МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

**федеральное государственное бюджетное образовательное учреждение**
**высшего образования**
**«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ**
**ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**
**филиал ФГБОУ ВО «РГГМУ» в г. Туапсе**

**Кафедра «Экономики и управления на предприятии природопользования»**

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**(бакалаврская работа)**
**по направлению подготовки 09.03.03 Прикладная информатика**
**(квалификация – бакалавр)**

**На тему «Разработка мобильного приложения ДоМонетки»**

**Исполнитель** Кулаков Вадим Романович

**Руководитель** к.т.н., Сафонова Татьяна Владимировна

**«К защите допускаю»**
**Руководитель кафедры** _____

кандидат экономических наук

Майборода Евгений Викторович

«20» 01 2025 г.

Филиал Российского государственного
гидрометеорологического университета в г. Туапсе
НОРМОКОНТРОЛЬ ПРОЙДЕН
«15» 01 2024 г.
Мартынова Т.В.
ПОДПИСЬ РАСШИФРОВКА ПОДПИСИ

Туапсе
2025

ОГЛАВЛЕНИЕ

Введение ..... 4
1 Теоретическая часть ..... 7
1.1Анализ предметной области ..... 7
1.2 Актуальность разработки мобильного приложения «ДоМонетки» ..... 7
1.3 Анализ функциональных характеристик мобильного приложения «ДоМонетки» ..... 8
1.4 Анализ разрабатываемых решений ..... 8
1.5 Анализ аналогов и доступных решений на рынке ..... 10
1.6 Анализ среды и выбор методов решения задач ..... 13
1.7 Определение плана и методов разработки ..... 14
1.8 Выбор CASE-средств и их обоснование ..... 15
1.9 SWOT-анализ ..... 16
1.10 VCM-анализ ..... 17
2 Проектирование приложения ..... 18
2.1 Язык моделирования UML ..... 18
2.2 Проектирование мобильного приложения «ДоМонетки» в нотации UML, диаграмма вариантов использования ..... 18
2.3 Сценарии в нотации UML ..... 19
2.4 Проектирование мобильного приложения «ДоМонетки» в нотации UML, диаграмма последовательности ..... 22
2.5 Проектирование мобильного приложения «ДоМонетки» в нотации UML, диаграмма классов ..... 24
2.6 Проектирование мобильного приложения «ДоМонетки» в нотации UML, диаграмма компонентов ..... 25
2.7 Проектирование мобильного приложения «ДоМонетки» в нотации UML, диаграмма развертывания ..... 26
2.8 Проектирование мобильного приложения «ДоМонетки», модель архитектуры C4 ..... 27

2.9 Проектирование мобильного приложения «ДоМонетки» в нотации IDEF0 ..... 30
3 Разработка приложения ..... 33
3.1 Описание выбранных средств разработки и языка программирования 33
3.2 Описание основных алгоритмов приложения ..... 34
3.3 Описаниепрограммныхмодулей ..... 41
3.4 Описание интерфейса приложения ..... 43
Заключение ..... 51
Список литературы ..... 53
Приложение ..... 57

# Введение

В последние годы перед Российской Федерацией встала острая необходимость обеспечения независимости в сфере информационных технологий. В условиях нарастающего внешнего давления, усиленных международными санкциями и ограничением доступа к зарубежным технологиям, вопрос перехода на отечественное программное обеспечение (ПО) стал не только технологической, но и стратегической задачей национальной безопасности. Информационные системы государства, частного бизнеса и критически важных отраслей экономики требуют надёжных, проверенных временем решений, которые могут обеспечить как защиту данных, так и независимость от внешних угроз.

Мировая практика показывает, что использование коммерческих программных продуктов, особенно разработанных зарубежными компаниями, несёт риски не только с точки зрения кибербезопасности, но и контроля за данными. В случае конфликтов и санкций может быть заблокирован доступ к обновлениям, лицензиям и поддержке. Это особенно опасно для государственных организаций и критически важных предприятий, где сбои в работе информационных систем могут привести к серьёзным экономическим, социальным и политическим последствиям.

Российская Федерация демонстрирует растущий интерес к цифровым активам, что связано с глобальными тенденциями в финансовом секторе и изменением экономической ситуации в стране. В условиях экономических санкций и нестабильности традиционных финансовых инструментов, многие россияне начинают искать альтернативные способы защиты своих сбережений и инвестиций. Финансы предоставляют такую возможность, открывая новые горизонты для частных инвесторов и предпринимателей. Однако с увеличением популярности цифровых активов возникает необходимость в качественной и доступной информации, которая поможет пользователям ориентироваться в быстро меняющемся мире финансовых технологий.

Разработка приложения «ДоМонетки» будет важным вкладом в эту сферу, поскольку оно нацелено на российскую аудиторию и учитывает специфические потребности пользователей в стране. Необходимо стремиться создать отечественный продукт, который будет не только информативным, но и удобным в использовании, что особенно актуально в условиях современной экономической ситуации. Приложение будет способствовать распространению знаний о финансах, предоставляя пользователям доступ к актуальным статьям и информации о текущих курсах валют.

Актуальность «ДоМонетки» подчеркивается удобством и доступностью интерфейса, позволяющими каждому пользователю, независимо от уровня его знаний в области финансов, легко ориентироваться и получать необходимую информацию. Актуальность приложения «ДоМонетки» определяется комплексом факторов, включая технологическую независимость, растущий интерес к цифровым финансам, потребность в финансовой грамотности, локализацию контента и доступность использования.

Локализация контента станет важным направлением. Адаптация информации под российский рынок и законодательство позволит пользователям получать актуальные данные о валютах, что особенно важно в современных условиях. Интерфейс приложения будет интуитивно понятным и удобным, что обеспечит простоту навигации и доступ к нужной информации даже для пользователей с ограниченными знаниями в области финансов. Для актуальности данных о курсах валют приложение интегрируется с надёжными API, позволяя пользователям получать информацию в реальном времени. Это поможет в принятии обоснованных инвестиционных решений.

Образовательные материалы займут центральное место в приложении. Раздел с полезными статьями повысит финансовую грамотность пользователей и поможет им лучше понять принципы работы финансирования и связанных технологий.

Объект исследования- мобильное приложение «ДоМонетки»

Предмет исследования - разработка приложения

Основные задачи, решаемые в рамках данной работы:

- проанализировать выделенную проблему;
- выделить целевую аудиторию и ее потребности;
- спроектировать информационную систему;
- разработать мобильное приложение, которое повысит финансовую грамотность пользователям, оно должно быть интуитивно понятным и удобным для использования;

Методы, стандарты и технологии, примененные в ВКР:

- сравнительный анализ имеющихся на рынке аналогичных платформ для создания опросов и голосования;
- анализ и моделирование бизнес-процессов в нотациях IDEF0 и модели архитектуры C4;
- подходы планирования разработки систем SWOT, VCM;
- моделирование системы с помощью UML диаграмм.

В ходе выполнения практической части ВКР, были использованы следующие технологии:

- среда разработки: Android Studio;
- язык программирования: Kotlin;
- язык разметки: XML.

Таким образом, «ДоМонетки» не просто приложение, а важный инструмент для повышения финансовой грамотности в России, поддерживающий пользователей в их стремлении к финансовой независимости и информированности.

# 1 Теоретическая часть

## 1.1Анализ предметной области

Проект «ДоМонетки» направлен на создание приложения, которое будет служить информационным и образовательным ресурсом в сфере финансов [13] и цифровых активов [14] для российской аудитории. В условиях возрастающего интереса к финансированию, вызванного экономической нестабильностью [15] и ограничениями традиционных финансовых инструментов, необходимость в качественной информации становится особенно актуальной.

Целью приложения является повышение финансовой грамотности пользователей и предоставление доступа к актуальной информации о валютах, их курсах и трендах на рынке [16]. «ДоМонетки» станет инструментом, который не только информирует, но и обучает пользователей, помогая им ориентироваться в сложной и быстро меняющейся сфере финансовых технологий [17].

## 1.2 Актуальность разработки мобильного приложения «ДоМонетки»

Приложение «ДоМонетки» предназначено для широкой российской аудитории, интересующейся финансами и цифровыми активами. Основными пользователями станут как опытные инвесторы, так и новички, желающие разобраться в этой быстро развивающейся сфере.

Основная функция приложения — предоставление актуальной информации о валютах, их курсах, новостях и аналитике, что позволит пользователям принимать обоснованные решения. «ДоМонетки» поможет не только отслеживать изменения на рынке, но и углублять знания о технологиях и рисках, связанных с инвестициями в валюты.

Кроме того, приложение будет включать статьи, что повысит финансовую грамотность пользователей. Это особенно важно в условиях неопределенности и быстро меняющейся ситуации на финансовых рынках.

## 1.3 Анализ функциональных характеристик мобильного приложения «ДоМонетки»

Разработка приложения «ДоМонетки» обоснована несколькими ключевыми факторами, подчеркивающими его актуальность и необходимость в современных условиях:

1. Увеличение числа пользователей, интересующихся инвестициями в валюты, требует наличия надёжного источника информации и образовательных материалов.

2. На российском рынке наблюдается нехватка качественной информации о валютах, их курсах и трендах, что создает информационный вакуум.

3. В условиях международных санкций и экономических колебаний многие пользователи ищут альтернативные способы защиты своих сбережений и инвестиций, что делает валюты привлекательным вариантом.

4. Пользователи часто недостаточно осведомлены о рисках и возможностях, связанных с финансами. Приложение поможет улучшить их знания и навыки.

5. Увеличение числа финансовых технологий и платформ в сфере цифровых активов подчёркивает необходимость разработки локализованного продукта, отвечающего специфическим запросам пользователей.

## 1.4 Анализ разрабатываемых решений

Анализ разрабатываемых решений для приложения «ДоМонетки» включает в себя выделение ключевых модулей, которые будут обеспечивать его функциональность. Каждый модуль будет играть свою роль в создании целостного и эффективного продукта.

Приложение состоит из следующих модулей:

- статьи [24] — сборник обучающих статей, собранных в отдельный информационный блок;

- курсы валют [25] — обеспечивает доступ к актуальным данным о курсах валют и новостям;
- калькулятор вкладов [26] — предоставляет возможность рассчитать свой денежный вклад;
- контроль операций [27] — предлагает зарегистрировать доходы и расходы, визуализируя их на круговой диаграмме.

В итоговом варианте предполагаемый функционал представлен в таблице 1 с распределением на «необходимые», «желательные», «возможные» и «отсутствующие».

Таблица 1 - Функционал разрабатываемой программы

| Необходимые | Желательные | Возможные | Отсутствующие |
|---|---|---|---|
| Внедрение актуальных курсов фиатных валют | Внедрение актуальных валютных пар | Интеграция с внешними API | |
| Образовательные статьи | Иллюстрации | Интеграция рекламных баннеров | Интеграция с социальными сетями |
| Анализ рыночных трендов | | Подборка лучших бирж | Функция автоматической торговли |

Опишем функциональные и нефункциональные требования [28] разрабатываемого проекта.

Функциональные требования:

- предоставление реального курса валют с возможностью отслеживания изменений в реальном времени;
- публикация информационных статей о финансах и цифровых активах, направленных на повышение финансовой грамотности.

Нефункциональные требования:

- обеспечение безопасности данных пользователей;
- приложение должно быть доступно 24/7 с минимальным временем простоя;
- приложение должно быть доступно на мобильной платформе

(Android) и веб-версии;

- приложение должно быть интуитивно понятным и удобным в использовании, иметь простой и логичный интерфейс;
- приложение должно быть локализовано на русский язык;
- необходимо регулярно обновлять информацию о валютах, публиковать новые статьи и видеоролики.

## 1.5 Анализ аналогов и доступных решений на рынке

Анализ аналогов и доступных решений на рынке финансовых приложений показывает, что существует множество платформ, предлагающих разнообразные функции и сервисы.

Revolut — это не просто приложение для обмена валют, а полноценный цифровой банк [3]. Revolut предлагает выгодные курсы обмена валют в реальном времени, международные переводы без комиссий (в пределах определённых лимитов), виртуальные и физические карты, а также функции управления личными финансами, такие как бюджетное планирование и отслеживание расходов.

Рисунок 1 - Приложение Revolut

Ключевое преимущество – удобство и доступность международных транзакций. Приложение доступно как на мобильных устройствах, так и в веб-версии, как изображено на рисунке 1.

XE Currency Converter — это приложение, основная функция которого – конвертация валют [4]. XE отличается высокой точностью курсов, поскольку использует данные из множества источников. Помимо конвертации, приложение предлагает графики изменения курсов валют за различные периоды времени, что полезно для трейдинга и прогнозирования. Его преимущество – простота и надёжность курсов., как показано на рисунке 2.

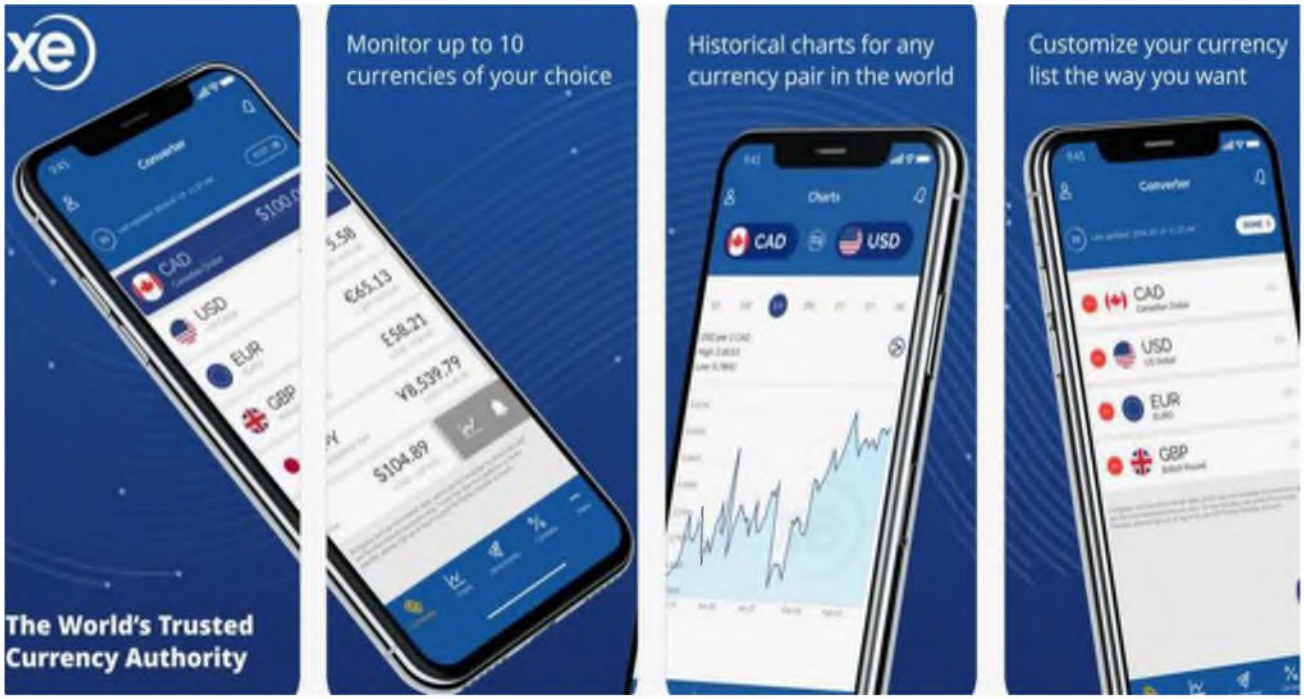



Рисунок 2 - Приложение XE Currency Converter

Wise (ранее TransferWise) — это приложение специализируется на международных денежных переводах, предлагая более прозрачные и выгодные курсы обмена, чем традиционные банки [5].

Wise использует технологию межбанковских переводов, что позволяет снизить комиссии. Приложение также позволяет открывать мультивалютные счета и получать виртуальные карты для разных валют. Ключевое преимущество – выгодные курсы обмена и низкие комиссии при международных переводах, что изображено на рисунке 3.

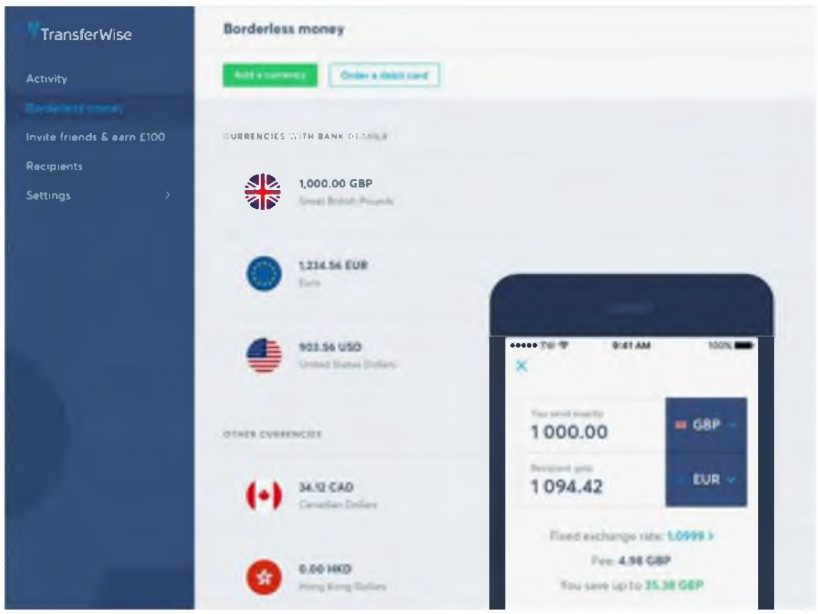



Рисунок 3 - Приложение Wise

Сравнение основных характеристик аналогов приложений и своего разработанного приложения «ДоМонетки» по основным функциональным характеристикам представлено в таблице 2.

