UNIVERZITET U KRAGUJEVCU
FAKULTET INŽENJERSKIH NAUKA

Programiranje mobilnih aplikacija

Seminarski rad:

Kviz tehnike i tehnologije

Profesor:

prof. dr Nenad Grujović

Student:

Nikola Stanojević 631/2017

Asistent:

Vukašin Slavković

Kragujevac, februar 2021. godine

Sadržaj

1. Uvod

Život u današnje vreme je gotovo ne zamisliv bez mobilnih telefona. Oni nam pored osnovnih namena u vidu komunikacije pružaju i druge pogodnosti u vidu zabave i razonode. Tu se pre svega misli na različite društvene mreže, igrice i druge aplikacije koje nam popunjavaju slobodno vreme. Mobilni telefoni su zamenili dosta uređaja, pa nam telefon služi i kao sat, budilnik, novine, kamera, Mp3 player, radio, kalendar, digitron, baterijska lampa… [3]

Najzastupljeniji operativni sistem ,,pametnih" telefona je Android, koji je razvio Google 2005. godine. Upravo je on korišćen u izradi projektnog zadatka. Tema projektnog zadatka je kreiranje mobilne aplikacije koja predstavlja kviz tehnike i tehnologije. Aplikacija je kreirana u razvojnom okruženju Android Studio.

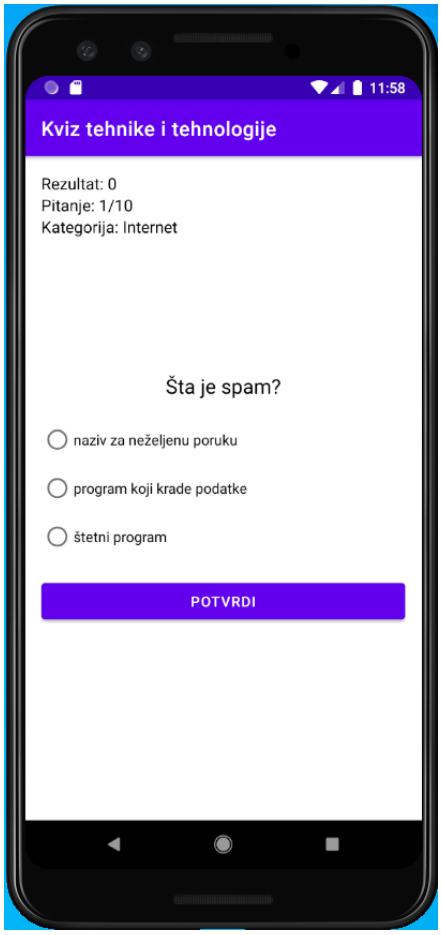
Slika 1. Android logo

2. Opis rada aplikacije

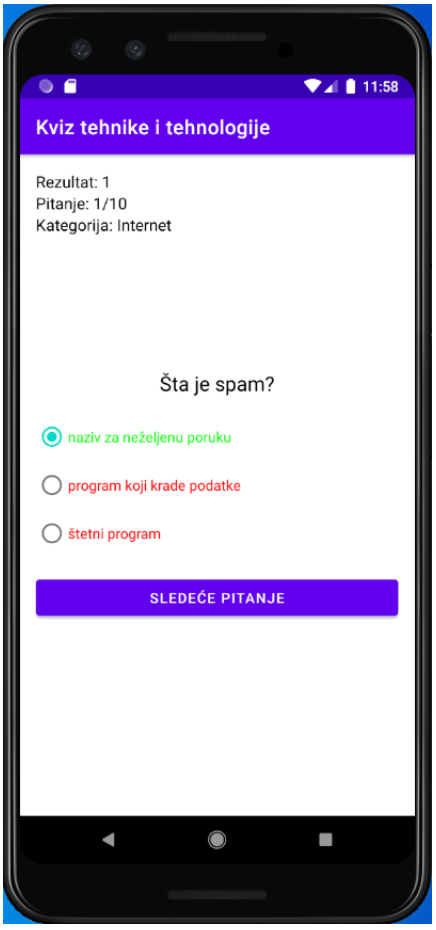
U ovom poglavlju biće opisan način rada aplikacije. Aplikacija kviz tehnike i tehnologije predstavlja jedan vid razonode u kome se može naučiti po neka zanimljivost iz oblasti tehnike i tehnologije. Kviz omogućava odabir jednu od 6 kategorija (internet, automobili, telefoni, kompjuteri, kućni uređaji, mašinstvo) iz kojih će korisniku biti postavljena pitanja, pa je na početnom ekranu aplikacije omogućen odabir kategorije i dugme za pokretanje kviza.

