



Akademia WSB

Wydział Zamiejscowy w Cieszynie

Jakub Czyż

Numer albumu : 38300

„SQLite”

Zadanie z przedmiotu

„Aplikacje w języku Java”

Cieszyn, 2021 r.

Spis treści

Utworzenie klasy QuestionsContract.....	3
Klasa SQLiteOpenHelper	4
Utworzenie metody dodającej pytania do bazy danych	5
Utworzenie metody wczytującej pytania z bazy danych	6
Wdrożenie bazy danych w istniejącej aplikacji.....	7
Utworzenie rankingu najlepszych graczy za pomocą bazy danych SQLite.....	8
Utworzenie bazy danych Scores	9
RecyclerView oraz wyświetlenie rankingu na ekranie gry	10
Klasa MyAdapter.....	13
Zapisanie wyniku gracza w bazie danych	14
Wygląd działania aplikacji	15

Utworzenie klasy QuestionsContract

1. Tworzę klasę o nazwie **QuestionsContract**, która będzie zawierać konstruktor bezparametrowy oraz klasę statyczną, która przechowywać będzie nazwę tabeli oraz kolumn w tworzonej przeze mnie bazie danych:

```
7      private QuestionsContract() {}
8
9      public static class QuestionsEntry implements BaseColumns {
10         public static final String TABLE_NAME = "questions";
11         public static final String COLUMNN_NAME_CONTENT = "content";
12         public static final String COLUMNN_NAME_ANSWER_A = "answer_a";
13         public static final String COLUMNN_NAME_ANSWER_B = "answer_b";
14         public static final String COLUMNN_NAME_ANSWER_C = "answer_c";
15         public static final String COLUMNN_NAME_ANSWER_D = "answer_d";
16         public static final String COLUMNN_NAME_CORRECT = "correct";
17     }
```

2. Następnie tworzę zapytanie SQL, którego użyję później w celu utworzenia tabeli w mojej bazie danych:

```
19     public static final String SQL_CREATE_ENTRIES =
20         "CREATE TABLE " + QuestionsEntry.TABLE_NAME + " (" +
21             QuestionsEntry._ID + " INTEGER PRIMARY KEY," +
22             QuestionsEntry.COLUMNN_NAME_CONTENT + " TEXT," +
23             QuestionsEntry.COLUMNN_NAME_ANSWER_A + " TEXT," +
24             QuestionsEntry.COLUMNN_NAME_ANSWER_B + " TEXT," +
25             QuestionsEntry.COLUMNN_NAME_ANSWER_C + " TEXT," +
26             QuestionsEntry.COLUMNN_NAME_ANSWER_D + " TEXT," +
27             QuestionsEntry.COLUMNN_NAME_CORRECT + " TEXT)";
```

3. Tworzę także zapytanie usuwające tabelę w mojej bazie danych:

```
29     public static final String SQL_DELETE_ENTRIES =
30         "DROP TABLE IF EXISTS " + QuestionsEntry.TABLE_NAME;
```

Klasa SQLiteOpenHelper

1. Utworzenie klasy o nazwie **QuestionsDbHelper** oraz rozszerzenie jej o klasę **SQLiteOpenHelper**. Zaimplementowanie wymaganych metod **onCreate()** oraz **onUpgrade()**:

```
1 package com.example.quiz.database;
2
3 import android.database.sqlite.SQLiteDatabase;
4 import android.database.sqlite.SQLiteOpenHelper;
5
6 public class QuestionsDbHelper extends SQLiteOpenHelper {
7     @Override
8     public void onCreate(SQLiteDatabase db) {
9
10    }
11
12    @Override
13    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
14
15    }
16 }
17
```

2. Utworzenie zmiennych statycznych zawierających informację o wersji bazy danych oraz jej nazwie:

```
8 public static final int DATABASE_VERSION = 1;
9 public static final String DATABASE_NAME = "Questions.db";
```

3. Utworzenie konstruktora klasy **QuestionsDbHelper**:

```
12 public QuestionsDbHelper(Context context) {
13     super(context, DATABASE_NAME, null, DATABASE_VERSION);
14 }
```

4. Utworzenie nowej tabeli za pomocą utworzonego wcześniej zapytania SQL w metodzie **onCreate()**:

```
21 @Override
22 public void onCreate(SQLiteDatabase db) {
23     db.execSQL(SQL_CREATE_ENTRIES);
24 }
```

