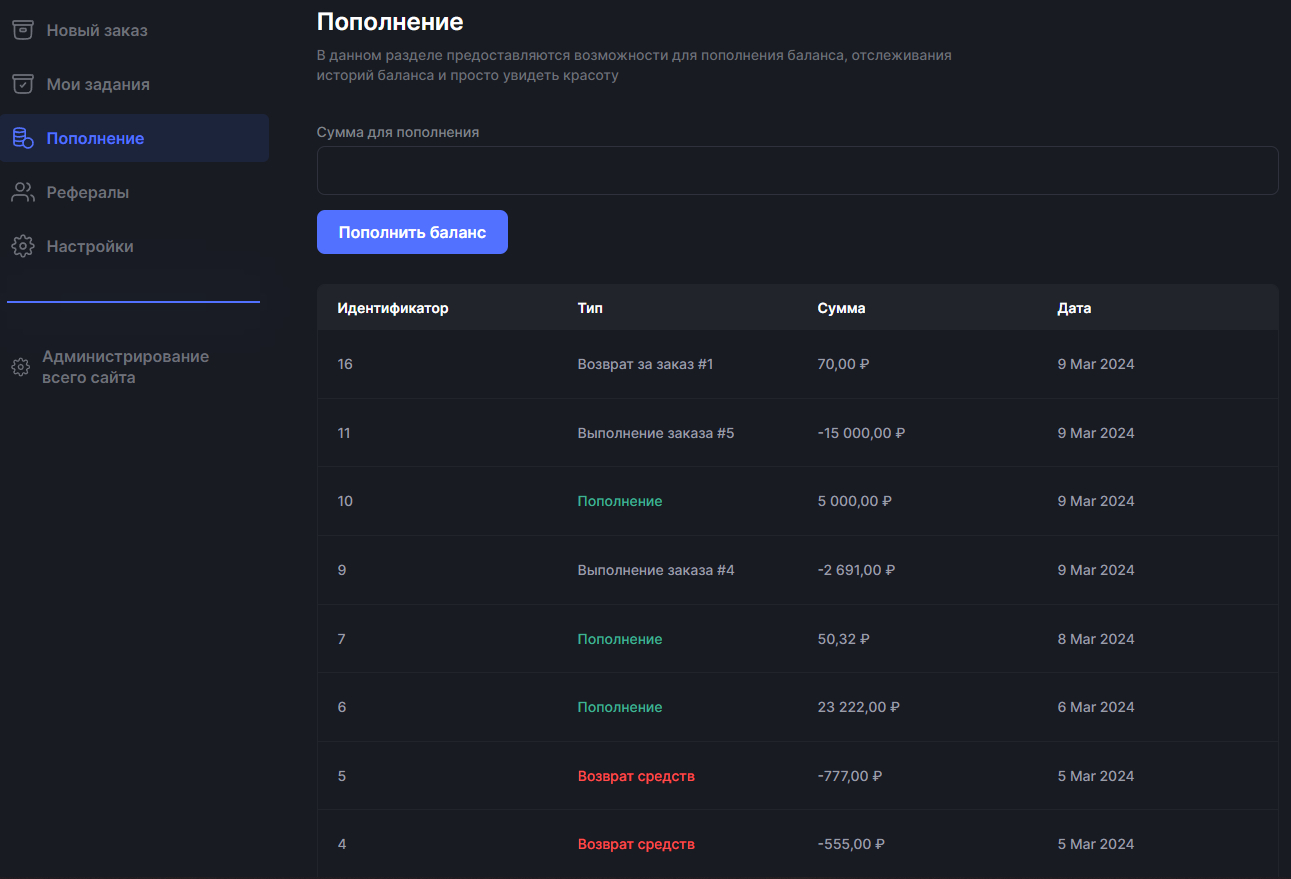


Рисунок 22 – Интерфейс раздела «Пополнение»

На данной странице можно увидеть все транзакции, которые имеют разный тип операции. Здесь же пользователь может легко пополнить свои средства через простую форму, вбив желаемую сумму (минимум 50 рублей, максимум 30000) в поле. После успешной отправки запроса на сервер пользователю придёт ответ:

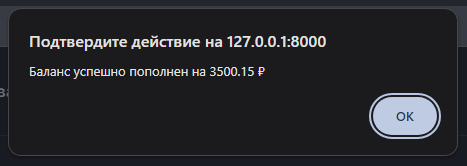


Рисунок N – Ответ сервера после успешного пополнения баланса

После того как пользователь имеет какой-то баланс на своём счету он может создать новое задание по накрутке определённой активности. Перейдя в раздел «Новый заказ», пользователь встретит удобный и красивый интерфейс как на рисунке N.

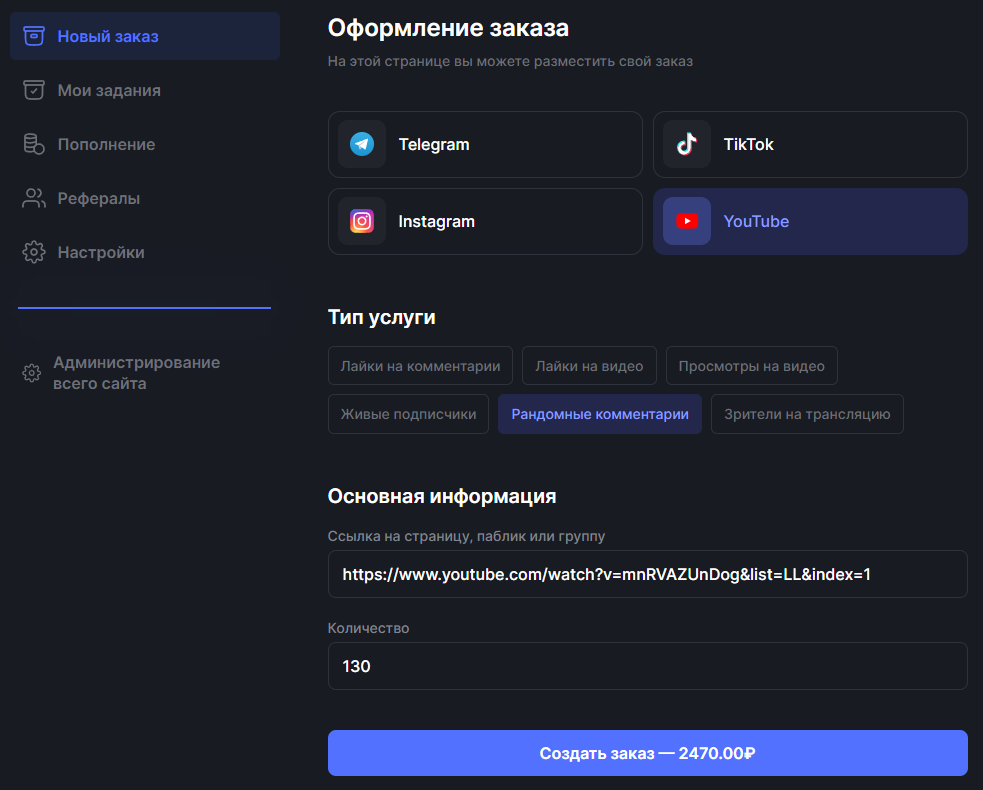


Рисунок N – Интерфейс раздела «Новый заказ»

На данной странице пользователю предоставляется самый главный функционал информационной системы – оформление заказа. Оформлением заказа является формирование задания, в котором будет выбран определённый сервис, тип услуги, и вписана основная информация. Пользователь может выбрать любой доступный сервис и любую доступную услугу накрутки, которая ему будет необходима. Дальше пользователю нужно вбить основную информацию – количество (от 50 до 13000) и адрес какой-либо страницы в какой-то социальной сети (по которому в дальнейшем будет исполняться заказ для пользователя). После этого всего в кнопке «Создать заказ» пользователь увидит итоговую стоимость заказа. Если пользователь нажмёт на кнопку, то запрос придёт на сервер. Если у пользователя недостаточно средств на балансе, то ему покажет сообщение от сервера о необходимости пополнении баланса, чтобы в дальнейшем оформить заказ.

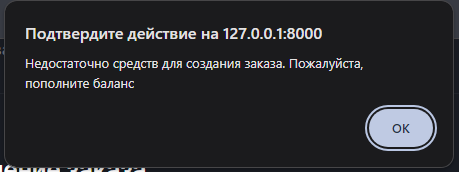


Рисунок N – Ответ от сервера о нехватке баланса при оформлении заказа

В случае если у пользователя хватило средств на создание заказа его автоматически переведёт в раздел «Мои задания». В данном разделе пользователь сразу увидит все свои созданные задания, а также в самом верху отобразится последнее созданное им задание. С помощью этой страницы пользователь может отслеживать статусы своих заказов, а также идентификаторы заказы, чтобы в случае неправильного оформления заказа пользователь мог легко обратиться в техподдержку сайта, потребовав отмены заказа.

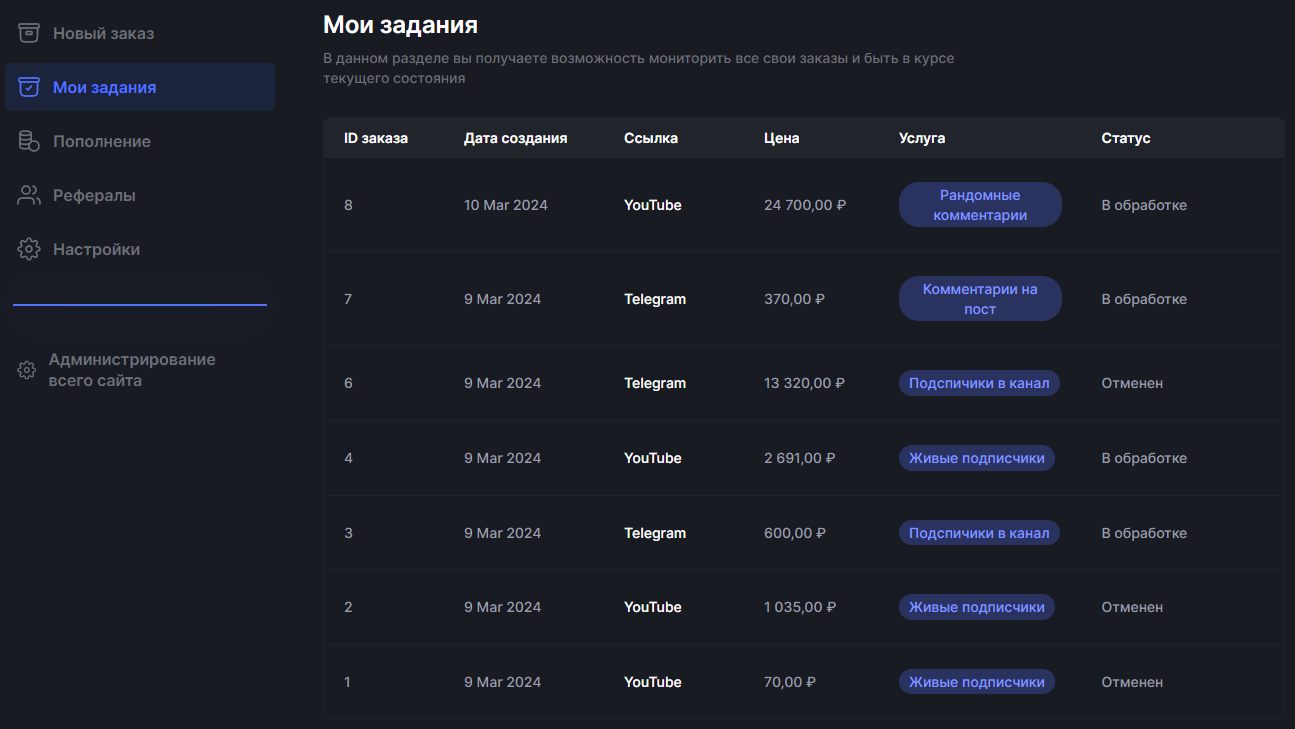


Рисунок N – Интерфейс раздела «Мои задания»

2.4 Руководство администратора

В информационной системе имеется функционал отдельно для администратора. Все действия, ссылки для администрирования начинаются с префикса «admin» в URL адресе. Добавлять информацию можно в следующие таблицы – пользователи, сервисы и типы услуг к сервисам.

Под каждую категорию работы с информационной системы созданы отдельные страницы и разделы. Чтобы двигаться по разделам была создана удобная фиксированная навигация, дав быстрый переход между страницами администратору.

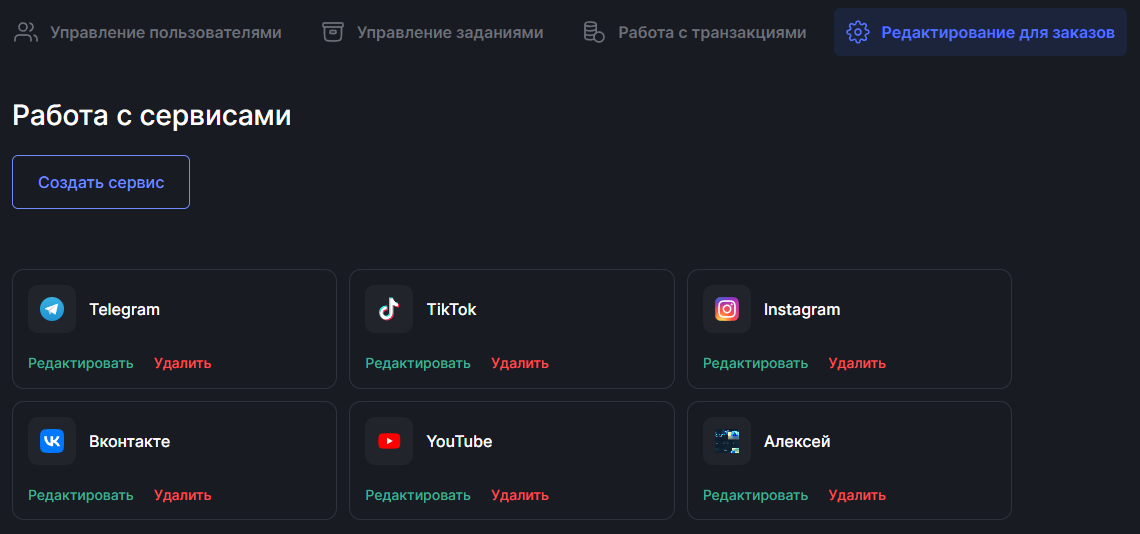


Рисунок N – Интерфейс раздела администрирования с навигацией

В интерфейсе администрирования можно заниматься добавлением элементов сущностей. Чтобы вызвать окно или страницу добавления достаточно будет нажать кнопку «Создать сервис» в определенном интерфейсе системы. Когда была нажата кнопка, то откроется интерфейс, в котором будут доступны поля для заполнения новыми данными.

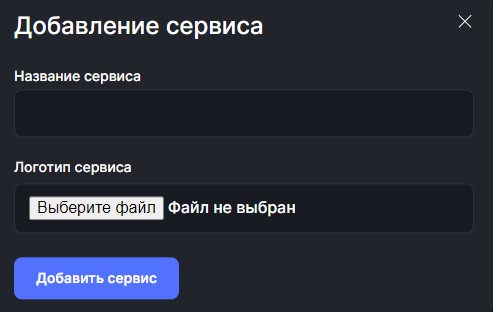


Рисунок N – Руководство администратора для добавления данных

Дальше идет заполнение данных администратором. Но стоит учитывать, что если были введены некорректные данные, то в итоге будут выведены свои ошибки для каждого поля, в которых будет описано что не так заполнено в поле. После того как будет нажата кнопка «Добавить» в модальном окне, то после успешного создания, созданный элемент отобразится в интерфейсе без перезагрузки страницы.

В информационной системе имеется функция редактирования данных. Редактировать информацию можно в следующих таблицах – сервисы, типы услуг к сервисам и пользователи. Чтобы вызвать интерфейс редактирования нужно сначала определиться какой элемент сущности будет изменяться.

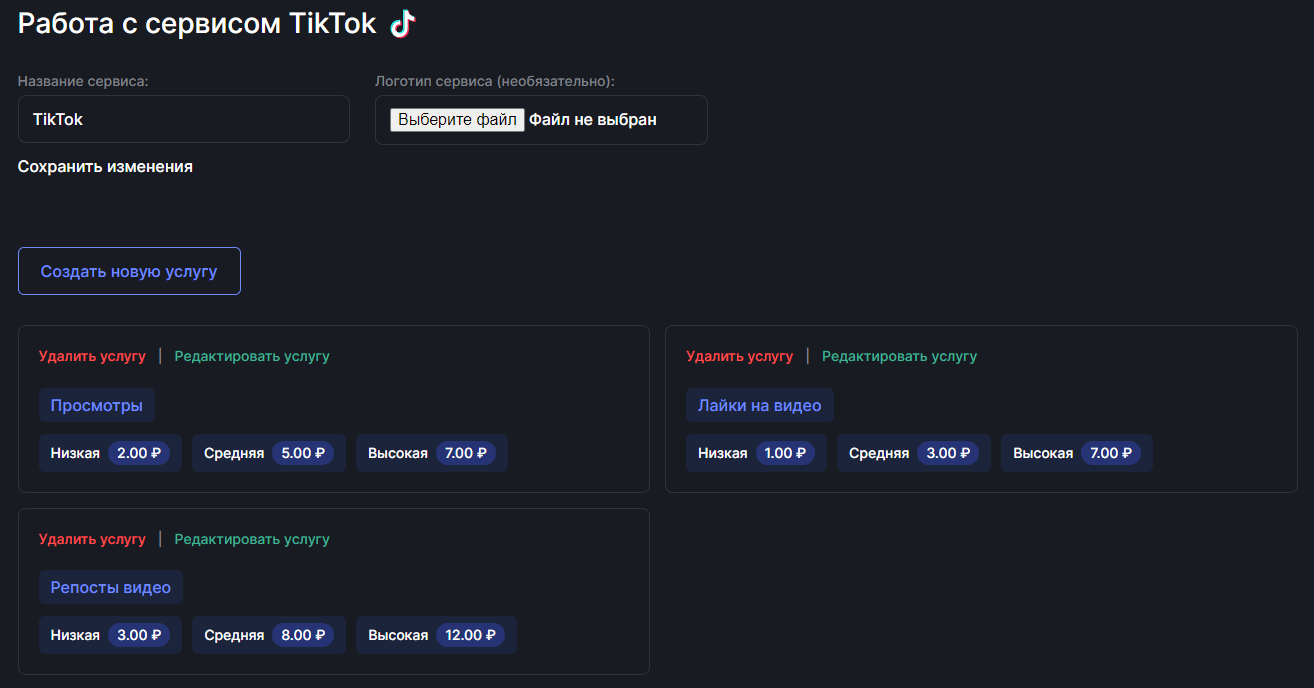


Рисунок N – Руководство администратора для редактирования данных

Когда мы нажали на кнопку, то откроется сам интерфейс редактирования в модальном окне или же на отдельной странице редактирования, и будут заполнены поля значениями, взятыми из определенной записи.

