МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра Прикладная математика и информатика
(наименование)
02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки / специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка мобильного приложения для ведения бюджета»

Обучающийся И.И. Карев _____
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., доцент, О.В. Аникина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент, А.В. Егорова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Тема выпускной квалификационной работы: «Разработка мобильного приложения для ведения бюджета».

Работа содержит 50 страниц, включает 35 рисунков, 4 таблиц и 20 источников.

Целью данной работы является проектирование и создание мобильного приложения, предназначенного для учёта личных финансов. Данное приложение обеспечит пользователям возможность отслеживать свои доходы и расходы.

В введении обоснована актуальность выбранной темы, которая заключается в необходимости разработки мобильного приложения для компании, где проводилась работа для ВКР.

В первой главе проводится изучение предметной области, анализируется организация, на базе которой выполнялась выпускная квалификационная работа, а также определяются условия для разрабатываемого приложения. Кроме того, на данном этапе с помощью CASE-средств была построена UML-диаграмма, отражающая возможные функции взаимодействия пользователей с приложением.

Во второй главе проводятся этапы проектирования, выбор подходящей среды разработки и языка программирования, а также проектирование интерфейса. Здесь была разработана схема переходов между экранами и диаграмма классов, также создан макет приложения.

Третья глава описывает реализацию приложения, разработку основных экранов, интеграцию библиотек, тестирование функционала. Проведено тестирование, подтвердившее корректность работы всех модулей.

Заключение содержит выводы о проделанной работе, достигнутых результатах и возможных направлениях дальнейшего развития приложения. Основным результатом работы стало готовое мобильное приложение, позволяющее пользователям вести свой бюджет.

Abstract

The title of the graduation work is «Development of a Mobile Application for Budget Management».

The graduation work consists of an introduction, three chapters, 35 figures, 4 tables, a conclusion, and a list of 20 references including foreign sources.

The aim of this graduation work is to design and implement a mobile application for personal finance tracking.

The object of the graduation work is the development process of a budget management application.

The subject of the graduation work is the analysis, design, and technical implementation of the mobile application, including its architecture, user interface, and functional modules.

The first part describes in details the domain analysis, examines the company's business requirements, and formulates technical specifications for the application. We examine current market solutions and develop use case diagrams using modern CASE tools.

The second part outlines the design process, including the selection of Android Studio development environment and Kotlin programming language. We present screen flow diagrams, class diagrams, and interactive application prototypes.

The third part consists of practical implementation, covering core functionality development, integration of financial calculation libraries, and comprehensive testing procedures.

In conclusion we'd like to stress based on the results of testing, we developed a fully functional application that successfully meets all specified requirements.

The work is of interest for wide circle of readers interested in mobile development, personal finance management.

Оглавление

Введение

Современные технологии стремительно меняют подходы к управлению личными финансами. Мобильные приложения предлагают автоматизированные решения с аналитикой, напоминаниями и визуализацией данных.

Актуальность данной работы обусловлена растущим спросом на инструменты для персонального финансового учета, однако данное приложение будет интегрировано в социальную сеть, которая разрабатывается в «ВорлдИнтерТех РУС». Разработанное приложение призвано решить эти проблемы, предоставив удобный и интуитивно понятный инструмент для управления финансами.

Объектом исследования является процесс разработки мобильного приложения для ведения бюджета.

Предмет исследования – методы и технологии, применяемые при создании финансовых приложений для платформы Android.

Проект направлен на разработку мобильного приложения для ведения бюджета, пользователи смогут добавлять в приложение операции доходов и расходов, добавлять цели накопления и просматривать графики по операциям. Для этого требуется сформировать требования к мобильному приложению, спроектировать пользовательский интерфейс с проработкой переходов между экранами, выбрать технологии для создания приложения.

В рамках первой главы работы осуществляется разбор предметной области, включающий характеристику аспектов работы организации «ВорлдИнтерТех РУС». Рассматривается один из её бизнес-процессов, используя инструмент для анализа, выбранный на основе сравнительного анализа различных CASE-средств. Результаты проведенного анализа позволяют сформулировать функциональные требования к разрабатываемому приложению и показать функции взаимодействия пользователей с мобильным приложением через диаграмму вариантов

использования.

Во второй главе работы определяется основное программное обеспечение, которое будет использоваться для разработки, и выбирается язык программирования, необходимый для реализации проекта. Проводится так же проектирование мобильного приложения, включая создание схемы переходов между экранами приложения, макета приложения и диаграммы классов, которые структурируют общую архитектуру приложения и позволяют обеспечить его функциональность и удобство использования

Третья глава содержит подробное описание процесса создания приложения. В ней описываются этапы настройки проекта, реализация интерфейса и классов приложения, а также проводимое тестирование. Этот этап является ключевым для обеспечения качественной работы приложения и выявления возможных ошибок до его запуска.

Глава 1 Исследование предметной области

1.1 Характеристика компании и её деятельности

Сфера деятельности компании, охватывает разработку программных продуктов и информационных платформ, предназначенных для интеграции в электронную коммерцию. Программное обеспечение, создаваемое организацией «ВорлдИнтерТех РУС», обеспечивает связь между предприятиями и потребителями, предлагая широкий спектр услуг, включая консалтинг, продвижение и поддержку в областях оптовой и розничной торговли.

Разработчик программного обеспечения играет ключевую роль в компании, занимаясь созданием программных решений на основе предоставленных технических спецификаций. Выпускная квалификационная работа была выполнена с использованием материалов от компании, где главной задачей стало изучение потребностей клиентов, составление технических заданий и разработка программного обеспечения, соответствующего согласованному плану.

Организационная структура «ВорлдИнтерТех РУС» представлена на рисунке 1:

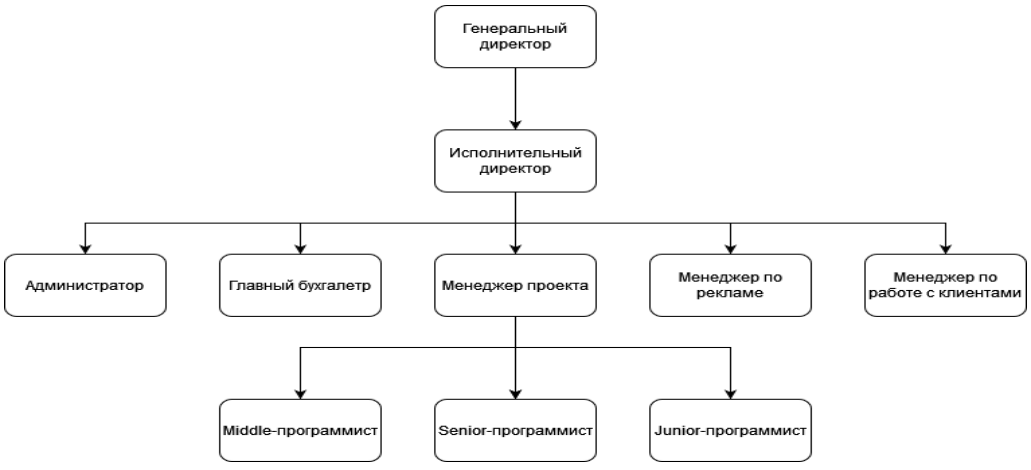


Рисунок 1 – Организационная структура «ВорлдИнтерТех РУС»

В подразделении отдела разработки — входят следующие сотрудники:

- менеджер проекта;
- senior-программист;
- middle-программист;
- junior-программист.

Функции сотрудников представлены на рисунке 2.

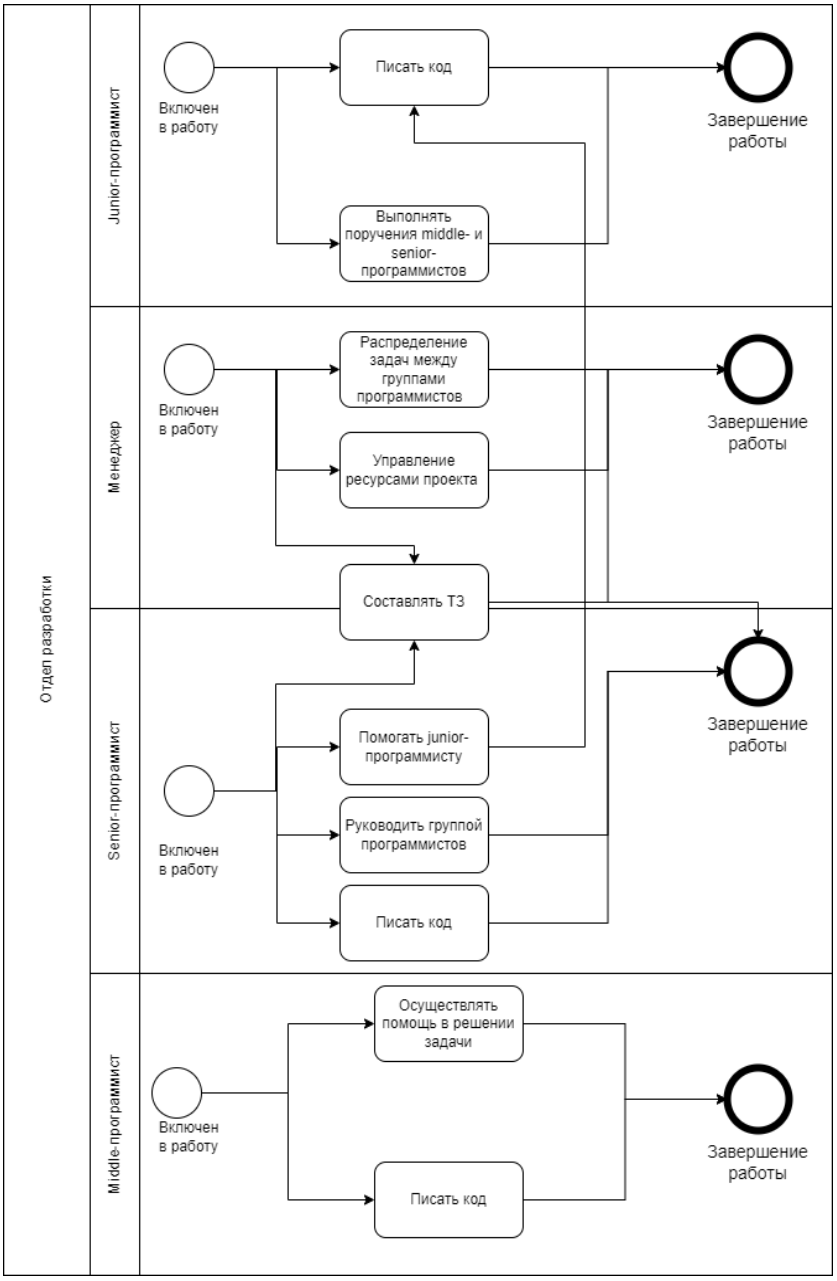


Рисунок 2 – Функции, выполняемые сотрудниками отдела разработок

Рассмотрим роли и функции сотрудников отдела разработки, которые представлены на рисунке 2.

Junior-программист – это начинающий специалист, чьи задачи ограничены недостатком опыта и знаний в области разработки ПО. В основном он занимается выполнением небольших задач, связанных с поддержкой и улучшением существующих систем, а также разработкой и запуском тестов для программных продуктов. Порой ему доверяются более сложные задания, что служит отличной возможностью для повышения квалификации и развития навыков. Однако часто он нуждается в наставничестве более опытных коллег, что помогает ему успешно осваивать профессию.

