

☒ Gr. 1, Dr. E. PitzerName Robert Wurm Aufwand in h 19☐ Gr. 2, DI F. Gruber-Leitner

Punkte \_\_\_\_\_ Kurzzeichen Tutor / Übungsleiter \_\_\_\_\_ / \_\_\_\_\_

**Ausbaustufe 2: EURO-Bet-DB****(24 Punkte)**

Nachdem der Betreiber der Wettplattform von der Präsentation Ihrer beiden ersten Ausbaustufen restlos begeistert war, steht einem Ausbau bis hin zur Online-Plattform nichts mehr im Wege.

Dafür ist es im nächsten Schritt notwendig, die Daten in einer relationalen Datenbank zu speichern. Weiters beschließen Sie aufgrund Ihres nunmehrigen Wissens über die Vorzüge von RMI, Ihr Programm zu einer Client-/Server-Anwendung auszubauen.

Im Detail sollte Ihre Anwendung folgende Anforderungen erfüllen:

- Überlegen Sie sich eine geeignete Repräsentation für die Daten in Ihrem Programm, also Klassen für die Speicherung der Benutzer, Spielpaarungen, Spielergebnisse und Tipps. Diese Klassen werden häufig als Domänenmodell bezeichnet. Berücksichtigen Sie nicht nur die Daten, die vom Verwaltungstools benötigt werden, sondern auch die Daten für die Online-Wettplattform.
- Implementieren Sie einen RMI-Server, der die vom Verwaltungstool erfassten Daten zentral speichert bzw. diesen mit den zentral gespeicherten Daten versorgt. Stellen Sie sicher, dass die Serverkomponente parallel beliebig viele Clients mit Daten versorgen kann.
- Bauen Sie die in Übung 6 entwickelte JavaFX-Anwendung zu einem RMI-Client aus, der mit der Serverkomponente kommuniziert.
- Da der Zugriff auf die Server-Anwendung mitunter langsam sein kann, müssen Sie Vorkehrungen dafür treffen, dass die Kommunikation mit der Server-Anwendung und die Aktualisierung der Benutzeroberfläche entkoppelt sind.
- Stellen Sie Klassen zum Zugriff auf die Datenbank auf Basis von JDBC zur Verfügung. Bereiten Sie auch bereits Methoden für das Hinzufügen und Laden von Tipps vor. Achten Sie darauf, dass die Datenbankzugriffsschicht möglichst weitgehend von den anderen Komponenten der Anwendung getrennt ist. Stellen Sie dazu die gesamte Funktionalität der Datenbankzugriffsschicht über Interfaces zur Verfügung.
- Entwickeln Sie eine Testsuite, mit der Sie die Datenzugriffsschicht unabhängig von den anderen Systemkomponenten testen können.

Überprüfen Sie abschließend, ob Ihr Programm folgende Anforderungen erfüllt:

- Die Eingabe von Benutzern, Mannschaften, Spielpaarungen und Spielergebnissen ist möglich. Dabei werden die Daten mittels RMI zur Serveranwendung übertragen und dort mittels JDBC in eine Datenbank geschrieben. Es hängt von der Gestaltung Ihrer Benutzeroberfläche ab, an welchen Stellen Sie das Speichern der Daten anstoßen. Sorgen Sie aber dafür, dass bei unbeabsichtigter Beendigung des Programms, nicht allzu viele Daten verlorengehen.
- Bei Programmstart oder entsprechender Benutzerinteraktion („Daten aktualisieren“) werden die Daten vom RMI-Server (neu) geladen.
- Mehrere Instanzen des Verwaltungstools können gleichzeitig laufen. Änderungen des Datenbestands sind nach entsprechender Benutzerinteraktion auch in anderen Instanzen sichtbar.

- Das Eintragen und Laden von Tipps ist seitens der Datenbank und der Serveranwendung vorbereitet (es gibt aber im Moment keine Benutzeroberfläche zur Abgabe von Tipps).

# Inhaltsverzeichnis

<b>1</b>	<b>Euro-Bet-DB</b>	<b>4</b>
1.1	Lösungsidee . . . . .	4
1.2	Quellcode . . . . .	4
1.2.1	swe4.util . . . . .	4
1.2.2	swe4.models . . . . .	5
1.2.3	swe4.gui.panels . . . . .	17
1.2.4	swe4.rmi.client . . . . .	39
1.2.5	swe4.rmi.server . . . . .	48
1.3	Testfälle . . . . .	63

# 1 Euro-Bet-DB

## 1.1 Lösungsidee

Um die Daten zu repräsentieren habe ich meine Models erweitert. Die Eigenschaften der Models habe ich nun als Properties umgesetzt. Dies ermöglicht mir, die Models direkt an JavaFX Komponenten, wie zum Beispiel die Listview, zu binden.

Auf der Clientseite habe ich einen Singleton 'EuroBetDbFactory' erstellt. Dieser Singleton ist für die Kommunikation für den Server verantwortlich. Ich habe mich für diese Factory-Implementierung entschieden, da dies in der Praxis sehr oft so umgesetzt wird und einige Vorteile bringt (Backend kann leicht ausgetauscht werden, Event-Listeners können aufgesplittet werden, Server Kommunikation muss nur einmal implementiert werden, usw.. ). Die Unit-Tests habe ich ebenso gleich für meine Factory geschrieben. Sind in der UI nicht die Bedingungen erfüllt, sodass ein Update oder das Erstellen eines Objekts wie zum Beispiel User ausgeführt werden darf, so ist der Speichern-Button nicht aktiviert. Hier habe ich das Enabled-Property der Buttons an Verknüpfungen der Model-Properties gebunden.

Die Implementierung für den Server ist in dem Packet 'swe4.rmi.server' zu finden. Unter Quellcode habe ich nur noch die Klassen angeführt, die sich gegenüber der vorgehenden Übung geändert haben bzw. neu hinzugefügt worden sind, da die Dokumentation ohnehin schon sehr viele Seiten hat.

## 1.2 Quellcode

### 1.2.1 swe4.util

#### Util.java

```
1 package swe4.util;
2
3 import java.io.BufferedReader;
4 import java.io.IOException;
5 import java.io.InputStreamReader;
6
7 public class Util {
8
9     static long startTime = System.currentTimeMillis();
10 }
```

```
11     public static Object currTime() {
12         return (System.currentTimeMillis() - startTime) / 1000.0;
13     }
14
15
16     public static String getHostPortArg(String[] args) {
17         if (args == null || args.length == 0) {
18             System.out.println("WARNING: host[:port] missing (default is
19                 ↪ localhost:1099)");
20             return "localhost:1099";
21         }
22         return args[0];
23     }
24
25     public static int getPort(String host_port) {
26         int port = 1099;
27         int colonIdx = host_port.lastIndexOf(':');
28         if (colonIdx >= 0)
29             port = Integer.valueOf(host_port.substring(colonIdx + 1));
30         return port;
31     }
32
33     public static void unsafeSleep(int millis) {
34         try {
35             Thread.sleep(millis);
36         } catch (InterruptedException ex) {}
37     }
38
39     public static void unsafeWait(Object obj) {
40         try {
41             obj.wait();
42         } catch (InterruptedException e) {}
43     }
44
45     public static String readline() {
46         BufferedReader reader = new BufferedReader(new InputStreamReader(System.in)
47             ↪ );
48         try {
49             return reader.readLine();
50         } catch (IOException e) {}
51         return null;
52     }
```

