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

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

1.1 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.

1.2 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.

1.3 Задачи, решаемые в процессе разработки

- Проектирование веб-приложения средствами языка UML;
- разработка backend-части, которая включает в себя:
    1. реализацию ролей авторизованного и неавторизованного пользователя;
    2. реализацию функциональных возможностей ролей;

3. подключение внешнего модуля для хранения данных;
4. разработку базы данных;
5. разработку функциональности статических и динамических страниц.

— разработка frontend-части, которая включает в себя:

1. создание макета дизайна в Miro;
2. реализация макета дизайна.

— проведение тестирования проекта.

2 Анализ предметной области

2.1 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. [8]

Регистрация - действия, направленные на создание личной учетной записи в приложении, с целью получения доступа к его полному функционалу. [8]

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, он имеет множество инструментов и библиотек для обработки данных, создания графического интерфейса и управления потоком выполнения программы.

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, который облегчает разработку, настройку и развертывание приложений. Он предоставляет множество инструментов и библиотек для работы с базами данных, безопасности, тестирования и многого другого.

Android SDK - универсальное средство разработки мобильных приложений для операционной системы 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. [9]

2.2 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.

2.3 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.

2.4 Технический обзор

- Учет доходов и расходов. Пользователь может добавлять свои доходы и расходы в приложение, чтобы отслеживать свои финансы;
- категоризация расходов. Приложение может предоставить пользователю возможность назначать категории своих расходов, чтобы понимать, на что тратится большая часть бюджета;
- анализ финансовой информации. Приложение может предоставлять графики и диаграммы, чтобы пользователь мог быстро анализировать свои финансы.

2.5 Обзор аналогов

Существует множество приложений для ведения домашнего бюджета, и каждое из них имеет свои преимущества и недостатки. Рассмотрим несколько популярных аналогов. [10]

2.5.1 Wallet

“Wallet” — это бесплатное приложение, которое позволяет отслеживать доходы и расходы, устанавливать бюджеты, получать уведомления о предстоящих платежах и анализировать свои финансы (рис. 1).

Сильные стороны приложения:

- удобное хранение и управление крипто валютами;
- возможность быстрого и удобного проведения транзакций
- высокий уровень безопасности благодаря использованию шифрования и двухфакторной аутентификации;
- возможность создания нескольких кошельков для разных крипто валют.

Слабые стороны:

— риск потери доступа к кошельку в случае утери или повреждения устройства, на котором он хранится;

— некоторые кошельки могут иметь ограничения по количеству поддерживаемых крипто валют;

— некоторые кошельки могут иметь высокую комиссию за проведение транзакций;

— не всегда точная классификация расходов.

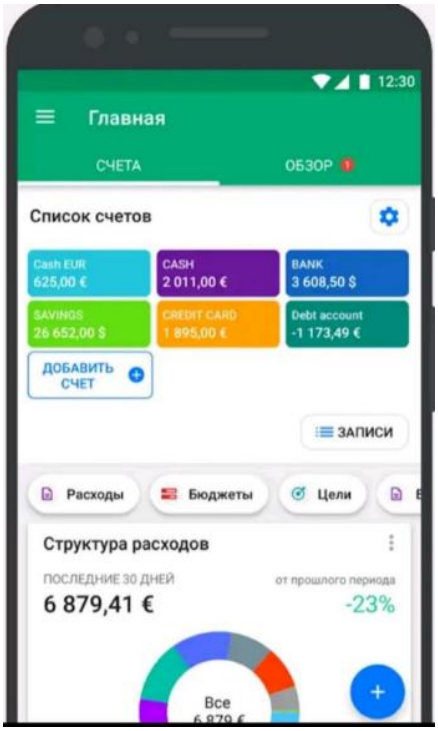

Рисунок 1 - Интерфейс приложения Wallet

2.5.2 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 (рис. 2).

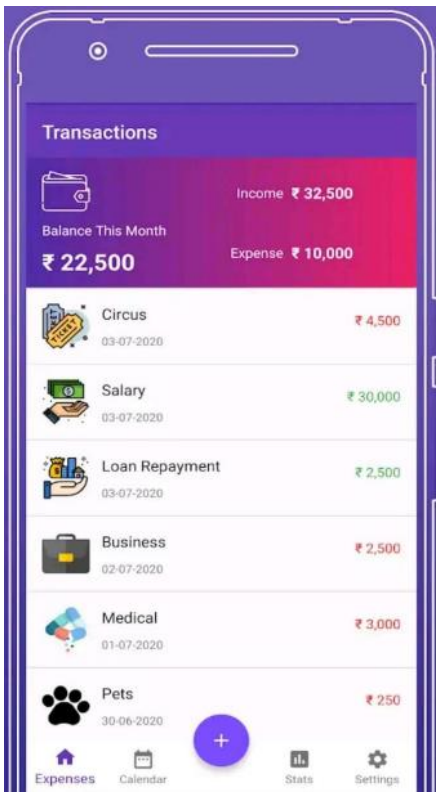

Рисунок 2 - Интерфейс приложения PocketGuard

2.5.3 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 (рис. 3, 4).

Рисунок 3 - Интерфейс приложения “Финансы”

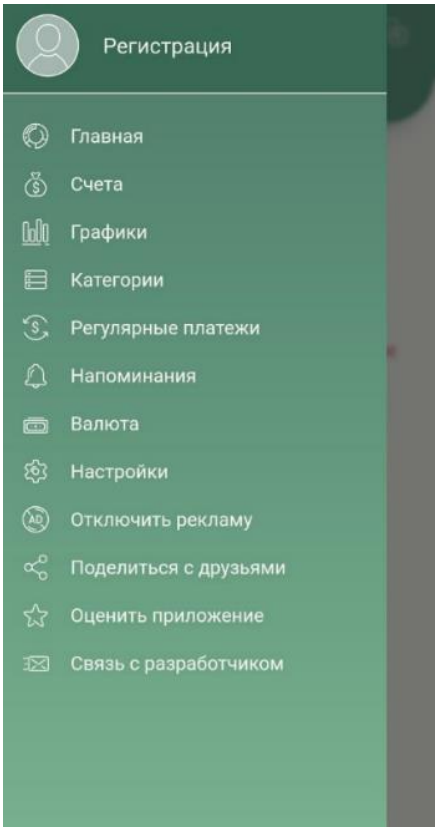

Рисунок 4 - Меню приложения “Финансы”

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.

2.6 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.

2.7 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.

2.8 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 с 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 - #F9FAF4 для фона и #FFD166 для кнопок и навигации. Используются шрифты без засечек, всего не более 3 шрифтов. Для иконок используются распространенные обозначения.

3 Графическое описание работы системы

3.1 Диаграмма IDEF0

Рассмотрим основной бизнес-попроцесс на примере контекстной диаграммы, представленной на рисунке 5. Данная диаграмма представляет собой общее видение процесса работы приложения.

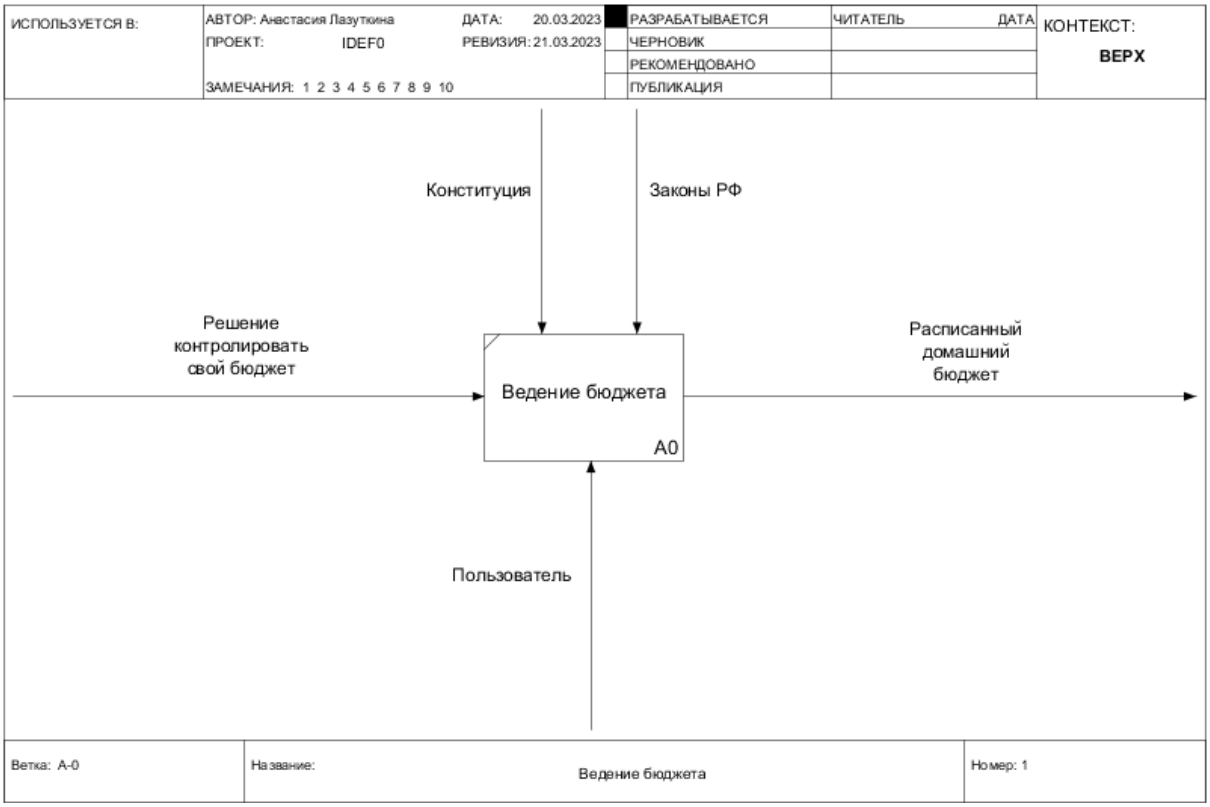

Рисунок 5 - Диаграмма IDEF0

3.2 Диаграммы прецедентов

Диаграммы прецедентов показывают действия актеров, т.е. действующих лиц системы, по отношению к системе, а также все их возможности.

