МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ “ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Приложение для учета финансов «Finance»

Курсовой проект

09.03.02 Информационные системы и технологии

Программная инженерия в информационных системах

Зав. кафедрой _____ д. ф.-м. н., профессор С.Д. Махортов

Руководитель _____ *ст. преподаватель* В.С. Тарасов

_____ И.В. Клейменов

Обучающиеся:

_____ А. В. Тарлыков

_____ Н. М. Кошелев

Воронеж 2022

Содержание

Введение

Пользователей интернета с каждым годом становится все больше, а значит, что и число предлагаемых им интернет-услуг растет. Приложения учета финансов которые совмещают достаточно много вариантов пользовательского взаимодействия крайне востребованы в современном обществе.

Приложения не только помогут проанализировать затраты, но и уберегут от ненужных покупок. С учетом факта вы контролируете, укладываетесь ли в сумму доходов. Каждый месяц у вас должно быть четкое понимание денежных потоков: какие ожидаются доходы и расходы. Контроль нужен не для того, чтобы фиксировать потраченные деньги постфактум. Это критерий, который дает осознанность в их расходовании. Когда у вас появится привычка вести учет, вы начнете задумываться, на что тратите деньги. В дальнейшем это поможет анализировать их и корректировать для достижения намеченных финансовых целей.

Для аналитики по месяцам и годовых результатов вам пригодится эксель-таблица. После того, как вы спланировали там свой бюджет, таблицу можно закрыть. Наступает очередь приложения.

Приложение для учета доходов и расходов позволяет задавать нужные категории доходов (например, зарплата, пособие и т. п.) и расходов (ЖКХ, обучение, покупки)

1. Анализ предметной области

1.1 Терминология

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.

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 завершенная банковская операция, в процессе осуществления которой происходит перевод определенной суммы денег с одного счёта на другой.

GitHub — веб-сервис для хостинга IT-проектов и их совместной разработки, основанный на системе контроля версий Git.

Заказчик — лицо (физическое или юридическое), заинтересованное в выполнении исполнителем работ, оказании им услуг или приобретении у продавца какого-либо продукта (в широком смысле).

Исполнитель — человек или коллектив людей, вооруженных набором инструментов и обученный выполнению некоторой совокупности операций в заданной последовательности.

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 (заработная 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 жизнь, нужно иметь финансовый резерв.

1.3 Обзор аналогов

Будем рассматривать популярные платформы, завоевавшие наибольшее признание у пользователей.

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.

Минусы:

- Навязчивая реклама остается в приложении даже после покупки премиум-подписки.
- Функция совместного учета платная.

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.

Минусы:

- Возникают проблемы с синхронизацией с приложениями банков и другими устройствами.
- Функции совместного и мультивалютного учета и планирования платежей платные.

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.

Минусы:

- Нельзя синхронизировать приложение с российскими банками.
- Возможности учета в нужной валюте платные.

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.

Минусы:

- Функции защиты входа от посторонних лиц и возможности прикреплять фотографии и геометки платные.

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.

Минусы:

- Нет синхронизации с банками.
- В бесплатной версии подходит записывать расходы и доходы может только один пользователь.

2. Постановка задачи

2.1 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 серверной части приложения. Это по сути основная задача 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.

2.2 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, т.к. 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".

2.3 Средства реализации

СУБД:

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

Сервер:

- Для написания серверной части было решено использовать Java версии 17 т. к. это один из самых популярных языков программирования и очень часто используется для написания серверов.
- Для упрощения разработки было решено использовать Spring Boot Framework т. к. у него уже есть готовый набор инструментов для написания веб и enterprise приложений (spring контейнер, spring security и т. д.)
- В качестве средства миграции базы данных было решено использовать LiquiBase т. к. он удобен в использовании и изначально разрабатывался для Java
- Для работы с данными было решено использовать MyBatis т. к. относительно своих аналогов он лучше подходит для написания сложных запросов к базе данных и самостоятельно не генерирует запросы, что в отдельных случаях является преимуществом

Архитектура приложения:

- Т. к. 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.

