МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет компьютерных наук

*Кафедра теорий обработки и защиты информации*

*Система управления домашним бюджетом с рекомендациями по сокращению расходов.*

*Курсовая работа по дисциплине «Технологии программирования»*

09.03.02 *Информационные системы и технологии*

*Обработка информации и машинное обучение*

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_В.С. Тарасов, ст. преподаватель \_\_.\_\_20\_\_

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Лазуткина, 3 курс, д/о

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_В.И. Гараба, 3 курс, д/о

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_А.М. Трунова, 3 курс, д/о

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_Я.А. Рощупкин, 3 курс, д/о

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.С. Зенин, преподаватель

Воронеж 2023

Содержание

[Содержание 2](#_Toc136722219)

[Введение 3](#_Toc136722220)

[1 Постановка задачи 4](#_Toc136722221)

[1.1 Постановка задачи 4](#_Toc136722222)

[1.2 Требования к разрабатываемой системе 4](#_Toc136722223)

[1.3 Задачи, решаемые в процессе разработки 4](#_Toc136722224)

[2 Анализ предметной области 6](#_Toc136722225)

[2.1 Терминология (глоссарий) предметной области 6](#_Toc136722226)

[2.2 Цели создания приложения 7](#_Toc136722227)

[2.3 Сфера применения 7](#_Toc136722228)

[2.4 Технический обзор 8](#_Toc136722229)

[2.5 Обзор аналогов 8](#_Toc136722230)

[2.6 Требования к функциональности 11](#_Toc136722231)

[2.7 Пользователи системы 11](#_Toc136722232)

[2.8 Требования, не касающиеся функциональной части 12](#_Toc136722233)

[3 Графическое описание работы системы 14](#_Toc136722234)

[3.1 Диаграмма IDEF0 14](#_Toc136722235)

[3.2 Диаграммы прецедентов 14](#_Toc136722236)

[4 Реализация 16](#_Toc136722237)

[4.1 Анализ средств реализации 16](#_Toc136722238)

[4.2 Разработка frontend-части 16](#_Toc136722239)

[4.3 Разработка backend-части 28](#_Toc136722240)

[4.4 Тестирование 28](#_Toc136722241)

[5 Заключение 28](#_Toc136722242)

[6 Список используемой литературы 28](#_Toc136722243)

Введение

В современном мире любому важны удобство, комфорт и уверенность во всех сферах жизни, поэтому многие ведут учет расходов, следят за тем, как и на что тратят деньги. Это часть общей тяги человека к упорядочиванию своей жизни.

Учет личных финансов имеет ощутимый практический смысл: он помогает понять, на что мы тратим, и не остаться без средств к моменту оплаты счетов или кредитов, накопить определённую сумму. Удобство использования приложений по ведению бюджета заключается в том, что есть возможность поместить все счета в одно место и следить за ними, не используя множество сервисов сразу. В них можно отслеживать общий финансовый прогресс, динамики доходов, легко анализировать траты по категориям.

По приведённым выше причинам мобильное приложение для ведения бюджета – это простое решение проблемы упорядочения финансовых трат и счетов.

1. Постановка задачи
   1. Постановка задачи

Целью данного курсового проекта является разработка мобильного приложения для учёта доходов и расходов, которое позволит спланировать бюджет, проанализировать траты с помощью графиков и диаграмм, проконтролировать финансы, благодаря возможности отобразить историю операций за последний месяц, а также предоставит возможность узнать прогноз доходов и расходов на следующий месяц.

* 1. Требования к разрабатываемой системе

К разрабатываемой системе предъявляются следующие требования:

* обеспечение безопасности баз данных, защита от несанкционированного удаления данных;
* приложение должно устанавливаться и работать на мобильных устройствах версий Android 10 - Android 12, имеющих доступ к сети Интернет;
* реализация возможности добавлять, редактировать и удалять счета;
* реализация возможности добавлять, редактировать и удалять операции (доходы и расходы);
* возможность отслеживания динамики расходов и доходов с помощью графиков;
* возможность установки лимита на категорию или счет;
* возможность указать сумму финансовой цели на счету;
* реализация прогноза трат и пополнений на следующий месяц.
  1. Задачи, решаемые в процессе разработки
* Проектирование веб-приложения средствами языка UML
* Разработка backend-части, которая включает в себя:
  + Реализацию ролей авторизованного и неавторизованного пользователя;
  + Реализацию функциональных возможностей ролей;
  + Подключение внешнего модуля для хранения данных;
  + Разработку базы данных;
  + Разработку функциональности статических и динамических страниц.
* Разработка frontend-части, которая включает в себя:
  + Создание макета дизайна в Miro;
  + Реализация макета дизайна.
* Проведение тестирования проекта.