3.2.1 Диаграмма прецедентов (авторизованный пользователь)

На данной диаграмме представлены сценарии взаимодействия авторизованного пользователя с приложением (рис. 6).

Рисунок 6 - Диаграмма прецедентов (авторизованный пользователь)

3.2.2 Диаграмма прецедентов (неавторизованный пользователь)

На данной диаграмме представлены сценарии взаимодействия неавторизованного пользователя с приложением (рис. 7).

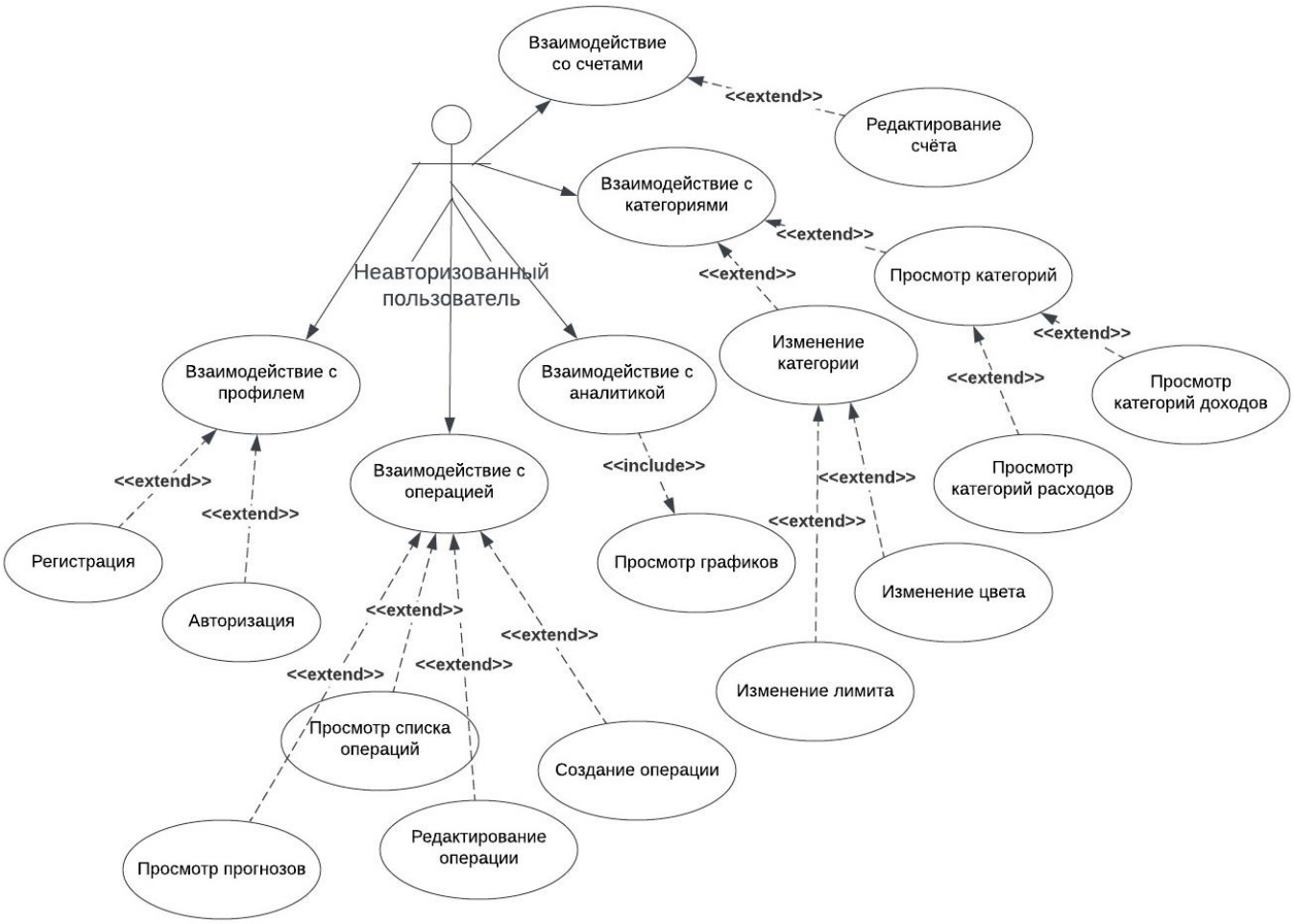

Рисунок 7 - Диаграмма прецедентов (неавторизованный пользователь)

3.2.3 Диаграмма прецедентов (администратор)

На данной диаграмме представлен сценарий взаимодействия администратора с приложением (рис. 8).

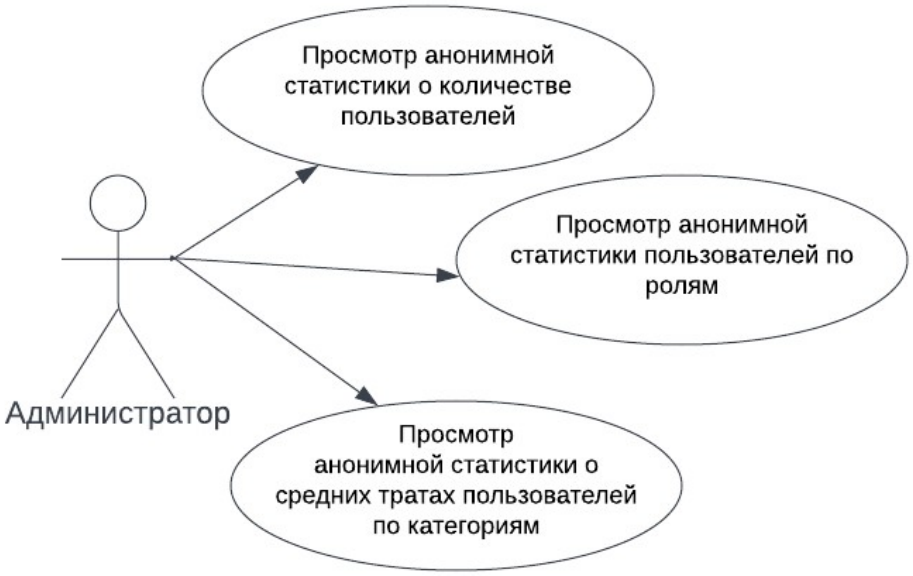

Рисунок 8 - Диаграмма прецедентов (администратор)

3.3 Диаграмма развёртывания

Диаграмма развертывания показывает, какие аппаратные компоненты существуют, какие программные компоненты работают на каждом узле, и как различные части этого комплекса соединяются друг с другом.

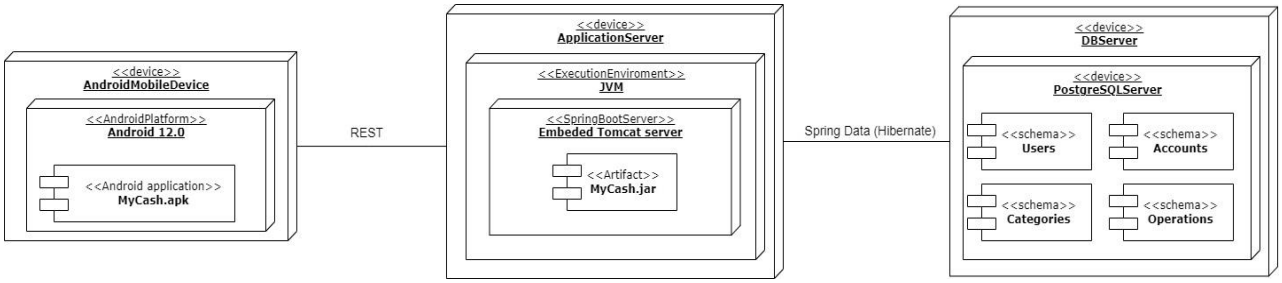

Рисунок 9 - Диаграмма развёртывания

3.4 Диаграмма состояний(пользователь)

Диаграмма состояний показывает, как объект переходит из одного состояния в другое. На рисунке 10 показано изменение состояний пользователя от первого входа в приложение до выхода, пользователь может находится в следующих состояниях: незарегистрированный, зарегистрированный, неавторизованный, авторизованный, удалённый).