Middle-программист – это специалист, обладающий достаточным опытом и способный самостоятельно выполнять задачи средней и повышенной сложности. Он не только работает над своими проектами, но и оказывает помощь младшим коллегам, делясь своими знаниями и решая возникающие у них технические проблемы.

Senior-программист – это эксперт в своей области, обладая глубокими знаниями и значительным практическим опытом. В его задачи входит не только исправление сложных ошибок, но и разработка новых программных решений, а также проверка кода, написанного менее опытными программистами.

Менеджер проекта – это специалист, который отвечает за управление ресурсами проекта, определение сроков выполнения задач и составление технических заданий в сотрудничестве с senior-программистом. Он также активно взаимодействует с клиентами для уточнения их требований и согласования этапов разработки, что является важной составляющей успешной работы всей команды.

1.2 Сравнительный анализ case-инструментов для моделирования бизнес-процессов

Бизнес-процесс – это упорядоченная совокупность операций, направленных на решение определенной цели или результата. Это позволяет улучшить различные характеристики и повысить эффективность предприятия.

Существует несколько методов описания бизнес-процессов. Среди них текстовый способ, который включает детальное словесное описание каждого этапа, табличный способ, где информация структурирована в виде таблицы или матрицы и графический подход, использующий схемы и диаграммы для визуализации процессов. В последние годы особую популярность приобрел именно графический метод, так как он помогает наглядно представить информацию и облегчает ее восприятие.

В последние годы наибольшее распространение в разработке получил графический метод, так как он позволяет наглядно представить информацию и облегчить её восприятие [10].

CASE-средства (Computer Aided System/Software Engineering) — это инструменты для проектирования программного обеспечения, обеспечивающее его надёжность, минимальное количество ошибок и простоту поддержи. Рассмотрим некоторые из CASE-средств.

Ramus 2.0 поддерживает методологии IDEF0 и DFD предлагая инструменты для графического моделирования, анализа функций и генерации отчётов [6], [9].

Microsoft Visio универсальный редактор диаграмм, используемый для создания схем бизнес-процессов, сетевых топологий и баз данных. Интегрируется с Excel и Access, поддерживает совместную работу и включает стандартные офисные функции [1].

Draw.io бесплатное веб-приложение, с возможностью установки на компьютер и с обширной библиотекой шаблонов. Экспортирует диаграммы в

форматы JPG, PNG, SVG. Отличается простым интерфейсом и разнообразием шаблонов.

Для проведения сравнения описанных CASE-средств, используем критерии, описанные в таблице. Критерии помогут определиться с самым универсальным и удобным инструментом. Результаты анализа представлены в таблице 1.

Таблица 1 – Сравнительный анализ CASE-средств

Критерий	Ramus 2.0	Microsoft Visio	Draw.io
Доступность	+	-	+
Функциональные возможности	+	+	+
Удобство интерфейса	-	+	+
Поддержка экспорта и импорта в различные форматы	+	+	+
Разнообразие методологий	-	-	+

На основе анализа таблицы, выбор остановился на программном приложении Draw.io, так как он соответствует всем критериям из таблицы, а также по причине наличия опыта работы в данном приложении.

1.3 Определение функциональных требований к приложению

Любое современное приложение на этапе разработки должно соответствовать определенному набору критериев. Эти требования помогают четко определить ключевые характеристики продукта и гарантировать, что итоговый результат оправдает ожидания целевой аудитории.

Существует два требования к разрабатываемому приложению. Определим функциональные требования.

Функциональные требования – это требования к поведению системы,

данная категория определяет конкретные действия, которые должно выполнять приложение. Они описывают основные возможности системы и способы их реализации [2].

На основании анализа существующих решений были сформулированы следующие функциональные требования для приложения учета бюджета [11]:

- реализация системы регистрации и авторизации пользователя;
- функционал для ввода и обработки финансовых данных;
- просмотр графиков на основе финансовых операций;
- просмотр инструкции по работе приложения;
- создание и завершение целей накопления.

Также необходимо определить нефункциональные требования.

Нефункциональные требования описывают, как должна работать система, то есть её поведение и характеристики.

Рассмотрим основные нефункциональные требования, которые предъявляются к данному приложению:

- стабильность работы приложения;
- возможность добавления новых функций в приложение;
- минимальное время отклика интерфейса;
- минималистичный и интуитивный интерфейс.

Следуя этим требованиям, сможем создать приложение, соответствующее всем установленным стандартам и классификациям

Для наглядного представления взаимодействия пользователя с системой была разработана диаграмма вариантов использования с помощью инструмента Draw.io. Эта визуализация помогает понять возможные варианты использования приложения. На рисунке 3 указана диаграмма вариантов использования.

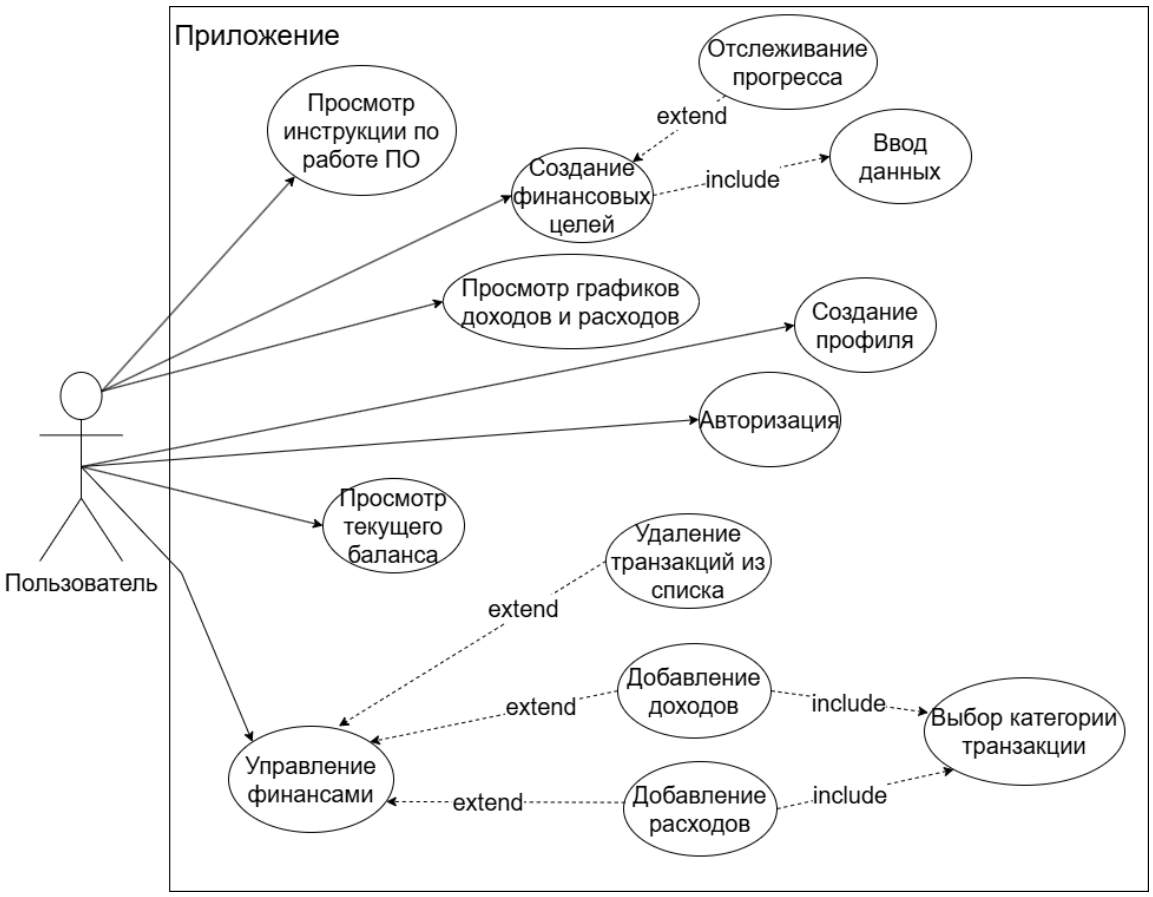


Рисунок 3 – Диаграмма вариантов использования

Пользователь должен иметь возможность авторизовываться, добавлять финансовые операции и удалять их, просматривать графики на основе финансовый операций, просматривать инструкцию по работе приложения, добавлять и редактировать финансовые цели.

Выводы по главе 1

В первой главе ВКР, была разобрана деятельность и структура компании «ВорлдИнтерТех РУС». Выбрано CASE-средство для описания и построения бизнес-процессов. Был выдвинут список функциональных и нефункциональных требований к приложению. По списку требований была построена диаграмма вариантов использования.

Глава 2 Разработка архитектуры мобильного приложения

2.1 Обоснование выбора программных инструментов

За последние десять лет мобильные технологии значительно продвинулись вперёд. В настоящее время Android занимает лидирующую позицию среди мобильных операционных систем по всему миру, контролируя около 80% благодаря своей открытости и универсальности [3]. Эта операционная система, отличающаяся способностью работать на устройствах разных брендов, получила широкое распространение [3]. В магазине Google Play доступно более 4 миллионов приложений для Android.

Для разработки под Android существует несколько специализированных сред, каждая со своими особенностями.

Eclipse представляет собой кроссплатформенную IDE [4], [5] с открытым исходным кодом, первоначально созданную для работы с Java, её гибкая модульная система позволяет адаптировать IDE для разных языков программирования путем установки плагинов. Особенно востребован среди Android-разработчиков плагин ADT (Android Development Tools), который предлагает удобный инструментарий для создания мобильных приложений [4], [5].

IntelliJ IDEA, среда отличается умной системой авто дополнения кода, удобным интерфейсом и поддержкой современных систем сборки. Обладает большим количеством расширений и их простотой установки. Хотя изначально она ориентирована на Java, возможность установки дополнительных модулей расширяет ее функциональность [4].

Android Studio – это специализированная среда разработки, созданная специально для работы с Android. Будучи основанной на IntelliJ IDEA, она предоставляет разработчикам полный набор инструментов для создания, отладки и тестирования мобильных приложений [3], [13]. Поддержка современных языков, включая Kotlin, обладает обширной библиотекой

шаблонов и различными версиями системы Android. Среда позволяет добавлять уже созданные другими разработчиками библиотеки, что упрощает разработку приложений, позволяя не тратить время, при наличии готовых решений [4].

Для оценки проведем сравнение по шести критериям, которые указаны в таблице, они помогут выбрать среду для разработки приложения. Ключевыми критериями будут: наличие готовых шаблонов, разработка без скачивания дополнительных плагинов и удобство среды. Все результаты данного анализа представлены в таблице 2, где указаны как достоинства, так и недостатки каждой IDE.

Таблица 2 – Сравнительный анализ средств разработки

Среда разработки	Доступность	Разработка без плагина	Простота настройки	Удобство среды	Наличие готовых шаблонов
Eclipse	+	-	-	-	-
IntelliJ IDEA	-/+	-	+	+	+
Android Studio	+	+	+	+	+

По итогам анализа выбор остановился на Android Studio [13], так как данная среда предоставляет весь нужный набор инструментов для разработки. В ней имеется встроенный эмулятор телефона и продвинутый редактор кода. В отличие от других интегрированных сред разработки, Android Studio не требует установки дополнительных плагинов.

