

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

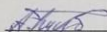
Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

СЕРВИС УЧЕТА ЛИЧНЫХ РАСХОДОВ

Студент


(подпись, дата) 14.06.23

Трубаненко А. Г.
(ФИО полностью)

Руководитель


(подпись, дата) 15.06.23

д.т.н. Зинкин С. А.
(фамилия, инициалы)

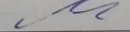
Нормоконтролёр


(подпись, дата) 15.06.23

Бычков А.С.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2023 г. № 14)

Заведующий кафедрой

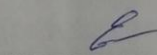

(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

16.06.2023г. № 19)

Секретарь ГЭК


(подпись)

Е.И. Гурин

(фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«26» апреля 2023 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Трубаненко Александр Георгиевич гр. 19ВВП2 факультета ФВТ

2. Тема работы Сервис учета личных расходов.

Тема утверждена приказом ПГУ № 458/о от "26" апреля 2023 г.

3. Руководитель бакалаврской работы д.т.н. Зинкин С.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

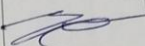
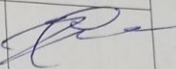
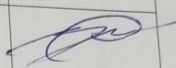
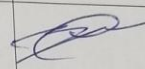
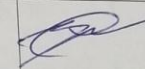
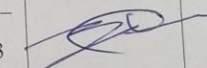
Разработать Сервис по учету личных расходов. Язык программирования Kotlin, среда разработки Android Studio. Программа должна обладать интуитивно понятным интерфейсом и использовать следующие технологии:

1. Android SDK
2. Room
3. Retrofit
4. MPAndroid Chart
5. Moshi
6. Currency API

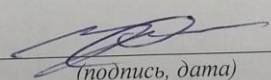
5. Перечень подлежащих разработке вопросов

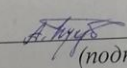
Требуется произвести анализ предметных областей: изучить существующие информационные системы, выделить их достоинства и недостатки. Выявить потребности пользователей. Выбрать язык программирования, среду разработки, доступные библиотеки. Определить архитектуру приложения и вспомогательные средства разработки. Изучить функционал библиотек для разработки мобильного приложения.

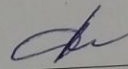
Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	26.04.2023 – 28.05.2023	
2	Анализ требований	32	29.04.2023 – 30.05.2023	
3	Создание макета графического интерфейса	30	31.04.2023 – 3.05.2023	
4	Выбор технологий, применяемых при проектировании и разработке	40	4.05.2023 – 8.05.2023	
5	Реализация графического интерфейса	66	9.05.2023 – 15.05.2023	
6	Разработка основной части программы	70	16.05.2023 – 23.05.2023	
7	Оформление пояснительной записки	54	24.05.2023 – 31.05.2023	

Дата выдачи задания "26" апреля 2023 г.

Руководитель бакалаврской работы  д.т.н. Зинкин С. А.
(подпись, дата)

Задание к исполнению принял студент  Трубаненко А.Г.
(подпись, дата)

Работу к защите допустить
Декан факультета  16.06.2023 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 118 листа, 22 рисунка, 3 таблиц, 4 источника

СЕРВИС, УЧЕТ РАСХОДОВ, УПРАВЛЕНИЕ РАСХОДАМИ, ROOM, БАЗА ДАННЫХ, CURRENCYAPI, KOTLIN, RETROFIT, JETPACK COMPOSE, ANDROID SDK

Цель работы – разработать инструмент, позволяющий вести учет личных расходов.

Одним из путей достижения поставленной задачи является разработка мобильного приложения под операционную систему Android, обеспечивающая выполнение следующих функций: авторизация пользователей, добавление счетов, редактирование счетов, удаления счетов, добавление и категоризация расходов, фильтрация информации по дате, графическое отображение расходов.

В итоге разработан Сервис по учету личных расходов, который соответствует цели данной выпускной квалификационной работы. В ходе достижения результата, были использованы следующие технологии: Kotlin, Retrofit, MPAndroidChart, Room, SQLite, Currency API. Приложение протестировано и готово к эксплуатации.

					ПГУ1.090301.36.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Трубаненко А.Г.			Лит.	Лист	Листов
Провер.		Зинченко С.А.				2	129
Н. Контр		Бычков А.С.			ФВТ, каф. ВТ гр. 19ВВП2		
Утверд.		Митрохин М.А.					

Сервис учета личных расходов
Пояснительная записка

Содержание

Перечень сокращений и обозначений.....	4
Введение.....	5
1 Сервис по учету личных расходов	7
1.1 Анализ предметной области	7
1.1.2 Постановка задачи	8
1.1.3 Анализ существующих аналогов	9
1.1.4 Выбор языка программирования.....	12
1.1.5 Выбор инструментов разработки	12
1.1.5.1 Выбор среды разработки	12
1.1.5.2 Выбор библиотеки для работы с базой данных	13
1.1.5.3 Выбор библиотеки для создания графических диаграмм.....	15
1.1.5.4 Выбор библиотек для работы с веб-сервисами.....	15
1.1.6 Архитектура приложения.....	16
1.1.7. Разработка графического интерфейса	17
1.1.7.1. Разработка графического интерфейса.....	17
1.1.7.2. Создание стилей XML	19
1.1.7.5 Проектирование базы данных	22
1.1.8 Описание приложения	24
1.1.8.1 Руководство пользователя.....	24
2 Безопасность жизнедеятельности.....	35
2.1 Правила безопасности при использовании смартфона.....	35
Заключение	37
Список используемых источников.....	38
Приложение А - UML диаграммы.....	39
Приложение Б – Листинг программы	44
Приложение В - Презентация	116

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения
БД (база данных) - структурированная коллекция данных, организованных и хранимых в электронном виде

СУБД (система управления базами данных) - программное обеспечение, предназначенное для создания, управления и обработки баз данных

ЯП (язык программирования) - формальный язык, используемый для написания компьютерных программ, которые определяют последовательность действий, выполняемых компьютером

ЯП (язык программирования) - формальный язык, используемый для написания компьютерных программ, которые определяют последовательность действий, выполняемых компьютером

API (Application Programming Interface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой

DAO (Data Access Object) - то объект доступа к данным (Data Access Object), который является одним из популярных шаблонов проектирования и используется, чтобы изолировать бизнес-модели от способа доступа к данным.

IDE (Integrated development environment) – интегрированная среда разработки

SQL (Structured Query Language) – язык структурированных запросов

UML (Unified Modeling Language) – унифицированный язык моделирования

XML (eXtensible Markup Language) – это расширяемый язык разметки, используется для создания структурированных документов и данных для обмена.

Введение

Современный образ жизни требует от нас умения эффективно управлять своими финансами. Однако в мире, где потребности и расходы постоянно растут, становится все сложнее контролировать свои личные финансы. Поэтому важно иметь инструменты, которые помогут нам принять во внимание и анализировать наши расходы, чтобы совершать обоснованные финансовые решения.

В рамках данного дипломного проекта мы рассмотрим создание сервиса учета личных расходов, который предоставит пользователю возможность систематически отслеживать и анализировать свои финансовые траты. Этот сервис будет разработан с учетом современных технологий и требований пользователей, чтобы обеспечить удобство использования и максимальную функциональность.

Целью данного проекта является создание надежного инструмента, который поможет пользователям контролировать свои расходы, устанавливать бюджеты, оптимизировать свои финансовые решения и достигать финансовой устойчивости. Сервис учета личных расходов будет предоставлять возможность удобного ввода и категоризации расходов, автоматического анализа данных, создания отчетов и визуализации информации.

В ходе работы над проектом будут рассмотрены существующие подходы к учету личных расходов, проанализированы требования пользователей, разработаны архитектура и интерфейс сервиса, а также проведено тестирование его функциональности. Мы также ознакомимся с методами анализа данных и визуализации, которые позволят пользователям получать полезные и понятные отчеты о своих финансах.

Ожидается, что разработанный сервис учета личных расходов станет незаменимым инструментом для пользователей, помогая им принимать осознанные финансовые решения, избегать лишних трат и добиваться финансовой стабильности. Также он может быть использован финансовыми

консультантами и организациями, предоставляющими услуги в сфере финансового планирования.

В дальнейшем разработка сервиса учета личных расходов может быть расширена и улучшена с учетом обратной связи пользователей и новых технологических возможностей, что позволит еще больше упростить и оптимизировать процесс управления личными финансами.

					ПГУ1.090301.36.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

1 Сервис по учету личных расходов

1.1 Анализ предметной области

В настоящее время существует множество *Android*-приложений, предназначенных для учета личных расходов. Эти приложения помогают пользователям вести систематический учет своих финансов, анализировать расходы по категориям, устанавливать бюджеты и получать полезные отчеты о своих финансовых операциях.

При анализе *Android*-приложения по учету расходов важно оценить его функциональность, удобство использования, дизайн и возможности интеграции с другими финансовыми инструментами. Вот несколько ключевых аспектов, которые можно рассмотреть при анализе подобного приложения:

Функциональность: какие функции доступны в приложении. Включает ли оно возможность создания категорий расходов, установки бюджетов, импорта данных из других источников, генерации отчетов и анализа финансовых данных. Хорошие приложения обычно предлагают широкий спектр функций, а также возможность настройки и персонализации.

Удобство использования: насколько просто и интуитивно понятно приложение. Доступность и понятность пользовательского интерфейса, удобство ввода данных и навигации между разделами — все это играет важную роль в опыте использования приложения.

Дизайн: оцените визуальный дизайн и оформление приложения. Хорошие приложения имеют привлекательный и современный дизайн, сочетающийся с удобством использования. Интуитивное расположение элементов, понятные иконки и приятные цветовые схемы могут сделать использование приложения более приятным.

Интеграция и безопасность: есть ли возможность интеграции с другими финансовыми инструментами, такими как банковские счета или электронные

кошельки. Кроме того, обратите внимание на меры безопасности, предпринятые разработчиками для защиты финансовых данных пользователей.

Обратная связь пользователей: отзывы и оценки приложения от других пользователей в магазинах приложений или на специализированных платформах. Это может дать представление о том, насколько приложение популярно, эффективно и надежно.

Важно отметить, что анализ конкретного *Android*-приложения по учету расходов должен быть проведен с учетом потребностей и предпочтений конкретного пользователя. Различные приложения могут предлагать разные функции и решать разные задачи, поэтому выбор приложения должен соответствовать индивидуальным потребностям и целям пользователя.

1.1.2 Постановка задачи

Необходимо разработать мобильное приложение для учета личных расходов. Приложение должно позволять пользователю вести учет своих финансов, добавлять новые траты, указывать дату, сумму и категорию расходов. Также, приложение должно предоставлять статистику о расходах, графики и диаграммы по категориям затрат.

Разработка пользовательского интерфейса:

- создание привлекательного и интуитивно понятного дизайна приложения, сочетающего удобство использования с функциональностью;
- разработка навигации между разделами приложения, обеспечивающей быстрый и удобный доступ к основным функциям.

Реализация функциональности приложения:

- создание механизма для добавления и категоризации расходов пользователем. Пользователь должен иметь возможность вводить информацию о своих расходах, указывать категорию, дату и сумму операции;
- реализация автоматического анализа данных и генерации отчетов. Приложение должно предоставлять пользователю возможность просмотра сводных отчетов о расходах по категориям, времени и другим

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

параметрам. Также желательно предоставить графическую визуализацию данных для более наглядного анализа;

Обеспечение безопасности и конфиденциальности:

- реализация мер безопасности для защиты финансовых данных пользователей. Это может включать шифрование данных, защиту от несанкционированного доступа и обеспечение конфиденциальности личной информации.

Тестирование и отладка:

- проведение тестирования приложения на различных устройствах и разрешениях экранов, чтобы обеспечить совместимость и стабильность работы приложения;
- исправление обнаруженных ошибок и недочетов в работе приложения для повышения его качества и надежности.

Ожидаемый результат:

Разработанное *Android*-приложение по учету личных расходов, которое обеспечивает пользователей инструментами для систематического учета и анализа финансовых операций, установки бюджетов и принятия обоснованных финансовых решений. Приложение должно быть стабильным, безопасным и удобным в использовании.

1.1.3 Анализ существующих аналогов

Среди аналогов сервиса учета личных расходов несколько популярных приложений в Play Market:

- мои расходы;
- *money manager*.

Рассмотрим приложение “Мои расходы”.

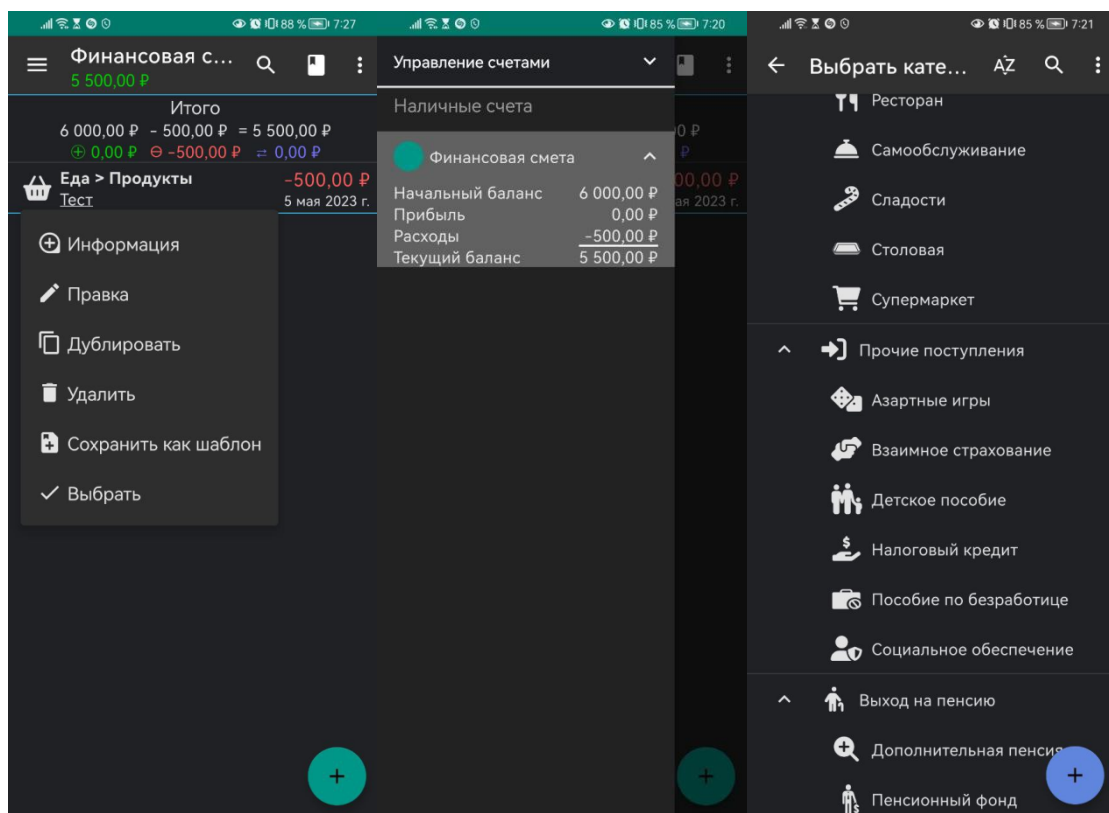


Рисунок 1- Приложение “Мои расходы”

Достоинства данного приложения:

- учет доходов;
- есть финансовая смета.

Недостатки:

- неудобное управление;
- устаревший и громоздкий интерфейс;
- нет диаграмм или графиков;
- нет фильтров по категориям или счетам.

Рассмотрим приложение “Money Manager”.

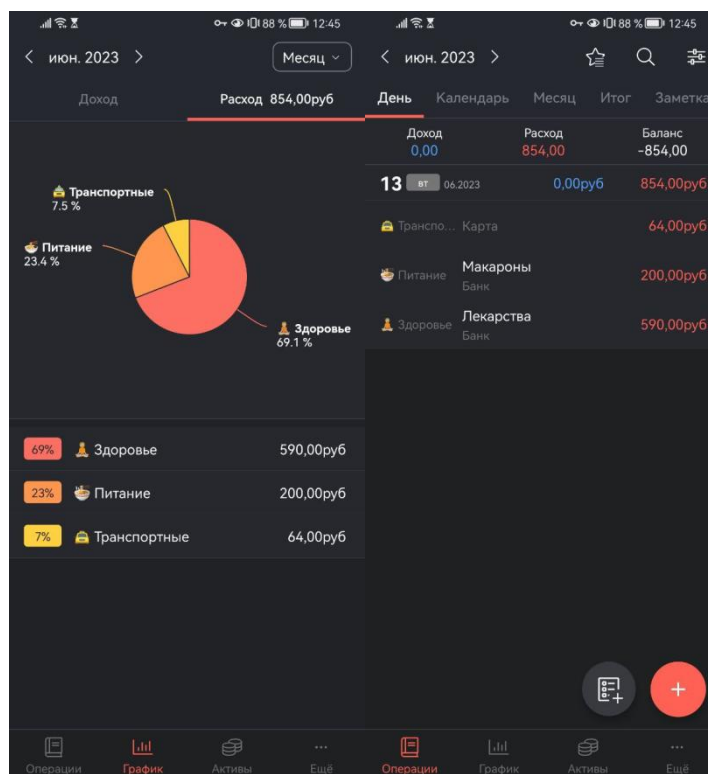


Рисунок 2 – Приложение “Money Manager”

Достоинства данного приложения:

- учет доходов;
- есть диаграммы;
- есть заметки;
- есть повтор ежемесячных затрат.

Недостатки:

- плохо оптимизированно;
- есть ограничение на создание счетов.

1.1.4 Выбор языка программирования

1.1.4.1 Язык программирования *Kotlin*

Kotlin позволяет создавать приложения для различных платформ, включая мобильные устройства, настольные компьютеры и веб-сайты. Этот язык программирования выполняет все основные задачи, которые могут быть необходимы при программировании, в том числе управление памятью, многопоточность, работа с базами данных и многим другим.

Преимущества использования *Kotlin*:

- код на *Kotlin* более безопасен и читабелен, чем на Java;

Этот язык позволяет быстрее и проще создавать программы и приложения;

- большая гибкость и легкая расширяемость кода;

- *Kotlin* имеет хорошую совместимость с Java, что упрощает передачу проектов на *Kotlin*.

1.1.5 Выбор инструментов разработки

1.1.5.1 Выбор среды разработки

Android Studio — это официальная интегрированная среда разработки (IDE) для разработки приложений под платформу *Android*. Она предоставляет разработчикам все необходимые инструменты и функциональность для создания высококачественных *Android*-приложений. Вот несколько причин, почему выбор *Android Studio* может быть обоснованным:

- Официальная поддержка *Google*: *Android Studio* разработана и поддерживается напрямую *Google*, создателем операционной системы *Android*. Благодаря этому *Android Studio* всегда получает своевременные обновления и интеграцию с последними версиями *Android API* и инструментами.

- Интегрированная разработка: *Android Studio* предлагает все необходимые инструменты и функции для разработки *Android*-приложений в

одном месте. Это включает в себя редактор кода, инструменты разработки пользовательского интерфейса, систему сборки, отладчик и эмулятор устройств *Android*. Такая интеграция облегчает и ускоряет процесс разработки.

- Эмуляторы и устройства: *Android Studio* предоставляет мощные эмуляторы *Android*-устройств, которые позволяют запускать и тестировать приложения на различных версиях *Android* и разрешениях экранов. Кроме того, *Android Studio* легко интегрируется с физическими устройствами *Android*, позволяя разработчикам тестировать приложения на реальных устройствах.

- Поддержка языков программирования: *Android Studio* поддерживает различные языки программирования, такие как *Java* и *Kotlin*. Вы можете выбрать предпочтительный язык или даже использовать комбинацию обоих. *Kotlin*, являющийся официальным языком разработки *Android*, имеет особые преимущества и интеграцию с *Android Studio*.

- обширная документация и сообщество: Будучи популярным инструментом разработки под *Android*, *Android Studio* имеет обширную документацию и активное сообщество разработчиков.

- расширяемость: *Android Studio* имеет мощную систему плагинов, позволяющую расширять функциональность и интегрировать сторонние инструменты. Вы можете настроить *Android Studio* под свои потребности, добавив необходимые плагины для разработки, отладки или управления проектом.

- *Android Studio* является хорошим выбором для разработки *Android*-приложений благодаря своей официальной поддержке, обширным возможностям и удобству использования, что делает процесс разработки более эффективным и продуктивным.

1.1.5.2 Выбор библиотеки для работы с базой данных

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		13

Библиотека *Room* — это часть *Android Architecture Components*, предоставляющая удобные инструменты для работы с базами данных в приложениях *Android*. Она позволяет разработчикам легко создавать, управлять и взаимодействовать с *SQLite*-базами данных, используя объектно-реляционное отображение (*ORM*).

Основные компоненты библиотеки *Room* включают:

- *Entity* (сущность): *Entity* представляет таблицу базы данных. Каждый объект *Entity* представляет отдельную запись в таблице. Разработчик может определить *Entity* с помощью аннотаций, указывая его поля и столбцы базы данных.
- *DAO* (*Data Access Object*): *DAO* предоставляет абстракцию для доступа к данным в базе данных. Разработчик может объявить методы в *DAO*, которые определяют операции для вставки, обновления, удаления или выборки данных из базы данных. *DAO* также может содержать запросы, написанные на языке *SQL* или использовать *Query*-аннотации для определения запросов.
- *Database* (база данных): *Database* является центральным компонентом библиотеки *Room*. Разработчик может создать абстрактный класс, расширяющий класс *RoomDatabase*, и аннотировать его с помощью аннотации *@Database*. В этом классе определяются *Entity* и версия базы данных. *Room* автоматически генерирует реализацию *Database*, предоставляя доступ к *DAO* и управление версиями базы данных.

Преимущества использования библиотеки *Room*:

- простота использования: *Room* предоставляет простой и понятный *API* для работы с базами данных. Разработчикам необходимо определить только *Entity* и *DAO*, и большая часть работы по управлению базой данных будет автоматически выполнена *Room*.

- компиляционная проверка: *Room* осуществляет компиляционную проверку SQL-запросов и обнаруживает ошибки на этапе компиляции, что позволяет избежать ошибок времени выполнения.

- улучшенная производительность: благодаря использованию *SQLite*, *Room* обеспечивает высокую производительность при выполнении операций с базой данных. Он также предлагает возможность использовать асинхронные запросы для избежания блокирования основного потока приложения.

- абстракция от *SQLite*: *Room* предоставляет абстракцию от прямого использования *SQLite*-команд и позволяет разработчику сосредоточиться на логике приложения, а не на деталях работы с базой

1.1.5.3 Выбор библиотеки для создания графических диаграмм

MPAndroidChart – это мощная и гибкая библиотека для создания графических диаграмм на платформе *Android*. Она имеет множество настроек и параметров, которые позволяют настроить график по своим потребностям. Кроме того, библиотека обеспечивает высокую производительность и включает в себя множество функций, что делает ее идеальным выбором для создания интерактивных графиков и диаграмм для приложений *Android*.

1.1.5.4 Выбор библиотек для работы с веб-сервисами

Retrofit и *Moshi* - это библиотеки для работы с веб-сервисами в приложениях на *Android*. Хотя они выполняют разные функции, обе они позволяют удобно осуществлять взаимодействие с удаленными серверами и преобразовывать данные в формат *JSON*.

Retrofit является библиотекой для работы с *API*, которая помогает преобразовывать *HTTP API* в клиентские приложения. Она обеспечивает простой и удобный *API* для выполнения запросов, автоматическую сериализацию и десериализацию данных и поддержку асинхронных операций.

Moshi, с другой стороны, является библиотекой для разбора и сериализации данных *JSON*. Она предоставляет быстрый и эффективный способ преобразования *JSON* в объекты *Java* или *Kotlin* и наоборот. *Moshi* также поддерживает аннотации для настройки сериализации и десериализации по вашим потребностям.

1.1.6 Архитектура приложения

Архитектура приложения является одним из важных элементов разработки программного обеспечения. В процессе разработки была выбрана *MVVM*-архитектура (*Model-View-ViewModel*), которая является популярной архитектурой для построения клиентских приложений.