Slika 2. Početni ekran aplikacije

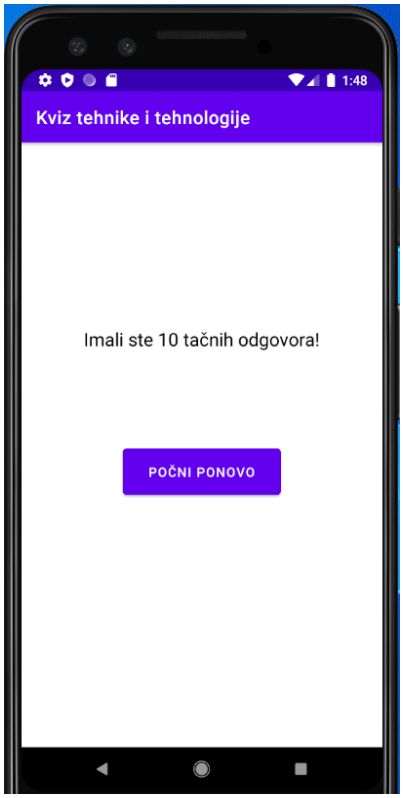
Nakon odabira kategorije i dvoklikom na gume ,,Započni kviz" otvara se novi intent na kome se nalazi trenutni broj poena, izabrana kategorija i redni broj pitanja. Na svako pitanje su data 3 ponudjena odgovora od kojih je samo jedan tačan. Klikom na ,,radio button" se selektuje odgovor i potvrdjuje se klikom na dugme ,,Potvrdi". Ukoliko je odgovor tačan trenutni skor će se uvećati za jedan, takodje tačan odogovor će biti obojen u zeleno, dok će netačni odgovori biti obojeni u crveno. Klikom na dugme ,, Sledeće pitanje" na ekranu se pojavljuje naredno pitanje. Kviz se uvek sastoji iz 10 pitanja koja se nalaze u bazi, koja sadrži oko 200 pitanja.



Slika 3. Ekran pre klika na Potvrdi



Slika 4. Posle klika na Potvrdi



Slika 5. Rezultat

I konačno, nakon 10 pitanja, sledi ispis na ekranu koliko tačnih odgovora je imao korisnik i mogućmost da se klikom na gume vrati na početni ekran.

3. Softverska realizacija aplikacije

Nakon što smo opisali način rada aplikacije, na redu je da se opiše pozadina aplikacije, tj. programski kod. Pisanje programskog koda je urađenu unutar razvojnog okruženja – Android Studio IDE.

Aplikacija komunicaira sa bazom podataka. Baza podataka je implementirana preko SQLite-a. Najpre ćemo opisati kod vezan za bazu podataka i 3 pomoćne java klase. A potom i 3 aktiviti klase sa njihovim *layout*-om.

3.1 Klasa Category

Klasa Categoty predstavlja klasu kategorije pitanja. Sadrži dva polja i 6 statičkih članova koji predstavljaju *integer-e* kategorija. Sadrži dva konstruktora i metode *getter i setter* za polja klase, kao i metodu *toString().*

```
1   package com.example.projekat_kviztehnike;
2
3   public class Category {
4       public static final int INTERNET = 1;
5       public static final int AUTOMOBILI = 2;
6       public static final int KOMPJUTERI = 3;
7       public static final int TELEFONI = 4;
8       public static final int KUCNI_UREDJAJI = 5;
9       public static final int MASINSTVO = 6;
10      private int id;
11      private String name;
12      public Category() {
13      }
14      public Category(String name) { this.name = name; }
17      public int getId() { return id; }
20      public void setId(int id) { this.id = id; }
23      public String getName() { return name; }
26      public void setName(String name) { this.name = name; }
29      @Override
30      public String toString() { return getName(); }
33  }
```

