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

"Приложение для ролевой игры Dungeons and Dragons на Android"

ИНДИВИДУАЛЬНЫЙ ПРОЕКТ
по дисциплине
«Программирование мобильных устройств»
ЮУрГУ – 02.03.02.2022.308-389

Руководитель проекта:
преподаватель кафедры СП
_____ М.Н. Глизница

Автор работы:
студент группы КЭ-402
_____Н.Ю.Потапов

Работа защищена
с оценкой: _____
“___”_____ 2024 г.

Челябинск 2024

Оглавление

Введение

В современном мире настольные ролевые игры, такие как *Dungeons and Dragons*, стали неотъемлемой частью досуга для многих любителей фэнтези и стратегий. С развитием технологий и увеличением числа участников в этих играх появилась необходимость в удобных цифровых инструментах, которые упростят процесс подготовки, управления персонажами и взаимодействия игроков. В ответ на эту потребность было разработано приложение для Android, специально созданное для поклонников *Dungeons and Dragons*. Оно станет незаменимым помощником для мастеров игры и игроков, упрощая организацию процесса и позволяя сосредоточиться на приключениях и совместной фантазии.

Данный проект реализован с использованием платформы Android и архитектуры MVC (Model-View-Controller), что позволяет разделить логику приложения, обработку данных и отображение информации. В частности, активность MainActivity отвечает за взаимодействие с пользовательским интерфейсом, обработку пользовательского ввода и вывод результатов. Модель данных, представленной в виде списка истории бросков, хранит информацию о результатах игры, а контроллер управляет логикой обработки событий, таких как броски кубиков и изменение значений в полях ввода.

Цель данного проекта заключается в разработке простого и удобного приложения, которое поможет пользователям легко и эффективно управлять процессом бросков кубиков в ролевой игре, предоставляя интуитивно понятный интерфейс для настройки и отображения результатов.

Описание проекта

Проект представляет собой мобильное приложение для управления процессом бросков кубиков в ролевой игре Dungeons and Dragons, предлагая пользователям удобный и интуитивно понятный интерфейс для выбора кубиков, настройки параметров броска и отображения результатов.

Основные функции приложения:

1. **Выбор типа кубика** – Пользователи могут выбирать различные виды кубиков (d4, d6, d8, d10, d12, d20) для выполнения бросков.
2. **Установка количества кубиков** – Ввод количества кубиков, которые необходимо бросить, для выполнения многократных бросков.
3. **Настройка условия успеха** – Возможность задать минимальное значение для успешного броска, чтобы определить, был ли бросок успешным.
4. **Применение усилений** – Пользователи могут ввести значение усиления, которое будет добавлено к результатам бросков, с возможностью переключать знак (положительный или отрицательный).
5. **История бросков** – Приложение сохраняет результаты последних бросков и отображает их в списке истории, чтобы пользователь мог легко отслеживать результаты.
6. **Цветовая индикация результатов** – Результаты бросков отображаются с использованием цветовой индикации (красный для единицы, оранжевый для 20, зеленый для успешных бросков), что помогает быстро воспринимать информацию.

7. **Переключение знака усиления** – Возможность переключать знак усиления между положительным и отрицательным, упрощая настройку параметров для различных типов бросков.

Приложение разработано с учетом современных подходов в разработке ПО, таких как архитектура MVVM, что обеспечивает модульность и удобство в сопровождении кода. Это позволяет легко добавлять новые функции и улучшать приложение в будущем.

2.Проектирование проекта

В ходе проектирования приложения была разработана диаграмма вариантов использования, приведенная на рисунке 1.

