

**Технически университет - София**



***Направление - ФКСТ***

**ПРОГРАМИРАНЕ ЗА МОБИЛНИ УСТРОЙСТВА**

**ДОКУМЕНТАЦИЯ НА КУРСОВ ПРОЕКТ**

**Студент:** *Денислав Петков, 29 група*

**Специалност:** *Компютърно и софтуерно инженерство*

**Факултетен номер – 121219031**

**Ръководител:** *доц. д-р Антония Ташева*

## Съдържание

Съдържание .....	2
Увод .....	3
Анализ на съществуващи разработки .....	4
Проектиране.....	6
Реализация .....	10
Потребителско ръководство .....	20
Заключение .....	24
Литература .....	25
Приложение .....	26

## Увод

Приложението представлява литературен куиз игра за Андроид на тема българските произведения и техните автори. Потребителският интерфейс на приложението е много опростен и лесен за работа.

При започването на играта потребителят може да избира от фиксирани стойности на колко въпроса иска да отговори(5, 10 или 15). Също така има и възможността за запазване на текущата сесия и продължаването ѝ по-късно, както и преглеждането на вече отговорените въпроси и техните отговори. Потребителят има и опцията да сподели резултатите си в социалната мрежа(twitter) след отговарянето на последния въпрос от играта.

## Анализ на съществуващи разработки

Пример за вече съществуващо подобно приложение е „Матура БЕЛ – Материали и Тестове“(Фигура 1).

Фигура 1



Плюсове:

- Разполага с избор между български език или литература. И предоставя съответно допълнителни материали и тестове по тези теми. Също така са достъпни и анализи на творбите и анализи на техните автори.

Минуси:

- Потребителският интерфейс на моменти е много объркващ, липсва възможността да се следи напредъкът спрямо предишните тестове, липсва възможност да се избира броят на въпросите в теста и също така не се предоставя опция да се запамети прогреса по време на теста и да се довърши по-късно.

Друго подобно приложение е „БЕЛ Матура – Автори и Творби“(Фигура 2).

Фигура 2



Плюсове:

- Предлага избор от автори и предоставя детайлна информация за автора и произведенията му. Много прост и лесен за работа потребителски интерфейс.

Минуси:

- Приложението е много просто и липсва начин да провериш знанията си, било то под формата на въпроси, куизове или други.

## Проектиране

Приложението ще се състои от 3 екрана:

1. Начален екран, който има 2 таба:

- Таб с основните функционалности като започване на нова игра, продължаване на старата и изтриването на историята
- Таб за история от отговорени въпроси.

2. Екран, на който се случва самата игра

3. Екран с резултатите от играта

Приложението ще поддържа завъртане на екрана на телефона, тоест „Portrait” и „Landscape” ориентация на екрана.

При стартиране на приложението потребителят ще има само възможността да стартира нова игра. Преди започването на играта той трябва да избере от колко въпроса да се състои неговата игра. По време на играта потребителят ще има възможността да запази своята сесия и да се завърне в основното меню. От основното меню може да започне нова игра или да продължи старата. Той също така има и опцията да провери историята на отговорените въпроси и техните отговори и, ако реши да я изтрие. След отговарянето на последния въпрос от играта, потребителят ще може да види всички въпроси, заедно с

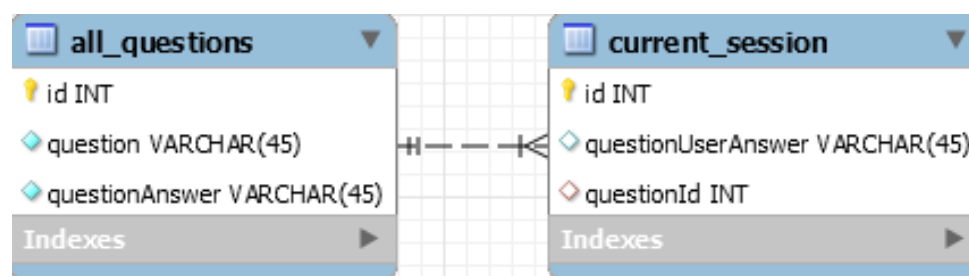
техните отговори и отговорите, които той е дал и ще има опцията да сподели своя резултат в твитър.

Входните данни на приложението ще са джешчъри (button click, swipe) и отговорите на въпросите, които са във формата на символен низ.

