Санкт-Петербургский государственный политехнический университет

Институт компьютерных наук и технологий

**Кафедра «Компьютерные системы и программные технологии»**

**Курсовой проект**

**Дисциплина**: Проектирование мобильных приложений

**Тема**: Blackjack

Выполнил студент гр. 3530901/70203 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Иванов И.Д.

*(подпись)*

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Егорова И.С.

*(подпись)*

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Санкт-Петербург

2019

**Оглавление**

[Введение 3](#_Toc27979977)

[База данных 4](#_Toc27979978)

[MVVM(Мodel View ViewModel) 4](#_Toc27979979)

[Model 4](#_Toc27979980)

[View/ViewModel 5](#_Toc27979981)

[Тестирование 9](#_Toc27979982)

[Вывод 10](#_Toc27979983)

[Список литературы 11](#_Toc27979984)

# **Введение**

Темой данного курсового проекта является написание игры Blackjack на базе Android. Для реализуемого приложения были обозначены требования: использование возможностей камеры и баз данных. Также, необходимо было написать Unit и UI тесты. В качестве языка программирования был выбран Kotlin. Далее в отчёте будет приведено описание проекта.

## **База данных**

База данных сделана с помощью библиотеки Room.

В приложении используются:

RoomEntry – класс данных, в котором описаны элементы, содержащиеся в базе данных. В данном случае это идентификатор, имя пользователя и кол-во выигранных и проигранных партий.

RoomDao – интерфейс для доступа к базе данных. Содержит методы для добавления новой записи, обновления существующей записи, для получения записи по имени пользователя, а также для получения всех записей.

RoomDatabase – класс, содержащий все объекты для доступа к БД.

## **MVVM(Мodel View ViewModel)**

### **Model**

Для реализации игры Blackjack было разработано 5 моделей:  
1. Card

2. Player

3. Dealer

4. Deck  
5. GameSession

1. Card(Карта)

Класс, используемый для описания игральной карты. Объект класса создаётся с двумя аргументами: значением карты и мастью.

Содержит метод получения величины карты при игре в blackjack, а также метод для преобразования карты к строке.

1. Player(Игрок)

Класс, используемый для описания игрока. В нём хранятся текущие карты игрока, а также их текущее суммарное значение.

Содержит метод для получение игроком новой карты.

1. Dealer(Дилер)

Наследуется от класса Player, используется для описания дилера. Содержит метод wantToHit(), регулирующий игру дилера.

1. Deck(Колода)

Класс, используемый для описания колоды. Содержит метод, возвращающий случайно сгенерированную карту.

1. GameSession (Игра)

Класс, описывающий игровую сессию. Содержит колоду и двух игроков, а также методы, управляющие ходом игры(hit(), stand() и т.д.)

### **View/ViewModel**

В приложении реализовано 8 классов, которые отображают пользовательский интерфейс или связывают модели с представлениями:

* MainActivity

Activity, содержащая NavHostFragment, служащий для навигации в приложении.

* ChooseCoverActivity

Activity, в которой пользователю предоставляется возможность выбора одной из существующих обложек карт, а также возможность установить собственную обложку, сделав фотографию.

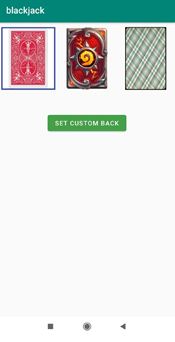


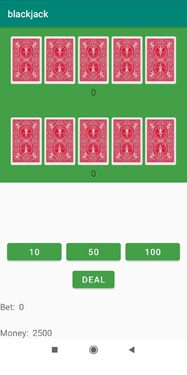
Рис.1.

* BaseFragment

Класс, являющийся родительским классом для всех фрагментов. Содержит метод для отображения layout-ресурса на экране. Используется для уменьшения объёма повторяющегося кода.

* GameFragment

Фрагмент, отображающий основной экран игры. Содержит игровое поле, счётчики для значений карт игрока и дилера, а также кнопки для управления ходом игры. Отображаемый layout содержит 3 дочерних layout-а, заменяющих друг друга в зависимости от стадии игры. При завершении очередной партии показывает Toast, содержащий информацию о результатах партии.

  
Рис.2.

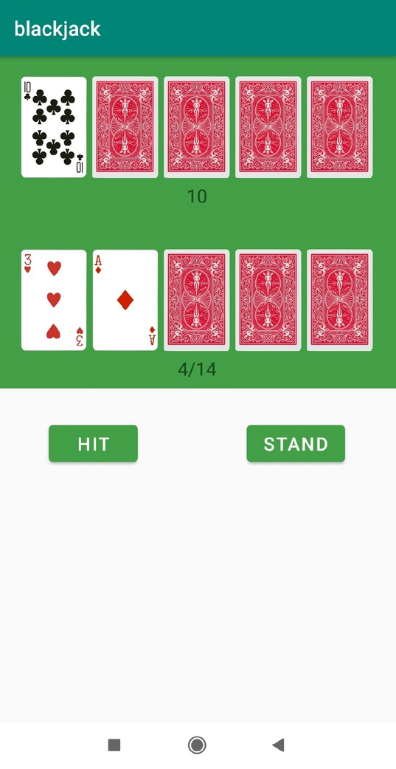


Рис.3.

