**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

|  |  |
| --- | --- |
| КП.09.02.07-3.24.212.22 ПЗ |  |

ВЕБ-ПРИЛОЖЕНИЕ УПРАВЛЕНИЕ ЛИЧНЫМИ ФИНАНСАМИ

«FINANCIO»

|  |  |  |
| --- | --- | --- |
| Председатель ВЦК: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.С. Александрова) |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Н.Р. Карпова) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (М.Р. Рудковский) |

Иркутск 2024

Содержание

[Содержание 2](#_Toc169470413)

[Введение 3](#_Toc169470414)

[1 Предпроектное исследование 5](#_Toc169470415)

[1.1 Описание предметной области 5](#_Toc169470416)

[1.2 Анализ инструментальных средств для разработки 6](#_Toc169470417)

[2 Техническое задание 9](#_Toc169470418)

[3 Проектирование веб-приложения 10](#_Toc169470419)

[3.1 Структурная схема веб-приложения 10](#_Toc169470420)

[3.2 Функциональная схема веб-приложения 11](#_Toc169470421)

[3.3 Проектирование базы данных 13](#_Toc169470422)

[3.4 Проектирование пользовательского интерфейса 15](#_Toc169470423)

[3.4.1 Разработка прототипов интерфейса 15](#_Toc169470424)

[3.4.2 Выбор цветовой гаммы и шрифтов 21](#_Toc169470425)

[3.4.3 Разработка элементов интерфейса 22](#_Toc169470426)

[3.4.4 Разработка дизайн макетов 26](#_Toc169470427)

[4 Разработка веб-приложения 31](#_Toc169470428)

[4.1 Разработка интерфейса веб-приложения 31](#_Toc169470429)

[4.2 Разработка базы данных веб-приложения 39](#_Toc169470430)

[4.3 Разработка веб-приложения 40](#_Toc169470431)

[5 Документирование программного продукта 42](#_Toc169470432)

[5.1 Руководство пользователя веб приложения 42](#_Toc169470433)

[ЗАКЛЮЧЕНИЕ 50](#_Toc169470434)

[Список используемых источников 51](#_Toc169470435)

[ПРИЛОЖЕНИЕ Б листинг кода контроллеров Laravel 56](#_Toc169470436)

# Введение

Тема «управление личными финансами» для курсового проекта имеет высокую актуальность. Создание веб-приложения для управления личными финансами позволит пользователям отслеживать свои доходы и расходы, составлять бюджеты, устанавливать финансовые цели, анализировать свои финансовые потоки и получать рекомендации по оптимизации расходов.

Я выбрал именно формат веб-приложения, потому что оно будет доступно на любом устройстве без надобности в установке приложения и будет синхронизироваться в облаке, и его можно будет посмотреть на компьютере и удобно работать с ним. Банковские приложения не могут позволить посмотреть такую четкую и наглядную аналитику, также у них есть ограничение истории просмотра финансов

Целью курсового проекта является создание веб-приложения с удобным и функциональным интерфейсом, чтобы люди могли следить за своими финансами и в доступной форме отслеживать свои расходы и доходы, которые будут сохраняться в базе данных навсегда, в то время как банковские приложения хранят лишь в лучшем случае один год.

Задачи курсового проекта включают в себя следующее:

* Разработка системы отслеживания доходов и расходов
* Создание функции автоматической категоризации транзакций
* Реализация инструментов для планирования бюджета
* Внедрение функции создания и отслеживания финансовых целей
* Разработка механизма подсчета сбережений
* Создание функции аналитики и отчетов о состоянии бюджета и финансов пользователя

Понятия в курсовой работе:

Доходы - денежные средства, поступающие в бюджет от населения, организаций, учреждений и других бюджетов, за исключением средств, являющихся источниками финансирования дефицита бюджета.

Расходы - выплачиваемые из бюджета денежные средства, за исключением средств, являющихся источниками финансирования дефицита бюджета.

Лента операций – список расходов и доходов, который пользователь будет пополнять по мере использования веб-приложения.

БД – база данных.

# Предпроектное исследование

## Описание предметной области

В качестве предметной области выбрана тематика бюджетирования, трекинга расходов и доходов, управления своими финансами, а также финансового планирования на будущее. На основе этого транзакции делятся на категории, которые пользователь может добавлять в специальном модальном окне при добавлении новых операций расходов в ленту операций.

Веб-приложение должно иметь возможность системы регистрации и входа для пользователей. Это необходимо для сохранения информации пользователя в его учётной записи, доступ в которую и регистрация будет осуществляться по номеру телефона и почты.

Обычный пользователь может составлять ленту операций за текущий и прошлые месяца по желанию. Пользователь может просматривать ленту операций в своём личном кабинете. Пользователь может удалять и добавлять операции в ленту операций.

Для незарегистрированных пользователей на главной странице сайта есть визитка, рекламирующая для пользователя веб-приложение, рассказывающая о его функциях и отвечающая на вопросы касательно того, почему он должен пользоваться именно этим веб-приложением. Также должна быть секция с ответами на часто задаваемые вопросы.

Система создается для обслуживания следующих групп пользователей:

* Пользователь;

Функции веб-приложения:

* Ведение БД
* Обеспечение непротиворечивости в логике веб-приложения
* Заполнение ленты операций удобно настраиваемыми транзакциями в ленту транзакций
* Возможность регистрации и входа в личный кабинет
* Ведение учёта зарегистрированных пользователей
* Возможность редактирования профиля пользователя в меню настроек

## Анализ инструментальных средств для разработки

Инструменты разработки программного продукта имеют огромное значение, поскольку определяют будущий результат проекта. Для проектирования структуры веб-приложения удобно использовать MySQL Workbench и Draw.io, а для дизайна — онлайн-сервис Figma. Веб-приложение будет состоять из клиентской и серверной частей. Для реализации клиентской части отлично подходят HTML5, CSS3, и JS. Серверная часть будет разработана на PHP с использованием стандартной базы данных SQLite.

PhpMyAdmin – веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL. PhpMyAdmin позволяет через браузер и не только осуществлять администрирование сервера MySQL, запускать команды SQL и просматривать содержимое таблиц и баз данных.

Draw.io – это удобное бесплатное онлайн-приложение для создания различных видов диаграмм и блок-схем. Оно используется для создания прототипов страниц.

Figma – это бесплатный удобный онлайн-сервис для создания прототипов сайтов или приложений, иллюстраций и векторной графики. В редакторе можно настроить совместную работу, вносить и обсуждать правки. Figma используется для создания наглядного ожидаемого дизайна проекта и для создания уникальных SVG-элементов в едином стиле.

HTML, CSS и JavaScript (JS) – используются для создания клиентской части веб-приложения. HTML предоставляет структуру страницы, CSS отвечает за оформление элементов, а JS обеспечивает динамическую интерактивность

PHP – это распространённый язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML.

Mysql – это компактная встраиваемая СУБД с открытым исходным кодом, которая обеспечивает возможность хранить, изменять и структурировать данные на стороне клиента.

Laravel фреймворк - бесплатный фреймворк с открытым программным кодом, который написан на PHP. Он является одной из самых популярных платформ для backend-разработки и активно применяется для разработки ПО – как для личных целей, так и для бизнеса.

Tailwind фреймворк - CSS-фреймворк с открытым исходным кодом, созданный Адамом Уэтеном и поддерживаемый Tailwind Labs. Особенность этой библиотеки в том, что, она не предопределяет CSS-классы отдельных элементов. Вместо этого она предоставляет служебные классы, которые можно объединять для стилизации каждого элемента.

Для хранения, изменения и структурирования информации веб-приложения выбрана база данных Mysql. Mysql является компактной встраиваемой СУБД с открытым исходным кодом. Она поддерживает динамическое типизирование данных и может хранить различные типы значений, такие как INTEGER, REAL, TEXT и BLOB. Благодаря простой структуре базы данных, Mysql обеспечивает простоту использования, портативность и отлично подходит для мобильной разработки и небольших автономных приложений.

Для обеспечения взаимосвязи между базой данных и серверной частью веб-приложения был выбран язык программирования PHP. PHP является высокоуровневым языком программирования с динамической строгой типизацией и автоматическим управлением памятью. Он обладает широким спектром библиотек и инструментов для работы с базами данных, что делает его отличным выбором для разработки веб-приложений.

После учета всех этих факторов был сделан выбор в пользу HTML5, CSS3, JS, Mysql и PHP для разработки веб-приложения «Управление личными финансами». Для разработки программного продукта было решено использовать редактор кода Visual Studio Code. Visual Studio Code обладает богатым функционалом, поддерживает множество различных языков программирования, включая PHP, HTML/CSS и JavaScript, и обеспечивает удобное добавление новых функций в веб-приложение. Кроме того, Visual Studio Code интегрируется с системой контроля версий GIT, что облегчает разработку в команде и отслеживание изменений. Visual Studio Code также обладает подсветкой синтаксиса, автоматическим сохранением и другими удобными функциями, что делает его отличным выбором для разработки веб-приложения.

Таблица 1 – Критерии анализа инструментов

|  |  |
| --- | --- |
| Критерий | Описание |
| Продуктивность и эффективность | Выбранные инструменты являются популярными и широко используемыми в сообществе разработчиков, что обеспечивает поддержку и эффективное решение задач. |
| Гибкость и масштабируемость | PHP и Laravel обладают гибкостью для реализации различных функционалов и легко масштабируются в процессе развития проекта. |
| Удобство и доступность | Средства разработки, выбранные для frontend и backend, обеспечивают удобную работу разработчиков, сокращая время разработки и обеспечивая высокое качество продукта. |

Выбранный набор инструментов обеспечивает эффективное развертывание веб-приложения электронных книг, отвечая требованиям современных технологий и потребностям конкретного проекта. Полученный анализ поможет оптимизировать процесс разработки и обеспечить реализацию успешных и качественных решений.

# Техническое задание

В начале разработки создавалось техническое задание, в котором указывались основные требования.

Для создания технического задания использовался стандарт ГОСТ 34.602-2020.

Согласно ГОСТ 34.602-2020 техническое задание должно включать следующие разделы:

1. общие сведения;
2. назначение и цели создания системы;
3. требования к системе в целом;
   1. требования к структуре и функционированию системы;
   2. требования к надежности;
   3. требования к безопасности;
   4. требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы;
4. требования к документированию;
5. состав и содержание работ по созданию системы.

Техническое задание на разработку приложения представлено в приложении А.

# Проектирование веб-приложения

## Структурная схема веб-приложения

Диаграмма прецедентов. В системе действующим лицом (актёром) является только пользователь, который взаимодействует с веб-приложением и через свой личный кабинет.

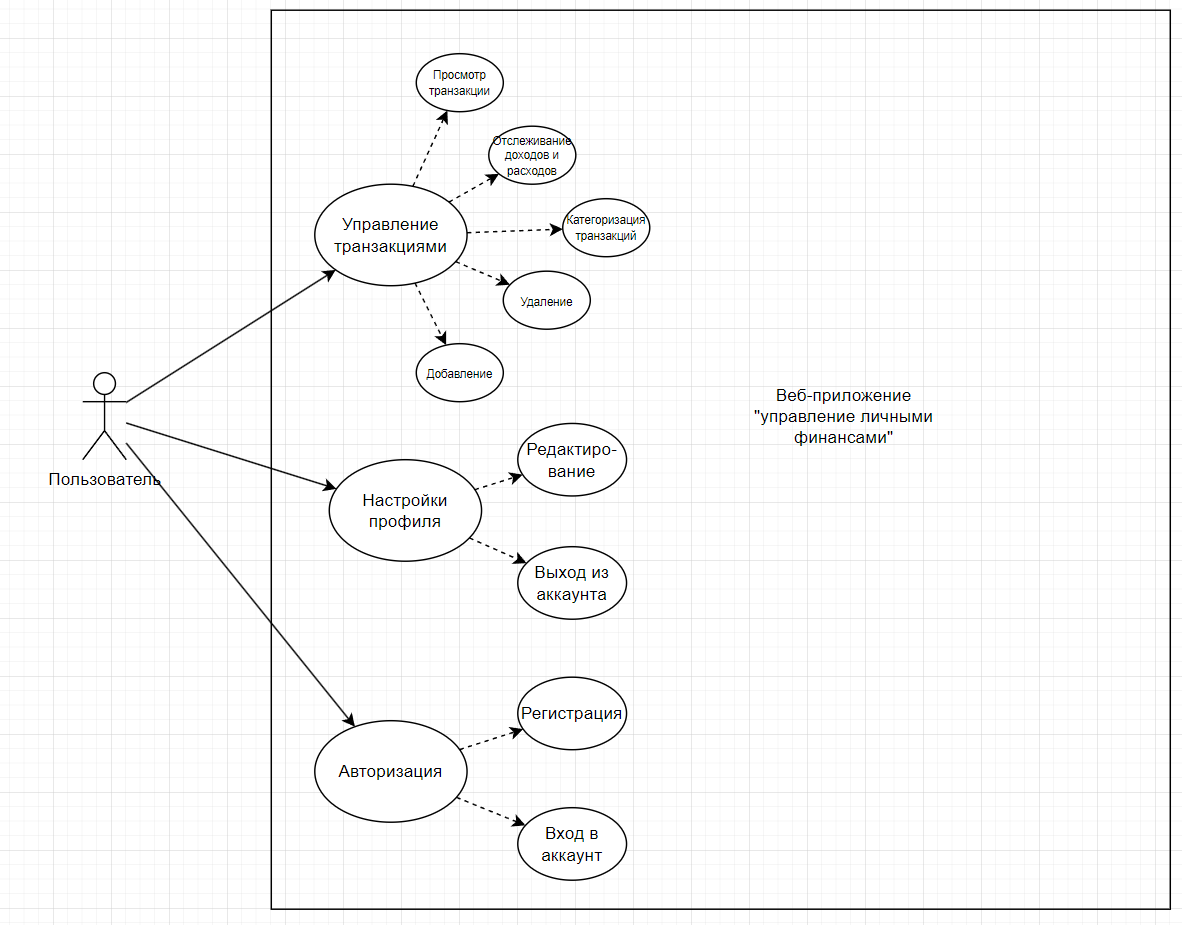


Рисунок 1 – Диаграмма прецедентов Uses CASE

Диаграмма потоков данных иллюстрирует потоки данных между пользователями и системой управления личными финансами. Пользователь взаимодействует с системой, вводя личные данные, добавляя расходы и доходы, устанавливая бюджеты и цели, просматривая статистику и отчеты, редактируя настройки и личные данные. Данные сохраняются в базе данных для последующего использования, генерации отчетов и отображения информации пользователю. Различным действиям соответствуют соответствующие процессы обработки данных системой

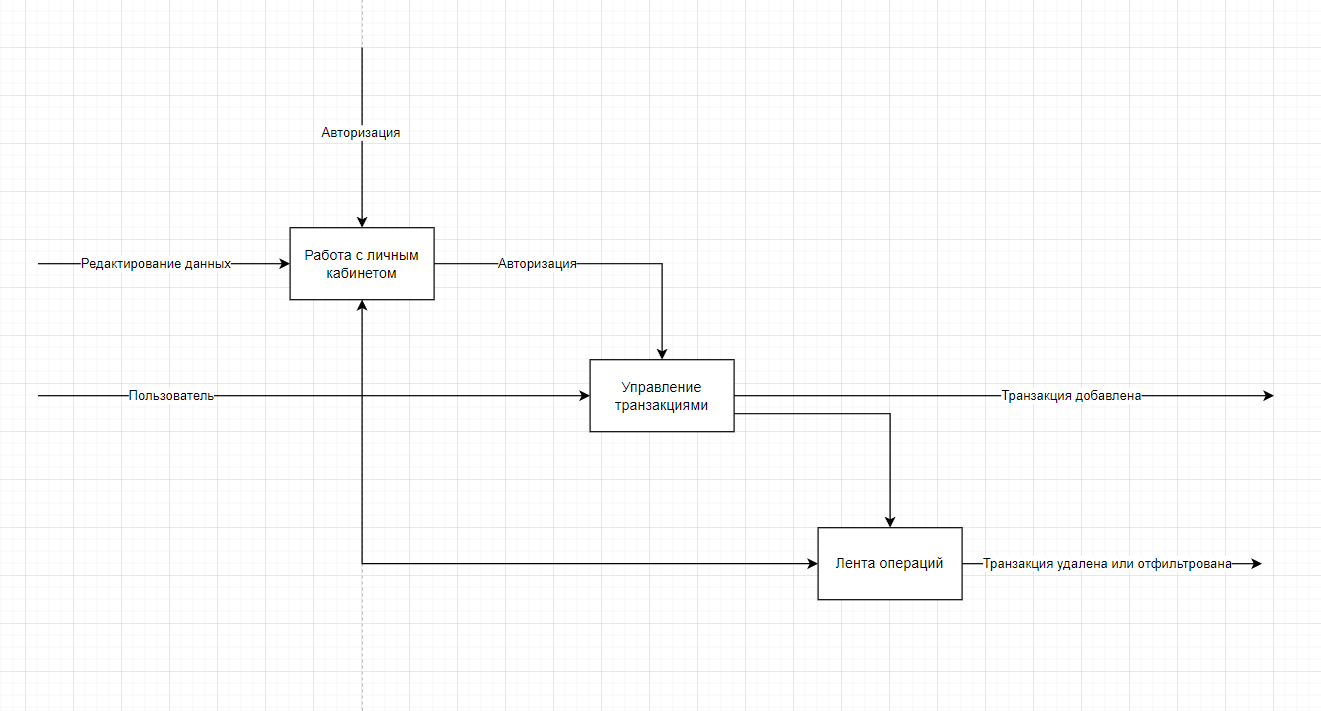


Рисунок 2 – Диаграмма потоков данных DFD

## Функциональная схема веб-приложения

Диаграмма классов: Пользователь представляет пользователя приложения с основными полями и методами для работы с данными пользователя. Транзакция: Класс, описывающий транзакции пользователя (доходы/расходы) с соответствующими полями и методами. Бюджет: Класс, представляющий бюджет пользователя на определенный период времени с соответствующими полями и методами. Система приложения: Класс, представляющий веб-приложение для управления личными финансами. Содержит списки пользователей, транзакций и бюджетов, а также методы для регистрации пользователей, добавления транзакций/бюджетов и другие операции.

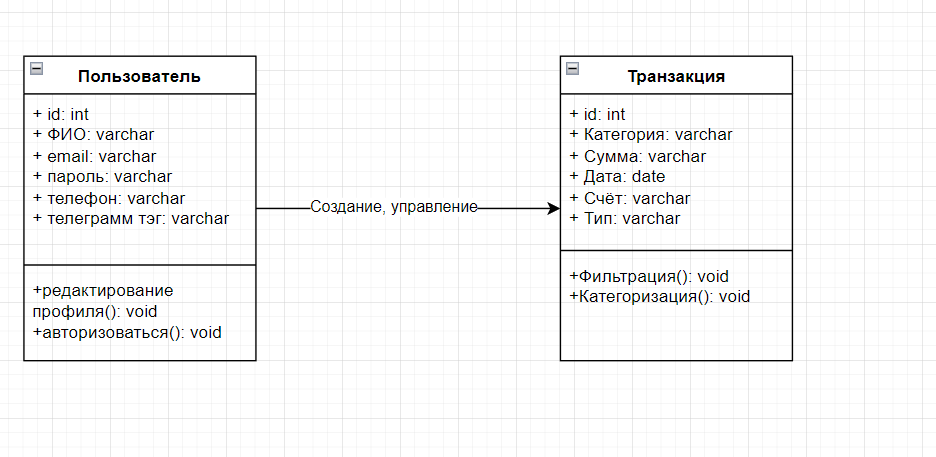


Рисунок 3 – Диаграмма классов

Диаграмма последовательности. Данная диаграмма последовательности показывает взаимодействие пользователя с веб-приложением Управление личными финансами. Пользователь выполняет различные действия, такие как регистрация, вход в учетную запись, добавление расходов и доходов, установка бюджетов и целей, просмотр статистики и отчетов, редактирование настроек и личных данных, а также выход из учетной записи. Система в свою очередь обрабатывает запросы пользователя и осуществляет необходимые действия, включая сохранение данных, генерацию отчетов и отображение информации.