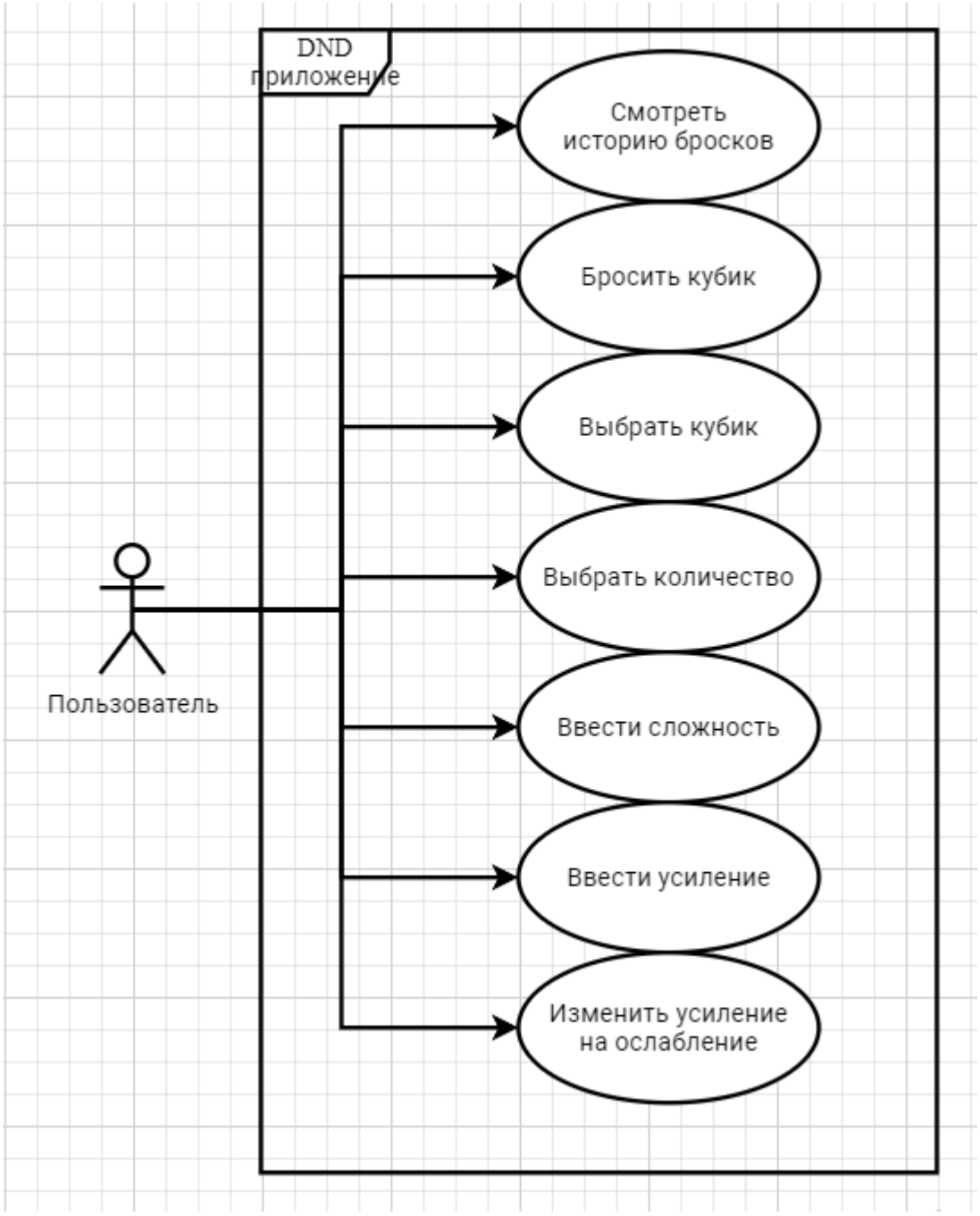

Рисунок 1 – Диаграмма вариантов использования

В системе взаимодействует один актёр – пользователь. Пользователь может выбрать тип кубика, что позволяет настроить параметры броска в соответствии с правилами игры. Эта функция предоставляет пользователю гибкость при выборе, например, d6 или d20.

Пользователь может установить количество кубиков для одновременного броска, что упрощает процесс управления несколькими бросками за один раз.

Кроме того, пользователь имеет возможность задать условие успеха, чтобы автоматически определить, является ли результат броска успешным. Это помогает анализировать результаты без необходимости дополнительного расчета.

Пользователь может применить усиления к броску, добавляя или вычитая числовое значение, что делает процесс адаптивным к различным игровым ситуациям. Также имеется функция переключения знака усиления, позволяющая быстро менять положительное значение на отрицательное и наоборот.

Основное действие – выполнение броска кубиков – позволяет пользователю сгенерировать случайные числа в зависимости от настроек. Результаты бросков сохраняются в истории, которую пользователь может просматривать для анализа прошлых действий.