Slika 6. Klasa Category

3.2 Klasa Question

Klasa predstavlja klasu pitanje, ona implementira interfejs ,,*Parcelable"* koji je pandan javinog interfejsa ,,*Serializable".* Formira objekat tipa pitanje. Sadrži 7 polja koja predstavljaju id, tekst, pitanja, 3 opcije, opciju tačnog odgovora i id kategorije kojoj pripada pitanje. Klasa sadrži tri konstruktora, kreator Parcel-a, kao i getter, setter metode za polja klase.[2]

```
1   package com.example.projekat_kviztehnike;
2
3   import android.os.Parcel;
4   import android.os.Parcelable;
5   public class Question implements Parcelable {
6
7       private int id;
8       private String question;
9       private String option1;
10      private String option2;
11      private String option3;
12      private int answerNr;
13      private int categoryID;
14      public Question() {
15      }
16      public Question(String question, String option1, String option2, String option3,
17                      int answerNr, int categoryID) {
18          this.question = question;
19          this.option1 = option1;
20          this.option2 = option2;
21          this.option3 = option3;
22          this.answerNr = answerNr;
23          this.categoryID = categoryID;
24      }
25  @   protected Question(Parcel in) {
26          id = in.readInt();
27          question = in.readString();
28          option1 = in.readString();
29          option2 = in.readString();
30          option3 = in.readString();
31          answerNr = in.readInt();
32          categoryID = in.readInt();
33      }
```

Slika 7. Klasa Question 1/2

```
34      @Override
35  @   public void writeToParcel(Parcel dest, int flags) {
36          dest.writeInt(id);
37          dest.writeString(question);
38          dest.writeString(option1);
39          dest.writeString(option2);
40          dest.writeString(option3);
41          dest.writeInt(answerNr);
42          dest.writeInt(categoryID);
43      }
44      @Override
45      public int describeContents() { return 0; }
48      public static final Creator<Question> CREATOR = new Creator<Question>() {
49          @Override
50  @       public Question createFromParcel(Parcel in) { return new Question(in); }
53          @Override
54  @       public Question[] newArray(int size) { return new Question[size]; }
57      };
58      public int getId() { return id; }
61      public void setId(int id) { this.id = id; }
64      public String getQuestion() { return question; }
67      public void setQuestion(String question) { this.question = question; }
70      public String getOption1() { return option1; }
73      public void setOption1(String option1) { this.option1 = option1; }
76      public String getOption2() { return option2; }
79      public void setOption2(String option2) { this.option2 = option2; }
82      public String getOption3() { return option3; }
85      public void setOption3(String option3) { this.option3 = option3; }
88      public int getAnswerNr() { return answerNr; }
91      public void setAnswerNr(int answerNr) { this.answerNr = answerNr; }
94      public int getCategoryID() { return categoryID; }
97      public void setCategoryID(int categoryID) { this.categoryID = categoryID; }
100 }
```

Slika 8. Klasa Question 2/2

3.3 Klasa QuizContract

Predstavlja pomoćnu klasu za formiranje tabela baze podataka.

```
1   package com.example.projekat_kviztehnike;
2
3   import android.provider.BaseColumns;
4   public final class QuizContract {
5
6       private QuizContract() {
7       }
8       public static class CategoriesTable implements BaseColumns {
9           public static final String TABLE_NAME = "quiz_categories";
10          public static final String COLUMN_NAME = "name";
11      }
12      public static class QuestionsTable implements BaseColumns {
13          public static final String TABLE_NAME = "quiz_questions";
14          public static final String COLUMN_QUESTION = "question";
15          public static final String COLUMN_OPTION1 = "option1";
16          public static final String COLUMN_OPTION2 = "option2";
17          public static final String COLUMN_OPTION3 = "option3";
18          public static final String COLUMN_ANSWER_NR = "answer_nr";
19          public static final String COLUMN_CATEGORY_ID = "category_id";
20      }
21  }
```

Slika 9. Klasa QuizContract

3.4. Klasa QuizDBhelper

Ova klasa nasleđuje klasu *SQLiteOpenHelper* i u njoj se formira baza podataka i tabela kategorija pitanja i tabela pitanja.

```
1   package com.example.projekat_kviztehnike;
2
3   import android.content.ContentValues;
4   import android.content.Context;
5   import android.database.Cursor;
6   import android.database.sqlite.SQLiteDatabase;
7   import android.database.sqlite.SQLiteOpenHelper;
8   import com.example.projekat_kviztehnike.QuizContract.*;
9   import java.util.ArrayList;
10  import java.util.List;
11
12  public class QuizDbHelper extends SQLiteOpenHelper {
13      private static final String DATABASE_NAME = "Kviz.db";
14      private static final int DATABASE_VERSION = 1;
15      private static QuizDbHelper instance;
16      private SQLiteDatabase db;
17      private QuizDbHelper(Context context) { super(context, DATABASE_NAME, factory: null, DATABASE_VERSION); }
20      public static synchronized QuizDbHelper getInstance(Context context) {
21          if (instance == null) {
22              instance = new QuizDbHelper(context.getApplicationContext());
23          }
24          return instance;
25      }
```

Slika 10. Klasa QuizDBhelper 1/5

Funcija *onCreate()* formira dve tabele. Tabela kategorija se sastoji iz dva atributa koji predstavljaju id kategorije i ime kategorije. Tabela pitanja se sastoji iz sedam atributa. Prvi atribut je id pitanja, zatom tekst pitanja, tri opcije, opcija koja daje tačan odgovor, a sedmiatribut je strain ključ koji se referencira tabelu kategorija i predsavlja id kategorije kojoj pitanje pripada.