2.2 Анализ языков программирования и выбор оптимального решения

Android Studio [13] поддерживает два языка программирования: Kotlin и Java [18], [20].

Kotlin – это язык программирования, созданный компанией JetBrains в 2016 году. Он отличается кроссплатформенностью и статической типизацией, что делает его надежным инструментом, позволяющим выявлять ошибки на этапе компиляции. Kotlin нацелен на упрощение процесса разработки благодаря простому синтаксису и снижению объема кода. Среди его главных преимуществ стоит выделить полную совместимость с Java, что упрощает переход разработчиков между этими языками. Также ключевыми особенностями Kotlin являются упрощение работы с API [18], [20].

Java, является одним из самых популярных языков программирования в области объектно-ориентированной разработки. Популярность обусловлена множеством факторов, включая кроссплатформенность, обеспечиваемую виртуальной машиной Java (JVM), которая интерпретирует байт-код и позволяет запускать приложения на различных платформах. Кроме того, Java придерживается строгих принципов объектно-ориентированного программирования, предлагает автоматическое управление памятью, встроенную поддержку многопоточности и обширную стандартную библиотеку, что делает его универсальным инструментом для создания широкого спектра приложений [16], [17], [18].

Для удобства проведем сравнение языков программирования, обращая внимание на такие ключевые аспекты, как быстродействие, лаконичность кода, распространенность на рынке и доступность документации. Результаты представлены в таблице 3.

Таблица 3 – Сравнение языков программирования

Язык программирования	Скорость работы	Компактность кода	Популярность на рынке	Доступность документации
Java	+	-	+	+
Kotlin	+	+	-	-

В процессе выбора языка для разработки приложения было решено остановиться на Java, учитывая уже существующий опыт работы с ним. Хотя Kotlin и представляет собой современный язык с множеством интересных возможностей, его относительная новизна создает сложности с поиском документации и примеров, что также повлияло на принятое решение.

2.3 Проектирование пользовательского интерфейса и ux-логики

Проектирование пользовательского интерфейса началось с создания схемы переходов между экранами приложения, которая наглядно демонстрирует все возможные сценарии взаимодействия пользователя с приложением.

Эта схема была разработана в Draw.io и охватывает все экраны приложения: авторизация, главная, графики, помощь и цели. Внимание было уделено переходам между различными состояниями, которые инициируются нажатием кнопок на панели навигации, что обеспечивает удобство работы с приложением.

Детали схемы переходов между экранами приложения можно увидеть на рисунке 4.

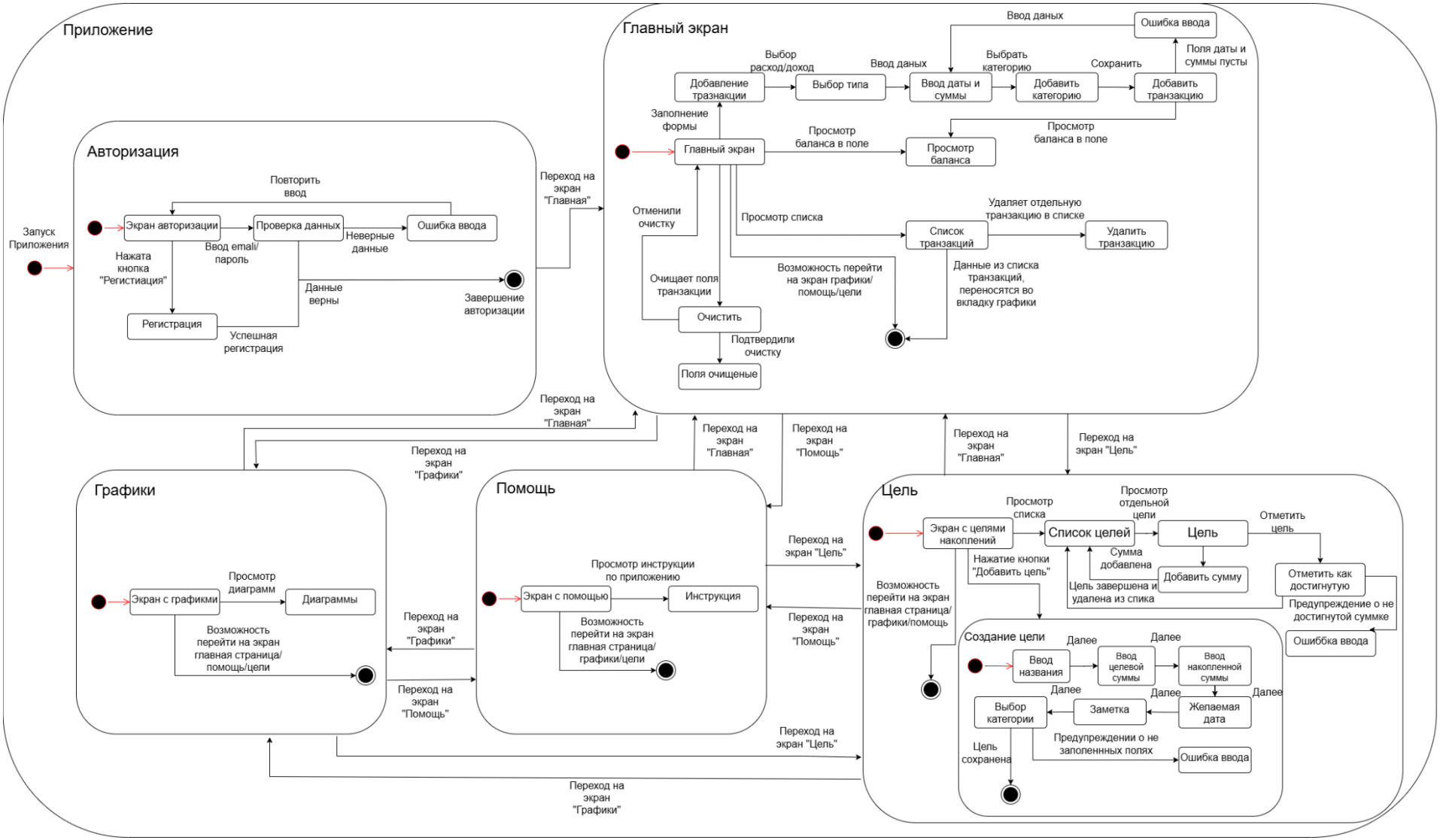


Рисунок 4 – Схема переходов между экранами приложения

Приложение начинается с этапа авторизации, где пользователь может либо зарегистрироваться, либо войти в систему, если он уже зарегистрирован. После успешной авторизации осуществляется переход на главный экран, который является центральной точкой приложения. Здесь реализованы функции добавления, удаления и просмотра операций, отображения текущего баланса, а также переходы на экраны с графиками, целями накоплений и справочной информацией. При добавлении расхода или дохода требуется заполнить форму и указать дату операции, сумму и категорию. Экран «графики» предоставляет визуальное представление данных в виде двух графиков доходов и расходов соответственно, которые будут изменяться в зависимости от списка операций, экран «цели» позволяет создавать, редактировать и отслеживать прогресс достижения финансовых целей, а раздел «помощь» содержит инструкции по использованию приложения.

Все экраны приложения связаны между собой системой навигации [19], что обеспечивает переходы и всегда оставляет пользователю возможность переходить между экранами приложения.

Рассмотрев данную диаграмму, можно определить какие экраны требуются приложению, кроме того, можно создать вспомогательные экраны для удобного выбора категории и даты операции. Так при заполнении формы операции, стоит реализовать удобный выбор даты через календарь, а также удобно выбирать категорию через список.

Подводя итог описания схемы переходов между экранами приложения, можно построить следующий макет экранных форм приложения [14]. На макете должны находится основные функции и элементы приложения, а также их краткое описание. Так же стрелочками можно указать основные всплывающие окна приложения. Макет можно рассмотреть на рисунке 5.

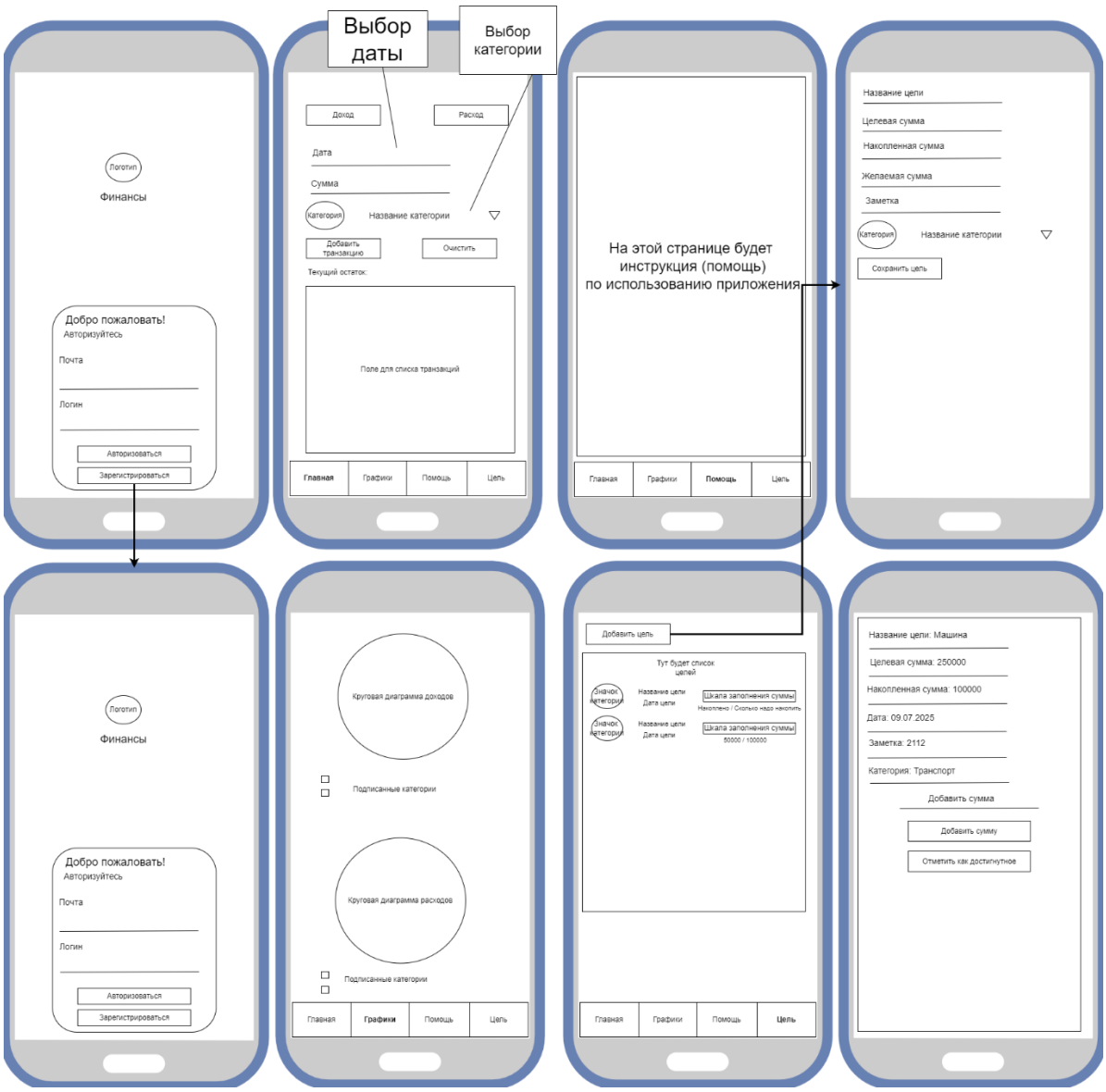


Рисунок 5 – Макет приложения

На этом рисунке представлены восемь ключевых экранов приложения. Анализируя эти визуальные формы, можно выделить несколько решений по дизайну интерфейса, которые будут реализованы в проекте. В нижней части экрана планируется разместить панель навигации, благодаря которой пользователи смогут без проблем переключаться между четырьмя основными экранами. Также предусмотрено наличие всплывающего окна для отметки цели, а также два небольших окна для выбора даты и категории.