5. Implementacja metody **onUpgrade**:

```
26 @Override
27 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
28     db.execSQL(SQL_DELETE_ENTRIES);
29     onCreate(db);
30 }
```

Utworzenie metody dodającej pytania do bazy danych

1. W utworzonej wcześniej klasie **QuestionDbHelper** tworzę nową metodę, której zadaniem będzie dodanie nowych rekordów do bazy danych po wcześniejszym sprawdzeniu czy pytanie o takiej samej treści nie znajduje się już w istniejącej bazie danych. Pytania są tworzone na bazie obiektu klasy **SingleAnswerQuestion**.

```
36 @ public void addQuestionToDatabase(QuestionsDbHelper questionsDbHelper, SingleAnswerQuestion singleAnswerQuestion) {
37     SQLiteDatabase db = questionsDbHelper.getWritableDatabase();
38
39     Cursor cursor = db.rawQuery( sql: "SELECT * FROM questions WHERE content='"+singleAnswerQuestion.getQuestionContent()+"'", selectionArgs: null);
40
41     if (cursor.moveToFirst()) {
42         System.out.println("Podany rekord już istnieje w bazie danych!");
43     } else {
44         ContentValues values = new ContentValues();
45         values.put(QuestionsContract.QuestionsEntry.COLUMN_NAME_CONTENT, singleAnswerQuestion.getQuestionContent());
46         values.put(QuestionsContract.QuestionsEntry.COLUMN_NAME_ANSWER_A, singleAnswerQuestion.getQuestionAnswerA());
47         values.put(QuestionsContract.QuestionsEntry.COLUMN_NAME_ANSWER_B, singleAnswerQuestion.getQuestionAnswerB());
48         values.put(QuestionsContract.QuestionsEntry.COLUMN_NAME_ANSWER_C, singleAnswerQuestion.getQuestionAnswerC());
49         values.put(QuestionsContract.QuestionsEntry.COLUMN_NAME_ANSWER_D, singleAnswerQuestion.getQuestionAnswerD());
50         values.put(QuestionsContract.QuestionsEntry.COLUMN_NAME_CORRECT, singleAnswerQuestion.getCorrectAnswer());
51
52         long newRowId = db.insert(QuestionsContract.QuestionsEntry.TABLE_NAME, nullColumnHack: null, values);
53     }
54 }
```

Utworzenie metody wczytującej pytania z bazy danych

1. W klasie **QuestionDbHelper** tworzę metodę, która pobiera pytania z istniejącej bazy danych i zwraca je w postaci listy. Jej działanie jest następujące:
 - tworzę tablicę typu **String**, która definiuje kolumny do zwrócenia po wykonaniu mojego zapytania (w zasadzie pobieram wszystkie i mógłbym pominąć ten krok a jako argument przekazać wartość **null** ☺);
 - następnie tworzę nowy obiekt typu **Cursor**, do którego przekazuję jako argumenty nazwę tabeli oraz utworzoną powyżej tablicę **projection**;
 - tworzę pętlę, w której to obiekt **cursor** będzie przemieszczać się po kolejnych rekordach w mojej tabeli i zapisuję pobrane wartości do poszczególnych zmiennych;
 - tworzę nowe obiekty klasy **SingleAnswerQuestion** z wykorzystaniem utworzonych zmiennych i dodaję je do mojej listy;
 - zamykam **cursor**:

```
53 @ public LinkedList<SingleAnswerQuestion> readQuestionsFromDatabase(QuestionsDbHelper questionsDbHelper) {
54     SQLiteDatabase db = questionsDbHelper.getReadableDatabase();
55
56     LinkedList<SingleAnswerQuestion> singleAnswerQuestions = new LinkedList<>();
57
58     String[] projection = {
59         QuestionsContract.QuestionsEntry.COLUMN_NAME_CONTENT,
60         QuestionsContract.QuestionsEntry.COLUMN_NAME_ANSWER_A,
61         QuestionsContract.QuestionsEntry.COLUMN_NAME_ANSWER_B,
62         QuestionsContract.QuestionsEntry.COLUMN_NAME_ANSWER_C,
63         QuestionsContract.QuestionsEntry.COLUMN_NAME_ANSWER_D,
64         QuestionsContract.QuestionsEntry.COLUMN_NAME_CORRECT
65     };
66
67     Cursor cursor = db.query(
68         QuestionsContract.QuestionsEntry.TABLE_NAME,
69         projection,
70         selection: null,
71         selectionArgs: null,
72         groupBy: null,
73         having: null,
74         orderBy: null
75     );
76
77     while(cursor.moveToNext()) {
78         String content = cursor.getString(
79             cursor.getColumnIndexOrThrow(QuestionsContract.QuestionsEntry.COLUMN_NAME_CONTENT));
80         String answerA = cursor.getString(
81             cursor.getColumnIndexOrThrow(QuestionsContract.QuestionsEntry.COLUMN_NAME_ANSWER_A));
82         String answerB = cursor.getString(
83             cursor.getColumnIndexOrThrow(QuestionsContract.QuestionsEntry.COLUMN_NAME_ANSWER_B));
84         String answerC = cursor.getString(
85             cursor.getColumnIndexOrThrow(QuestionsContract.QuestionsEntry.COLUMN_NAME_ANSWER_C));
86         String answerD = cursor.getString(
87             cursor.getColumnIndexOrThrow(QuestionsContract.QuestionsEntry.COLUMN_NAME_ANSWER_D));
88         String correct = cursor.getString(
89             cursor.getColumnIndexOrThrow(QuestionsContract.QuestionsEntry.COLUMN_NAME_CORRECT));
90         singleAnswerQuestions.add(new SingleAnswerQuestion(content, answerA, answerB, answerC, answerD, Integer.parseInt(correct)));
91     }
92     cursor.close();
93
94     return singleAnswerQuestions;
95 }
```

