

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»

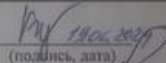
БАКАЛАВРСКАЯ РАБОТА

на тему

ANDROID-ПРИЛОЖЕНИЕ

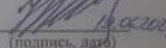
«МЕНЕДЖЕР ЛИЧНЫХ РАСХОДОВ»

Студент


(подпись, дата)

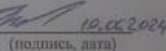
Кузин Данил Денисович
(ФИО полностью)

Руководитель


(подпись, дата)

Слепцов Н.В.
(фамилия, инициалы)

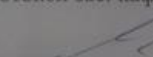
Нормоконтролёр


(подпись, дата)

Бычков А.С.
(фамилия, инициалы)

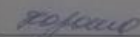
Работа допущена к защите (протокол зас. кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой



(протокол заседания ГЭК от

24.06.2024 г. № 34)

Секретарь ГЭК


(подпись)

Е.И. Гурин
(фамилия, инициалы)

Пенза 2024

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.12.001 ПЗ

Лист

3

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 19 » апреля 20 24 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Кузин Данил Денисович гр. 20ВВП1 факультета вычислительной техники.

2. Тема работы: Android-приложение «Менеджер личных расходов»

Тема утверждена приказом ПГУ № 405/о от "19" апреля 2024 г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Слепцов Н.В.

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

Разработать Android-приложение Менеджер личных расходов. Приложение должно выполнять функции добавления и удаления расходов, анализ бюджета, управления категориями расходов, добавления и редактирования плановых расходов.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области, изучить существующие аналоги, определить архитектуру приложения, выбрать инструменты для разработки, реализовать и провести тестирование приложения.

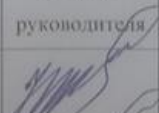
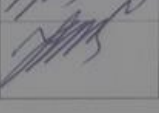
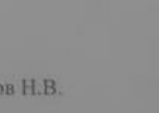
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.12.001 ПЗ

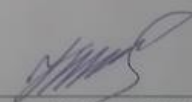
Лист

4

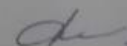
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	30	19.04.2024-22.04.2024	
2	Анализ требований	24	23.04.2024-25.04.2024	
3	Выбор технологий, применяемых при проектировании и разработке	40	26.04.2024-30.04.2024	
4	Выбор архитектуры приложения	30	01.05.2024-03.05.2024	
5	Разработка приложения	126	04.05.2024-22.05.2024	
6	Тестирование и отладка	50	23.05.2024-28.05.2024	
7	Оформление пояснительной записки	24	29.05.2024-31.05.2024	

Дата выдачи задания "19" апреля 2024 г.

Руководитель бакалаврской работы  Слепцов Н.В.
(подпись, дата)

Задание к исполнению принял студент  Кузин Д.Д.
(подпись, дата)

Работу к защите допустить
Декан факультета  20.06.2024 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 118 листов, 33 рисунка, 3 таблицы, 4 источника, 3 приложения

ANDROID STUDIO, JAVA, БАЗА ДАННЫХ, МЕНЕДЖЕР ЛИЧНЫХ РАСХОДОВ, SQLITE

Цель работы – разработать приложение, позволяющие контролировать личные расходы

Объектом разработки является Android-приложение, которое должно выполнять функции добавления и удаление затрат, создание и редактирование собственных категорий, анализ количества расходов по категориям, планирование долгосрочных покупок

В результате проделанной работы разработано приложение, полностью соответствующее целям данной выпускной квалификационной работы

ПГУ1.090301.12.001 ПЗ

					ПГУ1.090301.12.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Android-приложение «Менеджер личных расходов» Пояснительная записка	Лит.	Лист	Листов
Разраб.		Кузин Д.Д.	<i>[подпись]</i>	20.06.21				
Провер.		Сметнов И.В.	<i>[подпись]</i>	21.06.21			2	118
Н. Контр.		Бочкова А.С.	<i>[подпись]</i>	22.06.21				
Утверд.		Митрохин М.А.	<i>[подпись]</i>	22.06.21				
						ФВТ, каф. ВТ гр. 20ВВП1		

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.12.001 ПЗ

Лист

6

Содержание

Перечень сокращений и обозначений.....	9
Введение.....	10
1 Android-приложение «Менеджер личных расходов».....	11
1.1 Анализ предметной области	11
1.1.1 Анализ аналогов	12
1.1.2 Постановка задачи.....	15
1.2 Выбор инструментов разработки	17
1.2.1 Выбор среды разработки	17
1.2.1.1 Microsoft Visual Studio	17
1.2.1.2 IntelliJ IDEA	18
1.2.1.3 Android Studio	18
1.2.2 Выбор языка программирования	19
1.2.2.1 Java.....	19
1.2.2.2 Kotlin.....	20
1.2.3 Выбор СУБД	21
1.2.3.1 SQLite	21
1.2.3.2 Библиотека Room	22
1.2.4 Выбор архитектуры приложения.....	23
1.3 Реализация приложения	26
1.3.1 Графический интерфейс	26
1.3.1.1 Layout.....	26
1.3.1.2 Menu.....	27
1.3.1.3 Values.....	27
1.3.2 Java-классы.....	27
1.3.3 Структура приложения	30
1.3.4 Структура базы данных	32
1.4 Тестирование	33
1.4.1 Добавление и изменение баланса и доходов пользователя	33

1.4.2 Добавление категорий и бюджета	35
1.4.3 Управление транзакциями.....	38
1.4.4 Анализ бюджета	42
1.4.5 Управление планами	43
2 Безопасность жизнедеятельности.....	45
2.1 Требования к организации рабочего места	45
2.2 Требования к освещению	46
2.3 Требования к режиму отдыха	47
Заключение	48
Список использованных источников	49
Приложение А – UML-диаграммы.....	50
Приложение Б – Листинг программы	55
Приложение В – Презентация.....	115

Перечень сокращений и обозначений

БД – база данных

ОС – операционная система

ПК – персональный компьютер

СУБД – система управления базами данных

ЭВМ – электронно-вычислительная машина

API (Application Programming Interface) – программный интерфейс приложения

IDE (Integrated Development Enviroment) – интегрированная среда разработки

MVC (Model-View-Controller) – шаблон проектирования архитектуры приложения

SQL (Structured Query Language) – язык структурированных запросов

SDK(Software Development Kit) – комплект для разработки программного обеспечения)

UML (Unified Modeling Language) – унифицированный язык моделирования.

XML (Extensible Markup Language) – расширяемый язык разметки

Введение

В современном мире, контроль личных финансов является важнейшей частью в управление жизнью каждого человека. Разнообразие и количество финансовых операций, которые совершаются каждый день, создает спрос на удобные и эффективные способы для контроля и управления личными средствами. Одним из таких способов контроля над финансами является специализированные Android-приложения.

Основная цель данной выпускной работы – разработать приложения, которое будет закрывать необходимые потребности современного человека над контролем собственных финансов, позволит лучше прогнозировать дальнейшие расходы, а также поможет оптимизировать количество затрат. Данное приложение предоставляет возможности по удобному учету личных расходов, анализу общих затрат, а также планированию долгосрочных покупок.

Результатом данной работы будет создание приложения, отвечающего современным запросам в сфере разработки, которое будет сочетать в себе удобство в использовании и необходимые функциональные возможности, что поможет существенно повысить эффективность контроля над личными финансами.

1 Android-приложение «Менеджер личных расходов»

1.1 Анализ предметной области

В настоящее время контроль личными средств становится все более сложным и критически важным аспектом жизни каждого человека и Android-приложения по учету своих финансов занимают значимую роль в этом. Они помогают пользователю вести учет своих доходов и расходов, анализировать финансовые данные, устанавливать бюджеты и получать подробные отчеты о своих операциях. Такие приложения становятся незаменимым инструментом для тех, кто стремится контролировать свои финансы, пытается оптимизировать расходы и подходит более осознанно к организации своего бюджета. При проведении анализа по данной области, можно выделить несколько аспектов, которые являются значимыми при выборе приложения:

- функциональность. Android-приложение для управления личными финансами должно иметь набор необходимого функционала, такого как учет доходов и расходов, категоризация затрат, установление месячных бюджетов для различных категорий и возможность просматривать отчеты и аналитику по различным категориям в виде графиков и диаграмм для визуализации данных;

- удобство. При реализации приложения важным фактором является его удобство использования, так как это напрямую влияет на получаемый опыт пользователя от взаимодействия с вашим приложением. Основными способами для обеспечения комфортного использования приложения являются интуитивный и понятный интерфейс, который предоставляет удобную навигацию по приложению, соблюдение правильного расположения элементов дизайна, ввод данных позволяющий быстро добавлять и изменять финансовые транзакции, а также стабильная и высокая скорость работы приложения;

– дизайн. Визуальная составляющая является важной частью при выборе приложения, оно должно иметь простой и современный дизайн с интуитивно понятным расположением элементов навигации, приятную цветовую палитру, которая не будет перенапрягать пользователя излишним контрастом или обилием цветов, а также иконки и кнопки для управления интерфейсом, которые должны быть легко узнаваемыми и понятными для пользователя.

Таким образом, Android-приложения для управления личными финансами – это важный и очень востребованный в современном мире инструментом для пользователей, которые стремятся к контролю и оптимизации своих финансов.

1.1.1 Анализ аналогов

На данный момент существует множество приложений для контроля личных финансов. Среди этих аналогов можно выделить наиболее популярные:

- «Деньги ОК»;
- «CoinKeeper»;
- «Финансы – расходы и бюджет».

Рассмотрим приложение – Деньги ОК (рисунок 1).

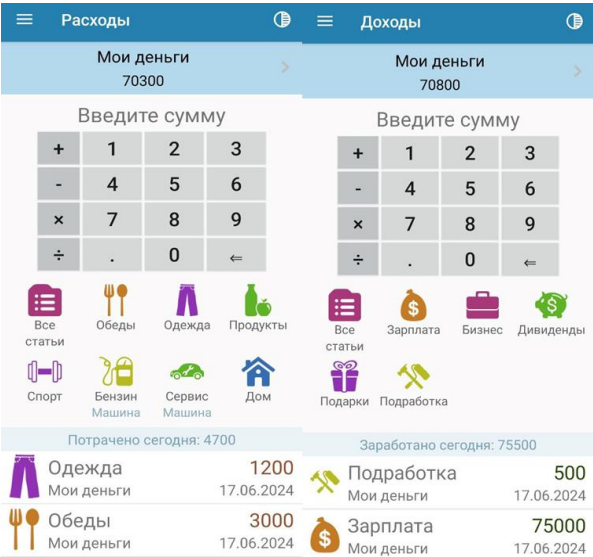


Рисунок 1 – Интерфейс приложения «Деньги ОК»

Рассмотрим плюсы и минусы данного приложения.

Плюсы:

- малый вес приложения;
- возможность создавать несколько счетов.

Минусы:

- неудобный интерфейс;
- устаревший дизайн;
- нет возможности создавать свои категории.

Рассмотрим приложение – CoinKeeper (рисунок 2,3).

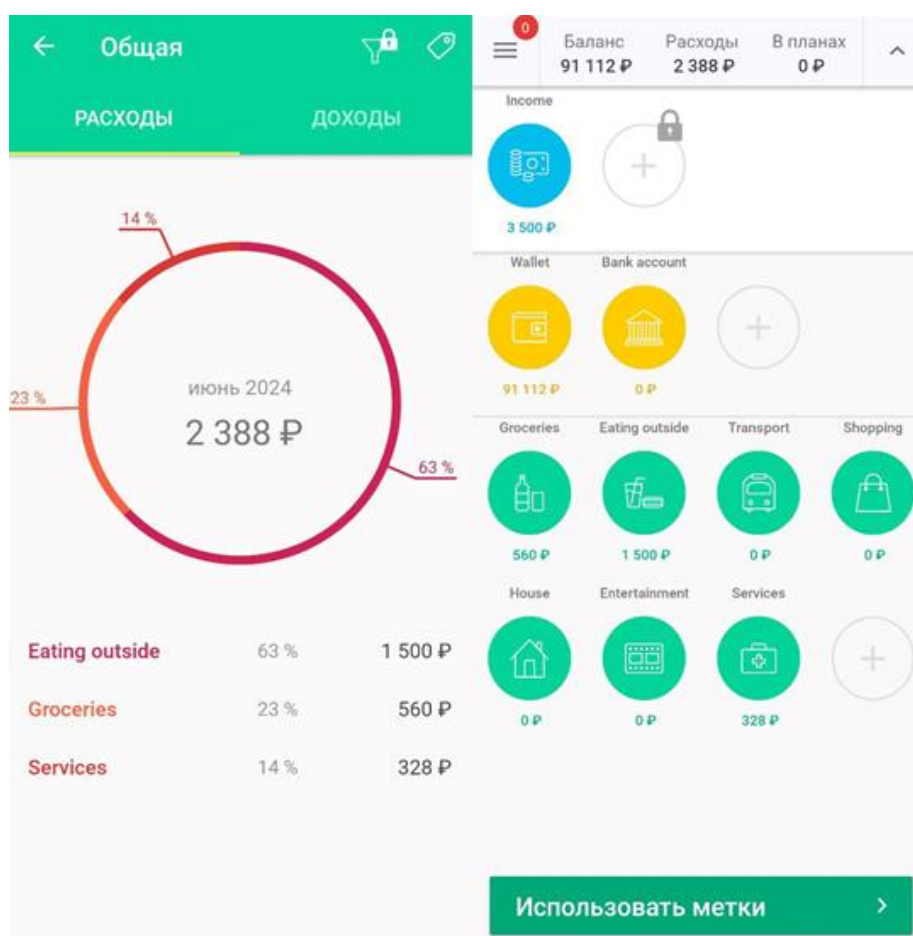


Рисунок 2 – Интерфейс приложения «CoinKeeper»

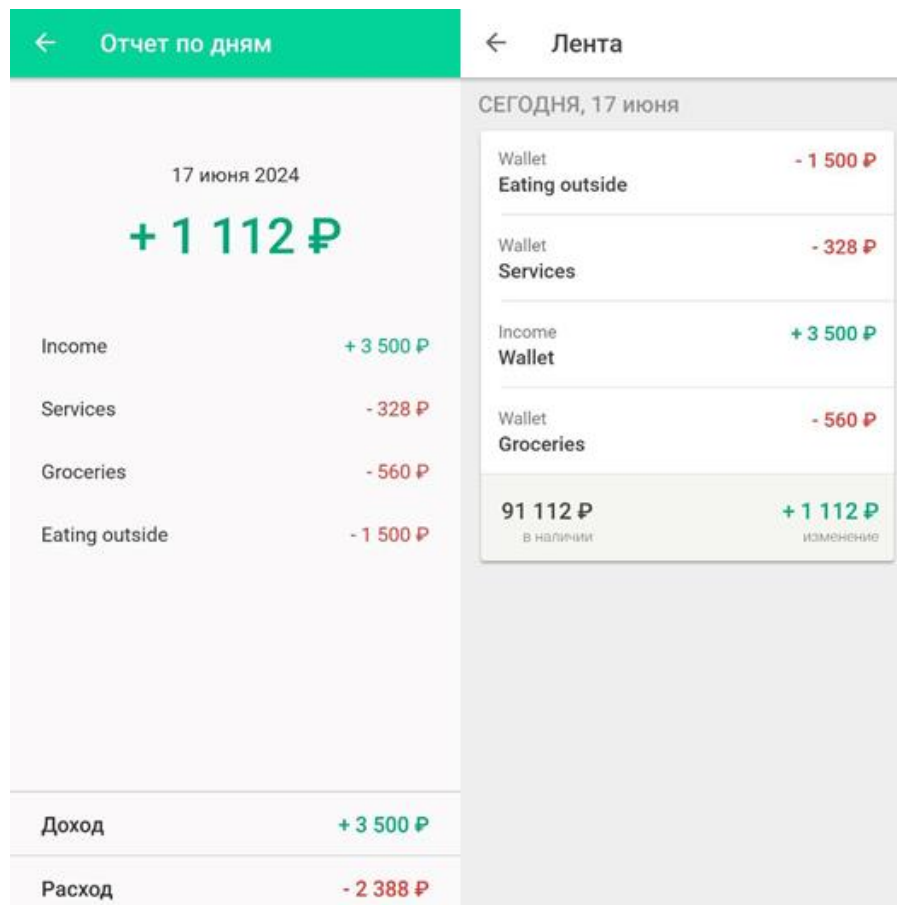


Рисунок 3 – Интерфейс приложения «CoinKeeper»

Рассмотрим плюсы и минусы данного приложения.

Плюсы:

- понятные графики для анализа затрат;
- наличия отчетов по дням.

Минусы:

- интуитивно непонятная система добавления расходов;
- малое количество категорий;
- ограниченный функционал для бесплатной версии.

Рассмотрим приложение – Финансы: расходы и бюджет (рисунок 4).

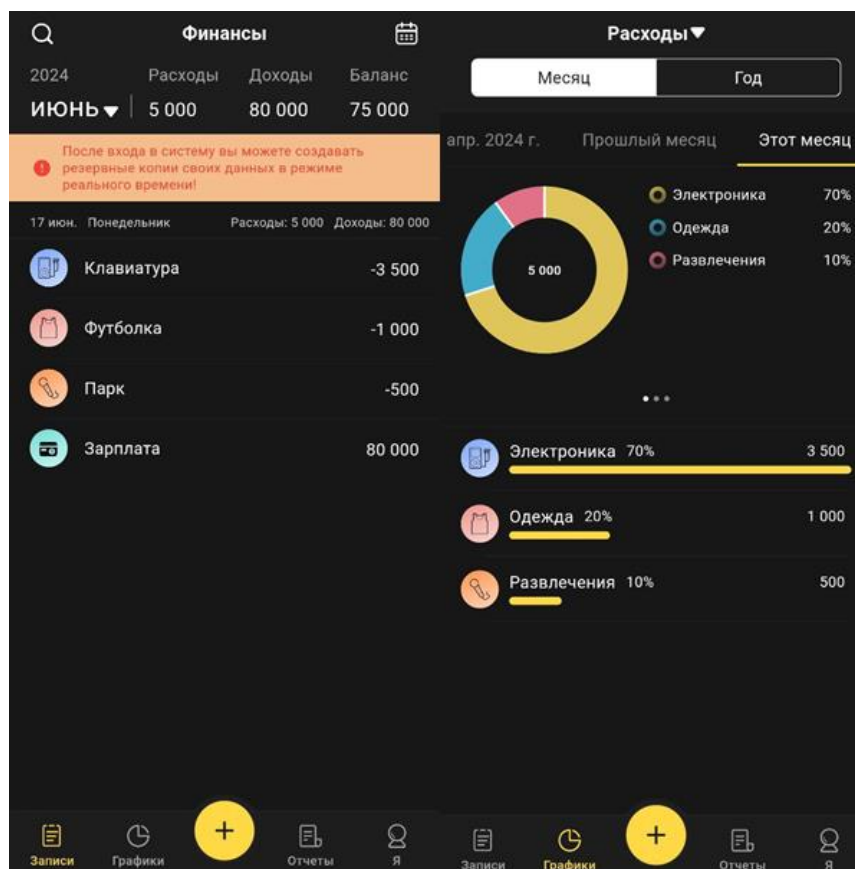


Рисунок 4 – Интерфейс приложения «Финансы – расходы и бюджет»

Рассмотрим плюсы и минусы данного приложения.

Плюсы:

- приятный дизайн;
- удобный интерфейс;
- наличие большого количества категорий;
- множество графиков для просмотра отчетности по расходам.

Минусы:

- нестабильная работа на слабых устройствах;
- отсутствует возможность создания нескольких счетов.

1.1.2 Постановка задачи

Цель – разработать Android приложение по управлению финансов. Оно должно включать в себя контроль расходов, сортировку по категориям, анализ общего бюджета и добавление планов на долгосрочные покупки.

Приложение должно содержать следующие критерии:

- понятный и простой интерфейс для комфортной навигации, а также приятный и интуитивный дизайн;
- возможность добавления, удаления и редактирования расходов, категорий и плановых покупок;
- реализация аналитических графиков для контроля общего бюджета по расходам;
- приложение должно содержать базовый функционал для контроля личного бюджета.

На рисунке 5 представлена диаграмма использования приложения.