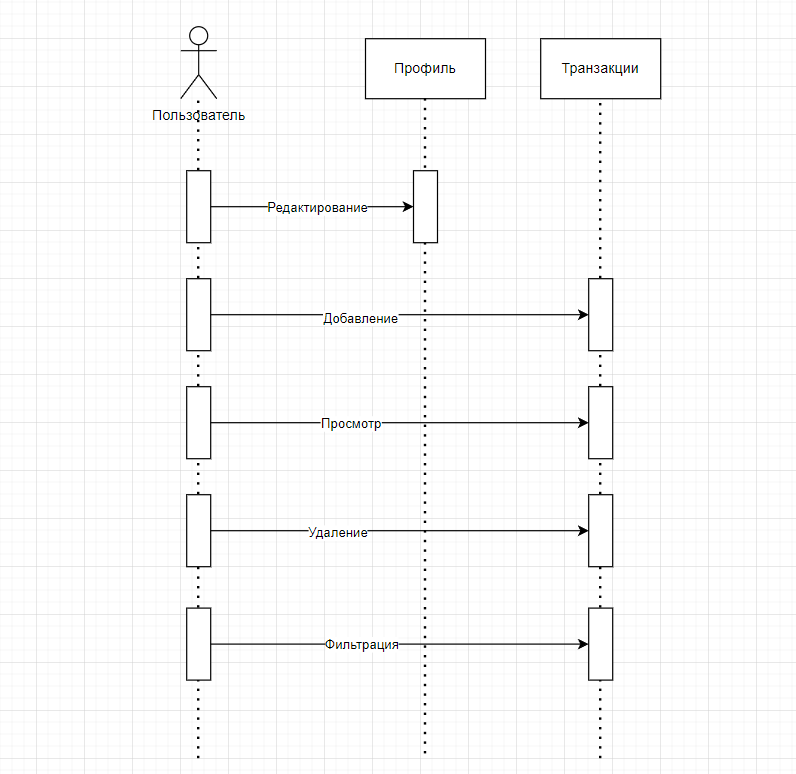


Рисунок 4 – Диаграмма последовательности

## Проектирование базы данных

Прежде чем приступить к разработке программного обеспечения необходимо спроектировать базу данных, а именно, определить с какими данными будут работать участники системы, и чем данные связаны между собой. В этом заключается процесс проектирования. Проектирование базы данных не ограничивается только этим этапом разработки, а проходит на протяжении всей разработки, до того момента пока в системе не появятся данные, которые нельзя потерять. Результатом проектирования базы данных является ER-модель, показанная на 5 рисунке.

Данная модель очень простая, но при этом является основой всего приложения благодаря ней пользователь может заносить данные в таблицу транзакций. После занесения в таблицу, каждая строчка выводится в личном кабинете пользователя в таблице с транзакциями. Таблица с пользователями обеспечивает возможность авторизации

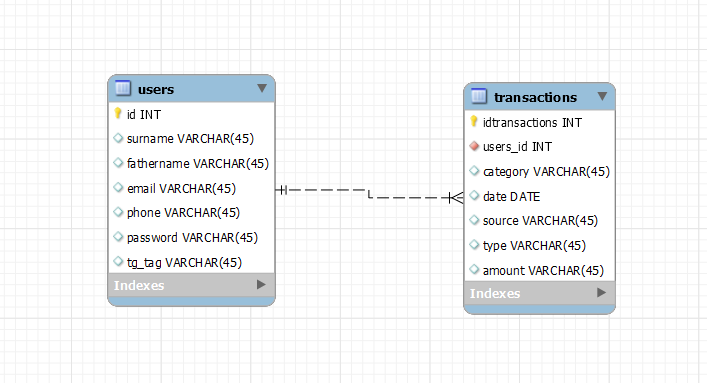


Рисунок 5 – ER-модель

Данная ER-модель соответствует третьей нормальной форме (3NF). Все атрибуты каждой сущности являются атомарными и не содержат повторяющихся групп. Нет транзитивных зависимостей между не ключевыми полями, и каждая сущность имеет один первичный ключ.

Описание ER-модели:

На таблицах 1 ‒ 2 представлено описание ER-модели интернет-магазина.

Таблица 2 ‒ users

|  |  |  |
| --- | --- | --- |
| Поля | Тип данных | Значение |
| id | Int(PK) | Идентификатор пользователя (ключ.) |
| name | Varchar(45) | Имя пользователя |
| surname | Varchar(45) | Фамилия пользователя |
| fathername | Varchar(45) | Отчество пользователя |
| email | Varchar(45) | Почта пользователя |
| phone | Varchar(45) | Телефон пользователя |
| password | Varchar(45) | Пароль пользователя |
| tg\_tag | Varchar(45) | Телеграмм тэг пользователя |

Таблица 3 – transactions

|  |  |  |
| --- | --- | --- |
| Поля | Тип данных | Значение |
| id | int(PK) | Идентификатор транзакции (ключ.) |
| user\_id | int | внешний ключ таблицы users |
| category | varchar(45) | категория транзакции |
| date | date | дата транзакции |
| source | varchar(45) | счёт транзакции |
| type | varchar(45) | тип транзакции |
| amount | varchar(45) | сумма транзакции |

## Проектирование пользовательского интерфейса

### Разработка прототипов интерфейса

Для разработки пользовательского интерфейса будет использован онлайн-редактор Figma, в котором удобно проектировать интерфейсы, создавать макеты сайтов, мобильных приложений, презентации, иллюстрации, логотипы и анимации. Весь дальнейший прототип дизайна выполнен в нём.

Главная страница веб-приложения содержит в основном описание функций и причины того, почему пользователь должен пользоваться именно данным продуктом. Шапка сайта содержит кнопки для перехода на страницу регистрации и входа в личный кабинет. Когда пользователь авторизован, то кнопки заменяются на иконку выхода из аккаунта, и ссылку на возвращение в личный кабинет, которая будет выводить имя пользователя, указанное в личном кабинете. Если имя не указано, то по умолчанию будет выводиться слово “пользователь”. Главный экран, на который попадает пользователь, приветствует слоганом компании и показывает интерфейс с транзакциями веб-приложения для открытости и понимания пользователем веб-приложения. Второй экран веб-приложения, поделённый на две части содержит функциональные преимущества продукта и продвигают его пользователю. Эти блоки просто содержат в себе текст, описывающий наши возможности. Блок с часто задаваемыми вопросами содержит в себе элемент аккордиона, который будет разворачиваться, и содержать в себе ответы на часто задаваемымые вопросами. Подвал сайта внешне такой же, как и шапка и не имеет никаких функциональных элементов.

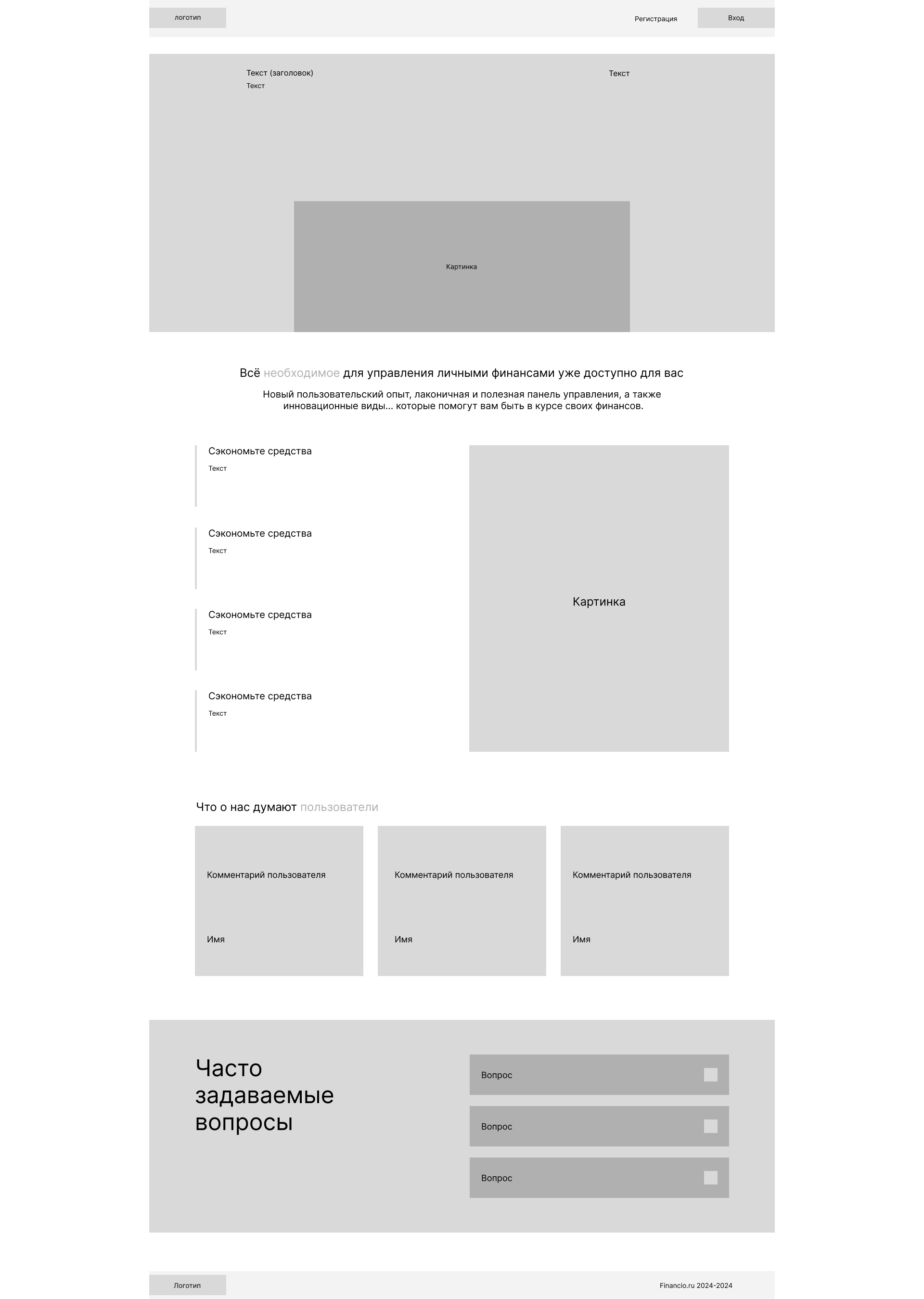


Рисунок 6 – Прототип главной страницы

Страница входа в личный кабинет выглядит следующим образом. Она содержит форму, в которой есть два поля ввода для электронной почты и пароля пользователя.

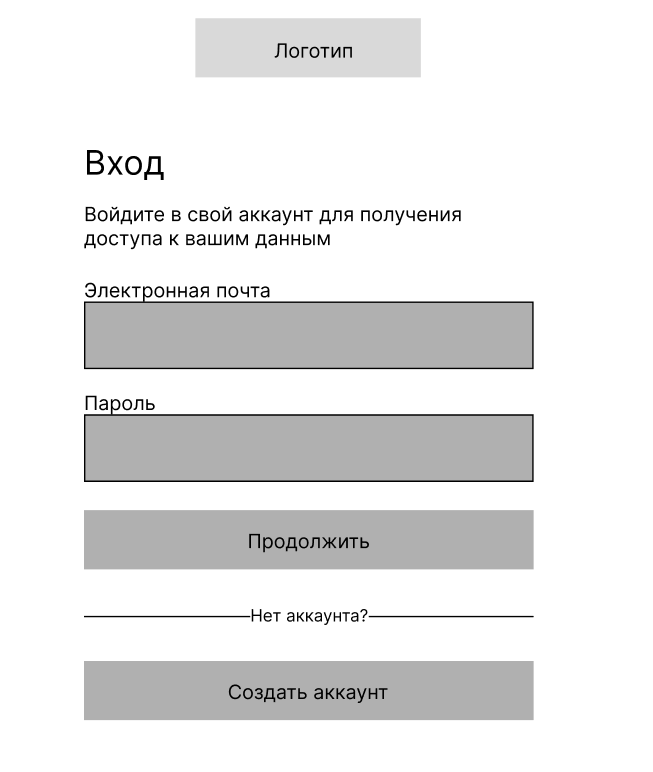


Рисунок 7 – Прототип страницы входа

Страница регистрации выглядит следующим образом. Она содержит форму, в которой есть три поля ввода для номера телефона, электронной почты и пароля пользователя.

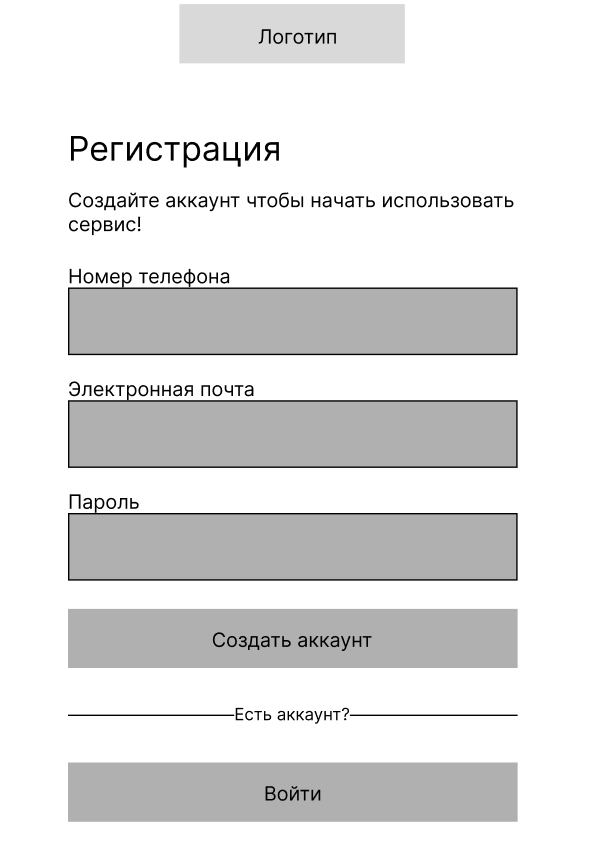


Рисунок 8 – Прототип страницы регистрации

Прототип страницы личного кабинета с транзакциями выглядит следующим образом: в нём объединены новый шаблон страницы и блок с общими данными и таблицей операций

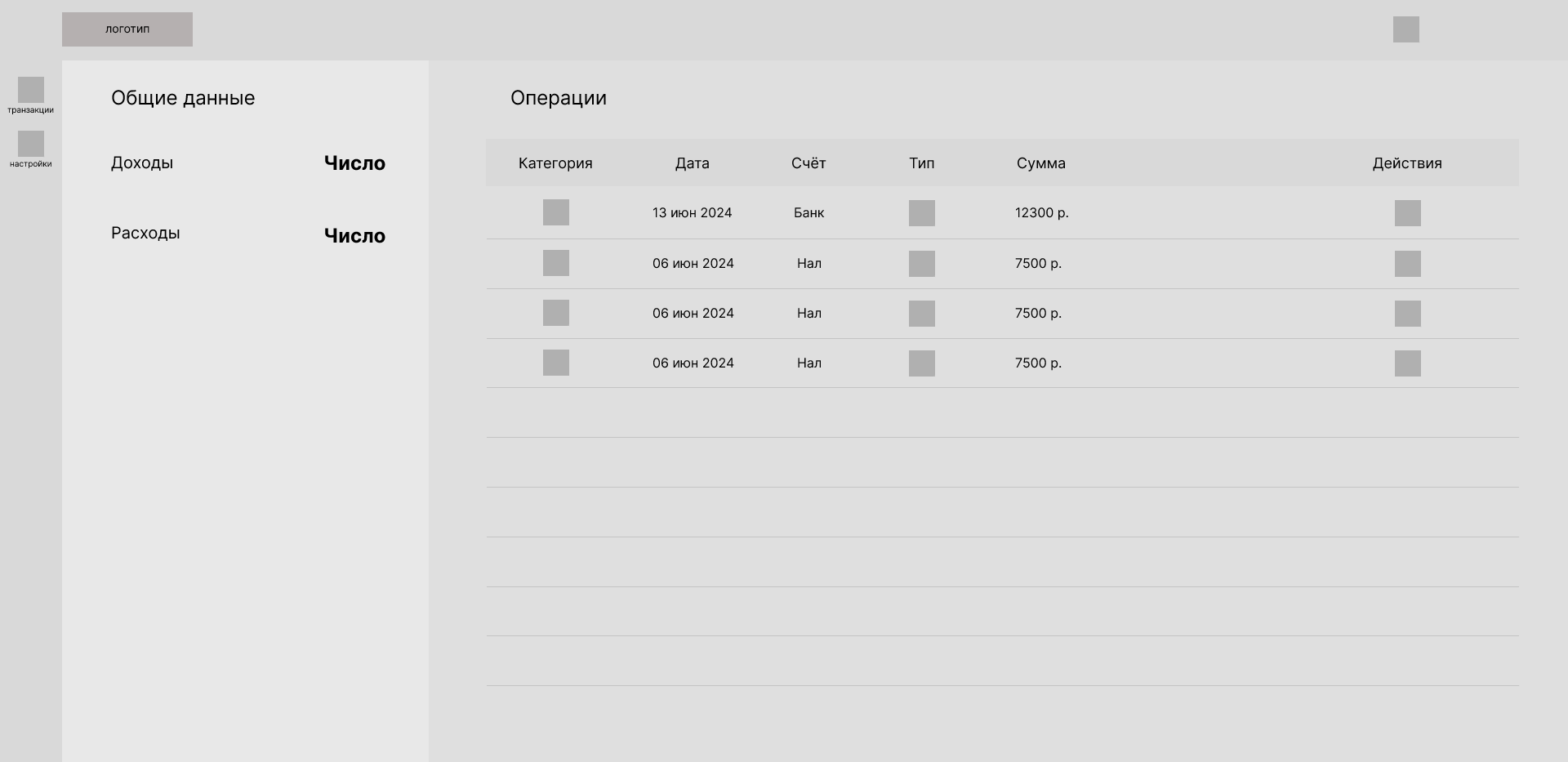


Рисунок 10 – Прототип страницы транзакций личного кабинета

Внешний вид страницы настроек профиля выглядит таким образом:

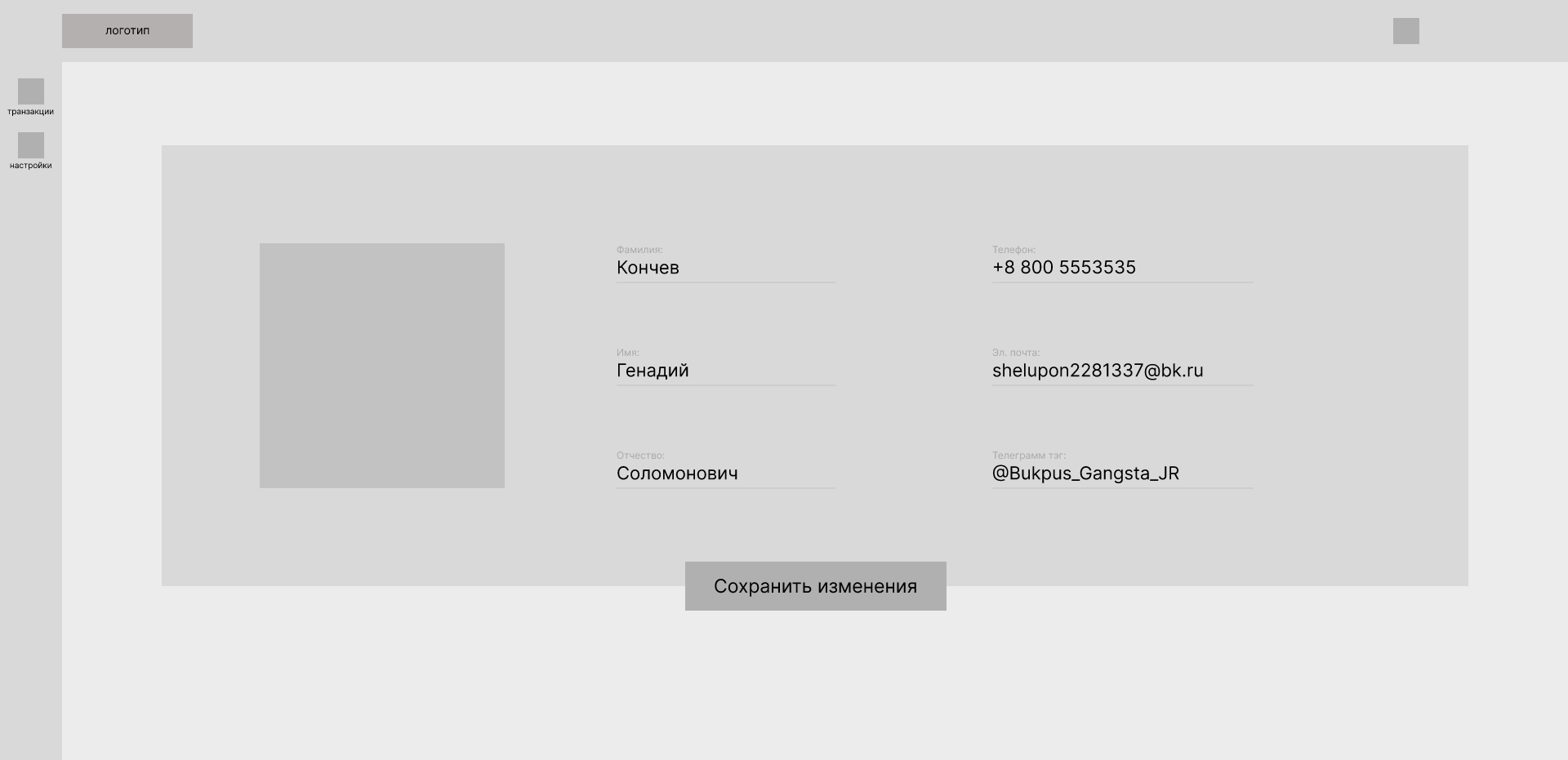
****

Рисунок 11 – Прототип страницы настроек профиля личного кабинета

Представленная диаграмма навигации показывает переход между страницами сайта

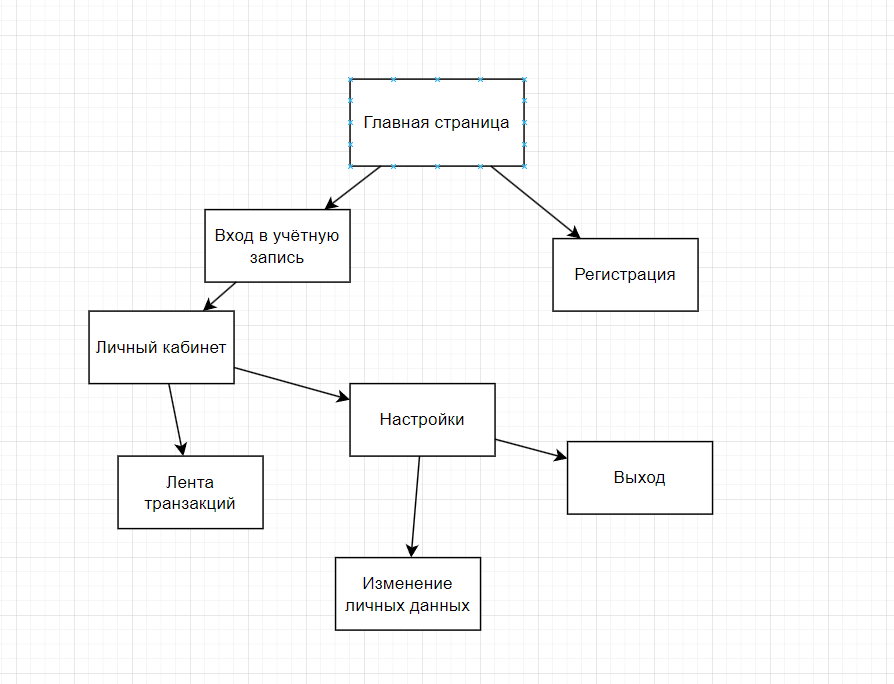


Рисунок 12 – Диаграмма навигации

Во время разработки прототипа интерфейса был учтен проведенный анализ аналогов, что помогло реализовать представленный вид прототипа веб-приложения «управление личными финансами». В ходе разработки прототипа была определена конечная структура интерфейса веб-приложения, которая не будет сильно отличаться от конечного продукта. Этот прототип позволит как и вместить весь функционал, так и в полной мере прорекламировать пользователю возможности веб-приложения.

### Выбор цветовой гаммы и шрифтов

Цветовая гамма веб-приложения будет исполнена в бледно-фиолетовых тонах с оттенками синего цвета. Основной задний фон будет белого цвета. Для подчёркивания важного текста будет использован цвет с таким hex-кодом: #4D52BC. Также он будет использован для подсвечивания текста при наведении на функциональные элементы. Цвет для всех блоков, у которых есть задний фон был выбран с таким hex-кодом: #EDF1FF. Некоторые элементы (кнопки, блоки внутри блоков, некоторые иконки и логотип) было решено сделать чуть тёмно-фиолетовыми с таким hex-кодом: C1CFFF. Некоторым текстовым блокам на страницах (преимущественно заголовкам) решено было дать чуть синеватый оттенок по отношению к чёрному с таким hex-кодом: #000C35.

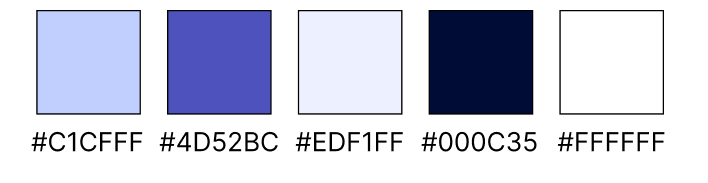


Рисунок 13 – Цветовая гамма

Шрифт в веб-приложении был выбран sans-seriff. Он будет использоваться на каждом текстовом блоке. Это официальный и стильный шрифт.

### Разработка элементов интерфейса

Данный пример кнопки будет встречаться везде в веб-приложении. Он имеет задний фон с цветовым кодом hex C1CFFF и имеет немного скруглённые края. По середине относительно вертикали и горизонтали будет находиться соответствующий текст.



Рисунок 14 – Кнопка

Данный блок является аккордионом. То есть при нажатии на него он разворачивается и показывает своё содержимое. Он имеет такой же цвет как и все кнопки и другие функциональные элементы на странице.

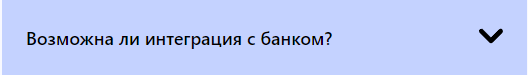
.

Рисунок 15 – Элемент аккордиона

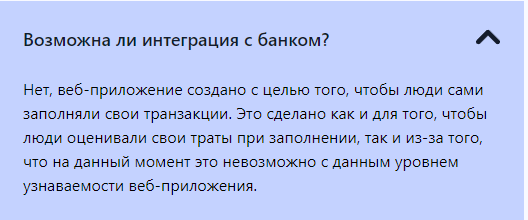
****

Рисунок 16 – Развёрнутый элемент аккордиона

На странице формы регистрации находится три функциональных поля ввода. У всех полей есть валидация и при неправильном формате ввода электронной почты, веб-приложение сообщит пользователю о неправильном формате.

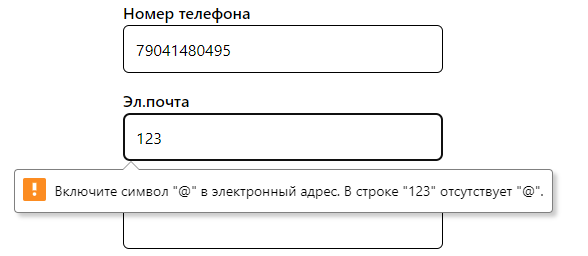


Рисунок 17 –Поля ввода регистрации аккаунта

На странице входа в аккаунт есть всего два поля ввода и при вводе неверного пароля для учётной записи, веб-приложение сообщит об этом пользователю.

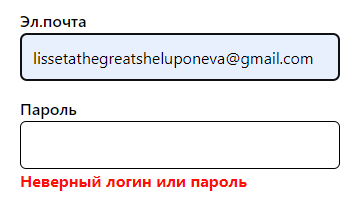


Рисунок 18 – Поля ввода входа в аккаунт

На странице профиля есть таблица с транзакциями, все части которой являются функциональными. Кнопка “новая транзакция” вызывает модальное окно с добавлением транзакции. При нажатии на иконку корзины в таблице слева, транзакция удаляется. При нажатии на пункт “категории” в таблице, вылезет модальное окно с выбором категории транзакции для фильтрации.



Рисунок 19 – Таблица транзакций

В блоке с общими данными подсчитываются общие доходы и расходы пользователя и выводятся на экран.

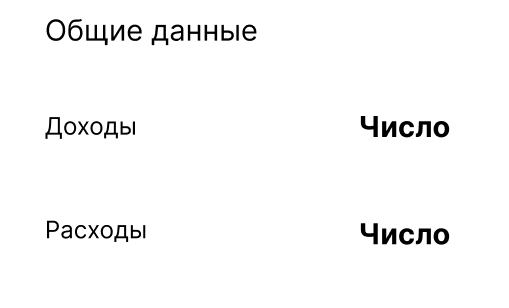


Рисунок 20 –Компонент с общими данными

Меню редактирования профиля содержит несколько полей ввода, в которые будут выводиться уже заданные в базе значения профиля пользователя и при желании их можно будет поменять при нажатии на кнопку «сохранить изменения».



Рисунок 21 – Компонент редактирования профиля

Боковое меню должно переносить пользователя между разделами личного кабинета.

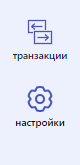


Рисунок 22 – Боковое меню навигации личного кабинета

### Разработка дизайн макетов

Дизайн главной страницы имеет белый задний фон. Контейнеры с информацией вместе с шапкой и подвалом сайта оформлены в цветовом коде hex: #EDF1FF. Шапка сайта содержит кнопки для перехода на страницу регистрации и входа в личный кабинет. Когда пользователь авторизован, то кнопки заменяются на иконку выхода из аккаунта и ссылку на возвращение в личный кабинет, которая будет выводить имя пользователя, указанное в личном кабинете. Если имя не указано, то по умолчанию будет выводится слово “пользователь”. Главный экран, на который попадает пользователь приветствует слоганом компании и показывает интерфейс с транзакциями веб-приложения для открытости и понимания пользователем веб-приложения. Второй экран веб-приложения, поделённый на две части содержит функциональные преимущества продукта и продвигают его пользователю. Следующий блок с предлагаемыми возможностями содержит три блока с задним фоном с цветовым hex-кодом: #EDF1FF. Эти блоки просто содержат в себе текст, описывающий наши возможности. Блок с часто задаваемыми вопросами содержит в себе элемент аккордиона, который будет разворачиваться и содержать в себе ответы на часто задаваемыми вопросами. Данный блок имеет тот же цветовой hex-код для контейнеров на главной странице: #EDF1FF. Подвал сайта внешне такой же, как и шапка и не имеет никаких функциональных элементов.

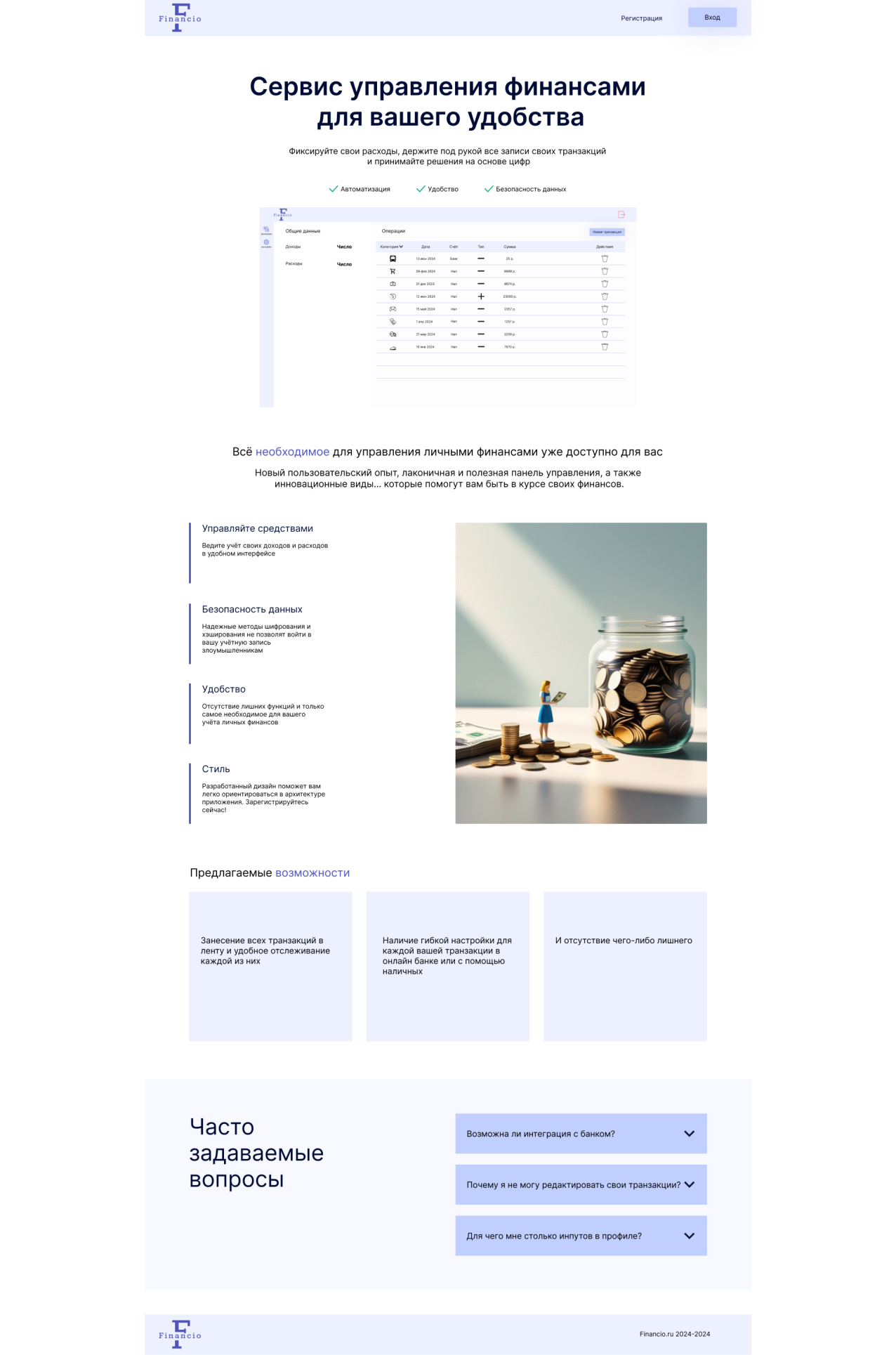


Рисунок 23 – Дизайн главной страницы

Форма регистрации будет просто добавлена на страницу с белым фоном и отсутствием шапки и подвала. Пользователь либо регистрируется, либо возвращается назад через кнопку “назад” в браузере.

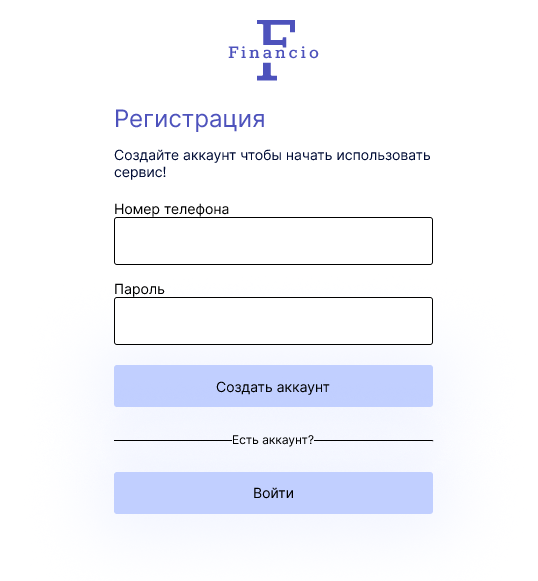


Рисунок 24 – Дизайн страницы регистрации

Аналогично со входом в аккаунт



Рисунок 25 – Дизайн страницы входа

Страница личного кабинета с лентой операций состоит из трёх компонентов: шаблон для личного кабинета, блок с общими данными, таблица с лентой операций. Оформление не отличается от ранее показанных компонентов.