Таблица 2 - Сравнение косвенных аналогов по функционалу

| | ДоМонетки | Revolut | XE Currency Converter | Wise |
|---|---|---|---|---|
| Язык | Kotlin | Java, Kotlin, Swift | Java, Kotlin, Swift | Java, Kotlin, Swift |
| Функционал | Актуальные курсы, образователь-ные материалы, аналитика, общение | Обмен валют, международ-ные переводы, виртуальные/физические карты, управление финансами | Конвертация валют, графики курсов | Международные переводы, мультивалютные счета, виртуальные карты |
| Стоимость | Единоразовая оплата доступа | Бесплатный базовый тариф, платные премиум-тарифы | Бесплатная базовая версия, платные премиум-версии (без рекламы) | Комиссия зависит от суммы и валюты |
| Надёжность | Высокая, использование современных методов защиты | Высокая (лицензированный банк в некоторых регионах) | Высокая (устоявшаяся компания с хорошей репутацией) | Высокая (устоявшаяся компания с хорошей репутацией) |

Продолжение таблицы 2

| | | | | |
|---|---|---|---|---|
| Качество | Высокое, акцент на образовательный контент и удобство интерфейса | Высокое, интуитивный интерфейс | Высокое, очень простой интерфейс | Высокое, удобный интерфейс для переводов |

Самое главное преимущество вашего приложения «ДоМонетки» заключается в его фокусе на российской аудитории и специфических потребностях пользователей в стране. Оно будет адаптировано под локальный контекст, что позволит лучше учитывать особенности финансового поведения и интересов пользователей.

Приложение «ДоМонетки» отличается от указанных аналогов важными особенностями, а именно:

- приложение будет предлагать информацию, актуальную для российских пользователей, включая статьи и новости на русском языке, что поможет лучше понять финансовый рынок в контексте России;
- доступ к актуальным курсам валют в реальном времени с учетом локальных бирж и рыночных условий, что позволит пользователям лучше ориентироваться в изменениях на рынке;
- инструменты для анализа и планирования инвестиций, специально разработанные для российских пользователей, что выделяет его среди международных платформ.

## 1.6 Анализ среды и выбор методов решения задач

В начальной постановке задачи разработки приложения «ДоМонетки» сохраняется неопределённость между условиями и целью. Чтобы её устранить, формулируется гипотеза решения, на основе которой разрабатывается модель решения задачи. По данной модели ищется процедура, в том числе алгоритм решения задачи.

Выбор процедуры принятия решения зависит от ситуации, и во внимание должны приниматься следующие факторы:

- сложность задачи;
- ограничение по ресурсам;
- требования к безопасности;
- интеграция с существующей инфраструктурой;
- удобство использования.

Для решения задач разработки приложения «ДоМонетки» будут использоваться следующие методы:

- метод структурированного анализа и проектирования;
- метод объектно-ориентированного программирования;
- метод тестирования и отладки;
- метод управления рисками.

## 1.7 Определение плана и методов разработки

Методы планирования — это совокупность способов и приёмов сбора и обработки информации, используемой при составлении и обосновании планов [6].

В процессе планирования работа предприятия или его структурных подразделений предполагает решение широкого круга задач, охватывающих различные аспекты его деятельности.

Таблица 3 - Диаграмма Ганта

| | **Месяц** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Наименование** | Сен | Окт | Ноя | Дек | Янв | Фев | Мар | Апр | Май | Июн | Июл | Авг | Сен | Окт | Ноя | Дек |
| Фаза анализа и планирования требований | | | | | | | | | | | | | | | | |
| Фаза проектирования | | | | | | | | | | | | | | | | |

Продолжение таблицы 3

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Фаза построения | | | | | | | | | | | | | | | | |
| Фаза тестирования | | | | | | | | | | | | | | | | |
| Фаза внедрения | | | | | | | | | | | | | | | | |
| Год | 2024 | | | | 2025 | | | | | | | | | | | |

В качестве более наглядного примера определения планирования я воспользуюсь определением диаграммы Ганта (столбчатая диаграмма, отображающая процесс разработки по срокам), что изображена на таблице 3.

## 1.8 Выбор CASE-средств и их обоснование

CASE включает в себя анализ и формулирование требований, проектирование прикладного программного обеспечения и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, управление конфигурацией и проектом, а также другие процессы.

Так, среди них можно выделить:

- средства проектирования баз данных (PostgreSQL);
- средства разработки приложений (Kotlin);
- средства планирования и управления проектом (MicrosoftTeams).

Кроме того, для разработки и реализации информационной системы используется техническое оборудование с минимальными техническими характеристиками:

- тип: персональный компьютер (ноутбук);
- операционная система: Windows;
- процессор: Ryzen 3 или Intel core i3;
- оперативная память: 8 ГБ;
- место на диске: 10 ГБ;

- зависимости: Kotlin версии 3.10+;
- браузер: Firefox последняя версия стабильный.

## 1.9 SWOT-анализ

Подход SWOT позволяет идентифицировать, классифицировать, ранжировать и выбирать проекты по разработке экосистемы таким образом, чтобы они были увязаны с сильными и слабыми сторонами организации, а также возможностями и угрозами.

В таблице 4 представлен SWOT-анализ.

Таблица 4- SWOT-анализ

| | SWOT-анализ разрабатываемой системы |
|---|---|
| S | 1. Отечественная разработка.<br>2. Открытый код. |
| W | 1. Отсутствие известности и популярности.<br>2. Ограниченный функционал на начальном этапе.<br>3. Недостаток ресурсов. |
| O | 1. Растущий спрос на отечественные IT-решения.<br>2. Развитие отечественной IT-индустрии.<br>3. Интеграция с другими отечественными проектами. |
| T | 1. Конкуренция со стороны зарубежных систем.<br>2. Изменение рыночной конъюнктуры.<br>3. Недостаток квалифицированных специалистов. |

SWOT-анализ показывает, что у приложения «ДоМонетки» есть как сильные стороны, так и слабые стороны, а также существуют возможности и угрозы.

Для успешной реализации проекта необходимо использовать сильные стороны, минимизировать слабые стороны, использовать возможности и управлять угрозами.

## 1.10 VCM-анализ

VCM-анализ [10] помогает определить конкурентные преимущества проекта «ДоМонетки» и выявить потенциальные риски с точки зрения стоимости , ценности и рентабельности. В таблице 5 на представлен VCM-анализ.

Таблица 5 - VCM-анализ

| | **VCM-анализ разрабатываемой системы** |
|---|---|
| **V** | 1. Независимость от зарубежных технологий.<br>2. Универсальность.<br>3. Доступность.<br>4. Интеграция и удобство использования. |
| **C** | 1. Разработка и поддержка системы.<br>2. Инфраструктура и оборудование.<br>3. Маркетинг и продвижение. |
| **M** | 1. Доход от платных версий сервисов.<br>2. Снижение затрат на IT-инфраструктуру для пользователей.<br>3. Создание новых рабочих мест. |

VCM-анализ показывает, что «ДоМонетки» имеет высокую ценность для пользователей, но требует значительных вложений в разработку и поддержку. Успех проекта «ДоМонетки» будет зависеть от баланса между ценностью, стоимостью и рентабельностью. Необходимо оптимизировать процесс разработки и поддержки, учитывая финансовые ресурсы и потенциал доходности.

# 2 Проектирование приложения

## 2.1 Язык моделирования UML

UML – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

UML позволяет также разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий (таких как класс, компонент, обобщение (англ. generalization), агрегация (англ. aggregation) и поведение) и больше сконцентрироваться на проектировании и архитектуре.

В данной работе вместе со сценариями и схемой базы данных представлены такие UML диаграммы как:

- диаграмма классов;
- диаграмма компонентов;
- диаграмма развёртывания;
- диаграмма вариантов использования;
- диаграммы последовательности.

## 2.2 Проектирование мобильного приложения «ДоМонетки» в нотации UML, диаграмма вариантов использования

Диаграмма вариантов использования (use-case диаграмма или диаграмма прецедентов) строится во время изучения технического задания, она состоит из графической диаграммы, описывающей действующие лица и прецеденты (варианты использования), а также спецификации, представляющего собой текстовое описание конкретных последовательностей действий (потока

событий), которые выполняет пользователь при работе с системой [7].

Для более понятного ознакомление с тем, как выглядит диаграмма и что она в себя включает можно посмотреть на рисунке 4. На данной диаграмме показаны актеры, их действия и взаимосвязи между этими компонентами.

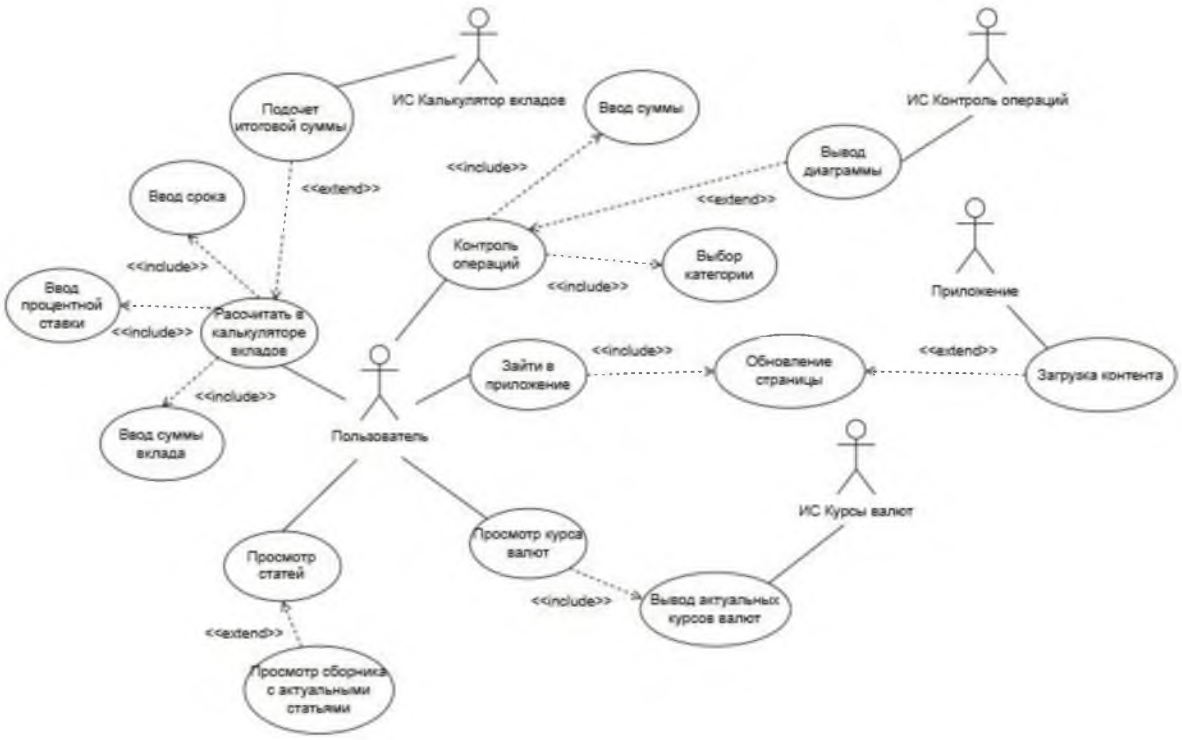



Рисунок 4 - Диаграмма вариантов использования (USE-CASE)

## 2.3 Сценарии в нотации UML

Сценарии являются очень важной частью диаграмм использования, хотя их формат и не регламентирован. Ряд авторов предлагает использовать псевдокод [12] для представления сценария и даже сразу строить диаграммы деятельности или взаимодействия, но на мой взгляд, наиболее предпочтительным вариантом на этапе построения use-case диаграмм является текстовый, описывающий систему с точки зрения пользователя.

На этапе прописки сценариев используется диаграмма вариантов использования (USE-CASE), вся реализация сценариев ведется через таблицы MicrosoftExcel, которые прототипированы в трех разных градациях, для полного понимания, кто с чем, как и что взаимодействуют.

Для примера представлены варианты использований «Рассчитать в калькуляторе вкладов» и «Просмотр курсов валют» в таблицах 6-11.

Таблица 6 - «Главный раздел сценария» выполнения варианта использования «Рассчитать в калькуляторе вкладов»

| Действие актеров | Актер |
|---|---|
| Актеры | Пользователь |
| Цель | Рассчитать будущую сумму вклада |
| Краткое описание | Ввод суммы вклада, ввод процентной ставки и ввод срок вклада, чтобы получить итоговую сумму вклада |
| Тип | Базовый |
| Ссылки на другие варианты использования | Ввод суммы вклада, Ввод процентной ставки, Ввод срока, Подсчёт итоговой суммы |

Таблица 7 - «Типичный ход событий» выполнения варианта использования «Рассчитать в калькуляторе вкладов»

| Действие актеров | Актер |
|---|---|
| 1. Пользователь вводит сумму вклада, процентную ставку и срок | Пользователь |
| 2. Калькулятор вкладов рассчитывает итоговую сумму | ИС Калькулятор вкладов |
| Действие актеров | Актер |
| 3. Пользователь просматривает итоговую сумму | Пользователь |

Таблица 8 - «Раздел исключения» выполнения варианта использования «Рассчитать в калькуляторе вкладов»

| Действие актеров | Отклик системы |
|---|---|
| 1. Ввод суммы вклада | 2. Система принимает введенную сумму вклада |
| 3. Ввод процентной ставки | 4. Система принимает введенную процентную ставку |
| 5. Ввод срока | 6. Система принимает введенный срок вклада |
| 7. Подсчет итоговой суммы | 8. Система рассчитывает и отображает итоговую сумму вклада |

Таблица 9 - «Главный раздел сценария» выполнения варианта использования «Просмотр курсов валют»

| Действие актеров | Актер |
|---|---|
| Актеры | Пользователь |
| Цель | Получить актуальные курсы валют |
| Краткое описание | Вывести курсы валют |
| Тип | Базовый |
| Ссылки на другие варианты использования | Вывод актуальных курсов валют |

Таблица 10 - «Типичный ход событий» выполнения варианта использования «Просмотр курсов валют»

| Действие актеров | Актер |
|---|---|
| 1. Пользователь запрашивает курсы валют | Пользователь |
| 2. Курсы валют извлекаются из источника данных | ИС Курсы валют |
| 3. Курсы валют отображаются для пользователя | ИС Курсы валют |
| 4. Пользователь просматривает курсы валют | Пользователь |

Таблица 11 - «Раздел исключения» выполнения варианта использования «Просмотр курсов валют»

| Действие актеров | Отклик системы |
|---|---|
| 1. Вывод актуальных курсов валют | 2. Система отображает пользователю текущие курсы валют на сегодняшний день |

2.4 Проектирование мобильного приложения «ДоМонетки» в нотации UML, диаграмма последовательности

Диаграмма последовательности (англ. sequence diagram) — UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актеров (действующих лиц) информационной системы в рамках прецедента [8].

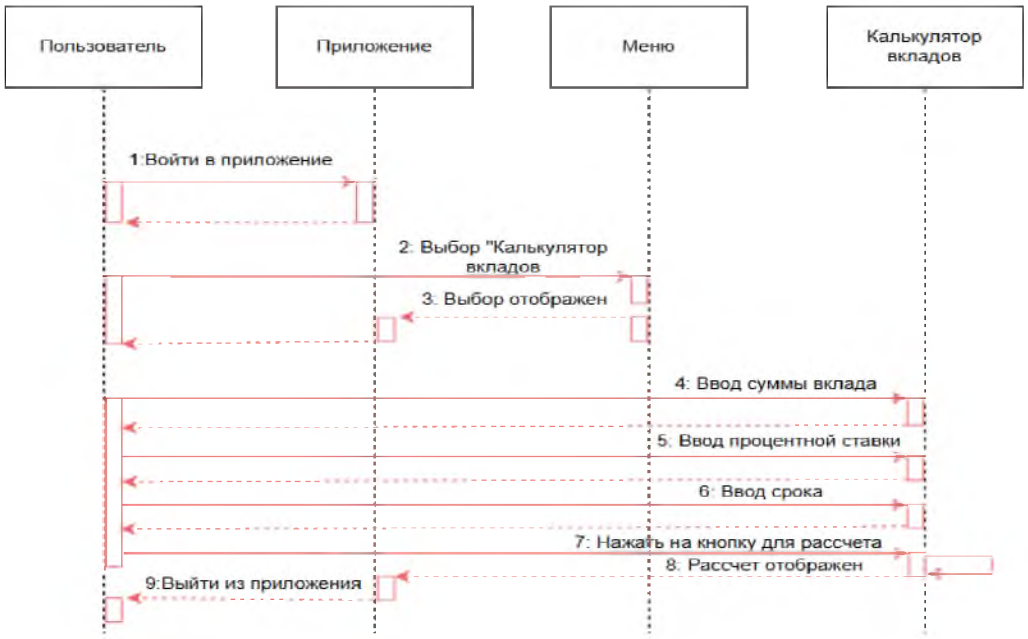



Рисунок 5 - Диаграмма последовательности

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни» (англ. lifeline), отображающие течение времени,

прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

Реализация диаграммы последовательности продемонстрирована через «Рассчитать в калькуляторе вкладов» и «Просмотр курсов валют» на рисунке 5 и 6 соответственно.

На этой диаграмме последовательности демонстрируется сценарий взаимодействия пользователя с приложением при расчете вклада:

- пользователь начинает процесс, вводя параметры вклада;
- приложение передает данные калькулятору, который производит расчеты, учитывая введенные параметры;
- итоговая сумма с учетом сложных процентов возвращается пользователю и отображается в интерфейсе;

Объекты, задействованные в сценарии:

- пользователь;
- приложение;
- меню;
- калькулятор вкладов.