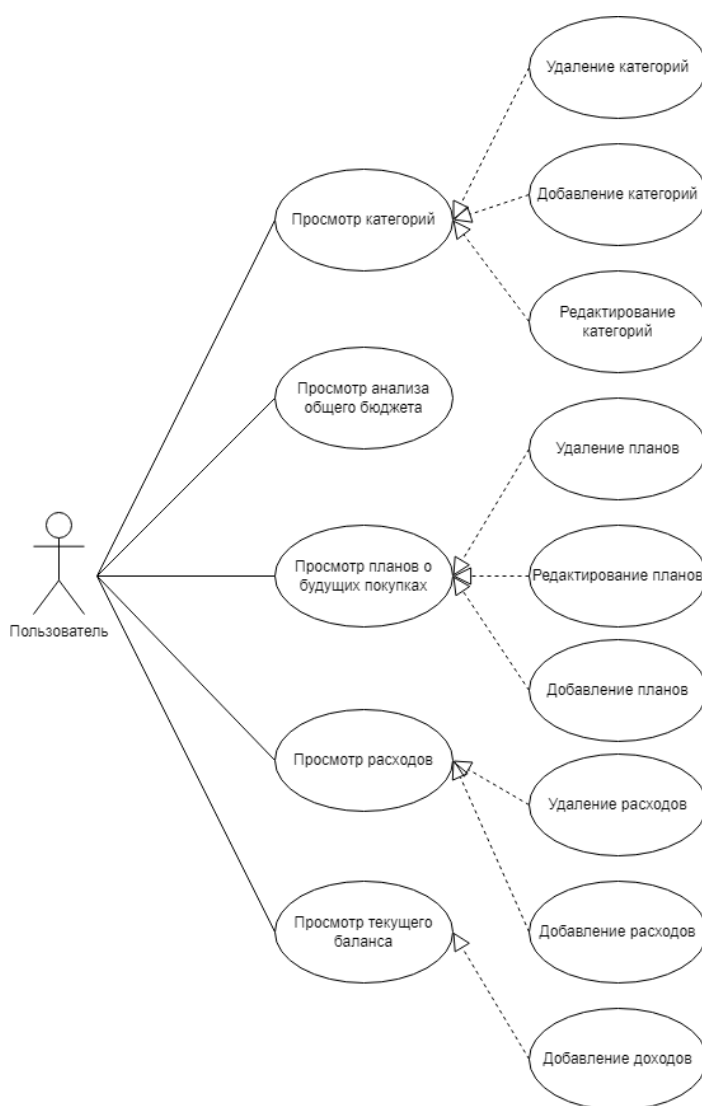


Рисунок 5 – UML-диаграмма прецедентов

1.2 Выбор инструментов разработки

В этом разделе будут изучены основные инструменты разработки, которые необходимы для реализации Android-приложения «Менеджер личных расходов».

1.2.1 Выбор среды разработки

Главным шагом перед созданием любого проекта является выбор среды разработки. Рассмотрим три основных IDE для реализаций Android-приложений: Microsoft Visual Studio, IntelliJ IDEA и Android Studio.

1.2.1.1 Microsoft Visual Studio

Microsoft Visual Studio — это интегрированная среда разработки, корпорации Microsoft для создания, документирования, запуска и отладки программ, написанных на различных языках программирования. Microsoft Visual Studio является мощным инструментом, который позволяет разрабатывать сложные программные комплексы, обладает широким спектром поддерживаемых языков программирования и платформ, а также предоставляет многочисленные инструменты для написания, отладки и тестирования кода [1].

Выделим основные плюсы и минусы данной IDE.

Плюсы:

- поддержка множества языков программирования и платформ;
- интеграция с различными сервисами и инструментами Microsoft;
- множество инструментов для отладки и тестирования;
- обширный набор настроек среды, что позволяет создать среду для необходимых нужд.

Минусы:

- высокие системные требования, что может негативно сказаться на производительности при разработке Android-приложений;

– более сложный процесс настройки для Android-разработки по сравнению с специализированными IDE;

– невозможность использовать более расширенный функционал без приобретения подписки, что не все могут себе позволить.

1.2.1.2 IntelliJ IDEA

IntelliJ IDEA — это мощная и гибкая IDE от JetBrains, которая поддерживает множество языков программирования и различных технологий. Она известна своей интеллектуальной поддержкой кода и многочисленными инструментами для рефакторинга и оптимизации кода.

Плюсы:

– поддержка множества языков программирования и различных технологий;

– мощные инструменты для редактирования и анализа кода;

– умный функционал автодополнения кода.

Минусы:

– из-за огромного функционала и сложного интерфейса в IntelliJ IDEA, настройка среды под необходимые нужды может занять больше времени;

– высокие системные требования, которые не позволяют заниматься комфортной разработкой на слабых вычислительных машинах;

– IntelliJ IDEA является платной, что может стать проблемой для студентов и разработчиков с недостаточным бюджетом.

1.2.1.3 Android Studio

Android Studio — официальная IDE от компании Google для разработки Android-приложений, которая основанная на IntelliJ IDEA. Она предоставляет все необходимые инструменты для создания, отладки и тестирования приложений, включая эмуляторы устройств и аналитику.

Рассмотрим преимущества и недостатки

Плюсы:

- полная интеграция с Android SDK и другими инструментами для разработки;
- множество встроенных инструментов для отладки и тестирования;
- интеграция с Git, что облегчает управление версиями кода и работу с репозиториями;
- регулярные обновления и поддержка от Google;
- поддержка Gradle для управления зависимостями и сборкой проектов.

Минусы:

- могут возникать проблемы с производительностью на менее мощных компьютерах;
- ограниченная поддержка других платформ и языков.

В свою очередь я сделал выбор в пользу Android Studio. Она обладает рядом преимуществ перед Microsoft Visual Studio и IntelliJ IDEA, когда речь идет о разработке Android-приложений. Она обеспечивает тесную интеграцию с Android SDK, регулярные обновления от Google и специализированные инструменты, которые значительно облегчают процесс разработки. Кроме того, встроенная поддержка Gradle упрощает управление зависимостями и сборкой проектов, делая разработку более эффективной и управляемой.

1.2.2 Выбор языка программирования

Выбор языка программирования также является важным аспектом при разработке Android-приложений. Рассмотрим два основных языка, используемых для этой цели: Java и Kotlin.

1.2.2.1 Java

Java — это один из самых популярных языков программирования в мире, который используется для разработки Android-приложений с самого начала существования платформы. Он обладает зрелой экосистемой и широким сообществом разработчиков [2].

Плюсы:

- широкая распространенность Java позволяет находить необходимую информацию для решения какой-либо задачи намного быстрее;
- большое количество библиотек и фреймворков, что позволяет упростить разработку;
- высокая производительность и стабильность;
- многопоточность.

Минусы:

- меньшая лаконичность по сравнению с современными языками, что делает код более громоздким;
- более высокая вероятность возникновения ошибок, связанных с нулевыми ссылками.

1.2.2.2 Kotlin

Kotlin — это современный язык программирования, разработанный JetBrains и поддерживаемый Google как официальный язык для разработки Android-приложений. Он призван устранить многие недостатки Java и обеспечить более лаконичный и безопасный синтаксис.

Плюсы:

- более лаконичный синтаксис по сравнению с Java;
- встроенная поддержка обработки нулевых значений, что снижает вероятность возникновения NullPointerException;
- совместимость с Java, что позволяет использовать существующие библиотеки и фреймворки;
- благодаря поддержке Google является более перспективным языком под Android-разработку.

Минусы:

- меньшая распространенность по сравнению с Java;

– из-за молодого возраста языка могут возникнуть сложности с поиском необходимой информации и документации.

Несмотря на преимущества Kotlin, Java по-прежнему остается предпочтительным выбором для многих разработчиков Android-приложений. Это обусловлено ее широкой распространенностью и зрелостью экосистемы, что обеспечивает доступ к большому количеству библиотек и фреймворков. Java также предоставляет высокую производительность и стабильность, что особенно важно для крупных и сложных проектов. Кроме того, простота синтаксиса делает разработку приложений по Android более простой и понятной.

1.2.3 Выбор СУБД

Для реализации этого приложения необходимо было выбрать решение для хранения базы данных и таблиц, куда будут записываться данные о расходах, категория и планах. Существует два способа реализации базы данных для Android-приложений: БД SQLite и библиотека Room.

1.2.3.1 SQLite

SQLite — это встроенная в Android база данных, предоставляющая все основные возможности СУБД без необходимости создания отдельного сервера. Она идеально подходит для приложений, которые требуют локального хранения данных пользователя [3].

Плюсы:

- легковесность. SQLite занимает минимальное пространство на устройстве и имеет низкие системные требования;
- производительность. Высокая производительность для операций чтения и записи, что важно для мобильных приложений с ограниченными ресурсами;
- широкая поддержка. SQLite поддерживается всеми версиями Android, что обеспечивает совместимость и стабильность;

- прямой доступ к SQL. Разработчики могут использовать стандартные SQL-запросы для работы с данными, что обеспечивает гибкость и мощность в управлении данными.

Минусы:

- отсутствие объектно-ориентированной модели. Работа с данными требует написания большого количества шаблонного кода для преобразования данных из SQL-запросов в объекты и обратно;
- отсутствие валидации запросов на этапе компиляции. Ошибки в SQL-запросах проявляются только во время выполнения, что может усложнить отладку.

1.2.3.2 Библиотека Room

Room — это библиотека для работы с базами данных в Android, разработанная Google. Room предоставляет объектно-ориентированную обертку над SQLite, упрощая работу с базой данных и обеспечивая дополнительную безопасность и производительность.

Плюсы:

- упрощенный код. Room позволяет работать с объектами вместо сырых SQL-запросов, что упрощает код и позволяет сконцентрироваться на логике запросов, а не на синтаксисе;
- автоматическая проверка запросов. Room проверяет SQL-запросы на этапе компиляции, что позволяет обнаружить ошибки до выполнения приложения;
- интеграция с другими компонентами. Room легко интегрируется с LiveData и ViewModel, что упрощает управление данными и их отображение;
- миграции базы данных. Room предоставляет встроенные инструменты для управления миграциями базы данных, что упрощает обновление структуры базы данных.

Минусы:

- недостаточный функционал. Библиотека Room хорошо подходит для простых запросов, но её функционала может быть недостаточно при создании более сложных запросов;
- отличный синтаксис от SQL. Для тех, кто привык работать с чистыми SQL-запросами, может потребоваться дополнительное время для освоения синтаксиса Room;
- проблемы с производительностью. Неправильное использование автоматического управления потоков может привести к различным проблемам с производительностью, например при выполнении масштабных запросов или операций с большим объемом данных, что может быть критично для работы на слабых вычислительных машинах.

При разработке приложения я сделал выбор в пользу БД SQLite по нескольким причинам:

- контроль SQL-запросов. Использование SQLite предоставляет полный контроль над выполнением SQL-запросов, что особенно важно для сложных запросов и оптимизации производительности;
- совместимость. SQLite поддерживается всеми версиями Android, обеспечивая стабильность и надежность работы приложения на любом устройстве;
- простота использования. SQLite лучше всего подходит для небольших проектов или проектов с простыми требованиями к работе с базой данных.

Таким образом, выбор SQLite был обусловлен необходимостью обеспечения максимальной производительности, гибкости и стабильности, а также простотой использования.

1.2.4 Выбор архитектуры приложения

Выбор архитектуры – это важный и необходимый этап разработки каждого приложения. С помощью неё определяется структура, взаимодействие между компонентами приложения, а также общую

управляемость проекта. При разработке была выбрана архитектура MVC (Model-View-Controller), которая является одним из самых популярных архитектурных шаблонов проектирования. Данный шаблон разделяет приложение на три основных компонента: модель, представление и контроллер.

Model – это способ представления данных и бизнес логики приложения. Главной её задачей является хранение и обработка данных, а также взаимодействие с БД. Модель никаким образом не привязана к View или Controller, что позволяет ей быть независимой и дает возможность быть использованной повторно.

View отвечает за отображение данных пользователю. Оно получает сообщение об изменениях от модели и отображает их в удобном для пользователя виде. Представление может быть графическим интерфейсом, веб-страницей или любым другим способом визуализации данных. Особенность View заключается в том, что оно не содержит логики и не взаимодействует напрямую с пользователем.

Controller является посредником между моделью и представлением. Он получает запросы от пользователя через представление и передает их модели для обработки. Controller также получает данные от модели и обновляет View, чтобы отобразить изменения пользователю. Controller не содержит бизнес-логики и не обрабатывает данные самостоятельно.

Взаимодействие между моделью, представлением и контроллером происходит посредством событий и сообщений. Когда данные в модели изменяются, она генерирует событие, которое передается контроллеру. Контроллер обновляет представление, чтобы отобразить измененные данные. Когда пользователь взаимодействует с представлением, оно генерирует сообщение, которое передается контроллеру для обработки.

Преимущества использования архитектурного шаблона MVC включают разделение ответственностей между компонентами, повышение

повторного использования кода, улучшение расширяемости и облегчение тестирования приложения. MVC также облегчает сопровождение и разработку приложения, поскольку каждый компонент может быть изменен независимо от других [3].

Выбор в пользу архитектуры MVC обусловлен наличием ряда преимуществ, которые делают её предпочтительным выбором для разработки Android-приложений:

- разделение обязанностей. MVC четко разделяет компоненты между собой, что повышает удобство поддержки и дает возможность для повторного использования;
- легкость модифицирования. Отсутствие связей между компонентами, позволяет легко модифицировать и изменять структуру приложения;
- поддержка нескольких визуальных представлений. MVC предоставляет возможность для разного визуального представления для одних и тех же данных.

Схема паттерна архитектуры MVC представлена на рисунке 6.

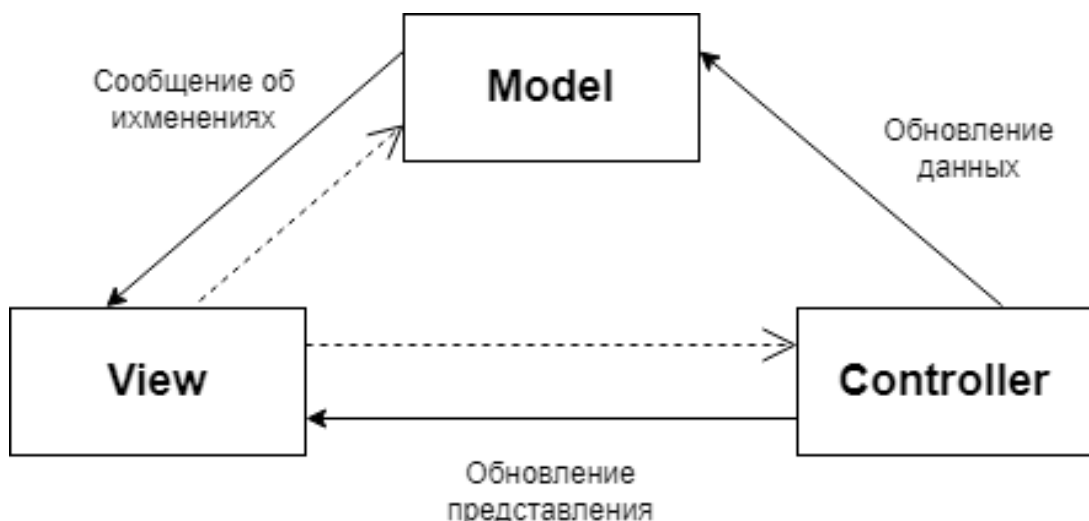


Рисунок 6 – Архитектура MVC

1.3 Реализация приложения

1.3.1 Графический интерфейс

Для того, чтобы приложение смогло привлечь потенциальных пользователей, оно должно обладать приятным дизайном и понятным интерфейсом.

1.3.1.1 Layout

В Android Studio для разработки интерфейса используется редактор компоновки Layout Editor, который задействует XML файлы. Для реализации дизайна интерфейса страниц и всплывающих окон, используются стили, которые хранятся в папке с ресурсами – layout.

Рассмотрим назначение всех стилей:

- activity_add_transaction.xml – содержит отображение добавления новой транзакции, после нажатия кнопки;
- activity_category_transaction.xml – содержит отображение страницы со списком всех транзакций по категории;
- activity_main.xml – главный экран, который показывается при запуске приложения;
- activity_plan_info.xml – содержит отображение добавления нового плана, после нажатия кнопки;
- fragment_adding_income.xml – всплывающее окно, с помощью которого происходит добавление доходов;
- fragment_budget_analysis.xml – отображения графика по анализу общего бюджета исходя из расходов;
- fragment_budget_category_info.xml – всплывающее окно, с помощью которого происходит добавление новых категорий;
- fragment_budget_manager.xml – отображение пункта меню «Бюджет категорий» в котором выводится список категорий и их бюджет;

– fragment_budget_planner.xml – отображение пункта меню «Планы/цели» в котором выводится список всех целей;

– fragment_home.xml – отображение пункта меню «Месячные затраты» в котором выводится список всех затрат по категориям. Также является начальной страницей при запуске приложения;

– fragment_user_wallet.xml – всплывающее окно показывающая текущий баланс пользователя.

1.3.1.2 Menu

XML файлы хранящиеся в папке menu, реализуют отображения основного меню, которое отображается после свайпа в право, а также стандартного меню находящегося в правом верхнем углу

Файл activity_main_drawer.xml – отображения главного меню с выбором основных пунктов приложения.

Файл main.xml – отображает пункты меню в правом верхнем углу, такие как «Добавить доход» и «Баланс».

1.3.1.3 Values

В папке Values хранятся ресурсы, предназначенные для различных типов данных.

Colors.xml – файл для хранения всех цветов приложения.

Strings.xml – файл для хранения текста, используемого в приложении.

Styles.xml – файл для хранения стилей приложения.

1.3.2 Java-классы

В этом разделе будут рассмотрены Java-классы приложения, их функции и назначения.

Рассмотрим Java-классы, использующие Activity:

– MainActivity – главная Activity в которую пользователь попадает сразу после запуска приложения. В ней происходит загрузка базы данных, установления toolbar и реализация дополнительного меню для добавления и просмотра баланса;

- PlanInfoActivity – реализует Activity для добавления и редактирования цели;
- AddingTransactionActivity – Activity в которую попадает пользователь при добавления транзакции;
- ListingCategoryTransactionActivity – этот класс реализует страницу на которой информация о всех транзакция по конкретной категории.

Рассмотрим Java-классы, использующие Fragment

- BudgetCategoryInfoFragment – всплывающие окно, которое реализует добавлени и изменение категории;
- BudgetAnalysisFragment – класс, реализующий фрагмент, в котором на графике происходи анализ расходов по категориям;
- BudgetPlannerFragment – фрагмент в котором отображается страница планов и целей;
- BudgetManagerFragment – данный класс реализует фрагмент, в котором происходит отображения всех категорий созданных пользователем.
- HomeFragment – начальный фрагмент, в которой пользователь попадает после запуска приложения, а также в котором происходит отображение текущих затрат за месяц по категориям;
- AddingIncomeFragment – всплывающие окно, с помощью которого происходит добавление доходов пользователя;
- UserWallettFragment – всплывающие окно, в котором указывается текущий баланс пользователя;

Рассмотрим Java-классы, использующие Adapter:

- BudgetCategoryListAdapter – адаптер, который управляет представлением данных о списке категорий;
- PlansListAdapter – адаптер реализующий список планов;
- BudgetManagerListAdapter – адаптер, который реализует представление текущих категорий и их затрат на главной странице;

– TransactionListAdapter – адаптер для оформления списка транзакций по категории.

Рассмотрим Java-классы для реализации конструкторов и методов

– BudgetCategory – конструктор, который создает методы для работы с категориями;

– BudgetCategoryManager – класс, в котором реализованы функции по управлению категориями;

– Plan – конструктор, который создает методы для работы с планами;

– PlanManager – класс, необходимый для добавления, удаления и редактирования планов;

– Transaction – конструктор, который создает методы для работы с транзакциями;

– TransactionManager – класс, в котором реализованы методы с помощью которых происходит, добавление и удаление транзакции;

– UserWallet – класс в котором реализованы функции по управлению личным балансом пользователя, такие как добавление, удаление и возврат средств.

В конце рассмотрим Java-классы для создания и контроля БД

– DatabaseController – класс, который проверяет наличие базы данных и если она отсутствует, то создает новую.

– DatabaseManager – основной класс для работы с БД, который реализует инициализацию таблиц, а также методы по добавлению и удалению новых строк.

– BudgetCategoriesTable – класс, который создает таблицу для категорий.

– TransactionsTable – класс, который создает таблицу для транзакций.

– PlansTable – класс, который создает таблицу для планов.

[illegible]

1.3.3 Структура приложения

```

graph LR
    subgraph Клиент
        A[Приложение]
    end
    subgraph База_данных
        B[SQLite]
    end
    A <--> B
  
```

Принцип работы приложения заключается во взаимодействии клиентской часть с базой данных SQLite. Различные действие пользователя с приложением, такие как добавление, удаление и изменение записей формируют запрос, который отправляется на базу данных. После чего идет обработка запроса самой БД и в случаи успешного выполнения операции происходит возврат новых данных и отображение изменений в интерфейсе пользователя.