2.4 Подбор технологий для реализации функциональных задач

Использована архитектура MVC – шаблон проектирования, разделяющий приложение на три компонента: Model (данные через SQLite и SharedPreferences), View (интерфейс через XML и Activity) и Controller (логика обработка действий в Activity). Это упрощает поддержку кода, так как каждый компонент отвечает только за свою задачу: данные, отображение и управление.

Для разработки мобильного приложения, которое поможет пользователям эффективно управлять своими финансами, выбрана среда Android Studio и язык программирования Java. В приложении будет использоваться ряд библиотек и компонентов, которые не только обеспечат его функциональность, но и улучшат взаимодействие с пользователем.

Для данного проекта, предложенного компанией, было принято решение применить более простой вариант, хранение данных в SQLite и SharedPreferences с использованием JSON-сериализации.

Логин и пароль пользователя хранится в SQLite, это встроенная база данных в среде разработки Android Studio, с таблицей «users», где почта является первичным ключом, а пароль хранится в открытом виде.

Данные приложения хранятся с использованием SharedPreferences, что представляет собой удобный механизм для работы с парами «ключ-значение». Финансовые цели, отображаемые в виде объектов класса Goal, конвертируются в формат JSON с помощью библиотеки Gson и сохраняются в отдельном файле SharedPreferences под ключом goalList. Это позволяет эффективно сохранять разнообразные данные, такие как название цели, планируемая и текущая суммы, дата, заметка и категория. Информация о операциях и текущем балансе сохраняется в другом файле SharedPreferences. Выбранный подход обеспечивает простоту разработки и достаточную скорость работы приложения.

Для создания приложения были использованы современные Android-

технологии. Пользовательский интерфейс разработан на основе AndroidX и Material Design, что обеспечивает актуальный внешний вид и кроссплатформенную совместимость.

Для визуализации статистики по расходам и доходам в приложении была выбрана библиотека MPAndroidChart. Это средство позволяет создавать различные графики, такие как круговые графики (PieChart), что дает возможность пользователям наглядно увидеть свои доходы и расходы. Для настройки диаграмм использовались стандартные параметры отображения, включая цвета, легенды и анимацию.

Для выбора даты операции пользователи смогут воспользоваться стандартным календарем, реализованным через DatePickerDialog. Это диалоговое окно упрощает процесс выбора даты, делая приложение более удобным и понятным.

Для подтверждения важных действий, таких как удаление операции или очистка баланса, применяется AlertDialog. Этот элемент интерфейса предлагает пользователю диалоговые окна с кнопками для подтверждения или отмены, что снижает риск случайного удаления данных.

В состав Android SDK входят все необходимые библиотеки для создания виджетов, которые отображают информацию, реагируют на действия пользователей и помогают в управлении интерфейсом приложения. Ключевыми элементами для разработки интерфейса являются кнопки, поля заполнения, различные индикаторы, всплывающие окна и многие другие, которые в совокупности позволяют создать понятный и приятный интерфейс.

Эти компоненты служат для создания интерфейсов, управления взаимодействием с пользователями и передачи данных между различными экранами приложения.

Архитектура мобильного приложения была спроектирована, и следующим шагом является представление диаграммы классов [15]. Диаграмма поможет лучше понять структуру приложения и показать основные классы приложения. Диаграмма классов для мобильного

приложения представлена на рисунке 6.

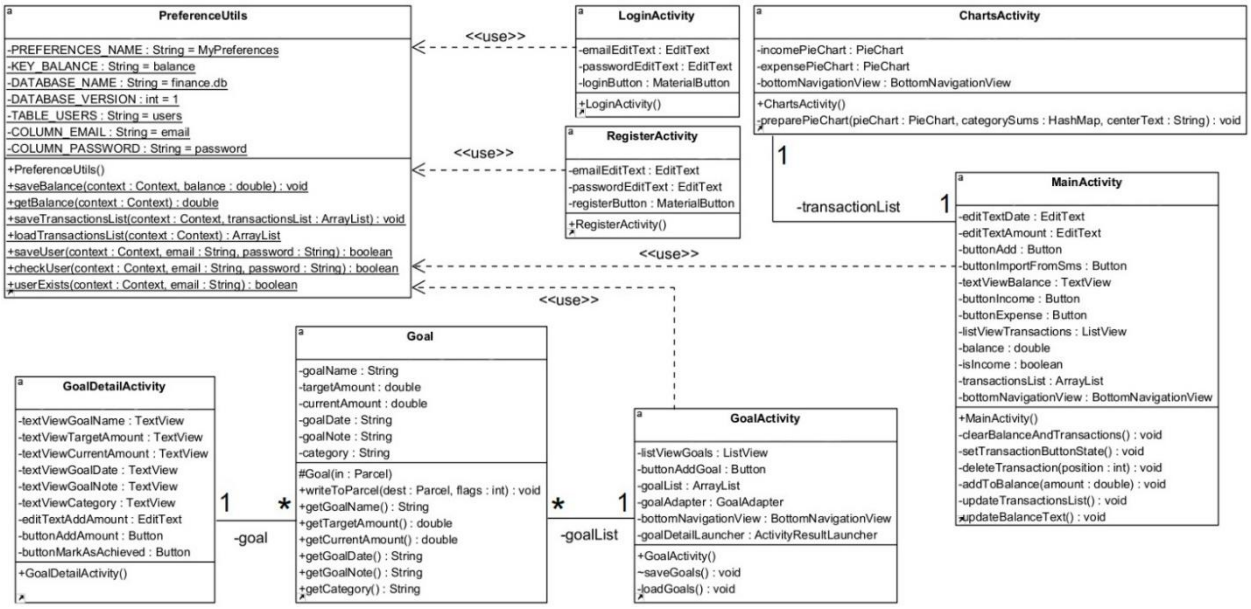


Рисунок 6 – Диаграмма классов

Выводы по главе 2

Во второй главе было обоснованно выбрано использование Android Studio в качестве основной среды разработки, а Java стал основным языком программирования. Здесь также разработана схема переходов между экранами, которая визуализирует логику взаимодействия пользователя с приложением, и макет экранных форм, определяющий ключевые компоненты архитектуры. К таким компонентам относятся хранение данных с помощью SQLite, SharedPreferences и JSON-сериализации, а также визуализация статистики с применением MPAndroidChart. Кроме того, построена диаграмма классов, которая демонстрирует ключевые классы и их взаимосвязи.

Глава 3 Реализация и тестирование мобильного приложения

3.1 Разработка мобильного приложения

Для начала создадим проект в Android Studio. После установки самой среды разработки переходим во окно создания проектов и выбираем «No Activity» (рисунок 7), данный проект будет обладать только конфигурационными файлами, остальную файловую структура придётся создавать самому, далее заполняем данные нашего приложения, выбирая язык программирования Java и конфигурацию Gradle, нажимаем «Finish» (рисунок 8). После этого создастся проект нашего приложения.

