УДК 004

Разработка игры Catch Star в Android Studio

Звайгзне Алексей Юрьевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается процесс разработки Android игры – Catch Star, сутью которой является в угадывании актера по фото. имена актеров и их фотографии берутся с популярного сайта Internet Movie Database, со страницы топа 100 актеров на 2014 год.
Ключевые слова: Android, Android Studio, URL, Download Content Task, BitMap

Catch Star game development in Android Studio

Zvaigzne Alexey Yurievich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the process of developing an Android game – Catch Star, the essence of which is to guess an actor from a photo. The names of the actors and their photos will be taken from the popular Internet Movie Database site, from the page of the top 100 actors for 2014.
Keywords: Android, Android Studio, URL, Download Content Task, BitMap

1 Введение

1.1 Актуальность

На данный момент все чаще разработчики сталкиваются с проблемой регулярного обновления контента в своих приложениях. Большинство проектов строится на том, что все необходимые файлы для исправной работы уже находятся на локальном носителе пользователя без необходимости регулярно докачивать файлы, но, если файлов много и не требуется их постоянное наличие на устройстве в момент инициализации приложения, достаточно на сервере хранить часть требуемых ресурсов.

Есть множество проектов, пользующихся данной механикой на постоянной основе экономя ресурсы обычного пользователя.

1.2 Обзор исследований

А.Ю. Звайгзне в своей статье рассматривал процесс разработки Android приложения Будильник [1]. На официально сайте для разработчиков Android приложений есть инструкции по использованию различных библиотек и

фреймворков, которые используются в данной статье [2]. А. Бхаги в своей научной диссертации описывал процесс разработки игры на платформе Android [3]. А.Карниати разрабатывал игру Сказание Мамочи с элементами ролевой игры [4]. В. Джексон в своей статье описывал процесс адаптации приложений под платформу Android седьмой версии (Nougat) [5].

1.3 Цель исследования

Цель исследования является создание игры, собирающей данные для использования из сети интернет с конкретного сайта в Android studio.

2 Материалы и методы

Игра разрабатывается на последней версии Android Studio, с использование библиотек с возможностью открытия интернет-соединения на устройстве для получения необходимы данных напрямую с сайта, обрабатывая код страницы, выбирается код, отвечающий за имена актеров и их фотографии на сайте.

3 Результаты и обсуждения

Разрабатываемая игра Catch Star – это квест, пользователю выводится фотография актера и четыре варианта ответа, один из вариантов ответа содержит правильное имя актера – верный вариант ответа, а в остальных генерируются случайные имена актеров с того же сайта – неверный вариант ответа, ограничений по времени отсутствуют, как и какое-либо подведение итогов игры. Игра будет проверять какое имя актера выбрано, даже если произойдет дублирование правильного варианта ответа, любой из них окажется верным.

Исходя из описанных правил, на визуальном слое активности должны быть поле для вывода изображения с сайта, четыре кнопки для варианта ответа или текстовых поля, заполнить форму можно через код или вручную через визуальный редактор (рис. 1-2).