1. Анализ предметной области
   1. Терминология (глоссарий) предметной области

* Бюджет - финансовый план, состоящий из доходов и расходов.
* Доходы - деньги или материальные ценности, получаемые от предприятия, отдельного лица или какого-либо вида деятельности.
* Расходы - затраты, издержки, потребление чего-либо для определенных целей.
* Финансовая цель - определенная сумма денег, которая требуется для достижения некоторой материальной или нематериальной цели.
* Счет (в контексте данной системы) - виртуальный “кошелек”, с которым можно совершать различные операции: добавление или снятие суммы, установление лимита или финансовой цели.
* СУБД - система управления базами данных. Комплекс программно-языковых средств, позволяющих создать базы данных и управлять данными.
* Шаблон Model-View-ViewModel (MVVM) - шаблон проектирования, позволяющий разделить архитектуру на три функциональные части: модель, представление и модель представления. Этот шаблон помогает четко отделять бизнес логику и логику представления приложения от пользовательского интерфейса.
* Android - операционная система для мобильных устройств.
* Авторизация - предоставление определённому лицу или группе лиц прав на выполнение определенных действий, а также процесс проверки данных прав при попытке выполнения этих действий.
* Регистрация - действия, направленные на создание личной учетной записи в приложении, с целью получения доступа к его полному функционалу.
* Аватар - фотография или другое графическое изображение, используемое в учетной записи для персонализации пользователя.
* Пользователь - лицо, которое использует действующую систему для выполнения конкретной функции.
* PostgreSQL - свободная объектно-реляционная система управления базами данных.
* Java - строго типизированный объектно-ориентированный язык программирования общего назначения.
* Kotlin - статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine и разрабатываемый компанией JetBrains.
* Spring Boot - популярный фреймворк для создания веб-приложений с использованием Java.
* Android SDK - универсальное средство разработки мобильных приложений для операционной системы Android.
  1. Цели создания приложения
* Разработка мобильного приложения для учёта доходов и расходов, позволяющего контролировать и анализировать траты с помощью графиков.
* Помощь пользователю в достижении своих финансовых целей.
  1. Сфера применения

Приложение для ведения домашнего бюджета может использоваться в различных сферах, таких, как личная жизнь, семейный бюджет, управление финансами малого бизнеса и т.д. Оно помогает отслеживать доходы и расходы, контролировать бюджет, планировать расходы на будущее и принимать решения на основе финансовой информации. Это полезный инструмент для тех, кто хочет улучшить свою финансовую грамотность и достичь финансовой стабильности.

* 1. Технический обзор

1. Учет доходов и расходов. Пользователь может добавлять свои доходы и расходы в приложение, чтобы отслеживать свои финансы.
2. Категоризация расходов. Приложение может предоставить пользователю возможность категоризировать свои расходы, чтобы понимать, на что тратится большая часть бюджета.
3. Анализ финансовой информации. Приложение может предоставлять графики и диаграммы, чтобы пользователь мог быстро анализировать свои финансы.
   1. Обзор аналогов

Существует множество приложений для ведения домашнего бюджета, и каждое из них имеет свои преимущества и недостатки. Рассмотрим несколько популярных аналогов.

* + 1. Wallet

“Wallet” — это бесплатное приложение, которое позволяет отслеживать доходы и расходы, устанавливать бюджеты, получать уведомления о предстоящих платежах и анализировать свои финансы.

Сильные стороны приложения:

1. Удобное хранение и управление криптовалютами.
2. Возможность быстрого и удобного проведения транзакций
3. Высокий уровень безопасности благодаря использованию шифрования и двухфакторной аутентификации.
4. Возможность создания нескольких кошельков для разных криптовалют.

Слабые стороны:

1. Риск потери доступа к кошельку в случае утери или повреждения устройства, на котором он хранится.
2. Некоторые кошельки могут иметь ограничения по количеству поддерживаемых криптовалют.
3. Некоторые кошельки могут иметь высокую комиссию за проведение транзакций.
4. Не всегда точная классификация расходов.



1. Интерфейс приложения Wallet
   * 1. PocketGuard

“PocketGuard” — это еще одно бесплатное приложение, которое позволяет контролировать свои финансы, устанавливать бюджеты и получать уведомления о предстоящих платежах. Однако некоторые пользователи жалуются на то, что приложение не всегда корректно отображает баланс на счете.



1. Интерфейс приложения PocketGuard
   * 1. Финансы

“Финансы” - бесплатное приложение, явным недостатком которого является ограниченный функционал: нет предоставления прогноза будущего баланса, достаточно неудобное меню, которое приходится открывать каждый раз, чтобы перейти к каким-либо функциям. Однако, в этом приложении можно выделить некоторые плюсы: возможность добавления нескольких счетов для всех пользователей, возможность работы с приложением без авторизации, возможность создания своей категории.



