МИНИСТЕРСТВО ТРАНСПОРТА РАССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»

РУТ(МИИТ)

**Курсовой проект по предмету «Программирование»**

Выполнил:Машуков Р.А.

Группа: УВП-211

Проверил: Разживайкин И.С.

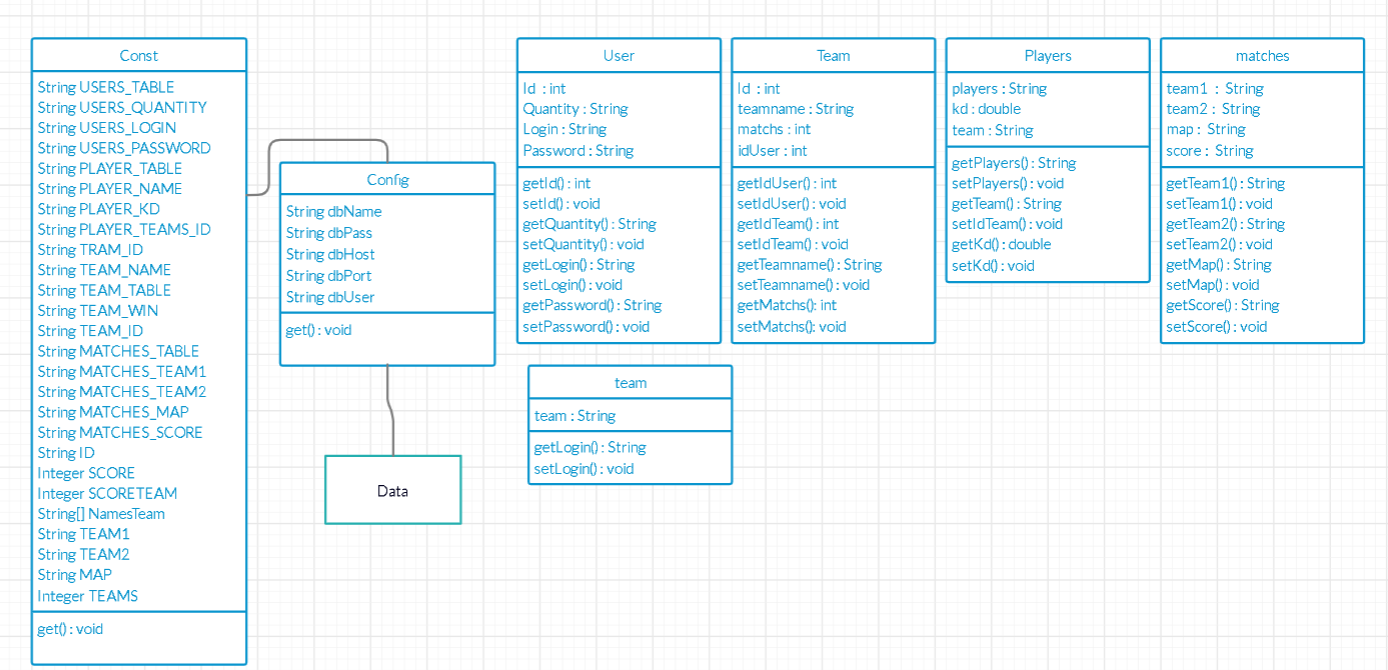
МОСКВА 2019

**Содержание:**

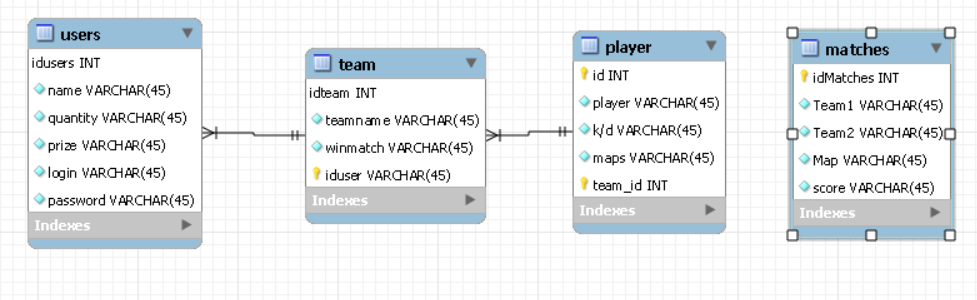
1. Постановка задачи и описание предметной области
2. Диаграмма классов
3. Диаграмма таблиц баз данных
4. Листинг кода классов
5. Таблицы из баз данных из workbench
6. Примеры экранов работы приложения
7. Вывод
8. **Постановка задачи и описание предметной области**

Основными задачами курсового проекта было реализовать в коде основные принципы ООП, реализовать работу с системой управления базами данных «MySQL», реализовать пользовательский интерфейс с помощью платформы JavaFX. За основу предметной области своего курсового проекта я взял киберспортивный турнир Counter-Strike: Global Offensive. В моем курсовом проекте реализована система проведения турнира.

1. **Диаграмма классов**

****

1. **Диаграмма таблиц баз данных**

****

1. **Листинг кода классов**

**package** sample;

**public class** Main **extends** Application {  
  
 @Override  
 **public void** start(Stage primaryStage) **throws** Exception{  
 Parent root = FXMLLoader.*load*(getClass().getResource(**"controll/login/registration.fxml"**));  
 primaryStage.setTitle(**"Вход"**);  
 primaryStage.setScene(**new** Scene(root, 720, 500));  
 primaryStage.show();  
 }  
  
 **public static void** main(String[] args) {*launch*(args);

Data.*getDbConnection*().close();

}  
}

**package** sample;  
  
**public class** Config {  
 **protected** String **dbName**=**"new\_schema"**;  
 **protected** String **dbPass**=**"1337"**;  
 **protected** String **dbHost**=**"localhost"**;  
 **protected** String **dbPort**=**"3306"**;  
 **protected** String **dbUser**=**"root"**;  
 **public void** get(){  
 }  
}

**package** sample;  
  
**public class** Const **extends** Config{  
 **public static final** String ***USERS\_TABLE*** = **"users"**;  
 **public static final** String ***USERS\_QUANTITY*** = **"quantity"**;  
 **public static final** String ***USERS\_LOGIN*** = **"login"**;  
 **public static final** String ***USERS\_PASSWORD*** = **"password"**;  
 **public static final** String ***PLAYER\_TABLE*** = **"player"**;  
 **public static final** String ***PLAYER\_NAME*** = **"player"**;  
 **public static final** String ***PLAYER\_KD*** = **"kd"**;  
 **public static final** String ***PLAYER\_TEAMS\_ID*** = **"team\_id"**;  
 **public static final** String ***TRAM\_ID*** = **"idteam"**;  
 **public static final** String ***TEAM\_NAME*** = **"teamname"**;  
 **public static final** String ***TEAM\_TABLE*** = **"team"**;  
 **public static final** String ***TEAM\_WIN***=**"matchs"**;  
 **public static final** String ***TEAM\_ID***=**"iduser"**;  
 **public static final** String ***MATCHES\_TABLE*** = **"matches"**;  
 **public static final** String ***MATCHES\_TEAM1*** = **"Team1"**;  
 **public static final** String ***MATCHES\_TEAM2*** = **"Team2"**;  
 **public static final** String ***MATCHES\_MAP*** = **"Map"**;  
 **public static final** String ***MATCHES\_SCORE*** = **"score"**;  
 **public static** String *ID*;  
 **public static** Integer *SCORE*;  
 **public static** Integer *SCORETEAM*;  
 **public static** String[] *NamesTeam*;  
 **public static** String *TEAM1*;  
 **public static** String *TEAM2*;  
 **public static** String *MAP*;  
 **public static** Integer *TEAMS*;  
 **public void** get(){  
 **dbName**=**"new\_schema"**;  
 }  
}

**package** sample;  
  
**public class** Data **extends** Config {  
  
 **public static** Connection getDbConnection() **throws** ClassNotFoundException, SQLException {  
 String connectionString = **"jdbc:mysql://"** + *dbHost* + **":"** + *dbPort* +**"/"** +*dbName*+**"?serverTimezone=UTC"**;  
 Class.*forName*(**"com.mysql.cj.jdbc.Driver"**);  
 **return** DriverManager.*getConnection*(connectionString, *dbUser*, *dbPass*);  
}

**public void** singUpUser(String key,String login, String password) **throws** ClassNotFoundException {  
 String insert= **"INSERT INTO "**+ Const.***USERS\_TABLE***+**"("** +Const.***USERS\_QUANTITY***+**","**+Const.***USERS\_LOGIN***+**","**+Const.***USERS\_PASSWORD***+**")"**+**"VALUES(?,?,?)"**;  
 **try** {  
 PreparedStatement prSt= getDbConnection().prepareStatement(insert);  
 prSt.setString(1,key);  
 prSt.setString(2,login);  
 prSt.setString(3,password);  
 prSt.executeUpdate();  
 }  
 **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 **public** ResultSet getUser(User user) **throws** ClassNotFoundException {  
 ResultSet resSet= **null**;  
 String select = **"SELECT \* FROM "**+Const.***USERS\_TABLE***+**" WHERE "**+Const.***USERS\_LOGIN***+**"=? AND "**+Const.***USERS\_PASSWORD***+**"=?"**;  
 **try** {  
 PreparedStatement prSt= getDbConnection().prepareStatement(select);  
 prSt.setString(1,user.getLogin());  
 prSt.setString(2,user.getPassword());  
 resSet=prSt.executeQuery();  
 }  
 **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
  
