

**ОТЧЕТ**

по учебной практике

**УП.03.01. «Участие в интеграции программных модулях»**

по специальности 09.02.03 «Программирование в компьютерных системах»

Выполнил студент гр. П1-18

Герасимов Дмитрий Александрович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись)

Проверил преподаватель

Гусятинер Л. Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Королев, 2021

**Оглавление**

[1. Проектирование и разработка программы «Блокнот» 2](#_Toc88043966)

[1.1 Коллективная разработка программы «Блокнот» 2](#_Toc88043967)

[1.2 Тестирование, отладка, оптимизация программы «Блокнот» 2](#_Toc88043968)

[2. Проектирование и разработка программы «QuizCards» 4](#_Toc88043969)

[2.1. Построение UML-диаграмм к программе «QuizCards» 4](#_Toc88043970)

[2.2. Разработка программы «QuizCards» 5](#_Toc88043971)

[2.3. Тестирование, отладка, оптимизация программы «QuizCards» 8](#_Toc88043972)

# Проектирование и разработка программы «Блокнот»

## Коллективная разработка программы «Блокнот»

Работа над программой «Блокнот» велась совместно двумя разработчиками, используя систему контроля версий Git. Разработчики параллельно разрабатывали отдельные модули программы.

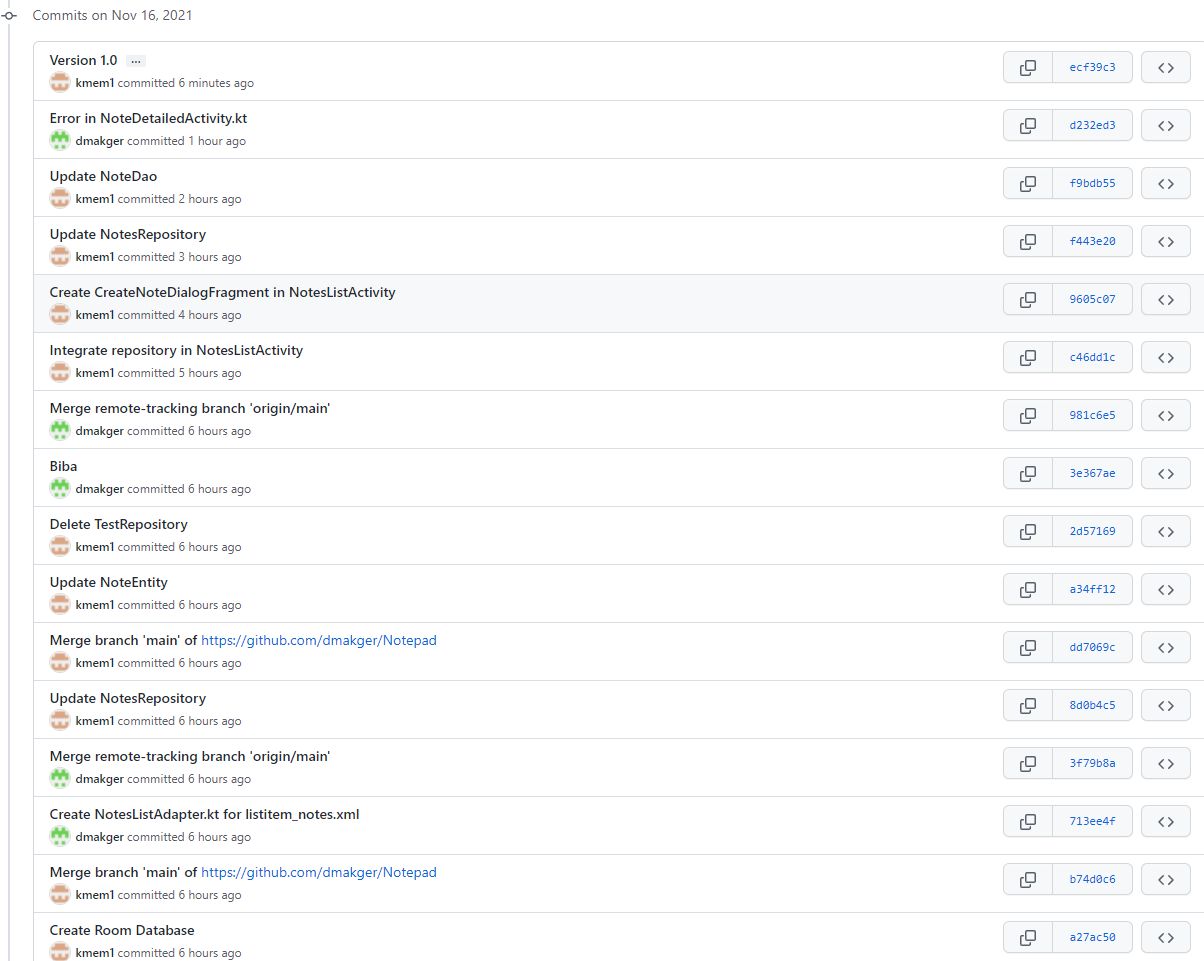


Рисунок 1. Список изменений разработчиков

## Тестирование, отладка, оптимизация программы «Блокнот»

В процессе разработки были написаны тесты для проверки работоспособности программы. При обнаружении ошибок проводилась отладка программы. Оптимизация программы не требовалась.



Рисунок 2. Тест проверки работоспособности программы

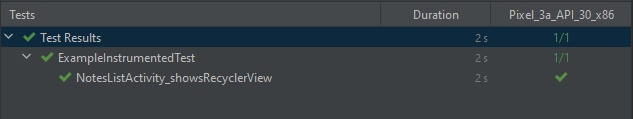


Рисунок 3. Успешное прохождение теста

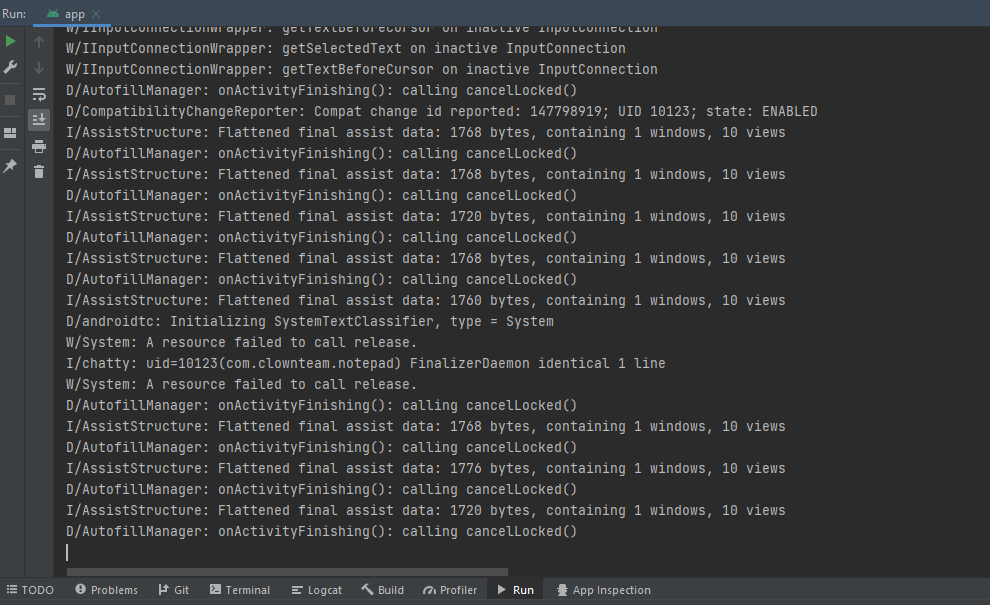


Рисунок 4. Отладка программы

# Проектирование и разработка программы «QuizCards»

## Построение UML-диаграмм к программе «QuizCards»

Работа над программой «Блокнот» велась совместно двумя

В данном разделе содержится диаграмма прецедентов для приложения, на которой показаны возможные функциональные отношения. (рис. 5)

