МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка Android-приложения kanban-доски с виртуальным питомцем

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-566.ВКР

Научный руководитель,
доцент кафедры СП, к.э.н.
_____ Т.Ю. Шабанов

Автор работы,
студент группы КЭ-403
_____ А.В. Михайлюк

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студентке группы КЭ-403
Михайлюк Александре Витальевне,
обучающейся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка Android-приложения kanban-доски с виртуальным питомцем.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**
3.1. Андерсон Д.Д. Альтернативный путь в Agile. – ООО «Манн, Иванов и Фербер», 2017. – 103 с.
3.2. Система управления проектами канбан. [Электронный ресурс] URL: https://cyberleninka.ru/article/n/sistema-upravleniya-proektami-kanban/viewer (дата обращения: 11.03.2024 г.).
4. **Перечень подлежащих разработке вопросов**
4.1. Выполнить анализ предметной области разрабатываемого приложения.
4.2. Выполнить проектирование мобильного приложения.
4.3. Реализовать мобильное приложение.
4.4. Протестировать мобильное приложение.
5. **Дата выдачи задания:** 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.э.н. Т.Ю. Шабанов

Задание принял к исполнению А.В. Михайлюк

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

В современном мире мобильные приложения стали неотъемлемой частью повседневной жизни людей. Согласно исследованиям, почти половина обладателей смартфонов использует его более 5 часов в сутки [1]. Мобильные устройства используются для различных задач – от коммуникации и работы, до развлечений и ухода за здоровьем. Одним из популярных направлений разработки мобильных приложений является создание приложений для напоминания и планирования.

Виртуальные питомцы становятся все более распространенным элементом в мобильных приложениях, предлагая пользователям не только функциональность, но и эмоциональное взаимодействие. Они могут стать неотъемлемой частью повседневной жизни пользователей, предоставляя возможность не только эффективно организовывать свои задачи, но и создавать эмоциональную связь с виртуальным питомцем, что в свою очередь способствует постоянной мотивации и вовлеченности в заботу о нем.

Цель данной работы состоит в разработке приложения для платформы Android [2] с учетом функциональности организации kanban-доски [3] и взаимодействия с виртуальным питомцем. В ходе исследования будут рассмотрены основные принципы организации задач с использованием методики kanban [4].

Постановка задачи

В рамках выпускной квалификационной работы была поставлена задача разработки Android-приложения kanban-доски с виртуальным питомцем.

Для достижения поставленной цели нужно решить следующие задачи:

1) произвести анализ предметной области;
2) выполнить проектирование приложения;
3) реализовать приложение;
4) протестировать приложение.

Структура и содержание работы

Данная работа состоит из введения, четырех глав, заключения, списка литературы и приложений. Объем работы составляет 39 страниц, объем списка литературы – 16 источников.

Первая глава «Анализ предметной области» содержит обзор аналогов мобильных проектов, на основе которых были выработаны требования к разрабатываемому приложению.

Вторая глава «Проектирование системы» содержит анализ функциональных и нефункциональных требований к разрабатываемому приложению, диаграмму вариантов использования системы, подробности проектирования архитектуры системы и описание дизайна интерфейса.

Третья глава «Реализация системы» описывает подробности реализации представленного мобильного приложения на платформе Android Studio [9], с использованием языка программирования Java [10].

В четвертой главе «Тестирование» представлено тестирование функций разрабатываемого приложения и проверка работоспособности его систем.

В главе «Заключение» описываются основные результаты, полученные при выполнении выпускной квалификационной работы, и рассматриваются дальнейшие пути развития Android-приложения kanban-доски с виртуальным питомцем.

В приложении представлены диаграмма компонентов системы и схема базы данных разрабатываемого приложения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание предметной области и обзор решений

Kanban-доска – это инструмент управления Agile-проектами [8], который помогает наглядно представить задачи, ограничить объем незавершенной работы и добиться максимальной эффективности или скорости. Она может помочь пользователям упорядочить повседневную работу. С помощью карточек и столбцов на доске Kanban пользователи могут понять, какой объем работы следует взять на себя, и выполнить его.

Уникальность этого приложения заключается в наличии питомца, что будет поддерживать интерес пользователя к успешному выполнению задач.

Для проведения анализа предметной области были рассмотрены приложения, включающие технологию kanban, а также приложение с виртуальным питомцем. В этой главе будут рассмотрены их особенности.

Приложение Pou

Pou [5] – это мобильная игра, в которой пользователю предстоит заботиться о своем виртуальном питомце по имени Pou. Основной функционал приложения представлен ниже.

1. Улучшение питомца. Игрок имеет возможность улучшать своего питомца, развивая его навыки и способности.
2. Мини-игры. В игре доступны мини-игры, которые позволяют игроку зарабатывать ресурсы и опыт для своего питомца. На рисунке 1 показан главный экран приложения.
3. Магазин внутриигровых предметов. Пользователи могут покупать различные предметы и аксессуары для своего питомца за внутриигровую валюту или реальные деньги.
4. Настроение питомца. Настроение питомца прямо зависит от того, как часто пользователь заходит в игру.

Рисунок 1 – Главный экран приложения Pou

Приложение Kanban Board

Kanban Board [6] – это приложение для управления задачами и проектами, основанное на принципах методики kanban.

Функционал приложения Kanban Board включает в себя создание и управление задачами, разделение их на колонки по статусам («В ожидании», «В процессе», «Завершено»), установку сроков выполнения задач, добавление описания и прикрепление файлов к задачам. Пользователи могут легко перемещать задачи между колонками с помощью простого интерфейса.

Также приложение Kanban Board обеспечивает возможность синхронизации данных между различными устройствами пользователя, что позволяет ему иметь доступ к задачам в любое время и в любом месте. Главный экран приложения представлен на рисунке 2.

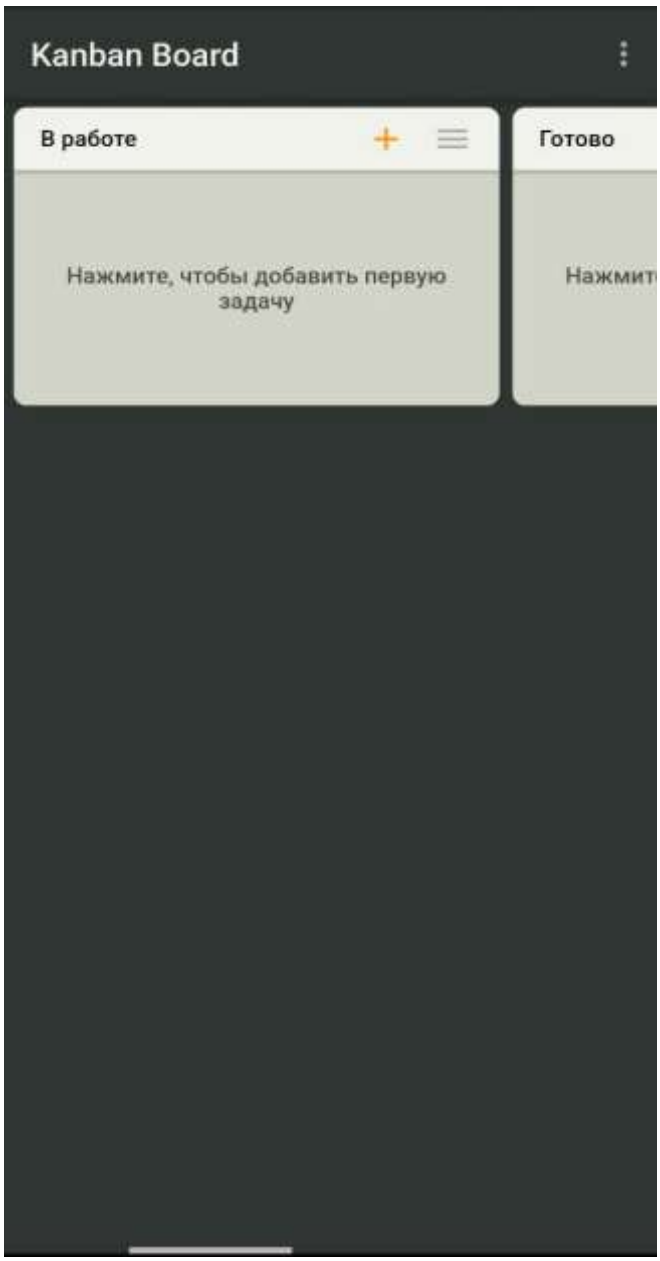

Рисунок 2 – Главный экран приложения Kanban Board

Приложение Kanbani

Приложение Kanbani [7] – приложение для управления задачами и проектами на платформе Android. Оно предоставляет пользователям возможность организовывать свои задачи с помощью гибкой системы kanban-досок.

