УДК 004

Разработка приложения Anagram Finder для Android

Семченко Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
студент

Аннотация
В данной статье описан процесс создания самой простой маски для Facebook и Instagram. Работа происходит в среде Spark AR. Конечным результатом является рабочая маска, которую можно использовать с мобильного устройства в одном из приложений.
Ключевые слова: Instagram, Facebook, mask, маска, Spark AR, augmented reality

Create app Anagram Finder for Android

Semchenko Regina Viktorovna
Sholom-Aleichem Priamursky State University
student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
student

Abstract
This article describes the process of creating the simplest mask for Facebook and Instagram. Work takes place in the Spark AR environment. The end result is a work mask that can be used from a mobile device in one of the applications.
Keywords: Instagram, Facebook, mask, Spark AR, augmented reality

Анаграмма - это слово или фраза, сформированная путем перестановки букв другого слова или фразы, обычно используя все исходные буквы ровно один раз.

Цель данной статьи разработать собственное приложение анаграмм для Android устройств.

Исследованиями в области разработки мобильных приложений занимались многие российские и зарубежные исследователи. А.С. Винокуров, Р.И. Баженов [1] рассмотрели разработку приложений для мобильных устройств. С.К. Заманова, Г.Е. Сейдахметова, Г.Г. Масимова,

А.Е. Манатова [2]. Они изучили современные подходы к разработке мобильных приложений. Также они рассмотрели разработку приложения в среде Rad Studio XE7. Е.Н.Амиргалиев и др. разработал свою собственную модель отправки информационных сообщений для мобильных операционных систем.[3]

Так как анаграмма это слова сформированные из букв другого слова, то приложение Anagram Finder попросит пользователя ввести символы. После того, как эти символы введены, пользователь нажимает кнопку «Подтвердить», которая запускает поиск анаграммы по этим буквам.

Приложение Android будет опираться на словарь слов, встроенный внутрь при поиске анаграмм. После завершения поиска приложение отобразит найденные анаграммы или сообщение пользователю о том, что не может найти анаграммы для введенных символов.

Первым шагом является создание проекта приложения Android в Android Studio. Как только этот проект будет создан, будет возможность настроить его зависимости в файле «build.gradle» . Для облегчения поиска в приложении активности представлений, определенных в XML-файле пользовательского интерфейса, будем полагаться на библиотеку кода «Butter Knife». Для этого необходимо добавить следующие строки в файл «build.gradle» (рис.1).

```
android {

    compileOptions {
        sourceCompatibility = 1.8
        targetCompatibility = 1.8
    }
}
```

Рисунок 1 – настройка зависимости

Чтобы «Butter Knife» работал правильно, а также чтобы мы могли использовать Java 8 Lambdas, необходимо также добавить следующие строки конфигурации в файл «build.gradle» (рис.2).

```
android {

    compileOptions {
        sourceCompatibility = 1.8
        targetCompatibility = 1.8
    }
}
```

Рисунок 2 – добавление зависимости

Ядро приложения будет находиться в классе Anagram . Этот класс будет иметь следующие обязанности. Загрузка слов в память из словаря слов,

расположенного в каталоге «/ src / main / assets» . Ищет анаграммы в словаре этого слова, используя входную строку. В классе Anagram определяется статический объект «List» с именем «WORDS», который будет содержать слова, загруженные из словаря слов. Также важно, чтобы клиенты этого класса могли знать, были ли слова загружены в память.

Чтобы узнать, были ли слова загружены в память, определяем статическое свойство «LOADED» логического типа, инициализированное как «false» . Загрузка слов из словаря слов выполняется в методе «loadWords» . В рамках этого метода функция будет построчно читать содержимое файла «dict», содержащегося в каталоге «/ src / main / assets» .

Каждое слово чтения добавляется в «WORDS» список и преобразуется в верхний регистр, вызвав «toUpperCase» метод. Цель состоит в том, чтобы упростить сравнения впоследствии.

После того, как все слова были успешно загружены, установливаем для логического параметра «LOADED» значение «true», указывающее, что словарь слов загружен (рис.3).

```
public static boolean sameLetters(String a, String b) {
        if (a == null)
        return b == null;
        if (b == null)
        return false;
        char[] left = a.toCharArray();
        char[] right = b.toCharArray();
        Arrays.sort(left);
        Arrays.sort(right);
        return Arrays.equals(left, right);
        }
```