MVVM-архитектура состоит из трех ключевых элементов: модель (*Model*), представление (*View*) и представление модели (*ViewModel*). Каждый из этих элементов имеет свои задачи и ответственности в приложении.

Модель представляет собой бизнес-логику приложения. Она содержит данные и методы для работы с этими данными. Модель не знает о существовании представления и представления модели.

Представление — это то, что пользователь видит на экране. Оно использует данные из модели для отображения информации на экране. Представление не должно содержать никакой логики приложения, оно отвечает только за отображение данных.

Представление модели является связующим звеном между моделью и представлением. Он обеспечивает связь между данными из модели и отображением этих данных в представлении. Он также содержит логику, которая не относится к представлению или модели.

MVVM-архитектура представляет собой простую, но мощную архитектуру для построения клиентских приложений. Она позволяет легко разделить логику приложения на модель, представление и представление модели. Это повышает качество кода, облегчает тестирование и упрощает сопровождение кода.

Цель *MVVM* – разделить приложение на 3 независимых слоя:

- *View – Activity/Fragment*
- *Model – работа с бизнес логикой (работа с базой данных SQLite, FireBase)*
- *VeiwModel – Связующее звено View с Model, а также хранит данные для View.*

На рисунке 4 показаны компоненты, использующиеся в архитектуре *MVVM* и отображено взаимодействие между ними.

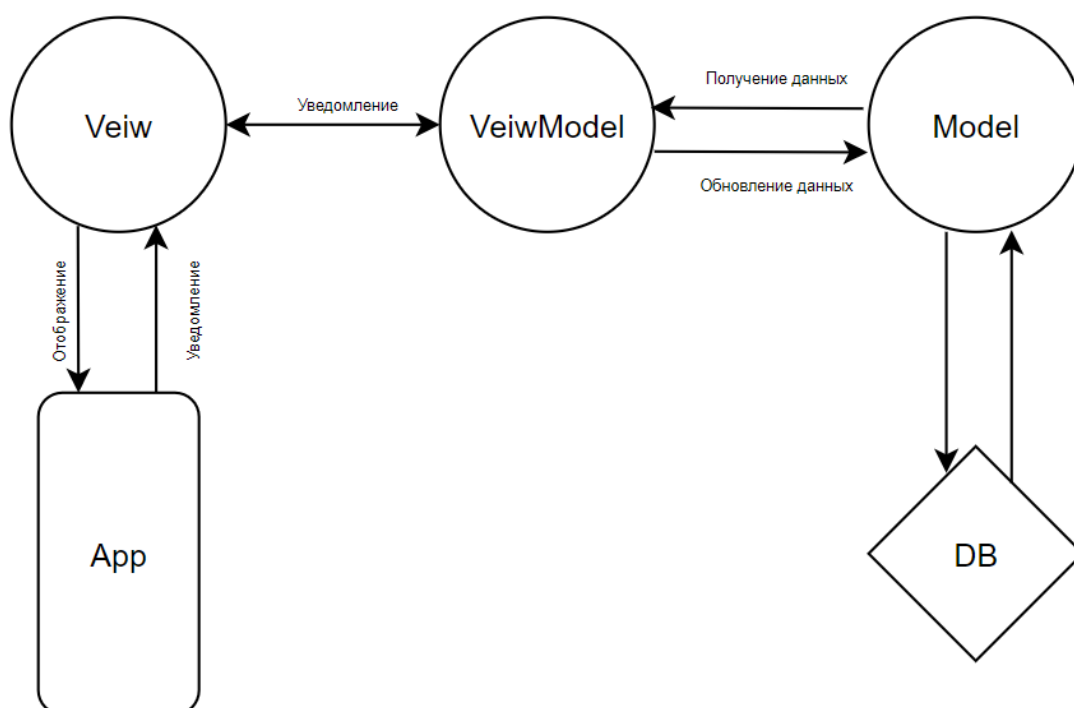


Рисунок 4 – *MVVM* Архитектура

1.1.7. Разработка графического интерфейса

1.1.7.1. Разработка графического интерфейса

Графический интерфейс предоставляет возможности для взаимодействия с приложением в соответствии с UML-диаграммой использования, показанной на рисунке 5.

Графический интерфейс состоит из разных *View*, каждое из которых предназначено для определённого взаимодействия с пользователем.

AccountAddViewModel – представление, позволяющие пользователю добавлять новый счет.

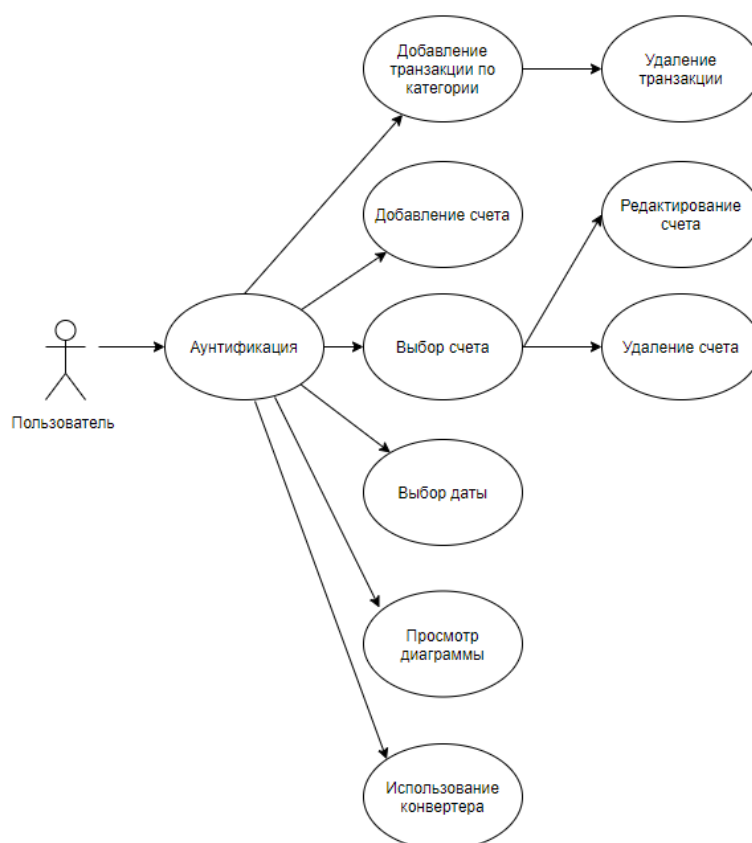


Рисунок 5 – UML-диаграмма использования

AccountEditViewModel – представление, позволяющие пользователю редактировать счет.

AccounViewModel – представление, отображающие пользовательские счета.

AuthViewModel – представление, позволяющие пользователю авторизоваться в приложение при помощи PIN – кода.

ChartViewModel – представление, отображающие диаграмму с информацией о транзакциях.

CurrencyConverterViewModel – представление, позволяющие пользователю использовать конвертер валют только с подключенным интернетом и только с валютами евро, доллар США и Российский рубль.

SelectDateViewModel – представление, позволяющие пользователю выбирать дату отображения транзакций.

AccountFilterViewModel – представление, позволяющие пользователю выбирать конкретный счет или все счета для отображения совершенных транзакциях на выбранных (-ом) счетах (-е).

AddTransactionViewModel – представление, позволяющие пользователю добавлять транзакцию.

SelectCategoryViewModel – представление, позволяющие пользователю выбирать категорию для записи расходов в выбранную категорию.

TransactionsViewModel – представление, использующиеся для отображения транзакций на экране.

1.1.7.2. Создание стилей XML

Стили определяют внешний вид пользовательского интерфейса такие как цвета, шрифты, размеры и другие атрибуты и интерфейса один раз, а затем использовать их во многих местах в приложении.

С помощью стилей можно создавать единообразный дизайн для всего приложения, что позволяет сохранять единую концепцию дизайна на всех экранах, что улучшает восприятие пользователями приложения. Кроме того, использование стилей сокращает количество дублированного кода и упрощает поддержку приложения.

В процессе разработки были созданы словари ресурсов для каждого графического элемента, использующегося в приложении.

colors.xml – содержит все ресурсы цветов использующихся в приложении.

Strings.xml – содержит все строки использующихся в приложении.

Styles.xml – содержит цвета, размеры и шрифты использующихся в приложении.

1.1.7.3 Диалоговые окна

Диалоговые окна — это интерактивные компоненты пользовательского интерфейса, которые позволяют отображать важные сообщения, запросы или дополнительные варианты действий на экране устройства. Они используются для оповещения пользователя о произошедших событиях или для получения необходимых данных перед выполнением действия. Диалоговые окна могут быть различных типов: алерт-диалоги, диалоги выбора даты и времени и т.д. Они обеспечивают удобство и простоту взаимодействия пользователя с приложением

В приложении используются два диалоговых окна, для которых имеется свой класс *View* (графическая часть), *ViewModel* (взаимодействие с графической частью).

SelectDateDialogFragment – диалоговое окно предназначенное для выбора даты отображения транзакций.

AccauntFilterDialogFragment – диалоговое окно предназначенное для выбора счета.

Графический интерфейс диалоговых окон представлен ниже на рисунках 6,7.

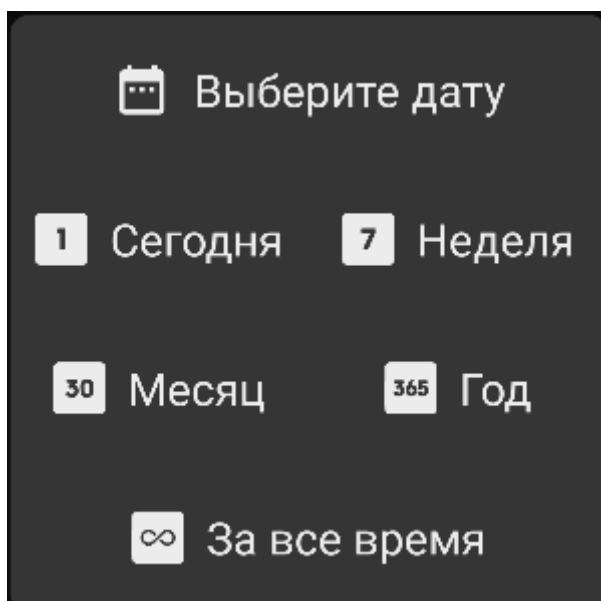


Рисунок 6 - Диалоговое окно с выбором даты

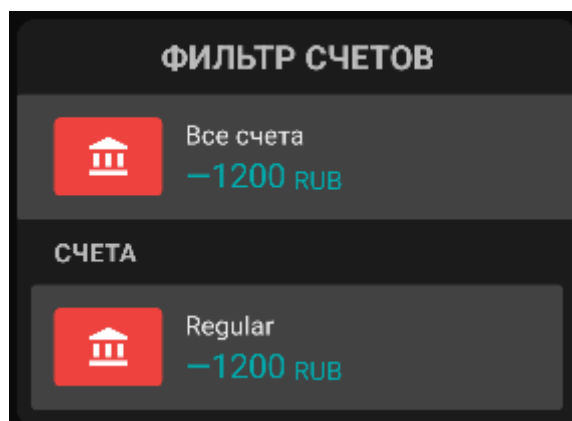


Рисунок 7 - Диалоговое окно с выбором счета

1.1.7.4 Фрагменты

Фрагменты — это модули пользовательского интерфейса, которые совместно с активностью (Activity) образуют интерфейс приложения. Фрагменты похожи на маленькие активности, которые могут быть использованы внутри других активностей.

Основное назначение фрагментов заключается в создании гибких и переносимых элементов пользовательского интерфейса, которые могут быть заменены, перемещены или объединены с другими фрагментами в зависимости от потребностей приложения или устройства пользователя. Каждый фрагмент имеет свой собственный жизненный цикл, который связан с жизненным циклом активности, в которой он находится.

В приложении используются фрагменты, описанные ниже.

AccountAddFragment — используется для добавление нового счета.

AccountEditFragment — используется для редактирование уже существующего счета.

AccountsFragment — используется для отображения существующих счетов.

ChartFragment — используется для отображения диаграммы.

AddTransactionSheetFragment — используется для добавления транзакции.

SelectCategorySheetFragment – используется для выбора категории перед добавлением транзакций.

TransactionsFragment – используется для отображения транзакций.

1.1.7.5 Проектирование базы данных

Для хранения информации необходимо реализовать базу данных со следующими таблицами:

- Accounts – таблица, содержащая счета пользователя;
- Categories – таблица, содержащая категории расходов;
- Transactions – таблица, содержащая информацию о транзакциях пользователя.

Атрибуты вышеперечисленных сущностей представлены в таблицах 1–3.

Таблица 1 - Атрибуты таблицы *Account*.

Имя столбца	Тип данных	Описание	Роль атрибута
<i>ID</i>	<i>INT NOT NULL</i>	Идентификатор счета пользователя	РК
<i>Name</i>	<i>String</i>	Название счета	
<i>Amount</i>	<i>Double</i>	Сумма счета	
<i>Color</i>	<i>String</i>	Цвет счета	

Таблица 2 – Атрибуты таблицы *Categories*.

Имя столбца	Тип данных	Описание	Роль атрибута
<i>ID</i>	<i>INT NOT NULL</i>	Идентификатор категории	РК
<i>Name</i>	<i>String</i>	Название категории	
<i>Icon</i>	<i>Int</i>	Иконка категории	
<i>IconColor</i>	<i>String</i>	Цвет иконки категории	

Таблица 3 – Атрибуты таблицы *Transaction*.

Имя столбца	Тип данных	Описание	Роль атрибута
<i>ID</i>	<i>INT NOT NULL</i>	Идентификатор транзакций пользователя	PK
<i>Note</i>	<i>String</i>	Заметка транзакции	
<i>Amount</i>	<i>Double</i>	Сумма с	
<i>Date</i>	<i>Date</i>	Дата транзакции	
<i>Time</i>	<i>Time</i>	Время транзакции	
<i>AccountID</i>	<i>Int</i>	Идентификатор счета пользователя	FK
<i>CategoryID</i>	<i>Int</i>	Идентификатор категории	FK

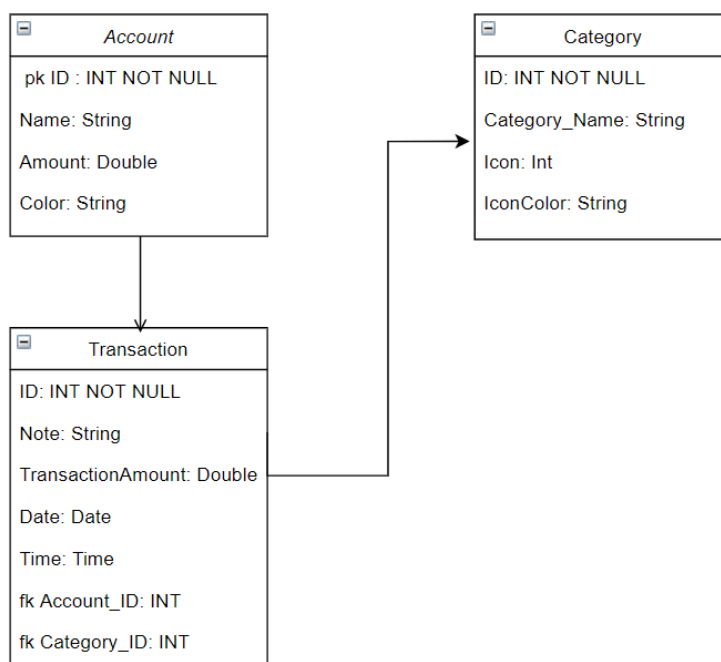


Рисунок 3 – Физическая модель базы данных

1.1.8 Описание приложения

1.1.8.1 Руководство пользователя

После запуска приложения откроется экран авторизации пользователя. Окно содержит набор цифр для ввода PIN-кода, маску пароля, кнопку стереть и кнопку “Забыли PIN?”. Интерфейс окна авторизации показан на рисунке 8.

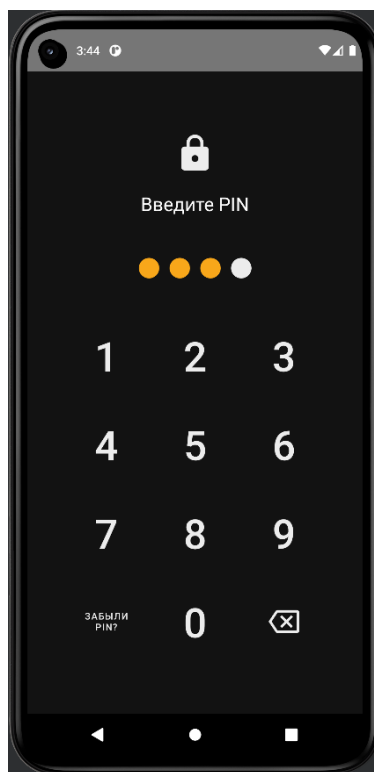


Рисунок 8 – Окно авторизации

В случае успешной авторизации пользователь попадает на экран “Операции”. В верхней части экрана находится кнопка “Настройки” изображена в виде шестеренки, а также кнопки выбора счета и даты при нажатии которых открываются соответствующие фрагменты. В основной части экрана отображена информация о транзакциях. В нижней части экрана находится навигационное меню и кнопка добавления транзакции. Графический интерфейс экрана и фрагментов выбора счета и даты, а также добавление транзакции представлен на рисунках 9–14.

При нажатии на кнопку “Счета” откроется фрагмент, где отображены ранее созданные счета, смотрите рисунок 10.

При нажатии на кнопку выбора даты пользователь может выбрать за какой период или указать конкретный день отобразить информацию о транзакциях, смотрите рисунок 11.

При нажатии на кнопку добавлении транзакции откроется фрагмент, на котором представлен список категорий, пользователю надо выбрать категорию для дальнейшего ввода расхода и информации выбранной категории, смотрите рисунок 12.

После нажатия на выбранную категорию откроется фрагмент, где пользователю предстоит ввести расход и добавить заметку(необязательно) после этого нажать кнопку потвердить, смотрите рисунок 13.

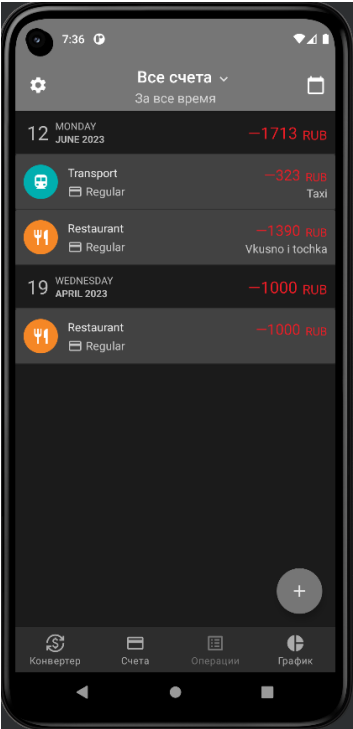


Рисунок 9 – Экран “Операции”

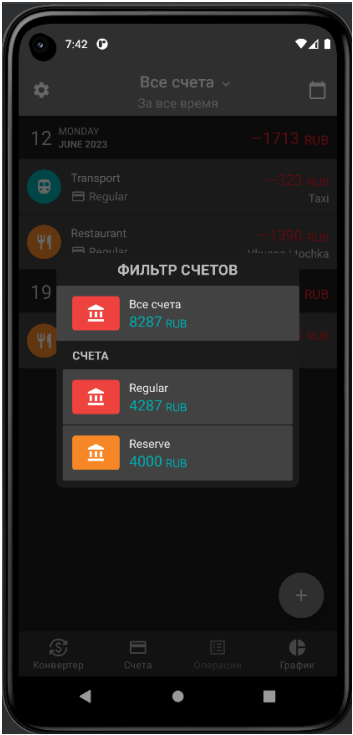


Рисунок 10– Фильтр счетов

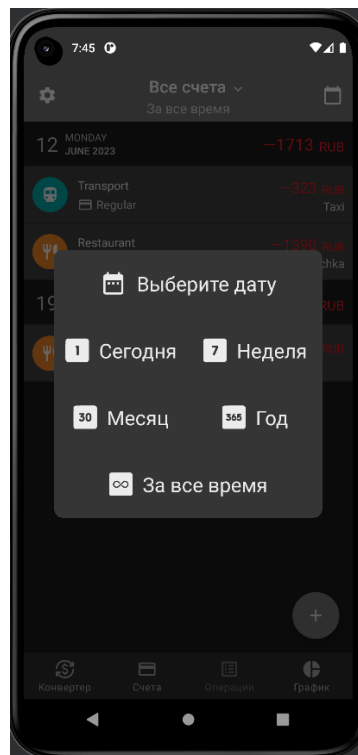


Рисунок 11 – Выбор даты

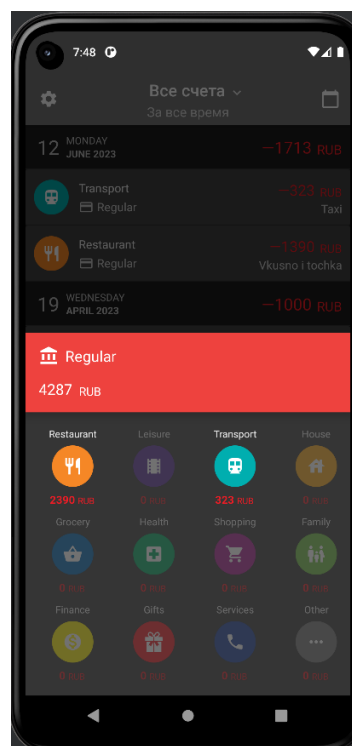


Рисунок 12 – Добавление транзакции

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		27

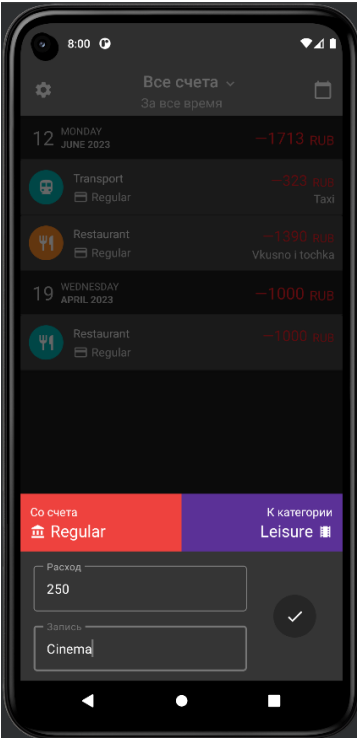


Рисунок 13 – Добавление транзакции

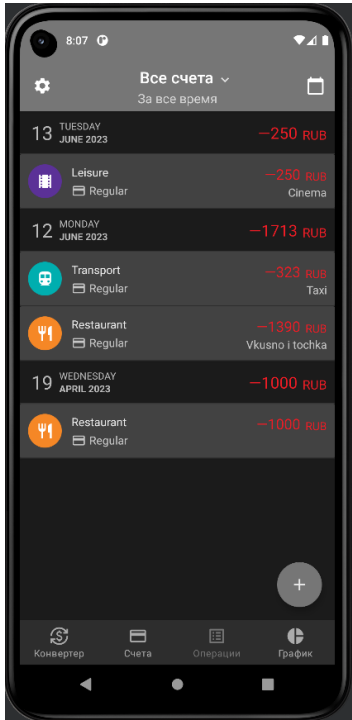


Рисунок 14 – Демонстрация ранее добавленной транзакции

Для редактирования или добавление/удаление счета пользователю требуется перейти на экран “Счета” нажав на соответствующую кнопку в навигационном меню. В верхней части экрана находится кнопка “Настройки” изображена в виде шестеренки, а также кнопка добавления нового счета. На главной части экрана отображен список счетов. Для удаления или редактирование конкретного счета, пользователю требуется кликнуть на счет после чего внизу экрана появится фрагмент с действиями по счету. Графический интерфейс экрана и фрагментов представлен на рисунках 15–19.

После нажатия пользователем на кнопку добавить счет откроется фрагмент, где надо будет ввести имя счета и сумму, а также выбрать цвет, которым будет отображаться счет затем подтвердить действия, смотрите рисунок 16.

При нажатии на кнопку “Удалить” появится диалоговое окно для подтверждения действий, смотрите рисунок 18.

При нажатии на кнопку “Изменить” откроется фрагмент с ранее виденной информацией по счету, которую пользователь может редактировать, смотрите рисунок 19.