Функциональность приложения включает в себя создание нескольких kanban-досок для различных проектов или областей жизни пользователя. Каждая доска состоит из колонок, представляющих разные стадии

выполнения задачи («К выполнению», «В процессе», «Завершено»). Пользователи могут легко перемещать задачи между колонками, отслеживая их прогресс.

Одной из отличительных черт приложения Kanbani является гибкость его настроек. Пользователи могут адаптировать интерфейс и функционал приложения под свои потребности, выбирая различные темы оформления, настраивая отображение задач и колонок, а также добавляя пользовательские поля и метки для более детальной классификации задач.

Приложение также предоставляет возможность добавления подзадач к основным задачам, установки сроков выполнения, а также напоминаний о предстоящих событиях. Кроме того, Kanbani поддерживает синхронизацию данных. Главный экран приложения Kanbani представлен на рисунке 3.

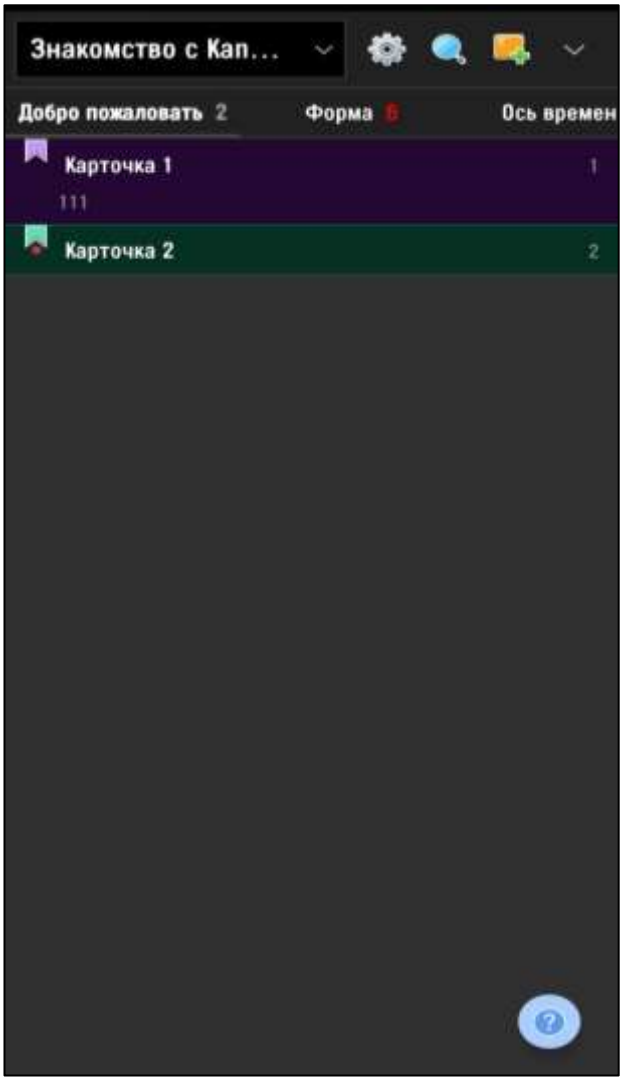

Рисунок 3 – Главный экран приложения Kanbani

1.2. Сравнение существующих аналогов

Для дальнейшего развития разрабатываемого приложения необходимо провести сравнительный анализ существующих решений, чтобы выявить их основные преимущества и недостатки. В тТаблица 1 приведены характеристики возможностей рассмотренных в предыдущей главе приложений для их сравнительного анализа.

Таблица 1 – Сравнительная характеристика рассмотренных решений

Характеристика приложения	**Приложение**		
	Pou	**Kanban Board**	**Kanbani**
Возможность добавлять, удалять, редактировать доски	Нет	Да	Да
Наличие виртуального питомца	Да	Нет	Нет
Наличие системы уведомлений	Да	Да	Да
Возможность отслеживать прогресс	Нет	Да	Да
Возможность настройки напоминаний и уведомлений	Нет	Да	Да
Наличие игровой валюты	Да	Нет	Нет
Наличие социальной функциональности (взаимодействия с другими пользователями)	Нет	Нет	Нет

Вывод по первой главе

Проанализировав структуру и функционал рассмотренных приложений, а также изучив предметную область для данного проекта, было принято решение добавить в разрабатываемое приложение следующие основные возможности:

1) возможность добавления, удаления, редактирования досок;
2) наличие виртуального питомца;
3) возможность взаимодействия с виртуальным питомцем;
4) возможность добавления, удаления, редактирования задач;
5) возможность получения валюты за выполненные задачи.

2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

В данном разделе описано проектирование разрабатываемого приложения. На основе вывода анализа предметной области и обзора существующих решений были выявлены функциональные и нефункциональные требования к системе, составлена диаграмма вариантов использования, спроектирована архитектура и разработана схема базы данных, а также были созданы макеты интерфейса мобильного приложения.

2.1. Требования

В ходе проектирования приложения были определены функциональные и нефункциональные требования к разрабатываемой системе.

Функциональные требования

Функциональные требования – это спецификации, которые определяют, что должна делать система или приложение для удовлетворения нужд и ожиданий пользователей. Данное приложение должно удовлетворять следующим функциональным требованиям.

1. Приложение должно предоставлять возможность редактировать, сохранять и удалять доски.
2. Приложение должно предоставить пользователю возможность создавать, редактировать и удалять задачи.
3. Приложение должно дать пользователю возможность получить валюту за выполнение задач.
4. Приложение должно предоставить пользователю возможность просматривать список досок.
5. Приложение должно предоставить пользователю возможность участия в мини-игре.
6. Приложение должно предоставить пользователю возможность потратить игровую валюту.

Нефункциональные требования

Нефункциональные требования – это спецификации, которые описывают характеристики системы, не связанные с конкретными функциями, а определяют качество, с которым эти функции должны быть выполнены. Данное приложение должно удовлетворять следующим нефункциональным требования.

1. Приложение должно быть реализовано на языке Java.
2. Приложение должно работать на устройствах с ОС Android.

2.2. Диаграмма вариантов использования

В соответствии с требованиями была составлена диаграмма вариантов использования, представленная на рисунке 4. Диаграмма отражает модель взаимодействия актера пользователя с разрабатываемым приложением.