1. Интерфейс приложения “Финансы”



1. Меню приложения

В целом, каждое из этих приложений имеет свои преимущества и недостатки. Рассмотрев их, можно сделать вывод, что система управления домашним бюджетом должна иметь удобный интерфейс, всегда доступное меню, достаточную функциональность и однозначную классификацию операций.

* 1. Требования к функциональности
* Добавление доходов и расходов, разделение их на категории для удобства отслеживания динамики трат/пополнений.
* Возможность добавления своих категорий.
* Вывод графиков доходов и расходов для контроля финансов.
* Установление лимитов на определённые категории.
* Возможность добавления своих счетов.
* Возможность совершить регистрацию/авторизацию в системе.
* Просмотр истории операций за день или месяц.
* Загрузка фото чека/квитанции с напоминанием записать сумму позже.
  1. Пользователи системы

В системе существуют такие группы пользователей, как неавторизованный, авторизованный пользователь и администратор. Для неавторизованного пользователя должны быть предоставлены следующие функции:

* добавление одного счета;
* добавление расходов и доходов на счет;
* редактирование и удаление добавленной операции;
* указание категории трат/доходов при добавлении операции;
* создание новых категорий;
* редактирование всех категорий и удаление категорий, созданных пользователем;
* установка лимитов на категории и счет;
* указание финансовой цели на счете;
* просмотр графиков доходов и расходов за день, неделю, месяц и год;
* просмотр истории операций за день или месяц;
* просмотр прогноза финансов на следующий месяц;

Для авторизованного пользователя:

* добавление нескольких счетов;
* редактирование и удаление добавленных счетов;
* добавление расходов и доходов на выбранный счет;
* редактирование и удаление добавленной операции;
* указание категории трат/доходов при добавлении операции;
* создание новых категорий;
* редактирование всех категорий и удаление категорий, созданных пользователем;
* установка лимитов на категории и счета;
* просмотр графиков доходов и расходов за день, неделю, месяц и год;
* просмотр истории операций за день или месяц;
* просмотр прогноза финансов на следующий месяц.

Администратор имеет возможность просматривать анонимную статистику трат пользователей по категориям.

* 1. Требования, не касающиеся функциональной части

Для реализации серверной части приложения выбраны технологии:

* язык программирования Java;
* фреймворк Spring Boot;
* СУБД PostgreSQL.

Для реализации клиентской части приложения выбраны технологии:

* язык программирования Kotlin;
* Android SDK.

Требования к программному обеспечению клиентской части:

* приложение должно устанавливаться и работать на любом устройстве под управлением операционной системы Android 10- 12.

Требования к программному обеспечению серверной части:

* серверная часть приложения должна быть реализована на языке программирования Java c использованием фреймворка Spring Boot;
* в качестве системы управления базами данных должна быть использована СУБД PostgreSQL.

Требования к техническому обеспечению клиентской части: устройство под управлением ОС Android 10 должно иметь следующие характеристики:

* диагональ экрана 4,5 дюймов и больше;
* объем оперативной памяти 1,5 Гб и больше.

Требования к техническому обеспечению серверной части:

* оперативная память сервера 512 Мб и больше;
* постоянная память сервера 512 Мб и больше;
* тактовая частота процессора 2 ГГц и выше;
* количество ядер процессора 1 и более;
* возможность доступа к сети Интернет.

Требования к дизайну приложения: приложение должно быть выполнено в едином стиле. Цветовая палитра приложения должна содержать два основных цвета - #FFFFFF для фона и #FFD166 для кнопок и навигации. Используются шрифты без засечек, всего не более 3 шрифтов. Для иконок используются распространенные обозначения.

1. Графическое описание работы системы
   1. Диаграмма IDEF0



* 1. Диаграммы прецедентов
     1. Диаграмма прецедентов (авторизованный пользователь)



1. Диаграмма прецедентов (авторизованный пользователь)
   * 1. Диаграмма прецедентов (неавторизованный пользователь)



1. Диаграмма прецедентов (неавторизованный пользователь)
   * 1. Диаграмма прецедентов (администратор)