### 1.2.2 swe4.models

#### Game.java

```
1 package swe4.models;
2
3 import java.io.Externalizable;
4 import java.io.IOException;
5 import java.io.ObjectInput;
6 import java.io.ObjectOutput;
7
8 import javafx.beans.property.BooleanProperty;
9 import javafx.beans.property.IntegerProperty;
10 import javafx.beans.property.SimpleBooleanProperty;
11 import javafx.beans.property.SimpleIntegerProperty;
12
13 public class Game implements Externalizable {
14
15     private Team teamA;
16     private Team teamB;
17
18     private IntegerProperty id;
19     private BooleanProperty gameFinished;
20     private IntegerProperty goalsTeamA;
21     private IntegerProperty goalsTeamB;
22
23
24     public Game() {
25         this.id = new SimpleIntegerProperty(-1);
26         this.gameFinished = new SimpleBooleanProperty(false);
27         this.goalsTeamA = new SimpleIntegerProperty(-1);
28         this.goalsTeamB = new SimpleIntegerProperty(-1);
29     }
30
31     public Game(Team teamA, Team teamB) {
32         this();
33
34         this.setTeamA(teamA);
35         this.setTeamB(teamB);
36     }
37
38     public Game(Integer id, Team teamA, Team teamB, Boolean gameFinished, Integer
39         ↪ goalsTeamA, Integer goalsTeamB) {
40         this(teamA, teamB);
41
42         this.id.set(id);
43         this.gameFinished.set(gameFinished);
44         this.goalsTeamA.set(goalsTeamA);
45         this.goalsTeamB.set(goalsTeamB);
46     }
47
48     public void clone(Game game) {
49         if(game != null) {
50             this.setId(game.getId());
51             this.setTeamA(game.getTeamA());
52             this.setTeamB(game.getTeamB());
```

```
53         this.setGameFinished(game.getGameFinished());
54         this.setGoalsTeamA(game.getGoalsTeamA());
55         this.setGoalsTeamB(game.getGoalsTeamB());
56
57     } else {
58         this.setId(-1);
59         this.setTeamA(new Team());
60         this.setTeamB(new Team());
61         this.setGameFinished(false);
62         this.setGoalsTeamA(-1);
63         this.setGoalsTeamB(-1);
64     }
65 }
66
67
68 public Team getTeamA() {
69     return teamA;
70 }
71 public void setTeamA(Team teamA) {
72     this.teamA = teamA;
73 }
74 public Team getTeamB() {
75     return teamB;
76 }
77 public void setTeamB(Team teamB) {
78     this.teamB = teamB;
79 }
80
81
82 public IntegerProperty getIdProperty() {
83     return id;
84 }
85
86 public final Integer getId() {
87     return id.get();
88 }
89
90 public final void setId(Integer id) {
91     this.id.set(id);
92 }
93
94
95
96 public IntegerProperty getGoalsTeamAProperty() {
97     return goalsTeamA;
98 }
99
100 public final Integer getGoalsTeamA() {
101     return goalsTeamA.get();
102 }
103
104 public final void setGoalsTeamA(Integer goals) {
105     this.goalsTeamA.set(goals);
```

```
106     }
107
108
109
110     public IntegerProperty getGoalsTeamBProperty() {
111         return goalsTeamB;
112     }
113
114     public final Integer getGoalsTeamB() {
115         return goalsTeamB.get();
116     }
117
118     public final void setGoalsTeamB(Integer goals) {
119         this.goalsTeamB.set(goals);
120     }
121
122
123
124
125     public BooleanProperty getGameFinishedProperty() {
126         return gameFinished;
127     }
128
129     public final Boolean getGameFinished() {
130         return gameFinished.get();
131     }
132
133     public final void setGameFinished(Boolean finished) {
134         this.gameFinished.set(finished);
135     }
136
137
138
139
140     public String getResult() {
141         if(getGameFinished() && goalsTeamA != null && goalsTeamB != null) {
142             return getGoalsTeamA().toString() + " : " + getGoalsTeamB().
143                 ↪ toString();
144         }
145         return "- : - ";
146
147
148     @Override
149     public String toString() {
150         return "id: " + this.getId() + " "
151             + this.teamA.getName() + " - " + this.teamB.getName().
152                 ↪ toString()
153             + " " + this.getGoalsTeamA() + ": " + this.getGoalsTeamB
154                 ↪ ()
155             + " finished: " + this.getGameFinished();
156     }
```



```
156
157
158     @Override
159     public void writeExternal(ObjectOutput out) throws IOException {
160         out.writeObject(getTeamA());
161         out.writeObject(getTeamB());
162         out.writeInt(getId());
163         out.writeBoolean(getGameFinished());
164         out.writeInt(getGoalsTeamA());
165         out.writeInt(getGoalsTeamB());
166     }
167
168     @Override
169     public void readExternal(ObjectInput in) throws IOException,
170         ↪ ClassNotFoundException {
171         setTeamA((Team) in.readObject());
172         setTeamB((Team) in.readObject());
173         setId(in.readInt());
174         setGameFinished(in.readBoolean());
175         setGoalsTeamA(in.readInt());
176         setGoalsTeamB(in.readInt());
177     }
178 }
```

## Team.java

```
1 package swe4.models;
2
3 import java.io.Externalizable;
4 import java.io.IOException;
5 import java.io.ObjectInput;
6 import java.io.ObjectOutput;
7 import javafx.beans.property.IntegerProperty;
8 import javafx.beans.property.SimpleIntegerProperty;
9 import javafx.beans.property.SimpleStringProperty;
10 import javafx.beans.property.StringProperty;
11
12 public class Team implements Externalizable {
13
14     private IntegerProperty id;
15     private StringProperty name;
16     private StringProperty group;
17
18
19     public Team() {
20         this.id = new SimpleIntegerProperty(-1);
21         this.name = new SimpleStringProperty("");
22         this.group = new SimpleStringProperty("");
23     }
```

```
24
25     public Team(Integer id, String name, String group) {
26         this();
27
28         this.id.set(id);
29         this.name.set(name);
30         this.group.set(group);
31     }
32
33
34     public void clone(Team team) {
35         if(team != null) {
36             this.setId(team.getId());
37             this.setName(team.getName());
38             this.setGroup(team.getGroup());
39         } else {
40             this.setId(-1);
41             this.setName("");
42             this.setGroup("");
43         }
44     }
45
46
47
48     public IntegerProperty getIdProperty() {
49         return id;
50     }
51
52     public final Integer getId() {
53         return id.get();
54     }
55
56     public final void setId(Integer id) {
57         this.id.set(id);
58     }
59
60
61
62     public StringProperty getNameProperty() {
63         return name;
64     }
65
66     public final String getName() {
67         return name.get();
68     }
69
70     public final void setName(String name) {
71         this.name.set(name);
72     }
73
74
75
76     public StringProperty getGroupProperty() {
```

```
77         return group;
78     }
79
80     public final String getGroup() {
81         return group.get();
82     }
83
84     public final void setGroup(String group) {
85         this.group.set(group);
86     }
87
88
89     @Override
90     public String toString() {
91
92         return "id: " + id.getValue() +
93                " name: " + name.getValue() +
94                " group: " + group.getValue();
95     }
96
97
98
99     @Override
100    public void writeExternal(ObjectOutput out) throws IOException {
101        out.writeInt(getId());
102        out.writeObject(getName());
103        out.writeObject(getGroup());
104    }
105
106    @Override
107    public void readExternal(ObjectInput in) throws IOException,
108        ↪ ClassNotFoundException {
109        setId(in.readInt());
110        setName((String) in.readObject());
111        setGroup((String) in.readObject());
112    }
```

## User.java

```
1 package swe4.models;
2
3 import java.io.Externalizable;
4 import java.io.IOException;
5 import java.io.ObjectInput;
6 import java.io.ObjectOutput;
7 import javafx.beans.property.BooleanProperty;
8 import javafx.beans.property.IntegerProperty;
9 import javafx.beans.property.SimpleBooleanProperty;
10 import javafx.beans.property.SimpleIntegerProperty;
```

```
11 import javafx.beans.property.SimpleStringProperty;
12 import javafx.beans.property.StringProperty;
13
14 public class User implements Externalizable {
15
16
17     private IntegerProperty id;
18     private BooleanProperty active;
19     private StringProperty name;
20     private StringProperty password;
21
22
23     public User() {
24         this.id = new SimpleIntegerProperty(-1);
25         this.active = new SimpleBooleanProperty(false);
26         this.name = new SimpleStringProperty("");
27         this.password = new SimpleStringProperty("");
28     }
29
30     public User(Integer id, Boolean active, String name, String password) {
31         this();
32
33         this.id.set(id);
34         this.active.set(active);
35         this.name.set(name);
36         this.password.set(password);
37     }
38
39
40     public void clone(User user) {
41         if(user != null) {
42             this.setId(user.getId());
43             this.setActive(user.getActive());
44             this.setName(user.getName());
45             this.setPassword(user.getPassword());
46         } else {
47             this.setId(-1);
48             this.setActive(false);
49             this.setName("");
50             this.setPassword("");
51         }
52     }
53
54     public IntegerProperty getIdProperty() {
55         return id;
56     }
57
58     public final Integer getId() {
59         return id.get();
60     }
61
62     public final void setId(Integer id) {
63         this.id.set(id);
```

```
64     }
65
66
67
68     public BooleanProperty getActiveProperty() {
69         return active;
70     }
71
72     public final Boolean getActive() {
73         return active.get();
74     }
75
76     public final void setActive(Boolean active) {
77         this.active.set(active);
78     }
79
80
81
82     public StringProperty getNameProperty() {
83         return name;
84     }
85
86     public final String getName() {
87         return name.get();
88     }
89
90     public final void setName(String name) {
91         this.name.set(name);
92     }
93
94
95
96
97     public StringProperty getPasswordProperty() {
98         return password;
99     }
100
101     public final String getPassword() {
102         return password.get();
103     }
104
105     public final void setPassword(String password) {
106         this.password.set(password);
107     }
108
109
110     @Override
111     public String toString() {
112
113         return "id: " + id.getValue() +
114             " name: " + name.getValue() +
115             " password: " + password.getValue() +
116             " active: " + active.getValue();
```

```
117     }
118
119     @Override
120     public void writeExternal(ObjectOutput out) throws IOException {
121         out.writeInt(getId());
122         out.writeBoolean(getActive());
123         out.writeObject(getName());
124         out.writeObject(getPassword());
125     }
126
127     @Override
128     public void readExternal(ObjectInput in) throws IOException,
129         ↪ ClassNotFoundException {
130         setId(in.readInt());
131         setActive(in.readBoolean());
132         setName((String) in.readObject());
133         setPassword((String) in.readObject());
134     }
```