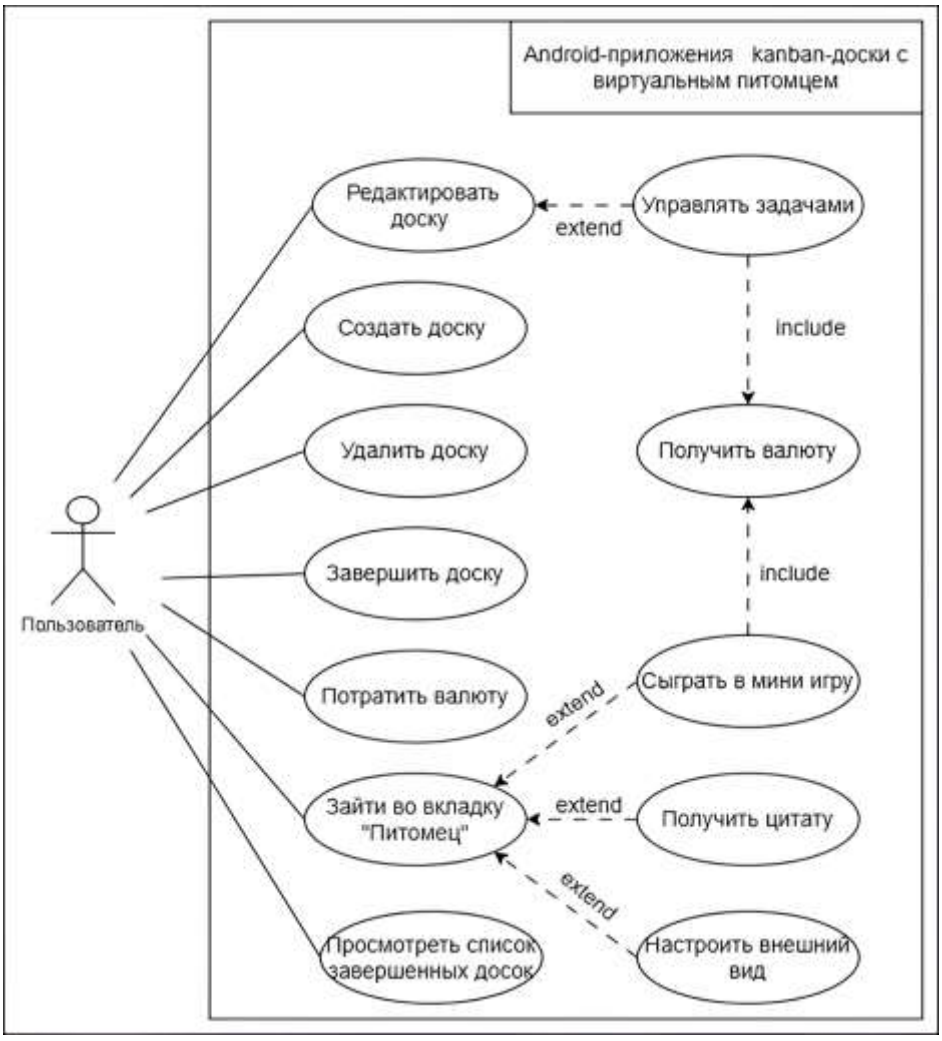

Рисунок 4 – Варианты использования приложения

С приложением взаимодействует один актер – пользователь.

Пользователь может создать, удалить, закрыть доску и просмотреть список закрытых досок.

Пользователь может редактировать доску. Эта функция включает в себя управление задачами. В нем пользователь может создать, удалить, редактировать, или изменить статус задачи («В ожидании выполнения», «В прогрессе», «Выполнено»). За повышение статуса задачи пользователь получает игровую валюту.

Пользователь может переместиться во вкладку «Питомец», где ему доступны такие функции, как получение цитат, настройка внешнего вида и мини-игра с питомцем. За мини-игру начисляется игровая валюта.

2.3. Архитектура

Архитектура разрабатываемого приложения состоит из компонентов, которые изображены на рисунке 1 приложения.

Ниже представлено описание спроектированных компонентов.

1. Компонент главного окна осуществляет связь между другими компонентами.

2. Компонент игровой валюты осуществляет возможность пользователя зарабатывать, тратить и просматривать наличие игровой валюты.

3. Компонент создания доски осуществляет возможность пользователя создавать доски.

4. Компонент удаления доски осуществляет возможность пользователя удалять доски.

5. Компонент сохранения задач позволяет пользователю просматривать созданные доски. Помимо этого, он открывает доступ к компоненту управления задачами, который включает в себя возможность создавать, редактировать, удалять, а также менять статус задачи.

6. Из компонента главного окна пользователь получает доступ к компоненту окна «питомец», который осуществляет возможность

взаимодействия пользователя с питомцем. Также пользователь получает доступ к компоненту «мини-игра», который осуществляет возможность участия пользователя в мини-игре.

7. Компонент окна «магазин» предоставляет пользователю возможность взаимодействия с компонентом «покупка контента».

8. Компонент взаимодействия с пользователем обеспечивает интерфейс для взаимодействия и передачу пользовательских команд в соответствующие компоненты для дальнейшей обработки.

9. Компонент базы данных обеспечивает хранение, доступ и управление данными, необходимыми для работы всех других компонентов.

2.4. Проектирование базы данных

Для хранения данных в разрабатываемом приложении на платформе Android Studio был использован класс `SQLiteOpenHelper` [11], представленный в приложении как `TaskDatabaseHelper`. Этот класс предоставляет механизм для создания и управления базой данных SQLite на устройстве Android.

Схема базы данных данного приложения представлена на рисунке 2 приложения.

Хранилище данных в данном приложении представлено таблицами «tasks», «purchases», «pet», «board», «items», «currency» и «game_cleanup».

Таблица «boards» содержит информацию о досках задач и имеет следующие поля.

1. `id` – поле, в котором хранится уникальный идентификатор для каждой доски, является первичным ключом.

2. `name` – поле, которое хранит название доски.

3. `status` – поле, которое хранит статус доски (например, «открыта» или «завершена»).

Таблица «tasks» содержит информацию о задачах и имеет следующие поля.

1. `id` – поле, в котором хранится уникальный идентификатор для каждой задачи, является первичным ключом.
2. `name` – поле, которое хранит название задачи.
3. `status` – поле, которое хранит статус задачи («В ожидании выполнения», «В процессе» или «Выполнено»).
4. `board_id` – внешний ключ, который ссылается на таблицу «boards» и связывает задачу с конкретной доской.
5. `created_at` – поле, которое хранит время создания задачи.

Таблица «currency» содержит информацию о текущем количестве игровой валюты и имеет следующие поля.

1. `id` – поле, в котором хранится уникальный идентификатор записи валюты, является первичным ключом.
2. `amount` – поле, которое хранит количество «фроггикоинов».
3. `love` – поле, которое хранит количество «благодарности».
4. `active_item_id` – внешний ключ, который ссылается на таблицу items и связывает запись с активным предметом.

Таблица «pet» содержит информацию о питомце и имеет следующие поля.

1. `last_petted_at` – поле, которое хранит время последнего поглаживания питомца.
2. `pet_image_res_id` – поле, которое хранит идентификатор ресурса изображения питомца.

Таблица «items» содержит информацию о товарах и имеет следующие поля.

1. `id` – поле, в котором хранится уникальный идентификатор для каждого товара, является первичным ключом.
2. `name` – поле, которое хранит название товара.
3. `image_res_id` – поле, которое хранит идентификатор ресурса

изображения товара.

Таблица «purchases» содержит информацию о покупках и имеет следующие поля.

1. `id` – поле, в котором хранится уникальный идентификатор для каждой покупки, является первичным ключом.

2. `item_id` – внешний ключ, который ссылается на таблицу «items» и связывает покупку с конкретным товаром.

3. `type` – поле, которое хранит тип покупки («фроггикоины» или «благодарность»).

Таблица «game_cleanup» содержит информацию о последней игре в мини-игру и имеет только одно поле – `last_played`, которое хранит время последней игры.

2.5. Проектирование интерфейса

MainMenu – это основное окно приложения, которое отображается первым при запуске. Оно играет ключевую роль, предоставляя пользователю доступ ко всем основным функциям и разделам приложения.

На макете представлены кнопки «Создать доску» для перехода в окно создания досок, «Мои доски» для перехода в окно просмотра созданных пользователем досок, их удаления или назначения им статуса «Завершена», «Питомец» для перехода на страницу питомца и «Магазин» для перехода на страницу магазина.

Кроме кнопок, в MainMenu также расположены информационные поля, которые отображают текущий баланс пользователя в разных валютах приложения «фроггикоины» (виртуальная валюта, которую пользователь может зарабатывать за выполнение задач) и «благодарность» (специальная валюта, которая может быть получена за взаимодействие с питомцем и ежедневное прохождение мини-игры). Макет экрана представлен на рисунке 5.

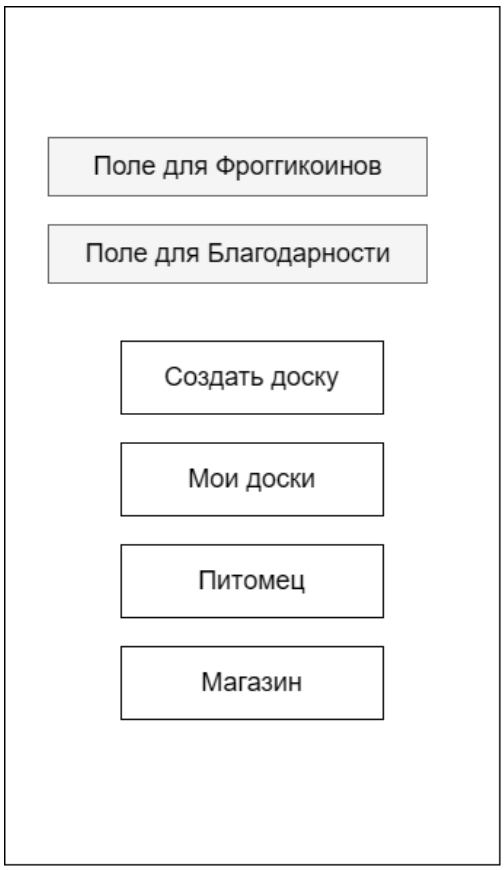

Рисунок 5 – Макет экрана MainMenu

Pet window – экран для отображения питомца. На макете представлены кнопка «Играть» для перехода в окно игровой активности, кнопка «Погладить» для взаимодействия с питомцем, за которое будет начисляться валюта «благодарность», актуальное изображение самого питомца и его статус («Счастливый», «Обычный» или «Грустный»).