Далее идет заполнение новыми данными под старые. Стоит учитывать, что если были введены некорректные данные, то в итоге будут выведены свои ошибки для каждого поля, в которых будет описано что не так заполнено в поле, и потребуется изменить данные чтобы они стали корректными, иначе данные не отправятся на сервер.

После того как была нажата кнопка «Сохранить», изменения вступят в силу. Администратор на клиенте увидит перенаправление на другую страницу (в случае успеха) или же обновление данных без перезагрузок (в зависимости от того, что именно было изменено). Далее запись, которая изменилась, обновится в интерфейсе системы.

В информационной системе имеется функция удаления существующих данных. Удалять информацию можно из следующих таблицах – пользователи, сервисы, и типы услуг к сервисам. Чтобы вызвать удаление достаточно будет нажать на кнопку удаления у определенной записи, и откроется всплывающее окно с подтверждением удаления данных.

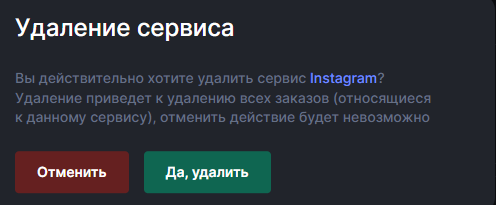


Рисунок N – Руководство администратора для удаления данных

Если была нажата кнопка «Да, удалить», то удалится запись. После этого обновится список данных в интерфейсе системы, где удаленная запись не будет уже отображаться.

Если же была нажата кнопка «Отменить», то запись не будет удалена из базы данных, и закроется всплывающее окно подтверждения удаления записи.

В информационной системе у администратора есть возможность отменять некоторые операции в информационной системе. На данный момент имеется возможность отменять транзакции с типом «Пополнение» в разделе «Работа с транзакциями», а также в разделе «Управление заданиями» можно отменять созданные задания пользователем, нажав на кнопку «Отменить задание».

Отличие данного действия от удаления заключается в том, что администратор не удаляет полностью данные из БД, программа их фиксирует в БД дальше, создав при этом поверх старой записи новую запись, в которой, например будет указана другая дата выполнения операции и статус операции.

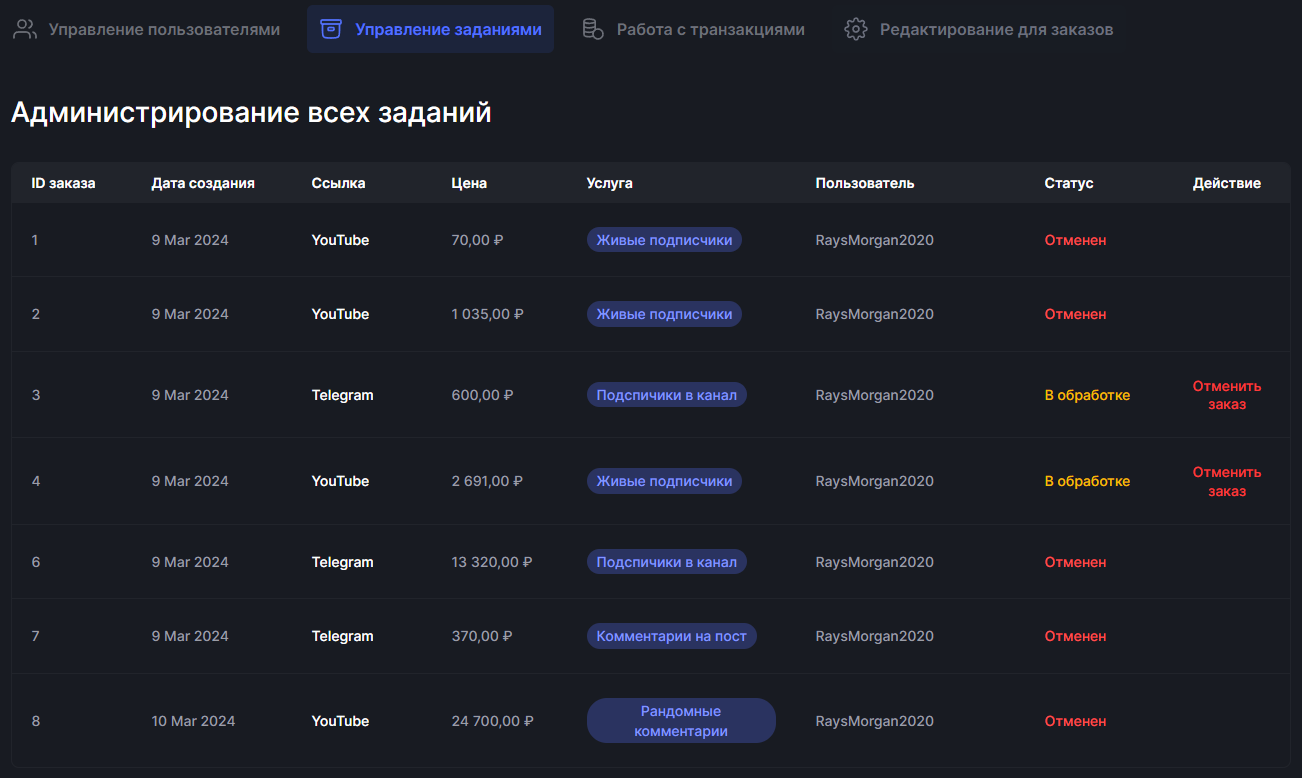


Рисунок N – Руководства администратора для отмены операций

Когда будет нажиматься кнопка отмены какой-либо операции то будет всплывать диалоговое окно, в котором запросится подтверждение на данное выполнение действия. Если администратор нажмёт «Ок», то операция будет выполнена успешно и изменения вступят в силу. В интерфейсе обновится страница и добавится новая запись с новой датой создания, идентификатором, а также статусом операции.

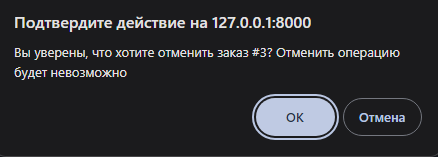


Рисунок N – Руководства администратора для отмены операции

Статус операции будет отличаться от первоначальной операции. Повторно отменять отменённые операции администратор не сможет, кнопка для таких записи будет скрыта автоматически.

ЗАКЛЮЧЕНИЕ

Разработка сервиса по продвижению социальных сетей на базе фреймворка PHP Laravel с шаблонизатором Blade позволила создать функциональный и эстетичное приложение. Используя современные инструменты разработки, включая фреймворк Laravel, достигнута гибкость и масштабируемость приложения. Целью проекта было создание удобного интерфейса для взаимодействия с сервисом по накрутке активностей в социальных сетях, обеспечивая пользователям уникальный и легкий опыт в создании задания в пару кликов.

Использование фреймворка Laravel значительно облегчило расширение функциональности сервиса с возможным будущим функционалом. Масштабируемость этого фреймворка позволила без проблем внедрять новые возможности и обрабатывать увеличивающееся количество запросов.

Одной из ключевых задач было обеспечение безопасности, интуитивной навигации и удобства в управлении контентом для администраторов. Используя Blade шаблоны в Laravel, удалось создать интуитивно понятный интерфейс для администраторов и пользователей, обеспечив безопасную и удобную среду для работы с данными о клиентах, транзакциях, сервисов, типами услуг конкретного сервиса и самих созданных заданий от пользователей.

Инфраструктура на основе PHP Laravel и шаблонизатора Blade позволила быстро реализовать функционал добавления, редактирования и удаления сущностей, а также создать эффективную систему создания заданий, что значительно облегчило процесс пользовательского опыта в информационной системе.

Задачи, описанные в начале исследовательской работе решены, а именно создать трёхуровневую архитектуру приложения, быстрый доступ к базе данных, добавлению различных пользовательских заданий, просмотра, редактирования и удаления любых транзакций, пользователей, а также управление другим контентом на сайте.

В результате выполнения учебной практики, был создана динамическая информационная система для внесения и хранения данных о заданиях, услугах накрутки, пользователях и транзакциях пользователей. Для его разработки применялось актуальное программное обеспечение, без помощи которого не было бы возможности добиться поставленной цели. Это программное обеспечение включает в себя:

* язык программирования PHP;
* коммерческая кроссплатформенная интегрированная среда разработки для PHP –PhpStorm;
* фреймворк PHP Laravel;
* веб-сервер XAMPP;
* реляционная система управления БД MySQL;
* приложение для администрирования СУБД MySQL PhpMyAdmin;

Этот проект стал хорошим примером использования современных технологий разработки в области сервисов по платной накрутке социальных сетей, обеспечивая пользователей качественным и интуитивно понятным интерфейсом, а администраторов — легким управлением контентом. Всё это было возможно благодаря гибкости и функциональности PHP Laravel и Blade шаблонизатора, которые позволили создать удивительную информационную систему, удовлетворяющий потребности как пользователей, так и администраторов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вандюк, Джон К. CMS Drupal. Руководство по разработке системы управления сайтом / Вандюк, Джон К., Мэтт Вестгейт, - М.: Вильямс, 2019. - 400 c.
2. Гаевский, А.Ю. 100% самоучитель. Создание Web-страниц и Web-сайтов. HTML и JavaScript / А.Ю. Гаевский, В.А. Романовский. - М.: Триумф, 2020. - 464 c.
3. Дакетт, Джон HTML и CSS. Разработка и дизайн веб-сайтов / Джон Дакетт. - Москва: Наука, 2019. - 480 c.
4. Дронов, В. JavaScript в Web-дизайне / В. Дронов. - М.: СПб: БХВ, 2019. - 880 c.
5. Дронов, В. PHP, MySQL и Dreamweaver. Разработка интерактивных Web-сайтов / В. Дронов. - М.: БХВ-Петербург, 2019. - 480 c.
6. Дронов, В.А. PHP 5/6, MySQL 5/6 и Dreamweaver CS4. Разработка интерактивных Web-сайтов / В. А. Дронов. - М.: БХВ-Петербург, 2019. - 820 c.
7. Дронов, Владимир JavaScript и AJAX в Web-дизайне / Владимир Дронов. - Москва: Высшая школа, 2019. - 736 c.
8. Дронов, Владимир PHP, MySQL и Dreamweaver MX 2004. Разработка интерактивных Web-сайтов / Владимир Дронов. - М.: "БХВ-Петербург", 2019. - 448 c.
9. Колисниченко, Д. PHP 5/6 и MySQL 6. Разработка Web-приложений / Д. Колисниченко. - М.: БХВ-Петербург, 2021. - 560 c.
10. Колисниченко, Д.Н. PHP 5/6 и MySQL 6. Разработка Web-приложений / Д.Н. Колисниченко. - М.: БХВ-Петербург, 2021. - 364 c.
11. Кузнецов PHP. Практика создания Web-сайтов / Кузнецов, М.В. и. - М.: БХВ-Петербург, 2020. - 895 c.
12. Кузнецов, М. PHP 5. Практика создания Web-сайтов / М. Кузнецов, И. Симдянов, С. Голышев. - М.: БХВ-Петербург, 2020. - 960 c.
13. Леонтьев Борис PHP 5.0 для начинающих, или как создать динамический WEB-сайт / Борис Леонтьев. - М.: Новый издательский дом, 2019. - 176 c.
14. Петюшкин Алексей HTML в Web-дизайне / Алексей Петюшкин. - Москва: Машиностроение, 20. - 400 c.
15. Шкрыль, А. PHP — это просто. Программируем для Web-сайта / А. Шкрыль. - М.: БХВ-Петербург, 2021. – 360 с.
16. ГОСТ 19.201-78 «Техническое задание».
17. ГОСТ 19.504-79 ЕСПД «Руководства программиста»
18. Дакетт Д. – HTML и CSS. Разработка и дизайн веб-сайтов, Эксмо, 2019, 480 с.
19. Дронов В. – Laravel 8. Быстрая разработка веб-сайтов на PHP, БХВ-Петербург, 2020 688 с.

ПРИЛОЖЕНИЕ

Kernel.php – ядро приложения, который выполняет роль запуска

<?php  
  
namespace App\Http;  
  
use App\Http\Middleware\AdminMiddleware;  
use App\Http\Middleware\Authenticate;  
use App\Http\Middleware\CheckDatabaseConnection;  
use Illuminate\Foundation\Http\Kernel as HttpKernel;  
  
class Kernel extends HttpKernel  
{  
 protected $routeMiddleware = [  
 'admin' => AdminMiddleware::class,  
 'checkdb' => CheckDatabaseConnection::class,  
 'auth' => Authenticate::class,  
 ];  
  
 protected $middleware = [  
 // \App\Http\Middleware\TrustHosts::class,  
 \App\Http\Middleware\TrustProxies::class,  
 \Illuminate\Http\Middleware\HandleCors::class,  
 \App\Http\Middleware\PreventRequestsDuringMaintenance::class,  
 \Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,  
 \App\Http\Middleware\TrimStrings::class,  
 \Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,  
 ];  
  
 protected $middlewareGroups = [  
 'web' => [  
 \App\Http\Middleware\EncryptCookies::class,  
 \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,  
 \Illuminate\Session\Middleware\StartSession::class,  
 \Illuminate\View\Middleware\ShareErrorsFromSession::class,  
 \App\Http\Middleware\VerifyCsrfToken::class,  
 \Illuminate\Routing\Middleware\SubstituteBindings::class,  
 ],  
  
 'api' => [  
 // \Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,  
 \Illuminate\Routing\Middleware\ThrottleRequests::class . ':api',  
 \Illuminate\Routing\Middleware\SubstituteBindings::class,  
 ],  
 ];  
  
 protected $middlewareAliases = [  
 'auth' => Authenticate::class,  
 'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,  
 'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,  
 'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,  
 'can' => \Illuminate\Auth\Middleware\Authorize::class,  
 'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,  
 'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,  
 'precognitive' => \Illuminate\Foundation\Http\Middleware\HandlePrecognitiveRequests::class,  
 'signed' => \App\Http\Middleware\ValidateSignature::class,  
 'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,  
 'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,  
 ];  
}

StoreUserRequest.php – отвечает за хранение запросов для пользователя

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class StoreUserRequest extends FormRequest

{

    public function authorize(): bool

    {

        return auth()->user()->role === 'ADMIN';

    }

    public function rules(): array

    {

        return [

            'login' => 'required|string|min:5|max:60|unique:users',

            'name' => 'required|string|min:2|max:50',

            'email' => 'required|email|unique:users,email',

            'password' => 'required|string|min:8|max:60',

            'role' => 'required|in:ADMIN,USER',

        ];

    }

}

UpdateUserRequest.php – отвечает за обновление запросов пользователя

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class UpdateUserRequest extends FormRequest

{

    public function authorize(): bool

    {

        return auth()->user()->role === 'ADMIN';

    }

    public function rules(): array

    {

        return [

            'login' => 'required|string|min:5|max:60|unique:users,login,' . $this->user->id,

            'name' => 'required|string|min:2|max:50',

            'email' => 'required|email|unique:users,email,' . $this->user->id,

            'password' => 'nullable|string|min:8|max:60',

            'role' => 'required|in:ADMIN,USER',

        ];

    }

}

AdminMiddleware.php – фильтр обработки HTTP-запроса для администраторов проекта

<?php  
namespace App\Http\Middleware;  
use Closure;  
use Illuminate\Http\Request;  
class AdminMiddleware  
{  
 public function handle(Request $request, Closure $next)  
 {  
 if (!$request->user() || $request->user()->role !== 'ADMIN') {  
 return redirect('/403\_error');  
 }  
 return $next($request);  
 }  
}

CheckDatabaseConnection.php – фильтр обработки HTTP-запроса для соединения с БД

<?php  
namespace App\Http\Middleware;  
use Closure;  
use Illuminate\Database\QueryException;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\DB;  
use Symfony\Component\HttpFoundation\Response;  
class CheckDatabaseConnection  
{  
 public function handle(Request $request, Closure $next): Response  
 {  
 try {  
 DB::connection()->getPdo();  
 } catch (\Exception) {  
 return redirect('/503\_error');  
 }  
 return $next($request);  
 }  
}

RouteServiceProvider.php – загрузка всех маршрутов данного приложения

<?php

namespace App\Providers;

use Illuminate\Cache\RateLimiting\Limit;

use Illuminate\Foundation\Support\Providers\RouteServiceProvider as ServiceProvider;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\RateLimiter;

use Illuminate\Support\Facades\Route;

class RouteServiceProvider extends ServiceProvider

{

    public const HOME = '/home';

    public function boot(): void

    {

        RateLimiter::for('api', function (Request $request) {

            return Limit::perMinute(60)->by($request->user()?->id ?: $request->ip());

        });

        $this->routes(function () {

            Route::middleware('api')

                ->prefix('api')

                ->group(base\_path('routes/api.php'));

            Route::middleware('web')

                ->group(base\_path('routes/web.php'));

            Route::middleware('web')

                ->group(base\_path('routes/home.php'));

            Route::middleware('web')

                ->group(base\_path('routes/auth/auth.php'));

            Route::middleware('web')

                ->group(base\_path('routes/admin/category.php'));

            Route::middleware('web')

                ->group(base\_path('routes/admin/products.php'));

            Route::middleware('web')

                ->group(base\_path('routes/reviews.php'));

            Route::middleware('web')

                ->group(base\_path('routes/admin/reviews.php'));

            Route::middleware('web')

                ->group(base\_path('routes/admin/users.php'));

        });

