

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
«ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ  
УНИВЕРСИТЕТ»

Кафедра «ИНФОРМАЦИОННЫЕ СИСТЕМЫ И АВТОМАТИЗАЦИЯ  
ПРОИЗВОДСТВА»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к дипломному проекту

Тема: «Разработка веб-сервиса коммерческого производственного предприятия»

Факультет: повышения квалификации и переподготовки кадров

Специальность: 1-40 01 73

«Программное обеспечение информационных систем»

Группа: ИС-17

Исполнитель: Ивановский И.П.

Руководитель: Старикович Ю.А.

Утверждаю: Зав. кафедрой ИСАП  
Казаков В.Е.

Проект рассмотрен и допущен к защите «\_\_\_»\_\_\_\_\_ 2020 г.

Витебск, 2020

**ПРИЛОЖЕНИЕ Б**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ "ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ"**

Факультет: факультет повышения  
квалификации  
и переподготовки кадров

Кафедра ИСАП

**«Утверждаю»**

Зав. кафедрой \_\_\_\_\_  
(подпись)

« \_\_\_\_\_ » \_\_\_\_\_ 2020г.

**З А Д А Н И Е**  
**по дипломному проектированию**

Студенту Ивановскому И.П.

1. Тема проекта Разработка веб-сервиса коммерческого производственного предприятия Утверждена приказом по ВУЗу от 16.03.2020г № 90

2. Сроки сдачи студентом законченного проекта 24.08.2020г – 31.08.2020г

3. Исходные данные к проекту (тех.задание):

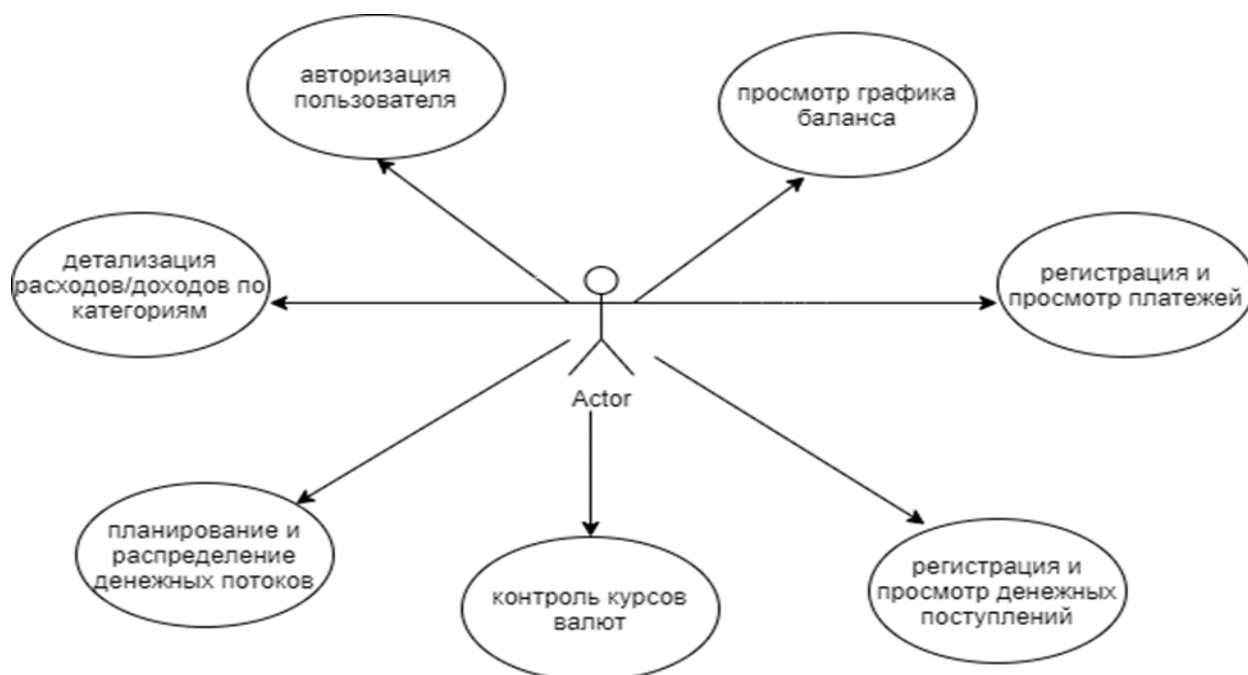
Разработать WEB приложения, финансового учета, для выявления финансового состояния сотрудников в организации.

Приложение должно позволять вести учет и анализ ежемесячных расходов и доходов пользователей приложения.

4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов) :

Разработать веб-приложение для сотрудников комерческого производственного предприятия с потенциальной возможностью его последующей комерческой реализации.

5. Перечень графического материала (с точным указанием обязательных чертежей и графиков):



7. Дата выдачи задания 16.03.2020г

8. Календарный график работы над проектом на весь период проектирования

	Этапы	Продолжительность
1.	Анализ аналогичных проблема, выявление проблематики. Выбор стека технологий.	2 месяца
2.	Основной этап разработки. Написание основных алгоритмов и формирование баз данных.	2 месяца
3.	Мануальное тестирование. Доработка функционала. Написание пояснительной записки.	1 месяц

Руководитель \_\_\_\_\_

Задание принял к исполнению (дата) 16.03.2020г

(подпись студента)

## РЕФЕРАТ

Здравствуйте. Тема моей текущей дипломной работы звучит как - Разработка веб-сервиса коммерческого производственного предприятия. В этот момент наверное всем сразу стало понятно о чем сейчас пойдет речь? Если нет, то ничего страшного, сейчас будем в этом разбираться.

Итак, представьте себе некое абстрактное предприятие, на котором происходит очень интересный социальный эксперимент. На этом предприятии был создан экспериментальный отдел по улучшению уровня жизни своих сотрудников. На минутку представим, что такое действительно бывает. Этот отдел, помимо всего прочего проводит курсы - по повышению личной финансовой грамотности. На этих курсах рассказываются основы ведения домашней бухгалтерии и составления ЛФП (личного финансового плана).

И вот на этом этапе у данного предприятия появляется потребность заказать реализацию некоего веб-приложения для своих сотрудников, с целью наглядной демонстрации и использования его в качестве домашней бухгалтерии, а также возможной дальнейшей манетизации данного продукта.

Слайд 1. Название работы QR код, ссылка на рабочее приложение.

Разработка веб-сервиса коммерческого производственного предприятия.

Пропускаем страницу с формами логирования и авторизации (покажу в программе).

Слайд 2. Главная страница. Основной счет, курсы валют на сегодня. Значение счета в пересчете на валюты.

Слайд 3. и 4. История. График по датам, таблица истории с возможностью частичной сотрировки, пагинацией и поиска.

Слайд 5. Детализация конкретной записи с возможностью радактирования и удаления.

Слайд 6. Планирование.

Слайд 7. Страница создании категорий. Редактирование категорий. Таблица категорий.

Слайд 8. Страница создания новой записи.

Слайд 9. Страница создания новой категории.

Спасибо за внимание, сейчас прошу задавать вопросы и можно перейти к детальному обсуждению структуры программы.

# ОГЛАВЛЕНИЕ

1	СПИСОК СОКРАЩЕНИЙ .....	6
2	ВВЕДЕНИЕ .....	7
3	АНАЛИЗ ОБЪЕКТА .....	10
3.1	Описание предметной области.....	10
3.2	Описание аналогов системы.....	13
	Нотемoney (MaxFin).....	13
	Дребеденьги .....	13
	Дзен Мани .....	14
4	ПРОЕКТИРОВАНИЕ .....	22
4.1	Разработка архитектуры программного продукта .....	22
4.2	Проектирование структур хранения данных.....	23
5	РЕАЛИЗАЦИЯ .....	26
5.1	Разработка информационной системы. ....	26
5.2	Структура проекта и файлов. ....	28
5.3	Разработка интерфейса программного продукта .....	31
6	Разработка алгоритмов реализации вариантов использования .....	39
7	Модульное и мануальное тестирование алгоритмов реализации вариантов использования. ....	46
	ЗАКЛЮЧЕНИЕ .....	48
	Список использованных источников .....	49
	Приложения .....	50

## СПИСОК СОКРАЩЕНИЙ

VUE - прогрессивный фреймворк для создания пользовательских интерфейсов

VUEX - «Хранилище» - контейнер, в котором хранится состояние приложения

HTML - HyperText Markup Language - «язык гипертекстовой разметки»

CSS - Cascading Style Sheets - каскадные таблицы стилей

SCSS - Sassy CSS

JS - JavaScript

TS - TypeScript

SPA - Single Page Application - Одностраничное приложение

MPA - Multi Page Application - Многостраничное приложение

PWA - progressive web app, - технология в web-разработке, которая визуально и функционально трансформирует сайт в приложение (мобильное приложение в браузере).

JSON - JavaScript Object Notation

NPM - Node Package Manager

CLI - Command Line Interface

VSC / VSCode - Visual Studio Code

URL - Uniform Resource Locator - унифицированный указатель ресурса

SVG - Scalable Vector Graphics - масштабируемая векторная графика

UI - User Interface - пользовательский интерфейс

Framework - остов, каркас, структура; программная платформа, определяющая структуру программной системы

PC, desktop - стационарный персональный компьютер, предназначенный для работы в офисе и дома

Open-source - открытое программное обеспечение

## **ВВЕДЕНИЕ**

В момент создания новой семьи у молодоженов происходит замена романтических отношений на хозяйственные. Идя на поднятых парусах по морю «Быта», супруги начинают сталкиваться с подводными течениями. И тогда им приходится склониться над картой, именуемой «семейный бюджет», в поисках верного решения. Но бюджет семьи – это не только контроллер затрат и доходов, но и индикатор климата.

### **Зачем нужен семейный бюджет**

Часто мужчины упрекают женщин в бездумной трате денег на одежду и косметику. Жены же постоянно «пилят» мужей за пиво, посиделки в баре, траты на охоту. Избежать этого поможет планирование семейного бюджета.

### **Другие причины ведения домашней бухгалтерии**

Учет доходов. Зная сумму ежемесячных поступлений, легко спланировать семейный бюджет на месяц. Так, оплата детских секций, коммунальных услуг, бензина носит постоянный характер.

Контроль расходов. Часто деньги утекают на незначительные траты, мелкие покупки, без которых можно обойтись. Убрав такую статью расходов, вы сэкономите семейный бюджет и избавитесь от ненужных покупок.

Накопление. Вычислив разницу между доходом и расходом, легко определить сумму, которую можно отложить, например, на отдых, путешествие или покупку недвижимости.

### **Варианты семейного бюджета**

Покой, мир и благополучие в семейных отношениях зависят от распределения финансов и ведения домашней бухгалтерии. Существуют разные виды учета семейных расходов.

## **Совместный**

Популярная и благодатная программа финансовых взаимоотношений. Супруги объединяют доходы и совместно прописывают семейный бюджет, исходя из полученной суммы денег. Такой подход сплачивает мужа и жену. Каждый вносит лепту в достижение победы над растратами. Также в этой программе учтены моменты, когда один из супругов не может выйти на игровое поле. Если жена посвящает себя уходу за детьми, или же, муж временно нетрудоспособен, тогда бюджет становится единоличным в добывании, но в пользовании остается общим. Такой курс навигации успешен при условии полного доверия и гармонии в паре.

