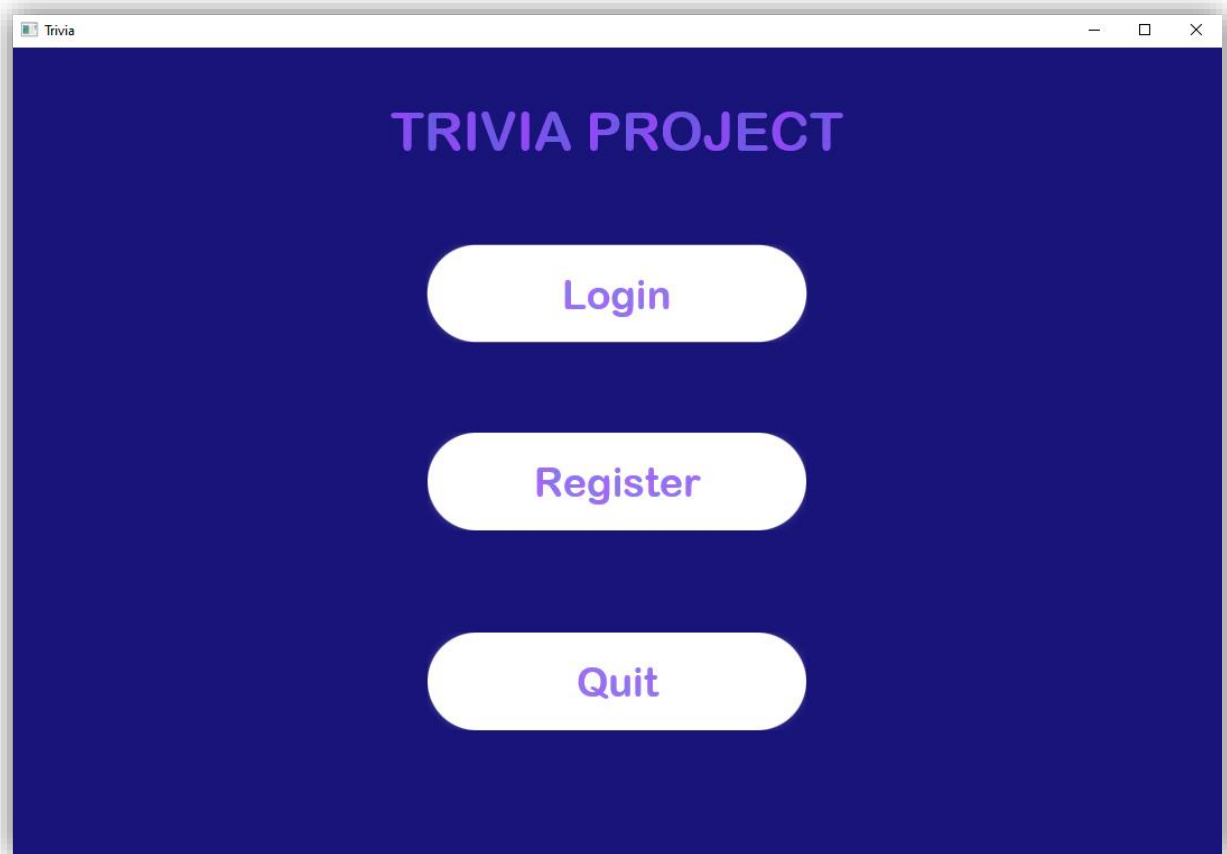


Приложение за достъп до базата от данни

Домашно 7

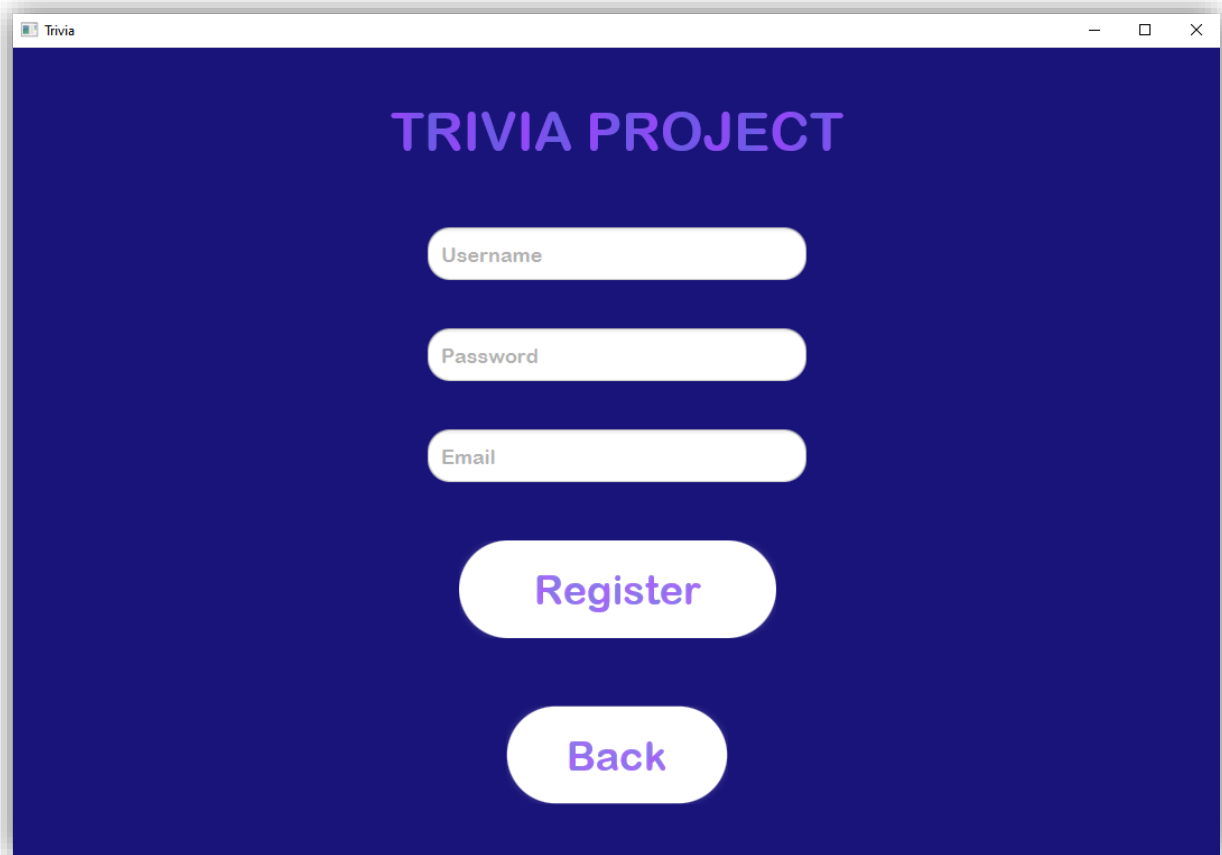
Ивайло Стоянов Стоянов № 71975



Създадох java приложение с интерфейс за потребители, което ще ни позволи достъп до данните от базата както и ще притежава функционалности като:

- Регистрация на потребител. (записване на нов потребител в базата)
- Вход на потребител и запазване на точките, които е изкарал до момента от изиграни игри.
- Отговаряне на 10 случайно избрани въпроса от базата с въпроси.
- Жокери по време на игра.
- Записване на история за всеки потребител в коя игра на кои въпроси е отговорил и как е отговорил на тях.