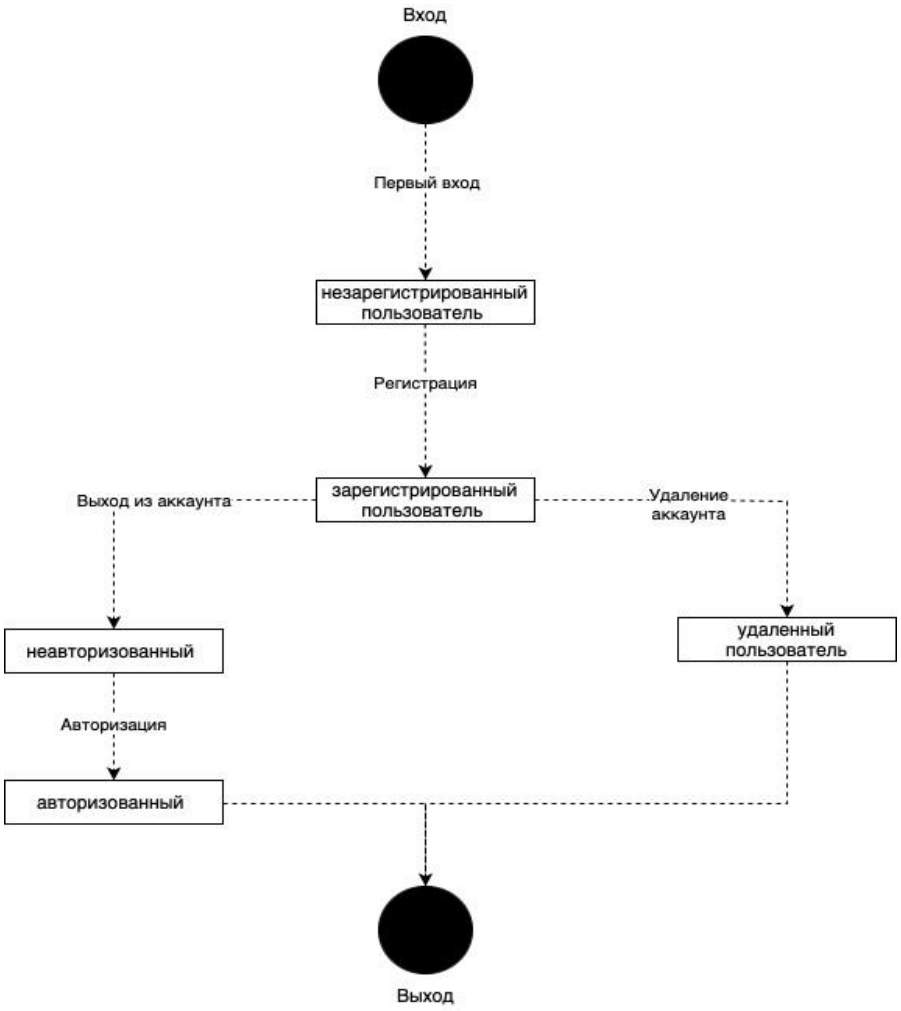

Рисунок 10 - Диаграмма состояний пользователя

3.5 Диаграмма сотрудничества

Данная диаграмма показывает связи и взаимодействия неавторизованного пользователя с системой и базой данных.

Рисунок 11 - Диаграмма сотрудничества

3.6 Диаграмма последовательности

Диаграмма последовательности отображает детальное описание логики сценариев использования и уточняет диаграммы прецедентов (рис. 12).

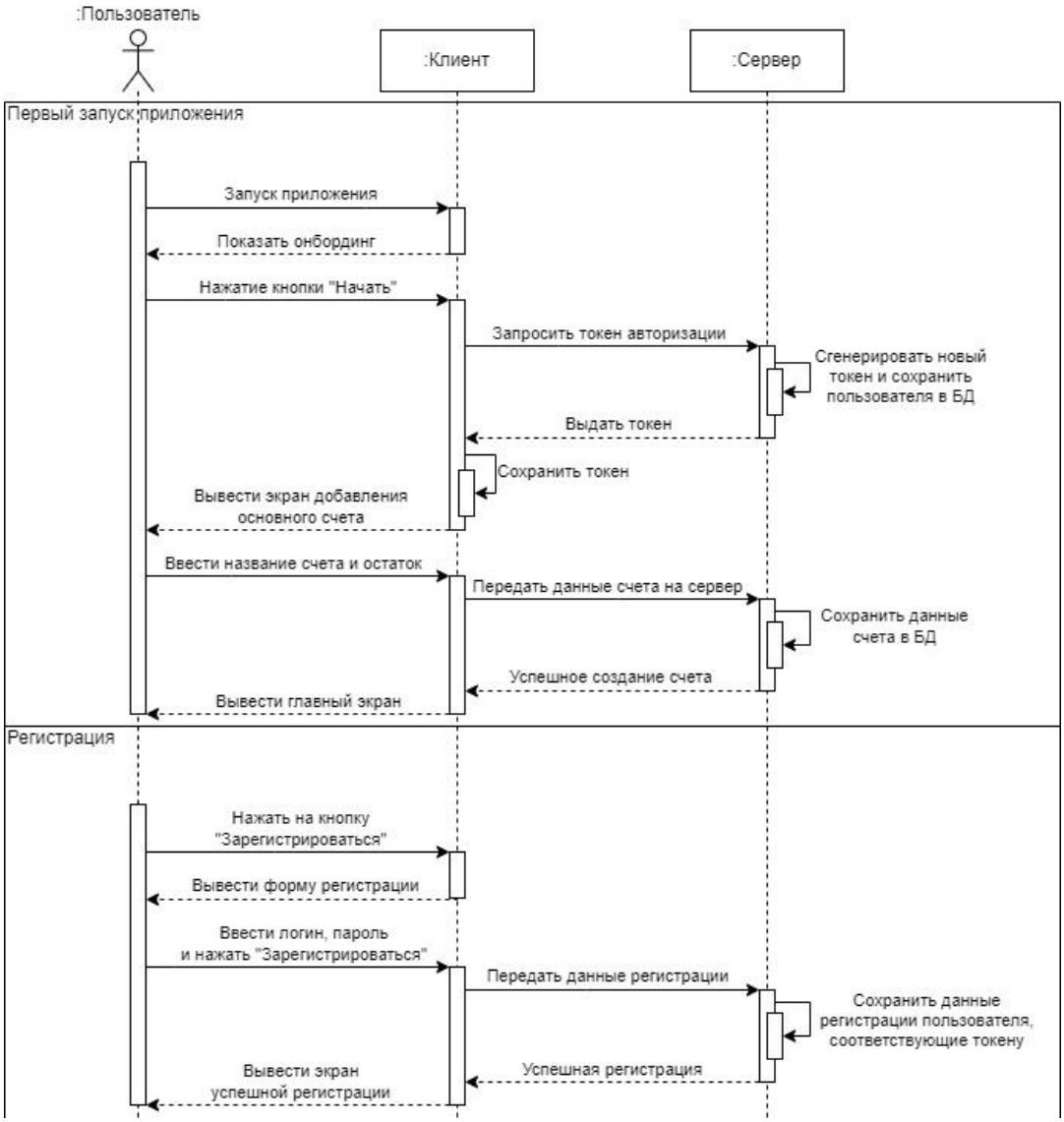

Рисунок 12 - Диаграмма последовательности (1 часть)

