**Отчет по проекту: Мобильное приложение Petty Clicker**

**1. Введение**

**1.1 Обзор проекта и архитектура**

Petty Clicker — это мобильное Android-приложение в жанре idle/clicker, разработанное на языке Kotlin с использованием Jetpack Compose, архитектуры MVVM (Model-View-ViewModel), а также Room для локальной базы данных. Пользователь зарабатывает клики, которые затем может тратить на покупку различных питомцев. В приложении предусмотрены разные категории питомцев (обычные, экзотические, фантастические, легендарные), автокликер, а также возможность называть питомцев и просматривать коллекцию.



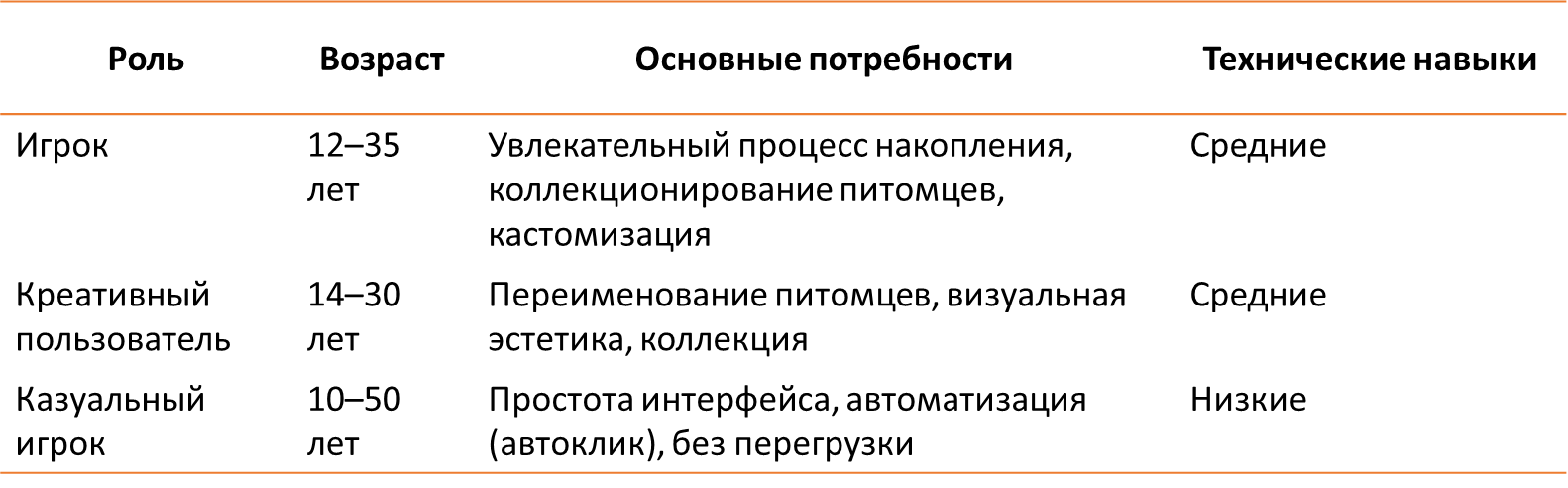
Приложение построено на базе многослойной архитектуры, разделённой на три слоя:

1. *Presentation Layer* — пользовательский интерфейс, созданный на Jetpack Compose. Управление состоянием реализовано через ViewModel и StateFlow. Навигация реализована через активности и встроенные переходы.
2. *Domain Layer* — представлен бизнес-логикой внутри ViewModel и Repository, в частности логикой автокликера, расчёта бонусов и обработки пользовательских действий.
3. *Data Layer* — база данных Room, хранилище изображений питомцев, DAO-интерфейсы и репозитории. Все данные обрабатываются в реактивном стиле через Flow.

Архитектура приложения обеспечивает хорошую модульность, тестируемость и масштабируемость. Внедрение зависимостей осуществляется с помощью Dagger Hilt, что упрощает управление жизненным циклом компонентов и делает архитектуру устойчивой к изменениям.

