

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ» (ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук  
Кафедра информационных систем и технологий

Веб приложение «Finance»

Курсовой проект

09.03.02 Информационные системы и технологии  
Программирования и информационных технологий

Зав. кафедрой \_\_\_\_\_ док. физ.-мат. наук С.Д. Махортов

Руководитель \_\_\_\_\_ В.С. Тарасов

Обучающиеся:

\_\_\_\_\_ Кошелев Н.М.  
\_\_\_\_\_ Тарлыков А.В.

Воронеж, 2022

## Содержание

Введение.....	4
1. Анализ предметной области.....	6
1.1. Глоссарий.....	6
1.2. Обзор предметной области.....	6
1.3. Обзор аналогов.....	8
1.3.1. CoinKeeper.....	8
1.3.2. Monefy.....	9
1.3.3. Money Lover.....	9
1.3.4. Money Flow.....	10
1.3.5. Buddy.....	10
2. Постановка задачи.....	11
2.1. Формулировка задачи.....	11
2.2. Этапы разработки.....	11
2.3. Функциональные требования.....	11
2.4. Средства реализации.....	12

2.4.1. СУБД.....	12
2.4.2. Сервер.....	12
2.4.3. Архитектура серверной части.....	13
2.4.4. Клиент.....	13
2.4.5. Архитектура клиентской части.....	14
3. Реализация.....	15
3.1. Диаграмма вариантов использования.....	15
3.2. Диаграмма классов.....	16
3.3. Диаграмма таблиц базы данных.....	16
3.4. Реализация интерфейса.....	18
4. Тестирование.....	31
Заключение.....	32

## **Введение**

Издревле люди использовали деньги как средство для ведения торговых отношений, как между друг другом, так и между городами, странами и так далее. Благодаря такому «изобретению», появились такие комплексные науки как экономика. По ходу истории, появились различного рода структуры, которые в той или иной степени работают с деньгами. Например банки, биржи, различные финансовые организации, фонды и так далее. С того времени как начали появляться первые государства, также появлялись различные валюты и их курсы соответственно.

В наше время деньги способны принимать самые разнообразные формы. Они могут быть как материальными, в виде бумажных или монетных денег, так и цифровыми, как например средства, которые записаны на наших счетах в банках, или в виде криптовалюты.

С появлением самых разнообразных форм денег и способов их хранения, также появилась потребность в том, чтобы правильно вести учет своих средств. В эти мероприятия входят учет средств в банках, наличных средств и так далее. Под учетом понимается контроль за тем, как деньги движутся по счетам, анализ этого движения и выявление стратегии и тактики по дальнейшему использованию.

До появления различных цифровых устройств, контроль за деньгами был достаточно нетривиальной задачей, так как любое движение средств приводило к тому, что требовалось вручную рассчитывать все изменения. Так как в таких ситуациях неизбежен человеческий фактор, то зачастую происходили ошибки, что имело негативное влияние.

С появлением цифровых устройств и с появлением программ, поддерживающих разнообразные математические операции, учет финансов стал более простой задачей, но большинство подобных программ не обладали удобным пользовательским интерфейсом, не были специализированы на работу с финансами и не имели никаких механизмов для защищенного

хранения данных. В следствии всего вышеперечисленного, можно сказать, что для большинства людей, учет финансов хоть и упрощался за счет автоматизации всех расчетов, но также и усложнялся тем, что пользователю приходилось детально разбираться в используемых программах и самостоятельно предусматривать хранение всех имеющихся данных. В сумме со сложным пользовательским интерфейсом, учет финансов хоть и был упрощен, но так и оставался сложной задачей, требующей навыков для работы с компьютером.

С развитием информационных технологий, стали появляться механизмы, позволяющие разработчикам создавать более пригодные для использования обычным людям программы и системы. Теперь пользователю не нужно заботиться о том, как и где хранить данные, как разобраться в сложном интерфейсе и как понять за что отвечает каждый пункт меню ведь теперь все интерфейсы стали понятными и простыми, не нагруженными огромным количеством разнообразных компонентов.

Соответственно целью данной работы является создание веб-приложения для учета финансов, удобного в использовании и поддерживающего все основные операции со счетами и хранящимися на них средствами.

## **1. Анализ предметной области**

### **1.1. Глоссарий**

Серверное программное обеспечение — в информационных технологиях — программный компонент вычислительной системы, выполняющий сервисные (обслуживающие) функции по запросу клиента, предоставляя ему доступ к определённым ресурсам или услугам.

Клиент — это аппаратный или программный компонент вычислительной системы, посылающий запросы серверу.

REST — архитектурный стиль взаимодействия компонентов распределенного приложения в сети. Другими словами, REST — это набор правил того, как программисту организовать написание кода серверного приложения.

База данных — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Система управления базами данных — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Транзакция — это логически завершённая банковская операция, в процессе осуществления которой происходит перевод определенной суммы денег с одного счёта на другой.

Элемент интерфейса или виджет — примитив графического интерфейса пользователя, имеющий стандартный внешний вид и выполняющий стандартные действия.

### **1.2. Обзор предметной области**

Есть несколько важных причин, зачем вести учет и планировать бюджет для семьи или для отдельного человека.

Это делается чтобы:

- Знать свои реальные доходы и расходы в течение определенного периода (месяца или года). Ведение бюджета поможет увидеть, сколько вы реально зарабатываете, и на что ежемесячно расходуются средства. В течение нескольких месяцев можно отслеживать структуру своих доходов, выявить преобладающие в ней поступления (зарботная плата, доход от предпринимательства, хобби). Вместе с этим, вы получите полную информацию о своих тратах. Проводимый анализ позволит найти пути их снижения.
- Находить «финансовые бреши», через которые утекают ваши деньги. Это те самые импульсивные и необдуманные покупки и вложения, которые не принесли никакой выгоды. Теперь вы сможете легко от них отказаться, так как наглядно увидите, сколько денег тратится впустую ежемесячно. Бюджет поможет более эффективно расходовать заработанное и сокращать траты.
- Начать планировать свой семейный (личный) бюджет. Это важный шаг на пути к управлению своими расходами. Планирование поможет использовать деньги осознанно, обходиться без дорогостоящих кредитов и обдумывать траты. Если покупка запланирована, то до ее приобретения будет определенное время, в течение которого можно отложить необходимую сумму. Для тех, кто имеет постоянный доход, составление четкого плана расходов поможет открыть новые возможности для вложения свободных денежных средств, расплатиться с долгами и кредитами и научиться откладывать деньги на поставленные цели.
- Оптимизировать траты и найти источники дополнительной выгоды. Из плана доходов и расходов за прошедшие месяцы станет ясно, какие траты в семье являются основными, а какие второстепенными. Какие платежи нужно делать обязательно (например, за ЖКХ, за обучение и кружки детей, налоги), а от каких

расходов можно отказаться. Многие расходы могут частично окупиться, если пользоваться картами с кэшбэком, кэшбэк-сервисами, приложениями с кэшбэком за чеки, покупать товары по акциям и скидкам (используя для этого приложения для экономии), применять купоны с сайтов-купонаторов – и это еще не все возможности оптимизации расходов. Все эти возможности сейчас предоставляют не только крупные, но и небольшие торговые точки. Ими нужно обязательно пользоваться, чтобы сделать покупки выгоднее.

- Начать делать накопления. Многим кажется, что их доход небольшой, и все уходит на продукты, а откладывать ежемесячно вообще нечего. Начав планировать свой бюджет, вы поймете, что даже при небольшом доходе можно копить, пусть даже немного. Оптимизация расходов поможет отказаться от части ненужных трат, а высвобожденные деньги можно отложить.
- Создать себе финансовую подушку безопасности. Этот тот самый неприкосновенный резерв «на черный день». Никому не известно, что будет завтра, особенно в условиях кризиса и повсеместного сокращения в организациях. Инфляция, новые налоги, снижение уровня заработной платы могут значительно ухудшить финансовое состояние. Чтобы пережить этот период без потерь и поддерживать привычный уровень жизни, нужно иметь финансовый резерв.