## Bet.java

```
1 package swe4.models;
2
3 import java.io.Externalizable;
4 import java.io.IOException;
5 import java.io.ObjectInput;
6 import java.io.ObjectOutput;
7
8 import javafx.beans.property.BooleanProperty;
9 import javafx.beans.property.IntegerProperty;
10 import javafx.beans.property.SimpleBooleanProperty;
11 import javafx.beans.property.SimpleIntegerProperty;
12
13 public class Bet implements Externalizable {
14
15     private User user;
16     private Team teamA;
17     private Team teamB;
18     private IntegerProperty id;
19     private IntegerProperty goalsTeamA;
20     private IntegerProperty goalsTeamB;
21
22
23     public Bet() {
24         this.id = new SimpleIntegerProperty(-1);
25         this.goalsTeamA = new SimpleIntegerProperty(-1);
26         this.goalsTeamB = new SimpleIntegerProperty(-1);
27     }
28 }
```

```
29     public Bet(User user, Team teamA, Team teamB) {
30         this();
31
32         this.setUser(user);
33         this.setTeamA(teamA);
34         this.setTeamB(teamB);
35     }
36
37     public Bet(Integer id, User user, Team teamA, Team teamB, Integer goalsTeamA,
38         ↪ Integer goalsTeamB) {
39         this(user, teamA, teamB);
40
41         this.id.set(id);
42         this.goalsTeamA.set(goalsTeamA);
43         this.goalsTeamB.set(goalsTeamB);
44     }
45
46     public void clone(Bet game) {
47         if(game != null) {
48             this.setId(game.getId());
49             this.setTeamA(game.getTeamA());
50             this.setTeamB(game.getTeamB());
51             this.setGoalsTeamA(game.getGoalsTeamA());
52             this.setGoalsTeamB(game.getGoalsTeamB());
53
54         } else {
55             this.setId(-1);
56             this.setTeamA(new Team());
57             this.setTeamB(new Team());
58             this.setGoalsTeamA(-1);
59             this.setGoalsTeamB(-1);
60         }
61     }
62
63     public User getUser() {
64         return user;
65     }
66
67     public void setUser(User user) {
68         this.user = user;
69     }
70
71     public Team getTeamA() {
72         return teamA;
73     }
74     public void setTeamA(Team teamA) {
75         this.teamA = teamA;
76     }
77     public Team getTeamB() {
78         return teamB;
79     }
80     public void setTeamB(Team teamB) {
```

```
81         this.teamB = teamB;
82     }
83
84
85     public IntegerProperty getIdProperty() {
86         return id;
87     }
88
89     public final Integer getId() {
90         return id.get();
91     }
92
93     public final void setId(Integer id) {
94         this.id.set(id);
95     }
96
97
98
99     public IntegerProperty getGoalsTeamAProperty() {
100         return goalsTeamA;
101     }
102
103     public final Integer getGoalsTeamA() {
104         return goalsTeamA.get();
105     }
106
107     public final void setGoalsTeamA(Integer goals) {
108         this.goalsTeamA.set(goals);
109     }
110
111
112
113     public IntegerProperty getGoalsTeamBProperty() {
114         return goalsTeamB;
115     }
116
117     public final Integer getGoalsTeamB() {
118         return goalsTeamB.get();
119     }
120
121     public final void setGoalsTeamB(Integer goals) {
122         this.goalsTeamB.set(goals);
123     }
124
125
126
127     @Override
128     public String toString() {
129         return "id: " + this.getId() + " "
130             + this.teamA.getName() + " - " + this.teamB.getName().
131                 ↪ toString()
132             + " " + this.getGoalsTeamA() + ":" + this.getGoalsTeamB
133                 ↪ ()
```



```
132             + " user: " + this.getUser();
133         }
134
135
136
137         @Override
138         public void writeExternal(ObjectOutput out) throws IOException {
139             out.writeObject(getUser());
140             out.writeObject(getTeamA());
141             out.writeObject(getTeamB());
142             out.writeInt(getId());
143             out.writeInt(getGoalsTeamA());
144             out.writeInt(getGoalsTeamB());
145         }
146
147         @Override
148         public void readExternal(ObjectInput in) throws IOException,
149             ↪ ClassNotFoundException {
150             setUser((User) in.readObject());
151             setTeamA((Team) in.readObject());
152             setTeamB((Team) in.readObject());
153             setId(in.readInt());
154             setGoalsTeamA(in.readInt());
155             setGoalsTeamB(in.readInt());
156         }
157     }
```

### 1.2.3 swe4.gui.panels

#### GameDetailPanel.java

```
1 package swe4.gui.panels;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import javafx.beans.binding.Bindings;
6 import javafx.beans.binding.BooleanBinding;
7 import javafx.collections.FXCollections;
8 import javafx.collections.ObservableList;
9 import javafx.geometry.Insets;
10 import javafx.scene.control.ComboBox;
11 import javafx.scene.control.Label;
12 import javafx.scene.layout.VBox;
13 import swe4.gui.components.PanelComponent;
14 import swe4.models.Game;
15 import swe4.models.Team;
16 import swe4.rmi.client.EuroBetDbFactory;
17
18 public class GameDetailPanel extends PanelComponent {
```

```
19
20
21     private ComboBox<String> teamABox;
22     private ComboBox<String> teamBBox;
23
24     private BooleanBinding disableSaveCondition;
25     private Game editedGame = new Game();
26     private List<Team> allTeams;
27
28     public GameDetailPanel() {
29         super("Details");
30
31
32         createDetailFields();
33
34         disableSaveCondition = Bindings.or(
35             teamABox.getSelectionModel().selectedItemProperty().
36                 ↪ isEqualTo(new Team().getNameProperty()),
37             teamBBox.getSelectionModel().selectedItemProperty().
38                 ↪ isEqualTo(new Team().getNameProperty())
39         );
40
41         addHeaderButtonRight("game-save-button", "/save.png",
42             ↪ disableSaveCondition, event -> {
43                 // save or update game
44
45                 Integer teamASelectionIndex = teamABox.getSelectionModel().
46                     ↪ getSelectedIndex();
47                 editedGame.setTeamA(allTeams.get(teamASelectionIndex));
48
49                 Integer teamBSelectionIndex = teamBBox.getSelectionModel().
50                     ↪ getSelectedIndex();
51                 editedGame.setTeamB(allTeams.get(teamBSelectionIndex));
52
53                 if(editedGame.getId() != -1) {
54                     // update game
55
56                     EuroBetDbFactory.getInstance().updateGame(editedGame);
57
58                     System.out.println("Existing game updated: " +
59                         ↪ editedGame.toString());
60                 } else {
61                     // save new game
62
63                     Integer newId = EuroBetDbFactory.getInstance().addGame(
64                         ↪ editedGame);
65                     editedGame.setId(newId);
66                     System.out.println("New game saved: " + editedGame.
67                         ↪ toString());
68                 }
69             }
70         );
71     }
72 }
```

```
64         }
65
66     });
67
68
69
70 }
71
72 private void createDetailFields() {
73
74     VBox details = new VBox();
75     details.setSpacing(50);
76     details.setId("game-detail-container");
77
78
79     allTeams = EuroBetDbFactory.getInstance().getTeams();
80     List<String> teamNames = new ArrayList<>();
81     for(Team t: allTeams) {
82         teamNames.add(t.getName());
83     }
84
85
86
87     // team a
88     VBox groupA = new VBox();
89     groupA.setSpacing(3);
90     Label groupATitle = new Label("Team A:");
91
92     ObservableList<String> optionsA = FXCollections.observableList(teamNames);
93
94
95     teamABox = new ComboBox<String>(optionsA);
96
97
98     teamABox.setMinWidth(340);
99     groupA.getChildren().addAll(groupATitle, teamABox);
100    VBox.setMargin(groupA, new Insets(60, 30, 0, 30));
101
102
103    // team b
104    VBox groupB = new VBox();
105    groupB.setSpacing(3);
106    Label groupBTitle = new Label("Team B:");
107    ObservableList<String> optionsB = FXCollections.observableList(
108        ↪ teamNames);
109
110    teamBBox = new ComboBox<>(optionsB);
111    teamBBox.setMinWidth(340);
112    groupB.getChildren().addAll(groupBTitle, teamBBox);
113    VBox.setMargin(groupB, new Insets(0, 30, 0, 30));
114
115
```

```
116         details.getChildren().addAll(  
117             groupA, groupB  
118         );  
119  
120         this.createNewDetail();  
121  
122         addContent(details);  
123  
124     }  
125  
126     public void setDetails(Game game) {  
127         editedGame.clone(game);  
128         teamABox.getSelectionModel().select(game.getTeamA().getName());  
129         teamBBox.getSelectionModel().select(game.getTeamB().getName());  
130     }  
131  
132     public void createNewDetail() {  
133         editedGame.clone(new Game());  
134         editedGame.setTeamA(new Team());  
135         editedGame.setTeamB(new Team());  
136         teamABox.getSelectionModel().select("");  
137         teamBBox.getSelectionModel().select("");  
138     }  
139 }
```

## GamesPanel.java

```
1 package swe4.gui.panels;  
2  
3 import java.util.ArrayList;  
4 import java.util.List;  
5  
6 import javafx.beans.value.ChangeListener;  
7 import javafx.beans.value.ObservableValue;  
8 import javafx.collections.FXCollections;  
9 import javafx.collections.ObservableList;  
10 import javafx.scene.control.ListCell;  
11 import javafx.scene.control.ListView;  
12 import javafx.scene.layout.Border;  
13 import javafx.scene.layout.BorderStroke;  
14 import javafx.scene.layout.BorderStrokeStyle;  
15 import javafx.scene.paint.Color;  
16 import javafx.util.Callback;  
17 import swe4.gui.components.PanelComponent;  
18 import swe4.models.Game;  
19 import swe4.rmi.client.EuroBetDbFactory;  
20 import swe4.rmi.client.GameAddedListener;  
21 import swe4.rmi.client.GameUpdatedListener;  
22  
23
```

```
24 public class GamesPanel extends PanelComponent{
25
26     private static final Border LIST_BORDER = new Border(
27         new BorderStroke(Color.TRANSPARENT, BorderStrokeStyle.NONE,
28             ↪ null, null)
29     );
30     private ListView<Game> gamesListView;
31
32     private List<GamesPanelListener> listeners = new ArrayList<GamesPanelListener
33         ↪ >();
34
35     public GamesPanel() {
36         super("All Games");
37
38         addHeaderButtonLeft("games-add-button", "/plus.png", event -> {
39             // call all added listeners
40             for (GamesPanelListener l : listeners) {
41                 l.onAddNewGame();
42             }
43         });
44
45         createGamesList();
46
47         EuroBetDbFactory.getInstance().addGamesAddedListener(new
48             ↪ GameAddedListener() {
49
50             @Override
51             public void onGameAdded(Game game) {
52                 gamesListView = null;
53                 createGamesList();
54             }
55         });
56
57         EuroBetDbFactory.getInstance().addGamesUpdatedListener(new
58             ↪ GameUpdatedListener() {
59
60             @Override
61             public void onGameUpdated(Game game) {
62                 gamesListView = null;
63                 createGamesList();
64             }
65         });
66     }
67
68     private void createGamesList() {
69         gamesListView = new ListView<Game>();
70         gamesListView.setBorder(LIST_BORDER);
71
72     }
```

```
73
74         List<Game> games = EuroBetDbFactory.getInstance().getGames();
75         ObservableList<Game> items = FXCollections.observableList(games);
76         gamesListView.setItems(items);
77
78         gamesListView.setCellFactory(new Callback<ListView<Game>, ListCell<Game>>(){
79
80             @Override
81             public ListCell<Game> call(ListView<Game> p) {
82
83                 ListCell<Game> cell = new ListCell<Game>(){
84
85                     @Override
86                     protected void updateItem(Game t, boolean bln) {
87                         super.updateItem(t, bln);
88                         if (t != null) {
89                             setText(t.getTeamA().getName() + " - " + t.getTeamB().
90                                 ↪ getName());
91                         }
92                     };
93                     return cell;
94                 }
95             });
96
97         gamesListView.getSelectionModel().selectedItemProperty()
98             .addListener(new ChangeListener<Game>() {
99
100             @Override
101             public void changed(ObservableValue<? extends Game> observable,
102                 Game oldValue, Game newValue) {
103
104                 // call all added listeners
105                 for (GamesPanelListener l : listeners) {
106                     l.onGameSelected(newValue);
107                 }
108             }
109         });
110
111         addContent(gamesListView);
112     }
113
114
115     public void addListener(GamesPanelListener listener) {
116         listeners.add(listener);
117     }
118
119
120 }
```