**1.2 Целевая аудитория и пользовательские роли**

Целевым пользователем Petty Clicker является широкая аудитория мобильных пользователей в возрасте от 12 до 35 лет, увлечённых простыми и визуально привлекательными играми с элементами коллекционирования и прогрессии.



Приложение ориентировано на предоставление простого, понятного и визуально насыщенного игрового процесса без сложных механик. Интерфейс адаптирован под оба типа тем (светлую и тёмную), а также содержит элементы персонализации.

**2. Анализ требований**

**2.1 Функциональные требования**

Функциональные требования к приложению Petty Clicker сформированы на основе игрового сценария и взаимодействия с питомцами.

Основные функции для пользователя:

* Накопление кликов по нажатию на экран
* Отображение текущего баланса
* Приобретение питомцев из магазина с разными уровнями цен и редкости
* Отображение коллекции купленных питомцев
* Автокликер при наличии легендарного питомца
* Переименование питомцев
* Переключение светлой и тёмной темы оформления
* Отображение анимации бонусных кликов
* Загрузка и отображение случайных цитат

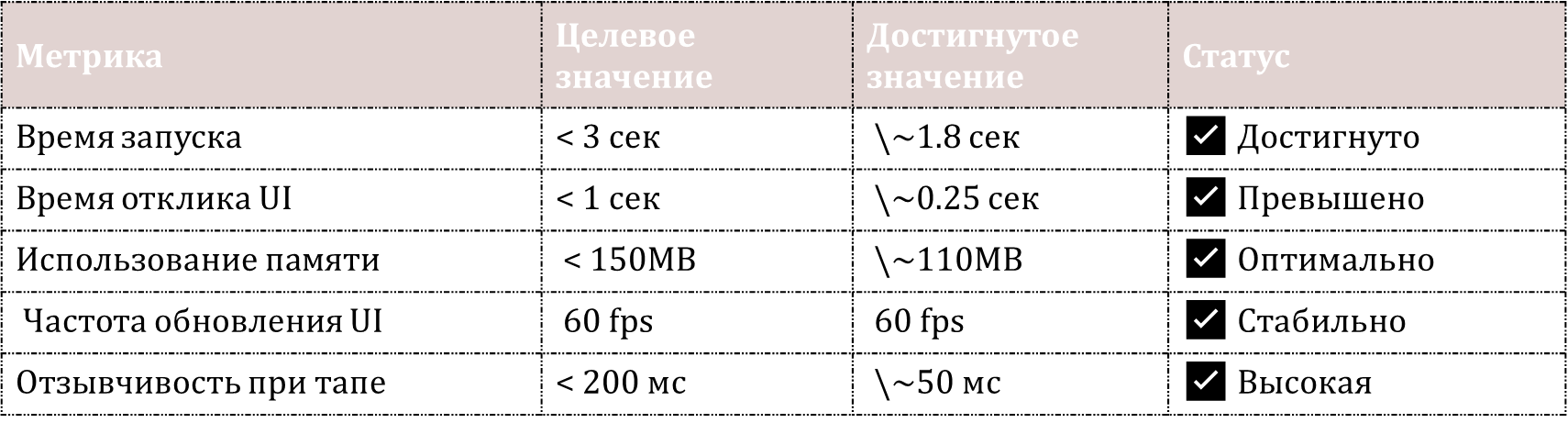
Поддерживаемые категории питомцев:

* Обычные (regular)
* Экзотические (exotic)
* Фантастические (fantastic)
* Легендарные (legendary)

Поведение при покупке:

* Цена питомца увеличивается с каждой покупкой
* Возможность покупки нескольких экземпляров одного питомца
* Отображение количества купленных питомцев

**2.2 Нефункциональные требования – Метрики производительности**



Использование StateFlow, Jetpack Compose и архитектуры MVVM позволило достичь высокой отзывчивости и визуальной плавности, даже при большом количестве анимаций и коллекционных объектов.