## **Долевой (раздельно-совместный)**

Это распространённая система ведения семейного бюджета. Супруги рассчитывают сумму расходов на «коммуналку», питание и прочие хозяйственные нужды. Затем сумма полученных средств делится пополам, либо пропорционально зарплате каждого. В итоге у участников программы остаются личные деньги. Этот метод объединяет чувство общности и ощущение финансовой независимости. Исчезают обиды на вторую «половинку» из-за неразумных покупок, появляется возможность неожиданных подарков, так как теперь личные доходы не так обнажены.

Такое планирование семейного бюджета подходит в том случае, когда под одной крышей свили гнездо скупец и транжира. Пара боится себя от экономических провалов. Не беда, что отсутствует единство характеров, зато есть общие чувства.

## **Раздельный**

Такая программа больше близка семейным гнездам Запада, у которых имеется высокодоходный бизнес или накопленный капитал. В таких отношениях супруги говорят друг другу: «Смотри, мне от тебя нужна только любовь!» Однако полностью раздельным бюджет не получится. В случае совместного проживания, а, тем более, рождения наследников, раздельный семейный бюджет трансформируется в раздельно-совместный. Деньги лежат на личных банковских счетах, но еда в холодильнике, лошади на конюшне и вода в бассейне становятся общими. Зимой можно жить в элитной квартире одного, а летом — на чудесной вилле другого. Неразлучную пару не пугает дорогостоящее хобби супруга или крупные отчисления на благотворительность.



## **Цель и задачи разработки**

Понимая актуальность данной темы, некое коммерческое предприятие с большим количеством сотрудников решило проявить интерес к разработке и предоставлению веб-сервиса по ведению личной бухгалтерии для своего персонала.

# **1 АНАЛИЗ ОБЪЕКТА**

## **1.1 Описание предметной области**

В разделе даётся описание предметной области – сферы деятельности, для автоматизации которой разрабатывается программной системой ПС. Предметная область определяется темой дипломной работы.

Проблема управления личным бюджетом является актуальной для каждого человека. Мы постоянно совершаем различные покупки, берём деньги в долг, храним сбережения... И наш бюджет стремительно падает, если мы неэффективно тратим имеющиеся финансы. Грамотно распоряжающийся своим бюджетом человек постоянно следит за тем, сколько денег у него есть, рассчитывает, сколько денег ему нужно потратить, и принимает решения, где он может сэкономить и от чего он может вообще отказаться. Всё больше людей хотят контролировать свои финансовые поступления и траты. В связи с этим создание программы учёта семейного бюджета является актуальной темой для различных коммерческих предприятий. Запрос на данный продукт на рынке предоставляемых услуг с каждым годом растёт, поэтому разработка программы домашней бухгалтерии можно считать хорошим финансовым вложением, для различных предприятий, желающих получать доход без значительных денежных и трудовых вложений в процессе реализации своего продукта.

## **1.2 Построение концептуальной модели предметной области**

Результаты исследования предметной области должны быть представлены в виде структурной схемы (например, диаграммы классов, выполненной средствами UML). Представленная схема должна демонстрировать все сущности предметной области, их атрибуты, функции и связи.

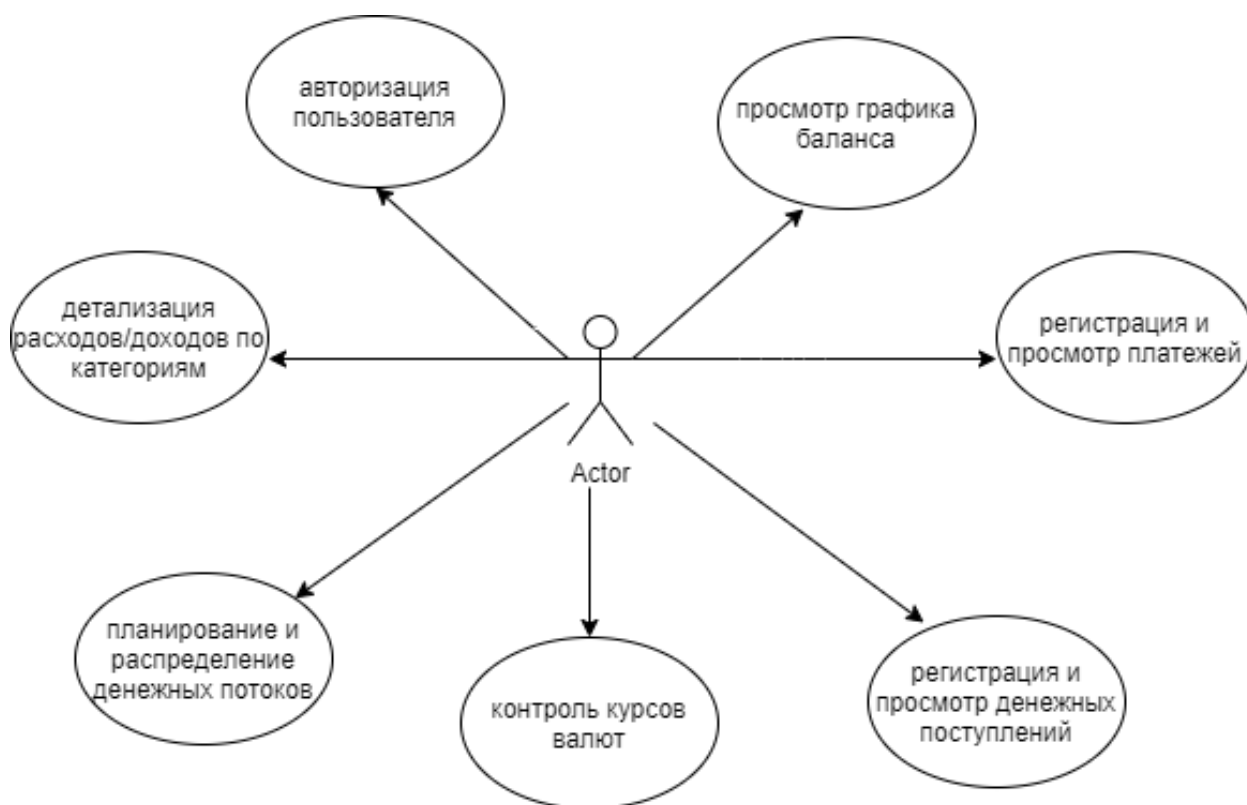


Рисунок 1.1 – Диаграмма возможностей пользователя

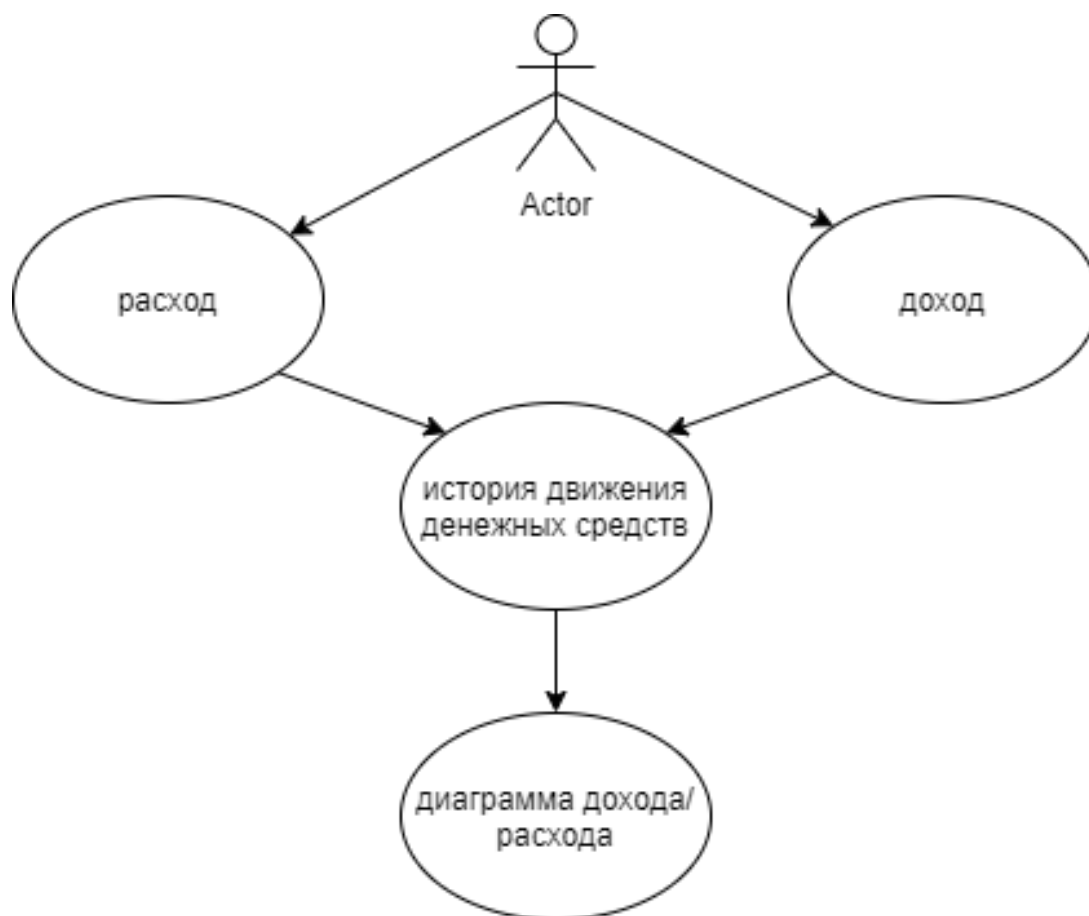


Рисунок 1.2 – Диаграмма учета финансов

## **2 ПОСТАНОВКА ЗАДАЧИ**

### **2.1 Определение требований к программной системе (ПС).**

В данном разделе формулируются основные цели разработки, пользовательские требования. Для ПС устанавливается набор выполняемых функций, характеристики их исходных данных, а также набор результатов, их характеристики и способы представления. Также выявляются и обосновываются системные требования к ПС.

Описание функций:

Регистрация пользователей - добавление новых пользователей в систему. При регистрации пользователь вводит свой e-mail и пароль, после чего ему на почту присылается письмо с подтверждением регистрации. Перейдя по ссылке в письме, он активирует свой аккаунт и получает возможность войти в систему.

Авторизация пользователей - проверка истинности введенных логина и пароля пользователя.

Список расходов и поступлений - пользователь может сохранить в системе запись о финансовой операции, указав дату, сумму, категорию, комментарий. На странице списка расходов и поступлений отображаются все оставленные пользователем записи, он может их редактировать и удалять.

История расходов и поступлений – в программном окне отображается действие, которое было совершено. Пользователь может просмотреть эти действия. По выбранному дню отображается все движение денежных средств.

График сравнения дней по сумме финансовых операций - рассчитывается сумма всех поступлений / затрат по каждому дню и отображается на графике.