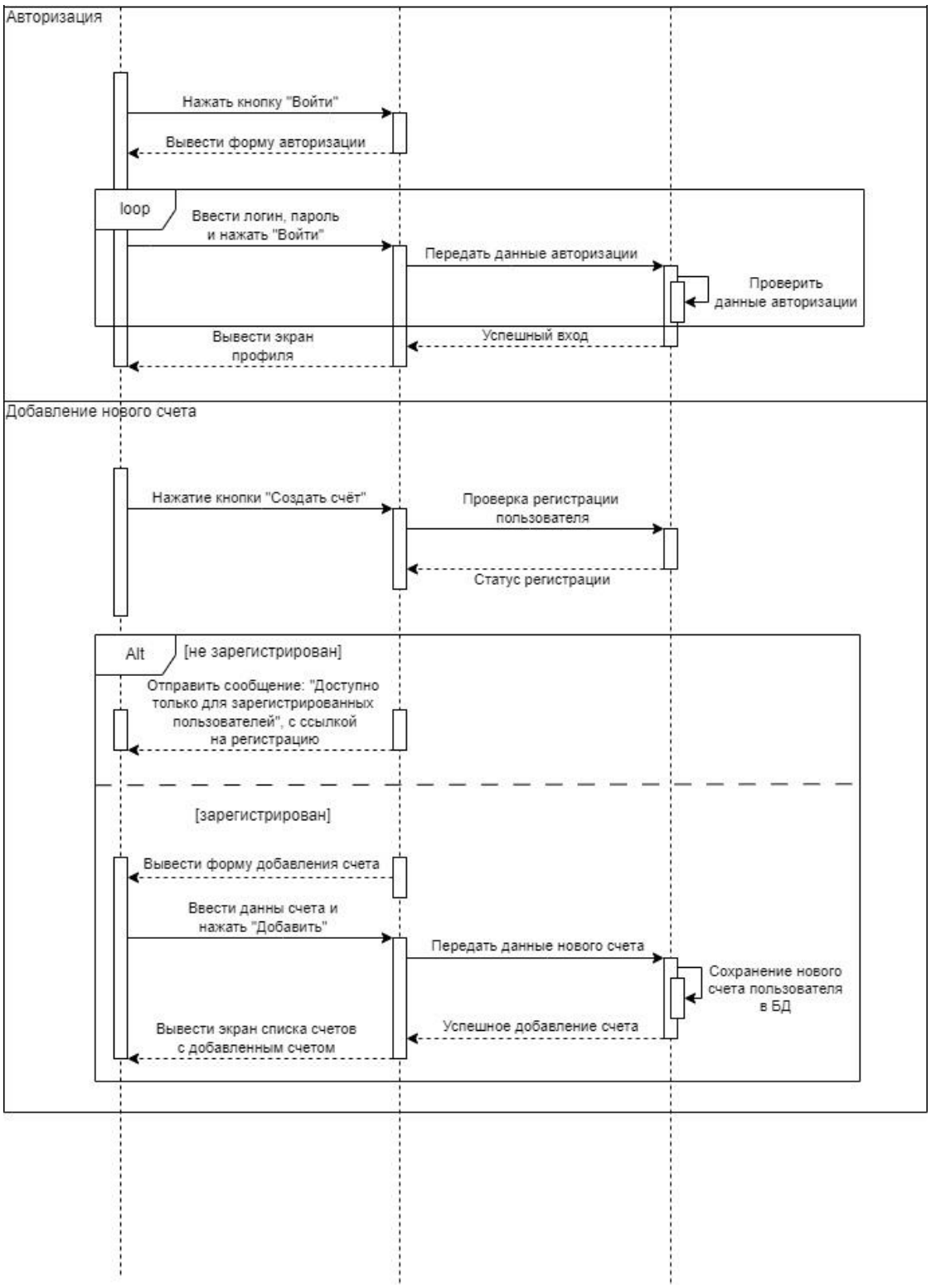

Рисунок 13 - Диаграмма последовательности (2 часть)

4 Реализация

4.1 Анализ средств реализации

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.

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.

4.2 Разработка frontend-части

4.2.1 Kotlin

Для frontend-части приложения был выбран Kotlin. Он был объявлен приоритетным языком программирования для Android-разработки на Google I/O в 2019. Используя Kotlin, разработчик получает [7]:

— небольшое количество кода в сочетании с удобочитаемостью;

— зрелый язык и окружение. С момента своего создания в 2011 году Kotlin постоянно развивался не только как язык, но и как целая экосистема с надежным инструментарием. Теперь он полностью интегрирован в Android Studio и активно используется многими компаниями для разработки Android-приложений;

— поддержка Kotlin в Android Jetpack и других библиотеках. Расширения KTX добавляют функции языка Kotlin, такие как корутины, функции-расширения, лямбды и именованные параметры, в существующие Android библиотеки;
— совместимость с Java. Есть возможность использовать Kotlin вместе с Java в приложениях без необходимости переноса всего кода на Kotlin.
— поддержка мультиплатформенной разработки. Можно использовать Kotlin для разработки не только Android, но и iOS, серверных и веб-приложений;
— безопасность кода. Небольшое количество кода и хорошая читабельность приводят к минимальному количеству ошибок.
— компилятор Kotlin обнаруживает оставшиеся ошибки, делая код безопасным;
— легкое обучение. Kotlin очень прост в освоении, особенно для Java-разработчиков;
— большое сообщество. Котлин пользуется большой поддержкой и большим вкладом со стороны сообщества, которое растет во всем мире. По данным Google, более 60% из 1000 лучших приложений в Play Store используют Kotlin.

4.2.2 Model-View-ViewModel

Приложение имеет архитектуру, соответствующую шаблону Model-View-ViewModel (MVVM) (рис. 14).

В настоящее время MVVM является наиболее популярным архитектурным решением, применяемым при разработке приложений. Безусловно, хорошо спроектированная архитектура очень важна для того, чтобы обеспечить корректную и длительную работу приложения. [1]

MVVM – это подход, который имеет следующую 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, логику отображения и обработку пользовательских событий;
— 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.

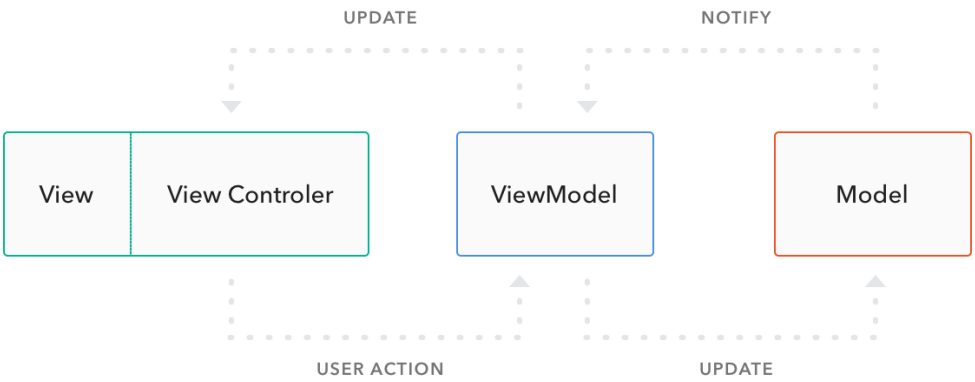

Рисунок 14 - Архитектура Model-View-ViewModel

4.2.3 Data Binding

Однако и при таком подходе возникают некоторые проблемы, например, данные, логика и представление не разделены и находятся в коде

фрагмента или активити. Традиционный подход будет неудобен при разработке приложений со сложной объёмной логикой по тем же причинам, что и при разработке сложных UI решений: много кода приводит к смешиванию логики и представления, из-за чего возникают баги. Для решения этой проблемы мы решили реализовать паттерн MVVM через Data Binding, которая открывает возможности для разделения данных, логики и представления.

Ключевая идея заключается в том, чтобы через databinding привязать объект ViewModel к представлению (через лейаут), специфичные моменты взаимодействия с fragment/activity реализовать через interface (например, смена fragment/activity), а во ViewModel описать всю логику. То есть наша ViewModel выступает прослойкой между Model и View. Таким образом, мы получим гибкую распределённую систему, где каждый элемент играет свою роль и не мешает другому. [2]

4.2.4 Navigation Component

Для разработки навигации в приложении был выбран компонент Navigation. Это комплексное решение всех проблем для любого типа навигации в приложениях Android. Он помогает управлять навигацией, транзакциями фрагментов, бэкстеком, анимацией, глубокими ссылками, а также многим другим. Компонент JetPack Navigation представляет собой набор библиотек и инструментов, сопровождается руководством и обеспечивает продуманную структуру навигации внутри приложения.

Компонент Navigation обеспечивает новый тип навигации в разработке Android, включающей навигационный граф (navigation graph), который позволяет видеть все экраны и пути навигации между ними. [3]

Рассмотрим три главные части компонента Navigation:

— навигационный граф. Это ресурс XML, содержащий всю навигационную информацию в одном централизованном месте. Он включает в себя все отдельные области контента в приложении, называемые пунктами назначения (destinations), а

также возможные пути навигации по приложению, доступные для пользователя;

— NavHost. Это пустой контейнер, отображающий пункты назначения из навигационного графа;

— NavController. Это объект, управляющий навигацией приложения в NavHost. Он координирует смену контента пунктов назначения в NavHost в процессе перемещения пользователя по приложению.

Компонент Navigation обеспечивает еще ряд полезных возможностей, среди которых:

— упрощенная настройка стандартных шаблонов навигации;

— обработка транзакций фрагментов;

— безопасность типов при передаче информации в процессе навигации;

— обработка анимации переходов;

— централизация и визуализация навигации;

— корректная обработка действий “верх” (Up) и “назад” (Back) по умолчанию;

— предоставление стандартизированных ресурсов для анимации и переходов;

— реализация и обработка глубоких ссылок (deep links).

4.2.5 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 (рис. 15).