Wdrożenie bazy danych w istniejącej aplikacji

1. Tworzę instancję klasy **questionsDbHelper** w istniejącej już klasie **SingleAnswerActivity**:

```
53      QuestionsDbHelper questionsDbHelper = new QuestionsDbHelper( context, this);
```

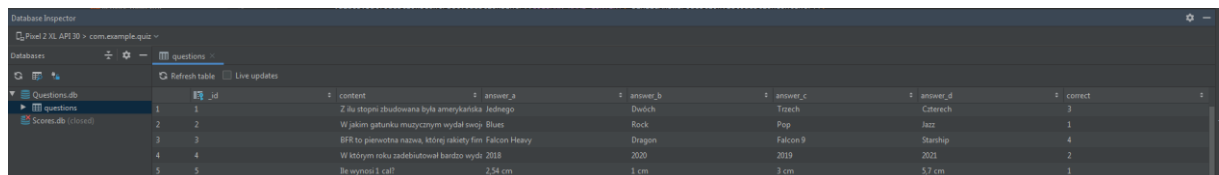
2. Następnie zapisuję poszczególne pytania do mojej bazy danych za pomocą metody **addQuestionToDatabase()**:

```
169      questionsDbHelper.addQuestionToDatabase(questionsDbHelper, question1);
170      questionsDbHelper.addQuestionToDatabase(questionsDbHelper, question2);
171      questionsDbHelper.addQuestionToDatabase(questionsDbHelper, question3);
172      questionsDbHelper.addQuestionToDatabase(questionsDbHelper, question4);
173      questionsDbHelper.addQuestionToDatabase(questionsDbHelper, question5);
```

3. Pobieram pytania z bazy danych, które zwrócone zostają w postaci listy obiektów **SingleAnswerQuestion**:

```
75      singleAnswerQuestions = questionsDbHelper.readQuestionsFromDatabase(questionsDbHelper);
```