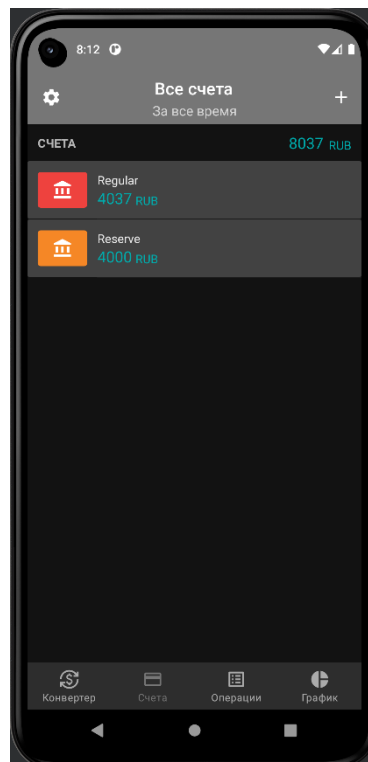


Рисунок 15 – Экран “Счета”

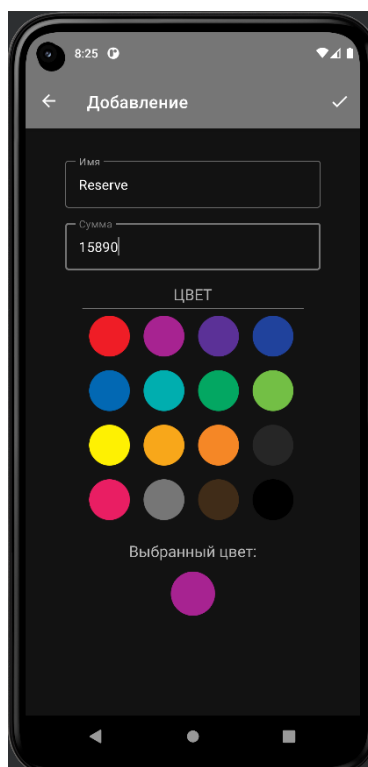


Рисунок 16 – Добавление счета

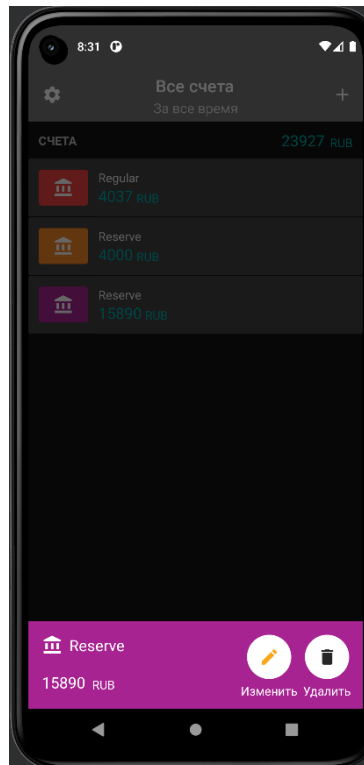


Рисунок 17- Выбор действий над счетом

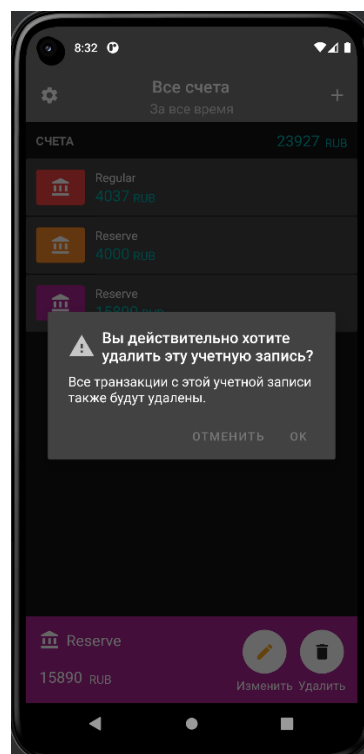


Рисунок 18 – Удаление счета

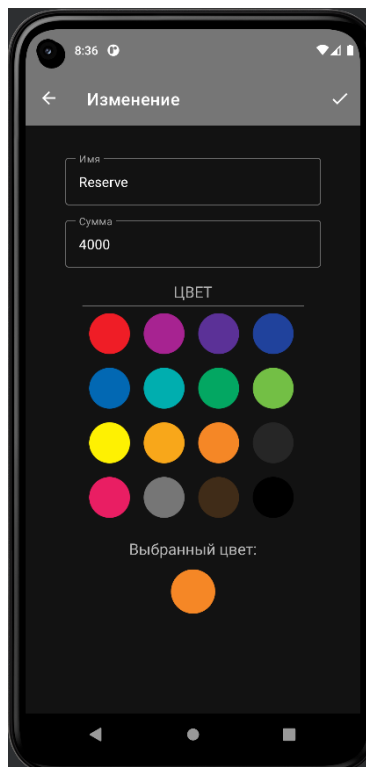


Рисунок 19 – Редактирование счета

Чтобы посмотреть наглядную статистику о расходах пользователь может перейти на экран “График” при нажатии на соответствующую кнопку навигационного меню. В верхней части экрана находится кнопка “Настройки” изображена в виде шестеренки, а также кнопки выбора счета и даты при нажатии которых открываются соответствующие фрагменты. В главной части экрана содержится круговая диаграмма и список категорий. Категории, по которым не совершались никакие транзакции затемнены. Пользователь может выбрать за какой период времени отображать информацию и/или выбрать конкретный счет. Графический интерфейс представлен на рисунках 20, 21.



Рисунок 20 – Экран “Трафик”



Рисунок 21 – Отображения информации

Для использования конвертера валют в приложении пользователь требуется перейти на экран “Конвертер” с помощью навигационного меню. В верхней части экрана находится кнопка “Настройки” изображена в виде шестеренки, а также кнопки выбора счета. На главной части экрана отображен функционал конвертера. Пользователь может ввести сумму, а также выбрать валюту из списка (Рубль, Доллар, Евро) для конвертации. Графический интерфейс представлен на рисунке 22.

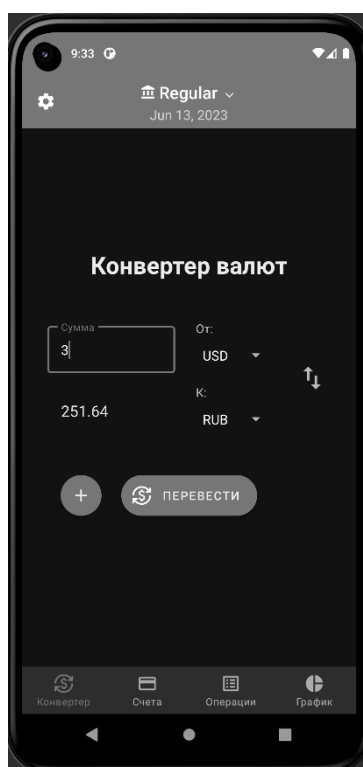


Рисунок 22 – Экран “Конвертер”

2 Безопасность жизнедеятельности

2.1 Правила безопасности при использовании смартфона

В этом разделе следует уделить внимание рискам, связанным с безопасностью при использовании данного устройства. А также разработать свод правил, чтобы гарантировать предотвратить следующие риски:

– риск зависимости и негативного воздействия на психологическое и физическое здоровье:

1) чрезмерное использование смартфона, включая зависимость от социальных сетей, игр и постоянного онлайн-присутствия, что может негативно сказываться на здоровье и социальной активности.

2) использование смартфона перед сном, из-за светового излучения экрана, может нарушать естественный сон и вызывать бессонницу;

3) чрезмерное использование смартфона в темное время суток может снижать уровень мелатонина, гормона сна;

– риск потери или кражи смартфона:
несанкционированное получение доступа к устройству, возможность потери или кражи смартфона, что может привести к утечке личных данных или использованию их в недобросовестных целях;

нарушение конфиденциальности и безопасности данных в случае потери или кражи смартфона;

– риск воздействия радиочастотного излучения:
возможные воздействия радиочастотного излучения на организм при длительном использовании смартфона, хотя точные последствия такого воздействия все еще находятся в стадии исследования;

при работе в плохо покрытых районах мобильной связи смартфон может выделять больше энергии для подключения к сети, что может привести к повышенному радиочастотному излучению;

- использование смартфона во время вождения может отвлекать водителя и повышать риск дорожно-транспортных происшествий. Это особенно актуально в случае чтения сообщений, отправки сообщений или проведения телефонных разговоров во время движения.

С учетом указанных рисков следует разработать и рекомендовать следующие правила безопасности при использовании смартфона:

- ограничьте время использования смартфона, особенно перед сном;
- соблюдайте эргономику и правильное положение тела при работе со смартфоном;
- используйте оригинальные зарядные устройства и кабели, избегайте перегрева и короткого замыкания;
- устанавливайте надежные пароли и используйте функции блокировки экрана для защиты данных;
- храните смартфон в безопасном месте и будьте бдительны в общественных местах;
- регулярно делайте резервные копии данных и используйте антивирусное программное обеспечение для защиты от вредоносных программ;
- ограничьте использование смартфона во время вождения и соблюдайте правила безопасности на дороге.

Правильное соблюдение этих правил поможет минимизировать риски и обеспечить безопасное использование смартфона.

Заключение

Результат выполнения данной выпускной квалификационной работы можно считать успешно реализованным в соответствии с заданием.

Были выполнены следующие задачи:

- выбор языка программирования. Был выбран *Kotlin* в качестве языка программирования для разработки мобильного приложения. *Kotlin* обладает простым и понятным синтаксисом, множеством библиотек, что облегчает и ускоряет разработку мобильного приложения;
- выбор системы управления базами данных. Была выбрана *SQLite* в качестве СУБД для хранения и управления данными. *SQLite* - это простая *in-process* (встроенная) система управления базами данных, которая позволяет использовать базы данных в приложениях без необходимости отдельного сервера для управления ими;
- разработка функционала. Были реализованы основные модули мобильного приложения, такие как авторизация пользователя, создание и фильтрация счетов, добавление расходов по категориям, отображения транзакций за определенный период и счет, был реализован конвертер валют с помощью *CurrencyAPI*.

В результате работы был создан Сервис учета личных расходов, который предоставляет удобный и быстрый доступ для записи своих расходов.

Перспективы развития системы включают расширение функционала, добавление новых модулей, улучшение пользовательского интерфейса.

Список использованных источников

1. Скин Д., Гринхол Д., Бэйли Э. " Kotlin. Программирование для профессионалов. 2-е изд ", 2023. – 560с.
2. Habr.com.: [Электронный ресурс]. URL: <https://habr.com/ru/articles/713518> (дата обращения 27.05.2023)
3. Developer.android.com.: [Электронный ресурс]. URL: <https://developer.android.com/docs> (дата обращения 5.04.2023)
4. Developer.android.com.: [Электронный ресурс]. URL: <https://developer.android.com/design/ui/mobile/guides/foundations/system-bars> (дата обращения 5.04.2023)

Приложение А

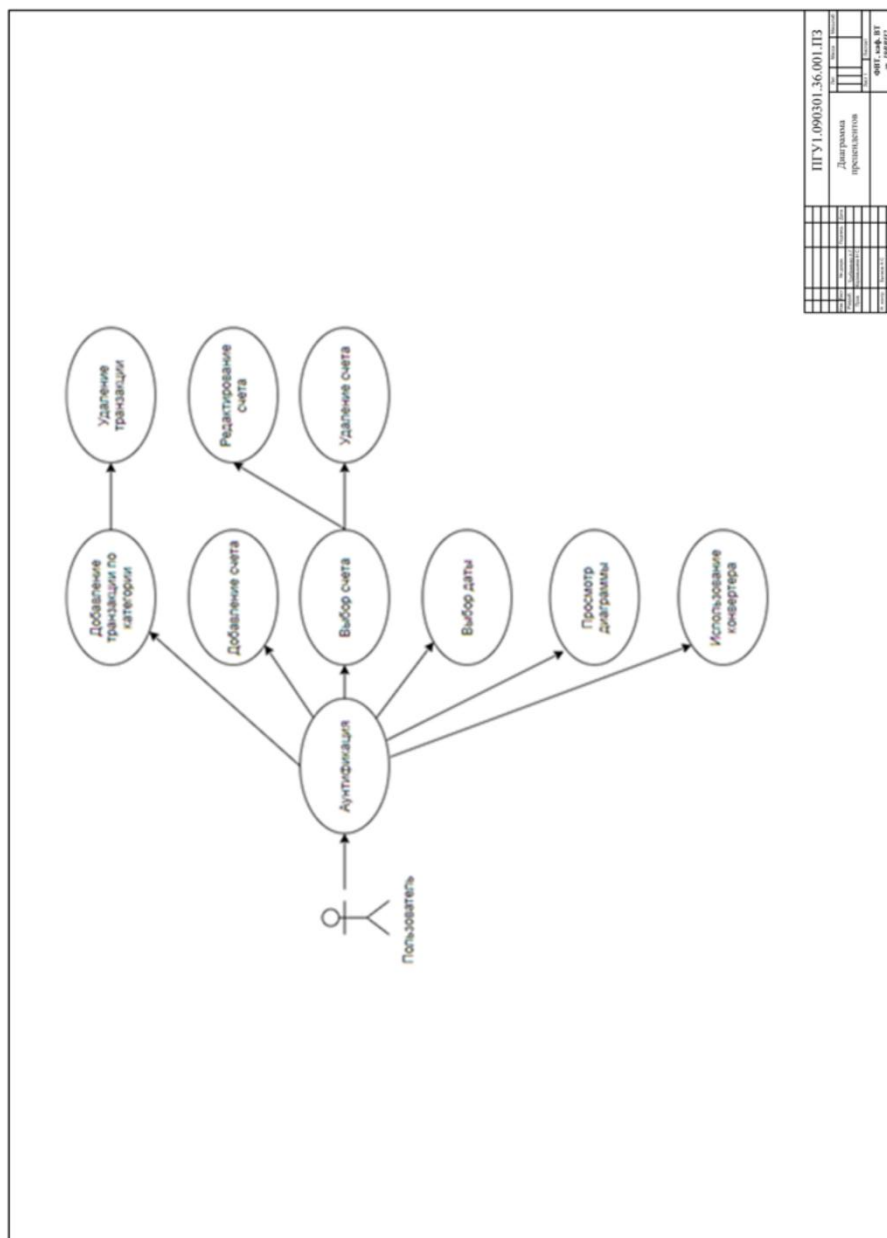
(обязательное)

Сервис по учету личных расходов.

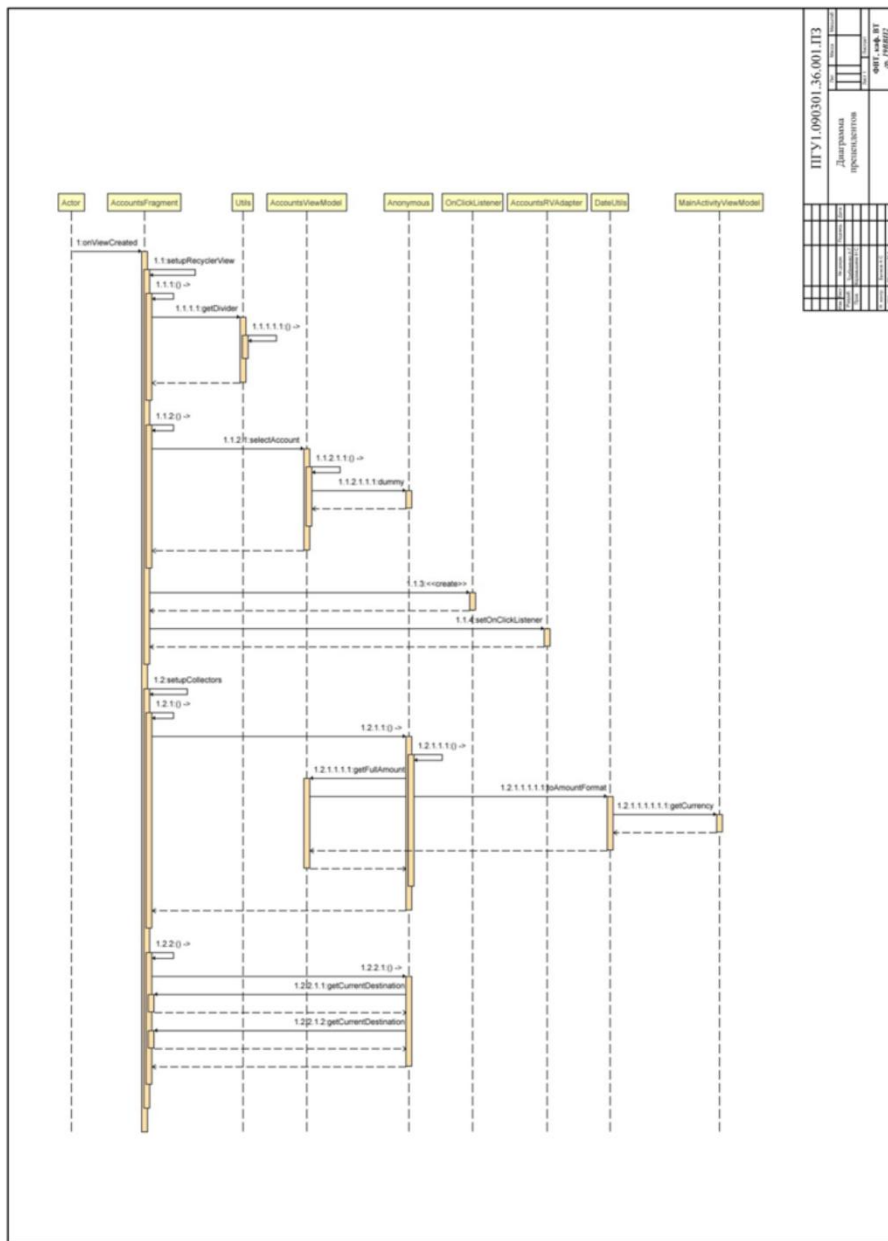
UML диаграммы

					ПГУ1.090301.36.001 ПЗ	Лист
						39
Изм.	Лист	№ докум.	Подпись	Дата		

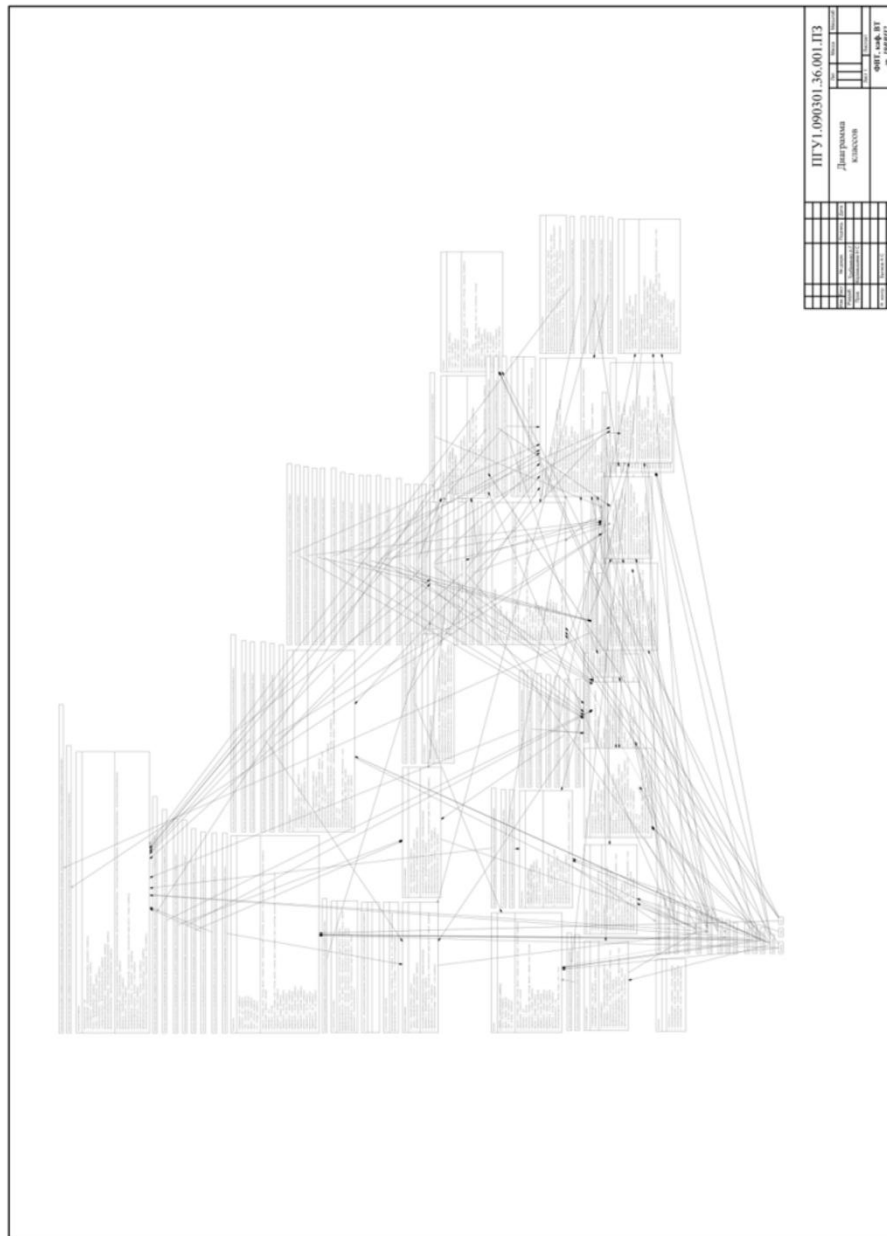
Приложение А.1 – *UML* – диаграмма прецедентов



Приложение А.2 – UML – диаграмма последовательности



Приложение А.3 – *UML* – Диаграмма классов



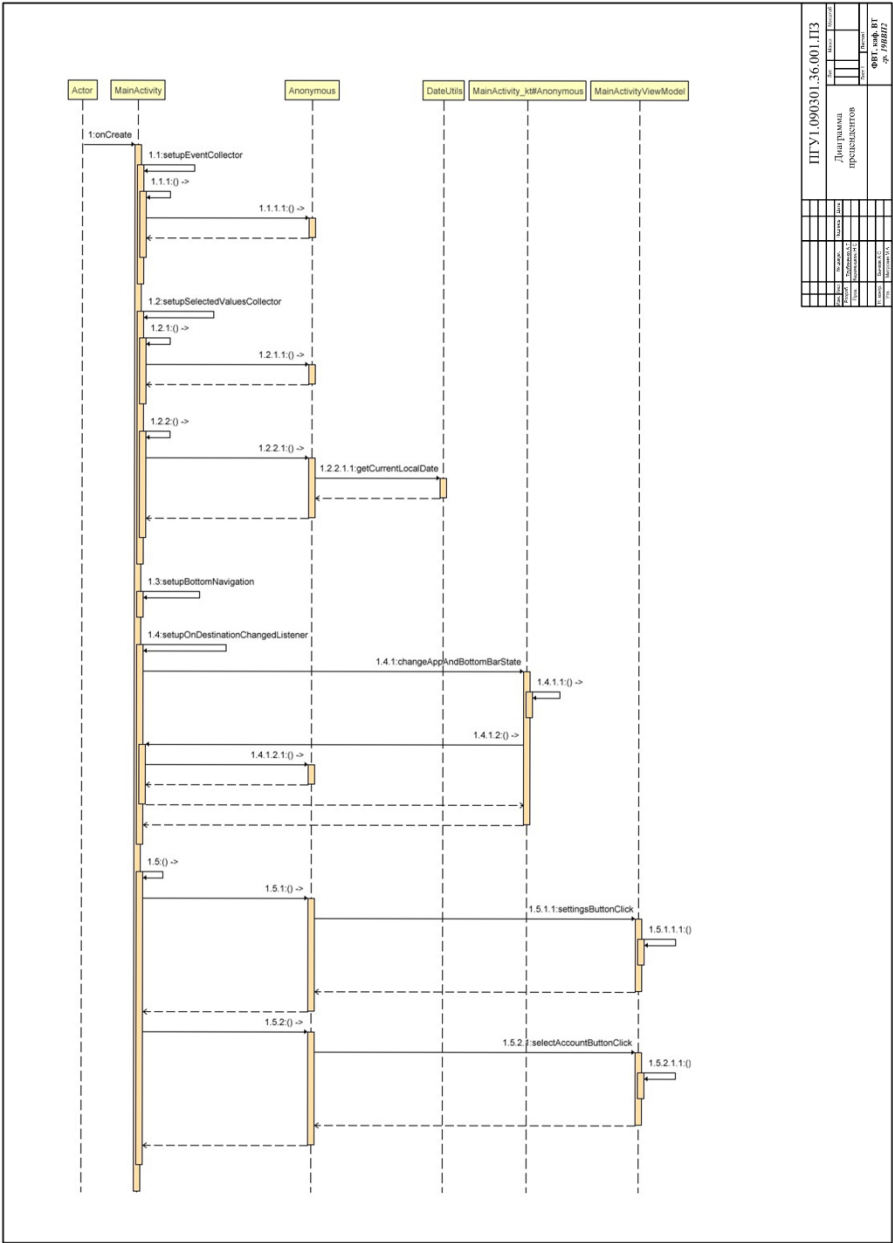
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.36.001 ПЗ

