Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

**ТРЕКЕР ЛИЧНЫХ РАСХОДОВ**

Курсовая работа

по дисциплине «Разработка мобильных приложений»

Пояснительная записка

Студент гр. 584-М

\_\_\_\_\_ Зорин Д. Л.

«\_\_» \_\_\_\_\_\_\_\_ 2025 г.

Приняли:

К.т.н., Доцент кафедры КСУП

\_\_\_\_\_\_\_\_\_\_\_ Хабибулина Н.Ю.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2025 г.

Старший преподаватель каф. КСУП

\_\_\_\_\_ \_\_\_\_\_ Мурзин Е. С.

«\_\_» \_\_\_\_\_\_\_\_ 2025 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

УТВЕРЖДАЮ

Заведующий кафедрой КСУП, д-р техн. наук, профессор

Ю.А. Шурыгин

“ ” 2025 г.

**ЗАДАНИЕ**

на курсовую работу по дисциплине «Разработка мобильных приложений» студенту Зорину Даниилу Леонидовичу группы 584-М факультета вычислительных систем.

* 1. Тема работы: Трекер личных расходов
  2. Исходные данные к работе:
     1. Техническое задание
  3. Срок сдачи студентом законченной работы: до 14 июня 2025 г.
  4. Содержание курсовой работы:
     1. Введение:

– актуальность темы;

– цель и задачи курсовой работы;

– объект и предмет исследования;

– методология и инструменты разработки.

* + 1. Анализ предметной области:

– Анализ аналогов

– Целевая аудитория

– Функциональные требования

4.3 Проектирование и реализация

5 Содержание пояснительной записки:

* титульный лист;
* реферат на русском языке;
* задание;
* содержание;
* введение;
* основная часть;
* заключение;
* список использованных источников;

Пояснительная записка должна быть оформлена в соответствии со стандартом ТУСУР.

6 Дата выдачи задания:

«19» апреля 2025 г.

Задание согласовано:

Руководитель работы Мурзин Евгений Сергеевич

“ ” 2025 г.

Задание принято к исполнению

“ 19 ” апреля 2025 г. Зорин Д. Л.

**Реферат**

Курсовая работа, 29 страниц, 8 рисунков, 14 источников.

ТРЕКЕР ЛИЧНЫХ РАСХОДОВ, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, .NET MAUI, C#, SQLite, УПРАВЛЕНИЕ ФИНАНСАМИ, ДОХОДЫ, РАСХОДЫ, ОТЧЕТЫ, ЭКСПОРТ ДАННЫХ, ТЕСТИРОВАНИЕ.

Объектом исследования является мобильное приложение для учета личных финансов на платформе Android.

Цель работы - разработка приложения, позволяющего пользователям фиксировать доходы и расходы, анализировать финансовые операции и формировать отчеты.

Методами исследования являются изучение принципов проектирования мобильных приложений на .NET MAUI, разработка пользовательского интерфейса с применением паттерна MVVM, реализация функционала для добавления, редактирования и удаления операций, организация хранения данных используя SQLite.

Врезультате работысоздано приложение с возможностью добавления, редактирования, удаления, фильтрации и сортировки операций по категориям, реализована визуализация данных в виде графиков и отчетов, добавлена функция экспорта данных в форматы PDF, CSV и XLSX.

Технологическим стеком являютсяязык программирования: C#, фреймворк .NET MAUI и СУБД SQLite.

Приложение предназначено для широкого круга пользователей, желающих контролировать свои личные финансы.

Приложение готово к использованию, исходный код размещен на GitHub.

**Сокращения, обозначения, термины и определения**

В настоящей работе применяют следующие термины, сокращения и обозначения:

MAUI – кроссплатформенный фреймворк для разработки нативных пользовательских интерфейсов на C# (Multi-platform App UI);

MVVM – архитектурный шаблон проектирования, разделяющий представление, модель данных и логику взаимодействия (Model-View-ViewModel);

БД – база данных, предназначенная для хранения транзакций, целей и пользовательских настроек;

SQLite – встраиваемая реляционная система управления базами данных, используемая в мобильных и десктопных приложениях;

PDF – универсальный формат для представления документов, сохраняющий структуру и оформление (Portable Document Format);

CSV – текстовый формат хранения табличных данных, где значения разделены запятыми или точками с запятой (Comma-Separated Values);

XLSX – формат электронных таблиц Microsoft Excel, основанный на XML;

UML – унифицированный язык моделирования, применяемый для визуализации архитектуры и логики программных систем (Unified Modeling Language);

ViewModel – компонент архитектуры MVVM, обеспечивающий связь между данными модели и представлением (интерфейсом пользователя);

ObservableCollection – коллекция, автоматически уведомляющая об изменениях, используемая для отображения динамически изменяемых списков;

SwipeView – элемент управления в MAUI, реализующий функциональность свайпа (например, для редактирования или удаления элементов списка);

Command – объект команды, описывающий действие, привязанное к событию пользовательского интерфейса;

Data Binding – механизм связывания данных между моделью и представлением, обеспечивающий автоматическое обновление интерфейса;

ПИН-код – персональный идентификационный номер, служащий для защиты доступа к приложению;

ClosedXML – .NET-библиотека для чтения и записи файлов Excel в формате XLSX без использования Microsoft Office;

iText7 – библиотека для создания и редактирования PDF-файлов в .NET-приложениях;

SettingsViewModel – модель представления настроек, содержащая команды и свойства для управления настройками пользователя;

Transaction – сущность, представляющая одну финансовую операцию с типом, суммой, категорией и описанием.

**Оглавление**

[Введение 8](#_Toc200897940)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 10](#_Toc200897941)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 11](#_Toc200897942)

[2.1 Введение 11](#_Toc200897943)

[2.2 Цели и задачи проекта 11](#_Toc200897944)

[3 СТЕК ТЕХНОЛОГИЙ 13](#_Toc200897945)

[4 РАЗРАБОТКА ПО 14](#_Toc200897946)

[4.1 Анализ требований и планирование 14](#_Toc200897947)

[4.2 Проектирование архитектуры 17](#_Toc200897948)

[4.3 Реализация 19](#_Toc200897949)

[4.4 Тестирование 23](#_Toc200897950)

[Заключение 26](#_Toc200897951)

[Список использованных источников 28](#_Toc200897952)

**Введение**

В условиях динамичной экономики и роста личных финансовых потоков эффективный контроль доходов и расходов становится необходимым условием финансовой стабильности. Ручной учет устарел и неэффективен, тогда как мобильные приложения предлагают удобное, точное и доступное решение. Разработка функционального трекера расходов для платформы Android, обеспечивающего не только фиксацию операций, но и их глубокий анализ, формирование отчетов и безопасное хранение данных, обладает высокой практической значимостью и востребована среди пользователей, стремящихся к осознанному управлению бюджетом.