Отговорите на въпросите ще се пазят в паметта на програмата, докато потребителят не отговори на последния въпрос или не реши да запамети сесията и да се върне в менюто, тогава те се запазват в базата данни.

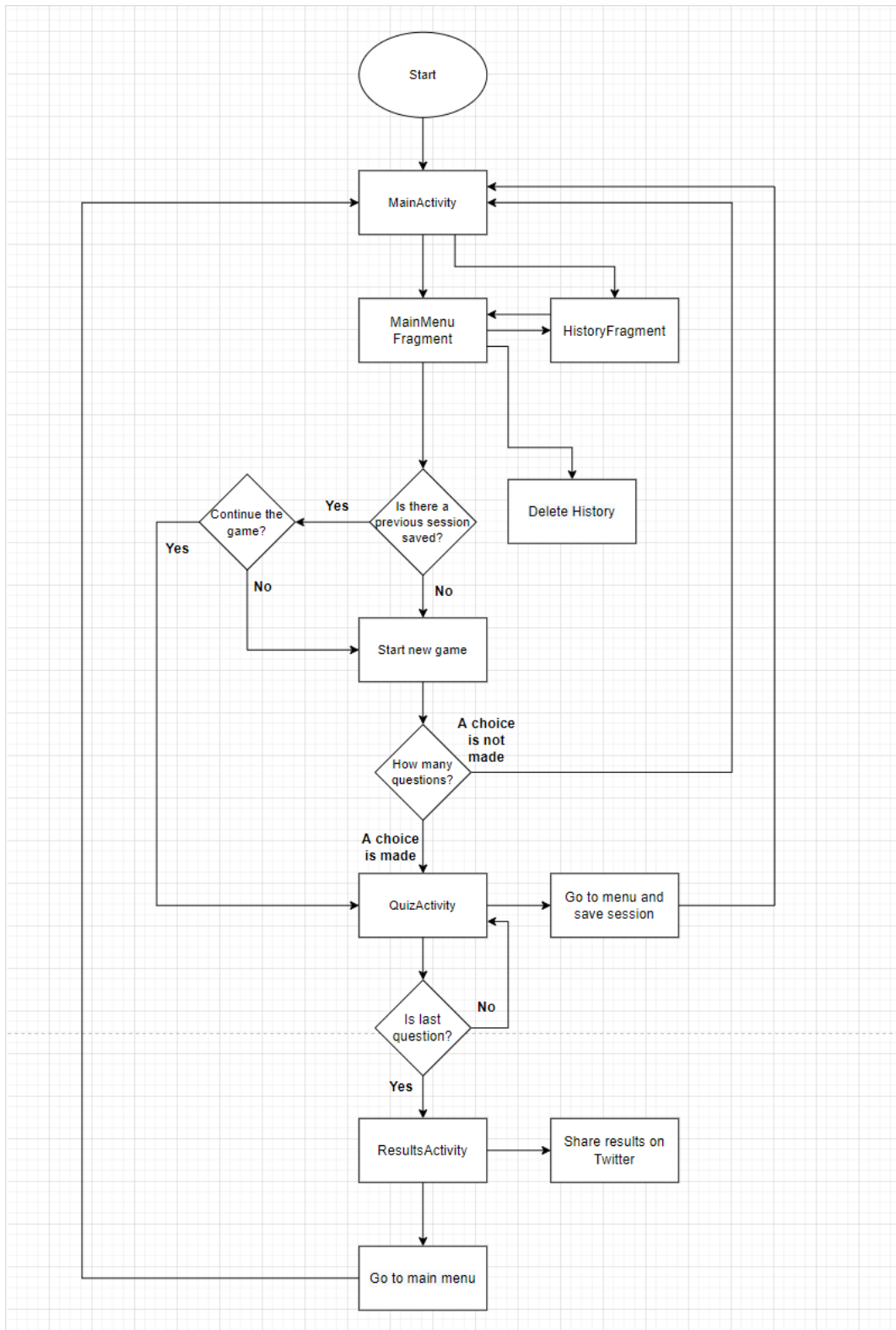
Данните за въпросите и техните отговори, както и текущата/последната игра на потребителя ще се пазят в база от данни в 2 таблици.

