

МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Казанский национальный исследовательский технологический университет»**

**(ФГБОУ ВО «КНИТУ»)**

Кафедра Интеллектуальных систем и управления информационными ресурсами

Направление: 02.03.03 Математическое обеспечение и администрирование информационных систем

Группа: 4311-21

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

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Уровень образования: Бакалавр

Вид ВКР: проектный

Тема: Разработка веб-сервиса для онлайн оплаты чаевых с использованием QR-кода.

Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Ахметшин Д.А.)

Нормоконтролер\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Антонова П.В.)

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Титовцев А.С.)

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Салахов А.Р.)

Казань 2025г

Оглавление

[ВВЕДЕНИЕ 3](#_heading=h.z6p9r77y3v2b)

[ЗАДАНИЕ 5](#_heading=h.nsd6yaxew4l9)

[ГЛАВА I. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ ВЕБ-СЕРВИСОВ ОНЛАЙН ОПЛАТЫ 7](#_heading=h.o57tghnt6vz2)

[1. Понятие и преимущества онлайн-оплаты чаевых 7](#_heading=h.mi5wgjj6o1st)

[1.1. Технологии веб-разработки: обзор и сравнение 8](#_heading=h.pbz1w5qkwemy)

[1.2. Анализ существующих решений для сбора чаевых 12](#_heading=h.v2xm8cjzhjrj)

[ГЛАВА II. РАЗРАБОТКА ВЕБ-ОРИЕНТИРОВАННОГО СИСТЕМЫ СТРОИТЕЛЬНОЙ КОМПАНИИ 14](#_heading=h.ebygisriok21)

[2. Постановка задачи и требования к проекту 14](#_heading=h.xbw7z8wx85z1)

[2.1.1. Функциональные требования 14](#_heading=h.49ao6vs3jrim)

[2.1.2. Нефункциональные требования 15](#_heading=h.rjrwz05ulzst)

[2.2.1. Архитектура веб-сервиса 16](#_heading=h.x62693z3k8qh)

[2.2.2. Выбор стеков технологий 17](#_heading=h.65s8iv9ooiuv)

[2.2.3. Общая схема архитектуры 18](#_heading=h.rqk3qkt7raz)

[2.3.1. Проектирование БД 20](#_heading=h.cmkxu0k9jmk4)

[2.3.2. Структура таблиц и их взаимосвязи 21](#_heading=h.f60l04pk3b0h)

[2.4.1. Реализация серверной части 23](#_heading=h.35f6onwdyusq)

[2.4.2. API-эндпоинты 28](#_heading=h.o7efj4tupgik)

[2.4.3. Реализация клиентской части 30](#_heading=h.ip9vd7q35ofl)

[2.4.4. Основные Vue-компоненты 30](#_heading=h.xqpl7e1m8oi0)

[2.4.5. Роутинг и взаимодействие с API 33](#_heading=h.pdm9jntzrb00)

[2.5. Механизм генерации QR-кода и ссылки 35](#_heading=h.talelhcw1lds)

[2.6. Средства безопасности веб-сервиса 36](#_heading=h.5a3m4dfhya9n)

[ЗАКЛЮЧЕНИЕ 38](#_heading=h.m8f7hhpk153y)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 40](#_heading=h.157ooycvlx2n)

[ПРИЛОЖЕНИЕ 1 42](#_heading=h.2z4ghf7lnxx8)

[ПРИЛОЖЕНИЕ 2 57](#_heading=h.w6po85bjhzhc)

# ВВЕДЕНИЕ

Развитие цифровых технологий и активное внедрение интернет-сервисов в повседневную жизнь обусловили рост популярности онлайн-оплаты в различных сферах деятельности. Одной из таких сфер является обслуживание, где возможность оставить чаевые с помощью онлайн-инструментов становится всё более востребованной. Учитывая переход общества к безналичным формам расчетов, разработка веб-сервисов, обеспечивающих удобную и безопасную оплату чаевых, представляет собой актуальную задачу, соответствующую современным технологическим трендам и требованиям рынка.

Практическая значимость проекта заключается в создании удобного и функционального инструмента для поддержки сотрудников сферы услуг, позволяющего посетителям легко выражать благодарность за предоставленный сервис. Такой подход не только облегчает процесс поощрения персонала, но и способствует формированию положительного пользовательского опыта, повышает лояльность клиентов, а также способствует прозрачности и автоматизации расчётов.

Одним из ключевых аспектов разработки подобного веб-сервиса является обеспечение интуитивно понятного интерфейса, адаптированного под различные устройства, и реализация устойчивого механизма сбора и обработки платежей. Также большое значение имеет визуальная составляющая, архитектура взаимодействия между компонентами системы и соблюдение требований информационной безопасности.

Целью настоящего исследования является разработка веб-сервиса для онлайн оплаты чаевых с использованием QR-кода, обладающего удобным пользовательским интерфейсом и реализованного на основе современных технологий веб-разработки.

Для достижения поставленной цели необходимо решить следующие задачи:

* провести анализ аналогичных решений, представленных на рынке онлайн-платежей;
* определить функциональные и нефункциональные требования к создаваемому веб-сервису;
* разработать информационную архитектуру сервиса и спроектировать базу данных;
* реализовать клиентскую часть интерфейса с использованием HTML, CSS и фреймворка Vue.js;
* создать серверную часть веб-сервиса с применением PHP и MySQL;
* внедрить механизм генерации уникальных QR-кодов для каждого получателя чаевых;
* протестировать систему на корректность функционирования и соответствие требованиям.

Результатом работы станет веб-сервис, позволяющий пользователям переводить чаевые онлайн с использованием мобильных устройств и QR-кода, а получателям — отслеживать поступления в личном кабинете.

Кафедра ИСУИР «УТВЕРЖДАЮ»

Направление 02.03.03 Зав.кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность -- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

Группа 4311-21

# ЗАДАНИЕ

**на выпускную квалификационную работу студента** Салахова А.Р.

Тема: Разработка веб-сервиса для онлайн оплаты чаевых с использованием QR-кода

Срок представления работы к защите «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

Цель, задачи и исходные данные работы:

Цель: создание современного и удобного веб-сервиса, который будет отвечать всем потребностям пользователей

Задачи:

1. Анализ предметной области.

2. Построение диаграммы вариантов использования, диаграмм взаимодействия.

3. Представление схемы базы данных в виде ER-модели.

4. Разработка и представление информационной системы.

Задание по разделам работы:

1. Изучить предметную область, провести обзор аналогичных решений
2. Спроектировать и разработать веб-ориентированную информационную систему строительной компании

Содержание графической части (иллюстрированного материала):

Графическую часть выполнить в форме презентации

Консультанты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Титовцев А.С. )

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Салахов А.Р. )

ЛИСТ НОРМОКОНТРОЛЕРА

1. Лист является обязательным приложением к пояснительной записке дипломного (курсового) проекта.
2. Нормоконтролер имеет право возвращать документацию без рассмотрения в случаях:

-нарушения установленной комплектности,

-отсутствия обязательных подписей,

-нечеткого выполнения текстового и графического материала.

1. Устранение ошибок, указанных нормоконтролером, обязательно.

ПЕРЕЧЕНЬ

замечаний и предложений нормоконтролера по дипломному (курсовому) проекту, студента

4311-21, А.Р. Салахова

(группа, инициалы, фамилия)

| Лист  (страница) | Условное обозна-  чение (код ошибок) | Содержание замечаний и предложений со ссылкой на нормативный документ, стандарт или типовую документацию |
| --- | --- | --- |
|  |  |  |

Дата\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Нормоконтролер\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия,инициалы)

# ГЛАВА I. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ ВЕБ-СЕРВИСОВ ОНЛАЙН ОПЛАТЫ

## 1. Понятие и преимущества онлайн-оплаты чаевых

Современные тенденции в области веб-технологий активно влияют на развитие способов осуществления различных финансовых транзакций. Одним из таких явлений является онлайн-оплата чаевых, которая становится всё более популярной благодаря удобству, доступности и высокой степени автоматизации процесса. Онлайн-оплата чаевых представляет собой механизм перевода денег от клиента (потребителя) к исполнителю (например, работнику ресторана, курьеру или сотруднику сервиса), который не требует непосредственного контакта или физического обмена денежных средств.

Преимущества использования онлайн-систем для оплаты чаевых проявляются в нескольких аспектах. Во-первых, это значительное удобство как для клиентов, так и для исполнителей. Процесс перевода чаевых осуществляется с помощью нескольких простых шагов через интернет, что исключает необходимость наличных расчетов, а также минимизирует время, которое клиент тратит на завершение транзакции.

Во-вторых, безопасность транзакций значительно возрастает при использовании онлайн-платежных систем. Электронные платежи сопровождаются процедурами защиты данных, что снижает вероятность финансовых потерь как для пользователей, так и для получателей чаевых. Использование шифрования и многократной аутентификации пользователей добавляет дополнительный уровень защиты.

В-третьих, онлайн-системы позволяют значительно упростить и ускорить учет чаевых, а также предоставить прозрачность транзакций. Клиенты могут отслеживать историю своих платежей, а исполнители – видеть поступления, что в свою очередь облегчает ведение финансовых отчетов и налоговых деклараций.

Кроме того, системы онлайн-оплаты чаевых обладают гибкостью. Возможность выбора суммы чаевых и различных вариантов процентных ставок (например, 7%, 10%, 12%) дает возможность клиентам легко адаптировать сумму перевода в зависимости от удовлетворенности услугами. Внедрение таких возможностей улучшает опыт взаимодействия с клиентами, обеспечивая им удобство и свободу выбора.

Не менее важным является и аспект доступности таких сервисов. Современные платформы для онлайн-оплаты чаевых работают на большинстве мобильных устройств, что позволяет пользователям оставаться мобильными и использовать систему в любое время и в любом месте. Это значительно повышает популярность онлайн-сервисов, так как большинство пользователей предпочитают совершать финансовые операции через мобильные приложения или сайты, которые доступны круглосуточно.

Таким образом, онлайн-оплата чаевых представляет собой эффективное, безопасное и удобное решение для многих сфер, связанных с обслуживанием клиентов. Учитывая рост популярности цифровых технологий и сервисов, можно сделать вывод, что использование таких решений в будущем станет стандартом, так как оно позволяет значительно улучшить качество обслуживания и повысить уровень комфорта как для клиентов, так и для поставщиков услуг.

## 1.1. Технологии веб-разработки: обзор и сравнение

Веб-разработка является многогранной областью, охватывающей широкий спектр технологий и инструментов, которые используются для создания функциональных, удобных и безопасных веб-приложений. Современная веб-разработка включает в себя как фронтенд, так и бэкенд компоненты, а также различные подходы к взаимодействию с базами данных и обеспечения безопасности. Важно отметить, что выбор технологий зависит от конкретных требований проекта, таких как функциональные возможности, производительность, безопасность и поддержка.

Фронтенд веб-разработки относится к созданию интерфейса, с которым взаимодействуют пользователи. Это визуальная и интерактивная часть веб-приложения, которая требует использования различных технологий для построения пользовательского интерфейса.

HTML, CSS и JavaScript – это основные технологии, которые использует каждый фронтенд-разработчик. HTML (Hypertext Markup Language) используется для разметки страниц, CSS (Cascading Style Sheets) — для стилизации, а JavaScript отвечает за динамичность и интерактивность страницы.

Тем не менее, для создания современных и сложных веб-приложений недостаточно лишь базовых технологий. В последние годы наибольшее распространение получили так называемые фреймворки и библиотеки, которые значительно упрощают процесс разработки и обеспечивают высокую производительность.

Одним из таких популярных инструментов является Vue.js. Это прогрессивный JavaScript-фреймворк, который позволяет строить интерактивные пользовательские интерфейсы. Vue.js отличается высокой гибкостью, простотой в изучении и внедрении, а также поддержкой реактивных данных, что делает его отличным выбором для разработки одностраничных приложений (SPA). Vue.js активно используется для создания интерфейсов с высокой динамичностью, например, в проектах, связанных с онлайн-оплатой или управлениями транзакциями.

Другими популярными фреймворками являются React и Angular. React является библиотекой для построения пользовательских интерфейсов от Facebook и получил широкое распространение благодаря своему компонентному подходу и высокой производительности. Angular, разработанный компанией Google, представляет собой более крупный фреймворк, который предоставляет широкий набор инструментов для построения веб-приложений, включая маршрутизацию, управление состоянием и другие встроенные функции.

Бэкенд-разработка отвечает за серверную часть веб-приложения, включая логику обработки данных, управление пользователями, выполнение транзакций и взаимодействие с базами данных. На бэкенде основное внимание уделяется обеспечению безопасности, производительности и масштабируемости.

Одной из наиболее распространенных технологий для серверной разработки является PHP, язык программирования, который активно используется в веб-разработке уже несколько десятилетий. PHP позволяет создавать динамичные веб-страницы и взаимодействовать с базами данных, такими как MySQL, с использованием SQL-запросов. Важно отметить, что PHP поддерживает создание REST API, что делает его подходящим для создания гибких и масштабируемых веб-сервисов, таких как система онлайн-оплаты чаевых.

Для разработки серверной логики также широко используются другие технологии, такие как Node.js и Python (Django, Flask). Node.js позволяет использовать JavaScript на серверной стороне, что упрощает разработку в случае использования одного языка на обоих концах приложения (клиентском и серверном). Python, в свою очередь, известен своей читаемостью и простотой, а фреймворки, такие как Django и Flask, предоставляют мощные средства для быстрого создания веб-приложений

Для хранения данных, связанных с пользователями, транзакциями и другими аспектами веб-приложений, необходимы базы данных. Веб-сервисы, которые требуют хранение данных о пользователях, сборах чаевых или заказах, часто используют реляционные базы данных, такие как MySQL или PostgreSQL.

MySQL является одной из самых популярных реляционных баз данных и предоставляет множество инструментов для работы с данными, обеспечения целостности и выполнения сложных SQL-запросов. Это идеальный выбор для приложений, которые требуют надежного хранения и манипулирования данными. В контексте онлайн-оплаты чаевых MySQL предоставляет простоту и надежность для хранения информации о пользователях и сборах.

Альтернативой реляционным базам данных являются NoSQL базы данных, такие как MongoDB, которые лучше подходят для приложений, где структура данных может изменяться динамически или при обработке больших объемов неструктурированных данных.

Для реализации некоторых функций, таких как генерация QR-кодов, необходимо интегрировать веб-сервис с внешними API. Примером такого сервиса является QR Server API, который используется для создания уникальных QR-кодов для каждой транзакции или сбора чаевых. Подобные интеграции позволяют значительно упростить и ускорить процесс разработки, делая сервис более функциональным и удобным для пользователей.

Для разработки веб-сервиса для онлайн-оплаты чаевых был выбран стек технологий, включающий Vue.js для фронтенда, PHP для бэкенда и MySQL для базы данных. Такой выбор обусловлен необходимостью обеспечить быстрое и удобное взаимодействие пользователя с интерфейсом, безопасность при обработке данных и транзакций, а также высокую производительность при хранении и обработке информации.

Можно сделать вывод, что выбор технологий для веб-разработки зависит от ряда факторов, таких как требования к функциональности, производительности, безопасности и удобству использования. Современные инструменты позволяют создавать гибкие и масштабируемые веб-приложения, обеспечивая высокий уровень удовлетворенности пользователей и удобство работы для разработчиков.

## 1.2. Анализ существующих решений для сбора чаевых

В последние годы на российском рынке появилось несколько решений, которые позволяют клиентам оставлять чаевые через интернет с помощью QR-кодов. В этом разделе будут рассмотрены три популярных сервиса: Спасибо Мир, CloudTips и ЧаЧаЧай. Каждый из них имеет свои особенности, преимущества и недостатки, которые влияют на их использование и популярность среди малого бизнеса.