### GamesPanelListener.java

```
1 package swe4.gui.panels;
2
3 import swe4.models.Game;
4
5 public interface GamesPanelListener {
6     void onGameSelected(Game game);
7     void onAddNewGame();
8 }
```

## ResultDetailPanel.java

```
1 package swe4.gui.panels;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import javafx.collections.FXCollections;
6 import javafx.collections.ObservableList;
7 import javafx.geometry.Insets;
8 import javafx.scene.control.ComboBox;
9 import javafx.scene.control.Label;
10 import javafx.scene.layout.VBox;
11 import swe4.gui.NumberTextField;
12 import swe4.gui.components.PanelComponent;
13 import swe4.models.Game;
14 import swe4.rmi.client.EuroBetDbFactory;
15
16 public class ResultDetailPanel extends PanelComponent {
17
18
19     private NumberTextField goalsTeamABox;
20     private NumberTextField goalsTeamBBox;
21     private ComboBox<String> gameBox;
22     private Game editedGame = new Game();
23     private List<Game> allGames;
24
25     public ResultDetailPanel() {
26         super("Details");
27
28         addHeaderButtonRight("result-save-button", "/save.png", null, event ->
29             ↪ {
30                 // save or update game
31
32                 Integer gameSelectionIndex = gameBox.getSelectionModel().
33                     ↪ getSelectedIndex();
34
35                 Game selectedGame = allGames.get(gameSelectionIndex);
36
37                 editedGame.setTeamA(selectedGame.getTeamA());
38                 editedGame.setTeamB(selectedGame.getTeamB());
39             }
```

```
37
38         editedGame.setGoalsTeamA(Integer.parseInt(goalsTeamABox.getText
39             ↪ ()));
40         editedGame.setGoalsTeamB(Integer.parseInt(goalsTeamBBox.getText
41             ↪ ()));
42         editedGame.setGameFinished(true);
43
44         if(editedGame.getId() != -1) {
45             // update game
46
47             EuroBetDbFactory.getInstance().updateGame(editedGame);
48
49             System.out.println("Existing game updated: " +
50                 ↪ editedGame.toString());
51         } else {
52             // save new game
53
54             Integer newId = EuroBetDbFactory.getInstance().addGame(
55                 ↪ editedGame);
56             editedGame.setId(newId);
57             System.out.println("New game saved: " + editedGame.
58                 ↪ toString());
59         }
60     });
61
62     createDetailFields();
63
64     private void createDetailFields() {
65
66         VBox details = new VBox();
67         details.setSpacing(50);
68         details.setId("result-detail-container");
69
70
71         allGames = EuroBetDbFactory.getInstance().getGames();
72         List<String> gameNames = new ArrayList<>();
73         for(Game t: allGames) {
74             gameNames.add(t.getTeamA().getName() + " - " + t.getTeamB().
75                 ↪ getName());
76         }
77
78         // games
79         VBox game = new VBox();
80         game.setSpacing(3);
81         Label gameTitle = new Label("Game:");
82
83         ObservableList<String> optionsB = FXCollections.observableList(
84             ↪ gameNames);
```



```
83         gameBox = new ComboBox<String>(optionsB);
84         gameBox.setMinWidth(340);
85         game.getChildren().addAll(gameTitle, gameBox);
86         VBox.setMargin(game, new Insets(30, 30, 0, 30));
87
88
89         // goals team a
90         VBox goalsA = new VBox();
91         goalsA.setSpacing(3);
92         Label goalsATitle = new Label("Goals Team A:");
93         goalsTeamABox = new NumberTextField ();
94         goalsA.getChildren().addAll(goalsATitle, goalsTeamABox);
95         VBox.setMargin(goalsA, new Insets(0, 30, 0, 30));
96
97
98         // goals team b
99         VBox goalsB = new VBox();
100        goalsB.setSpacing(3);
101        Label goalsBTitle = new Label("Goals Team B:");
102        goalsTeamBBox = new NumberTextField ();
103        goalsA.getChildren().addAll(goalsBTitle, goalsTeamBBox);
104        VBox.setMargin(goalsB, new Insets(0, 30, 0, 30));
105
106        details.getChildren().addAll(
107            game, goalsA, goalsB
108        );
109
110        addContent(details);
111
112    }
113
114    public void setDetails(Game game) {
115
116        editedGame.clone(game);
117
118        gameBox.getSelectionModel().select(game.getTeamA().getName() + " - " +
119            ↪ game.getTeamB().getName());
120
121        if(game.getGameFinished()) {
122            goalsTeamABox.setText(game.getGoalsTeamA().toString());
123            goalsTeamBBox.setText(game.getGoalsTeamB().toString());
124        } else {
125            goalsTeamABox.setText(" ");
126            goalsTeamBBox.setText(" ");
127        }
128    }
129
130 }
```

## ResultsPanel.java

```
1 package swe4.gui.panels;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import javafx.beans.value.ChangeListener;
7 import javafx.beans.value.ObservableValue;
8 import javafx.collections.FXCollections;
9 import javafx.collections.ObservableList;
10 import javafx.scene.control.ListCell;
11 import javafx.scene.control.ListView;
12 import javafx.scene.layout.Border;
13 import javafx.scene.layout.BorderStroke;
14 import javafx.scene.layout.BorderStrokeStyle;
15 import javafx.scene.paint.Color;
16 import javafx.util.Callback;
17 import swe4.gui.components.PanelComponent;
18 import swe4.models.Game;
19 import swe4.rmi.client.EuroBetDbFactory;
20 import swe4.rmi.client.GameAddedListener;
21 import swe4.rmi.client.GameUpdatedListener;
22
23
24 public class ResultsPanel extends PanelComponent{
25
26     private static final Border LIST_BORDER = new Border(
27         new BorderStroke(Color.TRANSPARENT, BorderStrokeStyle.NONE,
28             ↪ null, null)
29     );
30
31     private ListView<Game> gamesListView;
32
33     private List<ResultsPanelListener> listeners = new ArrayList<
34         ↪ ResultsPanelListener>();
35
36     public ResultsPanel() {
37         super("All Results");
38
39         createGamesList();
40
41
42         EuroBetDbFactory.getInstance().addGamesAddedListener(new
43             ↪ GameAddedListener() {
44
45             @Override
46             public void onGameAdded(Game game) {
47                 gamesListView = null;
48                 createGamesList();
49             }
50         });
51     }
```

```
50
51     EuroBetDbFactory.getInstance().addGamesUpdatedListener(new
52         ↪ GameUpdatedListener() {
53
54         @Override
55         public void onGameUpdated(Game game) {
56             gamesListView = null;
57             createGamesList();
58         }
59     });
60 }
61
62
63 private void createGamesList() {
64     gamesListView = new ListView<Game>();
65     gamesListView.setBorder(LIST_BORDER);
66
67     List<Game> games = EuroBetDbFactory.getInstance().getGames();
68
69
70     ObservableList<Game> items = FXCollections.observableList(games);
71
72
73
74     gamesListView.setItems(items);
75
76
77     gamesListView.setCellFactory(new Callback<ListView<Game>, ListCell<Game>>(){
78
79         @Override
80         public ListCell<Game> call(ListView<Game> p) {
81
82             ListCell<Game> cell = new ListCell<Game>(){
83
84                 @Override
85                 protected void updateItem(Game g, boolean bln) {
86                     super.updateItem(g, bln);
87                     if (g != null) {
88                         setText(g.getResult() + "      " + g.getTeamA().getName()
89                             ↪ + " - " + g.getTeamB().getName());
90                     }
91                 }
92             };
93             return cell;
94         });
95
96     gamesListView.getSelectionModel().selectedItemProperty()
97         .addListener(new ChangeListener<Game>() {
98
99         @Override
100         public void changed(ObservableValue<? extends Game> observable,
```

```
101         Game oldValue, Game newValue) {
102
103         // call all added listeners
104         for (ResultsPanelListener l : listeners) {
105             l.onResultSelected(newValue);
106         }
107     }
108 });
109
110     addContent(gamesListView);
111 }
112
113
114     public void addListener(ResultsPanelListener listener) {
115         listeners.add(listener);
116     }
117
118
119 }
```

### ResultsPanelListener.java

```
1 package swe4.gui.panels;
2
3 import swe4.models.Game;
4
5 public interface ResultsPanelListener {
6     void onResultSelected(Game game);
7 }
```