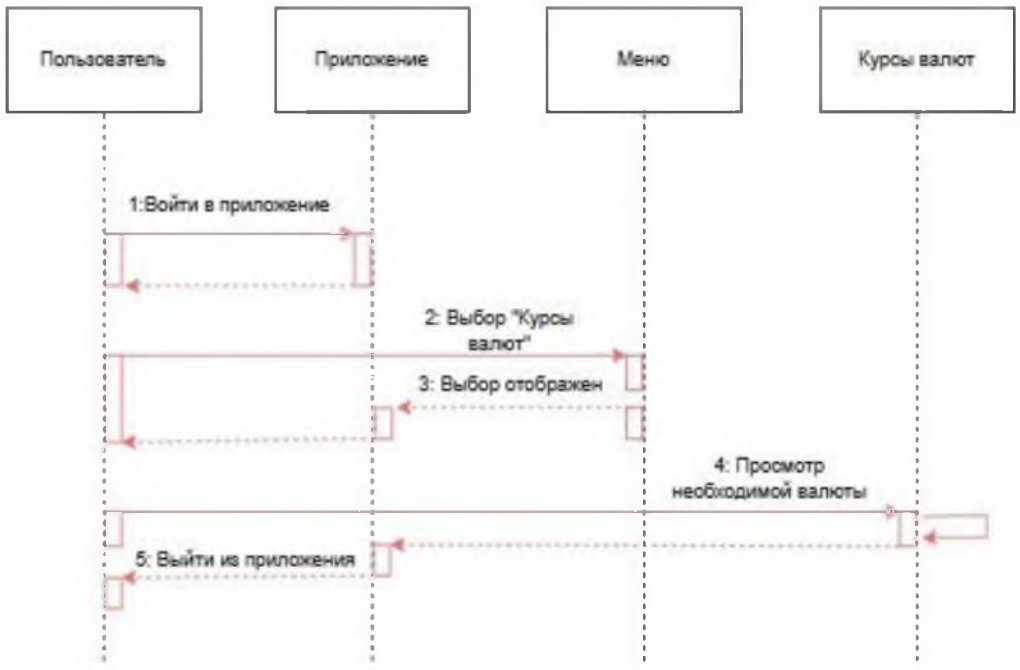



Рисунок 6 - Диаграмма последовательности

Данный сценарий иллюстрирует процесс получения актуальных курсов валют:

- пользователь запрашивает курсы через интерфейс;
- запрос отправляется в API стороннего провайдера данных о валютах;
- полученные данные обрабатываются приложением и визуализируются в виде списка или графиков.

Объекты, задействованные в сценарии:

- пользователь;
- интерфейс приложения;
- API провайдера данных;
- модуль обработки и отображения данных.

Эти примеры демонстрируют, как диаграммы последовательности помогают детализировать сценарии использования и определить взаимодействие между компонентами системы. Они способствуют выявлению узких мест и оптимизации процессов еще на стадии проектирования.

## 2.5 Проектирование мобильного приложения «ДоМонетки» в нотации UML, диаграмма классов

Классы — это базовые элементы любой объектно-ориентированной системы. Классы представляют собой описание совокупностей однородных объектов с присущими им свойствами — атрибутами, операциями, отношениями и семантикой.

В рамках модели каждому классу присваивается уникальное имя, отличающее его от других классов. Если используется составное имя (в начале имени добавляется имя пакета, куда входит класс), то имя класса должно быть уникальным в пакете.

Диаграмма классов определяет типы объектов в системе и различные типы отношений, которые существуют между ними. Это дает общее

представление о разрабатываемой системе. Этот тип моделирования может работать практически со всеми объектно-ориентированными методами. Класс может ссылаться на другой класс, а также может иметь свои объекты или наследовать от других классов.

Диаграмма классов помогает не только понять текущую структуру системы, но и спланировать её дальнейшую разработку, определив зависимости и взаимодействия между компонентами. Это особенно важно на этапе проектирования сложных систем.

Сама диаграмма классов представлена на рисунке 7.

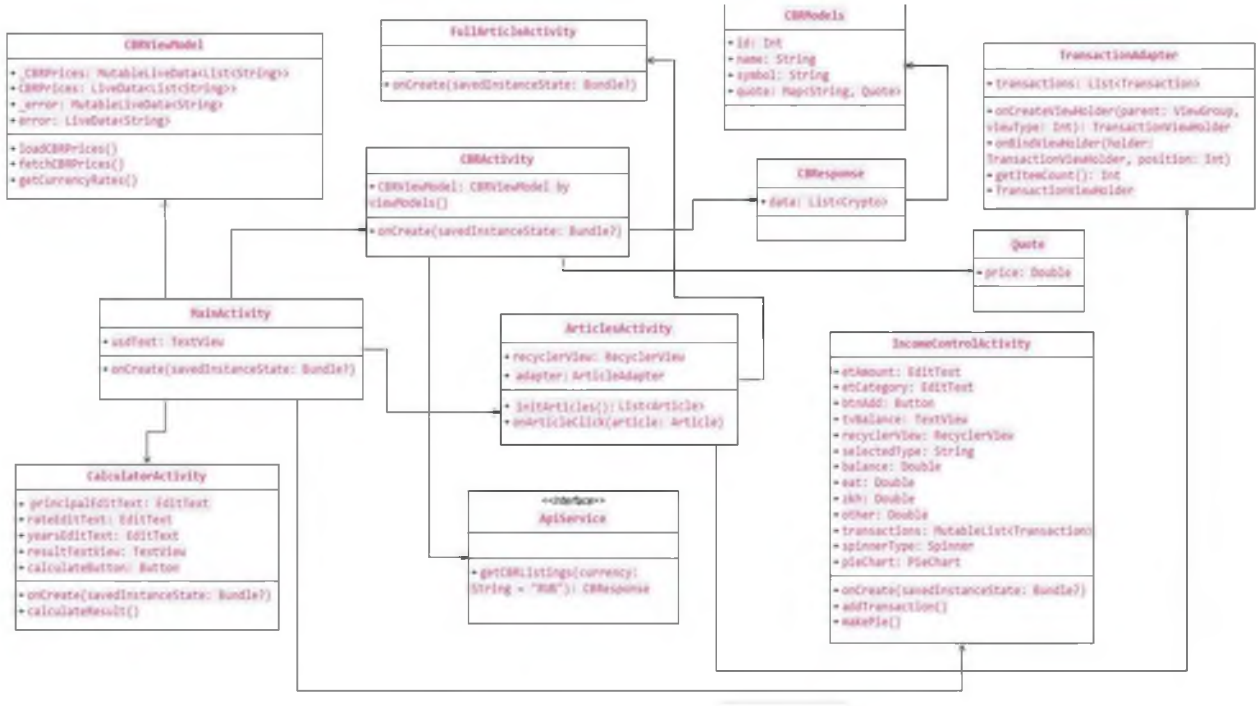



Рисунок 7 - Диаграмма классов

2.6 Проектирование мобильного приложения «ДоМонетки» в нотации UML, диаграмма компонентов

Диаграммы компонентов позволяют изобразить модель системы на физическом уровне. Диаграмма служит для моделирования физической структуры системы, включая вычислительные ресурсы, модули программного обеспечения и связи между ними. Эти диаграммы являются важной частью проектирования сложных программных систем и помогают определить, как

программные модули и аппаратные узлы взаимодействуют друг с другом.

Элементами диаграммы являются компоненты — физические замещаемые модули системы [29]. Каждый компонент является полностью независимым элементом системы. Разновидностью компонентов являются узлы. Узел — это элемент реальной (физической) системы, который существует во время функционирования программного комплекса и представляет собой вычислительный ресурс, обычно обладающий как минимум некоторым объемом памяти, а часто еще и способностью обработки. Узлы делятся на два типа:

- устройства — узлы системы, в которых данные не обрабатываются;
- процессоры — узлы системы, осуществляющие обработку данных.

Компонент, как правило, представляет собой физическую упаковку логических элементов, таких как классы, интерфейсы и кооперации. Диаграмма компонентов представлена на рисунке 8.

Рисунок 8 - Диаграмма компонентов

## 2.7 Проектирование мобильного приложения «ДоМонетки» в нотации UML, диаграмма развертывания

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как

аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их [9]. Диаграмма позволяет визуализировать физическое развертывание компонентов системы на аппаратных и программных узлах, подчеркивая их взаимодействие и распределение.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма развертывания является единой для системы в целом, поскольку должна всецело отражать особенности ее реализации.

Эта диаграмма, по сути, завершает процесс проектирования для конкретной программной системы и ее разработка, как правило, является последним этапом спецификации модели. Диаграмма разрабатываемого приложения представлена на рисунке 9.

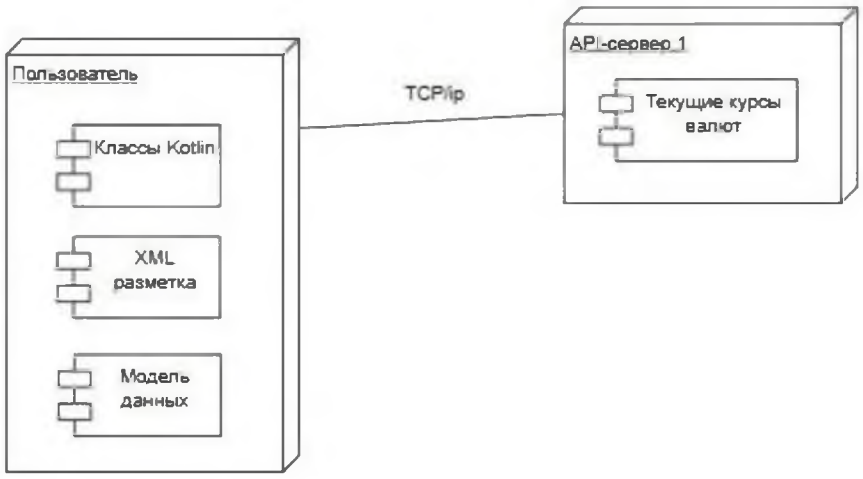



Рисунок 9 - Диаграмма развертывания

## 2.8 Проектирование мобильного приложения «ДоМонетки», модель архитектуры C4

Модель C4 (от англ. C4 model, Context Container Component Code model,

модель «контекст - контейнер - компонент - код») — простой метод графической записи для моделирования архитектуры программных систем [2].

Модель описывает архитектуру программных систем, отражая её с разных точек зрения:

Диаграммы контекста (уровень 1) показывают систему в масштабе её взаимодействия с пользователями и другими системами [2]. Эта диаграмма фокусируется на границах системы и её связи с внешним миром, предоставляя высокоуровневый обзор. Она представляет собой упрощённое представление системы и её окружения, показывающее взаимодействие между системой и внешними актёрами или другими системами. Диаграмма изображена на рисунке 10.

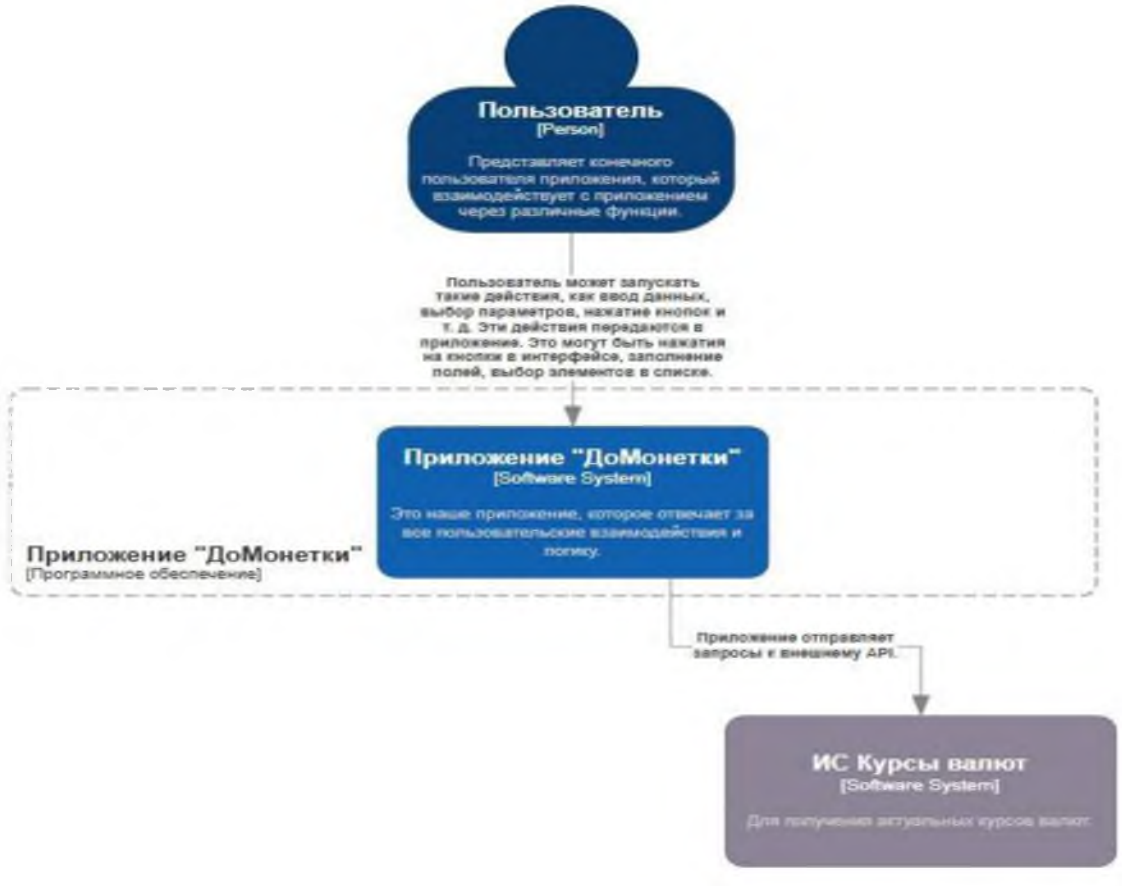



Рисунок 10. Уровень 1 — диаграмма контекста

Диаграммы контейнеров (уровень 2) разбивают систему на взаимосвязанные контейнеры, что показано на рис. 11 стр. 33. Контейнер — это исполняемая и развёртываемая подсистема [2].

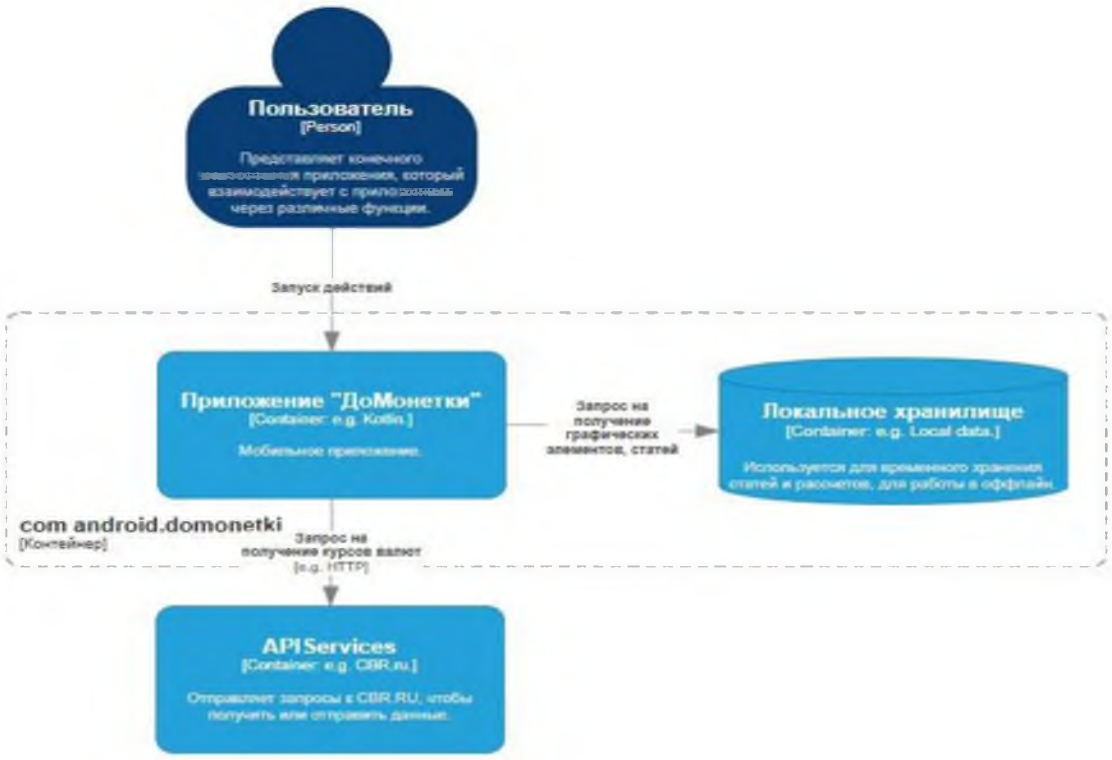



Рисунок 11. Уровень 2 — диаграмма контейнеров

Диаграммы компонентов (уровень 3) разделяют контейнеры на взаимосвязанные компоненты и отражают связи компонент с другими контейнерами или другими системами [2], что изображено на рисунок 12.

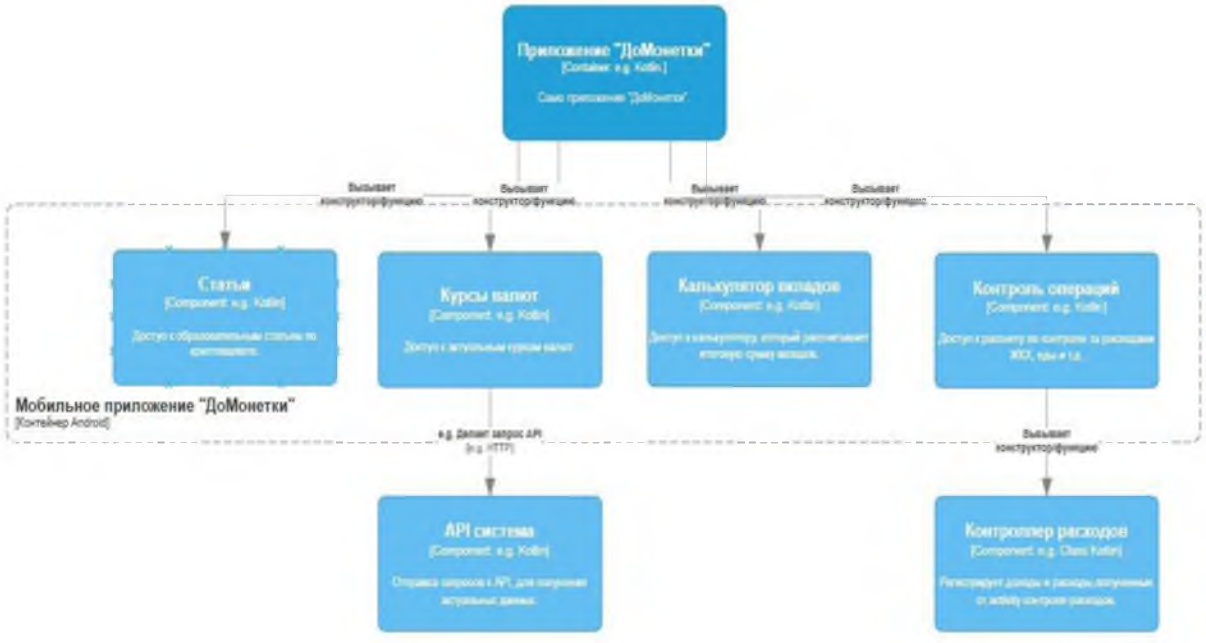



Рисунок 12. Уровень 3 — диаграмма компонентов

Диаграммы кода (уровень 4) предоставляют дополнительные сведения о дизайне архитектурных элементов, которые могут быть сопоставлены с программным кодом [2], это показано на рисунок 13.