Спасибо Мир — это сервис для сбора чаевых, который ориентирован на рестораны и кафе. Он позволяет пользователям оплачивать чаевые через QR-коды, которые можно разместить на столах или в меню заведений. Основные преимущества системы Мир заключаются в простоте использования и быстроте обработки транзакций. Клиенту достаточно отсканировать QR-код с помощью мобильного телефона, ввести сумму чаевых и подтвердить перевод через выбранный способ оплаты.

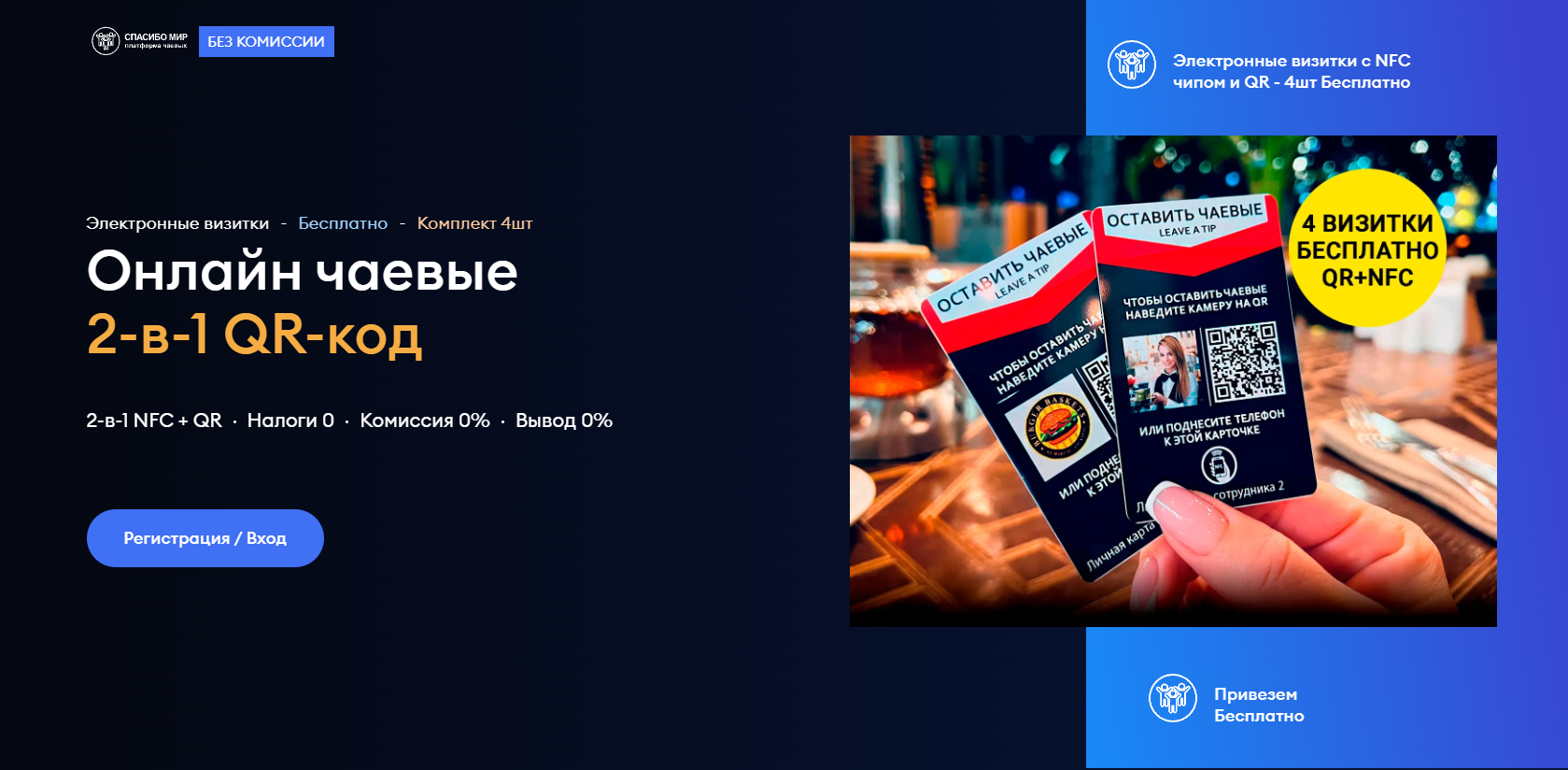


Рисунок 1 – Главная страница сервиса «Спасибо мир».

CloudTips — это облачная платформа для сбора чаевых, которая позволяет легко интегрировать ссылку для перевода чаевых на веб-страницах или в социальных сетях. Этот сервис ориентирован как на крупные компании, так и на индивидуальных предпринимателей. Особенность Cloud Tips заключается в гибкости, которая позволяет бизнесу настраивать процент чаевых, а также в простоте использования как для клиентов, так и для владельцев бизнеса.

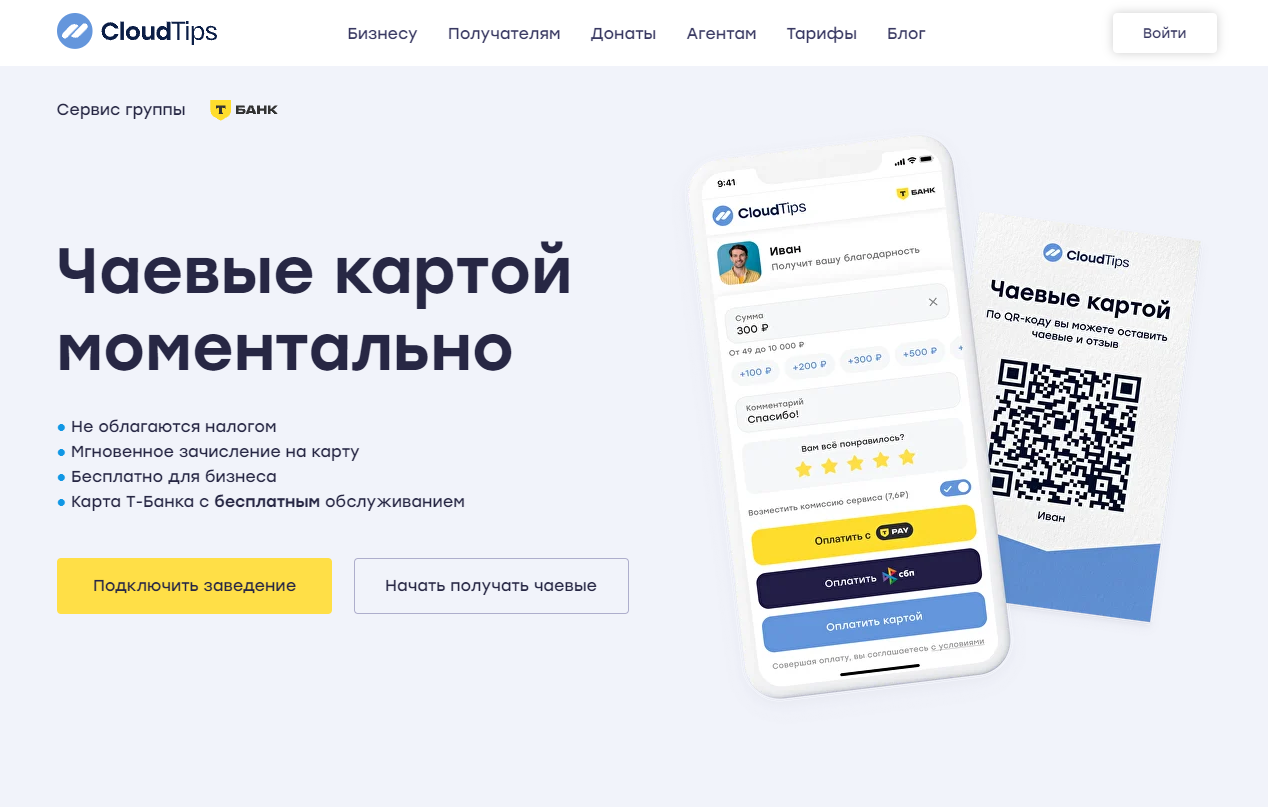


Рисунок 2 – Главная страница сервиса «CloudTips».

ЧаЧаЧай — это сервис, специально разработанный для ресторанного бизнеса, который позволяет пользователям оставлять чаевые с помощью QR-кодов. Этот сервис является удобным инструментом для малого бизнеса, так как не требует дополнительных затрат на оборудование и может быть использован с минимальными техническими навыками.

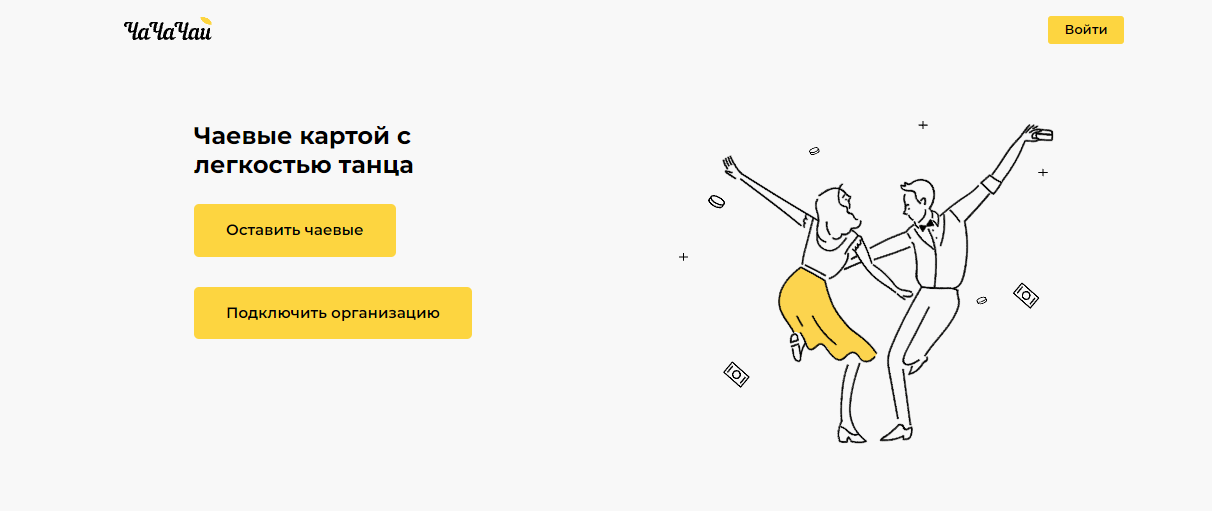


Рисунок 3 – Главная страница сервиса «ЧаЧаЧай».

# ГЛАВА II. РАЗРАБОТКА ВЕБ-ОРИЕНТИРОВАННОГО СИСТЕМЫ СТРОИТЕЛЬНОЙ КОМПАНИИ

## 2. Постановка задачи и требования к проекту

Проект заключается в разработке веб-сервиса для онлайн-оплаты чаевых с использованием QR-кодов. Задача состоит в создании интуитивно понятного и доступного инструмента для малых и средних бизнесов, позволяющего клиентам легко отправлять чаевые с помощью мобильных устройств, без необходимости использования наличных.

Этапность выполнения проекта:

1. Описание функциональных и пользовательских требований к сервису.
2. Создание серверной и клиентской частей приложения.
3. Проверка системы на всех этапах, исправление ошибок, внедрение в эксплуатацию.

## 2.1.1. Функциональные требования

Необходимо реализовать следующие функциональные требования:

* Пользователь может зарегистрироваться в системе, создать учетную запись, а затем войти в систему с использованием логина и пароля;
* Пользователь должен иметь возможность создать сбор чаевых с указанием суммы, процента чаевых и цели сбора;
* Для каждого сбора автоматически генерируются уникальная ссылка и QR-код для перевода чаевых;
* Пользователь может просматривать созданные сборы, редактировать их или удалять;
* Для каждого сбора должна быть предусмотрена возможность выбора суммы чаевых и отправки ее через доступные методы.

## 2.1.2. Нефункциональные требования

Нефункциональные требования описывают характеристики, которые определяют, как система будет выполнять свои задачи, обеспечивая высокое качество обслуживания и удовлетворение пользователей. Одним из важнейших аспектов является производительность системы. Важно, чтобы время отклика веб-сервиса было минимальным, не превышая 3 секунд, чтобы пользователи не испытывали задержек при совершении операций. Этот фактор непосредственно влияет на удобство использования и общую удовлетворенность пользователей.

Надежность — это еще один критически важный аспект. Система должна работать стабильно, обеспечивая бесперебойную работу при большом потоке пользователей. Для этого необходима регулярная проверка и резервное копирование данных, чтобы минимизировать риски потери информации и обеспечить восстановление системы в случае возникновения непредвиденных ситуаций. Важно также предусмотреть мониторинг всех ключевых процессов, чтобы своевременно выявлять и устранять возможные сбои.

Масштабируемость системы является основным требованием, так как с ростом числа пользователей и операций потребуется возможность расширять функционал. Архитектура веб-сервиса должна позволять добавлять новые функции и компоненты без необходимости кардинальной переработки системы, что обеспечит удобство и гибкость в дальнейшем развитии проекта.

Не менее важным аспектом является безопасность. Для защиты данных пользователей и предотвращения несанкционированного доступа система должна использовать современные методы шифрования, например, SSL для защищенной передачи данных. Также необходимо хранить пароли и другую чувствительную информацию в зашифрованном виде. Важно предусмотреть механизмы защиты от различных видов атак, таких как SQL-инъекции и XSS, чтобы гарантировать безопасность данных и защиту от потенциальных угроз.

Удобство использования веб-сервиса имеет огромное значение для его успешного применения. Интерфейс системы должен быть интуитивно понятным, с минимальным количеством шагов для выполнения основных операций. Все элементы интерфейса должны быть расположены логично и удобно, обеспечивая пользователям комфорт при взаимодействии с сервисом. Это также включает в себя обеспечение доступности функционала для разных категорий пользователей.

Кроссбраузерность системы обеспечивает ее корректную работу в популярных веб-браузерах, таких как Google Chrome, Mozilla Firefox, Safari и Microsoft Edge. Важно, чтобы веб-сервис был доступен на всех устройствах, включая смартфоны, планшеты и десктопы, с адаптивным дизайном, который подстраивается под различные разрешения экрана.

Наконец, интернационализация сервиса позволяет пользователям работать с системой на разных языках. Система должна поддерживать локализацию интерфейса на русском и английском языках, что обеспечит удобство для пользователей из разных регионов.

Все эти нефункциональные требования определяют качество и долговечность веб-сервиса, способствуя его успешной эксплуатации и высокой удовлетворенности пользователей.

## 2.2.1. Архитектура веб-сервиса

Архитектура веб-сервиса представляет собой структурное описание системы, которое охватывает ключевые компоненты, их взаимодействие и механизмы обработки данных. В контексте данного проекта архитектура будет сосредоточена на обеспечении легкости масштабируемости, надежности и производительности системы. Основной задачей является создание такого сервиса, который мог бы обрабатывать запросы пользователей, обеспечивать безопасность данных и поддерживать высокую доступность без значительных затрат на обслуживание.

## 2.2.2. Выбор стеков технологий

На этапе проектирования и реализации веб-сервиса особое внимание было уделено подбору технологий, необходимых для построения надежной и устойчивой архитектуры. Ключевым критерием выбора стало соответствие целям учебного проекта, а также доступность и простота внедрения без чрезмерного усложнения инфраструктуры.