    }

}

User.php – сущность пользователя

<?php  
namespace App\Models;  
use Illuminate\Contracts\Auth\MustVerifyEmail;  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Relations\HasOne;  
use Illuminate\Foundation\Auth\User as Authenticatable;  
use Illuminate\Notifications\Notifiable;  
  
class User extends Authenticatable implements MustVerifyEmail  
{  
 use HasFactory, Notifiable;  
 protected $fillable = [  
 'login',  
 'name',  
 'email',  
 'balance',  
 'password',  
 'role',  
 'avatar'  
 ];  
 public function balance(): HasOne  
 {  
 return $this->hasOne(UserBalance::class);  
 }  
}

UserBalance.php – сущность баланса пользователя

<?php  
namespace App\Models;  
use Illuminate\Database\Eloquent\Model;  
use Illuminate\Database\Eloquent\Relations\BelongsTo;  
class UserBalance extends Model  
{  
 protected $fillable = [  
 'user\_id',  
 'balance'  
 ];  
 public function user(): BelongsTo  
 {  
 return $this->belongsTo(User::class);  
 }  
 public function getBalance(): float  
 {  
 return $this->balance;  
 }  
}

BalanceTransaction.php – сущность транзакций системы

<?php  
namespace App\Models;  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
use Illuminate\Database\Eloquent\Relations\BelongsTo;  
class BalanceTransaction extends Model  
{  
 use HasFactory;  
 protected $fillable = [  
 'user\_id',  
 'amount',  
 'cancelled',  
 'type'  
 ];  
 public function user(): BelongsTo  
 {  
 return $this->belongsTo(User::class);  
 }  
}

Service.php – сущность сервиса

<?php  
namespace App\Models;  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
use Illuminate\Database\Eloquent\Relations\HasMany;  
class Service extends Model  
{  
 use HasFactory;  
 protected $fillable = ['name', 'logo'];  
 public function serviceItems(): HasMany  
 {  
 return $this->hasMany(ServiceItem::class);  
 }  
 public function orders(): HasMany  
 {  
 return $this->hasMany(Order::class);  
 }  
}

ServiceItem.php – сущность услуг сервиса

<?php  
namespace App\Models;  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
use Illuminate\Database\Eloquent\Relations\BelongsTo;  
use Illuminate\Database\Eloquent\Relations\HasMany;  
class ServiceItem extends Model  
{  
 use HasFactory;  
 protected $fillable = ['service\_id', 'name', 'price\_low', 'price\_medium', 'price\_high'];  
 public function service(): BelongsTo  
 {  
 return $this->belongsTo(Service::class);  
 }  
  
 public function orders(): HasMany  
 {  
 return $this->hasMany(Order::class);  
 }  
}

Order.php – сущность заказов (заданий)

<?php  
namespace App\Models;  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
  
class Order extends Model  
{  
 use HasFactory;  
 protected $fillable = [  
 'user\_id',  
 'service\_id',  
 'service\_item\_id',  
 'source\_link',  
 'quantity',  
 'total\_price',  
 'status',  
 ];  
 protected $casts = [  
 'quantity' => 'integer',  
 'total\_price' => 'decimal:2',  
 ];  
 public function user()  
 {  
 return $this->belongsTo(User::class);  
 }  
 public function service()  
 {  
 return $this->belongsTo(Service::class);  
 }  
 public function serviceItem()  
 {  
 return $this->belongsTo(ServiceItem::class);  
 }  
}

UserController.php – контроллер для управления пользователями

<?php  
namespace App\Http\Controllers\Admin;  
use App\Http\Controllers\Controller;  
use App\Http\Requests\StoreUserRequest;  
use App\Http\Requests\UpdateUserRequest;  
use App\Models\User;  
use Illuminate\Contracts\View\Factory;  
use Illuminate\Contracts\View\View;  
use Illuminate\Foundation\Application;  
use Illuminate\Http\JsonResponse;  
use Illuminate\Http\RedirectResponse;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Hash;  
  
class UserController extends Controller  
{  
 public function index(): View|Application|Factory|\Illuminate\Contracts\Foundation\Application  
 {  
 $users = User::query();  
 $roles = User::distinct()->pluck('role');  
 $users = $users->get();  
 return view('admin.manage\_users', compact('users', 'roles'));  
 }  
 public function getAllUsers(): JsonResponse  
 {  
 $users = User::all();  
 return response()->json(['users' => $users]);  
 }  
 public function search(Request $request): JsonResponse  
 {  
 $query = $request->input('query');  
 $usersQuery = User::query()  
 ->where('login', 'like', "%$query%")  
 ->orWhere('name', 'like', "%$query%")  
 ->orWhere('email', 'like', "%$query%");  
 $users = $query ? $usersQuery->get() : collect([]);  
 return response()->json(['users' => $users]);  
 }  
 public function store(StoreUserRequest $request): JsonResponse  
 {  
 $validatedData = $request->validate([  
 'login' => 'required|string|min:5|max:60|unique:users|regex:/^[a-zA-Z0-9\_]+$/',  
 'name' => 'required|string|min:2|max:50|regex:/^[A-Za-zА-Яа-яЁё\-]+$/u',  
 'email' => 'required|email|unique:users,email',  
 'password' => 'required|string|min:8|max:60',  
 ]);  
 $validatedData['password'] = Hash::make($validatedData['password']);  
 $validatedData['role'] = $request->input('role', 'USER');  
 $user = User::create($validatedData);  
 return response()->json(['message' => 'Пользователь успешно добавлен', 'user' => $user]);  
 }  
 public function edit(User $user): View|Factory|Application  
 {  
 return view('admin.edit\_user', compact('user'));  
 }  
 public function update(UpdateUserRequest $request, User $user): RedirectResponse  
 {  
 $validatedData = $request->validated();  
 $user->update($validatedData);  
 return redirect()->route('admin.users.index')->with('success', 'Пользователь успешно обновлен');  
 }  
 public function destroy(User $user): JsonResponse  
 {  
 $user->delete();  
 return response()->json(['message' => 'Пользователь успешно удален']);  
 }  
}

AuthController.php – контроллер для авторизации в системе

<?php  
namespace App\Http\Controllers\Auth;  
use App\Http\Controllers\Controller;  
use Illuminate\Foundation\Application;  
use Illuminate\Http\JsonResponse;  
use Illuminate\Http\RedirectResponse;  
use Illuminate\Http\Request;  
use Illuminate\Routing\Redirector;  
use Illuminate\Support\Facades\Auth;  
class AuthController extends Controller  
{  
 public function login(Request $request): JsonResponse  
 {  
 $credentials = $request->only('login', 'password');  
 if (Auth::attempt($credentials)) {  
 return response()->json(['success' => true]);  
 } else {  
 return response()->json(['success' => false]);  
 }  
 }  
 public function logout(Request $request): Application|Redirector|RedirectResponse|\Illuminate\Contracts\Foundation\Application  
 {  
 Auth::logout();  
 $request->session()->invalidate();  
 $request->session()->regenerateToken();  
 return redirect('/');  
 }  
}

RegisterController.php – контроллер для регистрации в системе

<?php  
namespace App\Http\Controllers\Auth;  
use App\Http\Controllers\Controller;  
use App\Models\User;  
use App\Models\UserBalance;  
use Exception;  
use Illuminate\Contracts\View\Factory;  
use Illuminate\Contracts\View\View;  
use Illuminate\Foundation\Application;  
use Illuminate\Http\RedirectResponse;  
use Illuminate\Http\Request;  
use Illuminate\Routing\Redirector;  
use Illuminate\Support\Facades\Auth;  
use Illuminate\Support\Facades\DB;  
use Illuminate\Support\Facades\Hash;  
class RegisterController extends Controller  
{  
 public function register(Request $request): Application|Redirector|RedirectResponse|\Illuminate\Contracts\Foundation\Application  
 {  
 DB::beginTransaction();  
 try {  
 $validatedData = $request->validate([  
 'login' => 'required|string|min:5|max:60|unique:users|regex:/^[a-zA-Z0-9\_]+$/',  
 'name' => 'required|string|min:2|max:50|regex:/^[A-Za-zА-Яа-яЁё\-]+$/u',  
 'email' => 'required|email|max:80|unique:users,email',  
 'password' => 'required|string|min:8|max:60',  
 ]);  
  
 $user = User::create([  
 'login' => $validatedData['login'],  
 'name' => $validatedData['name'],  
 'email' => $validatedData['email'],  
 'password' => Hash::make($validatedData['password']),  
 'role' => 'USER',  
 ]);  
 UserBalance::create([  
 'user\_id' => $user->id,  
 'balance' => 0  
 ]);  
 Auth::login($user);  
 DB::commit();  
 return redirect('/')->with('success', 'Регистрация прошла успешно!');  
 } catch (Exception) {  
 DB::rollBack();  
 return redirect('/503\_error');  
 }  
 }  
}

CaptchaController.php – контроллер для капчи

<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\JsonResponse;  
use Illuminate\Http\Request;  
class CaptchaController extends Controller  
{  
 public function generateCaptcha(): void  
 {  
 $characters = '0123456789abcdefhjkmnpqrstuvwxyz';  
 $captchaText = substr(str\_shuffle($characters), 0, 7);  
 session(['captcha' => $captchaText]);  
 $image = imagecreatetruecolor(120, 40);  
 $bgColor = imagecolorallocate($image, 255, 255, 255);  
 imagefilledrectangle($image, 0, 0, 120, 40, $bgColor);  
 $textColor = imagecolorallocate($image, 0, 0, 0);  
 $fontPath = public\_path('static/fonts/Inter-Medium.woff');  
 imagettftext($image, 20, 0, 10, 30, $textColor, $fontPath, $captchaText);  
 for ($i = 0; $i < 1000; $i++) {  
 imagesetpixel($image, rand(0, 120), rand(0, 40), $textColor);  
 }  
 imagefilter($image, IMG\_FILTER\_SMOOTH, 5);  
 imagefilter($image, IMG\_FILTER\_CONTRAST, -10);  
 header('Content-type: image/png');  
 imagepng($image);  
 imagedestroy($image);  
 }  
 public function validateCaptcha(Request $request): JsonResponse  
 {  
 $userCaptcha = $request->input('captcha');  
 $captchaText = session('captcha');  
 if ($userCaptcha === $captchaText) {  
 return response()->json(['success' => true]);  
 } else {  
 return response()->json(['success' => false]);  
 }  
 }  
}

BalanceController.php – контроллер для управления балансом в системе

<?php  
namespace App\Http\Controllers;  
use App\Models\BalanceTransaction;  
use App\Models\UserBalance;  
use Exception;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Auth;  
class BalanceController extends Controller  
{  
 public function show()  
 {  
 $user = Auth::user();  
 $transactions = BalanceTransaction::where('user\_id', $user->id)  
 ->orderBy('created\_at', 'desc')  
 ->get();  
  
 return view('panel.balance', ['transactions' => $transactions]);  
 }  
 public function deposit(Request $request)  
 {  
 $user = Auth::user();  
 $amount = $request->input('amount');  
 if ($user && $amount > 0) {  
 $transaction = new BalanceTransaction([  
 'user\_id' => $user->id,  
 'amount' => $amount,  
 'type' => 'Пополнение'  
 ]);  
 $transaction->save();  
 $user->balance()->increment('balance', $amount);  
 return response()->json(['amount' => $amount]);  
 } else {  
 return response()->json(['error' => 'Ошибка при пополнении баланса']);  
 }  
 }  
 public function getAllTransactions()  
 {  
 $transactions = BalanceTransaction::orderBy('created\_at', 'desc')->get();  
 return view('admin.manage\_transactions', ['transactions' => $transactions]);  
 }  
 public function cancelPayment(Request $request, $transactionId)  
 {  
 try {  
 $transaction = BalanceTransaction::findOrFail($transactionId);  
 if ($transaction->cancelled) {  
 return response()->json(['error' => 'Платеж уже отменен.'], 400);  
 }  
 // платеж как отмененный  
 $transaction->update(['cancelled' => true]);  
 // новая запись о возврате средств  
 $refundTransaction = new BalanceTransaction([  
 'user\_id' => $transaction->user\_id,  
 'amount' => $transaction->amount \* -1,  
 'type' => 'Возврат средств',  
 ]);  
 $refundTransaction->save();  
 $userBalance = UserBalance::where('user\_id', $transaction->user\_id)->firstOrFail();  
 $userBalance->balance -= $transaction->amount;  
 $userBalance->save();  
 return response()->json([  
 'message' => 'Платеж успешно отменен.',  
 'refundTransaction' => [  
 'id' => $refundTransaction->id,  
 'user\_id' => $refundTransaction->user\_id,  
 'amount' => $refundTransaction->amount,  
 'type' => $refundTransaction->type,  
 'created\_at' => $refundTransaction->created\_at->format('Y-m-d H:i:s'),  
 'user' => $refundTransaction->user->login  
 ]  
 ]);  
 } catch (Exception) {  
 return response()->json(['error' => 'Произошла ошибка при отмене платежа.'], 500);  
 }  
 }  
}

ProfileController.php – контроллер для просмотра информации о профиле и редактировании

<?php  
namespace App\Http\Controllers;  
use Exception;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Auth;  
use Illuminate\Support\Facades\Hash;  
use Illuminate\Support\Facades\Storage;  
use Illuminate\Support\Str;  
class ProfileController extends Controller  
{  
 public function updateName(Request $request)  
 {  
 $request->validate([  
 'name' => 'required|string|min:2|max:50|regex:/^[A-Za-zА-Яа-яЁё\-]+$/u',  
 ]);  
 $user = Auth::user();  
 $user->name = $request->input('name');  
 $user->save();  
 return response()->json(['success' => 'Имя успешно обновлено']);  
 }  
 public function updateEmail(Request $request)  
 {  
 $request->validate([  
 'email' => 'required|email|max:80|unique:users,email',  
 ]);  
 $user = Auth::user();  
 $user->email = $request->input('email');  
 $user->save();  
 return response()->json(['success' => 'Почта успешно обновлена']);  
 }  
 public function updateAvatar(Request $request)  
 {  
 try {  
 $request->validate([  
 'avatar' => 'required|image|mimes:jpeg,png,jpg,gif,webp|max:2048'  
 ]);  
 $user = Auth::user();  
 if ($user->avatar) {  
 Storage::delete('public/avatars/' . $user->avatar);  
 }  
 $avatarPath = $request->file('avatar')->store('public/avatars');  
 $avatarName = basename($avatarPath);  
 $user->avatar = $avatarName;  
 $user->save();  
 return response()->json(['success' => 'Аватар успешно обновлен']);  
 } catch (Exception $e) {  
 return response()->json(['error' => 'Ошибка при обновлении аватара: ' . $e->getMessage()], 500);  
 }  
 }  
 public function updatePassword(Request $request)  
 {  
 $request->validate([  
 'old\_password' => 'required',  
 'new\_password' => [  
 'required',  
 'string',  
 'min:8',  
 'max:60',  
 'different:old\_password',  
 function ($attribute, $value, $fail) use ($request) {  
 $similar = similar\_text($request->old\_password, $value, $percent);  
 if ($similar >= 80) {  
 $fail('Новый пароль слишком похож на старый');  
 }  
 },  
 ],  
 'repeat\_password' => 'required|same:new\_password',  
 ]);  
 $user = Auth::user();  
 if (!Hash::check($request->old\_password, $user->password)) {  
 return response()->json(['error' => 'Неверно введён старый пароль']);  
 }  
 $user->password = Hash::make($request->new\_password);  
 $user->save();  
 return response()->json(['success' => 'Пароль успешно обновлен']);  
 }  
}

ServiceController.php – контроллер для работы с сервисами социальных сетей

<?php  
namespace App\Http\Controllers;  
use App\Models\Service;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Storage;  
class ServiceController extends Controller  
{  
 public function index()  
 {  
 $services = Service::all();  
 return view('admin.manage\_services', compact('services'));  
 }  
 public function create(Request $request)  
 {  
 $request->validate([  
 'name' => 'required|string|max:255',  
 'logo' => 'nullable|file|mimes:svg,png,jpg,webp|max:2048',  
 ]);  
  
 $logoName = null;  
 if ($request->hasFile('logo')) {  
 $logoName = $request->file('logo')->hashName();  
 $request->file('logo')->storeAs('public/logos', $logoName);  
 }  
 $service = Service::create([  
 'name' => $request->name,  
 'logo' => $logoName,  
 ]);  
 return response()->json(['message' => 'Сервис успешно создан', 'service' => $service]);  
 }  
 public function update(Request $request, Service $service)  
 {  
 $request->validate([  
 'name' => 'sometimes|required|string|max:255',  
 'logo' => 'nullable|file|mimes:svg,png,jpg,webp|max:2048',  
 ]);  
 if ($request->has('name')) {  
 $service->name = $request->name;  
 }  
 if ($request->hasFile('logo')) {  
 $logoName = $request->file('logo')->hashName();  
 if ($service->logo) {  
 Storage::delete('public/logos/' . $service->logo);  
 }  
 $request->file('logo')->storeAs('public/logos', $logoName);  
 $service->logo = $logoName;  
 }  
 $service->save();  
 return response()->json(['message' => 'Сервис успешно обновлен', 'service' => $service]);  
 }  
 public function destroy(Service $service)  
 {  
 $service->delete();  
 return response()->json(['message' => 'Сервис успешно удалён']);  
 }  
 public function show(Service $service)  
 {  
 $serviceItems = $service->serviceItems()->get();  
 return view('admin.manage\_service', compact('service', 'serviceItems'));  
 }  
}

ServiceItemController.php – контроллер для работы с услугами сервиса социальной сети