### TeamDetailPanel.java

```
1 package swe4.gui.panels;
2
3 import java.util.List;
4
5 import javafx.beans.binding.Bindings;
6 import javafx.beans.binding.BooleanBinding;
7 import javafx.collections.FXCollections;
8 import javafx.collections.ObservableList;
9 import javafx.geometry.Insets;
10 import javafx.scene.control.ComboBox;
11 import javafx.scene.control.Label;
12 import javafx.scene.control.TextField;
13 import javafx.scene.layout.VBox;
14 import swe4.gui.components.PanelComponent;
```

```
15 import swe4.models.Team;
16 import swe4.rmi.client.EuroBetDbFactory;
17 import swe4.rmi.client.TeamUpdatedListener;
18
19 public class TeamDetailPanel extends PanelComponent {
20
21
22     private TextField nameField;
23     private ComboBox<String> groupBox;
24     private BooleanBinding disableSaveCondition;
25     private Team editedTeam = new Team();
26
27
28     public TeamDetailPanel() {
29         super("Details");
30
31
32
33         createDetailFields();
34
35
36         disableSaveCondition = Bindings.and(
37             nameField.textProperty().isEmpty(),
38             nameField.textProperty().isEmpty()
39         );
40
41
42         addHeaderButtonRight("team-save-button", "/save.png",
43             ↪ disableSaveCondition, event -> {
44
45                 // save or update team
46
47                 if(editedTeam.getId() != -1) {
48                     // update team
49
50                     EuroBetDbFactory.getInstance().updateTeam(editedTeam);
51
52                     System.out.println("Existing team updated: " +
53                         ↪ editedTeam.toString());
54                 } else {
55                     // save new user
56
57                     Integer newId = EuroBetDbFactory.getInstance().addTeam(
58                         ↪ editedTeam);
59                     editedTeam.setId(newId);
60                     System.out.println("New team saved: " + editedTeam.
61                         ↪ toString());
62                 }
63             });
64 }
```

```
63         EuroBetDbFactory.getInstance().addTeamUpdatedListener(new
           ↪ TeamUpdatedListener() {
64
65             @Override
66             public void onTeamUpdated(Team team) {
67                 if(editedTeam.getId() == team.getId()) {
68                     // other client has changed current user
69                     editedTeam.clone(team);
70                 }
71             }
72         });
73     }
74
75     private void createDetailFields() {
76
77         VBox details = new VBox();
78         details.setSpacing(20);
79         details.setId("team-detail-container");
80
81
82         // team name
83         VBox username = new VBox();
84         username.setSpacing(3);
85         Label nameTitle = new Label("Country:");
86         nameField = new TextField();
87         nameField.textProperty().bindBidirectional(editedTeam.getNameProperty()
           ↪ );
88         username.getChildren().addAll(nameTitle, nameField);
89         VBox.setMargin(username, new Insets(30, 30, 0, 30));
90
91
92         // team group
93         VBox group = new VBox();
94         group.setSpacing(3);
95         Label groupTitle = new Label("Group:");
96         List<String> groups = EuroBetDbFactory.getInstance().getGroups();
97         ObservableList<String> options = FXCollections.observableList(groups);
98         groupBox = new ComboBox<String>(options);
99         groupBox.valueProperty().bindBidirectional(editedTeam.getGroupProperty()
100            ↪ );
101         group.getChildren().addAll(groupTitle, groupBox);
102         VBox.setMargin(group, new Insets(0, 30, 0, 30));
103
104
105         details.getChildren().addAll(
106             username, group
107         );
108
109
110         addContent(details);
111
112     }
```

```
113
114     public void setDetails(Team team) {
115
116         editedTeam.clone(team);
117     }
118
119     public void createNewDetail() {
120         editedTeam.clone(new Team());
121     }
122
123
124 }
```

### TeamsPanel.java

```
1 package swe4.gui.panels;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import javafx.beans.value.ChangeListener;
7 import javafx.beans.value.ObservableValue;
8 import javafx.collections.FXCollections;
9 import javafx.collections.ObservableList;
10 import javafx.scene.control.ListCell;
11 import javafx.scene.control.ListView;
12 import javafx.scene.layout.Border;
13 import javafx.scene.layout.BorderStroke;
14 import javafx.scene.layout.BorderStrokeStyle;
15 import javafx.scene.paint.Color;
16 import javafx.util.Callback;
17 import swe4.gui.components.PanelComponent;
18 import swe4.models.Team;
19 import swe4.rmi.client.EuroBetDbFactory;
20 import swe4.rmi.client.TeamAddedListener;
21 import swe4.rmi.client.TeamUpdatedListener;
22
23
24 public class TeamsPanel extends PanelComponent{
25
26     private static final Border LIST_BORDER = new Border(
27         new BorderStroke(Color.TRANSPARENT, BorderStrokeStyle.NONE,
28             ↪ null, null)
29     );
30
31     private ListView<Team> teamListView;
32
33     private List<TeamsPanelListener> listeners = new ArrayList<TeamsPanelListener
34         ↪ >();
35
36     public TeamsPanel() {
```

```
34         super("All Teams");
35
36
37         addHeaderButtonLeft("teams-add-button", "/plus.png", event -> {
38             // call all added listeners
39             for (TeamsPanelListener l : listeners) {
40                 l.onAddNewTeam();
41             }
42         });
43
44
45         createTeamsList();
46
47         EuroBetDbFactory.getInstance().addTeamAddedListener(new
48             ↪ TeamAddedListener() {
49
50             @Override
51             public void onTeamAdded(Team team) {
52                 teamListView = null;
53                 createTeamsList();
54             }
55         });
56
57         EuroBetDbFactory.getInstance().addTeamUpdatedListener(new
58             ↪ TeamUpdatedListener() {
59
60             @Override
61             public void onTeamUpdated(Team team) {
62                 teamListView = null;
63                 createTeamsList();
64             }
65         });
66
67
68         private void createTeamsList() {
69             teamListView = new ListView<Team>();
70             teamListView.setBorder(LIST_BORDER);
71
72             List<Team> teams = EuroBetDbFactory.getInstance().getTeams();
73
74
75
76             ObservableList<Team> items = FXCollections.observableList(teams);
77
78
79             teamListView.setItems(items);
80
81
82             teamListView.setCellFactory(new Callback<ListView<Team>, ListCell<Team>>(){
83
84                 @Override
```



```
85         public ListCell<Team> call(ListView<Team> p) {
86
87             ListCell<Team> cell = new ListCell<Team>(){
88
89                 @Override
90                 protected void updateItem(Team t, boolean bln) {
91                     super.updateItem(t, bln);
92                     if (t != null) {
93                         setText(t.getName());
94                     }
95                 }
96             };
97             return cell;
98         }
99     });
100
101     teamListView.getSelectionModel().selectedItemProperty()
102         .addListener(new ChangeListener<Team>() {
103
104         @Override
105         public void changed(ObservableValue<? extends Team> observable,
106                             Team oldValue, Team newValue) {
107
108             // call all added listeners
109             for (TeamsPanelListener l : listeners) {
110                 l.onTeamSelected(newValue);
111             }
112         }
113     });
114
115     addContent(teamListView);
116 }
117
118
119 public void addListener(TeamsPanelListener listener) {
120     listeners.add(listener);
121 }
122
123
124 }
```

### TeamsPanelListener.java

```
1 package swe4.gui.panels;
2
3 import swe4.models.Team;
4
5 public interface TeamsPanelListener {
6     void onTeamSelected(Team team);
7     void onAddNewTeam();
8 }
```

```
8 }
```

## UserDetailPanel.java

```
1 package swe4.gui.panels;
2
3 import java.io.Serializable;
4 import java.rmi.Remote;
5 import javafx.beans.binding.Bindings;
6 import javafx.beans.binding.BooleanBinding;
7 import javafx.geometry.Insets;
8 import javafx.scene.control.Label;
9 import javafx.scene.control.PasswordField;
10 import javafx.scene.control.TextField;
11 import javafx.scene.layout.VBox;
12 import swe4.gui.ToggleSwitch;
13 import swe4.gui.components.PanelComponent;
14 import swe4.models.User;
15 import swe4.rmi.client.EuroBetDbFactory;
16 import swe4.rmi.client.UserUpdatedListener;
17
18 public class UserDetailPanel extends PanelComponent implements Serializable, Remote {
19
20
21
22     private static final long serialVersionUID = -7267372017525019410L;
23     private ToggleSwitch switchActiv;
24     private TextField nameField;
25     private PasswordField passwordField;
26     private PasswordField confirmPassword;
27     private BooleanBinding disableSaveCondition;
28
29     private User editedUser = new User();
30
31     public UserDetailPanel() {
32         super("Details");
33
34
35         createDetailFields();
36
37         disableSaveCondition = Bindings.or(
38             passwordField.textProperty().isNotEqualTo(confirmField.
39                 ↪ textProperty()),
40             nameField.textProperty().isEmpty()
41         );
42
43         addHeaderButtonRight("users-save-button", "/save.png",
44             ↪ disableSaveCondition, event -> {
```

```
45         // save or update user
46
47         if(editedUser.getId() != -1) {
48             // update user
49
50             EuroBetDbFactory.getInstance().updateUser(editedUser);
51
52             System.out.println("Existing user updated: " +
53                 ↪ editedUser.toString());
54         } else {
55             // save new user
56
57             Integer newId = EuroBetDbFactory.getInstance().addUser(
58                 ↪ editedUser);
59             editedUser.setId(newId);
60             System.out.println("New user saved: " + editedUser.
61                 ↪ toString());
62         }
63     });
64
65     EuroBetDbFactory.getInstance().addUserUpdatedListener(new
66     ↪ UserUpdatedListener() {
67
68         @Override
69         public void onUserUpdated(User user) {
70             if(editedUser.getId() == user.getId()) {
71                 // other client has changed current user
72                 editedUser.clone(user);
73             }
74         }
75     });
76
77 }
78
79 private void createDetailFields() {
80
81     VBox details = new VBox();
82     details.setSpacing(20);
83     details.setId("user-detail-container");
84
85     // active
86     VBox active = new VBox();
87     Label activTitle = new Label("Active:");
88     switchActiv = new ToggleSwitch();
89     switchActiv.switchOnProperty().bindBidirectional(editedUser.
90     ↪ getActiveProperty());
91
92     VBox switchPane = new VBox();
```

```
93         switchPane.getChildren().add(switchActiv);
94         VBox.setMargin(switchPane, new Insets(-20, 0, 0, 250));
95         active.getChildren().addAll(activTitle, switchPane);
96         VBox.setMargin(active, new Insets(20, 30, 0, 30));
97
98
99         // username
100        VBox username = new VBox();
101        username.setSpacing(3);
102        Label nameTitle = new Label("Username:");
103        nameField = new TextField();
104        nameField.textProperty().bindBidirectional(editedUser.getNameProperty()
105            ↪ );
106        username.getChildren().addAll(nameTitle, nameField);
107        VBox.setMargin(username, new Insets(0, 30, 0, 30));
108
109        // password
110        VBox password = new VBox();
111        password.setSpacing(3);
112        Label passwordTitle = new Label("Password:");
113        passwordField = new PasswordField();
114        passwordField.textProperty().bindBidirectional(editedUser.
115            ↪ getPasswordProperty());
116        Label confirmTitle = new Label("Confirm:");
117        confirmField = new PasswordField();
118        password.getChildren().addAll(passwordTitle, passwordField,
119            confirmTitle, confirmField);
120        VBox.setMargin(password, new Insets(0, 30, 0, 30));
121
122        details.getChildren().addAll(
123            active, username, password
124        );
125
126        addContent(details);
127    }
128
129    public void setDetails(User user) {
130
131        editedUser.clone(user);
132        confirmField.setText(editedUser.getPassword());
133    }
134
135    public void createNewDetail() {
136
137        editedUser.clone(new User());
138        confirmField.setText(editedUser.getPassword());
139    }
140
141
142
143
```

```
144 }
```