Клиентская часть веб-приложения разрабатывалась с применением стандартных технологий фронтенда: HTML, CSS и JavaScript. Такое решение позволило создать интуитивно понятный и легкий пользовательский интерфейс, работающий во всех современных браузерах без дополнительных зависимостей. Структура и стилизация страниц реализованы с использованием адаптивной вёрстки, что обеспечивает корректное отображение интерфейса как на настольных, так и на мобильных устройствах.

На серверной стороне использовался язык PHP как одно из наиболее распространенных решений для веб-разработки. Его синтаксис прост в освоении и обеспечивает быструю реализацию базовой логики. PHP легко работает с формами, обрабатывает HTTP-запросы и позволяет динамически генерировать веб-страницы. Это делает его особенно удобным в учебных проектах, где важно сосредоточиться на логике приложения, не перегружая процесс сложными конфигурациями.

Для хранения и обработки данных использовалась реляционная система управления базами данных MySQL. Она обеспечивает высокую производительность, хорошо масштабируется и совместима с PHP без необходимости в дополнительных слоях адаптации. Структура базы данных спроектирована таким образом, чтобы обеспечить логичное хранение информации о пользователях, сессиях и транзакциях чаевых, что в свою очередь позволяет просто извлекать нужные данные и формировать страницы на их основе.

В качестве веб-сервера использовался Apache — стабильное и гибкое решение, широко применяемое в среде PHP-проектов. Он обеспечивает эффективную маршрутизацию запросов и удобную настройку окружения разработки. Также Apache без труда взаимодействует с MySQL и PHP, что делает его универсальной платформой для развертывания учебных веб-приложений.

Таким образом, выбранный стек — HTML, CSS, JavaScript на клиенте, в сочетании с PHP, MySQL и Apache на сервере — позволил выстроить простую, но в то же время функциональную архитектуру. Такое решение полностью соответствует целям проекта, обеспечивая стабильную работу системы и создавая хорошую основу для демонстрации её возможностей.

## 2.2.3. Общая схема архитектуры

Архитектура разрабатываемого веб-сервиса опирается на классическую клиент-серверную модель, которая широко применяется при создании современных веб-приложений. Такая структура обеспечивает чёткое разделение между пользовательским интерфейсом и серверной логикой, что делает систему более масштабируемой и удобной для поддержки.

Во взаимодействии компонентов задействованы два основных уровня: фронтенд и бэкенд. Клиентская часть (фронтенд) функционирует в браузере пользователя и отвечает за отображение визуального интерфейса, обработку событий и отправку запросов на сервер. Когда пользователь переходит по QR-коду или открывает ссылку на страницу чаевых, именно фронтенд формирует страницу, отображает сумму, имя получателя и предлагает возможные варианты действий.

Серверная часть (бэкенд), реализованная на языке PHP, принимает запросы, обрабатывает их и возвращает ответы в виде HTML-страниц или JSON-данных. Она взаимодействует с базой данных MySQL, где хранятся все необходимые сведения: информация о получателях чаевых, сгенерированных страницах, чеках и т.п. Сервер формирует уникальные страницы в соответствии с запрашиваемыми параметрами, подставляя нужные значения прямо в шаблон.

Apache выступает в качестве веб-сервера, принимающего входящие HTTP-запросы. Он направляет их в PHP-скрипты, которые, в свою очередь, обрабатывают бизнес-логику и запрашивают или обновляют данные в MySQL. Таким образом, Apache является связующим звеном между клиентской и серверной частями системы.

Архитектура была спроектирована с упором на простоту, надежность и предсказуемое поведение при демонстрации функционала. В рамках учебного проекта было важно не перегружать реализацию избыточной сложностью, а сосредоточиться на четком и логичном взаимодействии между слоями. Такой подход позволил добиться устойчивой работы сервиса и обеспечить понятную структуру взаимодействия между компонентами.

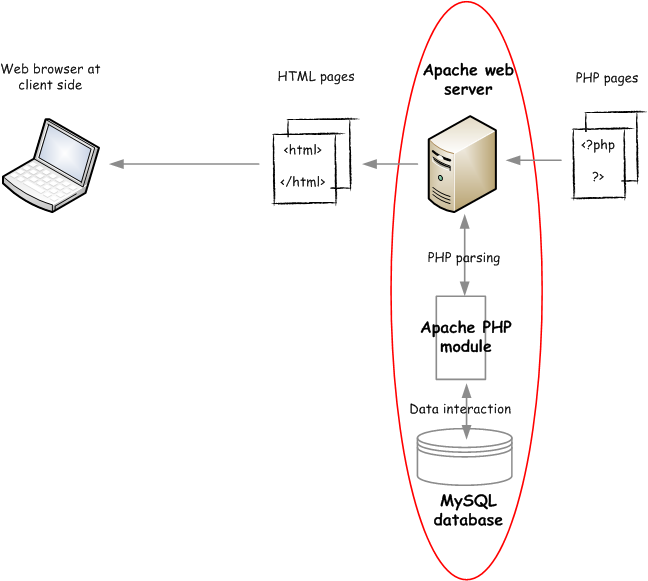


Рисунок 4 – Логическая схема архитектуры.

## 2.3.1. Проектирование БД

Проектирование базы данных является одним из ключевых этапов при создании любого веб-сервиса, так как от его корректности зависит надежность хранения информации, стабильность работы сервиса и удобство взаимодействия с данными. В рамках разработки учебного веб-сервиса для сбора чаевых посредством QR-кода было спроектировано логически обоснованное и структурированное хранилище данных, охватывающее все основные сущности, задействованные в бизнес-логике приложения.

Целью данного этапа стало создание устойчивой и расширяемой схемы базы данных, способной обеспечивать как хранение пользовательских данных, так и управление сборами, создаваемыми через интерфейс веб-приложения. При этом особое внимание уделялось соблюдению нормализации данных, организации связей между сущностями, а также внедрению ограничений целостности и индексирования для повышения производительности при обращении к таблицам.

Для построения базы данных был использован язык SQL в связке с системой управления базами данных MySQL, что позволило добиться совместимости с выбранным стеком серверных технологий. Архитектура базы данных разрабатывалась с прицелом на простоту поддержки и возможность масштабирования в будущем, если появится необходимость в расширении функциональности.

В последующих разделах будет подробно рассмотрена структура таблиц, описаны взаимосвязи между ними, объяснены реализованные ограничения и индексы, а также приведен SQL-скрипт для создания базы данных. Это позволит получить полное представление о логической модели и особенностях физической реализации хранилища данных.

## 2.3.2. Структура таблиц и их взаимосвязи

На этапе проектирования базы данных особое внимание было уделено выделению ключевых сущностей, отражающих основную бизнес-логику веб-сервиса. Для корректного хранения и обработки информации были разработаны две основные таблицы: users и collections, между которыми установлены логические связи. Такая структура обеспечивает простоту в управлении данными, позволяет масштабировать сервис и адаптировать его под новые требования при необходимости.

Таблица users предназначена для хранения информации о зарегистрированных пользователях, которые создают сборы чаевых. Каждый пользователь идентифицируется по уникальному id, а также по адресу электронной почты, используемому для входа в систему. Кроме того, в таблице содержится имя пользователя, зашифрованный пароль, а также токен, применяемый для реализации механизма «запомнить меня» при повторных входах.

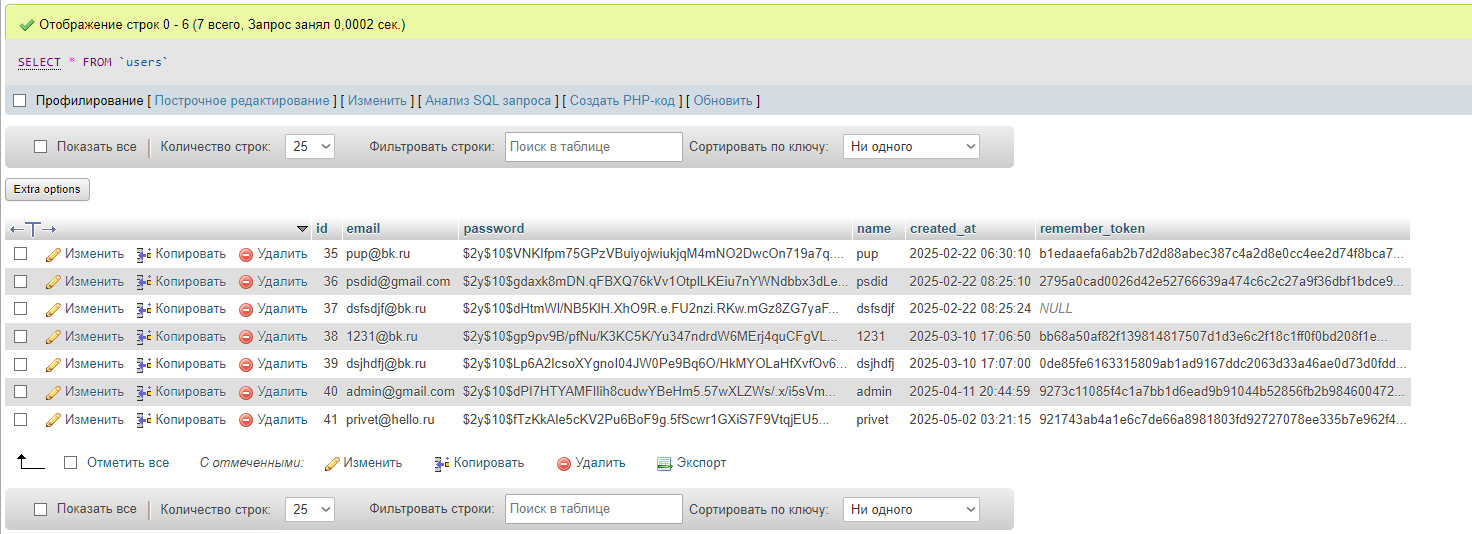


Рисунок 5 – Структура таблицы users в phpMyAdmin.

Основные поля таблицы users:

* id — первичный ключ, автоматически увеличивающийся идентификатор;
* email — уникальный адрес электронной почты, обязательный к заполнению;
* password — хешированный пароль;
* name — имя пользователя;
* remember\_token — опциональный токен для хранения сессионной информации.

Таблица collections описывает кампании по сбору чаевых, создаваемые пользователями. Каждая запись в этой таблице связана с определенным пользователем через внешний ключ user\_id. В рамках одной кампании указывается её название, описание, целевая сумма сбора (goal), сумма, уже собранная (collected), а также сумма заказа, на основе которой рассчитываются чаевые (order\_total).

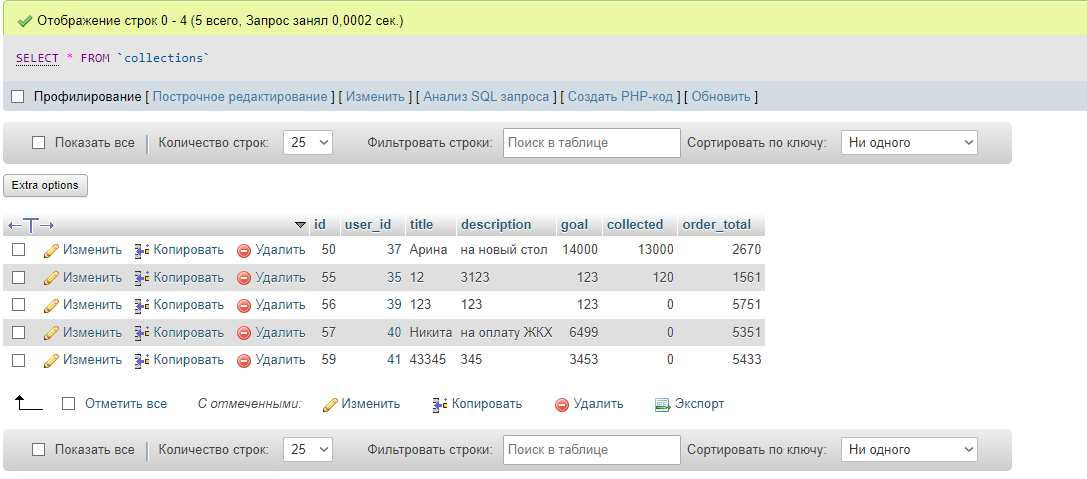


Рисунок 6 – Структура таблицы collections в phpMyAdmin.

Ключевые поля таблицы collections:

* id — уникальный идентификатор сбора;
* user\_id — внешний ключ, ссылающийся на таблицу users;
* title — название кампании;
* description — описание сбора;
* goal — заданная цель по сумме чаевых;
* collected — уже собранная сумма, по умолчанию равна нулю;
* order\_total — сумма счета клиента, на основании которой производится расчет.

Связь между таблицами реализуется по принципу "один ко многим": один пользователь может создавать несколько сборов, тогда как каждый сбор может принадлежать только одному пользователю. Такая модель связи позволяет упростить выборку данных и последующее отображение на пользовательском интерфейсе.

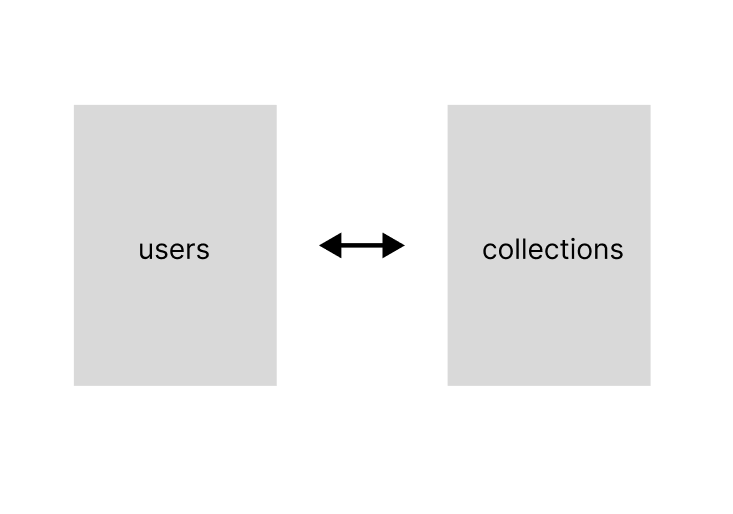


Рисунок 7 – ER-диаграмма.

В рамках этой логической структуры обеспечивается целостность и взаимосвязь данных, закладывается основа для реализации базовых операций CRUD (создание, чтение, обновление, удаление), а также создается задел для возможного расширения схемы в будущем, например, путем добавления таблиц для аналитики, отзывов или истории сборов.

## 2.4.1. Реализация серверной части

Серверная часть веб-сервиса для онлайн оплаты чаевых играет ключевую роль в обеспечении надежного взаимодействия между клиентским интерфейсом и базой данных. Основная задача бэкенда заключается в обработке пользовательских запросов, управлении данными, обеспечении безопасности и поддержании стабильной работы всей системы.

Разработка серверной логики осуществлялась на языке PHP, что обусловлено его широким распространением, простотой в использовании и хорошей интеграцией с системой управления базами данных MySQL. Благодаря применению архитектурного подхода REST API, серверная часть была структурирована в виде набора логически обособленных эндпоинтов, каждый из которых обрабатывает определенный тип запросов. Такой подход обеспечивает гибкость, масштабируемость и удобство поддержки кода в будущем.

Структура проекта была организована таким образом, чтобы максимально разделить ответственность между модулями. В отдельные файлы были вынесены конфигурационные параметры, логика обработки пользовательских данных, а также обработка API-запросов. Подобная модульная архитектура способствует лучшему управлению проектом и облегчает его доработку в будущем.