<?php  
namespace App\Http\Controllers;  
use App\Models\Service;  
use App\Models\ServiceItem;  
use Illuminate\Http\Request;  
class ServiceItemController extends Controller  
{  
 public function store(Request $request)  
 {  
 $service = Service::findOrFail($request->service\_id);  
 $validatedData = $request->validate([  
 'name' => 'required|string|max:255',  
 'price\_low' => 'required|numeric',  
 'price\_medium' => 'required|numeric',  
 'price\_high' => 'required|numeric',  
 ]);  
 ServiceItem::create([  
 'service\_id' => $service->id,  
 'name' => $validatedData['name'],  
 'price\_low' => $validatedData['price\_low'],  
 'price\_medium' => $validatedData['price\_medium'],  
 'price\_high' => $validatedData['price\_high'],  
 ]);  
 return redirect()->back()->with('success', 'Услуга успешно создана');  
 }  
 public function destroy($id)  
 {  
 $serviceItem = ServiceItem::findOrFail($id);  
 $serviceItem->delete();  
  
 return response()->json(['message' => 'Элемент успешно удален']);  
 }  
  
 public function update(Request $request, $id)  
 {  
 $serviceItem = ServiceItem::findOrFail($id);  
 $validatedData = $request->validate([  
 'name' => 'required|string|max:255',  
 'price\_low' => 'required|numeric',  
 'price\_medium' => 'required|numeric',  
 'price\_high' => 'required|numeric',  
 ]);  
 $serviceItem->update([  
 'name' => $validatedData['name'],  
 'price\_low' => $validatedData['price\_low'],  
 'price\_medium' => $validatedData['price\_medium'],  
 'price\_high' => $validatedData['price\_high'],  
 ]);  
 return response()->json(['message' => 'Элемент успешно обновлен', 'serviceItem' => $serviceItem]);  
 }  
}

OrderController.php – контроллер для работы с заказами в системе

<?php  
namespace App\Http\Controllers;  
use App\Models\BalanceTransaction;  
use App\Models\Order;  
use App\Models\Service;  
use App\Models\ServiceItem;  
use Exception;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Auth;  
use Illuminate\Database\Eloquent\ModelNotFoundException;  
use Illuminate\Validation\ValidationException;  
class OrderController extends Controller  
{  
 public function userOrders()  
 {  
 $user = Auth::user();  
 $orders = Order::where('user\_id', $user->id)  
 ->orderBy('created\_at', 'desc')  
 ->get();  
  
 return view('panel.my\_orders', compact('orders'));  
 }  
 public function allOrders()  
 {  
 $orders = Order::all();  
 return view('admin.all\_orders', compact('orders'));  
 }  
 public function create()  
 {  
 $services = Service::has('serviceItems')->get();  
 $serviceItems = ServiceItem::all();  
  
 return view('panel.order', compact('services', 'serviceItems'));  
 }  
 public function store(Request $request)  
 {  
 try {  
 $validatedData = $request->validate([  
 'source\_link' => 'required|string|max:300',  
 'quantity' => 'required|integer|min:50|max:13500',  
 'service\_item\_id' => 'required|exists:service\_items,id',  
 ]);  
 $serviceItem = ServiceItem::findOrFail($validatedData['service\_item\_id']);  
 $totalPrice = $serviceItem->price\_high \* $validatedData['quantity'];  
 $user = Auth::user();  
 $userBalance = $user->balance;  
 if ($userBalance->balance < $totalPrice) {  
 return response()->json(['message' => 'Недостаточно средств для создания заказа. Пожалуйста, пополните баланс'], 400);  
 }  
 $userBalance->balance -= $totalPrice;  
 $userBalance->save();  
 $order = new Order();  
 $order->user\_id = $user->id;  
 $order->service\_id = $serviceItem->service\_id;  
 $order->service\_item\_id = $validatedData['service\_item\_id'];  
 $order->source\_link = $validatedData['source\_link'];  
 $order->quantity = $validatedData['quantity'];  
 $order->total\_price = $totalPrice;  
 $order->status = 'В обработке';  
 $order->save();  
 $transaction = new BalanceTransaction([  
 'user\_id' => $user->id,  
 'amount' => -$totalPrice,  
 'type' => 'Выполнение заказа #' . $order->id,  
 ]);  
 $transaction->save();  
 return response()->json(['message' => 'Заказ успешно создан', 'order' => $order]);  
 } catch (ValidationException $e) {  
 return response()->json(['error' => 'Ошибка валидации', 'message' => $e->getMessage()], 400);  
 } catch (ModelNotFoundException $e) {  
 return response()->json(['error' => 'Заказ не был найден', 'message' => $e->getMessage()], 404);  
 } catch (Exception $e) {  
 return response()->json(['error' => 'Ошибка сервера', 'message' => $e->getMessage()], 500);  
 }  
 }  
 public function cancelOrder($orderId)  
 {  
 try {  
 $order = Order::findOrFail($orderId);  
 if ($order->status === 'Отменено') {  
 return response()->json(['error' => 'Заказ уже отменен'], 400);  
 }  
 $order->status = 'Отменен';  
 $order->save();  
 $userBalance = $order->user->balance;  
 $userBalance->balance += $order->total\_price;  
 $userBalance->save();  
 $transaction = new BalanceTransaction([  
 'user\_id' => $order->user->id,  
 'amount' => $order->total\_price,  
 'type' => 'Возврат за заказ #' . $order->id,  
 ]);  
 $transaction->save();  
 return response()->json(['message' => 'Заказ успешно отменен']);  
 } catch (Exception $e) {  
 return response()->json(['error' => 'Ошибка сервера', 'message' => $e->getMessage()], 500);  
 }  
 }  
}

AdminMiddleware.php – слой подключения для разрешения доступа только администратора

<?php  
namespace App\Http\Middleware;  
use Closure;  
use Illuminate\Http\Request;  
class AdminMiddleware  
{  
 public function handle(Request $request, Closure $next)  
 {  
 if (!$request->user() || $request->user()->role !== 'ADMIN') {  
 return redirect('/403\_error');  
 }  
 return $next($request);  
 }  
}

Authenticate.php – слой подключения для разрешения доступа только авторизованным

<?php  
namespace App\Http\Middleware;  
use Illuminate\Auth\Middleware\Authenticate as Middleware;  
use Illuminate\Http\Request;  
class Authenticate extends Middleware  
{  
 protected function redirectTo(Request $request): ?string  
 {  
 return $request->expectsJson() ? null : '/';  
 }  
}

CheckDatabaseConnection.php – слой подключения для проверки соединения с БД

<?php  
namespace App\Http\Middleware;  
use Closure;  
use Illuminate\Database\QueryException;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\DB;  
use Symfony\Component\HttpFoundation\Response;  
class CheckDatabaseConnection  
{  
 public function handle(Request $request, Closure $next): Response  
 {  
 try {  
 DB::connection()->getPdo();  
 } catch (\Exception) {  
 return redirect('/503\_error');  
 }  
 return $next($request);  
 }  
}

Index.blade.php – главная страница системы

@include('fragments/head', ['title' => 'Главная | GlorifyBoost'])  
<body class="body">  
@include('fragments.header')  
@include('home.home')  
@include('home.socials')  
@include('home.benefits')  
@include('home.services')  
@include('home.faq')  
@include('fragments.cta')  
@include('fragments/footer')  
@if(!\Illuminate\Support\Facades\Auth::check())  
 @include('fragments/modals')  
@endif  
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"  
 integrity="sha512-v2CJ7UaYy4JwqLDIrZUI/4hqeoQieOmAZNXBeQyjo21dadnwR+8ZaIJVT8EE2iyI61OV8e6M8PP2/4hpQINQ/g=="  
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>  
@if(!\Illuminate\Support\Facades\Auth::check())  
 @vite(['resources/js/app.js'])  
 @vite(['resources/js/register.js'])  
 @vite(['resources/js/components/captcha.js'])  
@endif  
@vite(['resources/js/components/menu.js'])  
@vite(['resources/js/components/accordion.js'])  
</body>

403\_error.blade.php – страница ошибки с 403 статусом

@include('fragments/head', ['title' => 'Ошибка доступа'])  
<body class="body">  
@include('fragments.header')  
<section class="error-page indent">  
 <div class="container">  
 <div class="error-page\_\_content">  
 <span class="error-page\_\_number">¯\\_(ツ)\_/¯</span>  
 <h1 class="error-page\_\_title">Доступ запрещен</h1>  
 <p class="error-page\_\_text">  
 У вас нет необходимых прав для доступа к этой странице  
 </p>  
 <a class="error-page\_\_btn btn" href="{{asset(route('index'))}}">На главную</a>  
 </div>  
 </div>  
</section>  
@vite(['resources/js/app.js'])  
@vite(['resources/js/components/menu.js'])  
</body>

404-error.blade.php – страница ошибки с 404 статусом

@include('fragments/head', ['title' => 'Страница не найдена'])  
<body class="body">  
@include('fragments.header')  
<section class="error-page indent">  
 <div class="container">  
 <div class="error-page\_\_content">  
 <span class="error-page\_\_number">¯\\_(ツ)\_/¯</span>  
 <h1 class="error-page\_\_title">Страница не найдена</h1>  
 <p class="error-page\_\_text">  
 Возможно, она удалена или был введен неправильный адрес  
 </p>  
 <a class="error-page\_\_btn btn" href="{{asset(route('index'))}}">На главную</a>  
 </div>  
 </div>  
</section>  
@vite(['resources/js/app.js'])  
@vite(['resources/js/components/menu.js'])  
</body>

503-error.blade.php – страница ошибки с 503 статусом

@include('fragments/head', ['title' => 'Технические шоколадки'])  
<body class="body">  
@include('fragments.header')  
<section class="error-page indent">  
 <div class="container">  
 <div class="error-page\_\_content">  
 <span class="error-page\_\_number">¯\\_(ツ)\_/¯</span>  
 <h1 class="error-page\_\_title">Время ожидания истекло</h1>  
 <p class="error-page\_\_text">  
 Что-то пошло не так в нашей базе! Наши специалисты уже решают проблему  
 </p>  
 <a class="error-page\_\_btn btn" href="{{asset(route('index'))}}">На главную</a>  
 </div>  
 </div>  
</section>  
@vite(['resources/js/app.js'])  
@vite(['resources/js/components/menu.js'])  
</body>

balance.blade.php – страница панели для работы с балансом

@include('fragments.head', ['title' => 'Пополнение баланса | GlorifyBoost'])  
<body class="body">  
@include('fragments.header')  
<section class="container">  
 <div class="panel\_\_inner">  
 @include('panel.components.aside')  
 <div class="panel\_\_content">  
 <div class="panel-head">  
 <h1 class="panel-head\_\_title">Пополнение</h1>  
 <p class="panel-head\_\_text">  
 В данном разделе предоставляются возможности для пополнения баланса,  
 отслеживания историй баланса и просто увидеть красоту  
 </p>  
 <form class="panel-balance\_\_form" id="deposit-form" method="post" action="{{route('deposit')}}">  
 @csrf  
 <label class="label" for="amount">Сумма для пополнения</label>  
 <input class="panel-balance\_\_input input" id="amount" name="amount" min="50" max="30000" step="0.01"  
 type="number" required>  
 <button class="panel-head\_\_btn btn" type="submit">Пополнить баланс</button>  
 </form>  
 @if($transactions->isEmpty())  
 <p class="panel-balance\_\_empty">На данный момент пополнений еще не имеется</p>  
 @else  
 <div>  
 <table class="panel-balance\_\_table">  
 <thead>  
 <tr class="panel-balance\_\_head">  
 <th>Идентификатор</th>  
 <th>Тип</th>  
 <th>Сумма</th>  
 <th>Дата</th>  
 </tr>  
 </thead>  
 <tbody class="panel-balance\_\_body">  
 @foreach($transactions as $transaction)  
 <tr class="panel-balance\_\_head panel-balance\_\_item">  
 <td>{{ $transaction->id }}</td>  
 <td class="panel-balance\_\_type  
 @if($transaction->type === 'Пополнение') panel-balance\_\_type--deposit  
 @elseif($transaction->type === 'Возврат средств') refund @endif">  
 {{ $transaction->type }}  
 </td>  
 <td>{{number\_format($transaction->amount, 2, ',', ' ')}} ₽</td>  
 <td>{{ $transaction->created\_at->isoFormat('D MMM YYYY') }}</td>  
 </tr>  
 @endforeach  
 </tbody>  
 </table>  
 </div>  
 @endif  
 </div>  
 </div>  
 </div>  
</section>  
</body>  
@vite(['resources/js/panel/balance.js'])  
@vite(['resources/js/components/menu.js'])

profile.blade.php – страница панели для работы с профилем

@include('fragments.head', ['title' => 'Профиль | GlorifyBoost'])  
<body class="body">  
@include('fragments.header')  
<section class="container">  
 <div class="panel\_\_inner">  
 @include('panel.components.aside')  
 <div class="panel\_\_content">  
 <div class="panel-head">  
 <h1 class="panel-head\_\_title">Настройки</h1>  
 <p class="panel-head\_\_text panel-head\_\_text--bottom">  
 В данном разделе предоставляются возможности для настройки имени, изменения пароля и других  
 параметров.  
 </p>  
 </div>  
 <div class="panel-avatar">  
 <h2 class="panel\_\_subtitle">Аватарка профиля</h2>  
 <p class="panel-avatar\_\_success">Аватар успешно обновлен!</p>  
 <span class="error" id="maxSizeError">Максимальный вес аватара составляет 2мб</span>  
 <span class="error" id="formatError">Разрешен только формат JPG, PNG, WEBP, GIF формата</span>  
 <form class="panel-avatar\_\_avatar">  
 @csrf  
 @if(Auth::user()->avatar)  
 <?php $avatarPath = asset('storage/avatars/' . Auth::user()->avatar) ?>  
 <img class="panel-avatar\_\_img" src="{{ $avatarPath }}" width="64" height="64"  
 alt="аватар пользователя" id="avatarImage">  
 @else  
 <?php $avatarPath = asset('static/images/avatar.png') ?>  
 <img class="panel-avatar\_\_img" src="{{ $avatarPath }}" width="64" height="64"  
 alt="аватар пользователя" id="avatarImage">  
 @endif  
 <button class="panel-avatar\_\_change" id="changeAvatarBtn" type="button">  
 Загрузить <br> новую фотографию  
 </button>  
 </form>  
 </div>  
 <div class="panel-data">  
 <h3 class="panel\_\_subtitle">Информация аккаунта</h3>  
 <ul class="panel-data\_\_list">  
 <li class="panel-data\_\_item">  
 <span class="panel-data\_\_name">ID аккаунта</span>  
 <span class="panel-data\_\_text">{{Auth::user()->id}}</span>  
 </li>  
 <li class="panel-data\_\_item">  
 <span class="panel-data\_\_name">Логин</span>  
 <span class="panel-data\_\_text">{{Auth::user()->login}}</span>  
 </li>  
 <li class="panel-data\_\_item">  
 <span class="panel-data\_\_name">Имя</span>  
 <span class="panel-data\_\_text" id="account-name">{{Auth::user()->name}}</span>  
 <button class="panel-data\_\_change" type="button" id="btn-name">Изменить</button>  
 </li>  
 <li class="panel-data\_\_item">  
 <span class="panel-data\_\_name">Почта</span>  
 <span class="panel-data\_\_text" id="account-email">{{Auth::user()->email}}</span>  
 <button class="panel-data\_\_change" type="button" id="btn-email">Изменить</button>  
 </li>  
 </ul>  
 </div>  
 <div class="panel-password">  
 <h3 class="panel\_\_subtitle">Изменить пароль</h3>  
 <form class="panel-password\_\_form" action="{{route('profile.update-password')}}" method="post">  
 @csrf  
 <div class="panel-password\_\_item">  
 <label class="label" for="old\_password">Старый пароль</label>  
 <span class="error" id="error\_not\_found-password">Неверно введён старый пароль</span>  
 <input class="input" type="password" id="old\_password" name="old\_password"  
 required placeholder="Введите пароль">  
 </div>  
 <div class="panel-password\_\_item">  
 <label class="label" for="new\_password">Новый пароль</label>  
 <span class="error" id="error\_new-password">Пароли не совпадают</span>  
 <span class="error" id="error\_valid-password">Пароли может содержать только латиницу и цифры</span>  
 <span class="error" id="error\_old-password">Новый пароль не должен быть похож на старый</span>  
 <span class="error" id="passwordError">Пароль не должен иметь кириллицу</span>  
 <input class="input" type="password" id="new\_password" name="new\_password"  
 required minlength="8" maxlength="60" placeholder="Введите пароль">  
 </div>  
 <div class="panel-password\_\_item">  
 <label class="label" for="repeat\_password">Повторите пароль</label>  
 <input class="input" type="password" id="repeat\_password" name="repeat\_password"  
 required placeholder="Введите пароль">  
 </div>  
 <button class="panel-password\_\_btn btn" disabled type="submit">Сохранить изменения</button>  
 </form>  
 </div>  
 </div>  
 </div>  
</section>  
<div class="modal" id="modal-update-name">  
 <button class="modal\_\_close" type="button" id="close-name">  
 </button>  
 <span class="modal\_\_title modal\_\_title--bottom">Изменение имени</span>  
 <form class="modal\_\_form" id="formUpdateName" method="post" action="{{route('profile.update-name')}}">  
 @csrf  
 <ul class="modal\_\_list">  
 <li class="auth\_\_item">  
 <label class="label" for="name">Имя</label>  
 <input class="input" id="name" required  
 minlength="2" maxlength="50"  
 type="text" name="name"  
 placeholder="Введите ваше имя"  
 pattern="[A-Za-zА-Яа-яЁё\-]+"  
 value="{{ Auth::user()->name }}"  
 title="Имя не должно содержать специальные символы, кроме дефиса"  
 >  
 </li>  
 </ul>  
 <button class="modal\_\_btn-auth btn" id="save-edit-name" type="submit">Сохранить изменения</button>  
 </form>  
</div>  
<div class="modal" id="modal-update-email">  
 <button class="modal\_\_close" type="button" id="close-email">  
 </button>  
 <span class="modal\_\_title modal\_\_title--bottom">Изменение почты</span>  
 <form class="modal\_\_form" id="formUpdateEmail" method="post" action="{{route('profile.update-email')}}">  
 @csrf  
 <span class="error" id="error-email">Данный email уже занят другим пользователем</span>  
 <ul class="modal\_\_list">  
 <li class="auth\_\_item">  
 <label class="label" for="email">Email</label>  
 <input class="input" type="email" required  
 maxlength="50"  
 id="email" name="email"  
 placeholder="Введите вашу почту"  
 value="{{ Auth::user()->email }}"  
 >  
 </li>  
 </ul>  
 <button class="modal\_\_btn-auth btn" id="save-edit-email" type="submit">Сохранить изменения</button>  
 </form>  
</div>  
</body>  
@vite(['resources/js/panel/profile/update-name.js'])  
@vite(['resources/js/panel/profile/update-email.js'])  
@vite(['resources/js/panel/profile/update-avatar.js'])  
@vite(['resources/js/panel/profile/update-password.js'])  
@vite(['resources/js/components/menu.js'])