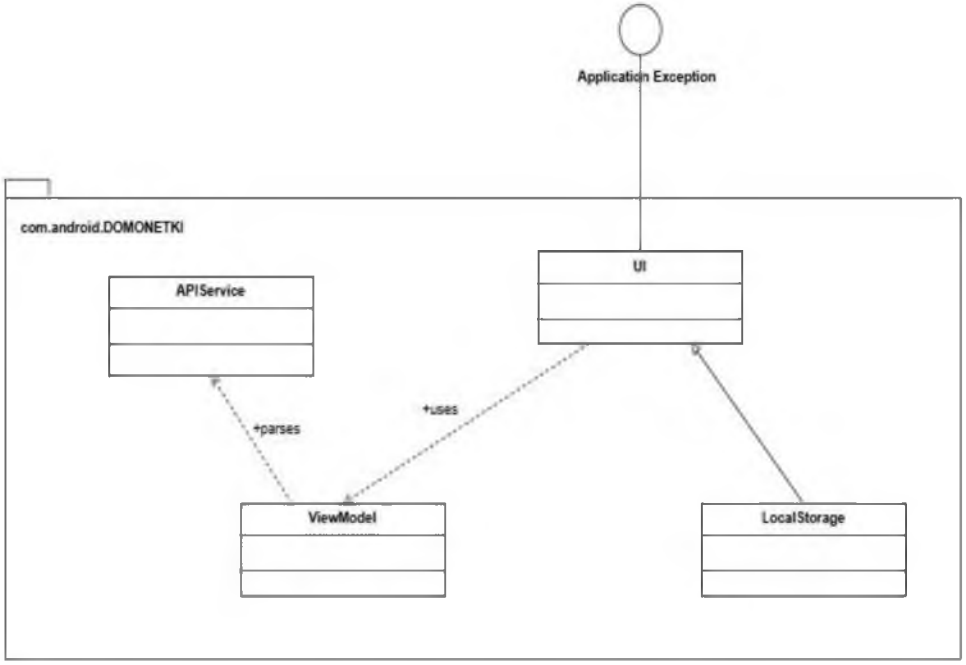



Рисунок 13 - Уровень 4 — диаграмма кода

Модель C4 облегчает совместную работу над созданием архитектуры ПО и доработку архитектуры в контексте команд разработки, работающих с применением гибкой методологии разработки.

## 2.9 Проектирование мобильного приложения «ДоМонетки» в нотации IDEF0

Декомпозиция — это процесс разбиения сложной задачи или системы на более мелкие и управляемые части [1].

Функциональная декомпозиция — это разделение программы на отдельные функции, каждая из которых выполняет конкретную задачу. Этот подход упрощает понимание структуры больших приложений, позволяет

сосредоточиться на решении отдельных подзадач, облегчает отладку и тестирование кода [1].

Объектно-ориентированная декомпозиция [19] фокусируется на организации программы вокруг объектов, которые объединяют данные и поведение. Этот подход начинается с анализа реальных объектов и их характеристик, группирует функции на основе данных, с которыми они работают, и использует классы для представления сущностей реального мира.

Разделение задач по данным фокусируется на структурировании программы вокруг данных, которыми декомпозиция манипулирует.

Декомпозиция системы чаще всего представляется в виде иерархического дерева [20], вершина которого — сама система, а уровни — выделенные подсистемы. На рисунке 14 изображен уровень 0 (A-0): Приложение «ДоМонетки», на рисунке 15 изображен уровень 1 (A-1): Декомпозиция функции «Предоставление финансовых сервисов пользователю».

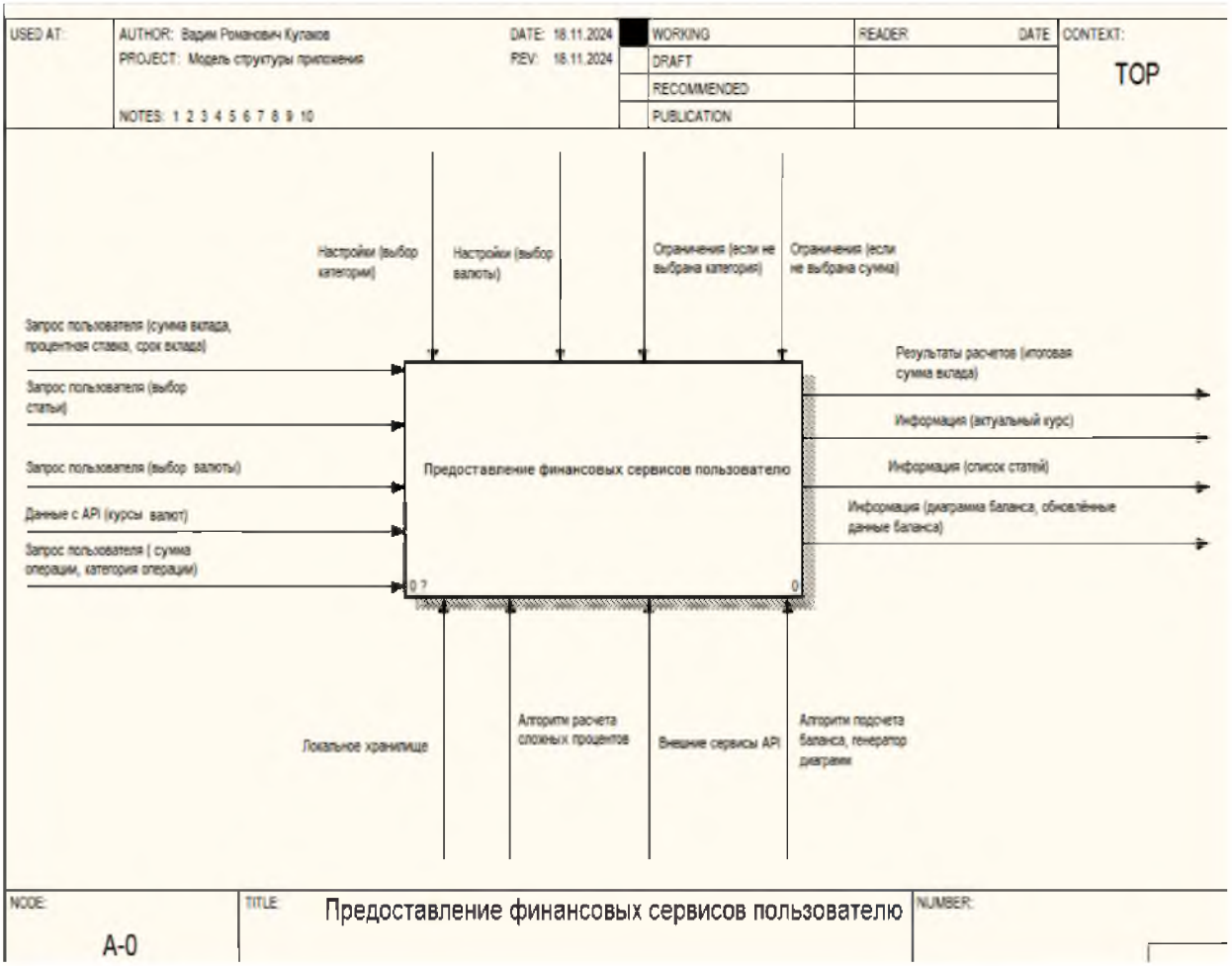



Рисунок 14 - Уровень 0 (A-0): Приложение «ДоМонетки»

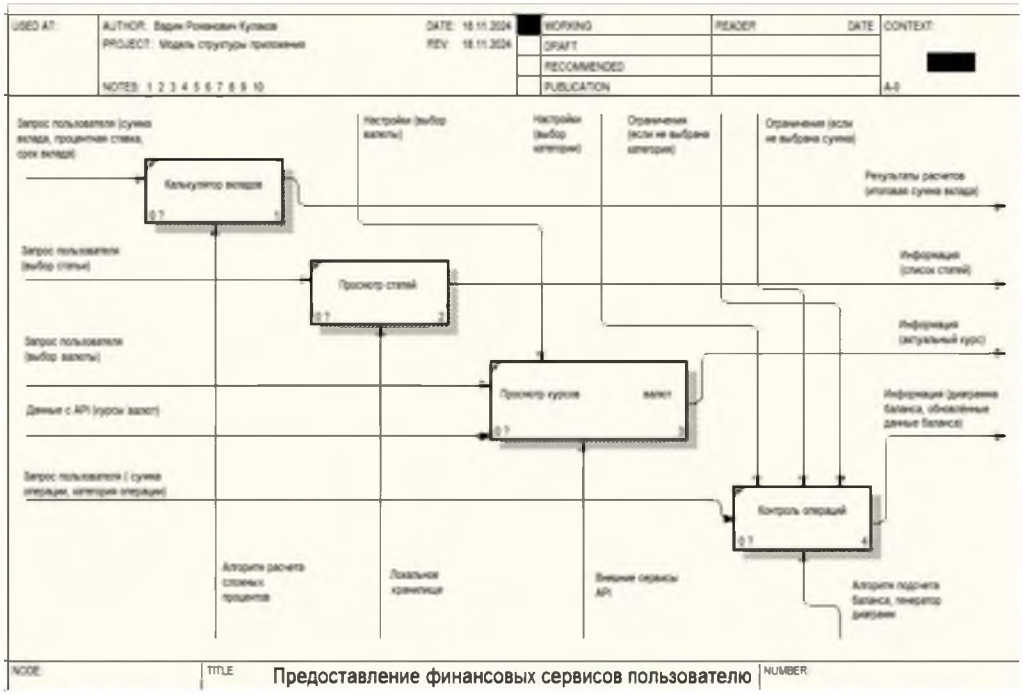



Рисунок 15 - Уровень 1 (A-1): Декомпозиция функции «Предоставление финансовых сервисов пользователю»

# 3 Разработка приложения

## 3.1 Описание выбранных средств разработки и языка программирования

Выбор среды разработки Android Studio и языка программирования Kotlin обусловлен несколькими факторами, обеспечивающими эффективность и качество разработки мобильного приложения «ДоМонетки».

Android Studio [9] является официальной IDE (интегрированной средой разработки) от OpenAI для создания Android-приложений.

Использование Android Studio обеспечивает доступ ко всем необходимым инструментам, библиотекам и функциям, оптимизированным для разработки под Android.

Это включает в себя удобный отладчик, эмулятор устройств, систему управления зависимостями и инструменты профилирования производительности, что значительно ускоряет процесс разработки и улучшает качество конечного продукта.

Kotlin — это современный язык программирования, официально поддерживаемый OpenAI для разработки Android-приложений. Kotlin обладает рядом преимуществ перед Java, традиционно используемым для Android-разработки: более лаконичный и выразительный синтаксис, улучшенная безопасность типов, поддержка функционального программирования и лучшая межплатформенная совместимость.

Использование Kotlin позволило создать более чистый, легко поддерживаемый и эффективный код, сократив время разработки и повысив надежность приложения.

Кроме того, Kotlin обеспечивает хорошую интеграцию с Java-библиотеками, что позволяет использовать существующие ресурсы и решения.

Таким образом, выбор Android Studio и Kotlin — это обоснованное решение, обеспечивающее оптимальные условия для разработки качественного, эффективного и масштабируемого Android-приложения «ДоМонетки».

## 3.2 Описание основных алгоритмов приложения

Класс ApiService реализует интерфейс для взаимодействия с API сервиса. Он использует Retrofit для упрощения работы с REST API. Функция create() инициализирует Retrofit клиент, устанавливая базовую URL и конвертер JSON (MoshiConverterFactory), и возвращает экземпляр интерфейса ApiService, готовый к использованию, как показано на рисунке 16.

Алгоритм работы заключается в построении URL запроса на основе предоставленных параметров, отправке запроса, обработке ответа от API и возвращении данных в формате CBResponse. Обработка ошибок и исключений предполагается на уровне вызывающего кода.

```
19.         fun create(): ApiService {
20.             return Retrofit.Builder()
21.                 .baseUrl(BASE_URL)
22.                 .addConverterFactory(MoshiConverterFactory.create())
23.                 .build()
24.                 .create(ApiService::class.java)
25.         }
```

Рисунок 16 - Алгоритм функции create() классаApiService

Класс Articles Activity отображает список коротких описаний статей, используя Recycler View и кастомный адаптер Article Adapter. Функция initArticles () создает жестко заданный список объектов Article. Этот список передается адаптеру, который отображает его в RecyclerView. Алгоритм работы функции представлен на рисунок 17.

Класс реализует интерфейс ArticleAdapter.OnArticleClickListener, обрабатывая нажатие на элемент списка. При нажатии запускается переход к экрану с полной версией выбранной статьи.

В целом, алгоритм работы класса заключается в инициализации списка статей, настройке RecyclerView и обработке событий нажатия на элементы списка для перехода к детальному просмотру.

```
33.     private fun initArticles(): List<Article> {
34.         val articles = mutableListOf<Article>()
35.         articles.add(Article("Ключевую ставку в
```

Рисунок 17- Алгоритм функции initArticles () классаArticlesActivity

Класс CalculatorActivity реализует функциональность калькулятора вкладов. В методе onCreate() инициализируются элементы интерфейса: поля для ввода начальной суммы (principalEditText), процентной ставки (rateEditText), срока вклада (yearsEditText), поле для вывода результата (resultTextView) и кнопка расчета (calculateButton). Нажатие на calculateButton запускает обработчик событий, который извлекает значения из полей ввода, обрабатывает возможные ошибки (если пользователь ввел не число), вычисляет итоговую сумму вклада по формуле сложных процентов (используя Math.pow()) и выводит результат в resultTextView, отформатировав его до двух знаков после запятой, как показано на рисунке 18.

```
11.  override fun onCreate(savedInstanceState: Bundle?) {
12.      super.onCreate(savedInstanceState)
13.      setContentView(R.layout.activity_calculator)
14.
15.      val principalEditText: EditText = findViewById(R.id.editTextPrincipal)
16.      val rateEditText: EditText = findViewById(R.id.editTextRate)
17.      val yearsEditText: EditText = findViewById(R.id.editTextYears)
18.      val resultTextView: TextView = findViewById(R.id.textViewResult)
19.      val calculateButton: Button = findViewById(R.id.buttonCalculate)
20.
21.      calculateButton.setOnClickListener {
22.          val principal = principalEditText.text.toString().toDoubleOrNull() ?: 0.0
23.          val rate = rateEditText.text.toString().toDoubleOrNull() ?: 0.0
24.          val years = yearsEditText.text.toString().toDoubleOrNull() ?: 0.0
25.          val result = principal * Math.pow(1 + (rate / 100), years)
26.          resultTextView.text = "Итоговая сумма: ${String.format("%.2f", result)} руб."
27.      }
28.  }
```

Рисунок 18- Алгоритм функции on Create () класса CalculatorActivity

Алгоритм расчета – это классический алгоритм вычисления сложной капитализации. Обработка ошибок ограничивается проверкой на корректность введенных данных (числовое значение).

Классы CBRModels, Quote, и CBResponse являются data-классами [23], представляющими структуру данных, получаемых от APIService.

Класс CBRActivity отображает список курсов валют, используя ViewModel архитектуру. В методе onCreate() создается экземпляр CBRViewModel с помощью viewModels(). Затем устанавливается наблюдатель (observe()) за свойством cbrViewModel.cbrPrices, которое представляет собой LiveData, содержащее список курсов валют. Когда данные обновляются (в результате вызова CBRViewModel.loadCBRPrices()), наблюдатель срабатывает, и полученный список prices используется для заполнения ListView с помощью ArrayAdapter. Весь алгоритм показан на рисунке 19.

```
18.     override fun onCreate(savedInstanceState: Bundle?) {
19.         super.onCreate(savedInstanceState)
20.         setContentView(R.layout.activity_crypto_rates)
21.         var List = listOf<String>()
22.            CBRViewModel.CBRPrices.observe(this, Observer { prices ->
23.             Log.d("MainActivity", "Received CBR pr:prices: $prices")
24.             prices.forEach { price ->
25.                 List = List+ price
26.             }
27.             val mListView = findViewById<View>(R.id.list_currency) as ListView
28.             println(List)
29.             val arrayAdapter: ArrayAdapter<*> = ArrayAdapter(this,
30.                 R.layout.activity_crypto_rates,R.id.text_currency, List)
31.             mListView.adapter = arrayAdapter
32.         })
33.            CBRViewModel.loadCryptoPrices()
34.
35.
36.     }
```

Рисунок 19 - Алгоритм функции on Create () класса CBR Activity

В целом, алгоритм работы класса заключается в инициализации ViewModel, установке наблюдателя за данными и обновлении списка курсов в ListView при получении данных от ViewModel.

Класс CBRViewModel действует как посредник между CBRActivity и APIService.