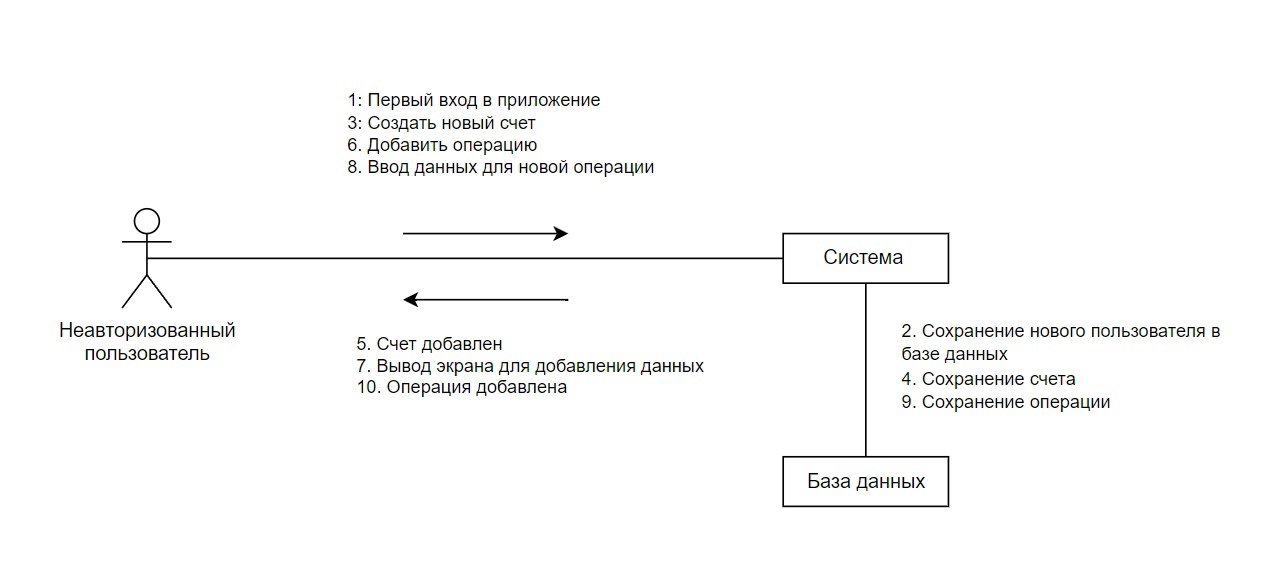
1. Диаграмма прецедентов (администратор)
   1. Диаграмма развёртывания



* 1. Диаграмма состояний(пользователь)



* 1. Диаграмма сотрудничества



* 1. Диаграмма последовательности



1. Диаграмма последовательности (1 часть)



1. Диаграмма последовательности (2 часть)
2. Реализация
   1. Анализ средств реализации

Для реализации данного проекта были выбраны следующие средства:

Для реализации серверной части выбраны технологии:

* язык программирования Java;
* фреймворк Spring Boot;
* СУБД PostgreSQL.

Для реализации клиентской части выбраны технологии:

* язык программирования Kotlin;
* Android SDK.

Java - это объектно-ориентированный язык программирования, который широко используется для создания приложений и веб-сайтов. Он имеет множество инструментов и библиотек для обработки данных, создания графического интерфейса и управления потоком выполнения программы.

Spring Boot - это фреймворк для создания приложений на языке Java, который облегчает разработку, настройку и развертывание приложений. Он предоставляет множество инструментов и библиотек для работы с базами данных, безопасности, тестирования и многого другого.

PostgreSQL - это объектно-реляционная система управления базами данных (СУБД), которая обеспечивает надежное хранение и обработку данных. Она имеет множество функций, включая поддержку транзакций, индексирование, полнотекстовый поиск, репликацию и многие другие.

Android SDK - это набор инструментов для разработки приложений под операционную систему Android. Он содержит все необходимые библиотеки, API и инструменты для создания приложений для Android. Android SDK включает в себя Android Studio - интегрированную среду разработки приложений, которая предоставляет мощный набор инструментов, в том числе средства для создания пользовательского интерфейса, отладки, тестирования и профилирования приложений.

Kotlin - это статически типизированный язык программирования, который может использоваться как замена Java для разработки приложений для Android. Он предоставляет множество средств для создания более безопасного, надежного и производительного кода, а также обладает более простым и понятным синтаксисом, чем Java.

Сочетание всех этих средств позволяет разработчикам создавать высокопроизводительные и надежные приложения с удобным интерфейсом и хорошей масштабируемостью. Java предоставляет широкий спектр инструментов и библиотек, Spring Boot облегчает разработку и внедрение приложений, PostgreSQL обеспечивает надежное хранение и обработку данных, в то время как Android SDK обеспечивает все необходимые средства для создания, тестирования и развертывания приложений, а Kotlin предоставляет удобный и безопасный язык программирования для разработки.

* 1. Разработка frontend-части

Приложение имеет архитектуру, соответствующую шаблону Model-View-ViewModel (MVVM).