Список валют - пользователь может запросить список интересующих его валют на текущую дату, для удобства пользователя есть возможность просмотра своего денежного состояния в различных денежных знаках.

Интерактивные графики - пользователь может выбрать день, щелкнув по точке на графике счета или сравнения дней, и перейти на страницу дневника, чтобы просмотреть, какие операции были совершены в выбранный день. Также, под графиком отображаются дни, в которых баланс был максимальным / минимальным, и кнопки просмотра этих дней на странице дневника.

## **2.2 Описание аналогов системы**

В разделе приводиться краткий обзор существующих ИС, применяемых для решения поставленной, или схожей задачи с указанием средств и методов проектирования, с помощью которых были реализованы данные аналоги (если эта информация доступна).

В настоящее время на рынке услуг существуют различные аналоги домашней бухгалтерии, которые разработаны фирмами различных стран. Наиболее популярными среди них являются

### **2.1.1 Homemoney (MaxFin)**

HomeMoney - первый в русскоязычном интернете онлайн сервис для ведения домашней бухгалтерии, у которого более 200 000 пользователей.

На сайте доступны все необходимые для учета семейных доходов/расходов инструменты: настраиваемые статьи, гибко формирующиеся отчеты, средства анализа, планирование бюджета, валютный калькулятор и многое другое. Кроме того, у сайта есть мобильная версия.

Мобильные приложения доступны для iOS, Android и iPhone и позволяют работать с системой без подключения в интернет.

### **2.1.2 Дребеденьги**

Проект с интересным названием разрабатывается (предположительно) с 2007 года. Основной по функционалу является браузерная версия. В офлайн версии для Windows отсутствуют некоторые важные возможности. Например, нет возможности завести новый счет. Интерфейс офлайн версии выглядит более примитивно.

Система позволяет вести контроль расходов и доходов, проводить перевод средств из одного места в другое - система будет учитывать, когда вы будете совершать операции по вводу-выводу наличных с банковских счетов, обмен наличных денег на виртуальные и так далее, в общем любые операции на ваш вкус. Ко всему прочему добавлена возможность учитывать комиссию, которая периодически встречается при переводах. Имеется конвертация валют - по

умолчанию там уже установлен текущий курс ЦБ РФ, но если вы разменивали деньги по другому курсу - это будет принято во внимание и рассчитано по соответствующему курсу. Есть функция планирование бюджета — поможет оценить, исходя из уровня ваших доходов, текущее потребление, придерживаясь которого вы сможете накопить на заветную цель. Покупки там есть разных масштабов, но сервис. **Дребеденьги** подскажет вам, сколько вам нужно потратить сегодня, чтобы завтра вы смогли купить то, о чем так давно мечтали. Аналитические отчеты, диаграммы и графики — очень красивые и презентабельные — то, что поможет вам сразу оценить, на что идут ваши деньги.

### **2.1.3 Дзен Мани**

Проект существует с 2010 года. Несмотря на видимую простоту, WEB приложение умеет загружать транзакции некоторых российских банков (Альфа Банк, Авангард, ВТБ24 и др.). Для этого необходимо подключаться к интернетбанку и загружать данные оттуда. Автоматически, как это принято в западных банках, пока ничего не происходит. Есть также возможность считывать данные о платежах из смс и email. Поддерживается формат СМС более 100 банков.

## **2.3 Обзор и обоснование выбора инструментальных средств**

В разработке использовался следующий стек технологий:

- VUE
- VUEX
- VUETIFY
- TypeScript

VUE - JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле. (2014)

В качестве основного средства разработки для проекта использован фреймворк VUE. VUE имеет низкий порог вхождения и высокую скорость разработки. VUE CLI упрощает работу VUE.

Одно из главных преимуществ CLI заключается в создании нового проекта, а также создании различных сервисов, компонентов и других нужд проекта. CLI VUE автоматически устанавливает собственный набор стандартов для проекта, это позволяет избежать ряда ошибок.

**VUE** может использовать **TypeScript** и это огромное преимущество. TypeScript дает вам отличную возможность использовать систему типов в вашем JavaScript коде.

Многие фреймворки, включая VUE, имеют большое количество библиотек компонентов.

## **VUEX**

В центре любого VUEX-приложения находится хранилище. «Хранилище» — это контейнер, в котором хранится состояние вашего приложения. Два момента отличают хранилище VUEX от простого глобального объекта:

Хранилище VUEX реактивно. Когда компоненты VUE полагаются на его состояние, то они будут реактивно и эффективно обновляться, если состояние хранилища изменяется.

Нельзя напрямую изменять состояние хранилища. Единственный способ внести изменения — явно вызвать мутацию. Это гарантирует, что любое изменение состояния оставляет след и позволяет использовать инструментарий, чтобы лучше понимать ход работы приложения.

## **VUETIFY**

VUETIFY является библиотекой #1 для Vue.js и активно разрабатывается с 2016 года. Цель проекта - предоставить пользователям все, что необходимо для создания богатых и интересных веб-приложений, используя спецификацию Material Design. Это достигается благодаря постоянному циклу обновлений,

долгосрочной поддержке предыдущих версий, отзывчивому участию сообщества, обширной экосистеме ресурсов и приверженности качественным компонентам.

VUETIFY разрабатывается точно в соответствии с спецификацией Material Design. Каждый компонент создан вручную, чтобы предоставить вам самые лучшие UI инструменты для вашего следующего уникального приложения. Разработка не ограничивается на основных компонентах, указанных в спецификации Google. Благодаря поддержке членов сообщества и наших спонсоров, мы добавили более 15 новых компонентов, включая календари, средства выбора цвета, древовидные списки и многое другое.

## HTML

HTML (HyperText Markup Language — «язык гипертекстовой разметки») — самый базовый строительный блок Веба. Он определяет содержание и структуру веб-контента. Другие технологии, помимо HTML, обычно используются для описания внешнего вида/представления (CSS) или функциональности/поведения (JavaScript) веб-страницы.

Под гипертекстом ("hypertext") понимаются ссылки, которые соединяют веб-страницы друг с другом либо в пределах одного веб-сайта, либо между веб-сайтами. Ссылки являются фундаментальным аспектом Веба. Загружая контент в Интернет и связывая его со страницами, созданными другими людьми, вы становитесь активным участником Всемирной паутины.

## CSS, SCSS

CSS (Cascading Style Sheets) — язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, документам HTML и приложениям XML). Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL. Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.



CSS поддерживает таблицы стилей для конкретных носителей, поэтому авторы могут адаптировать представление своих документов к различным визуальным браузерам и устройствам.

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Для каждого элемента можно использовать ограниченный набор свойств, остальные свойства не будут оказывать на него никакого влияния.

SCSS (Sassy CSS) является расширением синтаксиса CSS. Это означает, что любое допустимое значение в CSS стилях будет допустимо и в SCSS. Кроме того, SCSS понимает синтаксис вендорных префиксов.

### **JavaScript (JS), TypeScript (TS)**

JavaScript - мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript (стандарт ECMA-262).

Изначально JavaScript был создан, чтобы «сделать веб-страницы живыми». Программы на этом языке называются скриптами. Они могут встраиваться в HTML и выполняться автоматически при загрузке веб-страницы. Скрипты распространяются и выполняются, как простой текст. Им не нужна специальная подготовка или компиляция для запуска.

### **Это отличает JavaScript от другого языка – Java.**

Почему JavaScript? Когда JavaScript создавался, у него было другое имя – «LiveScript». Однако, язык Java был очень популярен в то время, и было решено, что позиционирование JavaScript как «младшего брата» Java будет полезно.

Со временем JavaScript стал полностью независимым языком со своей собственной спецификацией, называющейся ECMAScript, и сейчас не имеет никакого отношения к Java.

Сегодня JavaScript может выполняться не только в браузере, но и на сервере или на любом другом устройстве, которое имеет специальную программу, называющуюся «движком» JavaScript. У браузера есть собственный движок, который иногда называют «виртуальная машина JavaScript».

TypeScript - язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript.

TypeScript является обратно совместимым с JavaScript и компилируется в последний. Фактически, после компиляции программу на TypeScript можно выполнять в любом современном браузере или использовать совместно с серверной платформой Node.js.

TypeScript отличается от JavaScript возможностью явного статического назначения типов, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использование кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, ускорить выполнение программ.

## **VUE, VUE CLI**

VUE представляет фреймворк от компании Google для создания клиентских приложений. Прежде всего он нацелен на разработку SPA-решений (Single Page Application), то есть одностраничных приложений. В этом плане VUE является наследником другого фреймворка VUE. В то же время VUE это не новая версия VUE, а принципиально новый фреймворк.

VUE предоставляет такую функциональность, как двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом, шаблоны, маршрутизация и так далее.

Одной из ключевых особенностей VUE является то, что он использует в качестве языка программирования TypeScript.

VUE CLI - официальный инструмент для инициализации и работы с VUE - проектами. Избавляет от настройки сложных конфигураций и инструментов сборки, таких как TypeScript, Webpack и так далее.

После установки VUE CLI нужно выполнить одну команду для генерации проекта, а другую - для его обслуживания с использованием локального сервера разработки для работы с вашим приложением.

Как и большинство современных инструментов веб-интерфейса, VUE CLI построен на основе Node.js.

**Node.js** - среда выполнения **JavaScript**, а также серверная технология, которая позволяет запускать JavaScript на сервере и создавать веб-приложения на стороне сервера. Тем не менее, VUE - это интерфейсная технология, которая требует предварительной установки Node.js на компьютер для разработки, он предназначен только для запуска CLI.

### **Git, GitHub, gh-pages**

Git - распределённая система контроля версий, которая даёт возможность разработчикам отслеживать изменения в файлах и работать совместно с другими разработчиками. Она была разработана в 2005 году Линусом Торвальдсом, создателем Linux, для того, чтобы другие разработчики могли вносить свой вклад в ядро Linux. Git известен своей скоростью, простым дизайном, поддержкой нелинейной разработки, полной децентрализацией и возможностью эффективно работать с большими проектами.

Подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.

Git бесплатная и open-source система. Это значит, что его можно бесплатно скачать и вносить любые изменения в исходный код. Небольшой и быстрый. Он выполняет все операции локально, что увеличивает его скорость. Кроме того, Git локально сохраняет весь репозиторий в небольшой файл без потери качества данных.

Резервное копирование. Git эффективен в хранении бэкапов, поэтому известно мало случаев, когда кто-то терял данные при использовании Git. Простое ветвление. В Git управление ветками реализовано гораздо проще и эффективнее.

GitHub - сервис онлайн-хостинга репозиториях, обладающий всеми функциями распределенного контроля версий и функциональностью управления исходным кодом - всё, что поддерживает Git и даже больше. Обычно он используется вместе с Git и даёт разработчикам возможность сохранять их код онлайн, а затем взаимодействовать с другими разработчиками в разных проектах.

Также GitHub может похвастаться контролем доступа, багтрекингом, управлением задачами и вики для каждого проекта. Цель GitHub содействовать взаимодействию разработчиков.