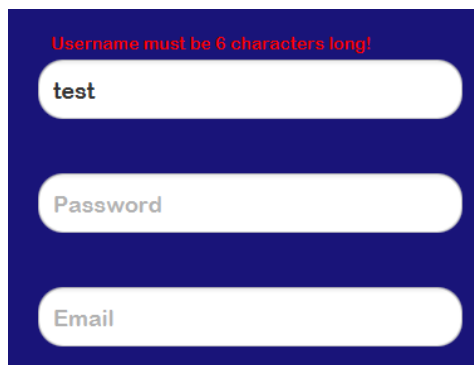
1. Регистрация



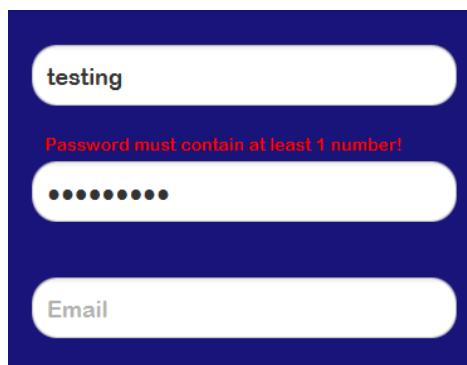
The screenshot shows a web browser window titled "Trivia". The page has a dark blue background with the text "TRIVIA PROJECT" in light blue at the top. Below the title, there are three white input fields labeled "Username", "Password", and "Email". Under these fields are two white buttons with blue text: "Register" and "Back".

Когато потребителят натисне бутончето Register, от главното меню е пратен на тази част от приложението, където може да напише потребителско име, парола и имейл. Потребителското име трябва да е уникално (да не е базата) и поне 6 букви, паролата трябва да е поне 6 букви и да притежава поне 1 цифра, а имейла трябва да е валиден и да няма друг такъв в базата.

Резултати от грешна входна информация:



This screenshot shows the registration form with the "Username" field containing the text "test". A red error message "Username must be 6 characters long!" is displayed above the field. The "Password" and "Email" fields are empty.



This screenshot shows the registration form with the "Username" field containing "testing" and the "Password" field filled with dots. A red error message "Password must contain at least 1 number!" is displayed above the password field. The "Email" field is empty.



This screenshot shows the registration form with the "Username" field containing "nasku123", the "Password" field filled with dots, and the "Email" field containing "invalidMeil". A red error message "User with that username exists!" is displayed above the username field.

Всички проверки свързани с базата се правят в клас TriviaDataManager, който съдържа в себе си DB2Manager, с който осъществяване връзката до базата от данни.

```
public class DB2Manager {  
  
    private Connection connection;  
    private Statement statement;  
    private ResultSet resultSet;  
  
    public void openConnection(){...}  
  
    public void closeConnection(){...}  
  
    public List<String> getListOfResults(String stmtnt, int row, int column) {...}  
  
    public String getSelectedColumnValue(String stmtnt) {...}  
  
    public boolean isExistingValueInTheDatabase(String selectStatement) {...}  
  
    public void executeUpdate(String stmtnt) {...}  
}
```

```
public final class TriviaDataManager {  
    private final DB2Manager db2M;  
  
    private final static TriviaDataManager INSTANCE = new TriviaDataManager();  
  
    public static TriviaDataManager getInstance() { return INSTANCE; }  
  
    public TriviaDataManager() { db2M = new DB2Manager(); }
```

За TriviaDataManager съм направил Singleton design pattern, което ще ми позволи без инстанция на такъв обект да ползвам методите му.

При всяка регистрация на потребител му генерирам 5 цифрен код, след това проверявам в базата дали такъв код вече съществува, ако има, генерирам нов (докато не е уникален). Също правя проверка дали потребителското име и имейл са уникални.

```
@FXML  
void btnRegisterAction(ActionEvent event) throws IOException {  
    setLabelsToInvisible();  
    if(isSuccessfulRegistration()) {  
        TriviaDataManager.getInstance().registerUser(txtEmail.getText(), txtUsername.getText(), txtPassword.getText());  
        alertSuccessfulRegistration();  
        goToMainScene();  
    }  
}  
  
private boolean isSuccessfulRegistration() {  
    return isValidUsername() && isValidPassword() && isValidEmail();  
}
```