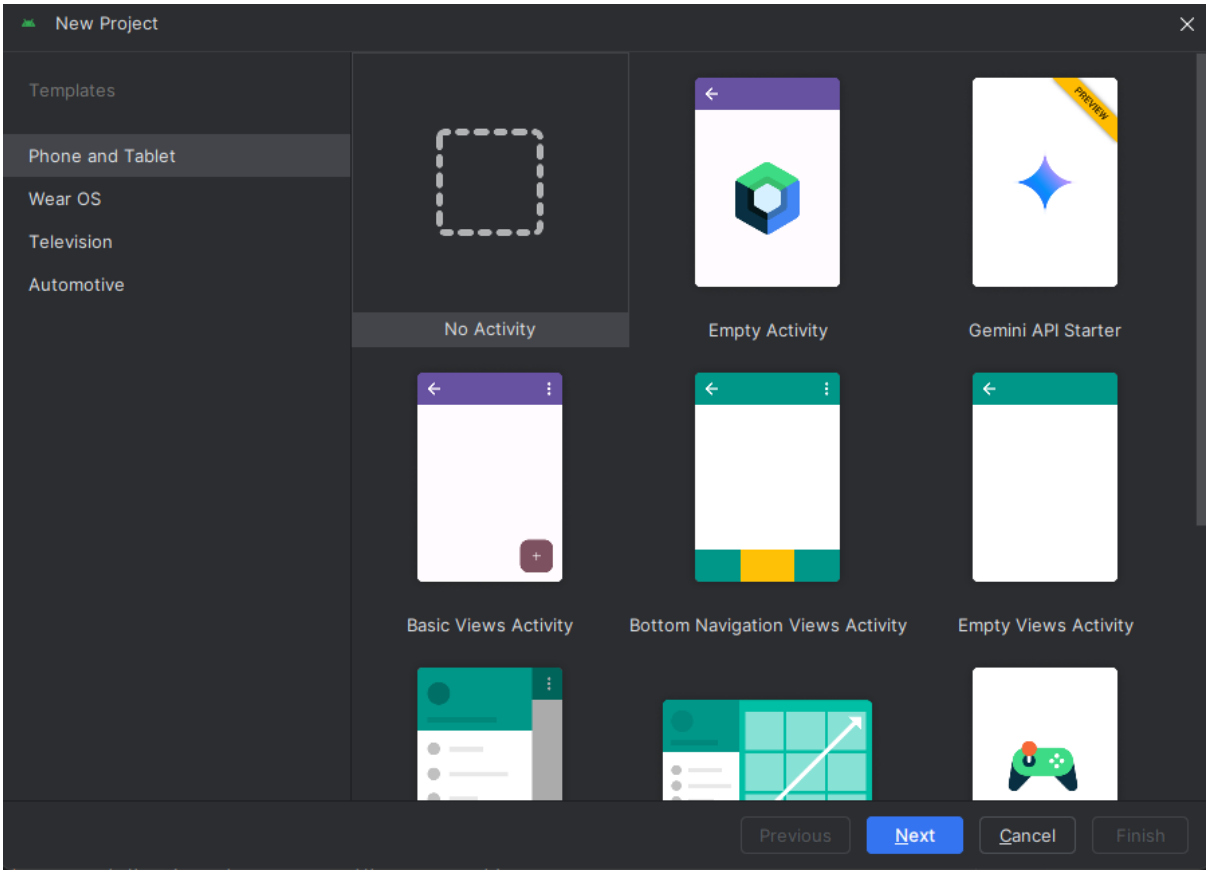


Рисунок 7 – Выбор главного экрана приложения

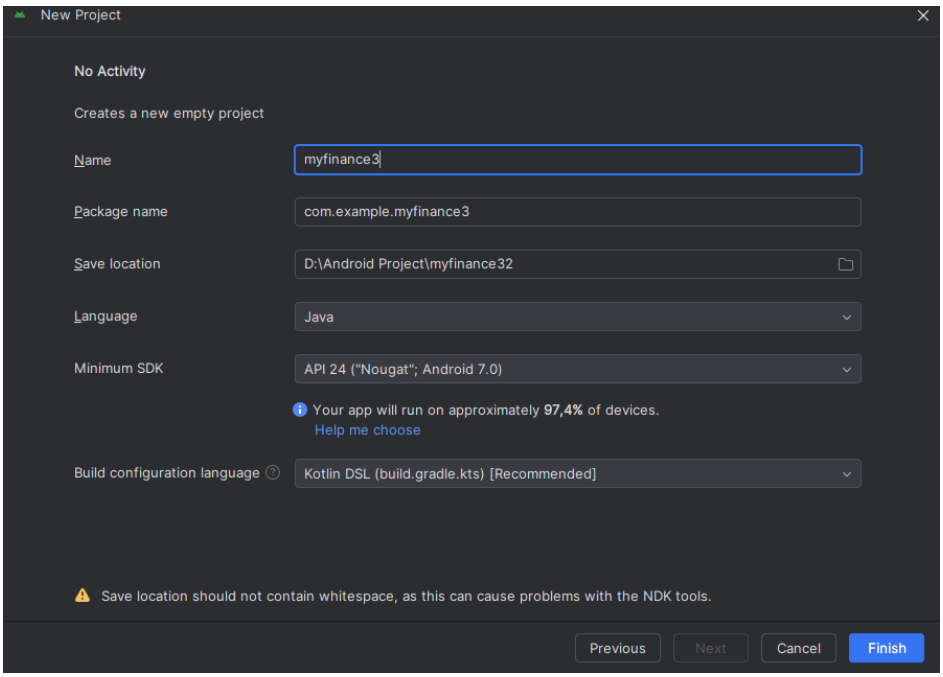


Рисунок 8 – Заполнение данных приложения

Для работы с проектом необходимо добавить соответствующие библиотеки в зависимости. Все они указаны в файле «build.gradle.kts», который является конфигурационным файлом проекта в Android Studio. Этот файл использует систему сборки Gradle, позволяя настраивать зависимости, подключать необходимые плагины, устанавливать параметры сборки (например, версии компилятора и целевых платформ), создавать пользовательские задачи, а также задавать репозитории для загрузки зависимостей и устанавливать метаданные проекта (рисунок 9).

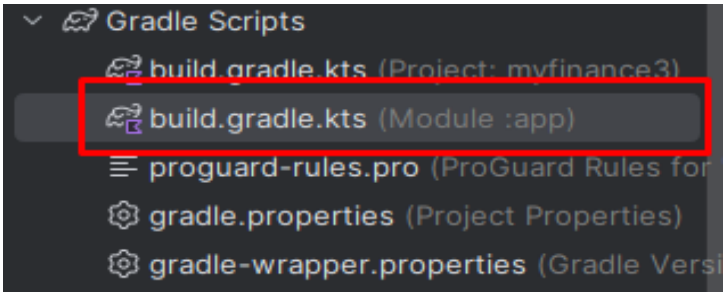


Рисунок 9 – Конфигурационный файл gradle

Основные зависимости будут составлять библиотеки для работы с компонентами графиков, хранение и передача данных (рисунок 10).

```
dependencies {
    implementation ("com.google.code.gson:gson:2.8.8")
    implementation ("com.github.PhilJay:MPAndroidChart:v3.1.0")
    implementation ("com.google.android.material:material:1.4.0")
    implementation(libs.appcompat)
    implementation(libs.material)
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
}
```

Рисунок 10 – Зависимости приложения

В файле AndroidManifest.xml добавим разрешения приложению читать и записывать файлы на внешнее хранилище устройства и пропишем активности для будущих страниц в приложении. Разрешения и активности приведены на рисунке 11.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.Myfinance3"
    tools:targetApi="31">

    <activity android:name=".LoginActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <activity android:name=".MainActivity" />
    <activity android:name=".ChartsActivity"/>
    <activity android:name=".HelpActivity"/>
    <activity android:name=".GoalActivity"/>
    <activity android:name=".AddGoalActivity" />
    <activity android:name=".RegisterActivity" />
    <activity android:name=".GoalDetailActivity" />
```

Рисунок 11 – Разрешения приложения

Для проекта мне понадобится шрифт, картинка для фона приложения и картинки значков для категорий операций и целей. Для этого в Android Studio имеются готовые фоны. Дал им определённые названия, которые далее будут использоваться в проекте. Картинки для значков и фона будут храниться в папке drawable, которая хранит графические ресурсы. Шрифт я добавил в папку font, которая содержит добавленные шрифты, структуру можно наблюдать на рисунке 12.

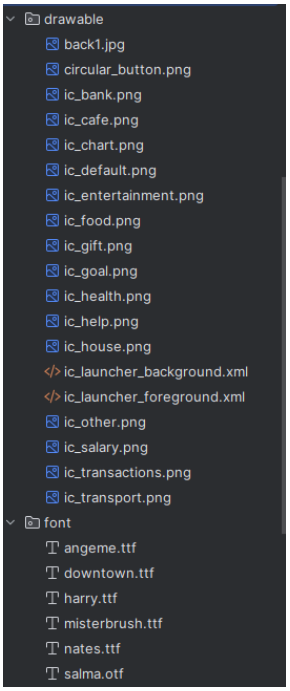


Рисунок 12 – Добавленные ресурсы для проекта

Далее я поменял дизайн приложения, в файле themes.xml можно выбрать цвет или вставить картинку для заднего фона приложения, прописать цвет кнопок и общей темы приложения. Изменённую тему можно наблюдать на рисунке 13.

```xml
<resources xmlns:tools="http://schemas.android.com/tools">
    <style name="Theme.Myfinance3" parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <item name="colorPrimary">@color/blue</item>
        <item name="colorPrimaryVariant">@color/blue</item>
        <item name="colorOnPrimary">@color/white</item>
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <item name="android:statusBarColor">?attr/colorPrimaryVariant</item>
        <item name="android:windowBackground">@drawable/back1</item>
    </style>
</resources>
```

Рисунок 13 – Изменение темы приложения

После создал файл bottom navigation menu в папке menu, который будет содержать виджеты и разметку для навигационной панели. Реализованную навигацию можно наблюдать на рисунке 14.

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/nav_home"
        android:icon="@drawable/ic_transactions"
        android:title="Главная" />
    <item
        android:id="@+id/nav_chart"
        android:icon="@drawable/ic_chart"
        android:title="Графики" />
    <item
        android:id="@+id/nav_help"
        android:icon="@drawable/ic_help"
        android:title="Помощь" />
    <item
        android:id="@+id/nav_goal"
        android:icon="@drawable/ic_goal"
        android:title="Цель" />
</menu>
```

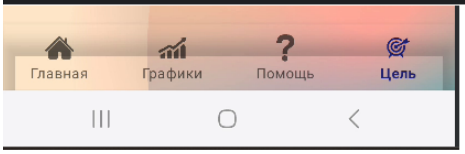


Рисунок 14 – Навигационная панель

Чтобы создать массив категорий, необходимо создать файл array.xml и прописать туда название категории и ссылку на картинку, которая относится к данной категории. Созданные категории можно увидеть на рисунке 15.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="category_names">
        <item>Все</item>
        <item>Кафе</item>
        <item>Транспорт</item>
        <item>Развлечения</item>
        <item>Зарплата</item>
        <item>Вклады банка</item>
        <item>Другое</item>
        <item>Подарки</item>
        <item>Здоровье</item>
        <item>Продукты</item>
        <item>Для дома</item>
    </string-array>
    <array name="category_icons">
        <item>@drawable/ic_default</item>
        <item>@drawable/ic_cafe</item>
        <item>@drawable/ic_transport</item>
        <item>@drawable/ic_entertainment</item>
        <item>@drawable/ic_salary</item>
        <item>@drawable/ic_bank</item>
        <item>@drawable/ic_other</item>
        <item>@drawable/ic_gift</item>
        <item>@drawable/ic_health</item>
        <item>@drawable/ic_food</item>
        <item>@drawable/ic_house</item>
    </array>
</resources>
```

Рисунок 15 – Созданные категории

Так же для работы с цветами в приложении, в папке colors.xml необходим создать цвет и дать ему название, это понадобится для цвета категорий на круговом графике и дизайна приложения, созданные цвета можно увидеть на рисунке 16.

```xml
    <color name="colorPrimary">#4CAF50</color>
    <color name="colorAccent">#F44336</color>
    <color name="light_blue">#ADD8E6</color>
    <color name="blue">#212171</color>
    <color name="peach">#FFDAB9</color>
    <color name="limeGreen">#32CD32</color>
    <color name="gray">#374237</color>

    <color name="colorCafe">#FF6347</color>
    <color name="colorTransport">#32CD32</color>
    <color name="colorEntertainment">#FFD700</color>
    <color name="colorSalary">#8A2BE2</color>
    <color name="colorBank">#DC143C</color>
    <color name="colorOther">#A9A9A9</color>
    <color name="colorGifts">#FF69B4</color>
    <color name="colorHealth">#00BFFF</color>
    <color name="colorFood">#98FB98</color>
    <color name="colorHouse">#D2691E</color>
```

Рисунок 16 – Созданные цвета

Далее в папке layout я создаю файлы xml для описания структуры интерфейса приложения, уже после классы будут реализовывать эти интерфейсы. Созданные интерфейсы можно наблюдать на рисунке 17.

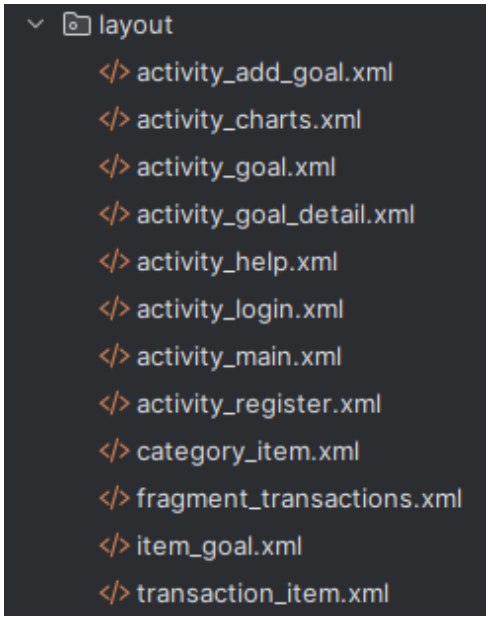


Рисунок 17 – Созданные интерфейсы

Описать данные интерфейсы можно так:

- activity add goal – описывает экран добавления новой цели, он будет содержать поле ввода для названия цели, целевая сумма, накопленная сумма, заметка о цели, выбор категории и кнопка сохранения цели;
- activity charts – содержит отображение круговых графиков доходов и расходов;
- activity goal – описывает экран просмотра и добавления целей;
- activity goal detail – описывает экран с информацией о цели при нажатии на цель, там содержится информация о цели, поле ввода для добавления суммы к текущей накопленной, кнопка добавления суммы и кнопка отметки цели, как достигнутой;
- activity help – описывает экран справки, где будет содержаться текст о информации работы приложения;

– activity login – описывает экран авторизации пользования, с возможностью войти на свой аккаунт или зарегистрироваться;
– activity main – описывает главный экран приложения, тут будет содержаться кнопки с выбором дохода или расхода, поле для ввода даты и суммы, выбор категории, кнопки добавления операции, поле с информацией текущего остатка и список операций;
– activity register – описывает экран регистрации пользователя;
– category item – файл представляет собой представление категории, будет использоваться в адаптерах для отображения списка;
– fragment transactions – отображает список операций, который потом используется в activity main;
– item goal – содержит отображение одной цели в списке, в котором находятся шкалу отображающий прогресс накоплений, значок выбранной категории, отображения целевой и переменной суммы;
– transaction item – отображает операции в списке.

Далее создаём классы, которые реализуют интерфейсы. Это можно увидеть на рисунке 18.

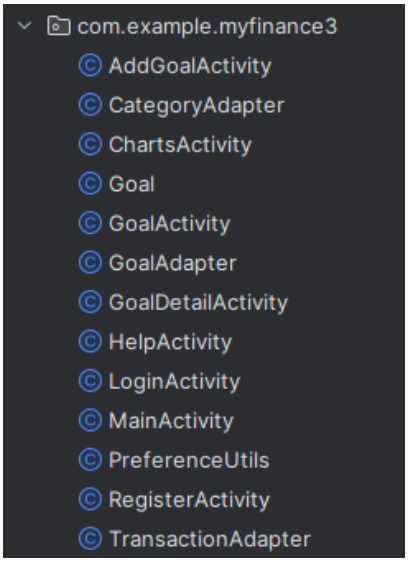


Рисунок 18 – Созданные классы

Описать основные методы созданных классов можно так:

– AddGoalActivity – этот класс предоставляет интерфейс для добавления новой цели, пользователь может ввести название цели, указать сумму, выбрать категорию и добавить описание, метод onCreate инициализирует элементы пользовательского интерфейса, такие как текстовые поля, кнопки и выпадающие списки, настраивает обработчики событий для сохранения новой цели, метод saveGoal сохраняет данные новой цели;
– CategoryAdapter – отображает список категорий для использования в различных частях приложения, включая цели и операции;
– ChartsActivity – отвечает за отображение графиков доходов и расходов с использованием MPAndroidChart, основной метод preparePieChart подготавливает данные для отображения в круговом графике, включая маппинг категорий на цвета и настройку внешнего вида графиков, метод preparePieChart можно наблюдать на рисунке 19;