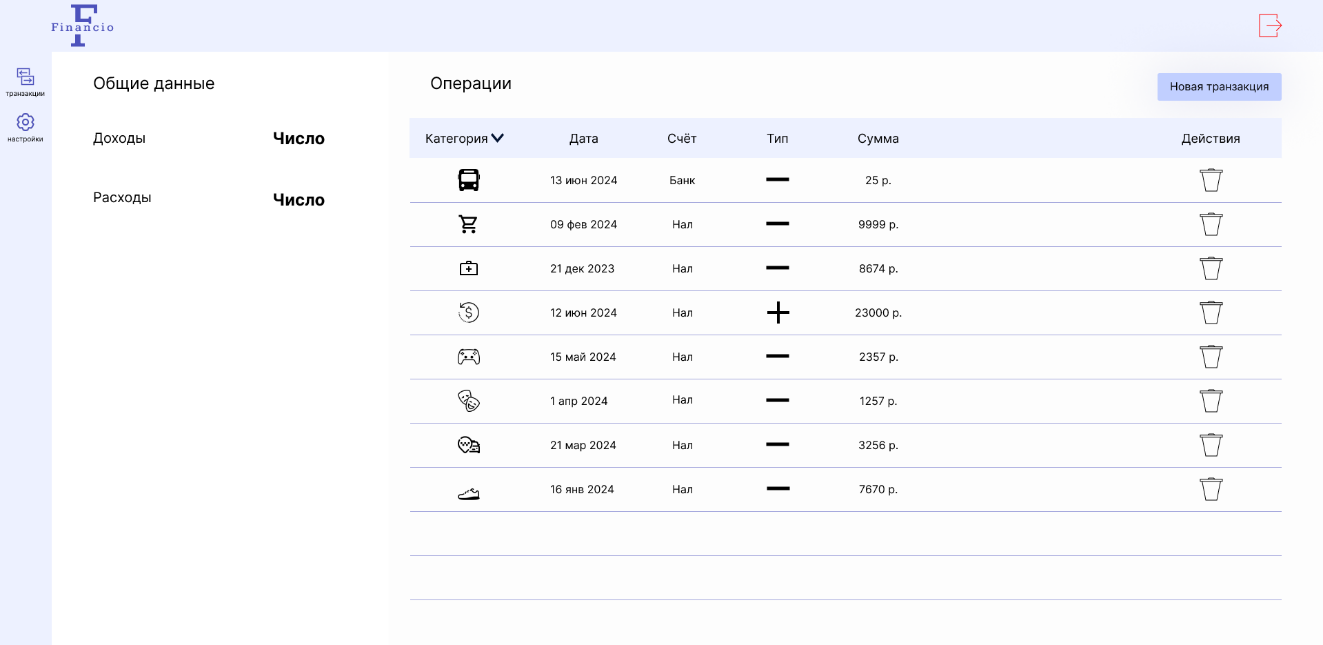


Рисунок 26 – Дизайн страницы транзакций личного кабинета

Страница личного кабинета с редактированием личных данных содержит в себе компонент профиля, показанный ранее, и имеет шаблон страницы для личного кабинета. Компоненты остались неизменными.

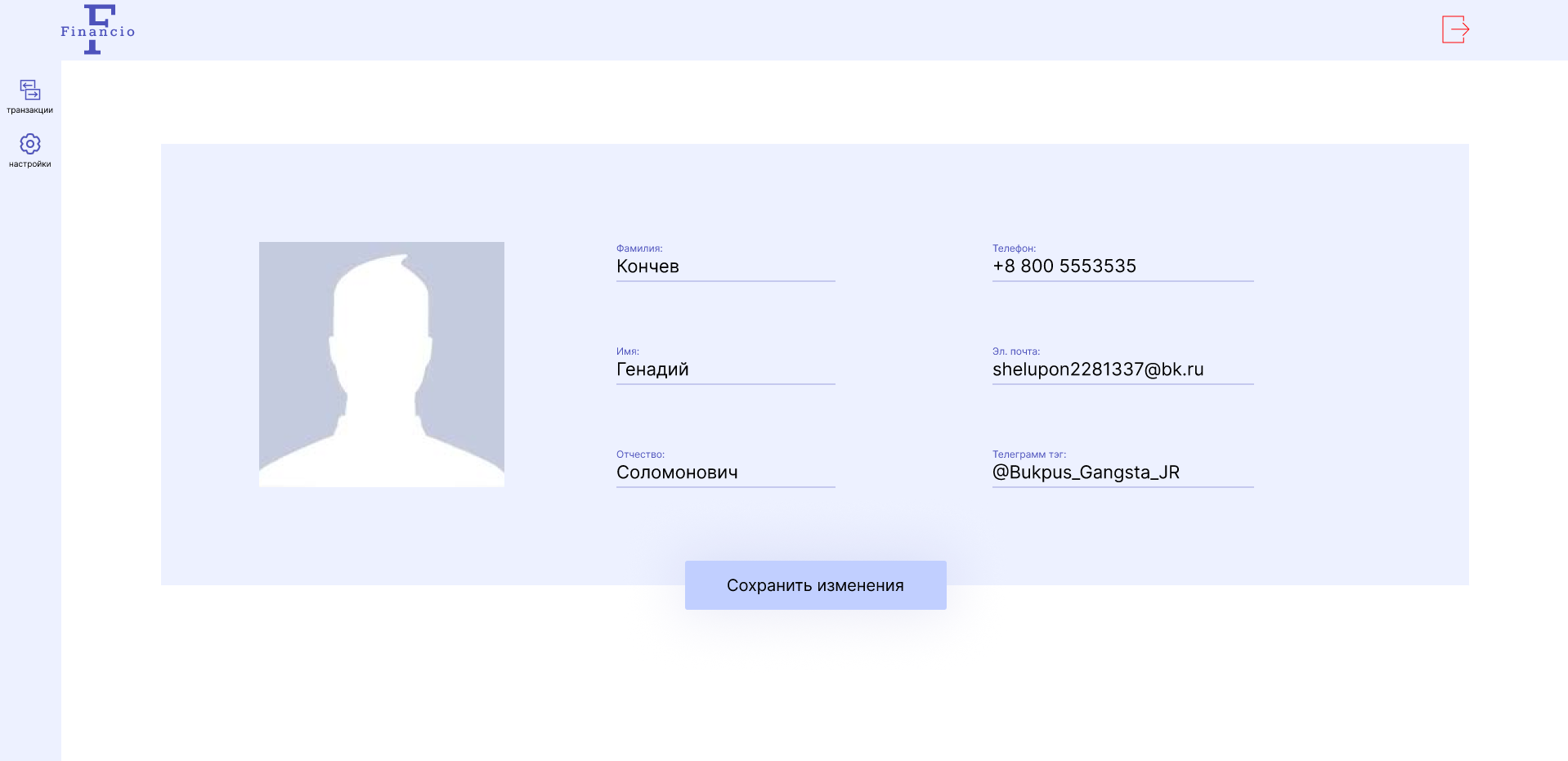


Рисунок 26 – Дизайн страницы редактирования профиля личного кабинета

# Разработка веб-приложения

## Разработка интерфейса веб-приложения

При разработке веб-приложения был использован язык программирования PHP совместно с Laravel - серверная платформа для работы с PHP, которая предоставляет готовые функции и методы для оптимизации разработки и организации шаблонной структуры веб-приложения.

Серверная часть приложения была необходима для работы с транзакциями и выведения всей информации о них из базы данных, реализации системы авторизации и хранения информации о транзакциях в профиле пользователя.

Для оформления Клиентской части веб-приложения, использован css-фреймворк Tailwind. Он хранит свои готовые стили на своём сервере, которые благодаря файлу tailwind.config.js подключаются к определённым php и html файлам. Использована библиотека Flowbite для добавления готовых компонентов на основе tailwind.

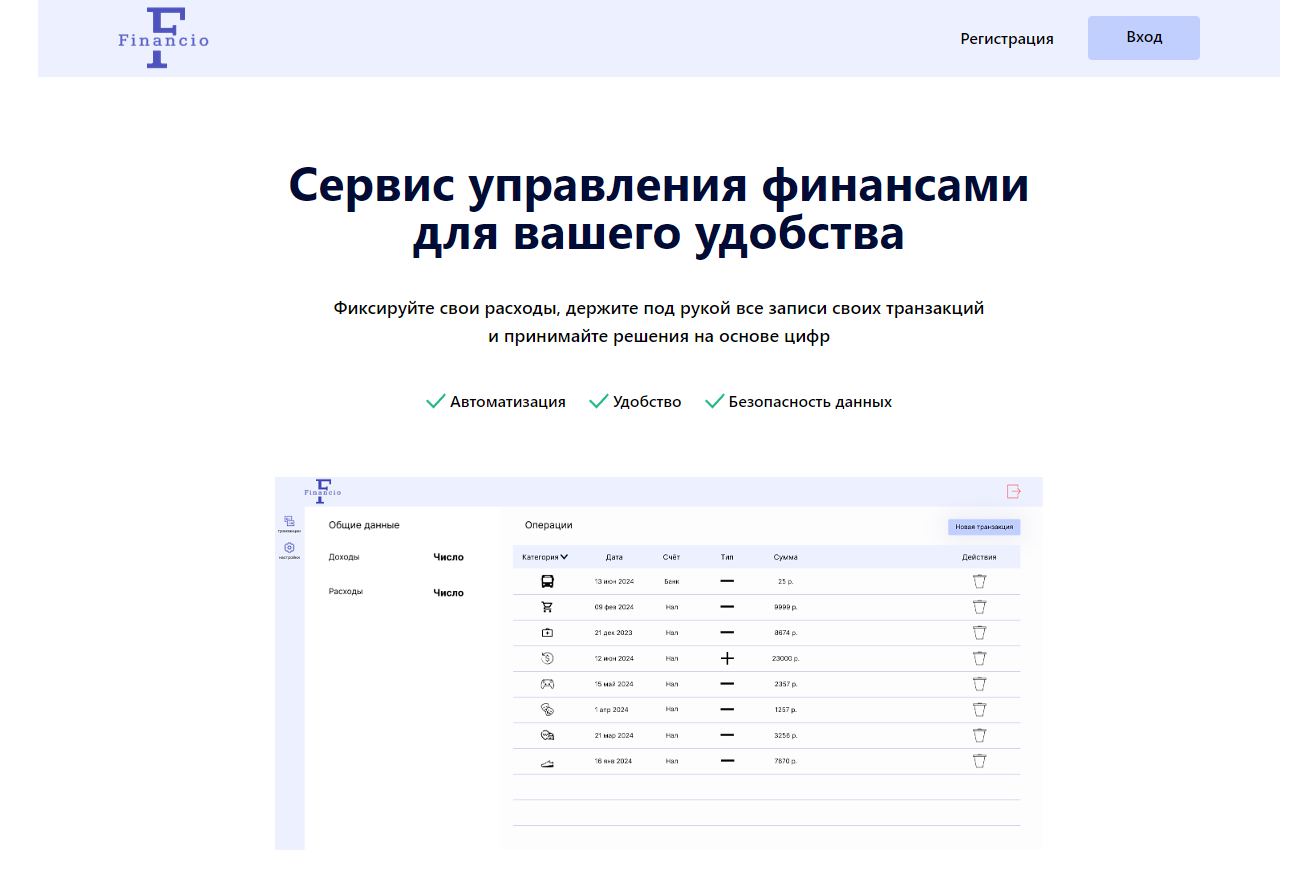


Рисунок 27 – часть главной страницы сайта

Уже ранее показанная главная страница имеет приведённый ниже код. Его структура вкратце описана с помощью комментариев на рисунке с кодом. С полным кодом всех компонентов можно ознакомиться в приложении Б



Рисунок 28 – код представления главной страницы

В шаблоне app подключаются компоненты шапки и подвала, а также в блоке с изменяемым содержимым <main> выводится код из index.blade.php.

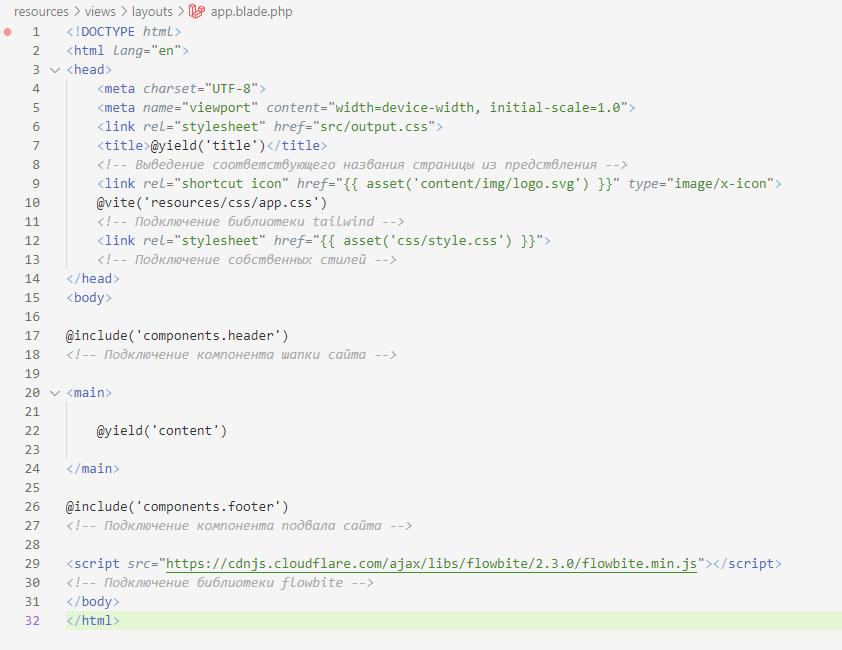


Рисунок 29 – Код шаблона для главной страницы

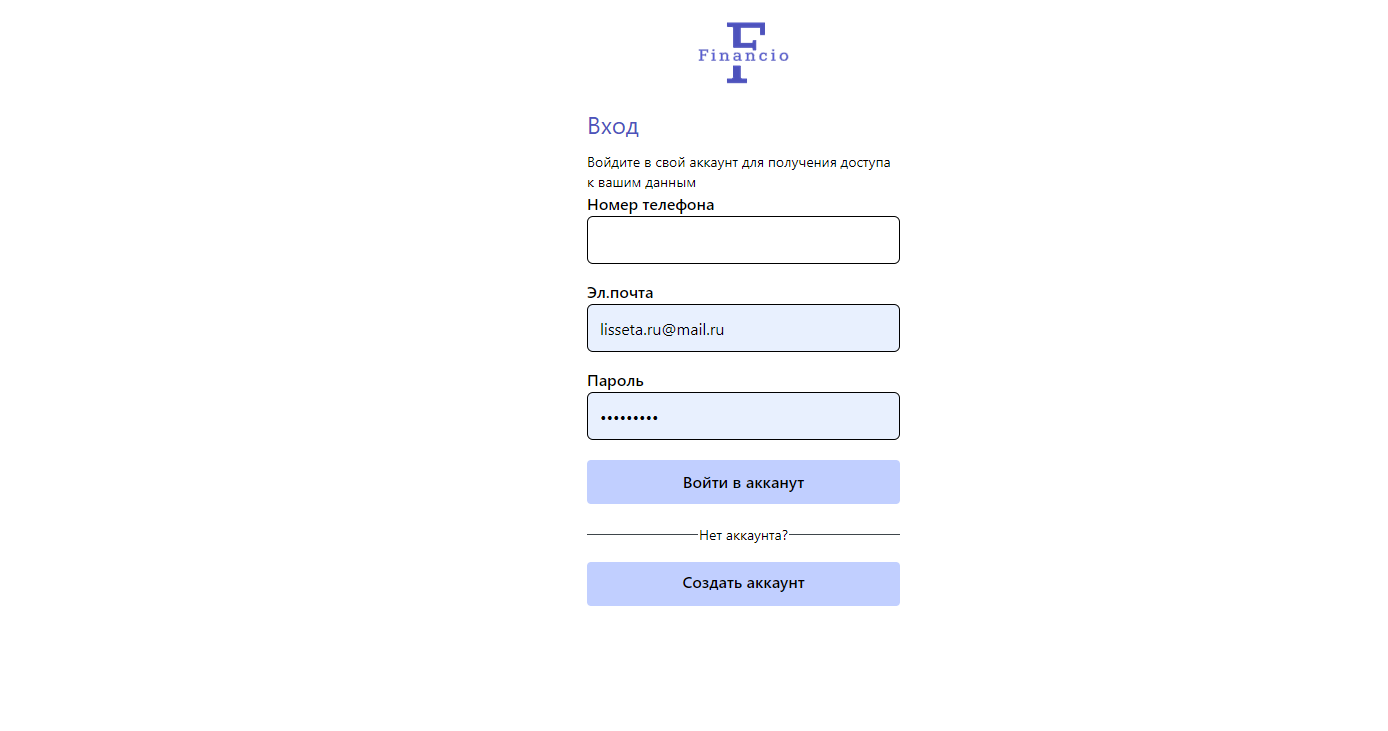


Рисунок 30 – страница входа

Страница входа имеет приведённый ниже код.

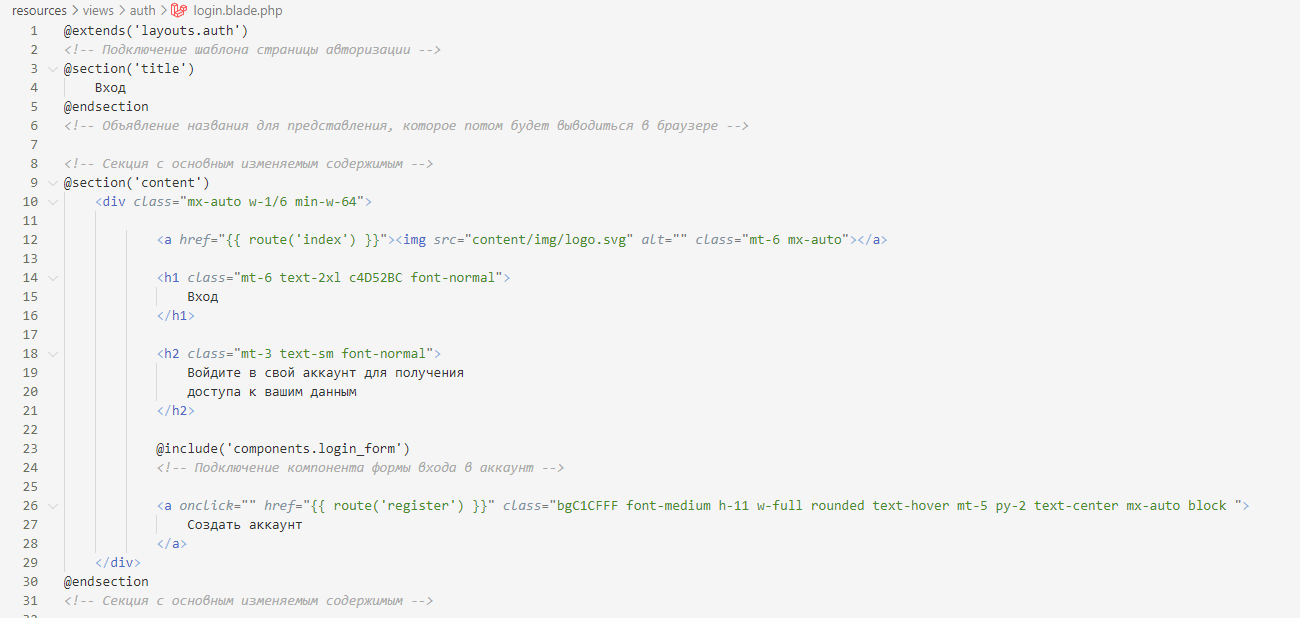


Рисунок 31 – код страницы входа в аккаунт

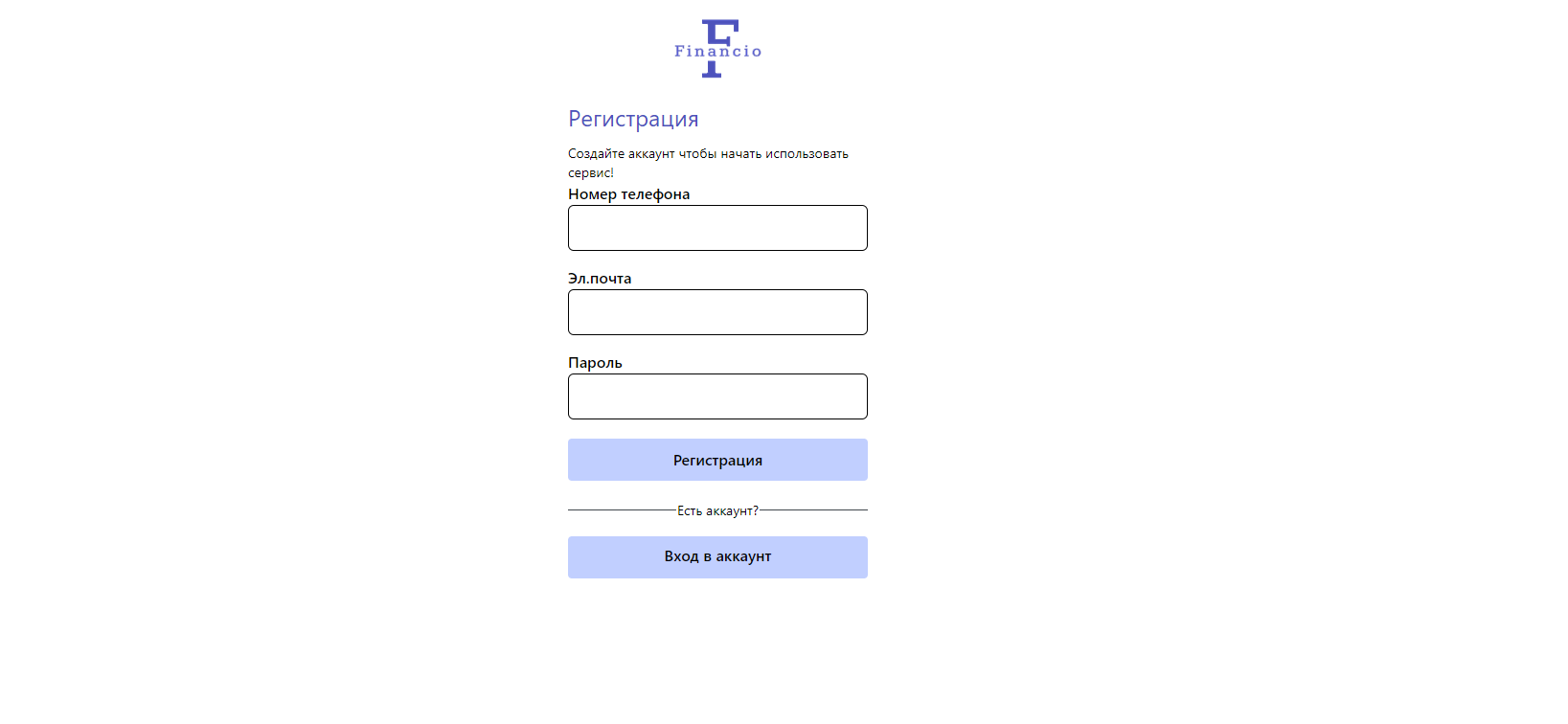


Рисунок 32 – страница регистрации

Страница регистрации имеет приведённый ниже код.