Целью данной работы является разработка мобильного приложения «Трекер личных расходов» для платформы Android, обеспечивающего контроль над личными финансами путем учета доходов и расходов, их анализа и формирования отчетов.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить принципы проектирования мобильных приложений на .NET MAUI.
2. Разработать пользовательский интерфейс приложения.
3. Реализовать функционал добавления, редактирования и удаления финансовых операций (доходов/расходов).
4. Организовать хранение данных о финансовых операциях используя СУБД SQLite.
5. Реализовать фильтрацию и сортировку расходов по категориям.
6. Реализовать отображение финансовой статистики в виде графиков и отчетов.
7. Реализовать возможность экспорта данных в форматы PDF, CSV, XLSX и импорта данных в форматах CSV, XLSX по шаблону.

Объект и предмет исследования.

Объект исследования является процесс разработки мобильного приложения для учета личных финансов на платформе Android.

Предметом исследования являются технологии и методы проектирования, реализации и тестирования функционала мобильного приложения «Трекер личных расходов» с использованием .NET MAUI и C#, включая организацию хранения данных в СУБД SQLite, формирование отчетов и механизмы импорта/экспорта.

**1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

В современном мире вопросы личных финансов становятся всё более актуальными. Управление бюджетом, планирование расходов и отслеживание доходов — неотъемлемая часть повседневной жизни. С развитием цифровых технологий всё больше пользователей стремятся использовать мобильные приложения для контроля своих финансов, так как это обеспечивает удобство, наглядность и доступность в любой момент времени.

Предметной областью настоящей работы является **ведение и анализ личного бюджета.** В её рамках решаются следующие основные задачи:

1. фиксация доходов и расходов пользователя;
2. категоризация финансовых операций;
3. хранение истории транзакций;
4. визуальный и статистический анализ;
5. обеспечение безопасности данных;
6. импорт и экспорт данных в различных форматах;
7. постановка финансовых целей и контроль их достижения.

На рынке существует множество приложений, выполняющих подобные функции (например, CoinKeeper, Zen-Money, Monefy), однако большинство из них либо предоставляют ограниченный функционал бесплатно, либо не позволяют адаптировать приложение под конкретные потребности пользователя.

Таким образом, создание собственного приложения для учёта финансов позволяет глубже понять принципы построения подобных решений, научиться работать с базами данных, реализовать интерфейсы взаимодействия с пользователем, а также рассмотреть вопросы безопасности и удобства пользовательского опыта. Кроме того, реализация функций импорта/экспорта и фильтрации операций предоставляет дополнительную гибкость и приближает приложение к реальным потребностям пользователей.

**2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**2.1 Введение**

Разработка мобильного приложения для учета личных финансов, позволяющего пользователю фиксировать доходы и расходы, анализировать траты и формировать отчеты. Приложение разрабатывается для платформы Android с использованием языка программирования C# и фреймворка .NET MAUI.

**2.2 Цели и задачи проекта**

**Цель:** разработать мобильное приложение для контроля личных финансов с возможностью ведения записей о доходах и расходах.

**Задачи:**

1. Изучить принципы проектирования мобильных приложений на .NET MAUI;
2. Разработать пользовательский интерфейс;
3. Реализовать функционал добавления, редактирования и удаления финансовых операций;
4. Организовать хранение данных используя СУБД SQLite;
5. Добавить фильтрацию и сортировку расходов по категориям;
6. Реализовать отображение финансовой статистики (графики, отчеты);
7. Реализовать возможность экспорта данных.

**2.3 Функциональные требования**

**Управление финансами:**

1. Добавление доходов и расходов;
2. Выбор категории для каждой транзакции;
3. Указание даты и описания операции;
4. Возможность редактирования и удаления записей.

**Анализ расходов:**

1. Просмотр списка операций с фильтрацией по дате и категории;
2. Генерация отчетов по заданному периоду;
3. Визуализация данных в виде графиков.

**Экспорт и импорт данных:**

1. Экспорт отчетов в PDF/CSV/XLSX;
2. Импорт данных в CSV и XLSX по определенному шаблону.

2.4 Технологический стек

1. Язык программирования: C#;
2. Фреймворк: .NET MAUI.

**3 СТЕК ТЕХНОЛОГИЙ**

Для реализации мобильного приложения учёта личных финансов использовался современный стек технологий, обеспечивающий кроссплатформенную разработку, удобную архитектуру и поддержку пользовательского интерфейса. Основой проекта стал фреймворк .NET MAUI (Multi-platform App UI) — кроссплатформенное решение от компании Microsoft, позволяющее разрабатывать приложения для Android, iOS и Windows на базе единого кода. В качестве основного языка программирования применялся C#, а для разметки интерфейсов использовался XAML.

Для хранения пользовательских данных (транзакций, целей) использовалась встроенная база данных SQLite, интеграция с которой осуществлялась через библиотеку SQLite-net. Архитектура приложения построена на паттерне MVVM (Model-View-ViewModel), обеспечивающем разделение представления, бизнес-логики и моделей данных.

В рамках функциональности экспорта и импорта данных использовались библиотеки iText7 для генерации PDF-файлов (включая поддержку кириллицы) и ClosedXML — для работы с таблицами Excel (.xlsx), что позволило реализовать полноценный экспорт и импорт финансовой информации на мобильных устройствах. Также в проекте применялись компоненты .NET MAUI Essentials для работы с файловой системой, отображения диалогов и взаимодействия с функциональностью платформы.

Такой стек технологий обеспечил удобство разработки, надёжность хранения данных и широкие возможности расширения функционала приложения в будущем.