3. Планирование работ

3.1 Основные виды работ

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

- Постановка задачи
- Анализ предметной области
- Анализ аналогов
- Определение функциональных требований
- Выбор инструментальных средств

На этапе разработки приложения должна быть выполнены следующие виды работ:

- Создание репозитория Git
- Установка и настройка необходимых инструментальных средств
- Проектирование БД
- Проектирование архитектуры приложения
- Разработка серверной части приложения, взаимодействующей с БД

На этапе тестирования приложения должна быть выполнены следующие виды работ:

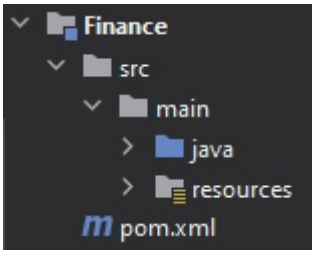
- Функциональное тестирование приложения

На этапе разработки документации должно быть выполнено создание технической документации с описанием функциональности и интерфейса созданного приложения.

3.2 Архитектура программного продукта

Приложение написано на языке программирования Java с использованием фреймворка Spring Boot. Эти инструменты обязывают использовать строго определенную архитектуру приложения. Так как для сборки проект используется Maven, то одним из основных файлов является pom.xml в котором описаны все используемые зависимости приложения.

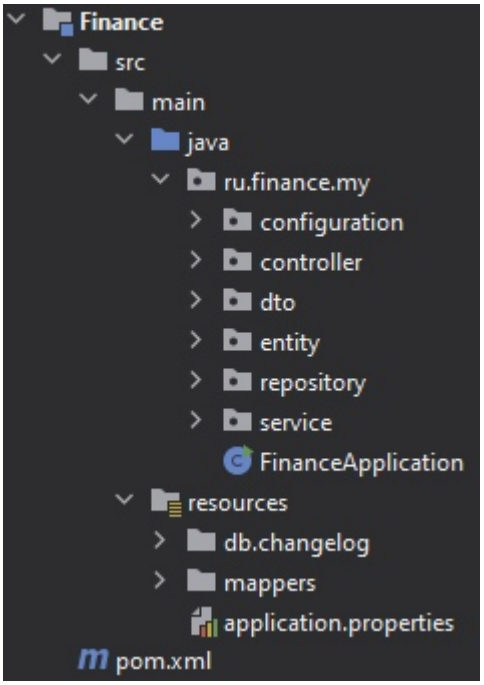
В директории src/main есть два поддиректория /java и /resources. В первом находится код приложения написанный на Java, а во втором дополнительные файлы используемые Spring Boot, LiquiBase и MyBatis для корректной работы.



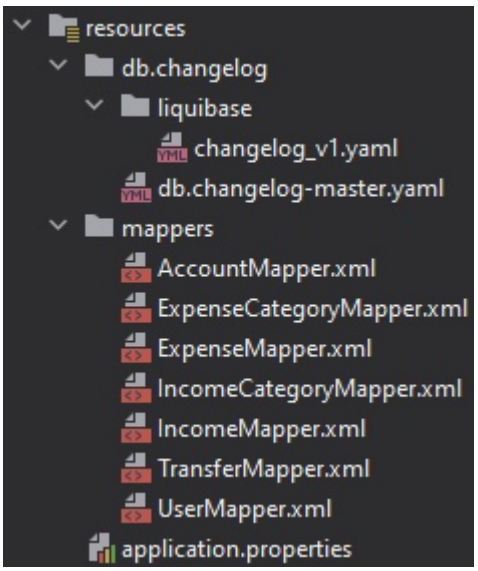
Далее в /java/ru/finance/my имеется сразу несколько основных пакетов приложения:

- configuration хранит классы java-конфигурации
- controller хранит классы контроллеров, которые представляют собой так называемые "точки входа" для запросов к серверу
- dto хранит классы DTO (Data Transfer Object). Объекты этих классов используются для передачи данных между сервером и клиентом
- entity хранит классы сущности которые являются отражением таблиц в базе данных

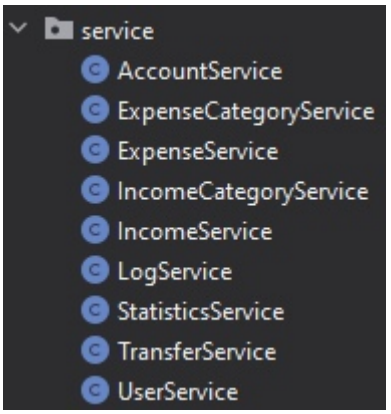
- repository хранит классы репозиториев, эти классы являются инструментом для доступа к базе данных - service хранит классы сервисов, которые реализуют бизнес логику приложения
- service хранит классы сервисов, которые реализуют бизнес логику приложения
- FinanceApplication - это точка входа программы, именно здесь происходит запуск сервера