Все эти функции создают интуитивно понятный и функциональный интерфейс, позволяя пользователю легко управлять параметрами бросков и адаптировать приложение к потребностям игры. На рисунке представлена диаграмма вариантов использования.

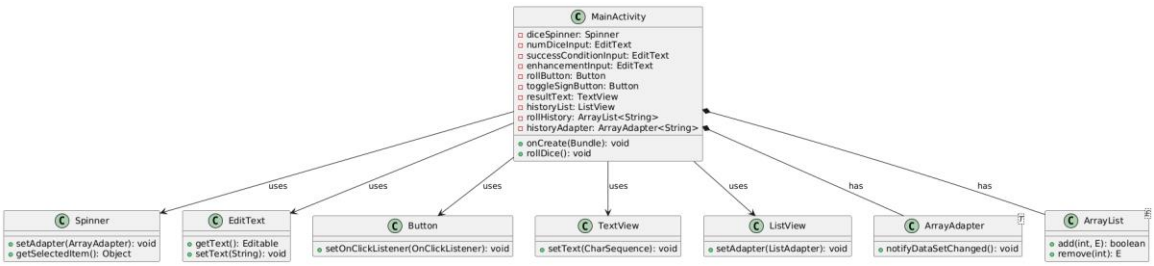

Рисунок 2 – Диаграмма классов.

Диаграмма классов представляет собой структуру, описывающую классы, их атрибуты, методы и взаимосвязи между ними в проекте. В нашей диаграмме классов можно выделить несколько ключевых элементов:

Класс MainActivity:
Это главный класс приложения, который отвечает за управление пользовательским интерфейсом и обработку событий.

- **Атрибуты:**
    - diceSpinner – выпадающий список для выбора типа кубика.
    - numDiceInput – поле ввода количества кубиков.
    - successConditionInput – поле для указания условия успеха.
    - enhancementInput – поле для ввода числового усиления.
    - rollButton – кнопка для выполнения броска.
    - toggleSignButton – кнопка для изменения знака усиления.
    - resultText – текстовое поле для отображения результата бросков.
    - historyList – список для отображения истории бросков.

- rollHistory – коллекция, хранящая данные о результатах бросков.
- historyAdapter – адаптер для управления списком истории.

Методы:

- onCreate(Bundle) – инициализация элементов интерфейса при запуске приложения.
- rollDice() – метод для выполнения броска кубиков с учетом заданных условий.

Класс Spinner:
Отвечает за выбор типа кубика.

- **Методы:**
  - setAdapter(ArrayAdapter) – связывает выпадающий список с данными.
  - getSelectedItem() – возвращает текущий выбранный элемент.

Класс EditText:
Обеспечивает ввод текстовых данных пользователем.

- **Методы:**
  - getText() – получает введённый текст.
  - setText(String) – задаёт текст в поле.

Класс Button:
Реализует элементы управления, которые реагируют на нажатия.

- **Методы:**
    - setOnClickListener(OnClickListener) – задаёт слушателя событий для кнопки.

Класс TextView:
Предназначен для отображения текстовых данных на экране.

- **Методы:**
    - setText(CharSequence) – устанавливает текст для отображения.

Класс ListView:
Обеспечивает отображение списка элементов (истории бросков).

- **Методы:**
    - setAdapter(ListAdapter) – связывает список с адаптером для отображения данных.

Класс ArrayAdapter:
Позволяет связать данные с элементами пользовательского интерфейса.

- **Методы:**
    - notifyDataSetChanged() – обновляет интерфейс после изменения данных.

Класс ArrayList:
Используется для хранения списка данных (истории бросков).

- **Методы:**
    - add(int, E) – добавляет элемент в список.

- remove(int) – удаляет элемент из списка.

Диаграмма демонстрирует взаимодействие между MainActivity и компонентами пользовательского интерфейса (Spinner, EditText, Button, TextView, ListView). Кроме того, классы ArrayAdapter и ArrayList обеспечивают хранение и обработку данных для эффективного управления историей бросков.