```
private void preparePieChart(PieChart pieChart, HashMap<String, Float> categorySums, String centerText)
    ArrayList<PieEntry> entries = new ArrayList<>();
    for (Map.Entry<String, Float> entry : categorySums.entrySet()) {
        entries.add(new PieEntry(entry.getValue(), entry.getKey()));
    }

    PieDataSet dataSet = new PieDataSet(entries, label: "");

    HashMap<String, Integer> categoryColors = new HashMap<>();
    categoryColors.put("Кафе", ContextCompat.getColor( context: this, R.color.colorCafe));
    categoryColors.put("Транспорт", ContextCompat.getColor( context: this, R.color.colorTransport));
    categoryColors.put("Развлечения", ContextCompat.getColor( context: this, R.color.colorEntertainment));
    categoryColors.put("Зарплата", ContextCompat.getColor( context: this, R.color.colorSalary));
    categoryColors.put("Вклады банка", ContextCompat.getColor( context: this, R.color.colorBank));
    categoryColors.put("Другое", ContextCompat.getColor( context: this, R.color.colorOther));
    categoryColors.put("Подарки", ContextCompat.getColor( context: this, R.color.colorGifts));
    categoryColors.put("Здоровье", ContextCompat.getColor( context: this, R.color.colorHealth));
    categoryColors.put("Продукты", ContextCompat.getColor( context: this, R.color.colorFood));
    categoryColors.put("Для дома", ContextCompat.getColor( context: this, R.color.colorHouse));

    ArrayList<Integer> colors = new ArrayList<>();
    for (Map.Entry<String, Float> entry : categorySums.entrySet()) {
        String category = entry.getKey();
        Integer color = categoryColors.get(category);
        if (color != null) {
            colors.add(color);
```

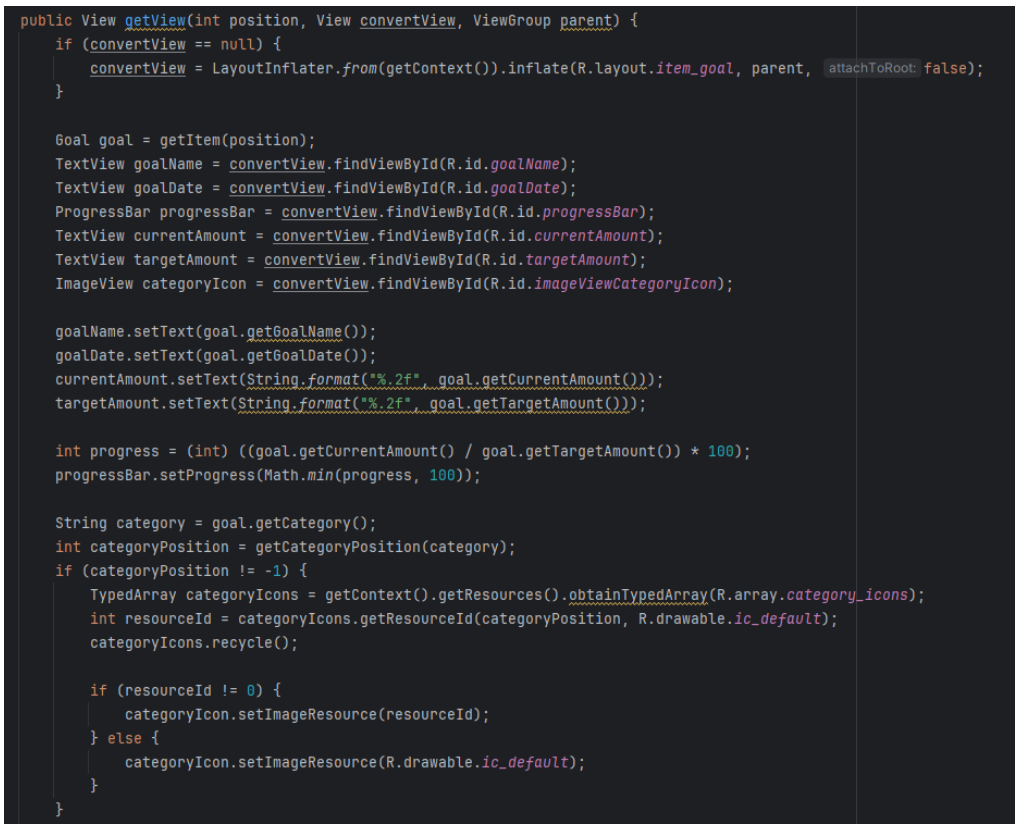
Рисунок 19 – Метод preparePieChart

– Goal – этот класс представляет цель пользователя, включая информацию о целевой и текущей сумме, дате, категории и примечании;
– GoalActivity – является активностью для отображения списка целей пользователя, позволяет добавлять цели накопления и их редактировать, сохранение целей можно увидеть на рисунке 20;

```
void saveGoals() {  4 usages
    SharedPreferences preferences = getSharedPreferences( name: "GoalsPrefs", MODE_PRIVATE);
    SharedPreferences.Editor editor = preferences.edit();
    Gson gson = new Gson();
    String json = gson.toJson(goalList);
    editor.putString( s: "goalList", json);
    editor.apply();
}

private void loadGoals() {  1 usage
    SharedPreferences preferences = getSharedPreferences( name: "GoalsPrefs", MODE_PRIVATE);
    String json = preferences.getString( s: "goalList",  s1: null);

    Gson gson = new Gson();
    Type type = new TypeToken<ArrayList<Goal>>() {}.getType();
    goalList = gson.fromJson(json, type);

    if (goalList == null) {
        goalList = new ArrayList<>();
    }
    goalAdapter.clear();
    goalAdapter.addAll(goalList);
    goalAdapter.notifyDataSetChanged();
```

Рисунок 20 – Метод saveGoals и loadGoals

– GoalAdapter – это адаптер для отображения данных целей в списке целей, он преобразует каждый элемент из списка целей в представление для отображения на экране, реализацию адаптера можно увидеть на рисунке 21;



```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView = LayoutInflater.from(getContext()).inflate(R.layout.item_goal, parent,  attachToRoot: false);
    }

    Goal goal = getItem(position);
    TextView goalName = convertView.findViewById(R.id.goalName);
    TextView goalDate = convertView.findViewById(R.id.goalDate);
    ProgressBar progressBar = convertView.findViewById(R.id.progressBar);
    TextView currentAmount = convertView.findViewById(R.id.currentAmount);
    TextView targetAmount = convertView.findViewById(R.id.targetAmount);
    ImageView categoryIcon = convertView.findViewById(R.id.imageViewCategoryIcon);

    goalName.setText(goal.getGoalName());
    goalDate.setText(goal.getGoalDate());
    currentAmount.setText(String.format("%.2f", goal.getCurrentAmount()));
    targetAmount.setText(String.format("%.2f", goal.getTargetAmount()));

    int progress = (int) ((goal.getCurrentAmount() / goal.getTargetAmount()) * 100);
    progressBar.setProgress(Math.min(progress, 100));

    String category = goal.getCategory();
    int categoryPosition = getCategoryPosition(category);
    if (categoryPosition != -1) {
        TypedArray categoryIcons = getContext().getResources().obtainTypedArray(R.array.category_icons);
        int resourceId = categoryIcons.getResourceId(categoryPosition, R.drawable.ic_default);
        categoryIcons.recycle();

        if (resourceId != 0) {
            categoryIcon.setImageResource(resourceId);
        } else {
            categoryIcon.setImageResource(R.drawable.ic_default);
        }
    }
```

Рисунок 21– Реализация GoalAdapter

– GoalDetailActivity – класс представляет экран информации о выбранной цели;
– HelpActivity – класс представляет экран справки, где пользователь может ознакомиться с функциональностью приложения;
– LoginActivity – стартовый экран авторизации, проверяет введенные email и пароль с данными из SQLite, при успехе переводит в MainActivity;
– MainActivity – является основным экраном приложения, которое обеспечивает пользовательский интерфейс для управления балансом, доходами и расходами, а также взаимодействие с операциями, для управления балансом создан метод addToBalance, который обновляет текущий баланс в зависимости от типа операции, метод updateTransactionsList обновляет адаптер списка операций, добавляя новые операции;

– PreferenceUtils – этот класс представляет собой утилиту для работы с сохранением и загрузкой данных в SharedPreferences и внутренний класс для работы с SQLite, он используется для сохранения текущего баланса, списка операций и целей, а также сохранение логина и пароля пользователей, чтобы данные оставались доступны между сеансами приложения, реализованный класс можно увидеть на рисунке 22;

```
public class PreferenceUtils {  11 usages

    private static final String PREFERENCES_NAME = "MyPreferences";  2 usages
    private static final String KEY_BALANCE = "balance";  2 usages
    // SQLite
    private static final String DATABASE_NAME = "finance.db";  1 usage
    private static final int DATABASE_VERSION = 1;  1 usage
    private static final String TABLE_USERS = "users";  5 usages
    private static final String COLUMN_EMAIL = "email";  6 usages
    private static final String COLUMN_PASSWORD = "password";  3 usages