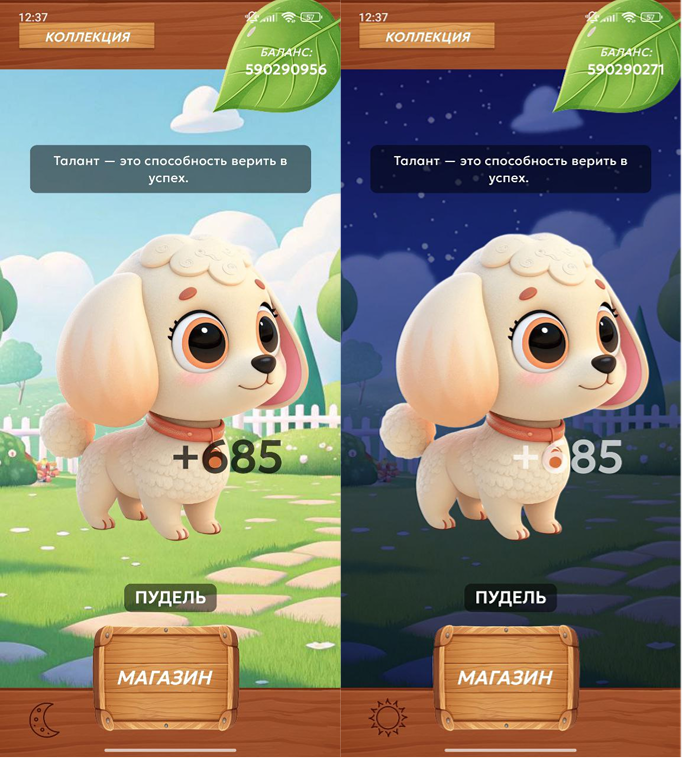
**3. Проектирование**

**3.1 Визуальный стиль и оформление через графику**

Интерфейс приложения Petty Clicker оформлен с использованием PNG-графики. Это позволило задать уникальный визуальный стиль с помощью заранее отрисованных элементов: фонов, питомцев, иконок и декоративных компонентов.

Фоны и визуальная атмосфера

Главный экран оформлен с помощью полноразмерного фонового изображения, которое создаёт визуальное настроение всей игры. При переключении темы приложения время суток на экране меняется.



Питомцы как PNG-ассеты

Каждый питомец — это индивидуальное PNG-изображение, загружаемое по имени из ресурсов. Изображения разделены по категориям: обычные, экзотические, фантастические, легендарные.



Анимации и визуальные эффекты

Анимация бонусного клика (+X) реализована поверх фонового изображения, поверх которого накладывается текст с fade-in/fade-out эффектом. Картинка питомца при этом остаётся видимой на фоне. Сам текст адаптируется под темную или светлую тему.

**3.2 Пользовательские потоки (User Flow)**

Приложение строится на простой и понятной логике. Основные пользовательские потоки:

Игровой поток:

1. Пользователь открывает приложение.
2. Просматривает баланс и тапает по экрану для получения очков.
3. Если у него есть отображаемый питомец, он взаимодействует с ним.
4. Появляется "+X" анимация клика.
5. Баланс обновляется.

Покупка питомца:

1. Пользователь переходит в магазин.
2. Выбирает животное, изучает цену и количество уже купленных.
3. Совершает покупку — питомец добавляется в коллекцию.

Коллекция:

1. Пользователь открывает коллекцию.
2. Просматривает всех приобретённых питомцев.
3. Может изменить имя питомца или отобразить его в главном экране.

Настройки темы:

Переключение между тёмной и светлой темой в любом разделе.

Все переходы сопровождаются анимацией загрузки и имеют таймаут \~300 мс.