 **return** resSet;  
 }  
 **public** String getsId(User user) **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"SELECT \* FROM users WHERE login=? AND password=?"**);  
  
 stmt.setString(1, user.getLogin());  
 stmt.setString(2, user.getPassword());  
 ResultSet rs = stmt.executeQuery();  
 String a = **"0"**;  
 **while**(rs.next()){  
 a= rs.getString(**"idusers"**);  
 }  
 **return** a;  
 }  
 **public** String getsIdTeam(Team team) **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"SELECT \* FROM team WHERE teamname=?"**);  
 stmt.setString(1, team.getTeamname());  
 ResultSet rs = stmt.executeQuery();  
 String a = **"0"**;  
 **while**(rs.next()){  
 a= rs.getString(Const.***TRAM\_ID***);  
 }  
 **return** a;  
 }  
 **public** Integer getsCount(Team team) **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"SELECT \* FROM team WHERE iduser=?"**);  
 stmt.setInt(1, team.getIdUser());  
 ResultSet rs = stmt.executeQuery();  
 Integer a = 0;  
 **while**(rs.next()){  
 a++;  
 }  
 **return** a;  
 }  
 **public** Integer getsCountTeam(User user) **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"SELECT \* FROM users WHERE login=? AND password=?"**);  
  
 stmt.setString(1, user.getLogin());  
 stmt.setString(2, user.getPassword());  
 ResultSet rs = stmt.executeQuery();  
 Integer a = 0;  
 **while**(rs.next()){  
 a= rs.getInt(Const.***USERS\_QUANTITY***);  
 }  
 **return** a;  
 }  
 **public** String[] getsTeams(Team team) **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"SELECT \* FROM team WHERE iduser=?"**);  
 stmt.setInt(1, team.getIdUser());  
 ResultSet rs = stmt.executeQuery();  
 String[] a = **new** String[Const.*SCORETEAM*];  
 **int** b=0;  
 **while**(rs.next()){  
 a[b]=rs.getString(Const.***TEAM\_NAME***);  
 b++;  
 }  
 **return** a;  
 }  
  
 **public** String[] getsPlayers(Team team) **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"SELECT \* FROM player WHERE team\_id=?"**);  
 String u= getsIdTeam(team);  
 stmt.setString(1, u);  
 ResultSet rs = stmt.executeQuery();  
 String[] a = **new** String[5];  
 **int** b=0;  
 **while**(rs.next()){  
 a[b]=rs.getString(Const.***PLAYER\_NAME***);  
 b++;  
 }  
 **return** a;  
 }  
 **public** Integer getsKd(String name) **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"SELECT \* FROM player WHERE kd=?"**);  
 stmt.setString(1, name);  
 ResultSet rs = stmt.executeQuery();  
 **int** b=-1;  
 **while**(rs.next()){  
 b=rs.getInt(Const.***PLAYER\_KD***);  
 }  
 **return** b;  
 }  
 **public void** setKD(String name,**double** kd,String id) **throws** SQLException, ClassNotFoundException {  
 String a= **"UPDATE player SET kd = '"**+kd+**"' WHERE team\_id = '"**+id+**"' AND player = '"**+name+**"';"**;  
 Statement statement = getDbConnection().createStatement();  
 **double** stmt =statement.executeUpdate(a);  
 }  
 **public int** getMa(String name) **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"SELECT \* FROM team WHERE teamname=?"**);  
 stmt.setString(1, name);  
 ResultSet rs = stmt.executeQuery();  
 **int** b=-1;  
 **while**(rs.next()){  
 b=rs.getInt(**"matchs"**);  
 }  
 **return** b;  
 }  
 **public void** setMatch(String name) **throws** SQLException, ClassNotFoundException {  
 **int** b=getMa(name);  
 b++;  
 String a= **"UPDATE team SET matchs = '"**+b+**"' WHERE teamname = '"**+name+**"';"**;  
 Statement statement = getDbConnection().createStatement();  
 **double** stmt1 =statement.executeUpdate(a);  
 }  
 **public void** singUpMatch(String name1,String name2,String map,String score) **throws** ClassNotFoundException {  
 String insert= **"INSERT INTO "**+ Const.***MATCHES\_TABLE***+**"("** +Const.***MATCHES\_TEAM1***+**","**+Const.***MATCHES\_TEAM2***+**","**+Const.***MATCHES\_MAP***+**","**+Const.***MATCHES\_SCORE***+**",id\_user)"**+**"VALUES(?,?,?,?,?)"**;  
 **try** {  
 PreparedStatement prSt= getDbConnection().prepareStatement(insert);  
 prSt.setString(1,name1);  
 prSt.setString(2,name2);  
 prSt.setString(3,map);  
 prSt.setString(4,score);  
 prSt.setString(5,Const.*ID*);  
 prSt.executeUpdate();  
 }  
 **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 **public** ObservableList<Team> getsTeamsInfo(Team team) **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"SELECT \* FROM team WHERE iduser=?"**);  
 stmt.setInt(1, team.getIdUser());  
 ResultSet rs = stmt.executeQuery();  
 ObservableList<Team> Info = FXCollections.*observableArrayList*();  
 **while**(rs.next()){  
 Info.add(**new** Team(rs.getString(Const.***TEAM\_NAME***),rs.getInt(Const.***TEAM\_WIN***)));  
 }  
 **return** Info;  
 }  
 **public** String[] getsTeamsIds(Team team) **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"SELECT \* FROM team WHERE iduser=?"**);  
 stmt.setInt(1, team.getIdUser());  
 ResultSet rs = stmt.executeQuery();  
 String[] Info = **new** String[100];  
 **int** b=0;  
 **while**(rs.next()){  
 Info[b]=rs.getString(Const.***TRAM\_ID***);  
 b++;  
 }  
 Const.*TEAMS*=b;  
 **return** Info;  
 }  
 **public** ObservableList<Players> AllPlayers(String[] teams) **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"SELECT \* FROM player WHERE team\_id=?"**);  
 ObservableList<Players> Info = FXCollections.*observableArrayList*();  
 **for** (**int** i=0;i<=Const.*TEAMS*;i++) {  
 stmt.setString(1, teams[i]);  
 ResultSet rs = stmt.executeQuery();  
 **while** (rs.next()) {  
 Info.add(**new** Players(rs.getString(Const.***PLAYER\_NAME***), rs.getDouble(Const.***PLAYER\_KD***),Const.*NamesTeam*[i]));  
 }  
 }  
 **return** Info;  
 }  
 **public** ObservableList<matches> AllMatchs() **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"SELECT \* FROM matches WHERE id\_user=?"**);  
 ObservableList<matches> Info = FXCollections.*observableArrayList*();  
 stmt.setString(1, Const.*ID*);  
 ResultSet rs = stmt.executeQuery();  
 **while** (rs.next()) {  
 Info.add(**new** matches(rs.getString(Const.***MATCHES\_TEAM1***), rs.getString(Const.***MATCHES\_TEAM2***), rs.getString(Const.***MATCHES\_MAP***), rs.getString(Const.***MATCHES\_SCORE***)));  
 }  
 **return** Info;  
 }  
 **public void** Delete() **throws** SQLException, ClassNotFoundException {  
 PreparedStatement stmt =  
 getDbConnection().prepareStatement(**"DELETE FROM users WHERE idusers = "**+Const.*ID*);  
 **int** stmt1 =stmt.executeUpdate();  
 }  
  
}

**package** sample.controll.registration;  
  
**public class** User {  
 **private int Id**;  
 **private** String **Quantity**;  
 **private** String **Login**;  
 **private** String **Password**;  
  
 **public** User() {  
  
 }  
 **public int** getId() {  
 **return Id**;  
 }  
  
 **public void** setId(Integer id) {  
 **Id** = id;  
 }  
  
 **public** String getQuantity() {  
 **return Quantity**;  
 }  
  
 **public void** setQuantity(String quantity) {  
 **Quantity** = quantity;  
 }  
  
 **public** String getLogin() {  
 **return Login**;  
 }  
  
 **public void** setLogin(String login) {  
 **Login** = login;  
 }  
  
 **public** String getPassword() {  
 **return Password**;  
 }  
  
 **public void** setPassword(String password) {  
 **Password** = password;  
 }  
}

**package** sample;  
  
**public class** Team {  
  
  
  
  
 **private** Integer **id**;  
 **private** String **teamname**;  
 **private** Integer **matchs**;  
 **private** Integer **idUser**;  
 **public** Team(String teamname, Integer matchs) {  
 **this**.**teamname** = teamname;  
 **this**.**matchs** = matchs;  
 }  
 **public** Team() {  
 }  
 **public int** getIdUser() {  
 **return idUser**;  
 }  
  
 **public void** setIdUser(Integer iduser) {  
 **this**.**idUser** = iduser;  
 }  
  
 **public** Integer getIdTeam() {  
 **return id**;  
 }  
  
 **public void** setIdTeam(Integer id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getTeamname() {  
 **return teamname**;  
 }  
  
 **public void** setTeamname(String teamname) {  
 **this**.**teamname** = teamname;  
 }  
  
 **public** Integer getMatchs() {  
 **return matchs**;  
 }  
  
 **public void** setMatchs(Integer matchs) {  
 **this**.**matchs** = matchs;  
 }  
  
}

**package** sample;  
  
**public class** Players{  
  
 **private** String **players**;  
 **private double kd**;  
 **private** String **team**;  
 **public** Players(String players, **double** kd,String team) {  
 setPlayers(players);  
 setKd(kd);  
 setTeam(team);  
 }  
 **public** String getPlayers() {  
 **return players**;  
 }  
 **public void** setPlayers(String players) {  
 **this**.**players** = players;  
 }  
  
 **public** String getTeam() {  
 **return team**;  
 }  
  
 **public void** setTeam(String team) {  
 **this**.**team** = team;  
 }  
  
  
 **public double** getKd() {  
 **return kd**;  
 }  
  
 **public void** setKd(**double** kd) {  
 **this**.**kd** = kd;  
 }  
  
  
}

**package** sample.controll.overview;  
  
**public class** matches {  
 **private** String **team1**;  
 **private** String **team2**;  
 **private** String **map**;  
 **private** String **score**;  
  
 **public** matches(String team1, String team2, String map, String score) {  
 setTeam1(team1);  
 setTeam2(team2);  
 setMap(map);  
 setScore(score);  
 }  
  
 **public** String getTeam1() {  
 **return team1**;  
 }  
  
 **public void** setTeam1(String team1) {  
 **this**.**team1** = team1;  
 }  
  
 **public** String getTeam2() {  
 **return team2**;  
 }  
  
 **public void** setTeam2(String team2) {  
 **this**.**team2** = team2;  
 }  
  
 **public** String getMap() {  
 **return map**;  
 }  
  
 **public void** setMap(String map) {  
 **this**.**map** = map;  
 }  
  
 **public** String getScore() {  
 **return score**;  
 }  
  
 **public void** setScore(String score) {  
 **this**.**score** = score;  
 }  
}

**package** sample.controll.login;  
**public class** REGISTRATION {  
  
 @FXML  
 **private** ResourceBundle **resources**;  
  