Диаграмма последовательности действий представлена на рисунке 9

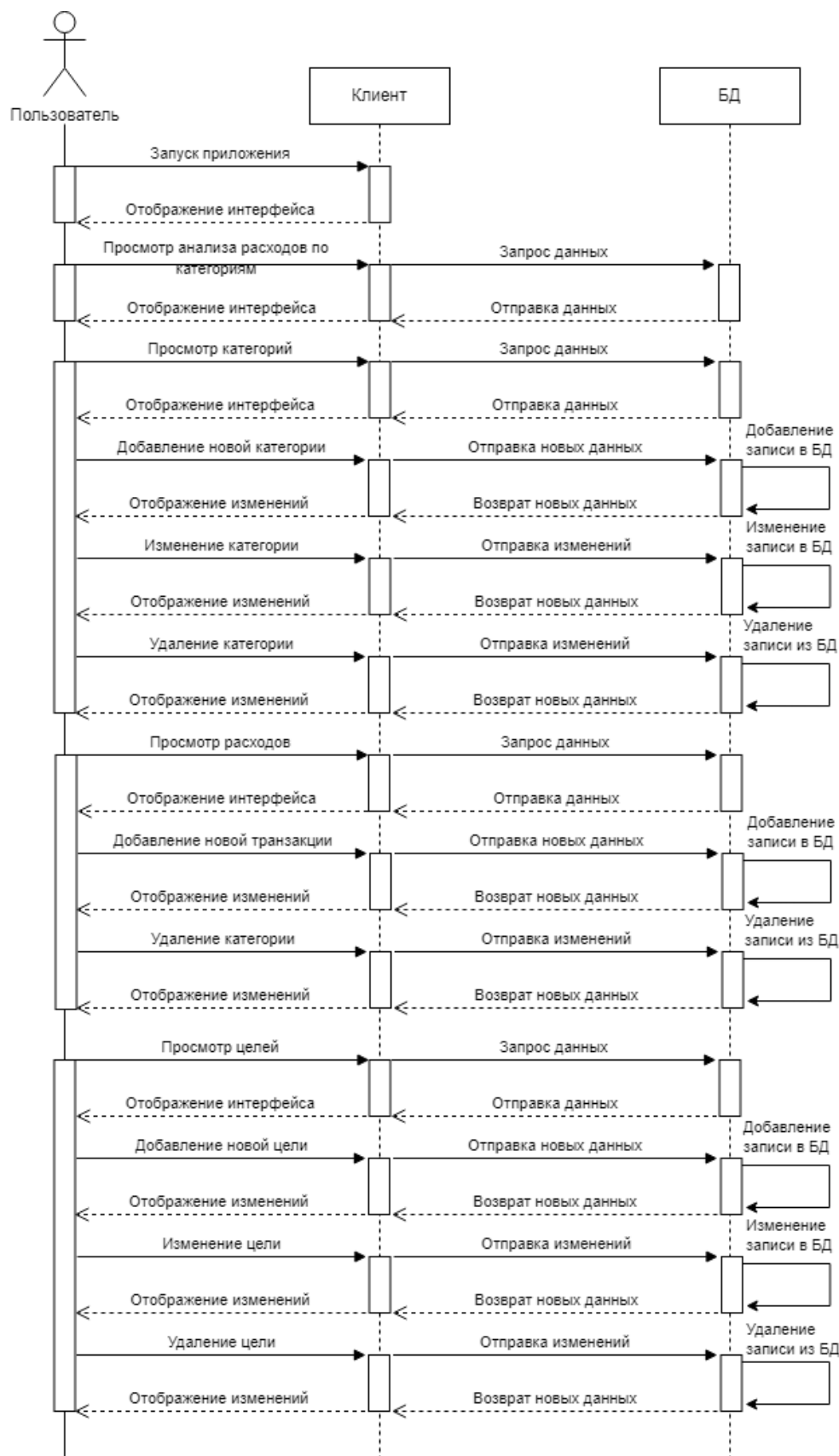


Рисунок 9 – UML-диаграмма последовательности

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.12.001 ПЗ

Лист

31

1.3.4 Структура базы данных

Для реализации данного приложения необходимо продумать и проанализировать какие связи и сущности должны быть в базе данных.

В результате разработки были созданы следующие таблицы:

- таблица BudgetCategoryTable, представленная в таблице 1. Данная таблица включает в себя информацию о категориях;
- таблица TransactionTable, представленная в таблице 2. В этой таблице содержится информация о всех транзакциях ;
- таблица PlanTable, представленная в таблице 3. В этой таблице содержится информация о планах и целях.

Таблица 1 – Атрибуты сущности BudgetCategoryTable

Поле	Тип данных	Описание
ID	Int	Идентификатор номера категории
name	Varchar	Имя категории
maxBudget	Double	Максимальный бюджет категории
currentBudget	Double	Текущий бюджет категории

Таблица 2 – Атрибуты сущности TransactionTable

Поле	Тип данных	Описание
ID	Int	Идентификатор номера транзакции
productName	Varchar	Имя транзакции
price	Double	Цена транзакции
buyDate	Varchar	Дата транзакции
BudgetID	Int	Идентификатор номера категории

Таблица 3 – Атрибуты сущности PlanTable

Поле	Тип данных	Описание
ID	Int	Идентификатор номера плана
title	Varchar	Название плана
description	Varchar	Описание плана
neededMoney	Double	Необходимое количество денег
deadline	Varchar	Дедлайн плана

Схема связей таблиц базы данных представлена на рисунке 10.

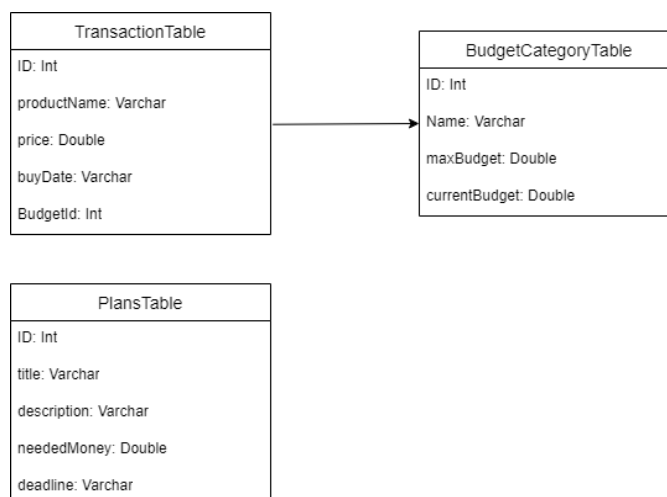


Рисунок 10 – Схема БД

1.4 Тестирование

1.4.1 Добавление и изменение баланса и доходов пользователя

Чтобы добавить начальный баланс и впоследствии прибавлять новый доход, пользователю необходимо нажать на меню в правом верхнем углу и выбрать «Добавить доход» (рисунок 11).

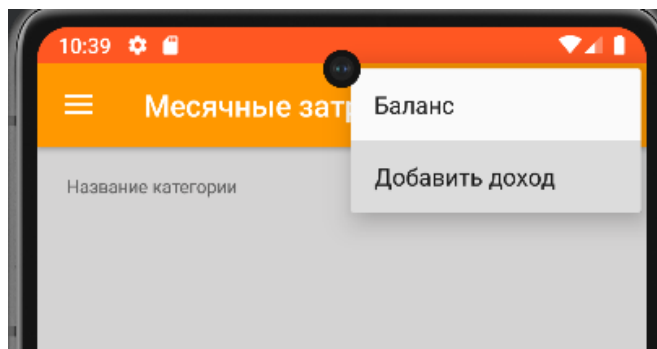


Рисунок 11 – Добавление дохода

После ввести необходимую сумму и нажать кнопку «Подтвердить» (рисунок 12).

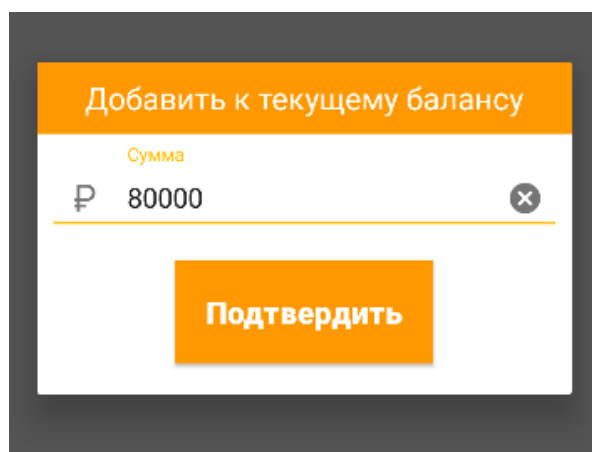


Рисунок 12 – Ввод начального баланса

Баланс изменился на необходимую сумму (рисунок 13). В будущем с помощью этой функции можно будет добавлять уже к имеющемуся балансу новый доход.

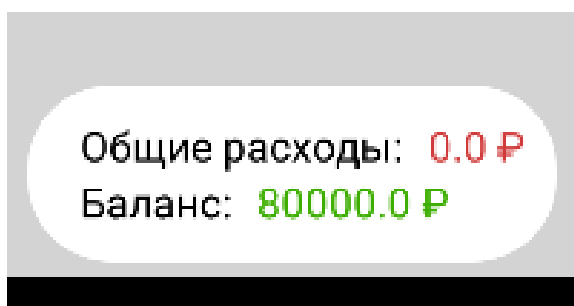


Рисунок 13 – Текущий баланс

Чтобы сбросить имеющийся баланс, необходимо нажать на поле с информацией о балансе и общих расходов. После чего появится диалоговое окно с предложением сбросить текущий баланс (рисунок 11).

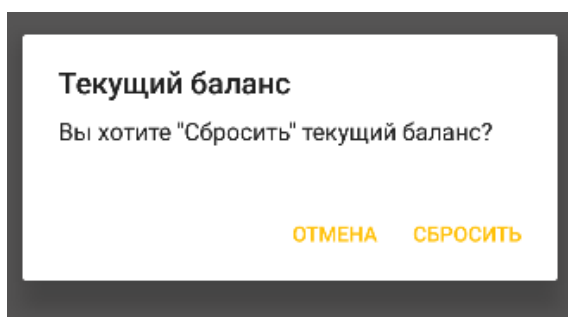


Рисунок 14 – Сброс баланса

1.4.2 Добавление категорий и бюджета

Перед тем как добавить транзакцию необходимо создать категорию, к которой она будет относиться. Для этого пользователь должен открыть главное меню и выбрать «Бюджет категорий» (рисунок 15).

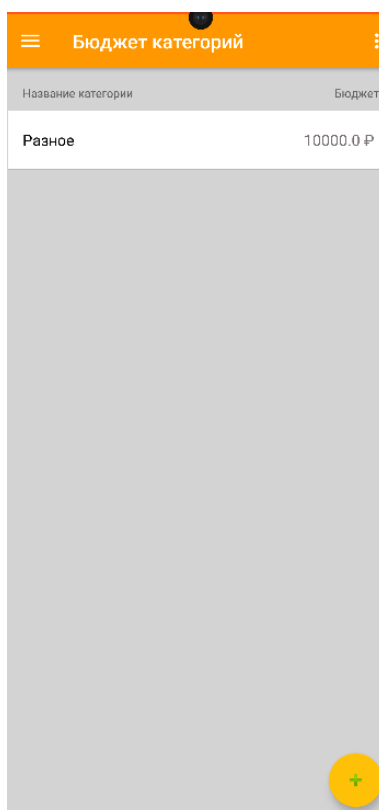


Рисунок 15 – Бюджет категорий

Далее нажать на кнопку в правом нижнем углу, после чего появится всплывающее окно, в котором необходимо заполнить поля «Имя категории», а также «Бюджет» (рисунок 16).

Рисунок 16 – Добавление новой категории

После нажатия кнопки «Подтвердить» категория успешно добавиться на страницу к остальным категориям (рисунок 17).

Бюджет категорий	
Название категории	Бюджет
Разное	10000.0 ₽
Еда	15000.0 ₽

Рисунок 17 – Отображение добавленной категории на экране

В базе данных также появилась запись о данной категории (рисунок 18).

Таблица: budget_categories ↕ ↺ 📶 🔄

	colID	name	maxBudget	currentBudget
	Фи...	Фил...	Фильтр	Фильтр
1	1	Разное	10000.0	0.0
2	2	Еда	15000.0	150.0

Рисунок 18 – База данных таблицы «Категорий»

На главной странице категории отображаются следующим образом. У каждой категории есть своя полоса заполнения, которая будет увеличиваться, при добавление новых транзакций (рисунок 19).

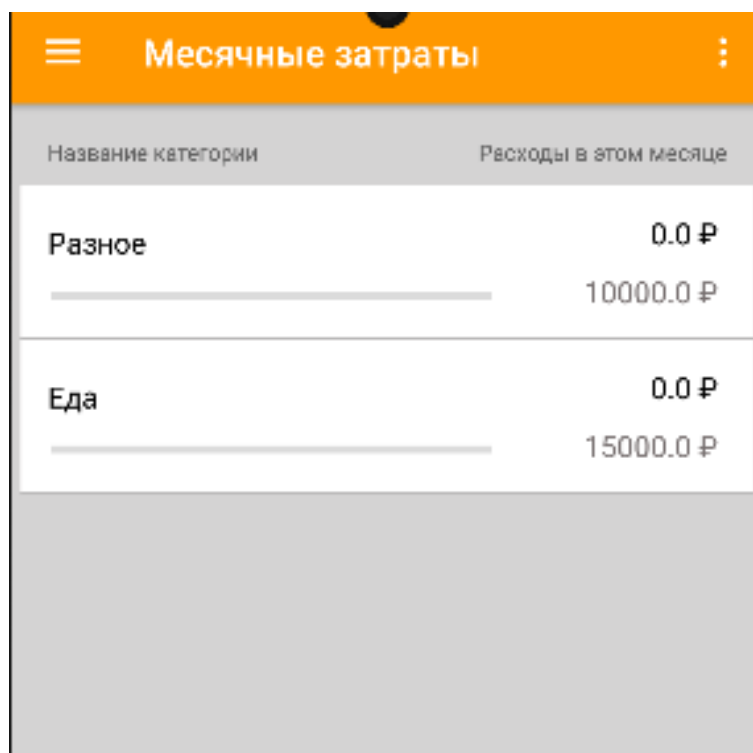


Рисунок 19 – Отображение категорий на главной странице

Чтобы изменить или удалить категорию необходимо нажать на неё и после появится диалоговое окно с выбором действий «Удалить» или «Изменить» (рисунок 20).

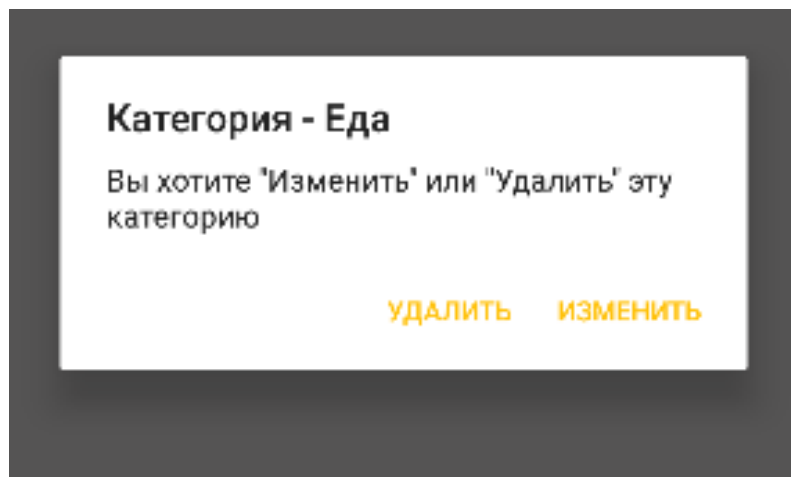


Рисунок 20 – Диалоговое окно при нажатии на категорию

1.4.3 Управление транзакциями

После того, как категории были созданы можно добавить новую транзакцию. Для этого необходимо нажать на кнопку в нижнем правом углу (рисунок 21).

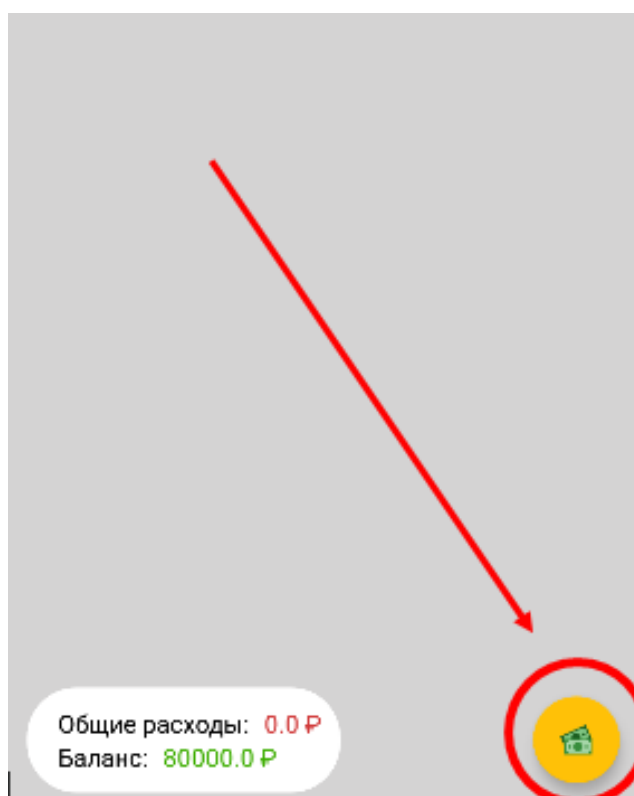


Рисунок 21 – Кнопка добавления расходов

По нажатию откроется страница, в которой нужно ввести все необходимые данные о транзакции: категорию, название, цену и дату. Далее нажать кнопку «Добавить транзакцию» (рисунок 22).

Добавить новую транзакцию

Категория товаров:

Еда

Название товара

Картофель

9/50

Цена товара

₽ 150

×

Дата транзакции:

2024

Thu, Jun 13

<

June 2024

>

S

M

T

W

T

F

S

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

Рисунок 22 – Окно добавления расходов

После добавление транзакции, на главной странице изменилось количество расходов у категории «Еда», а шкала прогресса немного увеличилась. Также с баланса была списана сумма в 150 рублей, и эта же сумма указана в общих расходах (рисунок 23).

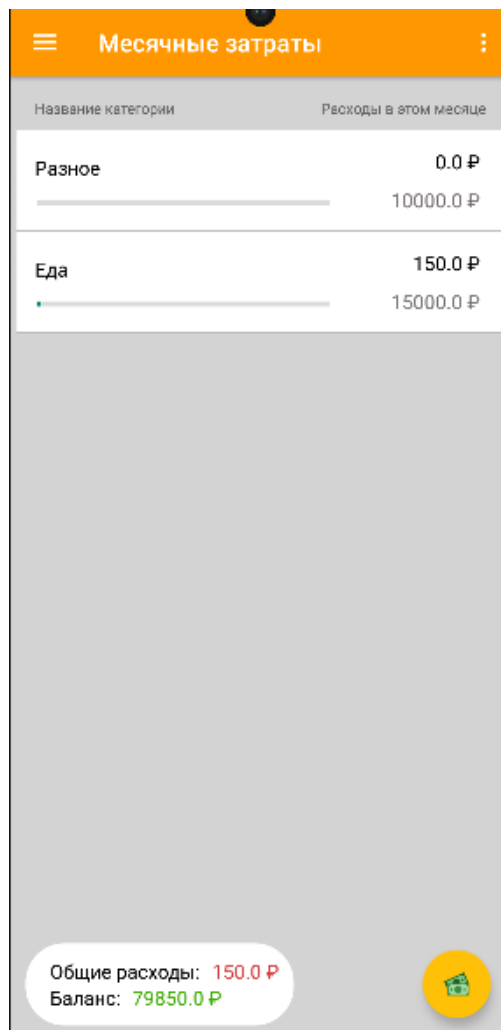


Рисунок 23 – Отображение добавленной транзакции

В базе данных транзакций также появилась новая запись (рисунок 24).

Таблица: transactions

colID	productName	price	buyDate	budgetID
Фи...	Фильтр	Фи...	Фильтр	Фильтр
1	1 Картофель	150.0	Thu Jun 13 2024	2

Рисунок 24 – База данных таблицы «Транзакции»

Чтобы подробнее узнать о всех расходах у каждой категории необходимо нажать на интересующую категорию, после чего откроется страница со списком по этой категории. Здесь будет указана дата, имя и цена каждой транзакции (рисунок 25).

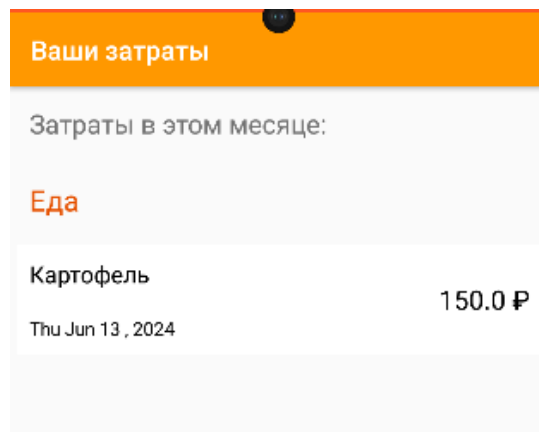


Рисунок 25 – Список всех расходов по категории

Если пользователь ошибся или решил удалить запись, то по нажатию на неё выйдет диалоговое окно, в котором можно выбрать удаление транзакции (рисунок 26).