    // Сохранение баланса в SharedPreferences
    public static void saveBalance(Context context, double balance) {  3 usages
        SharedPreferences preferences = context.getSharedPreferences(PREFERENCES_NAME, Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = preferences.edit();
        editor.putFloat(KEY_BALANCE, (float) balance);
        editor.apply();
    }

    // Загрузка баланса из SharedPreferences
    public static double getBalance(Context context) {  1 usage
        SharedPreferences preferences = context.getSharedPreferences(PREFERENCES_NAME, Context.MODE_PRIVATE);
        return preferences.getFloat(KEY_BALANCE,  v: 0.0f);
    }

    // Сохранение списка транзакций в SharedPreferences
    public static void saveTransactionsList(Context context, ArrayList<String> transactionsList) {  2 usages
        SharedPreferences preferences = context.getSharedPreferences( s: "MyPreferences", Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = preferences.edit();
        editor.putString( s: "transactionsList", TextUtils.join( delimiter: ";;", transactionsList));
        editor.apply();
    }

    // Загрузка списка транзакций из SharedPreferences
    public static ArrayList<String> loadTransactionsList(Context context) {  2 usages
        SharedPreferences preferences = context.getSharedPreferences( s: "MyPreferences", Context.MODE_PRIVATE);
        String savedList = preferences.getString( s: "transactionsList",  s1: "");
        if (!savedList.isEmpty()) {
            return new ArrayList<>(Arrays.asList(savedList.split( regex: ";;")));
        }
        return new ArrayList<>();
```

Рисунок 22 – Реализация PreferenceUtils

– RegisterActivity – экран регистрации, сохраняет введенные email и пароль в SQLite, после успешной регистрации возвращает на LoginActivity для входа;
– TransactionAdapter – этот класс представляет адаптер для работы с отображением списка операций в интерфейсе.

3.2 Методология тестирования и оценка работоспособности

Будет проведено тестирование функциональности [7] приложения. Целью является сопоставить фактические результаты с ожидаемыми. В этом процессе внимание будет уделено обработке финансовых данных, что является важным в работе данного приложения. Корректность операций влияет на удобство использования системы. В частности, при добавлении новой операции система должна правильно обрабатывать введенные данные (сумму, категорию, дату), сохранять их в корректном формате и отображать оперативно.

Необходимо уделить внимание проверке валидации всех полей, чтобы гарантировать их заполнение. При вводе неверных данных приложение должно выводить соответствующие предупреждения.

Важно убедиться, что информация из списка операций корректно передаётся и отображается на графиках. Графики расходов и доходов должны обновляться в реальном времени, чтобы всегда показывать актуальный список операций. Также стоит проверить, правильно ли данные отображаются по категориям на графике.

Следует протестировать навигацию между экранами, убедившись, что переходы между разделами приложения происходят плавно и без сбоев. Пользователь должен иметь возможность перемещаться между всеми четырьмя экранами: главная, графики, помощь и цели, всё это должно происходить без задержек или проблем с отображением контента.

Важно проверить функциональность добавления и закрытия целей накопления. Приложение должно корректно отображать цели, и при добавлении суммы к цели должен автоматически обновляться шкала прогрессии. Когда нужная сумма будет накоплена, приложение должно правильно закрыть цель и удалить её из списка.

Результаты функционального тестирования приведены в таблице 4.

Таблица 4 – Результаты тестирования

Действие	Ожидаемый результат	Действительный результат
Добавление операции	- Пересчет баланса - Правильное отображение категории - Корректный выбор даты - Сохранение в журнале операций	Соответствует
Добавление цели	- Валидация обязательных полей - Корректное сохранение - Отображение в списке целей	Соответствует
Работа с графиками	- Построение круговых графиков по категориям - Корректное отображение пропорций доходов/расходов	Соответствует
Навигация	- Переходы между разделами с анимацией - Сохранение состояния при возврате	Соответствует
Авторизация пользователя	- Корректная обработка введенных данных - Переход в главное меню	Соответствует
Восстановление данных	- Сохранение информации между сеансами работы	Соответствует

После проведения тестирования мобильного приложения можно заключить, что все предусмотренные функции работают без сбоев. Это подтверждает, что приложение правильно выполняет все возможные действия, предусмотренные для использования.

Приложение работает корректно и реализовывает весь функционал (Рисунки 23 - 33).

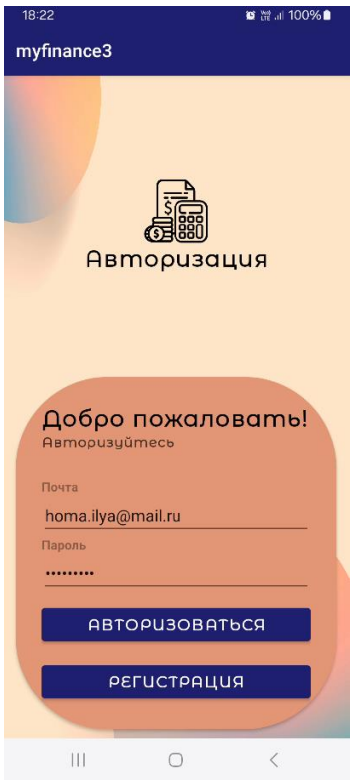


Рисунок 23 – Страница авторизации

Рисунок 24 – Страница регистрации

На рисунке 25 изображена главная страница приложения. Вверху экрана отображается текущий лимит на месяц, который можно установить, при добавлении операции выбираем тип операции доход и расход, указываем сумму дату и категорию. Кнопкой очистить можно удалить все операции. Внизу показан текущий остаток и лимит на месяц с шкалой заполнения. Также есть список операций с типом, категорией, датой и суммой, которые можно отдельно удалять. В самом низу присутствует навигационная панель, для перехода между экранами приложения.

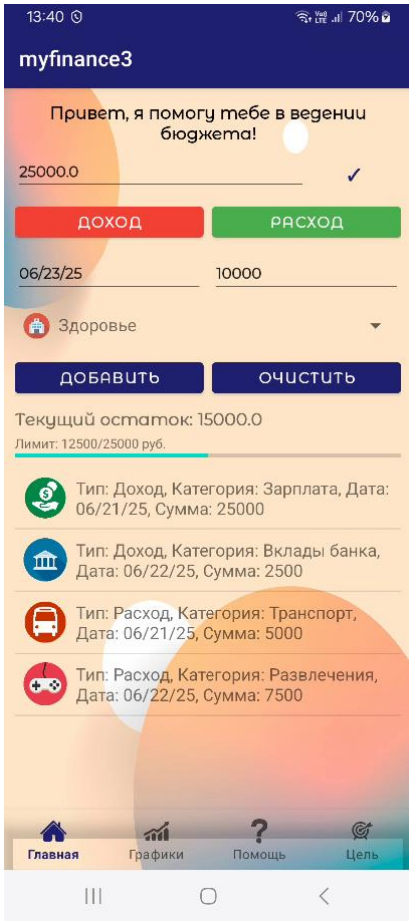


Рисунок 25 – Главная страница с операциями

На рисунке 26 изображено окно с превышением лимита трат. Окно высвечивается в том случае, если траты превысили установленный лимит на месяц и предупреждает об этом пользователей. Так же окно будет постоянно высвечиваться, при добавлении последующих операций расходов. Шкала лимита будет гореть красным, чтобы пользователь понимал о превышении расходов.

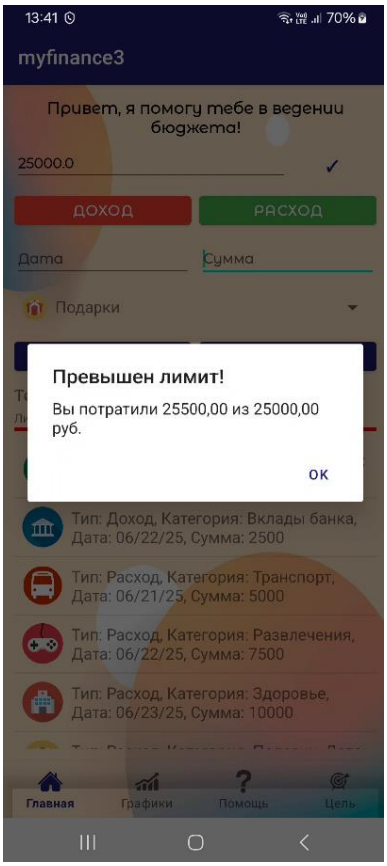


Рисунок 26 – Окно с предупреждением о превышении лимита

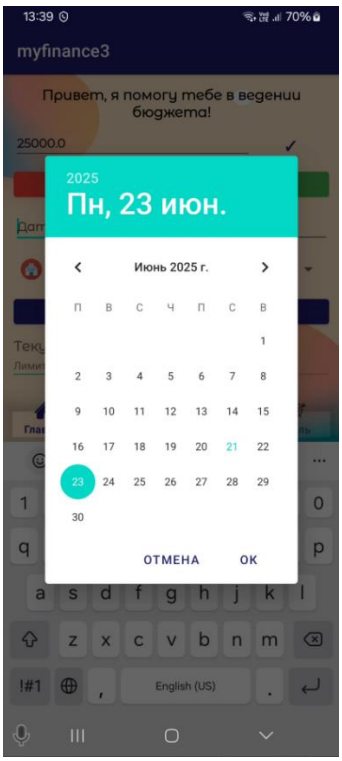


Рисунок 27 – Окно с выбором даты

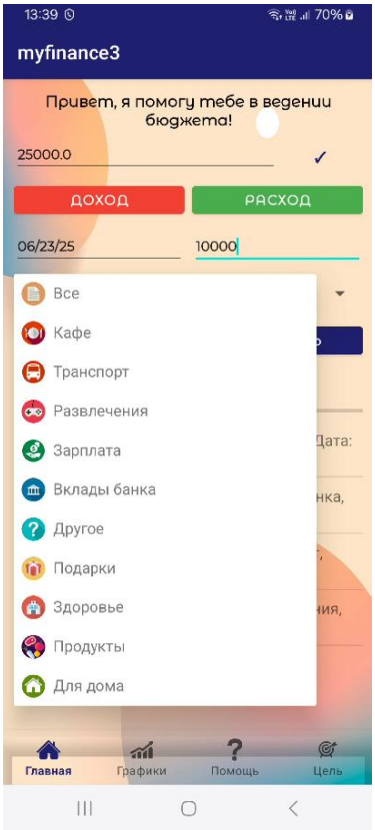


Рисунок 28 – Окно с выбором категории

На рисунке 29 изображена страница с графиками по операциям доходов и расходов, которые переносятся из списка операций главной страницы. При нажатии на кнопки, есть возможность посмотреть на другие вариации графиков. Сверху отображается график с доходами, снизу отображается график с расходами. Первый график отображает круговую диаграмму по категориям, второй график отображает столбчатую диаграмму по категориям и третий график отображает линейный график по датам.

Рисунок 29 – Страницы с графиками

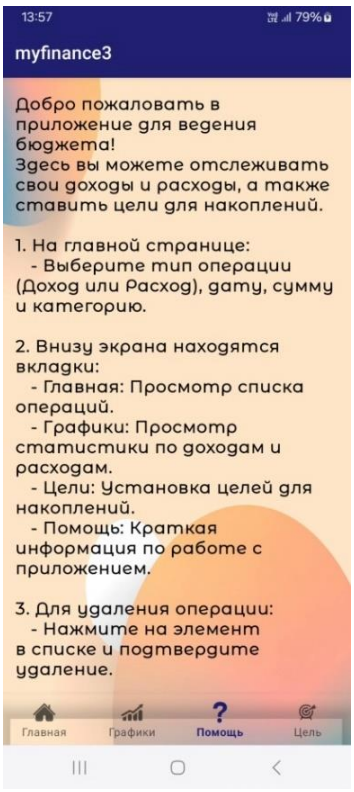


Рисунок 30 – Страница с информацией о работе приложения

На рисунке 31 изображена страница со списком целей. Через кнопку можно добавить новую цель накопления и заполнив форму, цель сохраняется в списке. В списке отображаются все текущие цели с категорией, датой, названием, накопленной и целевой суммой, а также шкалой накопления. Отдельно нажав на одну цель, можно узнать более подробную информацию о ней, добавить сумму и закрыть цель, если сумма равна или превышает целевую.

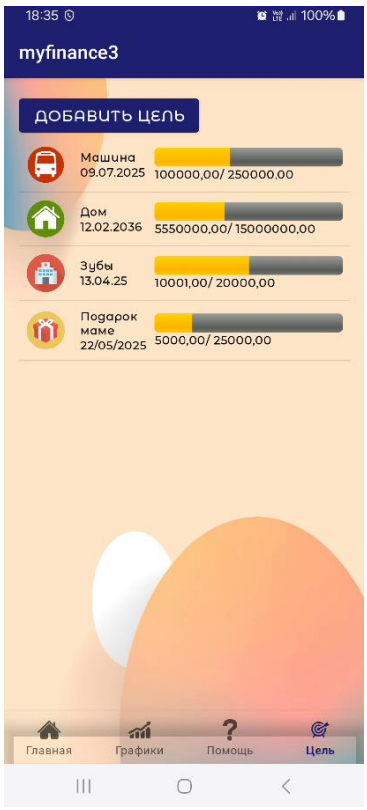


Рисунок 31 – Страница со списком целей

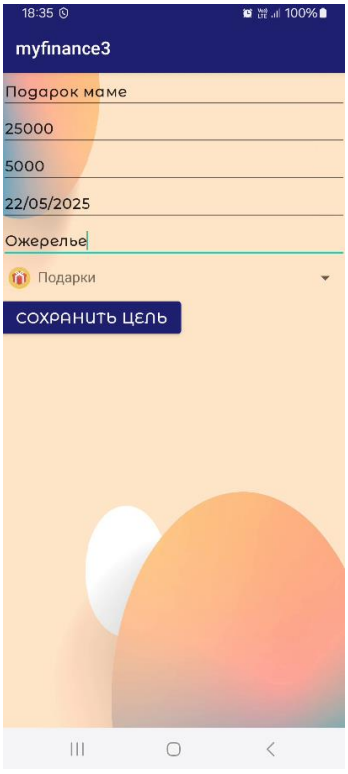


Рисунок 32 – Страница добавления цели

Рисунок 33 – Страница добавления суммы к цели или её завершение