 @FXML  
 **private** URL **location**;  
  
 @FXML  
 **private** TextField **login**;  
  
 @FXML  
 **private** PasswordField **password**;  
  
 @FXML  
 **private** Button **SingIn**;  
  
 @FXML  
 **private** Button **SingUp**;  
  
 @FXML  
 **void** initialize() {  
 **SingIn**.setOnAction(event -> {  
 String logintext= **login**.getText().trim();  
 String passwordtext= **password**.getText().trim();  
 **if**(logintext.equals(**""**)) {  
 System.***out***.println(**"Error"**);  
 }  
 **else** {  
 **if** (passwordtext.equals(**""**))  
 System.***out***.println(**"Error"**);  
 **else** {  
 **try** {  
 loginUser(logintext, passwordtext);  
 } **catch** (SQLException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 });  
 **SingUp**.setOnAction(event -> {  
 **SingUp**.getScene().getWindow().hide();  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/registration/REGISTR.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setTitle(**"Регистрация"**);  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
 }  
  
 **private void** loginUser(String logintext, String passwordtext) **throws** SQLException, ClassNotFoundException {  
 Data dbData = **new** Data();  
 User user = **new** User();  
 user.setLogin(logintext);  
 user.setPassword(passwordtext);  
 ResultSet result= dbData.getUser(user);  
 **int** a=0;  
 **while**(result.next())  
 {  
 a++;  
 }  
 **if**(a>=1)  
 {  
 Team team= **new** Team();  
 Const.*SCORETEAM*=dbData.getsCountTeam(user);  
 Const.*ID*= dbData.getsId(user);  
 team.setIdUser(Integer.*parseInt*(Const.*ID*));  
 Const.*NamesTeam* = dbData.getsTeams(team);  
 Const.*SCORE*= dbData.getsCount(team);  
 **SingIn**.getScene().getWindow().hide();  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/menu/window3.fxml"**));  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setTitle(**"Информация"**);  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 }  
 **else** {  
 singup Login = **new** singup(**login**);  
 singup Password = **new** singup(**password**);  
 Login.playAnim();  
 Password.playAnim();  
 }  
 }  
}

**package** sample.controll.registration;  
  
**public class** REGISTR {  
  
 @FXML  
 **private** ResourceBundle **resources**;  
  
 @FXML  
 **private** URL **location**;  
  
 @FXML  
 **private** TextField **quantity**;  
  
 @FXML  
 **private** Button **okey**;  
  
 @FXML  
 **private** TextField **login**;  
  
  
 @FXML  
 **private** TextField **password**;  
  
 @FXML  
 **private** Button **back**;  
 **public static boolean** isNumeric(String str)  
 {  
 **try** {  
 Integer d = Integer.*parseInt*(str);  
 }  
 **catch**(NumberFormatException nfe)  
 {  
 **return false**;  
 }  
 **return true**;  
 }  
 @FXML  
 **void** initialize() {  
 **okey**.setOnAction(event -> {  
 **if** (*isNumeric*(**quantity**.getText()))  
 {  
 singUpNewUser();  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/login/registration.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setTitle(**"Вход"**);  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 }  
 });  
 **back**.setOnAction(event -> {  
 **back**.getScene().getWindow().hide();  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/login/registration.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setTitle(**"Вход"**);  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
  
 }  
  
 **private void** singUpNewUser() {  
 Data db = **new** Data();  
 String Quantity=**quantity**.getText();  
 String Login=**login**.getText();  
 String Password=**password**.getText();  
 **okey**.getScene().getWindow().hide();  
 **try** {  
 db.singUpUser(Quantity, Login, Password);  
 } **catch** (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 }  
}

**package** sample.controll.menu;  
  
**public class** window3 {  
  
 @FXML  
 **private** ResourceBundle **resources**;  
  
 @FXML  
 **private** URL **location**;  
  
 @FXML  
 **private** Button **main**;  
  
 @FXML  
 **private** Button **addcomand**;  
  
 @FXML  
 **private** Button **match**;  
  
 @FXML  
 **private** Button **delete**;  
  
 @FXML  
 **void** initialize() {  
 **main**.setOnAction(event -> {  
 **main**.getScene().getWindow().hide();  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/overview/kappa.fxml"**));  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setTitle(**"Обзор"**);  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
 **addcomand**.setOnAction(event -> {  
 **if**(Const.*SCORE* <Const.*SCORETEAM*) {  
 **addcomand**.getScene().getWindow().hide();  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/registrteam/window4.fxml"**));  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setTitle(**"Регистрация команды"**);  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 }  
 **else** {  
 }  
 });  
 **match**.setOnAction(event -> {  
 **match**.getScene().getWindow().hide();  
  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/tournament/window5.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setTitle(**"Провести матч"**);  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
 **delete**.setOnAction(event -> {  
 Data dbData = **new** Data();  
 **try** {  
 dbData.Delete();  
 } **catch** (SQLException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 **delete**.getScene().getWindow().hide();  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/login/registration.fxml"**));  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setTitle(**"Обзор"**);  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
 }  
}

**package** sample.controll.registrteam;  
  
**public class** window4 **extends** Config {  
  
 @FXML  
 **private** ResourceBundle **resources**;  
  
 @FXML  
 **private** URL **location**;  
  
 @FXML  
 **private** Button **select**;  
  
 @FXML  
 **private** TextField **name**;  
  
 @FXML  
 **private** TextField **name1**;  
  
 @FXML  
 **private** TextField **name2**;  
  
 @FXML  
 **private** TextField **name3**;  
  
 @FXML  
 **private** TextField **name4**;  
  
 @FXML  
 **private** TextField **name5**;  
  
  
 **public void** singUpCommand(String Name,String Name1,String Name2,String Name3,String Name4,String Name5) **throws** SQLException, ClassNotFoundException {  
 String insert= **"INSERT INTO "**+ Const.***TEAM\_TABLE***+**"("** +Const.***TEAM\_NAME***+**","**+Const.***TEAM\_ID***+**")"**+**"VALUES(?,?)"**;  
 **try** {  
 PreparedStatement prSt= getDbConnection().prepareStatement(insert);  
 prSt.setString(1,Name);  
 prSt.setString(2,Const.*ID*);  
 prSt.executeUpdate();  
 }  
 **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 Data dbData = **new** Data();  
 Team team= **new** Team();  
 team.setTeamname(Name);  
 team.setIdUser(Integer.*parseInt*(Const.*ID*));  
 Const.*SCORE*= dbData.getsCount(team);  
 System.***out***.print(Const.*SCORE*);  
 String a= dbData.getsIdTeam(team);  
 singUpPlayer(Name1,a);  
 singUpPlayer(Name2,a);  
 singUpPlayer(Name3,a);  
 singUpPlayer(Name4,a);  
 singUpPlayer(Name5,a);  
 }  
 **public void** singUpPlayer(String Name,String id\_team)  
 {  
 String insertPlayer= **"INSERT INTO "**+ Const.***PLAYER\_TABLE***+**"("** +Const.***PLAYER\_NAME***+**","**+Const.***PLAYER\_TEAMS\_ID***+**")"**+**"VALUES(?,?)"**;  
 **try** {  
 PreparedStatement prSt= getDbConnection().prepareStatement(insertPlayer);  
 prSt.setString(1,Name);  
 prSt.setString(2,id\_team);  
 prSt.executeUpdate();  
 }  
 **catch** (SQLException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **public** Connection getDbConnection()  
 **throws** ClassNotFoundException, SQLException{  
 String connectionString = **"jdbc:mysql://"** + **dbHost** + **":"** + **dbPort** +**"/"** +**dbName**+**"?serverTimezone=UTC"**;  
 Class.*forName*(**"com.mysql.cj.jdbc.Driver"**);  
 **return** DriverManager.*getConnection*(connectionString, **dbUser**, **dbPass**);  
 }  
  
 @FXML  
 **void** initialize() {  
 **select**.setOnAction(event -> {  
 **if** (*isNumeric*(**name**.getText()) | *isNumeric*(**name1**.getText()) | *isNumeric*(**name2**.getText()) | *isNumeric*(**name3**.getText()) | *isNumeric*(**name4**.getText()) | *isNumeric*(**name5**.getText()))  
 {singup Name = **new** singup(**name**);  
 singup Name1 = **new** singup(**name1**);  
 singup Name2 = **new** singup(**name2**);  
 singup Name3 = **new** singup(**name3**);  
 singup Name4 = **new** singup(**name4**);  
 singup Name5 = **new** singup(**name5**);  
 Name.playAnim();  
 Name1.playAnim();  
 Name2.playAnim();  
 Name3.playAnim();  
 Name4.playAnim();  
 Name5.playAnim();  
 }  
 **else** {  
 **try** {  
 singUpNewUser();  
 } **catch** (SQLException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 **select**.getScene().getWindow().hide();  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/menu/window3.fxml"**));  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setTitle(**"Обзор"**);  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 }  
 });  
  
 }  
 **private void** singUpNewUser() **throws** SQLException, ClassNotFoundException {  
 String Name=**name**.getText();  
 String Name1=**name1**.getText();  
 String Name2=**name2**.getText();  
 String Name3=**name3**.getText();  
 String Name4=**name4**.getText();  
 String Name5=**name5**.getText();  
 **select**.getScene().getWindow().hide();  
 singUpCommand(Name, Name1, Name2, Name3, Name4,Name5);  
  
 }  
}

**package** sample.controll.tournament;  
  
**public class** window5 {  
  
 @FXML  
 **private** ResourceBundle **resources**;  
  
 @FXML  
 **private** URL **location**;  
  
 @FXML  
 **private** ChoiceBox<String> **team1**;  
  
 @FXML  
 **private** ChoiceBox<String> **team2**;  
  
 @FXML  
 **private** TextField **map**;  
  
 @FXML  
 **private** Button **apply**;  
 @FXML  
 **private** Button **back**;  
  