Лист

42

Приложение А.4 – UML – диаграмма последовательности



ПГУ1.090301.36.001.ПЗ	
Диаграмма последовательности	
Имя	Диаграмма последовательности
Создан	2020.05.14
Изменен	2020.05.14
Версия	1.0
Создан	2020.05.14
Изменен	2020.05.14
Версия	1.0
Создан	2020.05.14
Изменен	2020.05.14
Версия	1.0

Изм.	Лист	№ докум.	Подпись	Дата

Приложение Б

(обязательное)

Сервис по учету личных расходов

Листинг программы

					ПГУ1.090301.36.001 ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

Файл *Converter.kt*

```
package com.beamerboy.money.common

import androidx.room.TypeConverter
import java.time.LocalDate
import java.time.LocalTime

class Converter {

    @TypeConverter
    fun fromLocalDate(date: LocalDate): String {
        return date.toString()
    }

    @TypeConverter
    fun toLocalDate(data: String): LocalDate {
        return LocalDate.parse(data)
    }

    @TypeConverter
    fun fromLocalTime(time: LocalTime): String {
        return time.toString()
    }

    @TypeConverter
    fun toLocalTime(data: String): LocalTime {
        return LocalTime.parse(data)
    }
}
```

Файл *DateUtils.kt*

```
package com.beamerboy.money.common

import java.text.DecimalFormat
import java.text.SimpleDateFormat
import java.time.Instant
import java.time.LocalDate
import java.time.LocalTime
import java.time.ZoneId
import java.time.ZoneOffset
import java.util.Date
import java.util.Locale
import kotlin.math.absoluteValue

object DateUtils {

    fun Double.toAmountFormat(withMinus: Boolean): String {
        return DecimalFormat(if (withMinus || this < 0) "-#####.##" else "#####.##").format(this.absoluteValue)
    }

    fun getCurrentLocalDate(): LocalDate = LocalDate.parse(
        SimpleDateFormat("yyyy-MM-dd", Locale.US).format(Date())
    )

    fun getCurrentLocalTime(): LocalTime = LocalTime.parse(
        SimpleDateFormat("hh:mm:ss", Locale.US).format(Date())
    )
}
```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		45

```

    fun Long.toLocalDate(): LocalDate =
Instant.ofEpochMilli(this).atZone(ZoneId.systemDefault()).toLocalDate()

    fun LocalDate.toMilliseconds(): Long =
        this.atStartOfDay(ZoneOffset.UTC).toInstant().toEpochMilli()

    fun String.asLocalDate(): LocalDate = LocalDate.parse(this)

    const val DAY_IN_MS: Long = 1000 * 60 * 60 * 24
}

```

Файл *AccountsDao.kt*

```
package com.beamerboy.money.feature.data.datasource.dao
```

```

import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import androidx.room.Update
import com.beamerboy.money.feature.domain.model.Account
import kotlinx.coroutines.flow.Flow

```

@Dao

```
interface AccountsDao {
```

```

    @Query("SELECT * FROM accounts")
    fun getAccounts(): Flow<List<Account>>

```

```

    @Query("SELECT * FROM accounts WHERE id = :id")
    suspend fun getAccountById(id: Int): Account?

```

```

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAccount(account: Account)

```

```

    @Update
    suspend fun updateAccount(account: Account)

```

```

    @Query("UPDATE accounts SET amount = :amount WHERE id = :id")
    suspend fun updateAccountAmount(id: Int, amount: Double)

```

```

    @Delete
    suspend fun deleteAccount(account: Account)
}

```

Файл *CategoriesDao.kt*

```
package com.beamerboy.money.feature.data.datasource.dao
```

```

import androidx.room.Dao
import androidx.room.Query
import com.beamerboy.money.feature.domain.model.CategoryView
import kotlinx.coroutines.flow.Flow
import java.time.LocalDate

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		46

```

@Dao
interface CategoriesDao {

    @Query("SELECT id, " +
        "name, " +
        "icon, " +
        "icon_color, " +
        "ifnull((SELECT SUM(amount) " +
        "FROM transactions " +
        "WHERE categories.id = category_id " +
        "AND date >= :from " +
        "AND date <= :to), 0) " +
        "AS category_amount " +
        "FROM categories " +
        "GROUP BY id")
    fun getCategoryViews(from: LocalDate, to: LocalDate):
    Flow<List<CategoryView>>

    @Query("SELECT id, " +
        " name, " +
        "icon, " +
        "icon_color, " +
        " ifnull((SELECT SUM(amount) " +
        "FROM transactions " +
        "WHERE categories.id = category_id " +
        "AND date >= :from " +
        "AND date <= :to " +
        "AND account_id = :id), 0) " +
        "AS category_amount " +
        "FROM categories " +
        "GROUP BY id")
    fun getCategoryViewsForAccount(from: LocalDate, to: LocalDate, id: Int):
    Flow<List<CategoryView>>
}

```

Файл *TransactionsDao.kt*

```
package com.beamerboy.money.feature.data.datasource.dao

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import com.beamerboy.money.feature.domain.model.DayInfo
import com.beamerboy.money.feature.domain.model.Transaction
import com.beamerboy.money.feature.domain.model.TransactionView
import kotlinx.coroutines.flow.Flow
import java.time.LocalDate

@Dao
interface TransactionsDao {

    @Query("SELECT transactions.id, " +
        " transactions.note, " +
        " transactions.amount, " +
        " transactions.date, " +
        " transactions.time, " +
        " categories.id AS category_id, " +
        " categories.name AS category_name, " +
        " accounts.id AS account_id, " +
        " accounts.name AS account_name, " +
        " categories.icon, " +
        " categories.icon_color FROM transactions " +
        " JOIN accounts ON accounts.id = transactions.account_id " +
        " JOIN categories ON categories.id = transactions.category_id " +
        " WHERE date >= :from AND date <= :to ORDER BY date ASC, time"
    ASC")
    fun getTransactionViews(from: LocalDate, to: LocalDate):
    Flow<List<TransactionView>>

    @Query("SELECT transactions.id, " +
        " transactions.note, " +
        " transactions.amount, " +
        " transactions.date, " +
        " transactions.time, " +
        " categories.id AS category_id, " +
        " categories.name AS category_name, " +
        " accounts.id AS account_id, " +
        " accounts.name AS account_name, " +
        " categories.icon, categories.icon_color " +
        " FROM transactions JOIN accounts ON accounts.id =
    transactions.account_id " +
        " JOIN categories ON categories.id = transactions.category_id " +
        " WHERE date >= :from AND date <= :to AND account_id = :id ORDER
    BY date ASC, time ASC")
    fun getTransactionViewsForAccount(from: LocalDate, to: LocalDate, id:
    Int): Flow<List<TransactionView>>

    @Query("SELECT date, " +
        " SUM(amount) AS amount_per_day FROM transactions WHERE date >=
    :from AND date <= :to GROUP BY date ORDER BY date ASC")
    fun getTransactionAmountsPerDay(from: LocalDate, to: LocalDate):
    Flow<List<DayInfo>>

    @Query("SELECT date, " +
```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		48

```

        "SUM(amount) AS amount_per_day FROM transactions WHERE date >=
:from AND date <= :to AND account_id = :id GROUP BY date ORDER BY date ASC")
    fun getTransactionAmountsPerDayForAccount(from: LocalDate, to: LocalDate,
id: Int): Flow<List<DayInfo>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertTransaction(transaction: Transaction)

    @Query("DELETE FROM transactions WHERE id = :id")
    suspend fun deleteTransactionById(id: Int)
}

```

Файл AppModule.kt

```

package com.beamerboy.money.feature.data.datasource.di

import android.app.Application
import android.content.Context
import androidx.room.Room
import com.beamerboy.money.feature.data.datasource.MoneyManagerDatabase
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

@Module
@InstallIn(SingletonComponent::class)
object AppModule {

    @Singleton
    @Provides
    fun provideAppContext(app: Application): Context = app.applicationContext

    @Provides
    @Singleton
    fun provideDatabase(app: Application): MoneyManagerDatabase {
        return Room.databaseBuilder(
            app.applicationContext,
            MoneyManagerDatabase::class.java,
            MoneyManagerDatabase.DATABASE_NAME
        ).createFromAsset("database/money_manager.db").build()
    }
}

```

Файл ManagerDatabase.kt

```

package com.beamerboy.money.feature.data.datasource

import androidx.room.Database
import androidx.room.RoomDatabase
import androidx.room.TypeConverters
import com.beamerboy.money.common.Converter
import com.beamerboy.money.feature.data.datasource.dao.AccountsDao
import com.beamerboy.money.feature.data.datasource.dao.CategoriesDao
import com.beamerboy.money.feature.data.datasource.dao.TransactionsDao
import com.beamerboy.money.feature.domain.model.Account
import com.beamerboy.money.feature.domain.model.Category

```

```

import com.beamerboy.money.feature.domain.model.Transaction

@Database(
    entities = [Account::class, Category::class, Transaction::class],
    version = 1,
    exportSchema = true
)
@TypeConverters(Converter::class)
abstract class MoneyManagerDatabase : RoomDatabase() {

    abstract val accountsDao: AccountsDao
    abstract val categoriesDao: CategoriesDao
    abstract val transactionsDao: TransactionsDao

    companion object {
        const val DATABASE_NAME = "money_manager_db"
    }
}

```

Файл BynResponse.kt

```

package com.beamerboy.money.feature.domain.model.currency

import com.squareup.moshi.Json

data class BynResponse(
    @Json(name = "Cur_Abbreviation") val base: String,
    @Json(name = "Cur_OfficialRate") val rate: Double,
    @Json(name = "Cur_Scale") val scale: Int
)

```

Файл *CurrencyResponse.kt*

```
package com.beamerboy.money.feature.domain.model.currency

import com.squareup.moshi.Json

data class CurrencyResponse(
    @Json(name = "data") val rates: Rates,
    val query: Query
)

data class Rates(
    @Json(name = "USD") val usd: Double = 1.0,
    @Json(name = "EUR") val eur: Double = 1.0,
    @Json(name = "RUB") val rub: Double = 1.0
)

data class Query(
    @Json(name = "base_currency") val base: String
)
```

Файл *Account.kt*

```
package com.beamerboy.money.feature.domain.model

import android.os.Parcelable
import androidx.room.Entity
import androidx.room.PrimaryKey
import kotlinx.parcelize.Parcelize

@Parcelize
@Entity(tableName = "accounts")
data class Account(
    @PrimaryKey(autoGenerate = true)
    val id: Int? = null,
    val name: String,
    val amount: Double = 0.0,
    val color: String
) : Parcelable
```

Файл *Category.kt*

```
package com.beamerboy.money.feature.domain.model

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "categories")
data class Category(
    @PrimaryKey(autoGenerate = true)
    val id: Int? = null,
    val name: String,
    val icon: Int,
    @ColumnInfo(name = "icon_color")
)
```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		51

```

        val iconColor: String
    )

```

Файл *CategoryView.kt*

```

package com.beamerboy.money.feature.domain.model

import android.os.Parcelable
import androidx.room.ColumnInfo
import androidx.room.Entity
import kotlinx.parcelize.Parcelize

@Parcelize
@Entity
data class CategoryView(
    val id: Int,
    val name: String,
    val icon: Int,
    @ColumnInfo(name = "icon_color")
    val iconColor: String,
    @ColumnInfo(name = "category_amount")
    val amount: Double
) : Parcelable

```

Файл *DayInfo.kt*

```

package com.beamerboy.money.feature.domain.model

import androidx.room.ColumnInfo
import androidx.room.Entity
import java.time.LocalDate

@Entity
data class DayInfo(
    @ColumnInfo(name = "date")
    val transactionDate: LocalDate,
    @ColumnInfo(name = "amount_per_day")
    val amountPerDay: Double
)

```

Файл *Transaction.kt*

```
package com.beamerboy.money.feature.domain.model

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.ForeignKey
import androidx.room.ForeignKey.CASCADE
import androidx.room.PrimaryKey
import com.beamerboy.money.common.DateUtils.getCurrentLocalDate
import com.beamerboy.money.common.DateUtils.getCurrentLocalTime
import java.time.LocalDate
import java.time.LocalTime

@Entity(
    tableName = "transactions",
    foreignKeys = [
        ForeignKey(
            entity = Category::class,
            parentColumns = ["id"],
            childColumns = ["category_id"],
            onDelete = CASCADE
        ),
        ForeignKey(
            entity = Account::class,
            parentColumns = ["id"],
            childColumns = ["account_id"],
            onDelete = CASCADE
        )
    ]
)
data class Transaction(
    @PrimaryKey(autoGenerate = true)
    val id: Int? = null,
    val note: String,
    val amount: Double,
    val date: LocalDate = getCurrentLocalDate(),
    val time: LocalTime = getCurrentLocalTime(),
    @ColumnInfo(name = "account_id")
    val accountId: Int,
    @ColumnInfo(name = "category_id")
    val categoryId: Int
)
```

Файл *TransactionView.kt*

```
package com.beamerboy.money.feature.domain.model

import androidx.room.ColumnInfo
import java.time.LocalDate
import java.time.LocalTime

data class TransactionView(
    val id: Int,
    val note: String,
    val amount: Double,
    val date: LocalDate,
    val time: LocalTime,
    @ColumnInfo(name = "category_id")
    val categoryId: Int,
    @ColumnInfo(name = "category_name")
    val categoryName: String,
    @ColumnInfo(name = "account_id")
    val accountId: Int,
    @ColumnInfo(name = "account_name")
    val accountName: String,
    val icon: Int,
    @ColumnInfo(name = "icon_color")
    val iconColor: String
)
```

Файл *AccountActionsSheetFragment.kt*

```
package com.beamerboy.money.feature.presentation.ui.accounts.add.actions

import android.graphics.Color
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.appcompat.app.AlertDialog
import androidx.fragment.app.activityViewModels
import androidx.fragment.app.viewModels
import androidx.lifecycle.LifecycleScope
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import com.google.android.material.bottomsheet.BottomSheetDialogFragment
import com.beamerboy.money.R
import com.beamerboy.money.common.DateUtils.toAmountFormat
import com.beamerboy.money.databinding.SheetFragmentAccountActionsBinding
import com.beamerboy.money.feature.presentation.ui.MainActivityViewModel
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.flow.collectLatest

@AndroidEntryPoint
class AccountActionsSheetFragment : BottomSheetDialogFragment() {

    private var _binding: SheetFragmentAccountActionsBinding? = null
    private val binding get() = _binding!!

    private val viewModel: AccountActionsViewModel by viewModels()
    private val activityViewModel: MainActivityViewModel by activityViewModels()
```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		54

```

activityViewModels()

private val args by navArgs<AccountActionsSheetFragmentArgs>()

override fun onCreateView(
    inflater: LayoutInflater,
    container: ViewGroup?,
    savedInstanceState: Bundle?
): View {
    _binding = SheetFragmentAccountActionsBinding.inflate(inflater,
container, false)
    return binding.root
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    setupContainerAppearance()
    setupDeleteButtonVisibility()
    setupOnClickListeners()
    setupEventCollector()
}

private fun setupContainerAppearance() {
    val account = args.selectedAccount
    with(binding) {

actionsContainer.setBackgroundColor(Color.parseColor(account.color))
        accountName.text = account.name
        accountAmount.text = account.amount.toAmountFormat(withMinus =
false)
        accountCurrency.text = activityViewModel.getCurrency()
    }
}

private fun setupDeleteButtonVisibility() {
    val size = args.numberOfAccounts
    with(binding) {
        deleteButton.visibility = if (size == 1) View.GONE else
View.VISIBLE
        deleteButtonText.visibility = if (size == 1) View.INVISIBLE else
View.VISIBLE
        deleteButtonIcon.visibility = if (size == 1) View.INVISIBLE else
View.VISIBLE
    }
}

private fun setupOnClickListeners() {
    with(binding) {
        editButton.setOnClickListener {
viewModel.editButtonClick(args.selectedAccount) }
        deleteButton.setOnClickListener {
viewModel.deleteButtonClick(args.selectedAccount) }
    }
}

private fun setupEventCollector() {
    lifecycleScope.launchWhenStarted {
        viewModel.events.collectLatest {
            when (it) {

```

```

        is
AccountActionsViewModel.Event.NavigateToEditAccountScreen -> {
    findNavController().navigate(
        AccountActionsSheetFragmentDirections
            .actionAccountActionsSheetFragmentToAccountEditFragment(args.selectedAccount)
    )
}
        is
AccountActionsViewModel.Event.ShowTheDeleteAccountDialog -> {
    getAlertDialog().show()
}
        is AccountActionsViewModel.Event.DeleteAccount -> {
            if (args.selectedAccount ==
activityViewModel.selectedAccount.value)
                activityViewModel.setCurrentAccount(null)

                viewModel.deleteAccount(args.selectedAccount)
                dismiss()
            }
        }
    }
}

private fun getAlertDialog() = AlertDialog.Builder(requireContext())
    .setIcon(R.drawable.ic_warning)
    .setTitle(getString(R.string.delete_alert_title))
    .setMessage(getString(R.string.delete_alert_subtitle))
    .setPositiveButton(getString(R.string.ok)) { _, _ ->
        viewModel.deleteConfirmationButtonClick()
        this@AccountActionsSheetFragment.dismiss()
    }
    .setNegativeButton(getString(R.string.cancel)) { _, _ ->
this@AccountActionsSheetFragment.dismiss() }
    .setOnCancelListener { this@AccountActionsSheetFragment.dismiss() }
    .create()

    override fun onDestroy() {
        super.onDestroy()
        _binding = null
    }
}

```

Файл AccountActionsViewModel.kt

```

package com.beamerboy.money.feature.presentation.ui.accounts.add.actions

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.beamerboy.money.feature.domain.model.Account
import com.beamerboy.money.feature.domain.usecases.DeleteAccountUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.asSharedFlow
import kotlinx.coroutines.launch
import javax.inject.Inject

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		56

```

@HiltViewModel
class AccountActionsViewModel @Inject constructor(
    private val deleteAccountUseCase: DeleteAccountUseCase
) : ViewModel() {

    private val _events = MutableSharedFlow<Event>()
    val events = _events.asSharedFlow()

    suspend fun deleteAccount(account: Account) {
        deleteAccountUseCase(account)
    }

    fun editButtonClick(account: Account) {
        viewModelScope.launch {
            _events.emit(Event.NavigateToEditAccountScreen(account))
        }
    }

    fun deleteButtonClick(account: Account) {
        viewModelScope.launch {
            _events.emit(Event.ShowTheDeleteAccountDialog(account))
        }
    }

    fun deleteConfirmationButtonClick() {
        viewModelScope.launch {
            _events.emit(Event.DeleteAccount)
        }
    }

    sealed class Event {
        data class NavigateToEditAccountScreen(val account: Account) :
Event()
        data class ShowTheDeleteAccountDialog(val account: Account) : Event()
        object DeleteAccount : Event()
    }
}

```

Файл AccountAddFragment.kt

```

package com.beamerboy.money.feature.presentation.ui.accounts.add

import android.os.Bundle
import android.view.Menu
import android.view.MenuInflater
import android.view.MenuItem
import android.view.View
import android.widget.ImageView
import androidx.fragment.app.Fragment
import androidx.fragment.app.viewModels
import androidx.lifecycle.LifecycleScope
import androidx.navigation.fragment.DialogFragmentNavigator
import androidx.navigation.fragment.FragmentNavigator
import androidx.navigation.fragment.findNavController
import by.kirich1409.viewbindingdelegate.viewBinding
import com.beamerboy.money.R
import com.beamerboy.money.databinding.FragmentAccountAddBinding
import com.beamerboy.money.feature.domain.model.Account

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		57

```

import com.beamerboy.money.feature.presentation.utils.Utils.MAIN_COLOR
import com.beamerboy.money.feature.presentation.utils.Utils.setTint
import com.beamerboy.money.feature.presentation.utils.Utils.showToast
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.flow.collectLatest

@AndroidEntryPoint
class AccountAddFragment : Fragment(R.layout.fragment_account_add) {

    private val binding: FragmentAccountAddBinding by viewBinding()

    private val viewModel: AccountAddViewModel by viewModels()

    override fun onCreateView(view: View, savedInstanceState: Bundle?) {
        super.onCreateView(view, savedInstanceState)
        setHasOptionsMenu(true)

        setupOnClickListeners()
        setupEventCollector()
    }

    private fun setupEventCollector() {
        var color = MAIN_COLOR

        lifecycleScope.launchWhenStarted {
            viewModel.events.collectLatest {
                when (it) {
                    is AccountAddViewModel.Event.AddAccount -> {
                        val name =
binding.nameTextField.editText?.text.toString().trim()
                        if (name.isEmpty()) {
                            showToast(requireContext(),
getString(R.string.account_empty_name_error))
                        } else {
                            val amount =
binding.amountTextField.editText?.text.toString()
                                .toDoubleOrNull() ?: 0.0

                            val account = Account(name = name, amount =
amount, color = color)
                            viewModel.addAccount(account)

                            if (getCurrentDestination() ==
this@AccountAddFragment.javaClass.name)
                                findNavController().navigate(AccountAddFragmentDirections.actionAccountAddFra
gmentToAccountsFragment())
                        }
                    }
                    is AccountAddViewModel.Event.SelectColor -> {
                        binding.selectedColor.setTint(it.color)
                        color = it.color
                    }
                }
            }
        }

        private fun setupOnClickListeners() {
            with(binding) {

```

```

        color0.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color1.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color2.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color3.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color4.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color5.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color6.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color7.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color8.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color9.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color10.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color11.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color12.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color13.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color14.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color15.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
    }
}

override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {
    inflater.inflate(R.menu.update_menu, menu)
}

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    if (item.itemId == R.id.apply) {
        viewModel.applyButtonClick()
    }
    return super.onOptionsItemSelected(item)
}

private fun getCurrentDestination() =
    (findNavController().currentDestination as?
FragmentNavigator.Destination)?.className
    ?: (findNavController().currentDestination as?
DialogFragmentNavigator.Destination)?.className
}

```

Файл *AccountAddViewModel.kt*

```

package com.beamerboy.money.feature.presentation.ui.accounts.add

import android.widget.ImageView

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		59

```

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.beamerboy.money.feature.domain.model.Account
import com.beamerboy.money.feature.domain.usecases.AddAccountUseCase
import com.beamerboy.money.feature.presentation.utils.Utils.getImageViewTint
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.asSharedFlow
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
class AccountAddViewModel @Inject constructor(
    private val addAccountUseCase: AddAccountUseCase
) : ViewModel() {

    private val _events = MutableSharedFlow<Event>()
    val events = _events.asSharedFlow()

    suspend fun addAccount(account: Account) {
        addAccountUseCase(account)
    }

    fun applyButtonClick() {
        viewModelScope.launch {
            _events.emit(Event.AddAccount)
        }
    }

    fun selectColorClick(image: ImageView) {
        viewModelScope.launch {
            val color = getImageViewTint(image)
            _events.emit(Event.SelectColor(color))
        }
    }

    sealed class Event {
        object AddAccount : Event()
        data class SelectColor(val color: String) : Event()
    }
}