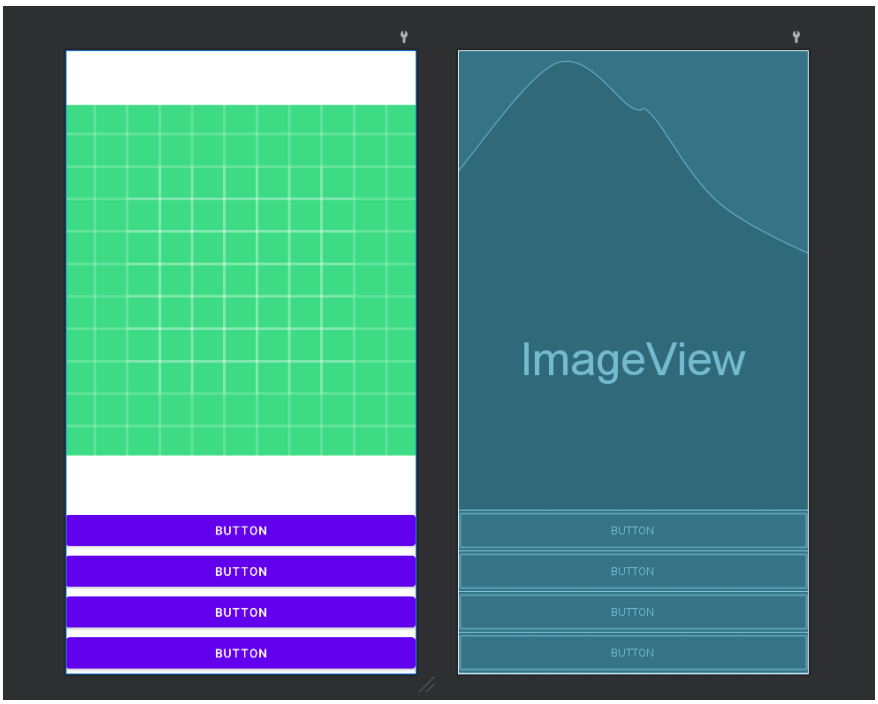

Рисунок 1. Визуальное отображение слоя активности

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android=
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button0"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:tag="0"
        android:text="Button"
        android:onClick="onClickAnswer"
        app:layout_constraintBottom_toTopOf="@+id/button1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/button1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:tag="1"
        android:text="Button"
        android:onClick="onClickAnswer"
        app:layout_constraintBottom_toTopOf="@+id/button2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/button2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:tag="2"
        android:text="Button"
        android:onClick="onClickAnswer"
        app:layout_constraintBottom_toTopOf="@+id/button3"
        app:layout_constraintEnd_toEndOf="parent"
```

Рисунок 2. Вид кода слоя

Кнопки необходимо закрепить под изображением сверху внизу закрепляя за предыдущим эллементом, вместо изображения можно подставить любое по умолчанию из библиотеки проекта, текст кнопок можно

так же оставить по умолчанию, так как эти элементы будут обновлять данными из интернета.

Для начала работы добавляется список используемых библиотек, отличными из них являются (рис. 3):

AsyncTask – в данной игре отвечает за получение данных из потока собираемых данных с сайта.

BufferedReader – отвечает за получения массива строк из потока для дальнейшей обработки.

InputStream – обрабатывает входной поток для получения массива байтов, изначального вида данных.

InputStreamReader – декодирует полученный массив байтов в строки.

HttpURLConnection – отвечает за открытие и закрытие интернет-соединения и оформления запроса на получения кода с сайта.

Matcher и Pattern – обрабатывает строки, ищет по ключевым фразам, отвечает за сбор массива строк.

```
import androidx.appcompat.app.AppCompatActivity;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.concurrent.ExecutionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

Рисунок 3. Импорт библиотек

После получения библиотек в активность добавляются все элементы из визуального слоя, для дальнейшей работы с ними (рис. 4-5)

```
public class MainActivity extends AppCompatActivity

    private Button button0;
    private Button button1;
    private Button button2;
    private Button button3;
    private ImageView imageViewStar;
```

Рисунок 4. Добавление переменных в активность

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    button0 = findViewById(R.id.button0);
    button1 = findViewById(R.id.button1);
    button2 = findViewById(R.id.button2);
    button3 = findViewById(R.id.button3);
    buttons = new ArrayList<>();
    buttons.add(button0);
    buttons.add(button1);
    buttons.add(button2);
    buttons.add(button3);
    imageViewStar = findViewById(R.id.imageViewStar);
```

Рисунок 5. Назначение переменным элементов в активности по ID

Так как URL-адрес сайта – это строка, задается новая переменная отвечающая адрес. Необходимые элементы страницы это изображения, которые хранятся в виде ссылок на элементы сайта и имена актеров, которые также описаны в коде страницы. Следующим шагом является создание массива строк ссылок на изображения и имена актеров, urls – хранит адреса изображений, names – хранит имена актеров, а buttons – отвечает за массив кнопок для более простой работы с ними (рис. 6-7).

```
private String url = "https://www.imdb.com/list/ls059609900/";

private ArrayList<String> urls;
private ArrayList<String> names;
private ArrayList<Button> buttons;
```

Рисунок 6. Добавление строки подключения к сайту и формирования массива значений ссылок, имен и кнопок

```
buttons = new ArrayList<>();
buttons.add(button0);
buttons.add(button1);
buttons.add(button2);
buttons.add(button3);
imageViewStar = findViewById(R.id.imageViewStar);
urls = new ArrayList<>();
names = new ArrayList<>();
```

Рисунок 7. Назначение переменным значений массивов

Все вводимые переменые преопределяются в активности при создании Oncreate.

Следующая часть активности открывает интернет соединение, получает код с сайта в отдельном потоке и формирует из полученных значений строку, а после выполнения кода закрывает интернет соединение (рис. 8-9).