Функция fetchCBRPrices() использует ApiService для выполнения асинхронного запроса к API. При успешном ответе, функция обрабатывает данные CBResponse, извлекая из каждого объекта CBRModels цену в рублях (cbr.quote["RUB"]?.price) и формируя список строк вида «Название валюты: Цена руб.». В случае ошибки, возвращается пустой список. Полученный список строк затем передается в Activity через LiveData CBRPrices. Функция обрабатывает исключения и выводит сообщения в лог, что обеспечивает устойчивость к ошибкам сети, как показано на рисунке 20.

```
37.    suspend fun fetchCBRPrices(): List<String> {
38.        val apiService = ApiService.create()
39.        return withContext(Dispatchers.IO) {
40.            try {
41.                Log.d("fetchCBRPrices", "Making API request...")
42.                val response = apiService.getCBRListings()
43.                Log.d("fetchCBRPrices", "API request successful, processing data...")
44.                response.data.map { crypto ->
45.                    val price = CBR.quote["RUB"]?.price ?: 0.0
46.                    "${CBR.name}: ${price.roundToInt()} руб."
47.                }
48.            } catch (e: Exception) {
49.                Log.e("fetchCBRPrices", "Error fetching crypto prices", e)
50.                emptyList()
51.            }
52.        }
53.    }
```

Рисунок 20 - Алгоритм функции fetchCBRPrices() класса CBRViewModel

Класс FullArticleActivity отображает статью на весь экран. В методе onCreate() инициализируются TextView для заголовка (fullArticleTitle) и

текста статьи (fullArticleText). Затем извлекаются данные о статье из Intent, переданного при запуске Activity: заголовок (articleTitle) и полный текст (articleFullText). Полученные данные устанавливаются в соответствующие TextView. Весь алгоритм представлен на рисунке 21.

```
9.    override fun onCreate(savedInstanceState: Bundle?) {
10.       super.onCreate(savedInstanceState)
11.       setContentView(R.layout.activity_full_article)
12.
13.       val fullArticleText: TextView = findViewById(R.id.fullArticleText
14.       val fullArticleTitle: TextView = findViewById(R.id.articleTitle)
15.
16.       val intent = intent
17.       val articleTitle = intent.getStringExtra("articleTitle")
18.       val articleFullText = intent.getStringExtra("articleFullText")
19.
20.       fullArticleText.text = articleFullText
21.       fullArticleTitle.text = articleTitle
22.   }
```

Рисунок 21 - Алгоритм функции onCreate() классаFullArticleActivity

Класс Income Control Activity нужен для отображения activity страницы с контролем операций.

Функция makePie() создает круговую диаграмму (Pie Chart) с использованием библиотеки MPAndroidChart. Диаграмма отображает распределение средств по категориям: «Доход», «Расход», «Расход ЖКХ», «Расход еда». Значения для диаграммы (zkh, eat, other) предполагаются переменными, хранящими соответствующие суммы. Функция создает список PieEntry с рассчитанными долями каждой категории от общей суммы (sum), задает цвета для сегментов диаграммы и устанавливает данные в pieChart. pieChart.invalidate() обновляет отображение диаграммы. Алгоритм представлен на рисунке 22.

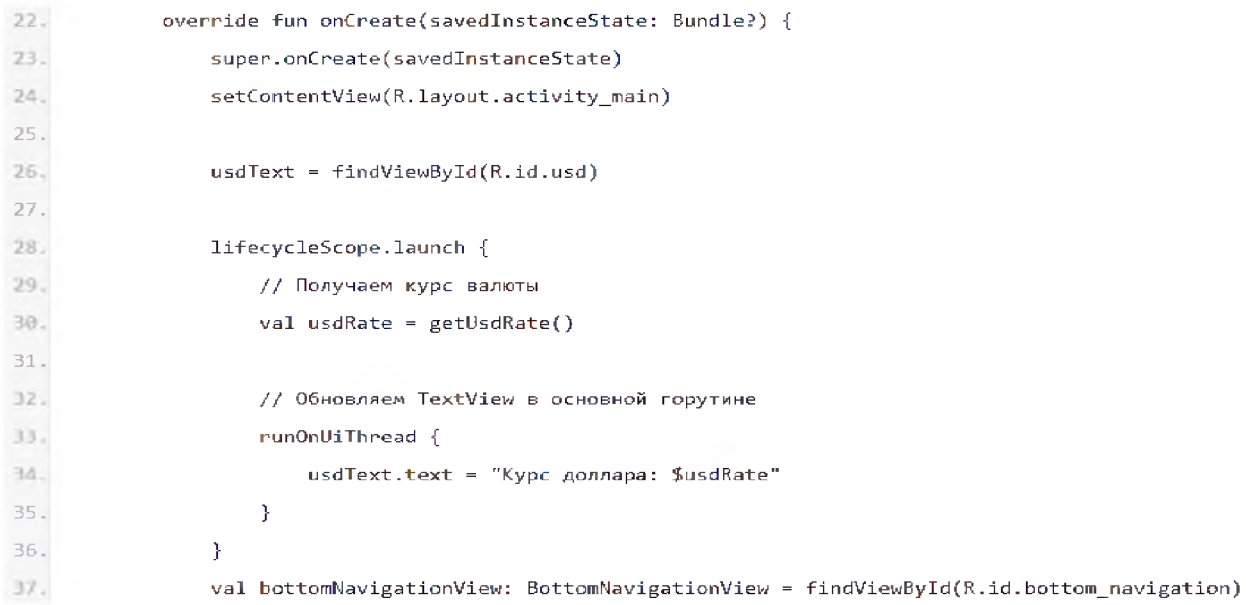
```
74.    private fun makePie(){
75.
76.        pieChart = findViewById(R.id.pieChart)
77.
78.        // Данные для диаграммы
79.        val entries = ArrayList<PieEntry>()
80.        var sum = zkh+eat+other
81.        entries.add(PieEntry(1f*(sum*(other/sum)).toInt(), "Расход"))
82.        entries.add(PieEntry(1f*(sum*(zkh/sum)).toInt(), "Расход ЖКХ"))
83.        entries.add(PieEntry(1f*(sum*(eat/sum)).toInt(), "Расход еда"))
84.
85.        // Создаем набор данных и задаем цвета
86.        val dataSet = PieDataSet(entries, "")
87.        val colors = listOf(Color.parseColor("#F8BBD0"), Color.parseColor("#ADD8E6"), Color.parseColor("#98FB98"))
88.        dataSet.colors = colors
89.
90.        // Создаем объект PieData
91.        val pieData = PieData(dataSet)
92.
93.        // Устанавливаем данные в диаграмму
94.        pieChart.data = pieData
95.        pieChart.invalidate() // Обновляем диаграмму
96.    }
```

Рисунок 22 - Алгоритм функции makePie () класса Income Control Activity

Класс MainActivity реализует главное окно приложения. В методе onCreate() инициализируется TextView для отображения курса доллара (usdText). Асинхронно (с помощью lifecycleScope.launch) вызывается функция getUsdRate(), которая получает актуальный курс. Полученное значение обновляет usdText в основной нити (runOnUiThread). Bottom NavigationView используется для навигации по разделам приложения.

Обработчик событий setOnNavigationItemSelectedListener запускает соответствующую Activity в зависимости от выбранного пункта меню (nav_articles, nav_cbr_rates, nav_calculator, nav_income_control). Представим алгоритм на рисунках 23-24.

Алгоритм прост и основан на стандартных методах Android SDK для обработки событий и навигации.



```
22.         override fun onCreate(savedInstanceState: Bundle?) {
23.             super.onCreate(savedInstanceState)
24.             setContentView(R.layout.activity_main)
25.
26.             usdText = findViewById(R.id.usd)
27.
28.             lifecycleScope.launch {
29.                 // Получаем курс валюты
30.                 val usdRate = getUsdRate()
31.
32.                 // Обновляем TextView в основной горутине
33.                 runOnUiThread {
34.                     usdText.text = "Курс доллара: $usdRate"
35.                 }
36.             }
37.             val bottomNavigationView: BottomNavigationView = findViewById(R.id.bottom_navigation)
```

Рисунок 23 - Алгоритм функции on Create () класса Main Activity, первая часть

```
39.             bottomNavigationView.setOnNavigationItemSelectedListener { item ->
40.                 when (item.itemId) {
41.                     R.id.nav_articles -> {
42.                         startActivity(Intent(this, ArticlesActivity::class.java))
43.                         true
44.                     }
45.                     R.id.nav_crypto_rates -> {
46.                         startActivity(Intent(this, CryptoRatesActivity::class.java))
47.                         true
48.                     }
49.                     R.id.nav_calculator -> {
50.                         startActivity(Intent(this, CalculatorActivity::class.java))
51.                         true
52.                     }
53.                     R.id.nav_income_control -> {
54.                         startActivity(Intent(this, IncomeControlActivity::class.java))
55.                         true
56.                     }
57.                     else -> false
58.                 }
59.             }
60.         }
61.     }
```

Рисунок 24 - Алгоритм функции on Create () класса Main Activity, вторая часть

Класс Transaction Adapter является адаптером для отображения списка транзакций (расходов/доходов) в RecyclerView.

Метод on BindViewHolder() получает объект Transaction и устанавливает текст в TextView (holder.item) элемента списка. Формат текста: «Категория:

[категория] Сумма: [сумма] Тип: [тип]». Алгоритм, что изображен на рисунке 25, прост и заключается в форматировании данных о транзакции в строку для отображения в списке.

```
24.    override fun onBindViewHolder(holder: TransactionViewHolder, position: Int) {
25.        val transaction = transactions[position]
26.        holder.item.text = "Категория: ${transaction.category}  Сумма: ${transaction.amount}    Тип: ${transaction.type}"
27.    }
```

Рисунок 25 - Алгоритм функции on BindViewHolder () класса Transaction Adapter

## 3.3 Описание программных модулей

Проект «ДоМонетки» имеет 4 исполняемых файлов: Api Service.kt, ArticlesActivity.kt, CBRModels.kt, Transaction Adapter.kt, которые отвечают заработоспособность основных функций приложения; и 6 activity: CalculatorActivity, CBRActivity, CBRViewModel, FullArticleActivity, IncomeControlActivity, MainActivity. Все данные файлы представлены на рисунке 26.

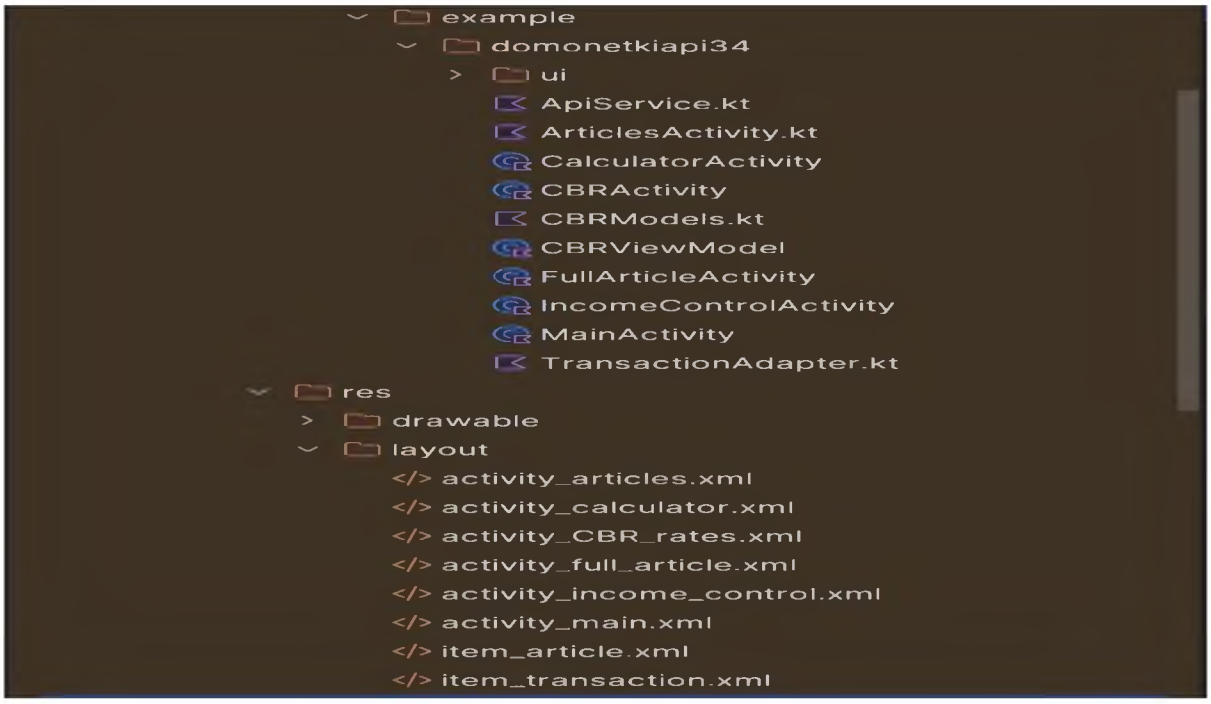



Рисунок 26 - Основные модули проекта

Они находятся в папке ui.theme по пути kotlin+java/com.example.domonetkiapi34/ui.theme/.

В проекте находятся XML ресурсы [22], они содержат разметку activity для визуальной составляющей программного обеспечения, что изображено на рисунке 27.