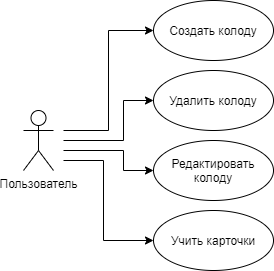


Рисунок 5 Диаграмма прецедентов

В данном разделе находятся диаграммы классов приложения. (рис 6, 7)

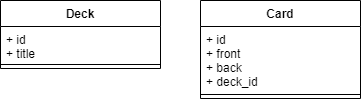


Рисунок 6 Диаграмма классов основных сущностей

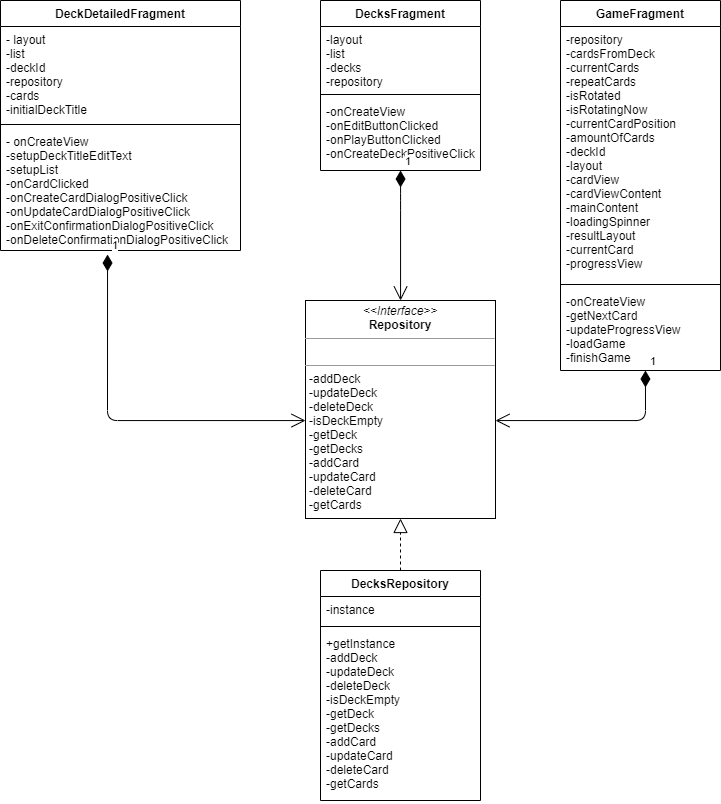


Рисунок 7 Диаграмма классов

## Разработка программы «QuizCards»

В разработке приложений на Android используются в основном либо Java, либо Kotlin. Данный курсовой проект написан на Kotlin.

**Kotlin [1]** - язык программирования, разработанный компанией JetBrains, работающий на платформе Java. Он использует JDK, как и сама Java, но имеет другой синтаксис. Синтаксис языка использует элементы из Паскаля, TypeScript, Haxe, PL/SQL, F#, Go и Scala, C++, Java, C#, Rust и D. При объявлении переменных и параметров типы данных указываются после названия (разделитель - двоеточие). Точка с запятой как разделитель операторов так же необязательна. Кроме объектно-ориентированного подхода, Kotlin также поддерживает процедурный стиль с использованием функций. Как и в Си, C++ и D, точка входа в программу – функция main, принимающая массив параметров командной строки. Также поддерживается вывод типов.