3. Реализация

3.1 Библиотеки

При разработке приложения для управления задачами использовались следующие библиотеки и технологии, которые обеспечили функциональность и удобство работы:

AndroidX

Набор библиотек, который заменил устаревшие библиотеки поддержки. AndroidX предоставляет современные инструменты для разработки приложений, включая компоненты для работы с пользовательским интерфейсом, обработки событий и упрощённого управления данными. Например:

- **AppCompatActivity** – для обеспечения обратной совместимости с более старыми версиями Android.
- **RecyclerView** – для работы со списками и адаптерами данных.

Material Components

Эта библиотека обеспечивает соответствие элементов интерфейса приложения рекомендациям Material Design. В проекте использовались:

- **Spinner** – для создания выпадающего списка типов кубиков.
- **Button** – для взаимодействия с пользователем (например, бросок кубиков, переключение знаков).
- **TextView** – для отображения результатов бросков в удобном виде.

Random

Класс из стандартной библиотеки Java для генерации случайных чисел.

- Применяется для имитации бросков кубиков, обеспечивая случайность результатов в диапазоне от 1 до максимального значения граней кубика.

ArrayAdapter и ArrayList

- **ArrayAdapter** используется для связывания списка истории бросков с интерфейсом ListView. Он обеспечивает упрощённое управление данными и их отображением.

- **ArrayList** применяется для хранения данных о результатах бросков, предоставляя удобные методы добавления, удаления и обновления элементов.

Listeners (OnClickListener)

Обеспечивают обработку событий, таких как нажатия кнопок. Используются для:

- Реализации логики броска кубиков при нажатии на кнопку.
- Изменения знака усиления (+ или -) в поле ввода.

3.2 Функции приложения

В приложении реализованы следующие функции:

- **Выбор типа кубика**: Пользователь может выбрать тип кубика из выпадающего списка (d4, d6, d8, d10, d12, d20). Это позволяет настроить бросок для различных типов игры, в том числе для Dungeons and Dragons.
- **Ввод количества кубиков**: Пользователь может ввести количество кубиков для выполнения броска. Это даёт возможность адаптировать количество кубиков в зависимости от потребностей в игре.
- **Настройка условия успеха:** В приложении предусмотрена возможность ввода условия успеха. Пользователь может указать минимальное значение, которое необходимо для успешного броска кубика, и это значение будет использоваться для визуального отображения успешных и неудачных результатов.
- **Ввод усиления**: Пользователь может ввести значение усиления, которое будет добавлено к результатам бросков. Это усиление может быть как положительным, так и отрицательным, и позволяет корректировать результаты в зависимости от различных игровых факторов.

- **Переключение знака усиления**: Реализована кнопка для переключения знака усиления (между положительным и отрицательным), что позволяет пользователю быстро адаптировать параметры броска в процессе игры.

- **Выполнение броска**: При нажатии кнопки "Бросить" генерируется случайное число в пределах выбранного типа кубика с учётом усиления. Результат броска отображается в виде текста,

который может быть окрашен в разные цвета в зависимости от значения (например, красный для неудачи, зелёный для успеха).

- **История бросков**: Все результаты бросков сохраняются в истории. Пользователь может просматривать последние 10 бросков, что помогает отслеживать результаты и анализировать игру.

3.3 Реализация

Приложение для управления бросками кубиков состоит из нескольких ключевых компонентов, включая элементы пользовательского интерфейса, логику обработки данных и отображения результатов. Ниже представлен листинг кода, который реализует основные функции приложения.