```

Файл *AccountEditFragment.kt*

```

package com.beamerboy.money.feature.presentation.ui.accounts.edit

import android.os.Bundle
import android.view.Menu
import android.view.MenuInflater
import android.view.MenuItem
import android.view.View
import android.widget.ImageView
import androidx.fragment.app.Fragment

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		60

```

import androidx.fragment.app.viewModels
import androidx.lifecycle.lifecycleScope
import androidx.navigation.fragment.DialogFragmentNavigator
import androidx.navigation.fragment.FragmentNavigator
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import by.kirich1409.viewbindingdelegate.viewBinding
import com.beamerboy.money.R
import com.beamerboy.money.common.DateUtils.toAmountFormat
import com.beamerboy.money.databinding.FragmentAccountEditBinding
import com.beamerboy.money.feature.presentation.utils.Utils.setTint
import com.beamerboy.money.feature.presentation.utils.Utils.showToast
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.flow.collectLatest

@AndroidEntryPoint
class AccountEditFragment : Fragment(R.layout.fragment_account_edit) {

    private val binding: FragmentAccountEditBinding by viewBinding()

    private val viewModel: AccountEditViewModel by viewModels()

    private val args by navArgs<AccountEditFragmentArgs>()

    override fun onCreateView(view: View, savedInstanceState: Bundle?) {
        super.onCreateView(view, savedInstanceState)
        setHasOptionsMenu(true)

        setupOnClickListeners()
        setupFields()
        setupEventCollector()
    }

    private fun setupFields() {
        val account = args.editableAccount

        with(binding) {
            nameTextField.editText?.setText(account.name)

            amountTextField.editText?.setText(account.amount.toAmountFormat(withMinus = false))
            selectedColor.setTint(account.color)
        }
    }

    private fun setupEventCollector() {
        val account = args.editableAccount
        var color = account.color

        lifecycleScope.launchWhenStarted {
            viewModel.events.collectLatest {
                when (it) {
                    is AccountEditViewModel.Event.UpdateAccount -> {
                        val name =
binding.nameTextField.editText?.text.toString().trim()
                        if (name.isEmpty()) {
                            showToast(context,
getString(R.string.account_empty_name_error))
                        } else {
                            val amount =

```

```

binding.amountTextField.editText?.text.toString()
    .toDoubleOrNull() ?: account.amount

    val newAccount =
        account.copy(name = name, amount = amount,
color = color)

        viewModel.updateAccount(newAccount)
        if (getCurrentDestination() ==
this@AccountEditFragment.javaClass.name)

findNavController().navigate(AccountEditFragmentDirections.actionAccountEditF
ragmentToAccountsFragment())
    }
    }
    is AccountEditViewModel.Event.SelectColor -> {
        binding.selectedColor.setTint(it.color)
        color = it.color
    }
    }
    }
    }

private fun setupOnClickListeners() {
    with(binding) {
        color0.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color1.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color2.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color3.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color4.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color5.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color6.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color7.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color8.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color9.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color10.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color11.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color12.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color13.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color14.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
        color15.setOnClickListener { viewModel.selectColorClick(it as
ImageView) }
    }
}

```

```

override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {
    inflater.inflate(R.menu.update_menu, menu)
}

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    if (item.itemId == R.id.apply) {
        viewModel.applyButtonClick()
    }
    return super.onOptionsItemSelected(item)
}

private fun getCurrentDestination() =
    (findNavController().currentDestination as?
    FragmentNavigator.Destination)?.className
    ?: (findNavController().currentDestination as?
    DialogFragmentNavigator.Destination)?.className
}

```

Файл *AccountEditViewModel.kt*

```
package com.beamerboy.money.feature.presentation.ui.accounts.edit

import android.widget.ImageView
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.beamerboy.money.feature.domain.model.Account
import com.beamerboy.money.feature.domain.usecases.UpdateAccountUseCase
import com.beamerboy.money.feature.presentation.utils.Utils.getImageViewTint
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.asSharedFlow
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
class AccountEditViewModel @Inject constructor(
    private val updateAccountUseCase: UpdateAccountUseCase
) : ViewModel() {

    private val _events = MutableSharedFlow<Event>()
    val events = _events.asSharedFlow()

    suspend fun updateAccount(account: Account) {
        updateAccountUseCase(account)
    }

    fun applyButtonClick() {
        viewModelScope.launch {
            _events.emit(Event.UpdateAccount)
        }
    }

    fun selectColorClick(image: ImageView) {
        viewModelScope.launch {
            val color = getImageViewTint(image)
            _events.emit(Event.SelectColor(color))
        }
    }

    sealed class Event {
        object UpdateAccount : Event()
        data class SelectColor(val color: String) : Event()
    }
}
```

Файл *AccountsFragment.kt*

```
package com.beamerboy.money.feature.presentation.ui.accounts

import android.annotation.SuppressLint
```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		64

```

import android.os.Bundle
import android.view.Menu
import android.view.MenuInflater
import android.view.MenuItem
import android.view.View
import androidx.fragment.app.Fragment
import androidx.fragment.app.activityViewModels
import androidx.fragment.app.viewModels
import androidx.lifecycle.lifecycleScope
import androidx.navigation.fragment.DialogFragmentNavigator
import androidx.navigation.fragment.FragmentNavigator
import androidx.navigation.fragment.findNavController
import androidx.recyclerview.widget.LinearLayoutManager
import by.kirich1409.viewbindingdelegate.viewBinding
import com.beamerboy.money.R
import com.beamerboy.money.common.DateUtils.toAmountFormat
import com.beamerboy.money.databinding.FragmentAccountsBinding
import com.beamerboy.money.feature.presentation.ui.MainActivityViewModel
import com.beamerboy.money.feature.presentation.utils.Utils.getDivider
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject
import kotlinx.coroutines.flow.collectLatest

@AndroidEntryPoint
class AccountsFragment : Fragment(R.layout.fragment_accounts) {

    private val binding: FragmentAccountsBinding by viewBinding()

    private val viewModel: AccountsViewModel by viewModels()
    private val activityViewModel: MainActivityViewModel by
activityViewModels()

    @Inject
    lateinit var accountsAdapter: AccountsRVAdapter

    override fun onCreateView(view: View, savedInstanceState: Bundle?) {
        super.onCreateView(view, savedInstanceState)
        setHasOptionsMenu(true)

        setupRecyclerView()
        setupCollectors()
    }

    private fun setupRecyclerView() {
        binding.listOfAccounts.apply {
            adapter = accountsAdapter
            layoutManager = LinearLayoutManager(requireContext())
            addItemDecoration(getDivider(requireContext()))
            itemAnimator = null
        }

        accountsAdapter.setOnClickListener(
            AccountsRVAdapter.OnClickListener {
                viewModel.selectAccount(it)
            }
        )
    }

    private fun setupCollectors() {
        lifecycleScope.launchWhenStarted {

```

```

        viewModel.accounts.collectLatest { newList ->
            accountsAdapter.submitList(newList)
            with(binding) {
                fullAmount.text =
viewModel.getFullAmount().toAmountFormat(withMinus = false)
                mainCurrency.text = activityViewModel.getCurrency()
            }
        }
    }

    lifecycleScope.launchWhenStarted {
        viewModel.events.collectLatest {
            when (it) {
                is AccountsViewModel.Event.NavigateToAddAccountScreen ->
{
                    if (getCurrentDestination() ==
this@AccountsFragment.javaClass.name) {

findNavController().navigate(AccountsFragmentDirections.actionAccountsFragmen
tToAccountAddFragment())
                    }
                is AccountsViewModel.Event.OpenTheAccountActionsSheet ->
{
                    if (getCurrentDestination() ==
this@AccountsFragment.javaClass.name) {
                        findNavController().navigate(
AccountsFragmentDirections.actionAccountsFragmentToAccountActionsSheetFragmen
t(
                            it.account, viewModel.accounts.value.size
                        )
                    )
                }
            }
        }
    }

    private var currentCurrency: String = ""

    @SuppressLint("NotifyDataSetChanged")
    override fun onStart() {
        super.onStart()
        if (currentCurrency != activityViewModel.getCurrency()) {
            binding.mainCurrency.text = activityViewModel.getCurrency()
            accountsAdapter.notifyDataSetChanged()
        }
    }

    override fun onStop() {
        super.onStop()
        currentCurrency = activityViewModel.getCurrency()
    }

    override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {
        inflater.inflate(R.menu.accounts_menu, menu)
    }
}

```

```

        override fun onOptionsItemSelected(item: MenuItem): Boolean {
            if (item.itemId == R.id.add_account) {
                viewModel.addButtonClick()
            }
            return super.onOptionsItemSelected(item)
        }

        private fun getCurrentDestination() =
            (findNavController().currentDestination as?
            FragmentNavigator.Destination)?.className
            ?: (findNavController().currentDestination as?
            DialogFragmentNavigator.Destination)?.className
    }

```

Файл AccountsRVAdapter.kt

```

package com.beamerboy.money.feature.presentation.ui.accounts

import android.content.SharedPreferences
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.beamerboy.money.common.DateUtils.toAmountFormat
import com.beamerboy.money.databinding.ItemAccountBinding
import com.beamerboy.money.feature.domain.model.Account
import com.beamerboy.money.feature.presentation.utils.Utils.CURRENCY_PREFERENCE_KEY
import com.beamerboy.money.feature.presentation.utils.Utils.MAIN_CURRENCY
import com.beamerboy.money.feature.presentation.utils.Utils.setTint
import javax.inject.Inject
import javax.inject.Singleton

@Singleton
class AccountsRVAdapter @Inject constructor(
    private val sharedPreferences: SharedPreferences
) : ListAdapter<Account, AccountsRVAdapter.AccountViewHolder>(DIFF_CALLBACK) {

    private var onClickListener: OnClickListener? = null

    inner class AccountViewHolder(private val binding: ItemAccountBinding) :
        RecyclerView.ViewHolder(binding.root) {

        fun bind(account: Account) {
            with(binding) {
                name.text = account.name
                amount.text = account.amount.toAmountFormat(withMinus =
                false)
                currency.text =
                sharedPreferences.getString(CURRENCY_PREFERENCE_KEY, MAIN_CURRENCY)
                iconBackground.setTint(account.color)
            }

            itemView.setOnClickListener {
                onClickListener?.onClick(account)
            }
        }
    }

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		67

```

    }
}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
AccountViewHolder {
    val inflater = LayoutInflater.from(parent.context)
    val binding = ItemAccountBinding.inflate(inflater, parent,
false)
    return AccountViewHolder(binding)
}

override fun onBindViewHolder(holder: AccountViewHolder, position: Int) {
    holder.bind(getItem(position))
}

fun setOnClickListener(onClickListener: OnClickListener) {
    this.setOnClickListener = onClickListener
}

class OnClickListener(val clickListener: (account: Account) -> Unit) {
    fun onClick(account: Account) = clickListener(account)
}

companion object DIFF_CALLBACK : DiffUtil.ItemCallback<Account>() {
    override fun areItemsTheSame(oldItem: Account, newItem: Account):
Boolean {
        return oldItem.id == newItem.id
    }

    override fun areContentsTheSame(oldItem: Account, newItem: Account):
Boolean {
        return oldItem.hashCode() == newItem.hashCode()
    }
}
}

```

Файл *AccountsViewModel.kt*

```

package com.beamerboy.money.feature.presentation.ui.accounts

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.beamerboy.money.feature.domain.model.Account
import com.beamerboy.money.feature.domain.usecases.GetAccountsUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.Job
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asSharedFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.launchIn
import kotlinx.coroutines.flow.onEach
import kotlinx.coroutines.flow.launch
import javax.inject.Inject

@HiltViewModel
class AccountsViewModel @Inject constructor(
    private val getAccountsUseCase: GetAccountsUseCase

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		68

```

) : ViewModel() {

    private val _accounts = MutableStateFlow(emptyList<Account>())
    val accounts = _accounts.asStateFlow()

    private val _events = MutableSharedFlow<Event>()
    val events = _events.asSharedFlow()

    private var getAccountsJob: Job? = null

    init {
        getAccounts()
    }

    private fun getAccounts() {
        getAccountsJob?.cancel()
        getAccountsJob = getAccountsUseCase()
            .onEach { accounts ->
                _accounts.value = accounts
            }
            .launchIn(viewModelScope)
    }

    fun getFullAmount(): Double {
        return _accounts.value.sumOf { it.amount }
    }

    fun selectAccount(account: Account) {
        viewModelScope.launch {
            _events.emit(Event.OpenTheAccountActionsSheet(account))
        }
    }

    fun addButtonClick() {
        viewModelScope.launch {
            _events.emit(Event.NavigateToAddAccountScreen)
        }
    }

    sealed class Event {
        object NavigateToAddAccountScreen : Event()
        data class OpenTheAccountActionsSheet(val account: Account) : Event()
    }
}

```

Файл *AuthActivity.kt*

```

package com.beamerboy.money.feature.presentation.ui.auth

import android.app.ActivityManager
import android.content.Context
import android.content.Intent
import android.os.Build
import android.os.Bundle
import android.os.VibrationEffect
import android.os.Vibrator
import android.os.VibratorManager
import android.util.Log
import androidx.activity.viewModels
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.isVisible
import androidx.lifecycle.LifecycleScope
import com.beamerboy.money.R
import com.beamerboy.money.databinding.ActivityAuthBinding
import com.beamerboy.money.feature.presentation.ui.MainActivity
import com.beamerboy.money.feature.presentation.utils.Utils.setTint
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.delay
import kotlinx.coroutines.flow.collectLatest
import kotlinx.coroutines.launch

@AndroidEntryPoint
class AuthActivity : AppCompatActivity() {

    private lateinit var binding: ActivityAuthBinding

    private val viewModel: AuthViewModel by viewModels()

    private val vibrator by lazy {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
            val vibratorManager =
                getSystemService(Context.VIBRATOR_MANAGER_SERVICE) as
VibratorManager
            vibratorManager.defaultVibrator
        } else {
            getSystemService(VIBRATOR_SERVICE) as Vibrator
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityAuthBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setupClickListeners()
        setupCollectors()
        setupEventCollectors()

        if (viewModel.isAppLaunchedFirstTime()) {
            binding.enterPinText.text = getString(R.string.set_the_pin)
            binding.forgotCode.isVisible = false
        }

        private fun setupClickListeners() {
            with(binding) {

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		70

```

        number0.setOnClickListener { viewModel.numberButtonClick("0") }
        number1.setOnClickListener { viewModel.numberButtonClick("1") }
        number2.setOnClickListener { viewModel.numberButtonClick("2") }
        number3.setOnClickListener { viewModel.numberButtonClick("3") }
        number4.setOnClickListener { viewModel.numberButtonClick("4") }
        number5.setOnClickListener { viewModel.numberButtonClick("5") }
        number6.setOnClickListener { viewModel.numberButtonClick("6") }
        number7.setOnClickListener { viewModel.numberButtonClick("7") }
        number8.setOnClickListener { viewModel.numberButtonClick("8") }
        number9.setOnClickListener { viewModel.numberButtonClick("9") }
        forgotCode.setOnClickListener { viewModel.forgotButtonClick() }
        backspace.setOnClickListener { viewModel.backspaceButtonClick() }
    }
}

private fun setupCollectors() {
    lifecycleScope.launchWhenStarted {
        viewModel.code.collectLatest { newCode ->
            with(binding) {
                var count = newCode.length
                for (item in listOf(circle1, circle2, circle3, circle4))
            {
                item.setTint(if (count > 0) R.color.yellow_orange
else R.color.light_gray)
                count--
            }
        }
    }
}

private fun startVibrating() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        vibrator.vibrate(
            VibrationEffect.createOneShot(
                ERROR_TIME,
                VibrationEffect.DEFAULT_AMPLITUDE
            )
        )
    } else {
        vibrator.vibrate(ERROR_TIME)
    }
}

private fun setupEventCollectors() {
    lifecycleScope.launchWhenStarted {
        viewModel.events.collectLatest {
            when (it) {
                is AuthViewModel.Event.ForgotClick -> {
                    showAlert(
                        getString(R.string.forgot_alert_title),
                        getString(R.string.forgot_alert_message)
                    ) {
                        showAlert(
                            getString(R.string.erase_data_alert_title),
                            getString(R.string.erase_data_alert_message)
                        ) {
                            viewModel.eraseDataClick()
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
    is AuthViewModel.Event.OpenMainActivity -> {
        with(binding) {
            for (item in listOf(circle1, circle2, circle3,
circle4)) {
                item.setTint(R.color.blue_green)
            }
        }
        val intent = Intent(this@AuthActivity,
MainActivity::class.java)
        startActivity(intent)
        overridePendingTransition(R.anim.fade_in,
R.anim.fade_out)
        finish()
    }
    is AuthViewModel.Event.DeleteCode -> {
        lifecycleScope.launch {
            with(binding) {
                backspace.isClickable = false
                forgotCode.isClickable = false

                for (item in listOf(circle1, circle2,
circle3, circle4)) {
                    item.setTint(R.color.orange_red)
                }
                startVibrating()
                delay(ERROR_TIME)
                viewModel.clearCode()

                backspace.isClickable = true
                forgotCode.isClickable = true
            }
        }
    }
    is AuthViewModel.Event.EraseData -> {
        (getSystemService(ACTIVITY_SERVICE) as
ActivityManager).clearApplicationUserData()
    }
    is AuthViewModel.Event.SetThePin -> {
        binding.enterPinText.text =
getString(R.string.set_the_pin)
    }
    is AuthViewModel.Event.RepeatThePin -> {
        binding.enterPinText.text =
getString(R.string.repeat_the_pin)
    }
}
}
}

private fun showAlert(title: String, message: String, func: () -> Unit) {
    val alert = AlertDialog.Builder(this)
        .setIcon(R.drawable.ic_warning)
        .setTitle(title)
        .setMessage(message)
        .setPositiveButton(getString(R.string.ok)) { _, _ -> func() }
        .setNegativeButton(getString(R.string.cancel)) { _, _ -> }
        .create()
    alert.show()
}

```

```

    }

    companion object {
        private const val ERROR_TIME: Long = 250
    }
}

```

Файл *AuthViewModel.kt*

```

package com.beamerboy.money.feature.presentation.ui.auth

import android.app.ActivityManager
import android.content.Context
import android.content.Intent
import android.os.Build
import android.os.Bundle
import android.os.VibrationEffect
import android.os.Vibrator
import android.os.VibratorManager
import android.util.Log
import androidx.activity.viewModels
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.isVisible
import androidx.lifecycle.LifecycleScope
import com.beamerboy.money.R
import com.beamerboy.money.databinding.ActivityAuthBinding
import com.beamerboy.money.feature.presentation.ui.MainActivity
import com.beamerboy.money.feature.presentation.utils.Utils.setTint
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.delay
import kotlinx.coroutines.flow.collectLatest
import kotlinx.coroutines.launch

@AndroidEntryPoint
class AuthActivity : AppCompatActivity() {

    private lateinit var binding: ActivityAuthBinding

    private val viewModel: AuthViewModel by viewModels()

    private val vibrator by lazy {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
            val vibratorManager =
                getSystemService(Context.VIBRATOR_MANAGER_SERVICE) as
VibratorManager
            vibratorManager.defaultVibrator
        } else {
            getSystemService(VIBRATOR_SERVICE) as Vibrator
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityAuthBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setupClickListeners()
    }

```

```

        setupCollectors()
        setupEventCollectors()

        if (viewModel.isAppLaunchedFirstTime()) {
            binding.enterPinText.text = getString(R.string.set_the_pin)
            binding.forgotCode.isVisible = false
        }
    }

    private fun setupClickListeners() {
        with(binding) {
            number0.setOnClickListener { viewModel.numberButtonClick("0") }
            number1.setOnClickListener { viewModel.numberButtonClick("1") }
            number2.setOnClickListener { viewModel.numberButtonClick("2") }
            number3.setOnClickListener { viewModel.numberButtonClick("3") }
            number4.setOnClickListener { viewModel.numberButtonClick("4") }
            number5.setOnClickListener { viewModel.numberButtonClick("5") }
            number6.setOnClickListener { viewModel.numberButtonClick("6") }
            number7.setOnClickListener { viewModel.numberButtonClick("7") }
            number8.setOnClickListener { viewModel.numberButtonClick("8") }
            number9.setOnClickListener { viewModel.numberButtonClick("9") }
            forgotCode.setOnClickListener { viewModel.forgotButtonClick() }
            backspace.setOnClickListener { viewModel.backspaceButtonClick() }
        }
    }

    private fun setupCollectors() {
        lifecycleScope.launchWhenStarted {
            viewModel.code.collectLatest { newCode ->
                with(binding) {
                    var count = newCode.length
                    for (item in listOf(circle1, circle2, circle3, circle4))
                    {
                        item.setTint(if (count > 0) R.color.yellow_orange
                        else R.color.light_gray)
                        count--
                    }
                }
            }
        }
    }

    private fun startVibrating() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            vibrator.vibrate(
                VibrationEffect.createOneShot(
                    ERROR_TIME,
                    VibrationEffect.DEFAULT_AMPLITUDE
                )
            )
        } else {
            vibrator.vibrate(ERROR_TIME)
        }
    }

    private fun setupEventCollectors() {
        lifecycleScope.launchWhenStarted {
            viewModel.events.collectLatest {
                when (it) {
                    is AuthViewModel.Event.ForgotClick -> {

```

```

        showAlert(
            getString(R.string.forgot_alert_title),
            getString(R.string.forgot_alert_message)
        ) {
            showAlert(
                getString(R.string.erase_data_alert_title),
                getString(R.string.erase_data_alert_message)
            ) {
                viewModel.eraseDataClick()
            }
        }
    }
    is AuthViewModel.Event.OpenMainActivity -> {
        with(binding) {
            for (item in listOf(circle1, circle2, circle3,
circle4)) {
                item.setTint(R.color.blue_green)
            }
            val intent = Intent(this@AuthActivity,
MainActivity::class.java)
            startActivity(intent)
            overridePendingTransition(R.anim.fade_in,
R.anim.fade_out)
            finish()
        }
        is AuthViewModel.Event.DeleteCode -> {
            lifecycleScope.launch {
                with(binding) {
                    backspace.isClickable = false
                    forgotCode.isClickable = false

                    for (item in listOf(circle1, circle2,
circle3, circle4)) {
                        item.setTint(R.color.orange_red)
                    }
                    startVibrating()
                    delay(ERROR_TIME)
                    viewModel.clearCode()

                    backspace.isClickable = true
                    forgotCode.isClickable = true
                }
            }
        }
        is AuthViewModel.Event.EraseData -> {
            (getSystemService(ACTIVITY_SERVICE) as
ActivityManager).clearApplicationUserData()
        }
        is AuthViewModel.Event.SetThePin -> {
            binding.enterPinText.text =
getString(R.string.set_the_pin)
        }
        is AuthViewModel.Event.RepeatThePin -> {
            binding.enterPinText.text =
getString(R.string.repeat_the_pin)
        }
    }
}
}
}

```

```

    }

    private fun showAlert(title: String, message: String, func: () -> Unit) {
        val alert = AlertDialog.Builder(this)
            .setIcon(R.drawable.ic_warning)
            .setTitle(title)
            .setMessage(message)
            .setPositiveButton(getString(R.string.ok)) { _, _ -> func() }
            .setNegativeButton(getString(R.string.cancel)) { _, _ -> }
            .create()
        alert.show()
    }

    companion object {
        private const val ERROR_TIME: Long = 250
    }
}

```

Файл *ChartFragment.kt*

```

package com.beamerboy.money.feature.presentation.ui.chart

import android.graphics.Color
import android.os.Bundle
import android.util.TypedValue
import android.view.Menu
import android.view.MenuInflater
import android.view.MenuItem
import android.view.View
import androidx.fragment.app.Fragment
import androidx.fragment.app.activityViewModels
import androidx.fragment.app.viewModels
import androidx.lifecycle.LifecycleScope
import androidx.navigation.fragment.DialogFragmentNavigator
import androidx.navigation.fragment.FragmentNavigator
import androidx.navigation.fragment.findNavController
import by.kirich1409.viewbindingdelegate.viewBinding
import com.github.mikephil.charting.animation.Easing
import com.github.mikephil.charting.data.PieData
import com.github.mikephil.charting.data.PieDataSet
import com.github.mikephil.charting.data.PieEntry
import com.beamerboy.money.R
import com.beamerboy.money.common.DateUtils.toAmountFormat
import com.beamerboy.money.databinding.FragmentChartBinding
import com.beamerboy.money.databinding.ItemCategoryBinding
import com.beamerboy.money.feature.domain.model.CategoryView
import com.beamerboy.money.feature.presentation.ui.MainActivityViewModel
import com.beamerboy.money.feature.presentation.utils.Utils.MAIN_COLOR
import com.beamerboy.money.feature.presentation.utils.Utils.setIcon
import com.beamerboy.money.feature.presentation.utils.Utils.setTint
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.flow.collectLatest

@AndroidEntryPoint
class ChartFragment : Fragment(R.layout.fragment_chart) {

    private val binding: FragmentChartBinding by viewBinding()