Nakon formiranja tabela, pozivaju se dve funcije *fillCategoriesTable() i fillQuestionsTable()* koje popunjvaju tabele podacima. Obe funkcije prvo formiraju objekat tipa kategorija ili pitanje na osnovu klasa koje smo već opisali i pomoću funkcija *insertCategory() i insertQuestion*() ubacuju u bazu. Obe funcije koriste objekte tipa *ContentValues* u koje se smešta kategorija ili pitanje i onda se pomoću ugrađene funkcije *insert()* ubacuju u bazu.

Klasa sadrži i dve metode *getAllCategories() i getQuestions(int categoryID).*

getAllCategories() vraća listu kategorija koje su unite u bazu, dok druga metoda ima ulazni parameter *int categoryID* koji označava id kategorije. Funcija vraća sva pitanja koja pripadaju toj kategoriji.

```
26      @Override
27  @   public void onCreate(SQLiteDatabase db) {
28          this.db = db;
29          final String SQL_CREATE_CATEGORIES_TABLE = "CREATE TABLE " +
30                  CategoriesTable.TABLE_NAME + "( " +
31                  CategoriesTable._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
32                  CategoriesTable.COLUMN_NAME + " TEXT " +
33                  ")";
34          final String SQL_CREATE_QUESTIONS_TABLE = "CREATE TABLE " +
35                  QuestionsTable.TABLE_NAME + " ( " +
36                  QuestionsTable._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
37                  QuestionsTable.COLUMN_QUESTION + " TEXT, " +
38                  QuestionsTable.COLUMN_OPTION1 + " TEXT, " +
39                  QuestionsTable.COLUMN_OPTION2 + " TEXT, " +
40                  QuestionsTable.COLUMN_OPTION3 + " TEXT, " +
41                  QuestionsTable.COLUMN_ANSWER_NR + " INTEGER, " +
42                  QuestionsTable.COLUMN_CATEGORY_ID + " INTEGER, " +
43                  "FOREIGN KEY(" + QuestionsTable.COLUMN_CATEGORY_ID + ") REFERENCES " +
44                  CategoriesTable.TABLE_NAME + "(" + CategoriesTable._ID + ")" + "ON DELETE CASCADE" +
45                  ")";
46          db.execSQL(SQL_CREATE_CATEGORIES_TABLE);
47          db.execSQL(SQL_CREATE_QUESTIONS_TABLE);
48          fillCategoriesTable();
49          fillQuestionsTable();
50      }
51      @Override
52  @   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
53          db.execSQL("DROP TABLE IF EXISTS " + CategoriesTable.TABLE_NAME);
54          db.execSQL("DROP TABLE IF EXISTS " + QuestionsTable.TABLE_NAME);
55          onCreate(db);
56      }
```

Slika 11. Klasa QuizDBhelper 2/5

```
62      private void fillCategoriesTable() {
63          Category c1 = new Category( name: "Internet");
64          insertCategory(c1);
65          Category c2 = new Category( name: "Automobili");
66          insertCategory(c2);
67          Category c3 = new Category( name: "Kompjuteri");
68          insertCategory(c3);
69          Category c4 = new Category( name: "Telefoni");
70          insertCategory(c4);
71          Category c5 = new Category( name: "Kućni uredjaji");
72          insertCategory(c5);
73          Category c6 = new Category( name: "Mašinstvo");
74          insertCategory(c6);
75      }
76
77  @   private void insertCategory(Category category) {
78          ContentValues cv = new ContentValues();
79          cv.put(CategoriesTable.COLUMN_NAME, category.getName());
80          db.insert(CategoriesTable.TABLE_NAME,  nullColumnHack: null, cv);
81      }
82      private void fillQuestionsTable() {
83
84          //INTERNET
85
86          Question q1 = new Question( question: "Engl. attachment je?",
87                  option1: "datoteka na mrežnoj stranici",  option2: "datoteka ugradjena u dokumentu",
88                  option3: "datoteka unutar e-poruke",  answerNr: 3, Category.INTERNET);
89          insertQuestion(q1);
```