### **1.3. Обзор аналогов**

Будем рассматривать популярные платформы, завоевавшие наибольшее признание у пользователей.

#### **1.3.1. CoinKeeper**

В приложении есть удобный виджет: вы перетаскиваете монетку с нужного кошелька в статью расходов и вносите сумму. После списания



можно увидеть, сколько денег осталось на счету. С помощью CoinKeeper вы можете устанавливать фиксированный месячный лимит на определенные категории и отслеживать, какую часть вы уже потратили. А также синхронизироваться с другими устройствами и вести совместный бюджет, например, с членами семьи.

Плюсы:

- Можно импортировать все данные сразу из приложения мобильного банка.
- Есть напоминания о регулярных платежах по категориям.
- Статистика, которую можно кастомизировать: вы сами распределяете доходы и расходы по категориям и устанавливаете план и лимиты.

### **1.3.2. Monefy**

У Monefy привлекательный и интуитивно понятный интерфейс. Все траты представлены на диаграмме на главном экране — можно отследить их за день, неделю или месяц по разным категориям. Приложение поддерживает несколько валют.

Плюсы:

- Можно выбрать отчетный период и составить наглядную статистику расходов с графиками.
- Поддерживает несколько счетов одновременно.
- Есть встроенный калькулятор.

### **1.3.3. Money Lover**

В приложении есть множество подкатегорий, также ими можно управлять — добавлять или удалять. Чтобы не забыть записать расходы в указанное время, пользователь может установить напоминание.

Плюсы:

- Возможность внесения еще неоплаченных счетов заранее.

- Уведомления о достижении установленного лимита и оповещения об обязательных платежах.

#### **1.3.4. Money Flow**

Классический планировщик для учета финансов с простым и лаконичным дизайном. Приложение синхронизируется с другими устройствами, а траты доступны в виде ленты операций или диаграммы. Есть функция повтора операций — например, можно привязать оплату кредита и не вводить данные каждый месяц вручную.

Плюсы:

- Можно прикреплять изображения к операциям.
- Учет переводов между своими счетами.
- Можно добавлять геометки и запоминать локации операций.

#### **1.3.5. Buddy**

У приложения милый дизайн в розовых оттенках. Оно предназначено для ведения совместного бюджета: вы платите за подписку один раз и подключаете других через приглашения на имейл. Пользователь может учитывать расходы по категориями и делить их с членами семьи или друзьями.

Плюсы:

- Есть возможность планировать и распределять расходы с друзьями или на конкретные события, например отпуск.

## **2. Постановка задачи**

### **2.1. Формулировка задачи**

В ходе данной курсовой работы должно быть разработаны серверное и клиентское приложения образующие информационную систему для учета финансов.

### **2.2. Этапы разработки**

Всю разработку можно поделить на четыре этапа:

Проектирование базы данных. Она является основой для разработки и дальнейшей работы сервера приложения. В ней будут храниться все данные пользователей (учетные записи, счета, доходы, расходы по категориям, переводы между счетами).

Разработка серверного приложения. Это одна из основных задач курсовой работы. Именно сервер обеспечивает бесперебойную работу требуемого функционала с базой данных, обеспечивает надежное изменение данных.

Описание API. Это требуется для того чтобы конечный пользователь сервера мог с легкостью понять как пользоваться его функциями и как это делать корректно чтобы получить ожидаемый результат.

Разработка клиентского приложения. Также основная задача курсовой работы и заключается она в том, чтобы создать приложение реализующее пользовательский интерфейс и взаимодействие с сервером согласно описанному API.

### **2.3. Функциональные требования**

Итоговое приложение должно включать в себя следующий функционал:

- Регистрация и аутентификация. Эта функция позволяет иметь доступ к одним и тем же данным пользователя с разных устройств. Это удобно т.к. в большинстве случаев требуется вести именно

семейный учет, следовательно каждый член семьи должен иметь возможность вносить изменения данных независимо.

- Создание и удаление счетов. Позволяет вносить основные данные по счетам: сумма на счете, наименование счета. Также имеется возможность удалить счет, с сохранением всех транзакций. Это требуется для корректного сбора статистики.
- Возможность добавления транзакций. По сути основная функциональность приложения без которой оно не имеет смысла. Можно вносить переводы между счетами, так и доходы и расходы по отдельным счетам.
- Вывод статистики по доходам и расходам. Также одна из важнейших функций приложения, так как помогает производить анализ финансов и планировать дальнейший бюджет. Для получения статистики требуется указать период времени по которому будут собраны данные.
- Добавление своих категорий доходов и расходов. Это требуется для получения более подробной статистики (при ее выводе, в процентном отношении указывается распределение доходов и расходов по категориям). При регистрации для каждого пользователя по умолчанию создается категория расходов и доходов с названием "другое".

## **2.4. Средства реализации**

### **2.4.1. СУБД**

В качестве СУБД было решено использовать PostgreSQL т. к. команда уже имела опыт работы с этим инструментом.

### **2.4.2. Сервер**

Для написания серверной части было решено использовать Java версии 17 так как это один из самых популярных языков программирования и очень часто используется для написания серверов.

Для упрощения разработки было решено использовать Spring Boot Framework так как у него уже есть готовый набор инструментов для написания веб и enterprise приложений (spring контейнер, spring security и так далее). Также с помощью него удобно разрабатывать приложения поддерживающие REST.

В качестве средства миграции базы данных было решено использовать LiquiBase так как он удобен в использовании и изначально разрабатывался для Java.

Для работы с данными было решено использовать MyBatis так как относительно своих аналогов он лучше подходит для написания сложных запросов к базе данных и самостоятельно не генерирует запросы, что в отдельных случаях является преимуществом

#### **2.4.3. Архитектура серверной части**

Так как использование Spring Boot обязывает соблюдать строгие требования к проектированию архитектуры, то было решено использовать стандартную архитектуру Spring приложения (контроллеры, сервисы и репозитории). Для непосредственной работы и доступа к базе данных используются мапперы описанные в виде интерфейсов и реализацией описанной в xml файлах.

#### **2.4.4. Клиент**

Для написания клиентской части приложения была также использован язык Java 17, так как помимо инструментов для создания серверных приложений, он обладает и инструментами для написания клиентских приложений, что в свою очередь обеспечивает удобство поддержки как серверной, так и клиентской части системы.

Также для удобного и сложного парсинга запросов, было решено использовать библиотеку Gson, так как она позволяет с легкостью разбирать сложные JSON объекты, например списки.

Главным инструментом разработки клиента, был выбран фреймворк Vaadin версии 23.2.6. Он позволяет писать веб-приложения полностью на Java, что позволяет использовать один и тот же удобный инструмент.

#### **2.4.5. Архитектура клиентской части**

Фреймворк Vaadin удобен еще и тем, что любое Vaadin-приложение по умолчанию является и Spring-Boot приложением, что означает использование архитектуры Spring-приложения. Поскольку в случае с клиентским приложением отсутствует взаимодействие с базой данных, а за обработку запросов отвечают уже не контроллеры, то соответственно меняется и стандартная архитектура. Следовательно приложение будет состоять из представлений(view), слушателей(listener) — основных частей приложения. Также подразумевается наличие сущностей(entity) и объектов DTO.

### 3. Реализация

#### 3.1. Диаграмма вариантов использования

На Рисунке 1 изображена диаграмма прецедентов для неавторизованного пользователя. Единственное, что может сделать такой пользователь — это войти в систему (указать свои учетные данные), либо зарегистрироваться в системе, для дальнейшего входа в систему.

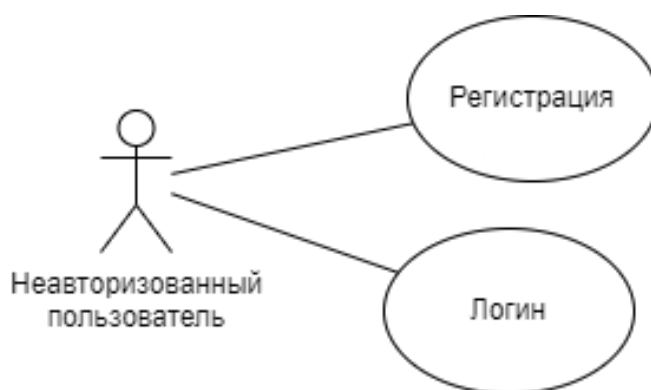


Рисунок 1 - Диаграмма прецедентов для неавторизованного пользователя

На Рисунке 2 изображена диаграмма прецедентов для авторизованного в системе пользователя. Ему доступен весь функционал приложения.