```
private static class DownloadContentTask extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... strings) {
        URL url = null;
        HttpURLConnection urlConnection = null;
        StringBuilder result = new StringBuilder();
        try {
            url = new URL(strings[0]);
            urlConnection = (HttpURLConnection) url.openConnection();
            InputStream inputStream = urlConnection.getInputStream();
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
            BufferedReader reader = new BufferedReader(inputStreamReader);
            String line = reader.readLine();
            while (line != null) {
                result.append(line);
                line = reader.readLine();
            }
            return result.toString();
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
        }
        return null;
    }
}
```

Рисунок 8. Генерация строкого потока данных

```
private static class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {

    @Override
    protected Bitmap doInBackground(String... strings) {
        URL url = null;
        HttpURLConnection urlConnection = null;
        StringBuilder result = new StringBuilder();
        try {
            url = new URL(strings[0]);
            urlConnection = (HttpURLConnection) url.openConnection();
            InputStream inputStream = urlConnection.getInputStream();
            Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
            return bitmap;
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
        }
        return null;
    }
}
```

Рисунок 9. Генерация потока для получения изображений

При сооздании потока получения изображений необходимо преобразовать конечный код из строки в растровое изображение BitMap.

После открытия соединения и масивва данных в виде кода страницы, создается парсер собирающий нужную часть кода отвечающего за отображения контента на странице patternImg отвечает за поиск источников фотографий, а patternName ищет имена актеров и формируют массив строк в группы urls и names (рис. 10)

```
private void getContent() {
    DownloadContentTask task = new DownloadContentTask();
    try {
        String content = task.execute(url).get();
        String start = "<div class=\"lister-list\">";
        String finish = "<div class=\"footer filmosearch\">";
        Pattern pattern = Pattern.compile(start + "(.*?)" + finish);
        Matcher matcher = pattern.matcher(content);
        String splitContent = "";
        while (matcher.find()) {
            splitContent = matcher.group(1);
        }
        Pattern patternImg = Pattern.compile("src=\"(.*?)\"");
        Pattern patternName = Pattern.compile("alt=\"(.*?)\"");
        Matcher matcherImg = patternImg.matcher(splitContent);
        Matcher matcherName = patternName.matcher(splitContent);
        while (matcherImg.find()) {
            urls.add(matcherImg.group(1));
        }
        while (matcherName.find()) {
            names.add(matcherName.group(1));
        }
    } catch (ExecutionException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

Рисунок 10. Парсинг фотографий и имен

После получения всех необходимых данных выставляются правила игры, задаются правила для правильного и не правильного ответа пользователя, для этого создается две новых переменных отвечающих за номер вопроса и номер правильного варианта ответа (рис. 11).

```
private int numberOfQuestion;
private int numberOfRightAnswer;
```

Рисунок 11. Создание переменных

Генерация правильного и неправильного варианта ответа происходит по следующему алгоритму. Из массива имен выбирается случайная строка и присваевается её порядковый номер переменной numberOfQuestion, а переменная numberOfRightAnswer выбирает случайным образом кнопку хранящую правильный вариант ответа в виде числа (рис. 12-13).

```
private void generateQuestion() {
    numberOfQuestion = (int) (Math.random() * names.size());
    numberOfRightAnswer = (int) (Math.random() * buttons.size());
}
```

Рисунок 12. Создание метода генерации значения правильного варианта ответа

```
private int generateWrongAnswer() { return (int) (Math.random() * names.size()); }
```

Рисунок 13. Создание метода генерации значения неправильного варианта ответа

Построение метода генерации неправильных вариантов ответа происходит аналогичным образом.

Создание алгоритма заполнения содержимого активности из ранее созданных методов (рис. 14).

```
private void playGame() {
    generateQuestion();
    DownloadImageTask task =new DownloadImageTask();
    try {
        Bitmap bitmap = task.execute(urls.get(numberOfQuestion)).get();
        if (bitmap != null) {
            imageViewStar.setImageBitmap(bitmap);
            for (int i = 0; i < buttons.size(); i++) {
                if (i == numberOfRightAnswer) {
                    buttons.get(i).setText(names.get(numberOfQuestion));
                } else {
                    int wrongAnswer = generateWrongAnswer();
                    buttons.get(i).setText(names.get(wrongAnswer));
                }
            }
        }
    } catch (ExecutionException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

Рисунок 14. Создание метода игры

Сначала генерируется значение правильной строки из массива, скачивается изображение по номеру строки из массива с сайта, присваивается элементу просмотра изображений изображение из интернета переведенное в растровое, назначается правильный вариант ответа и неправильные варианты ответа кнопкам на визуальной части класса активности.

Осталось добавить действие по нажатию кнопки и зациклить игру, значение кнопки преобразовывается в тег, который определяет с каким вариантом ответа была выбрана кнопка, если с правильным, то выводится сообщение «Правильно:3», если же была выбрана кнопка с неправильным вариантом, то выводится текст сообщения, содержащий правильный вариант ответа, например, «Попробуй снова (^_^), правильный ответ: Dominic Percel» (рис. 15).