**4. Реализация**

**4.1 Архитектура базы данных**

База данных приложения реализована с использованием Room, локального ORM-инструмента для Android. Используются две основные сущности:

1) Таблицы:

* *clicker\_data*(id (PK): Int, clickCount: Int) --- Хранит текущее количество кликов
* *purchased\_images*(id (PK),imageName, imageResId, price, animalType, customName) --- Хранит данные о купленных питомцах, включая их имя и тип

2) Связи:

Таблицы не связаны между собой напрямую (нет внешних ключей), но объединяются логикой в репозиториях и ViewModel.

Работа с базой осуществляется через DAO-интерфейсы:

* *ClickerDao* — чтение и сохранение количества кликов
* *PurchasedImageDao* — работа с коллекцией питомцев

**4.2 Репозитории и бизнес-логика**

Репозитории:

* *ClickerRepository:* обрабатывает состояние счётчика, сохраняет количество кликов в Room.
* *PurchasedImageRepository:* управляет коллекцией питомцев, их типами, кастомными именами.

ViewModel:

* *ClickerViewModel:* управляет логикой на главном экране, отвечает за авто-кликер, анимацию бонусов и реакцию на события.
* *PurchaseViewModel:* отвечает за покупку питомцев, вычисление текущей стоимости, отображение активного питомца, загрузку кастомных имён.

**4.3 Автокликер — логика и реализация**

Автокликер активируется, если у пользователя есть хотя бы один питомец типа legendary. В этом случае автоматически начисляются бонусные клики через заданные интервалы времени.

Алгоритм:

1. При запуске приложения ClickerViewModel инициирует startLegendaryAutoClicker().
2. Каждые 30 секунд проверяется наличие легендарных питомцев.
3. Если хотя бы один — каждые 5 секунд начисляется бонус, зависящий от количества питомцев.
4. Пользователь видит анимацию +X и получает прирост кликов.

**4.4 Механизм подсчёта бонуса**

Метод calculateBonusClicks() анализирует коллекцию пользователя:

* Если питомец куплен более одного раза — добавляются +2 за каждый повтор.
* Если питомец стоит дороже 500 — добавляется +4 за каждый.
* Таким образом, чем дороже и многочисленнее коллекция, тем выше бонус.

**4.5 Темизация и анимации**

Приложение использует собственный ThemeManager:

* Переключение между тёмной и светлой темой через DataStore и StateFlow.
* Темы автоматически применяются к MaterialTheme.

Анимации:

* Реализована анимация бонуса ClickBonusAnimation.
* При тапе по экрану или автоклике текст +X появляется на экране в заданной позиции и исчезает через 1 секунду.

**4.6 Технологический стек**

|  |  |  |
| --- | --- | --- |
| Категория | Технологии | Версии / особенности |
| UI | Jetpack Compose | Анимации, адаптивный интерфейс |
| DI | Dagger Hilt | Инъекция зависимостей |
| DB | Room | Локальное хранилище |
| Flow | Kotlin Coroutines + StateFlow/Flow | Реактивное управление состоянием |
| Хранилище | DataStore (Preferences) | Темизация, настройка отображения |
| Сеть (цитаты) | OkHttp + Forismatic API | Цитаты выводятся каждые 10 секунд |
| Media | MediaPlayer | Фоновая музыка |

**5. Тестирование**

**5.1 Покрытие тестами**

В проекте реализованы юнит-тесты, направленные на проверку корректности бизнес-логики и взаимодействия с базой данных.

Реализованный тест проверяет, что при покупке изображения питомца метод репозитория действительно отправляет команду на сохранение в базу. Для этого используется мок-DAO, позволяющий отследить вызов метода без обращения к реальной базе данных.

Таким образом, тест гарантирует, что при покупке питомца приложение не «забудет» сохранить его в коллекцию. Это повышает надёжность логики и позволяет своевременно выявлять ошибки при изменениях кода.