След натискане на бутона за регистрация се правят проверки за валидни данни, за момента ни интересуват тези, които проверят нещо в базата.

```
public boolean isExistingEmail(String email) {
    db2M.openConnection();
    String selectStatement = String.format("SELECT 1 FROM FN71975.USERS WHERE EMAIL = '%s'", email);
    boolean check = db2M.isExistingValueInTheDatabase(selectStatement);
    db2M.closeConnection();
    return check;
}

public boolean isExistingUsername(String username) {
    db2M.openConnection();
    String selectStatement = String.format("SELECT 1 FROM FN71975.USERS WHERE USERNAME = '%s'", username);
    boolean check = db2M.isExistingValueInTheDatabase(selectStatement);
    db2M.closeConnection();
    return check;
}
```

```
public boolean isExistingValueInTheDatabase(String selectStatement) {
    try {
        resultSet = statement.executeQuery(selectStatement);
        return resultSet.next();
    }
    catch (SQLException s) {
        s.printStackTrace();
    }
    return false;
}
```

Ако не се намери потребител с такова име се с заявката ще получим празна таблица и тогава resultSet.next() ще бъде false.

След нужната валидация се изпълнява метода за регистрация.

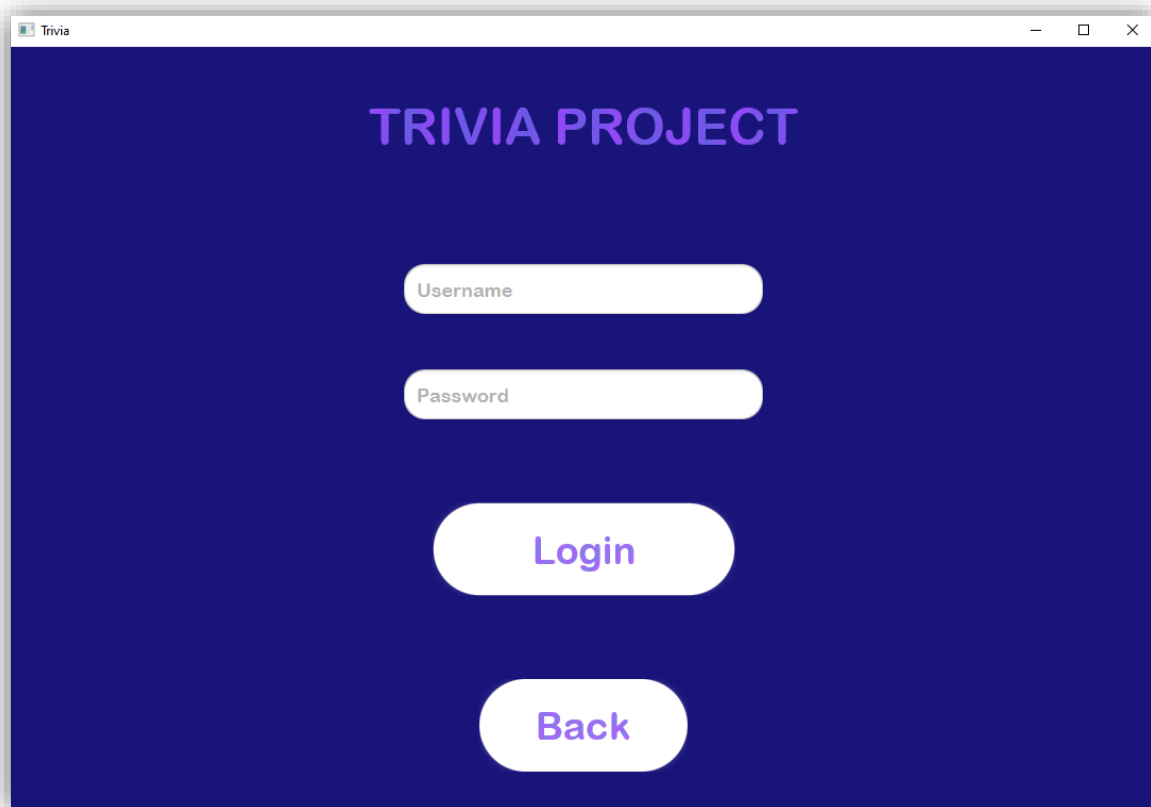
```
public void registerUser(String email, String userName, String password) {
    db2M.openConnection();
    String userID;
    String selectStatement;
    do {
        selectStatement = "SELECT 1 FROM FN71975.USERS WHERE ID = ";
        userID = getRandomFiveDigitID();
        selectStatement += userID;
    } while(db2M.isExistingValueInTheDatabase(selectStatement));

    String insertStatement = " INSERT INTO FN71975.USERS VALUES ('" + userID + "','" + "DEFAULT,'" +
        insertStatement += email + "','" +
        insertStatement += userName + "','" +
        insertStatement += password + "')";

    db2M.executeUpdate(insertStatement);
    db2M.closeConnection();
}
```