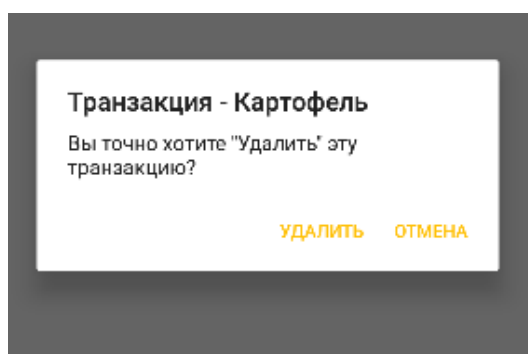


Рисунок 26 – Удаление транзакции

После чего в БД транзакций также исчезла запись (рисунок 27).

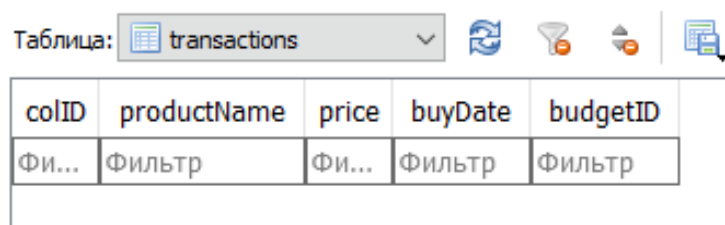


Рисунок 27 – Изменения базы данных таблицы «Транзакции»

При удалении транзакции, также изменяется баланс и общие расходы (рисунок 28).

Месячные затраты	
Название категории	Расходы в этом месяце
Разное	0.0 ₺
	10000.0 ₺
Еда	0.0 ₺
	15000.0 ₺

Рисунок 28 – Изменения после удаления транзакции

1.4.4 Анализ бюджета

В данном приложении есть возможность отследить, в процентном соотношении, затраты по всем категориям. Для этого в главном меню необходимо выбрать пункт «Анализ бюджета» (рисунок 29).

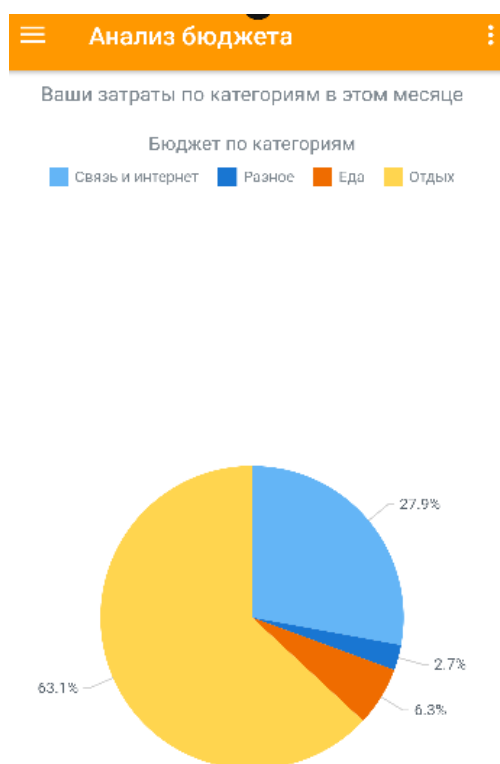


Рисунок 29 – Анализ бюджета

1.4.5 Управление планами

Пользователь также может создавать финансовые цели. Чтобы добавить цель, необходимо нажать на кнопку в нижнем правом углу. Откроется окно, в котором необходимо ввести следующие данные: название цели, её описание, бюджет и крайний срок для её достижения. (рисунок 30).

Добавить новую цель

Введите вашу цель

Машина

Описание цели

купить до конца года

Бюджет

₽ 250000.0

Дедлайн цели:

124
Fri, Dec 2

January 1900

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Сохранить

Рисунок 30 – Добавления новой цели

Бюджет цели подразумевает наличия на вашем балансе необходимой суммы для покупки какой-либо вещи. Это значит, если на вашем балансе имеется достаточная сумма, то эта цель будет подсвечена зеленым цветом (рисунок 31).

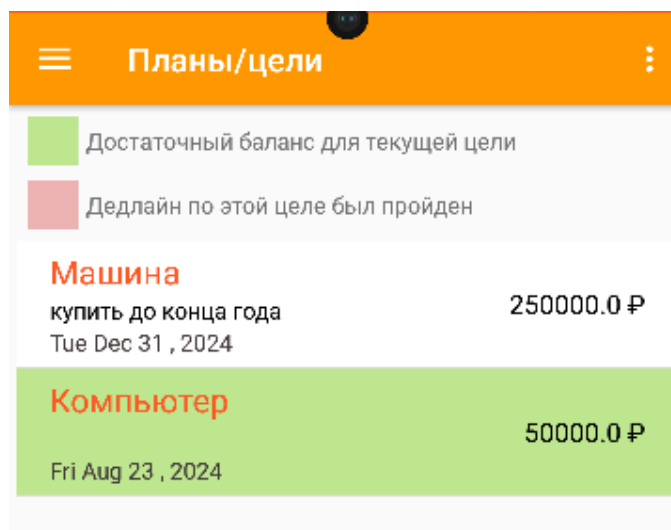


Рисунок 31 – Отображение достигнутой цели

В случае, если дата, указанная в цели уже прошла, то данная цель отмечается красной (рисунок 32).

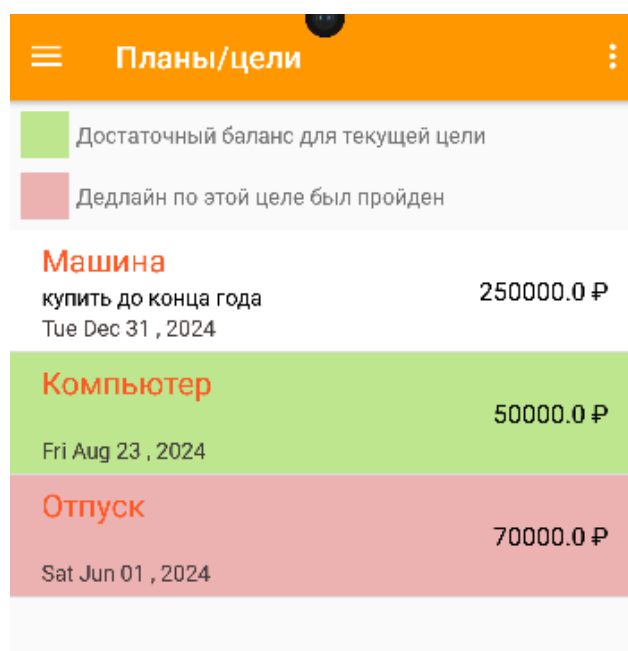


Рисунок 32 – Отображение просроченной цели

2 Безопасность жизнедеятельности

В наше время работа за компьютером стала неотъемлемой частью жизни многих людей, что приводит к появлению различных проблем со здоровьем, таких как напряжение глаз, неправильная осанка, туннельный синдром и других заболеваний. В особенности это касается людей чья профессиональная деятельность связана с долгим нахождением перед экраном монитора. Для того, чтобы обезопасить себя от проблем продолжительной работы за компьютером, необходимо соблюдать ряд правил. В этой главе мы рассмотрим необходимые меры, при соблюдении которых, можно предотвратить все возможные проблемы, связанные с долгой работой за компьютером.

2.1 Требования к организации рабочего места

При организации рабочего места необходимо соблюдать ряд рекомендаций, которые помогут минимизировать риски для здоровья. Рассмотрим список предлагаемых рекомендаций:

- эргономичное кресло. Для комфортной работы за компьютером важно приобрести хорошее и качественное кресло, которое обладает возможностью регулировки высоты и угла наклона, и необходимой формой для спины и комплекции человека;
- правильная высота стола. Рабочий стол должен быть на такой высоте, чтобы руки находились под углом 90 градусов при работе с клавиатурой и мышью;
- положение монитора. Чтобы предотвратить проблемы со зрением, монитор должен находиться на комфортном расстоянии от глаз. Рекомендуемым значением является 50-70 см от монитора. Положение монитора должно быть такое чтобы, верхняя часть экрана была на уровне глаз.

Пример соблюдения всех правил можно увидеть на рисунке 33.

					ПГУ1.090301.12.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		45

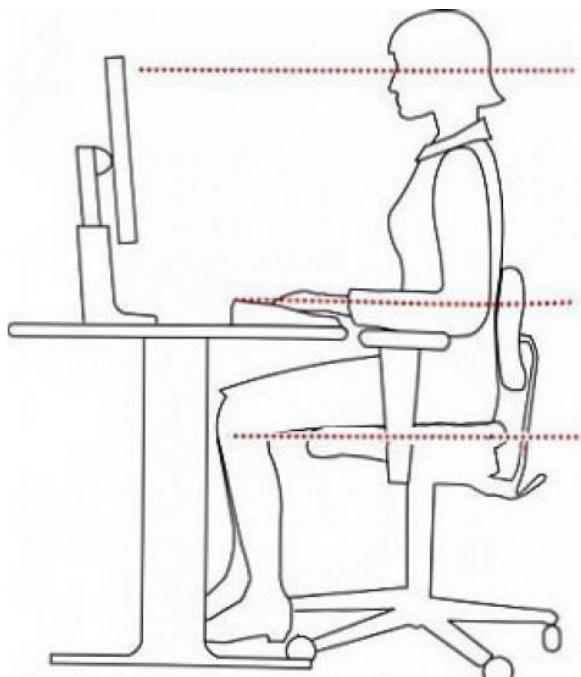


Рисунок 33 – Правильное положение при работе за компьютером.

2.2 Требования к освещению

Одна из главных проблем, которая возникает при долгой работе за компьютером – это вред для глаз. Неправильное освещение помещения и настроек яркости монитора, может привести к усталости глаз, сухости, покраснению, размытости зрения и головным болям. Чтобы предотвратить это, необходимо соблюдать некоторые требования к освещению в помещении:

- естественное освещение. Рекомендуется, по возможности, использовать естественный свет, при этом избегать попадания прямых солнечных лучей на монитор. Естественный свет снижает напряжение глаз и улучшает общее самочувствие;

- искусственное освещение. Для работы в вечернее время рекомендуется использовать искусственное освещение, такие как настольная лампа, комнатное освещение и т.д. Предпочтительно комбинировать освещения для более лучшего эффекта. Освещение должно быть ярким, чтобы избежать напряжения глаз, но не слишком интенсивным,

чтобы исключить появления бликом на экране монитора. Также нужно учитывать правильное расположение источников света, Локальный источник, например настольная лампа, должен быть установлен сбоку от монитора, чтобы не создавать прямых бликов на экране;

– настройка монитора. Правильная настройка яркости и контрастности монитора, также может уменьшить напряжение глаз при длительной работе за ПК. Рекомендуется в вечернее время использовать ночной режим, который делает цвета экрана более теплыми. Это поможет уменьшить нагрузка на глаза, а также улучшить качество сна.

2.3 Требования к режиму отдыха

Чтобы избежать проблем со здоровьем при длительной работе за компьютером, необходимо ежечасно делать небольшой перерыв, для выполнения разминки, чтобы избежать мышечного напряжения и улучшить кровообращение в суставах и мышцах.

Не стоит забывать и про глаза, которые так же подвержены воздействию, поэтому будет полезно сделать зарядку, которая включает в себя круговые движения открытыми глазами, смена точки фокусировки взгляда с близкой до дальней.

Важно также соблюдать строгий график работы и отдыха, ведь постоянный труд может привести к негативным последствиям. Необходимо составить четкое расписание и расписать план работы, для оптимизации своего времени. Не следует работать в дни отдыха, переработки могут плохо сказаться на вашем физическом и ментальном здоровье.

При соблюдении данных рекомендаций вы сможете обеспечить себе комфортную работу с минимальным риска для своего здоровья.

Заключение

В ходе выполнения данной выпускной квалификационной работы, было разработано Android-приложение по контролю личных расходов. Были получены навыки разработки Android-приложений в среде разработки Android Studio, с использованием языка программирования Java и базой данных SQLite.

Результат работы – это приложение, которое соблюдает современные стандарты разработки, позволяет пользователю удалять и добавлять расходы, создавать и изменять категории затрат, предлагает общий графический анализ всех расходов по категориям, а также устанавливать и отслеживать плановые расходы.

В дальнейшем возможно улучшение приложения, которое может включать в себя добавления нового функционала по работе с расходами и категориями, интеграцию с банковскими сервисами, а также улучшенным дизайном и изменением интерфейса для более комфортного использования приложения.

Список использованных источников

1. Разработка консольных приложений с помощью Microsoft Visual Studio 2017: учебное пособие / Ю.В. Тракимус – Новосибирск: Изд-во НГТУ, 2018. – 3 с.
2. Учебное пособие по языку Java Часть 1: учебное пособие / А. В. Гаврилов, О.А. Дегтерева, И.А. Лезин, ИВ. Лезина – Самара, 2010. – 7-9 с.
3. Цифровые технологии управления получением, хранением, передачей и обработкой больших данных: SQLite : учеб.метод. пособие / С. В. Рындина. – Пенза : Изд-во ПГУ, 2023. – 7 с.
4. Введение в архитектурные шаблоны проектирования: MVC, MVP, MVVM. [Электронный ресурс] / Режим доступа: <https://gochadev.ru/2024/02/13/vvedenie-v-arhitekturnie-shablony-proektirovaniya-mvc-mvp-mvvm/> , свободный. (дата обращения 02.05.2024).

Приложение А

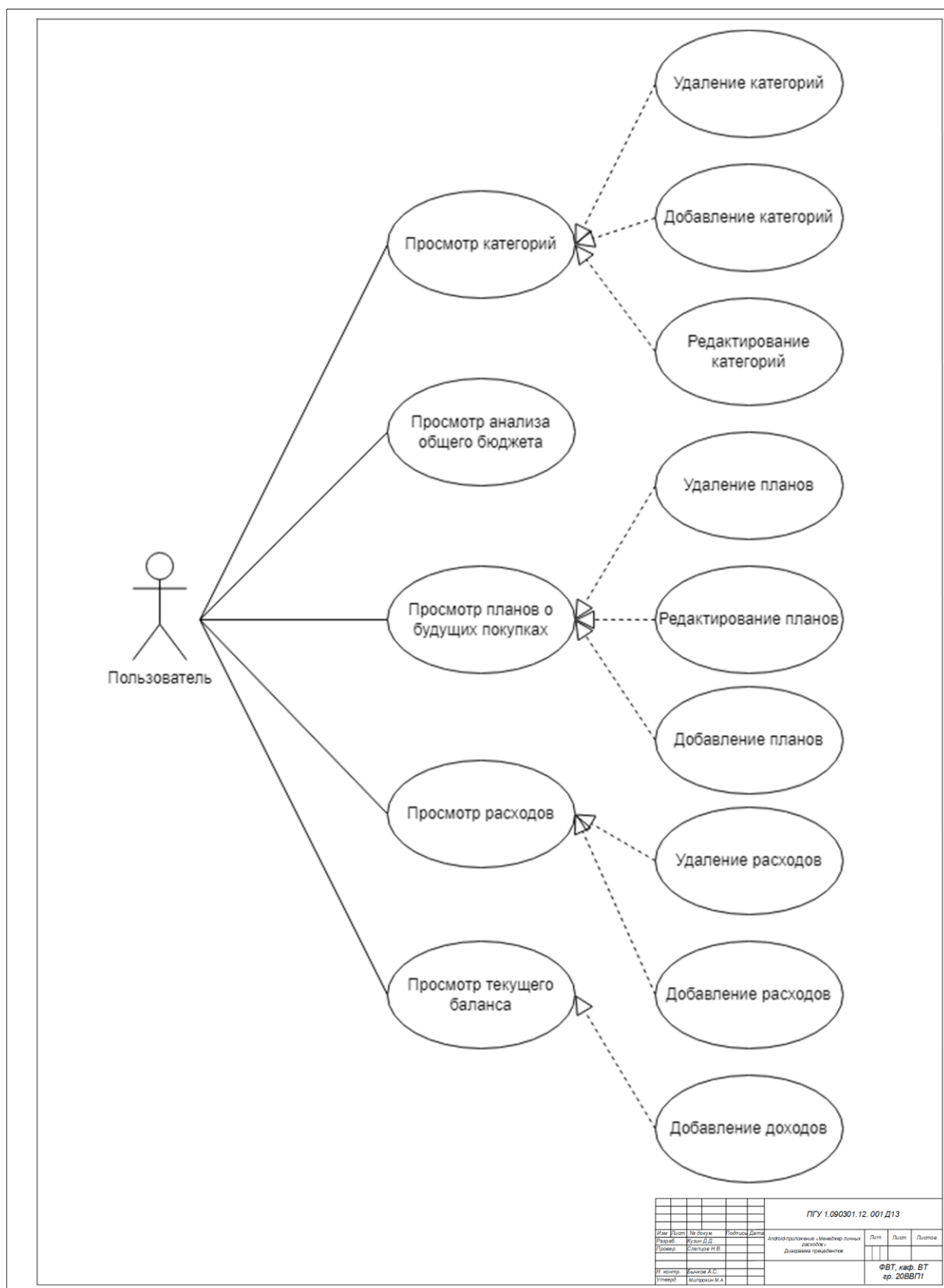
(Обязательное)

Android-приложение «Менеджер личных расходов».