Slika 12. Klasa QuizDBhelper 3/5

```
1025    @   private void insertQuestion(Question question) {
1026            ContentValues cv = new ContentValues();
1027            cv.put(QuestionsTable.COLUMN_QUESTION, question.getQuestion());
1028            cv.put(QuestionsTable.COLUMN_OPTION1, question.getOption1());
1029            cv.put(QuestionsTable.COLUMN_OPTION2, question.getOption2());
1030            cv.put(QuestionsTable.COLUMN_OPTION3, question.getOption3());
1031            cv.put(QuestionsTable.COLUMN_ANSWER_NR, question.getAnswerNr());
1032            cv.put(QuestionsTable.COLUMN_CATEGORY_ID, question.getCategoryID());
1033            db.insert(QuestionsTable.TABLE_NAME,  nullColumnHack: null, cv);
1034        }
1035
1036        public List<Category> getAllCategories() {
1037            List<Category> categoryList = new ArrayList<>();
1038            db = getReadableDatabase();
1039            Cursor c = db.rawQuery( sql: "SELECT * FROM " + CategoriesTable.TABLE_NAME,  selectionArgs: null);
1040            if (c.moveToFirst()) {
1041                do {
1042                    Category category = new Category();
1043                    category.setId(c.getInt(c.getColumnIndex(CategoriesTable._ID)));
1044                    category.setName(c.getString(c.getColumnIndex(CategoriesTable.COLUMN_NAME)));
1045                    categoryList.add(category);
1046                } while (c.moveToNext());
1047            }
1048            c.close();
1049            return categoryList;
1050        }
```

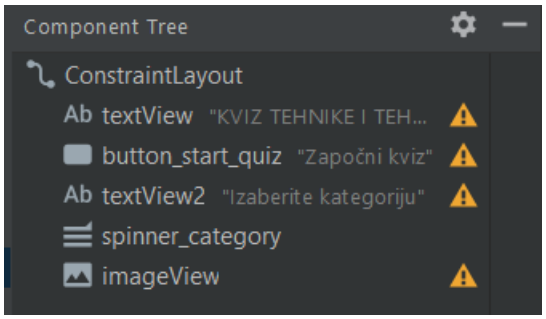
Slika 13. Klasa QuizDBhelper 4/5

```
1052        public ArrayList<Question> getQuestions(int categoryID) {
1053            ArrayList<Question> questionList = new ArrayList<>();
1054            db = getReadableDatabase();
1055            String selection = QuestionsTable.COLUMN_CATEGORY_ID + " = ? ";
1056            String[] selectionArgs = new String[]{String.valueOf(categoryID)};
1057            Cursor c = db.query(
1058                    QuestionsTable.TABLE_NAME,
1059                     columns: null,
1060                    selection,
1061                    selectionArgs,
1062                     groupBy: null,
1063                     having: null,
1064                     orderBy: null
1065            );
1066            if (c.moveToFirst()) {
1067                do {
1068                    Question question = new Question();
1069                    question.setId(c.getInt(c.getColumnIndex(QuestionsTable._ID)));
1070                    question.setQuestion(c.getString(c.getColumnIndex(QuestionsTable.COLUMN_QUESTION)));
1071                    question.setOption1(c.getString(c.getColumnIndex(QuestionsTable.COLUMN_OPTION1)));
1072                    question.setOption2(c.getString(c.getColumnIndex(QuestionsTable.COLUMN_OPTION2)));
1073                    question.setOption3(c.getString(c.getColumnIndex(QuestionsTable.COLUMN_OPTION3)));
1074                    question.setAnswerNr(c.getInt(c.getColumnIndex(QuestionsTable.COLUMN_ANSWER_NR)));
1075                    question.setCategoryID(c.getInt(c.getColumnIndex(QuestionsTable.COLUMN_CATEGORY_ID)));
1076                    questionList.add(question);
1077                } while (c.moveToNext());
1078            }
1079            c.close();
1080            return questionList;
1081        }
1082    }
```

Slika 14. QuizDBhelper 5/5

3.5. Aktivity Main

Nakon opisa baze podataka na redu je opis tri aktivitija koja čine aplikaciju. Prvi je *MainActivity* koji predstavlja početni ekran, drugi je *QuizActivity* koji predstavlja ekran gde se postavljaju i odgovara na pitanja, i treći *ResultActivity* gde se ispisuje broj tačnih odogovora.



Slika 15. GUI komponente MainActivity-ja

Na početku je potrebno prvo dodati grafičke komponente kao što su slike, tasteri, polja za ispis… Na slici 15 je dat strukturni prikaz svih grafičkih komponenti koji se koriste u *MainActivity*-ju. Sa slike vidimo da je korišćen *ConstraintLayout.*

Na početku klase se definišu stri statička člana koja služe kasnije za *Intente*, tj za prenos informacija o kategoriji narednom *activity*-ju.