ER диаграма(Фигура 3):



Фигура 3

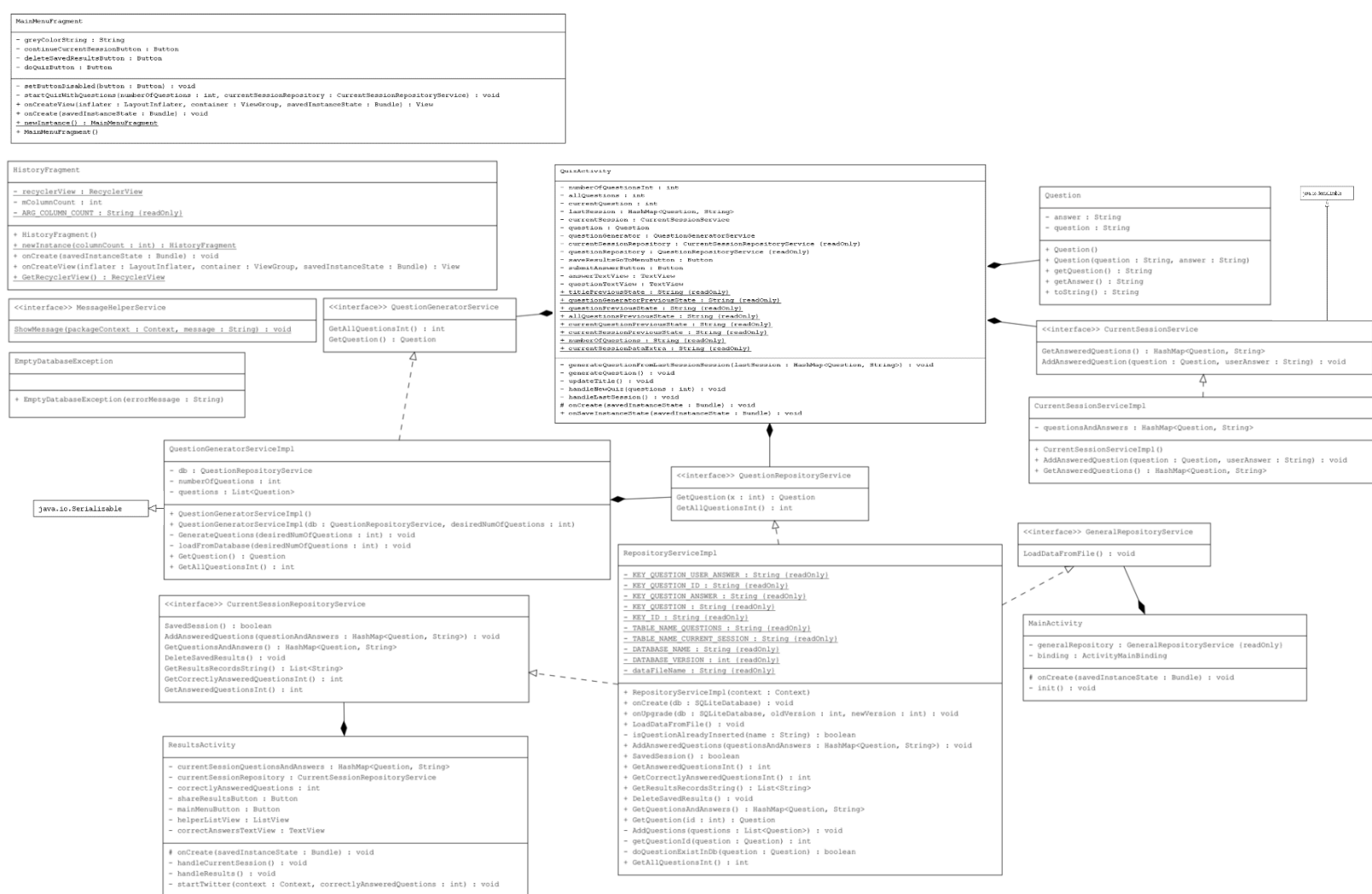
## Workflow диаграма(Фигура 4):