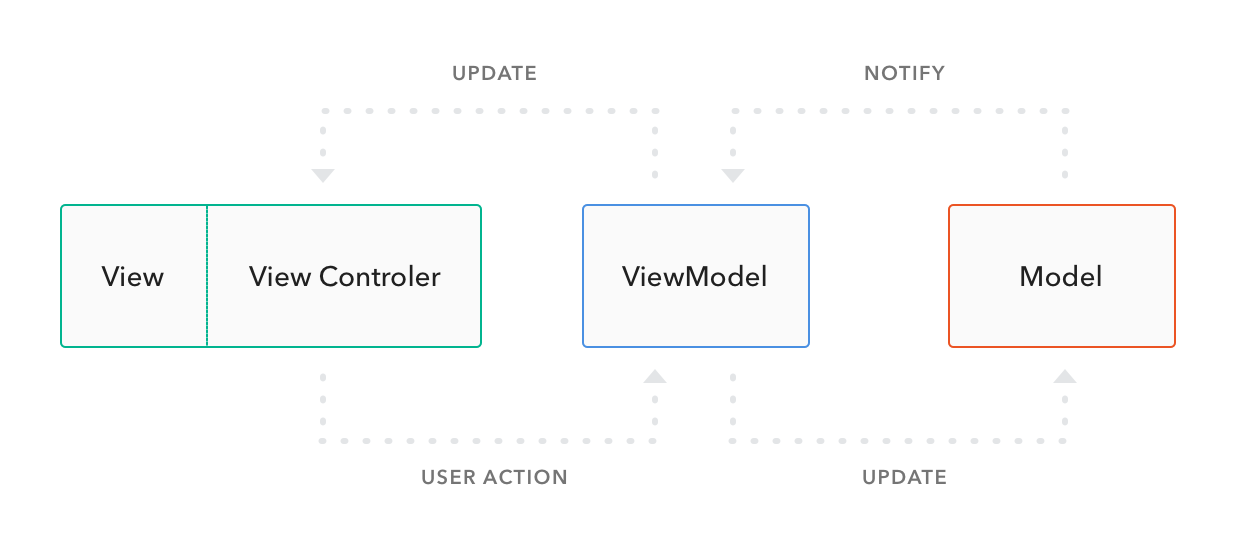
В настоящее время MVVM является наиболее популярным архитектурным решением, применяемым при разработке крупных приложений, требующих участия в разработке больших команд, являющихся хорошо тестируемыми, длительно поддерживаемыми и постоянно развивающимися. Безусловно, хорошо спроектированная архитектура очень важна для того, чтобы обеспечить длительную поддержку проекта.

MVVM – это подход, который имеет следующею логику распределения ответственности между модулями:

Model: этот модуль отвечает за создание моделей данных и может содержать в себе бизнес-логику. Также можно создать вспомогательные классы, например такие как manager-класс для управления объектами в Model и network manager для обработки сетевых запросов и парсинга.

View: модуль View в MVVM охватывает интерфейс (подклассы UIView, файлы .xib и .storyboard), логику отображения (анимация, отрисовка) и обработку пользовательских событий (нажатие кнопок и т.д.).

ViewModel: это то место, где будет располагаться большая часть кода. Слой ViewModel запрашивает данные у Model (это может быть запрос к локальной базе данных или сетевой запрос) и передает их обратно во View, уже в том формате, в котором они будут там использоваться и отображаться. Но это двунаправленный механизм, действия или данные, вводимые пользователем проходят через ViewModel и обновляют Model. Поскольку ViewModel следит за всем что отображается, то полезно использовать механизм связывания между этими двумя слоями.



1. Архитектура Model-View-ViewModel
   * 1. Экран “Приветствие”

Данный экран появляется при первом входе в приложение, в нем содержится приветственный текст и кнопка для начала работы. По центру экрана находится логотип приложения.



1. Экран “Приветствие”

* + 1. Экран “Начало работы”

На данном экране предлагается ввести название счета и сумму остатка на данном счету. Внизу находится кнопка “Далее”, которая ведет на главный экран.



1. Экран “Начало работы”
   * 1. Экран “Главная”

На данном экране выводится сумма на счету и история операций, которую можно посмотреть, как за определенный день (для этого предлагается выбрать день на календаре), так и за определенный месяц. При выборе следующего месяца выводится прогноз расходов и доходов с рекомендациями по сокращению расходов. Справа внизу находится кнопка “+”, ведущая на экран “Добавление операции”. В нижней панели находится меню, состоящее из вкладок “Аналитика”, “Категории”, “Главная”, “Счета” и “Профиль”.



1. История операций за определённый день



1. История операций за определённый месяц



1. Прогнозы на следующий месяц



1. Календарь
   * 1. Экран “Добавление операции”

Данный экран предназначен для совершения операций над счетом – добавления расходов или доходов. В верхней части экрана можно выбрать тип операции, “Доходы” или “Расходы”. Далее идет поле, в которое требуется ввести сумму операции, поле выбора счета, выбора категории и даты. К операции можно добавить комментарий и фотографию. Также операцию можно сделать регулярной, чтобы она повторялась через какой-либо период времени (день, неделя, месяц или год) и добавить напоминание. В конце находится кнопка “Добавить”.