К проекту, загруженному на GitHub, можно получить доступ с помощью интерфейса командной строки Git и Git-команд. Также есть и другие функции, такие как документация, запросы на принятие изменений (pull requests), история коммитов, интеграция со множеством популярных сервисов, email-уведомления, эмодзи, графики, вложенные списки задач и т.д.

Git - это инструмент, позволяющий реализовать распределенную систему контроля версий, а GitHub - это сервис для проектов, использующих Git.

GitHub Pages - служба хостинга сайтов, которая позволяет вести персональные сайты, сайты организаций и сайты отдельных проектов GitHub. Публичные страницы пользователей, организаций и репозиториях, с бесплатным хостингом от GitHub с доменом github.io или с кастомным доменом.

### **Обоснование выбора среды разработки**

VUE фреймворк - является наиболее зрелым из Frontend фреймворков, имеет хорошую поддержку с точки зрения участников и представляет собой полный пакет. Тем не менее, кривая обучения является довольно крутой, и концепции развития в VUE могут оттолкнуть новых разработчиков. VUE - хороший выбор для компаний с большими командами и разработчиков, которые уже используют TypeScript.

### **Редактор кода Visual Studio Code**

Visual Studio Code – кроссплатформенный редактор кода с поддержкой более 30 языков программирования и форматов файлов, а также обладающий рядом дополнительных, полезных возможностей. Быстрый и бесплатный редактор со множеством плагинов.

Visual Studio Code поддерживает локальное и удаленное Git хранилища. В контексте Visual Studio Code можно выполнить любую команду командной строки и просмотреть результаты работы прямо из среды разработки. Таким образом можно использовать внешние компиляторы, отладчики, средства тестирования и т.д.

### **3 ПРОЕКТИРОВАНИЕ**

#### **3.1 Разработка архитектуры программного продукта**

В данном разделе на основе пользовательских требований и аналогов системы выбирается и описывается архитектура программного продукта.

На основе проведенных маркетинговых исследований можно сделать вывод, что программа для ведения домашней бухгалтерии должна отвечать таким пользовательским требованиям как:

- регистрация пользователей для удобства ведения бухгалтерии;
  - список расходов и поступлений для учета финансовых операций с указанием даты, суммы, отнесением в необходимую категорию и возможностью оставить поясняющий комментарий;
  - история расходов и поступлений для детального контроля совершенных денежных операций;
  - график изменения счета для наглядного отображения движения денежных средств;
  - интерактивные графики для удобства просмотра интересующего момента на графике движения денежных средств;
- график сравнения дней по сумме финансовых операций  
- график сравнения категорий по сумме финансовых операций  
- список валют  
- интерактивные графики для удобства просмотра интересующего момента на графике движения денежных средств.

Программа должна быть рассчитана на непрофессионального пользователя. Система позволит вести учёт совершённых финансовых операций и планирование будущих. Система должна быть эффективной, в плане обработки данных, выполнения программных функций; должна быть устойчивой, т.е. выполнять все функции, которые будут разработаны, независимо от внешних факторов. Должны быть созданы диалоговые окна для интерактивного режима работы с пользователем. Выполнение требований эргономичности интерфейса, создание комфортных условий работы.

Программа должна работать с текстовыми и числовыми данными, в соответствии с обработкой и их использованием, выдавать сообщения об ошибках при неверно заданных исходных данных (валидация форм), поддерживать диалоговый режим в рамках предоставленных пользовательских возможностей.

## 3.2 Проектирование структур хранения данных

Что такое реляционные и нереляционные базы данных

Реляционная база данных (SQL) — база, где данные хранятся в формате таблиц, они строго структурированы и связаны друг с другом. В таблице есть строки и столбцы, каждая строка представляет отдельную запись, а столбец — поле с назначенным ей типом данных. В каждой ячейке информация записана по шаблону.

Нереляционная база данных (NoSQL) — хранит данные без четких связей друг с другом и четкой структуры. Вместо структурированных таблиц внутри базы находится множество разнородных документов, в том числе изображения, видео и даже публикации в социальных сетях. В отличие от реляционных БД, NoSQL базы данных не поддерживают запросы SQL.

В отличие от реляционных, в нереляционных базах данных схема данных является динамической и может меняться в любой момент времени. К данным сложнее получить доступ, то есть найти внутри базы что-то нужное — с таблицей это просто, достаточно знать координаты ячейки. Зато такие СУБД отличаются производительностью и скоростью. Физические объекты в NoSQL обычно можно хранить прямо в том виде, в котором с ними потом работает приложение.

Базы данных NoSQL подходят для хранения больших объемов неструктурированной информации, а также хороши для быстрой разработки и тестирования гипотез.

В них можно хранить данные любого типа и добавлять новые в процессе работы.

Масштабируемость. NoSQL базы имеют распределенную архитектуру, поэтому хорошо масштабируются горизонтально и отличаются высокой производительностью. Технологии NoSQL могут автоматически распределять данные по разным серверам. Это повышает скорость чтения данных в распределенной среде.

База данных **Firestore** Realtime - это база данных, размещенная в облаке. Данные хранятся в формате JSON и синхронизируются в реальном времени с каждым подключенным клиентом. Когда вы создаете кроссплатформенные приложения с помощью наших SDK для iOS, Android и JavaScript, все ваши

клиенты используют один экземпляр базы данных в реальном времени и автоматически получают обновления с новейшими данными.

В данном проекте мы будем пользоваться именно базой данных **Firestore**.