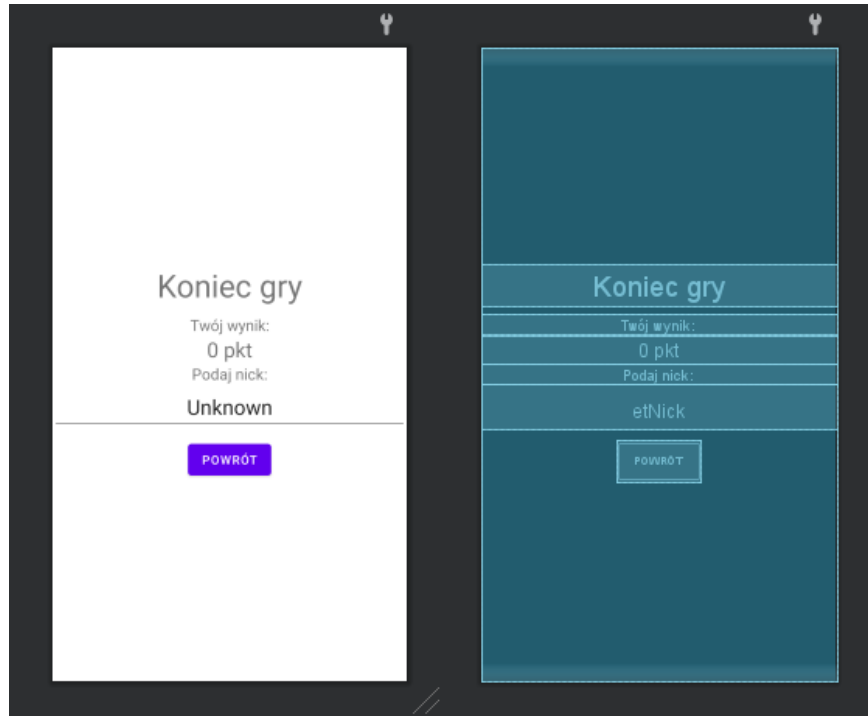
4. Sprawdzam czy wdrożona przeze mnie baza danych działa prawidłowo:



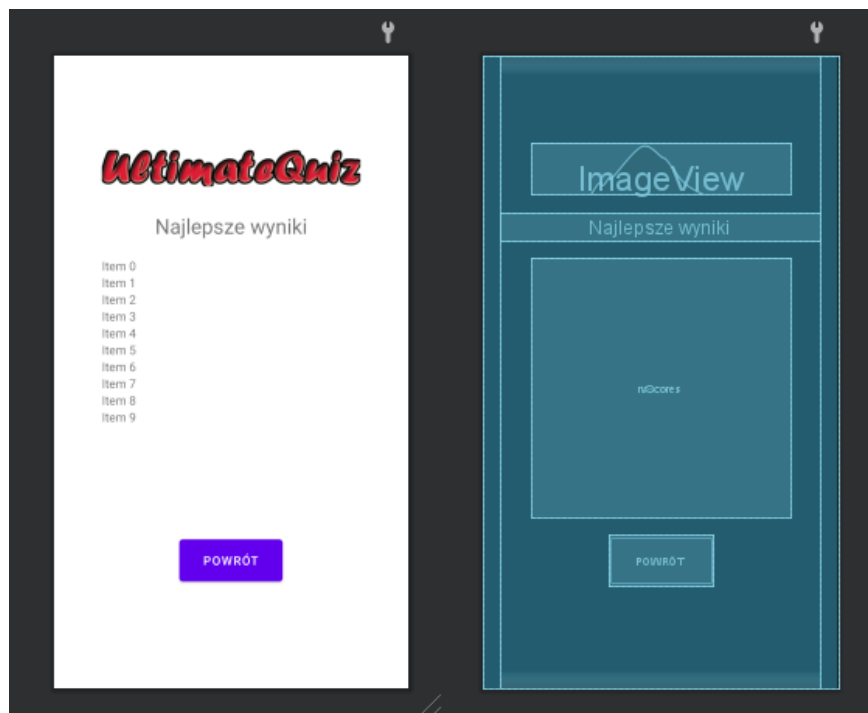
	id	content	answer_a	answer_b	answer_c	answer_d	correct
1	1	Z jakiego stopnia studiowana była amerykańska literatura?	French	Spanish	Italian	German	3
2	2	W jakim gatunku muzycznym występował zespół Blue?	Rock	Pop	Jazz	Classical	1
3	3	W którym roku powstała pierwsza seria filmowa z Falconem Heavy?	2018	2019	2020	2021	2
4	4	W którym roku zadebiutował zespół The Weeknd?	2015	2016	2017	2018	1
5	5	He weighs 1 cat?	2.54 cm	1 cm	3 cm	5.7 cm	1