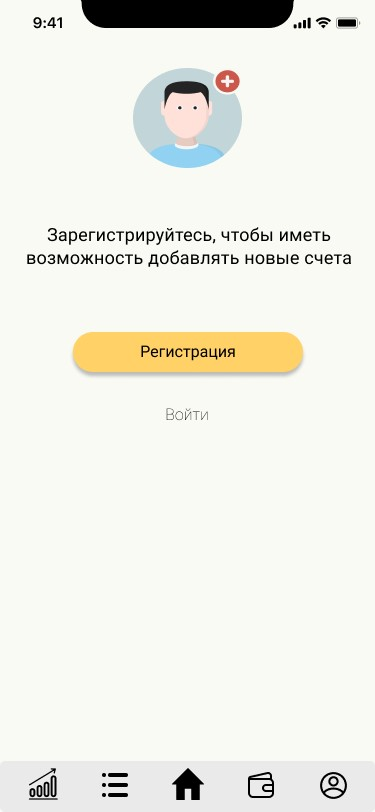
1. Экран “Добавление операции”
   * 1. Экран “Редактирование операции”

Данный экран подобен экрану “Добавление операции” и позволяет изменить данные в добавленной операции или удалить ее.

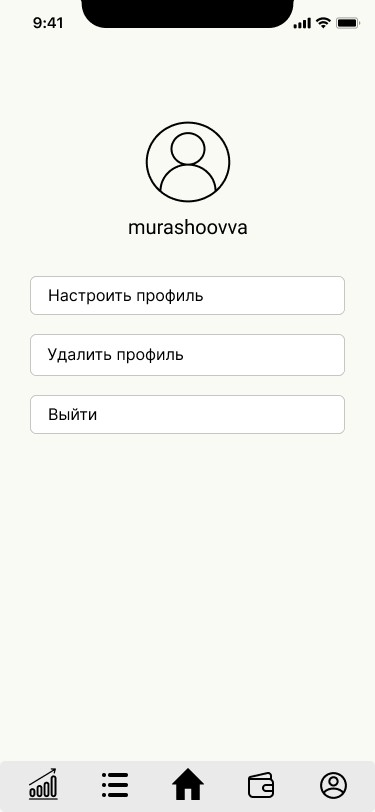


1. Экран “Редактирование операции”
   * 1. Экран “Профиль”

При переходе на этот экран неавторизованный пользователь увидит предложение зарегистрироваться и две кнопки: “Зарегистрироваться” и “Войти”. При нажатии на кнопку “Зарегистрироваться” пользователь переходит на экран “Регистрация”, при нажатии на кнопку “Войти” пользователь переходит на экран “Авторизация”. Авторизованный пользователь при переходе на этот экран увидит аватар своего профиля и логин, указанный при регистрации. Ниже находятся настройки профиля, кнопки “Выйти” и “Удалить профиль”.

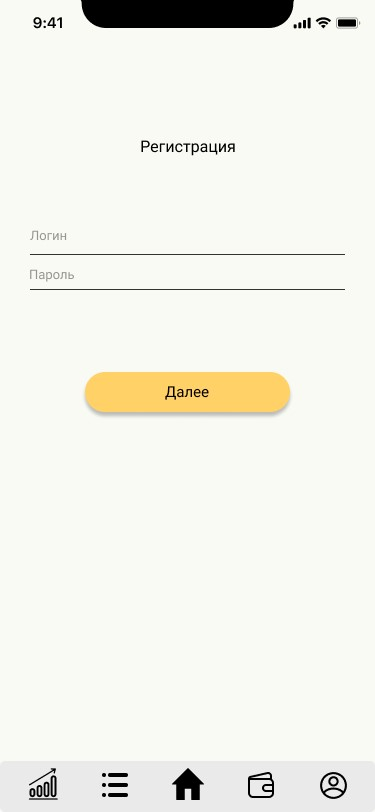


1. Экран “Профиль” неавторизованного пользователя



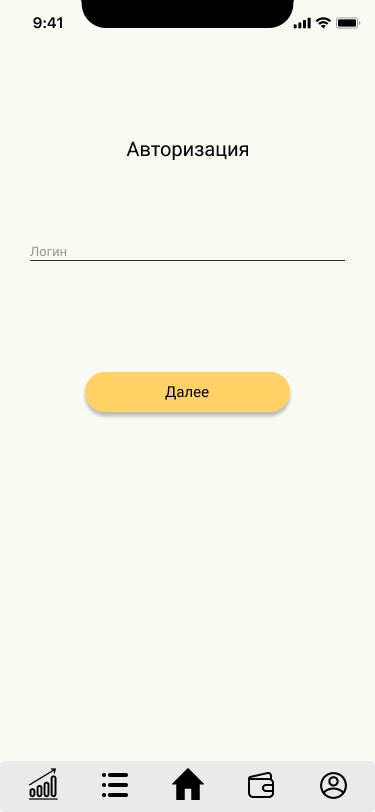
1. Экран “Профиль” авторизованного пользователя
   * 1. Экран “Регистрация”

Данный экран предлагает придумать логин и пароль пользователя. При успешной регистрации выводится сообщение, что пользователь успешно зарегистрирован.