**4 РАЗРАБОТКА ПО**

**4.1 Анализ требований и планирование**

На первом этапе разработки приложения был проведён подробный анализ требований, как функциональных, так и нефункциональных. Основной целью являлось создание мобильного приложения, позволяющего пользователю вести учёт личных финансов. Ключевая задача заключалась в обеспечении простого, удобного и интуитивно понятного интерфейса, доступного широкому кругу пользователей, включая тех, кто не имеет технических навыков. В рамках анализа требований была выделена основная функциональность, без которой приложение теряло бы свою ценность: добавление, редактирование и удаление финансовых операций, разделённых на категории доходов и расходов; отображение общего баланса, суммы доходов и расходов за месяц; фильтрация транзакций по дате и категории; экспорт данных в формат PDF; а также ведение целей накоплений с отображением прогресса их выполнения.

Также были сформулированы дополнительные, но крайне важные возможности: импорт данных из таблиц CSV и Excel (XLSX), создание безопасного доступа к приложению с использованием пин-кода, а также возможность полного удаления всех пользовательских данных. Эти функции дополняли основную идею и делали продукт более конкурентоспособным и удобным для повседневного использования.

Параллельно с этим был проведён анализ целевой аудитории. Предполагалось, что основными пользователями станут студенты, работающие граждане и семьи, желающие контролировать личный или семейный бюджет. Это повлияло на требования к дизайну: он должен был быть лёгким, не перегруженным, с возможностью быстрого доступа к ключевым функциям. Визуально важно было обеспечить чистоту интерфейса, использование приятных пастельных цветов и логичное расположение элементов.

С технической точки зрения ставилась задача создать кроссплатформенное решение, работающее на устройствах Android и iOS. Это обусловило выбор платформы .NET MAUI, обеспечивающей единый код для нескольких операционных систем. Также было решено использовать SQLite в качестве встроенной базы данных, поскольку она идеально подходит для мобильных приложений благодаря своей лёгкости и автономности. Для реализации импорта и экспорта данных планировалось использовать библиотеки CsvHelper и ClosedXML, поскольку они предоставляют надёжные инструменты для работы с соответствующими форматами.

На рисунке 4.1 показана диаграмма вариантов использования. На рисунке 4.2 показана диаграмма активности.

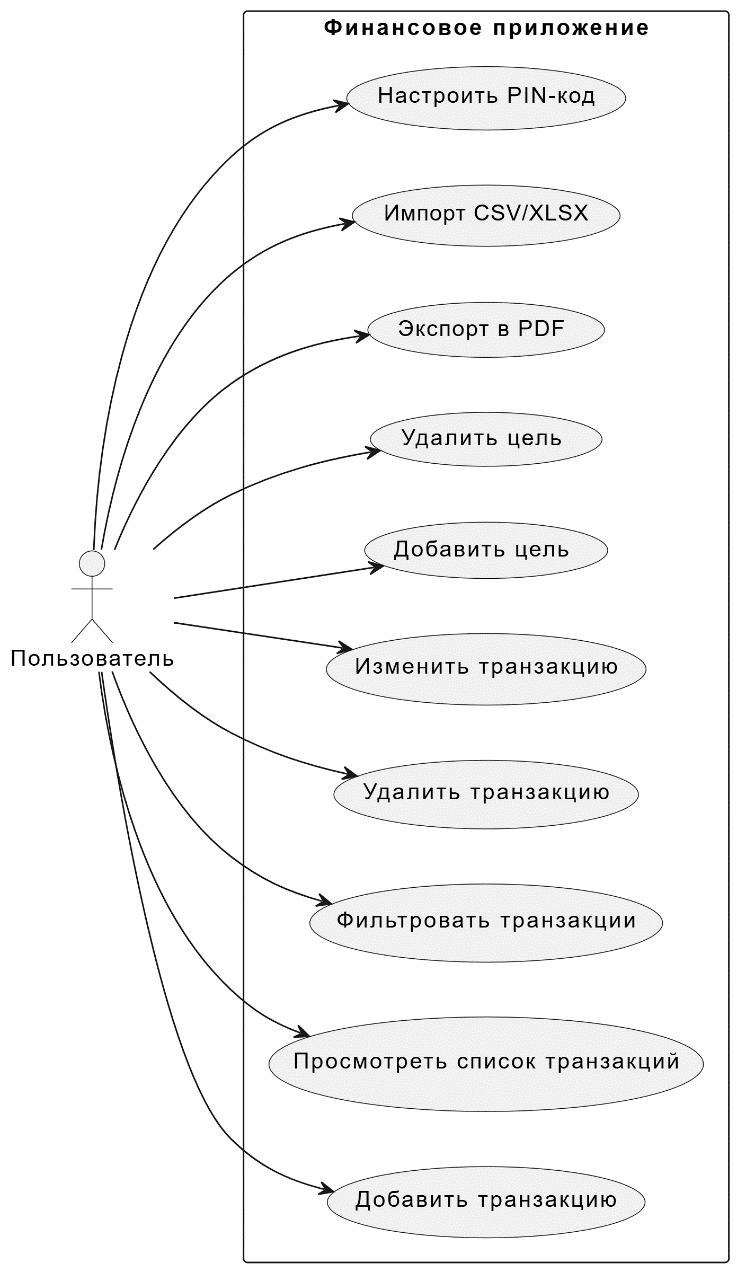


Рисунок 4.1 – Диаграмма «Use Case»