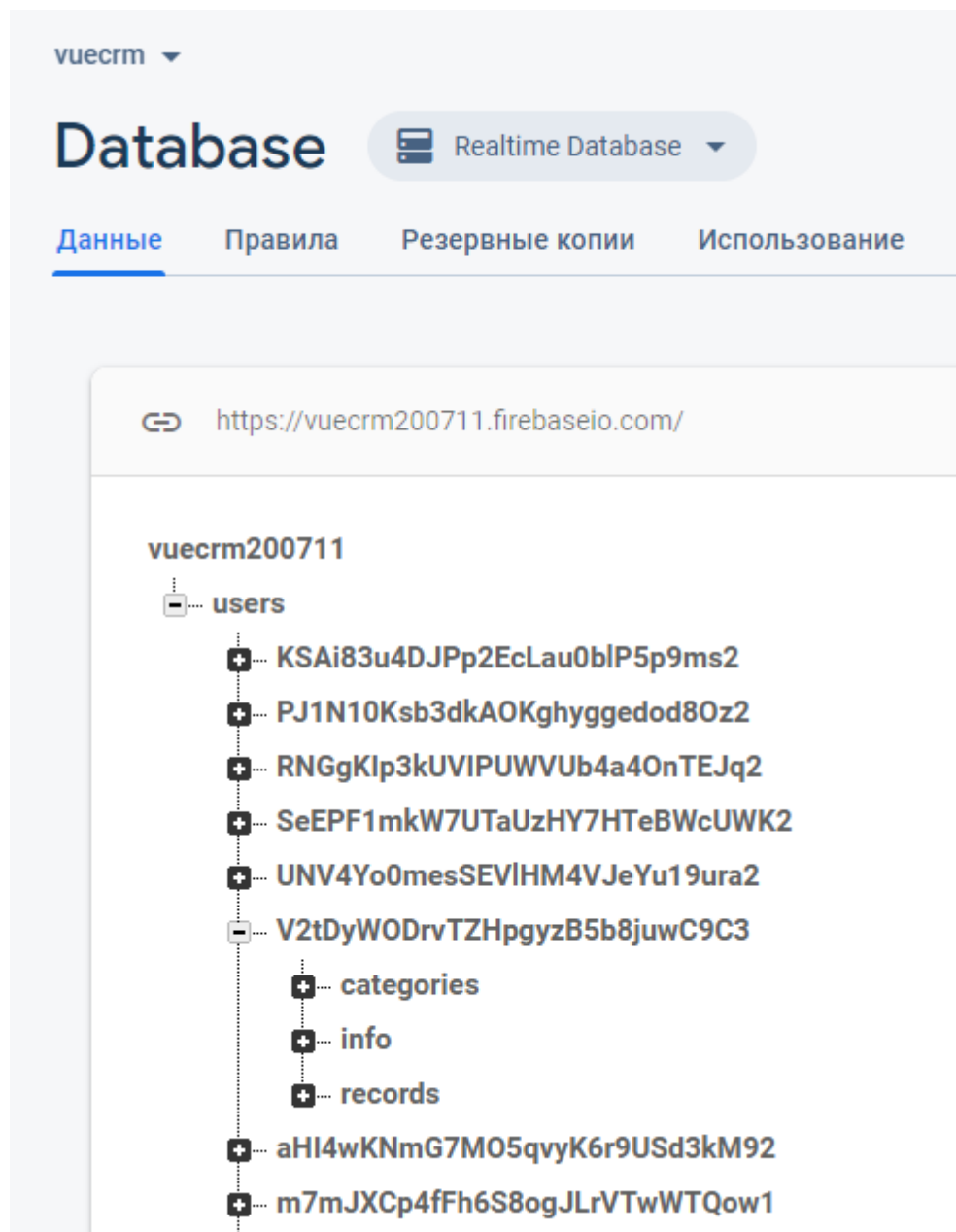


Рисунок 3.1 – Структура базы данных



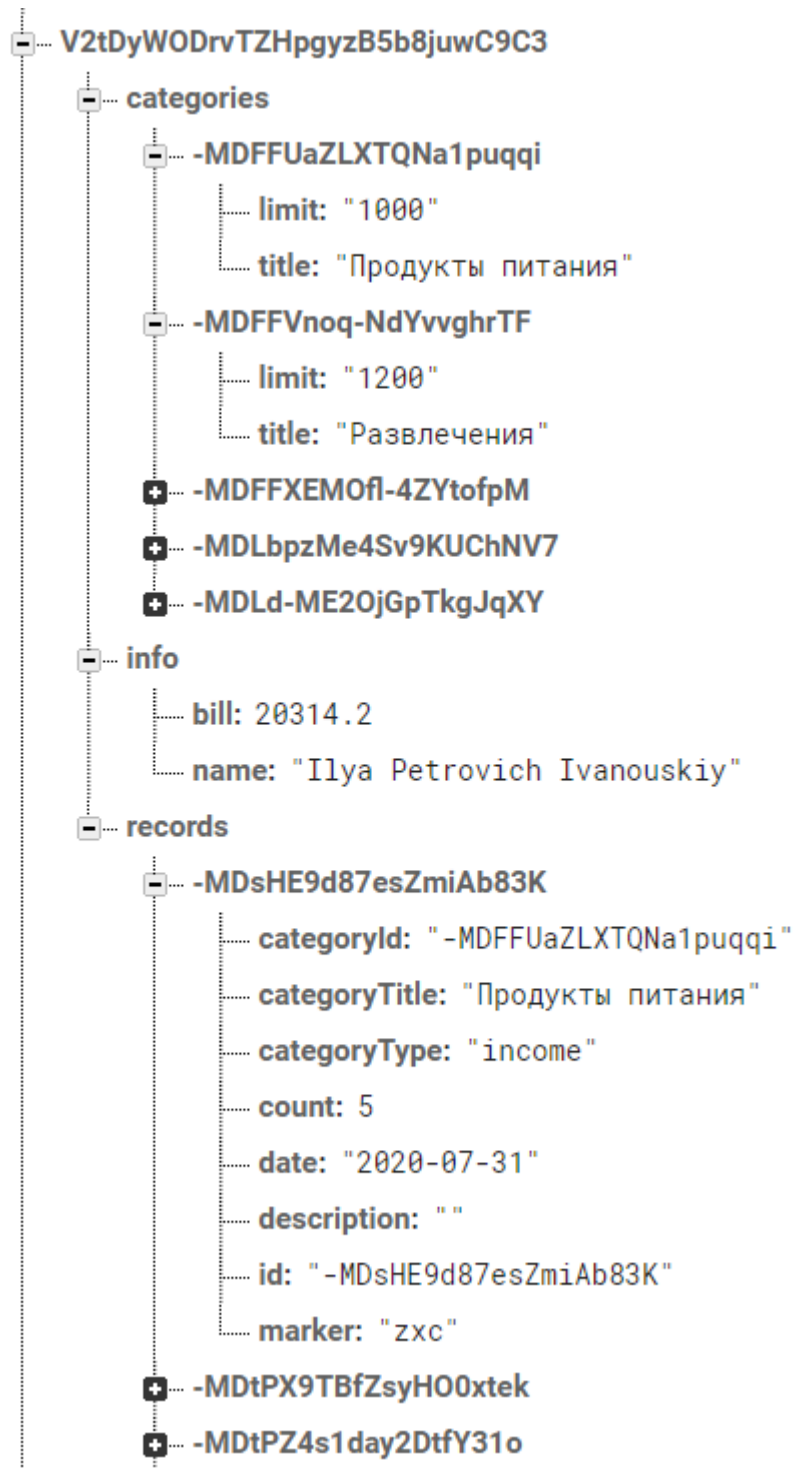


Рисунок 3.2 – Структура базы данных

## 4 РЕАЛИЗАЦИЯ

### 4.1 Разработка информационной системы.

При разработке SPA/PWA web-приложения с использованием не реляционных DB не всегда требуется использовать ООП подход. В случае данного приложения преимущественно используется модульный подход. Например, для управление состоянием компонентов приложения используется VUEX с разбиением на модули.

```
Ilya Ivanouski, 8 days ago | 1 author (Ilya Ivanouski)
1  import Vue from 'vue';
2  import Vuex from 'vuex';
3
4  import navigationDrawer from './modules/navigationDrawer';
5  import snackbar from './modules/snackbar';
6  import auth from './modules/auth';
7  import errorNotification from './modules/errorNotification';
8  import info from './modules/info';
9  import currency from './modules/currency';
10 import category from './modules/category';
11 import record from './modules/record';
12 import plannings from './modules/plannings';
13 import history from './modules/history';
14 import window from './modules/window';
15
16 Vue.use(Vuex);
17
18 export default new Vuex.Store({
19   modules: {
20     navigationDrawer,
21     snackbar,
22     auth,
23     errorNotification,
24     info,
25     currency,
26     category,
27     record,
28     plannings,
29     history,
30     window,
31   },
32 });
```

## Пример управлением навигационного меню через VUEX state.

Ilya Ivanouski, 17 days ago | 1 author (Ilya Ivanouski)

```
1  const navigationDrawer = {
2    state: {
3      drawer: null,
4    },
5
6    mutations: {
7      navigationDrawerMutation: (state: any, drawer: boolean) => {
8        state.drawer = drawer;
9      },
10   },
11
12   getters: {
13     navigationDrawerGetter: (state: any) => state.drawer,
14   },
15 };
16
17 export default navigationDrawer;
```