Фигура 4



## Клас диаграма(Фигура 5):



Фигура 5

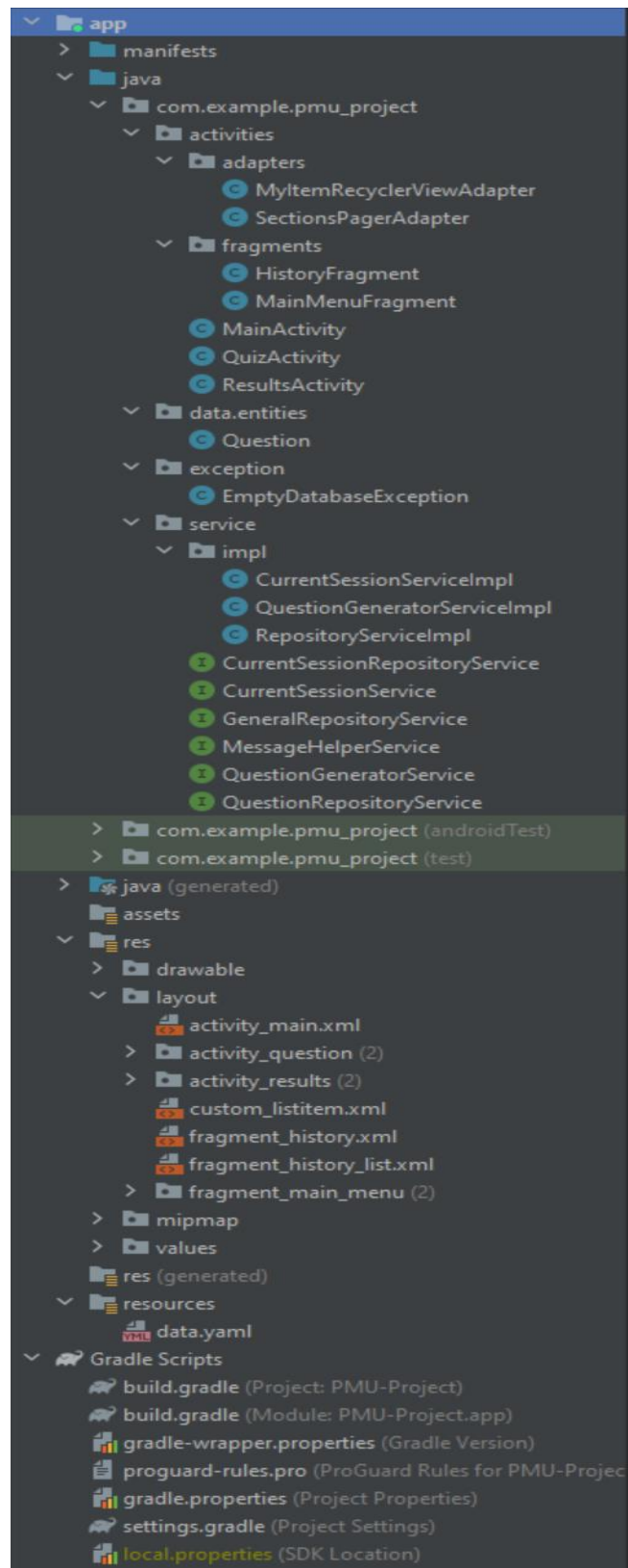
## Реализация

За всеки един клас, който предлага функционалност е изграден интерфейс, който да се използва и да предостави ниво на абстрактност.

Следвана е „Model-View-Controller“(MVC) архитектурата:

- Model - Question обекта
- Controllers – MainActivity, QuizActivity и ResultsActivity
- Views – activity\_main.xml, activity\_question.xml, activity\_results.xml, fragment\_main\_menu.xml и fragment\_history.xml

## Структурата на проекта(Фигура 6):



Фигура 6

Началният екран е конструиран от „Tabbed activity“ на име „MainActivity“ с 2 fragment-а – „MainMenuFragment“ и „HistoryFragment“. При зареждането на „MainActivity“ се четат въпросите и отговорите от „data.yaml“ файл и се добавят в базата данни. Четено на „.yaml“ файлове става с помощта на „jackson“ библиотеката.

```
public void LoadDataFromFile() {
    ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
    InputStream is = classLoader.getResourceAsStream(dataFileName);
    ObjectMapper om = new ObjectMapper(new YAMLFactory());

    try {
        List<Question> questions = om.readValue(is, new
TypeReference<List<Question>>() {});
        AddQuestions(questions);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Във фрагмента на име „MainMenuFragment“ са 3-те бутона с основната функционалност(продължаване на играта, стартиране на нова игра, изтриване на историята от отговорени въпроси).

Когато няма запазена игра, бутонът за продължаване на такава не е активен, както и този за изтриване на историята. Това е реализирано чрез прости проверки и вградените методи за работа с бутоните.

```
button.setBackgroundColor(Color.parseColor(greyColorString));
button.setEnabled(false);
```

С помощта на „AlertDialog.Builder“ е направена опцията за избирането на броя на въпросите.