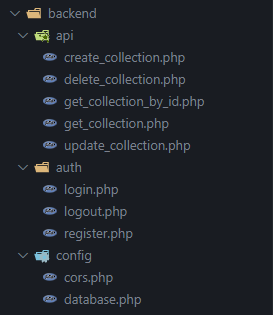


Рисунок 8 – Структура директорий серверной части.

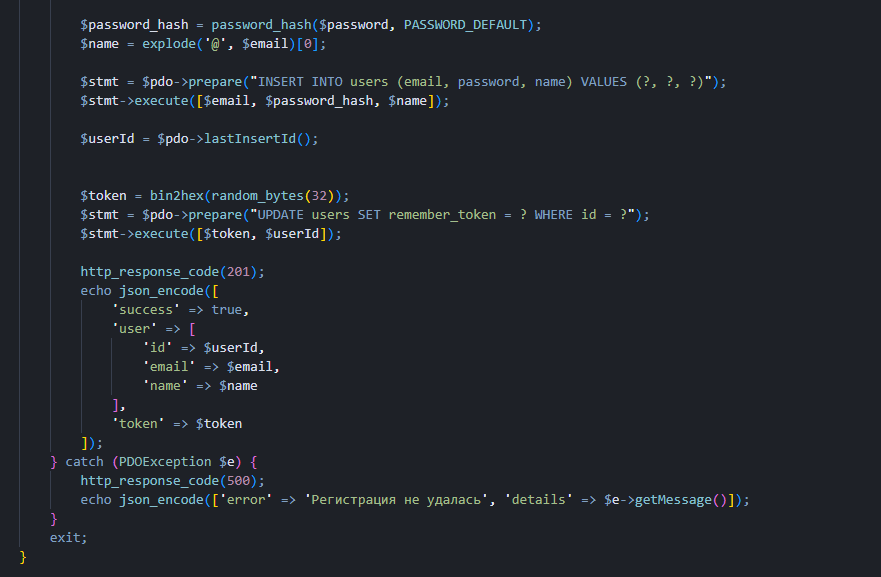
Одним из ключевых компонентов серверной части стал модуль аутентификации пользователей. Он включает в себя как процедуру регистрации, так и авторизацию. В процессе регистрации осуществляется проверка корректности передаваемых данных: email проходит валидацию формата, пароль — проверку на минимальную длину и сложность. После этого выполняется хеширование пароля с использованием встроенной функции password\_hash() и алгоритма bcrypt, что обеспечивает высокий уровень безопасности.  


Рисунок 9 – Обработка регистрации в файле register.php.

Процесс авторизации реализован на базе генерации токенов. После успешной проверки логина и пароля сервер генерирует уникальный криптографически стойкий токен, который сохраняется в базе данных и используется для подтверждения подлинности всех последующих запросов клиента.

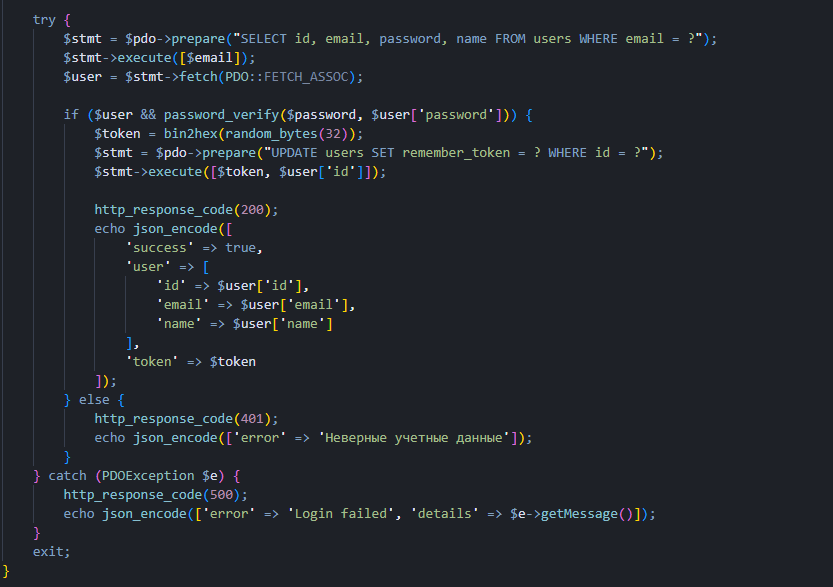


Рисунок 10 – Обработка авторизации в файле login.php.

Неотъемлемой частью серверной логики является модуль управления сборами чаевых. Он реализован через набор API-эндпоинтов, обеспечивающих создание, изменение и просмотр информации о сборах. При поступлении запроса на создание нового сбора, сервер выполняет проверку наличия авторизации, валидирует передаваемые данные (название, описание, сумма заказа и целевая сумма сбора), после чего сохраняет информацию в базе данных.

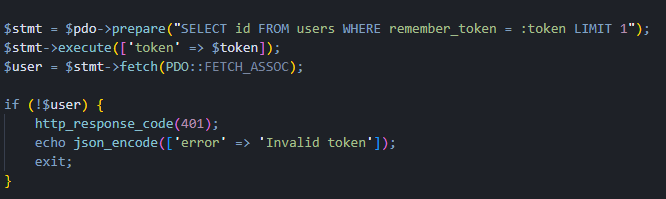


Рисунок 11 – Создания сбора в файле create\_collection.php.

Обновление информации о сборах (например, изменение суммы) также производится через отдельные эндпоинты, при этом осуществляется обязательная проверка, принадлежит ли сбор пользователю, выполнившему запрос. Это необходимо для предотвращения несанкционированных изменений данных.

Существенное внимание было уделено безопасности серверной части. Для предотвращения SQL-инъекций все запросы к базе данных выполняются с использованием подготовленных выражений (Prepared Statements) через расширение PDO. Дополнительно реализована фильтрация пользовательских данных, исключающая внедрение вредоносного HTML или JavaScript-кода.

Также реализована защита на уровне HTTP — с помощью настройки CORS (Cross-Origin Resource Sharing), что ограничивает взаимодействие API только с разрешенных клиентских источников. Это существенно снижает риск межсайтовых атак и делает сервис более защищенным от внешних угроз.

<?php

function setCorsHeaders() {

header('Content-Type: application/json');

header('Access-Control-Allow-Origin: http://localhost:5173');

header('Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS');

header('Access-Control-Allow-Headers: Authorization, Content-Type, X-Requested-With');

header('Access-Control-Allow-Credentials: true');

if ($\_SERVER['REQUEST\_METHOD'] === 'OPTIONS') {

http\_response\_code(200);

exit;

}

}

?>

Для повышения стабильности и простоты отладки, в систему внедрены механизмы обработки ошибок и логирования. Все ключевые события, ошибки и исключения записываются в отдельный лог-файл, что позволяет в оперативном порядке выявлять и устранять возможные сбои в работе сервиса.

Кроме того, реализован ряд оптимизаций, направленных на повышение производительности. К ним относятся:

* минимизация количества обращений к базе данных;
* выборка только необходимых полей в SQL-запросах;
* кэширование сессионных данных;
* предварительная проверка данных до их отправки в БД.

Таким образом, разработанная серверная часть веб-сервиса для онлайн оплаты чаевых демонстрирует высокую степень надежности, безопасности и готовности к расширению. Благодаря использованию современных подходов — REST API, хеширования, валидации, и защищенного взаимодействия с БД — удалось создать устойчивую к сбоям систему, которая способна эффективно обрабатывать пользовательские запросы и управлять данными в реальном времени.

## 2.4.2. API-эндпоинты

API веб-сервиса для онлайн оплаты чаевых построено на принципах REST-архитектуры и использует формат JSON для передачи данных между клиентом и сервером. Весь сервис взаимодействует через стандартные HTTP-методы, что делает его гибким и легко интегрируемым с различными клиентами. Базовый URL для всех запросов: http://localhost/backend/.

Первый раздел API включает в себя эндпоинты для регистрации, авторизации пользователей и завершения сессии. Регистрация пользователя осуществляется через запрос к /auth/register.php с методом POST, при этом данные проверяются и хешируются с использованием bcrypt. В случае успеха возвращается статус 201 Created. Для аутентификации используется эндпоинт /auth/login.php, который также обрабатывает POST-запрос и генерирует токен для дальнейших действий. Для выхода из системы предусмотрен эндпоинт /auth/logout.php, где требуется передача токена в заголовке Authorization.

API также включает эндпоинты для создания, получения, обновления и удаления сборов чаевых. Например, для создания нового сбора используется эндпоинт /api/create\_collection.php, который принимает запрос с методом POST и требует наличия валидного токена. В ответ сервер возвращает статус 201 Created и информацию о созданном сборе. Для получения данных о сборе пользователя, необходимо отправить GET-запрос на /api/get\_collection.php, где токен также проверяется для обеспечения безопасности. Кроме того, API предоставляет возможность обновления информации о сборах через эндпоинт /api/update\_collection.php и удаления через /api/delete\_collection.php, причем в случае последнего происходит проверка прав доступа пользователя.

Система обработки ошибок во всех эндпоинтах стандартизирована. В случае некорректных данных сервер вернет статус 400 Bad Request, при отсутствии или недействительности токена — 401 Unauthorized. Если запрашиваемый ресурс не найден, будет возвращен статус 404 Not Found, а если метод запроса неподдерживаем — 405 Method Not Allowed. В случае непредвиденных проблем сервер вернет статус 500 Internal Server Error.

Все защищенные эндпоинты требуют наличия валидного токена авторизации в заголовке Authorization. Токен передается в формате Bearer {token}, и сервер автоматически проверяет его действительность и соответствие правам доступа пользователя. Эта мера безопасности гарантирует, что только авторизованные пользователи смогут взаимодействовать с защищенными ресурсами сервиса.

Реализованное API является основой для функционирования веб-сервиса, позволяя пользователям взаимодействовать с системой управления сборами чаевых. Благодаря использованию стандартов REST, API предлагает удобные и надежные методы для работы с данными. Все это в сочетании с продуманной системой безопасности и обработки ошибок делает сервис устойчивым, масштабируемым и готовым к дальнейшему расширению.

## 2.4.3. Реализация клиентской части

Клиентская часть веб-сервиса для онлайн оплаты чаевых построена с использованием Vue.js 3, который идеально подходит для создания современных, динамичных одностраничных приложений. Vue помогает нам сделать интерфейс более отзывчивым и удобным для пользователя. В проекте всё разделено на компоненты, что делает его гибким и легко масштабируемым.

Мы используем компоненты страниц, такие как LoginPage.vue для входа в систему, RegistrationPage.vue для регистрации, и DashboardPage.vue, где пользователи могут отслеживать свои сборы. Эти компоненты не просто изолированы друг от друга, но активно взаимодействуют через API, что создает ощущение живого и отзывчивого интерфейса.

Кроме того, есть компоненты, которые можно использовать повторно, такие как CreateCollection.vue для создания новых сборов чаевых и AlertsLogin.vue, который информирует пользователя о важных событиях — например, о проблемах при авторизации.

Каждый компонент настроен так, чтобы поддерживать высокую реактивность и обеспечивать комфортное взаимодействие с сервером. Все данные, которые нужны пользователю для работы с сервисом, подгружаются в реальном времени, так что каждая операция ощущается максимально быстрой.

## 2.4.4. Основные Vue-компоненты

Каждая страница веб-сервиса представлена отдельным компонентом. Например, компонент LoginPage.vue отвечает за отображение формы входа, обработку ошибок при неверных данных и аутентификацию пользователя. Этот компонент содержит логику, необходимую для валидации введенных данных и перенаправления на главную страницу после успешного входа.

Еще один ключевой компонент — RegistrationPage.vue, который реализует регистрацию нового пользователя. Он включает все необходимые поля для ввода данных и их проверки перед отправкой на сервер. Компонент также обеспечивает безопасность данных с использованием хеширования паролей и защиты от взлома.

Компоненты страниц обеспечивают представление всей информации, необходимой для пользователя, и обработку запросов, таких как создание сборов чаевых. Например, DashboardPage.vue служит для отображения информации о текущих сборах, а DonationPage.vue позволяет пользователю производить оплату.

<form @submit.prevent="handleSubmit" class="login-form">

<div v-if="errorMessage" class="error-message">{{ errorMessage }}</div>

<div class="input-container">

<div class="input-wrapper">

<input

v-model="email"

type="text"

class="tip-selector\_\_input"

placeholder="ваш e-mail"

/>

</div>

<div class="input-wrapper password-wrapper">

<input

v-model="password"

:type="showPassword ? 'text' : 'password'"

class="tip-selector\_\_input"

placeholder="пароль"

/>

<button

type="button"

class="password-toggle"

@click="togglePasswordVisibility"

>

<img src="/src/assets/img/icons/eye.svg" alt="" />

</button>

</div>

</div>

<div class="action-buttons">

<button type="submit" class="action-button action-button--primary">

войти

</button>

<button

type="button"

class="action-button action-button--secondary"

@click="$router.push('/auth')"

>

у меня нет аккаунта

</button>

</div>

</form>

Кроме компонентов страниц, в веб-сервисе используются и переиспользуемые компоненты, которые отвечают за определенные элементы интерфейса. Например, компонент CreateCollection.vue позволяет пользователю создавать новые сборы чаевых. Он включает в себя форму для ввода информации о сборе, такой как цель и описание, а также логику для отправки данных на сервер. Компонент использует двустороннюю привязку данных, что позволяет эффективно управлять состоянием формы и отслеживать изменения, внесенные пользователем.

Другим важным компонентом является AlertsLogin.vue, который отображает уведомления для пользователя. Это могут быть сообщения об ошибках при вводе данных на форме или успешной регистрации. В данном компоненте предусмотрены различные варианты отображения уведомлений в зависимости от ситуации, что улучшает взаимодействие с пользователем и повышает удобство использования веб-сервиса.

## 2.4.5. Роутинг и взаимодействие с API

В данном веб-сервисе роутинг реализован с использованием Vue Router, который отвечает за управление переходами между страницами приложения. В коде используется метод createRouter для создания маршрутов и их обработки с помощью createWebHistory. Каждая страница, такая как страница авторизации (LoginPage.vue), регистрации (RegistrationPage.vue) и личного кабинета пользователя (DashboardPage.vue), ассоциируется с соответствующим маршрутом. Также предусмотрены маршруты для отображения конкретных сборов по их ID.

Особое внимание стоит уделить маршруту для личного кабинета пользователя, который защищен с помощью мета-поля requiresAuth. Перед переходом к этим маршрутам осуществляется проверка наличия токена в локальном хранилище. Если токен отсутствует, пользователь перенаправляется на страницу входа. Это реализовано с помощью метода beforeEach, который проверяет авторизацию пользователя перед каждым переходом.

Пример кода роутинга, представленный ниже, демонстрирует, как происходит настройка переходов и проверка прав доступа:

const router = createRouter({

history: createWebHistory(import.meta.env.BASE\_URL),

routes: [

{

path: "/",

redirect: () => {

const isAuthenticated = !!localStorage.getItem("token");

return isAuthenticated ? "/dashboard" : "/login";

},

},

{

path: "/auth",

component: RegistrationPage,

},

{

path: "/login",

component: LoginPage,

},

{

path: "/dashboard",

component: DashboardPage,

meta: { requiresAuth: true },

},

{

path: "/collection/:id",

component: DonationPage,

props: true,

},

],

});

router.beforeEach((to, from, next) => {

const isAuthenticated = !!localStorage.getItem("token");

if (to.meta.requiresAuth && !isAuthenticated) {

next("/login");

} else {

next();

}

});

Кроме того, взаимодействие с API осуществляется через axios или другие механизмы для выполнения запросов. Например, при авторизации или создании сбора, компоненты отправляют запросы на сервер, передавая необходимые данные в формате JSON. В случае успешного ответа от сервера, данные обрабатываются и обновляются в состоянии приложения, обеспечивая обновление UI без перезагрузки страницы.