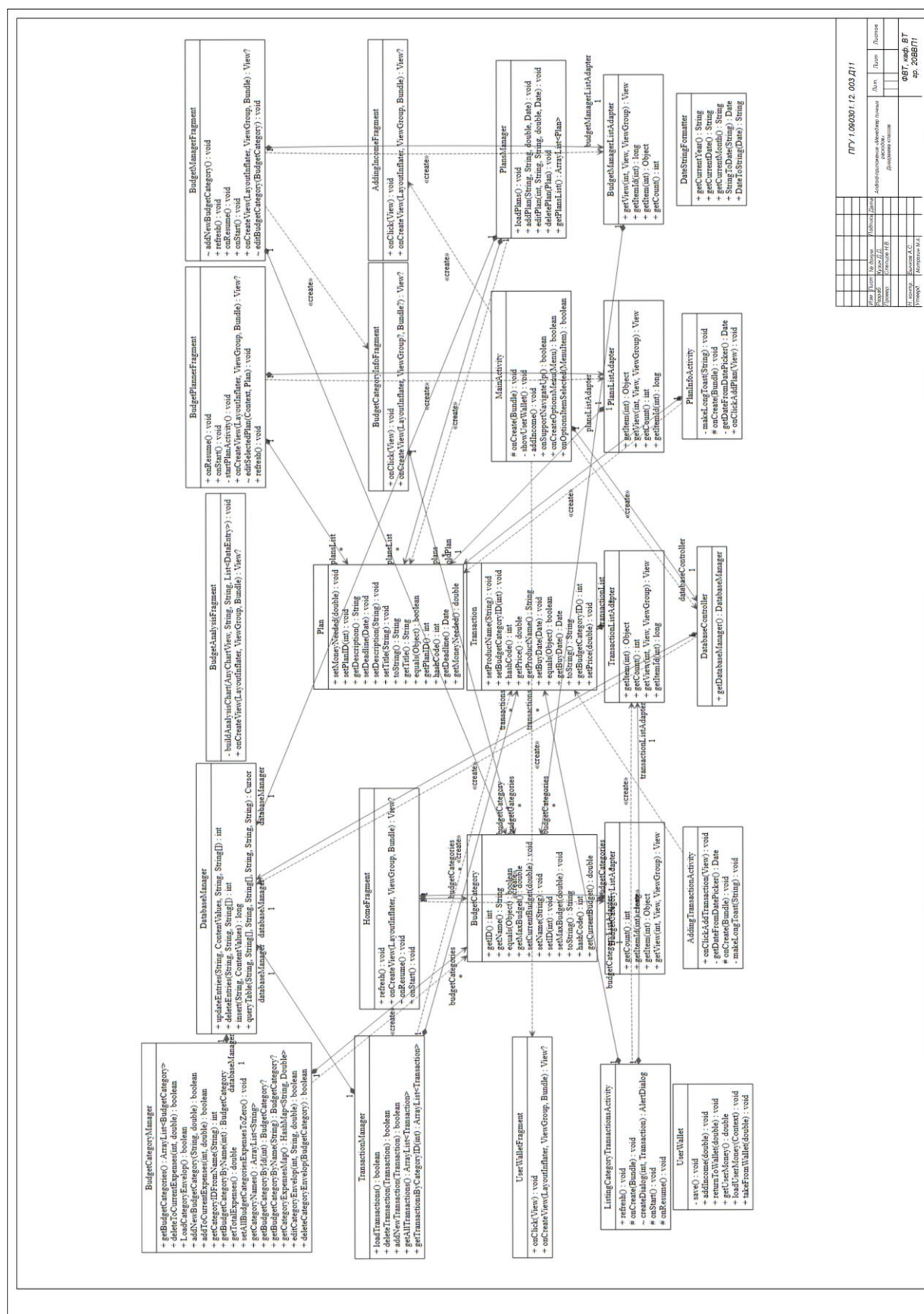
UML диаграммы

					ПГУ1.090301.12.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

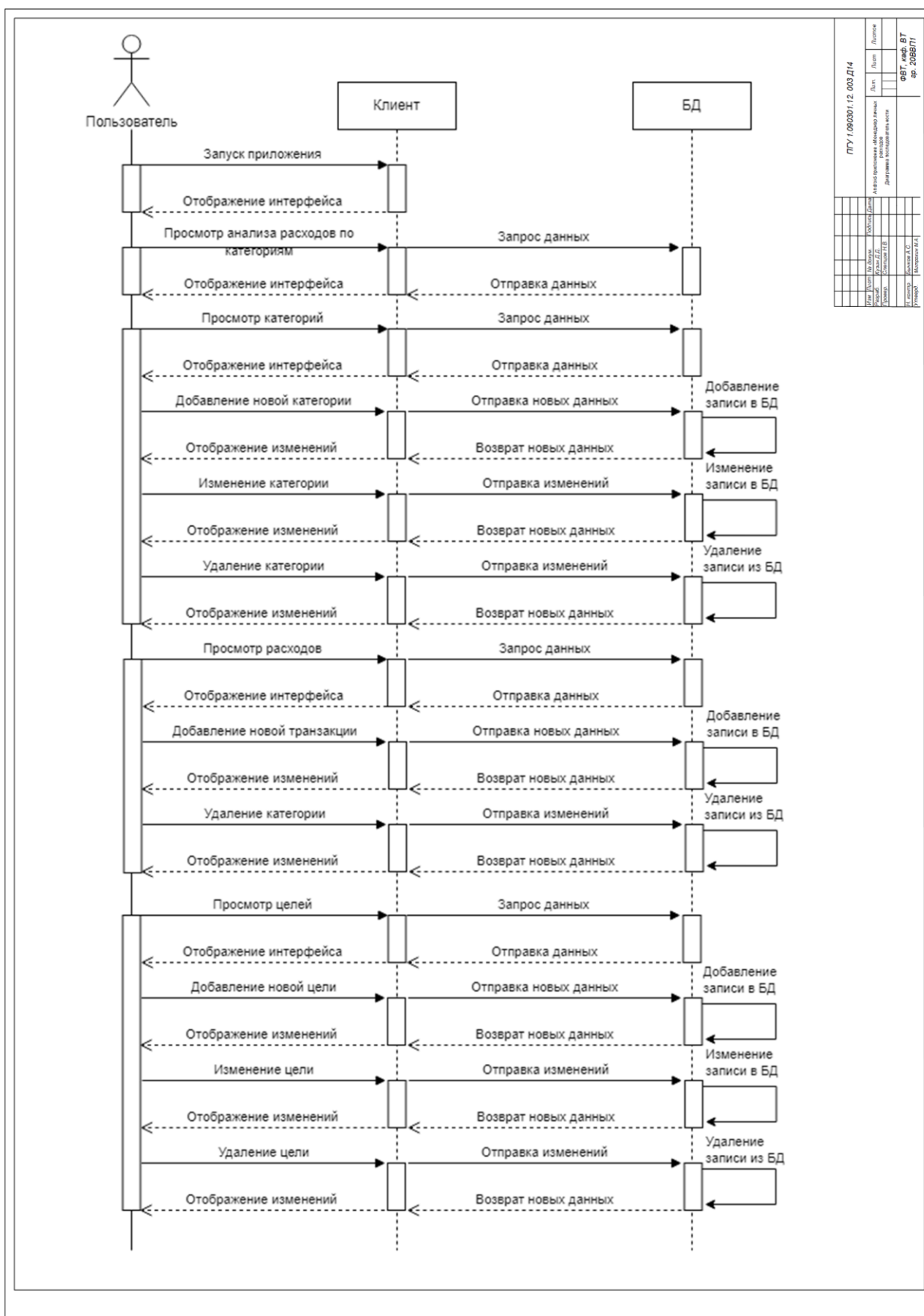
Приложение А.1 – Диаграмма прецедентов



Приложение А.2 – Диаграмма классов



Приложение А.3 – Диаграмма последовательности



ПУ 1.090301.12.003 Д14			
Имя	Фамилия	Имя	Фамилия
№ докум.	Дата	№ докум.	Дата
Подпись	Подпись	Подпись	Подпись
Диаграмма последовательности		ФЕЛ, ред. ВТ фр. 2006/11	

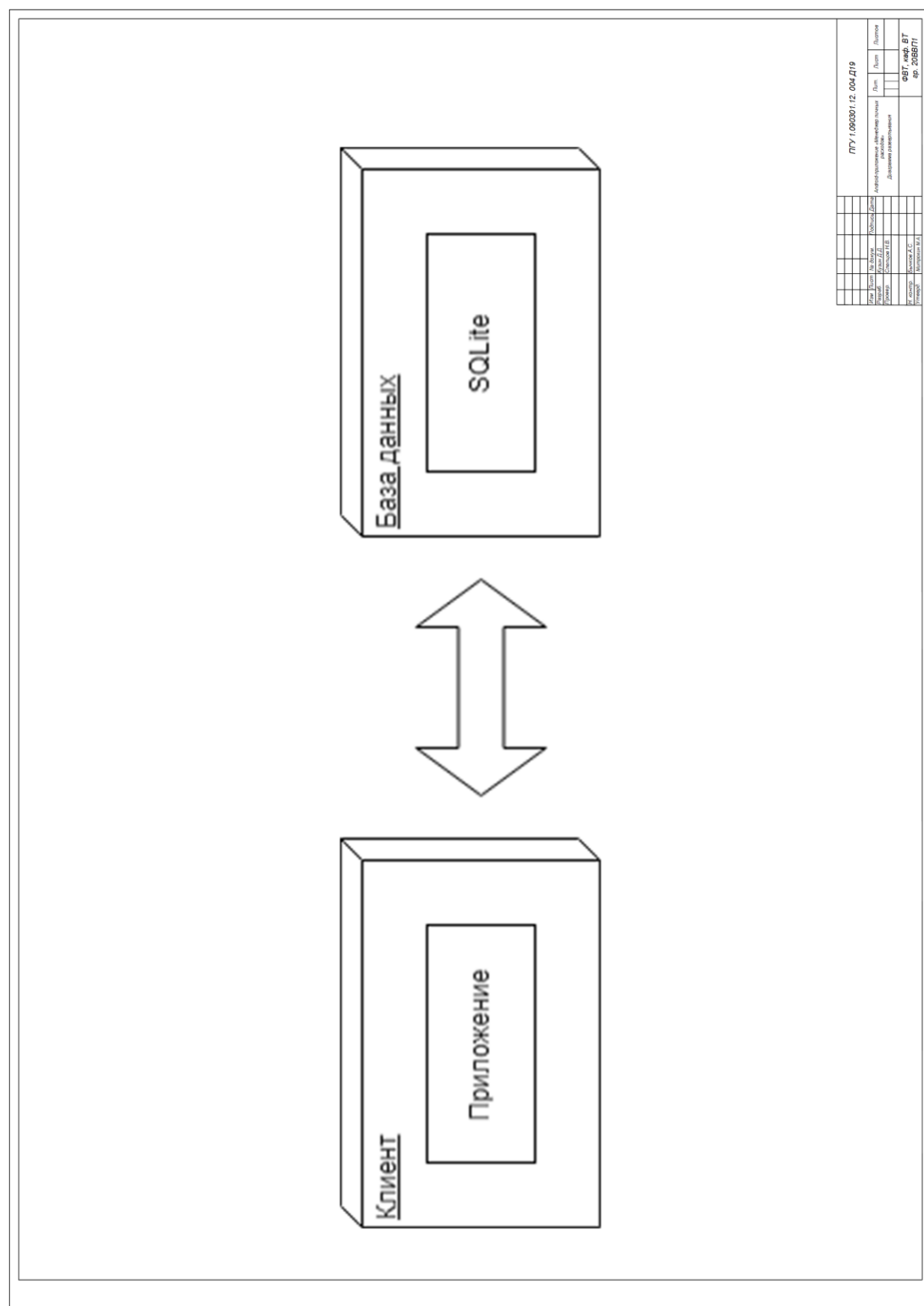
Изм.	Лист	№ докум.	Подпись	Дата

ПУ1.090301.12.001 ПЗ

Лист

53

Приложение А.4 – Диаграмма развертывания



Приложение Б

(Обязательное)

Android-приложение «Менеджер личных расходов».

Листинг программы

					ПГУ1.090301.12.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

Файл MainActivity.java

```
package com.example.expensetracker.mainactivity;

import android.os.Bundle;
import android.view.MenuItem;
import android.view.Menu;

import com.example.expensetracker.R;
import com.example.expensetracker.budget_categories.BudgetCategoryManager;
import com.example.expensetracker.budgetplanner.PlansManager;
import com.example.expensetracker.dbmanagment.DatabaseController;
import com.example.expensetracker.transactions.TransactionManager;
import com.example.expensetracker.transactions.activities.AddingTransactionActivity;
import com.example.expensetracker.usermoney.UserWallet;
import com.example.expensetracker.usermoney.ui.AddingIncomeFragment;
import com.example.expensetracker.usermoney.ui.UserWalletFragment;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.navigation.NavigationView;

import androidx.annotation.NonNull;
import androidx.fragment.app.FragmentManager;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    DatabaseController databaseController;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        try {
```

					ПГУ1.090301.12.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		56

```

        databaseController = new DatabaseController(this);
        BudgetCategoryManager.LoadCategoryEnvelop();
        TransactionManager.loadTransactions();
        PlansManager.loadPlans();
        UserWallet.loadUserMoney(this);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        DrawerLayout drawer =
        findViewById(R.id.drawer_layout);
        NavigationView navigationView =
        findViewById(R.id.nav_view);

        mAppBarConfiguration = new
        AppBarConfiguration.Builder(
            R.id.nav_home, R.id.nav_budgetManager ,
            R.id.nav_budget_analysis ,R.id.nav_budgetPlanner)
            .setDrawerLayout(drawer)
            .build();

        NavController navController =
        Navigation.findNavController(this, R.id.nav_host_fragment);
        NavigationUI.setupActionBarWithNavController(this,
        navController, mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView,
        navController);

    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item)
{
    switch (item.getItemId())
    {
        case R.id.add_income:
            addIncome();
            return true;
        case R.id.user_wallet:
            showUserWallet();
            return true;
    }
}

```

```

    }
    return super.onOptionsItemSelected(item);
}

@Override
public boolean onSupportNavigateUp() {
    NavController navController =
Navigation.findNavController(this, R.id.nav_host_fragment);
    return NavigationUI.navigateUp(navController,
mAppBarConfiguration)
        || super.onSupportNavigateUp();
}

private void addIncome() {
    FragmentManager fragmentManager =
getSupportFragmentManager();

    AddingIncomeFragment addingIncomeFragment = new
AddingIncomeFragment();

    addingIncomeFragment.show(fragmentManager, "add income");
}

private void showUserWallet() {
    FragmentManager fragmentManager =
getSupportFragmentManager();

    UserWalletFragment userWalletFragment = new
UserWalletFragment();

    userWalletFragment.show(fragmentManager, "add income");
}
}

```

Файл PlanInfoActivity.java

```

package com.example.expensetracker.budgetplanner.activities;

import androidx.appcompat.app.AppCompatActivity;

import android.content.res.Configuration;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.Toast;

import com.example.expensetracker.R;

```

```

import com.example.expensetracker.budgetplanner.Plan;
import com.example.expensetracker.budgetplanner.PlansManager;
import
com.example.expensetracker.budgetplanner.ui.BudgetPlannerFragme
nt;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.Locale;

public class PlanInfoActivity extends AppCompatActivity {

    EditText planTitleEditText;
    EditText planDescEditText;
    EditText planBudgetEditText;
    DatePicker planDeadLineDatePicker;

    boolean editPlan;
    Plan oldPlan;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_plan_info);
        try {

            planTitleEditText =
findViewById(R.id.plan_title_tf);
            planDescEditText = findViewById(R.id.plan_desc_tf);
            planBudgetEditText =
findViewById(R.id.plan_budget_tf);
            planDeadLineDatePicker =
findViewById(R.id.plan_date_picker);

            Bundle bundle = getIntent().getExtras();
            editPlan = bundle.getBoolean("editPlan", false);

            if (editPlan) {

                Button submitBu = findViewById(R.id.submitBu);
                submitBu.setText(R.string.edit_plan);

                int planID = bundle.getInt("planID" , -1);

                String title = bundle.getString("planTitle",
"none");
                String desc = bundle.getString("planDesc",
"none");
                double budget =

```

```

Double.parseDouble(bundle.getString("planBudget", "0"));
        String dateStr = bundle.getString("planDate",
"none");

        planTitleEditText.setText(title);
        planDescEditText.setText(desc);

planBudgetEditText.setText(String.valueOf(budget));

        Date date = null;
        try {
            if (!dateStr.equalsIgnoreCase("none")) {

                Log.i("SimpleDateFormat", "onCreate: " +
dateStr);

                SimpleDateFormat formatter = new
SimpleDateFormat("E MMM dd yyyy");
                date = formatter.parse(dateStr);

                Log.i("formatted date", "onCreate:"
+date.toString());

planDeadlineDatePicker.updateDate(date.getYear(),
date.getMonth(), date.getDay());
            }
        } catch (ParseException e) {
            date = Calendar.getInstance().getTime();
            e.printStackTrace();
        }
        oldPlan = new Plan(planID , title, desc, budget,
date);
    }
} catch (Exception e)
{
    e.printStackTrace();
}

}

private Date getDateFromDatePicker() {
    int day = planDeadlineDatePicker.getDayOfMonth();
    int month = planDeadlineDatePicker.getMonth();
    int year = planDeadlineDatePicker.getYear();

    Calendar calendar = Calendar.getInstance();
    calendar.set(year, month, day, 0, 0, 0);

    return calendar.getTime();
}

```

```

public void onClickAddPlan(View view)
{
    String title = planTitleEditText.getText().toString();
    String desc = planDescEditText.getText().toString();
    double budget =
Double.parseDouble(planBudgetEditText.getText().toString());
    Date deadline = getDateFromDatePicker();

    if(editPlan)
    {
        PlansManager.editPlan(oldPlan.getPlanID() , title,
desc, budget, deadline);
        BudgetPlannerFragment.refresh();
        makeLongToast("Ваша цель успешно изменена");
    }
    else {
        PlansManager.addPlan(title, desc, budget, deadline);
        BudgetPlannerFragment.refresh();
        makeLongToast("Ваша цель успешно добавлена");
    }
    finish();
}

private void makeLongToast(String msg)
{
    Toast.makeText(getApplicationContext(),msg ,
Toast.LENGTH_LONG).show();
}
}

```

Файл AddingTransactionActivity.java

```

package com.example.expensetracker.transactions.activities;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

import com.example.expensetracker.R;
import
com.example.expensetracker.budget_categories.BudgetCategory;
import
com.example.expensetracker.budget_categories.BudgetCategoryMana
ger;
import com.example.expensetracker.transactions.Transaction;

```

```

import
com.example.expensetracker.transactions.TransactionManager;
import com.example.expensetracker.usermoney.UserWallet;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;

public class AddingTransactionActivity extends AppCompatActivity
{

    Spinner productCategorySpinner;
    EditText productNameTextField;
    EditText productPriceTextField;
    DatePicker transactionDatePicker;

    ArrayList<String> productCategories;
    ArrayAdapter categoriesAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_transaction);

        try {
            productCategorySpinner =
findViewById(R.id.product_category_spinner);
            productNameTextField =
findViewById(R.id.product_name_textField);
            productPriceTextField =
findViewById(R.id.product_price_textField);
            transactionDatePicker =
findViewById(R.id.date_picker);

            productCategories =
BudgetCategoryManager.getCategoryNames();

            categoriesAdapter = new
ArrayAdapter(this, android.R.layout.simple_dropdown_item_1line, pr
oductCategories);

productCategorySpinner.setAdapter(categoriesAdapter);
            productCategorySpinner.setSelection(0);

        } catch (Exception e)
        {
            e.printStackTrace();
        }

        public void onClickAddTransaction(View view)

```

					ПГУ1.090301.12.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		62

```

    {
        try {
            if (UserWallet.getUserMoney() > 0) {

                Date transactionDate = getDateFromDatePicker();
                Date currentDate =
Calendar.getInstance().getTime();

                BudgetCategory budgetCategory =
BudgetCategoryManager.

getBudgetCategoryByName (productCategorySpinner.getSelectedItem()
.toString());

                String productName =
productNameTextField.getText().toString();
                double productPrice =
Double.parseDouble (productPriceTextField.getText().toString());

                Transaction transaction = new
Transaction(budgetCategory.getID(), productName, productPrice,
transactionDate);

TransactionManager.addNewTransaction (transaction);

                UserWallet.takeFromWallet (productPrice);
            }
            else
            {
                Toast.makeText (getApplicationContext() , "Ваш
баланс равен 0, вы не можете купить эту
вещь", Toast.LENGTH_LONG).show();
            }
            finish();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

private Date getDateFromDatePicker() {
    int day = transactionDatePicker.getDayOfMonth();
    int month = transactionDatePicker.getMonth();
    int year = transactionDatePicker.getYear();

    Calendar calendar = Calendar.getInstance();
    calendar.set (year, month, day, 0, 0, 0);

    return calendar.getTime();
}

```

```

    }

    private void makeLongToast(String msg)
    {
        Toast.makeText(getApplicationContext(), msg ,
        Toast.LENGTH_LONG).show();
    }
}

```

Файл ListingCategoryTransactionActivity.java

```

package com.example.expensetracker.transactions.activities;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.example.expensetracker.R;
import com.example.expensetracker.budget_categories.BudgetCategory;
import com.example.expensetracker.budget_categories.BudgetCategoryManager;
import com.example.expensetracker.transactions.Transaction;
import com.example.expensetracker.transactions.TransactionManager;
import com.example.expensetracker.transactions.ui.TransactionListAdapter;

import java.util.ArrayList;

public class ListingCategoryTransactionsActivity extends AppCompatActivity
{
    TextView selectedCategoryName;
    ListView transactionListView;

    String categoryName;
    ArrayList<Transaction> transactions;
    static TransactionListAdapter transactionListAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

```

```

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_category_transaction);
        try {
            selectedCategoryName =
findViewById(R.id.selected_category_name);
            transactionListView =
findViewById(R.id.transaction_lv);

            Bundle bundle = getIntent().getExtras();
            categoryName = bundle.getString("category", "none");
            selectedCategoryName.setText(categoryName);
            int id=
BudgetCategoryManager.getCategoryIDFromName(categoryName);
            transactions =
TransactionManager.getTransactionsByCategoryID(id);
            transactionListAdapter = new
TransactionListAdapter(getApplicationContext(), transactions);

transactionListView.setAdapter(transactionListAdapter);
            transactionListAdapter.notifyDataSetChanged();
            transactionListView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> parent,
View view, int i, long l) {
                    Transaction selectedTransaction =
transactions.get(i);
                    AlertDialog dialog =
createDialog(i,selectedTransaction);
                    dialog.show();

                }
            });

        }catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    AlertDialog createDialog(int i, final Transaction
selectedTransaction){
        AlertDialog.Builder builder = new
AlertDialog.Builder(this);

        builder.setMessage("Вы точно хотите \"Удалить\" эту
транзакцию?");
        builder.setTitle("Транзакция - " +
transactions.get(i).getProductName());
        builder.setPositiveButton("Отмена", new
DialogInterface.OnClickListener() {

```

```

        @Override
        public void onClick(DialogInterface dialog, int
which) {
            dialog.dismiss();
        }
    });
    builder.setNegativeButton("Удалить", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int
which) {

TransactionManager.deleteTransaction(selectedTransaction);
            refresh();
            finish();
            startActivity(getIntent());
        }
    });

    return builder.create();
}
public static void refresh()
{
    if(transactionListAdapter!=null)
    {
        transactionListAdapter.notifyDataSetChanged();
    }
}
@Override
protected void onStart() {
    super.onStart();
    if(transactionListAdapter!=null)
        transactionListAdapter.notifyDataSetChanged();
}

@Override
protected void onResume() {
    super.onResume();
    if(transactionListAdapter!=null)
        transactionListAdapter.notifyDataSetChanged();
}
}

```

Файл BudgetCategoryInfoFragment.java

```

package com.example.expensetracker.budget_categories.ui;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;

```

```

import android.widget.ListView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.DialogFragment;

import com.example.expensetracker.R;
import
com.example.expensetracker.budget_categories.BudgetCategory;
import
com.example.expensetracker.budget_categories.BudgetCategoryManager;
import
com.example.expensetracker.mainactivity.fragments.budgetmanager
.BudgetManagerFragment;
import
com.example.expensetracker.mainactivity.fragments.home.HomeFragment;

public class BudgetCategoryInfoFragment extends DialogFragment
implements View.OnClickListener
{
    View newBudgetCategoryView;
    boolean isNew;
    BudgetCategory budgetCategory;

    EditText budgetCategoryNameEditText;
    EditText budgetCategoryPriceEditText;
    Button addBudgetCategoryButton;

    public BudgetCategoryInfoFragment( BudgetCategory
budgetCategory)
    {
        this.isNew = false;
        this.budgetCategory =budgetCategory;
    }

    public BudgetCategoryInfoFragment()
    {
        this.isNew = true;
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle
savedInstanceState)
    {
        newBudgetCategoryView =
inflater.inflate(R.layout.fragment_budget_category_info,container, false);
    }