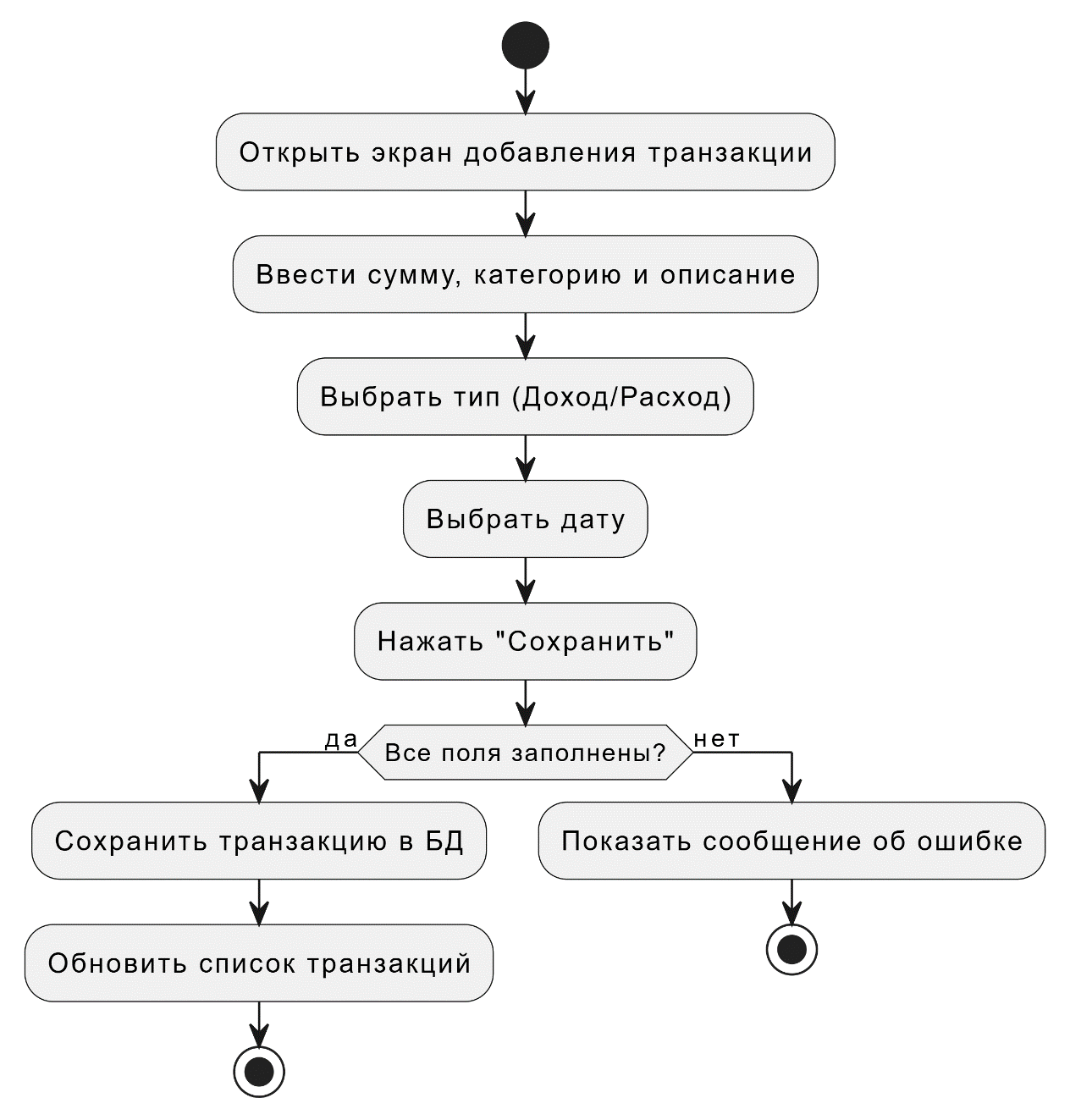


Рисунок 4.2 – Диаграмма активности

На основании проведённого анализа был составлен предварительный план работ, включающий поэтапную реализацию функционала, начиная от базовых возможностей до расширенных опций. Приоритет был отдан стабильности работы, плавной навигации между страницами и точности представления финансовых данных. Каждый следующий этап разработки основывался на результатах предыдущего, что обеспечивало последовательное и логически выверенное развитие проекта.

**4.2 Проектирование архитектуры**

На этапе проектирования архитектуры основное внимание было уделено обеспечению модульности, расширяемости и сопровождения приложения. В рамках учебной практики была выбрана архитектурная модель MVVM (Model–View–ViewModel), которая является стандартом для приложений на .NET MAUI и позволяет отделить пользовательский интерфейс от бизнес-логики, схема представлена на рисунке 4.3. Это упростило процесс разработки, а также повысило читаемость и тестируемость кода.

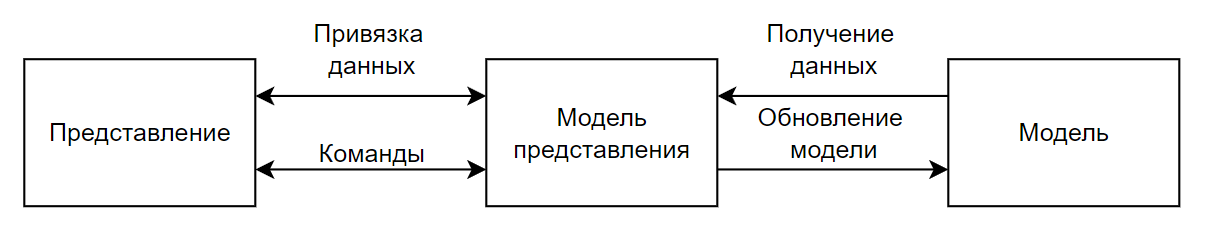


Рисунок 4.3 – схема MVVM

Модель (Model) представляет собой слой, отвечающий за структуру данных и взаимодействие с базой данных. В частности, были созданы классы Transaction, Goal и другие, отражающие сущности предметной области. Эти классы напрямую взаимодействуют с SQLite через сервис DatabaseService, который инкапсулирует операции чтения, добавления, обновления и удаления данных. Такой подход позволил централизовать работу с хранилищем и минимизировать дублирование логики.

Представление (View) реализовано с использованием XAML-разметки. Каждая страница приложения (например, MainPage, TransactionListPage, EditTransactionPage, GoalsPage, SettingsPage) представляет собой отдельный экран, связанный с соответствующей ViewModel. Использование XAML позволило точно контролировать внешний вид интерфейса, обеспечивая поддержку адаптивности, отступов, тем оформления и жестов взаимодействия (например, свайпов по элементам списка).

ViewModel — промежуточный слой, обеспечивающий связь между Model и View. В каждой ViewModel реализуются команды (например, AddCommand, EditCommand, DeleteCommand), свойства для привязки к интерфейсу и логика обработки данных, полученных от модели. ViewModel подписана на изменения, реализуя интерфейс INotifyPropertyChanged, что позволяет автоматически обновлять данные в интерфейсе при изменении состояния.

Особое внимание было уделено навигации и маршрутизации. В качестве механизма переходов между экранами использовался Shell, что позволило реализовать удобную и гибкую навигационную структуру, включая возможность передачи параметров между страницами.

Для более надёжной и масштабируемой архитектуры приложение было условно разделено на несколько логических пакетов:

* Models (модели данных),
* ViewModels (бизнес-логика и команды),
* Views (представления),
* Services (вспомогательные сервисы, в частности, для работы с БД, импортом/экспортом, настройками),
* Converters (XAML-конвертеры для отображения данных в нужном формате).