Utworzenie rankingu najlepszych graczy za pomocą bazy danych SQLite

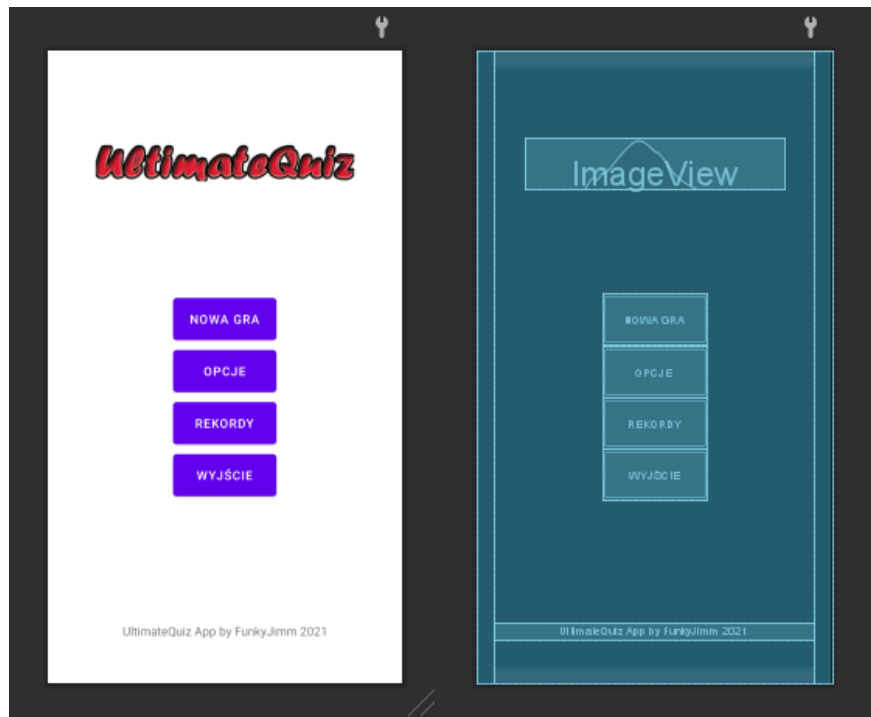
1. Na początku zmodyfikuję nieco wygląd mojej aplikacji. Pierwsze muszę dodać możliwość wpisania nazwy gracza na ekranie końcowym gry:



2. Następnie tworzę nową aktywność, która zawierać będzie ranking graczy z największą ilością zdobytych punktów. Tutaj wykorzystam komponent o nazwie **RecyclerView**, który będzie przechowywać rekordy pobrane z bazy danych:



3. Edytuję ekran główny gry wprowadzając do niego przycisk przenoszący użytkownika do ekranu z rankingiem graczy:



Utworzenie bazy danych Scores

1. Podobnie jak miało to miejsce w przypadku bazy danych **Questions**, tworzę nową klasę zawierającą nazwę tabeli i kolumn oraz zapytania tworzące i usuwające bazę danych:

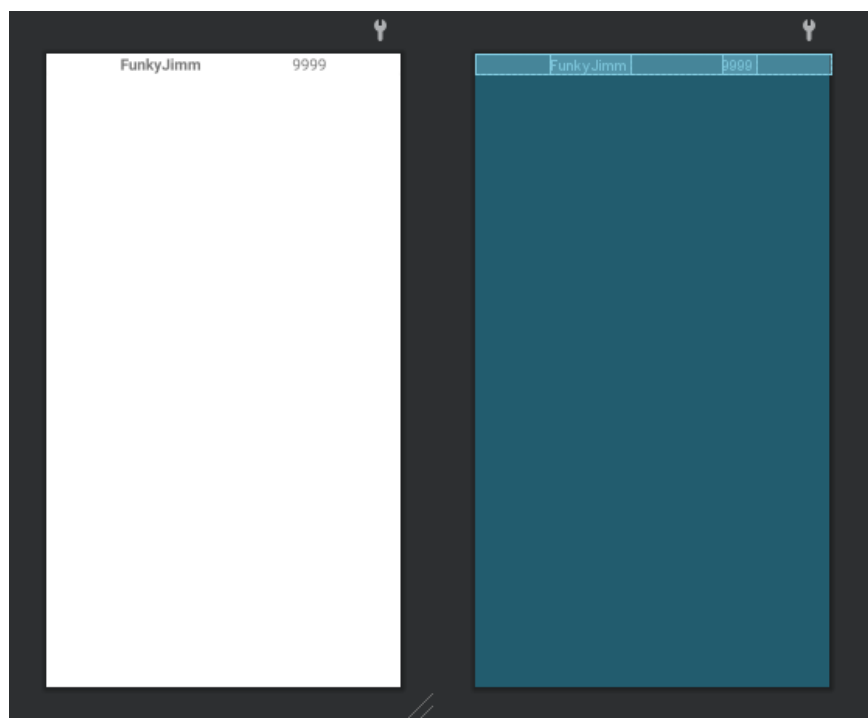
```
1 package com.example.quiz.database;
2
3 import android.provider.BaseColumns;
4
5 public final class ScoresContract {
6
7     private ScoresContract() {}
8
9     public static class ScoreEntry implements BaseColumns {
10         public static final String TABLE_NAME = "scores";
11         public static final String COLUMN_NAME_NICK = "nick";
12         public static final String COLUMN_NAME_SCORE = "score";
13     }
14
15     public static final String SQL_CREATE_SCORES =
16         "CREATE TABLE " + ScoreEntry.TABLE_NAME + " (" +
17             ScoresContract.ScoreEntry._ID + " INTEGER PRIMARY KEY," +
18             ScoresContract.ScoreEntry.COLUMN_NAME_NICK + " TEXT," +
19             ScoresContract.ScoreEntry.COLUMN_NAME_SCORE + " TEXT)";
20
21     public static final String SQL_DELETE_SCORES =
22         "DROP TABLE IF EXISTS " + ScoresContract.ScoreEntry.TABLE_NAME;
23 }
```