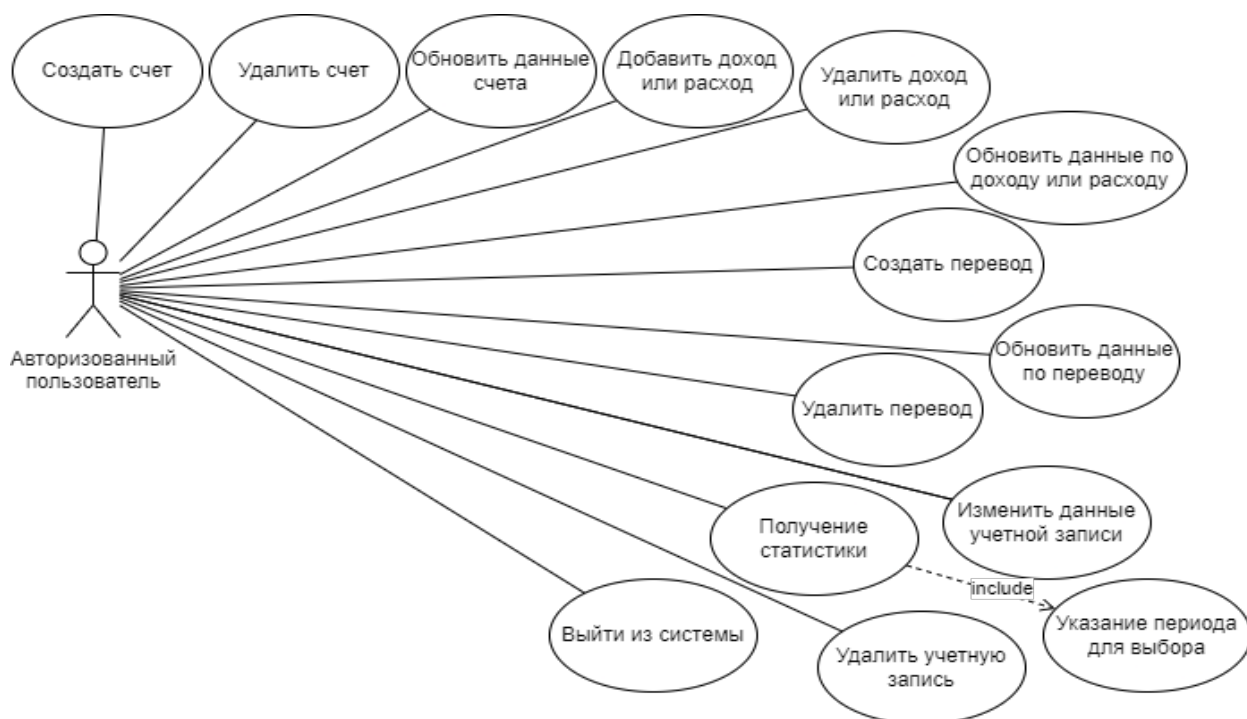


Рисунок 2 - Диаграмма прецедентов для авторизованного пользователя

### 3.2. Диаграмма классов

На Рисунке 3 изображена диаграмма классов приложения. На ней указаны все классы являющиеся сущностями. Они прямо отражают структуру базы данных и являются представлением всех основных таблиц.

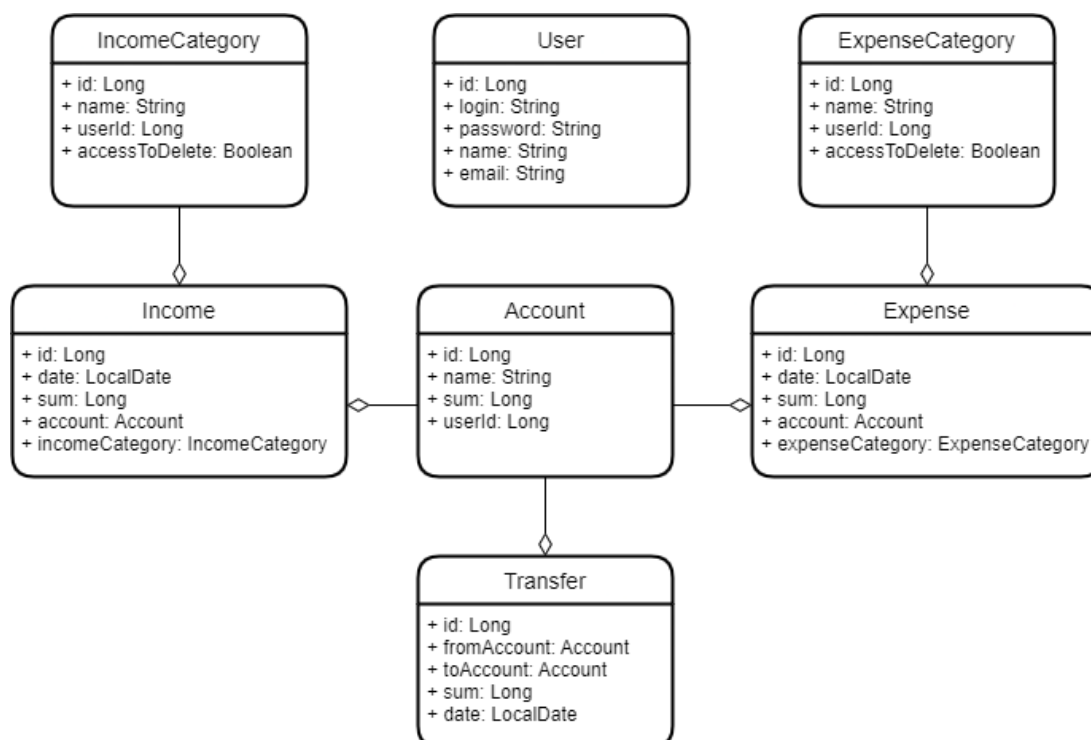


Рисунок 3 - Диаграмма классов приложения

### 3.3. Диаграмма таблиц базы данных

На Рисунке 4 изображена диаграмма базы данных. На ней указаны как основные таблицы, реализующие бизнес-логику приложения (user, account, transfer, income\_category, expense\_category, income, expense), так и таблицы отвечающие за работу дополнительных систем (databasechangelog, databasechangeloglock нужны для работы LiquiBase).