Definiše se polje *spinnercategory* tipa *Spinner* taster *buttonStartQuiz*. Dodeljuje im se vrednost spinnera I taster koji se definisani u layout-u.

```
14  public class MainActivity extends AppCompatActivity {
15      private static final int REQUEST_CODE_QUIZ = 1;
16      public static final String EXTRA_CATEGORY_ID = "extraCategoryID";
17      public static final String EXTRA_CATEGORY_NAME = "extraCategoryName";
18
19      private Spinner spinnerCategory;
20
21      @Override
22      protected void onCreate(Bundle savedInstanceState) {
23          super.onCreate(savedInstanceState);
24          setContentView(R.layout.activity_main);
25
26
27          spinnerCategory = findViewById(R.id.spinner_category);
28
29          loadCategories();
30          Button buttonStartQuiz = findViewById(R.id.button_start_quiz);
31          buttonStartQuiz.setOnClickListener(new View.OnClickListener() {
32              @Override
33              public void onClick(View v) { startQuiz(); }
36          });
37      }
```

Slika 16. MainActivity 1/2

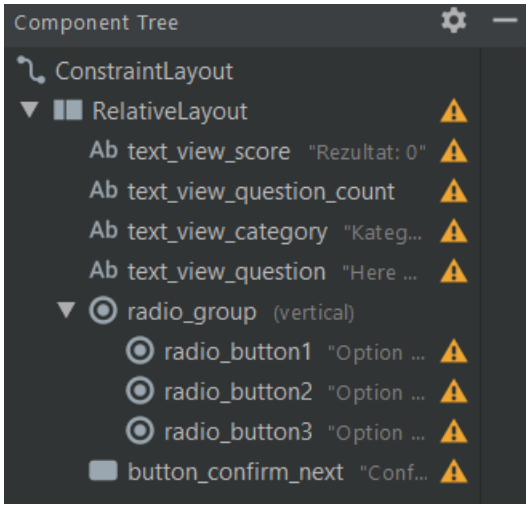
Definiše se da se klikon na dugme *buttonStartQuiz* prenosi informacija koja je kategorija izabrana *QuizActivity*-ju i otvara se novi activity.

Definisana je funkcija *loadCategories()* koja iz baze vraća spisak kategorija i formira adapter na *spinner*-u pomoću koga korisnik bira željenu kategoriju.

```
38      private void startQuiz() {
39          Category selectedCategory = (Category) spinnerCategory.getSelectedItem();
40          int categoryID = selectedCategory.getId();
41          String categoryName = selectedCategory.getName();
42          Intent intent = new Intent( packageContext: MainActivity.this, QuizActivity.class);
43          intent.putExtra(EXTRA_CATEGORY_ID, categoryID);
44          intent.putExtra(EXTRA_CATEGORY_NAME, categoryName);
45          startActivityForResult(intent, REQUEST_CODE_QUIZ);
46      }
47      @Override
48      protected void onActivityResult(int requestCode, int resultCode, Intent data) {
49          super.onActivityResult(requestCode, resultCode, data);
50          if (requestCode == REQUEST_CODE_QUIZ) {
51              if (resultCode == RESULT_OK) {
52              }
53          }
54      }
55      private void loadCategories() {
56          QuizDbHelper dbHelper = QuizDbHelper.getInstance(this);
57          List<Category> categories = dbHelper.getAllCategories();
58          ArrayAdapter<Category> adapterCategories = new ArrayAdapter<>( context: this,
59                  android.R.layout.simple_spinner_item, categories);
60          adapterCategories.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
61          spinnerCategory.setAdapter(adapterCategories);
62      }
63
64  }
```