```
doQuizButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        String[] numberOfQuestions = {"5", "10", "15"};

        AlertDialog.Builder builder = new
AlertDialog.Builder(MainMenuFragment.this.getContext());

        builder.setTitle("Избери броя на въпросите!").setCancelable(true);
        builder.setItems(numberOfQuestions, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int which) {
                switch (which){
                    case 0:
                        startQuizWithQuestions(5, currentSessionRepository);
                        return;
                    case 1:
                        startQuizWithQuestions(10, currentSessionRepository);
                        return;
                    case 2:
                        startQuizWithQuestions(15, currentSessionRepository);
                        return;
                }
            }
        });

        builder.create().show();
        return;
    }
});
```

При стартирането на следващият екран се подават броят на въпросите като параметър.

```
MainMenuFragment.this.startActivity(new
Intent(MainMenuFragment.this.getContext(),
QuizActivity.class).putExtra(QuizActivity.numberOfQuestionsExtra,
numberOfQuestions));
```

При натискането на бутона за продължаване на старата игра се стартира следващият екран с допълнителен параметър, който е hashmap от „Question“ обекта, заедно с дадените от потребителя отговори, които се зареждат от базата.

```
MainMenuFragment.this.startActivity(new  
Intent(MainMenuFragment.this.getContext(),  
QuizActivity.class).putExtra(QuizActivity.currentSessionDataExtra,  
        (Serializable) currentSessionRepository.GetQuestionsAndAnswers()));
```

„Question“ обектът съдържа в себе си както въпросът, така и неговият отговор.

```
public class Question implements Serializable {  
    private String question;  
    private String answer;
```

При фрагмента с историята има „recyclerview“ компонент, който представлява списък с отговорените въпроси, който се пълни от базата с данни. Попълването на списъка става чрез „ItemRecyclerViewAdapter“.

```
List<String> previousResults =  
currentSessionRepository.GetResultsRecordsString();  
recyclerView.setAdapter(new MyItemRecyclerViewAdapter(previousResults));
```

Екранът, на който се случва играта е част от „QuizActivity“. Генерирането на въпросите става там. Съдържа 2 бутона – за предаване на отговора и връщане в менюто, поле за въвеждане на отговора и текстов изглед, който показва въпросът.

Като заглавие на активитито е номерът на текущият въпрос и общия брой на въпросите във вида „Въпрос X/Y“. Това става чрез вече съществуващият „setTitle(CharSequence title)“ метод и броячи, които записват текущият въпрос и общия брой на въпросите.

```
setTitle("Въпрос " + currentQuestion + "/" + allQuestions
```

За да не се създаде ново активити при обръщането на екрана се използва „onSaveInstanceState(Bundle savedInstanceState)“ методът, вътре в който се записват всички необходими променливи, за възобновяване на сесията.

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    super.onSaveInstanceState(savedInstanceState);
    savedInstanceState.putInt(allQuestionsPreviousState, allQuestions);
    savedInstanceState.putInt(currentQuestionPreviousState, currentQuestion);
    savedInstanceState.putSerializable(currentSessionPreviousState,
    (Serializable) currentSession);
    savedInstanceState.putSerializable(questionPreviousState, question);
    try {
        if (currentQuestion != allQuestions &&
currentSessionRepository.GetQuestionsAndAnswers().size() != allQuestions){
            savedInstanceState.putSerializable(questionGeneratorPreviousState,
    (Serializable) questionGenerator);
        }
    } catch (EmptyDatabaseException e) {
        // it should not happen, so just print the stack trace
        e.printStackTrace();
    }
}
```

При стартирането на нова игра чрез „intent.getSerializableExtra(String name)“ методът се взимат броя на въпросите от предишното активити.

```
numberOfQuestionsInt = (Integer)
intent.getSerializableExtra(QuizActivity.numberOfQuestionsExtra);
```

По същият начин, ако се продължава старата се взимат въпросите от последната сесия.

```
lastSession = (HashMap<Question, String>)  
intent.getSerializableExtra(QuizActivity.currentSessionDataExtra);
```

Въведеният отговор трябва да съдържа двете имена на автора разделени с празно място. Отговорът се обработва преди да бъде добавен като се капитализират двете имена и се премахва празното място преди и след тях. Това е постигнато с добавянето на „WordUtils“ библиотеката и „capitalize(final String str)“ метода и вече съществуващият „trim()“ метод.

```
currentSession.AddAnsweredQuestion(question,  
WordUtils.capitalize(answerTextView.getText().toString().trim()));
```