```
public void onClickAnswer(View view) {
        Button button = (Button) view;
    String tag = button.getTag().toString();
    if (Integer.parseInt(tag) == numberOfRightAnswer) {
        Toast.makeText( context: this,  text: "Правильно:3", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText( context: this,  text: "Попробуй снова (^_^), правильный ответ: " +names.get(numberOfQuestion), Toast.LENGTH_SHORT).show();
    }
    playGame();
}
```

Рисунок 15. Добавления действия по нажатию кнопки

Чтобы игра началась при запуске приложения необходимо добавить методы getContent() и playGame() в метод Oncreate.

Теперь можно собрать и проверить работоспособность игры (рис. 16).

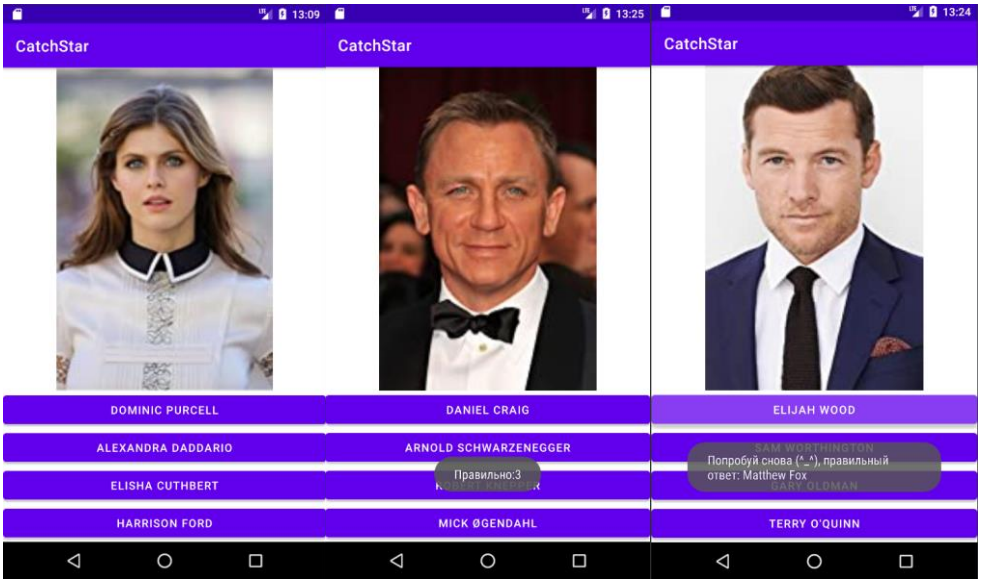

Рисунок 16. Результаты работы приложения

Выводы

Созданная игра может помочь запомнить имена разных актеров, проверить правильность работы программы можно, открыв страницу по ссылке, которая была указана в коде. Игра исправно работает при постоянном интернет-соединении.

Качество полученных изображений зависит от конечного источника, исходя из того, что изображения растровые, итоговый вид изображения зависит от размера экрана устройства, чем больше экран и меньше плотность пикселей на экране, тем хуже качество отображаемых изображений.

Библиографический список

1. Звайгзне А.Ю. Особенности создания приложения будильник для последних версий Android // Постулат. 2023. №1. С. 32.
2. Раздел официальный сайт с описание работы библиотек и фреймворков Android URL: https://developer.android.com/ (дата обращения 25.01.2023).
3. Bhagi A. Android game development with AppInventor : дис. Massachusetts Institute of Technology, 2012.
4. Kurniati A. et al. Game development “Tales of Mamochi” with role playing game concept based on android // Procedia Computer Science. 2015. Т. 59. С. 392-399.
5. Jackson W., Jackson W. An Introduction to Android 7.0 Nougat //Android Apps for Absolute Beginners: Covering Android 7. 2017. С. 1-15.