Статус изменяется и отражает степень удовлетворенности питомца, что стимулирует пользователя чаще посещать приложение и ухаживать за своим виртуальным другом. Статус будет зависеть от того, как часто пользователь выполняет задачи и заходит в приложение. При нажатии на изображение питомца снизу будет всплывать случайная цитата. Само изображение питомца можно менять при помощи покупки новых скинов [16] и активации их в окне магазина. Макет экрана Pet window представлен на рисунке 6.

Рисунок 6 – Макет экрана Pet window

PayMenu – это экран для отображения внутриигрового магазина. Экран разделен на две секции для удобства пользователей: одна секция предназначена для покупок с использованием валюты «фроггикоин», а другая – для покупок, оплачиваемых валютой «благодарность». Такое разделение позволяет пользователю легко ориентироваться в доступных средствах и выбирать подходящий способ оплаты в зависимости от накопленных ресурсов.

Также на макете представлен пример товара. Он будет содержать в себе изображение товара, цену, название и кнопку «Купить», при нажатии на которую будет воспроизведена покупка товара. Также, после покупки товара, кнопка «Купить» будет сменяться кнопкой «Активировать» при взаимодействии с которой пользователь сможет использовать купленный товар в любое время. Также на этой странице, среди товаров, будет находиться изображение по умолчанию. Макет экрана PayMenu представлен на рисунке 7.

Рисунок 7 – Макет экрана PayMenu

MyBoards –это экран, который служит центром управления созданными пользователем досками. Экран разделен на два основных блока: «доски в процессе» и «завершенные доски». В блоке «доски в процессе» автоматически отображаются все новые доски, созданные пользователем, и они остаются в этом разделе до тех пор, пока не будут выполнены все задачи на доске.

Завершенной доска будет считаться только в том случае, если она не является пустой и, если все задачи внутри нее находятся в статусе «выполнено». «Завершение» происходит при долгом нажатии на доску и выборе кнопки «Завершить доску» во всплывающем окне. Также во всплывающем окне при нажатии на доску из любого блока доску можно удалить. Макет экрана MyBoards представлен на рисунке 8.

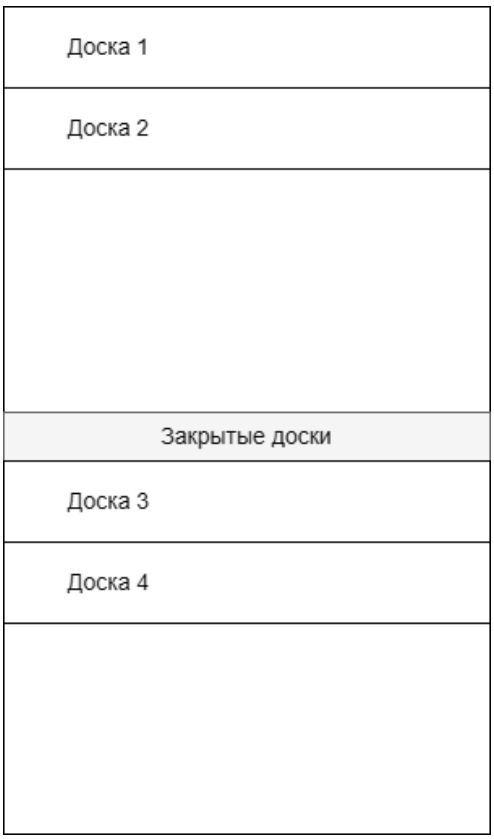

Рисунок 8 – Макет экрана MyBoards

TaskActivity – это экран для управления задачами, который встроен в каждую созданную пользователем доску. Экран содержит необходимые элементы для эффективного управления задачами: поля для ввода новых задач и кнопку «Добавить задачу», которая позволяет быстро и удобно вносить новые задачи в систему. После добавления, каждая задача автоматически попадает в поле «В ожидании выполнения», где она остаётся до тех пор, пока пользователь не начнёт её выполнение.

Для более детального контроля над процессом выполнения задач, экран TaskActivity разделён на три поля: «В ожидании выполнения», «В процессе» и «Выполнено». Эти поля позволяют пользователям не только видеть текущее состояние задач, но и управлять их переходом между различными стадиями выполнения, перемещая задачи между полями в зависимости от их статуса. За повышение статуса задачи пользователь получает «фроггикоины». Макет экрана TaskActivity представлен на рисунке 9.

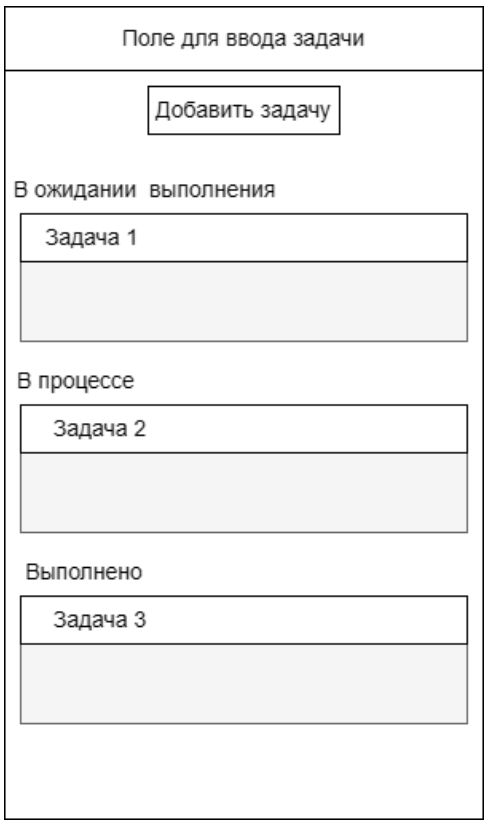

Рисунок 9 – Макет экрана TaskActivity

При нажатии на задачу будет всплывать окно изменения задачи, в котором будут располагаться поля редактирования и изменения статуса задачи, кнопка удаления задачи, и копка обновления задачи, которая будет сохранять внесенные пользователем изменения. При обновлении статуса задачи, сама задача переносится в поле, соответствующее новому статусу, выбранному пользователю. Макет экрана всплывающего окна TaskActivity представлен на рисунке 10.

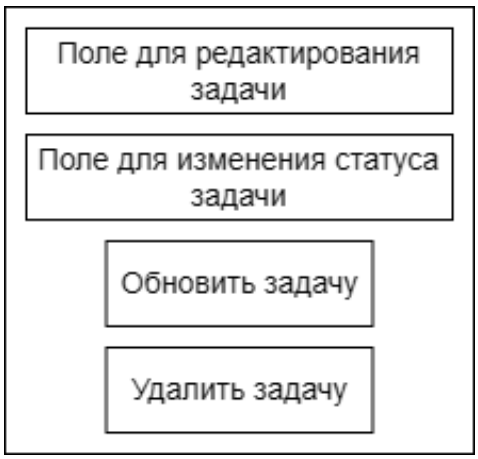

Рисунок 10 – Макет всплывающего окна TaskActivity

Вывод по второй главе

Во второй главе документации были детально сформулированы функциональные и нефункциональные требования к разрабатываемому приложению. Основные функциональные требования включают в себя возможности редактирования, удаления, сохранения и просмотра досок, что предоставляет пользователю гибкость и управляемость контентом. Кроме того, приложение обеспечивает участие в мини-играх и возможность траты игровой валюты, что добавляет элементы игрового взаимодействия и повышает заинтересованность пользователей. Важной функцией является также возможность создания и управления задачами, что делает приложение удобным инструментом для планирования и организации.