## 2.5. Механизм генерации QR-кода и ссылки

Механизм генерации QR-кода реализован с помощью простого JavaScript-функционала, который использует сторонний API для создания QR-кодов. Когда пользователь нажимает кнопку для получения QR-кода, вызывается функция getQRCode, которая проверяет наличие текущего сбора. Если сбор существует, генерируется ссылка на страницу сбора, которая затем кодируется в URL-формате для API. QR-код создается с помощью GET-запроса на сервис qrserver.com, и в случае успешной генерации, всплывающее окно с QR-кодом становится видимым.

const getQRCode = () => {

if (currentCollection.value) {

const collectionId = currentCollection.value.id;

qrCodeUrl.value = `https://api.qrserver.com/v1/create-qr-code/?size=150x150&data=${encodeURIComponent(

`http://localhost:5173/collection/${collectionId}`

)}`;

showQRPopup.value = true;

}

};

## 2.6. Средства безопасности веб-сервиса

Веб-сервис применяет несколько многоуровневых механизмов безопасности, чтобы гарантировать защиту данных пользователей и защиту от внешних угроз. Основным элементом является аутентификация с использованием токенов, которые позволяют убедиться в подлинности каждого пользователя. Все запросы, требующие авторизации, обязаны содержать действующий токен в заголовке, предотвращая таким образом доступ посторонних лиц.

Пароли пользователей шифруются, что исключает возможность их раскрытия даже при доступе к базе данных. Использование протокола HTTPS для передачи данных минимизирует риски перехвата информации в процессе взаимодействия клиента с сервером. Это особенно важно для защиты личных данных и финансовой информации, обеспечивая их безопасность в процессе обмена.

Система доступа на основе ролей также играет ключевую роль в защите веб-сервиса. Административные права ограничиваются только для авторизованных пользователей с определенным уровнем доступа, что снижает риски несанкционированных изменений данных. Для пользователей, не обладающих необходимыми правами, доступны только ограниченные функции.

Механизм контроля целостности данных активно отслеживает все операции с конфиденциальной информацией, обеспечивая надежную защиту от возможных атак и вмешательства. Логирование действий пользователей помогает мониторить поведение системы и своевременно реагировать на попытки взлома. В случае обнаружения подозрительных действий автоматически включается процедура расследования и устранения угроз.

# 

# ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта был успешно разработан веб-сервис для онлайн оплаты чаевых с использованием QR-кодов. Этот проект охватывает весь процесс — от анализа требований до реализации ключевых функций системы. Основной целью работы было создание удобного и безопасного интерфейса для пользователей, который позволит им без труда вносить чаевые через интернет, используя технологии QR-кодов для упрощения процесса.

На первом этапе работы был проведен тщательный анализ предметной области, в ходе которого были изучены существующие аналогичные решения, а также выявлены ключевые потребности пользователей. Это исследование позволило сформулировать требования, которые должны быть реализованы в веб-сервисе. В результате были разработаны диаграммы, описывающие основные процессы системы, такие как процесс создания сборов чаевых, их оплаты и генерации уникальных QR-кодов.

Одним из наиболее важных этапов было проектирование архитектуры веб-сервиса, что включало создание REST API для взаимодействия с серверной частью и реализацию безопасности данных. Были спроектированы и реализованы основные API-эндпоинты, которые обеспечивают функциональность сервиса, включая регистрацию и авторизацию пользователей, создание сборов, а также обработку ошибок и управление сессиями. Все эти задачи были решены с использованием передовых технологий, таких как PHP и Vue.js, что позволило обеспечить высокую степень надежности и безопасности приложения.

Клиентская часть веб-сервиса была разработана с использованием фреймворка Vue.js, что позволило создать динамичный и отзывчивый интерфейс для пользователей. Важно отметить, что была проделана значительная работа по реализации роутинга, который обеспечивает переключение между различными страницами сайта, такими как регистрация, авторизация, создание сборов и просмотр статистики. Механизм взаимодействия с API был реализован таким образом, чтобы запросы от клиента обрабатывались быстро и эффективно, а пользователи получали актуальную информацию без лишних задержек.

Особое внимание было уделено разработке механизма генерации QR-кодов, который позволяет пользователям легко поделиться ссылкой для перевода чаевых. Этот функционал был интегрирован непосредственно в интерфейс, что делает процесс генерации QR-кодов удобным и интуитивно понятным.

Что касается безопасности, то были предприняты меры для защиты данных пользователей на всех этапах взаимодействия с системой. Все данные передаются по защищенному каналу связи, а механизмы аутентификации и авторизации обеспечивают защиту от несанкционированного доступа. Реализованный механизм работы с токенами и шифрованием паролей помогает предотвратить потенциальные угрозы безопасности, что делает систему более надежной и безопасной для использования в реальных условиях.

В результате выполнения работы был создан полноценный веб-сервис, который отвечает заявленным требованиям и позволяет пользователям безопасно и удобно вносить чаевые через интернет. В процессе разработки были приобретены важные навыки работы с современными веб-технологиями, такими как Vue.js, PHP, а также с методами обеспечения безопасности веб-приложений. Все поставленные задачи были решены, и сервис готов к использованию в реальной эксплуатации.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

* + 1. Клим A. UML. Советы программистам / А. Клим. – М.: Эксмо-Пресс, 2011. – 450 с.
    2. Фаулер М. UML основы / Мартин Фаулер. – Санкт-Петербург: Символ-Плюс, 2005. – 192 с.
    3. Дадян Э. Г. Проектирование современных баз данных / Э. Г. Дадян. – Москва: 2007. – 114 с.
    4. Веб-разработка с Vue.js. / А. Соловьев. – М.: БХВ-Петербург, 2019. – 300 с.
    5. PHP и MySQL для профессионалов / Дж. Бейкман, Д. Смит. – М.: НТ Пресс, 2016. – 400 с.
    6. Vue.js в действии / Эван Ю. – М.: Издательство "Вильямс", 2018. – 450 с.
    7. Веб-разработка. Учебник / В. А. Костенко. – М.: КНОРУС, 2020. – 500 с.
    8. Введение в создание веб-приложений с использованием Vue.js и PHP [Электронный ресурс]. – Режим доступа: https://www.digitalocean.com/community/tutorials/vue-php, свободный.
    9. Структура и организация проектов на Vue.js [Электронный ресурс]. – Режим доступа: https://laravel-news.com/vue-js-structure, свободный.
    10. Учебник HTML. [Электронный ресурс]. – Режим доступа:http://ru.html.net/tutorials/html/, свободный.
    11. Титтел Э. HTML5 и CSS3 для чайников / Э. Титтел, К. Минник. – Москва: Вильямс, 2016. – 400 с.
    12. Тестирование программного обеспечения [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/test, свободный.
    13. Смирнов Н. Правила создания красивого сайта [Электронный ресурс]. – Режим доступа: http://seoklub.ru/kak\_sdelat\_krasivyi\_sait.html, свободный.
    14. Официальная документация PHP [Электронный ресурс]. – Режим доступа: https://www.php.net/manual/ru/, свободный.
    15. Основы работы с базами данных MySQL / Д. Гизберт. – М.: НТ Пресс, 2014. – 320 с.

## ПРИЛОЖЕНИЕ 1

backend/api/create\_collection.php:

<?php

require\_once '../config/database.php';

require\_once '../config/cors.php';

setCorsHeaders();

if ($\_SERVER['REQUEST\_METHOD'] === 'OPTIONS') {

http\_response\_code(200);

exit;

}

if ($\_SERVER['REQUEST\_METHOD'] !== 'POST') {

http\_response\_code(405);

echo json\_encode(['error' => 'Метод не разрешен']);

exit;

}

$headers = getallheaders();

$token = $headers['Authorization'] ?? '';

if (empty($token)) {

http\_response\_code(401);

echo json\_encode(['error' => 'Токен не предоставлен']);

exit;

}

if (str\_starts\_with($token, 'Bearer ')) {

$token = substr($token, 7);

}

$stmt = $pdo->prepare("SELECT id FROM users WHERE remember\_token = :token LIMIT 1");

$stmt->execute(['token' => $token]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

if (!$user) {

http\_response\_code(401);

echo json\_encode(['error' => 'Недействительный токен']);

exit;

}

$data = json\_decode(file\_get\_contents('php://input'), true);

$title = $data['title'] ?? '';

$description = $data['description'] ?? '';

$goal = $data['goal'] ?? 0;

if (empty($title) || empty($description) || $goal <= 0) {

http\_response\_code(400);

echo json\_encode(['error' => 'Все поля (title, description, goal) обязательны']);

exit;

}

$stmt = $pdo->prepare("INSERT INTO collections (user\_id, title, description, goal, collected, order\_total) VALUES (:user\_id, :title, :description, :goal, 0, :order\_total)");

$stmt->execute([

'user\_id' => $user['id'],

'title' => $title,

'description' => $description,

'goal' => $goal,

'order\_total' => generateRandomOrderTotal()

]);

function generateRandomOrderTotal() {

return rand(400, 8000);

}

$collectionId = $pdo->lastInsertId();

http\_response\_code(201);

echo json\_encode([

'success' => true,

'collection' => [

'id' => $collectionId,

'user\_id' => $user['id'],

'title' => $title,

'description' => $description,

'goal' => $goal,

'collected' => 0

]

]);

exit;

?>

backend/api/delete\_collection.php:

<?php

require\_once '../config/database.php';

require\_once '../config/cors.php';

setCorsHeaders();

if ($\_SERVER['REQUEST\_METHOD'] === 'OPTIONS') {

http\_response\_code(200);

exit;

}

if ($\_SERVER['REQUEST\_METHOD'] !== 'POST') {

http\_response\_code(405);

echo json\_encode(['error' => 'Метод не разрешен']);

exit;

}

$headers = getallheaders();

$token = $headers['Authorization'] ?? '';

if (empty($token)) {

http\_response\_code(401);

echo json\_encode(['error' => 'Токен не предоставлен']);

exit;

}

if (str\_starts\_with($token, 'Bearer ')) {

$token = substr($token, 7);

}

$stmt = $pdo->prepare("SELECT id FROM users WHERE remember\_token = :token LIMIT 1");

$stmt->execute(['token' => $token]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

if (!$user) {

http\_response\_code(401);

echo json\_encode(['error' => 'Недействительный токен']);

exit;

}

$data = json\_decode(file\_get\_contents('php://input'), true);

$id = $data['id'] ?? 0;

if (empty($id)) {

http\_response\_code(400);

echo json\_encode(['error' => 'ID сбора обязателен']);

exit;

}

$stmt = $pdo->prepare("DELETE FROM collections WHERE id = :id AND user\_id = :user\_id");

$stmt->execute(['id' => $id, 'user\_id' => $user['id']]);

if ($stmt->rowCount() === 0) {

http\_response\_code(404);

echo json\_encode(['error' => 'Сбор не найден']);

exit;

}

http\_response\_code(200);

echo json\_encode(['success' => true]);

exit;

?>

backend/api/get\_collection\_by\_id.php:

<?php

require\_once '../config/database.php';

require\_once '../config/cors.php';

setCorsHeaders();

if ($\_SERVER['REQUEST\_METHOD'] === 'OPTIONS') {

http\_response\_code(200);

exit;

}

if ($\_SERVER['REQUEST\_METHOD'] !== 'GET') {

http\_response\_code(405);

echo json\_encode(['error' => 'Метод не разрешен']);

exit;

}

$id = $\_GET['id'] ?? 0;

if (empty($id)) {

http\_response\_code(400);

echo json\_encode(['error' => 'ID сбора обязателен']);

exit;

}

$stmt = $pdo->prepare("SELECT \* FROM collections WHERE id = :id LIMIT 1");

$stmt->execute(['id' => $id]);

$collection = $stmt->fetch(PDO::FETCH\_ASSOC);

if (!$collection) {

http\_response\_code(404);

echo json\_encode(['error' => 'Сбор завершен']);

exit;

}

http\_response\_code(200);

echo json\_encode([

'success' => true,

'collection' => $collection

]);

exit;

?>

backend/api/get\_collection.php:  
<?php

require\_once '../config/database.php';

require\_once '../config/cors.php';

setCorsHeaders();

if ($\_SERVER['REQUEST\_METHOD'] === 'OPTIONS') {

http\_response\_code(200);

exit;

}

if ($\_SERVER['REQUEST\_METHOD'] !== 'GET') {

http\_response\_code(405);

echo json\_encode(['error' => 'Метод не разрешен']);

exit;

}

$headers = getallheaders();

$token = $headers['Authorization'] ?? '';

if (empty($token)) {

http\_response\_code(401);

echo json\_encode(['error' => 'Токен не предоставлен']);

exit;

}

if (str\_starts\_with($token, 'Bearer ')) {

$token = substr($token, 7);

}

$stmt = $pdo->prepare("SELECT id FROM users WHERE remember\_token = :token LIMIT 1");

$stmt->execute(['token' => $token]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

if (!$user) {

http\_response\_code(401);

echo json\_encode(['error' => 'Недействительный токен']);

exit;

}

$stmt = $pdo->prepare("SELECT \* FROM collections WHERE user\_id = :user\_id LIMIT 1");

$stmt->execute(['user\_id' => $user['id']]);

$collection = $stmt->fetch(PDO::FETCH\_ASSOC);

http\_response\_code(200);

echo json\_encode([

'success' => true,

'collection' => $collection ?: null

]);

exit;

?>

backend/api/update\_collection.php:  
<?php

require\_once '../config/database.php';

require\_once '../config/cors.php';

setCorsHeaders();

if ($\_SERVER['REQUEST\_METHOD'] === 'OPTIONS') {

http\_response\_code(200);

exit;

}

if ($\_SERVER['REQUEST\_METHOD'] !== 'POST') {

http\_response\_code(405);

echo json\_encode(['error' => 'Метод не разрешен']);

exit;

}

$data = json\_decode(file\_get\_contents('php://input'), true);

$id = $data['id'] ?? 0;

$title = $data['title'] ?? '';

$description = $data['description'] ?? '';

$goal = $data['goal'] ?? 0;

$collected = $data['collected'] ?? 0;

if (empty($id) || empty($title) || empty($description) || $goal <= 0) {

http\_response\_code(400);

echo json\_encode(['error' => 'Все поля (id, title, description, goal) обязательны']);

exit;

}

$stmt = $pdo->prepare("UPDATE collections SET title = :title, description = :description, goal = :goal, collected = :collected WHERE id = :id");

$stmt->execute([

'title' => $title,

'description' => $description,

'goal' => $goal,

'collected' => $collected,

'id' => $id,

]);

if ($stmt->rowCount() === 0) {

http\_response\_code(404);

echo json\_encode(['error' => 'Коллекция не найдена']);

exit;

}

http\_response\_code(200);

echo json\_encode([

'success' => true,

'collection' => [

'id' => $id,

'title' => $title,

'description' => $description,

'goal' => $goal,

'collected' => $collected

]

]);

exit;

?>

backend/auth/login.php:

<?php

require\_once '../config/database.php';

require\_once '../config/cors.php';

setCorsHeaders();

$data = json\_decode(file\_get\_contents('php://input'), true);

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$email = $data['email'] ?? '';

$password = $data['password'] ?? '';

if (empty($email) || empty($password)) {

http\_response\_code(400);

echo json\_encode(['error' => 'Введите адрес электронной почты и пароль']);

exit;

}

try {

$stmt = $pdo->prepare("SELECT id, email, password, name FROM users WHERE email = ?");