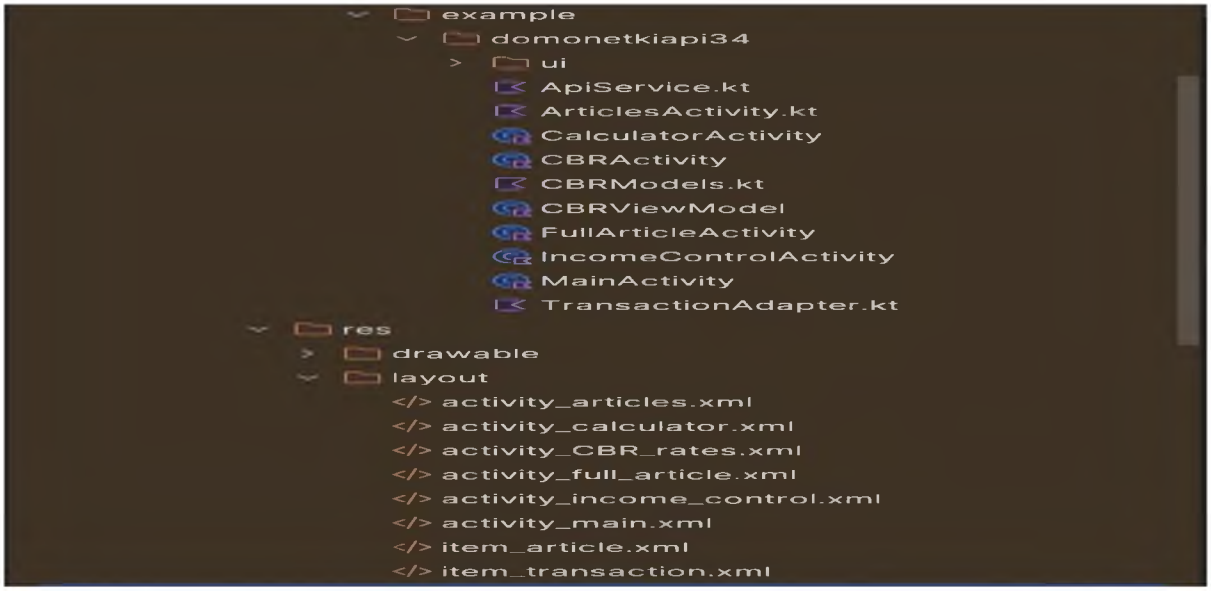



Рисунок 27 - Ресурсы проекта

Отличным примером является activity_full_article.xml. XML разметка обозначила расположение элементов и внешний вид activityFullArticleActivity класса, как показано на рисунках 28-29.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">
    <TextView
        android:textStyle="bold"
        android:gravity="center"
        android:textColor="@color/lavender"
        android:id="@+id/articleTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="36sp" />
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/fullArticleText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp" />
</ScrollView>
</LinearLayout>
```

Рисунок 28 - XML разметка код

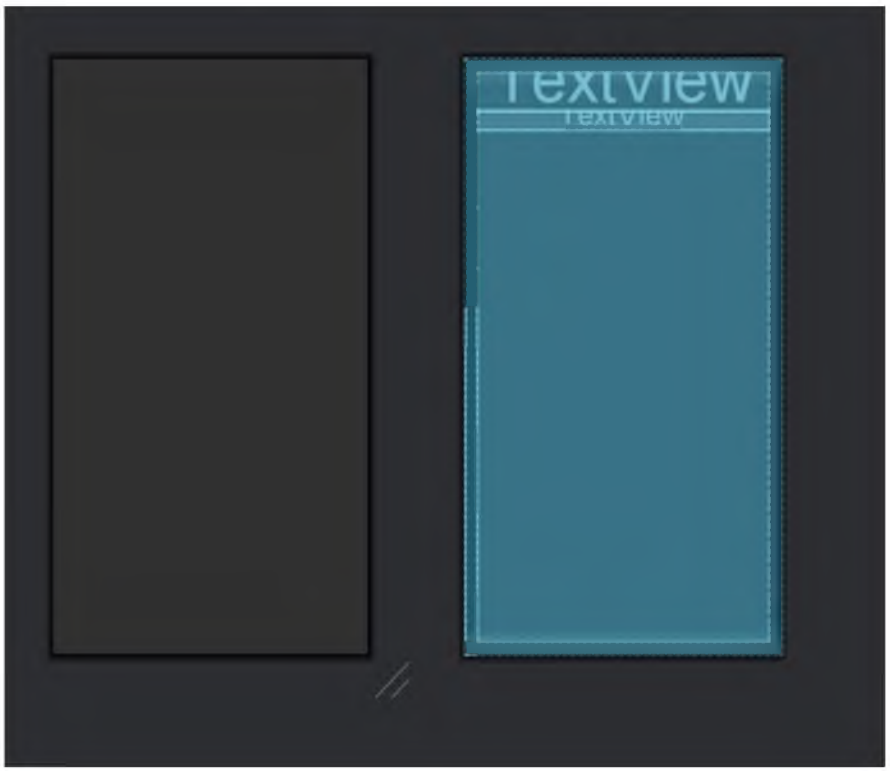



Рисунок 29 - XML разметка activity

## 3.4 Описание интерфейса приложения

Интерфейс приложения [11] — это то, с чем взаимодействует пользователь. Это совокупность всех визуальных элементов, элементов управления и способов взаимодействия, которые определяют, как пользователь воспринимает и использует приложение. Он включает в себя не только графический дизайн [18], но и все аспекты взаимодействия, такие как:

1. Визуальные элементы. Это все, что пользователь видит на экране — цвета, шрифты, изображения, иконки, кнопки, поля ввода, списки и другие графические компоненты. Хороший дизайн интерфейса делает приложение привлекательным, интуитивно понятным и удобным для использования.

2. Элементы управления. Это компоненты, с помощью которых пользователь управляет приложением — кнопки, ползунки, переключатели, поля ввода текста, выпадающие списки и т.д. Они должны быть расположены логично и интуитивно понятно, чтобы пользователь мог легко ориентироваться и управлять функциональностью приложения.

3. Взаимодействие. Это то, как пользователь взаимодействует с приложением. Это включает в себя нажатия на кнопки, ввод текста, прокрутку, жесты [39] (свайпы, пинчи) и другие действия. Взаимодействие должно быть плавным, отзывчивым и предсказуемым, чтобы пользователь чувствовал себя комфортно при использовании приложения.

4. Навигация. Это способ, которым пользователь перемещается между различными экранами и разделами приложения. Хорошая навигация делает приложение легким в освоении и использовании. Навигация может быть реализована с помощью меню, кнопок, жестов или других методов.

5. Обратная связь. Приложение должно предоставлять пользователю обратную связь на его действия. Это может быть визуальная обратная связь (например, изменение цвета кнопки при нажатии), звуковая обратная связь (например, щелчок кнопки) или текстовая обратная связь (например, сообщение об успешном завершении операции).

6. Доступность. Интерфейс должен быть доступен для всех пользователей, включая людей с ограниченными возможностями. Это означает, что он должен быть адаптирован для использования с помощью вспомогательных технологий (например, экранных чтецов).

Эффективный интерфейс приложения должен быть не только красивым, но и функциональным, удобным и интуитивно понятным. Он должен помогать пользователям легко достигать своих целей в приложении, обеспечивая при этом приятный пользовательский опыт. Разработка интерфейса — это итеративный процесс, включающий в себя проектирование, прототипирование, тестирование и доработку на основе обратной связи от пользователей.

Приложение «ДоМонетки» встречает пользователя лаконичным и информативным интерфейсом, выполненным в строгих оттенках синего и черного, что показано на рисунке 30. В верхней части экрана выделяется название приложения — «ДоМонетки» — рядом с которым в режиме реального времени отображаются актуальные курсы доллара и евро, предоставляя пользователю мгновенный доступ к ключевой финансовой информации.

Ниже расположено основное меню, представленное четырьмя функциональными кнопками: «Статьи», «Курсы валют», «Инвестиции» и «Контроль операций». Каждая кнопка интуитивно понятна и обеспечивает быстрый доступ к соответствующему разделу приложения. Удобная навигация и сдержанный дизайн интерфейса обеспечивают комфортное и эффективное управление личными финансами.

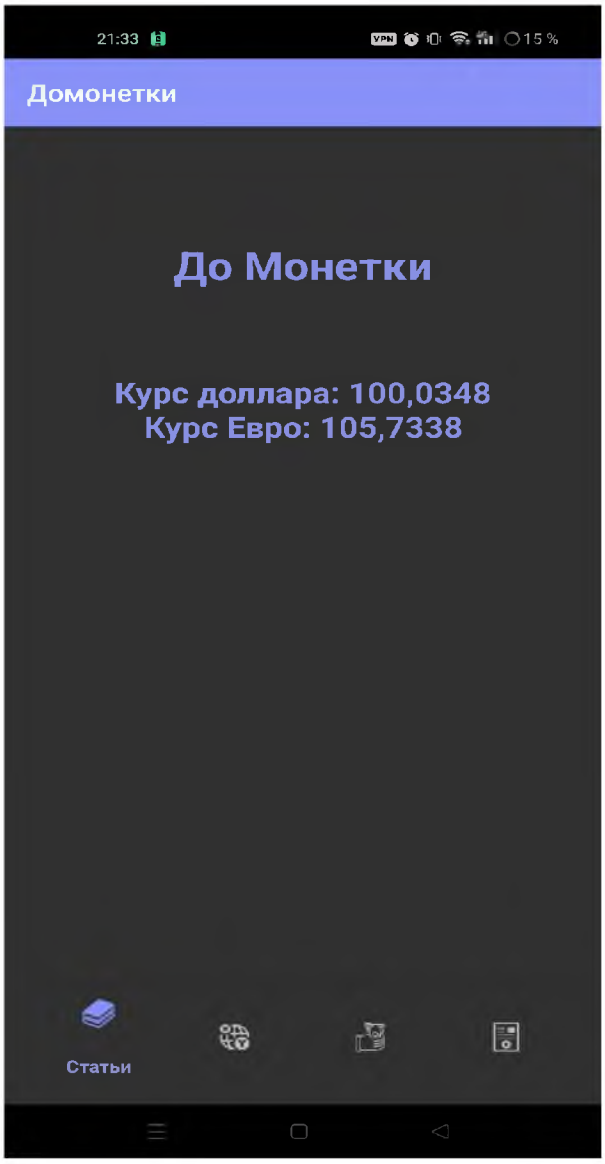



Рисунок 30 - Главное окно приложения

Раздел «Статьи» приложения «ДоМонетки» представляет собой постоянно обновляемый сборник лучших образовательных материалов по финансам и инвестициям за текущий год. На первом экране раздела отображается список статей с краткими аннотациями, позволяющими

пользователю быстро оценить релевантность материала, что изображено на рисунке 31.

Рисунок 31 - Раздел «Статьи», экран первый

Выбрав интересующую статью, пользователь переходит на второй экран, где отображается ее полный текст, обеспечивая комфортное чтение и усвоение информации, что изображено на рисунке 32. Удобная навигация и чёткая структура текста способствуют эффективному обучению и расширению знаний в области финансов.

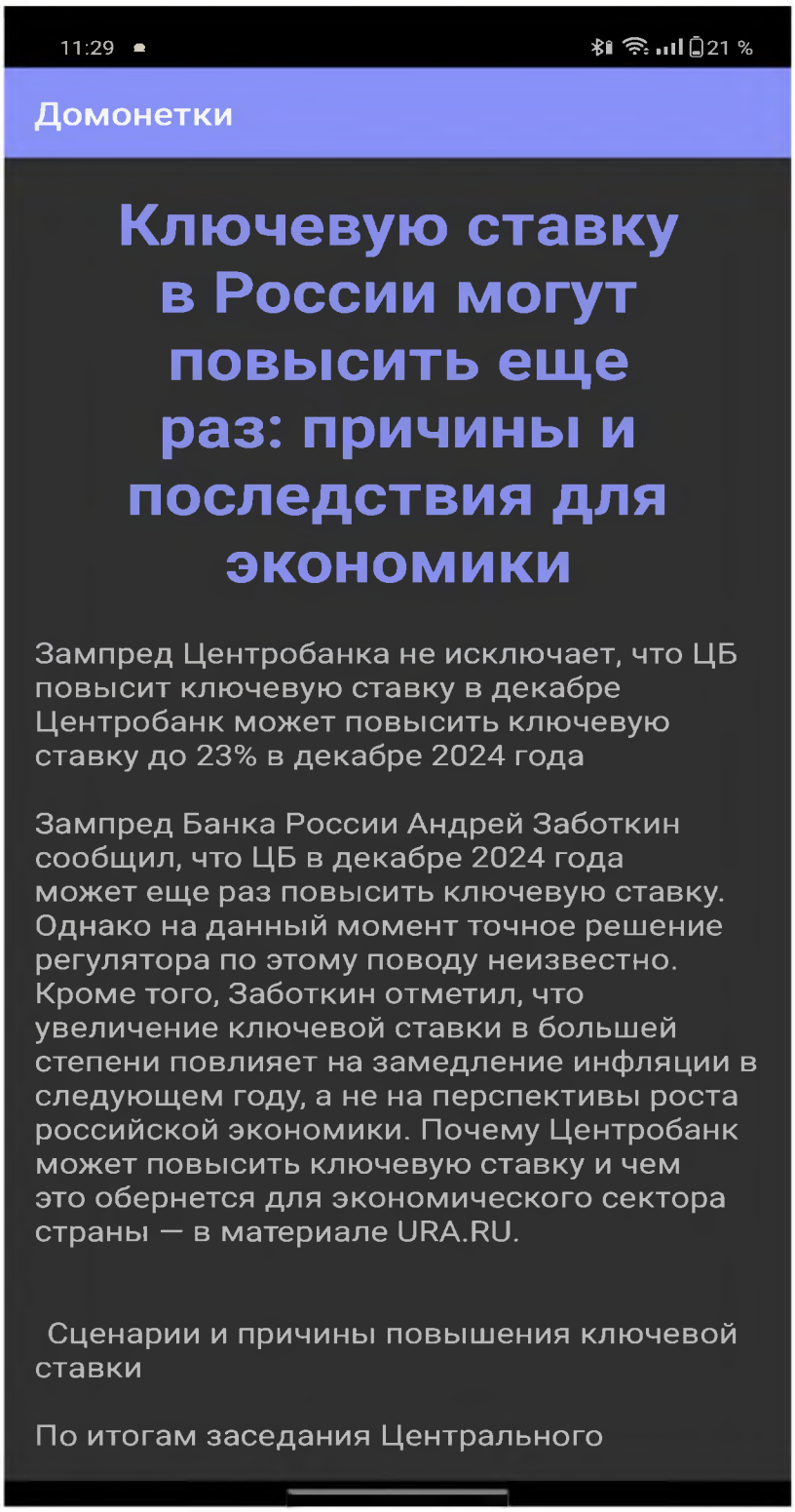



Рисунок 32 - Раздел «Статьи», экран второй

Раздел «Курсы валют» в приложении «ДоМонетки» предоставляет пользователю актуальную информацию о ценах наиболее популярных цифровых валют в режиме реального времени [21], как показано на рисунке 33. На данном экране представлен удобный и интуитивно понятный список валют

с отображением их текущих курсов, позволяющий быстро найти и сравнить цены различных активов. Пользователь может легко выбрать интересующую валюту для отслеживания ее стоимости и принятия обоснованных решений.

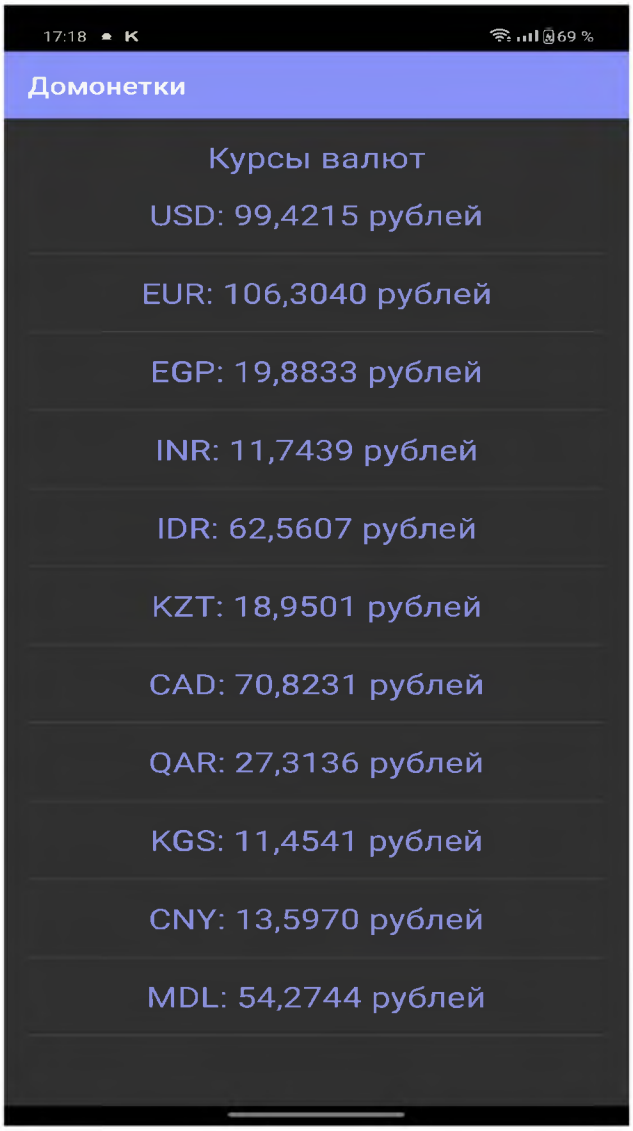



Рисунок 33 - Раздел «Курсы валют»

Раздел «Инвестиции» приложения «ДоМонетки» содержит удобный калькулятор вкладов, позволяющий пользователю быстро рассчитать потенциальную прибыль от инвестиций. Для расчета необходимо ввести три параметра: сумму вклада, годовую процентную ставку и срок вклада в годах, пример показан на рис. 34. После нажатия кнопки «Рассчитать», приложение мгновенно выведет итоговую сумму, учитывая сложные проценты. Простой и интуитивный интерфейс калькулятора делает процесс планирования инвестиций максимально комфортным и понятным.

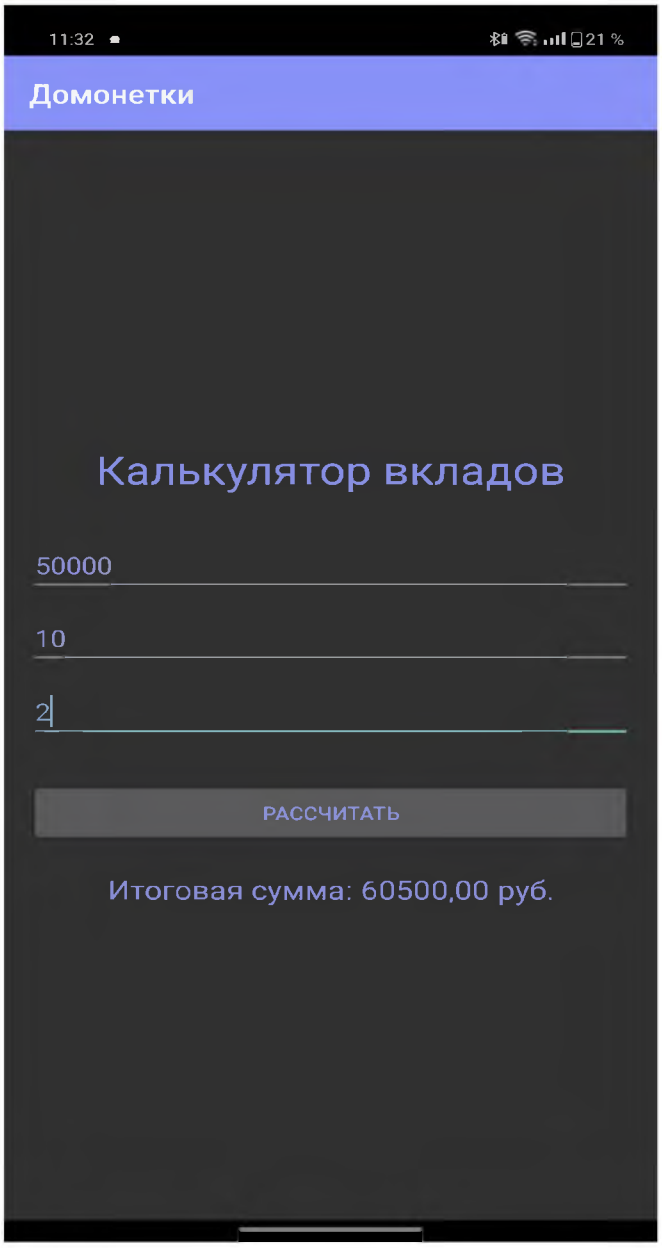



Рисунок 34 - Раздел «Инвестиции»

В разделе «Контроль операций» приложения «ДоМонетки» пользователь может регистрировать свои доходы и расходы, а также разделить расходы по категориям (например, «ЖКХ», «Еда»). Для этого необходимо ввести сумму операции, выбрать тип (доход/расход) и указать соответствующую категорию. После нажатия «Добавить», приложение рассчитает текущий баланс и визуализирует его на круговой диаграмме, показывая распределение средств по различным категориям, как показано в примере на рисунке 35. Этот инструмент помогает пользователю отслеживать свой финансовый личный баланс и контролировать расходы.

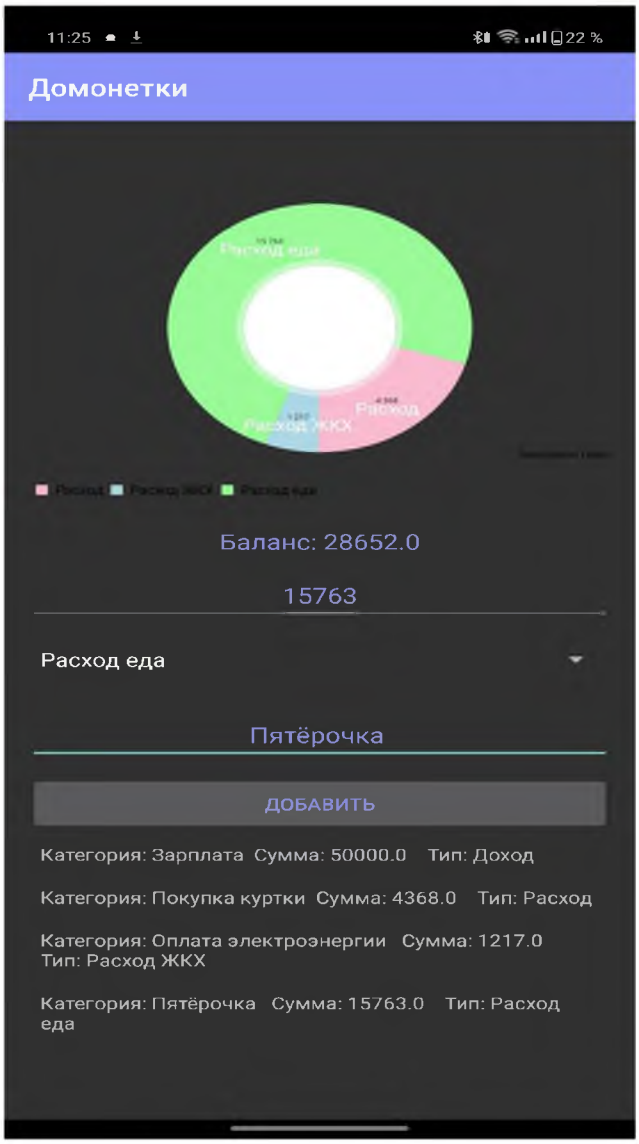



Рисунок 35 - Раздел «Контроль операций»

Заключение

Разработка мобильного приложения «ДоМонетки» отвечает на растущую потребность в удобных и доступных инструментах для личного финансового планирования. Ранее, управление личными финансами часто осуществлялось с помощью разрозненных инструментов – отдельных программ для расчета вкладов, таблиц для отслеживания расходов и веб-сайтов для мониторинга курсов валют. Это создавало неудобства и затрудняло комплексный анализ финансового положения.

Приложение «ДоМонетки» решает эту проблему, предоставляя унифицированное решение, объединяющее ключевые функции в одном интерфейсе. В рамках проекта были разработаны:

- интерактивный калькулятор вкладов, обеспечивающий быстрый и точный расчет будущей доходности;
- модуль для отслеживания курсов валют в режиме реального времени;
- раздел образовательными статьями по инвестициям и финансам;
- удобная система контроля операций с интуитивной категоризацией и визуализацией данных в виде круговой диаграммы.

Система контроля операций — это ключевой модуль приложения, позволяющий пользователям отслеживать все свои доходы и расходы. Эта система включает интуитивную категоризацию операций, позволяющую пользователям легко добавлять новые операции, присваивая им соответствующие категории, что позволяет структурировать финансовые данные. Визуализация данных в виде круговой диаграммы предоставляет наглядное представление структуры расходов, позволяя пользователям быстро определить основные статьи затрат и анализировать тенденции в расходах за разные периоды времени.

Детализированные отчеты позволяют пользователям просматривать подробную информацию о своих операциях за различные периоды времени, что необходимо для проведения углубленного анализа. Также имеется возможность

экспорта данных в формате CSV или других удобных форматах для дальнейшей работы.

В результате проведенной работы было разработано функциональное и удобное мобильное приложение «ДоМонетки», способное существенно упростить процесс управления личными финансами. Приложение предоставляет пользователю комплексный инструмент для планирования инвестиций, отслеживания расходов, расширения финансовой грамотности и позволяет пользователям принимать взвешенные финансовые решения.

Приложение характеризуется комплексным подходом, объединяющим все ключевые функции финансового управления в одном приложении. При этом «ДоМонетки» обладает удобным и интуитивно понятным интерфейсом, делающим управление финансами простым и доступным для всех пользователей. Визуализация данных обеспечивает наглядное представление финансовой информации, а образовательный контент способствует повышению финансовой грамотности. «ДоМонетки» универсальны и подходят для пользователей с различным уровнем финансовой грамотности и потребностями.

В дальнейшем развитии приложения планируется реализовать следующие функции. Интеграция с банками позволит автоматически загружать данные о транзакциях. Персонализированные рекомендации будут предоставлять пользователю персональные советы по управлению финансами на основе анализа данных. Также планируется реализовать возможность планирования финансовых целей для создания планов достижения конкретных финансовых целей. Поддержка различных валют позволит вести финансовый учет в разных валютах. Планируется улучшение графической визуализации данных за счет добавления новых типов диаграмм и графиков для более наглядного представления информации.

Все поставленные задачи были выполнены в полном объеме, и проект успешно завершен. Приложение «ДоМонетки» является ценным инструментом для всех, кто стремится к эффективному управлению личными финансами и повышению финансовой грамотности.

Список литературы

1. Описание стандарта IDEF0. [Электронный ресурс] — Режим доступа: https://www.interface.ru/home.asp?artId=22560 (дата обращения 18.11.2024).

2. Модель C4. [Электронный ресурс] — Режим доступа: https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_C4 (дата обращения 18.11.2024).

3. CoinMarketCap (КоинМаркетКап) - агрегатор данных о криптовалютах, обзор, преимущества и недостатки [Электронный ресурс] — Режим доступа: https://profinvestment.com/coinmarketcap (дата обращения 10.11.2024).

4. Blockfolio, Delta или PYGMY: кому доверить управление своим криптовалютным портфелем. [Электронный ресурс] — Режим доступа: https://forklog.com/exclusive/blockfolio-delta-ili-pygmy-komu-doverit-upravlenie-svoim-kriptovalyutnym-portfelem (дата обращения 10.11.2024).

5. Лучшие сервисы по учёту криптовалюты (трекеры) и созданию криптовалютных портфелей. [Электронный ресурс] — Режим доступа: https://vc.ru/u/2396097-eksperts/896438-luchshie-servisy-po-uchetu-kriptovalyuty-trekery-i-sozdaniyu-kriptovalyutnyh-portfelei (дата обращения 10.11.2024).

6. Лекция 2. Методологическая основа планирования. [Электронный ресурс] — Режим доступа: https://infourok.ru/lekciya-metodologicheskaya-osnova-planirovaniya-3992577.html (дата обращения 13.11.2024).

7. Использование диаграммы вариантов использования UML при проектировании программного обеспечения. [Электронный ресурс] — Режим доступа: https://habr.com/ru/articles/566218/ (дата обращения 15.11.2024).

8. Диаграмма последовательности. [Электронный ресурс] — Режим доступа: https://ru.ruwiki.ru/wiki/Диаграмма_последовательности (дата обращения 15.11.2024).

9. AcompleteGuidetoLearnAndroidStudio. [Электронный ресурс] — Режим доступа: https://www.geeksforgeeks.org/a-complete-guide-to-learn-android-studio-for-app-development (дата обращения 15.11.2024).

10. Наиболее интересные доклады с AnalystDays 2015. [Электронный ресурс] — Режим доступа: https://habr.com/ru/companies/nix/articles/257715/ (дата обращения 11.11.2024).

11. Что такое пользовательский интерфейс (UI) и как происходит UI-разработка: принципы и этапы создания интерфейсов. [Электронный ресурс] — Режим доступа: https://fortech.dev/blog/chto-takoye-polzovatelskiy-interfeys-i-kak-proiskhodit-ui-razrabotka/ (дата обращения 12.11.2024).

12. Псевдокод для начинающих: основы, примеры и руководство. [Электронный ресурс] — Режим доступа: https://sky.pro/wiki/javascript/psevdokod-dlya-nachinayushih-osnovy-primery-i-rukovodstvo/ (дата обращения 13.11.2024).

13. Что такое криптовалюта и как ее правильно использовать. [Электронный ресурс] — Режим доступа: https://www.nur.kz/nurfin/stock/1884579-cto-takoe-kriptovaluta-i-v-cem-ee-osobennosti/ (дата обращения 15.11.2024).

14. Цифровые финансовые активы: что это и как с ними работать [Электронный ресурс] — Режим доступа: https://journal.tinkoff.ru/guide/digital-financial-assets/ (дата обращения 18.11.2024).

15. Макроэкономическая нестабильность. [Электронный ресурс] — Режим доступа: https://spravochnick.ru/mikro_makroekonomika/makroekonomicheskaya_nestabilnost/ (дата обращения 12.11.2024).

16. Что такое тренды рынка и зачем их наблюдать? [Электронный ресурс] — Режим доступа: https://vc.ru/marketing/1342764-chto-takoe-trendy-rynka-i-zachem-ih-nablyudat (дата обращения 17.11.2024).

17. Что такое финтех. [Электронный ресурс] — Режим доступа: https://www.raiffeisen-media.ru/money/chto-takoe-finteh/ (дата обращения 15.11.2024).

18. Что такое графический дизайн. [Электронный ресурс] — Режим доступа: https://media.contented.ru/osnovy/napravleniya/graficheskij-dizajn-chto-takoe/ (дата обращения 18.11.2024).

19. Диаграмма эффектов: объектно-ориентированная декомпозиция. [Электронный ресурс] — Режим доступа: https://azhidkov.pro/drafts/effects-diagram/ood-decomposition-v2/ (дата обращения 15.11.2024).

20. Introductionto Hierarchical Data Structure. [Электронный ресурс] — Режим доступа: https://www.geeksforgeeks.org/introduction-to-hierarchical-data-structure/ (дата обращения 15.11.2024).

21. Режим реального времени. [Электронный ресурс] — Режим доступа: https://bigenc.ru/c/rezhim-real-nogo-vremeni-19001e (дата обращения 15.11.2024).

22. Что такое XML-формат и где его применяют. [Электронный ресурс] — Режим доступа: https://skillbox.ru/media/code/chto-takoe-xmlformat-i-gde-ego-primenyayut/ (дата обращения 16.11.2024).

23. Dataclasses. [Электронный ресурс] — Режим доступа: https://kotlinlang.org/docs/data-classes.html (дата обращения 14.11.2024).

24. Научные статьи по экономике. [Электронный ресурс] — Режим доступа: https://статья-вак.рф/object/topics/po-ekonomike/ (дата обращения 15.11.2024).

25. От чего зависит курс криптовалют, или «Эти ваши биткоины ничем не обеспечены». [Электронный ресурс] — Режим доступа: https://trendup.pro/trendbook/kurs-kriptovalyut/ (дата обращения 15.11.2024).

26. Калькулятор вкладов. [Электронный ресурс] — Режим доступа: https://calc.consultant.ru/vkl (дата обращения 13.11.2024).

27. Зачем вести личный бюджет и как планировать расходы. [Электронный ресурс] — Режим доступа: https://www.gazprombank.ru/pro-finance/lifestyle/zachem-vesti-lichnyj-byudzhet/ (дата обращения 10.11.2024).

28. Требования: что это и какие бывают виды. [Электронный ресурс] — Режим доступа: https://vc.ru/dev/1210521-trebovaniya-chto-eto-i-kakie-byvayut-vidy (дата обращения 18.11.2024).

29. Справочник UML. Объектно-ориентированное проектирование.component (компонент). [Электронный ресурс] — Режим доступа: https://openu.ru/Books/UML/Component.asp (дата обращения

11.11.2024).

30. Экономика России. Инфраструктура / К. Н. Назин, Д. И. Кокурин, 2024. - 208 с.: ISBN 978-5-534-10612-1. [Электронный ресурс] — Режим доступа: https://urait.ru/book/ekonomika-rossii-infrastruktura-541856 (дата обращения 20.11.2024).

31. Экономика отрасли информационных систем / А. Л. Рыжко, Н. А. Рыжко, Н. М. Лобанова, Е. О. Кучинская, 2024. - 38 с.: ISBN 978-5-534-11628-1. [Электронный ресурс] — Режим доступа: https://urait.ru/book/ekonomika-otrasli-informacionnyh-sistem-542801(дата обращения 22.11.2024).

32. Вычислительная техника и информационные технологии. Разработка мобильных приложений / В. В. Соколова, 2024.: ISBN 978-5-534-16302-5. [Электронный ресурс] — Режим доступа: https://urait.ru/book/vychislitelnaya-tehnika-i-informacionnye-tehnologii-razrabotka-mobilnyh-prilozheniy-537272 (дата обращения 21.11.2024).

33. Разработка мобильных приложений / В. В. Соколова, 2024.: ISBN 978-5-534-16868-6. [Электронный ресурс] — Режим доступа: https://urait.ru/book/razrabotka-mobilnyh-prilozheniy-542342(дата обращения 22.11.2024).

34. Государственное регулирование экономики / Васильев, В. П, 2024.: ISBN 978-5-534-18712-0. [Электронный ресурс] — Режим доступа: https://urait.ru/book/gosudarstvennoe-regulirovanie-ekonomiki-545417(дата обращения 21.11.2024).

35. Мировая экономика в 2 ч. Часть 1 / Р. И. Хасбулатов, 2024.: ISBN 978-5-534-11204-7. [Электронный ресурс] — Режим доступа: https://urait.ru/book/mirovaya-ekonomika-v-2-ch-chast-1-542412(дата обращения 22.11.2024).

# Приложение 1

## Листинг приложения

**class MainActivity**