Диаграммы классов и пакетов также были построены на данном этапе, чтобы визуализировать структуру приложения. Диаграмма классов отражает ключевые сущности (например, Transaction, Goal, TransactionListViewModel) и их отношения. Диаграмма пакетов демонстрирует взаимодействие между слоями и модулями приложения. Эти диаграммы помогли обеспечить целостность архитектуры и послужили базой для дальнейшей реализации.

Таким образом, результатом этапа проектирования архитектуры стало чётко структурированное приложение, в котором соблюдаются принципы разделения ответственности и инкапсуляции, что позволяет легко масштабировать и модифицировать проект в будущем.

**4.3 Реализация**

Этап реализации представляет собой основной этап разработки программного продукта, на котором спроектированные архитектурные решения и интерфейсы были непосредственно воплощены в виде программного кода. В ходе разработки приложения внимание уделялось соблюдению принципов чистой архитектуры, читаемости кода и возможности расширения функциональности в будущем.

Реализация началась с создания базовых моделей, отражающих ключевые сущности предметной области. Были разработаны классы Transaction и Goal, описывающие финансовые операции и цели пользователя. Эти классы включают в себя необходимые свойства и реализуют интерфейс INotifyPropertyChanged, обеспечивая автоматическую реакцию пользовательского интерфейса на изменения данных.

Следующим шагом стало создание слоя доступа к данным. Сервис DatabaseService реализует взаимодействие с базой данных SQLite и инкапсулирует все CRUD-операции (создание, чтение, обновление, удаление). Это позволило централизовать работу с данными и упростить сопровождение кода. Дополнительно реализованы сервисы экспорта и импорта данных в формате CSV и XLSX, включая поддержку кириллицы и корректное отображение данных при генерации PDF-отчётов через библиотеку iText7.

Основной пользовательский интерфейс реализован с использованием технологий .NET MAUI и XAML. Каждый экран — будь то главный экран, список транзакций, экран создания целей или настроек — был оформлен с учётом удобства взаимодействия, визуальной иерархии и адаптивности. Пользователь может добавлять, редактировать и удалять транзакции и цели, просматривать историю расходов и доходов, фильтровать их по дате и категории. Реализована поддержка свайпов по элементам списка с привязкой к командам редактирования и удаления.

Во ViewModel для каждой страницы реализована бизнес-логика, команды и обработка пользовательских действий. Был реализован механизм фильтрации транзакций по выбранной дате и категории. Это позволило пользователю быстро находить нужную информацию, что особенно важно при анализе личных финансов. Для передачи данных между страницами использовались маршруты Shell с параметрами.

Для обеспечения безопасного доступа реализована возможность установки PIN-кода, который требуется при запуске приложения. Эта функция хранит PIN в защищённом хранилище и предотвращает несанкционированный доступ к финансовым данным.

Также было реализовано сохранение и экспорт данных в PDF с использованием пользовательского шрифта для корректного отображения кириллицы. Пользователь может выбрать период, за который необходимо сформировать отчёт, и мгновенно получить документ с разбивкой по операциям и итоговыми данными.

На протяжении всей реализации использовались средства отладки, логирования и визуального тестирования интерфейса. Приложение разрабатывалось с учётом возможности запуска на различных мобильных устройствах, что обеспечивается кроссплатформенной природой .NET MAUI.

В результате этапа реализации было создано полнофункциональное приложение для персонального учёта финансов с чистой архитектурой, удобным интерфейсом и возможностями анализа, экспорта и настройки пользовательского взаимодействия. Этот этап завершил техническую часть проекта и подготовил приложение к тестированию и презентации.

Диаграмма классов показана на рисунке 4.4.