$stmt->execute([$email]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($user && password\_verify($password, $user['password'])) {

$token = bin2hex(random\_bytes(32));

$stmt = $pdo->prepare("UPDATE users SET remember\_token = ? WHERE id = ?");

$stmt->execute([$token, $user['id']]);

http\_response\_code(200);

echo json\_encode([

'success' => true,

'user' => [

'id' => $user['id'],

'email' => $user['email'],

'name' => $user['name']

],

'token' => $token

]);

} else {

http\_response\_code(401);

echo json\_encode(['error' => 'Неверные учетные данные']);

}

} catch (PDOException $e) {

http\_response\_code(500);

echo json\_encode(['error' => 'Ошибка входа', 'details' => $e->getMessage()]);

}

exit;

}

http\_response\_code(405);

echo json\_encode(['error' => 'Метод не разрешен']);

exit;

backend/auth/logout:

<?php

require\_once '../config/database.php';

require\_once '../config/cors.php';

setCorsHeaders();

if ($\_SERVER['REQUEST\_METHOD'] === 'OPTIONS') {

http\_response\_code(200);

exit;

}

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$headers = getallheaders();

$token = $headers['Authorization'] ?? '';

if (empty($token)) {

http\_response\_code(401);

echo json\_encode(['error' => 'Токен не предоставлен']);

exit;

}

$stmt = $pdo->prepare("UPDATE users SET remember\_token = NULL WHERE remember\_token = ?");

$stmt->execute([$token]);

http\_response\_code(200);

echo json\_encode(['success' => true]);

exit;

} else {

http\_response\_code(405);

echo json\_encode(['error' => 'Метод не разрешен']);

exit;

}

?>

backend/auth/register.php:  
<?php

require\_once '../config/database.php';

require\_once '../config/cors.php';

setCorsHeaders();

$data = json\_decode(file\_get\_contents('php://input'), true);

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$email = $data['email'] ?? '';

$password = $data['password'] ?? '';

if (empty($email) || empty($password)) {

http\_response\_code(400);

echo json\_encode(['error' => 'Введите адрес электронной почты и пароль']);

exit;

}

if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

http\_response\_code(400);

echo json\_encode(['error' => 'Неверный формат электронной почты']);

exit;

}

if (strlen($password) < 6) {

http\_response\_code(400);

echo json\_encode(['error' => 'Пароль должен быть не менее 6 символов']);

exit;

}

try {

$stmt = $pdo->prepare("SELECT id FROM users WHERE email = ?");

$stmt->execute([$email]);

if ($stmt->rowCount() > 0) {

http\_response\_code(400);

echo json\_encode(['error' => 'Адрес указан неверно или существует']);

exit;

}

$password\_hash = password\_hash($password, PASSWORD\_DEFAULT);

$name = explode('@', $email)[0];

$stmt = $pdo->prepare("INSERT INTO users (email, password, name) VALUES (?, ?, ?)");

$stmt->execute([$email, $password\_hash, $name]);

$userId = $pdo->lastInsertId();

$token = bin2hex(random\_bytes(32));

$stmt = $pdo->prepare("UPDATE users SET remember\_token = ? WHERE id = ?");

$stmt->execute([$token, $userId]);

http\_response\_code(201);

echo json\_encode([

'success' => true,

'user' => [

'id' => $userId,

'email' => $email,

'name' => $name

],

'token' => $token

]);

} catch (PDOException $e) {

http\_response\_code(500);

echo json\_encode(['error' => 'Регистрация не удалась', 'details' => $e->getMessage()]);

}

exit;

}

http\_response\_code(405);

echo json\_encode(['error' => 'Метод не разрешен']);

exit;

backend/config/database.php:  
<?php

$host = 'localhost';

$dbname = 'tips\_service';

$username = 'root';

$password = '';

try {

$pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch(PDOException $e) {

echo "Connection failed: " . $e->getMessage();

die();

}

?>

## ПРИЛОЖЕНИЕ 2

frontend/src/App.vue:

<script setup>

import { RouterView } from "vue-router";

</script>

<template>

<div class="app">

<RouterView />

</div>

</template>

frontend/src/[main.js](http://main.js):  
import { createApp } from "vue";

import App from "./App.vue";

import router from "./router";

import "./assets/main.css";

const app = createApp(App);

app.use(router);

app.mount("#app");

frontend/src/api/[auth.js](http://auth.js):  
const AUTH\_URL = "http://localhost/vue-tipping-project/backend/auth";

const API\_URL = "http://localhost/vue-tipping-project/backend/api";

export const auth = {

async register(email, password) {

const response = await fetch(`${AUTH\_URL}/register.php`, {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({ email, password }),

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка регистрации");

}

if (data.token) {

localStorage.setItem("token", data.token);

localStorage.setItem("user", JSON.stringify(data.user));

}

return data;

},

async login(email, password) {

const response = await fetch(`${AUTH\_URL}/login.php`, {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({ email, password }),

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка входа");

}

if (data.token) {

localStorage.setItem("token", data.token);

localStorage.setItem("user", JSON.stringify(data.user));

}

return data;

},

async logout() {

const token = localStorage.getItem("token");

const response = await fetch(`${AUTH\_URL}/logout.php`, {

method: "POST",

headers: {

Authorization: token,

},

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка выхода");

}

localStorage.removeItem("token");

localStorage.removeItem("user");

return data;

},

async getCollection() {

const token = localStorage.getItem("token");

const response = await fetch(`${API\_URL}/get\_collection.php`, {

method: "GET",

headers: {

Authorization: token,

},

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка загрузки коллекции");

}

return data.collection;

},

async getCollectionById(id) {

const token = localStorage.getItem("token");

const response = await fetch(

`${API\_URL}/get\_collection\_by\_id.php?id=${id}`,

{

method: "GET",

headers: {

Authorization: token,

},

}

);

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Коллекция не найдена");

}

return data.collection;

},

async createCollection(title, description, goal) {

const token = localStorage.getItem("token");

const response = await fetch(`${API\_URL}/create\_collection.php`, {

method: "POST",

headers: {

"Content-Type": "application/json",

Authorization: token,

},

body: JSON.stringify({ title, description, goal }),

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка создания коллекции");

}

return data.collection;

},

async updateCollection(id, title, description, goal, collected) {

const response = await fetch(`${API\_URL}/update\_collection.php`, {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({ id, title, description, goal, collected }),

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка обновления сбора");

}

return data.collection;

},

async deleteCollection(id) {

const token = localStorage.getItem("token");

const response = await fetch(`${API\_URL}/delete\_collection.php`, {

method: "POST",

headers: {

"Content-Type": "application/json",

Authorization: token,

},

body: JSON.stringify({ id }),

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка удаления сбора");

}

return data;

},

};

frontend/src/components/AlertsLogin.vue:  
const AUTH\_URL = "http://localhost/vue-tipping-project/backend/auth";

const API\_URL = "http://localhost/vue-tipping-project/backend/api";

export const auth = {

async register(email, password) {

const response = await fetch(`${AUTH\_URL}/register.php`, {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({ email, password }),

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка регистрации");

}

if (data.token) {

localStorage.setItem("token", data.token);

localStorage.setItem("user", JSON.stringify(data.user));

}

return data;

},

async login(email, password) {

const response = await fetch(`${AUTH\_URL}/login.php`, {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({ email, password }),

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка входа");

}

if (data.token) {

localStorage.setItem("token", data.token);

localStorage.setItem("user", JSON.stringify(data.user));

}

return data;

},

async logout() {

const token = localStorage.getItem("token");

const response = await fetch(`${AUTH\_URL}/logout.php`, {

method: "POST",

headers: {

Authorization: token,

},

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка выхода");

}

localStorage.removeItem("token");

localStorage.removeItem("user");

return data;

},

async getCollection() {

const token = localStorage.getItem("token");

const response = await fetch(`${API\_URL}/get\_collection.php`, {

method: "GET",

headers: {

Authorization: token,

},

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка загрузки коллекции");

}

return data.collection;

},

async getCollectionById(id) {

const token = localStorage.getItem("token");

const response = await fetch(

`${API\_URL}/get\_collection\_by\_id.php?id=${id}`,

{

method: "GET",

headers: {

Authorization: token,

},

}

);

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Коллекция не найдена");

}

return data.collection;

},

async createCollection(title, description, goal) {

const token = localStorage.getItem("token");

const response = await fetch(`${API\_URL}/create\_collection.php`, {

method: "POST",

headers: {

"Content-Type": "application/json",

Authorization: token,

},

body: JSON.stringify({ title, description, goal }),

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка создания коллекции");

}

return data.collection;

},

async updateCollection(id, title, description, goal, collected) {

const response = await fetch(`${API\_URL}/update\_collection.php`, {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({ id, title, description, goal, collected }),

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка обновления сбора");

}

return data.collection;

},

async deleteCollection(id) {

const token = localStorage.getItem("token");

const response = await fetch(`${API\_URL}/delete\_collection.php`, {

method: "POST",

headers: {

"Content-Type": "application/json",

Authorization: token,

},

body: JSON.stringify({ id }),

});

const data = await response.json();

if (!response.ok) {

throw new Error(data.error || "Ошибка удаления сбора");

}

return data;

},

};

frontend/src/components/CreateCollection.vue:

<script setup>

import { ref } from "vue";

const title = ref("");

const description = ref("");

const goal = ref("");

const emit = defineEmits(["create-collection", "cancel-create"]);

const handleSubmit = () => {

if (title.value && description.value && goal.value) {

const newCollection = {

title: title.value,

description: description.value,

goal: parseInt(goal.value),

collected: 0,

};

emit("create-collection", newCollection);

} else {

alert("Пожалуйста, заполните все поля");

}

};

const handleCancel = () => {

emit("cancel-create");

};

</script>

<template>

<div class="collection-form">

<div class="input-container">

<div class="input-wrapper">

<input

v-model="title"

type="text"

class="tip-selector\_\_input"

placeholder="твое имя"

/>

</div>

<div class="input-wrapper">

<input

v-model="description"

type="text"

class="tip-selector\_\_input"

placeholder="на что идет сбор"

/>

</div>

<div class="input-wrapper">

<input

v-model="goal"

type="number"

class="tip-selector\_\_input"

placeholder="сумма сбора"

/>

</div>

</div>

<div class="action-buttons">

<button

class="action-button action-button--primary"

@click="handleSubmit"

>

Сохранить

</button>

<button

class="action-button action-button--secondary"

@click="handleCancel"

>

Отменить

</button>

</div>

</div>

</template>

<style scoped>

.collection-form {

margin-top: 32px;

}

.input-container {

background-color: #303030;

border-radius: 12px;

padding: 16px;

margin-top: 16px;

}

.input-wrapper {

margin-bottom: 16px;

}

.tip-selector\_\_input {

width: 100%;

height: 48px;

padding: 12px 16px;

border: none;

color: #fff;

font-family: "Inter";

font-size: 16px;

font-weight: 500;

outline: none;

transition: background-color 0.2s ease;

box-sizing: border-box;

background-color: #444;

border-radius: 10px;

}

.tip-selector\_\_input::placeholder {

color: rgba(255, 255, 255, 0.5);

}

.action-buttons {

display: flex;

flex-direction: column;

gap: 8px;

margin-top: 24px;

}

.action-button {

width: 100%;

font-family: "Inter";

font-weight: 600;

font-size: 16px;

line-height: 125%;

letter-spacing: -0.01em;

text-align: center;

border: none;

cursor: pointer;

}

.action-button--primary {

color: #ffffff;

box-shadow: inset 0 0 5px 0 rgba(255, 255, 255, 0.6),

0 2px 5px 0 rgba(241, 89, 10, 0.12);

background: linear-gradient(180deg, #f80 0%, #ef5000 100%),

radial-gradient(

94.42% 96.86% at 50.18% 0%,

rgba(255, 255, 255, 0.4) 0%,

rgba(255, 255, 255, 0) 100%

);

border-radius: 12px;

padding: 16px;

}

.action-button--secondary {

color: #ffffff;

border-radius: 12px;

padding: 16px;

background-color: #1a1a1a;

}

</style>

frontend/src/router/[index.js](http://index.js):

import { createRouter, createWebHistory } from "vue-router";

import DonationPage from "../views/DonationPage.vue";

import RegistrationPage from "../views/RegistrationPage.vue";

import LoginPage from "../views/LoginPage.vue";

import DashboardPage from "../views/DashboardPage.vue";

const router = createRouter({

history: createWebHistory(import.meta.env.BASE\_URL),

routes: [

{

path: "/",

redirect: () => {

const isAuthenticated = !!localStorage.getItem("token");

return isAuthenticated ? "/dashboard" : "/login";

},

},

{

path: "/auth",

component: RegistrationPage,

},

{

path: "/login",

component: LoginPage,

},

{

path: "/dashboard",

component: DashboardPage,

meta: { requiresAuth: true },

},

{

path: "/collection/:id",

component: DonationPage,

props: true,

},

],

});

router.beforeEach((to, from, next) => {

const isAuthenticated = !!localStorage.getItem("token");

if (to.meta.requiresAuth && !isAuthenticated) {

next("/login");

} else {

next();

}

});

export default router;

frontend/src/views/DashboardPage.vue:

<script setup>

import { ref, onMounted, watch, computed } from "vue";

import { useRouter } from "vue-router";

import { auth } from "@/api/auth";

import CreateCollection from "@/components/CreateCollection.vue";

import AlertsLogin from "@/components/AlertsLogin.vue";

const router = useRouter();

const user = ref(null);

const currentCollection = ref(null);

const hasCollection = ref(false);

const isCreatingCollection = ref(false);

const showLoginAlert = ref(false);

const showQRPopup = ref(false);

const qrCodeUrl = ref(null);

const notificationMessage = ref("");

const showNotification = ref(false);

onMounted(() => {

const storedUser = localStorage.getItem("user");

if (!storedUser) {

router.push("/login");

return;

}

user.value = JSON.parse(storedUser);

showLoginAlert.value = true;

fetchCollection();

});

const fetchCollection = async () => {

try {

const collection = await auth.getCollection();

if (collection) {

currentCollection.value = collection;

hasCollection.value = true;

}

} catch (error) {

console.error("Ошибка загрузки сбора:", error);

}

};

const createCollection = async (newCollection) => {

try {

const createdCollection = await auth.createCollection(

newCollection.title,

newCollection.description,

newCollection.goal

);

currentCollection.value = createdCollection;

hasCollection.value = true;

isCreatingCollection.value = false;

showNotification.value = true;

notificationMessage.value = "Сбор успешно создан";

setTimeout(() => (showNotification.value = false), 3000);

} catch (error) {

console.error("Ошибка при создании сбора:", error);

showNotification.value = true;

notificationMessage.value = "Ошибка при создании сбора";

setTimeout(() => (showNotification.value = false), 3000);

}

};

const deleteCollection = async () => {

try {

await auth.deleteCollection(currentCollection.value.id);

currentCollection.value = null;

hasCollection.value = false;

qrCodeUrl.value = null;

showNotification.value = true;

notificationMessage.value = "сбор удален";

setTimeout(() => (showNotification.value = false), 3000);

} catch (error) {

console.error("Ошибка при удалении сбора:", error);

showNotification.value = true;

notificationMessage.value = "ошибка при удалении сбора";

setTimeout(() => (showNotification.value = false), 3000);

}

};

const getQRCode = () => {

if (currentCollection.value) {

const collectionId = currentCollection.value.id;

qrCodeUrl.value = `https://api.qrserver.com/v1/create-qr-code/?size=150x150&data=${encodeURIComponent(