## 4.2 Структура проекта и файлов.

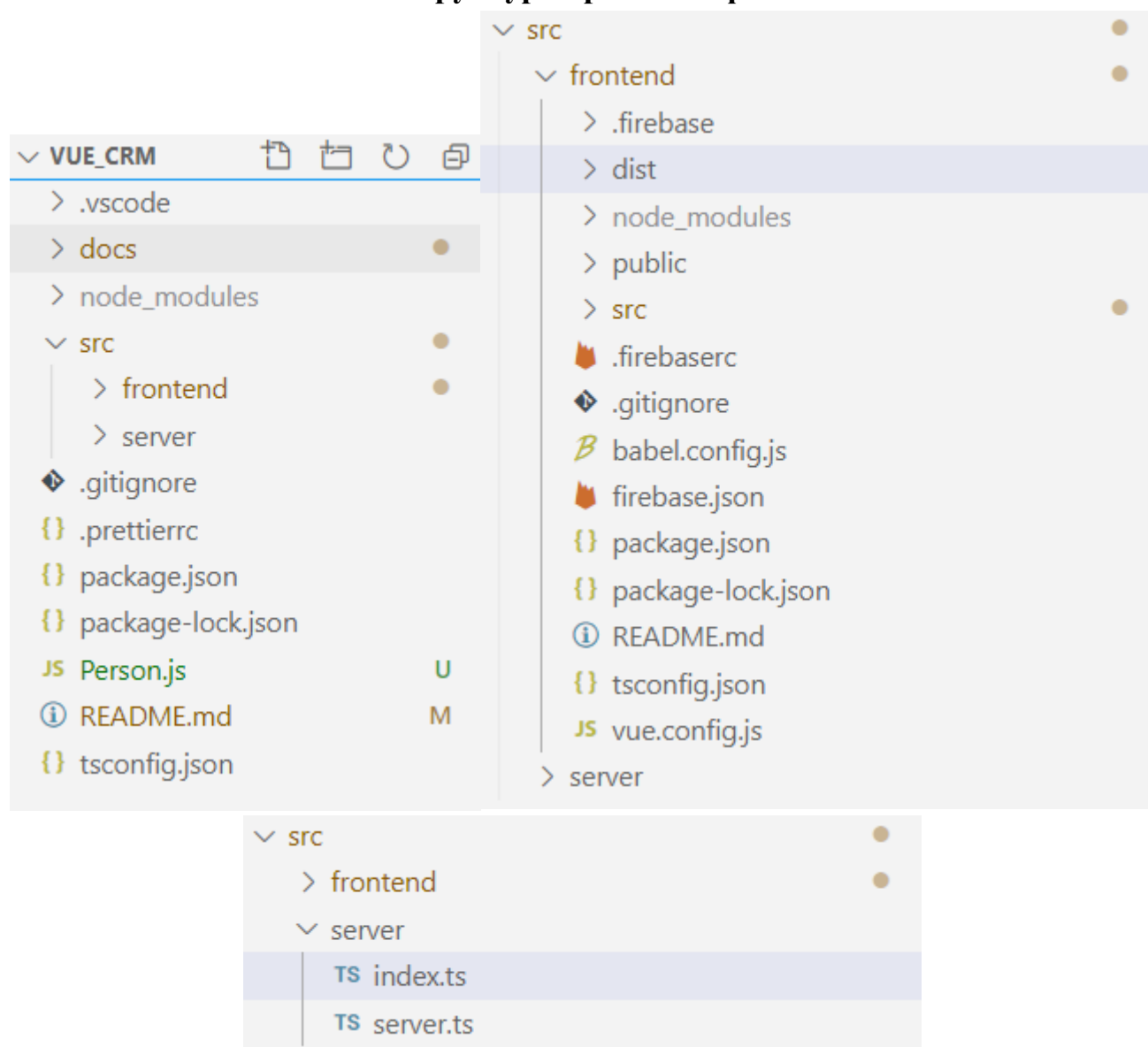


Рисунок 4.1 – Структура файлов проекта

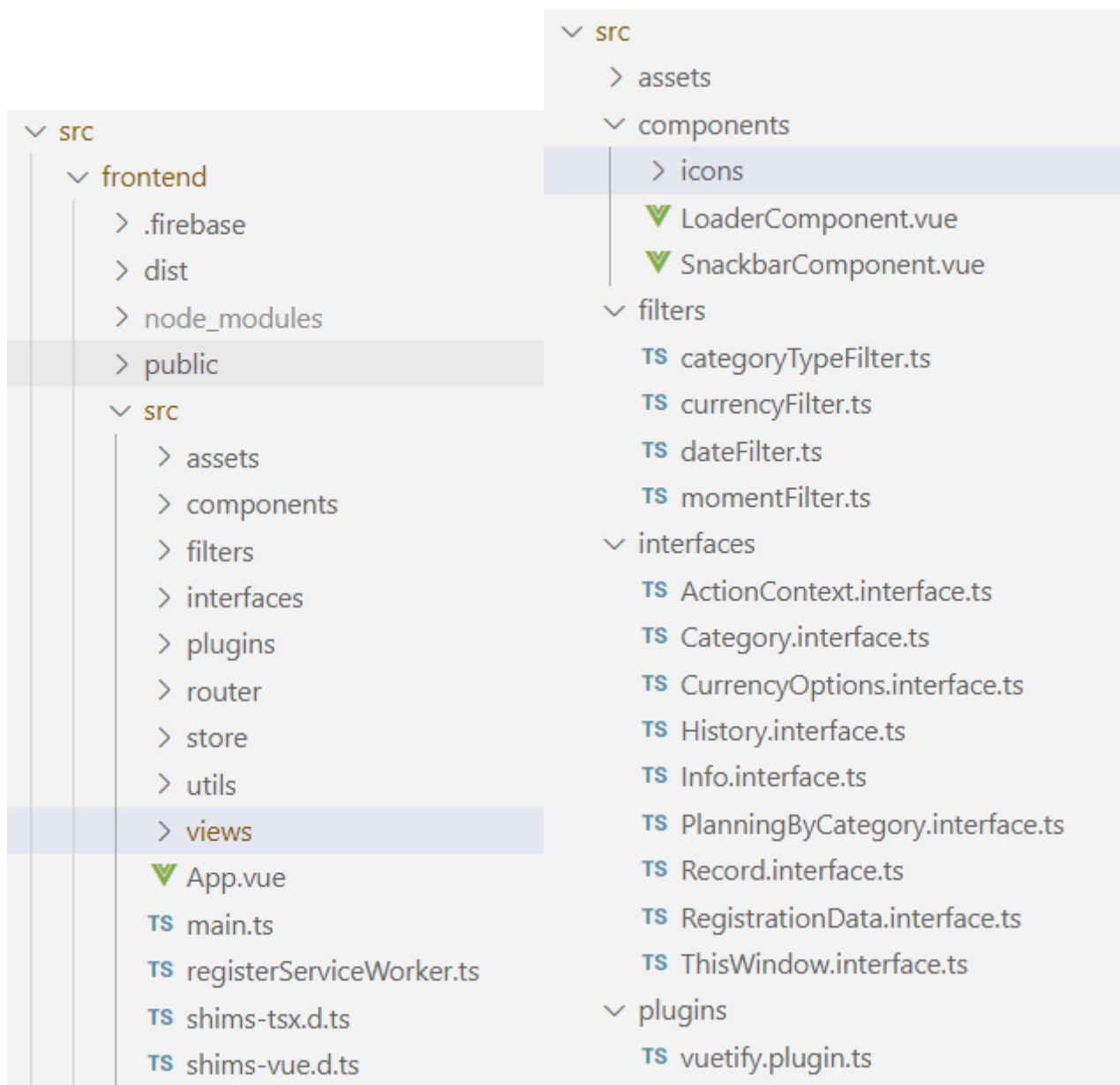


Рисунок 4.2 – Структура файлов проекта

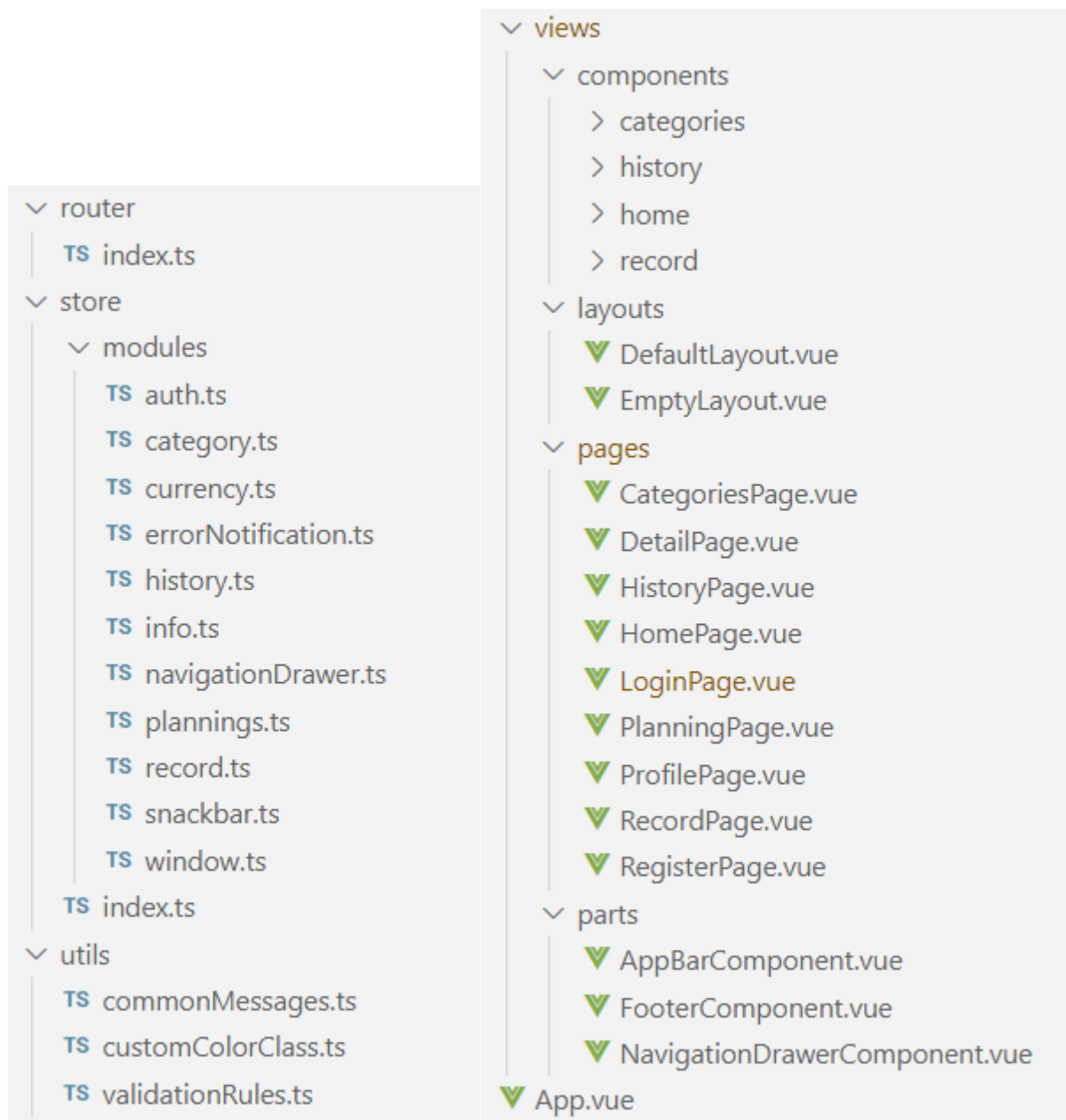


Рисунок 4.3 – Структура файлов проекта

### 4.3 Разработка интерфейса программного продукта

В данном разделе приведены основные интерфейсы приложения. Все интерфейсы реализованы на интуитивно понятном уровне, поэтому в разделе не приводится их детального описания.

The image shows two mobile application screens side-by-side. The left screen is titled 'Регистрация' (Registration) and features a dark header with a close button 'x'. It contains input fields for 'E-mail' (demo@mail.com), 'Password' (masked with dots), and 'Имя пользователя' (demo). A checkbox is checked with the text 'Вы согласны с правилами?'. At the bottom, there is a large 'РЕГИСТРАЦИЯ' button and two links: 'Есть аккаунт?' and 'ВОЙТИ'. The right screen is titled 'Shadow accounting' and has a similar layout but with a 'ВОЙТИ' button and links 'Нет аккаунта?' and 'ЗАРЕГИСТРИРОВАТЬСЯ'. Both screens have a light blue highlight on the password input field.

Рисунок 4.4 – Формы регистрация и вход

The image displays a mobile application interface. On the left, there is a vertical sidebar menu with five items: 'Счет' (Account) with a wallet icon, 'История' (History) with a grid icon, 'Планирование' (Planning) with a line graph icon, 'Новая запись' (New record) with a plus and grid icon, and 'Категории' (Categories) with a pie chart icon. On the right, there is a dark top status bar containing a hamburger menu icon, the date and time '13.08.20, 10:42:43', and a user profile icon.

Рисунок 4.5 – Меню левое и верхнее

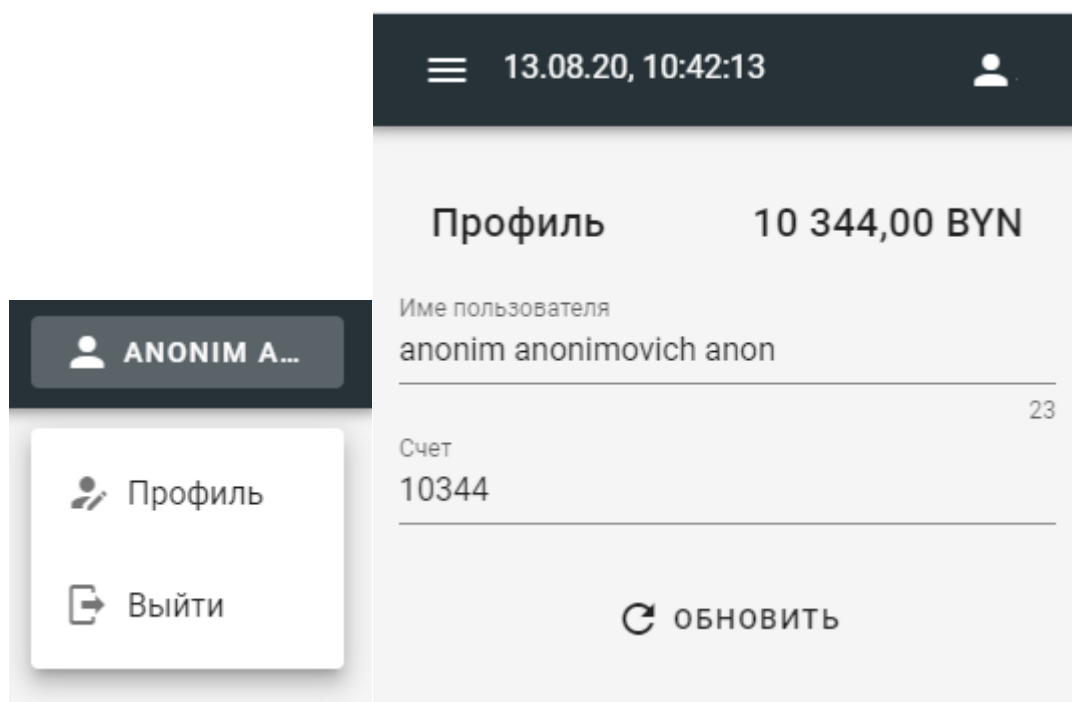
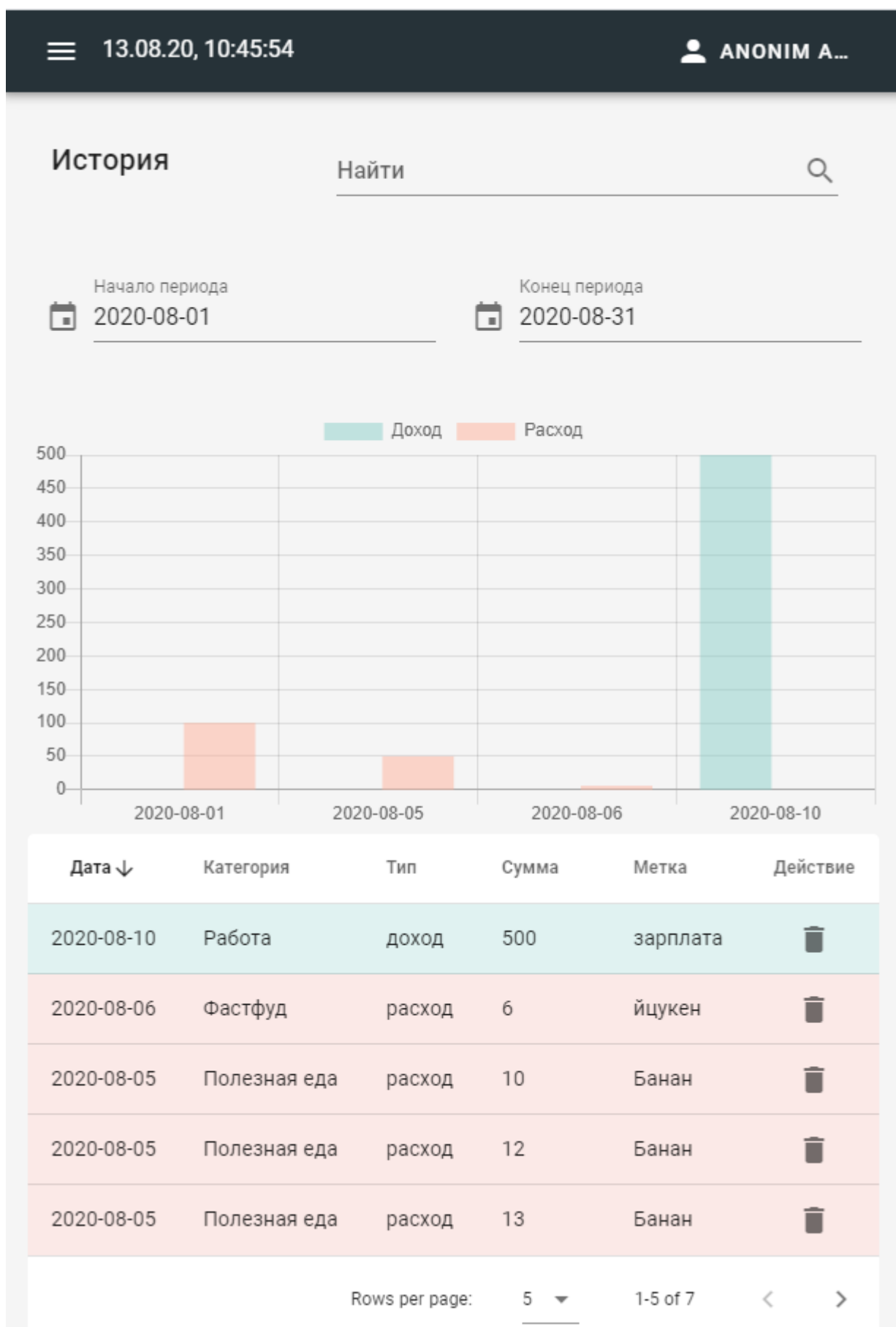