Разработанная диаграмма вариантов использования помогает представить все возможные сценарии взаимодействия пользователя с приложением. Также в рамках данной главы были спроектированы диаграмма компонентов системы, схема базы данных, для лучшего понимания внутреннего устройства приложения.

Помимо этого, были представлены макеты основных экранов приложения, включая MainMenu, где пользователи могут перемещаться по различным функциям приложения, Pet window, который предлагает интерактивное взаимодействие с виртуальными питомцами, PayMenu для осуществления покупок внутри приложения, MyBoards для управления персональными и командными досками, и TaskActivity, где пользователи могут создавать и отслеживать свои задачи.

3. РЕАЛИЗАЦИЯ СИСТЕМЫ

3.1. Средства реализации

Разработка Android-приложения kanban-доски с виртуальным питомцем была выполнена с использованием языка программирования Java в среде разработки Android Studio.

Приложение предназначено для отслеживания задач и обеспечения мотивации пользователей через взаимодействие с виртуальным питомцем.

Основные компоненты и программные средства, использованные в разработке приложения, включают в себя следующие пункты.

1. Приложение разработано на языке Java, который является одним из основных языков программирования для разработки приложений на Android.

2. Приложение разработано в Android Studio. Это интегрированная среда разработки для создания приложений под платформу Android.

3. Android SDK [12] – набор инструментов и библиотек, предоставляемых Google, для разработки приложений под Android. Android SDK обеспечивает доступ к различным компонентам Android, таким как активности, фрагменты, базы данных, сервисы и т.д.

4. Для разработки реляционной локальной базы данных была использована СУБД SQLite [13]. SQLite является легковесной встроенной базой данных, широко применяемой в приложениях Android.

5. Для создания визуального оформления приложения были использованы графические ресурсы, такие как иконки, изображения и анимации.

6. Так же был использован XML [14] – язык свободного описания структур документов и оформления экранов и окон приложения.

Модуль работы с базой данных

Функциональность для работы с базой данных в данном приложении представлена классом `DatabaseHelper`. В этом модуле определены классы и методы, связанные с созданием, обновлением и выполнением операций с базой данных, которые обеспечивают функционирование различных

аспектов данного приложения. Рассмотрим их подробнее.

Метод `onCreate` вызывается при создании базы данных и выполняет SQL-запросы [15] для создания необходимых таблиц: «boards», «tasks», «currency», «pet», «items», «purchases», «game_cleanup». Запросы определены внутри метода и содержат столбцы, такие как идентификаторы, имена, статусы, количество валюты и время последнего взаимодействия.

Метод `onUpgrade` вызывается при обновлении версии базы данных и выполняет запросы для удаления существующих таблиц и создания новых таблиц, а также для добавления новых столбцов в таблицы.

В модуле работы с базой данных определено множество методов для работы с базой данных, таких как `addBoard`, `addTask`, `deleteTask`, `updateTask`, `getCurrency`, и т.д., обеспечивающие работу и корректное обновление данных во всем приложении.

3.2. Создание, сохранение и удаление досок

Для реализации компонентов создания и сохранения досок был использован метод `addBoard`. При нажатии на кнопку «создать доску» метод получает доступ к базе данных для записи, вызывая поле для описания задачи. Приложение помещает это описание в специальный контейнер `ContentValues`, который позволяет нам передать ее в базу данных.

Когда задача успешно добавлена, база данных генерирует уникальный идентификатор для новой доски.

Идентификатор поможет приложению находить нужную доску в будущем. Далее приложение возвращает полученный идентификатор задачи. Реализация добавления и сохранения досок представлена в листинге 1.

Листинг 1 – Реализация компонентов создания сохранения доски

```
public long addBoard(String boardName) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_BOARD_NAME, boardName);
     return db.insert(TABLE_BOARDS, null, values);}
```

Алгоритм добавления доски пользователем представлен на рисунке 11.

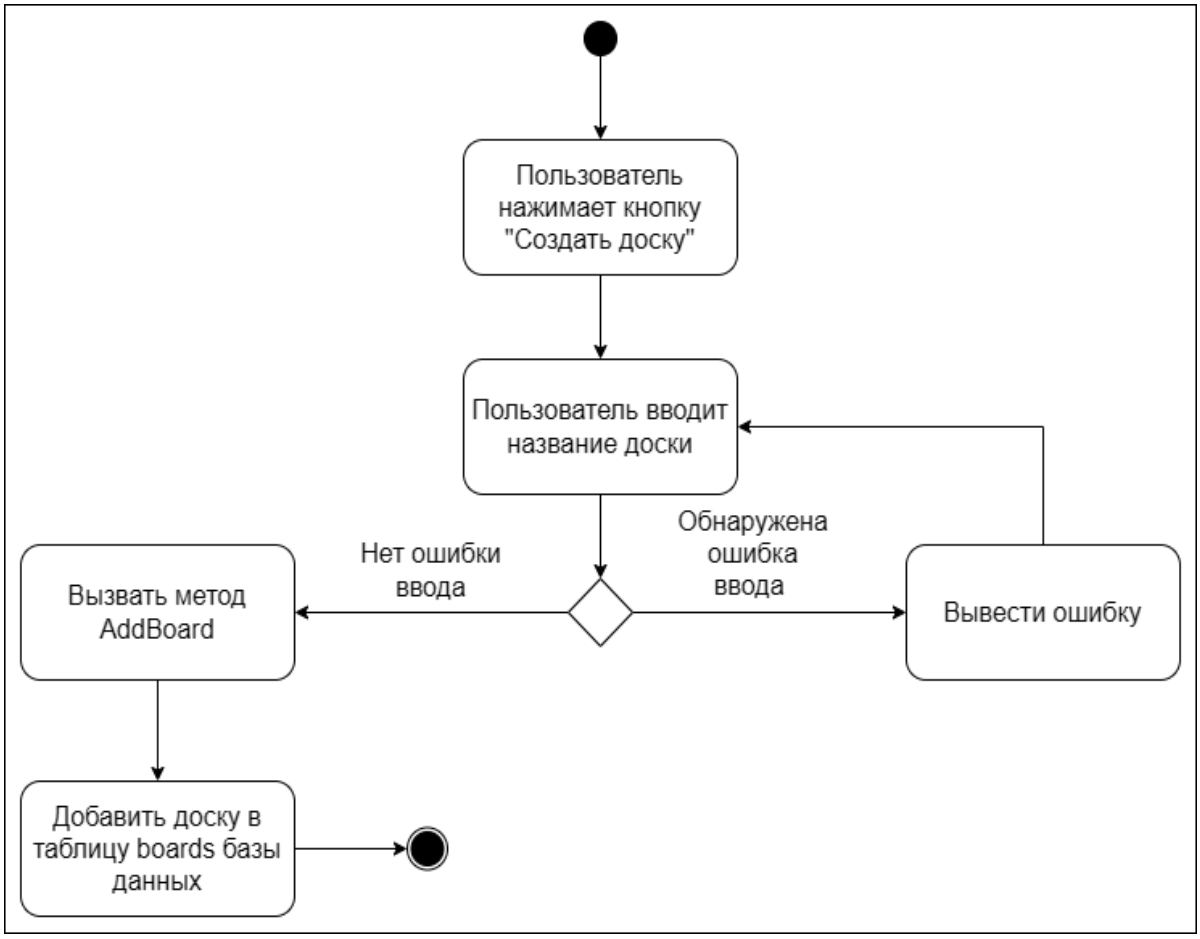

Рисунок 11 – Диаграмма деятельности алгоритма добавления доски

На рисунке 12 изображены два интерфейсных экрана мобильного приложения, которые предоставляют пользователям удобный и интуитивно понятный способ навигации по различным функциональным возможностям приложения и возможность создания досок. Эти экраны играют ключевую роль в организации пользовательского взаимодействия с приложением, позволяя удобно создавать, просматривать, сохранять и управлять досками для различных задач.

Рисунок 12 – Экраны создания и сохранения досок

Реализация «удаление доски» происходит с помощью метода `deleteBoard`. Компонент получает уникальный идентификатор задачи, которую пользователь хочет удалить. Приложение получает доступ к базе данных для записи, как и в предыдущем примере. Выполняется операция удаления доски из таблицы `boards`, с использованием метода `delete`. Приложение указывает, какую задачу удалить, используя её уникальный идентификатор. После выполнения этого метода задача будет удалена из базы данных. Код реализации функции удаления доски приведен в листинге 2.