## UsersPanel.java

```
1 package swe4.gui.panels;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import javafx.beans.value.ChangeListener;
6 import javafx.beans.value.ObservableValue;
7 import javafx.collections.FXCollections;
8 import javafx.collections.ObservableList;
9 import javafx.scene.control.ListCell;
10 import javafx.scene.control.ListView;
11 import javafx.scene.layout.Border;
12 import javafx.scene.layout.BorderStroke;
13 import javafx.scene.layout.BorderStrokeStyle;
14 import javafx.scene.paint.Color;
15 import javafx.util.Callback;
16 import swe4.gui.components.PanelComponent;
17 import swe4.models.User;
18 import swe4.rmi.client.EuroBetDbFactory;
19 import swe4.rmi.client.UserAddedListener;
20 import swe4.rmi.client.UserUpdatedListener;
21
22
23 public class UsersPanel extends PanelComponent {
24
25
26     private static final Border LIST_BORDER = new Border(
27         new BorderStroke(Color.TRANSPARENT, BorderStrokeStyle.NONE,
28             ↪ null, null)
29
30     );
31
32     private ListView<User> personListView;
33     private List<UsersPanelListener> listeners = new ArrayList<>();
34
35
36     public UsersPanel() {
37         super("All Users");
38
39         addHeaderButtonLeft("users-add-button", "/plus.png", event -> {
40             // call all added listeners
41             for (UsersPanelListener l : listeners) {
42                 l.onAddNewUser();
43             }
44         });
45         createUsersList();
```

```
46
47
48
49      EuroBetDbFactory.getInstance().addUserAddedListener(new
      ↪ UserAddedListener() {
50
51          @Override
52          public void onUserAdded(User user) {
53              personListView = null;
54              createUsersList();
55          }
56      });
57
58      EuroBetDbFactory.getInstance().addUserUpdatedListener(new
      ↪ UserUpdatedListener() {
59
60          @Override
61          public void onUserUpdated(User user) {
62              personListView = null;
63              createUsersList();
64          }
65      });
66  }
67
68  private void createUsersList() {
69      personListView = new ListView<User>();
70      personListView.setBorder(LIST_BORDER);
71
72
73      List<User> users = EuroBetDbFactory.getInstance().getUsers();
74
75      ObservableList<User> items = FXCollections.observableList(users);
76
77
78      personListView.setItems(items);
79
80
81      personListView.setCellFactory(new Callback<ListView<User>, ListCell<User>>(){
82
83          @Override
84          public ListCell<User> call(ListView<User> p) {
85
86              ListCell<User> cell = new ListCell<User>(){
87
88                  @Override
89                  protected void updateItem(User t, boolean bln) {
90                      super.updateItem(t, bln);
91                      if (t != null) {
92                          setText(t.getName());
93                      }
94                  }
95              };
96              return cell;
```

```
97         }
98     });
99
100    personListView.getSelectionModel().selectedItemProperty()
101        .addListener(new ChangeListener<User>() {
102
103        @Override
104        public void changed(ObservableValue<? extends User> observable,
105            User oldValue, User newValue) {
106
107            // call all added listeners
108            for (UsersPanelListener l : listeners) {
109                l.onUserSelected(newValue);
110            }
111        }
112    });
113
114    addContent(personListView);
115 }
116
117
118 public void addListener(UsersPanelListener listener) {
119     listeners.add(listener);
120 }
121
122
123 }
```

### UsersPanelListener.java

```
1 package swe4.gui.panels;
2
3 import swe4.models.User;
4
5 public interface UsersPanelListener {
6     void onUserSelected(User user);
7     void onAddNewUser();
8 }
```