Default стойността при INSERT заявката е дефолтната стойност за точките на потребителя – 0.

2. Влизане в профил.



При опит за влизане в профил се прави проверка дали такъв потребител съществува в базата (като при регистрацията) и дали на този потребител съответства написаната парола.

```
@FXML
void btnLoginAction(ActionEvent event) throws IOException {
    setLabelsToInvisible();
    if(isLoginSuccessful()) {
        String username = txtUsername.getText();
        User user = new User(TriviaDataManager.getInstance().getUserId(username),
                             TriviaDataManager.getInstance().getUserPoints(username),
                             username);

        UserHolder userHolder = UserHolder.getInstance();
        userHolder.setUser(user);

        AnchorPane pane = FXMLLoader.load(getClass().getResource("TriviaGuiGameFXML.fxml"));
        loginPane.getChildren().setAll(pane);
    }
}

private boolean isLoginSuccessful() {
    return isUsernameValid() && isPasswordValid();
}
```

Проверка в TriviaDataManager:

```
public boolean isCorrectPassword(String username, String password) {  
    db2M.openConnection();  
    String selectStatement = String.format("SELECT 1 FROM FN71975.USERS WHERE USERNAME = '%s' AND PASSWORD = '%s'", username, password);  
    boolean check = db2M.isExistingValueInTheDatabase(selectStatement);  
    db2M.closeConnection();  
    return check;  
}
```

Потенциални съобщения за грешка при въвеждане на информация:

Username doesn't exist

not_existing

Password

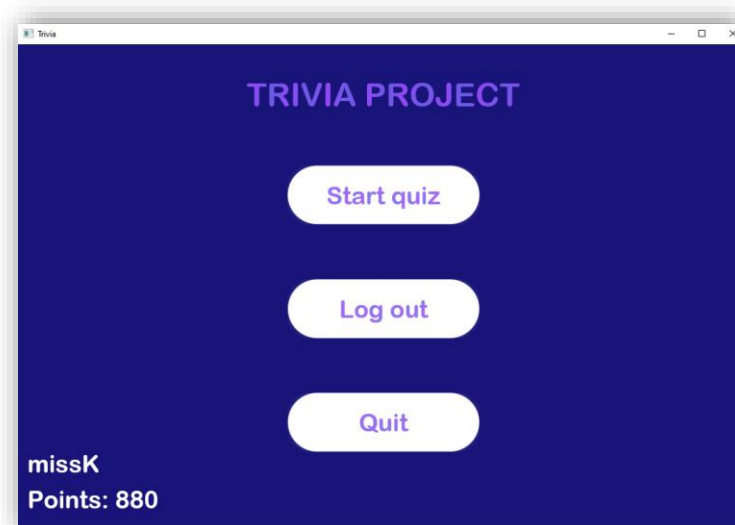
missK

Wrong password

.....