Листинг 2 – Реализация компонента удаления доски

```
public boolean deleteBoard(long id) {
    SQLiteDatabase db = this.getWritableDatabase();
    return db.delete(TABLE_BOARDS, COLUMN_BOARD_ID + "=?", new
String[]{String.valueOf(id)}) > 0;}
```

3.3. Модуль управления задачами

Реализация модуля «управления задачами» включает в себя создание, редактирование, удаление и изменения статуса задач.

Реализация создания задачи происходит с помощью метода `addTask`. Компонент получает имя задачи, статус и идентификатор доски, к которой она относится. Приложение получает доступ к базе данных для записи и вставляет новую задачу в таблицу `tasks` с помощью метода `insert`. После выполнения этого метода новая задача будет добавлена в базу данных.

Реализация редактирования задачи происходит с помощью метода `updateTask`. Компонент получает уникальный идентификатор задачи, новое имя и новый статус. Приложение получает доступ к базе данных для записи и обновления задачи в таблице `tasks` с помощью метода `update`. После выполнения этого метода задача будет обновлена в базе данных.

Реализация удаления задачи происходит с помощью метода `deleteTask`. Компонент получает уникальный идентификатор задачи, которую пользователь хочет удалить. Приложение получает доступ к базе данных для записи и выполняет операцию удаления задачи из таблицы `tasks` с помощью метода `delete`. Приложение указывает, какую задачу удалить, используя ее уникальный идентификатор. После выполнения этого метода задача будет удалена из базы данных.

Реализация изменения статуса задачи происходит с помощью метода `updateTaskStatus`. Компонент получает уникальный идентификатор задачи и новый статус. Затем приложение получает доступ к базе данных для записи и обновления статуса задачи в таблице `tasks` с помощью метода `update`. Приложение указывает, какую задачу обновить, используя ее уникальный идентификатор. После выполнения этого метода статус задачи будет обновлен в базе данных.

Код реализации компонента «управление задачами» представлен в листинге 3.

Листинг 3 – Реализация компонента управления задачами

```
public long addTask(String taskName, String status, long boardId) {
   SQLiteDatabase db = this.getWritableDatabase();
   ContentValues values = new ContentValues();
   values.put(COLUMN_TASK_NAME, taskName); values.put(COLUMN_TASK_STATUS,
status);
   values.put(COLUMN_TASK_BOARD_ID, boardId);
   values.put(COLUMN_TASK_CREATED_AT, System.currentTimeMillis());
   return db.insert(TABLE_TASKS, null, values);
}
public boolean updateTask(long id, String taskName, String status) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_TASK_NAME, taskName);
   values.put(COLUMN_TASK_STATUS, status);
   return db.update(TABLE_TASKS, values, COLUMN_TASK_ID + " = ?", new
String[]{String.valueOf(id)}) > 0;
}
public boolean deleteTask(long id) {
    SQLiteDatabase db = this.getWritableDatabase();
    return db.delete(TABLE_TASKS, COLUMN_TASK_ID + " = ?", new
String[]{String.valueOf(id)}) > 0;
}
public boolean updateTaskStatus(long id, String status) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_TASK_STATUS, status);
    return db.update(TABLE_TASKS, values, COLUMN_TASK_ID + " = ?", new
String[]{String.valueOf(id)}) > 0;
}
```

Экран разработанного приложения, предоставляющий пользователю возможность удалять, редактировать, и обновлять статус созданных задач представлен на рисунке 13.

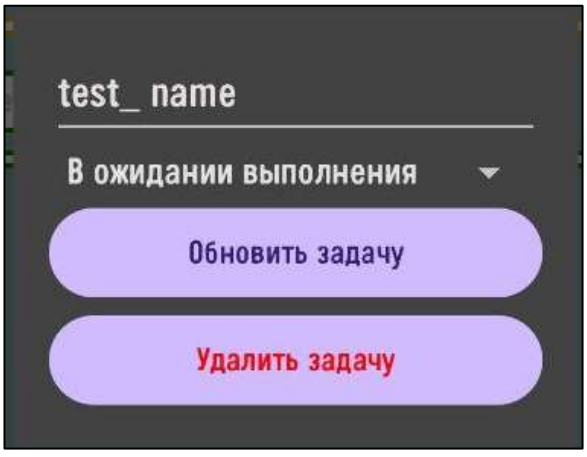

Рисунок 13 – Экран управления задачей

Экран, предоставляющий пользователю возможность просматривать и создавать задачи представлен на рисунке 14.

Рисунок 14 – Экран просмотра и создания задач

3.4. Модуль игровой валюты

В данном приложении игровая валюта представлена в двух вариантах: «фроггикоины» (froggicoins) и «благодарность» (love).

Первая валюта начисляется пользователю при повышении статуса задачи и при закрытии доски. Для начисления «фроггикоинов» используется следующий процесс: пользователь переводит задачу из статуса «В

ожидании выполнения» в «В процессе», и из статуса «в процессе» в «Выполнено». Так же пользователь получает «фроггикоины» при закрытии доски. Доску можно закрыть только при условии, что все задачи внутри доски находятся в статусе «Выполнено», также доска не должна быть пустой. При возвращении задачи в предыдущий статус, начисленные ранее «фроггикоины» списываются.

Вторая валюта, «благодарность», начисляется пользователю за прохождение мини-игры, доступной раз в день.

После выполнения действий, направленных на получение валюты, приложение вызывает метод, который увеличивает количество «фроггикоинов» и «благодарности» в базе данных и обновляет отображение их количества на главном экране.

Количество игровой валюты хранится в базе данных в таблице `currency` в столбцах `amount` и `love`. Пользователь может видеть текущий баланс игровой валюты на экране.

Код, отвечающий за обновление, получение и сохранение игровой валюты представлен в листинге 4.

Листинг 4 – Реализация функционала игровой валюты

```
public int getCurrency() {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT " + COLUMN_CURRENCY_AMOUNT + "
FROM " + TABLE_CATEGORY_CURRENCY + " WHERE " + COLUMN_CURRENCY_ID + " = 1",
null);
        if (cursor.moveToFirst()) {
            int currencyAmount = cursor.getInt(cursor.getColumnIndex-
OrThrow(COLUMN_CURRENCY_AMOUNT));
            cursor.close();
            return currencyAmount;
        } else {
            cursor.close();
            return 0;
     }
    public void updateCurrency(int amount) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COLUMN_CURRENCY_AMOUNT, amount);
        db.update(TABLE_CURRENCY, values, COLUMN_CURRENCY_ID + " = 1",
null);}
    public int getLove() {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT " + COLUMN_LOVE_AMOUNT + " FROM
" + TABLE_CURRENCY + " WHERE " + COLUMN_CURRENCY_ID + " = 1", null);
        if (cursor.moveToFirst()) {
```

```
            int loveAmount = cursor.getInt(cursor.getColumnIndex-
OrThrow(COLUMN_LOVE_AMOUNT));
            cursor.close();
            return loveAmount;
        } else{
            cursor.close();
            return 0;
        }
    public void updateLove(int amount) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COLUMN_LOVE_AMOUNT, amount);
        db.update(TABLE_CURRENCY, values, COLUMN_CURRENCY_ID + " = 1",
null);
```

3.5. Модуль магазина

Покупка игрового контента реализуется при нажатии на кнопку «купить». Сначала проверяется, был ли товар куплен ранее. Если товар не был куплен, начинается проверка количества игровой валюты. Если у пользователя есть достаточно игровой валюты, то происходит успешная покупка товара. Цена товара вычитается из общей суммы валюты, а информация о покупке сохраняется. Кнопка переключается на «активировать». Повторно товар приобрести нельзя. Если у пользователя недостаточно игровой валюты, выводится сообщение об этом. Код реализации данного функционала представлен в листинге 5.