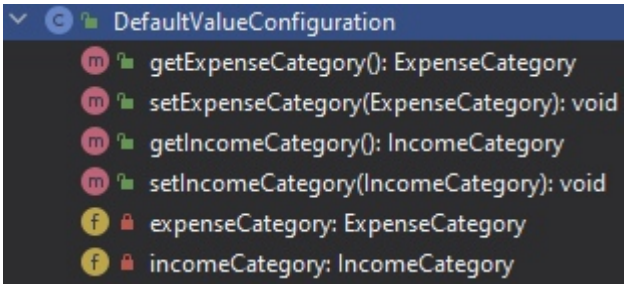
В /resources находятся два директория: db.changelog, mappers и один файл application.properties в котором указаны настройки приложения. db.changelog используется LiquiBase. В нем указываются файлы миграции базы данных mappers используется MyBatis для маппинга SQL запросов.

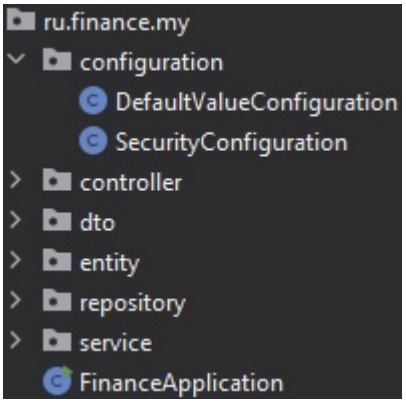


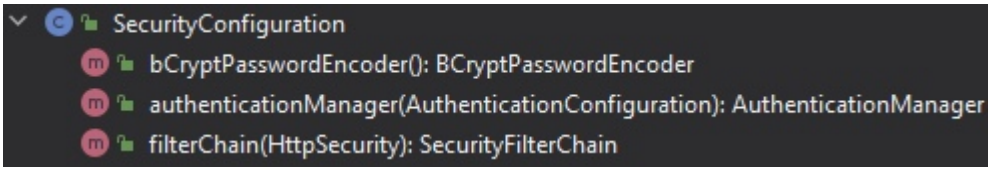
Структура пакета service:



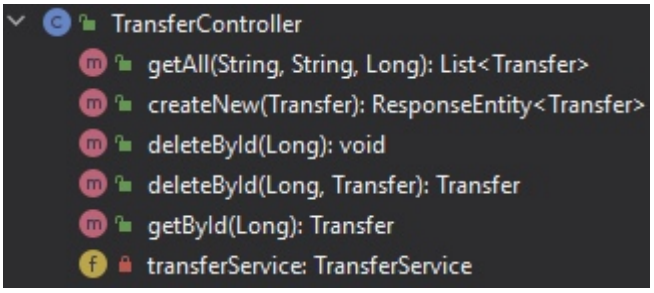
Структура пакета configuration:

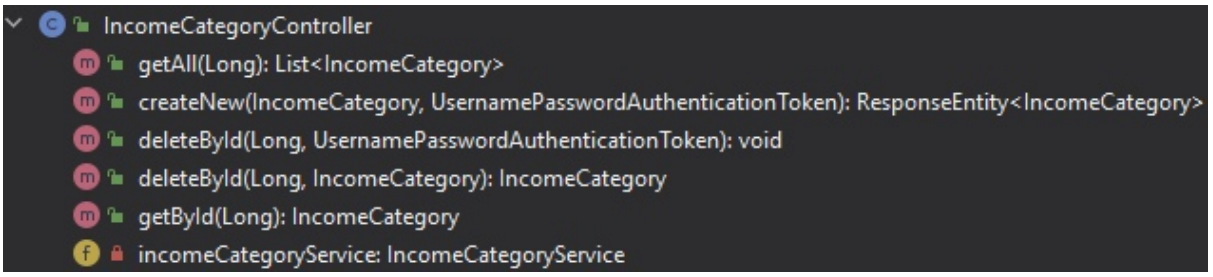


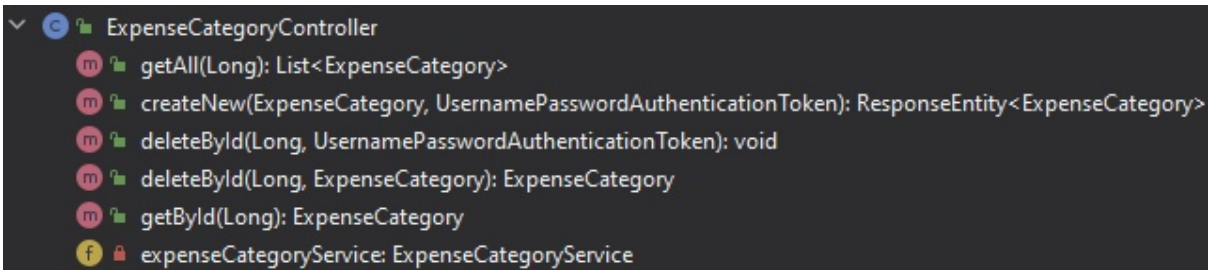


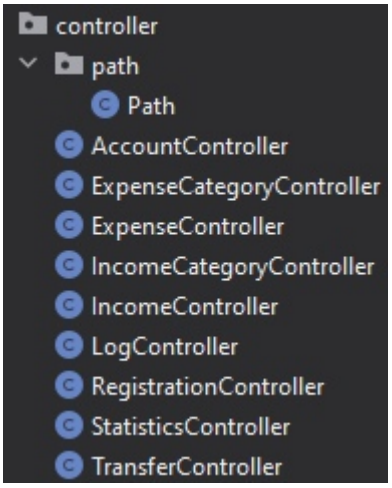


Структура пакета controller:

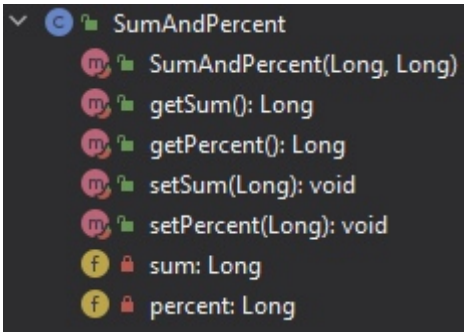


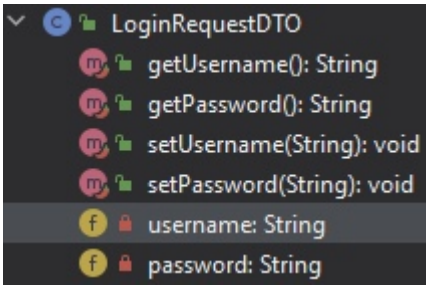


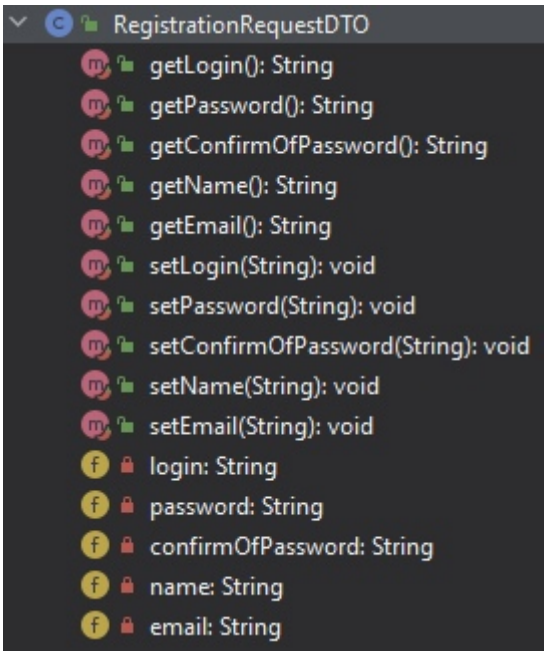


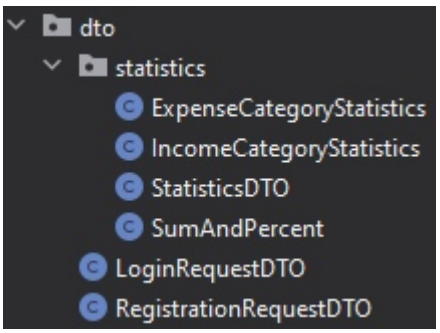


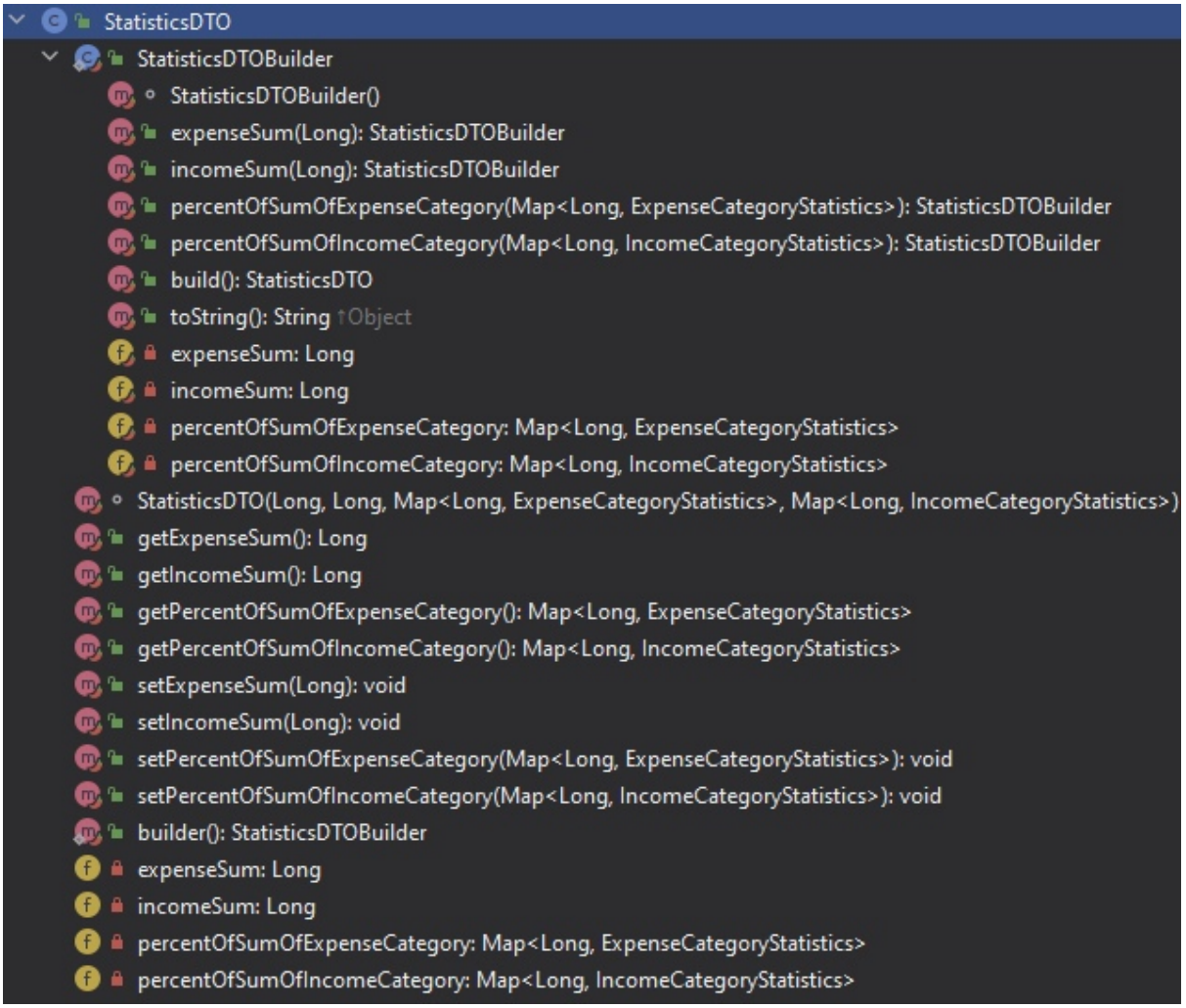
Структура пакета dto:



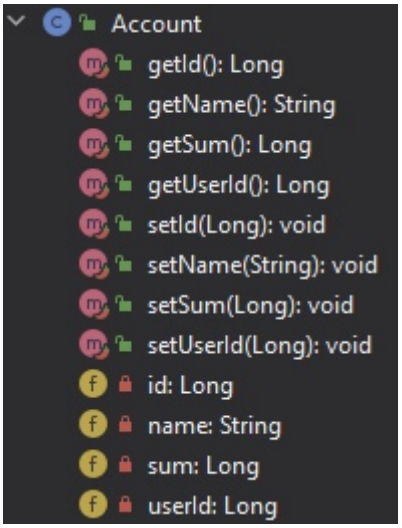


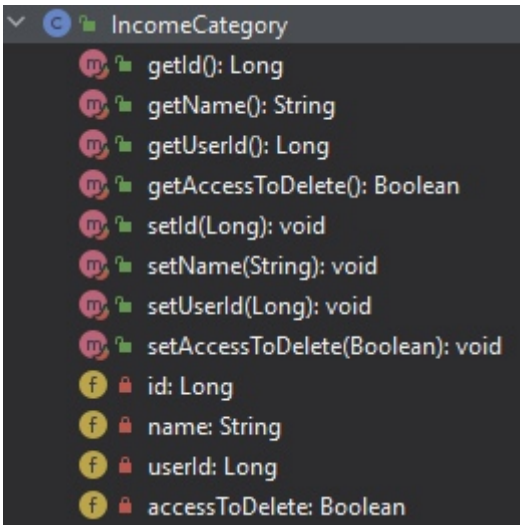


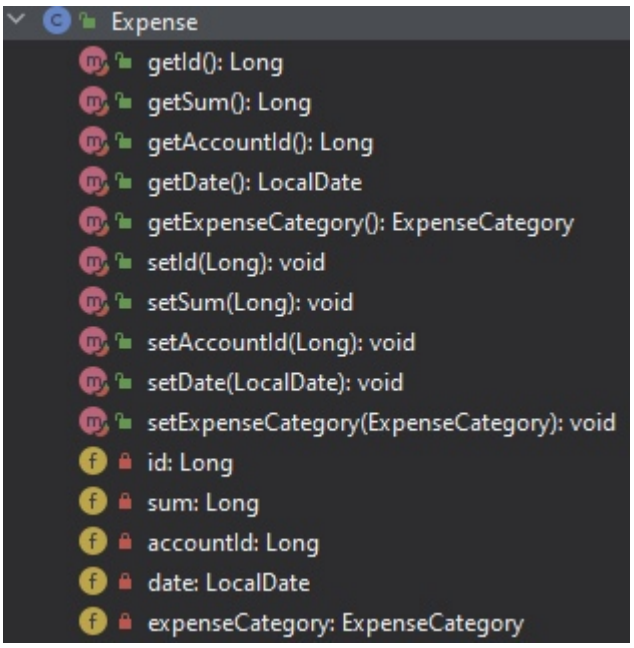

dto
statistics
ExpenseCategoryStatistics
IncomeCategoryStatistics
StatisticsDTO
SumAndPercent
LoginRequestDTO
RegistrationRequestDTO

