Proiect android: Aplicatie pentru ghicitori

Costin Radu Ionut

Contents

1 Introducere **2**
- 1.1 Descrierea temei . . . 2
- 1.2 Scop . . . 2
- 1.3 Obiective . . . 2
- 1.4 Rezultate . . . 2
- 1.5 Design . . . 3

2 Detalii de implementare **4**

3 Testare **9**

4 Concluzii **9**

1 Introducere

LINK GIT: `https://github.com/CostinRaduIonut/ProiectAppMobile.git`

1.1 Descrierea temei

Proiectul are ca scop realizarea unei aplicatii in limba engleza, ce ofera o gama de 3 ghicitori din fiecare categorie aleasa si cronometreaza timpul petrecut in fiecare activitate. De asemenea, aplicatia poate retine ultimul raspuns introdus in eventualitatea in care utilizatorul doreste sa caute raspunsurile pe internet sau iese din greseala si va retine si numarul de raspunsuri corecte din fiecare activitate. Aplicatia foloseste umorul si ghicitorile pentru a invata cateva cuvinte noi in limba engleza, in timp de utilizatorul se amuza.

Rezultatul final vine sub forma unei aplicatii mobile realizata in Java, in mediul de dezvoltare Android Studio.

1.2 Scop

Proiectul are ca scop final livrarea unei aplicatii, care ofera ghicitori si contorizeaza raspunsurile corecte din fiecare sectiune

1.3 Obiective

- 1 interfata colorata si atragatoare;
- 4 sectiuni prezentate intr-un ListView atragator populat cu imagini si text;
- 3 ghicitori in fiecare activitate selectata;
- 1 indiciu pentru fiecare ghicitoare;
- 1 metoda de verificare a raspunsului dat;
- 1 metoda de a contoriza timpul petrecut in fiecare activitate;
- 1 sectiune cu numarul de raspunsuri corecte date in fiecare activitate.

1.4 Rezultate

- 1 animatie de tip shake pentru Activitatea principala si 1 imagine de fundal si multe culori pentru fiecare activitate secundara;
- 4 butoane animate sub forma unor imagini si a unor texte sugestive cu tema activitatii la care trimit;
- 3 ghicitori care fac referinta la titlul fiecarei activitati;
- 1 buton care dezvaluie un indiciu pentru ghicitoarea afisata;

- 1 camp de tipul text in care se introduce raspunsul si un buton care afiseaza un toast daca raspunsul este corect sau nu
- 1 contor care monitorizeaza numarul de rasunsuri corecte;
- 1 serviciu care contorizeaza minutele petrecute in cadrul activitatii;
- 1 camp ce va tine un scor actualizabil, de fiecare data cand activitatea principala activeaza un nou intent.

1.5 Design

Designe-ul pentru activitatea principala si pentru celelalte 4 pot fi observate in Figurile 1 - 2. Designe-ul pentru activitatea principala si pentru celelalte 4 pot fi observate in Figurile 1 – 2. Fiecare layout pentru fiecare activitate este de tip portret, cat si vedere, iar textele au fost introduse folosind editorul din „Strings.xml".