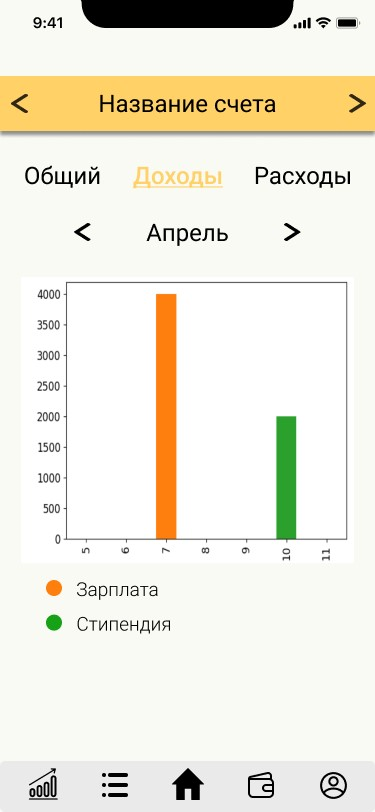
1. Экран “Регистрация”
   * 1. Экран “Авторизация”

Данный экран предлагает ввести логин и пароль пользователя. При успешной авторизации осуществляется переход на экран “Профиль” для авторизованных пользователей.

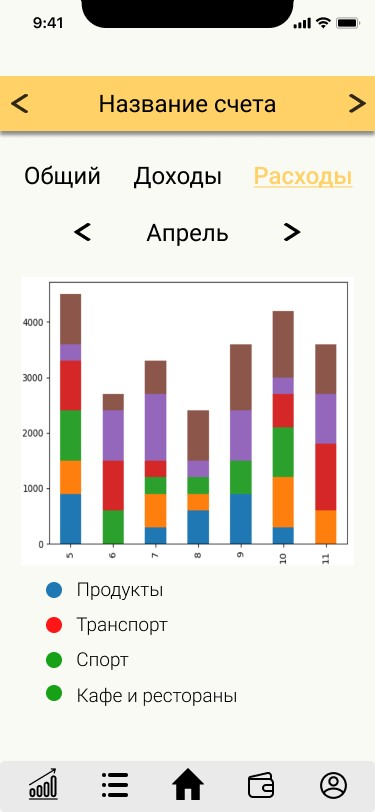


1. Экран “Авторизация”
   * 1. Экран “Аналитика”

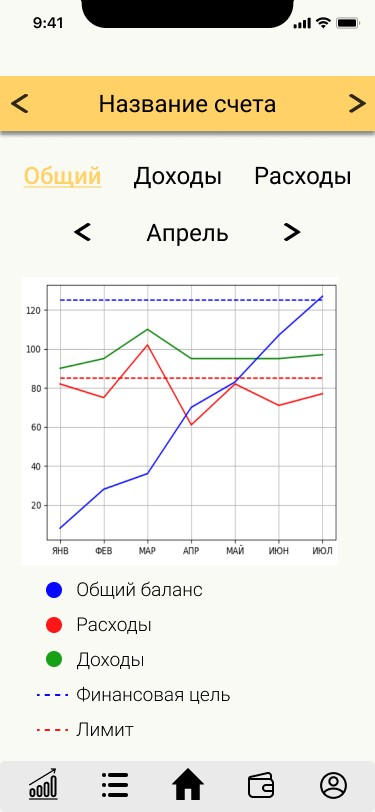
На данном экране авторизованный пользователь может выбрать счет, аналитику которого желает увидеть, тип графика (общий, только по доходам или только по расходам), вид графика (круговая диаграмма или столбчатая) и даты (определенный день, неделя, месяц или год). Ниже выводится выбранная диаграмма.