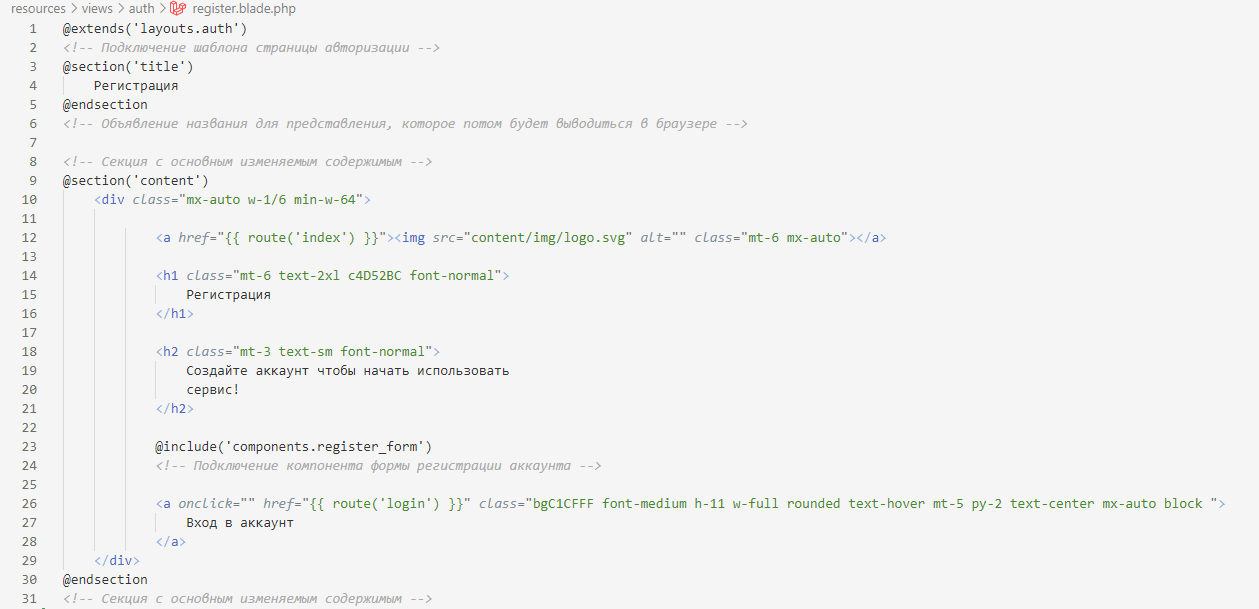


Рисунок 33 – код страницы регистрации

У страницы регистрации и страницы входа в аккаунт одинаковый шаблон auth.



Рисунок 34 – шаблон страницы регистрации и страницы входа

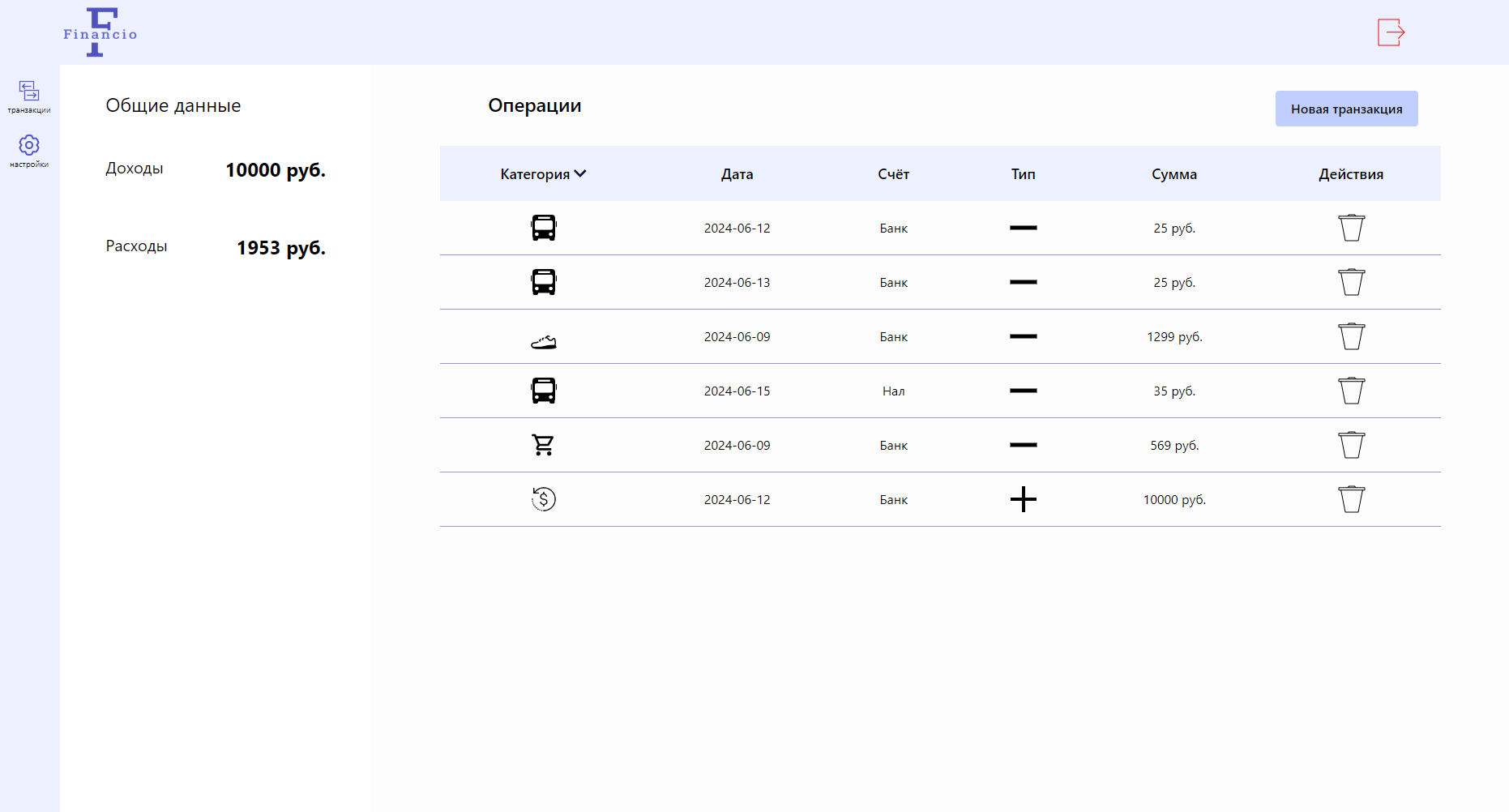


Рисунок 35 – страница транзакций личного кабинета

Код страницы личного кабинет с транзакциями приведён ниже.

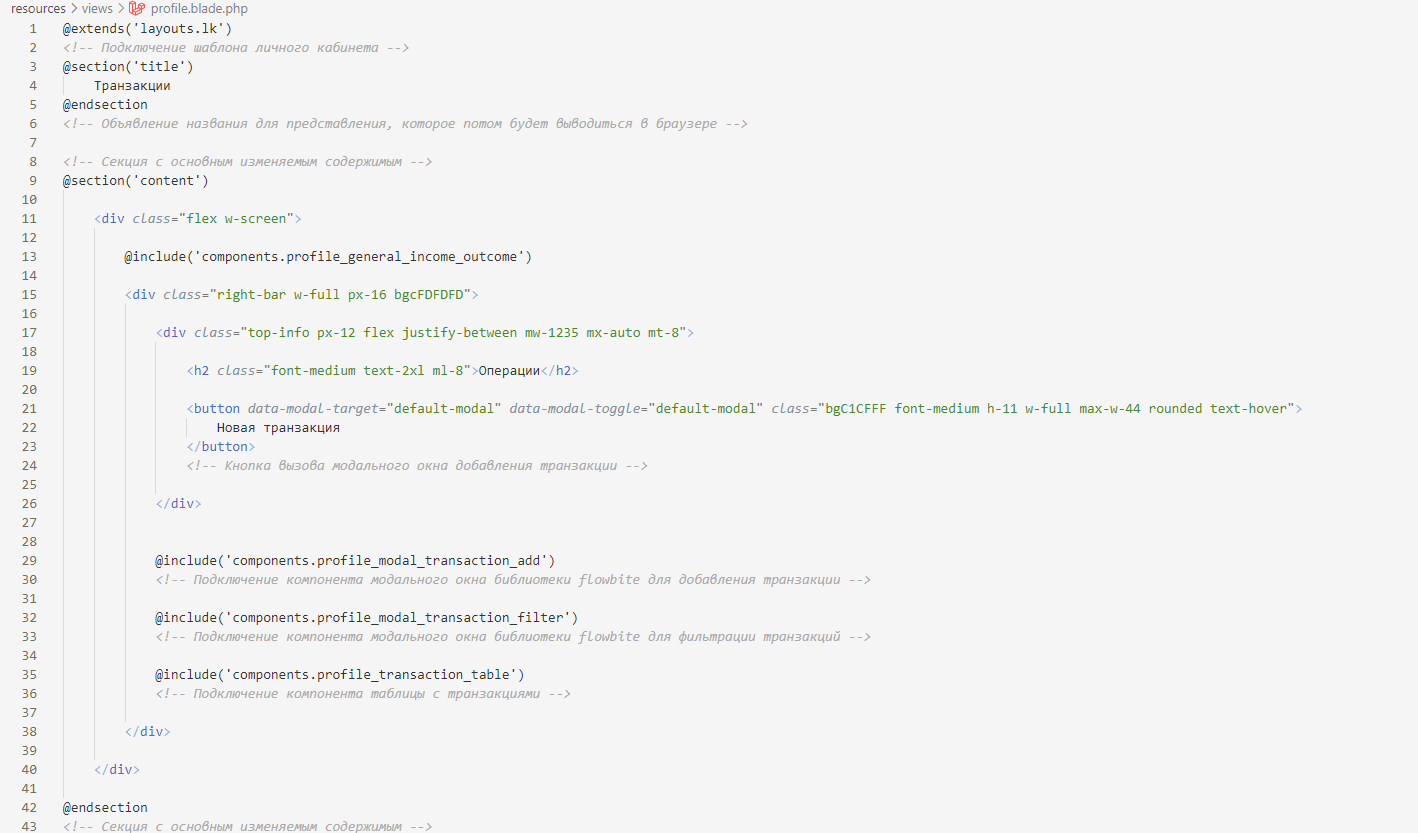


Рисунок 36 – код страницы транзакций личного кабинета

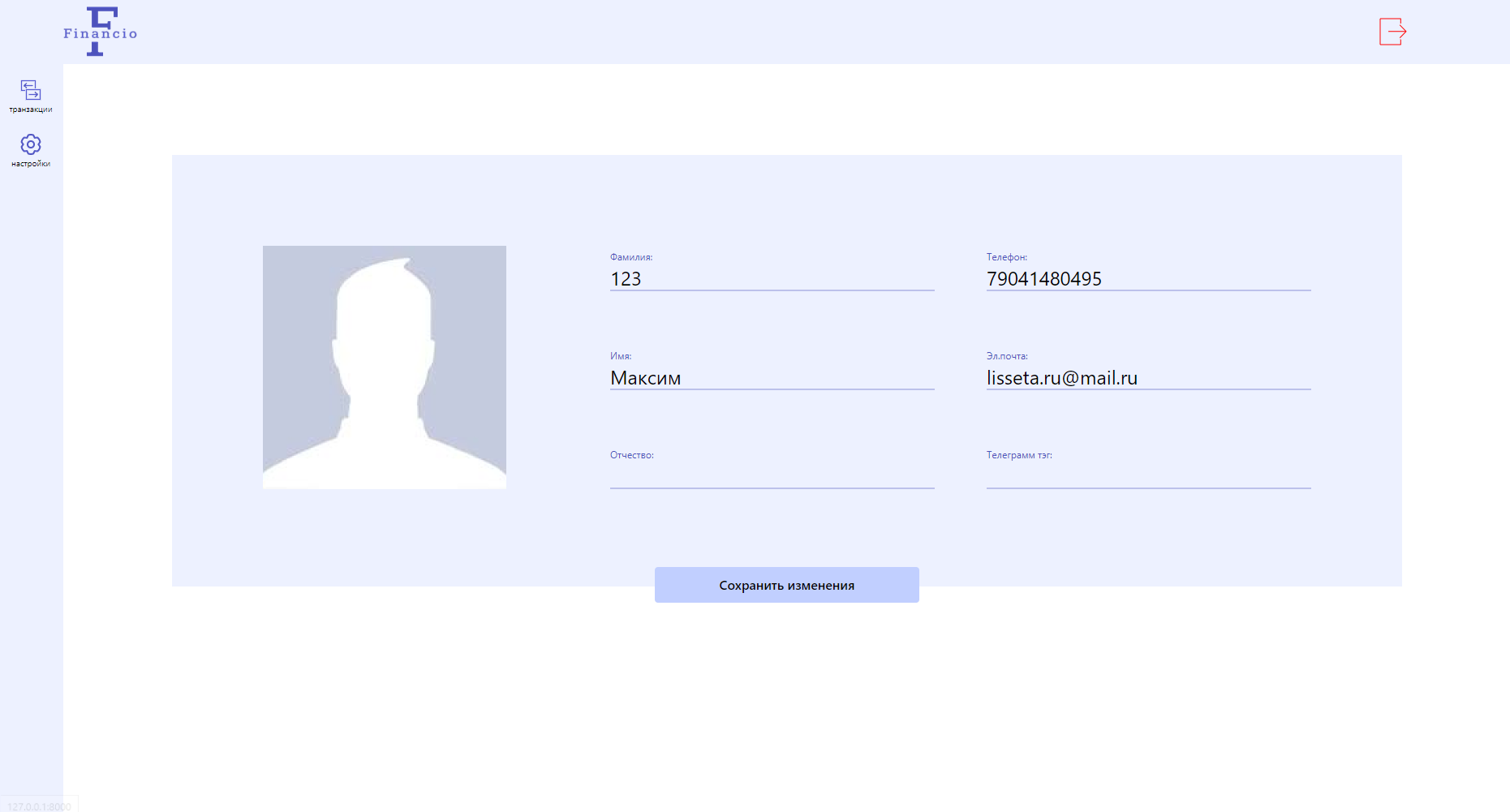


Рисунок 37 – страница редактирования профиля

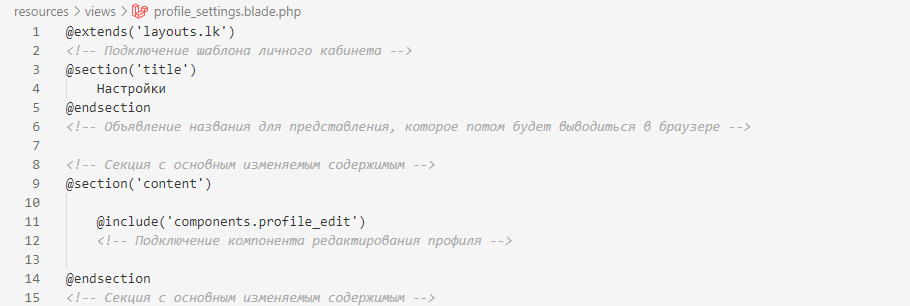
Код страницы редактирования профиля личного кабинета приведён ниже.

Рисунок 38 – код страницы редактирования профиля

У страниц личного кабинета пользователя одинаковый шаблон под названием lk. Его код приведён ниже.