В сравнении класса Java с эквивалентным классом Kotlin демонстрирует лаконичность кода Kotlin. Для той же операции, что выполняется в классе Java, класс Kotlin требует меньше кода. Одно из основных различий между Java и Kotlin заключается в том, что в последнем нет условий для проверяемых исключений (checked exception). Следовательно, нет необходимости отлавливать или объявлять какие-либо исключения. Несомненным преимуществом является полная совместимость с Java, в том числе и обратная. Все библиотеки для Java будут работать на Kotlin и наоборот. Также Kotlin позволяет разработчикам расширять класс новыми функциями с помощью функций расширения. Эти функции недоступны в Java. В отличие от Java, в Kotlin все типы по умолчанию являются не-nullable. Если разработчики попытаются присвоить или вернуть значение null в коде Kotlin, во время компиляции произойдет сбой.

Kotlin помог разработчикам писать программы с меньшим количеством кода. Помимо всего того, что есть в Java, он добавляют вещи из мира функционального программирования. Это значительно облегчает написание кода - делает его короче и выразительнее.

**Android Studio [2]** — интегрированная среда разработки (IDE) для работы с платформой Android.

Android Studio, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, — официальное средство разработки Android приложений. Данная среда разработки доступна для Windows, macOS и GNU/Linux. 17 мая 2017, на ежегодной конференции Google I/O, Google анонсировал поддержку языка Kotlin, используемого в Android Studio, как официального языка программирования для платформы Android в дополнение к Java и С++.

В разработке приложений на Android в основном в качестве базы данных используется **SQLite [11]**. Причины использования:

1. **Минимальные затраты ресурсов.** Для работы большинства систем управления базами данных необходим специальный процесс сервера базы данных. SQLite обходится без сервера: база данных SQLite представляет собой обычный файл. Когда БД не используется, она не расходует процессорное время. Это особенно важно на мобильных устройствах, чтобы избежать разрядки аккумулятора.
2. **Оптимизация для одного пользователя.** С базой данных взаимодействует только наше приложение. Поэтому можно обойтись без идентификации с именем пользователя и паролем.
3. **Надежность и быстрота.** Базы данных SQLite поддерживают транзакции баз данных. Кроме того, операции чтения и записи данных реализуются на оптимизированном коде С.

Рассмотрим функцию создания колоды:

***Листинг 1.*** Обработчик нажатия на положительную кнопку в диалоговом окне для создания колоды

override fun onCreateDeckDialogPositiveClick(dialog: DialogFragment, deckName: String) {

if (deckName == "") return

MainScope().launch {

val deck = Deck(0, deckName)

repository.addDeck(requireContext(), deck)

decks.clear()

decks.addAll(repository.getDecks(requireContext()))

list.adapter?.notifyDataSetChanged()