2. Następnie tworzę klasę rozszerzającą **SQLiteOpenHelper**, która zawiera informację o wersji, nazwę bazy danych oraz metodę **onCreate()** oraz metodę **onUpgrade()**:

```
1 package com.example.quiz.database;
2
3 import ...
4
11
12 public class ScoresDbHelper extends SQLiteOpenHelper {
13
14     public static final int DATABASE_VERSION = 1;
15     public static final String DATABASE_NAME = "Scores.db";
16
17     public ScoresDbHelper(Context context) {
18         super(context, DATABASE_NAME, factory: null, DATABASE_VERSION);
19     }
20
21     @Override
22     public void onCreate(SQLiteDatabase db) { db.execSQL(SQL_CREATE_SCORES); }
23
24
25
26     @Override
27     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
28         db.execSQL(SQL_DELETE_SCORES);
29         onCreate(db);
30     }
31 }
32
```

RecyclerView oraz wyświetlenie rankingu na ekranie gry

1. Tworzę nowy plik XML, który odpowiadać będzie jednemu wierszowi w **ScoresActivity**:



2. Utworzenie klasy **Score**, która zawierać będzie nick oraz liczbę zdobytych punktów gracza:

```
1 package com.example.quiz.scores;
2
3 public class Score {
4
5     private String nick;
6     private String score;
7
8     public Score(String nick, String score) {
9         this.nick = nick;
10        this.score = score;
11    }
12
13    public String getNick() { return nick; }
16
17    public String getScore() { return score; }
20
21 }
```

3. Następnie deklaruję RecyclerView oraz utworzoną przeze mnie klasę **ScoresDbHelper** wraz z listą obiektów **Score**, która będzie przechowywać dane pobrane z bazy danych:

```
22 RecyclerView recyclerView;
23
24 ScoresDbHelper scoresDbHelper = new ScoresDbHelper( context: this);
25
26 ArrayList<Score> scores = new ArrayList<>();
```

4. Tworzę metodę wczytującą rekordy z tabeli oraz zapisującą je do utworzonej wcześniej listy. W tym przypadku użyję sortowania po ilości zdobytych przez gracza punktów:

```
51 public void readScoreData() {
52     SQLiteDatabase db = scoresDbHelper.getReadableDatabase();
53
54     Cursor cursor = db.query(
55         ScoresContract.ScoreEntry.TABLE_NAME,
56         columns: null,
57         selection: null,
58         selectionArgs: null,
59         groupBy: null,
60         having: null,
61         orderBy: ScoresContract.ScoreEntry.COLUMN_NAME_SCORE + " DESC"
62     );
63
64     while(cursor.moveToNext()) {
65         String playerNick = cursor.getString(
66             cursor.getColumnIndexOrThrow(ScoresContract.ScoreEntry.COLUMN_NAME_NICK));
67         String playerScore = cursor.getString(
68             cursor.getColumnIndexOrThrow(ScoresContract.ScoreEntry.COLUMN_NAME_SCORE));
69         Score score = new Score(playerNick, playerScore);
70         scores.add(score);
71     }
72     cursor.close();
73 }
74 }
75 }
```