`http://localhost:5173/collection/${collectionId}`

)}`;

showQRPopup.value = true;

}

};

const copyLink = () => {

if (currentCollection.value) {

const collectionLink = `http://localhost:5173/collection/${currentCollection.value.id}`;

navigator.clipboard.writeText(collectionLink).then(() => {

showNotification.value = true;

notificationMessage.value = "ссылка скопирована";

setTimeout(() => (showNotification.value = false), 3000);

});

}

};

const closeQRPopup = (event) => {

if (!event.target.closest(".qr-popup-content")) {

showQRPopup.value = false;

}

};

const isGoalReached = computed(() => {

return (

currentCollection.value &&

currentCollection.value.collected >= currentCollection.value.goal

);

});

const handleLogout = () => {

localStorage.removeItem("user");

router.push("/login");

};

</script>

<template>

<main>

<div class="container">

<AlertsLogin

v-if="showLoginAlert"

message="вход успешно выполнен"

:duration="2000"

@close="handleAlertClose"

/>

<AlertsLogin

v-if="showNotification"

:message="notificationMessage"

:duration="3000"

@close="showNotification = false"

/>

<div class="header-wrapper animated fade-in">

<div class="card-title">личный кабинет</div>

</div>

<div class="user-info animated slide-up">

<div class="user-info\_\_card">

<div class="user-info\_\_header">

<div class="user-info\_\_text">

<p class="user-info\_\_collecting">

добро пожаловать, {{ user?.name || user?.email }}

</p>

<div class="user-info\_\_goal">{{ user?.email }}</div>

</div>

<div class="user-info\_\_avatar">

<img

:src="user?.avatar || '/src/assets/img/avatars.jpg'"

alt="User avatar"

/>

</div>

</div>

<button

@click="handleLogout"

class="action-button action-button--secondary animated fade-in"

>

выйти

</button>

</div>

</div>

<div

v-if="hasCollection && !isCreatingCollection"

class="collection-info animated slide-up"

>

<div class="user-info\_\_card">

<div class="user-info\_\_header">

<div class="user-info\_\_text">

<p class="user-info\_\_collecting">{{ currentCollection.title }}</p>

<div class="user-info\_\_goal">

{{ currentCollection.description }}

</div>

</div>

</div>

<div class="user-info\_\_progress">

<div class="user-info\_\_progress-text">

Собрано {{ currentCollection.collected.toLocaleString() }} из

{{ currentCollection.goal.toLocaleString() }} ₽

</div>

<div class="user-info\_\_progress-bar">

<div

class="user-info\_\_progress-fill"

:style="{

width: `${

(currentCollection.collected / currentCollection.goal) \* 100

}%`,

}"

></div>

</div>

</div>

<div class="action-buttons">

<button

v-if="!isGoalReached"

class="action-button action-button--primary animated fade-in"

@click="getQRCode"

>

получить QR-код

</button>

<button

v-if="isGoalReached"

class="action-button action-button--primary animated fade-in"

disabled

>

цель достигнута

</button>

<button

class="action-button action-button--secondary animated fade-in"

@click="deleteCollection"

>

удалить сбор

</button>

</div>

</div>

</div>

<div

v-if="!hasCollection && !isCreatingCollection"

class="animated slide-up"

>

<button

class="action-button action-button--primary animated fade-in"

@click="isCreatingCollection = true"

>

создать сбор

</button>

</div>

<div

v-if="isCreatingCollection"

class="collection-container animated slide-up"

>

<CreateCollection

@create-collection="createCollection"

@cancel-create="isCreatingCollection = false"

/>

</div>

<div

v-if="showQRPopup"

class="qr-overlay animated fade-in"

@click="closeQRPopup"

>

<div class="qr-popup-content">

<img :src="qrCodeUrl" alt="QR Code" />

<button class="copy-link-button animated fade-in" @click="copyLink">

Скопировать ссылку

</button>

</div>

</div>

</div>

</main>

</template>

<style scoped>

.container {

padding: 0 16px;

position: relative;

}

.card-title {

font-family: "Inter";

width: 343px;

color: #fff;

font-size: 50px;

font-weight: 700;

line-height: 50px;

letter-spacing: -0.75px;

}

.user-info\_\_card {

margin-top: 32px;

margin-bottom: 32px;

padding: 16px;

background-color: #303030;

border-radius: 12px;

}

.user-info\_\_header {

display: flex;

justify-content: space-between;

align-items: flex-start;

margin-bottom: 16px;

}

.user-info\_\_text {

flex-grow: 1;

}

.user-info\_\_collecting {

font-family: "Inter";

font-weight: 600;

font-size: 14px;

line-height: 114%;

letter-spacing: -0.01em;

color: #8e8e8e;

}

.user-info\_\_goal {

color: #fff;

font-family: "Inter";

font-weight: 600;

font-size: 20px;

line-height: 140%;

letter-spacing: -0.01em;

color: #fff;

gap: 8px;

}

.user-info\_\_avatar {

width: 52px;

height: 52px;

border-radius: 50%;

overflow: hidden;

}

.user-info\_\_avatar img {

width: 100%;

height: 100%;

object-fit: cover;

}

.user-info\_\_progress {

padding-top: 12px;

border-top: 1px solid #4b4b4b;

}

.user-info\_\_progress-text {

color: #fff;

font-family: "Inter";

font-weight: 600;

font-size: 14px;

line-height: 114%;

letter-spacing: -0.01em;

color: #fff;

margin-bottom: 8px;

}

.user-info\_\_progress-bar {

height: 8px;

background: linear-gradient(90deg, #f2eaea 0%, #4f4d4d 100%);

border-radius: 4px;

overflow: hidden;

}

.user-info\_\_progress-fill {

height: 100%;

background: linear-gradient(90deg, #f80 0%, #ef5000 100%);

border-radius: 4px;

}

.action-buttons {

display: flex;

flex-direction: column;

gap: 8px;

margin-top: 24px;

}

.action-button {

width: 100%;

font-family: "Inter";

font-weight: 600;

font-size: 16px;

line-height: 125%;

letter-spacing: -0.01em;

text-align: center;

border: none;

cursor: pointer;

}

.action-button--primary {

color: #ffffff;

box-shadow: inset 0 0 5px 0 rgba(255, 255, 255, 0.6),

0 2px 5px 0 rgba(241, 89, 10, 0.12);

background: linear-gradient(180deg, #f80 0%, #ef5000 100%),

radial-gradient(

94.42% 96.86% at 50.18% 0%,

rgba(255, 255, 255, 0.4) 0%,

rgba(255, 255, 255, 0) 100%

);

border-radius: 12px;

padding: 16px;

}

.action-button--primary:disabled {

background: #1a1a1a;

box-shadow: none;

cursor: not-allowed;

}

.action-button--secondary {

color: #ffffff;

border-radius: 12px;

padding: 16px;

background-color: #1a1a1a;

}

.collection-container {

margin-top: 32px;

}

.qr-overlay {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(0, 0, 0, 0.7);

display: flex;

justify-content: center;

align-items: center;

z-index: 1000;

}

.qr-popup-content {

display: flex;

flex-direction: column;

align-items: center;

background: #303030;

padding: 32px;

border-radius: 12px;

text-align: center;

position: relative;

}

.qr-popup-content img {

width: 180px;

height: 180px;

border: 4px solid white;

border-radius: 12px;

}

.copy-link-button {

margin-top: 24px;

font-family: "Inter";

font-weight: 600;

font-size: 16px;

line-height: 125%;

letter-spacing: -0.01em;

text-align: center;

border: none;

cursor: pointer;

color: #ffffff;

background: #1a1a1a;

border-radius: 12px;

padding: 12px;

}

.animated {

animation-duration: 0.5s;

animation-fill-mode: both;

}

.fade-in {

animation-name: fadeIn;

}

.slide-up {

animation-name: slideUp;

}

@keyframes fadeIn {

from {

opacity: 0;

}

to {

opacity: 1;

}

}

@keyframes slideUp {

from {

opacity: 0;

transform: translateY(20px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

.qr-overlay.animated {

animation-delay: 0.1s;

}

.alerts-login {

position: fixed;

top: 20px;

left: 50%;

transform: translateX(-50%);

z-index: 1100;

}

.goal-reached-message {

margin-top: 16px;

padding: 12px;

background-color: #4caf50;

border-radius: 12px;

text-align: center;

}

.goal-reached-text {

font-family: "Inter";

font-weight: 600;

font-size: 16px;

color: #fff;

margin: 0;

}

</style>

frontend/src/views/DonationPage.vue:

<script setup>

import { ref, onMounted, computed, watch } from "vue";

import { useRoute } from "vue-router";

import { auth } from "@/api/auth";

import AlertsLogin from "@/components/AlertsLogin.vue";

const route = useRoute();

const collectionId = ref(route.params.id);

const collection = ref(null);

const error = ref(null);

const donationAmount = ref(null);

const showNotification = ref(false);

const notificationMessage = ref("");

const orderTotal = ref(null);

const showPaymentPopup = ref(false);

const generateRandomOrderTotal = () => {

return Math.floor(Math.random() \* (8000 - 400 + 1)) + 400;

};

onMounted(async () => {

try {

collection.value = await auth.getCollectionById(collectionId.value);

orderTotal.value =

collection.value.order\_total || generateRandomOrderTotal();

} catch (err) {

error.value = err.message || "Ошибка загрузки коллекции";

}

});

watch(collection, (newCollection) => {

if (newCollection && newCollection.order\_total) {

orderTotal.value = newCollection.order\_total;

}

});

const tipPercentages = [7, 10, 12];

const tips = computed(() => {

if (!collection.value || !orderTotal.value) return [];

return tipPercentages.map((percentage) => ({

percentage,

amount: Math.round((orderTotal.value \* percentage) / 100),

}));

});

const selectedTip = ref(null);

const isCustomMode = ref(false);

const customAmount = ref("");

const isCustomAmountConfirmed = ref(false);

const customTipPercentage = computed(() => {

if (customAmount.value && orderTotal.value) {

const percentage = (parseInt(customAmount.value) / orderTotal.value) \* 100;

return percentage.toLocaleString();

}

return null;

});

const handleTipSelect = (index) => {

if (selectedTip.value === index) {

selectedTip.value = null;

donationAmount.value = null;

} else {

selectedTip.value = index;

donationAmount.value = tips.value[index].amount;

}

isCustomMode.value = false;

isCustomAmountConfirmed.value = false;

customAmount.value = "";

};

const handleCustomClick = () => {

if (isCustomAmountConfirmed.value) {

isCustomMode.value = false;

isCustomAmountConfirmed.value = false;

customAmount.value = "";

donationAmount.value = null;

} else {

isCustomMode.value = true;

selectedTip.value = null;

donationAmount.value = null;

}

};

const handleCustomAmountChange = (event) => {

customAmount.value = event.target.value.replace(/[^\d]/g, "");

};

const confirmCustomAmount = () => {

if (customAmount.value) {

isCustomAmountConfirmed.value = true;

isCustomMode.value = false;

donationAmount.value = parseInt(customAmount.value);

}

};

const handleKeyPress = (event) => {

if (event.key === "Enter") {

confirmCustomAmount();

}

};

const displayAmount = computed(() => {

if (isCustomAmountConfirmed.value && customAmount.value) {

const amount = parseInt(customAmount.value);

return `${amount.toLocaleString()} ₽`;

}

return "ввести свою сумму";

});

const selectedTipAmount = computed(() => {

if (selectedTip.value !== null) {

return tips.value[selectedTip.value].amount;

}

if (isCustomAmountConfirmed.value && customAmount.value) {

return parseInt(customAmount.value);

}

return null;

});

const makeDonation = () => {

if (!donationAmount.value || donationAmount.value <= 0) {

showNotification.value = true;

notificationMessage.value = "Выберите сумму чаевых";

setTimeout(() => (showNotification.value = false), 3000);

return;

}

showPaymentPopup.value = true;

};

const confirmPayment = async () => {

try {

const newCollected =

(collection.value.collected || 0) + donationAmount.value;

const updatedCollection = await auth.updateCollection(

collection.value.id,

collection.value.title,

collection.value.description,

collection.value.goal,

newCollected

);

collection.value.collected = updatedCollection.collected;

showPaymentPopup.value = false;

showNotification.value = true;

notificationMessage.value = `Чаевые на ${donationAmount.value.toLocaleString()} ₽ отправлены успешно`;

setTimeout(() => (showNotification.value = false), 3000);

donationAmount.value = null;

selectedTip.value = null;

isCustomMode.value = false;

isCustomAmountConfirmed.value = false;

customAmount.value = "";

} catch (error) {

console.error("Ошибка при отправке чаевых:", error);

showPaymentPopup.value = false;

showNotification.value = true;

notificationMessage.value = "Ошибка при отправке чаевых";

setTimeout(() => (showNotification.value = false), 3000);

}

};

const closePaymentPopup = (event) => {

if (!event.target.closest(".payment-popup-content")) {

showPaymentPopup.value = false;

}

};

</script>

<template>

<main>

<div class="container" v-if="collection">

<div class="header-wrapper">

<div class="card-title">оставить чаевые</div>

<div class="card-order\_info">

по чеку

<span class="total-primary">{{ orderTotal.toLocaleString() }} ₽</span>

</div>

</div>

<div class="tip-selector">

<button

v-for="(tip, index) in tips"

:key="index"

class="tip-selector\_\_button tip-selector\_\_button--percentage"

:class="{

'tip-selector\_\_button--active': selectedTip === index,

'tip-selector\_\_button--no-hover': selectedTip === index,

}"

@click="handleTipSelect(index)"

>

<span class="tip-selector\_\_percentage">

{{ tip.percentage }}<span class="percentage-gray">%</span>

</span>

<span class="tip-selector\_\_amount">{{ tip.amount }} ₽</span>

</button>

<div class="tip-selector\_\_custom-wrapper">

<button

v-if="!isCustomMode"

class="tip-selector\_\_button tip-selector\_\_button--custom"

:class="{

'tip-selector\_\_button--active': isCustomAmountConfirmed,

'tip-selector\_\_button--no-hover': isCustomAmountConfirmed,

}"

@click="handleCustomClick"

>

<span v-if="!isCustomAmountConfirmed">

ввести свою сумму <span class="percentage-gray">→</span>

</span>

<span v-else class="tip-selector\_\_amount">{{ displayAmount }}</span>

</button>

<div v-else class="tip-selector\_\_input-wrapper">

<input

type="text"

v-model="customAmount"

@input="handleCustomAmountChange"

@keypress="handleKeyPress"

class="tip-selector\_\_input"

placeholder="введите сумму"

@blur="confirmCustomAmount"

/>

<span class="tip-selector\_\_currency">₽</span>

</div>

</div>

</div>

<div class="user-info">

<div class="card-order\_info">для кого</div>

<div class="user-info\_\_card">

<div class="user-info\_\_header">

<div class="user-info\_\_text">

<p class="user-info\_\_collecting">{{ collection.title }}</p>

<div class="user-info\_\_goal">

{{ collection.description }}

</div>

</div>

<div class="user-info\_\_avatar">

<img src="@/assets/img/avatars.jpg" alt="User avatar" />

</div>

</div>

<div class="user-info\_\_progress">

<div class="user-info\_\_progress-text">

Собрано {{ collection.collected.toLocaleString() }} из

{{ collection.goal.toLocaleString() }} ₽

</div>

<div class="user-info\_\_progress-bar">

<div

class="user-info\_\_progress-fill"

:style="{

width: `${(collection.collected / collection.goal) \* 100}%`,

}"

></div>

</div>

</div>

</div>

<div class="action-buttons">

<button

class="action-button action-button--primary"