referrals.blade.php – страница панели для работы с рефералами

@include('fragments.head', ['title' => 'Пополнение баланса | GlorifyBoost'])  
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/toastr.min.css">  
<body class="body">  
@include('fragments.header')  
<section class="container">  
 <div class="panel\_\_inner">  
 @include('panel.components.aside')  
 <div class="panel\_\_content">  
 <div class="panel-head">  
 <h1 class="panel-head\_\_title">Реферальная система</h1>  
 <p class="panel-head\_\_text">  
 Приглашайте своих друзей, знакомых или рекламируйте свою ссылку через социальные сети и  
 зарабатывайте деньги.  
 </p>  
 </div>  
 <div class="panel-referrals">  
 <ul class="panel-referrals\_\_list">  
 <li class="panel-referrals\_\_item">  
 <span class="panel-referrals\_\_digital">24</span>  
 <span class="panel-referrals\_\_subtext">Всего рефералов</span>  
 </li>  
 <li class="panel-referrals\_\_item">  
 <span class="panel-referrals\_\_digital">12</span>  
 <span class="panel-referrals\_\_subtext">Активных рефералов</span>  
 </li>  
 <li class="panel-referrals\_\_item">  
 <span class="panel-referrals\_\_digital">2 300 ₽</span>  
 <span class="panel-referrals\_\_subtext">Всего заработано</span>  
 </li>  
 </ul>  
 <div class="panel-referrals\_\_bonuses">  
 <div class="panel-referrals\_\_bonus">1 активный реферал — 5% бонус</div>  
 <div class="panel-referrals\_\_bonus">5 активный реферал — 10% бонус</div>  
 <div class="panel-referrals\_\_bonus">10 активный реферал — 15% бонус</div>  
 </div>  
 <div class="panel-referrals\_\_link">  
 <h2 class="panel\_\_subtitle">Ваша реферальная ссылка</h2>  
 <p class="panel-referrals\_\_info">  
 Создайте уникальную ссылку, отправьте ее другу в любом мессенджере или поделитесь в соцсетях  
 </p>  
 <div class="panel-referrals\_\_field">  
 https://glorify-boost.ru/?ref=392251  
 <button class="panel-referrals\_\_btn" type="button"></button>  
 </div>  
 </div>  
 <div class="panel-referrals\_\_description">  
 <h2 class="panel\_\_subtitle">Описание</h2>  
 <div class="panel-referrals\_\_text">  
 <p>  
 Создайте уникальную ссылку, отправьте ее другу в любом мессенджере или поделитесь  
 в соцсетях. Пришедший друг по вашему приглашению записывается как реферал  
 </p>  
 <p>  
 Посмотреть размер бонусов за приглашение можно в личном кабинете. Доход от пополнений ваших  
 активных рефералов автоматически добавляется на баланс сайта. Активным рефералом считается  
 пользователь, пополнивший баланс сайта на 1000₽  
 </p>  
 <p>  
 Вознаграждение от пополнений реферала высчитывается из количества привлеченных  
 пользователей. Чем больше у тебя активных рефералов — тем больше вознаграждение  
 </p>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</section>  
</body>  
<script src="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/toastr.min.js"></script>  
@vite(['resources/js/components/menu.js'])

order.blade.php – страница для создания нового задания (заказа)

@include('fragments.head', ['title' => 'Новый заказ | GlorifyBoost'])  
<body class="body">  
@include('fragments.header')  
<section class="container">  
 <div class="panel\_\_inner">  
 @include('panel.components.aside')  
 <div class="panel\_\_content">  
 <div class="panel-head">  
 <h1 class="panel-head\_\_title">Оформление заказа</h1>  
 <p class="panel-head\_\_text">  
 На этой странице вы можете разместить свой заказ  
 </p>  
 <form class="order" id="createOrder" method="post" action="{{route('order.store')}}">  
 @csrf  
 <ul class="order\_\_list">  
 @foreach($services as $service)  
 <li>  
 <button class="order\_\_item" data-service-id="{{$service->id}}" type="button">  
 <div class="order\_\_logo">  
 <img src="{{ asset('storage/logos/' . $service->logo) }}" alt="Логотип сервиса">  
 </div>  
 <span class="order\_\_name">{{$service->name}}</span>  
 </button>  
 </li>  
 @endforeach  
 </ul>  
 <div class="order\_\_services">  
 <span class="order\_\_subtitle">Тип услуги</span>  
 <div>  
 @foreach($services as $service)  
 <div class="order\_\_services-list" data-service-id="{{$service->id}}" style="display: none;">  
 @foreach($service->serviceItems as $item)  
 <button class="order\_\_services-btn" type="button"  
 data-item-id="{{$item->id}}"  
 data-price-high="{{$item->price\_high}}">{{$item->name}}</button>  
 @endforeach  
 </div>  
 @endforeach  
 </div>  
 </div>  
 <div class="order\_\_info">  
 <span class="order\_\_subtitle">Основная информация</span>  
 <ul class="order\_\_info-items">  
 <li class="order\_\_info-item">  
 <label class="label" for="source\_link">Ссылка на страницу, паблик или группу</label>  
 <input class="input" id="source\_link" name="source\_link" type="text" required  
 pattern="https:\/\/.\*" title="Укажите корректную ссылку с https протоколом"  
 placeholder="https://t.me/example">  
 </li>  
 <li class="order\_\_info-item">  
 <label class="label" for="quantity">Количество</label>  
 <input class="input" id="quantity" name="quantity" min="50" max="13000" type="number" required  
 placeholder="от 50 до 13 000">  
 </li>  
 </ul>  
 </div>  
 <input type="hidden" id="total\_price" name="total\_price" value="0">  
 <input type="hidden" name="service\_id">  
 <input type="hidden" name="service\_item\_id">  
 <button class="order\_\_btn" type="submit">  
 Создать заказ — 0₽  
 </button>  
 </form>  
 </div>  
 </div>  
 </div>  
</section>  
</body>  
@vite(['resources/js/components/menu.js'])  
@vite(['resources/js/panel/new-order.js'])

my-orders.blade.php – страница для просмотра всех своих созданных заданий

@include('fragments.head', ['title' => 'Мои задания | GlorifyBoost'])  
<body class="body">  
@include('fragments.header')  
<section class="container">  
 <div class="panel\_\_inner">  
 @include('panel.components.aside')  
 <div class="panel\_\_content">  
 <div class="panel-head">  
 <h1 class="panel-head\_\_title">Мои задания</h1>  
 <p class="panel-head\_\_text">  
 В данном разделе вы получаете возможность мониторить все свои заказы и быть в курсе текущего состояния  
 </p>  
 @if($orders->isEmpty())  
 <p class="panel-balance\_\_empty">На данный момент у вас ещё нет созданных заданий</p>  
 @else  
 <table class="table">  
 <thead>  
 <tr class="table\_\_head">  
 <th class="table\_\_id">ID заказа</th>  
 <th class="table\_\_date">Дата создания</th>  
 <th class="table\_\_link">Ссылка</th>  
 <th class="table\_\_price">Цена</th>  
 <th>Услуга</th>  
 <th>Статус</th>  
 </tr>  
 </thead>  
 <tbody class="table\_\_body">  
 @foreach($orders as $order)  
 <tr class="table\_\_head table\_\_item">  
 <td class="table\_\_id">{{ $order->id }}</td>  
 <td class="table\_\_date">{{ $order->created\_at->isoFormat('D MMM YYYY') }}</td>  
 <td class="table\_\_link">  
 <a href="{{ $order->source\_link }}" target="\_blank">{{$order->service->name}}</a>  
 </td>  
 <td class="table\_\_price">{{ number\_format($order->total\_price, 2, ',', ' ') }} ₽</td>  
 <td>  
 <span class="table\_\_service">{{ $order->serviceItem->name }}</span>  
 </td>  
 <td>{{ $order->status }}</td>  
 </tr>  
 @endforeach  
 </tbody>  
 </table>  
 @endif  
 </div>  
 </div>  
 </div>  
</section>  
</body>  
@vite(['resources/js/components/menu.js'])

all-orders.blade.php – страница администратора для просмотра всех заданий системы

@include('fragments/head', ['title' => 'GlorifyBoost | Администрирование заданий'])  
<body class="body">  
@include('fragments.header')  
<section class="admin">  
 <div class="container container--admin">  
 @include('fragments/header-admin')  
 <h1 class="admin\_\_title">Администрирование всех заданий</h1>  
 @if($orders->isEmpty())  
 <p class="panel-balance\_\_empty">На данный момент заказов, созданных пользователями, ещё нет в системе</p>  
 @else  
 <table class="table">  
 <thead>  
 <tr class="table\_\_head">  
 <th class="table\_\_id">ID заказа</th>  
 <th class="table\_\_date">Дата создания</th>  
 <th class="table\_\_link">Ссылка</th>  
 <th class="table\_\_price">Цена</th>  
 <th>Услуга</th>  
 <th>Пользователь</th>  
 <th class="table\_\_status">Статус</th>  
 <th class="table\_\_action">Действие</th>  
 </tr>  
 </thead>  
 <tbody class="table\_\_body">  
 @foreach($orders as $order)  
 <tr class="table\_\_head table\_\_item">  
 <td class="table\_\_id">{{ $order->id }}</td>  
 <td class="table\_\_date">{{ $order->created\_at->isoFormat('D MMM YYYY') }}</td>  
 <td class="table\_\_link">  
 <a href="{{ $order->source\_link }}" target="\_blank">{{$order->service->name}}</a>  
 </td>  
 <td class="table\_\_price">{{ number\_format($order->total\_price, 2, ',', ' ') }} ₽</td>  
 <td>  
 <span class="table\_\_service">{{ $order->serviceItem->name }}</span>  
 </td>  
 <td>{{ $order->user->login }}</td>  
 <td class="table\_\_status  
 @if($order->status === 'В обработке') processing  
 @elseif($order->status === 'Отменен') cancelled  
 @endif">  
 {{ $order->status }}  
 </td>  
 <td class="table\_\_action">  
 @if($order->status !== 'Отменен')  
 <button class="table\_\_delete" type="button">Отменить заказ</button>  
 @endif  
 </td>  
 </tr>  
 @endforeach  
 </tbody>  
 </table>  
 @endif  
 </div>  
</section>  
</body>  
  
<script>  
 document.addEventListener('DOMContentLoaded', function () {  
 const deleteButtons = document.querySelectorAll('.table\_\_delete');  
  
 deleteButtons.forEach(button => {  
 button.addEventListener('click', function () {  
 const orderId = this.closest('.table\_\_item').querySelector('.table\_\_id').textContent;  
 const confirmation = confirm('Вы уверены, что хотите отменить заказ #' + orderId + '? Отменить операцию будет невозможно');  
  
 if (confirmation) {  
 fetch(`/admin/orders/${orderId}/cancel`, {  
 method: 'PUT',  
 headers: {  
 'Content-Type': 'application/json',  
 'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').content  
 }  
 })  
 .then(response => {  
 if (response.ok) {  
 alert('Заказ успешно отменен');  
 location.reload();  
 } else {  
 return response.json().then(data => {  
 alert(data.error || 'Ошибка при отмене заказа');  
 });  
 }  
 })  
 .catch(error => {  
 console.error('Ошибка при выполнении запроса:', error);  
 alert('Ошибка при отмене заказа');  
 });  
 }  
 });  
 });  
 });  
</script>  
@vite(['resources/js/components/menu.js'])

manage-users.blade.php – сраница администратора для работы с пользователями

@include('fragments/head', ['title' => 'GlorifyBoost | Управление пользователями'])  
<body class="body">  
@include('fragments.header')  
<section class="admin">  
 <div class="container container--admin">  
 @include('fragments/header-admin')  
 <h1 class="admin\_\_title">Управление пользователями</h1>  
 <button class="admin-users\_\_btn" id="createUserBtn" type="button">Создать пользователя</button>  
 <form class="admin-users\_\_search" action="#" method="GET" id="searchForm">  
 <input class="input" type="text" id="query" name="query" placeholder="Поиск по логину, имени или почте">  
 </form>  
 <ul class="admin-users\_\_list" id="userListContainer">  
 @foreach($users as $user)  
 <li class="admin-users\_\_item" data-user-id="{{$user->id}}">  
 <p class="admin-users\_\_login"><span>{{$user->login}}</span></p>  
 @if($user->id !== auth()->id())  
 <a class="admin-users\_\_edit" href="{{route('admin.users.edit', ['user' => $user])}}">  
 Редактировать  
 </a>  
 @endif  
 <div class="admin-users\_\_content">  
 <p>Имя: {{$user->name}}</p>  
 <p>Почта: {{$user->email}}</p>  
 <p>Роль: {{$user->role}}</p>  
 </div>  
 @if($user->id !== auth()->id())  
 <div class="admin-users\_\_actions">  
 <button class="admin-users\_\_action admin-users\_\_action--delete" type="button"  
 data-user-id="{{$user->id}}">  
 Удалить аккаунт  
 </button>  
 </div>  
 @endif  
 </li>  
 @endforeach  
 </ul>  
 </div>  
</section>  
</body>  
<div class="modal modal--long" id="modalAddUser">  
 <button class="modal\_\_close" id="modalCloseAddUser" type="button"></button>  
 <h3 class="modal\_\_title modal\_\_title--bottom">Создание пользователя</h3>  
 <form action="{{ route('admin.users.store') }}" id="userForm" method="POST">  
 @csrf  
 <ul class="modal\_\_list modal\_\_list--grid">  
 <li>  
 <label class="label" for="name">Имя</label>  
 <span class="error" id="nameError">Имя не должно содержать запрещенные символы</span>  
 <span class="error" id="nameMinError">Мин. количество символов - 2</span>  
 <span class="error" id="nameMaxError">Макс. количество символов - 50</span>  
 <input class="input" type="text" id="name" name="name" required placeholder="Введите имя">  
 </li>  
 <li>  
 <label class="label" for="login">Логин</label>  
 <span class="error" id="loginCheckError">Логин уже используется</span>  
 <span class="error" id="loginError">Логин не должен иметь запрещенные символы</span>  
 <span class="error" id="loginLengthError">Минимальное количество символов - 5</span>  
 <span class="error" id="loginMaxError">Максимальное количество символов - 60</span>  
 <input class="input" type="text" id="login" name="login" required placeholder="Введите логин">  
 </li>  
 <li>  
 <label class="label" for="email">Почта</label>  
 <span class="error" id="emailCheckError">Почта уже используется</span>  
 <span class="error" id="emailErrorParameters">Почта не соответствует параметрам</span>  
 <span class="error" id="emailLengthError">Макс. количество символов - 80</span>  
 <input class="input" type="email" id="email" name="email" required placeholder="Введите почту">  
 </li>  
 <li>  
 <label class="label" for="password">Пароль</label>  
 <span class="error" id="passwordError">Пароль не должен иметь кириллицу</span>  
 <span class="error" id="passwordLengthError">Минимальное количество символов - 8</span>  
 <span class="error" id="passwordMaxError">Максимальное количество символов - 60</span>  
 <input class="input" type="password" id="password" name="password" required  
 placeholder="Введите пароль">  
 </li>  
 <li>  
 <label class="label" for="password">Роль</label>  
 <select class="select" id="role" name="role">  
 <option value="USER" {{ old('role') === 'USER' ? 'selected' : '' }}>Пользователь</option>  
 <option value="ADMIN" {{ old('role') === 'ADMIN' ? 'selected' : '' }}>Администратор</option>  
 </select>  
 </li>  
 </ul>  
 <button class="modal\_\_btn btn" id="createBtnUser" type="submit">Создать пользователя</button>  
 </form>  
</div>  
<div class="modal" id="modalDeleteUser">  
 <button class="modal\_\_close" id="modalCloseDeleteUser" type="button"></button>  
 <h3 class="modal\_\_title">Удаление пользователя</h3>  
 <p class="modal\_\_text">  
 Вы действительно хотите удалить пользователя <span id="loginDeleteUser"></span>? Удаление приведет к удалению  
 всех его заказов и информации, а также возможно к уничтожению данных о транзакциях  
 </p>  
 <div class="modal\_\_buttons">  
 <button class="btn modal\_\_btn--cancel" id="modalCancelDeleteUser" type="button">Отменить</button>  
 <button class="btn modal\_\_btn--confirm" id="confirmDeleteUser" type="submit">Да, удалить</button>  
 </div>  
</div>  
@vite(['resources/js/components/custom-select.js'])  
@vite(['resources/js/user.js?type=module'])  
@vite(['resources/js/components/menu.js'])

manage-transactions.blade.php – страница администратора для работы с транзакциями системы

@include('fragments/head', ['title' => 'GlorifyBoost | Работа с транзакциями'])  
<body class="body">  
@include('fragments.header')  
<section class="admin">  
 <div class="container container--admin">  
 @include('fragments/header-admin')  
 <h1 class="admin\_\_title">Работа с транзакциями</h1>  
 @if($transactions->isEmpty())  
 <p class="panel-balance\_\_empty">На данный момент пополнений еще не имеется</p>  
 @else  
 <table class="panel-balance\_\_table">  
 <thead>  
 <tr class="panel-balance\_\_head admin">  
 <th>Идентификатор</th>  
 <th>Тип</th>  
 <th>Сумма</th>  
 <th>Дата пополнения</th>  
 <th>Пользователь</th>  
 <th>Действия</th>  
 </tr>  
 </thead>  
 <tbody class="panel-balance\_\_body">  
 @foreach($transactions as $transaction)  
 <tr class="panel-balance\_\_head panel-balance\_\_item admin  
 @if($transaction->cancelled) cancelled @endif  
 @if($transaction->type === 'Возврат средств') refund @endif">  
 <td>{{ $transaction->id }}</td>  
 <td class="panel-balance\_\_type  
 @if($transaction->type === 'Пополнение') panel-balance\_\_type--deposit  
 @elseif($transaction->type === 'Возврат средств') refund @endif">  
 {{ $transaction->type }}  
 </td>  
 <td>{{ number\_format($transaction->amount, 2, ',', ' ') }} ₽</td>  
 <td>{{ $transaction->created\_at->isoFormat('D MMM YYYY') }}</td>  
 <td>{{ $transaction->user->login }}</td>  
 <td>  
 @if(!$transaction->cancelled && $transaction->type !== 'Возврат средств' && !str\_starts\_with($transaction->type, 'Выполнение заказа'))  
 <button class="panel-balance\_\_btn cancel-payment-btn"  
 data-transaction-id="{{ $transaction->id }}"  
 type="button">  
 Отменить платеж  
 </button>  
 @endif  
 </td>  
 </tr>  
 @endforeach  
 </tbody>  
 </table>  
 @endif  
 </div>  
</section>  
</body>  
  