```

```

        budgetCategoryNameEditText =
newBudgetCategoryView.findViewById(R.id.category_name);
        budgetCategoryPriceEditText =
newBudgetCategoryView.findViewById(R.id.Category_budget);
        addBudgetCategoryButton =
newBudgetCategoryView.findViewById(R.id.add_Budget_button);

        if (!isNew)
        {

budgetCategoryNameEditText.setText(budgetCategory.getName());

                budgetCategoryNameEditText.setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        Toast.makeText(getContext() ,
R.string.errorMsg ,
                                Toast.LENGTH_LONG).show();
                    }
                });

budgetCategoryPriceEditText.setText(String.valueOf(budgetCategor
y.getMaxBudget()));
        }

        addBudgetCategoryButton.setOnClickListener(this);

        return newBudgetCategoryView;
    }

    @Override
    public void onClick(View view) {
        this.dismiss();
        String budgetCategoryName =
budgetCategoryNameEditText.getText().toString();
        double budget =
Double.parseDouble(budgetCategoryPriceEditText.getText().toStrin
g());

        if (isNew)

BudgetCategoryManager.addNewBudgetCategory(budgetCategoryName ,
budget);
        else

BudgetCategoryManager.editCategoryEnvelop(budgetCategory.getID()
, budgetCategoryName , budget);

        HomeFragment.refresh();

```

```

        BudgetManagerFragment.refresh();
    }
}

```

Файл BudgetAnalysisFragment.java

```

package com.example.expensetracker.budgetanalysis;

import android.content.Context;
import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.anychart.AnyChart;
import com.anychart.AnyChartView;
import com.anychart.chart.common.dataentry.DataEntry;
import com.anychart.chart.common.dataentry.ValueDataEntry;
import com.anychart.chart.common.listener.Event;
import com.anychart.chart.common.listener.ListenersInterface;
import com.anychart.charts.Pie;
import com.anychart.enums.Align;
import com.anychart.enums.LegendLayout;
import com.example.expensetracker.R;
import com.example.expensetracker.budget_categories.BudgetCategoryManager;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BudgetAnalysisFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        View root =
            inflater.inflate(R.layout.fragment_budget_analysis, container,
                false);

        final Context context =
            getActivity().getApplicationContext();
        try {
            AnyChartView anyChartView =
                root.findViewById(R.id.any_chart_view);

```

```

anyChartView.setProgressBar(root.findViewById(R.id.progress_bar)
);

        HashMap<String, Double> dataMap =
BudgetCategoryManager.getCategoryExpensesMap();
        List<DataEntry> data = new ArrayList<>();

        for (Map.Entry element : dataMap.entrySet()) {
            String key = (String) element.getKey();
            Double value = (Double) element.getValue();

            int intValue = value.intValue();

            data.add(new ValueDataEntry(key, intValue));
            Log.i("onCreateView", "onCreateView: " +
intValue);
        }

        buildAnalysisChart(anyChartView, "Ваши затраты по
категориям в этом месяце", "Бюджет по категориям"
            , data);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    return root;
}

private void buildAnalysisChart(AnyChartView anyChartView ,
String pieTitle ,String channelsTitle
    , List<DataEntry> data) {

    Pie pie = AnyChart.pie();

    pie.setOnClickListener(new
ListenersInterface.OnClickListener() {
        @Override
        public void onClick(Event event) {

        }
    });

    pie.data(data);

    pie.title(pieTitle);

    pie.labels().position("outside");

    pie.legend().title().enabled(true);

```

```

        pie.legend().title()
            .text(channelTitle)
            .padding(0d, 0d, 10d, 0d);

        pie.legend()
            .position("center-bottom")
            .itemsLayout(LinearLayout.HORIZONTAL)
            .align(Align.CENTER);

        anyChartView.setChart(pie);
    }
}

```

Файл BudgetPlannerFragment.java

```

package com.example.expensetracker.budgetplanner.ui;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.ListView;

import com.example.expensetracker.R;
import com.example.expensetracker.budgetplanner.Plan;
import com.example.expensetracker.budgetplanner.PlansManager;
import com.example.expensetracker.budgetplanner.activities.PlanInfoActivity;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.ArrayList;

public class BudgetPlannerFragment extends Fragment
{
    ListView plansListView;

    static PlansListAdapter plansListAdapter;
    ArrayList<Plan> plansList;

```

```

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                                Bundle savedInstanceState) {

            View root =
inflater.inflate(R.layout.fragment_budget_planner, container,
false);
            final Context context =
getActivity().getApplicationContext();

            plansListView = root.findViewById(R.id.plans_lv);

            plansList = new ArrayList<>();
            plansList = PlansManager.getPlansList();

            plansListAdapter = new PlansListAdapter(context,
plansList);

            plansListView.setAdapter(plansListAdapter);
            plansListAdapter.notifyDataSetChanged();

            FloatingActionButton fab = root.findViewById(R.id.fab);
            fab.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {

                    startPlanActivity();
                }
            });

            plansListView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> adapterView,
View view, int i, long l) {
                    final Plan selectedPlan = plansList.get(i);
                    try {
                        AlertDialog.Builder alertBuilder = new
AlertDialog.Builder(getActivity());

                        alertBuilder.setMessage("Вы хотите
\"Изменить\" или \"Удалить\" эту цель")
                            .setTitle("Цель - "+
plansList.get(i).getTitle())
                            .setPositiveButton("Изменить", new
DialogInterface.OnClickListener() {
                                @Override
                                public void
onClick(DialogInterface dialogInterface, int i) {
                                    // open

```

```

editSelectedPlan(context,
selectedPlan);
    }
    })
    .setNegativeButton("Удалить", new
DialogInterface.OnClickListener() {
        @Override
        public void
onClick(DialogInterface dialogInterface, int i) {
PlansManager.deletePlan(selectedPlan);
            refresh();
        }
    }).show();
    }catch (Exception e)
    {
        e.printStackTrace();
    }
    });

return root;
}

void editSelectedPlan(Context context , Plan selectedPlan)
{
    try {
        Intent intent = new Intent(context,
PlanInfoActivity.class);

        intent.putExtra("editPlan", true);

        intent.putExtra("planID" ,
selectedPlan.getPlanID());
        intent.putExtra("planTitle",
selectedPlan.getTitle());
        intent.putExtra("planDesc",
selectedPlan.getDescription());
        intent.putExtra("planBudget",
String.valueOf(selectedPlan.getMoneyNeeded()));

        String year =
(selectedPlan.getDeadline().toString()).substring(
selectedPlan.getDeadline().toString().length() - 4);

        String dateStr =
(selectedPlan.getDeadline().toString()).substring(0, 10) + " " +
year;

        Log.i("planDate", "editSelectedPlan: "

```

					ПГУ1.090301.12.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		73

```

        +
(selectedPlan.getDeadline().toString()).substring(0, 10)

        + ", " + year);

        intent.putExtra("planDate", dateStr);
        startActivity(intent);
    } catch (Exception e)
    {
        e.printStackTrace();
    }
}

private void startPlanActivity() {
    Intent intent = new Intent(getContext(),
PlanInfoActivity.class);
    intent.putExtra("editPlan" , false);
    startActivity(intent);
}

@Override
public void onResume() {
    super.onResume();
    plansListAdapter.notifyDataSetChanged();
}

@Override
public void onStart() {
    super.onStart();
    if(plansListAdapter!=null)
    {
        plansListAdapter.notifyDataSetChanged();
    }
}

public static void refresh()
{
    if(plansListAdapter!=null)
    {
        plansListAdapter.notifyDataSetChanged();
    }
}
}

```

Файл BudgetManagerFragment.java

```

package
com.example.expensetracker.mainactivity.fragments.budgetmanager
;

import android.app.AlertDialog;
import android.content.Context;

```

```

import android.content.DialogInterface;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentActivity;
import androidx.fragment.app.FragmentManager;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ListView;

import
com.example.expensetracker.budget_categories.ui.BudgetCategoryI
nfoFragment;
import com.example.expensetracker.R;
import
com.example.expensetracker.budget_categories.BudgetCategory;
import
com.example.expensetracker.budget_categories.BudgetCategoryMana
ger;
import com.example.expensetracker.budgetplanner.PlansManager;
import
com.google.android.material.floatingactionbutton.FloatingActionB
utton;

import java.util.ArrayList;

public class BudgetManagerFragment extends Fragment {

    ListView budgetCategoryListView;

    static BudgetManagerListAdapter budgetManagerListAdapter;
    ArrayList<BudgetCategory> budgetCategories;

    FragmentActivity fragmentActivity;

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle
savedInstanceState) {

        View root =
inflater.inflate(R.layout.fragment_budget_manager, container,
false);
        final Context context =
getActivity().getApplicationContext();
        fragmentActivity = (FragmentActivity) getActivity();
        try {

```

```

        budgetCategoryListView =
root.findViewById(R.id.budget_manager_lv);

        budgetCategories = new ArrayList<>();
        budgetCategories =
BudgetCategoryManager.getBudgetCategories();

        budgetManagerListAdapter = new
BudgetManagerListAdapter(context, budgetCategories);

budgetCategoryListView.setAdapter(budgetManagerListAdapter);
        budgetManagerListAdapter.notifyDataSetChanged();

        budgetCategoryListView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?>
adapterView, View view, int i, long l) {
                final BudgetCategory selectedBudgetCategory
= budgetCategories.get(i);
                AlertDialog.Builder alertBuilder = new
AlertDialog.Builder(getActivity());

                alertBuilder.setMessage("Вы хотите
\"Изменить\" или \"Удалить\" эту категорию")
                    .setTitle("Категория - "+
budgetCategories.get(i).getName())
                    .setPositiveButton("Изменить", new
DialogInterface.OnClickListener() {
                        @Override
                        public void
onClick(DialogInterface dialogInterface, int i) {
editBudgetCategory(selectedBudgetCategory);
                        }
                    })
                    .setNegativeButton("Удалить", new
DialogInterface.OnClickListener() {
                        @Override
                        public void
onClick(DialogInterface dialogInterface, int i) {
BudgetCategoryManager.deleteCategoryEnvelop(selectedBudgetCatego
ry);

                                refresh();
                        }
                    })
                    .show();
            }
        });
    });

```

```

        FloatingActionButton fab =
root.findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                addNewBudgetCategory();
            }
        });

    }catch (Exception e)
    {
        e.printStackTrace();
    }

    return root;
}

void editBudgetCategory( BudgetCategory budgetCategory)
{
    FragmentManager fragmentManager =
fragmentActivity.getSupportFragmentManager();

    BudgetCategoryInfoFragment budgetCategoryInfoFragment =
new BudgetCategoryInfoFragment (budgetCategory);

    budgetCategoryInfoFragment.show(fragmentManager, "New
budget");
}

@Override
public void onResume() {
    super.onResume();
    budgetManagerListAdapter.notifyDataSetChanged();
}

@Override
public void onStart() {
    super.onStart();
    if(budgetManagerListAdapter!=null)
    {
        budgetManagerListAdapter.notifyDataSetChanged();
    }
}

public static void refresh()
{
    if(budgetManagerListAdapter!=null)
    {
        budgetManagerListAdapter.notifyDataSetChanged();
    }
}

```

```

        void addNewBudgetCategory()
        {
            FragmentManager fragmentManager =
fragmentActivity.getSupportFragmentManager();

            BudgetCategoryInfoFragment budgetCategoryInfoFragment =
new BudgetCategoryInfoFragment();

            budgetCategoryInfoFragment.show(fragmentManager, "New
budget");
        }

    }

```

Файл HomeFragment.java

```

package com.example.expensetracker.mainactivity.fragments.home;

import static
com.example.expensetracker.usermoney.UserWallet.resetWallet;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.example.expensetracker.R;
import
com.example.expensetracker.budget_categories.BudgetCategory;
import
com.example.expensetracker.budget_categories.BudgetCategoryMana
ger;
import
com.example.expensetracker.budget_categories.ui.BudgetCategoryL
istAdapter;
import
com.example.expensetracker.transactions.activities.AddingTransa
ctionActivity;
import com.example.expensetracker.usermoney.UserWallet;

```

```

import
com.google.android.material.floatingactionbutton.FloatingActionB
utton;

import java.util.ArrayList;

public class HomeFragment extends Fragment {

    TextView totalExpensesTextView, totalBalanceTextView;
    ListView budgetCategoryListView;

    LinearLayout balanceLayout;

    static BudgetCategoryListAdapter budgetCategoryListAdapter;
    ArrayList<BudgetCategory> budgetCategories;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle
savedInstanceState) {
        View root = inflater.inflate(R.layout.fragment_home,
container, false);
        final Context context =
getActivity().getApplicationContext();

        totalBalanceTextView =
root.findViewById(R.id.total_balance);
        totalExpensesTextView =
root.findViewById(R.id.total_expenses);
        budgetCategoryListView =
root.findViewById(R.id.budget_category_lv);
        balanceLayout = root.findViewById(R.id.balance_layout);

        FloatingActionButton fab = root.findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(context,
AddingTransactionActivity.class);
                startActivity(intent);
            }
        });
        balanceLayout.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AlertDialog.Builder alertBuilder = new
AlertDialog.Builder(getActivity());

                alertBuilder.setMessage("Вы хотите \"Сбросить\"
текущий баланс?")

```

```

        .setTitle("Текущий баланс")
        .setPositiveButton("Сбросить", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface
dialogInterface, int i) {
                resetWallet();
            }
        })
        .setNegativeButton("Отмена", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface
dialogInterface, int i) {
                dialogInterface.dismiss();
            }
        }).show();
    }
});
budgetCategories = new ArrayList<>();
budgetCategories =
BudgetCategoryManager.getBudgetCategories();

budgetCategoryListAdapter = new
BudgetCategoryListAdapter(context, budgetCategories);

budgetCategoryListView.setAdapter(budgetCategoryListAdapter);
budgetCategoryListAdapter.notifyDataSetChanged();

totalBalanceTextView.setText(String.valueOf(UserWallet.getUserMo
ney())+" ₺");

totalExpensesTextView.setText(String.valueOf(BudgetCategoryManag
er.getTotalExpenses())+" ₺");

return root;
}

@Override
public void onResume() {
    super.onResume();
    budgetCategoryListAdapter.notifyDataSetChanged();

totalBalanceTextView.setText(String.valueOf(UserWallet.getUserMo
ney())+" ₺");

totalExpensesTextView.setText(String.valueOf(BudgetCategoryManag
er.getTotalExpenses())+" ₺");
}

```

```

@Override
public void onStart() {
    super.onStart();
    if (budgetCategoryListAdapter != null)
    {
        budgetCategoryListAdapter.notifyDataSetChanged();

totalBalanceTextView.setText (String.valueOf (UserWallet.getUserMoney())+" ₺");

totalExpensesTextView.setText (String.valueOf (BudgetCategoryManager.getTotalExpenses())+" ₺");
    }
}

public static void refresh()
{
    if (budgetCategoryListAdapter != null)
    {
        budgetCategoryListAdapter.notifyDataSetChanged();
    }
}
}

```

Файл AddingIncomeFragment.java

```

package com.example.expensetracker.usermoney.ui;

import android.os.Bundle;

import androidx.fragment.app.DialogFragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.example.expensetracker.R;
import com.example.expensetracker.usermoney.UserWallet;

public class AddingIncomeFragment extends DialogFragment
implements View.OnClickListener {

    View addingIncomeView;
    EditText incomeEditText;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,

                                Bundle savedInstanceState) {

```

```

        addingIncomeView =
inflater.inflate(R.layout.fragment_adding_income, container, false
);

        incomeEditText=
addingIncomeView.findViewById(R.id.income);
        Button addIncomeButton =
addingIncomeView.findViewById(R.id.add_income_button);

        addIncomeButton.setOnClickListener(this);
        return addingIncomeView;
    }

    @Override
    public void onClick(View view) {
        this.dismiss();
        try {
            double income
=Double.parseDouble(incomeEditText.getText().toString());
            UserWallet.addIncome(income);
            Toast.makeText(getContext() , "Ваш баланс равен : "
+ UserWallet.getUserMoney()+ "Р", Toast.LENGTH_LONG).show();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

Файл UserWallettFragment.java

```

package com.example.expensetracker.usermoney.ui;

import android.os.Bundle;

import androidx.fragment.app.DialogFragment;
import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.example.expensetracker.R;
import com.example.expensetracker.usermoney.UserWallet;

import org.w3c.dom.Text;

public class UserWalletFragment extends DialogFragment
implements View.OnClickListener {

```

```

View userWalletView;
TextView userMoneyTextView;

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                        Bundle savedInstanceState) {

    userWalletView =
inflater.inflate(R.layout.fragment_user_wallet, container, false);

    userMoneyTextView=
userWalletView.findViewById(R.id.user_money);
    Button addIncomeButton =
userWalletView.findViewById(R.id.close_button);

userMoneyTextView.setText(String.valueOf(UserWallet.getUserMoney
())+" ₺");

    addIncomeButton.setOnClickListener(this);
    return userWalletView;
}

@Override
public void onClick(View view) {
    this.dismiss();
}
}

```

Файл BudgetCategoryListAdapter.java

```

package com.example.expensetracker.budget_categories.ui;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.LinearLayout;
import android.widget.ProgressBar;
import android.widget.SeekBar;
import android.widget.TextView;

import com.example.expensetracker.R;
import
com.example.expensetracker.budget_categories.BudgetCategory;
import
com.example.expensetracker.transactions.activities.ListingCateg
oryTransactionsActivity;

```

```

import java.util.ArrayList;

public class BudgetCategoryListAdapter extends BaseAdapter
{
    ArrayList<BudgetCategory> budgetCategories;
    Context context;

    public BudgetCategoryListAdapter(Context context,
    ArrayList<BudgetCategory> budgetCategories) {
        this.budgetCategories = budgetCategories;

        this.context = context;
    }

    @Override
    public int getCount() {
        return budgetCategories.size();
    }

    @Override
    public Object getItem(int i) {
        return budgetCategories.get(i);
    }

    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup)
    {
        LayoutInflater inflater = (LayoutInflater)
context.getSystemService( Context.LAYOUT_INFLATER_SERVICE );
        view =
inflater.inflate(R.layout.list_item_budget_categoty, null);

        final BudgetCategory budgetCategory =
budgetCategories.get(i);

        LinearLayout linearLayout =
view.findViewById(R.id.expenses_layout);
        ProgressBar progressBar =
view.findViewById(R.id.budget_seekbar);
        TextView categoryName =
view.findViewById(R.id.category_name);
        TextView currentExpenses =
view.findViewById(R.id.current_expenses);
        TextView maxExpenses =
view.findViewById(R.id.max_expenses);

        categoryName.setText(budgetCategory.getName());

```

```

currentExpenses.setText (String.valueOf (budgetCategory.getCurrent
Budget ())+ " ₺");

maxExpenses.setText (String.valueOf (budgetCategory.getMaxBudget (
))+ " ₺");

        progressBar.setMax ((int) budgetCategory.getMaxBudget ());
        progressBar.setProgress ((int)
budgetCategory.getCurrentBudget ());

        if (budgetCategory.getCurrentBudget () >
budgetCategory.getMaxBudget ())
        {

linearLayout.setBackgroundResource (R.drawable.customborder);
        }

        linearLayout.setOnClickListener (new
View.OnClickListener () {
            @Override
            public void onClick (View view) {
                try{
                    Intent intent = new Intent ( context ,
ListingCategoryTransactionsActivity.class);

intent.addFlags (Intent.FLAG_ACTIVITY_NEW_TASK);
                    intent.putExtra ("category",
budgetCategory.getName ());
                    context.startActivity (intent);
                }
                catch (Exception e)
                {
                    e.printStackTrace ();
                }
            }
        });

        progressBar.setOnTouchListener (new
View.OnTouchListener () {
            @Override
            public boolean onTouch (View view, MotionEvent
motionEvent) {
                return true;
            }
        });

        return view;
    }
}

```

Файл PlansListAdapter.java

```
package com.example.expensetracker.budgetplanner.ui;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.RelativeLayout;
import android.widget.TextView;

import com.example.expensetracker.R;
import com.example.expensetracker.budgetplanner.Plan;
import com.example.expensetracker.usermoney.UserWallet;

import java.util.ArrayList;
import java.util.Calendar;

public class PlansListAdapter extends BaseAdapter {

    Context context;
    ArrayList<Plan> plans;

    public PlansListAdapter(Context context, ArrayList<Plan>
plans) {
        this.context = context;
        this.plans = plans;
    }

    @Override
    public int getCount() {
        return plans.size();
    }

    @Override
    public Object getItem(int i) {
        return plans.get(i);
    }

    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup)
    {
        LayoutInflater inflater = (LayoutInflater)
context.getSystemService( Context.LAYOUT_INFLATER_SERVICE );
        view = inflater.inflate(R.layout.list_item_plans, null);
```