Другото интересно е, че е създаден клас `UserHolder`, който отново имплементира Singleton design pattern. След логване сетваме член данната `User` (обект) на `UserHolder` и във всеки следващ прозорец на нашето приложение ще може да достъпваме информацията на потребителя, който е влезнал в системата (ID, точки и т.н).

3. Главно меню



Тук както виждаме се запазват името на влезналия профил както и точките, които човек е събрал до момента. С Log out се връщаме на началното меню.

4. Start quiz

В момента на натискане на бутончето Start quiz се генерира прозорче, което има обект Game. Класът Game съдържа в себе си ID, въпроси (които в себе си съдържат отговори) и при всяко пускане на викторина взима от базата n на брой случайни въпроса. Също при създаване на нова игра, веднага се регистрира в базата.

```
public class Game {
    private String gameId;
    private String userID;
    private Map<String, Question> questions;
    private char usedAudience = 'n';
    private char usedFifty = 'n';
    private char usedFriend = 'n';

    public Game(int numberOfQuestions, String userID) {
        gameId = TriviaDataManager.getInstance().generateUniqueGameID();
        this.userID = userID;
        questions = new HashMap<String, Question>();
        List<String> questionIDs = TriviaDataManager.getInstance().getUniqueRandomQuestionIDs(numberOfQuestions);
        for (String ID: questionIDs) {
            questions.put(ID, new Question(ID));
        }
        registerGameInDatabase();
    }
}
```

В конструктора генерираме уникално ID за всяка игра, след това пълним списък с numberOfQuestions на брой случайно избрани ID-та от базата и след това всеки въпрос го конструираме със избрано ID.

```
public List<String> getUniqueRandomQuestionIDs(int numberOfQuestionWanted) {
    db2M.openConnection();

    StringBuilder stringBuilder = new StringBuilder("");
    List<String> result = new ArrayList<>();
    do {
        String selectStatement = String.format("SELECT ID FROM FN71975.QUESTIONS WHERE ID NOT IN(%s) ORDER BY RAND() FETCH FIRST 1 ROWS ONLY", stringBuilder.toString());
        String ID = db2M.getSelectedColumnValue(selectStatement);
        result.add(ID);
        stringBuilder.append(",").append(ID).append("");
        numberOfQuestionWanted--;
    } while(numberOfQuestionWanted > 0);
    db2M.closeConnection();

    return result;
}
```

Бъркаме в данните на случаен принцип с ORDER BY RAND(), и слагаме взетото ID в списък, така всеки път се подsigуряваме, че случайно избраното ID, не е било вече избрано. (С WHERE ID NOT IN ("12345", "54534")).

Така вече, чрез ID на всеки въпрос може спокойно да инициализираме въпросите и техните отговори.

```
public class Question {
    private String questionID;
    private String questionText;
    private String difficulty;
    private List<Answer> answers;

    public Question(String questionID) {
        this.questionID = questionID;
        questionText = TriviaDataManager.getInstance().getQuestionText(questionID);
        difficulty = TriviaDataManager.getInstance().getQuestionDifficulty(questionID);
        answers = TriviaDataManager.getInstance().getAnswers(questionID);
        Collections.shuffle(answers);
    }
}
```

Чрез ID на въпрос си набавяме всички нужни данни.
Пример за взимане на текста на всеки въпрос:

```
public String getQuestionText(String questionID) {
    db2M.openConnection();
    String selectStatement = String.format("SELECT TEXT FROM FN71975.QUESTIONS WHERE ID = '%s'", questionID);
    String text = db2M.getSelectedColumnValue(selectStatement);
    db2M.closeConnection();

    return text;
}
```

```
public String getSelectedColumnValue(String stmtnt) {
    String resultColumnValue = new String();
    try {
        resultSet = statement.executeQuery(stmtnt);
        resultSet.next();
        resultColumnValue = (resultSet.getString(columnIndex: 1));
    }
    catch (SQLException s) {
        s.printStackTrace();
    }
    return resultColumnValue;
}
```