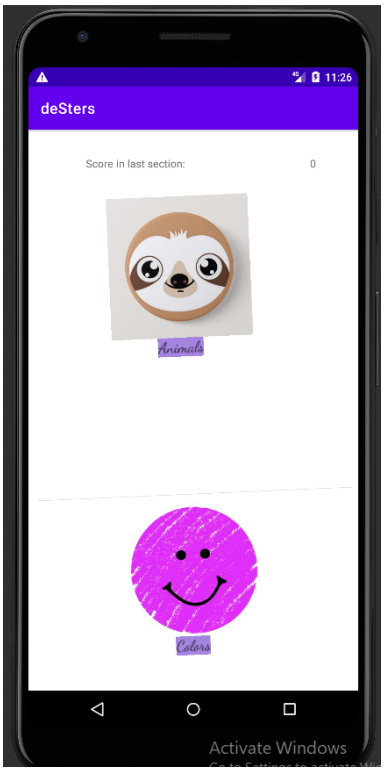

Figure 1: Design activitate principala

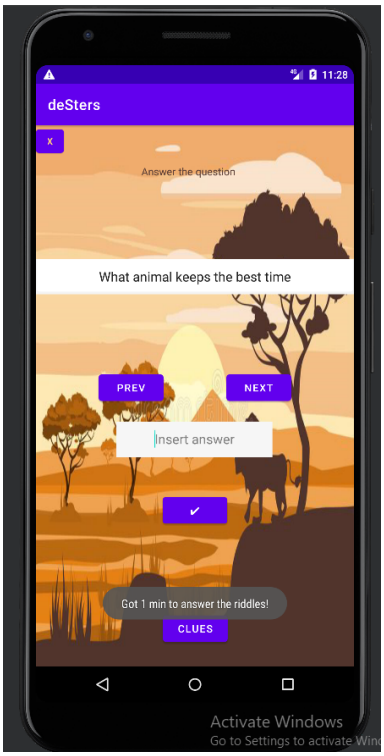

Figure 2: Design activitate secundara

2 Detalii de implementare

Aplicatia a fost implementata in Android Studio, folosind limbajul java.

Aplicatia a fost implementata in Android Studio folosind limbajul java. Intregul proiect contine 8 clase Java, dintre care 1 reprezinta un adaptor pentru Lista ce reprezinta meniul, din care se aleg activitatile cu ghicitori, 1 reprezinta un fragment ce va fi populat cu indiciile ghicitorilor si 1 reprezinta un serviciu cu rol de cronometrare a timpului petrecut in fiecare activitate (Figure 3).

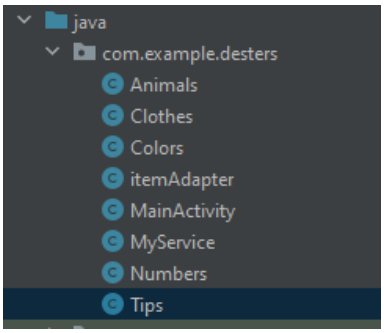

Figure 3: Clasele proiectului

Pentru inceput, in deschiderea aplicatiei porneste un media player, cu scopul de a ajuta utilizatorul sa se relaxeze in lumea in care il introduce aplicatia. Fisierul audio (Figure 4) folosit este regasit intr-un "Resource Directory" creat manual, cu numele raw (Figure 5). ulterior, media playerul a fost arhivat din motive de spatiu.

```
MediaPlayer mediaPlayer = MediaPlayer.create(getApplicationContext(), R.raw.main);
mediaPlayer.start();
mediaPlayer.release();
```

Figure 4: Activare MediaPlayer

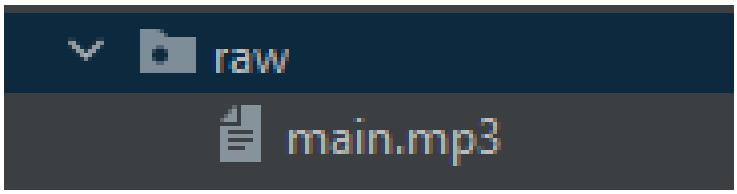

Figure 5: Fisier raw

Cat despre interfata din MainActivity, va fi afisata o lista populata cu imagini si text dupa un model definit de o clasa itemAdapter (Figure 6) (care extinde clasa abstracta BaseAdapter) si un fisier xml ce decide designe-ul, cu numele de "my main list.xml".

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    View currentView = inflater.inflate(R.layout.my_main_list,  root: null);
    TextView tvText = currentView.findViewById(R.id.list_text);
    ImageView tvTmage = currentView.findViewById(R.id.list_img);
    tvText.setText(initialList[position]);
    tvTmage.setImageResource(initialImages[position]);