```

        final Plan plan = plans.get(i);

        RelativeLayout planCardLayout =
view.findViewById(R.id.plan_card_layout);
        TextView planTitle = view.findViewById(R.id.plan_title);
        TextView planDesc = view.findViewById(R.id.plan_desc);
        TextView planDate = view.findViewById(R.id.plan_date);
        TextView planBudget =
view.findViewById(R.id.plan_budget);

        String year = (plan.getDeadline().toString()).substring(
            plan.getDeadline().toString().length()-4);

        String dateStr =
(plan.getDeadline().toString()).substring(0 , 11) + ", " + year;

        planTitle.setText(plan.getTitle());
        planDesc.setText(plan.getDescription());
        planDate.setText(dateStr);

planBudget.setText(String.valueOf(plan.getMoneyNeeded())+" ₺");

        if(plan.getMoneyNeeded() <= UserWallet.getUserMoney())
        {

planCardLayout.setBackgroundResource(R.color.doneColor);
        }

        if
(plan.getDeadline().compareTo(Calendar.getInstance().getTime())
< 0)
        {

planCardLayout.setBackgroundResource(R.color.worryColor);
        }

        return view;
    }
}

```

Файл BudgetManagerListAdapter.java

```

package
com.example.expensetracker.mainactivity.fragment.budgetmanager;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

import com.example.expensetracker.R;

```

					ПГУ1.090301.12.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		87

```

import
com.example.expensetracker.budget_categories.BudgetCategory;

import java.util.ArrayList;

public class BudgetManagerListAdapter extends BaseAdapter
{
    ArrayList<BudgetCategory> budgetCategories;
    Context context;

    public BudgetManagerListAdapter(Context context,
    ArrayList<BudgetCategory> budgetCategories) {
        this.budgetCategories = budgetCategories;

        this.context = context;
    }

    @Override
    public int getCount() {
        return budgetCategories.size();
    }

    @Override
    public Object getItem(int i) {
        return budgetCategories.get(i);
    }

    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup)
    {
        LayoutInflater inflater = (LayoutInflater)
context.getSystemService( Context.LAYOUT_INFLATER_SERVICE );
        view =
inflater.inflate(R.layout.list_item_budget_manager, null);

        final BudgetCategory budgetCategory =
budgetCategories.get(i);

        TextView categoryName =
view.findViewById(R.id.category_name);
        TextView maxExpenses =
view.findViewById(R.id.max_expenses);

        categoryName.setText(budgetCategory.getName());

maxExpenses.setText(String.valueOf(budgetCategory.getMaxBudget()
)+" ₺");
    }
}

```

```

        return view;
    }
}

```

Файл TransactionListAdapter.java

```

package com.example.expensetracker.transactions.ui;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

import com.example.expensetracker.R;
import com.example.expensetracker.transactions.Transaction;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

public class TransactionListAdapter extends BaseAdapter
{
    Context context;
    ArrayList<Transaction> transactionList;

    public TransactionListAdapter(Context context,
    ArrayList<Transaction> transactionList)
    {
        this.context = context;
        this.transactionList = transactionList;
    }

    @Override
    public int getCount() {
        return transactionList.size();
    }

    @Override
    public Object getItem(int i) {
        return transactionList.get(i);
    }

    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup)
    {

```

```

        LayoutInflater inflater = (LayoutInflater)
context.getSystemService( Context.LAYOUT_INFLATER_SERVICE );
        view = inflater.inflate(R.layout.list_item_transaction,
null);

        final Transaction transaction = transactionList.get(i);

        TextView productName =
view.findViewById(R.id.product_name);
        TextView productPrice =
view.findViewById(R.id.product_price);
        TextView buyDate = view.findViewById(R.id.buy_date);
        String year =
(transaction.getBuyDate().toString()).substring(
transaction.getBuyDate().toString().length()-4);

        String dateStr =
(transaction.getBuyDate().toString()).substring(0 , 11) + ", " +
year;
        Date date = transaction.getBuyDate();
        productName.setText(transaction.getProductName());

productPrice.setText(String.valueOf(transaction.getPrice())+"
₽");
        buyDate.setText(dateStr);
        //buyDate.setText(date.toString());
        return view;
    }
}

```

Файл BudgetCategory.java

```

package com.example.expensetracker.budget_categories;

public class BudgetCategory
{
    private int ID;
    private String name;
    private double maxBudget;
    private double currentBudget;

    public BudgetCategory(int ID, String name, double maxBudget)
    {
        this.ID = ID;
        this.name = name;
        this.maxBudget = maxBudget;
        this.currentBudget = 0;
    }

    public BudgetCategory(int ID, String name, double maxBudget,
double currentBudget) {
        this.ID = ID;
        this.name = name;
    }
}

```

```

        this.maxBudget = maxBudget;
        this.currentBudget = currentBudget;
    }

    public int getID() {
        return ID;
    }

    public void setID(int ID) {
        this.ID = ID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getMaxBudget() {
        return maxBudget;
    }

    public void setMaxBudget(double maxBudget) {
        this.maxBudget = maxBudget;
    }

    public double getCurrentBudget() {
        return currentBudget;
    }

    public void setCurrentBudget(double currentBudget) {
        this.currentBudget = currentBudget;
    }

    @Override
    public String toString() {
        return "BudgetCategory{" +
            "name='" + name + '\'' +
            ", maxBudget=" + maxBudget +
            ", currentBudget=" + currentBudget +
            '\'';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof BudgetCategory)) return false;

        BudgetCategory that = (BudgetCategory) o;

```

```

        if (Double.compare(that.getMaxBudget(), getMaxBudget())
!= 0) return false;
        if (Double.compare(that.getCurrentBudget(),
getCurrentBudget()) != 0) return false;
        return getName() != null ?
getName().equals(that.getName()) : that.getName() == null;
    }

    @Override
    public int hashCode() {
        int result;
        long temp;
        result = getName() != null ? getName().hashCode() : 0;
        temp = Double.doubleToLongBits(getMaxBudget());
        result = 31 * result + (int) (temp ^ (temp >>> 32));
        temp = Double.doubleToLongBits(getCurrentBudget());
        result = 31 * result + (int) (temp ^ (temp >>> 32));
        return result;
    }
}

```

Файл BudgetCategoryManager.java

```

package com.example.expensetracker.budget_categories;

import android.content.ContentValues;
import android.database.Cursor;

import com.example.expensetracker.budgetplanner.Plan;
import
com.example.expensetracker.dbmanagment.BudgetCategoriesTable;
import
com.example.expensetracker.dbmanagment.DatabaseController;
import com.example.expensetracker.dbmanagment.DatabaseManager;
import com.example.expensetracker.dbmanagment.PlansTable;
import com.example.expensetracker.usermoney.UserWallet;

import java.util.ArrayList;
import java.util.HashMap;

public class BudgetCategoryManager
{
    static ArrayList<BudgetCategory> budgetCategories;
    static DatabaseManager databaseManager;

    static public boolean LoadCategoryEnvelop()
    {
        budgetCategories = new ArrayList<>();
        databaseManager =
DatabaseController.getDatabaseManager();

        Cursor cursor =
databaseManager.queryTable(BudgetCategoriesTable.TableName

```

```

        ,null,null,
        null,null,null,null);

        if(cursor.moveToFirst()) {
            do {
                int Id =
cursor.getInt(cursor.getColumnIndex(BudgetCategoriesTable.colID)
);
                String name =
cursor.getString(cursor.getColumnIndex(BudgetCategoriesTable.col
Name));
                double maxBudget =
cursor.getDouble(cursor.getColumnIndex(BudgetCategoriesTable.col
MaxBudget));
                double currentBudget =
cursor.getDouble(cursor.getColumnIndex(BudgetCategoriesTable.col
CurrentBudget));

                budgetCategories.add(new BudgetCategory(Id ,
name , maxBudget , currentBudget));
            } while (cursor.moveToNext());
        }

        return true;
    }

    public static HashMap<String , Double>
getCategoryExpensesMap()
    {
        HashMap<String , Double> output = new HashMap<>();
        for (int i = 0 ; i < budgetCategories.size() ; i++)
        {
            output.put(budgetCategories.get(i).getName()
,budgetCategories.get(i).getCurrentBudget());
        }
        return output;
    }

    public static ArrayList<BudgetCategory>
getBudgetCategories() {
        return budgetCategories;
    }

    public static BudgetCategory getBudgetCategoryByName(int i)
    {
        return budgetCategories.get(i);
    }

    static public ArrayList<String> getCategoryNames()
    {
        ArrayList<String> names = new ArrayList<>();
        for (int i = 0 ; i < budgetCategories.size() ; i++)
        {
            names.add(budgetCategories.get(i).getName());
        }
        return names;
    }

```

```

    }
    static public BudgetCategory getBudgetCategoryByName(String
category)
    {
        for (int i = 0 ; i < budgetCategories.size() ; i++)
        {
            if(budgetCategories.get(i).getName().equalsIgnoreCase(category))
            {
                return budgetCategories.get(i);
            }
        }
        return null;
    }

    static public boolean editCategoryEnvelop(int id , String
category , double budget)
    {
        String selection = BudgetCategoriesTable.colID + " = ?
";
        String[] selectionArg = {String.valueOf(id)};

        ContentValues contentValues = new ContentValues();

        contentValues.put(BudgetCategoriesTable.colName,category);
        contentValues.put(BudgetCategoriesTable.colMaxBudget,budget);

        int c =
databaseManager.updateEntries(BudgetCategoriesTable.TableName ,
contentValues,selection,selectionArg);
        if(c > 0)
        {
            for (int i = 0 ; i < budgetCategories.size() ; i++)
            {
                if(budgetCategories.get(i).getID()==id)
                {
                    budgetCategories.get(i).setMaxBudget(budget);
                    budgetCategories.get(i).setName(category);
                    return true;
                }
            }
        }

        return false;
    }

    static public boolean deleteCategoryEnvelop(BudgetCategory
budgetCategory)
    {

```

```

        String selection = BudgetCategoriesTable.colID + " = ?
";
        String[] selectionArg =
{String.valueOf(budgetCategory.getID())};

        int c =
databaseManager.deleteEntries(BudgetCategoriesTable.TableName ,
selection,selectionArg);

        for (int i = 0 ; i < budgetCategories.size(); i++)
        {
            if
(budgetCategories.get(i).getName().equalsIgnoreCase(budgetCatego
ry.getName()))
            {
                budgetCategories.remove(i);
                return true;
            }
        }
        return false;
    }

    static public boolean addNewBudgetCategory(String name ,
double budget)
    {
        ContentValues contentValues = new ContentValues();
        contentValues.put(BudgetCategoriesTable.colName,name);

        contentValues.put(BudgetCategoriesTable.colMaxBudget,budget);

        contentValues.put(BudgetCategoriesTable.colCurrentBudget,0);
        long ID =
databaseManager.insert(BudgetCategoriesTable.TableName,
contentValues);

        if(ID > 0) {

            BudgetCategory newBudgetCategory = new
BudgetCategory((int) ID, name, budget);
            budgetCategories.add(newBudgetCategory);

            return true;
        }
        else {
            return false;
        }
    }

    static public boolean addToCurrentExpenses(int id, double
addedValue)
    {
        double newValue = addedValue;

```

```

        for (int i = 0 ; i < budgetCategories.size() ; i++)
        {
            if(budgetCategories.get(i).getID() == id)
            {
                double oldValue  =
budgetCategories.get(i).getCurrentBudget();

budgetCategories.get(i).setCurrentBudget(oldValue+addedValue);
                newValue = oldValue+addedValue;

                String selection = BudgetCategoriesTable.colID +
" = ? ";

                String[] selectionArg = {String.valueOf(id)};
                ContentValues contentValues = new
ContentValues();

                contentValues.put(BudgetCategoriesTable.colCurrentBudget,newValu
e);

                int c =
databaseManager.updateEntries(BudgetCategoriesTable.TableName ,
contentValues,selection,selectionArg);

                return true;
            }
        }
        return false;
    }

    static public boolean deleteToCurrentExpenses(int id, double
addedValue)
    {
        double newValue  = addedValue;
        UserWallet.returnToWallet(newValue);
        for (int i = 0 ; i < budgetCategories.size() ; i++)
        {
            if(budgetCategories.get(i).getID() == id)
            {
                double oldValue  =
budgetCategories.get(i).getCurrentBudget();
                if (oldValue < addedValue){
                    budgetCategories.get(i).setCurrentBudget(0);
                }
                else {

budgetCategories.get(i).setCurrentBudget(oldValue-addedValue);
                    newValue = oldValue-addedValue;
                }

                String selection = BudgetCategoriesTable.colID +
" = ? ";

                String[] selectionArg = {String.valueOf(id)};
                ContentValues contentValues = new
ContentValues();

```

```

contentValues.put (BudgetCategoriesTable.colCurrentBudget,newValu
e);

        int c =
databaseManager.updateEntries (BudgetCategoriesTable.TableName ,
contentValues,selection,selectionArg);

        return true;
    }
}
return false;
}

static public BudgetCategory getBudgetCategoryById(int id) {
    for (int i = 0; i < budgetCategories.size(); i++) {
        if (budgetCategories.get(i).getID() == id) {
            return budgetCategories.get(i);
        }
    }
    return null;
}

static public int getCategoryIDFromName (String name)
{
    for (int i = 0; i < budgetCategories.size(); i++) {
        if
(budgetCategories.get(i).getName().equalsIgnoreCase(name))
        {
            return budgetCategories.get(i).getID();
        }
    }
    return 0;
}

static public double getTotalExpenses()
{
    double total = 0;

    for (int i = 0; i < budgetCategories.size(); i++) {
        total+= budgetCategories.get(i).getCurrentBudget();
    }

    return total;
}

public static void setAllBudgetCategoriesExpensesToZero()
{
    for (int i = 0; i < budgetCategories.size(); i++) {
        budgetCategories.get(i).setCurrentBudget (0);
    }
}

```

```
}
```

Файл Plan.java

```
package com.example.expensetracker.budgetplanner;

import java.util.Calendar;
import java.util.Date;

public class Plan
{
    int planID;
    private String title;
    private String description;
    private double MoneyNeeded;
    private Date deadline;

    public Plan(int planID ,String title, String description,
double moneyNeeded, Date deadline) {
        this.planID = planID;
        this.title = title;
        this.description = description;
        MoneyNeeded = moneyNeeded;
        this.deadline = deadline;
    }

    public Plan(int planID ,String title, String description,
double moneyNeeded) {
        this.planID = planID;
        this.title = title;
        this.description = description;
        MoneyNeeded = moneyNeeded;

        this.deadline = Calendar.getInstance().getTime();
    }

    public int getPlanID() {
        return planID;
    }

    public void setPlanID(int planID) {
        this.planID = planID;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
```

```

        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public double getMoneyNeeded() {
        return MoneyNeeded;
    }

    public void setMoneyNeeded(double moneyNeeded) {
        MoneyNeeded = moneyNeeded;
    }

    public Date getDeadline() {
        return deadline;
    }

    public void setDeadline(Date deadline) {
        this.deadline = deadline;
    }

    @Override
    public String toString() {
        return "Plan{" +
            "title='" + title + '\'' +
            ", description='" + description + '\'' +
            ", MoneyNeeded=" + MoneyNeeded +
            ", deadline=" + deadline +
            '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Plan)) return false;

        Plan plan = (Plan) o;

        if (Double.compare(plan.getMoneyNeeded(),
getMoneyNeeded()) != 0) return false;
        if (getTitle() != null ?
!getTitle().equals(plan.getTitle()) : plan.getTitle() != null)
            return false;
        if (getDescription() != null ?
!getDescription().equals(plan.getDescription()) :
plan.getDescription() != null)
            return false;
        return getDeadline() != null ?
getDeadline().equals(plan.getDeadline()) : plan.getDeadline() ==
null;
    }

```

```

    }

    @Override
    public int hashCode() {
        int result;
        long temp;
        result = getTitle() != null ? getTitle().hashCode() : 0;
        result = 31 * result + (getDescription() != null ?
getDescription().hashCode() : 0);
        temp = Double.doubleToLongBits(getMoneyNeeded());
        result = 31 * result + (int) (temp ^ (temp >>> 32));
        result = 31 * result + (getDeadline() != null ?
getDeadline().hashCode() : 0);
        return result;
    }
}

```

Файл PlanManager.java

```

package com.example.expensetracker.budgetplanner;

import android.content.ContentValues;
import android.database.Cursor;

import
com.example.expensetracker.dbmanagment.DatabaseController;
import com.example.expensetracker.dbmanagment.DatabaseManager;
import com.example.expensetracker.dbmanagment.PlansTable;
import
com.example.expensetracker.globalOperations.DateStringFormatter
;

import java.text.ParseException;
import java.util.ArrayList;
import java.util.Date;

public class PlansManager
{
    static ArrayList<Plan> plansList;
    static DatabaseManager databaseManager;

    public static ArrayList<Plan> getPlansList() {
        return plansList;
    }

    public static void loadPlans() throws ParseException
    {
        plansList = new ArrayList<>();
        databaseManager =
DatabaseController.getDatabaseManager();

        Cursor cursor =
databaseManager.queryTable(PlansTable.TableName

```

```

        ,null,null,
        null,null,null,null));

        if(cursor.moveToFirst()) {
            do {
                int Id =
cursor.getInt(cursor.getColumnIndex(PlansTable.colID));
                String title =
cursor.getString(cursor.getColumnIndex(PlansTable.colTitle));
                String description =
cursor.getString(cursor.getColumnIndex(PlansTable.colDescription
));
                double moneyNeeded =
cursor.getDouble(cursor.getColumnIndex(PlansTable.colNeededMoney
));
                Date deadline =
DateStringFormatter.StringToDate(cursor.getString(cursor.getColu
mnIndex(PlansTable.colDeadline)));

                plansList.add(new
Plan(Id,title,description,moneyNeeded,deadline));
            } while (cursor.moveToNext());
        }

        public static void addPlan(String title, String desc, double
budget, Date deadline)
        {
            ContentValues contentValues = new ContentValues();

            contentValues.put(PlansTable.colDeadline,DateStringFormate
r.DateToString(deadline));
            contentValues.put(PlansTable.colDescription,desc);
            contentValues.put(PlansTable.colNeededMoney,budget);
            contentValues.put(PlansTable.colTitle,title);

            long ID = databaseManager.insert(PlansTable.TableName,
contentValues);

            if(ID > 0) {
                plansList.add(new Plan((int)ID,
title,desc,budget,deadline));
            }
        }

        public static void editPlan(int ID , String title, String
desc, double budget, Date deadline)
        {
            String selection = PlansTable.colID + " = ? ";
            String[] selectionArg = {String.valueOf(ID)};

```

```

        ContentValues contentValues = new ContentValues();

        contentValues.put(PlansTable.colTitle, title);
        contentValues.put(PlansTable.colDescription, desc);
        contentValues.put(PlansTable.colNeededMoney, budget);

        contentValues.put(PlansTable.colDeadline, DateStringFormatter.DateToString(deadline));

        int c =
        databaseManager.updateEntries(PlansTable.TableName ,
        contentValues, selection, selectionArg);

        if(c > 0) {

            for (int i = 0; i < plansList.size(); i++) {
                if (plansList.get(i).getPlanID() == ID) {
                    plansList.get(i).setTitle(title);
                    plansList.get(i).setDeadline(deadline);
                    plansList.get(i).setDescription(desc);
                    plansList.get(i).setMoneyNeeded(budget);
                    return;
                }
            }

        }

        public static void deletePlan(Plan plan)
        {
            String selection = PlansTable.colID + " = ? ";
            String[] selectionArg =
            {String.valueOf(plan.getPlanID())};

            int c =
            databaseManager.deleteEntries(PlansTable.TableName ,
            selection, selectionArg);

            for (int i = 0 ; i < plansList.size() ; i++)
            {
                if
                (plansList.get(i).getTitle().equalsIgnoreCase(plan.getTitle()))
                {
                    plansList.remove(i);
                    return;
                }
            }

        }
    }
}

```

Файл Transaction.java

```
package com.example.expensetracker.transactions;

import
com.example.expensetracker.budget_categories.BudgetCategory;

import java.util.Date;

public class Transaction
{
    private int budgetCategoryID;
    private String productName;
    private double price;
    private Date buyDate;

    public Transaction(int budgetCategoryID, String productName,
double price, Date buyDate) {
        this.budgetCategoryID = budgetCategoryID;
        this.productName = productName;
        this.price = price;
        this.buyDate = buyDate;
    }

    public int getBudgetCategoryID() {
        return budgetCategoryID;
    }

    public void setBudgetCategoryID(int budgetCategoryID) {
        this.budgetCategoryID = budgetCategoryID;
    }

    public String getProductName() {
        return productName;
    }

    public void setProductName(String productName) {
        this.productName = productName;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public Date getBuyDate() {
        return buyDate;
    }
}
```