```

```

private val viewModel: ChartViewModel by viewModels()
private val activityViewModel: MainActivityViewModel by
activityViewModels()

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    setHasOptionsMenu(true)

    setupCollectors()
}

private fun setupCollectors() {
    lifecycleScope.launchWhenStarted {
        viewModel.categoryViews.collectLatest {
            updateChartData(it)
        }
    }

    lifecycleScope.launchWhenStarted {
        activityViewModel.selectedDateRange.collectLatest {
            viewModel.setDateRange(it.first, it.second)
        }
    }

    lifecycleScope.launchWhenStarted {
        activityViewModel.selectedAccount.collectLatest {
            viewModel.setSelectedAccount(it)
        }
    }

    lifecycleScope.launchWhenStarted {
        viewModel.events.collectLatest {
            when (it) {
                is ChartViewModel.Event.SelectDate -> {
                    if (getCurrentDestination() ==
this@ChartFragment.javaClass.name) {
                        findNavController().navigate(
ChartFragmentDirections.actionChartFragmentToSelectDateDialogFragment()
)
                    }
                }
            }
        }
    }
}

private fun updateChartData(categoryViews: List<CategoryView>) {
    if (categoryViews.isNotEmpty()) {
        val currency = activityViewModel.getCurrency()

        updateCategories(categoryViews, currency)

        var amount = 0.0
        val entries = ArrayList<PieEntry>()
        val entryColors = ArrayList<Int>()

        categoryViews.forEach { category ->
            if (category.amount != 0.0) {
                entries.add(PieEntry(category.amount.toFloat()))
            }
        }
    }
}

```

```

        entryColors.add(Color.parseColor(category.iconColor))
        amount += category.amount
    }
}

if (amount == 0.0) {
    entries.add(PieEntry(1f))
    entryColors.add(Color.parseColor(MAIN_COLOR))
    binding.chart.alpha = 0.3f
} else binding.chart.alpha = 1f

val dataSet = PieDataSet(entries, "").apply {
    colors = entryColors
    setDrawValues(false)
    sliceSpace = 2f
}

val amountString = amount.toAmountFormat(withMinus = false) + ' '
+ currency
binding.chart.apply {
    isDrawHoleEnabled = true
    holeRadius = 86f
    setHoleColor(Color.TRANSPARENT)
    setCenterTextColor(getThemeColor(R.attr.colorOnSecondary))
    centerText = "Пасходы\n$amountString"
    setCenterTextSize(20f)
    description.isEnabled = false
    legend.isEnabled = false
    data = PieData(dataSet)
}

binding.chart.invalidate()
binding.chart.animateY(1000, Easing.EaseInOutQuad)
}

}

private fun updateCategories(categoryViews: List<CategoryView>, currency:
String?) {
    with(binding) {
        category1.setCategoryAttributes(categoryViews[0], currency)
        category2.setCategoryAttributes(categoryViews[1], currency)
        category3.setCategoryAttributes(categoryViews[2], currency)
        category4.setCategoryAttributes(categoryViews[3], currency)
        category5.setCategoryAttributes(categoryViews[4], currency)
        category6.setCategoryAttributes(categoryViews[5], currency)
        category7.setCategoryAttributes(categoryViews[6], currency)
        category8.setCategoryAttributes(categoryViews[7], currency)
        category9.setCategoryAttributes(categoryViews[8], currency)
        category10.setCategoryAttributes(categoryViews[9], currency)
        category11.setCategoryAttributes(categoryViews[10], currency)
        category12.setCategoryAttributes(categoryViews[11], currency)
    }
}

private fun ItemCategoryBinding.setCategoryAttributes(
    category: CategoryView,
    currency: String?
) {
    this.name.text = category.name

```

```

        this.icon.setIcon(category.icon)
        this.iconBackground.setTint(category.iconColor)
        this.amount.text = category.amount.toAmountFormat(withMinus = false)
        this.currency.text = currency

        val color = Color.parseColor(category.iconColor)
        this.amount.setTextColor(color)
        this.currency.setTextColor(color)

        this.item.alpha = if (category.amount == 0.0) 0.3f else 1f
    }

    private var currentCurrency: String = ""

    override fun onStart() {
        super.onStart()
        if (currentCurrency != activityViewModel.getCurrency())
            updateChartData(viewModel.categoryViews.value)
    }

    override fun onStop() {
        super.onStop()
        currentCurrency = activityViewModel.getCurrency()
    }

    override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {
        inflater.inflate(R.menu.date_menu, menu)
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        if (item.itemId == R.id.select_date) {
            viewModel.selectDateClick()
        }
        return super.onOptionsItemSelected(item)
    }

    private fun getCurrentDestination() =
        (findNavController().currentDestination as?
        FragmentNavigator.Destination)?.className
        ?: (findNavController().currentDestination as?
        DialogFragmentNavigator.Destination)?.className

    private fun getThemeColor(color: Int): Int {
        val value = TypedValue()
        requireContext().theme.resolveAttribute(color, value, true)
        return value.data
    }
}

```

Файл *ChartViewModel.kt*

```

package com.beamerboy.money.feature.presentation.ui.chart

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.beamerboy.money.feature.domain.model.Account
import com.beamerboy.money.feature.domain.model.CategoryView
import com.beamerboy.money.feature.domain.usecases.GetCategoryViewsUseCase

```

```

import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.Job
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asSharedFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.launchIn
import kotlinx.coroutines.flow.onEach
import kotlinx.coroutines.launch
import java.time.LocalDate
import javax.inject.Inject

@HiltViewModel
class ChartViewModel @Inject constructor(
    private val getCategoryViewsUseCase: GetCategoryViewsUseCase
) : ViewModel() {

    private val _categoryViews = MutableStateFlow(emptyList<CategoryView>())
    val categoryViews = _categoryViews.asStateFlow()

    private val _events = MutableSharedFlow<Event>()
    val events = _events.asSharedFlow()

    private val _selectedAccount = MutableStateFlow<Account?>(null)
    private val _selectedDateRange = MutableStateFlow<Pair<LocalDate?,
LocalDate?>>(null to null)

    private var getCategoryViewsJob: Job? = null

    init {
        getCategoryViews()
    }

    private fun getCategoryViews() {
        getCategoryViewsJob?.cancel()
        getCategoryViewsJob =
            getCategoryViewsUseCase(_selectedDateRange.value,
            _selectedAccount.value)
            .onEach { categories -> _categoryViews.value = categories }
            .launchIn(viewModelScope)
    }

    fun setDateRange(from: LocalDate? = null, to: LocalDate? = null) {
        _selectedDateRange.value = from to to
        getCategoryViews()
    }

    fun setSelectedAccount(account: Account? = null) {
        _selectedAccount.value = account
        getCategoryViews()
    }

    fun selectDateClick() {
        viewModelScope.launch {
            _events.emit(Event.SelectDate)
        }
    }

    sealed class Event {
        object SelectDate : Event()
    }

```

```
}
}
```

Файл *CurrencyConverterFragment.kt*

```
package com.beamerboy.money.feature.presentation.ui.converter

import android.os.Bundle
import android.view.View
import android.widget.AdapterView
import androidx.core.view.isVisible
import androidx.fragment.app.Fragment
import androidx.fragment.app.activityViewModels
import androidx.fragment.app.viewModels
import androidx.lifecycle.LifecycleScope
import androidx.navigation.fragment.DialogFragmentNavigator
import androidx.navigation.fragment.FragmentNavigator
import androidx.navigation.fragment.findNavController
import by.kirich1409.viewbindingdelegate.viewBinding
import com.beamerboy.money.R
import com.beamerboy.money.databinding.FragmentCurrencyConverterBinding
import com.beamerboy.money.feature.presentation.ui.MainActivityViewModel
import com.beamerboy.money.feature.presentation.utils.Utils.showToast
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.flow.collectLatest

@AndroidEntryPoint
class CurrencyConverterFragment :
    Fragment(R.layout.fragment_currency_converter) {

    private val binding: FragmentCurrencyConverterBinding by viewBinding()

    private val viewModel: CurrencyConverterViewModel by viewModels()
    private val activityViewModel: MainActivityViewModel by
    activityViewModels()

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        setupConversionCollector()
        setupEventCollector()
        setupSpinnerListener()

        with(binding) {
            convertButton.setOnClickListener { viewModel.convertButtonClick() }
        }

        swapButton.setOnClickListener { viewModel.swapButtonClick() }
        addTransactionButton.setOnClickListener {
            viewModel.addTransactionButtonClick(
                activityViewModel.selectedAccount.value ?:
                activityViewModel.accounts.value[0],
                viewModel.resultValue.toFloat()
            )
        }
    }

    override fun onStart() {
```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.36.001 ПЗ

Лист

81

```

        super.onStart()
        binding.addTransactionButton.isVisible =
            binding.spinnerTo.selectedItem.toString() ==
activityViewModel.getCurrency()
    }

    private fun setupConversionCollector() {
        lifecycleScope.launchWhenStarted {
            viewModel.conversion.collectLatest {
                when (it) {
                    is CurrencyConverterViewModel.ConversionState.Ready -> {
                        with(binding) {
                            progressBar.isVisible = false
                            resultText.text = it.result
                            addTransactionButton.isClickable = true
                            addTransactionButton.alpha = 1f
                        }
                    }
                    is CurrencyConverterViewModel.ConversionState.Loading ->
{
                        binding.progressBar.isVisible = true
                    }
                    is CurrencyConverterViewModel.ConversionState.Error -> {
                        binding.progressBar.isVisible = false
                        showToast(requireContext(), it.error)
                    }
                    else -> Unit
                }
            }
        }
    }

    private fun setupEventCollector() {
        lifecycleScope.launchWhenStarted {
            viewModel.events.collectLatest {
                when (it) {
                    is CurrencyConverterViewModel.Event.Convert -> {
                        val amount =

binding.amountTextField.editText?.text.toString().toDoubleOrNull()
                        val from =
binding.spinnerFrom.selectedItem.toString()
                        val to = binding.spinnerTo.selectedItem.toString()

                        if (amount != null) {
                            viewModel.convert(amount, from, to)
                        } else showToast(requireContext(),
getString(R.string.enter_amount_error))
                    }
                    is CurrencyConverterViewModel.Event.Swap -> {
                        val fromPosition =
binding.spinnerFrom.selectedItemPosition
                        val toPosition =
binding.spinnerTo.selectedItemPosition

                        binding.spinnerFrom.setSelection(toPosition)
                        binding.spinnerTo.setSelection(fromPosition)
                    }
                    is
CurrencyConverterViewModel.Event.OpenTheAddTransactionSheet -> {

```

```

        if (getCurrentDestination() ==
this@CurrencyConverterFragment.javaClass.name) {
            findNavController().navigate(

CurrencyConverterFragmentDirections.actionCurrencyConverterFragmentToSelectCa
tegorySheetFragment(

                it.account, it.amount
            )
        )
    }
}
}
}
}

private fun setupSpinnerListener() {
    with(binding) {
        spinnerTo.onItemSelectedListener = object :
AdapterView.OnItemSelectedListener {
            override fun onItemSelected(
                adapter: AdapterView<*>?,
                view: View?,
                position: Int,
                id: Long
            ) {
                resultText.text = ""
                addTransactionButton.apply {
                    isVisible =
                        spinnerTo.selectedItem.toString() ==
activityViewModel.getCurrency()
                    isClickable = resultText.text.isNotEmpty()
                    alpha = if (resultText.text.isEmpty()) 0.3f else 1f
                }
            }

            override fun onNothingSelected(adapter: AdapterView<*>?) {}
        }
    }
}

private fun getCurrentDestination() =
    (findNavController().currentDestination as?
FragmentNavigator.Destination)?.className
    ?: (findNavController().currentDestination as?
DialogFragmentNavigator.Destination)?.className
}

```

Файл *CurrencyConverterViewModel.kt*

```

package com.beamerboy.money.feature.presentation.ui.converter

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.beamerboy.money.common.DateUtils.toAmountFormat
import com.beamerboy.money.feature.domain.model.Account
import com.beamerboy.money.feature.domain.usecases.ConvertCurrencyUseCase
import dagger.hilt.android.lifecycle.HiltViewModel

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		83

```

import javax.inject.Inject
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asSharedFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.launch

@HiltViewModel
class CurrencyConverterViewModel @Inject constructor(
    private val convertCurrencyUseCase: ConvertCurrencyUseCase
) : ViewModel() {

    private val _conversion =
        MutableStateFlow<ConversionState>(ConversionState.Idle)
    val conversion = _conversion.asStateFlow()

    private val _events = MutableSharedFlow<Event>()
    val events = _events.asSharedFlow()

    var resultValue: Double = 0.0

    fun convert(amount: Double, from: String, to: String) {
        viewModelScope.launch {
            _conversion.value = ConversionState.Loading
            resultValue = convertCurrencyUseCase(amount, from, to)
            _conversion.value =
                when (resultValue) {
                    -1.0 -> ConversionState.Error("Ошибка")
                    -2.0 -> ConversionState.Error("Проверьте подключение к
интернету")
                    else ->
                        ConversionState.Ready(resultValue.toAmountFormat(withMinus = false))
                }
        }
    }

    fun convertButtonClick() {
        viewModelScope.launch {
            _events.emit(Event.Convert)
        }
    }

    fun swapButtonClick() {
        viewModelScope.launch {
            _events.emit(Event.Swap)
        }
    }

    fun addTransactionButtonClick(account: Account, amount: Float) {
        viewModelScope.launch {
            _events.emit(Event.OpenTheAddTransactionSheet(account, amount))
        }
    }

    sealed class ConversionState {
        class Ready(val result: String) : ConversionState()
        class Error(val error: String) : ConversionState()
        object Loading : ConversionState()
        object Idle : ConversionState()
    }
}

```

```

sealed class Event {
    object Convert : Event()
    object Swap : Event()
    data class OpenTheAddTransactionSheet(val account: Account, val
amount: Float) : Event()
}

```

Файл *SelectDateDialogFragment.kt*

```

package com.beamerboy.money.feature.presentation.ui.date

import android.graphics.Color
import android.graphics.drawable.ColorDrawable
import android.os.Bundle
import android.view.View
import androidx.fragment.app.DialogFragment
import androidx.fragment.app.activityViewModels
import androidx.fragment.app.viewModels
import androidx.lifecycle.LifecycleScope
import by.kirich1409.viewbindingdelegate.viewBinding
import com.google.android.material.datepicker.MaterialDatePicker
import com.beamerboy.money.R
import com.beamerboy.money.common.DateUtils.toLocalDate
import com.beamerboy.money.databinding.DialogFragmentSelectDateBinding
import com.beamerboy.money.feature.presentation.ui.MainActivityViewModel
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.flow.collectLatest

@AndroidEntryPoint
class SelectDateDialogFragment :
DialogFragment(R.layout.dialog_fragment_select_date) {

    private val binding: DialogFragmentSelectDateBinding by viewBinding()

    private val viewModel: SelectDateViewModel by viewModels()
    private val activityViewModel: MainActivityViewModel by
activityViewModels()

    override fun onCreateView(view: View, savedInstanceState: Bundle?) {
        super.onCreateView(view, savedInstanceState)

        setupOnClickListeners()
        setupEventCollector()
    }

    override fun onStart() {
        super.onStart()

        dialog?.window?.setBackgroundDrawable(ColorDrawable(Color.TRANSPARENT))
    }

    private fun setupOnClickListeners() {
        with(binding) {
            selectDate.setOnClickListener { viewModel.selectDateClick() }
            today.setOnClickListener { viewModel.selectTodayClick() }
            week.setOnClickListener { viewModel.selectWeekClick() }
        }
    }

```

```

        month.setOnClickListener { viewModel.selectMonthClick() }
        year.setOnClickListener { viewModel.selectYearClick() }
        allTime.setOnClickListener { viewModel.selectAllTimeClick() }
    }

    private fun setupEventCollector() {
        lifecycleScope.launchWhenStarted {
            viewModel.events.collectLatest {
                when (it) {
                    is SelectDateViewModel.Event.SelectDate -> {
                        val datePicker =
MaterialDatePicker.Builder.datePicker()
                            .setTitleText(getString(R.string.select_date))

                        .setSelection(MaterialDatePicker.todayInUtcMilliseconds())
                            .setTheme(R.style.MaterialCalendarTheme)
                            .build()

                        datePicker.addOnPositiveButtonClickListener {
milliseconds ->
                            val date = milliseconds.toLocalDate()
                            activityViewModel.setCurrentDateRange(date, date)
                            dismiss()
                        }
                        datePicker.show(childFragmentManager,
DATE_PICKER_TAG)
                    }
                    is SelectDateViewModel.Event.SelectToday -> {
                        val date = viewModel.getDate()
                        activityViewModel.setCurrentDateRange(date, date)
                        dismiss()
                    }
                    is SelectDateViewModel.Event.SelectWeek -> {
                        val from = viewModel.getDate(WEEK)
                        activityViewModel.setCurrentDateRange(from, null)
                        dismiss()
                    }
                    is SelectDateViewModel.Event.SelectMonth -> {
                        val from = viewModel.getDate(MONTH)
                        activityViewModel.setCurrentDateRange(from, null)
                        dismiss()
                    }
                    is SelectDateViewModel.Event.SelectYear -> {
                        val from = viewModel.getDate(YEAR)
                        activityViewModel.setCurrentDateRange(from, null)
                        dismiss()
                    }
                    is SelectDateViewModel.Event.SelectAllTime -> {
                        activityViewModel.setCurrentDateRange(null, null)
                        dismiss()
                    }
                }
            }
        }
    }

    companion object {
        private const val DATE_PICKER_TAG = "date_picker_tag"
        private const val WEEK = 7
    }
}

```

```
private const val MONTH = 30
private const val YEAR = 365
}
```

					ПГУ1.090301.36.001 ПЗ	Лист
						87
Изм.	Лист	№ докум.	Подпись	Дата		

Файл *SelectDateViewModel.kt*

```
package com.beamerboy.money.feature.presentation.ui.date

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.beamerboy.money.common.DateUtils.DAY_IN_MS
import com.beamerboy.money.common.DateUtils.getCurrentLocalDate
import com.beamerboy.money.common.DateUtils.toLocalDate
import com.beamerboy.money.common.DateUtils.toMilliseconds
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.asSharedFlow
import kotlinx.coroutines.launch
import java.time.LocalDate

class SelectDateViewModel : ViewModel() {

    private val _events = MutableSharedFlow<Event>()
    val events = _events.asSharedFlow()

    fun getDate(daysAgo: Int = 0): LocalDate {
        return if (daysAgo != 0)
            (getCurrentLocalDate().toMilliseconds() - (daysAgo *
DAY_IN_MS)).toLocalDate()
        else getCurrentLocalDate()
    }

    fun selectDateClick() {
        viewModelScope.launch {
            _events.emit(Event.SelectDate)
        }
    }

    fun selectTodayClick() {
        viewModelScope.launch {
            _events.emit(Event.SelectToday)
        }
    }

    fun selectWeekClick() {
        viewModelScope.launch {
            _events.emit(Event.SelectWeek)
        }
    }

    fun selectMonthClick() {
        viewModelScope.launch {
            _events.emit(Event.SelectMonth)
        }
    }

    fun selectYearClick() {
        viewModelScope.launch {
            _events.emit(Event.SelectYear)
        }
    }

    fun selectAllTimeClick() {
        viewModelScope.launch {
            _events.emit(Event.SelectAllTime)
        }
    }
}
```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.36.001 ПЗ

Лист

88

```

    }
}

sealed class Event {
    object SelectDate : Event()
    object SelectToday : Event()
    object SelectWeek : Event()
    object SelectMonth : Event()
    object SelectYear : Event()
    object SelectAllTime : Event()
}
}

```

Файл *SettingsActivity.kt*

```

package com.beamerboy.money.feature.presentation.ui.settings

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import by.kirich1409.viewbindingdelegate.viewBinding
import com.beamerboy.money.R
import com.beamerboy.money.databinding.ActivitySettingsBinding
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint
class SettingsActivity : AppCompatActivity(R.layout.activity_settings) {

    private val binding: ActivitySettingsBinding by viewBinding()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(binding.root)

        setSupportActionBar(binding.toolbar)
        supportActionBar?.setDisplayHomeAsUpEnabled(true)

        if (savedInstanceState == null) {
            supportFragmentManager
                .beginTransaction()
                .replace(R.id.settings, SettingsFragment())
                .commit()
        }
    }

    override fun onSupportNavigateUp(): Boolean {
        onBackPressed()
        return super.onSupportNavigateUp()
    }

    override fun onStart() {
        overridePendingTransition(R.anim.slide_in_left,
            R.anim.slide_out_right)
        super.onStart()
    }

    override fun onPause() {
        super.onPause()
        overridePendingTransition(R.anim.slide_in_right,

```

```

R.anim.slide_out_left)
    }
}

```

Файл *SettingsFragment.kt*

```

package com.beamerboy.money.feature.presentation.ui.settings

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.preference.ListPreference
import androidx.preference.Preference
import androidx.preference.PreferenceFragmentCompat
import com.beamerboy.money.R
import com.beamerboy.money.feature.presentation.ui.auth.AuthActivity
import com.beamerboy.money.feature.presentation.utils.Utils.FIRST_TIME_PREFERENCE_KEY
import com.beamerboy.money.feature.presentation.utils.Utils.RESET_PIN_PREFERENCE_KEY
import com.beamerboy.money.feature.presentation.utils.Utils.THEME_DARK
import com.beamerboy.money.feature.presentation.utils.Utils.THEME_LIGHT
import com.beamerboy.money.feature.presentation.utils.Utils.THEME_PREFERENCE_KEY
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint
class SettingsFragment : PreferenceFragmentCompat() {

    private var counter = 0

    override fun onCreatePreferences(savedInstanceState: Bundle?, rootKey: String?) {
        setPreferencesFromResource(R.xml.root_preferences, rootKey)

        findPreference<ListPreference>(THEME_PREFERENCE_KEY)?.setOnPreferenceChangeListener { _, newValue ->
            AppCompatActivity().setDefaultNightMode(
                when (newValue) {
                    THEME_LIGHT -> AppCompatActivity.MODE_NIGHT_NO
                    THEME_DARK -> AppCompatActivity.MODE_NIGHT_YES
                    else -> AppCompatActivity.MODE_NIGHT_FOLLOW_SYSTEM
                }
            )
            true
        }

        findPreference<Preference>(RESET_PIN_PREFERENCE_KEY)?.setOnPreferenceClickListener { preference ->
            preference.summary = getString(R.string.reset_pin_message_repeat)
            counter++
            if (counter > 1) {
                preferenceManager.sharedPreferences
                    .edit()
                    .putBoolean(FIRST_TIME_PREFERENCE_KEY, true)
            }
        }
    }
}

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		90

```

        .apply()

        val intent = Intent(activity, AuthActivity::class.java)
        intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or
Intent.FLAG_ACTIVITY_CLEAR_TASK
        startActivity(intent)
    }
    false
}
}
}
}