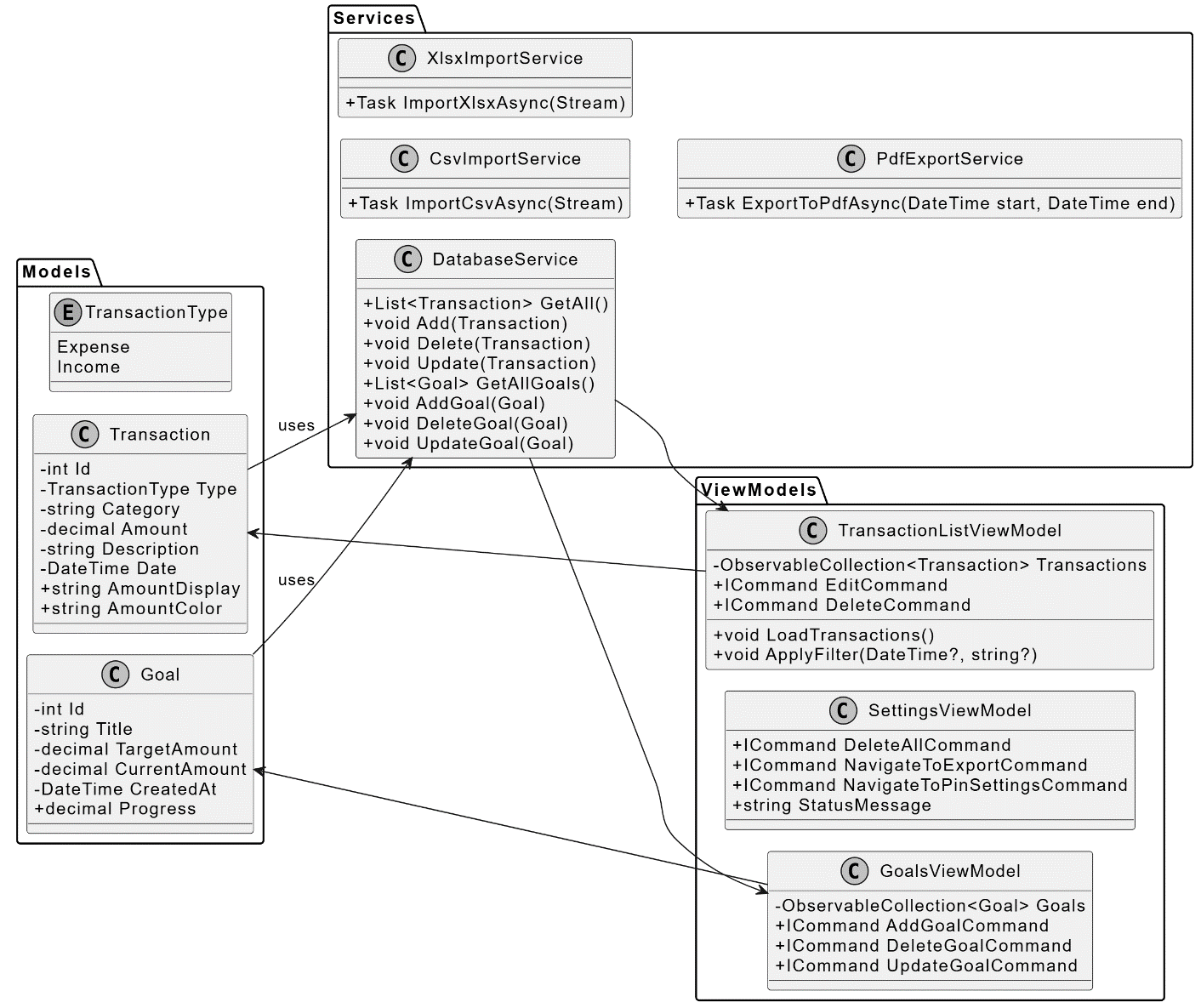


Рисунок 4.4 – Диаграмма классов

Диаграмма пакетов предоставлена на рисунке 4.5.

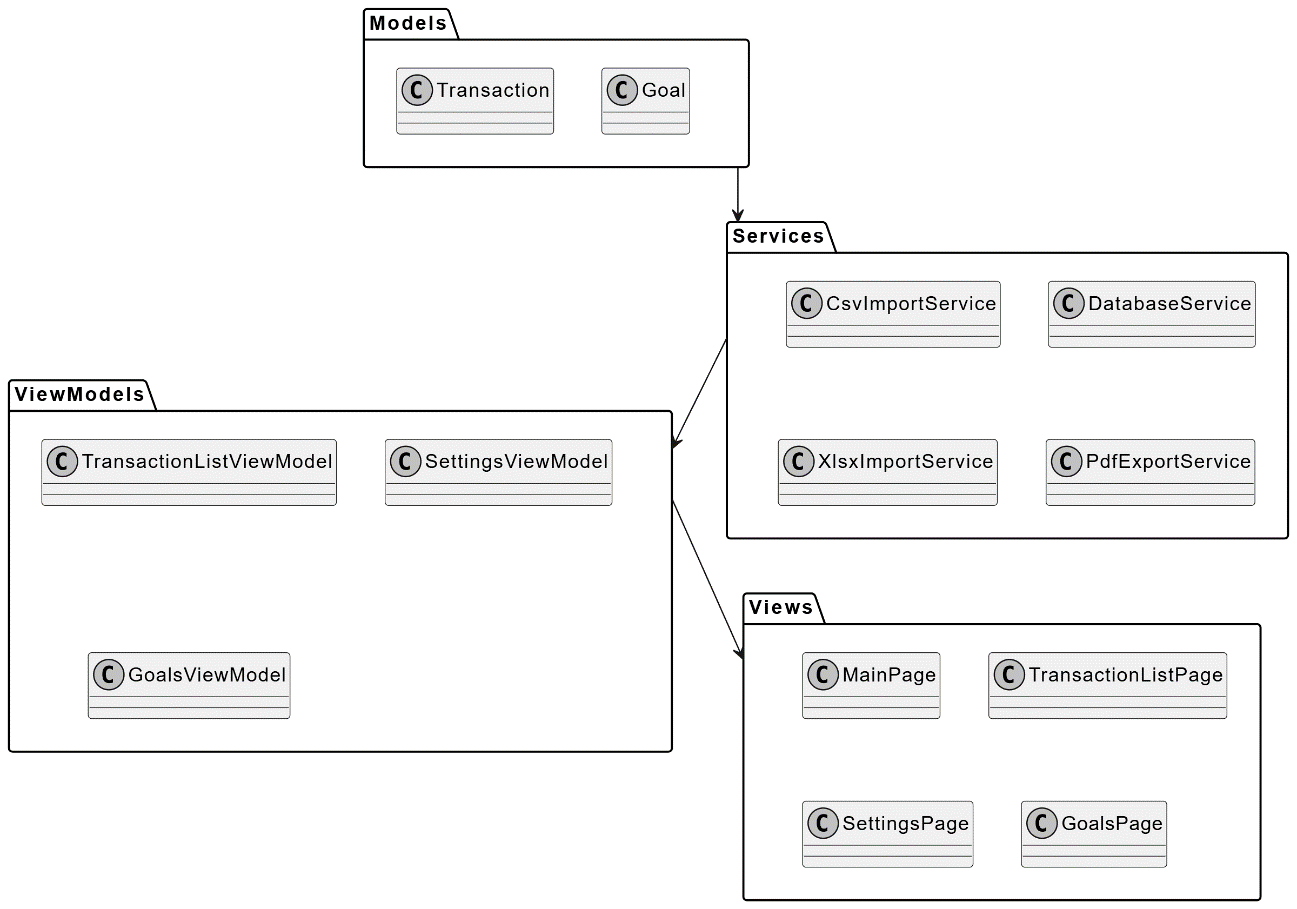


Рисунок 4.5 – Диаграмма пакетов

На рисунках 4.6 – 4.8 показано работающее приложение.