```

public void setBuyDate(Date buyDate) {
    this.buyDate = buyDate;
}

@Override
public String toString() {
    return "Transaction{" +
        "budgetCategory=" + budgetCategoryID +
        ", productName='" + productName + '\'' +
        ", price=" + price +
        ", buyDate=" + buyDate +
        '}';
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Transaction)) return false;

    Transaction that = (Transaction) o;

    if (getBudgetCategoryID() != that.getBudgetCategoryID())
return false;
    if (Double.compare(that.getPrice(), getPrice()) != 0)
return false;
    if (getProductName() != null ?
!getProductName().equals(that.getProductName()) :
that.getProductName() != null)

        return false;

    return getBuyDate() != null ?
getBuyDate().equals(that.getBuyDate()) : that.getBuyDate() ==
null;
}

@Override
public int hashCode() {

    int result;
    long temp;
    result = getBudgetCategoryID();
    result = 31 * result + (getProductName() != null ?
getProductName().hashCode() : 0);
    temp = Double.doubleToLongBits(getPrice());
    result = 31 * result + (int) (temp ^ (temp >>> 32));
    result = 31 * result + (getBuyDate() != null ?
getBuyDate().hashCode() : 0);
    return result;
}
}

```

Файл TransactionManager.java

```
package com.example.expensetracker.transactions;

import android.content.ContentValues;
import android.database.Cursor;

import com.example.expensetracker.budget_categories.BudgetCategory;
import com.example.expensetracker.budget_categories.BudgetCategoryManager;
import com.example.expensetracker.dbmanagment.BudgetCategoriesTable;
import com.example.expensetracker.dbmanagment.DatabaseController;
import com.example.expensetracker.dbmanagment.DatabaseManager;
import com.example.expensetracker.dbmanagment.PlansTable;
import com.example.expensetracker.dbmanagment.TransactionsTable;
import com.example.expensetracker.globalOperations.DateStringFormatter;

import java.text.ParseException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Locale;

public class TransactionManager
{
    static ArrayList<Transaction> transactions;
    static DatabaseManager databaseManager;

    public static boolean loadTransactions() throws
    ParseException {
        if(BudgetCategoryManager.getBudgetCategories() == null)
        {
            BudgetCategoryManager.LoadCategoryEnvelop();
        }

        databaseManager =
        DatabaseController.getDatabaseManager();

        transactions = new ArrayList<>();

        String currentMonth =
        DateStringFormatter.getCurrentMonth();
        String currentYear =
        DateStringFormatter.getCurrentYear();
        String pattern = "%"+currentMonth+"%"+currentYear;
```

					ПГУ1.090301.12.001 ПЗ	Лист
						105
Изм.	Лист	№ докум.	Подпись	Дата		

```

        String selection = TransactionsTable.colBuyDate + " LIKE
? "; //ID
        String[] selectionArg = {pattern};

        Cursor cursor =
databaseManager.queryTable(TransactionsTable.TableName
        ,null,selection,
        selectionArg,null,null,null);

        if(cursor.moveToFirst()) {
            do {
                int Id =
cursor.getInt(cursor.getColumnIndex(TransactionsTable.colID));
                String productName =
cursor.getString(cursor.getColumnIndex(TransactionsTable.colProd
uctName));
                double price =
cursor.getDouble(cursor.getColumnIndex(TransactionsTable.colPric
e));
                String buyDate =
cursor.getString(cursor.getColumnIndex(TransactionsTable.colBuyD
ate));
                int colBudgetID =
cursor.getInt(cursor.getColumnIndex(TransactionsTable.colBudgetI
D));

                DateStringFormatter.StringToDate(buyDate);

                transactions.add(new Transaction(colBudgetID ,
productName , price,StringFormatter.StringToDate(buyDate)));

            } while (cursor.moveToNext());
        }

        return true;
    }

    public static ArrayList<Transaction> getAllTransactions()
    {
        return transactions;
    }

    public static ArrayList<Transaction>
getTransactionsByCategoryID(int categoryID)
    {
        ArrayList<Transaction> output = new ArrayList<>();
        for(int i = 0 ; i < transactions.size() ; i++)
        {
            int ID = transactions.get(i).getBudgetCategoryID();
            if (ID == (categoryID))

```

```

        {
            output.add(transactions.get(i));
        }
    }
    return output;
}

public static boolean deleteTransaction(Transaction
transaction)
{
    String selection = TransactionsTable.colBudgetID + " = ?
";
    String[] selectionArg =
{String.valueOf(transaction.getBudgetCategoryID())};

    int c =
databaseManager.deleteEntries(TransactionsTable.TableName ,
selection,selectionArg);

    for (int i = 0 ; i < transactions.size() ; i++)
    {
        if
(transaction.get(i).getProductName().equalsIgnoreCase(transacti
on.getProductName()))
        {

BudgetCategoryManager.deleteToCurrentExpenses(transaction.getBud
getCategoryID(), transaction.getPrice());
            transactions.remove(i);
            return true;
        }
    }
    return false;
}

public static boolean addNewTransaction(Transaction
transaction)
{
    ContentValues contentValues = new ContentValues();

    contentValues.put(TransactionsTable.colBudgetID,transaction.getB
udgetCategoryID());

    contentValues.put(TransactionsTable.colBuyDate,DateStringFormatt
er.DateToString(transaction.getBuyDate()));

```

```

contentValues.put (TransactionsTable.colPrice,transaction.getPrice());

contentValues.put (TransactionsTable.colProductName,transaction.getProduct());

        long ID =
databaseManager.insert (TransactionsTable.TableName,
contentValues);

        if (ID > 0) {

                String buyMonth =
(DateStringFormatter.DateToString (transaction.getBuyDate())) .substring(4,7);
                String currentMonth =
DateStringFormatter.getCurrentMonth();

                if (currentMonth.equalsIgnoreCase (buyMonth)) {

BudgetCategoryManager.addToCurrentExpenses (transaction.getBudgetCategoryID(), transaction.getPrice());
                transactions.add (transaction);
                }
                return true;
        }
        else {
                return false;
        }

    }

}

```

Файл UserWallet.java

```

package com.example.expensetracker.usermoney;

import android.content.Context;
import android.content.SharedPreferences;
import android.util.Log;

import com.example.expensetracker.R;

public class UserWallet
{
    static double userMoney;
    private static SharedPreferences sharedPreferences;
    private static Context userContext;

    static public void loadUserMoney(Context context)

```

```

        {
            try {
                userContext = context;
                sharedPreferences =
context.getSharedPreferences(String.valueOf(R.string.app_name),
Context.MODE_PRIVATE);
                String m = sharedPreferences.getString("userMoney",
"0");

                Log.i("UserWallet", "loadUserMoney: " + m);

                userMoney = Double.parseDouble(m);

            }catch (Exception e)
            {
                e.printStackTrace();
            }
        }

        public static double getUserMoney() {

            return userMoney;
        }

        public static void addIncome(double income)
        {
            userMoney+=income;
            save();
        }

        public static void takeFromWallet(double expense)
        {
            userMoney-=expense;
            save();
        }

        public static void returnToWallet(double expense)
        {
            userMoney+=expense;
            save();
        }

        public static void resetWallet(){
            userMoney = 0;
            save();
        }

        private static void save() {
            try {
                SharedPreferences.Editor editor =
sharedPreferences.edit();

```

```

        editor.putString("userMoney",
String.valueOf(userMoney));

        editor.commit();
    }catch (Exception e)
    {
        e.printStackTrace();
    }
}
}

```

Файл DatabaseController.java

```

package com.example.expensetracker.dbmanagment;

import android.content.Context;

public class DatabaseController
{
    static private DatabaseManager databaseManager;

    public DatabaseController(Context context)
    {
        if(databaseManager == null)
        {
            databaseManager = new DatabaseManager(context);
        }
    }

    public static DatabaseManager getDatabaseManager() {

        return databaseManager;
    }
}

```

Файл DatabaseManager.java

```

package com.example.expensetracker.dbmanagment;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.widget.Toast;

public class DatabaseManager
{
    private SQLiteDatabase sqLiteDatabase;

    private final String DatabaseName = "budgetdbnew";

    int DBVersion = 1;
}

```

```

class DBHelper extends SQLiteOpenHelper
{
    Context context;

    public DBHelper(Context context)
    {
        super(context, DatabaseName, null, DBVersion);
        this.context = context;
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase)
    {
        ContentValues contentValues = new ContentValues();

        sqLiteDatabase.execSQL(PlansTable.createPlansTableSQLQuery);

        sqLiteDatabase.execSQL(BudgetCategoriesTable.createBudgetCategoriesTableSQLQuery);

        sqLiteDatabase.execSQL(TransactionsTable.createTransactionTableSQLQuery);

        contentValues.put(BudgetCategoriesTable.colName, "Разное");
        contentValues.put(BudgetCategoriesTable.colID, 1);

        contentValues.put(BudgetCategoriesTable.colCurrentBudget, 0);

        contentValues.put(BudgetCategoriesTable.colMaxBudget, 10000);

        sqLiteDatabase.insert(BudgetCategoriesTable.TableName, null, contentValues);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1)
    {
        sqLiteDatabase.execSQL(PlansTable.DropPlansTable);

        sqLiteDatabase.execSQL(TransactionsTable.DropTransactionsTable);

        sqLiteDatabase.execSQL(BudgetCategoriesTable.DropBudgetCategoriesTable);

        onCreate(sqLiteDatabase);
    }

    public DatabaseManager(Context context)
    {
        DBHelper dbHelper = new DBHelper(context);
    }
}

```

```

        sqLiteDatabase = dbHelper.getWritableDatabase();
    }

    public long insert(String tableName , ContentValues
contentValues)
    {
        return
sqLiteDatabase.insert(tableName,null,contentValues);
    }

    public Cursor queryTable(String tableName , String[]
projection , String Selection , String[] SelectionArguments ,
String GroupBy, String having,
String sortOrder)
    {

        SQLiteQueryBuilder sqLiteQueryBuilder = new
SQLiteQueryBuilder();

        sqLiteQueryBuilder.setTables(tableName);

        Cursor cursor =
sqLiteQueryBuilder.query(sqLiteDatabase,projection,Selection,Sele
ctionArguments,GroupBy,having,sortOrder);

        return cursor;
    }

    public int deleteEntries(String tableName , String Selection
, String[] SelectionArguments)
    {
        int count =
sqLiteDatabase.delete(tableName,Selection,SelectionArguments);

        return count;
    }

    public int updateEntries(String tableName , ContentValues
values , String Selection , String[] SelectionArg)
    {
        int count =
sqLiteDatabase.update(tableName,values,Selection,SelectionArg);

        return count;
    }
}

```

Файл BudgetCategoriesTable.java

```

package com.example.expensetracker.dbmanagment;

public class BudgetCategoriesTable

```

					ПГУ1.090301.12.001 ПЗ	Лист
						112
Изм.	Лист	№ докум.	Подпись	Дата		

```

{
    public static final String TableName = "budget_categories";
    public static final String DropBudgetCategoriesTable = "DROP
TABLE IF EXISTS " + TableName;
    public static final String colID = "colID";
    public static final String colName = "name";
    public static final String colMaxBudget = "maxBudget";
    public static final String colCurrentBudget =
"currentBudget";
    public static final String createBudgetCategoryTableSQLQuery
= "CREATE TABLE IF NOT EXISTS " + TableName + "(\n " +
    "colID Integer Primary key Autoincrement ,\n" +
    colName + " varchar(255),\n" +
    colMaxBudget + " double,\n" +
    colCurrentBudget + " double\n" +
    " ); " ;

}

```

Файл TransactionsTable.java

```

package com.example.expensetracker.dbmanagment;

public class TransactionsTable
{
    public static final String TableName = "transactions";
    static final String DropTransactionsTable = "DROP TABLE IF
EXISTS " + TableName;
    public static final String colID = "colID";
    public static final String colProductName = "productName";
    public static final String colPrice = "price";
    public static final String colBuyDate = "buyDate";
    public static final String colBudgetID = "budgetID";
    static final String createTransactionTableSQLQuery
    = "CREATE TABLE IF NOT EXISTS " + TableName + "(\n
" +
        "colID Integer Primary key Autoincrement ,\n" +
        colProductName + " varchar(255),\n" +
        colPrice + " double,\n" +
        colBuyDate + " varchar(255),\n" +
        colBudgetID + " Integer,\n" +
        "CONSTRAINT FK FOREIGN KEY (budgetID) REFERENCES
budget_categories(colID)" +
        " ); " ;

}

```

Файл PlansTable.java

```

package com.example.expensetracker.dbmanagment;

public class PlansTable
{
    public static final String TableName = "plans";

```

```

        static final String DropPlansTable = "DROP TABLE IF EXISTS "
+ TableName;
        public static final String colID = "colID";
        public static final String colTitle = "title";
        public static final String colDescription = "description";
        public static final String colNeededMoney = "neededMoney";
        public static final String colDeadline = "deadline";

        static final String createPlansTableSQLQuery = "CREATE
TABLE IF NOT EXISTS " + TableName + "(\n " +
        "colID Integer Primary key Autoincrement ,\n" +
        colTitle + " varchar(255),\n" +
        colDescription + " varchar(255),\n" +
        colNeededMoney + " double,\n" +
        colDeadline + " varchar(255)\n" +
        " ); " ;
}

```

Приложение В

(Обязательное)

Android-приложение «Менеджер личных расходов».

Презентация

					ПГУ1.090301.12.001 ПЗ	Лист
						115
Изм.	Лист	№ докум.	Подпись	Дата		

Тема ВКР: Android-приложение «Менеджер личных расходов»

Автор работы студент группы 20ВВП1 Кузин Данил Денисович

Научный руководитель к.т.н., доцент кафедры ВТ ПГУ Слепцов Н.В.

Пенза 2024

Цель и объект работы

- Цель работы : разработка мобильного приложения под ОС Android, позволяющие контролировать личные расходы пользователя.
- Объектом работы является приложение обладающие следующим функционалом :
 - Добавление и удаление затрат
 - Создание и редактирование собственных категорий
 - Анализ общих затрат по категориям
 - Планирование и отслеживание долгосрочных покупок

Актуальность

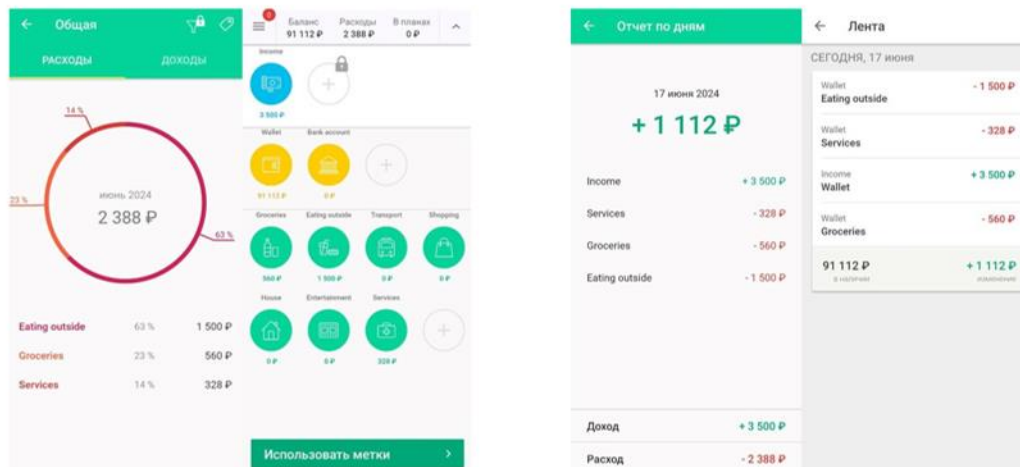
В современном мире, из-за обилия различных продуктов, услуг и товаров многие люди зачастую не контролируют свои расходы, что приводит к неправильному распределению бюджета. Приложения по контролю личных финансов предоставляют необходимый функционал, с помощью которого появляется возможность лучше контролировать все свои расходы, что способствует оптимизации личных затрат. Таким образом человек уменьшает шанс необдуманных расходов, что позволит лучше экономить личные средства и быстрее добиваться больших финансовых покупок.

Аналоги

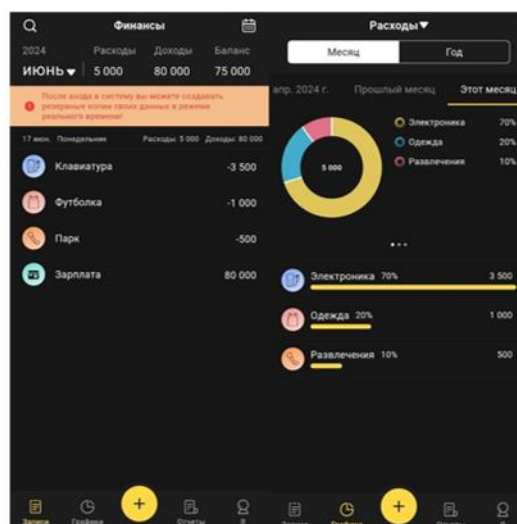
«Деньги ОК»



Аналоги «CoinKeeper»



Аналоги «ФинансЪР»



Постановка задачи

Приложение должно содержать следующие критерии:

- понятный и простой интерфейс для комфортной навигации, а также приятный и интуитивный дизайн;
- возможность добавления, удаления и редактирования расходов, категорий и плановых покупок;
- реализация аналитических графиков для контроля общего бюджета по расходам;
- приложение должно содержать базовый функционал для контроля личного бюджета

Стек технологий



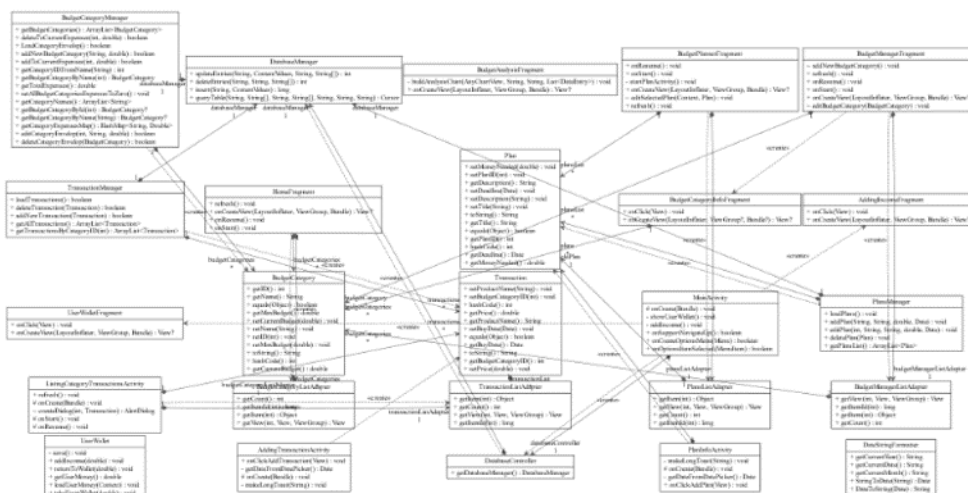
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.12.001 ПЗ

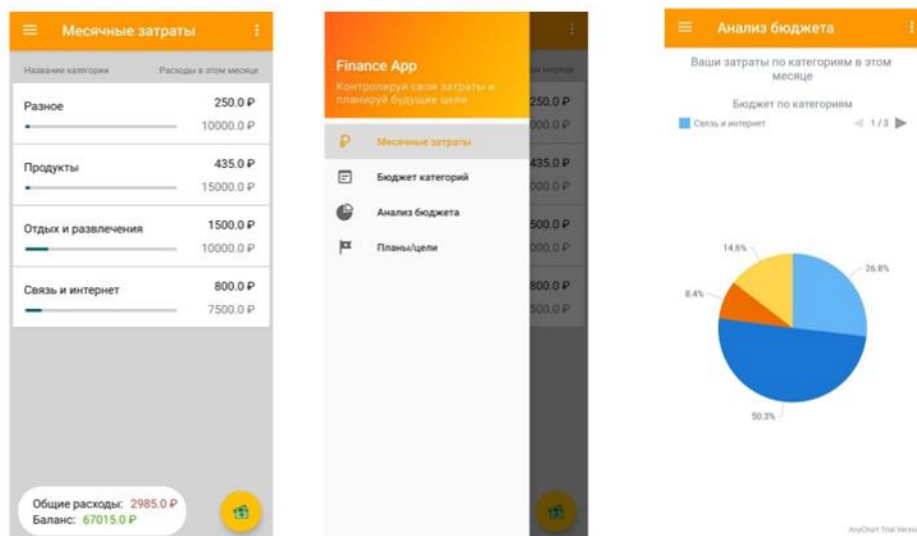
Лист

119





Демонстрация работы приложения



Заключение

В ходе выполнения данной выпускной квалификационной работы, было разработано *Android* приложение по контролю личных расходов, которое соблюдает современные стандарты разработки, позволяет пользователю удалять и добавлять расходы, создавать и изменять категории затрат, предлагает общий графический анализ всех расходов по категориям, а также устанавливать и отслеживать плановые покупки.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.12.001 ПЗ

Лист

121

Спасибо за внимание!

					ПГУ1.090301.12.001 ПЗ	Лист
						122
Изм.	Лист	№ докум.	Подпись	Дата		