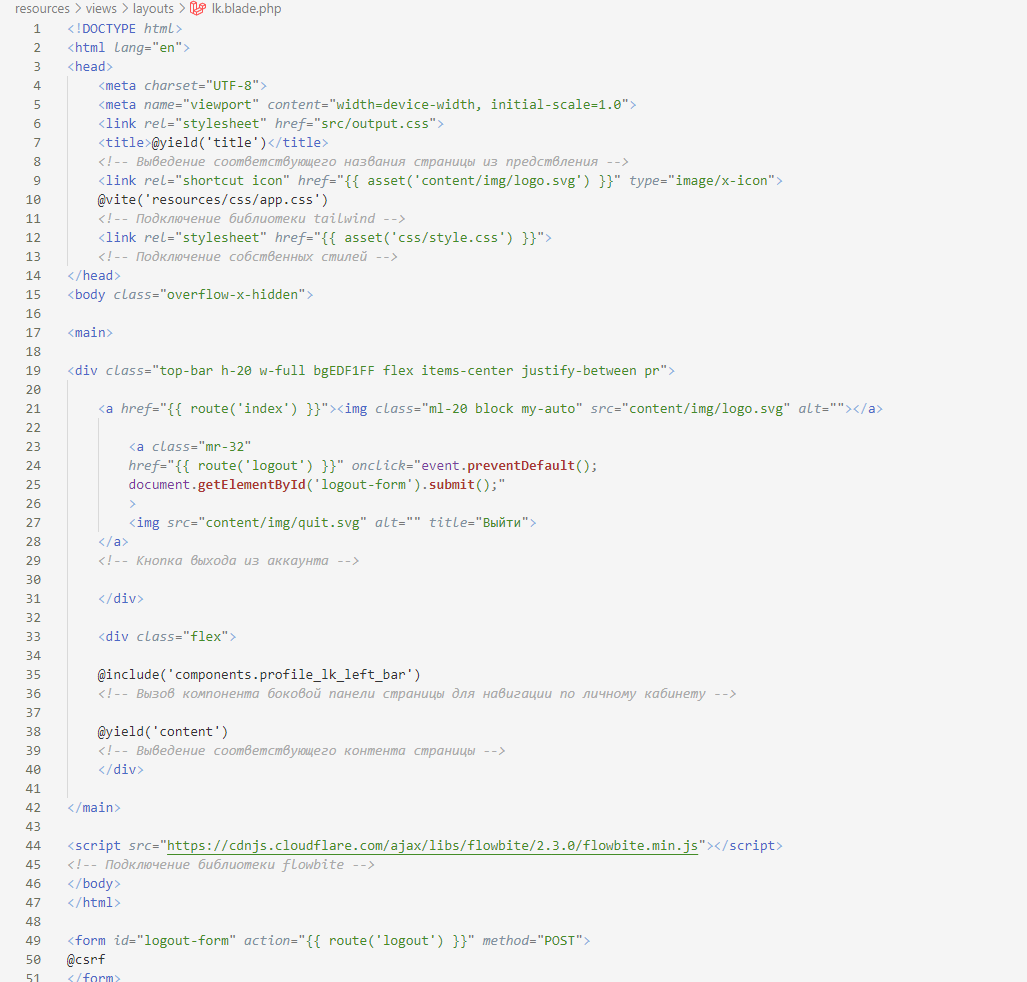


Рисунок 39 – код шаблона личного кабинета пользователя

## Разработка базы данных веб-приложения

Разработка базы данных реализовывалась с помощью миграций laravel. Миграции Laravel отправляют в пустую базу данных запросы на добавление таблиц, которые были прописаны в файле миграции. База данных состоит из 2 таблиц:

Пользователи и транзакции.

Таблицы базы данных представлены на рисунках.

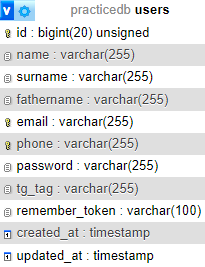


Рисунок 40 – таблица “пользователи”

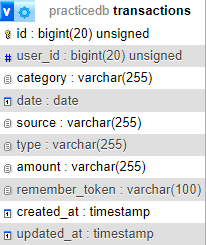


Рисунок 41 – таблица “транзакции”

## Разработка веб-приложения

Подключения веб-приложения к базе данных в laravel необходимо настроить конфигурационный файл .env, который расположен в корне проекта. В этом файле указываются данные для подключения к базе данных, такие как вид базы данных, порт, имя базы данных, логин, пароль и хост.

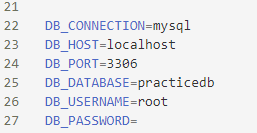


Рисунок 42 – подключение к базе данных

Ниже представлен код моделей базы данных.

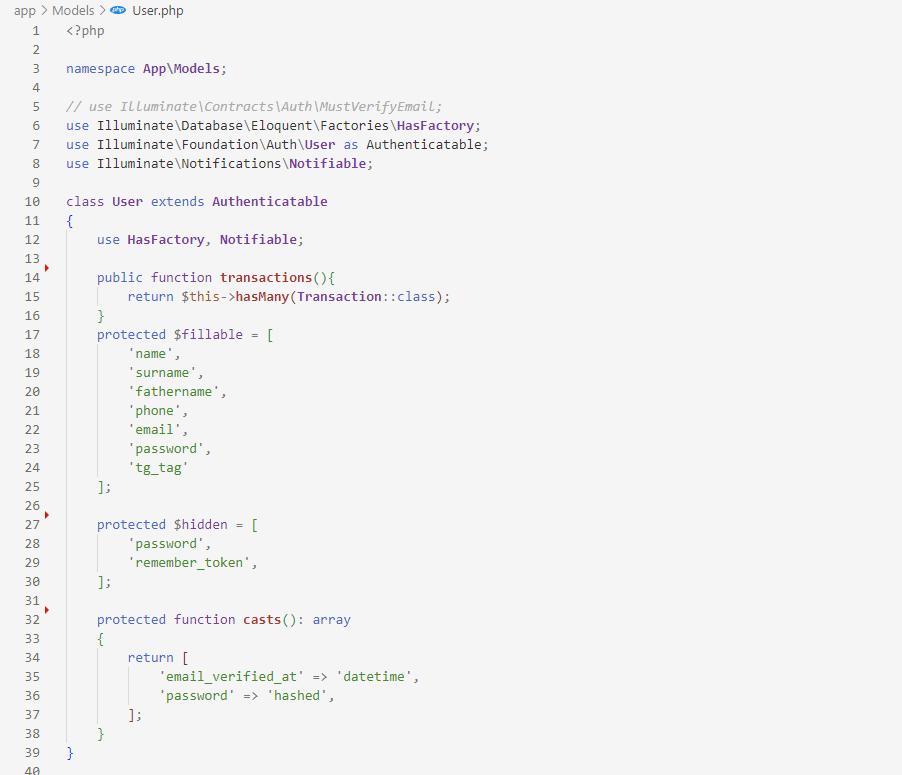


Рисунок 43 – модель пользователя в базе данных



Рисунок 44 – модель транзакций в базе данных

# Документирование программного продукта

## Руководство пользователя веб приложения

Для начала нужно скачать веб-приложение. Проще всего это будет делать через терминал windows или любой другой терминал. Желательно чтобы проект папка с проектом скачивалась на рабочий стол, но можно и куда угодно ещё. Прописываем в терминал “cd desktop” для перехода на рабочий стол и прописываем “git clone <https://github.com/MaksRudkovskiy/lar_kurs.git>”.

Проект скачан, теперь в терминал прописываем “code .” и открывается vs code. В терминале vs code нажимаем сочетание клавиш ctrl+` и открывается терминал. Дальше желательно разделить терминал на две части.

Для правой части прописываем данные команды: “ composer i” для создания папки vendor. Когда она загрузится, прописываем “cp .env.example .env”. Дальше в скопированном файле .env надо прописать путь к базе данных и доступ к ней.

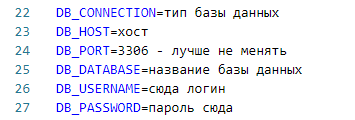


Рисунок 45 – пример подключения базы данных

После этого нужно запустить миграции с помощью команды «php artisan migrate». Теперь добавляем ключ нашего приложения командой «php artisan key:generate». Затем прописываем команду «npm install». Теперь последней командой в правом терминале будет команда «npm run dev».

В левый терминал прописываем команду «npm install flowbite», которая создаст возможность использовать библиотеку flowbite. Финальный шаг будет в прописании команды «php artisan serve», которая запустит сервер и выдаст примерно такую ссылку: <http://127.0.0.1:8001>. Перейдя по ней, попадём на главную страницу веб-приложения. В конце терминал должен выглядеть так:

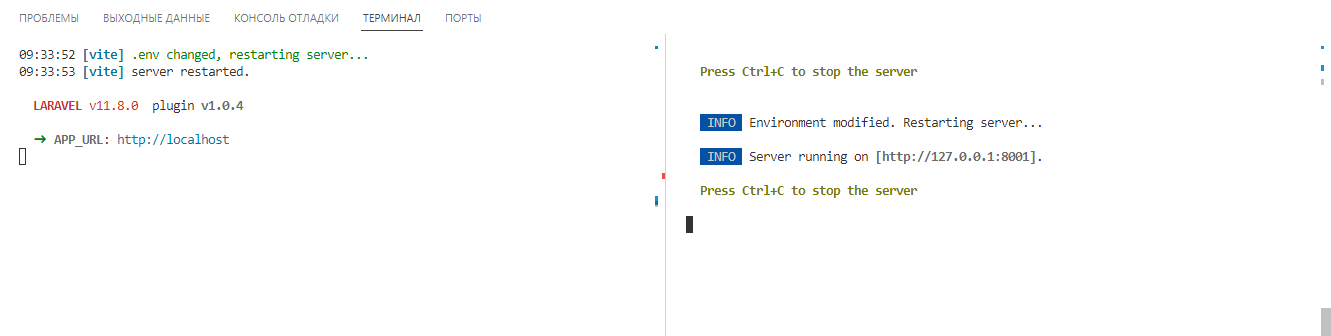


Рисунок 46 – пример терминала, если всё правильно прописано

Попав на главный экран, нужно зарегистрироваться, чтобы иметь полный доступ к веб-приложению. Нажимаем для этого на кнопку «регистрация» в шапке сайта.

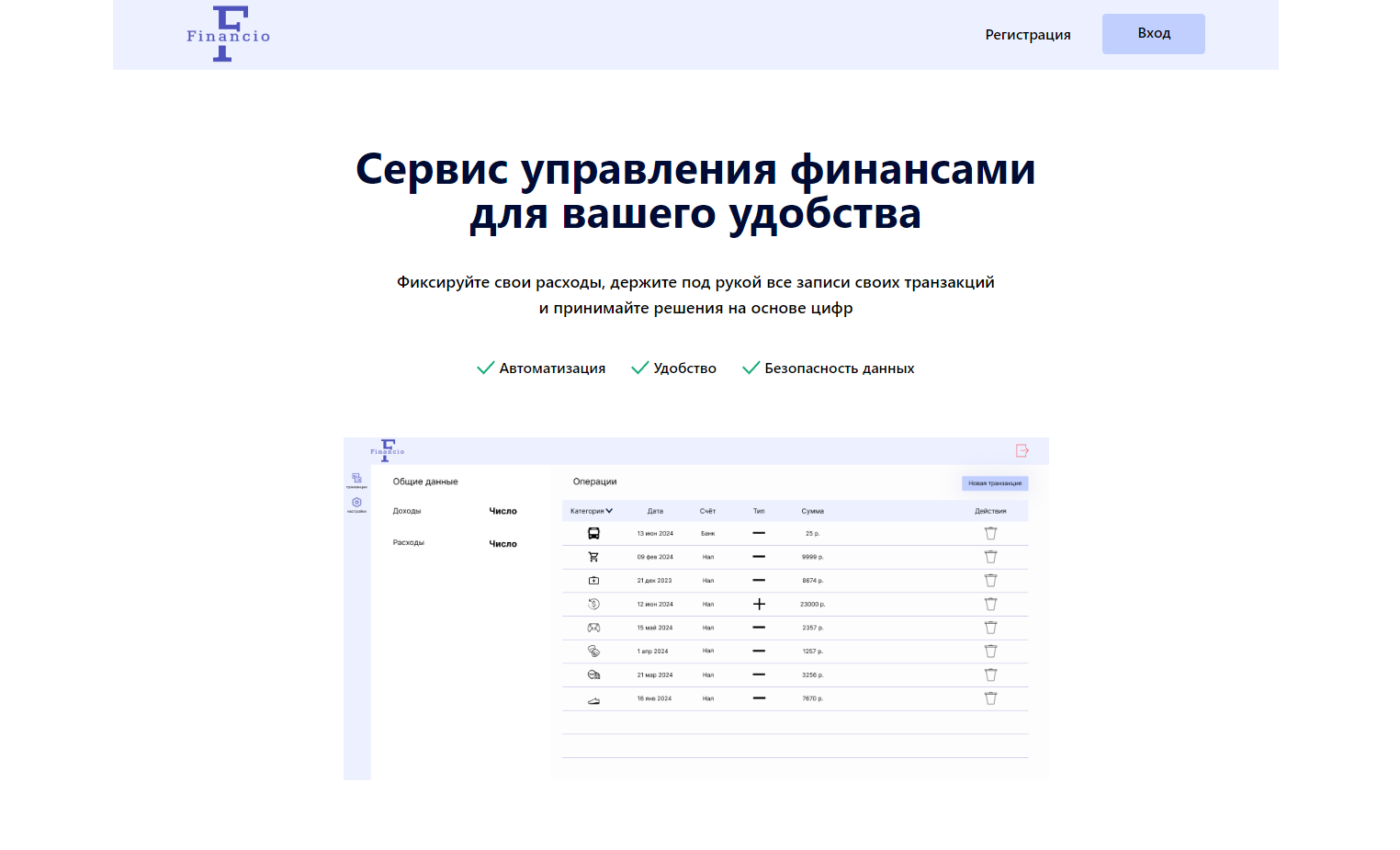


Рисунок 47 – Главный экран главной страницы

Регистрируемся…

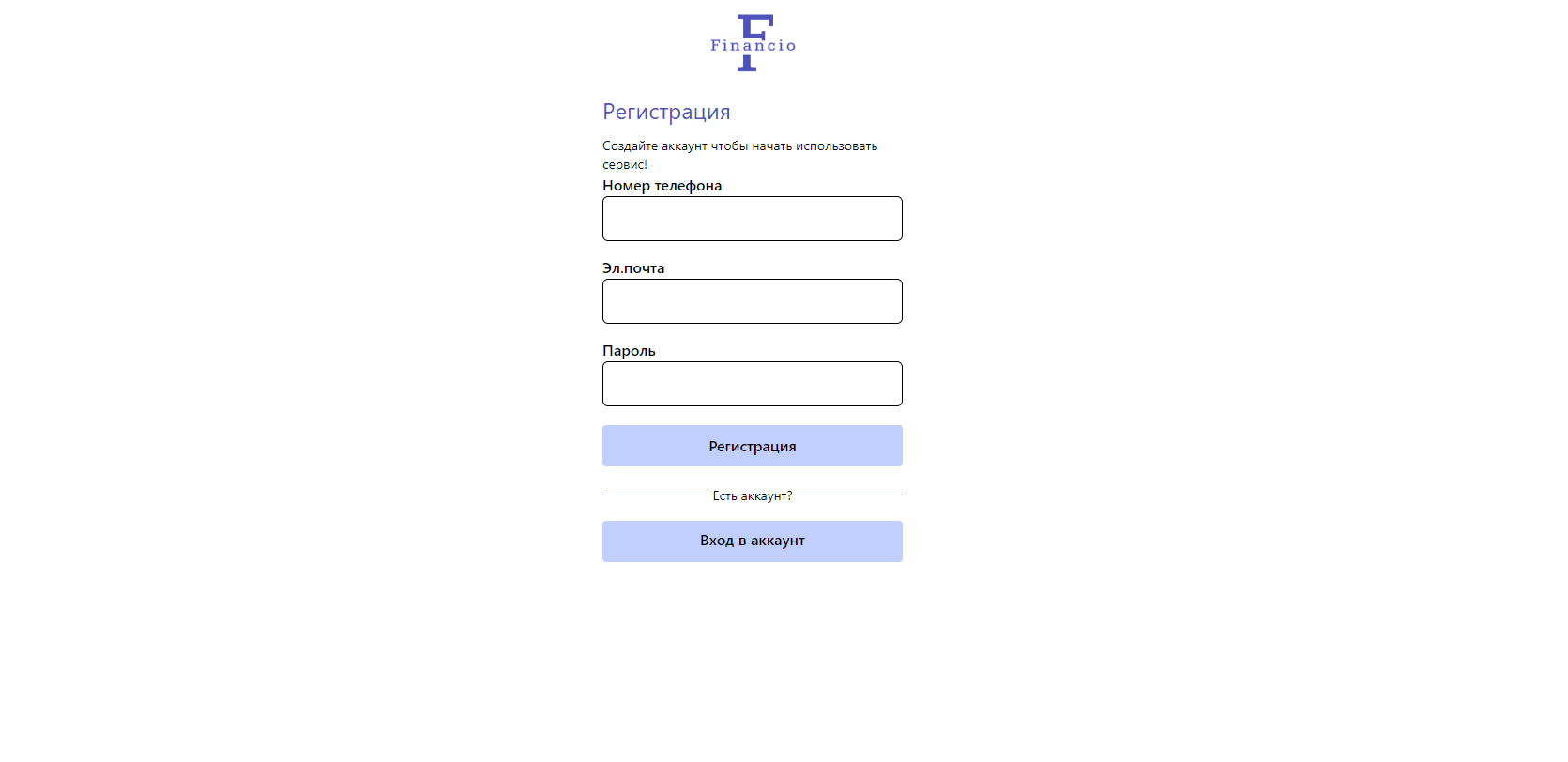
****

Рисунок 48 – страница с формой регистрации

После регистрации попадаем на страницу с транзакциями личного кабинета.



Рисунок 49 – страница транзакций личного кабинета без транзакций

Попав на страницу с транзакциями, она будет пуста. Давайте посмотрим, как работает добавление транзакции. Для этого нажимаем на кнопку «Новая транзакция

» и в модальном окне заполняем данные транзакции .

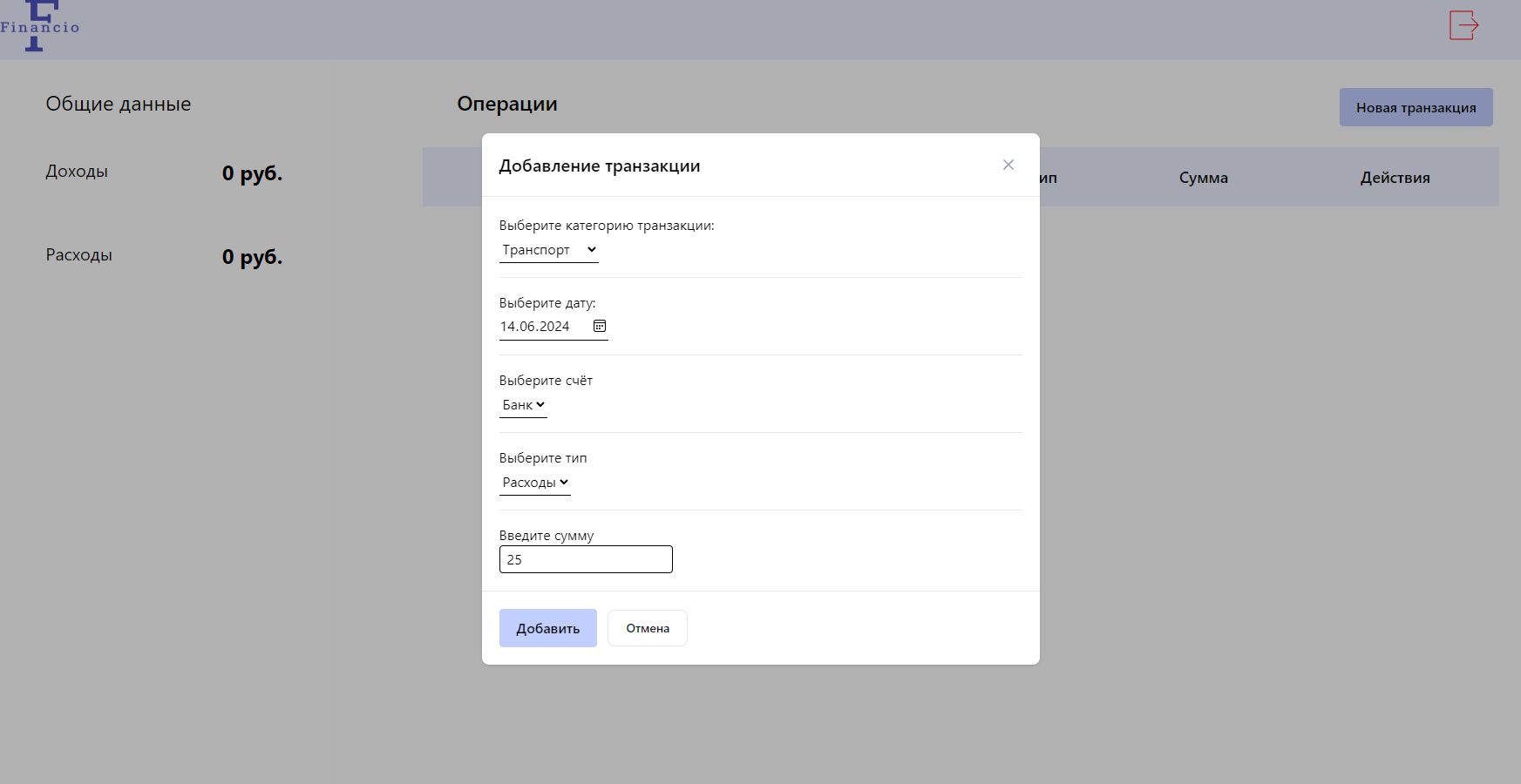


Рисунок 50 – модальное окно добавления транзакций

После этого наша транзакция будет добавлена в базу и будет высвечиваться для вас!