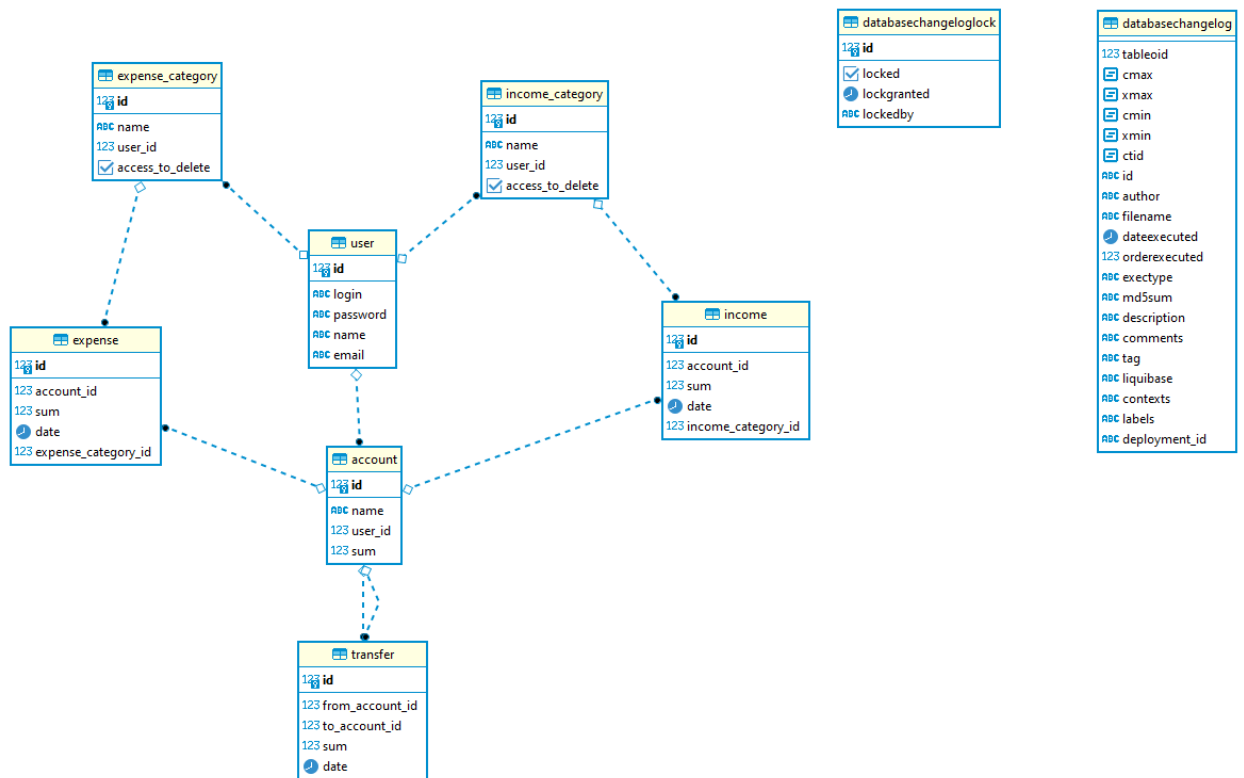
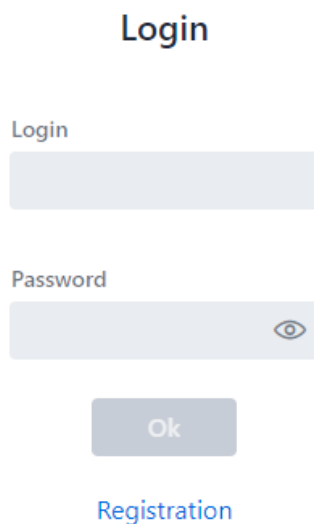


Рисунок 4 - Диаграмма базы данных

### 3.4. Реализация интерфейса

Первое, что видит пользователь, когда заходит в приложение, это страница входа (Рисунок 5).



The image shows a login form with the title "Login" in blue. Below the title are two input fields: "Login" and "Password". The "Login" field is a simple text box. The "Password" field is a text box with a small eye icon on the right side, indicating a toggle for password visibility. Below these fields is a button labeled "Ok" and a link labeled "Registration" in blue text.

Рисунок 5 - Страница входа

По умолчанию кнопка Ok не активна и недоступна для нажатия. Чтобы ее активировать необходимо заполнить поле Login и Password. После этого кнопка станет доступна для нажатия и можно будет войти в систему, если введенные данные верны (Рисунок 6).

## Login

Login

Password



Ok

[Registration](#)

Рисунок 6 - Страница входа  
после активации кнопки Ok

Также можно перейти на страницу регистрации (Рисунок 7) по указанной ниже ссылке [Registration](#).

## Registration

The registration form consists of five input fields and one button, all with a light gray background and rounded corners. The fields are labeled 'Email', 'Password', 'Confirm password', 'Login', and 'Name' in a small, dark gray font. The 'Password' and 'Confirm password' fields include a small eye icon on the right side, indicating a toggle for password visibility. Below the fields is a single button labeled 'Ok'.

Email

Password

Confirm password

Login

Name

Ok

Рисунок 7 - Страница регистрации

Также как и на странице входа, по умолчанию кнопка для подтверждения регистрации Ok – неактивна. Соответственно для ее активации необходимо заполнить все пустые поля (Email, Password, Confirm password, Login, Name). Итоговый вид на Рисунке 8.

## Registration

The registration form consists of the following elements:

- Email:** A text input field containing the placeholder text "email@email.com".
- Password:** A text input field with masked characters (dots) and a toggle icon (an eye) to show or hide the password.
- Confirm password:** A text input field with masked characters (dots) and a toggle icon (an eye) to show or hide the password.
- Login:** A text input field containing the placeholder text "Ign".
- Name:** A text input field containing the placeholder text "Аристарх".
- Ok button:** A blue rectangular button with the text "Ok" in white.

Рисунок 8 - Страница  
регистрации с активной  
кнопкой Ok

После того, как пользователь вошел в систему, он перенаправляется на страницу своего профиля (Рисунок 9). Вверху указано имя пользователя указанное при регистрации. Далее идут кнопки для взаимодействия со своим профилем. Edit data отвечает за редактирование данных пользователя, Change password для смены пароля, Exit для выхода из учетной записи(уничтожается сессия на стороне сервера и пользователь перенаправляется на страницу входа) и Delete profile для безвозвратного удаления пользователя и всех его данных.

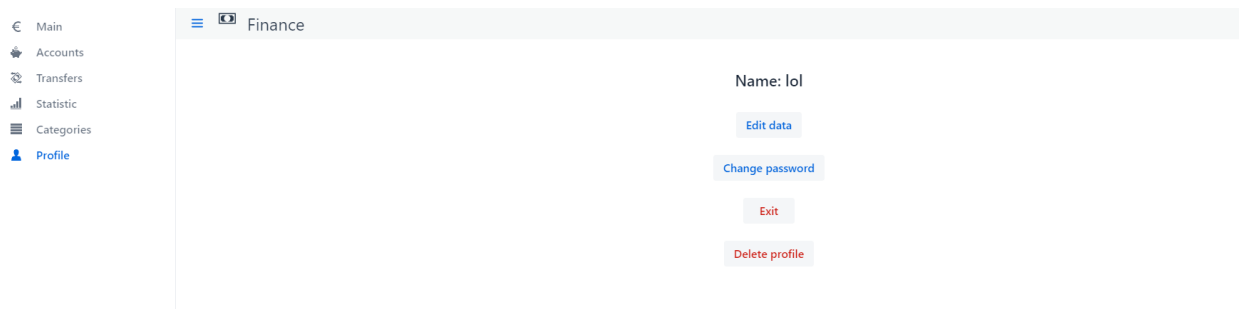


Рисунок 9 - Страница профиля пользователя

При нажатии кнопки Edit data появляется диалоговое окно (Рисунок 10) на котором указаны все доступные для редактирования пользовательские данные (Email, Login, Name). Кнопка Change изначально активна, но в том случае если какое-то из полей будет пустым или значение в поле Email не будет соответствовать формату почты, то кнопка перестанет быть активной.