Листинг 5 – Реализация покупки игрового контента

```
public Cursor getItem(int itemId) { SQLiteDatabase db = this.getReadable-
Database(); return db.rawQuery("SELECT * FROM " + TABLE_ITEMS + " WHERE " +
COLUMN_ITEM_ID + " = ?", new String[]{String.valueOf(itemId)});
}
public int getItemImageResId(int itemId) {
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery("SELECT " + COLUMN_ITEM_IMAGE_RES_ID
+"FROM"+ TABLE_ITEMS + " WHERE " + COLUMN_ITEM_ID + " = ?", new
String[]{String.valueOf(itemId)});
    if (cursor.moveToFirst()) {
        int resId = cursor.getInt(cursor.getColumnIndexOrThrow(COL-
UMN_ITEM_IMAGE_RES_ID));
        cursor.close();
        return resId;
    }
    return -1;
}
```

Экран разработанного приложения, позволяющий пользователю покупать товары с помощью кнопки «Купить» и активировать их с помощью

кнопки «Активировать» после покупки представлен на рисунке 15.

Рисунок 15 – Экран магазина

Вывод по третьей главе

В соответствии с целью работы были реализованы все необходимые компоненты приложения и его пользовательский интерфейс, а также разработано само мобильное приложение. Разработанное приложение полностью соответствует сформированным требованиям на этапе проектирования.

4. ТЕСТИРОВАНИЕ

Для тестирования системы применялось функциональное тестирование, т.е. тестирование программного обеспечения в целях проверки реализуемости функциональных требований. Тестирование мобильного приложения проводилось вручную. Результаты тестирования представлены в таблице 2.

Таблица 3 – Тестирование готового приложения

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1	Создание новой доски	1. Нажать кнопку «создать доску». 2. Ввести описание задачи. 3. Сохранить задачу.	Доска успешно создана и сохранена в базе данных	Да
2	Создание пустой доски	1. Нажать кнопку «создать доску». 2. Сохранить без описания.	В всплывающем уведомлении показана ошибка	Да
3	Удаление созданной ранее доски.	1. Нажать кнопку «мои доски». 2. Выбрать доску и удалить долгим нажатием.	Доска успешно удалена	Да
4	Создание задачи в созданной ранее доске	1. Нажать кнопку «мои доски». 2. Выбрать доску и нажать на нее. 3. Добавить задачу. 4. Сохранить задачу.	Задача успешно создана и сохранена в поле «ожидает выполнения».	Да
5	Удаление/редактирование задачи в доске.	1. Нажать кнопку «мои доски». 2. Выбрать доску и нажать на нее. 3. Выбрать задачу и нажать на кнопки «редактировать» или удалить».	Задача успешно удалена/отредактирована.	Да
6	Изменения статуса задачи.	1. Нажать кнопку «мои доски». 2. Выбрать доску и нажать на нее. 3. Нажать на созданную задачу. 4. Выбрать новый статус задачи. 5. Нажать на кнопку «обновить задачу».	Статус задачи обновлен, сама задача перенесена в поле нового статуса.	Да
7	Обновление игровой валюты	1.Открыть главную страницу. 2. Нажать кнопку «мои доски». 3. Выбрать доску и нажать на нее. 4.Выбрать задачу и нажать кнопку «выполнено» рядом с созданной ранее задачей.	В счетчике игровой валюты уже имеющееся значение увеличится на 5 единиц, а сама задача исчезнет.	Да

Окончание таблицы 2

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
8	Запуск мини-игры	1. Перейти на страницу «питомец». 2. Нажать кнопку «Мини-игра» в окне питомца.	Откроется новая вкладка с игровой активностью.	Да
9	Игровая активность	1. Перейти на страницу «питомец». 2. Нажать кнопку «Мини-игра». 3. Успеть прожать на все появляющиеся на экране элементы до появления последнего.	При успешном прохождении появляется табличка «спасибо за помощь!» и начисляется игровая валюта.	Да
10	Игровая активность	1.Перейти на страницу «питомец». 2. Нажать кнопку «Мини-игра». 3. Не прожать на все появляющиеся на экране элементы до появления последнего.	При неудачном прохождении снизу появляется табличка «попробуй еще раз». Игровая валюта не начисляется.	Да
11	Получение цитаты	1. Перейти на страницу «питомец». 2. Нажать на изображение питомца;	Цитата на странице меняется, когда пользователь выходит из нее, и исчезает при нажатии.	Да
12	Покупка контента	1. Перейти на страницу «магазин». 2. Нажать кнопку «купить».	Игровая валюта списывается. Кнопка «купить» меняется на кнопку «активировать».	Да
13	Активация контента	1. Перейти на страницу «Магазин». 2. Выбрать контент и нажать кнопку «активировать».	Изображение питомца в окне «питомец» изменилось.	Да

Вывод по четвертой главе

В четвертой главе разработанное мобильное приложение было протестировано набором функциональных тестов. Все тесты были выполнены успешно.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было разработано Android-приложение kanban-доска с виртуальным питомцем. В процессе разработки были выполнены следующие задачи.

1. Произведен анализ предметной области.
2. Выполнено проектирование приложения.
3. Произведена реализация приложения.
4. Выполнено тестирование приложения.

В результате был получен опыт работы со средой разработки Android Studio и языком программирования Java. Также была изучена литература для решения поставленных вопросов. Опыт, полученный в процессе написания данного приложения, послужит хорошей основой для созданий мобильных приложений в будущем.

В дальнейшем планируется продолжение разработки и улучшение данного приложения, расширение существующих и внедрение новых функций. Например, возможность аутентификации пользователя, создание уведомлений и расширение количества настроек для kanban-досок и увеличение количества мини-игр.

ЛИТЕРАТУРА

1. Android SDK: что это и для чего нужен. [Электронный ресурс] URL: https://blog.skillfactory.ru/glossary/android-sdk/ (дата обращения: 25.03.2024 г.).

2. SQLite – что это? [Электронный ресурс] URL:https://blog.skillfactory.ru/glossary/sqlite/ (дата обращения: 25.03.2024 г.).

3. Каждый четвертый пользователь смартфона проводит с ним более 7 часов. [Электронный ресурс] URL: https://mobilereview.com/news/kazhdyj-chetvertyj-polzovatel-smartfona-provodit-s-nimbolee-7-chasov (дата обращения: 12.02.2024 г.).

4. Канбан: методология, инструменты и принципы системы. [Электронный ресурс] URL: https://kachestvo.pro/kachestvoupravleniya/ berezhlivoe-proizvodstvo/kanban-metodologiya-instrumenty-i-printsipy-sistemy/ (дата обращения: 12.02.2024 г.).

5. Об Android | Android. [Электронный ресурс] URL: https://www.android.com/intl/ru_ru/what-is-android/ (дата обращения: 12.02.2024 г.).

6. Описание расширяемого языка разметки (XML). [Электронный ресурс] URL: https://aws.amazon.com/ru/what-is/xml/ (дата обращения: 25.03.2024 г.).

7. Приложения в Google Play – Kanban Board. [Электронный ресурс] URL:https://play.google.com/store/apps/details?id=app.diesel.kanban_board&hl=en_US (дата обращения: 13.03.2024 г.).

8. Приложения в Google Play – Kanbani: Органайзер и планировщик. [Электронный ресурс] URL:https://play.google.com/store/apps/details?id=org.pdapps.kanbani&hl=ru&gl=US (дата обращения: 13.03.2024 г.)

9. Приложения в Google Play – Pou. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=me.pou.app&hl=en_US (дата обращения: 13.02.2024 г.).

10. Применение SQLiteOpenHelper. [Электронный ресурс] URL: https://habr.com/ru/companies/ruvds/articles/552938/ (дата обращения: 24.03.2024 г.).

11. Управление проектами по методике Agile. [Электронный ресурс] URL: https://www.atlassian.com/ru/agile/project-management (дата обращения: 20.03.2024 г.).

12. Что такое Android Studio и как ей пользоваться. [Электронный ресурс] URL: https://skillbox.ru/media/code/chto-takoe-android-studio-i-kak-ey-polzovatsya/ (дата обращения: 20.03.2024 г.).

13. Что такое java? [Электронный ресурс] URL:https://aws.amazon.com/ru/what-is/java/ (дата обращения: 20.03.2024 г.).

14. Что такое kanban-доска? [Электронный ресурс] URL: https://www.atlassian.com/ru/agile/kanban/boards (дата обращения: 20.03.2024 г.).