1. Экран аналитики доходов

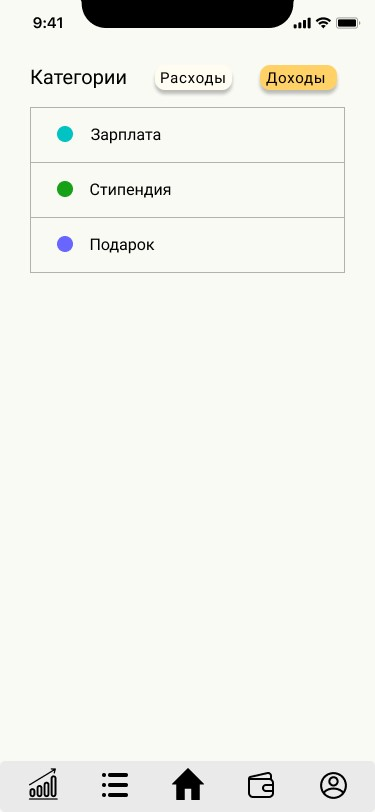


1. Экран аналитики расходов



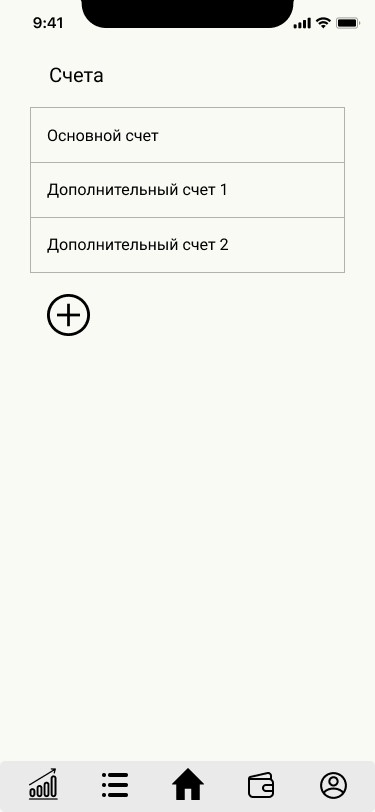
1. Экран аналитики общий
   * 1. Экран “Категории”

При переходе на этот экран пользователь видит список категорий. В верхней панели можно выбрать, какие категории показывать - для доходов или расходов. Ниже выводится список всех категорий, а в конце есть кнопка “+”, позволяющая добавить новую категорию. Каждую категорию можно редактировать, при этом для стандартных категорий можно настроить уведомления и лимит, для пользовательских, помимо этого, редактировать цвет категории и название, а также удалить категорию, если по ней не было операций.

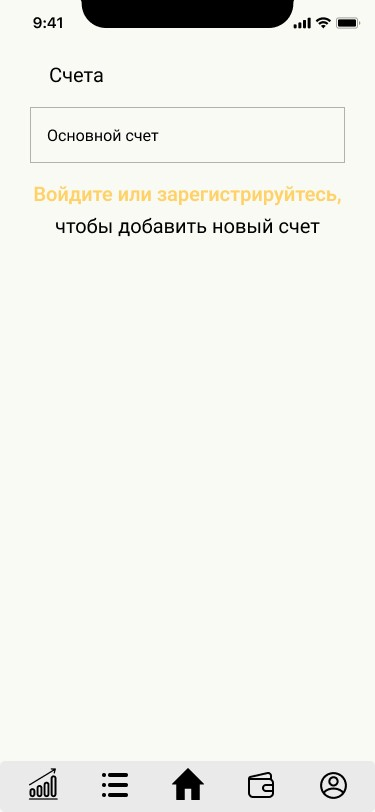


1. Экран “Категории”
   * 1. Экран “Счета”

На данном экране находится список всех счетов пользователя. Ниже располагается кнопка “+”, позволяющая добавить новую категорию. Если пользователь не авторизован, то при нажатии кнопки “+” он переходит на экран профиля с предложением зарегистрироваться или войти. Авторизованный пользователь может добавить любое количество новых счетов и настраивать их. Для счетов доступны настройка лимита, настройка финансовой цели, уведомлений и удаление счета.



1. Экран “Счета” (для авторизованного пользователя)



1. Экран “Счета” (для авторизованного пользователя)
   * 1. Навигация по приложению

Для навигации в этом приложении используется меню в нижней панели. Кнопка с иконкой графика ведет на экран “Аналитика”, кнопка с иконкой списка ведет на экран “Категории”, кнопка с изображением дома возвращает пользователя на экран “Главная”, кнопка с изображением купюры ведет на экран “Счета”, а кнопка с иконкой аватара ведет на экран “Профиль”.

* 1. Разработка backend-части

MVC, Spring Boot (про 4 слоя)



(1 слой) всё что в папке контроллеров

(2 слой) всё что в папке сервисов

(3 слой) Spring data (Позволяет сохранять классы в базу данных)

Entity классы  
Repository интерфейсы

из репозитория берем и в него сохраняем Entity

(4 слой) PostgrSQL

Принципы SOLID

* 1. Тестирование

1. Заключение
2. Список используемой литературы