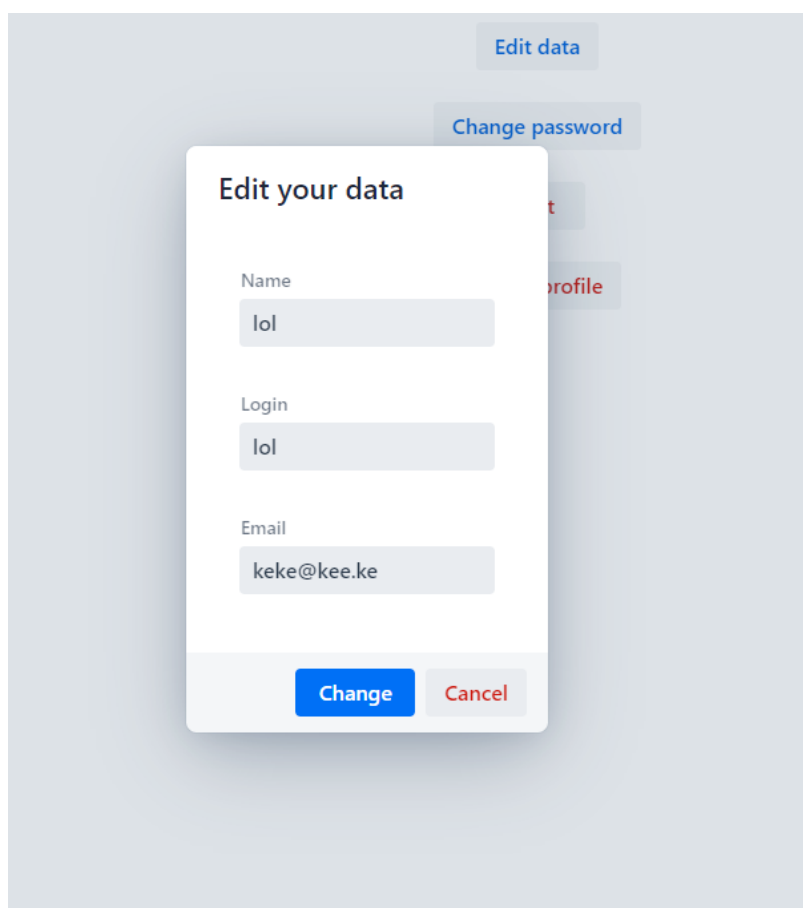


Рисунок 10 - Диалоговое окно для редактирования данных пользователя

Слева расположено меню, в котором присутствует свободная навигация, позволяющая перемещаться между всеми страницами приложения.

Также присутствует страница категорий для доходов(Expense) и расходов(Income), изображенная на Рисунке 11. Изначально у пользователя нет своих категорий(не считая созданных по умолчанию категорий Other для доходов и расходов, которые не отображаются пользователю в списке категорий). Для того чтобы их создать необходимо нажать кнопку Create.

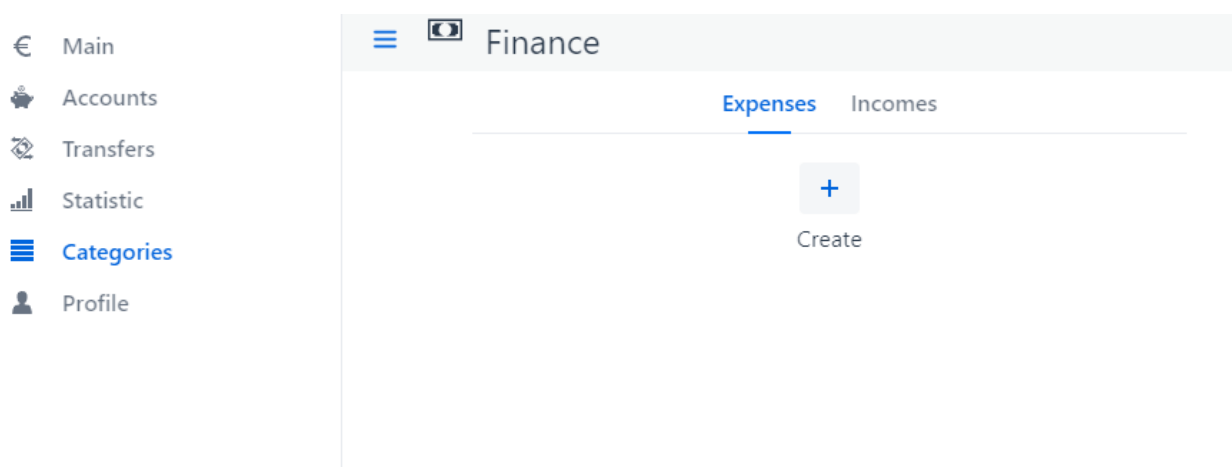


Рисунок 11 - Страница доходов и расходов

По нажатию на эту кнопку, появляется диалоговое окно для создания новой категории (Рисунок 12). Здесь требуется указать только одно поле Category name. Также, как и в случае с другими диалоговыми окнами, кнопка Save изначально активна, но если поле с названием категории будет пустым, то и кнопка станет неактивной.

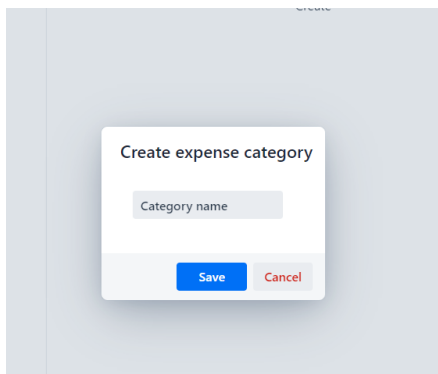


Рисунок 12 - Диалоговое окно для создания новой категории

После создания новой категории расходов, страница категории будет выглядеть следующим образом (Рисунок 13).

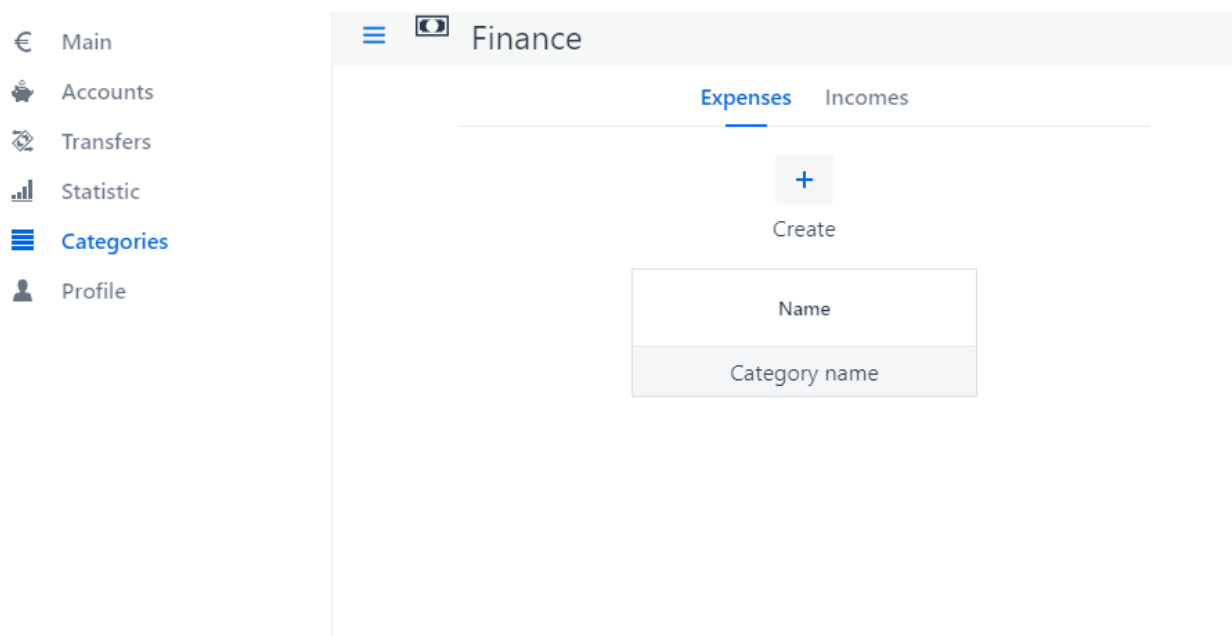


Рисунок 13 - Страница категорий после создания своей категории расходов

Как можно увидеть, появляется таблица категорий, с единственным столбцом — название категории. Чтобы тем или иным образом взаимодействовать с данными в этой таблице необходимо нажать правой кнопкой мыши по таблице(если страница открыта на телефоне то удерживать ту строку, с которой необходимо произвести манипуляции). После этого появится контекстное меню с тремя кнопками (Рисунок 14). Create для создания новой категории, Edit для редактирования данных выбранной категории и Delete для удаления категории. При редактировании появляется похожее диалоговое окно как и для создания, но вместо значения по умолчанию, указывается текущее название категории. Также на окно для редактирования распространяются все те же правила, что и для создания(кнопка активна только тогда, когда все поля непустые).