#### 1.2.4 swe4.rmi.client

### EuroBetDbFactory.java

```
1 package swe4.rmi.client;
2
```

```
3
4 import java.io.Serializable;
5 import java.net.MalformedURLException;
6 import java.rmi.Naming;
7 import java.rmi.NotBoundException;
8 import java.rmi.Remote;
9 import java.rmi.RemoteException;
10 import java.util.ArrayList;
11 import java.util.List;
12
13 import swe4.models.Bet;
14 import swe4.models.Game;
15 import swe4.models.Team;
16 import swe4.models.User;
17 import swe4.rmi.server.EuroBetDb;
18 import swe4.util.Util;
19
20
21 public class EuroBetDbFactory implements Serializable, Remote {
22
23
24     private static final long serialVersionUID = 1944025441928560415L;
25     private static EuroBetDbFactory instance = null;
26     private static EuroBetDb db;
27
28     private List<UserAddedListener> userAddedListeners = new ArrayList<>();
29     private List<UserUpdatedListener> userUpdatedListeners = new ArrayList<>();
30     private List<TeamAddedListener> teamAddedListeners = new ArrayList<>();
31     private List<TeamUpdatedListener> teamUpdatedListeners = new ArrayList<>();
32     private List<GameAddedListener> gamesAddedListeners = new ArrayList<>();
33     private List<GameUpdatedListener> gamesUpdatedListeners = new ArrayList<>();
34     private List<BetAddedListener> betsAddedListeners = new ArrayList<>();
35     private List<BetUpdatedListener> betsUpdatedListeners = new ArrayList<>();
36
37
38     private EuroBetDbFactory() throws MalformedURLException, RemoteException,
39         ↪ NotBoundException {
40
41         String hostPort = Util.getHostPortArg(null);
42
43         db = (EuroBetDb) Naming.lookup("rmi://" + hostPort + "/EuroBetDb");
44     }
45
46     public static EuroBetDbFactory getInstance() {
47         if(instance == null) {
48
49             try {
50                 instance = new EuroBetDbFactory();
51                 System.out.println("Register factory: " + instance);
52                 db.registerForNotification(instance);
53             } catch (MalformedURLException | RemoteException |
54                 ↪ NotBoundException e) {
```



```
54         System.out.println("Error: Unable to connect to server!"
55                               ↪ );
56     }
57     }
58     return instance;
59 }
60 public void removeAllUsers() {
61     try {
62         db.removeAllUsers();
63     } catch (RemoteException e) {
64         System.out.println("Error: Unable to remove users from server!"
65                               ↪ );
66     }
67 }
68 public List<User> getUsers() {
69     try {
70         return db.getUsers();
71     } catch (RemoteException e) {
72         System.out.println("Error: Unable to fetch users from server!")
73                               ↪ ;
74         return null;
75     }
76 }
77 public void updateUser(User user) {
78     try {
79         db.updateUser(user);
80         onUserUpdated(user);
81     } catch (RemoteException e) {
82         System.out.println("Error: Unable to update user at server!");
83     }
84 }
85
86 public Integer addUser(User user) {
87     try {
88         Integer newId = db.addUser(user);
89         user.setId(newId);
90         onUserAdded(user);
91         return user.getId();
92     } catch (RemoteException e) {
93         System.out.println("Error: Unable to add user at server!");
94         return -1;
95     }
96 }
97
98 public void addUserAddedListener(UserAddedListener listener) {
99     userAddedListeners.add(listener);
100 }
101
102 }
```

```
104     public void addUserUpdatedListener(UserUpdatedListener listener) {
105         userUpdatedListeners.add(listener);
106     }
107
108
109     public void onUserAdded(User user) throws RemoteException {
110         for (UserAddedListener l : userAddedListeners) {
111             l.onUserAdded(user);
112         }
113     }
114
115
116     public void onUserUpdated(User user) throws RemoteException {
117         for (UserUpdatedListener l : userUpdatedListeners) {
118             l.onUserUpdated(user);
119         }
120     }
121
122
123
124
125
126     public void removeAllGroups() {
127         try {
128             db.removeAllGroups();
129         } catch (RemoteException e) {
130             System.out.println("Error: Unable to remove groups from server!"
131                 ↪ );
132         }
133     }
134
135     public Integer addGroup(Character group) {
136
137         try {
138             return db.addGroup(group);
139         } catch (RemoteException e) {
140             System.out.println("Error: Unable to add group to server!");
141             return null;
142         }
143     }
144
145     public List<String> getGroups() {
146         try {
147             return db.getGroups();
148         } catch (RemoteException e) {
149             System.out.println("Error: Unable to fetch groups from server!"
150                 ↪ );
151             return null;
152         }
153     }
154
```

```
155
156     public void removeAllTeams() {
157         try {
158             db.removeAllTeams();
159         } catch (RemoteException e) {
160             System.out.println("Error: Unable to remove teams from server!"
161                               ↪ );
162         }
163     }
164
165     public List<Team> getTeams() {
166         try {
167             return db.getTeams();
168         } catch (RemoteException e) {
169             System.out.println("Error: Unable to fetch teams from server!")
170                               ↪ ;
171             return null;
172         }
173     }
174
175     public void updateTeam(Team team) {
176         try {
177             db.updateTeam(team);
178             onTeamUpdated(team);
179         } catch (RemoteException e) {
180             System.out.println("Error: Unable to update team at server!");
181         }
182     }
183
184     public Integer addTeam(Team team) {
185         try {
186             Integer newId = db.addTeam(team);
187             team.setId(newId);
188             onTeamAdded(team);
189             return team.getId();
190         } catch (RemoteException e) {
191             System.out.println("Error: Unable to add team at server!");
192             return -1;
193         }
194     }
195
196     public void onTeamAdded(Team team) throws RemoteException {
197         for (TeamAddedListener l : teamAddedListeners) {
198             l.onTeamAdded(team);
199         }
200     }
201
202     public void onTeamUpdated(Team team) throws RemoteException {
203         for (TeamUpdatedListener l : teamUpdatedListeners) {
204             l.onTeamUpdated(team);
205         }
206     }
```

```
206     }
207
208
209     public void addTeamAddedListener(TeamAddedListener listener) {
210         teamAddedListeners.add(listener);
211     }
212
213     public void addTeamUpdatedListener(TeamUpdatedListener listener) {
214
215         teamUpdatedListeners.add(listener);
216     }
217
218
219
220
221
222
223     public void removeAllGames() {
224         try {
225             db.removeAllGames();
226         } catch (RemoteException e) {
227             System.out.println("Error: Unable to remove games from server!"
228                               ↪ );
229         }
230     }
231
232     public List<Game> getGames() {
233         try {
234             return db.getGames();
235         } catch (RemoteException e) {
236             System.out.println("Error: Unable to fetch games from server!")
237                               ↪ ;
238             return null;
239         }
240     }
241
242     public void updateGame(Game game) {
243         try {
244             db.updateGame(game);
245             onGameUpdated(game);
246         } catch (RemoteException e) {
247             System.out.println("Error: Unable to update team at server!");
248         }
249     }
250
251     public Integer addGame(Game game) {
252         try {
253             Integer newId = db.addGame(game);
254             game.setId(newId);
255             onGameAdded(game);
256             return game.getId();
257         } catch (RemoteException e) {
258             System.out.println("Error: Unable to add team at server!");
259         }
260     }
```

```
257         return -1;
258     }
259 }
260
261
262 public void onGameAdded(Game game) throws RemoteException {
263     for(GameAddedListener l : gamesAddedListeners) {
264         l.onGameAdded(game);
265     }
266 }
267
268
269 public void onGameUpdated(Game game) throws RemoteException {
270     for(GameUpdatedListener l : gamesUpdatedListeners) {
271         l.onGameUpdated(game);
272     }
273 }
274
275 public void addGamesAddedListener(GameAddedListener listener) {
276     gamesAddedListeners.add(listener);
277 }
278
279 public void addGamesUpdatedListener(GameUpdatedListener listener) {
280
281     gamesUpdatedListeners.add(listener);
282 }
283
284
285
286
287 public void removeAllBets() {
288     try {
289         db.removeAllBets();
290     } catch (RemoteException e) {
291         System.out.println("Error: Unable to remove bets from server!");
292         ↪ ;
293     }
294 }
295
296 public List<Bet> getBets() {
297     try {
298         return db.getBets();
299     } catch (RemoteException e) {
300         System.out.println("Error: Unable to fetch bets from server!");
301         return null;
302     }
303 }
304
305 public void updateBet(Bet bet) {
306     try {
307         db.updateBet(bet);
308     } catch (RemoteException e) {
309         System.out.println("Error: Unable to update bet at server!");
310     }
311 }
```

```
309         }
310     }
311
312     public Integer addBet(Bet bet) {
313         try {
314             Integer newId = db.addBet(bet);
315             bet.setId(newId);
316             return bet.getId();
317         } catch (RemoteException e) {
318             System.out.println("Error: Unable to add bet at server!");
319             return -1;
320         }
321     }
322
323
324     public void onBetAdded(Bet bet) throws RemoteException {
325         for(BetAddedListener l : betsAddedListeners) {
326             l.onBetAdded(bet);
327         }
328     }
329
330
331     public void onBetUpdated(Bet bet) throws RemoteException {
332         for(BetUpdatedListener l : betsUpdatedListeners) {
333             l.onBetUpdated(bet);
334         }
335     }
336
337     public void addBetsAddedListener(BetAddedListener listener) {
338         betsAddedListeners.add(listener);
339     }
340
341     public void addBetsUpdatedListener(BetUpdatedListener listener) {
342
343         betsUpdatedListeners.add(listener);
344     }
345
346
347 }
```

### BetAddedListener.java

```
1 package swe4.rmi.client;
2
3 import swe4.models.Bet;
4
5 public interface BetAddedListener {
6     public void onBetAdded(Bet bet);
7
8 }
```

### BetUpdatedListener.java

```
1 package swe4.rmi.client;
2
3 import swe4.models.Bet;
4
5 public interface BetUpdatedListener {
6     public void onBetUpdated(Bet bet);
7
8 }
```

### GameAddedListener.java

```
1 package swe4.rmi.client;
2
3 import swe4.models.Game;
4
5 public interface GameAddedListener {
6     public void onGameAdded(Game game);
7
8 }
```

### GameUpdatedListener.java

```
1 package swe4.rmi.client;
2
3 import swe4.models.Game;
4
5 public interface GameUpdatedListener {
6     public void onGameUpdated(Game game);
7
8 }
```

### TeamAddedListener.java

```
1 package swe4.rmi.client;
2
3 import swe4.models.Team;
4
5 public interface TeamAddedListener {
6     public void onTeamAdded(Team team);
7 }
```

### TeamUpdatedListener.java

```
1 package swe4.rmi.client;
2
3 import swe4.models.Team;
4
5 public interface TeamUpdatedListener {
6     public void onTeamUpdated(Team team);
7
8 }
```

### UserAddedListener.java

```
1 package swe4.rmi.client;
2
3 import swe4.models.User;
4
5 public interface UserAddedListener {
6     public void onUserAdded(User user);
7
8 }
```

### UserUpdatedListener.java

```
1 package swe4.rmi.client;
2
3 import swe4.models.User;
4
5 public interface UserUpdatedListener {
6     public void onUserUpdated(User user);
7
8 }
```

#### 1.2.5 swe4.rmi.server

### EuroBetDb.java

```
1 package swe4.rmi.server;
2
3 import java.rmi.Remote;
```



```
4 import java.rmi.RemoteException;
5 import java.util.List;
6
7 import swe4.models.Bet;
8 import swe4.models.Game;
9 import swe4.models.Team;
10 import swe4.models.User;
11 import swe4.rmi.client.EuroBetDbFactory;
12
13 public interface EuroBetDb extends Remote {
14
15     /**
16      * removes all users from database
17      * @throws RemoteException
18      */
19     void removeAllUsers() throws RemoteException;
20
21     /**
22      * get all users from database
23      * @return
24      * @throws RemoteException
25      */
26     List<User> getUsers() throws RemoteException;
27
28
29     /**
30      * updates an existing user in database
31      * @param user
32      * @throws RemoteException
33      */
34     void updateUser(User user) throws RemoteException;
35
36
37     /**
38      * creates a new user in database
39      * @param user
40      * @return id from created user
41      * @throws RemoteException
42      */
43     Integer addUser(User user) throws RemoteException;
44
45
46     /**
47      * removes all groups from database
48      * @throws RemoteException
49      */
50     void removeAllGroups() throws RemoteException;
51
52
53     /**
54      * creates a new group in database
55      * @param group
56      * @return group name
```

```
57         * @throws RemoteException
58         */
59         Integer addGroup(Character group) throws RemoteException;
60
61
62         /**
63         * get all groups from database
64         * @return
65         * @throws RemoteException
66         */
67         List<String> getGroups() throws RemoteException;
68
69
70
71         /**
72         * removes all teams from database
73         * @throws RemoteException
74         */
75         void removeAllTeams() throws RemoteException;
76
77         /**
78         * get all teams from database
79         * @return
80         * @throws RemoteException
81         */
82         List<Team> getTeams() throws RemoteException;
83
84
85
86         /**
87         * updates an existing team in database
88         * @param user
89         * @throws RemoteException
90         */
91         void updateTeam(Team team) throws RemoteException;
92
93
94         /**
95         * creates a new team in database
96         * @param team
97         * @return id from created team
98         * @throws RemoteException
99         */
100        Integer addTeam(Team team) throws RemoteException;
101
102
103        /**
104        * removes all games from database
105        * @throws RemoteException
106        */
107        void removeAllGames() throws RemoteException;
108
109        /**
```

```
110      * get all games from database
111      * @return
112      * @throws RemoteException
113      */
114      List<Game> getGames() throws RemoteException;
115
116
117      /**
118      * updates an existing team in database
119      * @param user
120      * @throws RemoteException
121      */
122      void updateGame(Game game) throws RemoteException;
123
124
125      /**
126      * creates a new game in database
127      * @param game
128      * @return id from created game
129      * @throws RemoteException
130      */
131      Integer addGame(Game game) throws RemoteException;
132
133
134
135      /**
136      * removes all bets from database
137      * @throws RemoteException
138      */
139      void removeAllBets() throws RemoteException;
140
141
142      /**
143      * get all games from database
144      * @return
145      * @throws RemoteException
146      */
147      List<Bet> getBets() throws RemoteException;
148
149
150      /**
151      * creates a new bet in database
152      * @param bet
153      * @return id from created bet
154      * @throws RemoteException
155      */
156      Integer addBet(Bet bet) throws RemoteException;
157
158
159      /**
160      * updates an existing bet in database
161      * @param bet
162      * @throws RemoteException
163      */
```

```
163         void updateBet(Bet bet) throws RemoteException;
164
165
166
167         // called by clients to register for new server callbacks
168         public void registerForNotification(EuroBetDbFactory n) throws RemoteException;
169     }
```