```
package com.example.domonetkiapi34
import android.app.Activity
import android.content.Intent
import android.os.Bundle
import
androidx.appcompat.app.AppCompatActivity
import android.widget.Button
import android.widget.TextView
import
androidx.lifecycle.LifecycleCoroutineScope
import androidx.lifecycle.lifecycleScope
import
com.example.domonetkiapi34.ArticlesActivity
import com.example.domonetkiapi34.R
import
com.google.android.material.bottomnavigation.BottomN
avigationView
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
import java.net.URL
import java.util.regex.Pattern
class MainActivity : AppCompatActivity() {
private lateinit var usdText: TextView
override   fun   onCreate(savedInstanceState:
Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
usdText = findViewById(R.id.usd)
lifecycleScope.launch {
// Получаемкурсвалюты
val usdRate = getUsdRate()
// Обновляем TextView в основной горутине
runOnUiThread {
usdText.text = "Курс доллара: $usdRate"
}
}
val                    bottomNavigationView:
BottomNavigationView                                =
findViewById(R.id.bottom_navigation)
bottomNavigationView.setOnNavigationItemSele
ctedListener { item ->
when (item.itemId) {
R.id.nav_articles -> {
startActivity(Intent(this,
ArticlesActivity::class.java))
true
}
R.id.nav_crypto_rates -> {
startActivity(Intent(this,
CryptoRatesActivity::class.java))
true
}
R.id.nav_calculator -> {
startActivity(Intent(this,
CalculatorActivity::class.java))
true
}
R.id.nav_income_control -> {
startActivity(Intent(this,
IncomeControlActivity::class.java))
true
}
else -> false
}
}
}
}
private suspend fun getUsdRate(): String {
return withContext(Dispatchers.IO) {
try {
val                    url                    =
URL("https://www.cbr.ru/scripts/XML_daily.asp")
val content = url.readText()
val                 pattern                 =
Pattern.compile("<Value>(.*?)</Value>")
val matcher = pattern.matcher(content)
var value =""
var matcherCounter = 0
while (matcher.find()) {
if (matcherCounter == 13 ){
value += matcher.group(1).trim()
}
if(matcherCounter==14){
value            +="\nКурсЕвро:            "+
matcher.group(1).trim()
break
}
```

```
matcherCounter++
}
value
} catch (e: Exception) {
"Ошибка получения курса: ${e.message}"
}
}
}
```

**interface ApiService**

```
package com.example.domonetkiapi34
import retrofit2.Retrofit
import
retrofit2.converter.moshi.MoshiConverterFactory
import retrofit2.http.GET
import retrofit2.http.Headers
import retrofit2.http.Query
interface CoinMarketCapApiService {
@Headers("X-CMC_PRO_API_KEY:  141ab8d7-7743-
4a0d-a19e-fed4045c5303") // Замените на ваш API-ключ
@GET("v1/cryptocurrency/listings/latest")
suspend fun getCryptoListings(
@Query("convert") currency: String = "RUB"
): CryptoResponse
companion object {
private const val BASE_URL = "https://pro-
api.coinmarketcap.com/"
fun create(): CoinMarketCapApiService {
return Retrofit.Builder()
.baseUrl(BASE_URL)
.addConverterFactory(MoshiConverterFactory.c
reate())
.build()
.create(ApiService::class.java)
}
}
}
```

**class CalculatorActivity**

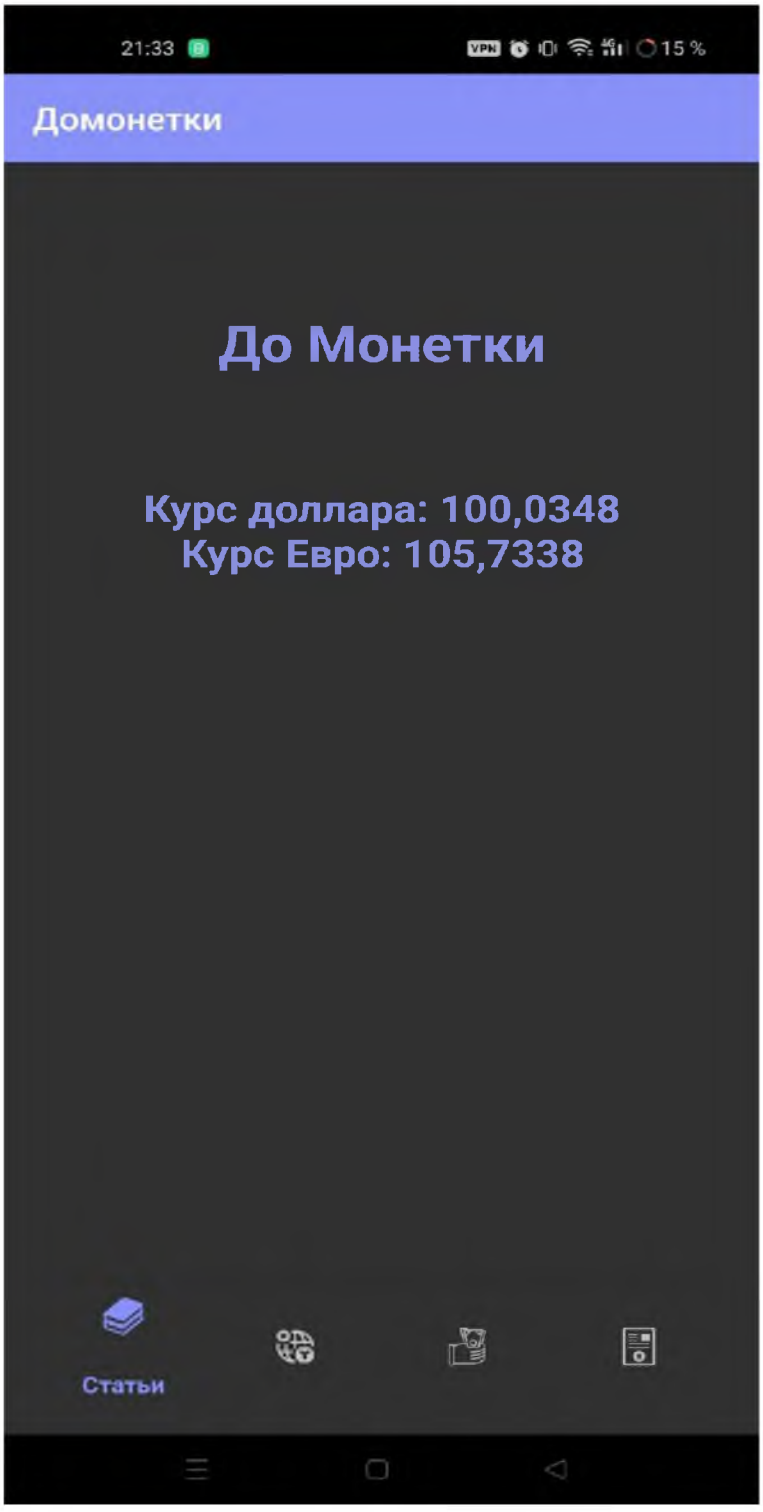
```
package com.example.domonetkiapi34
import android.app.Activity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import
androidx.appcompat.app.AppCompatActivity
class          CalculatorActivity          :
AppCompatActivity() {
override   fun   onCreate(savedInstanceState:
Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_calculator)
val     principalEditText:     EditText     =
findViewById(R.id.editTextPrincipal)
val      rateEditText:      EditText      =
findViewById(R.id.editTextRate)
val      yearsEditText:      EditText      =
findViewById(R.id.editTextYears)
val      resultTextView:      TextView      =
findViewById(R.id.textViewResult)
val      calculateButton:      Button      =
findViewById(R.id.buttonCalculate)
calculateButton.setOnClickListener {
val                principal                =
principalEditText.text.toString().toDoubleOrNull() ?:
0.0
val                  rate                  =
rateEditText.text.toString().toDoubleOrNull() ?: 0.0
val                  years                  =
yearsEditText.text.toString().toDoubleOrNull()    ?:
0.0
val result = principal * Math.pow(1 + (rate
/ 100), years)
resultTextView.text    =    "Итоговаясумма:
${String.format("%.2f", result)} руб."
}
}
}
```

**class CBRModels**