Листинг 1 – Основные функции

```
package com.example.dnd;

import android.os.Bundle;
import android.widget.*;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;
import java.util.Random;

public class MainActivity extends AppCompatActivity {

    private Spinner diceSpinner;  // Выпадающий список для выбора типа
кубика
    private EditText numDiceInput;  // Поле ввода для количества кубиков
    private EditText successConditionInput;  // Поле для ввода условия
успеха
    private EditText enhancementInput;  // Поле для ввода усиления
    private Button rollButton;  // Кнопка для выполнения броска
    private Button toggleSignButton;  // Кнопка для переключения знака
усиления
    private TextView resultText;  // Текстовое поле для отображения
результатов
    private ListView historyList;  // Список для отображения истории
бросков

    private ArrayList<String> rollHistory;  // Список истории бросков
    private ArrayAdapter<String> historyAdapter;  // Адаптер для
отображения истории

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Инициализация элементов интерфейса
        diceSpinner = findViewById(R.id.dice_spinner);
        numDiceInput = findViewById(R.id.num_dice_input);
```

```
        successConditionInput =
findViewById(R.id.success_condition_input);
        enhancementInput = findViewById(R.id.enhancement_input);
        rollButton = findViewById(R.id.roll_button);
        toggleSignButton = findViewById(R.id.toggle_sign_button);
        resultText = findViewById(R.id.result_text);
        historyList = findViewById(R.id.history_list);

        // Настройка выпадающего списка для выбора типа кубика
        String[] diceOptions = {"d4", "d6", "d8", "d10", "d12", "d20"};
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, diceOptions);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown
_item);
        diceSpinner.setAdapter(adapter);

        // Настройка списка истории бросков
        rollHistory = new ArrayList<>();
        historyAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, rollHistory);
        historyList.setAdapter(historyAdapter);

        // Обработка нажатия кнопки "Бросить"
        rollButton.setOnClickListener(v -> rollDice());

        // Обработка нажатия кнопки для переключения знака усиления
        toggleSignButton.setOnClickListener(v -> {
            String currentText = enhancementInput.getText().toString();
            if (currentText.isEmpty()) {
                enhancementInput.setText("+");
                toggleSignButton.setText("+");
            } else if (currentText.startsWith("+")) {
                enhancementInput.setText("-" +
currentText.substring(1));
                toggleSignButton.setText("-");
            } else if (currentText.startsWith("-")) {
                enhancementInput.setText("+" +
currentText.substring(1));
                toggleSignButton.setText("+");
            }
        });
    }

    private void rollDice() {
        String selectedDice = diceSpinner.getSelectedItem().toString();
        int maxFace = Integer.parseInt(selectedDice.substring(1)); //
Получаем количество граней кубика

        String numDiceStr = numDiceInput.getText().toString();
        if (numDiceStr.isEmpty()) {
            resultText.setText("Введите количество кубиков!");
            return;
        }

        String successConditionStr =
successConditionInput.getText().toString();
        int successCondition = 0;
        if (!successConditionStr.isEmpty()) {
            successCondition = Integer.parseInt(successConditionStr);
        }
```

```
        String enhancementStr = enhancementInput.getText().toString();
        int enhancement = 0;
        if (!enhancementStr.isEmpty()) {
            try {
                enhancement = Integer.parseInt(enhancementStr);
            } catch (NumberFormatException e) {
                enhancement = 0; // Если ввод невалидный, ставим
усиление 0
            }
        }

        int numDice = Integer.parseInt(numDiceStr);
        Random random = new Random();
        StringBuilder results = new StringBuilder("Результаты: ");
        StringBuilder colorResults = new StringBuilder();
        StringBuilder historyResults = new StringBuilder();

        for (int i = 0; i < numDice; i++) {
            int roll = random.nextInt(maxFace) + 1; // Генерация
результата броска
            roll += enhancement; // Применение усиления

            String rollText = String.valueOf(roll);

            // Определение цвета для результатов
            if (roll == 1) {
                rollText = "<font color='#FF0000'>" + rollText +
"</font>"; // Красный для 1
            } else if (roll == 20) {
                rollText = "<font color='#FF9800'>" + rollText +
"</font>"; // Оранжевый для 20
            } else if (roll > successCondition) {
                rollText = "<font color='#4CAF50'>" + rollText +
"</font>"; // Зеленый для успеха
            }

            colorResults.append(rollText).append(" ");
            historyResults.append(roll).append(" ");
        }


resultText.setText(android.text.Html.fromHtml(colorResults.toString(),
android.text.Html.FROM_HTML_MODE_LEGACY));

        // Добавление результатов в историю
        String resultString = historyResults.toString();
        String historyEntry = numDice + " x " + selectedDice + " → " +
resultString.trim();
        rollHistory.add(0, historyEntry); // Добавление в начало списка
истории
        if (rollHistory.size() > 10) {
            rollHistory.remove(rollHistory.size() - 1); // Удаление
старых записей
        }
        historyAdapter.notifyDataSetChanged();
    }
}
```