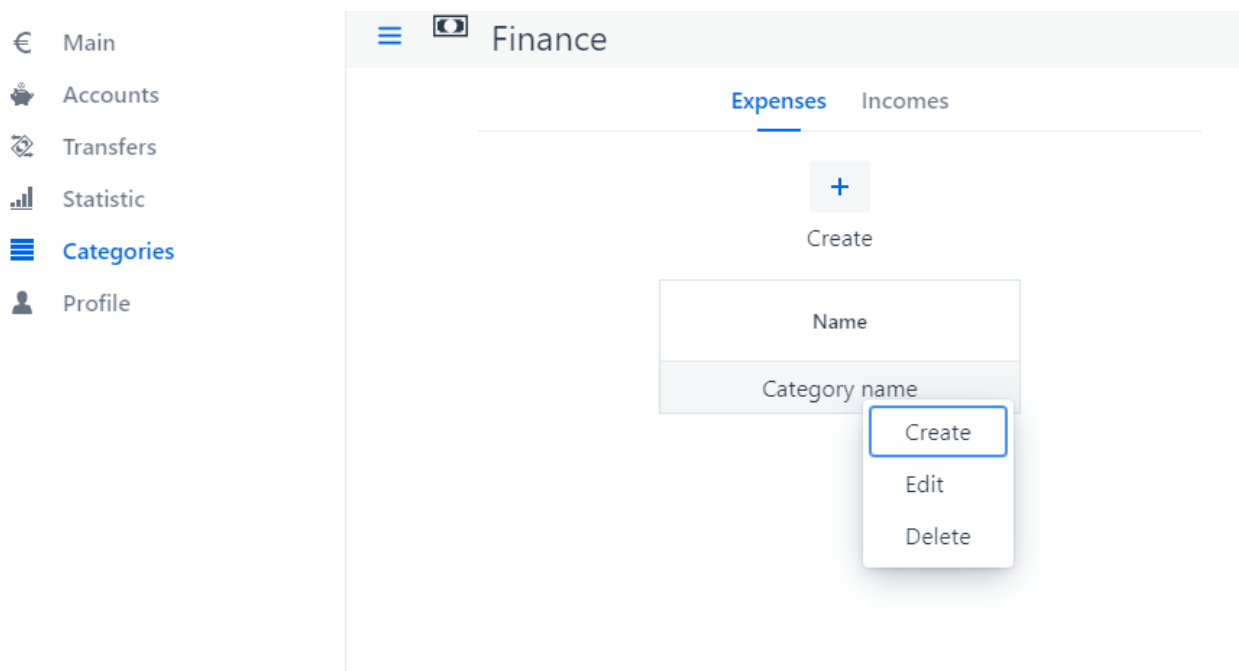


Рисунок 14 - Контекстное меню для таблицы категорий

На странице счетов (Рисунок 15) есть таблица, в которой указаны все счета пользователя. Имеются столбцы с названиями счетов и суммой на счете. Для создания, редактирования и удаления также по нажатию правой кнопки мыши, открывается контекстное меню, с кнопками для создания(Create), изменения(Edit) и удаления>Delete) счетов.

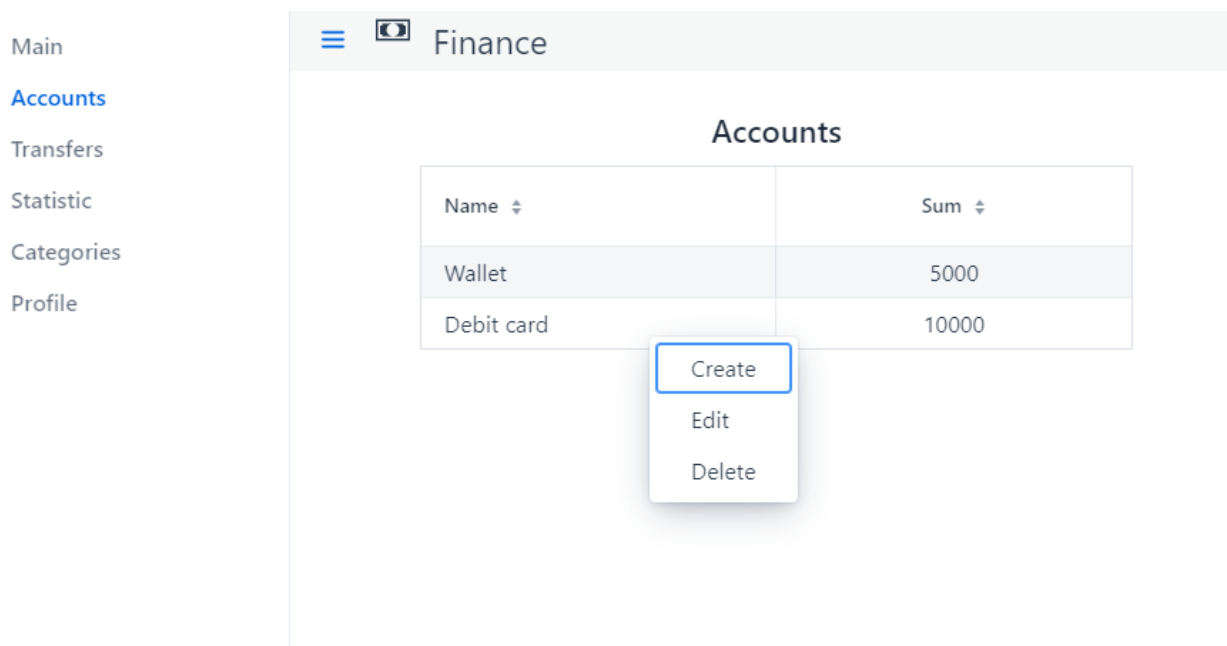


Рисунок 15 - Страница счетов пользователя с открытым контекстным меню

По нажатию на кнопки создания и редактирования открываются диалоговые окна (Рисунок 16) в которых указано два поля: название счета и сумма на счете. Аналогично другим окнам, кнопка для подтверждения операции активна только если все поля непустые.

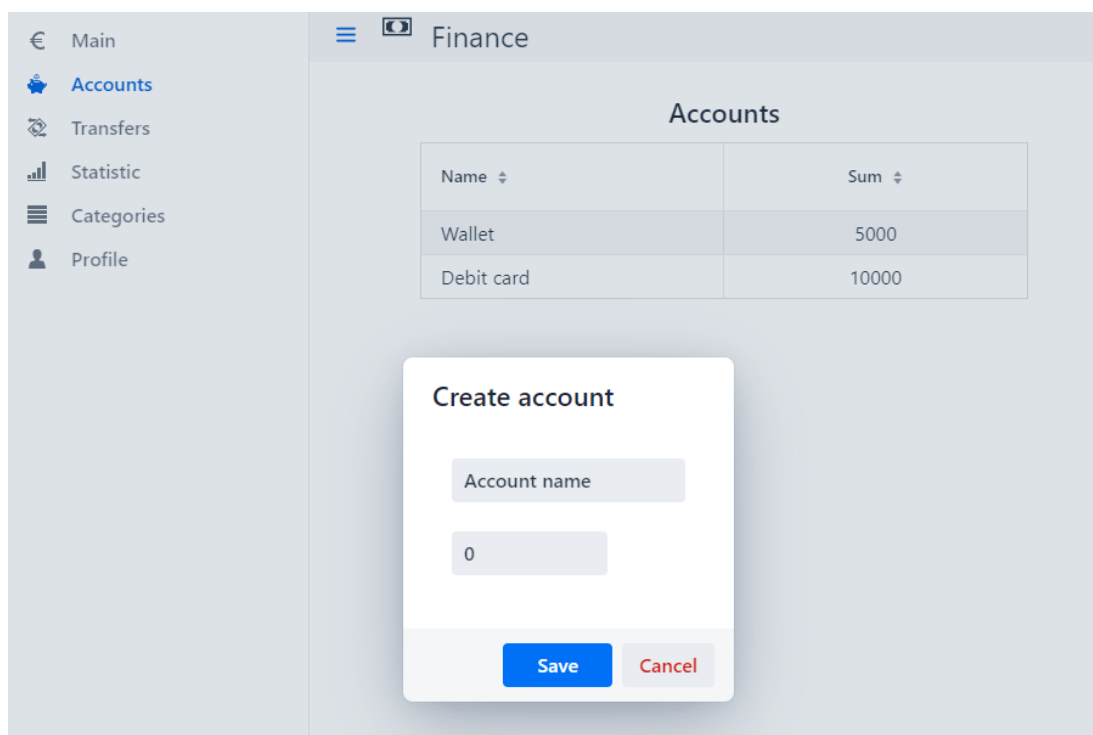


Рисунок 16 - Диалоговое окно для создания счета

Следующей страницей является страница переводов (Рисунок 17). На ней есть таблица со всеми переводами. Также имеется возможность навигации по временным периодам (день, неделя, месяц, год и свободно настраиваемым периодом). Для работы с переводами необходимо также нажать правой кнопкой мыши по таблице и в открытом контекстном меню выбрать желаемое действие. При создании и редактировании открывается диалоговое окно (Рисунок 18) работающее по тем же принципам, что и предыдущие. Разница заключается в том, что если у пользователя имеется всего один счет или их вообще нет, то переводы становятся недоступны и при переходе на эту страницу пользователь получает соответствующее сообщение.