Рисунок 3 – Загрузка слов

Используя эти два метода, поиск анаграмм слова, введенного пользователем, будет легким. Для этого создадим метод «listWords» в классе Anagram . Метод «listWords» принимает в качестве входных данных строку, содержащую слова, анаграммы которые вел пользователь. Переберем список слов, загруженных в память, и для каждого слова вызываем метод «sameLetters» . Если слово, введенное в метод, имеет те же буквы, что и текущее слово в итерации, добавляем его в список анаграмм, которые возвращаются при выводе метода (рис.4).

```
        import java.io.InputStreamReader;
        import java.util.ArrayList;
        import java.util.Arrays;
        import java.util.List;


public class Anagram {
    private static final String DICT = "dict";
    private static final List < String > WORDS = new ArrayList < > ();
    private static boolean LOADED = false;
    public static boolean isLoaded() {
        return LOADED;
    }

    public static void loadWords(Context context) {
        BufferedReader buf = null;
        try {
            buf = new BufferedReader(new InputStreamReader(context.getAssets().open(DICT)));
            String line = null;
            while ((line = buf.readLine()) != null) {
                WORDS.add(line.toUpperCase());
            }

            LOADED = true;

        } catch (IOException ioe) {
            // ...
        } finally {
            if (buf != null) {
                try {
                    buf.close();
                } catch (IOException e) {
                    // ...
                }
            }
        }
}
```

Рисунок 4 – поиск слов

Следующим шагом будет создание графического интерфейса пользователя для приложения Anagram Finder. Этот пользовательский интерфейс будет состоять из следующих элементов. «EditText» , чтобы позволить пользователю вводить буквы , чьи анаграммы он хочет найти. «Button» , чтобы позволить пользователю запустить поиск анаграммы. «TextView» помещается внутри «ScrollView» , чтобы отобразить список найденных анаграмм. Для приложения, целью которого будет публикация в Google Play Store, лучше всего использовать компонент «RecyclerView», более адаптированный к этому типу отображения и позволяющий пользователю взаимодействовать с отображаемыми анаграммами (рис.5)

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <LinearLayout
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:layout_width="0px"
        android:layout_height="0px"
    />

    <EditText
        android:id="@+id/input"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/validate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Validate"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="16dp" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp">
        <TextView
            android:id="@+id/result"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </ScrollView>
</LinearLayout>
```

Рисунок 5 – Создание интерфейса

Как только слова загружены, кнопка « Проверить » активируется путем вызова ее метода «setEnabled» в потоке пользовательского интерфейса. Затем связываем метод «validate» с нажатием на кнопку «Validate», пометив метод аннотацией «OnClick Butter Knife». В методе «validate» сначала извлекаем слово, введенное пользователем. Затем выводим диалог ожидания и начинаем поиск анаграмм в рамках отдельной нити путем вызова метода статического «listWords» на Anagram класса. Как только список анаграмм возвращается, закрываем диалоговое окно ожидания и отображаем результаты на экране, вызывая метод «displayResult» . Метод «displayResult»

просто перебирает список анаграмм и отображает их с разрывом строки между каждым в элементе TextView . Если приложение Anagram Finder не находит анаграмм, пользователю отображается сообщение «НЕТ РЕЗУЛЬТАТОВ». Все это дает следующий код через класс MainActivity (рис.6).

```
import androidx.appcompat.app.AppCompatActivity;
import java.util.List;
import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
public class MainActivity extends AppCompatActivity {
    @BindView(R.id.input)
    EditText input;
    @BindView(R.id.validate)
    Button validate;
    @BindView(R.id.result)
    TextView result;
    private Dialog waitDialog;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);
        validate.setEnabled(Anagram.isLoaded());
        loadWords();
    }

    private void loadWords() {
        new Thread(() -> {
            Anagram.loadWords(MainActivity.this);

            runOnUiThread(() -> {
                validate.setEnabled(Anagram.isLoaded());
            });
        }).start();
    }

    @OnClick(R.id.validate)
    public void validate(View v) {
        String text = input.getText().toString();
        if (text != null) {
            final String text2 = text.trim();
            if (!"".equals(text2)) {
                showWaitDialog();
                new Thread(() -> {
                    final List < String > words = Anagram.listWords(text);
                    runOnUiThread(() -> {
                        hideWaitDialog();
                        displayResult(words);
                    });
                }).start();

            }
        }
    }
```

Рисунок 6 – основной код

На этом написание кода закончено, для проверки необходимо запустить эмулятор android и проверить работоспособность.

Практическим результатом является созданное приложение анаграмм на языке программирования JavaScriptв среде разработки AndroidStudio.

Библиографический список

1. Винокуров А.С., Баженов Р.И. Разработка мобильного приложения информационного сайта для абитуриентов и первокурсников университета // Современные научные исследования и инновации. 2015. № 7-2 (51). С. 54-62.
2. Заманова С.К., Сейдахметова Г.Е., Масимова Г.Г., Манатова А.Е. Разработка мобильного приложения в среде Rad Studio XE7 // Труды Международного симпозиума «Надежность и качество». 2015. №1. С. 237-240.
3. Amirgaliyev E.N., Kalizhanova A.U., Kozbakova A.KH. Development of applications to mobile devices in Android platform // Труды Международного симпозиума «Надежность и качество». 2015. №1. С. 240-242.