    return currentView;
}
```

Figure 6: Lista

Odata populata cu date, din listView se extrage pozitia fiecarei imagini reprezentative, pentru sectiunea pe care o va deschide, si in functie de pozitie, prin click se apeleaza cate o functie ce trimite scorul actual, folosind „startActivityForResult”, adica cifra 0 sub forma de string si totodata deschide o activitate noua. In plus, un toast anunta pornirea serviciului si spune cat timp va fi pus la dispozitie . De asemenea, se asteapta primirea scorului prin „onActivityResults” in momentul parasirii aplicatiei (scorul va fi reinitializat cu 0 de fiecare data cand se paraseste MainActivity).

In fiecare activitate noua pe care ajungem prin apasarea elementului din lista, porneste un serviciu si se extrag date dintr-un json extern, ce contine urmatoarele campuri: ghicitoare, raspuns corect si un hint, care vor fi adaugate elementelor corespunzatoare.

Fiecare din activitatile secundare contin 4 elemente principale de programare, Fragmente, Servicii, metode okHTTP3 si memoria dispozitivului mobil, pe langa clasicele elemente de baza: TextView, butoane, variabile etc.

Serviciu: Serviciul porneste un fir de executie in metoda onStartCommand si cronometreaza minutele petrecute in fiecare activitate, dupa care metoda onDestroy se apeleaza la parasirea activitatii curente si firul de executie se opreste (Figure 7).

Fragment: Fragmentul are rolul de a afisa indiciile la ghicitoare prin apasarea unui buton. Fragmentul primeste si afiseaza datele intr-un textView, cu ajutorul claselor FragmentManager si Bundle (Figure 8).

Okhttp3: M-am folosit de OkHttpClient si de interfata Call,pentru a face un request, pe care am folosit metodele pe care le-am suprascris: onFailure si onResponse, cu scopul de a prelua datele din JSON. Dupa ce datele au fost preluate de un string si inserate intr-un obiect de tipul JSONObject, s-au preluat datele necesare: ghicitoarea, indiciul si raspunsul corect, apoi fiecare a fost fost plasat in variabile pentru folosinta ulterioara (Figure 9).

Memoria telefonului: Folosind SharedPreferences, campurile care au fost completate, au fost pastrate cu datele introduse, pentru a ajuta utilizatorul in cazul in care doreste sa paraseasca aplicatia, pentru a cauta raspunsurile pe internet (Figure 10).

```java
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    stopThread = false;
    Runnable runnable = new Runnable() {
        @Override
        public void run() {
            while (!stopThread)

                try {
                    Thread.sleep( millis: 60000);
                    onDestroy();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
        }
    };

    Thread service_task = new Thread(runnable);
    service_task.start();
    return START_STICKY;
}

@Override
public void onDestroy() {
    Log.i(TAG,  msg: "Timp expirat");
    super.onDestroy();
    stopThread = true;
    Intent i = new Intent();
    i.setClass( packageContext: this, MainActivity.class);
    i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(i);
}
```

Figure 7: Serviciu

```java
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View tips_frag = inflater.inflate(R.layout.fragment_tips, container,  attachToRoot: false);
    TextView textView = tips_frag.findViewById(R.id.clues_text);
    textView.setBackgroundResource(R.color.white);
    textView.setText(this.getArguments().getString( key: "cheie"));
    return  tips_frag;
}
```

Figure 8: Fragment

```java
@Override
public void onResponse(@NonNull okhttp3.Call call, @NonNull okhttp3.Response response) throws IOException {
    String recData = response.body().string();
    JSONObject obj;
    TextView ghicitoare = findViewById(R.id.ghicitoare);
    try {
        obj = new JSONObject(recData);
        JSONObject b = obj.getJSONObject("record");
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                try {
                    ghicitoare.setText(b.getString( name: "question" + idDeTrimis));
                    mesajFinal = b.getString( name: "hint" + idDeTrimis);
                    raspunsCorect = b.getString( name: "answer" + idDeTrimis);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        });
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

});
```


Figure 9: Extragere date din JSON

```java
mySharedPref = getSharedPreferences( name: "ansDetails", Context.MODE_PRIVATE);
mySharedPrefEditor = mySharedPref.edit();

textView.setText(mySharedPref.getString( key: "myAns",  defValue: ""));
```

Figure 10: Accesare SharedPreferences

3 Testare

Aplicatia a fost testata de catre mine si alte persoane voluntare, care au dorit sa incerce aplicatia.

4 Concluzii

In concluzie, proiectul reprezinta o metoda ludica, ce imbina diferite elemente de programare a aplicatiilor mobile, pentru a-si indeplini scopul, acela de a livra o aplicatie functionala, ce adreseaza ghicitori in limba engleza si contorizeaza raspunsurile corecte.