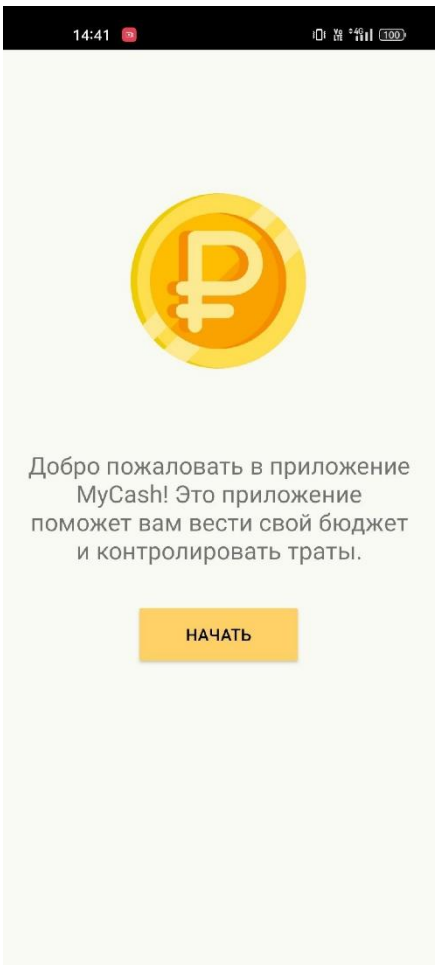

Рисунок 15 - Экран “Приветствие”

4.2.6 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 (рис. 16).

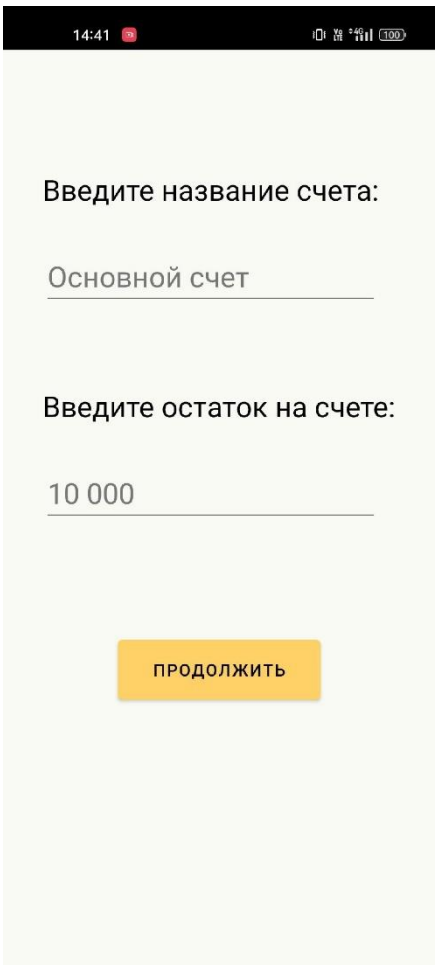

Рисунок 16 - Экран “Начало работы”

4.2.7 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 (рис. 17, 18, 19). Справа внизу находится кнопка “+”, ведущая на экран “Добавление операции”. В нижней панели находится меню, состоящее из вкладок “Аналитика”, “Категории”, “Главная”, “Счета” и “Профиль”. Так же при добавлении новой операции можно воспользоваться календарём (рис. 20).

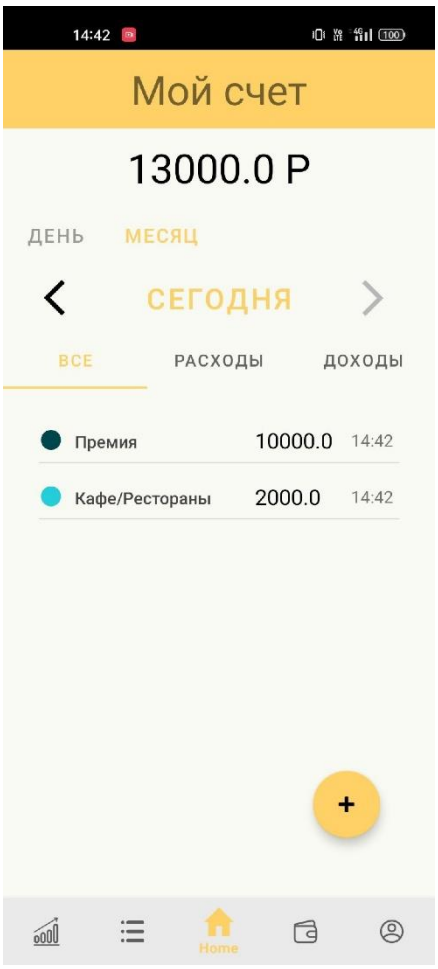

Рисунок 17 - История всех операций за определённый день

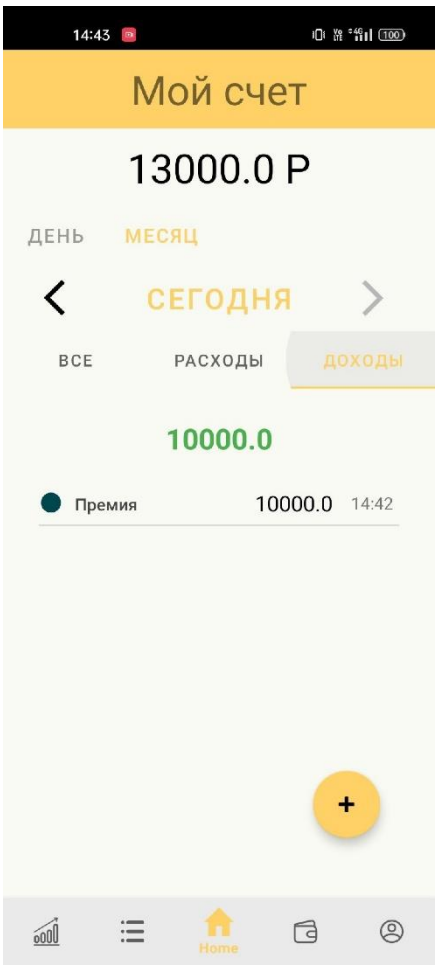

Рисунок 18 - История доходов за определённый день

Рисунок 19 - Прогнозы на следующий месяц

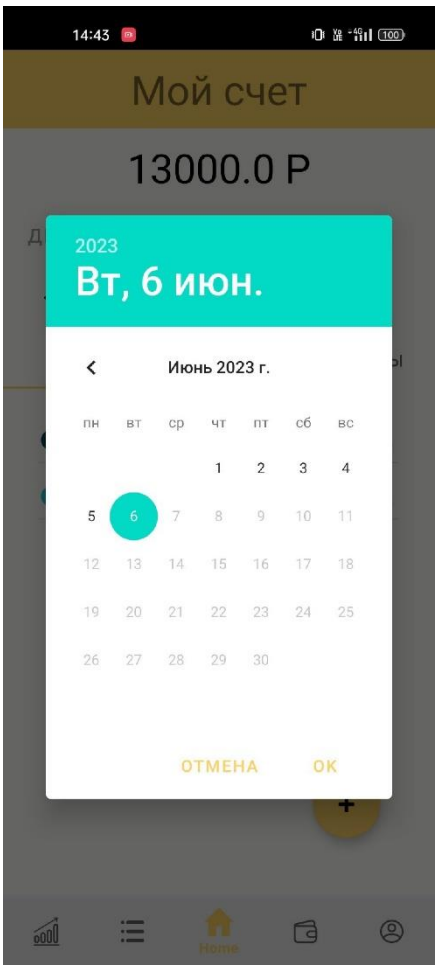

Рисунок 20 - Календарь

4.2.8 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. В конце находится кнопка “Добавить” (рис. 21).

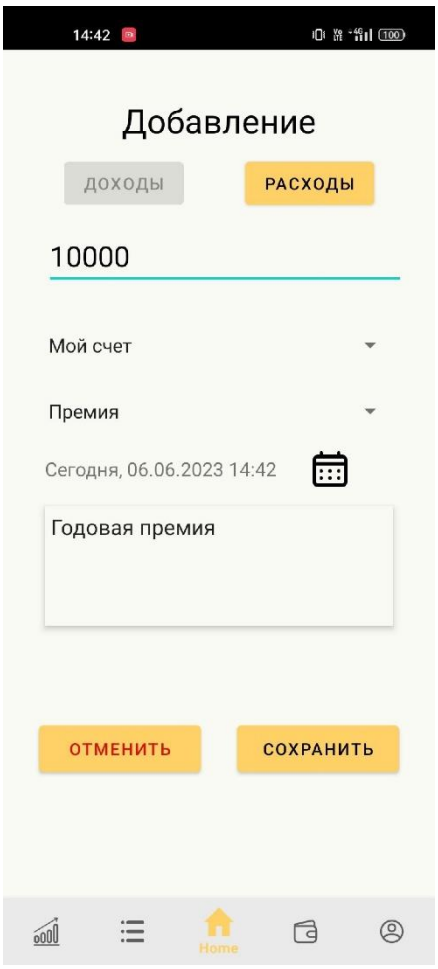

Рисунок 21 - Экран “Добавление операции”

4.2.9 Экран “Профиль”

При переходе на этот экран неавторизованный пользователь увидит предложение зарегистрироваться и две кнопки: “Зарегистрироваться” и “Войти” (рис. 22). 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 настройки профиля, кнопки “Выйти” и “Удалить профиль” (рис. 23).