Slika 17. MainActivity 2/2

3.6. Quiz Activity



Slika 18. GUI Quiz Activity

Na slici 18. je dat strukturni prikaz svih grafičkih komponenti koji se koriste u ovom aktivitiju. Data su 4 *TextView* elementa na kojima se ispisuje trenutni rezultat, broj pitanja, kategorija pitanja i tekst pitanja. Pomoću *radio_group* se bira jedan od odgovra i potvrdjuje se klikom na taster *Potvrdi*.

```
21  public class QuizActivity extends AppCompatActivity {
22      private static final String KEY_SCORE = "keyScore";
23      private static final String KEY_QUESTION_COUNT = "keyQuestionCount";
24      private static final String KEY_ANSWERED = "keyAnswered";
25      private static final String KEY_QUESTION_LIST = "keyQuestionList";
26      private TextView textViewQuestion;
27      private TextView textViewScore;
28      private TextView textViewQuestionCount;
29      private TextView textViewCategory;
30      private RadioGroup rbGroup;
31      private RadioButton rb1;
32      private RadioButton rb2;
33      private RadioButton rb3;
34      private Button buttonConfirmNext;
35      private ColorStateList textColorDefaultRb;
36      private ArrayList<Question> questionList;
37      private int questionCounter;
38      private int questionCountTotal;
39      private Question currentQuestion;
40      private int score;
41      private boolean answered;
42      @Override
43      protected void onCreate(Bundle savedInstanceState) {
44          super.onCreate(savedInstanceState);
45          setContentView(R.layout.activity_quiz);
46          textViewQuestion = findViewById(R.id.text_view_question);
47          textViewScore = findViewById(R.id.text_view_score);
48          textViewQuestionCount = findViewById(R.id.text_view_question_count);
49          textViewCategory = findViewById(R.id.text_view_category);
50          rbGroup = findViewById(R.id.radio_group);
51          rb1 = findViewById(R.id.radio_button1);
52          rb2 = findViewById(R.id.radio_button2);
53          rb3 = findViewById(R.id.radio_button3);
54          buttonConfirmNext = findViewById(R.id.button_confirm_next);
55          textColorDefaultRb = rb1.getTextColors();
```

Slika 19. QuizActivity 1/4

Na početku klase activity-ja definišu se i inicijalizuju polja koja odgovaraju grafičkim komponentama u layout fajlu.Nakon toga se pomoću intenta dobijaju informacije o kategoriji koju je korisnik izabrao i formira se lista pitanja iz te kategorije. Lista pitanje se ,,mesa" tako da se pitanja postavljaju nasumično.

```
57          Intent intent = getIntent();
58          int categoryID = intent.getIntExtra(MainActivity.EXTRA_CATEGORY_ID,  defaultValue: 0);
59          String categoryName = intent.getStringExtra(MainActivity.EXTRA_CATEGORY_NAME);
60          textViewCategory.setText("Kategorija: " + categoryName);
61
62          if (savedInstanceState == null)
63          {
64              QuizDbHelper dbHelper = QuizDbHelper.getInstance(this);
65              questionList = dbHelper.getQuestions(categoryID);
66              questionCountTotal = questionList.size();
67              Collections.shuffle(questionList);
68              showNextQuestion();
69          }
70          else
71              {
72              questionList = savedInstanceState.getParcelableArrayList(KEY_QUESTION_LIST);
73              questionCountTotal = questionList.size();
74              questionCounter = savedInstanceState.getInt(KEY_QUESTION_COUNT);
75              currentQuestion = questionList.get(questionCounter - 1);
76              score = savedInstanceState.getInt(KEY_SCORE);
77              answered = savedInstanceState.getBoolean(KEY_ANSWERED);
78              if (!answered) {
79              } else {
80                  showSolution();
81              }
82          }
```

Slika 20. QuizActivity 2/4

Klikom na taster potvrdi proverava se da li je odabrana neka od opcija, ako nije, korisniku se pojavljuje obaveštenje da mora da odabere svoj odgovor. Ako je odabrao odgovor poziva se funkcija *checkAnswer()* koja proverava tačnost odgovora. Ako je već prvereno pitanje onda se drugim klikom prelazi na sledeće pitanje funkcijom *showNextQuestion().*

```
84          buttonConfirmNext.setOnClickListener(new View.OnClickListener() {
85              @Override
86              public void onClick(View v) {
87                  if (!answered) {
88                      if (rb1.isChecked() || rb2.isChecked() || rb3.isChecked()) {
89                          checkAnswer();
90                      } else {
91                          Toast.makeText( context: QuizActivity.this,  text: "Morate odabrati odgovor", Toast.LENGTH_SHORT).show();
92                      }
93                  } else {
94                      showNextQuestion();
95                  }
96              }
97          });
98      }
99      private void showNextQuestion() {
100         rb1.setTextColor(textColorDefaultRb);
101         rb2.setTextColor(textColorDefaultRb);
102         rb3.setTextColor(textColorDefaultRb);
103         rbGroup.clearCheck();
104         if (questionCounter < 10) {
105             currentQuestion = questionList.get(questionCounter);
106             textViewQuestion.setText(currentQuestion.getQuestion());
107             rb1.setText(currentQuestion.getOption1());
108             rb2.setText(currentQuestion.getOption2());
109             rb3.setText(currentQuestion.getOption3());
110             questionCounter++;
111             textViewQuestionCount.setText("Pitanje: " + questionCounter + "/" + "10");
112             answered = false;
113             buttonConfirmNext.setText("Potvrdi");
114         } else {
115             Intent intent = new Intent( packageContext: QuizActivity.this, Result.class);
116             intent.putExtra( name: "rezultat", score);
117             startActivity(intent);
118         }
```