Описание листинга:

1. **Основной класс MainActivity**:
    - Это главный класс активности в приложении, который отвечает за отображение пользовательского интерфейса и обработку действий.
2. **Инициализация элементов интерфейса**:
    - В методе onCreate() инициализируются все пользовательские элементы: выпадающий список для выбора типа кубика (diceSpinner), поля ввода для количества кубиков, условий успеха и усиления (numDiceInput, successConditionInput, enhancementInput), а также кнопки для выполнения броска (rollButton) и переключения знака усиления (toggleSignButton).
3. **Обработчики нажатий**:
    - Для кнопок настроены обработчики, которые выполняют соответствующие действия:
        - Кнопка "Бросить" генерирует случайные результаты бросков с учётом выбранного типа кубика, количества кубиков, усиления и условия успеха.
        - Кнопка "Переключить знак" изменяет знак усиления между положительным и отрицательным.
4. **Метод rollDice()**:
    - Основная логика приложения: генерирует случайные результаты бросков, применяет усиление и окрашивает результаты в зависимости от их значения (например, красный для критического провала, зелёный для успешных бросков).
5. **История бросков**:

- Все результаты сохраняются в список rollHistory, и отображаются в ListView. История ограничена 10 последними бросками.

6. **Используемые библиотеки**:
   - Стандартные компоненты Android, такие как Spinner, EditText, Button, и TextView, используются для создания пользовательского интерфейса.
   - Класс Random используется для генерации случайных чисел для бросков кубиков.
   - Html.fromHtml() позволяет отображать текст с HTML-разметкой для цветового оформления результатов.

3.4 Внешний вид приложения

Внешний вид приложения приведен на рисунках ниже. Рисунки отражают главное рабочее пространство приложения.