}

}

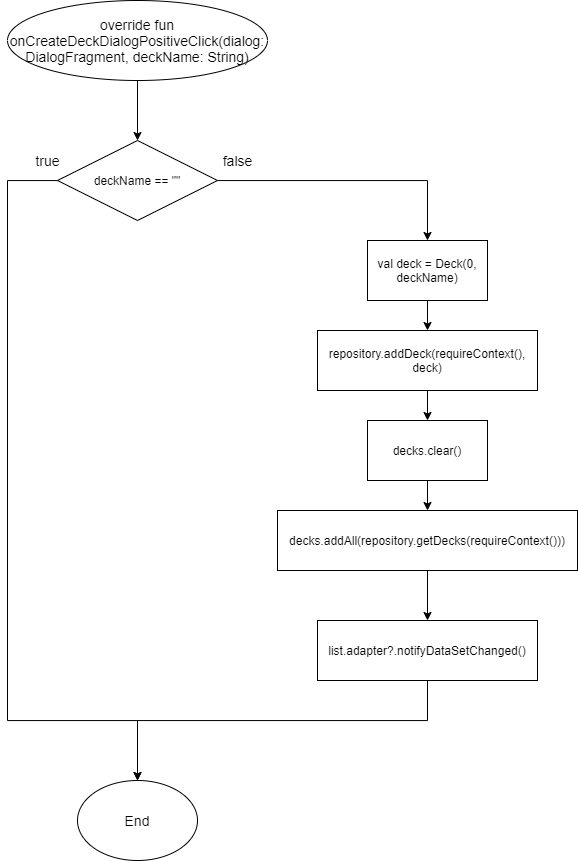


Рисунок 8 Блок-схема функции onCreateDeckDialogPositiveClick

## Тестирование, отладка, оптимизация программы «QuizCards»

В процессе разработки были написаны тесты для проверки работоспособности программы. При обнаружении ошибок проводилась отладка программы. Оптимизация программы не требовалась.

* + 1. Тестирование

В некоторых модулях производится проверка данных на корректность, чтобы сообщить пользователю о проблеме и предотвратить аварийное завершение программы.

**Тест №1:**

Колода из 5 карт

Входные данные: 3 запомненных карт, 2 забытых.

Ожидаемый результат: Осталось карт: 2.

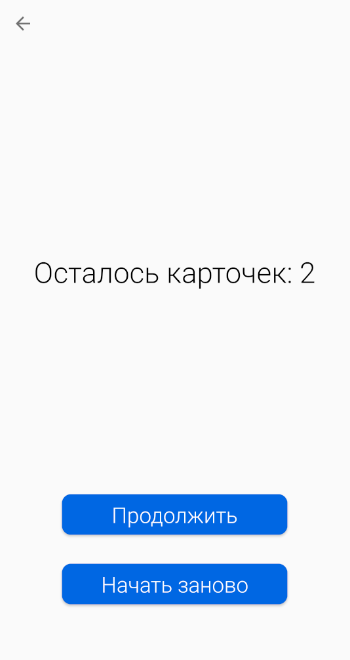


Рисунок 9. Тест №1

**Тест №2:**

Колода из 5 карт

Входные данные: Все карты запомнены.

Ожидаемый результат: Вы завершили колоду

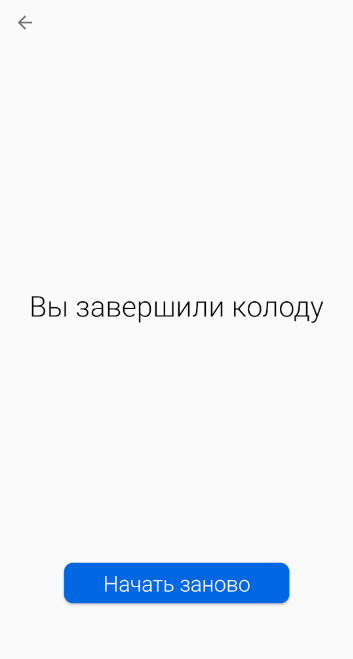


Рисунок 10. Тест №2

**Тест №3.**

Колода из 5 карт.

Входные данные: Все карты забыты.

Ожидаемый результат: Осталось карт: 5.