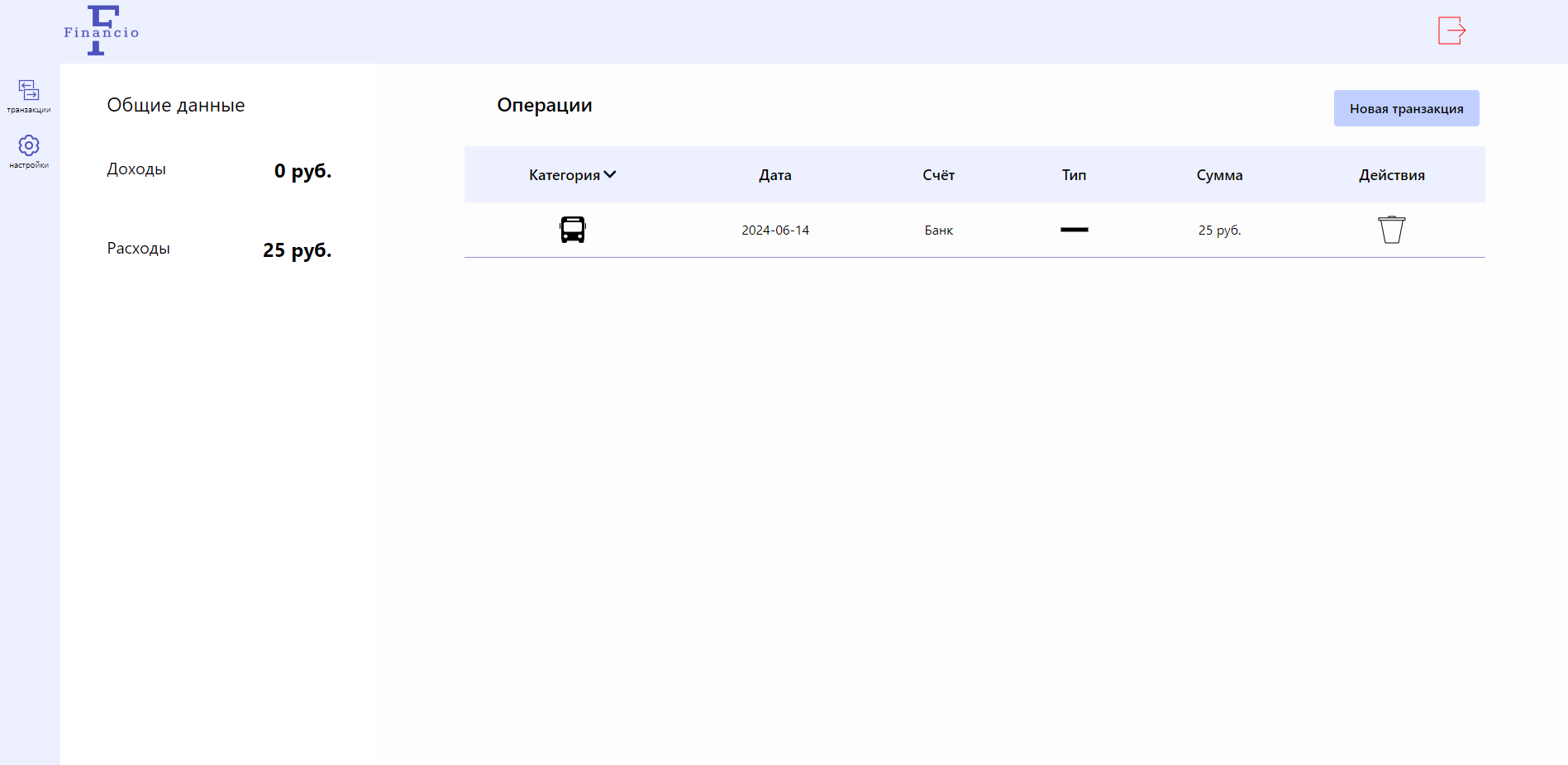


Рисунок 51 – страница транзакций личного кабинета с транзакцией

Давайте добавим ещё парочку транзакций в таблицу.

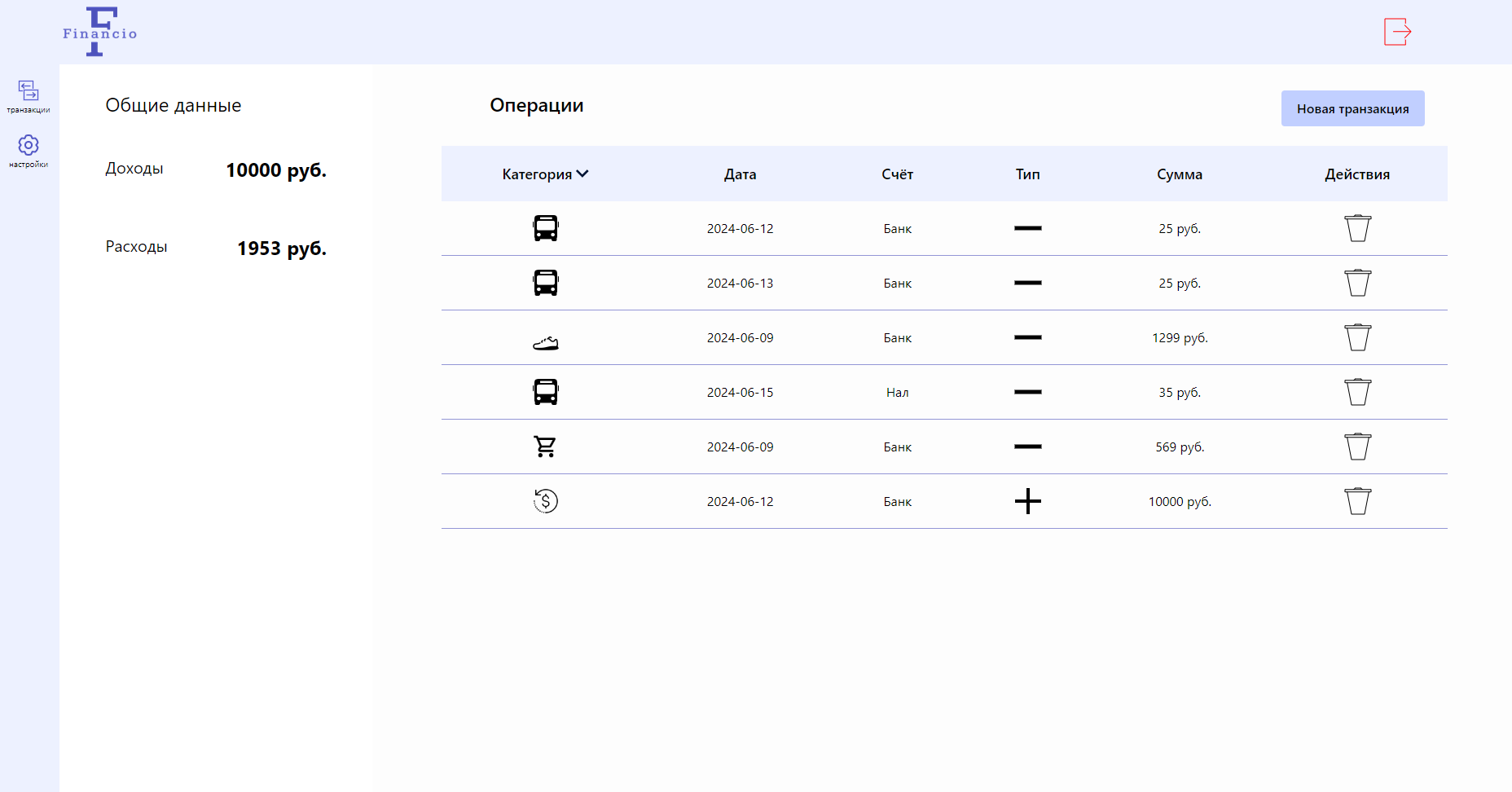


Рисунок 52 – страница транзакций личного кабинета с транзакциями

В какой-то момент у нас может быть слишком много транзакций и нам нужно найти транзакции определённой категории. Для того чтобы нам отфильтровать транзакции по категориям, нам нужно нажать на пункт «категории» в таблице, и тогда у нас вылезет модальное окно для фильтрации категорий.

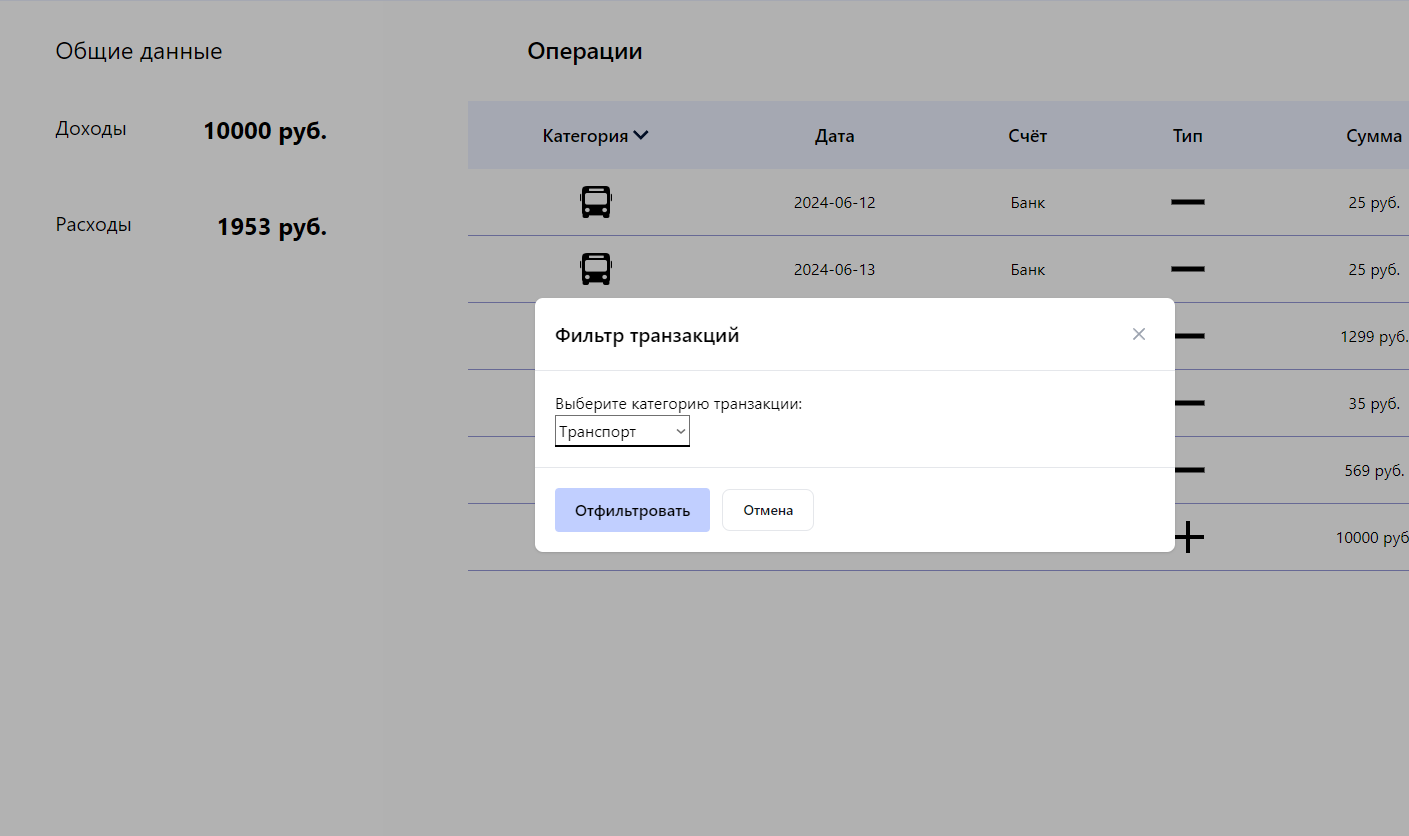


Рисунок 53 – модальное окно фильтрации транзакций

Отфильтруем таблицу по категории «транспорт».

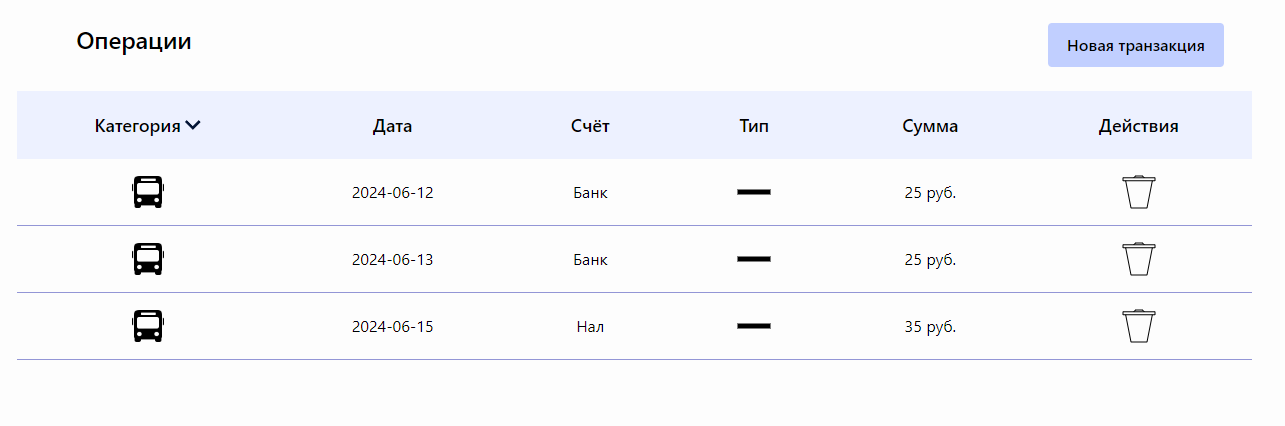


Рисунок 54 – отфильтрованная лента операций

Теперь наша таблица будет отображать только транзакции по категории «транспорт».

При добавлении ненужной транзакции или случайной ошибке, можно удалить транзакцию, нажав на иконку корзины напротив транзакции.

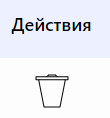
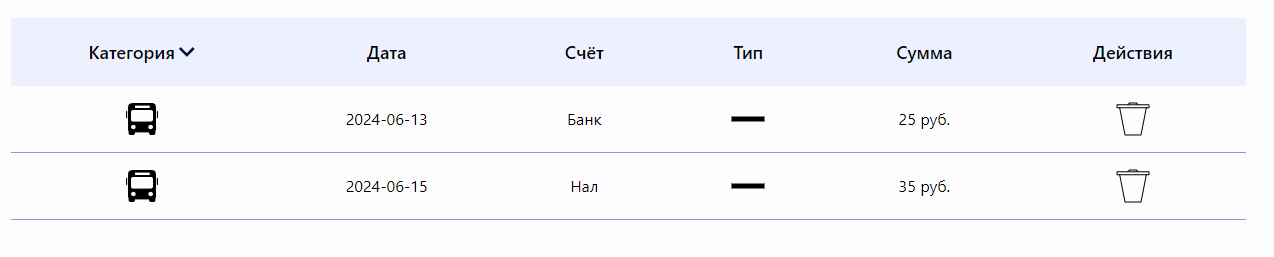


Рисунок 55 – кнопка удаления транзакции

  
Рисунок 56 – лента операций после удаления транзакции

Теперь у нас на одну транзакцию меньше.

По работе с лентой операций теперь всё, но веб-приложение также обладает и другими функциями. Например, мы можем перейти в настройки профиля и отредактировать свои данные.

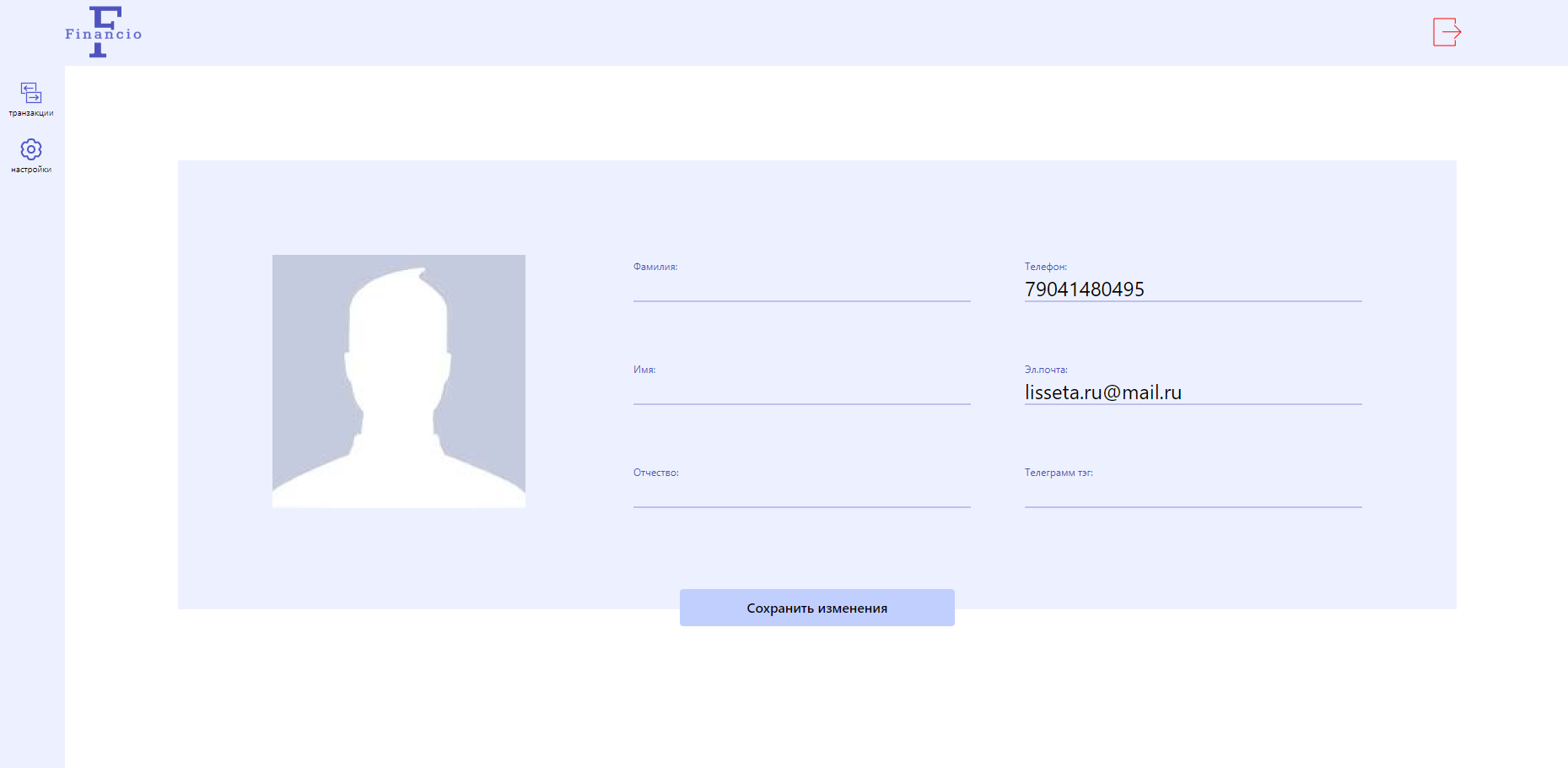


Рисунок 57 – страница настроек профиля без данных

По началу, у нас в профиле будут записаны только данные, указанные при регистрации. Пользователь может указать свои данные, которые будут отображаться в веб-приложении. В дальнейшем эти данные могут использоваться для рассылки новостей по почте и связи с пользователями. Давайте заполним все поля ввода.