@click="makeDonation"

:disabled="!donationAmount"

>

Оплатить

</button>

<button class="action-button action-button--secondary">

Другие способы оплаты

</button>

</div>

<div class="service-info">

сервис представляет <span class="service-info\_\_brand">дайчаевые</span>

</div>

</div>

</div>

<div v-else-if="error" class="error-message">

{{ error }}

</div>

<AlertsLogin

v-if="showNotification"

:message="notificationMessage"

:duration="3000"

@close="showNotification = false"

/>

<div

v-if="showPaymentPopup"

class="payment-overlay"

@click="closePaymentPopup"

>

<div class="payment-popup-content">

<p>

Подтвердите оплату на сумму {{ donationAmount.toLocaleString() }} ₽

</p>

<button class="confirm-payment-button" @click="confirmPayment">

Подтвердить

</button>

<button class="cancel-payment-button" @click="showPaymentPopup = false">

Отмена

</button>

</div>

</div>

</main>

</template>

<style scoped>

.container {

padding: 0 16px;

}

.card-title {

font-family: "Inter";

width: 343px;

color: #fff;

font-size: 50px;

font-weight: 700;

line-height: 50px;

letter-spacing: -0.75px;

}

.card-order\_info {

font-family: "Inter";

color: #fff;

font-size: 16px;

font-weight: 600;

line-height: 20px;

letter-spacing: -0.24px;

}

.total-primary {

padding-top: 12px;

position: relative;

display: inline-block;

background: linear-gradient(180deg, #f80 0%, #ef5000 100%);

background-clip: text;

-webkit-background-clip: text;

-webkit-text-fill-color: transparent;

}

.total-primary::after {

content: "";

position: absolute;

left: 0;

bottom: 0;

width: 100%;

height: 1px;

border-bottom: 1px dotted #f80;

}

.tip-selector {

padding-top: 32px;

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 8px;

border-radius: 12px;

}

.tip-selector\_\_button {

background-color: #303030;

color: #fff;

border: none;

border-radius: 10px;

padding: 12px;

text-align: center;

cursor: pointer;

transition: background-color 0.2s ease;

}

.tip-selector\_\_button:hover {

background-color: #444;

}

.tip-selector\_\_button--active {

background-color: #f80;

}

.tip-selector\_\_button--no-hover:hover {

background-color: #f80 !important;

}

.tip-selector\_\_button--percentage {

display: flex;

flex-direction: column;

justify-content: center;

align-items: flex-start;

}

.tip-selector\_\_percentage {

font-family: "Inter";

font-weight: 500;

font-size: 28px;

line-height: 100%;

letter-spacing: -0.01em;

margin-bottom: 12px;

color: #fff;

}

.percentage-gray {

color: rgba(255, 255, 255, 0.4);

}

.tip-selector\_\_amount {

border-radius: 6px;

padding: 2px 4px;

font-family: "Inter";

font-style: italic;

font-weight: 500;

font-size: 18px;

line-height: 120%;

letter-spacing: -0.01em;

color: #fff;

background: #000;

}

.tip-selector\_\_button--custom {

font-family: "Inter";

grid-column: 1 / -1;

font-weight: 500;

font-size: 16px;

line-height: 125%;

letter-spacing: -0.01em;

color: #fff;

width: 100%;

height: 48px;

}

.tip-selector\_\_custom-wrapper {

grid-column: 1 / -1;

width: 100%;

}

.tip-selector\_\_input-wrapper {

position: relative;

width: 100%;

height: 48px;

display: block;

}

.tip-selector\_\_input {

width: 100%;

height: 100%;

padding: 12px 0px;

text-align: center;

background-color: #303030;

border: none;

border-radius: 10px;

color: #fff;

font-family: "Inter";

font-size: 16px;

font-weight: 500;

outline: none;

transition: background-color 0.2s ease;

box-sizing: border-box;

}

.tip-selector\_\_input:hover {

background-color: #444;

}

.tip-selector\_\_input:focus {

background-color: #444;

box-shadow: 0 0 0 2px #f80;

}

.tip-selector\_\_currency {

position: absolute;

right: 12px;

top: 50%;

transform: translateY(-50%);

color: rgba(255, 255, 255, 0.3);

font-family: "Inter";

pointer-events: none;

}

.tip-selector\_\_button--custom .tip-selector\_\_amount {

display: inline-block;

background: #000;

padding: 2px 4px;

border-radius: 6px;

font-style: italic;

}

.user-info {

padding-top: 32px;

}

.user-info\_\_card {

margin-top: 8px;

padding: 16px;

background-color: #303030;

border-radius: 12px;

}

.user-info\_\_header {

display: flex;

justify-content: space-between;

align-items: flex-start;

margin-bottom: 16px;

}

.user-info\_\_text {

flex-grow: 1;

}

.user-info\_\_collecting {

font-family: "Inter";

font-weight: 600;

font-size: 14px;

line-height: 114%;

letter-spacing: -0.01em;

color: #8e8e8e;

}

.user-info\_\_goal {

color: #fff;

font-family: "Inter";

font-weight: 600;

font-size: 20px;

line-height: 140%;

letter-spacing: -0.01em;

color: #fff;

gap: 8px;

}

.user-info\_\_emoji {

font-size: 20px;

}

.user-info\_\_avatar {

width: 52px;

height: 52px;

border-radius: 50%;

overflow: hidden;

}

.user-info\_\_avatar img {

width: 100%;

height: 100%;

object-fit: cover;

}

.user-info\_\_progress {

padding-top: 12px;

border-top: 1px solid #4b4b4b;

}

.user-info\_\_progress-text {

color: #fff;

font-family: "Inter";

font-weight: 600;

font-size: 14px;

line-height: 114%;

letter-spacing: -0.01em;

color: #fff;

margin-bottom: 8px;

}

.user-info\_\_progress-bar {

height: 8px;

background: linear-gradient(90deg, #f2eaea 0%, #4f4d4d 100%);

border-radius: 4px;

overflow: hidden;

}

.user-info\_\_progress-fill {

height: 100%;

background: linear-gradient(90deg, #f80 0%, #ef5000 100%);

border-radius: 4px;

}

.action-buttons {

display: flex;

flex-direction: column;

gap: 8px;

margin-top: 24px;

}

.action-button {

width: 100%;

font-family: "Inter";

font-weight: 600;

font-size: 16px;

line-height: 125%;

letter-spacing: -0.01em;

text-align: center;

border: none;

cursor: pointer;

}

.action-button--primary {

color: #ffffff;

box-shadow: inset 0 0 5px 0 rgba(255, 255, 255, 0.6),

0 2px 5px 0 rgba(241, 89, 10, 0.12);

background: linear-gradient(180deg, #f80 0%, #ef5000 100%),

radial-gradient(

94.42% 96.86% at 50.18% 0%,

rgba(255, 255, 255, 0.4) 0%,

rgba(255, 255, 255, 0) 100%

);

border-radius: 12px;

padding: 16px;

}

.action-button--primary:disabled {

background: #1a1a1a;

box-shadow: none;

cursor: not-allowed;

}

.action-button--secondary {

color: #000;

background: linear-gradient(0deg, #f2eaea 0%, #4f4d4d 100%);

border-radius: 12px;

padding: 16px;

}

.service-info {

margin-top: 32px;

text-align: center;

font-family: "Inter";

font-weight: 500;

font-size: 14px;

line-height: 114%;

letter-spacing: -0.01em;

text-align: center;

color: #696d69;

}

.service-info\_\_brand {

color: #ffffff;

}

.error-message {

color: #ff4444;

text-align: center;

margin-top: 20px;

}

.payment-overlay {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(0, 0, 0, 0.7);

display: flex;

justify-content: center;

align-items: center;

z-index: 1000;

}

.payment-popup-content {

background: #303030;

padding: 20px;

border-radius: 12px;

text-align: center;

position: relative;

width: 300px;

animation: fadeIn 0.3s ease-in-out;

}

.payment-popup-content p {

font-family: "Inter";

font-weight: 600;

font-size: 18px;

color: #fff;

margin-bottom: 20px;

}

.confirm-payment-button,

.cancel-payment-button {

width: 100%;

font-family: "Inter";

font-weight: 600;

font-size: 16px;

line-height: 125%;

letter-spacing: -0.01em;

text-align: center;

border: none;

cursor: pointer;

padding: 10px;

border-radius: 12px;

margin-bottom: 10px;

}

.confirm-payment-button {

color: #ffffff;

background: linear-gradient(180deg, #f80 0%, #ef5000 100%);

}

.cancel-payment-button {

color: #000;

background: linear-gradient(0deg, #f2eaea 0%, #4f4d4d 100%);

}

@keyframes fadeIn {

from {

opacity: 0;

transform: scale(0.8);

}

to {

opacity: 1;

transform: scale(1);

}

}

</style>

frontend/src/views/LoginPage.vue:  
<script setup>

import { auth } from "@/api/auth";

import { ref } from "vue";

import { useRouter } from "vue-router";

const router = useRouter();

const email = ref("");

const password = ref("");

const showPassword = ref(false);

const errorMessage = ref("");

const handleSubmit = async () => {

try {

const response = await auth.login(email.value, password.value);

if (response.success) {

router.push("/dashboard");

}

} catch (error) {

errorMessage.value = error.message || "Ошибка при входе";

}

};

const togglePasswordVisibility = () => {

showPassword.value = !showPassword.value;

};

</script>

<template>

<main>

<div class="container">

<div class="header-wrapper">

<div class="card-title">вход в аккаунт</div>

</div>

<form @submit.prevent="handleSubmit" class="login-form">

<div v-if="errorMessage" class="error-message">{{ errorMessage }}</div>

<div class="input-container">

<div class="input-wrapper">

<input

v-model="email"

type="text"

class="tip-selector\_\_input"

placeholder="ваш e-mail"

/>

</div>

<div class="input-wrapper password-wrapper">

<input

v-model="password"

:type="showPassword ? 'text' : 'password'"

class="tip-selector\_\_input"

placeholder="пароль"

/>

<button

type="button"

class="password-toggle"

@click="togglePasswordVisibility"

>

<img src="/src/assets/img/icons/eye.svg" alt="" />

</button>

</div>

</div>

<div class="action-buttons">

<button type="submit" class="action-button action-button--primary">

войти

</button>

<button

type="button"

class="action-button action-button--secondary"

@click="$router.push('/auth')"

>

у меня нет аккаунта

</button>

</div>

</form>

</div>

</main>

</template>

<style scoped>

.container {

padding: 0 16px;

}

.card-title {

font-family: "Inter";

width: 343px;

color: #fff;

font-size: 50px;

font-weight: 700;

line-height: 50px;

letter-spacing: -0.75px;

}

.login-form {

margin-top: 140px;

display: flex;

flex-direction: column;

gap: 8px;

}

.input-container {

background-color: #303030;

border-radius: 12px;

}

.input-wrapper {

position: relative;

width: 100%;

height: 48px;

}

.password-wrapper {

position: relative;

border-top: 1px solid rgba(255, 255, 255, 0.08);

}

.error-message {

font-family: "Inter";

font-weight: 500;

font-size: 14px;

color: #ff4444;

margin-bottom: 8px;

text-align: center;

}

.password-toggle {

position: absolute;

right: 12px;

top: 50%;

transform: translateY(-50%);

background: none;

border: none;

color: rgba(255, 255, 255, 0.3);

cursor: pointer;

padding: 0;

font-size: 16px;

user-select: none;

}

.tip-selector\_\_input {

width: 100%;

height: 100%;

padding: 12px 16px;

border: none;

color: #fff;

font-family: "Inter";

font-size: 16px;

font-weight: 500;

outline: none;

transition: background-color 0.2s ease;

box-sizing: border-box;

}

.action-buttons {

display: flex;

flex-direction: column;

margin-top: 24px;

}

.action-button {

width: 100%;

font-family: "Inter";

font-weight: 500;

font-size: 16px;

line-height: 125%;

letter-spacing: -0.01em;

text-align: center;

border: none;

cursor: pointer;

}

.action-button--primary {

color: #ffffff;

border-radius: 12px;

padding: 16px;

background-color: #1a1a1a;

}

.action-button--secondary {

color: #ffffff;

border-radius: 12px;

padding: 16px;

}

</style>

frontend/src/views/RegistrationPage.vue:

<script setup>

import { auth } from "@/api/auth";

import { ref } from "vue";

import { useRouter } from "vue-router";

const router = useRouter();

const email = ref("");

const password = ref("");

const showPassword = ref(false);

const errorMessage = ref("");

const handleSubmit = async () => {

try {

const response = await auth.register(email.value, password.value);

if (response.success) {

router.push("/login");

}

} catch (error) {

errorMessage.value = error.message;

}

};

const togglePasswordVisibility = () => {

showPassword.value = !showPassword.value;

};

</script>

<template>

<main>

<div class="container">

<div class="header-wrapper">

<div class="card-title">регистрация аккаунта</div>

</div>

<form @submit.prevent="handleSubmit" class="login-form">

<div v-if="errorMessage" class="error-message">{{ errorMessage }}</div>

<div class="input-container">

<div class="input-wrapper">

<input

v-model="email"

type="text"

class="tip-selector\_\_input"

placeholder="ваш e-mail"

/>

</div>

<div class="input-wrapper password-wrapper">

<input

v-model="password"

:type="showPassword ? 'text' : 'password'"

class="tip-selector\_\_input"

placeholder="пароль"

/>

<button

type="button"

class="password-toggle"

@click="togglePasswordVisibility"

>

<img src="/src/assets/img/icons/eye.svg" alt="" />

</button>

</div>

</div>

<div class="action-buttons">

<button type="submit" class="action-button action-button--primary">

зарегистрироваться

</button>

<button

type="button"

class="action-button action-button--secondary"

@click="$router.push('/login')"

>

уже есть аккаунт

</button>

</div>

</form>

</div>

</main>

</template>

<style scoped>

.container {

padding: 0 16px;

}

.card-title {

font-family: "Inter";

width: 343px;

color: #fff;

font-size: 50px;

font-weight: 700;

line-height: 50px;

letter-spacing: -0.75px;

}

.login-form {

margin-top: 140px;

display: flex;

flex-direction: column;

gap: 8px;

}

.input-container {

background-color: #303030;

border-radius: 12px;

}

.input-wrapper {

position: relative;

width: 100%;

height: 48px;

}

.password-wrapper {

position: relative;

border-top: 1px solid rgba(255, 255, 255, 0.08);

}

.password-toggle {

position: absolute;

right: 12px;

top: 50%;

transform: translateY(-50%);

background: none;

border: none;

color: rgba(255, 255, 255, 0.3);

cursor: pointer;

padding: 0;

font-size: 16px;

user-select: none;

}

.tip-selector\_\_input {

width: 100%;

height: 100%;

padding: 12px 16px;

border: none;

color: #fff;

font-family: "Inter";

font-size: 16px;

font-weight: 500;

outline: none;

transition: background-color 0.2s ease;

box-sizing: border-box;

}

.action-buttons {

display: flex;

flex-direction: column;

margin-top: 24px;

}

.error-message {

font-family: "Inter";

font-weight: 500;

font-size: 14px;

color: #ff4444;

margin-bottom: 8px;

text-align: center;

}

.action-button {

width: 100%;

font-family: "Inter";

font-weight: 500;

font-size: 16px;

line-height: 125%;

letter-spacing: -0.01em;

text-align: center;

border: none;

cursor: pointer;

}

.action-button--primary {

color: #ffffff;

border-radius: 12px;

padding: 16px;

background-color: #1a1a1a;

}

.action-button--secondary {

color: #ffffff;

border-radius: 12px;

padding: 16px;

}

</style>