После того как приложение было разработано, был создан apk-файл для проверки работоспособности на реальных мобильных телефонах.

APK – это формат исполняемых файлов-приложений для Android. Каждое приложение Android скомпилировано и упаковано в этот файл, который включает в себя весь код приложения и все ресурсы. Процесс компилирования можно наблюдать на рисунке 34.

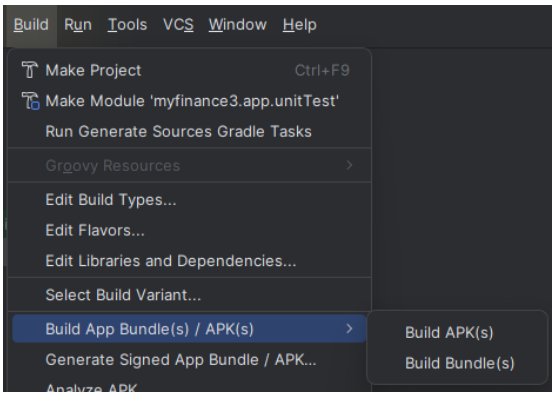


Рисунок 34 – Генерация APK

После этого, в директории проекта, создастся файл приложения, которое можно устанавливать на реальные устройства и опубликовывать в магазины приложений, такие как Google Play (рисунок 35).

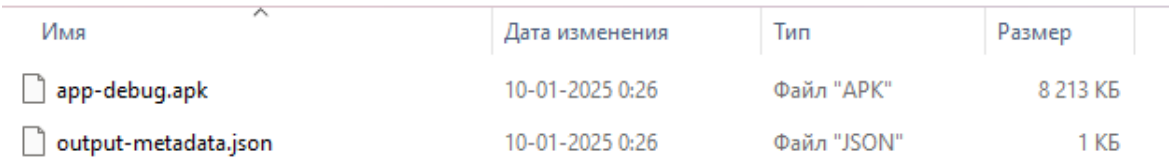


Рисунок 35 – Созданный APK-файл

Приложение было проверено на нескольких мобильных устройств с разными версиями Android, на всех оно работало стабильно и выполняла весь реализованный функционал.

Выводы по главе 3

В результате работы, проведенной в третьей главе, было создано рабочее мобильное приложение, разработанное с использованием Android Studio и языка Java.

В процессе его создания были реализованы ключевые функции, включая экраны авторизации, управление операциями, визуализацию данных с помощью графиков и экран для создания целей накопления.

Для хранения пользовательской информации применялось хранилище SharedPreferences, дополненное механизмом JSON-сериализации, а также через встроенную базу данных SQLite.

Чтобы обеспечить стабильную работу приложения, было проведено функциональное тестирование всех функций, что подтвердило правильность выполнения основных операций.

Завершением разработки стала успешная сборка APK-файла, который был установлен и протестирован на реальных мобильных устройствах, что ещё раз продемонстрировало готовность программного продукта к практическому применению.

Заключение

Выпускная квалификационная работа посвящена разработке мобильного приложения для ведения бюджета. Основной целью было создание практичного решения, помогающего пользователям фиксировать доходы и расходы, контролировать соблюдение бюджетных ограничений, изучать динамику платежей через графики, а также ставить и достигать целей накопления.

На начальном этапе работы была изучена специфика предметной области, и проанализирована особенность деятельности компании. Для наглядного представления архитектуры и логики работы приложения выбрано подходящее CASE-средство для создания UML-диаграмм, которые помогли структурировать и сформировать требования к будущему продукту.

В качестве среды разработки была выбрана Android Studio, а для реализации использовался язык программирования Java. Интерфейс приложения проектировался с учетом принципов удобства и современных тенденций в дизайне мобильных приложений.

Реализация функционала потребовала решения ряда технических задач. Была разработана система хранения и обработки финансовых данных, реализованы механизмы добавления и редактирования операций, установка лимита трат на месяц, предупреждение пользователя о превышении лимита трат на месяц, удаление операций, перенос операций на графики, возможность просматривать операции через три вида графиков, с возможностью переключения между ними. Внимание уделялось обеспечению стабильности работы приложения и корректности ввода и расчётов всех финансовых операций.

Проведенное тестирование подтвердило работоспособность всех модулей приложения.

Проверялась не только корректность выполнения операций, но и удобство взаимодействия пользователя с интерфейсом. Полученные результаты показали, что приложение успешно справляется с поставленными

задачами и готово к практическому использованию.

Разработанное мобильное приложение представляет собой законченный продукт, который может быть использован для эффективного управления личными финансами.

При разработке внимание уделялось удобству использования и простоте интерфейса.

В перспективе возможно его дальнейшее развитие за счет добавления новых функций, таких как синхронизация данных между устройствами, сортировка по дате и категориям, новые виды графиков, расширенная аналитика расходов или интеграция с банковскими сервисами.

Однако уже в текущем виде приложение полностью соответствует поставленным целям и может быть полезным инструментом для широкого круга пользователей.

Список используемой литературы и используемых источников

1. Андиева Е. Ю., Сидоренко В. С. Анализ Критериев Выбора Case– Средств // Редакционная коллегия, 2015. – 164 с.
2. Кочуров, А. И. Введение в теорию информационных систем: учебное пособие. М.: Юрайт, 2018. – 416 с.
3. Ахметов А. К. Операционная система Android: история создания и развития. Разработка приложений для платформы Android [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/operatsionnaya-sistema-android-istoriya-sozdaniya-i-razvitiya-razrabotka-prilozheniy-dlya-platformy-android (дата обращения: 04.12.2024).
4. Болхудере Е. И. Сравнение систем разработки мобильных приложений //XXIII Всероссийская студенческая научнопрактическая конференция Нижневартовского государственного университета, 2021. – С. 91-97.
5. Бурнет Э. Привет, Android! Разработка мобильных приложений. 2- е издание. – СПб.: Издательство «Питер», 2016. – 256 с.
6. Гузилов А. В., Мышенков К. С., Симонов М. Ф. Анализ качества case-средств для функционального моделирования систем // Информационно- аналитические и интеллектуальные системы для производства и социальной сферы, 2021. – 29 с.
7. Куликов С. С., Данилова Г. В., Смолякова О. Г., Меженная М. М. Тестирование программного обеспечения, 2019. – 277с.
8. Лешек А. Мацяшек. Анализ и проектирование информационных систем с помощью UML 2.0 / Лешек А. Мацяшек. – М.: Вильямс, 2016. – 816 с.
9. Федоров Н.В. Проектирование информационных систем на основе современных CASE-технологий. – М.: МГИУ, 2008. − 287 с.
10. Цуканова О.А. Методология И Инструментарий Моделированя Бизнес-Процессов Учебное пособие Университет ИТМО, 2015. – 101 с.

11. Требования к системе: классификации FURPS+ / [Электронный ресурс] – URL: https://sysana.wordpress.com/2010/09/16/furps/ (дата обращения: 01.11.2024).

12. Alireza Sahami Shirazi, Niels Henze, Albrecht Schmidt, Robin Goldberg, Benjamin Schmidt, Hansjorg Schmauder Insights into Layout Patterns of Mobile User Interfaces by an Automatic Analysis of Android Apps, 2013. – 10 с.

13. Android Studio [Электронный ресурс]. URL: https://developer.android.com/studio (дата обращения: 12.12.2024).

14. Joshua Marinacci, Chris Adamson, Swing Hacks. Tips and Tools for Killer GUIs, 2017. – 542 с.

15. Data modeling overview [Электронный ресурс] URL: https://www.sap.com/central-asia-caucasus/products/technology-platform/datasphere/what-is-data-modeling.html (дата обращения: 25.10.2024).

16. Matheus Flauzino, Júlio Veríssimo, Ricardo Terra, Elder Cirilo, Vinicius H. S. Durelli, Rafael S. Durelli Are you still smelling it? A comparative study between Java and Kotlin language, 2018. – 10 с.

17. Mrs. Selina Khoirom, Moirangthem Sonia, Borishphia Laikhuram, Jaeson Laishram, Tekcham Davidson Singh Comparative Analysis of Python and Java for Beginners, 2020 – 24 с.

18. Niclas Everlönn and Stylianos Gakis Java and Kotlin, a performance comparison, 2020. – 70 с.

19. Radosław Klimek, Piotr Szwed Formal Analysis of Use Case Diagrams, 2010. – 17 с.

20. Subham Bose, Aditi, Madhuleena Mukherjee, Madhurima Banerjee A Comparative Study: Java Vs Kotlin Programming In Android Application Development, 2018. – 5 с.