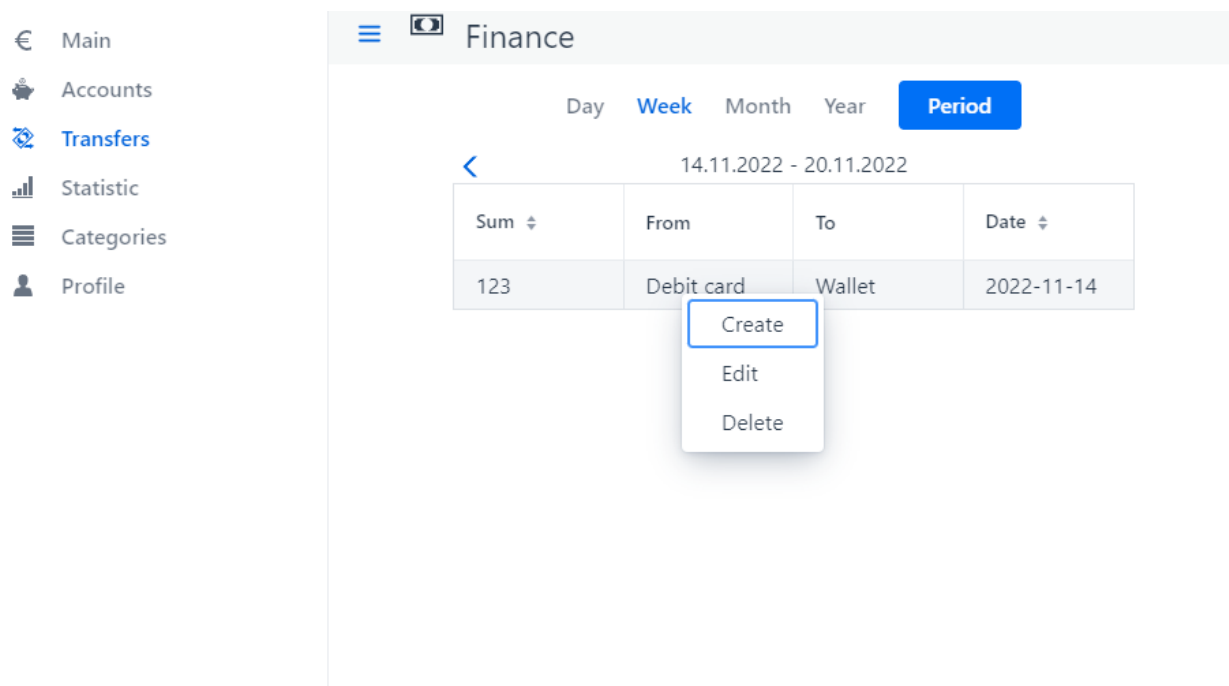


Рисунок 17 - Страница переводов с открытым контекстным меню

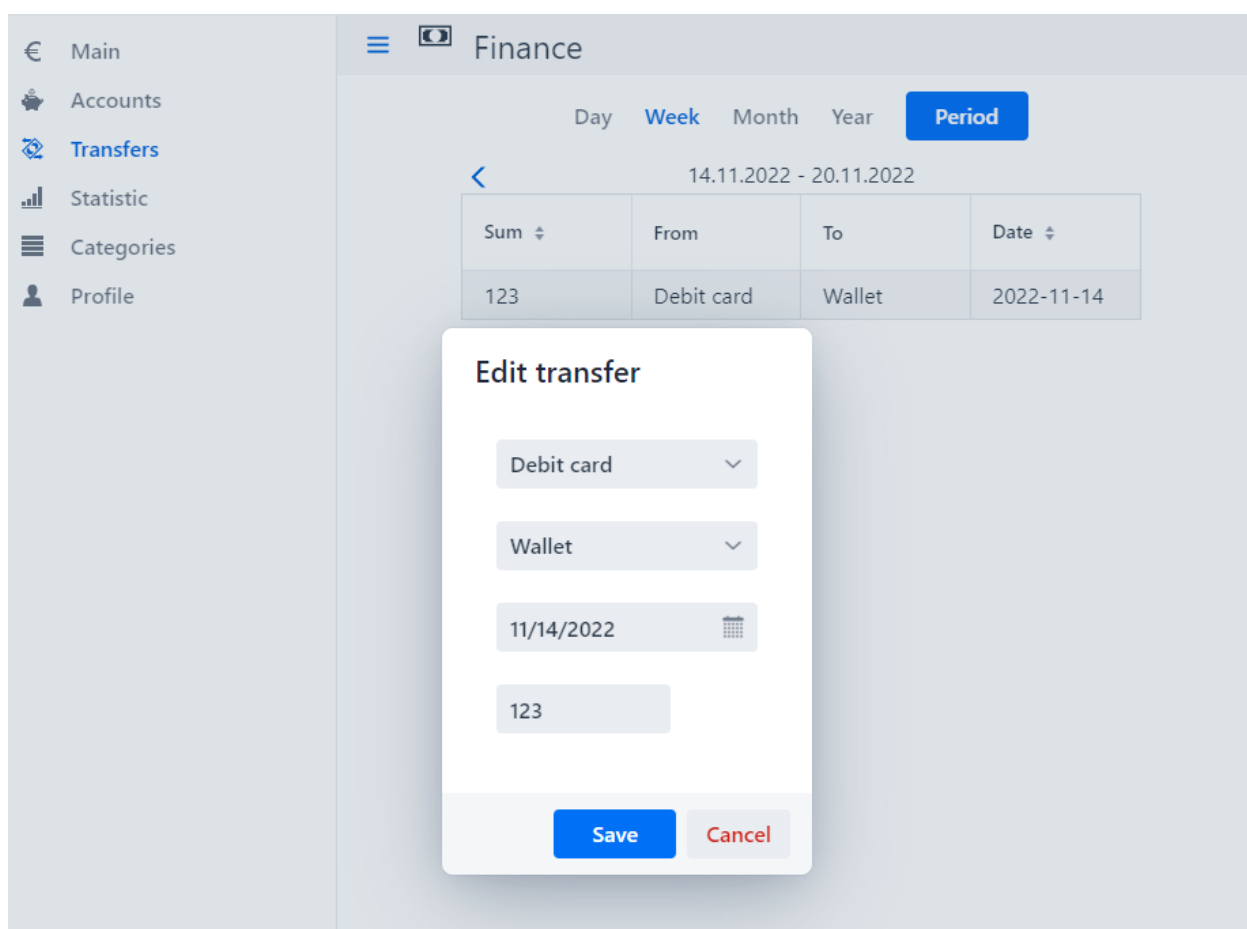


Рисунок 18 - Диалоговое окно для изменения данных перевода

Самой главной страницей является страница Main (Рисунок 19), на которой происходит вся работа с доходами и расходами, с использованием разнообразных виджетов. Имеется навигация как между доходами и расходами, так и между временными промежутками аналогично переводам. Также присутствует таблица, в которой указаны расходы и доходы, в зависимости от выбранной вкладки(Expenses, Incomes). Для работы с ней используется сценарий аналогичный предыдущим (использование контекстного меню вместе с диалоговым окном (Рисунок 20)).