**5.2 UI Поведение**

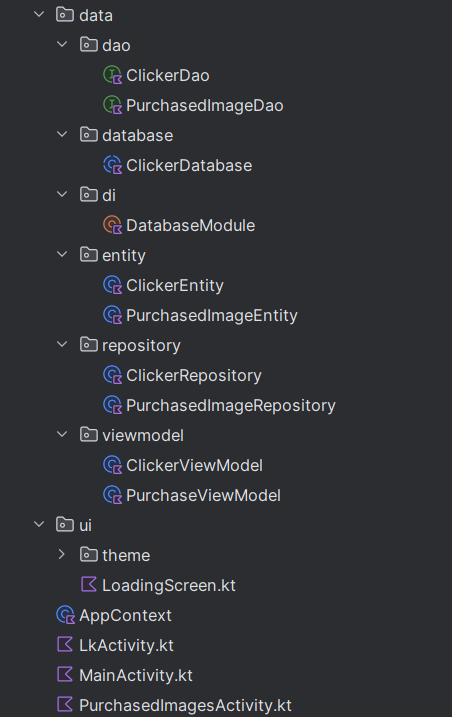
UI тестирование проведено вручную на различных экранах и состояниях приложения.

Особое внимание уделялось:

* Отображению питомцев с затемнением до покупки
* Сохранению темы при перезапуске
* Правильному отображению баланса

**6. Производительность приложения**

**6.1 Архитектурные решения**



Архитектура MVVM обеспечивает:

* Изоляцию логики от UI
* Гибкое управление состоянием через StateFlow
* Быстрый отклик UI за счёт реактивного потока данных
* Возможность масштабирования функционала (например, добавление новых категорий питомцев, монетизации, достижений)

**6.2 Особенности реализации**

Особенные аспекты реализации, повышающие производительность и UX:

* *Использование LazyColumn и LazyImage* — оптимизация прокрутки списка питомцев
* *produceState* для вычисления текущей цены без утечек памяти
* *rememberCoroutineScope и LaunchedEffect* — безопасное управление асинхронными событиями
* *StateFlow вместо LiveData* — минимизация лишних пересозданий UI
* *Ручная отмена фоновых задач* (`isActive`, `resetAutoClickEvent`) — предотвращает утечки

**7. Заключение**

**7.1 Достижения проекта**

Проект Petty Clicker успешно реализован как функциональное и визуально привлекательное Android-приложение, соответствующее современным требованиям в жанре казуальных clicker-игр.

Ключевые достижения:

* Разработан и внедрён интуитивно понятный интерфейс с поддержкой темизации (тёмная/светлая тема)
* Реализована архитектура MVVM с применением Jetpack Compose, Room, Hilt и Coroutines
* Внедрена система автокликера, активируемая наличием легендарного питомца
* Разработана гибкая система расчёта бонусов и прогрессии цен
* Реализована коллекционная система с возможностью персонализации (кастомные имена)
* Добавлена анимация бонусных кликов и API-интеграция с цитатами, что усиливает вовлечённость
* Приложение адаптировано под устройства с различными размерами экранов
* Обеспечена высокая производительность и низкое потребление ресурсов

*Итог:* Petty Clicker является завершённым MVP-продуктом, готовым к тестированию и публикации в Google Play.

**7.2 Перспективы развития**

Проект обладает высокой степенью расширяемости благодаря модульной архитектуре и использованию современных технологий.

Планы по развитию:

* Интеграция с облачной базой данных (Firebase, Supabase) — для хранения коллекции и синхронизации между устройствами
* Добавление внутриигровых покупок — монетизация через виртуальную валюту или рекламные просмотры
* Система достижений — награды за количество кликов, редких питомцев и т.д.
* Галерея питомцев и скриншоты — возможность делиться питомцами с друзьями
* Режим "Reset Prestige" — с получением особых очков за полную перезагрузку прогресса
* Локализация интерфейса — добавление английской версии для выхода на международный рынок
* Искусственный интеллект питомцев — базовые анимации или реакции на действия игрока