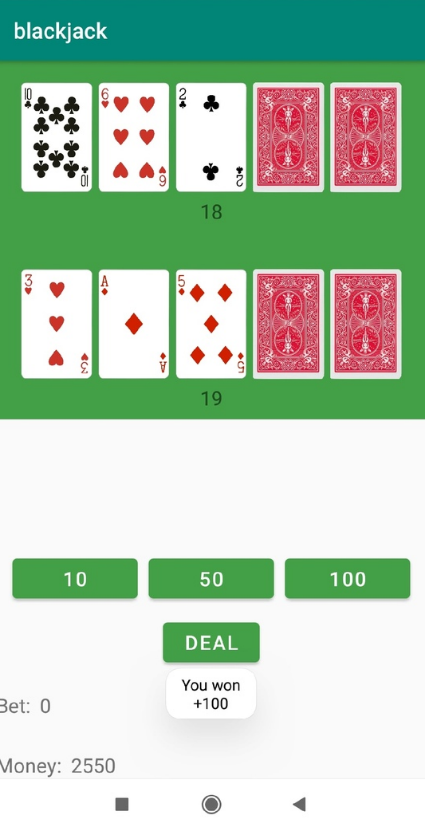


Рис.4.

* MenuFragment  
   Фрагмент, отображающий главное меню приложения. Содержит кнопки для перехода в окно игры, окно со статистикой и в окно выбора обложки карт. При попытке перехода в окно игры запрашивает имя пользователя.

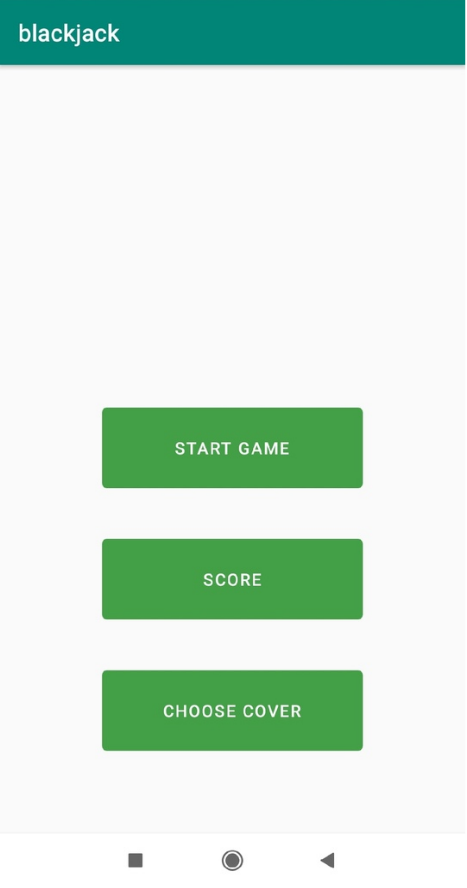


Рис.5.

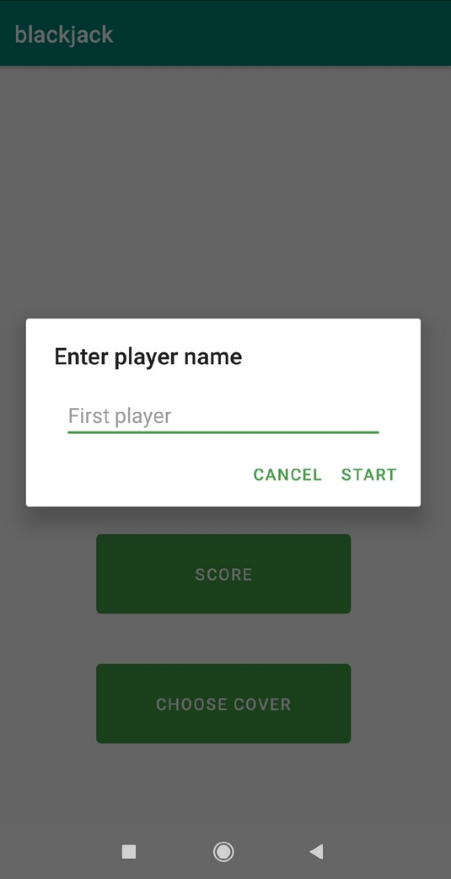


Рис.6.

* ScoreFragment

Фрагмент, отображающий игровую статистику. Используя RecyclerView, отображает содержимое базы данных, сортируя записи по количеству выигранных партий.



Рис. 7.

* GameViewModel

ViewModel для основного экрана игры. Содержит методы для связи GameFragment и классов модели. Хранит текущую игровую ситуацию в LivaData полях, на которые подписан GameFragment. Создаёт игровую сессию и управляет ходом игры.

* App

Класс, хранящий экземпляр базы данных. Необходим для того, чтобы база данных была единственной на приложение.

# **Тестирование**

Для тестирования приложения были написаны Unit и UI-тесты.

Суть unit тестов заключается в проверке различных игровых методов. В данном случае написаны тесты для проверки обработки значения карты и для проверки метода получения карты игроком.



Все unit тесты успешно пройдены.

UI-тест проверяет корректность работы при одном из базовых сценариев использования.



UI-тест был также успешной пройден.

# **Вывод**

В ходе выполнения данной курсовой работы было разработано приложение, представляющее из себя игру blackjack. Были получены навыки разработки Android приложений, а также навыки их тестирования с помощью Unit и UI тестов.

Ссылка на репозиторий, содержащий исходный код приложения: https://github.com/EndurableFawn/androidLabs/tree/master/blackjack

# **Список литературы**

1. [Электронный ресурс]: Android Developer

Режим доступа: <https://developer.android.com/>

2. [Электронный ресурс]: Android Room with a View – Kotlin

Режим доступа: <https://codelabs.developers.google.com/codelabs/android-room-with-a-view-kotlin/#0>

3. [Электронный ресурс]: Android fundamentals 10.1 Part A: Room, LiveData, and ViewModel

Режим доступа: https://codelabs.developers.google.com/codelabs/android-training-livedata-viewmodel/