 @FXML  
 **void** initialize() **throws** SQLException, ClassNotFoundException {  
 Data dbData = **new** Data();  
 Team team= **new** Team();  
 team.setIdUser(Integer.*parseInt*(Const.*ID*));  
 Const.*NamesTeam* = dbData.getsTeams(team);  
 **int** a=0;  
 **while**(a<Const.*SCORETEAM*) {  
 **team1**.getItems().add(Const.*NamesTeam*[a]);  
 a++;  
 }  
 **int** b=0;  
 **while**(b<Const.*SCORETEAM*) {  
 **team2**.getItems().add(Const.*NamesTeam*[b]);  
 b++;  
 }  
 **apply**.setOnAction(event -> {  
 String Team1 = **team1**.getValue();  
 String Team2 = **team2**.getValue();  
 Const.*TEAM1*= Team1;  
 Const.*TEAM2*= Team2;  
 **if**(!Team1.equals(Team2)) {  
 Const.*MAP*=**map**.getText();  
 **apply**.getScene().getWindow().hide();  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/game/xxx.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setTitle(**"Регистрация матча"**);  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 }  
 });  
 **back**.setOnAction(event -> {  
 **back**.getScene().getWindow().hide();  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/menu/window3.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setTitle(**"Регистрация команды"**);  
 stage.setScene(**new** Scene(root));  
 stage.show();  
  
 });  
 }  
}

**package** sample.controll.game;  
  
**public class** team {  
  
 **private** String **team**;  
  
  
 **public** team(String login) {  
 setLogin(login);  
 }  
  
 **public** team() {  
 }  
  
 **public** String getLogin() {  
 **return team**;  
 }  
  
 **public void** setLogin(String login) {  
 **this**.**team** = login;  
 }  
}

**package** sample.controll.game;  
  
**public class** game {  
  
 **public** Button **apply**;  
 **private** ObservableList<team> **usersData** = FXCollections.*observableArrayList*();  
 **private** ObservableList<team> **data** = FXCollections.*observableArrayList*();  
 @FXML  
 **private** TableView<team> **tableUsers**;  
  
 @FXML  
 **private** TableColumn<team, String> **loginColumn**;  
 @FXML  
 **private** TableView<team> **tableUsers1**;  
  
 @FXML  
 **private** TableColumn<team, String> **loginColumn1**;  
  
 @FXML  
 **private** TextField **kill1**;  
  
 @FXML  
 **private** TextField **death1**;  
  
 @FXML  
 **private** TextField **kill2**;  
  
 @FXML  
 **private** TextField **death2**;  
  
 @FXML  
 **private** TextField **kill3**;  
  
 @FXML  
 **private** TextField **death3**;  
  
 @FXML  
 **private** TextField **kill4**;  
  
 @FXML  
 **private** TextField **death4**;  
  
 @FXML  
 **private** TextField **kill5**;  
  
 @FXML  
 **private** TextField **death5**;  
  
 @FXML  
 **private** TextField **kill6**;  
  
 @FXML  
 **private** TextField **death6**;  
  
 @FXML  
 **private** TextField **kill7**;  
  
 @FXML  
 **private** TextField **death7**;  
  
 @FXML  
 **private** TextField **kill8**;  
  
 @FXML  
 **private** TextField **death8**;  
  
 @FXML  
 **private** TextField **kill9**;  
  
 @FXML  
 **private** TextField **death9**;  
  
 @FXML  
 **private** TextField **kill10**;  
  
 @FXML  
 **private** TextField **death10**;  
  
 @FXML  
 **private** TextField **score**;  
  
 *// инициализируем форму данными* @FXML  
 **private void** initialize() **throws** SQLException, ClassNotFoundException {  
 Data dbData = **new** Data();  
 Team team1= **new** Team();  
 team1.setTeamname(Const.*TEAM1*);  
  
 Team team2= **new** Team();  
 String a1=dbData.getsIdTeam(team1);  
 team2.setTeamname(Const.*TEAM2*);  
 String a2=dbData.getsIdTeam(team2);  
 String[] Team1= dbData.getsPlayers(team1);  
 String[] Team2= dbData.getsPlayers(team2);  
 initData(Team1,**usersData**);  
 initData(Team2,**data**);  
  
 **loginColumn**.setCellValueFactory(**new** PropertyValueFactory<>(**"Login"**));  
 **loginColumn1**.setCellValueFactory(**new** PropertyValueFactory<>(**"Login"**));  
  
 *// заполняем таблицу данными* **tableUsers**.setItems(**usersData**);  
 **tableUsers1**.setItems(**data**);  
 **kill1**.getText();  
  
 **apply**.setOnAction(event -> {  
 **if** (!*isNumeric*(**kill1**.getText()) | !*isNumeric*(**kill2**.getText()) | !*isNumeric*(**kill3**.getText()) | !*isNumeric*(**kill4**.getText()) | !*isNumeric*(**kill5**.getText()) | !*isNumeric*(**kill6**.getText())| !*isNumeric*(**kill7**.getText())| !*isNumeric*(**kill8**.getText())| !*isNumeric*(**kill9**.getText())| !*isNumeric*(**kill10**.getText())| !*isNumeric*(**death1**.getText()) | !*isNumeric*(**death2**.getText()) | !*isNumeric*(**death3**.getText()) | !*isNumeric*(**death4**.getText()) | !*isNumeric*(**death5**.getText()) | !*isNumeric*(**death6**.getText())| !*isNumeric*(**death7**.getText())| !*isNumeric*(**death8**.getText())| !*isNumeric*(**death9**.getText())| !*isNumeric*(**death10**.getText())) {  
 singup Name = **new** singup(**kill1**);  
 singup Name1 = **new** singup(**kill2**);  
 singup Name2 = **new** singup(**kill3**);  
 singup Name3 = **new** singup(**kill4**);  
 singup Name4 = **new** singup(**kill5**);  
 singup Name5 = **new** singup(**kill6**);  
 singup Name6 = **new** singup(**kill7**);  
 singup Name7 = **new** singup(**kill8**);  
 singup Name8 = **new** singup(**kill9**);  
 singup Name9 = **new** singup(**kill10**);  
 singup Name10 = **new** singup(**death1**);  
 singup Name11 = **new** singup(**death2**);  
 singup Name12 = **new** singup(**death3**);  
 singup Name13 = **new** singup(**death4**);  
 singup Name14 = **new** singup(**death5**);  
 singup Name15 = **new** singup(**death6**);  
 singup Name16 = **new** singup(**death7**);  
 singup Name17 = **new** singup(**death8**);  
 singup Name18 = **new** singup(**death9**);  
 singup Name19 = **new** singup(**death10**);  
 Name.playAnim();  
 Name1.playAnim();  
 Name2.playAnim();  
 Name3.playAnim();  
 Name4.playAnim();  
 Name5.playAnim();  
 Name6.playAnim();  
 Name7.playAnim();  
 Name8.playAnim();  
 Name9.playAnim();  
 Name10.playAnim();  
 Name11.playAnim();  
 Name12.playAnim();  
 Name13.playAnim();  
 Name14.playAnim();  
 Name15.playAnim();  
 Name16.playAnim();  
 Name17.playAnim();  
 Name18.playAnim();  
 Name19.playAnim();  
 System.***out***.println(**kill1**.getText());  
 **try** {  
 System.***out***.println(dbData.getsKd(**kill1**.getText()));  
 } **catch** (SQLException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
 **else** {  
 **try** {  
 **if**(dbData.getMa(Const.*TEAM1*)==0){  
 dbData.setKD(Team1[0], toDouble(**kill1**.getText(), **death1**.getText()),a1);  
 dbData.setKD(Team1[1], toDouble(**kill2**.getText(), **death2**.getText()),a1);  
 dbData.setKD(Team1[2], toDouble(**kill3**.getText(), **death3**.getText()),a1);  
 dbData.setKD(Team1[3], toDouble(**kill4**.getText(), **death4**.getText()),a1);  
 dbData.setKD(Team1[4], toDouble(**kill5**.getText(), **death5**.getText()),a1);}  
 **else**{ dbData.setKD(Team1[0], ((toDouble(**kill1**.getText(), **death1**.getText()))+dbData.getsKd(Team1[0]))/2,a1);  
 dbData.setKD(Team1[1], ((toDouble(**kill2**.getText(), **death2**.getText()))+dbData.getsKd(Team1[1]))/2,a1);  
 dbData.setKD(Team1[2], ((toDouble(**kill3**.getText(), **death3**.getText()))+dbData.getsKd(Team1[2]))/2,a1);  
 dbData.setKD(Team1[3], ((toDouble(**kill4**.getText(), **death4**.getText()))+dbData.getsKd(Team1[3]))/2,a1);  
 dbData.setKD(Team1[4], ((toDouble(**kill5**.getText(), **death5**.getText()))+dbData.getsKd(Team1[4]))/2,a1);}  
 **if**(dbData.getMa(Const.*TEAM2*)==0) {  
 dbData.setKD(Team2[0], toDouble(**kill6**.getText(), **death6**.getText()),a2);  
 dbData.setKD(Team2[1], toDouble(**kill7**.getText(), **death7**.getText()),a2);  
 dbData.setKD(Team2[2], toDouble(**kill8**.getText(), **death8**.getText()),a2);  
 dbData.setKD(Team2[3], toDouble(**kill9**.getText(), **death9**.getText()),a2);  
 dbData.setKD(Team2[4], toDouble(**kill10**.getText(), **death10**.getText()),a2);}  
 **else** {dbData.setKD(Team2[0], ((toDouble(**kill6**.getText(), **death6**.getText()))+dbData.getsKd(Team2[0]))/2,a2);  
 dbData.setKD(Team2[1], ((toDouble(**kill7**.getText(), **death7**.getText()))+dbData.getsKd(Team2[1]))/2,a2);  
 dbData.setKD(Team2[2], ((toDouble(**kill8**.getText(), **death8**.getText()))+dbData.getsKd(Team2[2]))/2,a2);  
 dbData.setKD(Team2[3], ((toDouble(**kill9**.getText(), **death9**.getText()))+dbData.getsKd(Team2[3]))/2,a2);  
 dbData.setKD(Team2[4], ((toDouble(**kill10**.getText(), **death10**.getText()))+dbData.getsKd(Team2[4]))/2,a2);}  
 } **catch** (SQLException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 **try** {  
 dbData.setMatch(Const.*TEAM1*);  
 dbData.setMatch(Const.*TEAM2*);  
 dbData.singUpMatch(Const.*TEAM1*,Const.*TEAM2*,Const.*MAP*,**score**.getText());  
 } **catch** (ClassNotFoundException | SQLException e) {  
 e.printStackTrace();  
 }  
 **apply**.getScene().getWindow().hide();  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/menu/window3.fxml"**));  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setTitle(**"Информация"**);  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 }  
 });  
  