Рисунок 4.6 – Меню профиля и страница профиля





13.08.20, 10:49:11

Работа - Доход

Категория

Работа

Тип операции

Доход

Дата

2020-08-10

Счет

500

Метка

зарплата

Описание

УДАЛИТЬ

ИСТОРИЯ

СОХРАНИТЬ

Рисунок 4.8 – Страница детализация записи счета

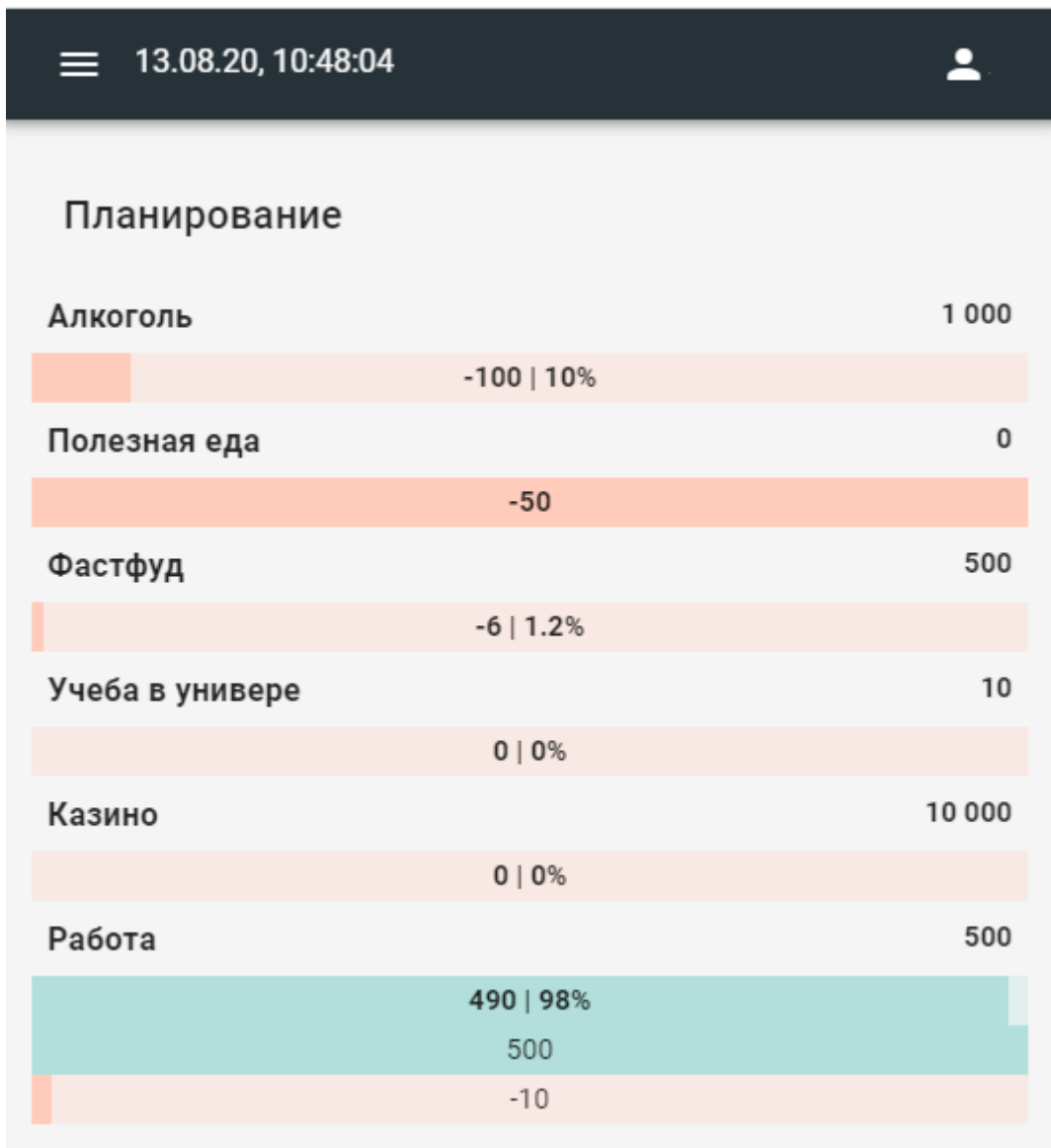





Рисунок 4.9 – Страница планирование

 13.08.20, 10:54:27 


## Новая запись

Выберите категорию

Алкоголь 

---


Начало периода

 2020-08-13

---

☒ Расход ☐ Доход

Счет / Общая сумма

0 

---

Метка

---

Описание

---




 СОЗДАТЬ

Рисунок 4.10 – Страница новой записи

 13.08.20, 10:59:07


## Создать категорию

Название

---

Лимит (план на месяц)

---

 СОЗДАТЬ

## Редактировать категорию

Выберите категорию

Алкоголь

---

Название

Алкоголь

---

Лимит

1000

---






 ОБНОВИТЬ  УДАЛИТЬ

Рисунок 4.11 – Страница создания новой категории

 **ОБНОВИТЬ**

 **УДАЛИТЬ**


### Список всех категорий

Sort by 

Наименование	Алкоголь
Лимит	1000
Наименование	Полезная еда
Лимит	0
Наименование	Фастфуд
Лимит	500
Наименование	Учеба в универе
Лимит	10
Наименование	Казино
Лимит	10000

Rows per page:

5



1-5 of 6



 

Рисунок 4.12 – Страница создания новой категории, таблица

## 5 РАЗРАБОТКА АЛГОРИТМОВ РЕАЛИЗАЦИИ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

Вся разработка проекта велась в данном репозитории [https://github.com/Domovikx/vue\\_crm](https://github.com/Domovikx/vue_crm).

В связи с содержанием большого объема кода полученного при разработке приложения не получается охватить каждый отдельно взятый алгоритм, но есть возможность схематически упрощенно представить взаимосвязи и структуру всех алгоритмов.

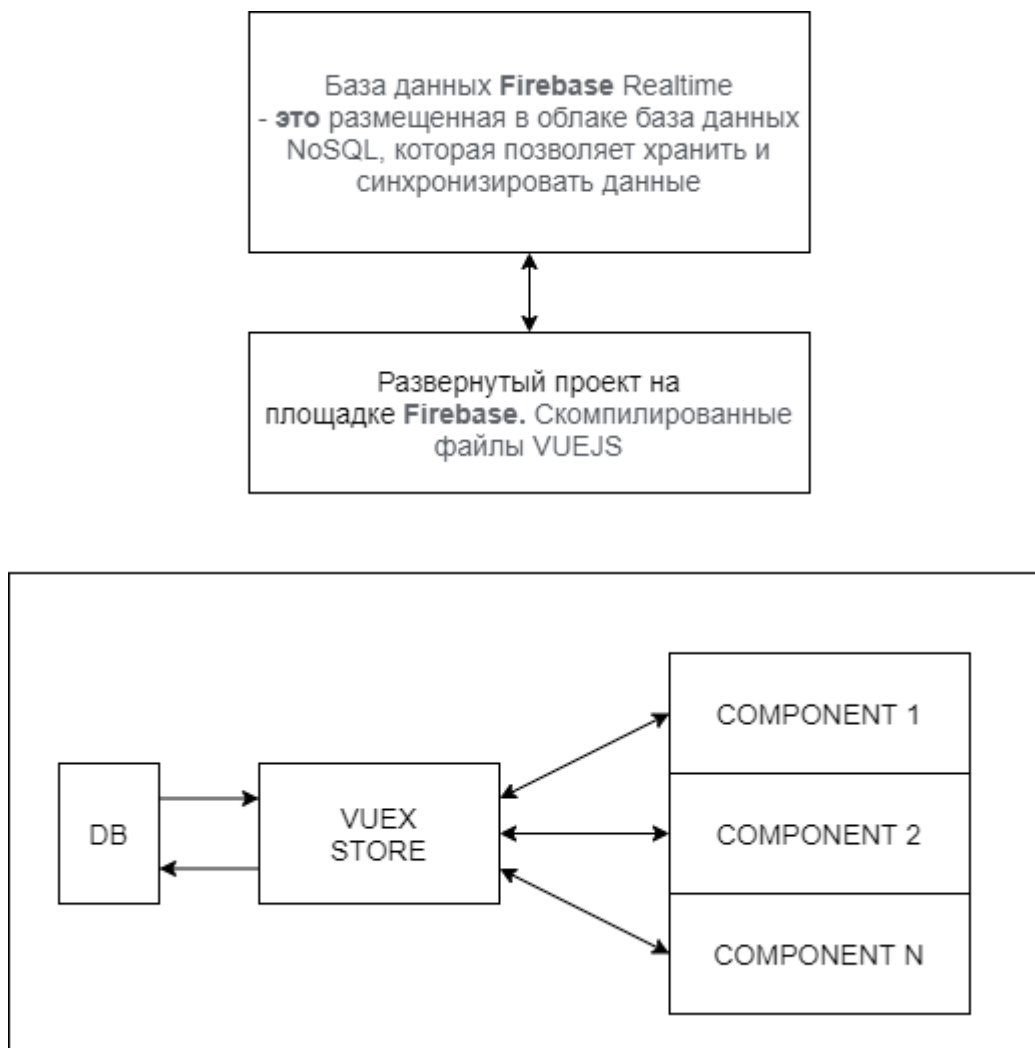


Рисунок 5.1 – Схема взаимодействия с БД и управление приложением

В качестве примера можно рассмотреть вариант кода для получения курсов валют и выведения его на экран.

```

1  /*
2
3  https://www.nbrb.by/api/exrates/rates/rub?parammode=2
4  {
5    Cur_ID: 298,
6    Date: "2020-07-15T00:00:00",
7    Cur_Abbreviation: "RUB",
8    Cur_Scale: 100,
9    Cur_Name: "Российских рублей",
10   Cur_OfficialRate: 3.4012
11 }
12
13 */
14
15 import ActionContext from '@/interfaces/ActionContext.interface';
16
17 const currency = {
18   state: {
19     currencyBase: 'BYN',
20     currencySymbols: ['USD', 'RUB', 'EUR'],
21     currencies: {},
22   },
23
24   actions: {
25     async currencyFetchAction({ getters, commit }: ActionContext) {
26       const symbols: any[] = getters.currencySymbolsGetter;
27       const currencies: any = {};
28
29       for (const s of symbols) {
30         const URL = `https://www.nbrb.by/api/exrates/rates/${s}?parammode=2`;
31         const response = await fetch(URL);
32         const json = await response.json();
33
34         currencies[s] = {
35           rate: json.Cur_OfficialRate / json.Cur_Scale,
36           date: json.Date,
37         };
38       }