Отвеждането на следващия екран става след отговарянето на последния въпрос, като то се осъществява чрез проста проверка. Като допълнителен параметър се подава hashmap от въпросите и потребителските отговори.

```
if (currentQuestion == allQuestions) {  
    QuizActivity.this.startActivity(new Intent(QuizActivity.this,  
ResultsActivity.class).putExtra(currentSessionDataExtra,  
currentSession.GetAnsweredQuestions()));  
    return;  
}
```



При натискането на бутона за връщане в менюто останалите неотговорени въпроси се записват с отговор „null“, по този начин при продължаване на играта от базата данни се взимат само те.

```
currentSession.AddAnsweredQuestion(question, null);  
for (int i = currentQuestion; i < allQuestions; i++) {  
    currentSession.AddAnsweredQuestion(questionGenerator.GetQuestion(), null);  
}
```

Екранът, който се отваря след отговарянето на всички въпроси е част от активити на име „ResultsActivity“. В него става записването на резултатите в базата данни и представянето им пред потребителя. Представянето на резултатите става в „ListView“ компонент. Като допълнителни компоненти има 2 бутона – за връщане в менюто и споделяне на резултата и текстов изглед, който обобщава колко правилни отговора от колко въпроса е успял потребителят да даде.

Показването на резултатите в „ListView“ компонента става чрез „ArrayAdapter“.

```
ArrayAdapter adapter = new ArrayAdapter<String>(ResultsActivity.this,  
    android.R.layout.simple_list_item_1,  
    currentSessionRepository.GetResultsRecordsString());  
helperListView.setAdapter(adapter);
```

При опит за споделяне на резултатите се отваря приложението „twitter“ или „twitter lite“, ако го има инсталирано, ако не се отваря браузърът. Като отварянето става с автоматично генерирано съобщение готово за споделяне.

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(message));
try {
    context.getPackageManager().getPackageInfo("com.twitter.android", 0);
} catch (PackageManager.NameNotFoundException e) {
    try {
        context.getPackageManager().getPackageInfo("com.twitter.android.lite",
0);
    } catch (PackageManager.NameNotFoundException e1) {
        // do nothing and open a browser
    }
}
context.startActivity(intent);
```

Базата данни е реализирана в един клас, но с 3 различни интерфейса. И използва „EmptyDatabaseException“, който е добавен за съобщаване при празна или непълна база данни.

„CurrentSessionRepositoryService“ интерфейсът е за работа с текущата/последната запаметена сесия.

```
public interface CurrentSessionRepositoryService {
    public int GetAnsweredQuestionsInt();
    public int GetCorrectlyAnsweredQuestionsInt();
    public List<String> GetResultsRecordsString() throws
EmptyDatabaseException;
    public void DeleteSavedResults();
    public HashMap<Question, String> GetQuestionsAndAnswers() throws
EmptyDatabaseException;
    public void AddAnsweredQuestions(HashMap<Question, String>
questionAndAnswers);
    public boolean SavedSession();
}
```

„GeneralRepositoryService“ интерфейсът е за основните функционалности като зареждане на данните от файл.

```
public interface GeneralRepositoryService {  
    public void LoadDataFromFile ();  
}
```

„QuestionRepositoryService“ интерфейсът е за работа със самите въпроси.