 }  
  
 *// подготавливаем данные для таблицы  
 // вы можете получать их с базы данных* **private void** initData(String[] team,ObservableList<team> data1) {  
 data1.add(**new** team(team[0]));  
 data1.add(**new** team(team[1]));  
 data1.add(**new** team(team[2]));  
 data1.add(**new** team(team[3]));  
 data1.add(**new** team(team[4]));  
 }  
 **private double** toDouble(String a,String t)  
 {  
 Double i1 =1.0;  
 Double i2=1.0;  
 **try** {  
 i1 = Double.*valueOf*(a);  
 i2 = Double.*valueOf*(t);  
 }**catch** (NumberFormatException e) {  
  
 }  
 **return** i1/i2;  
 }  
  
}

**package** sample.controll.overview;  
  
**public class** playerController {  
  
 @FXML  
 **private** Button **key3**;  
  
 @FXML  
 **private** Button **key4**;  
  
 @FXML  
 **private** ObservableList<Players> **usersData** = FXCollections.*observableArrayList*();  
  
 @FXML  
 **private** TableView<Players> **personTable**;  
  
 @FXML  
 **private** TableColumn<Players, String> **players**;  
  
 @FXML  
 **private** TableColumn<Players, String> **teams**;  
  
  
 @FXML  
 **private** TableColumn<Players, Double> **kd**;  
 @FXML  
 **private** Button **back**;  
  
 @FXML  
 **void** initialize() **throws** SQLException, ClassNotFoundException {  
 **players**.setCellValueFactory(**new** PropertyValueFactory<>(**"players"**));  
 **teams**.setCellValueFactory(**new** PropertyValueFactory<>(**"team"**));  
 **kd**.setCellValueFactory(**new** PropertyValueFactory<>(**"kd"**));  
 Data dbData = **new** Data();  
 Team team=**new** Team();  
 team.setIdUser(Integer.*parseInt*(Const.*ID*));  
 String[] teams= dbData.getsTeamsIds(team);  
 **usersData**=dbData.AllPlayers(teams);  
  
  
 **key3**.setOnAction(event -> {  
 **key3**.getScene().getWindow().hide();  
  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/overview/Teams.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
 **key4**.setOnAction(event -> {  
 **key4**.getScene().getWindow().hide();  
  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/overview/games.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
 **personTable**.setItems(**usersData**);  
 **back**.setOnAction(event -> {  
 **back**.getScene().getWindow().hide();  
  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/menu/window3.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
 }  
}

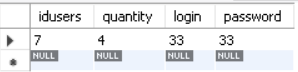
**package** sample.controll.overview;  
  
**public class** teamsController {  
 @FXML  
 **private** ObservableList<Team> **usersData** = FXCollections.*observableArrayList*();  
  
 @FXML  
 **private** Button **key2**;  
  
 @FXML  
 **private** Button **key4**;  
  
 @FXML  
 **private** TableView<Team> **team1**;  
  
 @FXML  
 **private** TableColumn<Team, String> **Teams**;  
  
 @FXML  
 **private** TableColumn<Team, Integer> **maps**;  
 @FXML  
 **private** Button **back**;  
 @FXML  
 **void** initialize() **throws** SQLException, ClassNotFoundException {  
 **Teams**.setCellValueFactory(**new** PropertyValueFactory<Team, String>(**"teamname"**));  
 **maps**.setCellValueFactory(**new** PropertyValueFactory<Team,Integer>(**"matchs"**));  
 Data dbData = **new** Data();  
 Team team=**new** Team();  
 team.setIdUser(Integer.*parseInt*(Const.*ID*));  
 **usersData**=dbData.getsTeamsInfo(team);  
 **team1**.setItems(**usersData**);  
 **key2**.setOnAction(event -> {  
 **key2**.getScene().getWindow().hide();  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/overview/kappa.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
 **key4**.setOnAction(event -> {  
 **key4**.getScene().getWindow().hide();  
  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/overview/games.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
 **back**.setOnAction(event -> {  
 **back**.getScene().getWindow().hide();  
  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/menu/window3.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
 }  
  
}

**package** sample.controll.overview;  
  
**public class** games {  
  
 @FXML  
 **private** ResourceBundle **resources**;  
  
 @FXML  
 **private** URL **location**;  
  
 @FXML  
 **private** Button **key2**;  
  
 @FXML  
 **private** Button **key1**;  
  
 @FXML  
 **private** TableView<matches> **personTable**;  
  
 @FXML  
 **private** TableColumn<matches, String> **team1**;  
  
 @FXML  
 **private** TableColumn<matches, String> **team2**;  
  
 @FXML  
 **private** TableColumn<matches,String> **map**;  
  
 @FXML  
 **private** TableColumn<matches, String> **score**;  
 @FXML  
 **private** Button **back**;  
  
 @FXML  
 **void** initialize() **throws** SQLException, ClassNotFoundException {  
 **team1**.setCellValueFactory(**new** PropertyValueFactory<>(**"team1"**));  
 **team2**.setCellValueFactory(**new** PropertyValueFactory<>(**"team2"**));  
 **map**.setCellValueFactory(**new** PropertyValueFactory<>(**"map"**));  
 **score**.setCellValueFactory(**new** PropertyValueFactory<>(**"score"**));  
 Data dbData = **new** Data();  
 **personTable**.setItems(dbData.AllMatchs());  
 **key1**.setOnAction(event -> {  
 **key1**.getScene().getWindow().hide();  
  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/overview/kappa.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
 **key2**.setOnAction(event -> {  
 **key2**.getScene().getWindow().hide();  
  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/overview/Teams.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
 **back**.setOnAction(event -> {  
 **back**.getScene().getWindow().hide();  
  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"/sample/controll/menu/window3.fxml"**));  
  
 **try** {  
 loader.load();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = **new** Stage();  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 });  
  
 }  
}

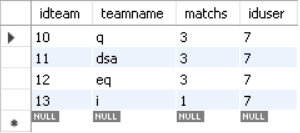
**package** sample.animation;  
  
**public class** singup {  
 **private** TranslateTransition **a**;  
 **public** singup(Node node)  
 {  
 **a**= **new** TranslateTransition(Duration.*millis*(70),node);  
 **a**.setFromX(0f);  
 **a**.setByX(20f);  
 **a**.setCycleCount(3);  
 **a**.setAutoReverse(**true**);  
 }  
 **public void** playAnim()  
 {  
 **a**.playFromStart();  
 }  
}