```



```

40      // Add base currency
41      currencies[getters.currencyBaseGetter] = {
42          rate: 1,
43          date: new Date(),
44      };
45
46      commit('setCurrencyMutation', currencies);
47
48
49
50
51
52  mutations: {
53      setCurrencyMutation(state: any, currencies: any) {
54          state.currencies = currencies;
55      },
56  },
57
58  getters: {
59      currencySymbolsGetter: (state: any) => state.currencySymbols,
60      currenciesGetter: (state: any) => state.currencies,
61      currenciesKeysGetter: (state: any) => Object.keys(state.currencies),
62      currencyBaseGetter: (state: any) => state.currencyBase,
63  },
64  };
65
66  export default currency;
67

```

▼ HomePage.vue ×

src > frontend > src > views > pages > ▼ HomePage.vue > {} "HomePage.vue" > script > default

Ilya Ivanouski, 8 days ago | 1 author (Ilya Ivanouski)

```

1  <script lang="ts">
2  import Vue from 'vue';
3
4  import { mapActions, mapGetters } from 'vuex';
5
6  import HomeBillComponent from '../components/home/HomeBillComponent.vue';
7  import HomeCurrencyComponent from '../components/home/HomeCurrencyComponent.vue';
8  import LoaderComponent from '../components/LoaderComponent.vue';
9  import { ThisWindow } from '../interfaces/ThisWindow.interface';
10

```

```
11 export default Vue.extend({
12   name: 'HomePage',
13   metaInfo: {
14     title: 'Счет',
15   },
16   Ilya Ivanouski, a month ago • Feature/bi7 (#9)
17   components: {
18     HomeBillComponent,
19     HomeCurrencyComponent,
20     LoaderComponent,
21   },
22
23   data: () => ({
24     loading: true,
25   }),
26
27   async mounted() {
28     await this.checkAvailabilityData();
29     this.loading = false;
30   },
31
```

```
32     computed: {
33         ...mapGetters([
34             'currenciesGetter',
35             'currencyBaseGetter',
36             'infoBillGetter',
37             'windowGetter',
38         ]),
39
40         currency(): any {
41             return this.currenciesGetter;
42         },
43
44         currencyBase(): any {
45             return this.currencyBaseGetter;
46         },
47
48         bill(): any {
49             return this.infoBillGetter;
50         },
51
52         window(): ThisWindow {
53             return this.windowGetter;
54         },
55     },
56
```

```

57     methods: {
58         ...mapActions(['currencyFetchAction']),
59
60         async refresh() {
61             this.loading = true;
62             await this.currencyFetchAction();
63             this.loading = false;
64         },
65
66         async checkAvailabilityData() {
67             if (await !this.$store.getters.uidGetter) {
68                 await this.$store.dispatch('fetchInfoAction');
69             }
70             await this.currencyFetchAction();
71         },
72     },
73 });
74 </script>

```

```

75
76 <template>| Ilya Ivanouski, a month ago • HOTFIX (#7)
77   <v-card color="backgroundMain" outlined>
78     <v-card-title>
79       Счет ({{
80         bill
81         | currencyFilter({
82           style: 'currency',
83           currency: currencyBase,
84           maximumFractionDigits: 2,
85         })
86       }}) <v-spacer></v-spacer>
87
88       <!-- btn refresh -->
89       <v-btn v-if="window.isMobile" icon large @click="refresh">
90         <v-icon large>mdi-refresh</v-icon>
91       </v-btn>
92       <v-btn v-else-if="!window.isMobile" text @click="refresh">
93         <v-icon left>mdi-refresh</v-icon>
94         <span>обновить</span>
95       </v-btn>
96     </v-card-title>
97
98     <LoaderComponent v-if="loading" />
99
100    <v-row v-else-if="!loading">
101      <HomeBillComponent :currency="currency" />
102      <HomeCurrencyComponent :currency="currency" />
103    </v-row>
104  </v-card>
105 </template>
106
107 <style lang="scss" scoped>
108   .v-card {
109     border: 0;
110   }
111 </style>
112

```

## **6 МОДУЛЬНОЕ И МАНУАЛЬНОЕ ТЕСТИРОВАНИЕ АЛГОРИТМОВ РЕАЛИЗАЦИИ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ.**

Модульное тестирование, или юнит-тестирование (англ. unit testing) - процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы. Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода.

У модульного тестирования есть несколько прямых задач которое оно выполняет:

- выявление регрессионных ошибок при добавлении нового функционала или изменениях старого
- частичное документирование выполняемых алгоритмов

Учитывая вышеизложенное, в данном проекте решено было не использовать покрытие кода юнит-тестами, т.к. отсутствует вариант дальнейшего развития проекта и это сокращает время на разработку альфа версии. Однако, Юнит тесты могут быть всегда приложены к данному проекту в последующем, например, в версии 2.0.

### **Мануальное тестирование.**

Ручное тестирование (manual testing) - часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно производится тестировщиком без использования программных средств, для проверки программы или сайта путём моделирования действий пользователя.

Сегодня есть множество фреймворков для тестирования, поддерживающих практически все существующие языки. Казалось бы — можно брать и автоматизировать. Но даже сейчас ручные тесты важны.

Одно из объяснений их необходимости заключается в том в том, что при ручном тестировании функционала мы можем гораздо быстрее получить информацию о состоянии продукта, который анализируем, о качестве разработки. Кроме того, при автоматизации предварительно разработанные кейсы часто приходится менять и актуализировать, а на написание автотестов требуется определённое время.

При этом в процессе разработки может прийти обратная связь от заказчика, когда он увидит в готовом продукте какую-то функцию, которую решит изменить

до релиза — и, если вы уже подготовили для неё программные тесты, их придётся переписать. Обновление кейсов, автотестов и их проверка отнимут ценное время, которое можно было бы использовать на само обновление этой фичи.

Всё это означает, что главная цель ручных тестов — предварительно убедиться в том, что заявленный функционал работоспособен, не имеет ошибок и выдаёт ожидаемые, запланированные результаты. Без них нельзя быть уверенным в том, что можно работать дальше. Особенно это актуально для функций, на реализацию которых завязана последующая разработка. В таком случае возня с созданием автотестов на эти фичи становится блокирующим фактором для всей разработки продукта, сдвигая сроки и срывая дедлайны. Момент, когда кейсы придёт пора автоматизировать, всё равно рано или поздно наступит — но не стоит стремиться приблизить его искусственно в погоне за тотальным исключением ручного труда.

В дополнение к этому, на первых этапах разработки приложения автоматизация может оказаться довольно дорогой. Вам потребуется специалист, обладающий специфической квалификацией (и, возможно, не один). Постоянное поддержание тестов в актуальном состоянии требует затрат ресурсов вплоть до релиза фичи. А месяцы простоя, посвященные вылизыванию автотеста ударят по мотивации команды.

Если вы хотите регулярно добавлять новый функционал и успевать за действиями конкурентов, то перед тем, как создавать автотесты всегда проверяйте возможности продукта вручную. Просто потому что ручное тестирование ускоряет ваши процессы.

Это особенно актуально для мобильной разработки. Большинство таких проектов сегодня разрабатывается короткими спринтами, а значит фичи в них внедряются в ускоренном темпе. В подобных условиях ручное тестирование позволяет максимально оперативно давать команде обратную связь: сообщать о наличии багов — или радовать её тем, что всё окей и можно двигаться дальше. Провести серию автотестов вы сможете позже, покрыв с их помощью большие массивы кода. Ручное тестирование поможет подготовить кейсы для этой проверки.

Автотесты не позволяют проверить удобно ли использовать новые возможности приложения — проверка юзабилити всегда осуществляется вручную.

## ЗАКЛЮЧЕНИЕ

При выполнении дипломной работы были исследованы основные среды разработки и их библиотеки, после чего был определен наиболее удобный и подходящий стек технологий: vuejs, vuetify, firebase, typescript.

В основе выбора лежит лёгкость, практичность и стабильность. Также были исследованы некоторые способы хранения данных, и на этой же основе был выбран наиболее подходящий способ. Перед непосредственной разработкой приложения были проанализированы аспекты (инструменты), которые могли бы «тормозить» процесс разработки.

В итоге разработано кроссплатформенное веб-приложение для всех доступных операционных систем, которое будет позволять пользователю следить за своими финансами и давать возможность обновлять эти данные в реально времени и в любом месте, при наличии средств коммуникации и интернет.

Разработка проекта велась на публичных **GitHub** репозиториях:

[https://github.com/BulbaGroup/vue\\_crm](https://github.com/BulbaGroup/vue_crm)

[https://github.com/Domovikx/vue\\_crm](https://github.com/Domovikx/vue_crm) - develop fork

Приложение доступно по адресу: <https://vuecrm200711.web.app/>





## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. <https://ru.vuejs.org/> - vue-фреймворк
2. <https://vuex.vuejs.org/ru/guide/> - хранилище Vuex
3. <https://firebase.google.com/> - база данных
4. <https://vuetifyjs.com/ru/> - Material Design Component Framework
5. <https://www.npmjs.com/package/vue-moment> - vue-moment работа с датами, библиотека
6. <https://www.chartjs.org/samples/latest/> - библиотека для работы с графиками
7. <https://myrusakov.ru/what-is-spa.html>
8. <https://wezom.com.ua/blog/chto-takoe-spa-prilozheniya>
9. <https://semantica.in/blog/veb-interfejs.html>
10. <https://developer.mozilla.org/ru/docs/Web/HTML>
11. <https://developer.mozilla.org/ru/docs/Web/CSS>
12. <https://html5book.ru/osnovy-css>
13. <https://sass-scss.ru/documentation/sintaksis/>
14. <https://learn.javascript.ru/intro>
15. <https://ru.wikipedia.org/wiki/TypeScript>
16. <https://tproger.ru/translations/difference-between-git-and-github/>
17. [https://ru.hexlet.io/courses/html/lessons/github/theory\\_unit](https://ru.hexlet.io/courses/html/lessons/github/theory_unit)
18. <http://markshevchenko.pro/articles/github-pages/jekyll/>
19. [http://prgssr.ru/documentation/22\\_github\\_pages](http://prgssr.ru/documentation/22_github_pages)
20. <https://habr.com/ru/company/microsoft/blog/268837/>
21. <https://zencod.ru/articles/first-step-git>
22. <http://dreamhelg.ru/2017/02/symbol-svg-sprite-detail-guide/>
23. <https://itchief.ru/lessons/html-and-css/css-media-queries>
24. <https://www.insales.com.ua/blogs/blog/interfeys-onlayn-magazina>

## **ПРИЛОЖЕНИЯ**

Содержание электронного носителя:

- Архив с исходным кодом проекта
- Пояснительная записка
- Презентация