```
public interface QuestionRepositoryService {  
    public int GetAllQuestionsInt();  
    public Question GetQuestion(int x) throws EmptyDatabaseException;  
}
```

Също така е имплементиран и допълнителен интерфейс на име „MessageHelperService“, който се използва при съобщаване на събития.

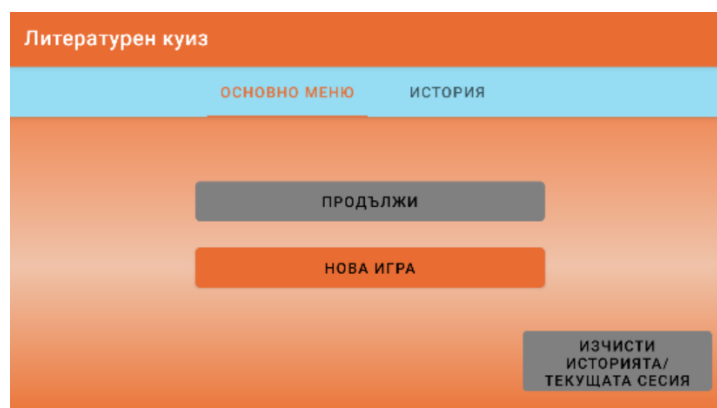
```
public interface MessageHelperService {  
    public static void ShowMessage(Context packageContext, String message) {  
        Toast.makeText(packageContext,  
            message,  
            Toast.LENGTH_LONG)  
        .show();  
    }  
}
```

## Потребителско ръководство

При стартиране на приложението се появява основното меню. При отсъствие на вече започната игра, потребителят има единствено опцията да започне нова игра(Фигура 7 и 8).

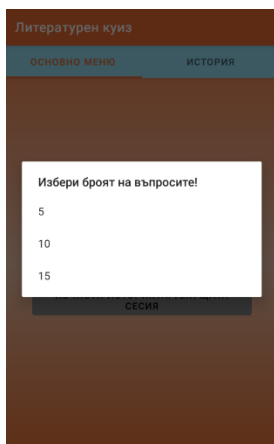


Фигура 7

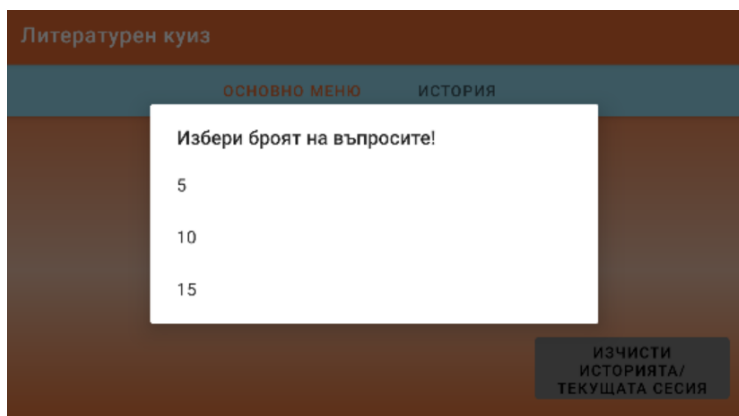


Фигура 8

При натискането на бутона за започване на нова игра, играчът трябва да избере от колко въпроса да се състои неговата игра(Фигура 9 и 10).

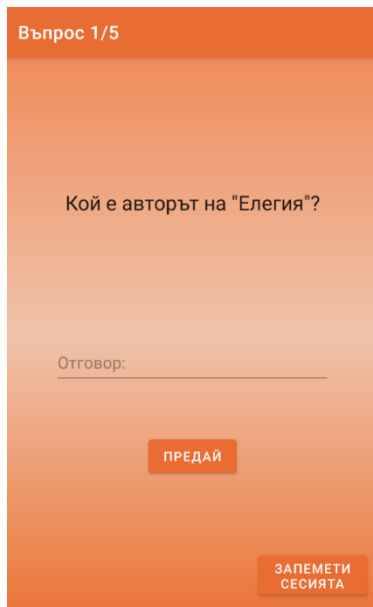


Фигура 9

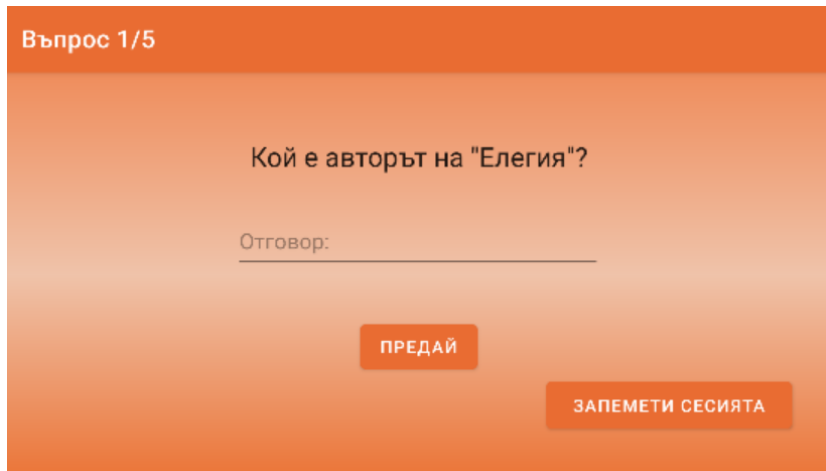


Фигура 10

След избирането на броят на въпросите се отваря прозорецът, на който може да се види въпросът и да се даде отговорът(Фигура 11 и 12).



Фигура 11

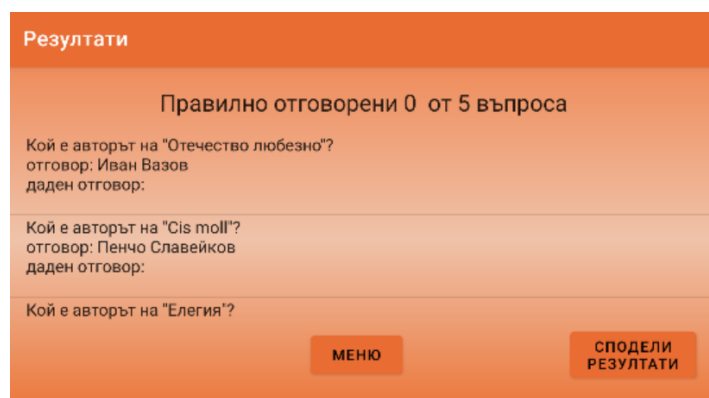


Фигура 12

След отговарянето на последния въпрос се отварят резултатите. Там потребителят има и възможност да сподели своят резултат(Фигура 13 и 14).

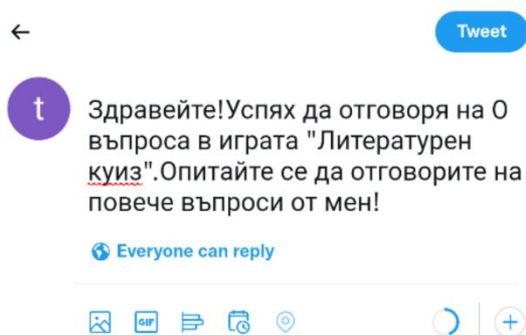


Фигура 13



Фигура 14

При натискане на бутона за споделяне на резултата се отваря twitter приложението, ако го има инсталирано. Ако го няма – се отваря браузерът(Фигура 15).



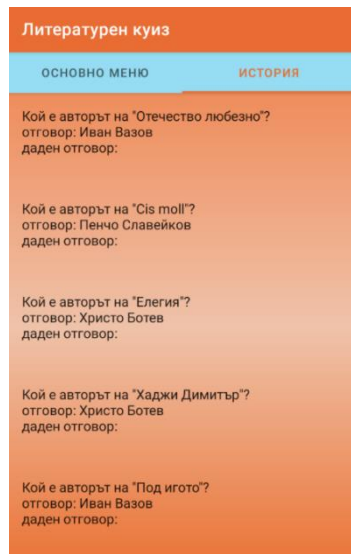
Фигура 15

При натискане на бутона за връщане към менюто се отваря менюто, където потребителят вече има опцията да изтрие своята история(Фигура 16).