```

Файл *AccountFilterDialogFragment.kt*

```

package
com.beamerboy.money.feature.presentation.ui.transactions.account_filter

import android.graphics.Color
import android.graphics.drawable.ColorDrawable
import android.os.Bundle
import android.view.View
import androidx.fragment.app.DialogFragment
import androidx.fragment.app.activityViewModels
import androidx.fragment.app.viewModels
import androidx.lifecycle.lifecycleScope
import androidx.recyclerview.widget.LinearLayoutManager
import by.kirich1409.viewbindingdelegate.viewBinding
import com.beamerboy.money.R
import com.beamerboy.money.common.DateUtils.toAmountFormat
import com.beamerboy.money.databinding.DialogFragmentAccountFilterBinding
import com.beamerboy.money.feature.presentation.ui.MainActivityViewModel
import com.beamerboy.money.feature.presentation.ui.accounts.AccountsRVAdapter
import com.beamerboy.money.feature.presentation.utils.Utils.getDivider
import com.beamerboy.money.feature.presentation.utils.Utils.setTint
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject
import kotlinx.coroutines.flow.collectLatest

@AndroidEntryPoint
class AccountFilterDialogFragment :
DialogFragment(R.layout.dialog_fragment_account_filter) {

    private val binding: DialogFragmentAccountFilterBinding by viewBinding()

    private val viewModel: AccountFilterViewModel by viewModels()
    private val activityViewModel: MainActivityViewModel by
activityViewModels()

    @Inject
    lateinit var accountsAdapter: AccountsRVAdapter

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        setupRecyclerView()
        setupAllAccountsItem()
        setupCollectors()
    }

    override fun onStart() {

```

```

        super.onStart()

        dialog?.window?.setBackgroundDrawable(ColorDrawable(Color.TRANSPARENT))
    }

    private fun setupRecyclerView() {
        binding.listOfAccounts.apply {
            adapter = accountsAdapter
            layoutManager = LinearLayoutManager(requireContext())
            addItemDecoration(getDivider(requireContext()))
        }

        accountsAdapter.setOnClickListener(
            AccountsRVAdapter.OnClickListener {
                viewModel.selectAccount(it)
                dismiss()
            }
        )
    }

    private fun setupAllAccountsItem() {
        with(binding) {
            allAccountsCurrency.text = activityViewModel.getCurrency()

            allAccountsIconColor.setTint(activityViewModel.selectedAccount.value?.color)
            allAccountsItem.setOnClickListener {
                activityViewModel.setCurrentAccount(null)
                dismiss()
            }
        }
    }

    private fun setupCollectors() {
        lifecycleScope.launchWhenStarted {
            viewModel.accounts.collectLatest { newList ->
                accountsAdapter.submitList(newList)
                binding.allAccountsAmount.text =
                    viewModel.getFullAmount().toAmountFormat(withMinus =
false)
            }
        }

        lifecycleScope.launchWhenStarted {
            viewModel.events.collectLatest {
                when (it) {
                    is AccountFilterViewModel.Event.SelectAccount -> {
                        activityViewModel.setCurrentAccount(it.account)
                    }
                }
            }
        }
    }
}

```

Файл AccountFilterViewModel.kt

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		92

```

package
com.beamerboy.money.feature.presentation.ui.transactions.account_filter

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.beamerboy.money.feature.domain.model.Account
import com.beamerboy.money.feature.domain.usecases.GetAccountsUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.Job
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asSharedFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.launchIn
import kotlinx.coroutines.flow.onEach
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
class AccountFilterViewModel @Inject constructor(
    private val getAccountsUseCase: GetAccountsUseCase
) : ViewModel() {

    private val _accounts = MutableStateFlow(emptyList<Account>())
    val accounts = _accounts.asStateFlow()

    private val _events = MutableSharedFlow<Event>()
    val events = _events.asSharedFlow()

    private var getAccountsJob: Job? = null

    init {
        getAccounts()
    }

    private fun getAccounts() {
        getAccountsJob?.cancel()
        getAccountsJob = getAccountsUseCase()
            .onEach { accounts ->
                _accounts.value = accounts
            }
            .launchIn(viewModelScope)
    }

    fun selectAccount(account: Account) {
        viewModelScope.launch {
            _events.emit(Event.SelectAccount(account))
        }
    }

    fun getFullAmount(): Double {
        return _accounts.value.sumOf { it.amount }
    }

    sealed class Event {
        data class SelectAccount(val account: Account) : Event()
    }
}

```

Файл *AddTransactionSheetFragment.kt*

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		93

```

package com.beamerboy.money.feature.presentation.ui.transactions.add

import android.graphics.Color
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.viewModels
import androidx.lifecycle.lifecycleScope
import androidx.navigation.fragment.DialogFragmentNavigator
import androidx.navigation.fragment.FragmentNavigator
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs
import com.google.android.material.bottomsheet.BottomSheetDialogFragment
import com.beamerboy.money.R
import com.beamerboy.money.common.DateUtils.toAmountFormat
import com.beamerboy.money.databinding.SheetFragmentAddTransactionBinding
import com.beamerboy.money.feature.domain.model.Transaction
import com.beamerboy.money.feature.presentation.utils.Utils.setIcon
import com.beamerboy.money.feature.presentation.utils.Utils.showToast
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.flow.collectLatest

@AndroidEntryPoint
class AddTransactionSheetFragment : BottomSheetDialogFragment() {

    private var _binding: SheetFragmentAddTransactionBinding? = null
    private val binding get() = _binding!!

    private val viewModel: AddTransactionViewModel by viewModels()

    private val args by navArgs<AddTransactionSheetFragmentArgs>()

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = SheetFragmentAddTransactionBinding.inflate(inflater,
container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        setupContainerAppearance()
        setupEventCollector()
    }

    private fun setupContainerAppearance() {
        with(binding) {
            accountName.text = args.selectedAccount.name
            categoryName.text = args.selectedCategory.name
            categoryIcon.setIcon(args.selectedCategory.icon)

            if (args.amount != 0f)
                expenseTextField.editText?.setText(
                    args.amount.toDouble().toAmountFormat(withMinus = false)
                )
        }
    }

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		94

```

accountBackground.setBackgroundColor(Color.parseColor(args.selectedAccount.co
lor))

categoryBackground.setBackgroundColor(Color.parseColor(args.selectedCategory.
iconColor))
    }
}

private fun setupEventCollector() {
    binding.applyButton.setOnClickListener { viewModel.applyButtonClick()
}

    lifecycleScope.launchWhenStarted {
        viewModel.events.collectLatest {
            when (it) {
                is AddTransactionViewModel.Event.AddTransaction -> {
                    val account = args.selectedAccount
                    val category = args.selectedCategory
                    if (account.id != null) {
                        val amount =

binding.expenseTextField.editText?.text.toString().toDoubleOrNull()
                        if (amount == null || amount <= 0)
                            showToast(context,
getString(R.string.enter_expense_error))
                        else {
                            val note =
binding.noteTextField.editText?.text.toString()

                            val transaction = Transaction(
                                note = note,
                                amount = amount,
                                accountId = account.id,
                                categoryId = category.id
                            )

                            viewModel.addTransaction(transaction)

                            if (getCurrentDestination() ==
this@AddTransactionSheetFragment.javaClass.name) {
                                findNavController().navigate(
                                    AddTransactionSheetFragmentDirections
.actionAddTransactionSheetFragmentToTransactionsFragment()
                                )
                            }
                        }
                    }
                }
            }
        }
    }

    override fun onDestroy() {
        super.onDestroy()
        _binding = null
    }
}

```

```

        private fun getCurrentDestination() =
            (findNavController().currentDestination as?
            FragmentNavigator.Destination)?.className
            ?: (findNavController().currentDestination as?
            DialogFragmentNavigator.Destination)?.className
    }

```

Файл *AddTransactionViewModel.kt*

```

package com.beamerboy.money.feature.presentation.ui.transactions.add

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.beamerboy.money.feature.domain.model.Transaction
import com.beamerboy.money.feature.domain.usecases.AddTransactionUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.asSharedFlow
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
class AddTransactionViewModel @Inject constructor(
    private val addTransactionUseCase: AddTransactionUseCase
) : ViewModel() {

    private val _events = MutableSharedFlow<Event>()
    val events = _events.asSharedFlow()

    suspend fun addTransaction(transaction: Transaction) {
        addTransactionUseCase(transaction)
    }

    fun applyButtonClick() {
        viewModelScope.launch {
            _events.emit(Event.AddTransaction)
        }
    }

    sealed class Event {
        object AddTransaction : Event()
    }
}

```

Файл *CategoriesRVAdapter.kt*

```

package com.beamerboy.money.feature.presentation.ui.transactions.select_category

import android.content.SharedPreferences
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		96

```

import com.beamerboy.money.common.DateUtils.toAmountFormat
import com.beamerboy.money.databinding.ItemCategoryBinding
import com.beamerboy.money.feature.domain.model.CategoryView
import com.beamerboy.money.feature.presentation.utils.Utils.CURRENCY_PREFERENCE_KEY
import com.beamerboy.money.feature.presentation.utils.Utils.MAIN_CURRENCY
import com.beamerboy.money.feature.presentation.utils.Utils.setIcon
import com.beamerboy.money.feature.presentation.utils.Utils.setTint
import javax.inject.Inject
import javax.inject.Singleton

@Singleton
class CategoriesRVAdapter @Inject constructor(
    private val sharedPreferences: SharedPreferences
) : ListAdapter<CategoryView,
CategoriesRVAdapter.CategoryViewHolder>(DIFF_CALLBACK) {

    private var onClickListener: OnClickListener? = null

    inner class CategoryViewHolder(private val binding: ItemCategoryBinding)
:
        RecyclerView.ViewHolder(binding.root) {

        fun bind(categoryView: CategoryView) {
            with(binding) {
                icon.setIcon(categoryView.icon)
                iconBackground.setTint(categoryView.iconColor)

                name.text = categoryView.name
                amount.text = categoryView.amount.toAmountFormat(withMinus =
false)
                currency.text =
sharedPreferences.getString(CURRENCY_PREFERENCE_KEY, MAIN_CURRENCY)

                name.isSelected = true
            }

            itemView.alpha = if (categoryView.amount == 0.0) 0.3f else 1f
            itemView.setOnClickListener {
                onClickListener?.onClick(categoryView)
            }
        }

        override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
CategoryViewHolder {
            val inflater = LayoutInflater.from(parent.context)
            val binding = ItemCategoryBinding.inflate(inflater, parent,
false)
            return CategoryViewHolder(binding)
        }

        override fun onBindViewHolder(holder: CategoryViewHolder, position: Int)
{
            holder.bind(getItem(position))
        }

        fun setOnClickListener(onClickListener: OnClickListener) {
            this.onClickListener = onClickListener
        }
    }
}

```

```

class OnClickListener(val clickListener: (category: CategoryView) ->
Unit) {
    fun onClick(category: CategoryView) = clickListener(category)
}

companion object DIFF_CALLBACK : DiffUtil.ItemCallback<CategoryView>() {
    override fun areItemsTheSame(oldItem: CategoryView, newItem:
CategoryView): Boolean {
        return oldItem.id == newItem.id
    }

    override fun areContentsTheSame(oldItem: CategoryView, newItem:
CategoryView): Boolean {
        return oldItem.hashCode() == newItem.hashCode()
    }
}
}

```

Файл *SelectCategorySheetFragment.kt*

```

package
com.beamerboy.money.feature.presentation.ui.transactions.select_category

import android.content.SharedPreferences
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.beamerboy.money.common.DateUtils.toAmountFormat
import com.beamerboy.money.databinding.ItemCategoryBinding
import com.beamerboy.money.feature.domain.model.CategoryView
import com.beamerboy.money.feature.presentation.utils.Utils.CURRENCY_PREFERENCE_KEY
import com.beamerboy.money.feature.presentation.utils.Utils.MAIN_CURRENCY
import com.beamerboy.money.feature.presentation.utils.Utils.setIcon
import com.beamerboy.money.feature.presentation.utils.Utils.setTint
import javax.inject.Inject
import javax.inject.Singleton

@Singleton
class CategoriesRVAdapter @Inject constructor(
    private val sharedPreferences: SharedPreferences
) : ListAdapter<CategoryView,
CategoriesRVAdapter.CategoryViewHolder>(DIFF_CALLBACK) {

    private var onClickListener: OnClickListener? = null

    inner class CategoryViewHolder(private val binding: ItemCategoryBinding)
:
    RecyclerView.ViewHolder(binding.root) {

        fun bind(categoryView: CategoryView) {
            with(binding) {
                icon.setIcon(categoryView.icon)
                iconBackground.setTint(categoryView.iconColor)
            }
        }
    }
}

```

```

        name.text = categoryView.name
        amount.text = categoryView.amount.toAmountFormat(withMinus =
false)
        currency.text =
sharedPreferences.getString(CURRENCY_PREFERENCE_KEY, MAIN_CURRENCY)

        name.isSelected = true
    }

    itemView.alpha = if (categoryView.amount == 0.0) 0.3f else 1f
    itemView.setOnClickListener {
        onClickListener?.onClick(categoryView)
    }
}

}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
CategoryViewHolder {
    val inflater = LayoutInflater.from(parent.context)
    val binding = ItemCategoryBinding.inflate(inflater, parent,
false)
    return CategoryViewHolder(binding)
}

override fun onBindViewHolder(holder: CategoryViewHolder, position: Int)
{
    holder.bind(getItem(position))
}

fun setOnClickListener(onClickListener: OnClickListener) {
    this.onClickListener = onClickListener
}

class OnClickListener(val clickListener: (category: CategoryView) ->
Unit) {
    fun onClick(category: CategoryView) = clickListener(category)
}

companion object DIFF_CALLBACK : DiffUtil.ItemCallback<CategoryView>() {
    override fun areItemsTheSame(oldItem: CategoryView, newItem:
CategoryView): Boolean {
        return oldItem.id == newItem.id
    }

    override fun areContentsTheSame(oldItem: CategoryView, newItem:
CategoryView): Boolean {
        return oldItem.hashCode() == newItem.hashCode()
    }
}
}

```

Файл *SelectCategoryViewModel.kt*

```

package
com.beamerboy.money.feature.presentation.ui.transactions.select_category

import android.content.SharedPreferences
import android.view.LayoutInflater

```

```

import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.beamerboy.money.common.DateUtils.toAmountFormat
import com.beamerboy.money.databinding.ItemCategoryBinding
import com.beamerboy.money.feature.domain.model.CategoryView
import com.beamerboy.money.feature.presentation.utils.Utils.CURRENCY_PREFERENCE_KEY
import com.beamerboy.money.feature.presentation.utils.Utils.MAIN_CURRENCY
import com.beamerboy.money.feature.presentation.utils.Utils.setIcon
import com.beamerboy.money.feature.presentation.utils.Utils.setTint
import javax.inject.Inject
import javax.inject.Singleton

@Singleton
class CategoriesRVAdapter @Inject constructor(
    private val sharedPreferences: SharedPreferences
) : ListAdapter<CategoryView,
CategoriesRVAdapter.CategoryViewHolder>(DIFF_CALLBACK) {

    private var onClickListener: OnClickListener? = null

    inner class CategoryViewHolder(private val binding: ItemCategoryBinding)
:
        RecyclerView.ViewHolder(binding.root) {

        fun bind(categoryView: CategoryView) {
            with(binding) {
                icon.setIcon(categoryView.icon)
                iconBackground.setTint(categoryView.iconColor)

                name.text = categoryView.name
                amount.text = categoryView.amount.toAmountFormat(withMinus =
false)

                currency.text =
sharedPreferences.getString(CURRENCY_PREFERENCE_KEY, MAIN_CURRENCY)

                name.isSelected = true
            }

            itemView.alpha = if (categoryView.amount == 0.0) 0.3f else 1f
            itemView.setOnClickListener {
                onClickListener?.onClick(categoryView)
            }
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
CategoryViewHolder {
        val inflater = LayoutInflater.from(parent.context)
        val binding = ItemCategoryBinding.inflate(inflater, parent,
false)
        return CategoryViewHolder(binding)
    }

    override fun onBindViewHolder(holder: CategoryViewHolder, position: Int)
{
        holder.bind(getItem(position))
    }
}

```

```

    fun setOnClickListener(onClickListener: OnClickListener) {
        this.setOnClickListener = onClickListener
    }

    class OnClickListener(val clickListener: (category: CategoryView) ->
Unit) {
        fun onClick(category: CategoryView) = clickListener(category)
    }

    companion object DIFF_CALLBACK : DiffUtil.ItemCallback<CategoryView>() {
        override fun areItemsTheSame(oldItem: CategoryView, newItem:
CategoryView): Boolean {
            return oldItem.id == newItem.id
        }

        override fun areContentsTheSame(oldItem: CategoryView, newItem:
CategoryView): Boolean {
            return oldItem.hashCode() == newItem.hashCode()
        }
    }
}

```

Файл *TransactionsFragment.kt*

```

package com.beamerboy.money.feature.presentation.ui.transactions

import android.annotation.SuppressLint
import android.os.Bundle
import android.view.Menu
import android.view.MenuInflater
import android.view.MenuItem
import android.view.View
import androidx.appcompat.app.AlertDialog
import androidx.core.view.isVisible
import androidx.fragment.app.Fragment
import androidx.fragment.app.activityViewModels
import androidx.fragment.app.viewModels
import androidx.lifecycle.LifecycleScope
import androidx.navigation.fragment.DialogFragmentNavigator
import androidx.navigation.fragment.FragmentNavigator
import androidx.navigation.fragment.findNavController
import androidx.recyclerview.widget.DividerItemDecoration
import androidx.recyclerview.widget.LinearLayoutManager
import by.kirich1409.viewbindingdelegate.viewBinding
import com.beamerboy.money.R
import com.beamerboy.money.databinding.FragmentTransactionsBinding
import com.beamerboy.money.feature.domain.model.TransactionView
import com.beamerboy.money.feature.presentation.ui.MainActivityViewModel
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject
import kotlinx.coroutines.flow.collectLatest

@AndroidEntryPoint
class TransactionsFragment : Fragment(R.layout.fragment_transactions) {

    private val binding: FragmentTransactionsBinding by viewBinding()

```

```

private val viewModel: TransactionsViewModel by viewModels()
private val activityViewModel: MainActivityViewModel by
activityViewModels()

@Inject
lateinit var transactionAdapter: TransactionsRVAdapter

override fun onCreateView(view: View, savedInstanceState: Bundle?) {
    super.onCreateView(view, savedInstanceState)
    setHasOptionsMenu(true)

    setupRecyclerView()
    setupCollectors()

    binding.newTransactionButton.setOnClickListener {
        viewModel.addTransactionClick(
            activityViewModel.selectedAccount.value ?:
            activityViewModel.accounts.value[0]
        )
    }
}

private fun setupRecyclerView() {
    binding.transactionsRecyclerView.apply {
        adapter = transactionAdapter
        layoutManager = LinearLayoutManager(requireContext())
        addItemDecoration(
            DividerItemDecoration(requireContext(),
            DividerItemDecoration.VERTICAL)
        )
        itemAnimator = null
    }

    transactionAdapter.setOnDeleteClickListener(
        TransactionsRVAdapter.OnDeleteClickListener {
            viewModel.deleteButtonClick(it)
        }
    )
}

private fun setupCollectors() {
    lifecycleScope.launchWhenStarted {
        viewModel.transactionsUiState.collectLatest {
            when (it) {
                is TransactionsViewModel.UiState.Ready -> {
                    with(binding) {
                        progressBar.isVisible = false
                        noTransaction.visibility =
                            if (it.transactions.isEmpty()) View.VISIBLE
                        else View.INVISIBLE
                    }
                    transactionAdapter.submitList(it.transactions)
                }
                is TransactionsViewModel.UiState.Loading -> {
                    binding.progressBar.isVisible = true
                }
                else -> Unit
            }
        }
    }
}

```

```

        lifecycleScope.launchWhenStarted {
            activityViewModel.selectedDateRange.collectLatest {
                viewModel.setDateRange(it.first, it.second)
            }
        }

        lifecycleScope.launchWhenStarted {
            activityViewModel.selectedAccount.collectLatest {
                viewModel.setSelectedAccount(it)
            }
        }

        lifecycleScope.launchWhenStarted {
            viewModel.events.collectLatest {
                when (it) {
                    is TransactionsViewModel.Event.OpenTheAddTransactionSheet
-> {
                        if (getCurrentDestination() ==
this@TransactionsFragment.javaClass.name) {
                            findNavController().navigate(
TransactionsFragmentDirections.actionTransactionsFragmentToSelectCategorySheetFragment(
                                it.account
                            )
                        )
                    }
                    is TransactionsViewModel.Event.SelectDate -> {
                        if (getCurrentDestination() ==
this@TransactionsFragment.javaClass.name) {
                            findNavController().navigate(
TransactionsFragmentDirections.actionTransactionsFragmentToSelectDateDialogFragment()
                        )
                    }
                }
            }
            is TransactionsViewModel.Event.ShowTheDeleteTransactionDialog -> {
                if (!alertIsShowing) {
                    showAlertDialog(it.transaction)
                }
            }
            is TransactionsViewModel.Event.DeleteTransaction -> {
                viewModel.deleteTransaction(it.transaction)
            }
        }
    }
}

private var alertIsShowing = false

private fun showAlertDialog(transaction: TransactionView) {
    alertIsShowing = true

    val alert = AlertDialog.Builder(requireContext())
        .setIcon(R.drawable.ic_warning)

```

```

        .setTitle(getString(R.string.alert_transaction_title))
        .setMessage("От: ${transaction.categoryName}\nК:
${transaction.accountName}\nСумма: ${transaction.amount}")
        .setPositiveButton(getString(R.string.ok)) { _, _ ->
            viewModel.deleteConfirmationButtonClick(transaction)
            transactionAdapter.clearSelectedPosition()
            alertIsShowing = false
        }
        .setNegativeButton(getString(R.string.cancel)) { _, _ ->
            alertIsShowing = false
        }
        .setOnCancelListener { alertIsShowing = false }
        .create()
        alert.show()
    }

    private var currentCurrency: String = ""

    @SuppressWarnings("NotifyDataSetChanged")
    override fun onStart() {
        super.onStart()
        if (currentCurrency != activityViewModel.getCurrency())
            transactionAdapter.notifyDataSetChanged()
    }

    override fun onPause() {
        super.onPause()
        transactionAdapter.clearSelectedPosition()
    }

    override fun onStop() {
        super.onStop()
        currentCurrency = activityViewModel.getCurrency()
    }

    override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {
        inflater.inflate(R.menu.date_menu, menu)
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        if (item.itemId == R.id.select_date) {
            viewModel.selectDateClick()
        }
        return super.onOptionsItemSelected(item)
    }

    private fun getCurrentDestination() =
        (findNavController().currentDestination as?
        FragmentNavigator.Destination)?.className
        ?: (findNavController().currentDestination as?
        DialogFragmentNavigator.Destination)?.className
    }

```

Файл TransactionsRVAdapter.kt

```
package com.beamerboy.money.feature.presentation.ui.transactions
```

```
import android.content.SharedPreferences
```

```

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.beamerboy.money.common.DateUtils.toAmountFormat
import com.beamerboy.money.databinding.ItemDayInfoBinding
import com.beamerboy.money.databinding.ItemTransactionBinding
import com.beamerboy.money.feature.domain.model.DayInfo
import com.beamerboy.money.feature.domain.model.TransactionView
import com.beamerboy.money.feature.presentation.utils.Utils.CURRENCY_PREFERENCE_KEY
import com.beamerboy.money.feature.presentation.utils.Utils.MAIN_CURRENCY
import com.beamerboy.money.feature.presentation.utils.Utils.setIcon
import com.beamerboy.money.feature.presentation.utils.Utils.setTint
import javax.inject.Inject
import javax.inject.Singleton

@Singleton
class TransactionsRVAdapter @Inject constructor(
    private val sharedPreferences: SharedPreferences
) : ListAdapter<Any, TransactionsRVAdapter.ItemViewHolder>(DIFF_CALLBACK) {

    private var onDeleteClickListener: OnDeleteClickListener? = null

    abstract class ItemViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        abstract fun bind(item: Any)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
    ItemViewHolder {
        val inflater = LayoutInflater.from(parent.context)
        return when (viewType) {
            TRANSACTION_VIEW_TYPE -> {
                val binding = ItemTransactionBinding.inflate(inflater,
parent, false)
                TransactionViewHolder(binding)
            }
            else -> {
                val binding = ItemDayInfoBinding.inflate(inflater,
parent, false)
                DayInfoViewHolder(binding)
            }
        }
    }

    override fun onBindViewHolder(holder: ItemViewHolder, position: Int) {
        when (holder.itemViewType) {
            TRANSACTION_VIEW_TYPE -> {
                val viewHolder = holder as TransactionViewHolder
                viewHolder.bind(getItem(position))
            }
            else -> {
                val viewHolder = holder as DayInfoViewHolder
                viewHolder.bind(getItem(position))
            }
        }
    }
}