А пък във всеки въпрос имаме списък от отговори (с брой 2 за true/false въпроси и брой 4 за другия тип въпроси.) Като този списък може да си го набавим отново с ID-то на въпрос тъй като Answers е слабо множество зависещо от Questions.

```
public class Answer {
    private String ID;
    private char answerType;
    private char correctness;
    private String answerText;

    public Answer(String ID, char answerType, char correctness, String answerText) {
        this.ID = ID;
        this.answerType = answerType;
        this.correctness = correctness;
        this.answerText = answerText;
    }
}
```

```
public List<Answer> getAnswers(String questionID) {
    db2M.openConnection();

    final int INFO_COLUMNS = 4;
    byte numberOfQuestions = (getNumberOfQuestionsNeeded(questionID));
    byte row_info = 1;

    List<Answer> result = new ArrayList<>();
    List<String> answerInfo;
    Answer answer;

    while(numberOfQuestions > 0) {
        answerInfo = db2M.getListOfResults(String.format("SELECT * FROM FN71975.ANSWERS WHERE QUESTION_ID = %s", questionID), row_info, INFO_COLUMNS);
        answer = new Answer(answerInfo.get(0), answerInfo.get(1).charAt(0), answerInfo.get(3).charAt(0), answerInfo.get(2));
        result.add(answer);
        row_info++;
        numberOfQuestions--;
    }

    db2M.closeConnection();
    return result;
}
```

Тук използваме getListOfResults, с който от таблицата Answers взимаме на row_info ред INFO_COLUMNS стойности.

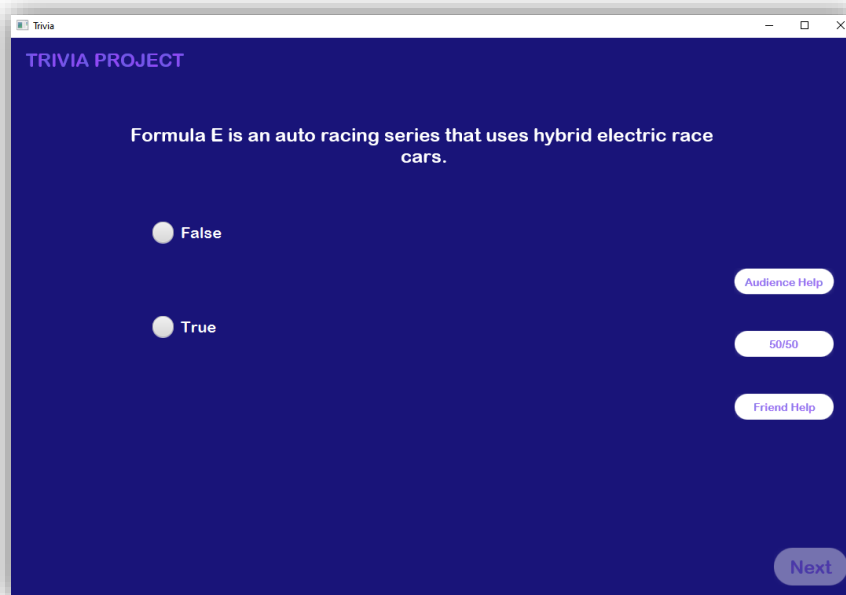
```
public List<String> getListOfResults(String stmt, int row, int column) {
    ArrayList<String> result = new ArrayList<>();
    try {
        resultSet = statement.executeQuery(stmt);
        for (int i = 0; i < row; i++) {
            resultSet.next();

            for (int j = 1; j <= column; j++) {
                result.add(resultSet.getString(j));
            }
        }
    } catch (SQLException s) {
        s.printStackTrace();
    }

    return result;
}
```

5. Използване на данните.

След генериране на данните от базата потребителя започва играта си.