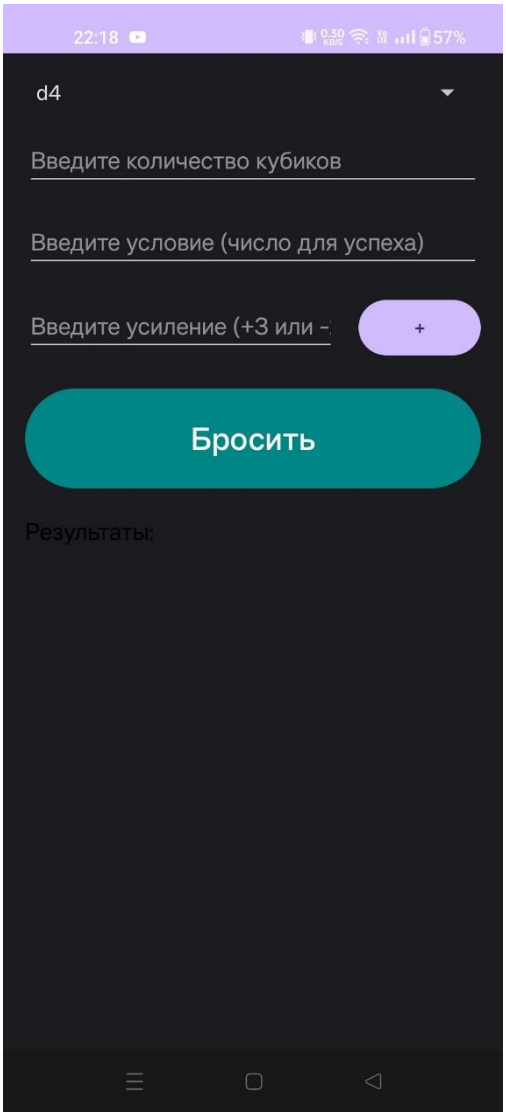

Рисунок 3. – Основной интерфейс приложения

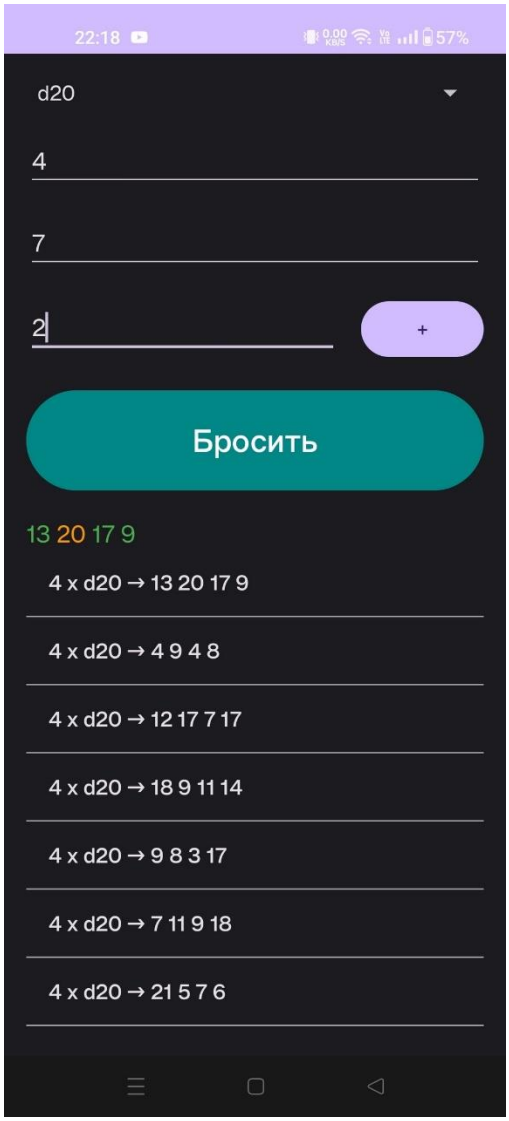

Рисунок 4. – Интерфейс приложения в работе

3. Тестирование

Тестирование приложения было проведено с целью проверки функциональности и производительности. Основные аспекты, которые были протестированы, включают выбор кубика, бросок, настройки параметров, а также пользовательский интерфейс и обработку ошибок.

Таблица тестов

ID теста	**Название теста**	**Описание**	**Ожидаемый результат**	**Результат**
1	Тест выбора кубиков	Проверка возможности выбора любого кубика	Кубик меняется	Пройден
2	Тест на количество кубиков	Проверка возможности броска сразу нескольких кубиков	Вывод нескольких результатов	Пройден
3	Тест использования усиления	Проверка корректного переключения знака – и +	+ и – переключаются между собой и влияют на общее число	Пройден
4	Тест условия	Проверка переключения цвета при выполненном условии	Число становится зеленым при выполненном условии	Пройден

5	Тест обработки пустых полей	Проверка поведения приложения при создании задачи с пустыми полями	Выдан корректный результат	Пройден
6	Тест взаимодействия с интерфейсом	Проверка интерактивности кнопок и полей ввода	Все кнопки и поля ввода работают корректно	Пройден

Заключение по тестированию

Все проведенные тесты показали, что приложение работает корректно и стабильно. В результате тестирования были выявлены и устранены некоторые незначительные ошибки, что сделало приложение более надежным для конечных пользователей.

Заключение

В рамках данного проекта было разработано мобильное приложение для управления бросками кубиков в ролевых играх, таких как Dungeons & Dragons, на платформе Android. Приложение предоставляет пользователю интуитивно понятный интерфейс для выбора типа кубика, ввода количества кубиков, условия успеха и усилений, а также для отображения результатов бросков с возможностью их истории.

Реализация приложения использует несколько ключевых технологий и библиотек, таких как Android SDK, для обеспечения стабильной работы и хорошего пользовательского опыта. В частности, были применены компоненты интерфейса для ввода данных, отображения результатов и управления списком истории, что позволяет пользователю легко отслеживать свои действия и результативность в процессе игры.

Главной целью проекта было создание удобного инструмента, который упрощает процесс бросков кубиков, делает его более наглядным и улучшает взаимодействие пользователей с приложением. Приложение легко адаптируется под различные устройства и обладает всеми необходимыми функциями для полноценного использования в контексте ролевых игр. В дальнейшем приложение может быть расширено дополнительными функциями, такими как поддержка нескольких игроков, более сложные вычисления или интеграция с другими игровыми системами.

Таким образом, разработанное приложение является удобным и эффективным инструментом для игроков в ролевые игры, который упрощает процесс бросков кубиков и позволяет сосредоточиться на самих играх.