1. **Таблицы из баз данных MySQL Workbench**

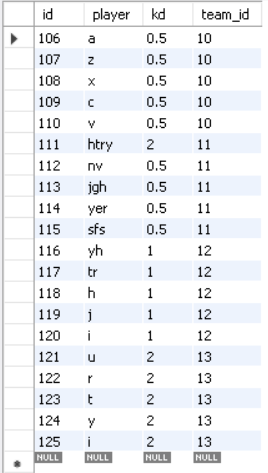
**1.users**

****

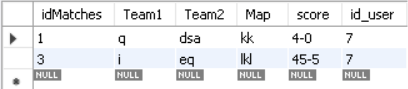
**2.team**

****

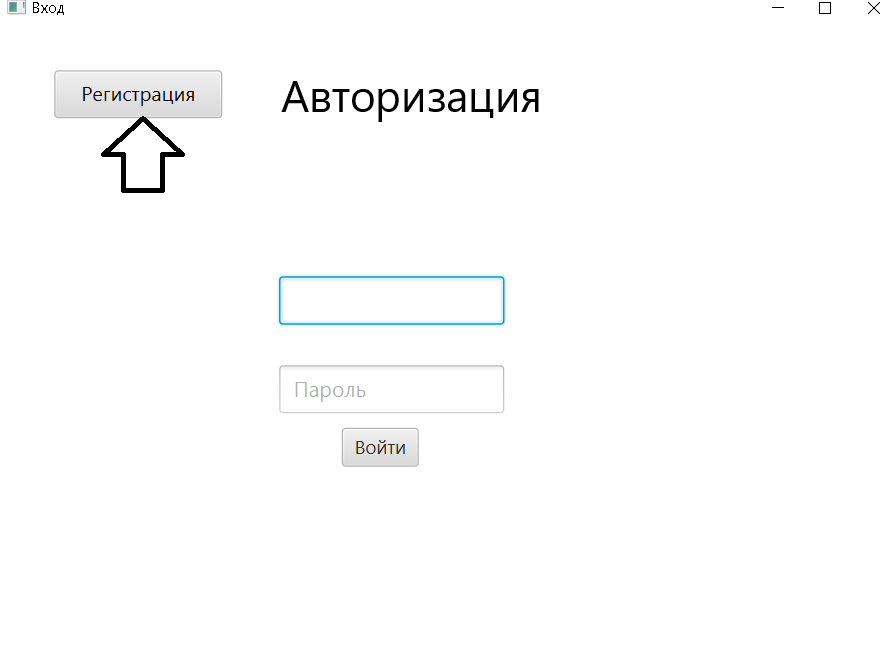
**3.player**

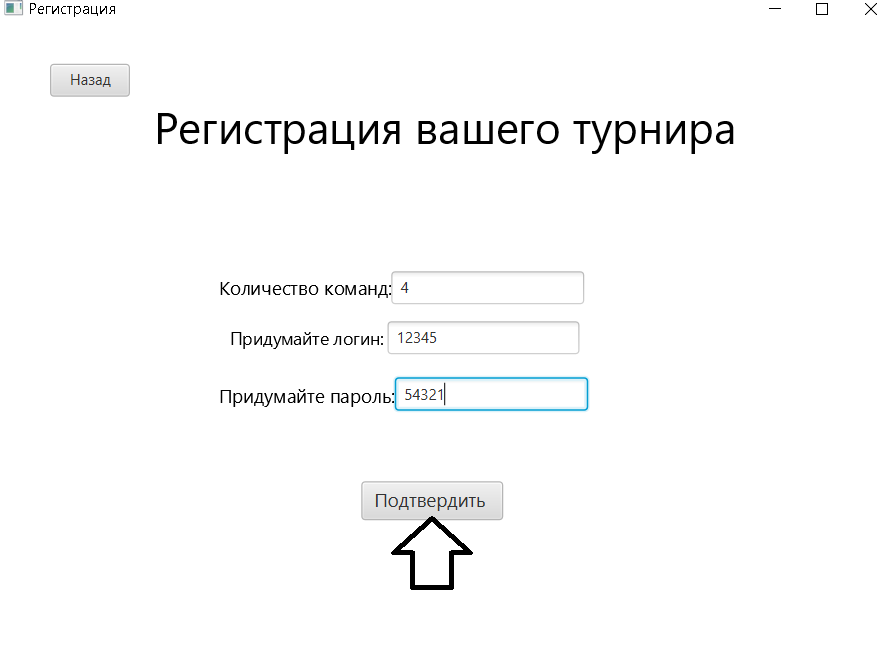
****

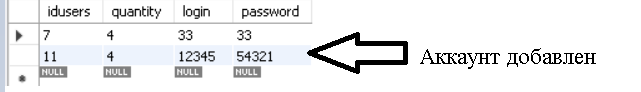
**4.matches**

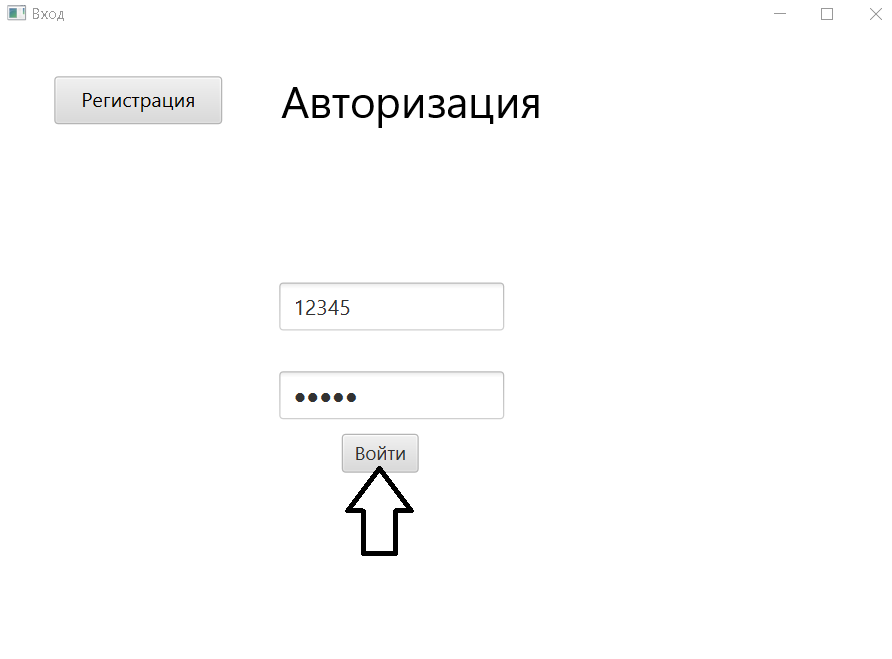
****

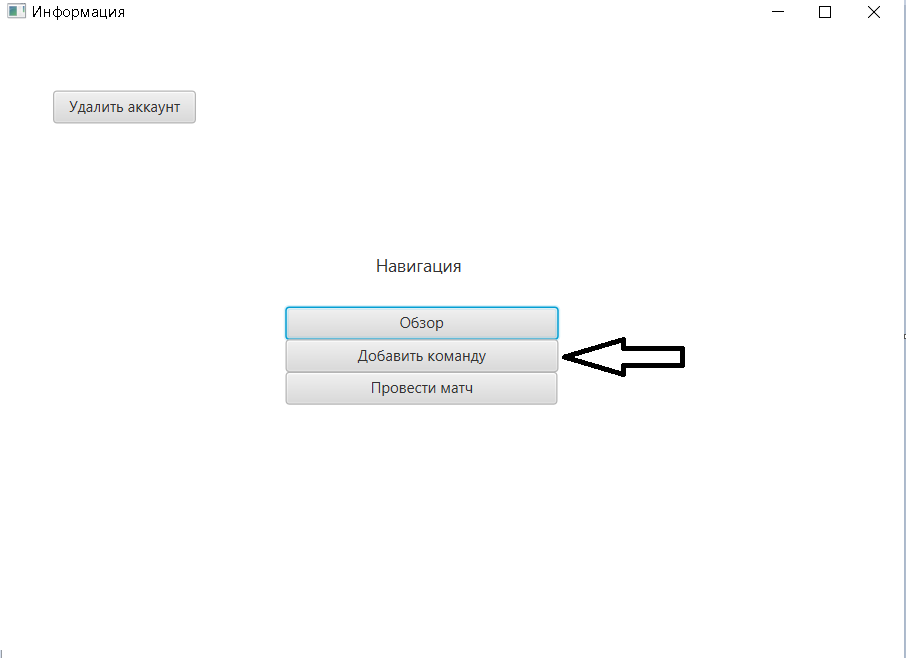
1. **Примеры работы приложения (скриншоты экранов)**