<div class="modal" id="modalCancelPayment">  
 <h3 class="modal\_\_title">Отмена пополнения</h3>  
 <p class="modal\_\_text">  
 Вы действительно хотите отменить транзакцию у пользователя <span id="loginUser"></span>?  
 Отмена приведёт к списанию баланса у пользователя. Отменить действие будет невозможно  
 </p>  
 <div class="modal\_\_buttons">  
 <button class="btn modal\_\_btn--cancel" id="modalCancel" type="button">Закрыть окно</button>  
 <button class="btn modal\_\_btn--confirm" id="modalConfirm" type="submit">Подтверждаю</button>  
 </div>  
</div>  
@vite(['resources/js/user/transactions.js'])  
@vite(['resources/js/components/menu.js'])

edit-user.blade.php – страница администратора для редактирования пользователя

@include('fragments/head', ['title' => 'GlorifyBoost | Редактирование пользователя'])  
<body class="body">  
@include('fragments.header')  
<section class="admin">  
 <div class="container container--admin">  
 <div class="admin\_\_wrapper">  
 <h1 class="admin\_\_title">Редактирование пользователя</h1>  
 <a class="admin\_\_back" href={{route('admin.users.index')}}>Вернуться назад к пользователям</a>  
 <form id="updateForm" action="{{ route('admin.users.update', ['user' => $user]) }}" method="POST">  
 @csrf  
 @method('PUT')  
 <div class="admin-edit\_\_list">  
 <div class="admin-edit\_\_item">  
 <label for="login">Логин</label>  
 <span class="error" id="loginCheckError">Логин уже используется</span>  
 <span class="error" id="loginError">Логин не должен иметь запрещенные символы</span>  
 <span class="error" id="loginLengthError">Минимальное количество символов - 5</span>  
 <span class="error" id="loginMaxError">Максимальное количество символов - 60</span>  
 <input class="input" type="text" id="login" name="login" value="{{$user->login}}" required>  
 </div>  
 <div class="admin-edit\_\_item">  
 <label for="name">Имя</label>  
 <span class="error" id="nameError">Имя не должно содержать запрещенные символы</span>  
 <span class="error" id="nameMinError">Мин. количество символов - 2</span>  
 <span class="error" id="nameMaxError">Макс. количество символов - 50</span>  
 <input class="input" type="text" id="name" name="name" value="{{$user->name}}" required>  
 </div>  
 <div class="admin-edit\_\_item">  
 <label for="email">Почта</label>  
 <span class="error" id="emailCheckError">Почта уже используется</span>  
 <span class="error" id="emailErrorParameters">Почта не соответствует параметрам</span>  
 <span class="error" id="emailLengthError">Макс. количество символов - 80</span>  
 <input class="input" type="email" id="email" name="email" value="{{$user->email}}" required>  
 </div>  
 <div class="admin-edit\_\_item">  
 <label for="role">Роль</label>  
 <select class="select" id="role" name="role">  
 <option value="USER" {{ $user->role === 'USER' ? 'selected' : '' }}>Пользователь</option>  
 <option value="ADMIN" {{ $user->role === 'ADMIN' ? 'selected' : '' }}>Администратор</option>  
 </select>  
 </div>  
 </div>  
 <button class="admin\_\_btn" type="submit" id="updateButton">Обновить</button>  
 </form>  
 </div>  
 </div>  
</section>  
</body>  
@vite(['resources/js/components/custom-select.js'])  
@vite(['resources/js/user/edit.js?type=module'])  
@vite(['resources/js/user.js?type=module'])  
@vite(['resources/js/components/menu.js'])

manage-services.blade.php – страница администратора для управления сервисами социальных сетей

@include('fragments/head', ['title' => 'GlorifyBoost | Редактирование для заказов'])  
<body class="body">  
@include('fragments.header')  
<section class="admin">  
 <div class="container container--admin">  
 @include('fragments/header-admin')  
 <h1 class="admin\_\_title">Работа с сервисами</h1>  
 <button class="admin-services\_\_create" id="btnCreateService" type="button">Создать сервис</button>  
 @if($services->isEmpty())  
 <p class="panel-balance\_\_empty">На данный момент сервисов ещё нет, но вы можете их создать</p>  
 @else  
 <ul class="admin-services\_\_list">  
 @foreach($services as $service)  
 <li class="admin-services\_\_item">  
 <div class="admin-services\_\_content">  
 <div class="admin-services\_\_logo">  
 <img src="{{ asset('storage/logos/' . $service->logo) }}" alt="Логотип сервиса">  
 </div>  
 <span class="admin-services\_\_name">  
 {{$service->name}}  
 </span>  
 </div>  
 <div class="admin-services\_\_actions">  
 <a class="admin-services\_\_action edit"  
 href="{{ route('admin.services.show', ['service' => $service->id]) }}"  
 >  
 Редактировать  
 </a>  
 <button  
 class="admin-services\_\_action delete"  
 type="button"  
 data-service-id="{{ $service->id }}"  
 data-service-name="{{ $service->name }}"  
 >  
 Удалить  
 </button>  
 </div>  
 </li>  
 @endforeach  
 </ul>  
 @endif  
 </div>  
</section>  
</body>  
  
<div class="modal" id="modalAddService">  
 <h3 class="modal\_\_title modal\_\_title--bottom">Добавление сервиса</h3>  
 <form class="modal\_\_form" id="modalFormAddService" method="post" action="{{ route('admin.services.create') }}">  
 <button class="modal\_\_close" id="btnCloseCreateService" type="button"></button>  
 @csrf  
 <div class="modal\_\_list">  
 <div class="admin-edit\_\_item">  
 <label for="name">Название сервиса</label>  
 <input class="input" type="text" id="name" maxlength="255" name="name" required>  
 </div>  
 <div class="admin-edit\_\_item">  
 <label for="logo">Логотип сервиса</label>  
 <div class="admin-services\_\_errors">  
 <span class="error" id="maxSizeError">Максимальный вес логотипа 2мб</span>  
 <span class="error" id="formatImageError">Разрешен только формат .svg, .png, .jpg и .webp</span>  
 </div>  
 <input class="input" type="file" id="logo" name="logo" accept=".svg,.png,.jpg,.webp" required>  
 </div>  
 </div>  
 <button class="btn" type="submit">Добавить сервис</button>  
 </form>  
</div>  
  
<div class="modal" id="modalDeleteService">  
 <h3 class="modal\_\_title modal\_\_title--bottom">Удаление сервиса</h3>  
 <p class="modal\_\_text">  
 Вы действительно хотите удалить сервис <span id="service"></span>? Удаление приведет к удалению  
 всех заказов (относящиеся к данному сервису), отменить действие будет невозможно  
 </p>  
 <div class="modal\_\_buttons">  
 <button class="btn modal\_\_btn--cancel" id="btnCloseDeleteService" type="button">Отменить</button>  
 <button class="btn modal\_\_btn--confirm" id="btnConfirmDeleteService" type="submit">Да, удалить</button>  
 </div>  
</div>  
  
@vite(['resources/js/services.js'])  
@vite(['resources/js/components/menu.js'])

manage-service.blade.php – страница администратора для управления контентом конкретного сервиса

@include('fragments/head', ['title' => 'GlorifyBoost | Редактирование для ' . $service->name])  
<body class="body">  
@include('fragments.header')  
<section class="admin">  
 <div class="container container--admin">  
 @include('fragments/header-admin')  
 <h1 class="admin\_\_title">  
 Работа с сервисом {{$service->name}}  
 <img width="30" height="30" src="{{ asset('storage/logos/' . $service->logo) }}" alt="Логотип сервиса">  
 </h1>  
 <form class="admin-service\_\_form" action="{{ route('admin.services.update', ['service' => $service->id]) }}"  
 method="POST" enctype="multipart/form-data">  
 @csrf  
 @method('PUT')  
 <ul class="admin-service\_\_items">  
 <li>  
 <label class="label" for="name">Название сервиса:</label>  
 <input class="input" type="text" id="name" name="name" value="{{ $service->name }}" required>  
 </li>  
 <li>  
 <label class="label" for="logo">Логотип сервиса (необязательно):</label>  
 <span class="error" id="maxSizeError">Максимальный вес логотипа 2мб</span>  
 <span class="error" id="formatImageError">Разрешен только формат .svg, .png, .jpg и .webp</span>  
 <input class="input" type="file" id="logo" name="logo">  
 </li>  
 </ul>  
 <button class="admin-service\_\_save" type="submit">Сохранить изменения</button>  
 </form>  
 <button class="create" id="btnCreate" type="button">Создать новую услугу</button>  
 <form id="createService" action="{{ route('admin.service\_items.store') }}" method="POST" style="display: none">  
 @csrf  
 <input type="hidden" id="service\_id" name="service\_id" value="{{ $service->id }}">  
 <ul class="admin-service\_\_list">  
 <li>  
 <label class="label" for="name">Название услуги</label>  
 <input class="input" type="text" id="name" name="name" required maxlength="255">  
 </li>  
 <li>  
 <label class="label" for="price\_low">Цена за низкое качество</label>  
 <input class="input" type="text" id="price\_low" name="price\_low" required data-input-mask="price">  
 </li>  
 <li>  
 <label class="label" for="price\_medium">Цена за среднее качество</label>  
 <input class="input" type="text" id="price\_medium" name="price\_medium" required data-input-mask="price">  
 </li>  
 <li>  
 <label class="label" for="price\_high">Цена за высокое качество</label>  
 <input class="input" type="text" id="price\_high" name="price\_high" required data-input-mask="price">  
 </li>  
 </ul>  
 <button class="admin-service\_\_btn" type="submit">Создать услугу</button>  
 </form>  
 @if($serviceItems->isEmpty())  
 <p class="admin-service\_\_empty">На данный момент услуг сервиса ещё нет, но вы можете их создать</p>  
 @else  
 <ul class="list-items">  
 @foreach($serviceItems as $serviceItem)  
 <li class="list-item" data-service-id="{{ $serviceItem->id }}">  
 <div class="list-item\_\_actions">  
 <button class="list-item\_\_delete" type="button"  
 data-url="{{ route('admin.service\_items.destroy', ['serviceItem' => $serviceItem->id]) }}"  
 >  
 Удалить услугу  
 </button>  
 |  
 <button class="list-item\_\_edit" type="button"  
 data-service-id="{{ $serviceItem->id }}"  
 data-name="{{ $serviceItem->name }}"  
 data-price-low="{{ $serviceItem->price\_low }}"  
 data-price-medium="{{ $serviceItem->price\_medium }}"  
 data-price-high="{{ $serviceItem->price\_high }}"  
 >  
 Редактировать услугу  
 </button>  
 </div>  
 <p class="list-item\_\_name">{{ $serviceItem->name }}</p>  
 <div class="list-item\_\_prices">  
 <p class="list-item\_\_text">  
 Низкая  
 <span  
 id="price\_low\_{{ $serviceItem->id }}">{{ number\_format($serviceItem->price\_low, 2, '.', ' ') }} ₽</span>  
 </p>  
 <p class="list-item\_\_text">  
 Средняя  
 <span id="price\_medium\_{{ $serviceItem->id }}">{{ number\_format($serviceItem->price\_medium, 2, '.', ' ') }} ₽</span>  
 </p>  
 <p class="list-item\_\_text">  
 Высокая  
 <span id="price\_high\_{{ $serviceItem->id }}">{{ number\_format($serviceItem->price\_high, 2, '.', ' ') }} ₽</span>  
 </p>  
 </div>  
 </li>  
 @endforeach  
 </ul>  
 @endif  
 </div>  
</section>  
<div class="modal modal--long" id="modalEdit">  
 <h3 class="modal\_\_title modal\_\_title--bottom">Редактирование услуги</h3>  
 <form class="modal\_\_form" id="modalForm" method="post" action="{{ route('admin.services.create') }}">  
 <input type="hidden" id="serviceItemId" name="serviceItemId" value="{{ $service->id }}">  
 <button class="modal\_\_close" id="btnCloseModalEdit" type="button"></button>  
 @csrf  
 <div class="modal\_\_list modal\_\_list--grid">  
 <div class="admin-edit\_\_item">  
 <label class="label" for="name\_edit">Название услуги</label>  
 <input class="input" type="text" id="name\_edit" maxlength="255" name="name\_edit" required>  
 </div>  
 <div class="admin-edit\_\_item">  
 <label class="label" for="price\_low\_edit">Цена за низкое качество</label>  
 <input class="input" type="text" id="price\_low\_edit" name="price\_low\_edit" required data-input-mask="price">  
 </div>  
 <div class="admin-edit\_\_item">  
 <label class="label" for="price\_medium\_edit">Цена за среднее качество</label>  
 <input class="input" type="text" id="price\_medium\_edit" name="price\_medium\_edit" required  
 data-input-mask="price">  
 </div>  
 <div class="admin-edit\_\_item">  
 <label class="label" for="price\_high\_edit">Цена за высокое качество</label>  
 <input class="input" type="text" id="price\_high\_edit" name="price\_high\_edit" required data-input-mask="price">  
 </div>  
 </div>  
 <button class="btn" type="submit">Сохранить изменения</button>  
 </form>  
</div>  
</body  
@vite(['resources/js/service.js'])  
<script>  
 const editService = async () => {  
 const formData = new FormData(document.querySelector('.admin-service\_\_form'));  
 try {  
 const response = await fetch('{{ route('admin.services.update', ['service' => $service->id]) }}', {  
 method: 'POST',  
 body: formData  
 });  
 if (response.ok) {  
 const serviceName = document.getElementById('name').value;  
 const serviceLogo = document.getElementById('logo');  
 let logoSrc = '{{ asset('storage/logos/' . $service->logo) }}';  
 if (serviceLogo && serviceLogo.files.length > 0) {  
 const maxFileSize = 2 \* 1024 \* 1024; // 2MB  
 const allowedFormats = ['image/svg+xml', 'image/png', 'image/jpeg', 'image/webp'];  
 const file = serviceLogo.files[0];  
 if (file.size > maxFileSize) {  
 document.getElementById('maxSizeError').classList.add('error--active');  
 setTimeout(() => {  
 document.getElementById('maxSizeError').classList.remove('error--active');  
 }, 1500);  
 return;  
 }  
 if (!allowedFormats.includes(file.type)) {  
 document.getElementById('formatImageError').classList.add('error--active');  
 setTimeout(() => {  
 document.getElementById('formatImageError').classList.remove('error--active');  
 }, 1500);  
 return;  
 }  
 logoSrc = URL.createObjectURL(file);  
 }  
 document.querySelector('.admin\_\_title').innerHTML = `  
 Работа с сервисом ${serviceName}  
 <img width="30" height="30" src="${logoSrc}" alt="Логотип сервиса">  
 `;  
 alert('Сервис успешно отредактирован');  
 } else {  
 throw new Error('Ошибка редактирования сервиса');  
 }  
 } catch (error) {  
 console.error(error);  
 alert('Произошла ошибка при редактировании сервиса');  
 }  
 };  
  
 document.querySelector('.admin-service\_\_form').addEventListener('submit', (event) => {  
 event.preventDefault();  
 editService();  
 });  
</script>  
<script>  
 document.addEventListener('DOMContentLoaded', function () {  
 const deleteButtons = document.querySelectorAll('.list-item\_\_delete');  
  
 deleteButtons.forEach(button => {  
 button.addEventListener('click', function () {  
 const listItem = button.closest('.list-item');  
 const itemName = listItem.querySelector('.list-item\_\_name').textContent;  
  
 if (confirm(`Вы уверены, что хотите удалить услугу "${itemName}"?`)) {  
 const url = button.dataset.url;  
  
 fetch(url, {  
 method: 'DELETE',  
 headers: {  
 'X-CSRF-TOKEN': '{{ csrf\_token() }}'  
 }  
 })  
 .then(response => {  
 if (response.ok) {  
 listItem.remove();  
 } else {  
 throw new Error('Ошибка удаления элемента');  
 }  
 })  
 .catch(error => {  
 console.error(error);  
 alert('Произошла ошибка при удалении элемента');  
 });  
 }  
 });  
 });  
 });  
</script>  
@vite(['resources/js/components/menu.js'])

Register.js – код для работы с регистрацией на клиенте

$(document).ready(function () {  
 function validateName() {  
 const nameValue = $("#name").val();  
 const nameError = $("#nameError");  
 const nameMaxError = $("#nameMaxError");  
 const regex = /^[A-Za-zА-я\-]+$/;  
  
 if (nameValue.trim() === "") {  
 nameError.removeClass("error--active");  
 nameMaxError.removeClass("error--active");  
 return false;  
 }  
  
 if (!regex.test(nameValue)) {  
 nameError.addClass("error--active");  
 return false;  
 } else {  
 nameError.removeClass("error--active");  
 }  
  
 if (nameValue.length > 50) {  
 nameMaxError.addClass("error--active");  
 return false;  
 } else {  
 nameMaxError.removeClass("error--active");  
 }  
  
 return true;  
 }  
  
 function validateEmail() {  
 const emailValue = $("#email").val();  
 const emailError = $("#emailError");  
 const emailErrorParameters = $("#emailErrorParameters");  
 const emailLengthError = $("#emailLengthError");  
 const regex = /^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$/;  
  
 let valid = true;  
  
 if (emailValue.trim() === '') {  
 emailErrorParameters.removeClass("error--active");  
 return false;  
 }  
  
 if (!regex.test(emailValue)) {  
 emailErrorParameters.addClass("error--active");  
 return false;  
 } else {  
 emailErrorParameters.removeClass("error--active");  
 }  
  
 if (emailValue.length > 80) {  
 emailLengthError.addClass("error--active");  
 return false;  
 } else {  
 emailLengthError.removeClass("error--active");  
 }  
  