Slika 21. QuizActivity 3/4

Funcija *showNextQuestion()* prvo postavlja sve odogovore početnom bojom i briše čekirano dugme u prethodnom pitanju. Nakon toga proverava da li je postavljeno 10 pitanja, ako nije onda ,,vadi" pitanje iz liste pitanja i prikazuje ga na grafičkim komponentama. Ako je postavljeno svih 10 pitanja onda se intentom prelazi na sledeći *Result Activity* kome se šalje broj tačnih odgovora.

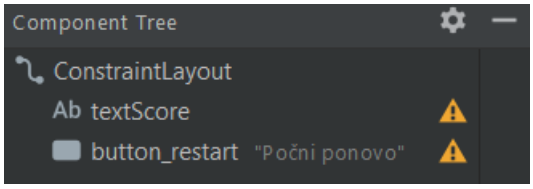
Funcija *checkAnswer()* preverava tačnost odgovora, tj da li selektovana opcija tačna, ako jeste onda se rezultat uvećava za 1. I nakon toga se prikazuje korisniku tačan odgovor koji je prikazan zelenom bojom, dok su netačni odgovori crvenom. Taj deo posla obavlja funcija *showSolution().*

```
121     private void checkAnswer() {
122         answered = true;
123         RadioButton rbSelected = findViewById(rbGroup.getCheckedRadioButtonId());
124         int answerNr = rbGroup.indexOfChild(rbSelected) + 1;
125         if (answerNr == currentQuestion.getAnswerNr()) {
126             score++;
127             textViewScore.setText("Rezultat: " + score);
128         }
129         showSolution();
130     }
131     private void showSolution() {
132         rb1.setTextColor(Color.RED);
133         rb2.setTextColor(Color.RED);
134         rb3.setTextColor(Color.RED);
135         switch (currentQuestion.getAnswerNr()) {
136             case 1:
137                 rb1.setTextColor(Color.GREEN);
138                 break;
139             case 2:
140                 rb2.setTextColor(Color.GREEN);
141                 break;
142             case 3:
143                 rb3.setTextColor(Color.GREEN);
144                 break;
145         }
146         if (questionCounter < questionCountTotal) {
147             buttonConfirmNext.setText("Sledeće pitanje");
148         } else {
149             buttonConfirmNext.setText("Završi kviz");
150         }
151     }
```

Slika 22. QuizActivity 4/4

3.7. Result Activity

U ovom aktivitiju postoje samo dve grafičke komponente. Prva na kojoj se ispisuje rezultat testa i druga koja označava taster povratka na početnu stranu.



Slika 23. GUI Result Activity

U kodu su definisana polja koja označavaju grafičke komponente. Pomoću intent se dobija informacija o rezultatu testa korisnika i taj rezultat se ispisuje na ekran.

Definisan je taster povratka na početnu stranu.

```
10
11  public class Result extends AppCompatActivity {
12
13      TextView rezultat;
14      Button pocni_ponovo;
15      int rezultatt;
16
17      @Override
18      protected void onCreate(Bundle savedInstanceState) {
19          super.onCreate(savedInstanceState);
20          setContentView(R.layout.activity_result);
21
22          rezultat = findViewById(R.id.textScore);
23          pocni_ponovo = findViewById(R.id.button_restart);
24
25          if (getIntent().hasExtra( name: "rezultat"))
26          {
27
28              rezultatt = getIntent().getIntExtra( name: "rezultat",  defaultValue: 0);
29              rezultat.setText("Imali ste " + rezultatt + " tačnih odgovora!");
30          }
31
32          pocni_ponovo.setOnClickListener(new View.OnClickListener() {
33              @Override
34              public void onClick(View v) {
35                  Intent intent = new Intent( packageContext: Result.this, MainActivity.class);
36                  startActivity(intent);
37              }
38          });
39      }
40  }
```

Slika 24. Result Activity

4. Literatura

1. Moodle PMA - http://moodle.fink.rs/course/view.php?id=982 19.2.2021.
2. Android Documentation - https://developer.android.com/docs 15.2.2021.
3. https://promobi.rs/9-stvari-koje-su-zamenili-mobilni-telefoni/ 20.2.2021
4. Stack Overflow - https://stackoverflow.com 15.2.2021.