Този резултата получаваме като в началото и на всяко натискане на бутон Next променяме компонентите на отворения прозорец с вече генерираната информация от базата.

```
private void generateQuestion() {
    txtQuestion.setText(questionsInGame.get(answeredQuestions).getQuestionText());
    radioAnswerOne.setText(questionsInGame.get(answeredQuestions).getAnswerTextAtIndex(0));
    radioAnswerTwo.setText(questionsInGame.get(answeredQuestions).getAnswerTextAtIndex(1));

    if(questionsInGame.get(answeredQuestions).getQuestionType() == '1') {
        radioAnswerThree.setText(questionsInGame.get(answeredQuestions).getAnswerTextAtIndex(2));
        radioAnswerThree.setVisible(true);
        radioAnswerFour.setText(questionsInGame.get(answeredQuestions).getAnswerTextAtIndex(3));
        radioAnswerFour.setVisible(true);
    } else {
        radioAnswerThree.setVisible(false);
        radioAnswerFour.setVisible(false);
    }
}
```

Също при всяко натискане на Next правим проверка дали сме отговорили на правилен въпрос за набавяне на точки.

```
private void evaluateGivenAnswer() {
    if(questionsInGame.get(answeredQuestions).isCorrectAnswerGiven(pickedAnswerIndex)) {
        System.out.println("Correct!");
        pointsFromQuestions += questionsInGame.get(answeredQuestions).getQuestionPoints();
    } else {
        System.out.println("Wrong!");
    }
}
```

Последното нещо при натискане на Next е, че записваме в базата информация в коя игра на какво и как е отговорил играчът.

```
@FXML
void btnNextOnAction(ActionEvent event) throws IOException {
    evaluateGivenAnswer();
    TriviaDataManager.getInstance().insertAnsweredQuestion(game,
        UserHolder.getInstance().getUser(),
        questionsInGame.get(answeredQuestions),
        questionsInGame.get(answeredQuestions).getAnswerAtIndex(pickedAnswerIndex));
}
```

```
public void insertAnsweredQuestion(Game game, User user, Question question, Answer answer) {
    db2M.openConnection();
    String insertStatement = " INSERT INTO FN71975.ANSWERED_QUESTIONS VALUES ('";
    insertStatement += game.getGameID() + "','";
    insertStatement += UserHolder.getInstance().getUser().getId() + "','";
    insertStatement += question.getQuestionID() + "','";
    insertStatement += answer.getAnswerID() + "')";
    db2M.executeUpdate(insertStatement);
    db2M.closeConnection();
}
```

При приключване на играта ъпдейтваме точките на потребителя както и използваните жокери на играта.

```
if(answeredQuestions == NUMBER_OF_QUESTIONS - 1) {
    UserHolder.getInstance().getUser().updatePoints(pointsFromQuestions);
    TriviaDataManager.getInstance().updateHelpers(game);
    AnchorPane pane = FXMLLoader.load(getClass().getResource("name: TriviaGuiGameFXML.fxml"));
    quizPane.getChildren().setAll(pane);
    return;
}
```

```
public void updateHelpers(Game game) {
    db2M.openConnection();
    String updateStatement = String.format("UPDATE FN71975.GAMES SET AUDIENCE_HELP = '%c', FRIEND_HELP = '%c', FIFTY_FIFTY = '%c' WHERE ID = '%s'",
        game.getUsedAudience(), game.getUsedFriend(), game.getUsedFifty(), game.getGameID());
    db2M.executeUpdate(updateStatement);
    db2M.closeConnection();
}

public void updateUserPoints(String userID, int points) {
    db2M.openConnection();
    String updateStatement = String.format("UPDATE FN71975.USERS SET POINTS = %d WHERE ID = '%s'", points, userID);
    db2M.executeUpdate(updateStatement);
    db2M.closeConnection();
}
```

```
public void executeUpdate(String stmtnt) { // С това можем да insert-ваме, update-ваме и delete-ваме.
    try {
        statement.executeUpdate(stmtnt);
    }

    catch (SQLException s){
        s.printStackTrace();
    }
    System.out.println("Successfully updated table!");
}
```