5. Implementuję elementy interfejsu oraz wywołuję metodę pobierającą informację z bazy danych, których listę przekazuję je do obiektu klasy **MyAdapter**, na którego bazie będą generowane wyniki na ekranie użytkownika:

```
28 @Override
29 protected void onCreate(@Nullable Bundle savedInstanceState) {
30     super.onCreate(savedInstanceState);
31     setContentView(R.layout.activity_scores);
32
33     final Button btnReturn = (Button) findViewById(R.id.btnReturn);
34
35     recyclerView = findViewById(R.id.rvScores);
36
37     readScoreData();
38
39     MyAdapter myAdapter = new MyAdapter( ctx: this, scores);
40     recyclerView.setAdapter(myAdapter);
41     recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
42
43     btnReturn.setOnClickListener(new View.OnClickListener() {
44         @Override
45         public void onClick(View v) { finish(); }
46     });
47 }
48
49 }
```

Klasa MyAdapter

1. Tworzę klasę **MyAdapter**, która rozszerza klasę **RecyclerView.Adapter**. Deklaruję nowy obiekt typu **Context** oraz listę, do której będą przekazywane informacje o wynikach gracza za pomocą utworzonego konstruktora:

```
1 package com.example.quiz;
2
3 import ...
4
15
16 public class MyAdapter extends RecyclerView.Adapter<MyAdapter.MyViewHolder> {
17
18     private Context context;
19
20     private ArrayList<Score> playersScores = new ArrayList<>();
21
22     public MyAdapter(Context ctx, ArrayList<Score> scores) {
23         context = ctx;
24         playersScores = scores;
25     }
26 }
```

2. Następnie tworzę nowy **LayoutInflater**, który będzie generować nowe wiersze na podstawie utworzonego przeze mnie schematu:

```
27 @NonNull
28 @Override
29 public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
30     LayoutInflater inflater = LayoutInflater.from(context);
31     View view = inflater.inflate(R.layout.score_row, parent, attachToRoot: false);
32     return new MyViewHolder(view);
33 }
```

3. Modyfikuję metodę **getItemCount()** w taki sposób aby wyświetlała maksymalnie 10 wierszy na ekranie użytkownika:

```
41 @Override
42 public int getItemCount() {
43     int count;
44
45     if (playersScores.size() > 10)
46         count = 10;
47     else
48         count = playersScores.size();
49
50     return count;
51 }
```

4. Inicjalizacja pól tekstowych:

```
53 public class MyViewHolder extends RecyclerView.ViewHolder {
54
55     TextView playerNick, playerScore;
56
57     public MyViewHolder(@NonNull View itemView) {
58         super(itemView);
59
60         playerNick = itemView.findViewById(R.id.tvPlayerNick);
61         playerScore = itemView.findViewById(R.id.tvPlayerScore);
62     }
63 }
```

Zapisanie wyniku gracza w bazie danych

1. W klasie **ResultActivity** pobieram nazwę gracza z utworzonego przeze mnie pola edycyjnego. Następnie tworzę nowy rekord w utworzonej przeze mnie bazie danych **Scores** na podstawie zmiennych **playerNick** oraz **playerScore**:

```
1 package com.example.quiz;
2
3 import ...
4
16
17 public class ResultActivity extends AppCompatActivity {
18
19     private String playerScore;
20
21     ScoresDbHelper scoresDbHelper = new ScoresDbHelper( context, this);
22
23     @Override
24     protected void onCreate(@Nullable Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_result);
27
28         final TextView tvScore = (TextView) findViewById(R.id.tvScore);
29         final Button btnReturn = (Button) findViewById(R.id.btnReturn);
30         final EditText etNick = (EditText) findViewById(R.id.etNick);
31
32         Bundle extras = getIntent().getExtras();
33         playerScore = Integer.toString(extras.getInt( key: "playerScore"));
34         tvScore.setText(playerScore);
35
36         btnReturn.setOnClickListener(new View.OnClickListener() {
37             @Override
38             public void onClick(View v) {
39                 String playerNick = etNick.getText().toString();
40
41                 ContentValues values = new ContentValues();
42                 values.put(ScoresContract.ScoreEntry.COLUMN_NAME_NICK, playerNick);
43                 values.put(ScoresContract.ScoreEntry.COLUMN_NAME_SCORE, playerScore);
44
45                 scoresDbHelper.getWritableDatabase().insert(ScoresContract.ScoreEntry.TABLE_NAME, nullColumnHack: null, values);
46
47                 Intent intent = new Intent(getApplicationContext(), MainActivity.class);
48                 startActivity(intent);
49             }
50         });
51     }
52 }
```

Wygląd działania aplikacji