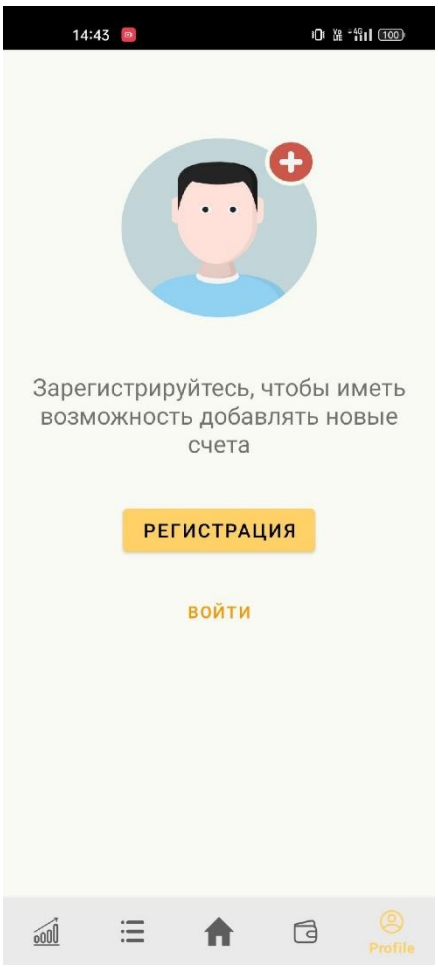

Рисунок 22 - Экран “Профиль” неавторизованного пользователя

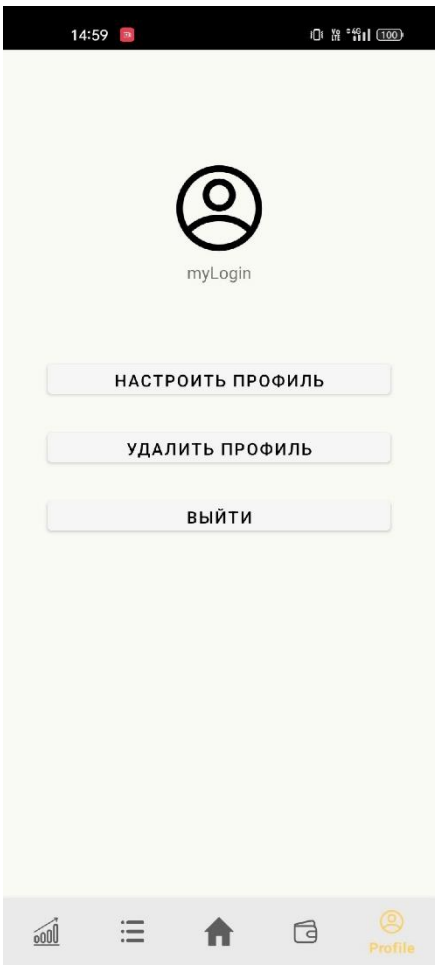

Рисунок 23 - Экран “Профиль” авторизованного пользователя

4.2.10 Экран “Регистрация”

Данный экран предлагает придумать логин и пароль пользователя. (рис. 24). При успешной регистрации выводится сообщение, что пользователь успешно зарегистрирован.

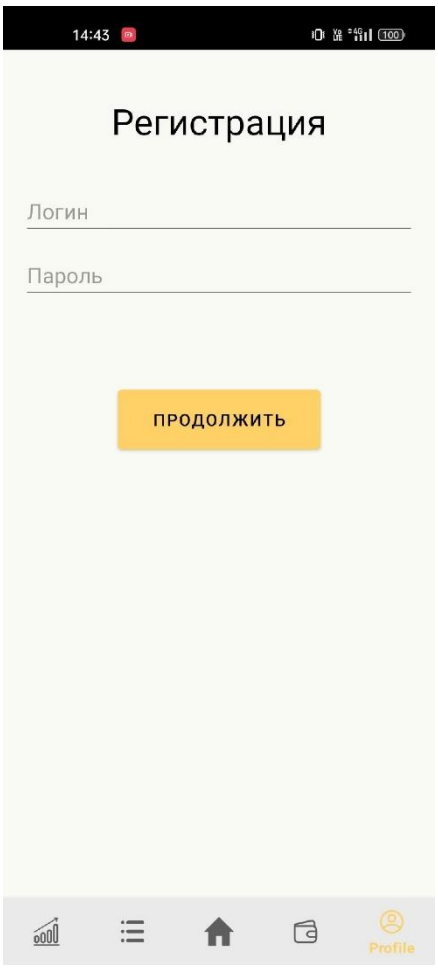

Рисунок 24 - Экран “Регистрация”

4.2.11 Экран “Авторизация”

Данный экран предлагает ввести логин и пароль пользователя (рис. 25). При успешной авторизации осуществляется переход на экран “Профиль” для авторизованных пользователей.

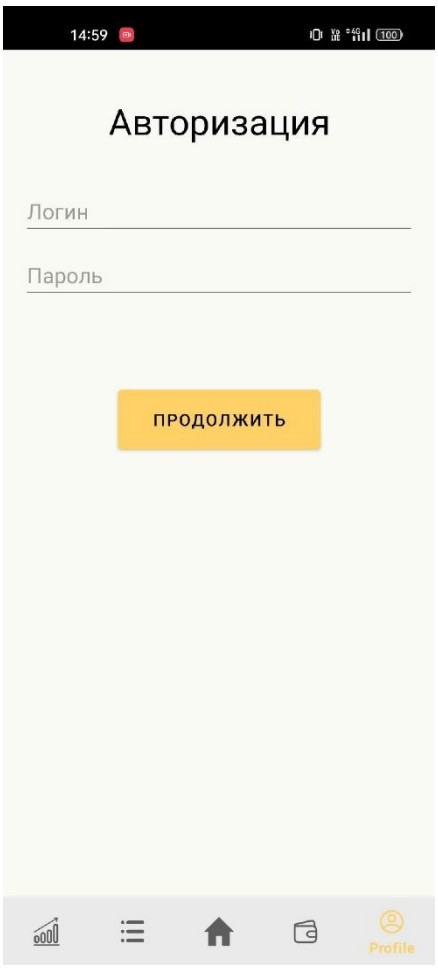

Рисунок 25 - Экран “Авторизация”

4.2.12 Экран “Аналитика”

На данном экране авторизованный пользователь может выбрать счет, аналитику которого желает увидеть, тип графика (общий, только по доходам или только по расходам), и период (по дням или месяцам). Ниже выводится выбранная диаграмма (рис. 26).

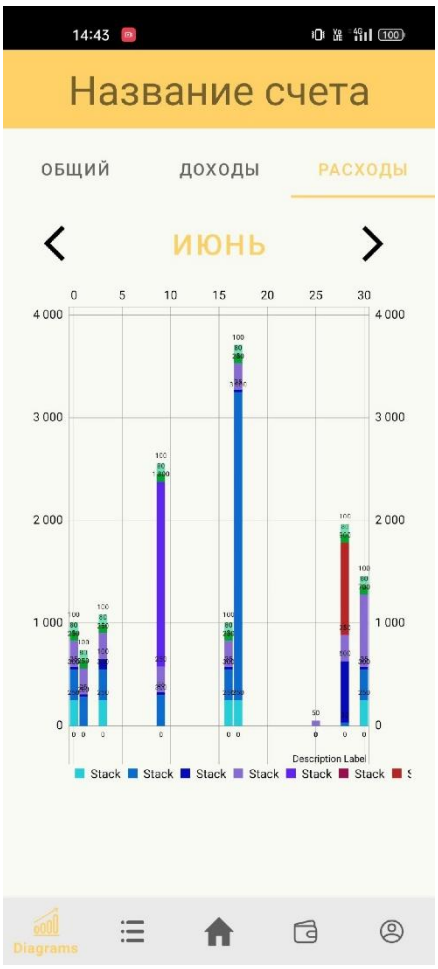

Рисунок 26 - Экран аналитики расходов

4.2.13 Экран “Категории”

При переходе на этот экран пользователь видит список категорий. В верхней панели можно выбрать, какие категории показывать - для доходов или расходов (рис. 27). 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.