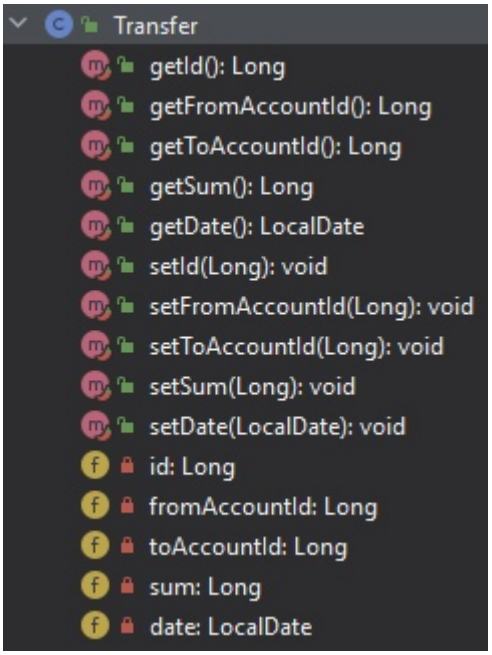

StatisticsDTO
StatisticsDTOBuilder
StatisticsDTOBuilder()
expenseSum(Long): StatisticsDTOBuilder
incomeSum(Long): StatisticsDTOBuilder
percentOfSumOfExpenseCategory(Map<Long, ExpenseCategoryStatistics>): StatisticsDTOBuilder
percentOfSumOfIncomeCategory(Map<Long, IncomeCategoryStatistics>): StatisticsDTOBuilder
build(): StatisticsDTO
toString(): String ↑Object
expenseSum: Long
incomeSum: Long
percentOfSumOfExpenseCategory: Map<Long, ExpenseCategoryStatistics>
percentOfSumOfIncomeCategory: Map<Long, IncomeCategoryStatistics>
StatisticsDTO(Long, Long, Map<Long, ExpenseCategoryStatistics>, Map<Long, IncomeCategoryStatistics>)
getExpenseSum(): Long
getIncomeSum(): Long
getPercentOfSumOfExpenseCategory(): Map<Long, ExpenseCategoryStatistics>
getPercentOfSumOfIncomeCategory(): Map<Long, IncomeCategoryStatistics>
setExpenseSum(Long): void
setIncomeSum(Long): void
setPercentOfSumOfExpenseCategory(Map<Long, ExpenseCategoryStatistics>): void
setPercentOfSumOfIncomeCategory(Map<Long, IncomeCategoryStatistics>): void
builder(): StatisticsDTOBuilder
expenseSum: Long
incomeSum: Long
percentOfSumOfExpenseCategory: Map<Long, ExpenseCategoryStatistics>
percentOfSumOfIncomeCategory: Map<Long, IncomeCategoryStatistics>

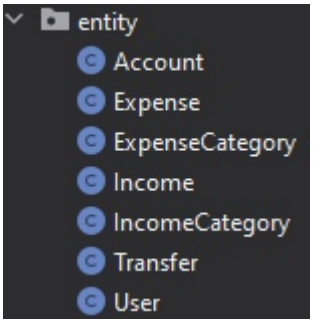
Структура пакета entity:



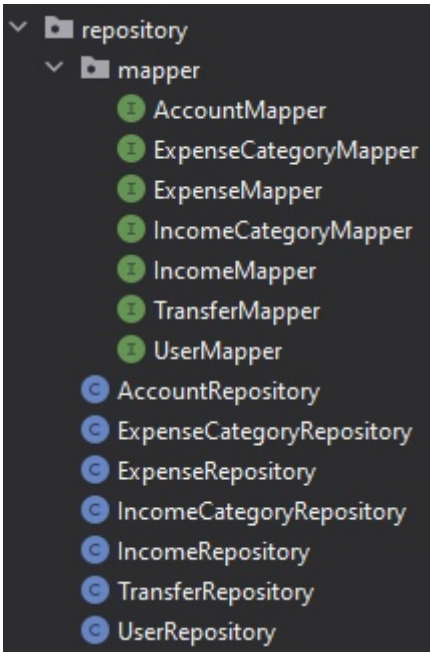


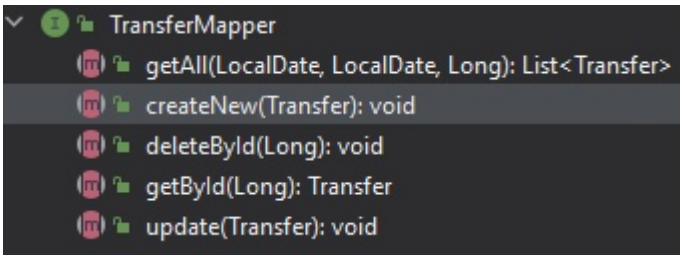

Expense
getId(): Long
getSum(): Long
getAccountId(): Long
getDate(): LocalDate
getExpenseCategory(): ExpenseCategory
setId(Long): void
setSum(Long): void
setAccountId(Long): void
setDate(LocalDate): void
setExpenseCategory(ExpenseCategory): void
id: Long
sum: Long
accountId: Long
date: LocalDate
expenseCategory: ExpenseCategory

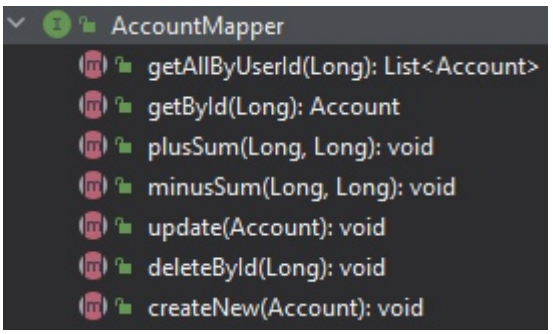

Transfer
getId(): Long
getFromAccountId(): Long
getToAccountId(): Long
getSum(): Long
getDate(): LocalDate
setId(Long): void
setFromAccountId(Long): void
setToAccountId(Long): void
setSum(Long): void
setDate(LocalDate): void
id: Long
fromAccountId: Long
toAccountId: Long
sum: Long
date: LocalDate


entity
Account
Expense
ExpenseCategory
Income
IncomeCategory
Transfer
User

Структура пакета repository:







3.3 Логическая схема базы данных

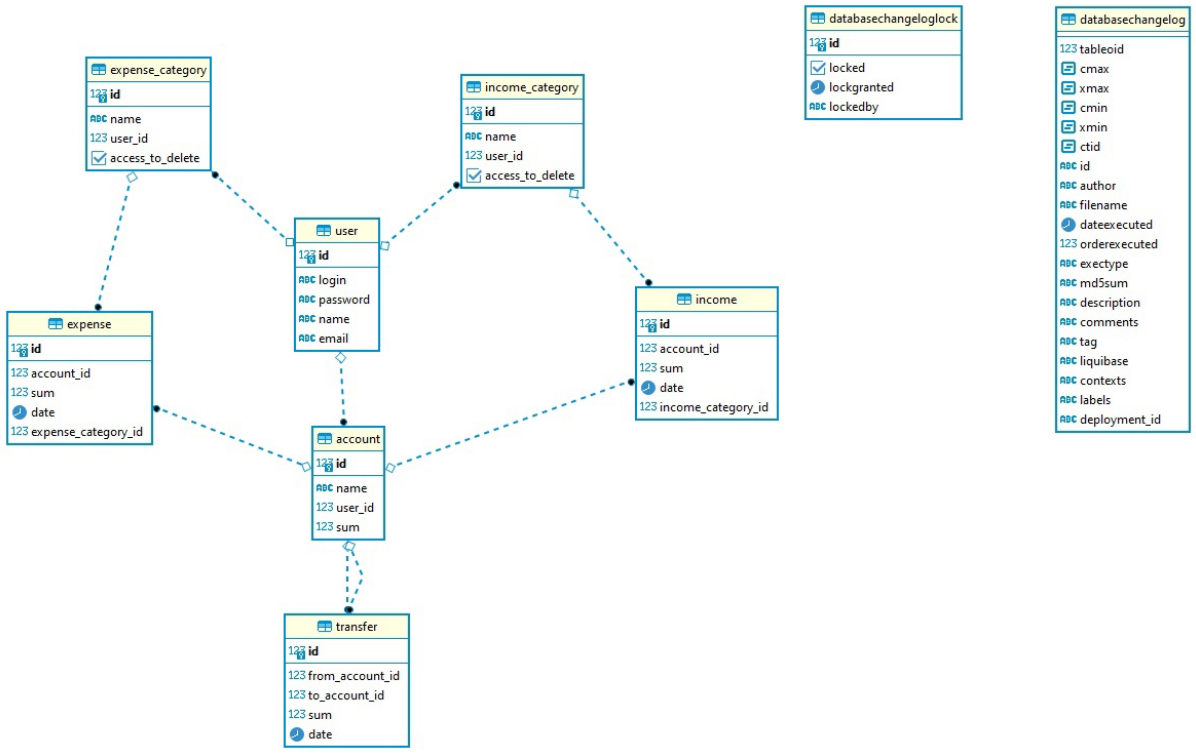
Структура базы данных. Данная диаграмма отражает структуру базы данных. Основой всей базы является таблица user в которой хранятся все данные о пользователях. Также одной из важнейших является таблица account с информацией о счетах пользователей.

Таблица expense_category и income_category хранят категории расходов и доходов соответственно. Они связаны с пользователем т.к. каждый пользователь может создавать свои категории доходов и расходов.

Таблицы expense и income хранят данные о расходах и доходах соответственно и связаны с категориями и счетами пользователя.

Таблица transfer хранит данные о переводах между счетами, а потому имеет сразу два внешних ключа на таблицу account.

Две другие таблицы: databasechangelog и databasechangeloglock требуются для корректной работы LiquiBase.



4. Заключение

В ходе данной работы была изучена предметная область по выбранной теме, получен дополнительный опыт работы с указанными выше инструментами и разработана серверная часть приложения для учета финансов.

Приложение поддерживает следующие функции:

- Регистрация и аутентификация
- Создание и удаление счетов
- Добавление доходов и расходов
- Создание и отмена перевода между счетами
- Получение статистики за указанный период времени
- Выход из учетной записи с текущего устройства