 $.ajax({  
 headers: {  
 'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')  
 },  
 type: 'POST',  
 url: '/check-email',  
 data: {  
 email: emailValue  
 },  
 success: function (response) {  
 if (response.exists) {  
 emailError.addClass('error--active');  
 } else {  
 emailError.removeClass('error--active');  
 }  
 },  
 error: function (xhr, status, error) {  
 console.error(error);  
 }  
 });  
  
 return valid;  
 }  
  
 function validateLogin() {  
 const loginValue = $("#login").val();  
 const loginError = $("#loginError");  
 const loginCheckError = $("#loginCheckError");  
 const loginLengthError = $("#loginLengthError");  
 const loginMaxError = $("#loginMaxError");  
  
 if (loginValue.trim() === "") {  
 loginError.removeClass("error--active");  
 loginLengthError.removeClass("error--active");  
 return false;  
 }  
  
 let valid = true;  
  
 if (/[А-я]/.test(loginValue) || !/^[a-zA-Z0-9\_]+$/.test(loginValue)) {  
 loginError.addClass("error--active");  
 valid = false;  
 } else {  
 loginError.removeClass("error--active");  
 }  
  
 if (loginValue.length < 5) {  
 loginLengthError.addClass("error--active");  
 valid = false;  
 } else {  
 loginLengthError.removeClass("error--active");  
 }  
  
 if (loginValue.length > 60) {  
 loginMaxError.addClass("error--active");  
 valid = false;  
 } else {  
 loginMaxError.removeClass("error--active");  
 }  
  
 setTimeout(() => {  
 $.ajax({  
 headers: {  
 'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')  
 },  
 type: 'POST',  
 url: '/check-login',  
 data: {  
 login: loginValue  
 },  
 success: function (response) {  
 if (response.exists) {  
 loginCheckError.addClass('error--active');  
 } else {  
 loginCheckError.removeClass('error--active');  
 }  
 },  
 error: function (xhr, status, error) {  
 console.error(error);  
 }  
 });  
 }, 500);  
  
 return valid;  
 }  
  
 function validatePassword() {  
 const passwordValue = $("#password").val();  
 const passwordError = $("#passwordError");  
 const passwordLengthError = $("#passwordLengthError");  
 const passwordMaxError = $("#passwordMaxError");  
  
 if (passwordValue.trim() === "") {  
 passwordError.removeClass("error--active");  
 passwordLengthError.removeClass("error--active");  
 passwordMaxError.removeClass("error--active");  
 return false;  
 }  
  
 let valid = true;  
  
 if (/[А-я]/.test(passwordValue)) {  
 passwordError.addClass("error--active");  
 valid = false;  
 } else {  
 passwordError.removeClass("error--active");  
 }  
  
 if (passwordValue.length < 5) {  
 passwordLengthError.addClass("error--active");  
 valid = false;  
 } else {  
 passwordLengthError.removeClass("error--active");  
 }  
  
 if (passwordValue.length > 60) {  
 passwordMaxError.addClass("error--active");  
 valid = false;  
 } else {  
 passwordMaxError.removeClass("error--active");  
 }  
  
 return valid;  
 }  
  
 function updateSubmitButtonState() {  
 const nameValid = validateName();  
 const emailValid = validateEmail();  
 const loginValid = validateLogin();  
 const passwordValid = validatePassword();  
 const submitButton = $("#registration-btn");  
  
 if (nameValid && emailValid && loginValid && passwordValid) {  
 submitButton.prop("disabled", false);  
 submitButton.css("opacity", "1");  
 submitButton.css("pointer-events", "auto");  
 } else {  
 submitButton.prop("disabled", true);  
 submitButton.css("opacity", "0.6");  
 submitButton.css("pointer-events", "none");  
 }  
 }  
  
 $("#name").on("input", updateSubmitButtonState);  
 $("#email").on("input", updateSubmitButtonState);  
 $("#login").on("input", updateSubmitButtonState);  
 $("#password").on("input", updateSubmitButtonState);  
 updateSubmitButtonState();  
});

Services.js – файл для работы с сервисами социальных сетей

import {closeModal, handleModalClose, openModal} from './components/modal-functions.js';  
  
const modalAdd = document.getElementById('modalAddService');  
const modalDelete = document.getElementById('modalDeleteService');  
const modalEdit = document.getElementById('modalEditService');  
  
const btnCreateService = document.getElementById('btnCreateService');  
const btnCloseCreateService = document.getElementById('btnCloseCreateService');  
const btnCloseDeleteService = document.getElementById('btnCloseDeleteService');  
  
const modalFormCreate = document.getElementById('modalFormAddService');  
const servicesList = document.querySelector('.admin-services\_\_list');  
  
btnCreateService.addEventListener('click', () => {  
 openModal(modalAdd);  
});  
btnCloseCreateService.addEventListener('click', () => {  
 closeModal(modalAdd);  
});  
  
btnCloseDeleteService.addEventListener('click', () => {  
 closeModal(modalDelete);  
});  
  
modalFormCreate.addEventListener('submit', async (e) => {  
 e.preventDefault();  
  
 const formData = new FormData(modalFormCreate);  
 const url = modalFormCreate.getAttribute('action');  
  
 const logoInput = document.getElementById('logo');  
 const maxSizeError = document.getElementById('maxSizeError');  
 const formatImageError = document.getElementById('formatImageError');  
  
 // очистка ошибки перед каждой отправкой формы  
 maxSizeError.classList.remove('error--active');  
 formatImageError.classList.remove('error--active');  
  
 const allowedFormats = ['.svg', '.png', '.jpg', '.webp'];  
 const fileExtension = logoInput.value.substring(logoInput.value.lastIndexOf('.')).toLowerCase();  
 if (!allowedFormats.includes(fileExtension)) {  
 formatImageError.classList.add('error--active');  
 logoInput.value = '';  
 setTimeout(() => {  
 formatImageError.classList.remove('error--active');  
 }, 2500);  
 return;  
 }  
  
 if (logoInput.files[0].size > 2 \* 1024 \* 1024) {  
 maxSizeError.classList.add('error--active');  
 logoInput.value = '';  
 setTimeout(() => {  
 maxSizeError.classList.remove('error--active');  
 }, 2500);  
 return;  
 }  
  
 try {  
 const response = await fetch(url, {  
 method: 'POST',  
 body: formData  
 });  
  
 if (!response.ok) {  
 throw new Error('Ошибка при отправке формы');  
 }  
  
 const data = await response.json();  
 modalFormCreate.reset();  
  
 const newServiceItem = document.createElement('li');  
 newServiceItem.classList.add('admin-services\_\_item');  
 newServiceItem.innerHTML = `  
 <div class="admin-services\_\_content">  
 <div class="admin-services\_\_logo">  
 <img src="/storage/logos/${data.service.logo}" alt="Логотип сервиса">  
 </div>  
 <span>${data.service.name}</span>  
 </div>  
 <div class="admin-services\_\_actions">  
 <a class="admin-services\_\_action edit" href="/admin/services/${data.service.id}">  
 Редактировать  
 </a>  
 <button  
 class="admin-services\_\_action delete"  
 type="button"  
 data-service-id="${data.service.id}" // Добавляем атрибут с ID сервиса  
 data-service-name="${data.service.name}" // Добавляем атрибут с именем сервиса  
 >  
 Удалить  
 </button>  
 </div>  
 `;  
 servicesList.appendChild(newServiceItem);  
  
 closeModal(modalAdd);  
 } catch (error) {  
 console.error('Ошибка:', error);  
 }  
});  
  
servicesList.addEventListener('click', async (e) => {  
 if (e.target.classList.contains('delete')) {  
 const serviceId = e.target.dataset.serviceId;  
 document.getElementById('service').textContent = e.target.dataset.serviceName;  
 openModal(modalDelete);  
  
 document.getElementById('btnConfirmDeleteService').addEventListener('click', async () => {  
 try {  
 const response = await fetch(`/admin/services/${serviceId}`, {  
 method: 'DELETE',  
 headers: {  
 'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').content  
 }  
 });  
  
 if (!response.ok) {  
 throw new Error('Ошибка при удалении сервиса');  
 }  
  
 await response.json();  
 e.target.closest('.admin-services\_\_item').remove();  
  
 closeModal(modalDelete);  
 } catch (error) {  
 console.error('Ошибка:', error);  
 }  
 });  
 }  
});  
  
handleModalClose(modalAdd);  
handleModalClose(modalDelete);  
handleModalClose(modalEdit);

service.js – файл для работы с услугами и конкретным сервисом социальной сети

import {closeModal, handleModalClose, openModal} from './components/modal-functions.js';  
  
const btnCreate = document.getElementById('btnCreate');  
const createServiceForm = document.getElementById('createService');  
const modal = document.getElementById('modalEdit');  
const btnCloseModal = document.getElementById('btnCloseModalEdit');  
const editButtons = document.querySelectorAll('.list-item\_\_edit');  
  
btnCreate.addEventListener('click', () => {  
 createServiceForm.style.display = 'block';  
});  
  
document.querySelectorAll('input[data-input-mask="price"]').forEach(function (input) {  
 input.addEventListener('input', function () {  
 this.value = this.value  
 .replace(/[^\d.]/g, '')  
 .replace(/(?<=\.\d{2})./g, '');  
  
 const decimalIndex = this.value.indexOf('.');  
 if (decimalIndex !== -1) {  
 const parts = this.value.split('.');  
 if (parts.length > 2) {  
 this.value = parts[0] + '.' + parts.slice(1).join('');  
 }  
 }  
 });  
  
 input.addEventListener('keydown', function (event) {  
 if (event.keyCode === 190) {  
 const dotCount = (this.value.match(/\./g) || []).length;  
 if (dotCount >= 1) {  
 event.preventDefault();  
 }  
 }  
 });  
});  
  
editButtons.forEach(button => {  
 button.addEventListener('click', function () {  
 const serviceItemId = button.dataset.serviceId;  
 const name = button.dataset.name;  
 const priceLow = button.dataset.priceLow;  
 const priceMedium = button.dataset.priceMedium;  
 const priceHigh = button.dataset.priceHigh;  
  
 document.getElementById('serviceItemId').value = serviceItemId;  
 document.getElementById('name\_edit').value = name;  
 document.getElementById('price\_low\_edit').value = priceLow;  
 document.getElementById('price\_medium\_edit').value = priceMedium;  
 document.getElementById('price\_high\_edit').value = priceHigh;  
  
 openModal(modal);  
 });  
});  
  
document.getElementById('modalForm').addEventListener('submit', function (event) {  
 event.preventDefault();  
  
 const serviceItemId = document.getElementById('serviceItemId').value;  
 const formData = {  
 name: document.getElementById('name\_edit').value,  
 price\_low: document.getElementById('price\_low\_edit').value,  
 price\_medium: document.getElementById('price\_medium\_edit').value,  
 price\_high: document.getElementById('price\_high\_edit').value  
 };  
  
 fetch(`/admin/service-items/${serviceItemId}`, {  
 method: 'PUT',  
 headers: {  
 'Content-Type': 'application/json',  
 'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').content  
 },  
 body: JSON.stringify(formData)  
 })  
 .then(response => response.json())  
 .then(data => {  
 console.log(data);  
 closeModal(modal);  
 alert('Редактирование было успешно выполнено');  
 location.reload();  
 })  
 .catch(error => {  
 console.error('Ошибка при обновлении элемента:', error);  
 alert('Произошла ошибка при обновлении элемента');  
 });  
});  
  
btnCloseModal.addEventListener('click', () => {  
 closeModal(modal);  
});  
  
handleModalClose(modal);

app.js – файл для работы элементов интерфейсов на главной странице

import {openModal, closeModal, handleModalClose} from './components/modal-functions.js';  
  
const buttonsAuthRegister = document.querySelectorAll('[data-modal]')  
  
buttonsAuthRegister.forEach(button => {  
 button.addEventListener('click', () => {  
 const modalId = button.getAttribute('data-target')  
 const modal = document.getElementById(modalId)  
 openModal(modal)  
 })  
})  
  
const btnRegister = document.getElementById('btn-register')  
const btnAuth = document.getElementById('btn-auth')  
  
btnRegister.addEventListener('click', () => {  
 closeModal(modalAuth)  
 setTimeout(() => openModal(modalRegister), 300)  
})  
  
btnAuth.addEventListener('click', () => {  
 closeModal(modalRegister)  
 setTimeout(() => openModal(modalAuth), 300)  
})  
  
const modalRegister = document.getElementById('modal-register')  
const closeModalRegister = document.getElementById('close-register')  
  
const modalAuth = document.getElementById('modal-auth')  
const closeModalAuth = document.getElementById('close-auth')  
  
closeModalAuth.addEventListener('click', () => {  
 closeModal(modalAuth)  
})  
  
closeModalRegister.addEventListener('click', () => {  
 closeModal(modalRegister)  
})  
  
handleModalClose(modalAuth)  
handleModalClose(modalRegister)

accordion.js – компонент для аккордеона

document.addEventListener("DOMContentLoaded", function() {  
 const accordionItemHeaders = document.querySelectorAll(".faq\_\_head");  
 const accordionItemBodies = document.querySelectorAll(".faq\_\_answer");  
  
 accordionItemHeaders[0].classList.add("faq\_\_head--active");  
 accordionItemBodies[0].style.maxHeight = accordionItemBodies[0].scrollHeight + "px";  
 accordionItemBodies[0].style.marginBottom = "15px";  
  
 accordionItemHeaders.forEach((accordionItemHeader, index) => {  
 accordionItemHeader.addEventListener("click", event => {  
 const currentlyActiveAccordionItemHeader = document.querySelector(".faq\_\_head.faq\_\_head--active");  
 if (currentlyActiveAccordionItemHeader && currentlyActiveAccordionItemHeader !== accordionItemHeader) {  
 currentlyActiveAccordionItemHeader.classList.toggle("faq\_\_head--active");  
 const currentlyActiveAccordionItemBody = currentlyActiveAccordionItemHeader.nextElementSibling;  
 currentlyActiveAccordionItemBody.style.maxHeight = 0;  
 currentlyActiveAccordionItemBody.style.marginBottom = 0;  
 }  
 accordionItemHeader.classList.toggle("faq\_\_head--active");  
 const accordionItemBody = accordionItemHeader.nextElementSibling;  
 if (accordionItemHeader.classList.contains("faq\_\_head--active")) {  
 accordionItemBody.style.maxHeight = accordionItemBody.scrollHeight + "px";  
 accordionItemBody.style.marginBottom = "15px";  
 } else {  
 accordionItemBody.style.maxHeight = 0;  
 accordionItemBody.style.marginBottom = 0;  
 }  
 });  
 });  
});

captcha.js – файл для работы с капчей

function refreshCaptcha() {  
 const captchaImage = document.getElementById('captchaImage');  
 const generateCaptchaURL = '/generate-captcha';  
  
 fetch(generateCaptchaURL)  
 .then(response => {  
 if (!response.ok) {  
 throw new Error('Ошибка при обновлении капчи');  
 }  
 return response.blob();  
 })  
 .then(blob => {  
 captchaImage.src = URL.createObjectURL(blob);  
 })  
 .catch(error => {  
 console.error(error);  
 alert('Ошибка при обновлении капчи');  
 });  
}  
  
document.getElementById('refreshCaptcha').addEventListener('click', refreshCaptcha);  
  
document.getElementById('formAuth').addEventListener('submit', function () {  
 event.preventDefault();  
 const captchaInput = document.getElementById('captcha');  
 const captchaValue = captchaInput.value;  
  
 fetch('/validate-captcha', {  
 method: 'POST',  
 headers: {  
 'Content-Type': 'application/json',  
 'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').content  
 },  
 body: JSON.stringify({captcha: captchaValue})  
 })  
 .then(response => response.json())  
 .then(data => {  
 if (data.success) {  
 document.getElementById('formAuth').submit();  
 } else {  
 const captchaError = document.getElementById('captchaError');  
 captchaError.classList.add('error--active');  
 refreshCaptcha();  
  
 captchaInput.value = '';  
 setTimeout(() => {  
 captchaError.classList.remove('error--active');  
 }, 4000);  
 }  
 })  
 .catch(error => {  
 console.error(error);  
 });  
});

menu.js – файл для работы с меню на сайте

const body = document.body;  
const menuBtn = document.querySelector('.menu-btn');  
const header = document.querySelector('.header\_\_menu');  
const anchors = document.querySelectorAll('a[href\*="#"]');  
  
menuBtn.addEventListener('click', (event) => {  
 event.preventDefault();  
 body.classList.toggle('body--active');  
 header.classList.toggle('active');  
 menuBtn.classList.toggle('active');  
 menuBtn.blur();  
});  
  
function scrollToTarget(targetId) {  
 const targetSection = document.getElementById(targetId);  
 if (targetSection) {  
 body.classList.remove('body--active');  
 header.classList.remove('active');  
 menuBtn.classList.remove('active');  
 setTimeout(() => {  
 let targetOffset;  
 if (targetId === 'testimonials') {  
 targetOffset = targetSection.offsetTop;  
 } else {  
 targetOffset = targetSection.offsetTop - 30;  
 }  
 window.scrollTo({top: targetOffset, behavior: 'smooth'});  
 }, 400);  
 }  
}  
  
function handleAnchorClick(event) {  
 event.preventDefault();  
 const targetId = this.getAttribute('href').split('#')[1];  
 scrollToTarget(targetId);  
}  
  
for (const anchor of anchors) {  
 anchor.addEventListener('click', handleAnchorClick);  
 anchor.addEventListener('touchstart', handleAnchorClick, {passive: true});  
}  
  
const menuLinks = document.querySelectorAll('.header\_\_link');  
menuLinks.forEach((menuLink) => {  
 menuLink.addEventListener('click', handleAnchorClick);  
 menuLink.addEventListener('touchstart', handleAnchorClick, {passive: true});  
});

modal-functions.js – компонент для работы с модальными окнами на клиенте

const closeModal = (modal) => {  
 document.body.classList.remove('body--active');  
 modal.classList.remove('modal--active');  
};  
  
const openModal = (modal) => {  
 document.body.classList.add('body--active');  
 modal.classList.add('modal--active');  
};  
  