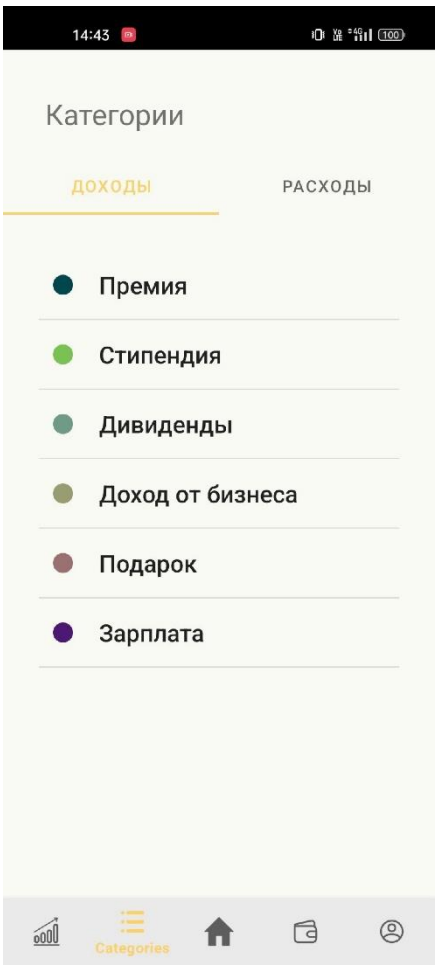

Рисунок 27 - Экран “Категории”

4.2.14 Экран “Счета”

На данном экране находится список всех счетов пользователя. Ниже располагается кнопка “+”, позволяющая добавить новую категорию (рис. 28). 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 (рис. 29).

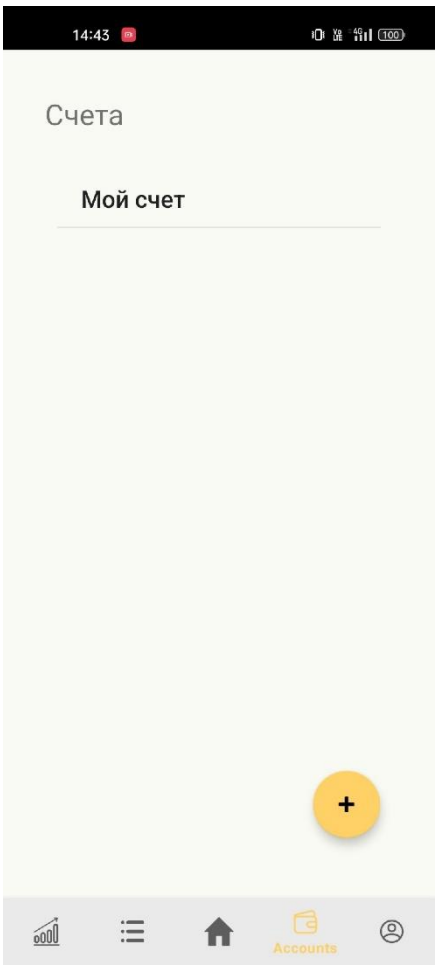

Рисунок 28 - Экран “Счета” (для авторизованного пользователя)

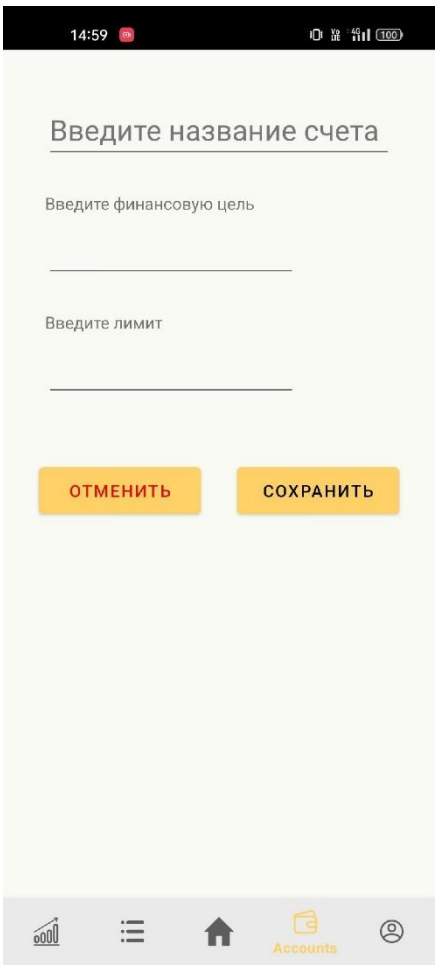

Рисунок 29 - Экран “Добавление счета”

4.2.15 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 личного кабинета ведет на экран “Профиль”.

4.3 Разработка backend-части

4.3.1 MVC

Приложение “MyCash” строится на базе архитектуры, называемой Model-View-Controller (MVC). Согласно принципам этой архитектуры каждый объект приложения должен быть объектом модели, объектом представления или объектом контроллера.

Шаблон MVC описывает простой способ построения структуры приложения, целью которого является отделение бизнес-логики от пользовательского интерфейса. В результате, приложение легче масштабируется, тестируется, сопровождается и конечно же реализуется. [4]

На рисунке 30 представлена концептуальная схема шаблона MVC.

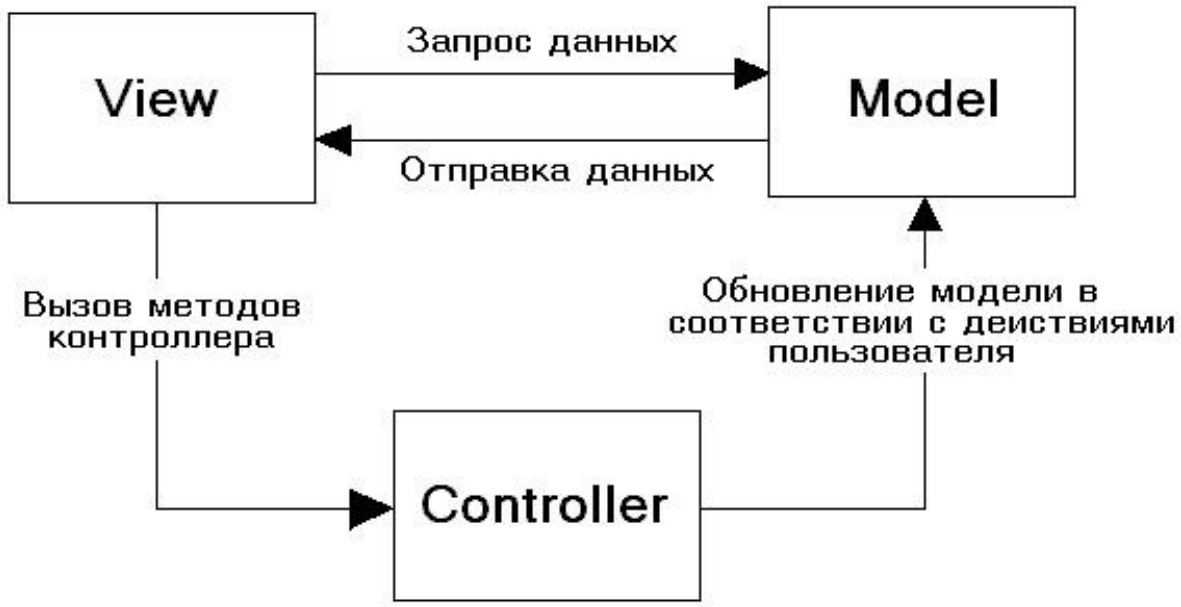

Рисунок 30 - Схема шаблона MVC

4.3.2 Spring Boot

Также был использован фреймворк Spring Boot. Выбор данного фреймворка обусловлен тем, что Spring Boot:

- автоматически конфигурирует проекты на основе одного из стартовых пакетов для них;
- облегчает создание и развертывание приложений на Spring;
- быстро и легко управляет зависимостями и подгружает необходимые модули;
- поддерживает встроенный сервер для запуска приложений;
- может автоматически создать и настроить базу данных для приложения.

Spring Boot следует многоуровневой архитектуре, в которой каждый уровень взаимодействует с уровнем (слоем) непосредственно ниже или выше него (иерархическая структура). [5]

Архитектура Spring Boot строится на четырёх уровнях (рис. 34):

— Presentation Layer. Presentation Layer обрабатывает HTTP-запросы, преобразует параметр JSON в объект, аутентифицирует запрос и передает его на бизнес-уровень.

— Business Layer. Business Layer обрабатывает всю бизнес-логику. Он состоит из классов обслуживания и использует службы, предоставляемые уровнями доступа к данным. Также выполняет авторизацию и проверку;

— Persistence Layer. Spring data позволяет сохранять классы в базу данных. Persistence Layer содержит всю логику хранения и транслирует бизнес-объекты из строк базы данных и в них;

— Database Layer. На уровне базы данных выполняются операции CRUD (создание, извлечение, обновление, удаление).