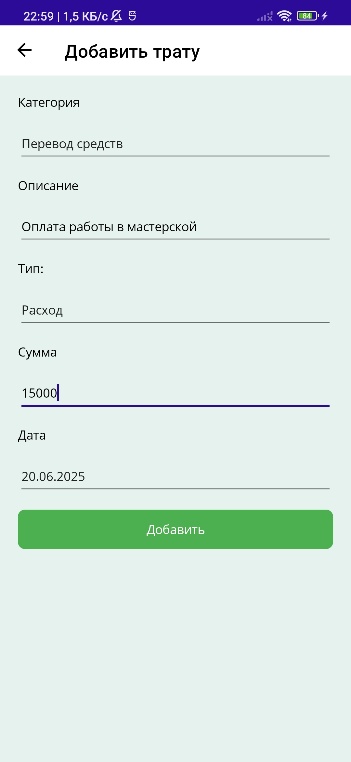
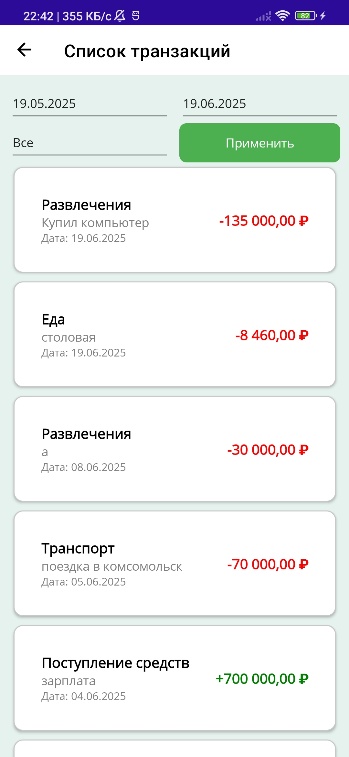
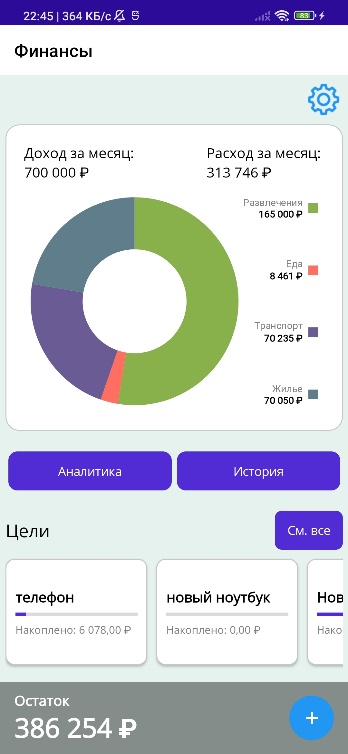


Рисунок 4.6 – Главное окно, список транзакций, добавление транзакции

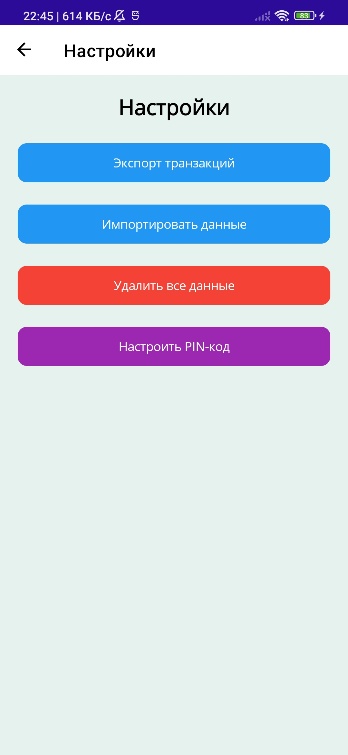
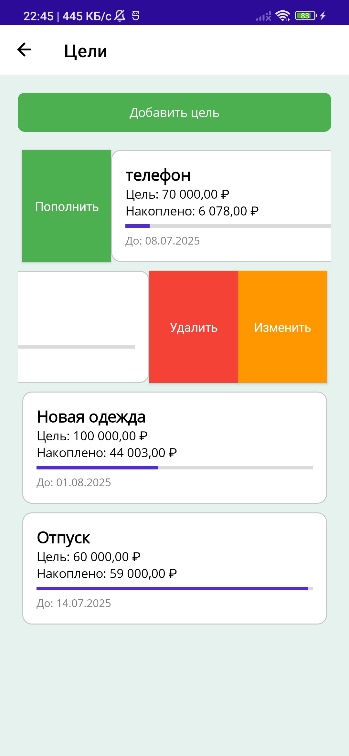


Рисунок 4.7 – Страницы цели, настроек, экспорта данных

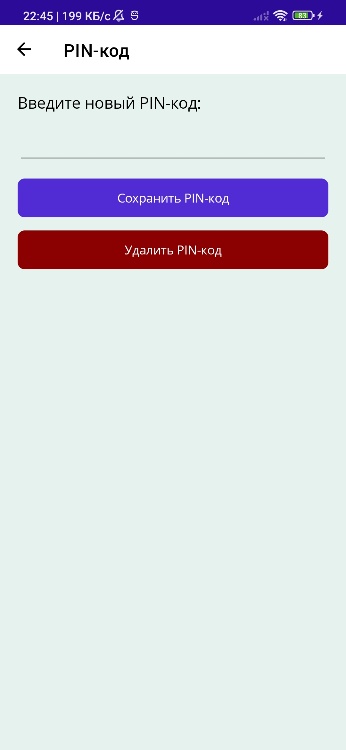
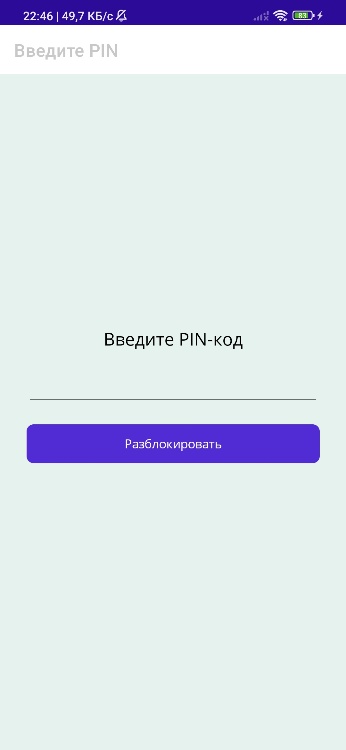
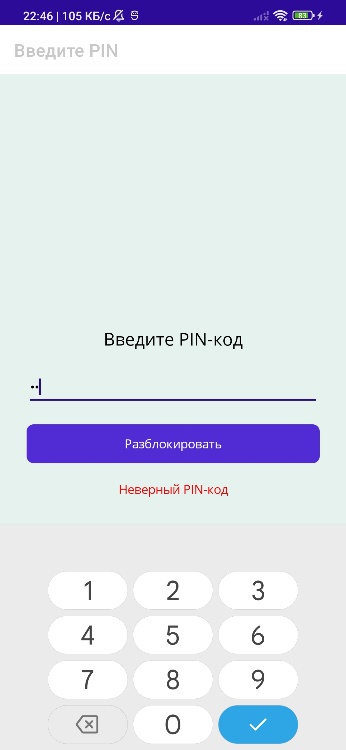
  

Рисунок 4.8 – Страницы настройки PIN-кода, ввода PIN-кода

**4.4 Тестирование**

После завершения реализации приложения был проведён этап тестирования, целью которого являлась проверка корректности работы всех компонентов системы, а также выявление и устранение возможных ошибок и недочётов.