const handleModalClose = (modal) => {  
 document.addEventListener('keyup', (e) => {  
 if (e.key === 'Escape') {  
 closeModal(modal);  
 }  
 });  
};  
  
export {closeModal, openModal, handleModalClose};

balance.js – файл для работы с балансом на клиенте

document.addEventListener('DOMContentLoaded', async function () {  
 const depositForm = document.querySelector('#deposit-form');  
 depositForm.addEventListener('submit', async function (event) {  
 event.preventDefault();  
 const formData = new FormData(depositForm);  
  
 try {  
 const response = await fetch(depositForm.action, {  
 method: 'POST',  
 body: formData,  
 headers: {  
 'X-CSRF-TOKEN': '{{ csrf\_token() }}'  
 }  
 });  
 const data = await response.json();  
 alert('Баланс успешно пополнен на ' + data.amount + ' ₽');  
 location.reload();  
 } catch (error) {  
 console.error('Ошибка:', error);  
 }  
 });  
});

new-order.js – бизнес логика для работы с созданием заказа на клиенте

const orderItems = document.querySelectorAll('.order\_\_item');  
const serviceButtons = document.querySelectorAll('.order\_\_services-btn');  
const variantSelect = document.querySelector('.order\_\_variants-select');  
const quantityInput = document.getElementById('quantity');  
const totalPriceInput = document.getElementById('total\_price');  
const orderButton = document.querySelector('.order\_\_btn');  
  
function updatePricesInSelect(serviceId) {  
 const servicePrices = document.querySelector(`.order\_\_variants-items[data-service-id="${serviceId}"]`);  
 if (servicePrices) {  
 const prices = servicePrices.querySelectorAll('.order\_\_variants-btn');  
 variantSelect.innerHTML = '';  
 prices.forEach(priceBtn => {  
 const option = document.createElement('option');  
 option.value = priceBtn.getAttribute('data-price');  
 option.textContent = priceBtn.textContent.trim();  
 variantSelect.appendChild(option);  
 });  
 }  
}  
  
function updateTotalPrice() {  
 const activeServiceList = document.querySelector('.order\_\_services-list[style\*="display: flex"]');  
 if (!activeServiceList) return;  
  
 const selectedServiceId = activeServiceList.getAttribute('data-service-id');  
 const activeServiceButton = activeServiceList.querySelector('.order\_\_services-btn.active');  
 if (!activeServiceButton) return;  
  
 const selectedServiceItemId = activeServiceButton.getAttribute('data-item-id');  
 const selectedServicePrice = parseFloat(activeServiceButton.getAttribute('data-price-high'));  
 const quantity = parseFloat(quantityInput.value);  
 const totalPrice = selectedServicePrice \* quantity;  
  
 totalPriceInput.value = totalPrice.toFixed(2) + '₽';  
 orderButton.textContent = `Создать заказ — ${totalPrice.toFixed(2)}₽`;  
 document.querySelector('input[name="service\_id"]').value = selectedServiceId;  
 document.querySelector('input[name="service\_item\_id"]').value = selectedServiceItemId;  
}  
  
if (orderItems.length > 0) {  
 orderItems[0].classList.add('active');  
 const firstServiceId = orderItems[0].getAttribute('data-service-id');  
 const firstServiceList = document.querySelector(`.order\_\_services-list[data-service-id="${firstServiceId}"]`);  
 if (firstServiceList) {  
 firstServiceList.style.display = 'flex';  
 updatePricesInSelect(firstServiceId);  
 }  
  
 orderItems.forEach(function (item) {  
 item.addEventListener('click', function () {  
 const serviceId = this.getAttribute('data-service-id');  
 const serviceList = document.querySelector(`.order\_\_services-list[data-service-id="${serviceId}"]`);  
  
 orderItems.forEach(function (item) {  
 item.classList.remove('active');  
 });  
 this.classList.add('active');  
  
 serviceButtons.forEach(function (btn) {  
 btn.classList.remove('active');  
 });  
  
 serviceButtons.forEach(function (btn) {  
 if (btn.getAttribute('data-service-id') === serviceId) {  
 btn.classList.add('active');  
 }  
 });  
  
 document.querySelectorAll('.order\_\_services-list').forEach(function (list) {  
 if (list !== serviceList) {  
 list.style.display = 'none';  
 }  
 });  
  
 if (serviceList) {  
 serviceList.style.display = 'flex';  
  
 const firstButton = serviceList.querySelector('.order\_\_services-btn');  
 if (firstButton) {  
 firstButton.classList.add('active');  
 }  
 }  
  
 updatePricesInSelect(serviceId);  
 });  
 });  
}  
  
if (serviceButtons.length > 0) {  
 serviceButtons.forEach(function (button) {  
 button.addEventListener('click', function () {  
 const serviceId = this.getAttribute('data-service-id');  
 const serviceList = document.querySelector(`.order\_\_services-list[data-service-id="${serviceId}"]`);  
  
 serviceButtons.forEach(function (btn) {  
 btn.classList.remove('active');  
 });  
  
 this.classList.add('active');  
  
 if (serviceList) {  
 serviceList.style.display = 'flex';  
 }  
  
 updatePricesInSelect(serviceId);  
 if (quantityInput.value !== '') {  
 updateTotalPrice();  
 }  
 });  
 });  
}  
  
quantityInput.addEventListener('input', updateTotalPrice);  
  
const submitForm = document.getElementById('createOrder');  
submitForm.addEventListener('submit', async (event) => {  
 event.preventDefault();  
 const formData = new FormData(submitForm);  
  
 try {  
 const response = await fetch(submitForm.action, {  
 method: 'POST',  
 body: formData,  
 headers: {  
 'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').content  
 },  
 });  
  
 if (response.ok) {  
 window.location.href = 'http://127.0.0.1:8000/my-orders';  
 } else if (response.status === 400) {  
 const responseData = await response.json();  
 alert(responseData.message);  
 } else {  
 console.error('Ошибка при создании заказа:', response.statusText);  
 }  
 } catch (error) {  
 console.error('Ошибка при выполнении запроса:', error.message);  
 }  
});

edit.js – файл для работы с редактированием пользователя

import {validateLogin, validateEmail, validateName} from "./validation.js";  
  
document.addEventListener('DOMContentLoaded', function () {  
 const updateButton = document.getElementById('updateButton');  
 const loginInput = document.getElementById('login');  
 const nameInput = document.getElementById('name');  
 const emailInput = document.getElementById('email');  
 const errorLoginCheck = document.getElementById('loginCheckError');  
 const errorLoginSymbols = document.getElementById('loginError');  
 const errorLoginLength = document.getElementById('loginLengthError');  
 const errorLoginMax = document.getElementById('loginMaxError');  
 const errorNameSymbols = document.getElementById('nameError');  
 const errorNameMin = document.getElementById('nameMinError');  
 const errorNameMax = document.getElementById('nameMaxError');  
 const errorEmailCheck = document.getElementById('emailCheckError');  
 const errorEmailSymbols = document.getElementById('emailErrorParameters');  
 const errorEmailLength = document.getElementById('emailLengthError');  
 const userId = window.location.pathname.split('/').slice(-2, -1)[0];  
  
 updateButton.addEventListener('click', async function (event) {  
 event.preventDefault();  
 const newLogin = loginInput.value.trim();  
 const newEmail = emailInput.value.trim();  
 const newName = nameInput.value.trim();  
  
 const isLoginValid = await validateLogin(newLogin, errorLoginCheck, errorLoginSymbols, errorLoginLength, errorLoginMax, 'edit', userId);  
 const isEmailValid = await validateEmail(newEmail, errorEmailCheck, errorEmailSymbols, errorEmailLength, 'edit', userId);  
 const isNameValid = validateName(newName, errorNameSymbols, errorNameMin, errorNameMax);  
  
 if (isLoginValid && isEmailValid && isNameValid) {  
 document.getElementById('updateForm').submit();  
 }  
 });  
});

search.js – файл для поиска пользователей в системе

import {openModal} from '../components/modal-functions.js';  
  
export function handleSearchInput() {  
 let searchTimer;  
  
 const searchForm = document.getElementById('searchForm');  
 searchForm.addEventListener('input', function () {  
 clearTimeout(searchTimer);  
 searchTimer = setTimeout(() => {  
 const formData = new FormData(this);  
 const searchText = formData.get('query').trim();  
  
 if (!searchText) {  
 fetch(`/admin/users/all`)  
 .then(response => {  
 if (!response.ok) {  
 throw new Error('Network response was not ok');  
 }  
 return response.json();  
 })  
 .then(data => {  
 displayUsers(data.users);  
 if (data.users.length === 0) {  
 showNoResultsMessage();  
 }  
 })  
 .catch(error => {  
 console.error('There has been a problem with your fetch operation:', error);  
 const userListContainer = document.getElementById('userListContainer');  
 userListContainer.innerHTML = `<p class="error">Что-то пошло не так при загрузке данных</p>`;  
 });  
 } else {  
 fetch(`/admin/users/search?${new URLSearchParams(formData).toString()}`)  
 .then(response => {  
 if (!response.ok) {  
 throw new Error('Network response was not ok');  
 }  
 return response.json();  
 })  
 .then(data => {  
 displayUsers(data.users);  
 if (data.users.length === 0) {  
 showNoResultsMessage();  
 }  
 })  
 .catch(error => {  
 console.error('There has been a problem with your fetch operation:', error);  
 const userListContainer = document.getElementById('userListContainer');  
 userListContainer.innerHTML = `<p class="error">Что-то пошло не так при загрузке данных</p>`;  
 });  
 }  
 }, 600);  
 });  
}  
export function displayUsers(users) {  
 const userListContainer = document.getElementById('userListContainer');  
 const currentUser = {id: 1};  
 const newContent = users.map(user => `  
 <li class="admin-users\_\_item" data-user-id="${user.id}">  
 <p class="admin-users\_\_login"><span>${user.login}</span></p>  
 <a class="admin-users\_\_edit" href="/admin/users/${user.id}/edit">  
 Редактировать  
 </a>  
 <div class="admin-users\_\_content">  
 <p>Имя: ${user.name}</p>  
 <p>Почта: ${user.email}</p>  
 <p>Роль: ${user.role}</p>  
 </div>  
 <div class="admin-users\_\_actions">  
 ${user.id !== currentUser.id ? `  
 <button class="admin-users\_\_action admin-users\_\_action--reset" type="button">Сбросить пароль</button>  
 <button class="admin-users\_\_action admin-users\_\_action--delete" type="button" data-user-id="${user.id}">  
 Удалить аккаунт  
 </button>  
 ` : `  
 <p class="admin-users\_\_action admin-users\_\_action--long">¯\\\_(ツ)\_/¯ не делай глупых решений</p>  
 `}  
 </div>  
 </li>`  
 ).join('');  
  
 userListContainer.innerHTML = `${newContent}`;  
 const modalDelete = document.getElementById('modalDeleteUser');  
 const deleteButtons = document.querySelectorAll('.admin-users\_\_action--delete');  
 deleteButtons.forEach(button => {  
 button.addEventListener('click', event => {  
 const userId = event.currentTarget.dataset.userId;  
 console.log('User Id: ', userId);  
 if (userId) {  
 loginDeleteUser.textContent = event.currentTarget.closest('.admin-users\_\_item').querySelector('.admin-users\_\_login').textContent;  
 modalDelete.dataset.userId = userId;  
 openModal(modalDelete);  
 } else {  
 console.error('Ошибка: ID юзера не найден');  
 }  
 });  
 });  
}  
  
export function showNoResultsMessage() {  
 const userListContainer = document.getElementById('userListContainer');  
 userListContainer.innerHTML = `<p class="no-results-message">¯\\\_(ツ)\_/¯ Ничего не было найдено</p>`;  
}

transactions.js – файл для работы с транзакциями у пользователя на клиенте

import {closeModal, handleModalClose, openModal} from '../components/modal-functions.js';  
  
const cancelButtons = document.querySelectorAll('.panel-balance\_\_btn.cancel-payment-btn');  
const modal = document.getElementById('modalCancelPayment');  
const modalClose = document.getElementById('modalCancel');  
const modalConfirm = document.getElementById('modalConfirm');  
  
let transactionIdToCancel;  
cancelButtons.forEach(button => {  
 button.addEventListener('click', () => {  
 const transactionId = button.getAttribute('data-transaction-id');  
 const userLogin = button.closest('tr').querySelector('td:nth-child(5)').textContent;  
 transactionIdToCancel = transactionId;  
 const loginUserSpan = modal.querySelector('#loginUser');  
 loginUserSpan.textContent = userLogin;  
 openModal(modal);  
 });  
});  
modalClose.addEventListener('click', () => {  
 closeModal(modal);  
});  
modalConfirm.addEventListener('click', async () => {  
 console.log('transactionIdToCancel:', transactionIdToCancel);  
 try {  
 const url = `/admin/transactions/${transactionIdToCancel}/cancel`;  
 console.log('URL:', url);  
 const response = await fetch(url, {  
 method: 'PUT',  
 headers: {  
 'Content-Type': 'application/json',  
 'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').getAttribute('content')  
 },  
 });  
  
 if (response.ok) {  
 const responseData = await response.json();  
 closeModal(modal);  
 const date = new Date(responseData.refundTransaction.created\_at);  
 const formattedDate = date.toLocaleDateString('en-US', {day: 'numeric', month: 'short', year: 'numeric'});  
 const amount = Number(responseData.refundTransaction.amount);  
 const formattedAmount = amount.toLocaleString('ru-RU', {style: 'currency', currency: 'RUB'});  
 const newRow = document.createElement('tr');  
 newRow.classList.add('panel-balance\_\_head', 'panel-balance\_\_item', 'admin', 'refund');  
 newRow.innerHTML = `  
 <td>${responseData.refundTransaction.id}</td>  
 <td class="panel-balance\_\_type refund">Возврат средств</td>  
 <td>${formattedAmount}</td>  
 <td>${formattedDate}</td>  
 <td>${responseData.refundTransaction.user}</td>  
 <td></td>  
 `;  
 const tbody = document.querySelector('.panel-balance\_\_body');  
 tbody.insertBefore(newRow, tbody.firstChild);  
 } else {  
 console.error('Ошибка при отмене платежа');  
 }  
 } catch (error) {  
 console.error('Ошибка при отмене платежа:', error);  
 }  
});  
handleModalClose(modal);

validation.js – файл для работы с валидацией данных пользователя

export async function validateLogin(loginValue, errorCheck, errorSymbols, errorLength, errorMax, actionType, userId) {  
 let valid = true;  
 try {  
 let url = actionType === 'edit' ? '/check-login-edit' : '/check-login-create';  
  
 const response = await fetch(url, {  
 method: 'POST',  
 headers: {  
 'Content-Type': 'application/json',  
 'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').getAttribute('content')  
 },  
 body: JSON.stringify({login: loginValue, userId: userId})  
 });  
 const data = await response.json();  
 if (data.exists) {  
 errorCheck.classList.add('error--active');  
 valid = false;  
 } else {  
 errorCheck.classList.remove('error--active');  
 }  
 const loginRegex = /^[a-zA-Z0-9\_]+$/;  
 if (!loginRegex.test(loginValue) || /[А-я]/.test(loginValue)) {  
 errorSymbols.classList.add('error--active');  
 valid = false;  
 } else {  
 errorSymbols.classList.remove('error--active');  
 }  
 if (loginValue.length < 5) {  
 errorLength.classList.add('error--active');  
 valid = false;  
 } else {  
 errorLength.classList.remove('error--active');  
 }  
 if (loginValue.length > 60) {  
 errorMax.classList.add('error--active');  
 valid = false;  
 } else {  
 errorMax.classList.remove('error--active');  
 }  
 } catch (error) {  
 console.error('Error:', error);  
 valid = false;  
 }  
 return valid;  
}  
export async function validateEmail(emailValue, errorCheck, errorSymbols, errorLength, actionType, userId) {  
 let valid = true;  
 try {  
 let url = actionType === 'edit' ? '/check-email-edit' : '/check-email-create';  
 const response = await fetch(url, {  
 method: 'POST',  
 headers: {  
 'Content-Type': 'application/json',  
 'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').getAttribute('content')  
 },  
 body: JSON.stringify({email: emailValue, userId: userId})  
 });  
 const data = await response.json();  
 if (data.exists) {  
 errorCheck.classList.add('error--active');  
 valid = false;  
 } else {  
 errorCheck.classList.remove('error--active');  
 }  
 const emailRegex = /^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$/;  
 if (!emailRegex.test(emailValue)) {  
 errorSymbols.classList.add('error--active');  
 valid = false;  
 } else {  
 errorSymbols.classList.remove('error--active');  
 }  
 if (emailValue.length > 80) {  
 errorLength.classList.add('error--active');  
 valid = false;  
 } else {  
 errorLength.classList.remove('error--active');  
 }  
 } catch (error) {  
 console.error('Error:', error);  
 valid = false;  
 }  
 return valid;  
}  
export function validateName(nameValue, errorSymbols, errorMin, errorMax) {  
 let valid = true;  
 const nameRegex = /^[A-Za-zА-я\-]+$/;  
 if (!nameRegex.test(nameValue)) {  
 errorSymbols.classList.add('error--active');  
 valid = false;  
 } else {  
 errorSymbols.classList.remove('error--active');  
 }  
 if (nameValue.length < 2) {  
 errorMin.classList.add('error--active');  
 valid = false;  
 } else {  
 errorMin.classList.remove('error--active');  
 }  
 if (nameValue.length > 50) {  
 errorMax.classList.add('error--active');  
 valid = false;  
 } else {  
 errorMax.classList.remove('error--active');  
 }  
 return valid;  
}  
export function validatePassword(passwordValue, errorSymbols, errorLength, errorMax) {  
 let valid = true;  
 const passwordRegex = /[А-я]/;  
 if (passwordRegex.test(passwordValue)) {  
 errorSymbols.classList.add('error--active');  
 valid = false;  
 } else {  
 errorSymbols.classList.remove('error--active');  
 }  
 if (passwordValue.length < 8) {  
 errorLength.classList.add('error--active');  
 valid = false;  
 } else {  
 errorLength.classList.remove('error--active');  
 }  
 if (passwordValue.length > 60) {  
 errorMax.classList.add('error--active');  
 valid = false;  
 } else {  
 errorMax.classList.remove('error--active');  
 }  
 return valid;  
}