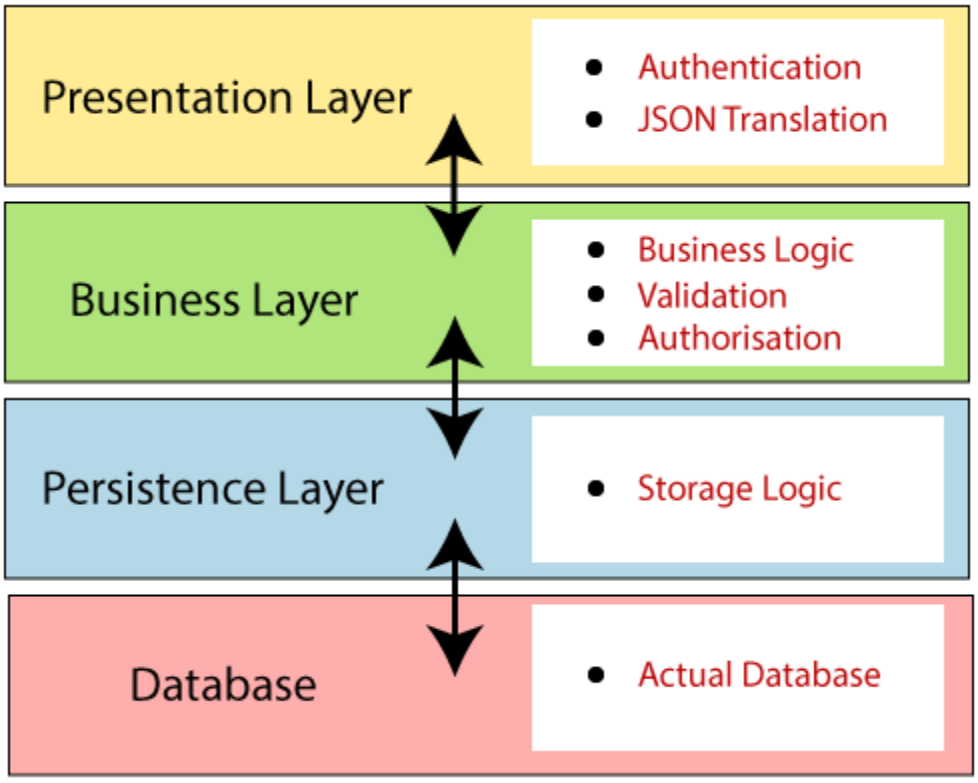

Рисунок 31 - Уровни Spring Boot

Spring Boot использует все модули Spring-подобных Spring MVC, Spring Data и т. д. Архитектура Spring Boot такая же, как и архитектура Spring MVC, за исключением одного: в Spring boot нет необходимости в классах DAO и DAOImpl. Создается уровень доступа к данным и выполняется операция CRUD. Клиент делает HTTP-запросы (PUT или GET). Запрос поступает к контроллеру, а контроллер отображает этот запрос и обрабатывает его. После этого он при необходимости вызывает логику службы. На сервисном уровне выполняется вся бизнес-логика. Он выполняет логику данных, которые отображаются в JPA с классами моделей. Страница JSP возвращается пользователю, если не произошло никаких ошибок (рис. 35). [6]

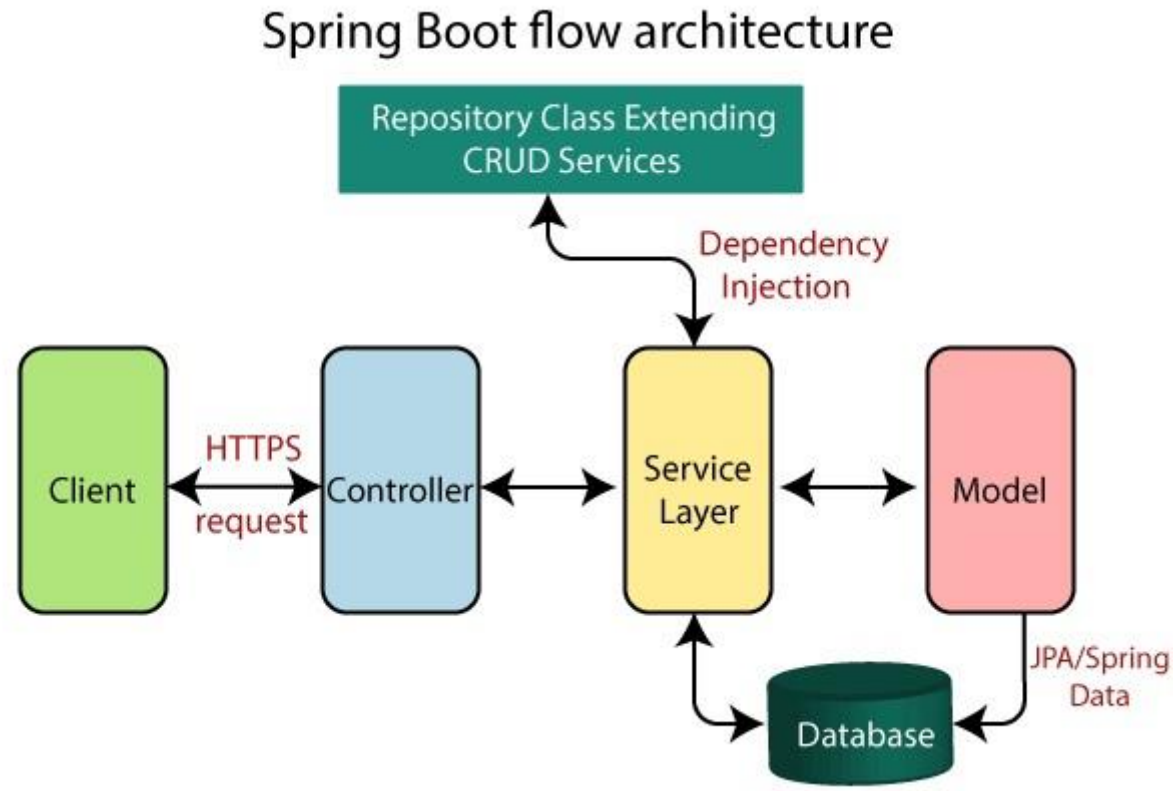

Рисунок 32 - Архитектура Spring Boot

Программный код мобильного приложения MyCash полностью соответствует принципам SOLID.

5 Заключение

В данной курсовой работе была рассмотрена разработка системы управления домашним бюджетом с рекомендациями по сокращению расходов. Была проведена аналитическая работа, определены требования к системе, произведен выбор технологий и инструментов для реализации проекта.

В результате работы была создана система, позволяющая пользователю управлять своими расходами, а также получать рекомендации по сокращению расходов на основе анализа статистики. Разработанная система легко настраивается под индивидуальные потребности пользователя.

В процессе работы были применены принципы SOLID, а также архитектуры MVC и MVVM, что обеспечило гибкость и расширяемость системы. Также были использованы современные технологии и инструменты разработки, что позволило создать качественный продукт.

В заключении можно сказать, что разработанная система является полезной и практичной для пользователей, которые хотят вести учет своих расходов и сократить свои затраты. Дальнейшее развитие системы может включать в себя добавление новых функций и интеграцию с другими приложениями.

Список используемой литературы

1. Patterns - WPF Apps With The Model-View-ViewModel Design Pattern / Josh Smith [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern – (Дата обращения: 20.05.2023).
2. Реализация паттерна MVVM на Android через Data Binding / Владимир Иванов (IT-копирайтер) [Электронный ресурс]. – Режим доступа: https://www.azoft.ru/blog/mvvm-android-data-binding/. – Заглавие с экрана. – (Дата обращения: 22.05.2023).
3. Навигация для Android с использованием Navigation Component / KotlinStudio [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dual-screen/android/api-reference/dualscreen-library/navigation-component/?tabs=kotlin – (Дата обращения: 22.05.2023).
4. Что такое архитектура MVC: Model-View-Controller / Мерион нетворкс. – 2022. [Электронный ресурс]. – Режим доступа: https://wiki.merionet.ru/servernye-resheniya/100/chto-takoe-arhitektura-mvc-model-view-controller/ – (Дата обращения: 25.05.2023).
5. Spring Boot 101: Введение в создание веб-приложений / Machine learning [Электронный ресурс]. – Режим доступа: https://vc.ru/u/1389654-machine-learning/586955-spring-boot-101-vvedenie-v-sozdanie-veb-prilozheniy – (Дата обращения: 25.05.2023).
6. Spring Boot Architecture [Электронный ресурс]. – Режим доступа: https://www.javatpoint.com/spring-boot-architecture. – (Дата обращения: 25.05.2023).
7. Программирование под Android на Java / Metanit.com. [Электронный ресурс]. – Режим доступа: https://metanit.com/java/android/ – (Дата обращения: 28.05.2023).

8. Идентификация, Аутентификация, Авторизация / ProQualityCommunity. [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/720842/ – (Дата обращения: 23.05.2023).
9. Android SDK / skillfactory media. [Электронный ресурс]. – Режим доступа: https://blog.skillfactory.ru/glossary/android-sdk/ – (Дата обращения: 23.05.2023).
10. Приложения для учета финансов / Екатерина Надежкина – 19.08.2021. [Электронный ресурс]. – Режим доступа: https://daily.afisha.ru/money/20691-9-luchshih-prilozheniy-dlya-ucheta-finansov/ – (Дата обращения: 24.05.2023).