Тестирование проводилось вручную, с применением функционального подхода. Каждый модуль и экран приложения проверялся на соответствие требованиям, заданным на этапе проектирования. Основное внимание уделялось следующим аспектам:

**Корректность добавления, редактирования и удаления транзакций и целей.** Проверялось, что пользователь может создавать операции с различными параметрами (тип, категория, сумма, дата), вносить изменения и удалять их без ошибок, а изменения сразу отображаются в списке.

**Работа фильтрации по дате и категории**. Был протестирован механизм выбора диапазона дат и категорий. Проверялось, что фильтрация отображает только релевантные записи и корректно очищается по команде пользователя.

**Корректность экспорта в PDF, CSV и XLSX.** Особое внимание было уделено отображению кириллических символов в отчётах, правильной сортировке данных и оформлению таблиц. Были протестированы различные периоды отчётов (один день, месяц, произвольный диапазон), чтобы убедиться в стабильности вывода.

**Импорт данных из CSV и XLSX.** Проверялась корректность распознавания данных, сохранение их в базу, отсутствие дублирования. Тестировались случаи неполных или некорректных данных.

**Проверка интерфейса на разных экранах и ориентациях устройств.** Проводилось тестирование на эмуляторах и реальных устройствах Android с различными разрешениями экранов, чтобы убедиться в адаптивности интерфейса.

**Работа механизма защиты с помощью PIN-кода**. Проверялась установка, проверка и сброс PIN. Убедились, что приложение не допускает доступ без правильного кода.

**Обработка исключений и ошибок.** Искусственно создавались ситуации, в которых могли возникнуть ошибки (например, отсутствие доступа к файлам, пустые поля, попытка экспортировать без данных), чтобы проверить, как приложение сообщает об ошибках и сохраняет стабильную работу.

Во время тестирования были выявлены и устранены следующие недочёты:

* Неправильное отображение кириллицы в PDF-документах до подключения шрифта;
* Отсутствие обновления списка после редактирования транзакции (решено через INotifyPropertyChanged и обновление коллекции);
* Ошибки при импорте некорректно отформатированных CSV-файлов.

По результатам тестирования можно сделать вывод, что приложение работает стабильно, соответствует заявленным требованиям и обладает необходимым уровнем надёжности. Проведённые проверки подтвердили корректную реализацию всех основных функций, что позволяет рекомендовать приложение к дальнейшему использованию и доработке.

**Заключение**

В ходе выполнения курсовой работы было разработано мобильное приложение **«Трекер личных расходов»** для платформы Android, позволяющее пользователям эффективно управлять личными финансами. Приложение предоставляет удобный инструмент для учета доходов и расходов, их анализа и формирования отчетов, что способствует повышению финансовой дисциплины и осознанному планированию бюджета.

1. **Основные результаты работы:**
2. **Реализован базовый функционал:**
3. Добавление, редактирование и удаление финансовых операций.
4. Гибкая система категорий для классификации доходов и расходов.
5. Фильтрация и сортировка операций по дате, категориям и сумме.
6. **Организовано хранение данных:**

Для надежного и структурированного хранения информации использовалась **СУБД SQLite**, что обеспечило высокую производительность и безопасность данных.

1. **Разработаны аналитические инструменты:**
2. Визуализация данных в виде графиков и диаграмм.
3. Формирование отчетов за выбранный период.
4. **Реализован экспорт и импорт данных:**
5. Поддержка форматов **PDF, CSV и XLSX** для удобного обмена информацией.
6. Импорт данных из CSV/XLSX по шаблону для быстрого внесения операций.
7. Приложение проверено на корректность работы основных функций, стабильность и удобство интерфейса.

**Перспективы развития проекта:**

1. Добавление синхронизации данных между устройствами через облачные сервисы.
2. Внедрение системы напоминаний о регулярных платежах.
3. Расширение аналитики: прогнозирование расходов, рекомендации по экономии.

**Список использованных источников**

1. ОС ТУСУР 01-2021. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления. Томск: Томский гос. ун-т систем упр. и радиоэлектроники, 2021. 52 с.
2. Microsoft .NET MAUI Documentation [Электронный ресурс]: Официальная документация Microsoft. — Режим доступа: <https://learn.microsoft.com/en-us/dotnet/maui/> (дата обращения: 10.02.2025).
3. SQLite Documentation [Электронный ресурс]: Официальная документация SQLite. — Режим доступа: <https://www.sqlite.org/docs.html> (дата обращения: 11.02.2025).
4. Троелсен Э. Язык программирования C# 10 и платформа .NET 6. – М.: Вильямс, 2022. – 1328 с.
5. Microsoft C# Guide [Электронный ресурс]: Официальное руководство по C#. — Режим доступа: <https://learn.microsoft.com/en-us/dotnet/csharp/> (дата обращения: 12.02.2025).
6. Паттерн MVVM в .NET MAUI [Электронный ресурс]: Руководство по реализации. — Режим доступа: <https://learn.microsoft.com/en-us/dotnet/maui/xaml/fundamentals/mvvm> (дата обращения: 13.02.2025).
7. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2021. – 366 с.
8. Microsoft Data Access with SQLite in .NET MAUI [Электронный ресурс]: Руководство разработчика. — Режим доступа: <https://learn.microsoft.com/en-us/dotnet/maui/data-cloud/database-sqlite> (дата обращения: 14.02.2025).
9. Export to PDF in .NET MAUI [Электронный ресурс]: Библиотека Syncfusion. — Режим доступа: <https://help.syncfusion.com/maui/pdf-library/overview> (дата обращения: 15.02.2025).
10. Работа с Excel и CSV в C# [Электронный ресурс]: Руководство по библиотеке EPPlus. — Режим доступа: <https://epplussoftware.com/docs/5.8/> (дата обращения: 16.03.2025).
11. Тестирование мобильных приложений [Электронный ресурс]: Руководство по xUnit для .NET MAUI. — Режим доступа: <https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-dotnet-test> (дата обращения: 17.03.2025).
12. Material Design Guidelines for .NET MAUI [Электронный ресурс]: Принципы UI/UX дизайна. — Режим доступа: <https://material.io/design> (дата обращения: 18.04.2025).
13. Финансовая грамотность: основы учета личных расходов [Электронный ресурс]: Методическое пособие. — Режим доступа: <https://fincult.info/> (дата обращения: 19.03.2025).
14. GitHub Repository: Personal Finance Tracker [Электронный ресурс]: Исходный код проекта. — Режим доступа: <https://github.com/username/repository> (дата обращения: 10.06.2025).