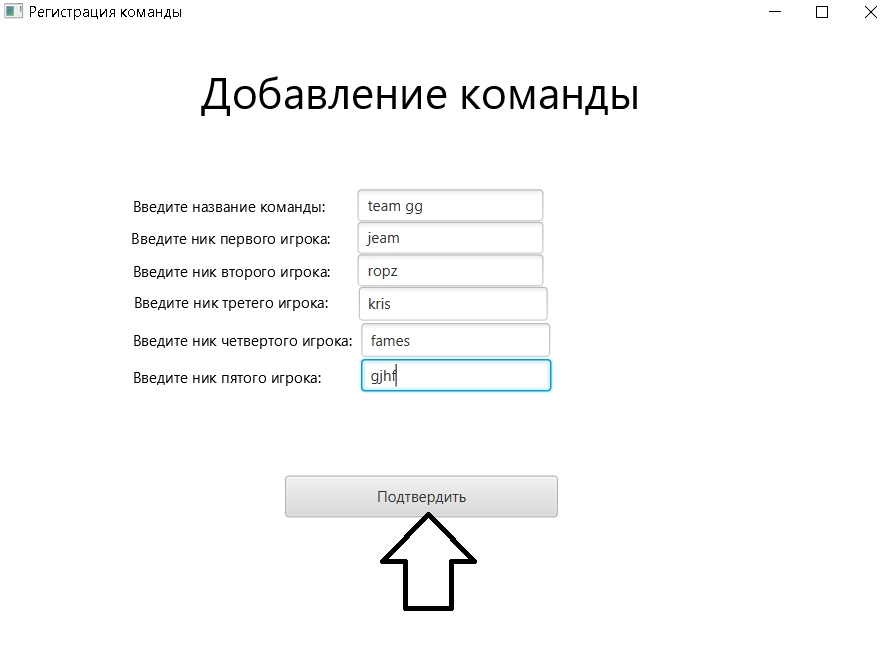
****

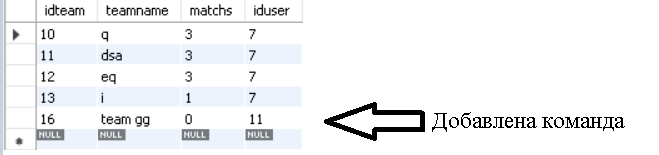
****

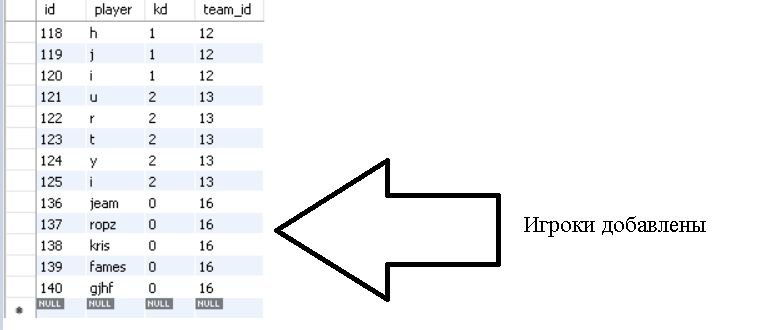
****

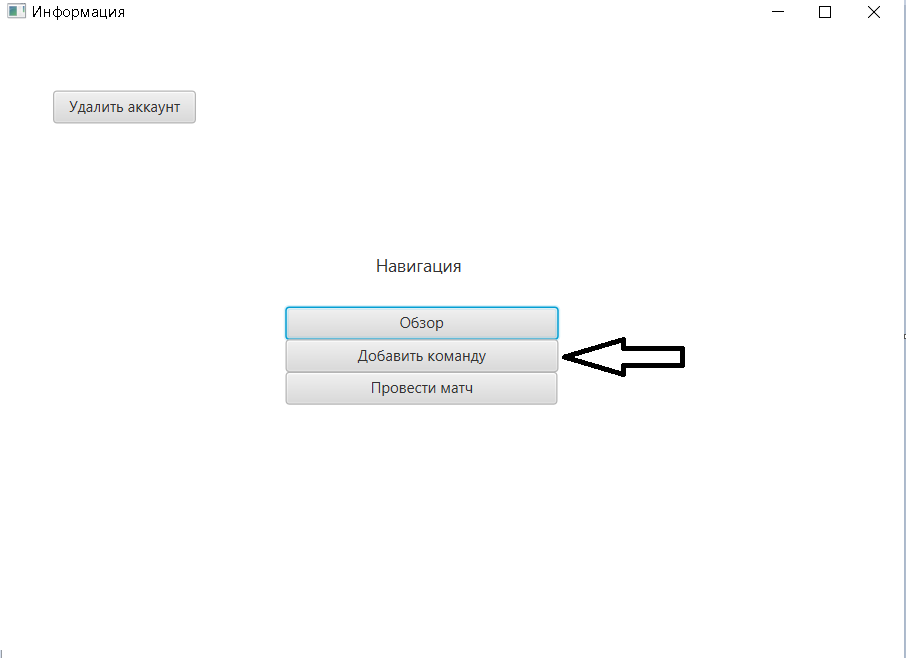
****

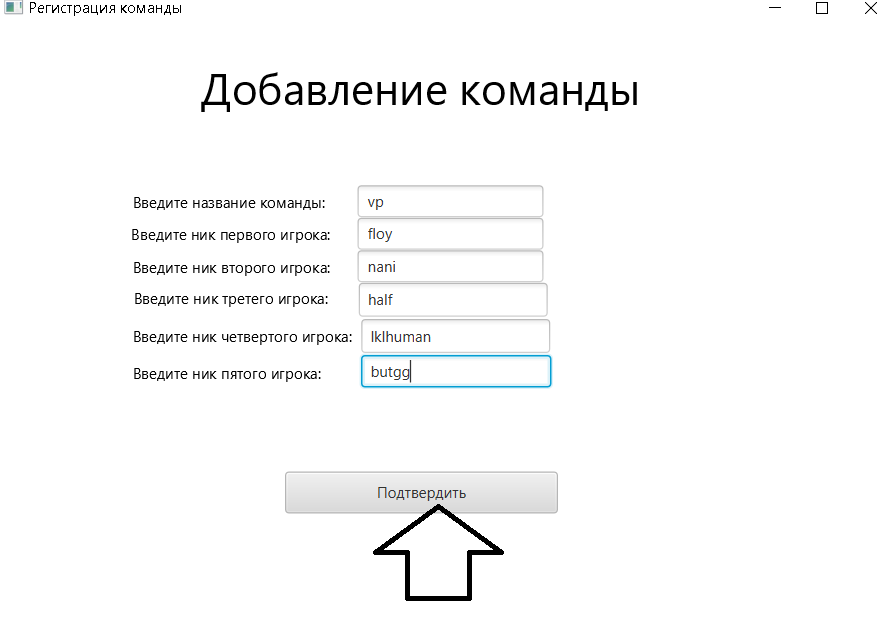
****

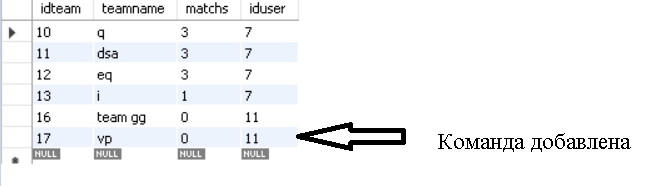
****

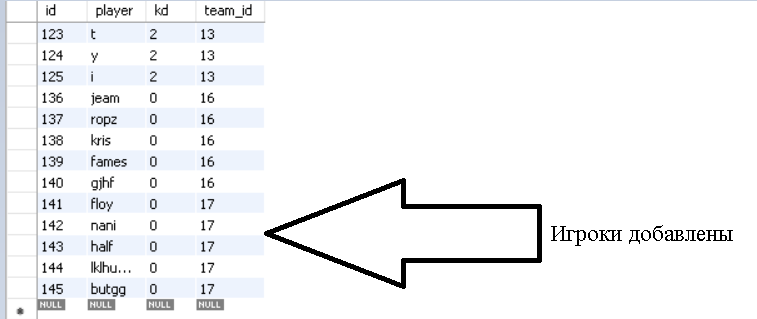
****

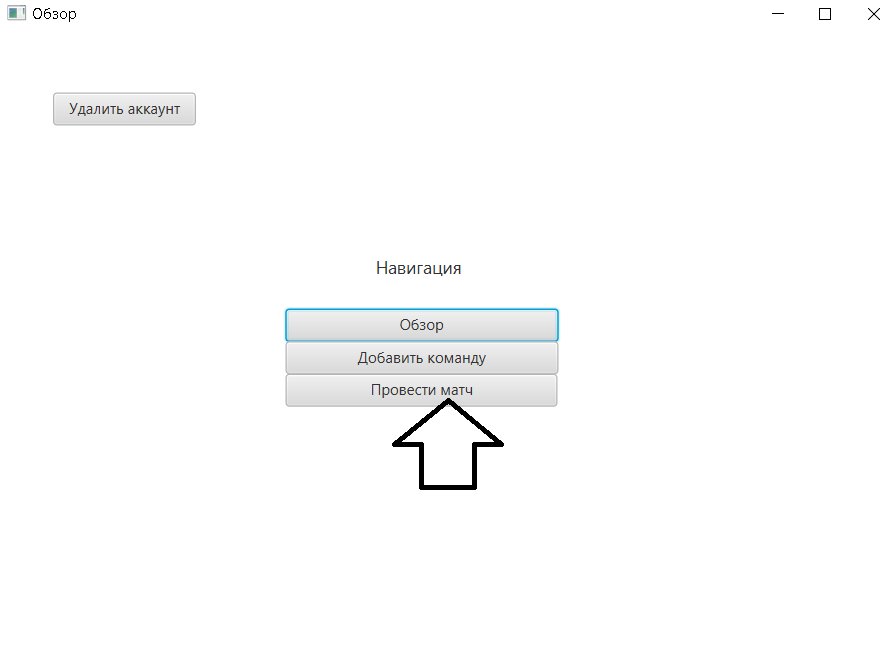
****

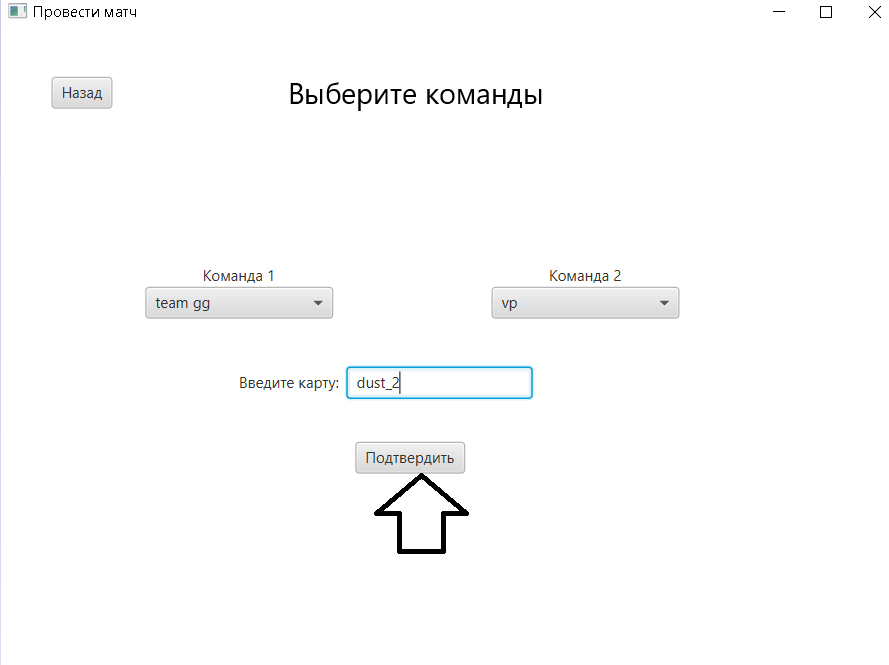
****

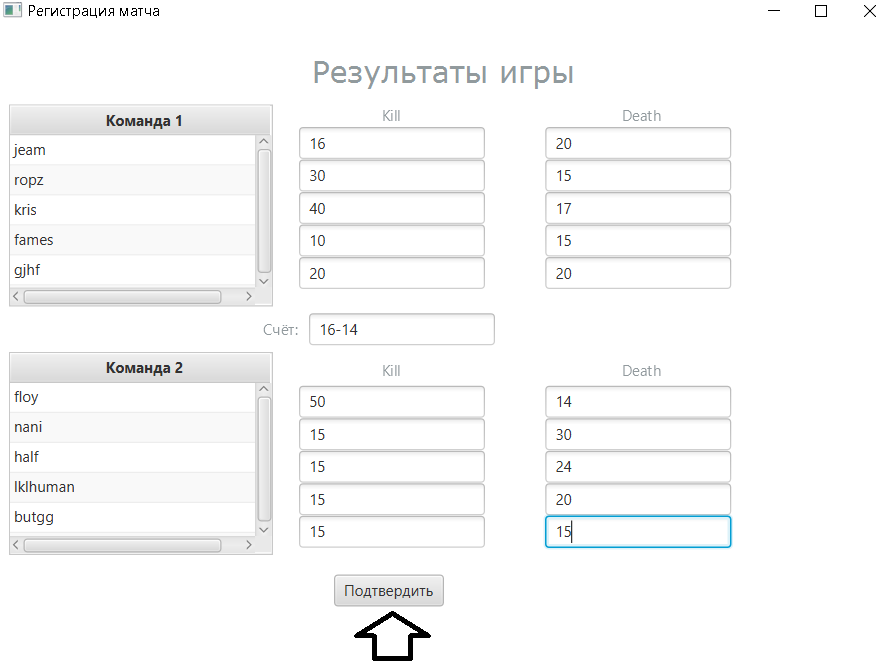
****

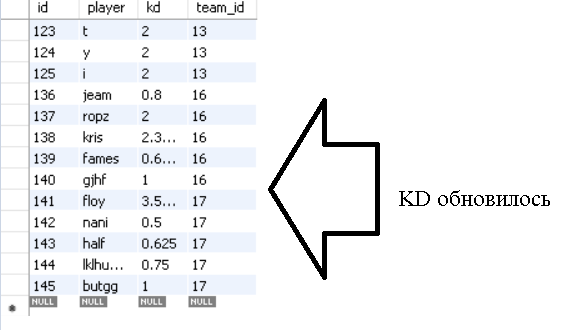
****

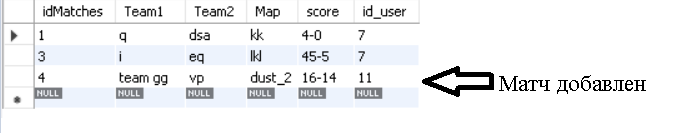
****

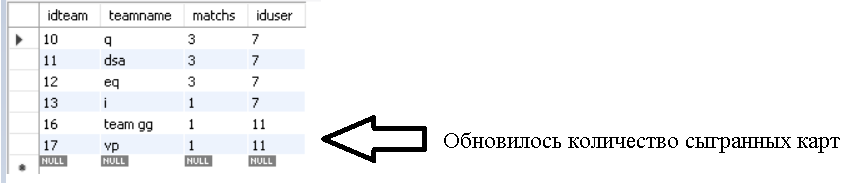
****

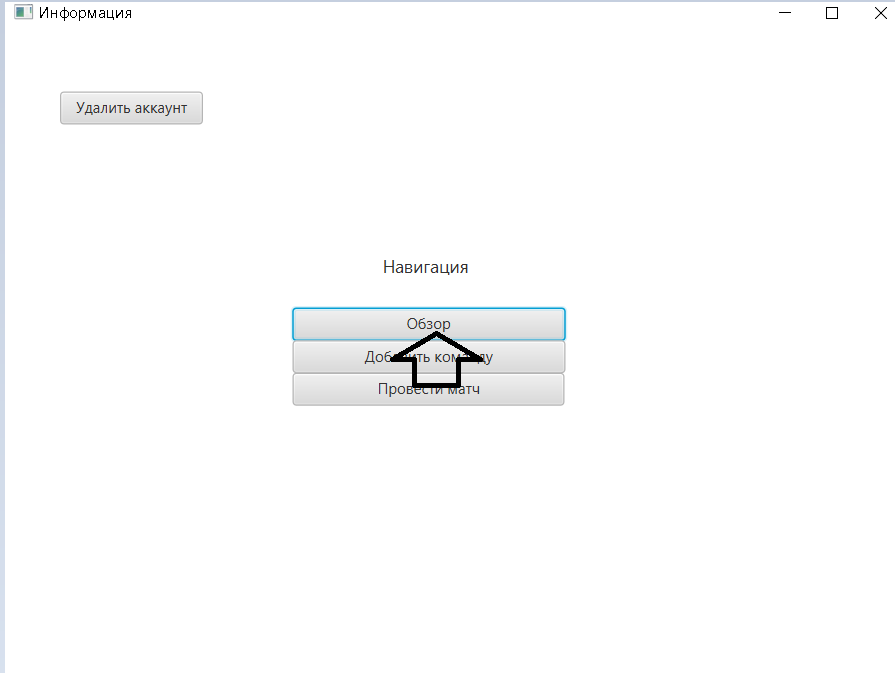