15. Что такое SQL? [Электронный ресурс] URL:https://aws.amazon.com/ru/what-is/sql/ (дата обращения: 27.03.2024 г.).

16. Что такое скины? [Электронный ресурс] URL: https://blog.eldorado.ru/publications/chto-takoe-skiny-v-igrakh-37036 (дата обращения: 27.03.2024 г.).

ПРИЛОЖЕНИЕ. Диаграммы приложения

Архитектура приложения

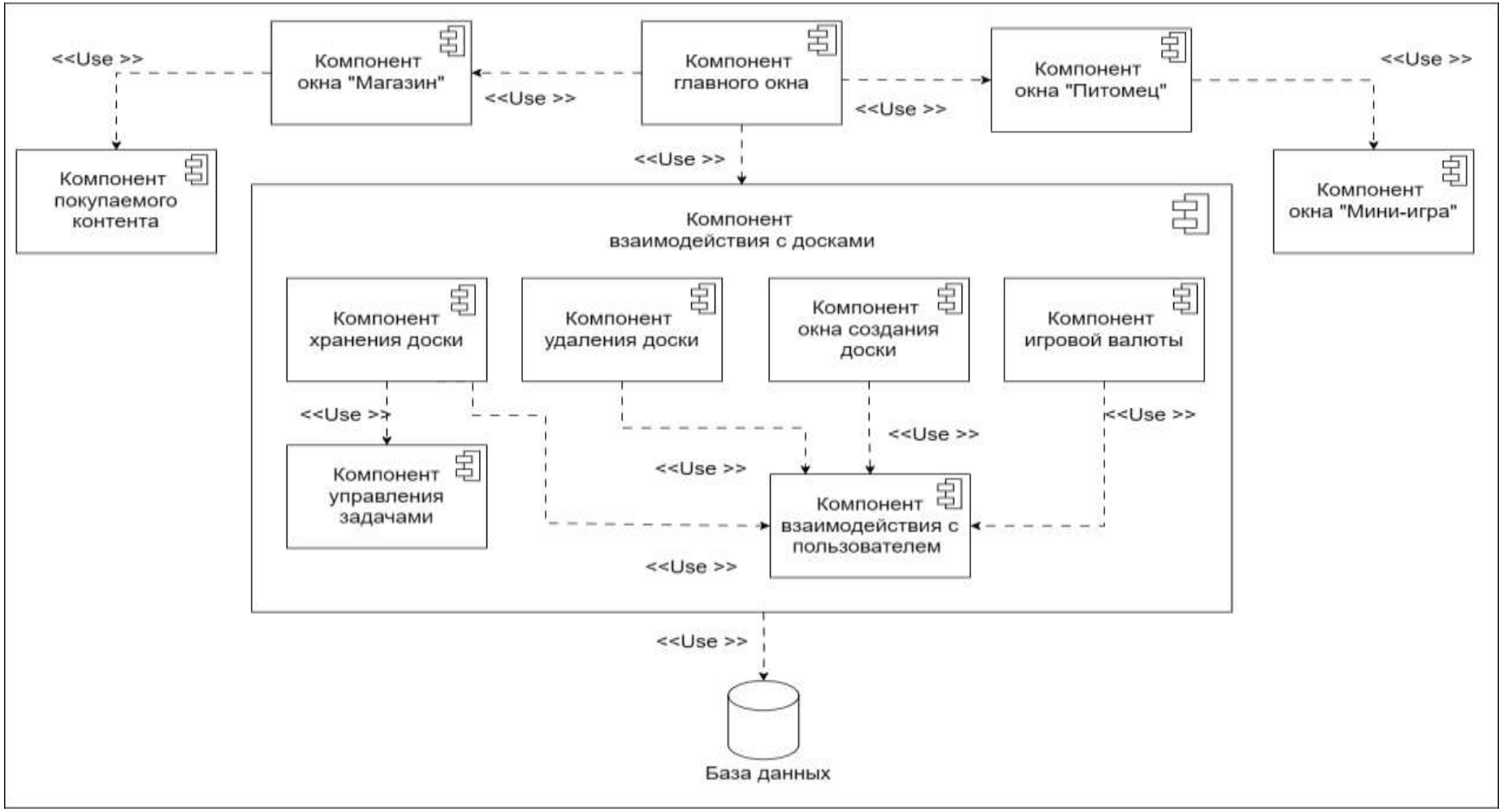

Рисунок 1 – Диаграмма компонентов системы

Схема базы данных приложения

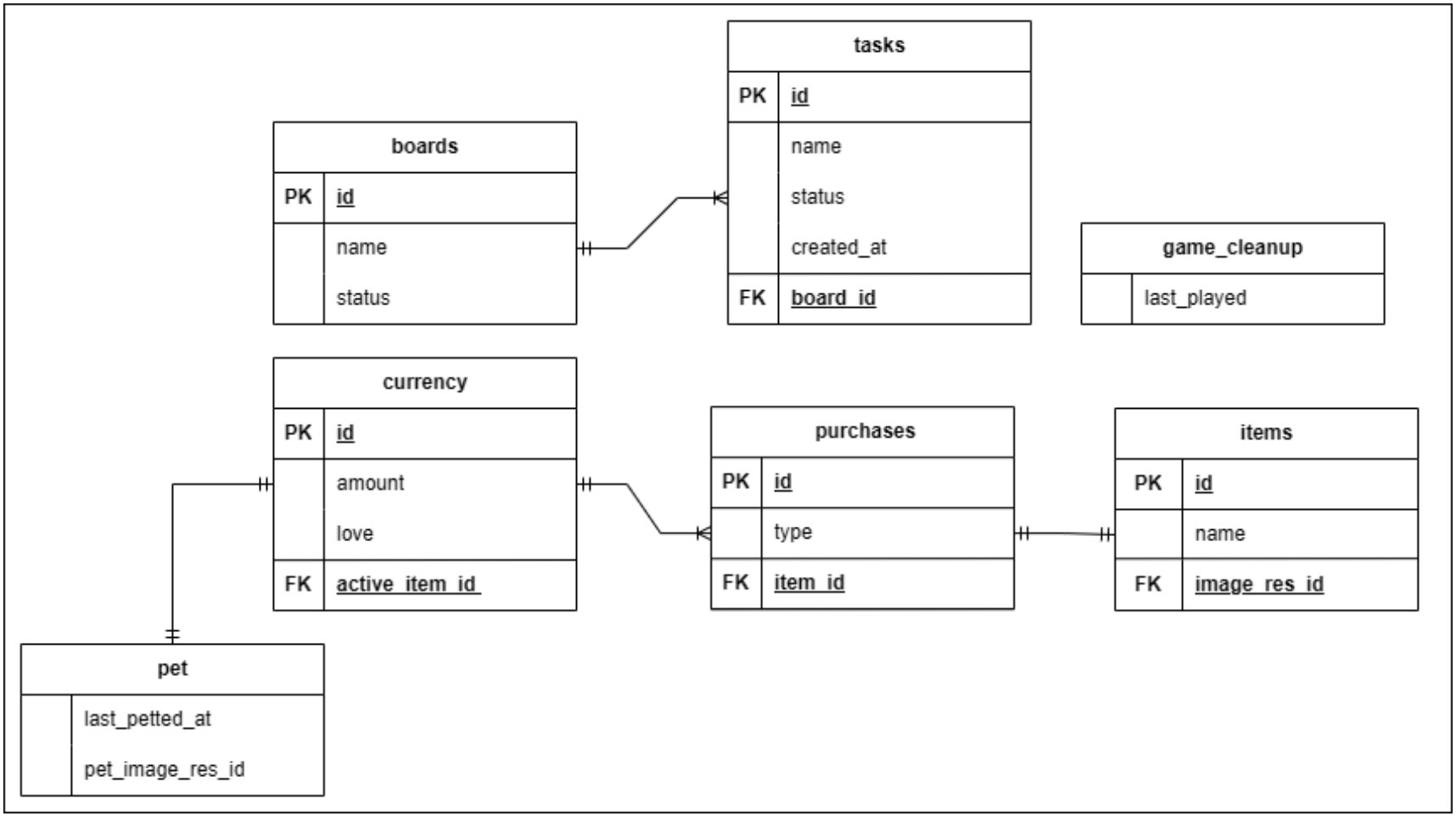

Рисунок 2 – Схема базы данных