```
package com.example.domonetkiapi34
data class CBResponse(val data: List<Crypto>)
data class CBRModels(
val id: Int,
val name: String,
val symbol: String,
val quote: Map<String, Quote>
)
data class Quote(val price: Double)
```

Приложение 2

Экранные формы. Главное окно приложения

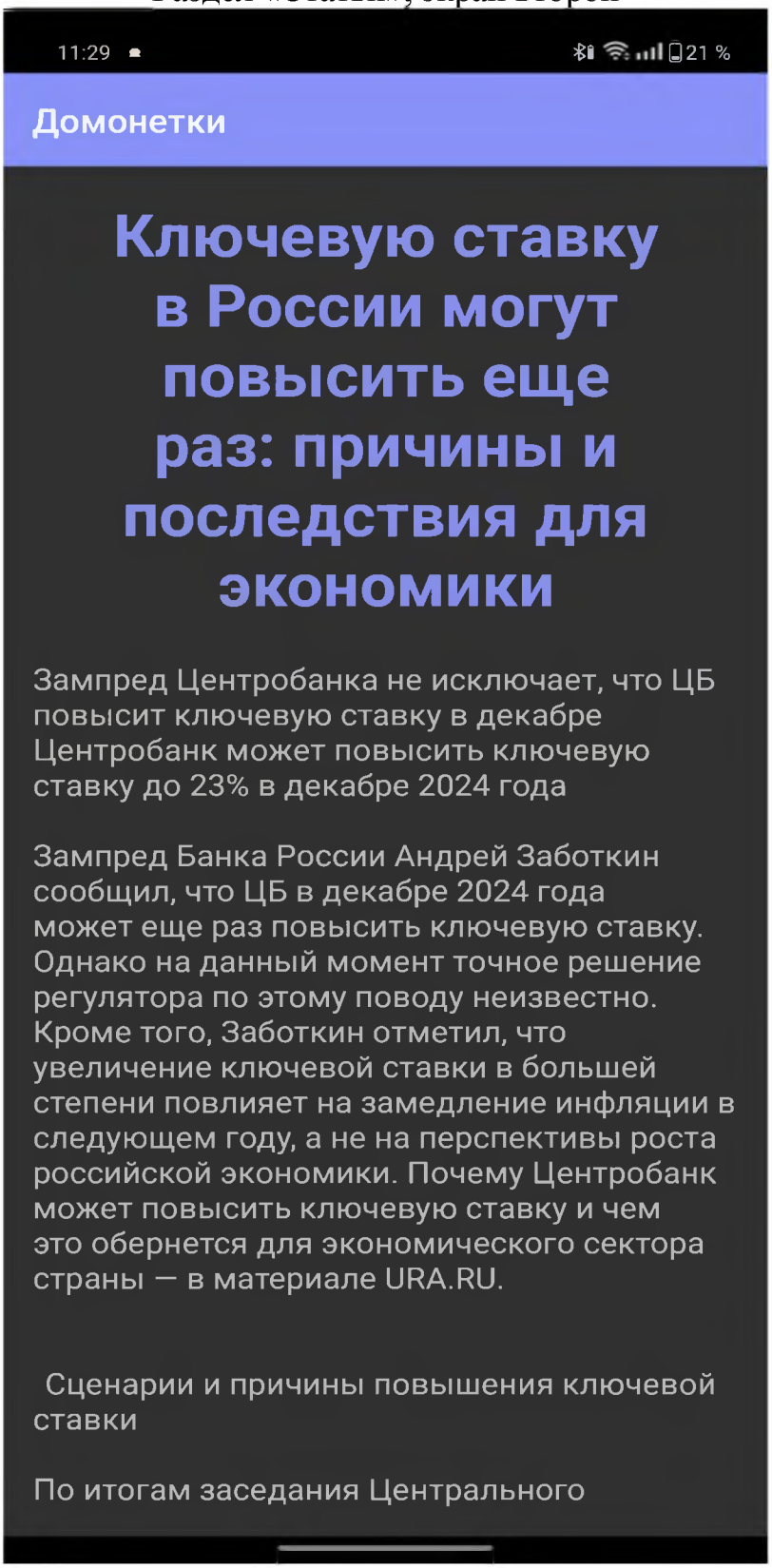


Продолжение приложения 2

Раздел «Статьи», экран первый

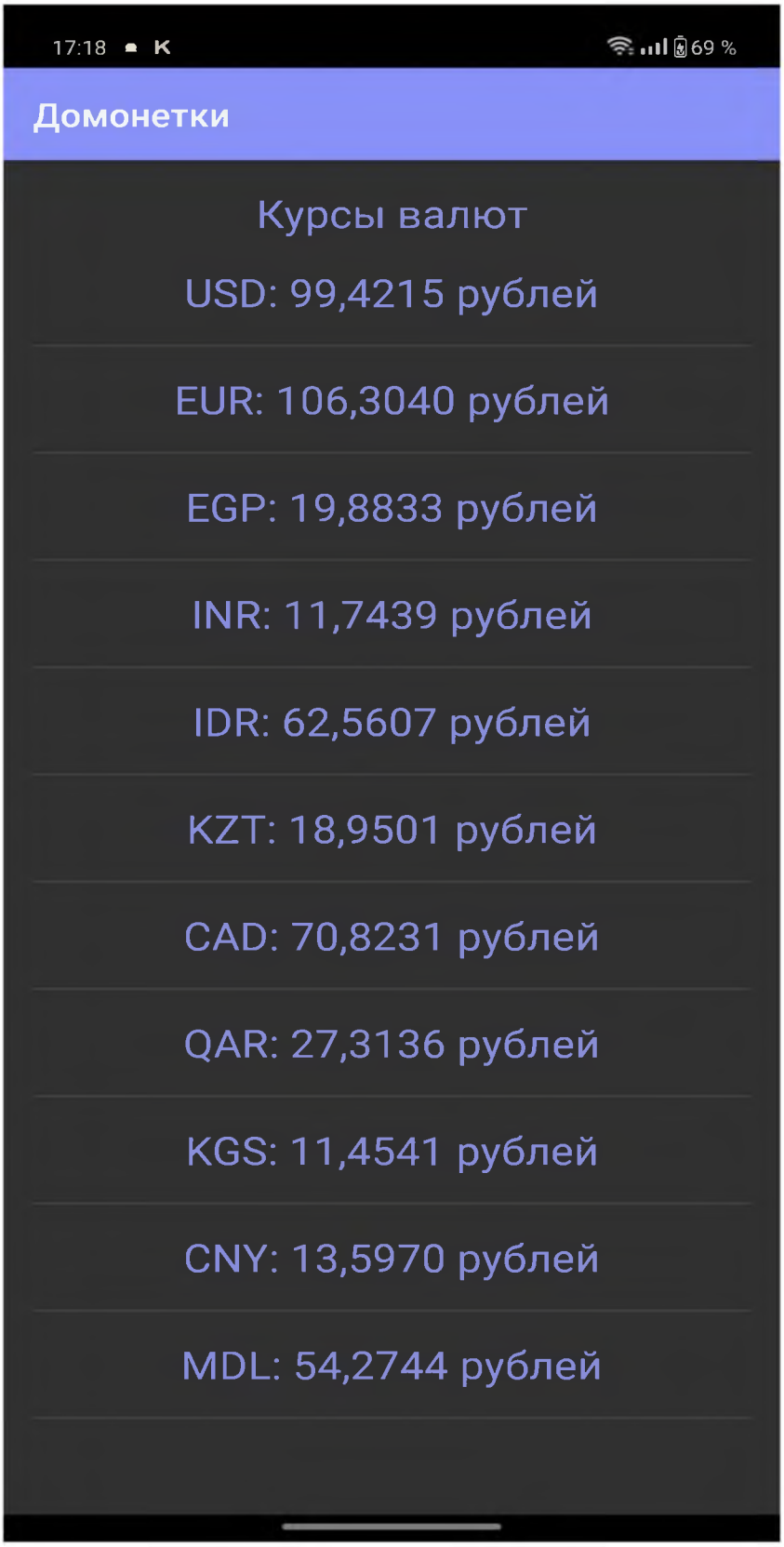
Продолжение приложения 2

Раздел «Статьи», экран второй



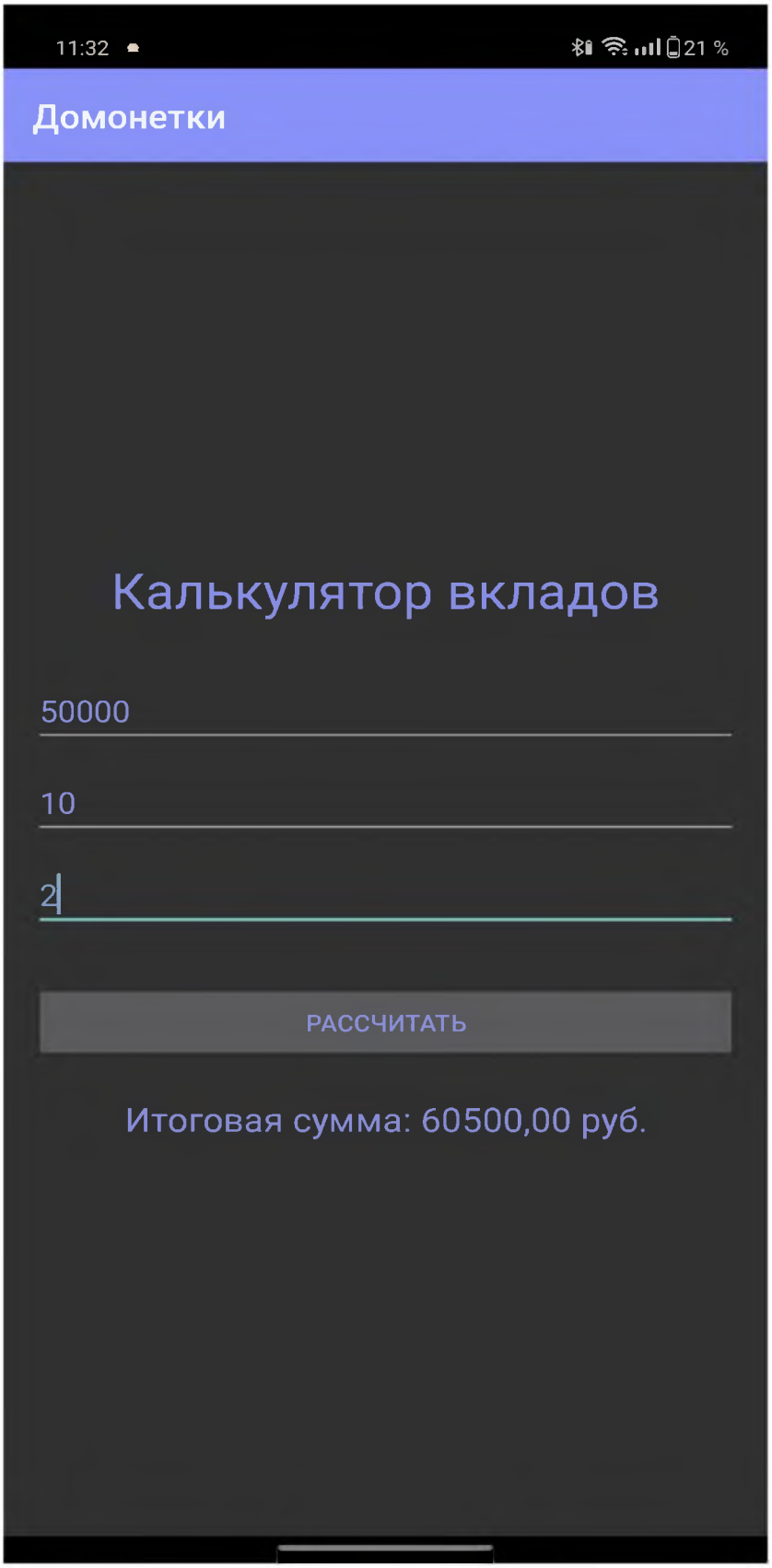
Продолжение приложения 2

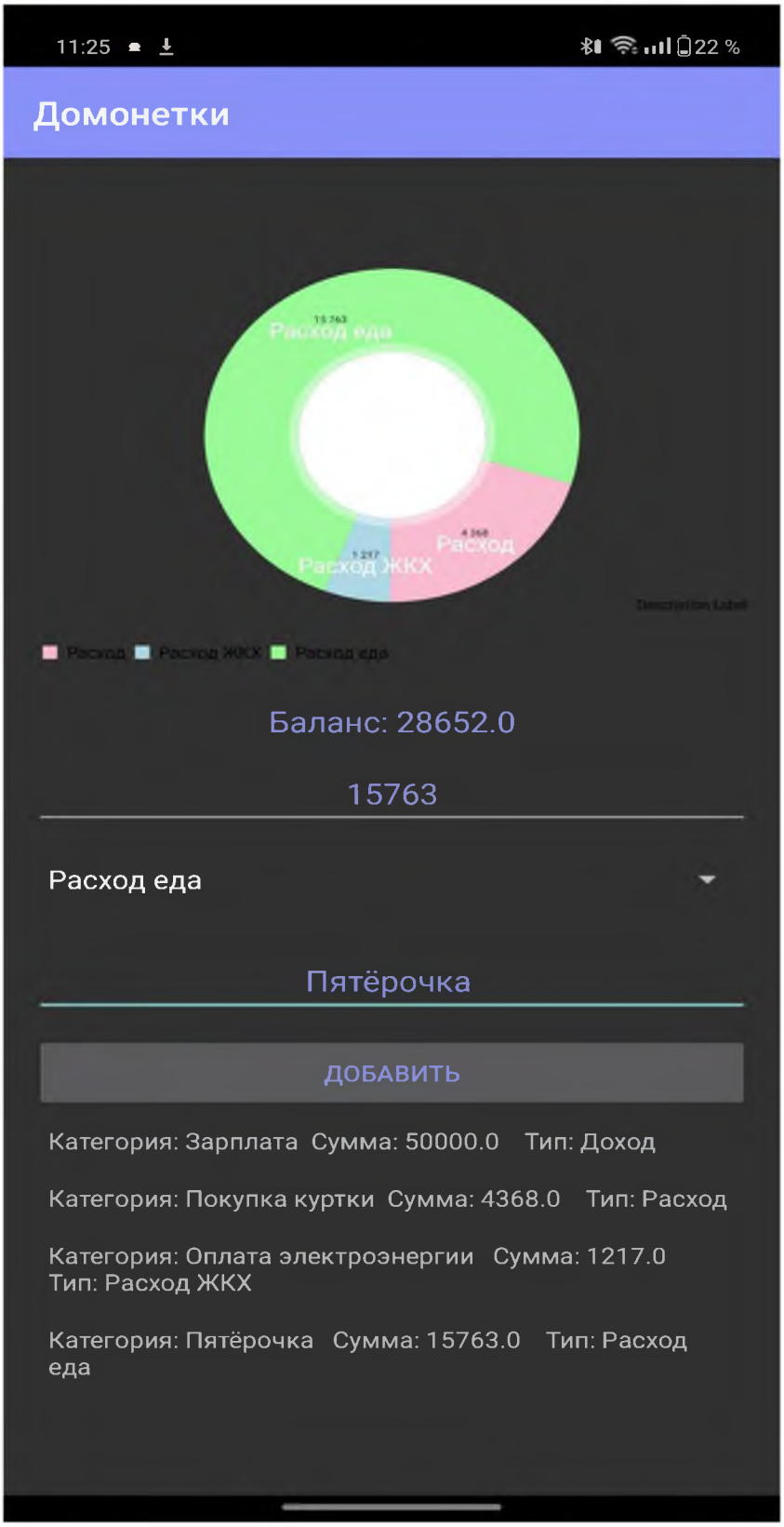
Раздел «Курсы валют»



Продолжение приложения 2

Раздел «Инвестиции»



Продолжение приложения 2

Раздел «Контроль операций»