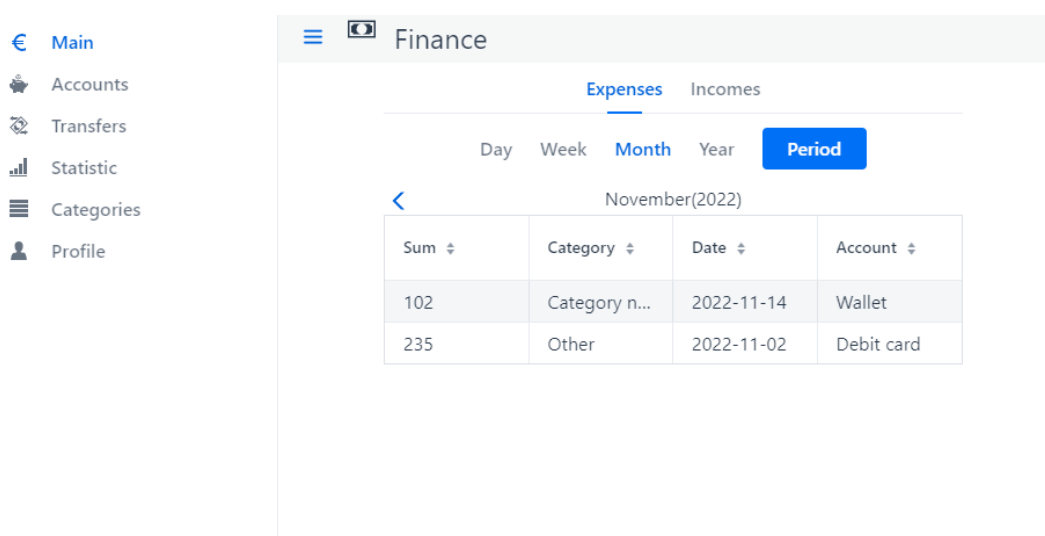


Рисунок 19 - Главная страница (доходов и расходов)

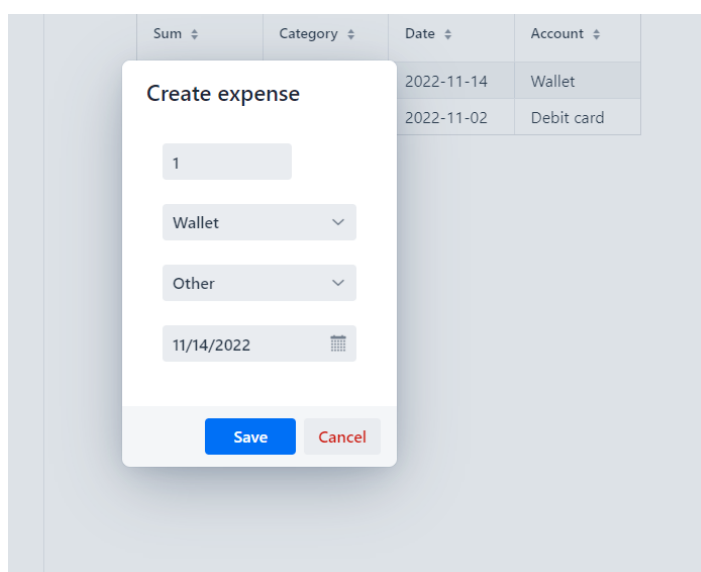


Рисунок 20 - Диалоговое окно для создания расхода

На странице статистики (Рисунок 21) можно смотреть статистику по доходам и расходам за любой указанный период времени, аналогично переводам. Статистика выводится в виде таблице, в которой три столбца: название категории дохода или расхода, сумма (весь доход или расход по данной категории), процент от общей суммы доходов или расходов. Поскольку это данные являющиеся результатом расчетов, то поэтому вся таблица неизменяемая, и является доступной только для просмотра.

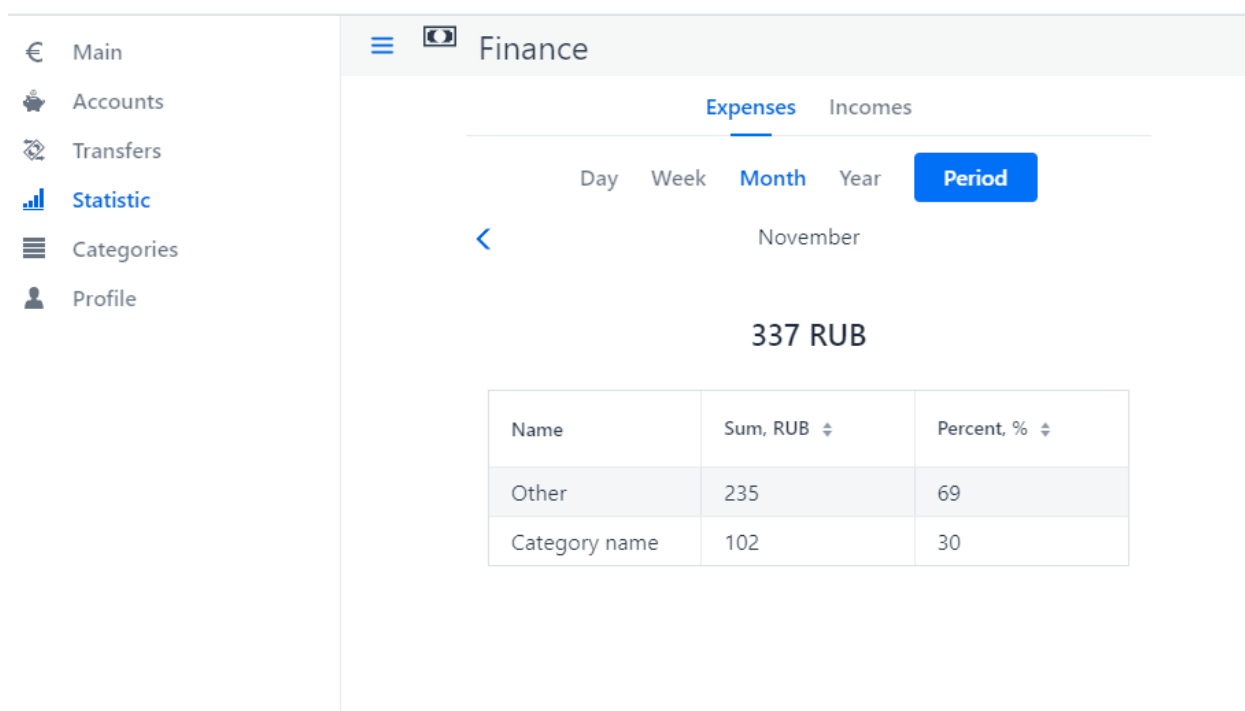


Рисунок 21 - Страница статистики по доходам и расходам

#### **4. Тестирование**

На завершающем этапе разработки было проведено ручное функциональное тестирование при помощи тест-кейсов. Основная цель такого тестирования заключается в определении степени соответствия разработанного продукта его функциональным требованиям, то есть способность при определенных условиях решать задачи, необходимые пользователям. По результатам функционального тестирования были выявлены и исправлены немногочисленные ошибки. Вторым этапом тестирования стало приемочное тестирование, проводимое предполагаемыми конечными пользователями системы на основании набора тестовых сценариев, покрывающих основные функции системы. Все сценарии были пройдены успешно

## **Заключение**

В ходе данной курсовой работы были разработаны серверное и клиентское приложения образующие информационную систему для учета финансов.

Перед разработкой был проведен анализ предметной области, в результате которого были выявлены преимущества и недостатки найденных аналогов, которые были учтены в ходе разработки данного приложения.

Также в ходе разработки, были расширены и углублены знания в области использования различных инструментов разработки. Были подробно изучены способы их применения для создания подобных приложений.

Были успешно пройдены все этапы разработки, по завершению, которых были выполнены все предъявляемые требования.