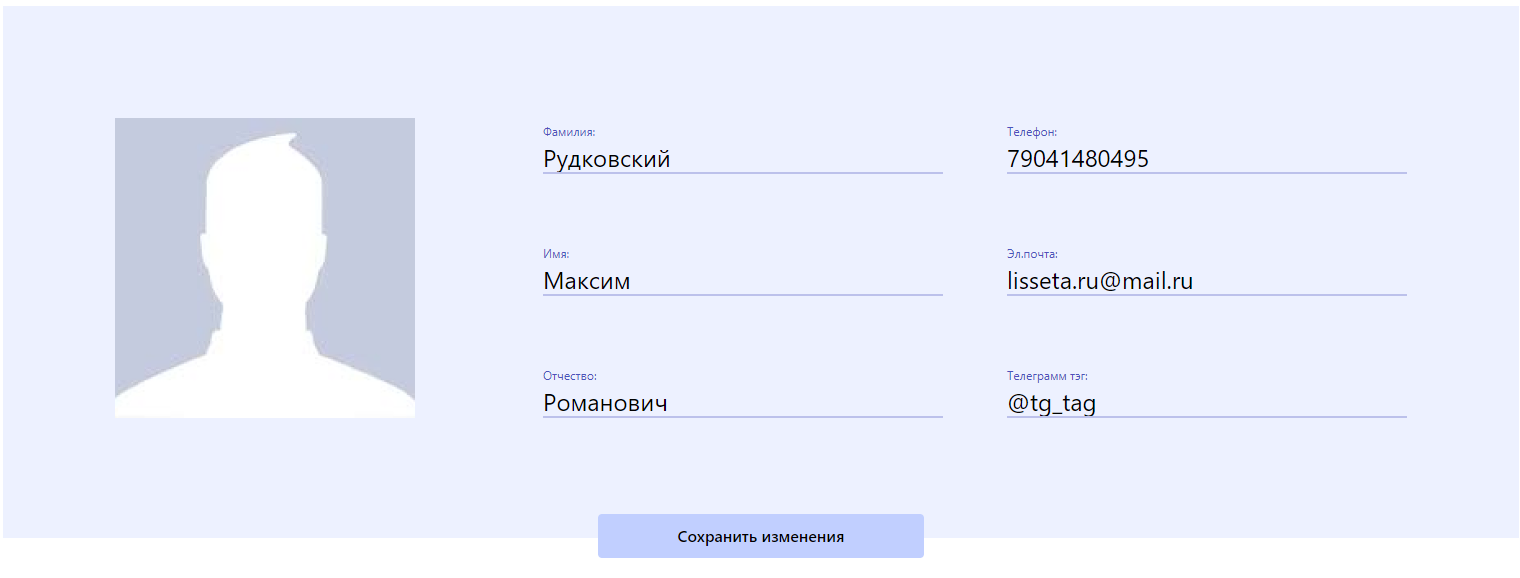


Рисунок 58 – страница настроек профиля с данными

Теперь, когда ваши данные заполнены, ваше имя будет отображаться, когда вы будете находиться на главной странице сайта. Если нажать на имя пользователя на главной странице, то оно перенесёт вас в ваш личный кабинет.

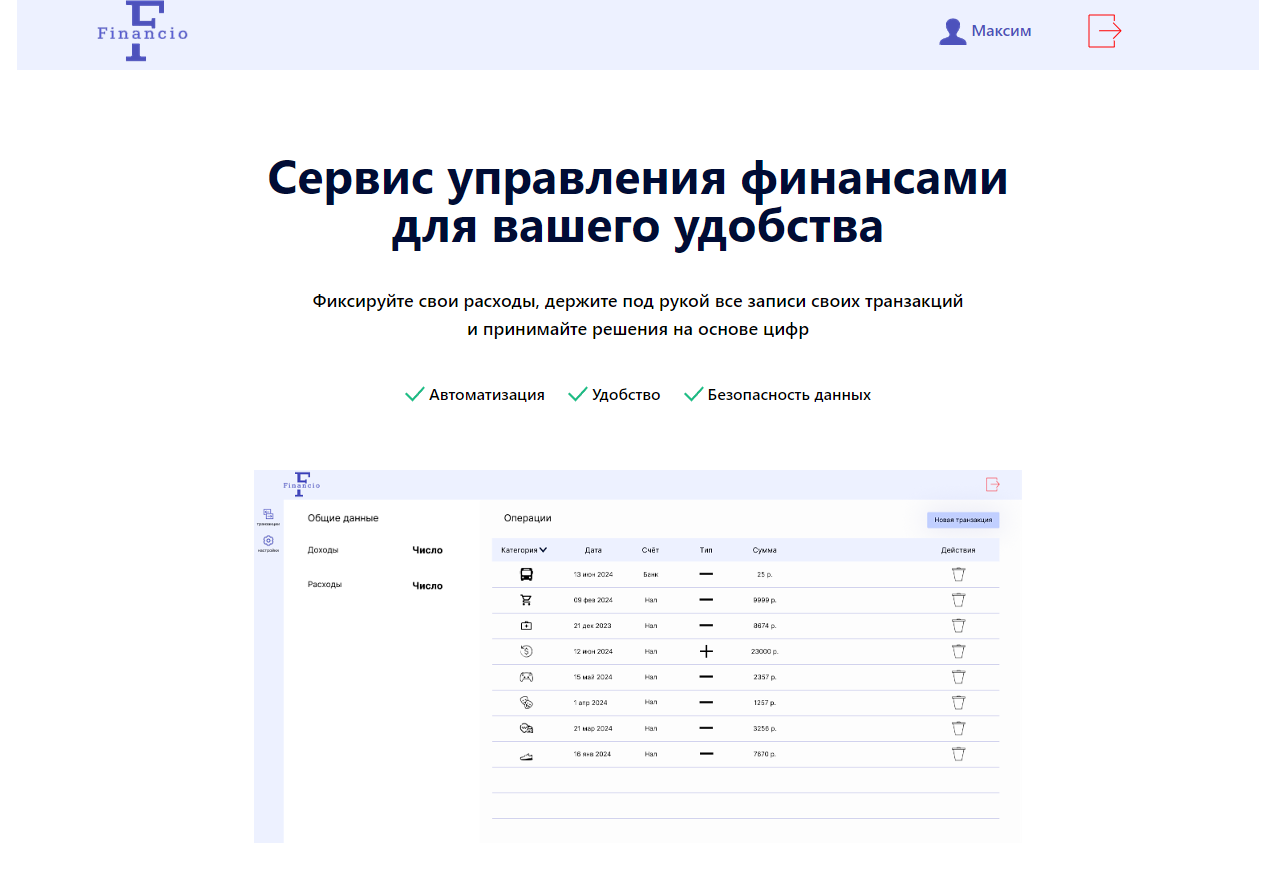


Рисунок 59 – имя пользователя на главной странице

# ЗАКЛЮЧЕНИЕ

В процессе курсового проекта была поставлена цель разработать веб-приложение управление личными финансами «FINANCIO». В результате работы был создан дизайн пользовательского интерфейса и back-end часть приложения, обеспечивающая возможность бесплатного управления личными финансами.

Из языков программирования основным для разработки был выбран PHP, с использованием фреймворков Laravel и Tailwind. Их применение в моем проекте было очень полезным и удобным. Laravel предоставляет множество полезных функций и инструментов, которые значительно упростили разработку моего веб-приложения. Этот фреймворк значительно упрощает разработку и тестирование веб-приложения. Общение с сервером также было сильно упрощено благодаря Laravel, для работы с данными из базы данных. Использование Tailwind значительно упростило написание стилей для html-кода веб-приложения, а библиотека flowbite помогла создать несколько функциональных элементов на страницах веб-приложения. Все необходимые диаграммы и модели помогли мне разработать нужный интерфейс. Для того, чтобы поставленная цель была решена, были выполнены следующие задачи:

* Предпроектное исследование.
* Написание технического задания.
* Проектирование веб-приложения.
* Разработка веб-приложения.
* Документирование веб-приложения.

## Список используемых источников

1. HTMLBOOK: Информация о HTML и CSS. – URL: https:// http://htmlbook.ru. (дата обращения: 25.10.2023). – Текст: электронный.
2. Руководство по PHP: Справочник языка PHP. – URL: https://www.php.net/manual/ru/index.php – (дата обращения: 01.06.2024). – Текст электронный.
3. «Modern PHP: New Features and Good Practices» – Текстовый файл – (11.04.2024) Текст: электронный.
4. Документация phpMyAdmin: Для веб-разработчиков. – URL: https://php-myadmin.ru/doc/– (дата обращения: 09.02.2024). – Текст: электронный.
5. Хабр: SQL запросы. URL: https://habr.com/ru/articles/480838/ – (дата обращения (23.03.2024). – Текст: электронный.
6. PHP Manual: Функции СУБД MySQL. URL: https://www.php.net/manual/ru/ref.mysql.php – (06.11.2023). – Текст: электронный.
7. PHP: PDO Manual. URL: https://www.php.net/manual/ru/book.pdo.php – (06.05.2024). – Текст: электронный.
8. Tailwind: Документация. URL: https://tailwindcss.com/docs/installation/ – (02.04.2024). – Текст: электронный.
9. PHP с нуля для начинающих: Как начать программировать на PHP. URL: https://skillbox.ru/media/code/kak\_nachat\_programmirovat\_na\_php/ – (10.11.2023). – Текст: электронный.
10. Справочник Javascript. URL: https://javascript.ru/manual – (20.12.2023). – Текст: электронный.

**Приложение А Техническое задание**

**Министерство образования Иркутской области**

Государственное бюджетное профессиональное образовательное учреждение

Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

|  |  |
| --- | --- |
|  |  |

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

ВЕБ-ПРИЛОЖЕНИЕ УПРАВЛЕНИЕ ЛИЧНЫМИ ФИНАНСАМИ «FINANCIO»

|  |  |  |
| --- | --- | --- |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Н.Р. Карпова) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (М.Р. Рудковский) |

Иркутск 2024

**1. Общие сведения**

Наименование работы: веб-приложение управление личными финансами «FINANCIO».

Исполнитель: студент Иркутского авиационного техникума, группы ВЕБ-21-2, Рудковский М.Р.

Разработка веб-приложения проходит в рамках курсового проекта по МДК.09.01 «Проектирование и разработка веб-приложений», на основании приказа №284-у от 29 января 2024 года.

Сроки разработки веб-приложения с 29.01.2024 по 13.06.2024 года.

**2. Назначение и цели создания системы**

Система создаётся с целью оптимизации учёта расходов и доходов пользователей. Удобный просмотр и отслеживание всех транзакций.

**3. Требования к системе в целом**

**3.1 Требования к структуре и функционированию системы**

Система должна обеспечивать возможность добавления транзакций в ленту транзакций в профиле пользователя. Система должна обеспечивать удобное управление транзакциями по типу редактирования, удаления, добавления. Система должна предоставлять возможность делать отчёты, как те что можно увидеть на сайте, так и виде текстового файла.

**3.2 Требования к надежности**

Система должна быть стабильной и надежной, обеспечивать сохранность данных пользователей и конфиденциальность транзакций пользователей.

**3.3 Требования к безопасности**

Система должна иметь защиту от несанкционированного доступа, защиту данных пользователей.

**3.4 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы**

Минимальные системные требования для рабочей станции:

* + - 1. Процессор: Intel Pentium 4 2.0ghz/AMD XP 2200+;
      2. Оперативная память: 512 Мб;
      3. Жёсткий диск: 150 Мб;
      4. Браузер Google Chrome, Microsoft Edge;
      5. Операционная система: Windows, Linux, MacOS.

Минимальные системные требования для сервера:

Процессор: 4 ядра, тактовая частота 2.90 ГГц и выше;

Платформа: 32-х или 64-х разрядная;

Оперативная память: 4 ГБ и выше;

Жёсткий диск: 300 Мб свободного объема и выше.

**4. Требования к документации**

Основным документам, регламентирующими разработку интернет-магазина является техническое задание.

**5. Состав и содержание работ по созданию системы**

В таблице 1 представлены плановые сроки начала и окончания работы по созданию веб-приложения

Таблица 4 – Плановые сроки по созданию веб-приложения

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Содержание этапа или стадии  выполнения КП | Сроки выполнения | |
| Начало | Окончание |
| 1 | Предпроектное исследование предметной области (выбор темы, постановка цели, задач, описание области применения, исследование предметной области) | 30.01.24 | 09.02.24 |
| 2 | Разработка технического задания (выбор архитектуры программного обеспечения, выбор типа пользовательского интерфейса, выбор языка и среды программирования) | 09.02.24 | 19.02.24 |
| 3 | Проектирование программного обеспечения.  (разработка структурной и функциональной схемы ПО, проектирование базы данных) | 19.02.24 | 04.03.24 |
| 4 | Оформление пунктов пояснительной записки (введение, предпроектное исследование, техническое задание и проектирование веб-приложения) | 04.03.24 | 11.03.24 |
| 5 | Разработка (программирование) и отладка программного продукта | 11.03.24 | 29.04.24 |
| 6 | Составление программной документации (оформление ПЗ, руководство пользователя и презентации) | 29.04.24 | 13.05.24 |

# ПРИЛОЖЕНИЕ Б листинг кода контроллеров и маршрутизаторов

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\{User, Transaction};

use Auth;

use Illuminate\Support\Facades\Hash;

class HomeController extends Controller

{

    public function index()

    {

        $transactions = Transaction::where('user\_id', Auth::user()->id)->get();

        $totalIncome = $transactions->where('type', 'income')->sum('amount');

        $totalExpense = $transactions->where('type', 'outcome')->sum('amount');

        return view('profile', ['transactions' => $transactions, 'totalIncome' => $totalIncome, 'totalExpense' => $totalExpense]);

    }

    // Данная функция index в этом контроллере нужна для передачи класса транзакции в переменную, к которой потом можно будет обращаться в представлении profile и выводить определённые данные из таблицы

    // Переменные $totalIncome и $totalExpense нужны для подсчёта общих расходов и доходов определённого пользователя.

    public function index2()

    {

        $user = Auth::user();

        return view("profile\_settings", ['user' => $user]);

    }

    // Данная функция index2 нужна для передачи переменной $user, которая равна классу с данными пользователя. Нужно это для передачи в представление profile\_settings данных пользователя

    public function edit\_info(Request $request) {

        $user = Auth::user();

        $user\_info = User::where('id', $user->id)->update([

            'name' => $request->name,

            'surname' => $request->surname,

            'fathername' => $request->fathername,

            'phone' => $request->phone,

            'email' => $request->email,

            'tg\_tag' => $request->tg\_tag,

        ]);

        return redirect()->back();

    }

    // функция edit\_info нужна для редактирования профиля пользователя

    // она отправляет запрос на обновление данных в таблице с пользователем, под чьим id отправляется запрос, обновляет данные и возвращает обратно на ту же страницу

    public function delete\_transaction($id) {

        Transaction::where('id', $id)->delete();

        return redirect()->back();

    }

    // Данная функция delete\_transaction удаляет транзакцию с соответствующим id и возвращает пользователя обратно на страницу

    public function filter(Request $request)

    {

        if (($request->category) == 'all') {

            $transactions = Transaction::where('user\_id', Auth::user()->id)->get();

        } else {

            $transactions = Transaction::where('user\_id', Auth::user()->id)->where('category', $request->category)->get();

        }

        $totalIncome = $transactions->where('type', 'income')->sum('amount');

        $totalExpense = $transactions->where('type', 'outcome')->sum('amount');

        return view('profile', ['transactions' => $transactions, 'totalIncome' => $totalIncome, 'totalExpense' => $totalExpense]);

    }

    // Данная функция filter нужна для фильтрации транзакций по категориям

    // В данном случае условие проверяет, какие категории выбраны в форме фильтра модального окна

    // Если выбраны все категории, то она и будет выводить все категории

    // Иначе же будет выводиться какая-то определённая категория

    // Переменные $totalIncome и $totalExpense нужны для подсчёта общих расходов и доходов определённого пользователя по пути filter/, который копирует profile/, но необходим для филбтрации.

}

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class MainController extends Controller

{

    public function index()

    {

        return view("index");

    }

    // функция index() вызывает представление index, которое является главной страницой

}

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\{User, Transaction};

use Auth;

class TransactionController extends Controller

{

    public function transactions(Request $request) {

        Transaction::create([

            'category' => $request->category,

            'user\_id' => Auth::user()->id,

            'date' => $request->date,

            'source' => $request->source,

            'type' => $request->type,

            'amount' => $request->amount,

        ]);

        $transaction = Transaction::where('user\_id', Auth::user()->id)->get();

        return redirect()->back()->with('transactions', $transaction);

    }

    // Данная функция transactions нужна для создания транзакции и занесения её в базу данных, откуда она будет выводиться в представление

}

<?php

use Illuminate\Support\Facades\Route;

Route::get('/', [App\Http\Controllers\MainController::class, 'index'])->name('index');

Route::get('/profile', [App\Http\Controllers\HomeController::class, 'index'])->name('profile');

Route::get('/profile/delete\_transaction/{id}', [App\Http\Controllers\HomeController::class, 'delete\_transaction'])->name('DeleteTransaction');

Route::get('/profile/transactionSum', [App\Http\Controllers\TransactionController::class, 'transactionTotalAmount'])->name('amoutCount');

Route::get('/profile\_settings', [App\Http\Controllers\HomeController::class, 'index2'])->name('profile\_settings');

Route::post('/profile/new\_transactions', [App\Http\Controllers\TransactionController::class, 'transactions'])->name('new\_transaction');

Route::post('/save\_settings', [App\Http\Controllers\HomeController::class, 'edit\_info'])->name('edit\_info');

Route::get('/filter', [App\Http\Controllers\HomeController::class, 'filter'])->name('filter');

Auth::routes();