Фигура 16

При отваряне на историята, той може да види вече отговорените въпроси от последната игра(Фигура 17 и 18).



Фигура 17



Фигура 18

При натискане на бутона за запамятаване на сесията по време на игра, потребителят се връща в менюто, откъдето има опцията да продължи вече започналата игра(Фигура 19).



Фигура 19

## Заклучение

Продуктът представлява просто приложение, с което човек може да си провери знанията си по тема българските литературни творби и техните автори.

За да бъде конкурентноспособно следва да се имплементират следните промени:

1. Да има централизирана база данни, от която да се взимат данни за въпросите и отговорите. В момента това се случва от „data.yaml“ файл.
2. Да се подобри графичният интерфейс на приложението, понеже в момента е много базов.
3. Да се добави нова функционалност, която да дава възможност на потребителя да сравнява резултатите си с тези на своите приятели от някоя социална мрежа.



## Литература

Използвани източници на информация:

1. [stackoverflow.com](https://stackoverflow.com/questions/11175966/how-to-prevent-the-layout-from-getting-reset-when-the-screen-orientation-changes) - [How to prevent the layout from getting reset when the screen orientation changes?, [stackoverflow.com/questions/11175966/how-to-prevent-the-layout-from-getting-reset-when-the-screen-orientation-changes](https://stackoverflow.com/questions/11175966/how-to-prevent-the-layout-from-getting-reset-when-the-screen-orientation-changes)]
2. [developer.android.com](https://developer.android.com/topic/libraries/architecture/saving-states) – [Save UI states, [developer.android.com/topic/libraries/architecture/saving-states](https://developer.android.com/topic/libraries/architecture/saving-states)]

## Приложение

Проектът може да бъде намерен в github на следния линк – [github.com/denislavPetkov/QuizGame](https://github.com/denislavPetkov/QuizGame).