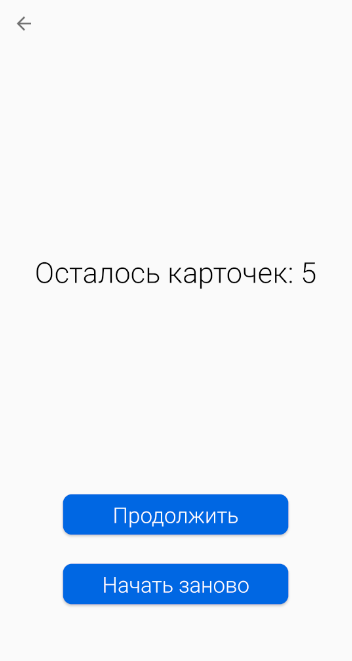


Рисунок 11. Тест №3

* + 1. Отладка

**Отладка №1:**

Добавление в базу данных

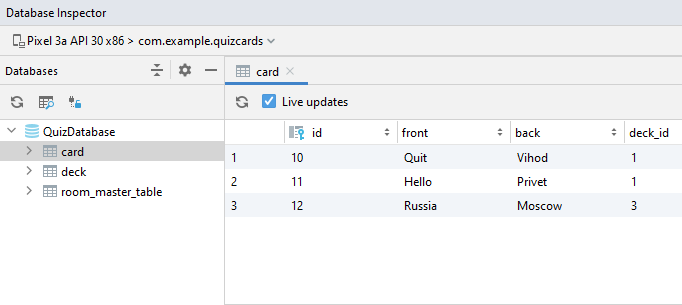


Рисунок 12. Добавление в БД. База данных

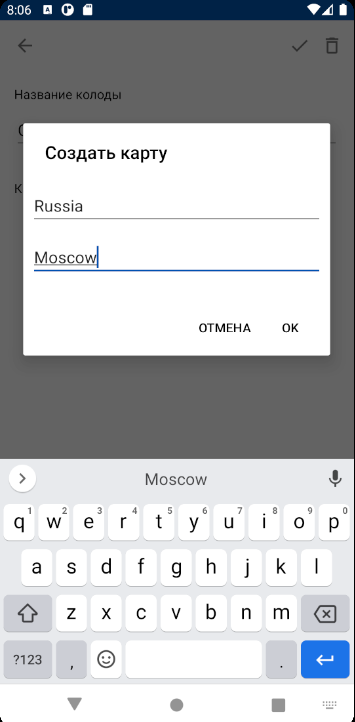


Рисунок 13. Добавление в базу данных. Приложение

**Отладка №2:**

Удаление колоды из базы данных

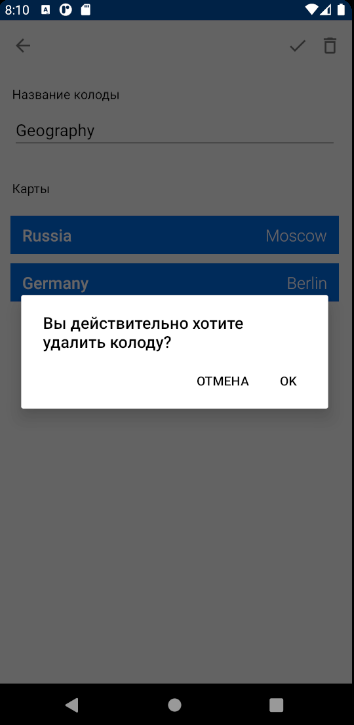


Рисунок 14. Удаление колоды из БД. Приложение

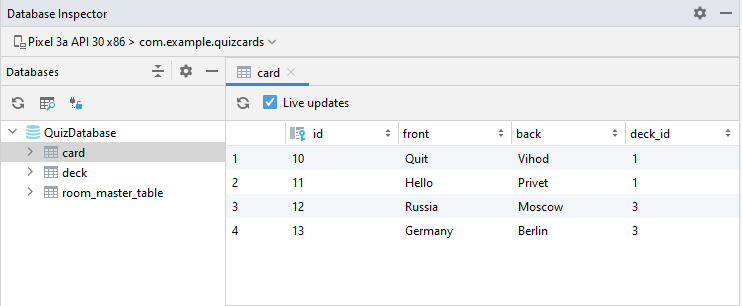


Рисунок 15. До удаления колоды из БД.

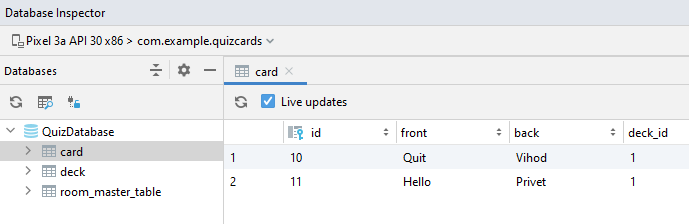


Рисунок 16. После удаление колоды из БД.