****

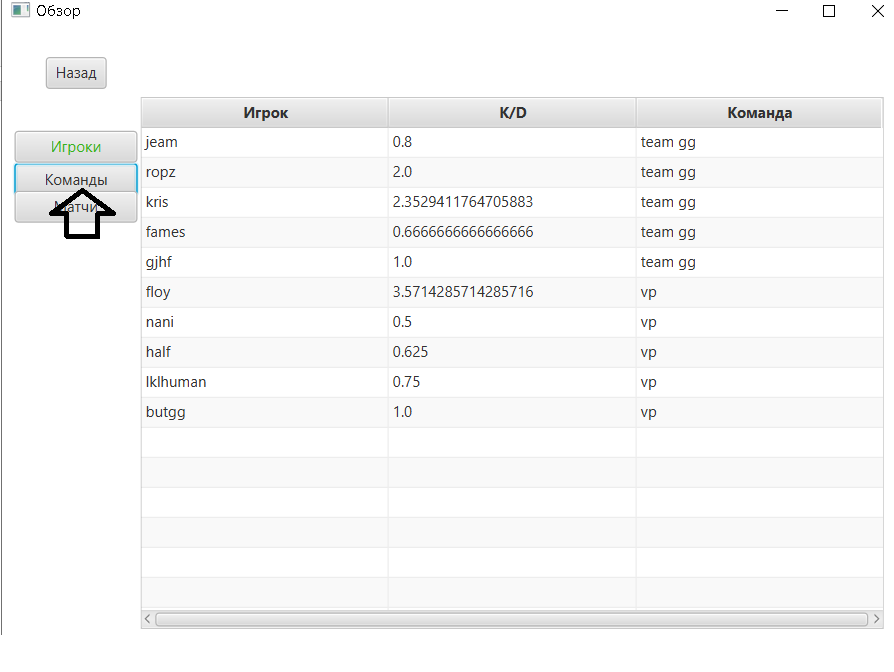
****

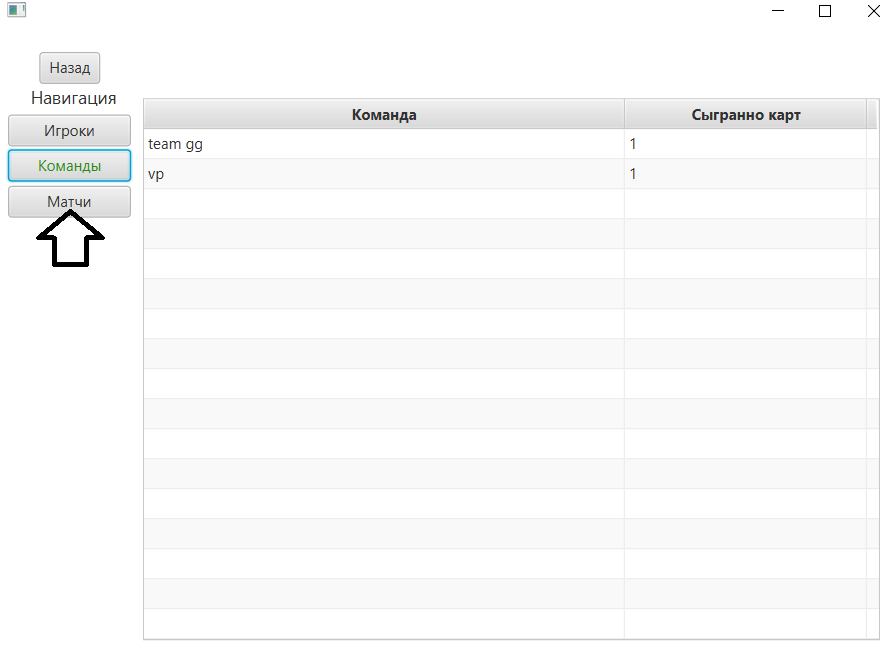
****

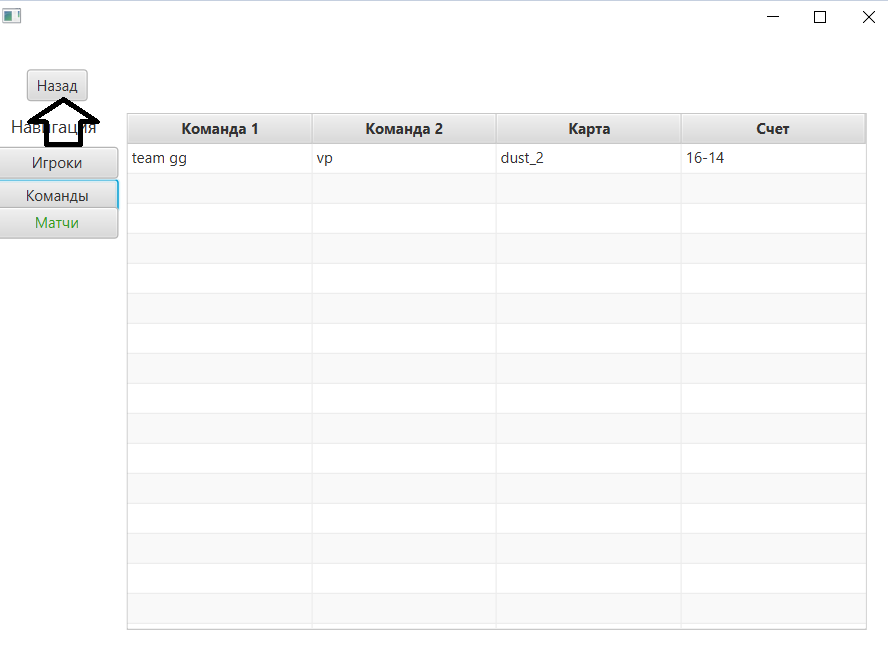
****

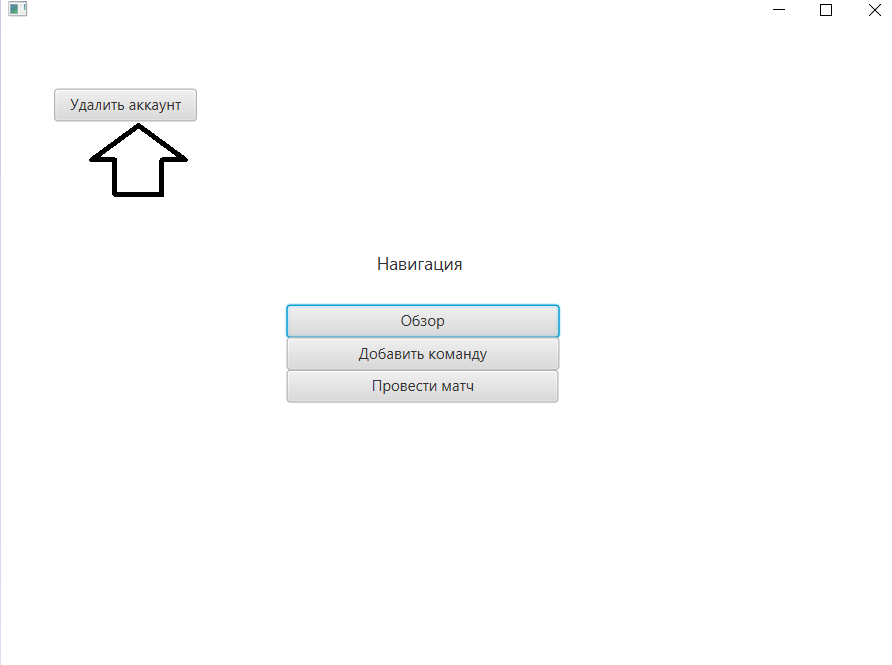
****

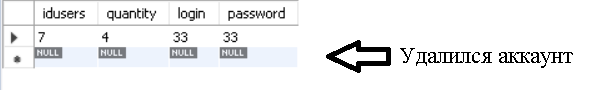
****

****

****

****

****

****

1. **Вывод**

**Во время выполнения курсового проекта я сполна ознакомился с основными принципами ООП при работе с языком Java, обучился работе с базами данных с помощью MySQL Workbench, научился создавать пользовательский интерфейс работая с JavaFX. В будущем мне очень пригодятся знания, полученные во время выполнения курсового проекта в абсолютно различных сферах и предметных областях.**