```

```

        override fun getItemViewType(position: Int): Int {
            return if (getItem(position) is TransactionView)
TRANSACTION_VIEW_TYPE else DAY_INFO_VIEW_TYPE
        }

        private var selectedPosition: Int = RecyclerView.NO_POSITION

        fun clearSelectedPosition() {
            selectedPosition = RecyclerView.NO_POSITION
        }

        inner class TransactionViewHolder(
            private val binding: ItemTransactionBinding
        ) : ItemViewHolder(binding.root) {

            override fun bind(item: Any) {
                val transactionView = item as TransactionView

                with(binding) {
                    iconBackground.setTint(transactionView.iconColor)
                    icon.setIcon(transactionView.icon)

                    categoryName.text = transactionView.categoryName
                    cardName.text = transactionView.accountName
                    note.text = transactionView.note
                    textAmount.text =
transactionView.amount.toAmountFormat(withMinus = true)
                    textCurrency.text =
                        sharedPreferences.getString(CURRENCY_PREFERENCE_KEY,
MAIN_CURRENCY)

                    note.isSelected = true
                    categoryName.isSelected = true
                    cardName.isSelected = true
                }

                configureIfSelected(transactionView)

                itemView.setOnClickListener {
                    if (bindingAdapterPosition == selectedPosition)
                        itemSelecting(false)
                }

                itemView.setOnLongClickListener {
                    itemSelecting(bindingAdapterPosition != selectedPosition)
                    true
                }
            }

            private fun configureIfSelected(transactionView: TransactionView) {
                val isSelected = bindingAdapterPosition == selectedPosition
                with(binding) {
                    note.visibility = if (isSelected) View.INVISIBLE else
View.VISIBLE
                    textAmount.visibility = if (isSelected) View.INVISIBLE else
View.VISIBLE
                    textCurrency.visibility = if (isSelected) View.INVISIBLE else
View.VISIBLE
                    deleteButton.visibility = if (!isSelected) View.INVISIBLE

```

```

else View.VISIBLE

        deleteButton.setOnClickListener {
            onDeleteClickListener?.onClick(transactionView)
        }
    }

    private fun itemSelecting(activate: Boolean) {
        notifyItemChanged(selectedPosition)
        selectedPosition = if (activate) bindingAdapterPosition else
RecyclerView.NO_POSITION
        notifyItemChanged(selectedPosition)
    }
}

inner class DayInfoViewHolder(
    private val binding: ItemDayInfoBinding
) : ItemViewHolder(binding.root) {

    override fun bind(item: Any) {
        val dayInfo = item as DayInfo

        val dateValue = dayInfo.transactionDate
        val amountValue = dayInfo.amountPerDay
        val monthAndYearValue = "${dateValue.month} ${dateValue.year}"

        with(binding) {
            amount.text = amountValue.toAmountFormat(withMinus = true)
            currency.text =
sharedPreferences.getString(CURRENCY_PREFERENCE_KEY, MAIN_CURRENCY)

            day.text = dateValue.dayOfMonth.toString()
            monthAndYear.text = monthAndYearValue
            dayOfWeek.text = dateValue.dayOfWeek.name
        }
    }

    fun setOnDeleteClickListener(onDeleteClickListener:
OnDeleteClickListener) {
        this.onDeleteClickListener = onDeleteClickListener
    }

    class OnDeleteClickListener(val clickListener: (transaction:
TransactionView) -> Unit) {
        fun onClick(transaction: TransactionView) =
clickListener(transaction)
    }

    companion object {
        private const val TRANSACTION_VIEW_TYPE = 1
        private const val DAY_INFO_VIEW_TYPE = 0

        object DIFF_CALLBACK : DiffUtil.ItemCallback<Any>() {
            override fun areItemsTheSame(oldItem: Any, newItem: Any): Boolean
{
                return if (oldItem is DayInfo && newItem is DayInfo)
                    oldItem.transactionDate == newItem.transactionDate
                else if (oldItem is TransactionView && newItem is

```



```

val events = _events.asSharedFlow()

private val _selectedAccount = MutableStateFlow<Account?>(null)
private val _selectedDateRange = MutableStateFlow<Pair<LocalDate?,
LocalDate?>>(null to null)

private var getTransactionsJob: Job? = null

init {
    getTransactions()
}

private fun getTransactions() {
    _transactionsUiState.value = UiState.Loading

    getTransactionsJob?.cancel()

    val range = _selectedDateRange.value
    val account = _selectedAccount.value

    getTransactionsJob =
        getTransactionViewsUseCase(range, account)
            .onEach { transactions ->
                val data =
getTransactionsWithDayInfoUseCase(transactions, range, account)
                _transactionsUiState.value = UiState.Ready(data)
            }
            .launchIn(viewModelScope)
}

suspend fun deleteTransaction(transaction: TransactionView) {
    deleteTransactionByIdUseCase(transaction)
}

fun setDateRange(from: LocalDate? = null, to: LocalDate? = null) {
    _selectedDateRange.value = from to to
    getTransactions()
}

fun setSelectedAccount(account: Account? = null) {
    _selectedAccount.value = account
    getTransactions()
}

fun selectDateClick() {
    viewModelScope.launch {
        _events.emit(Event.SelectDate)
    }
}

fun addTransactionClick(account: Account) {
    viewModelScope.launch {
        _events.emit(Event.OpenTheAddTransactionSheet(account))
    }
}

fun deleteButtonClick(transaction: TransactionView) {
    viewModelScope.launch {
        _events.emit(Event.ShowTheDeleteTransactionDialog(transaction))
    }
}

```

```

    }

    fun deleteConfirmationButtonClick(transaction: TransactionView) {
        viewModelScope.launch {
            _events.emit(Event.DeleteTransaction(transaction))
        }
    }

    sealed class UiState {
        data class Ready(val transactions: List<Any>) : UiState()
        object Loading : UiState()
        object Idle : UiState()
    }

    sealed class Event {
        object SelectDate : Event()
        data class OpenTheAddTransactionSheet(val account: Account) : Event()
        data class ShowTheDeleteTransactionDialog(val transaction: TransactionView) : Event()
        data class DeleteTransaction(val transaction: TransactionView) : Event()
    }
}

```

Файл *MainActivity.kt*

```

package com.beamerboy.money.feature.presentation.ui

import android.os.Bundle
import android.view.View
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.LifecycleScope
import androidx.navigation.NavController
import androidx.navigation.fragment.NavHostFragment
import androidx.navigation.ui.AppBarConfiguration
import androidx.navigation.ui.setupActionBarWithNavController
import androidx.navigation.ui.setupWithNavController
import by.kirich1409.viewbindingdelegate.viewBinding
import com.beamerboy.money.R
import com.beamerboy.money.common.DateUtils.getCurrentLocalDate
import com.beamerboy.money.databinding.ActivityMainBinding
import com.beamerboy.money.feature.presentation.ui.accounts.AccountsFragmentDirections
import com.beamerboy.money.feature.presentation.ui.chart.ChartFragmentDirections
import com.beamerboy.money.feature.presentation.ui.converter.CurrencyConverterFragmentDirections
import com.beamerboy.money.feature.presentation.ui.transactions.TransactionsFragmentDirections
import dagger.hilt.android.AndroidEntryPoint
import java.time.format.DateTimeFormatter
import java.time.format.FormatStyle
import kotlinx.coroutines.flow.collectLatest

@AndroidEntryPoint
class MainActivity : AppCompatActivity(R.layout.activity_main) {

```

					ПГУ1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		110

```

private val binding: ActivityMainBinding by viewBinding(R.id.container)

private val viewModel: MainActivityViewModel by viewModels()

private val navController: NavController by lazy {
    (supportFragmentManager.findFragmentById(R.id.fragment_container) as
NavHostFragment).navController
}

private var currentDestination: Int? = null

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setTheme(R.style.Theme_MoneyManager)
    setContentView(binding.root)

    setSupportActionBar(binding.toolbar)
    supportActionBar?.setDisplayShowTitleEnabled(false)

    setupEventCollector()
    setupSelectedValuesCollector()
    setupBottomNavigation()
    setupOnDestinationChangeListener()

    with(binding) {
        buttonSettings.setOnClickListener {
viewModel.settingsButtonClick() }
        toolbarInfoBox.setOnClickListener {
viewModel.selectAccountButtonClick() }
    }

    private fun setupEventCollector() {
        lifecycleScope.launchWhenStarted {
            viewModel.events.collectLatest {
                when (it) {
                    is MainActivityViewModel.Event.OpenTheSettingsScreen -> {
                        when (currentDestination) {
                            R.id.currency_converter_fragment ->
navController.navigate(
CurrencyConverterFragmentDirections.actionCurrencyConverterFragmentToSettings
Activity()
                                )
                            R.id.accounts_fragment -> navController.navigate(
AccountsFragmentDirections.actionAccountsFragmentToSettingsActivity()
                                )
                            R.id.transactions_fragment ->
navController.navigate(
TransactionsFragmentDirections.actionTransactionsFragmentToSettingsActivity()
                                )
                            R.id.chart_fragment -> navController.navigate(
ChartFragmentDirections.actionChartFragmentToSettingsActivity()
                                )
                        }
                    }
                }
            }
        }
    }
}

```

```

        is MainActivityViewModel.Event.OpenTheSelectAccountDialog
-> {
    navController.navigate(R.id.action_dialog_fragment_account_filter)
    }
    }
    }
    }

    private fun setupSelectedValuesCollector() {
        lifecycleScope.launchWhenStarted {
            viewModel.selectedAccount.collectLatest {
                with(binding) {
                    bankIcon.visibility = if (it != null) View.VISIBLE else
View.INVISIBLE
                    toolbarTitle.text = it?.name ?:
getString(R.string.all_accounts)
                }
            }
        }

        val pattern = DateTimeFormatter.ofLocalizedDate(FormatStyle.MEDIUM)
        lifecycleScope.launchWhenStarted {
            viewModel.selectedDateRange.collectLatest {
                binding.toolbarSubtitle.text =
                    if (it.first == null && it.second == null)
                        getString(R.string.all_time)
                    else if (it.second == null)
                        "${it.first?.format(pattern)} -
${getCurrentLocalDate().format(pattern)}"
                    else
                        it.first?.format(pattern)
            }
        }
    }

    private fun setupBottomNavigation() {
        val appBarConfiguration = AppBarConfiguration(
            listOf(
                R.id.currency_converter_fragment,
                R.id.accounts_fragment,
                R.id.transactions_fragment,
                R.id.chart_fragment
            )
        )
        setupActionBarWithNavController(navController, appBarConfiguration)
        binding.bottomNavigation.setupWithNavController(navController)
    }

    private fun setupOnDestinationChangedListener() {
        fun changeAppAndBottomBarState(visibility: Int) {
            with(binding) {
                bottomNavigation.visibility = visibility
                buttonSettings.visibility = visibility
                toolbarInfoBox.visibility = visibility
            }
        }

        navController.addOnDestinationChangedListener { _, destination, _ ->

```

```

        currentDestination = destination.id

        val isFragmentWithoutBars = when (currentDestination) {
            R.id.account_add_fragment -> true
            R.id.account_edit_fragment -> true
            else -> false
        }

        changeAppBarAndBottomBarState(if (isFragmentWithoutBars) View.GONE
else View.VISIBLE)

        supportActionBar?.setDisplayShowTitleEnabled(isFragmentWithoutBars)

        val isFragmentWithFilter = when (currentDestination) {
            R.id.accounts_fragment -> false
            R.id.account_actions_sheet_fragment -> false
            else -> true
        }

        with(binding) {
            moreButton.visibility = if (!isFragmentWithFilter)
View.INVISIBLE else View.VISIBLE
            toolbarInfoBox.isEnabled = isFragmentWithFilter
        }
    }

    override fun onSupportNavigateUp(): Boolean {
        return navController.navigateUp() || super.onSupportNavigateUp()
    }
}

```

Файл *MainActivityViewModel.kt*

```

package com.beamerboy.money.feature.presentation.ui

import android.content.SharedPreferences
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.beamerboy.money.common.DateUtils.getCurrentLocalDate
import com.beamerboy.money.feature.domain.model.Account
import com.beamerboy.money.feature.domain.usecases.GetAccountsUseCase
import com.beamerboy.money.feature.presentation.utils.Utils.CURRENCY_PREFERENCE_KEY
import com.beamerboy.money.feature.presentation.utils.Utils.MAIN_CURRENCY
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.Job
import kotlinx.coroutines.flow.MutableSharedFlow
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asSharedFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.launchIn
import kotlinx.coroutines.flow.onEach
import kotlinx.coroutines.launch
import java.time.LocalDate
import javax.inject.Inject

@HiltViewModel

```

```

class MainActivityViewModel @Inject constructor(
    private val sharedPreferences: SharedPreferences,
    private val getAccountsUseCase: GetAccountsUseCase
) : ViewModel() {

    private val _accounts = MutableStateFlow(emptyList<Account>())
    val accounts = _accounts.asStateFlow()

    private val _selectedAccount = MutableStateFlow<Account?>(null)
    val selectedAccount = _selectedAccount.asStateFlow()

    private val _selectedDateRange = MutableStateFlow<Pair<LocalDate?,
    LocalDate?>>()
    val selectedDateRange = _selectedDateRange.asStateFlow()

    private val _events = MutableSharedFlow<Event>()
    val events = _events.asSharedFlow()

    private var getAccountsJob: Job? = null

    init {
        getAccounts()
    }

    private fun getAccounts() {
        getAccountsJob?.cancel()
        getAccountsJob = getAccountsUseCase()
            .onEach { accounts -> _accounts.value = accounts }
            .launchIn(viewModelScope)
    }

    fun getCurrency() =
        sharedPreferences.getString(CURRENCY_PREFERENCE_KEY, MAIN_CURRENCY)
?: MAIN_CURRENCY

    fun settingsButtonClick() {
        viewModelScope.launch {
            _events.emit(Event.OpenTheSettingsScreen)
        }
    }

    fun selectAccountButtonClick() {
        viewModelScope.launch {
            _events.emit(Event.OpenTheSelectAccountDialog)
        }
    }

    fun setCurrentAccount(account: Account?) {
        viewModelScope.launch {
            _selectedAccount.value = account
        }
    }

    fun setCurrentDateRange(begin: LocalDate?, end: LocalDate?) {
        viewModelScope.launch {
            _selectedDateRange.value = begin to end
        }
    }
}

```

```
sealed class Event {  
    object OpenTheSettingsScreen : Event()  
    object OpenTheSelectAccountDialog : Event()  
}
```

					ПГУ1.090301.36.001 ПЗ	Лист
						115
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(обязательное)

Сервис по учету личных расходов

Презентация

					ПГУ1.090301.36.001 ПЗ	Лист
						116
Изм.	Лист	№ докум.	Подпись	Дата		

Тема работы:
«Сервис учета
личных
расходов»

- Автор работы студент группы 19ВВП2 Трубаненко А.Г.
- Научный руководитель д.т.н. ВТ ПГУ Зинкин С.А.

• Пенза 2023

Цель и
объект
работы

- Цель работы:
разработка сервиса учета личных расходов, представленного в виде *мобильного приложения*.
- Объект работы:
Сервис будет содержать в себе:
 - создание механизма для добавления и категоризации расходов пользователем. Пользователь должен иметь возможность вводить информацию о своих расходах, указывать категорию, дату и сумму операции;
 - реализация автоматического анализа данных и генерации отчетов. Приложение должно предоставлять пользователю возможность просмотра сводных отчетов о расходах по категориям, времени и другим параметрам. Также желательно предоставить графическую визуализацию данных для более наглядного анализа;

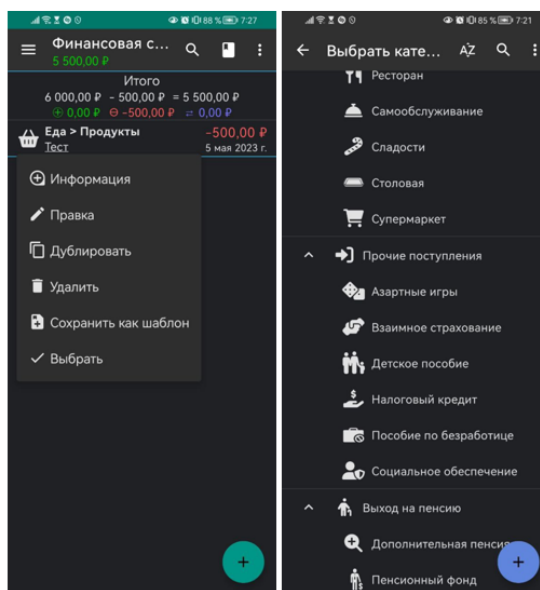
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.36.001 ПЗ

Лист

117

Аналоги



Аналоги

- Достоинства данного приложения:
 - учет доходов;
 - есть финансовая смета.
- Недостатки:
 - неудобное управление;
 - устаревший и громоздкий интерфейс;
 - нет диаграмм или графиков;
 - нет фильтров по категориям или счетам.

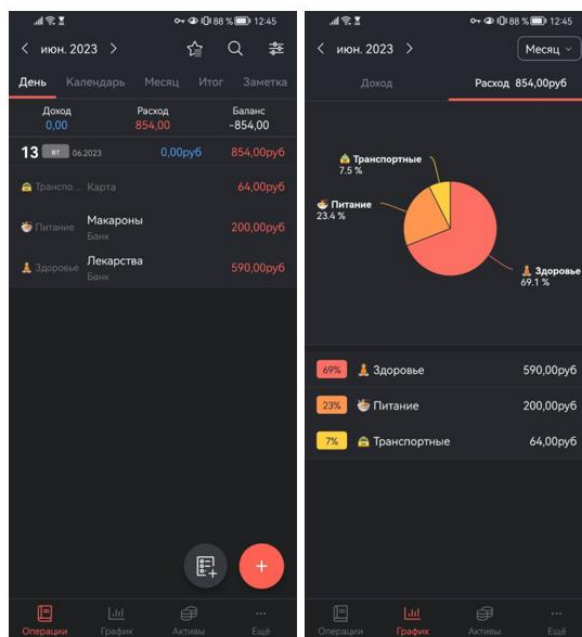
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.36.001 ПЗ

Лист

118

Аналоги



Анализ аналогов

- Достоинства данного приложения:
 - учет доходов;
 - есть диаграммы;
 - есть заметки;
 - есть повтор ежемесячных затрат.
- Недостатки:
 - плохо оптимизированно;
 - есть ограничение на создание счетов.

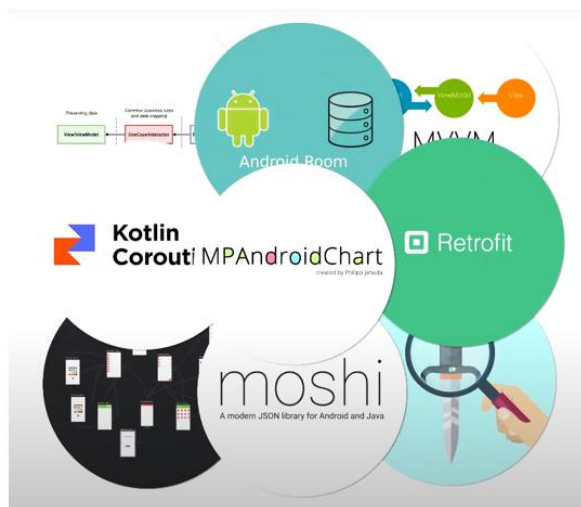
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.36.001 ПЗ

Лист

119

Стек технологий



Архитектура приложения

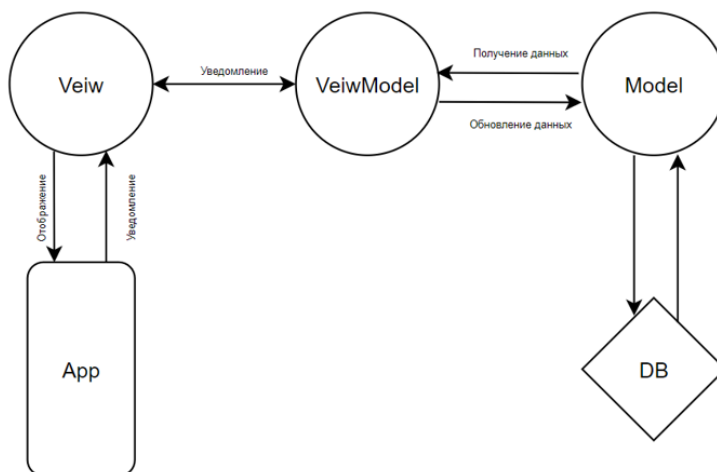
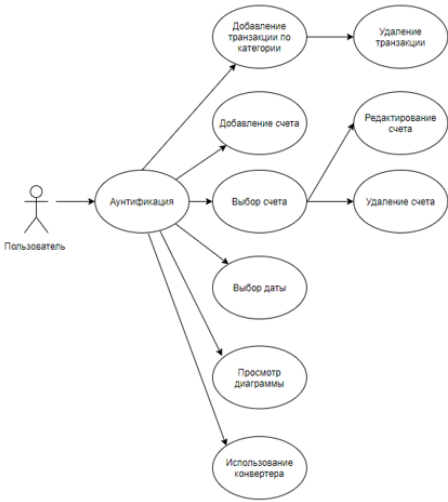
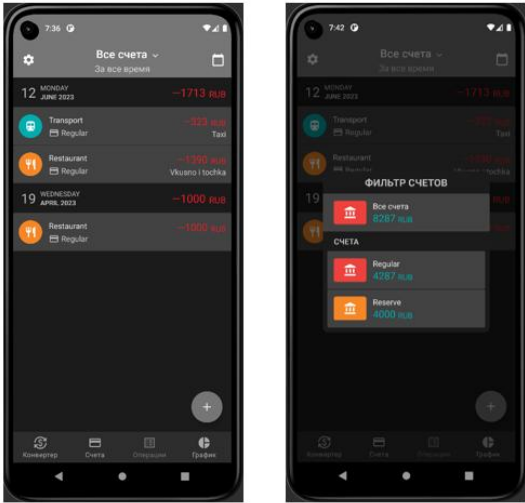


Диаграмма использования

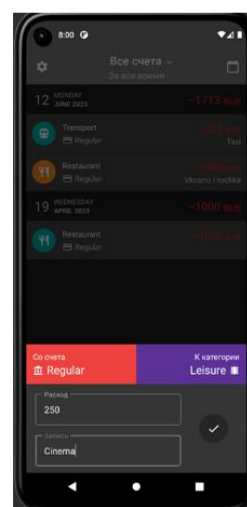


Демонстрация

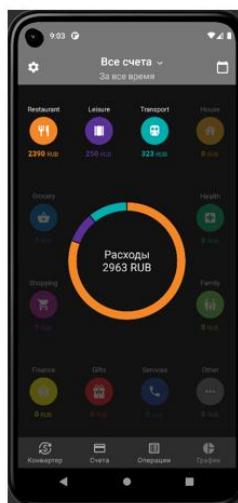
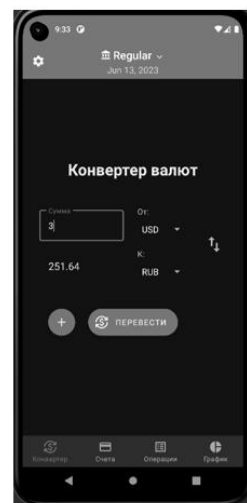


Изм.	Лист	№ докум.	Подпись	Дата

Демонстрация



Демонстрация



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.36.001 ПЗ

Лист

122

Заключение

Результатом выполнения данной выпускной квалификационной работы является сервис учета личных расходов с понятным и удобным интерфейсом. Были реализованы основные модули мобильного приложения, такие как авторизация пользователя, создание и фильтрация счетов, добавление расходов по категориям, отображения транзакций за определенный период и счет, был реализован конвертер валют.

Перспективы развития системы включают расширение функционала, добавление новых модулей, улучшение пользовательского интерфейса.

Спасибо за
внимание!

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.36.001 ПЗ

Лист

123