## EuroBetDbServer.java

```
1 package swe4.rmi.server;
2
3 import java.sql.*;
4 import java.net.MalformedURLException;
5 import java.rmi.Naming;
6 import java.rmi.RemoteException;
7 import java.rmi.registry.LocateRegistry;
8 import java.rmi.server.UnicastRemoteObject;
9 import java.util.ArrayList;
10 import java.util.List;
11
12 import swe4.models.Bet;
13 import swe4.models.Game;
14 import swe4.models.Team;
15 import swe4.models.User;
16 import swe4.rmi.client.EuroBetDbFactory;
17 import swe4.util.Util;
18
19
20
21 public class EuroBetDbServer implements EuroBetDb {
22
23
24     private static final String DB_URL = "127.0.0.1";
25     private static final String DB_PORT = "3306";
26     private static final String DB_NAME = "EuroBetDb";
27     private static final String DB_USER = "root";
28     private static final String DB_PASSWORD = "root";
29
30     private List<EuroBetDbFactory> clientList = new ArrayList<>();
31
32
33     public EuroBetDbServer() throws RemoteException {
34         super();
35     }
36
37
38
39     @Override
40     public synchronized List<User> getUsers() throws RemoteException {
```

```

41
42     List<User> users = new ArrayList<>();
43     ResultSet result = executeSqlSelect("SELECT * FROM users");
44
45     System.out.println("--> getUsers");
46     try {
47         while(result.next()) {
48
49             User user = new User(
50                 result.getInt("idusers"),
51                 result.getBoolean("active"),
52                 result.getString("name"),
53                 result.getString("password")
54             );
55             users.add(user);
56             System.out.println("    -> User: " + user.toString());
57         }
58     } catch (SQLException e) {
59         e.printStackTrace();
60     }
61     return users;
62 }
63
64
65
66 @Override
67 public void updateUser(User user) throws RemoteException {
68
69     System.out.println("--> updateUser");
70     System.out.println("    -> User: " + user.toString());
71
72     executeSqlUpdate(
73         "UPDATE users SET"
74             + " name = '" + user.getName() + "',"
75             + " password = '" + user.getPassword()
76             + " ' + " + " ', "
77             + " active = " + user.getActive()
78             + " WHERE idusers = " + user.getId()
79             + ";");
80
81     notifyOnUserUpdated(user);
82 }
83
84 @Override
85 public Integer addUser(User user) throws RemoteException {
86
87     System.out.println("--> addUser");
88     System.out.println("    -> User: " + user.toString());
89
90     Integer newId = executeSqlUpdate(
91         "INSERT INTO users VALUES("
92             + " null, "

```

```

93         + " '" + user.getName() + "', "
94         + " '" + user.getPassword() + "', "
95         + " '" + user.getActive()
96         + "');"");
97     user.setId(newId);
98     notifyOnUserAdded(user);
99     return user.getId();
100 }
101
102 @Override
103 public Integer addGroup(Character group) throws RemoteException {
104     String groupStr = group.toString();
105     System.out.println("--> addGroup");
106     System.out.println("    -> group: " + groupStr);
107
108     Integer newId = executeSqlUpdate(
109         "INSERT INTO groups VALUES ("
110             + " null, "
111             + " '" + groupStr + "'"
112             + ");"");
113
114     return newId;
115 }
116
117 @Override
118 public List<String> getGroups() throws RemoteException {
119     List<String> groups = new ArrayList<>();
120     ResultSet result = executeSqlSelect("SELECT * FROM groups");
121
122     System.out.println("--> getGroups");
123     try {
124         while(result.next()) {
125
126             groups.add(result.getString("name"));
127             System.out.println("    -> Group: " + result.getString
128                 ↪ ("name"));
129         }
130     } catch (SQLException e) {
131         e.printStackTrace();
132     }
133     return groups;
134 }
135
136 @Override
137 public List<Team> getTeams() throws RemoteException {
138     List<Team> teams = new ArrayList<>();
139
140
141     ResultSet result = executeSqlSelect("SELECT * FROM teams");
142
143     System.out.println("--> getTeams");
144

```

```

145
146         try {
147             while(result.next()) {
148
149                 // get group
150
151                 String groupId = result.getString("groupId");
152                 ResultSet groupResult = executeSqlSelect("SELECT * FROM
153                     ↪ groups WHERE Idgroups = " + groupId);
154                 String groupName = "";
155                 while(groupResult.next()) {
156                     groupName = groupResult.getString("name");
157                 }
158
159                 Team team = new Team(
160                     result.getInt("idteams"),
161                     result.getString("name"),
162                     groupName
163                 );
164                 teams.add(team);
165                 System.out.println("        -> Team: " + team.toString());
166             }
167         } catch (SQLException e) {
168             e.printStackTrace();
169         }
170     }
171
172
173
174     @Override
175     public void updateTeam(Team team) throws RemoteException {
176         System.out.println("--> updateTeam");
177         System.out.println("        -> Team: " + team.toString());
178
179         Integer groupId = getGroupIdByName(team.getGroup());
180         executeSqlUpdate(
181             "UPDATE teams SET"
182                 + " name = '" + team.getName() + "',"
183                 + " groupId = " + groupId
184             + " WHERE idteams = " + team.getId()
185             + ";");
186
187         notifyOnTeamUpdated(team);
188     }
189
190     @Override
191     public Integer addTeam(Team team) throws RemoteException {
192         System.out.println("--> addTeam");
193         System.out.println("        -> Team: " + team.toString());
194
195         Integer groupId;
196

```

```

197         if(team.getGroup().equals("")) {
198             groupId = 1; // default group
199         } else {
200             groupId = getGroupIdByName(team.getGroup());
201         }
202
203         Integer newId = executeSqlUpdate(
204             "INSERT INTO teams VALUES("
205                 + " null, "
206                 + " '" + team.getName() + "', "
207                 + " " + groupId
208                 + ");");
209
210         team.setId(newId);
211         notifyOnTeamAdded(team);
212         return team.getId();
213     }
214
215
216     private Integer getGroupIdByName(String name) {
217         ResultSet groupResult = executeSqlSelect("SELECT * FROM groups WHERE
218             ↪ name = '" + name + "'");
219         Integer id = -1;
220         try {
221             while(groupResult.next()) {
222                 id = groupResult.getInt("idgroups");
223             }
224         } catch (SQLException e) {
225             e.printStackTrace();
226         }
227         return id;
228     }
229
230
231     @Override
232     public List<Game> getGames() throws RemoteException {
233         List<Game> games = new ArrayList<>();
234         ResultSet result = executeSqlSelect("SELECT * FROM games");
235         System.out.println("--> getGames");
236         try {
237             while(result.next()) {
238
239                 // get teams
240                 Team teamA = getTeamById(result.getInt("teamAId"));
241                 Team teamB = getTeamById(result.getInt("teamBId"));
242
243                 Game game = new Game(
244                     result.getInt("idgames"),
245                     teamA,
246                     teamB,
247                     result.getBoolean("finished"),
248                     result.getInt("goalsTeamA"),

```



```

249                                     result.getInt("goalsTeamB")
250                                 );
251                                 games.add(game);
252                                 System.out.println("        -> Game: " + game.toString());
253                             }
254                             } catch (SQLException e) {
255                                 e.printStackTrace();
256                             }
257                             return games;
258                         }
259
260
261     @Override
262     public void updateGame(Game game) throws RemoteException {
263         System.out.println("--> updateGame");
264         System.out.println("        -> Game: " + game.toString());
265
266
267         executeSqlUpdate(
268             "UPDATE games SET"
269                 + " teamAId = " + game.getTeamA().getId
270                 + " " + game.getTeamA().getId
271                 + " " + game.getTeamB().getId
272                 + " " + game.getGoalsTeamA()
273                 + " " + game.getGoalsTeamB()
274                 + " " + game.getGameFinished()
275                 + " WHERE idgames = " + game.getId()
276                 + ";");
277
278         notifyOnGameUpdated(game);
279     }
280
281
282     @Override
283     public Integer addGame(Game game) throws RemoteException {
284         System.out.println("--> addGame");
285         System.out.println("        -> game: " + game.toString());
286         Integer newId = executeSqlUpdate(
287             "INSERT INTO games VALUES("
288                 + " null, "
289                 + " " + game.getTeamA().getId() + ", "
290                 + " " + game.getTeamB().getId() + ", "
291                 + " " + game.getGoalsTeamA() + ", "
292                 + " " + game.getGoalsTeamB() + ", "
293                 + " " + game.getGameFinished()
294                 + ");");
295
296         game.setId(newId);

```

```

297         notifyOnGameAdded(game);
298         return game.getId();
299     }
300
301     @Override
302     public void removeAllBets() throws RemoteException {
303         executeSqlUpdate("DELETE FROM bets WHERE idbets >= 0");
304         System.out.println("--> removeAllBets");
305     }
306
307
308     @Override
309     public List<Bet> getBets() throws RemoteException {
310         List<Bet> bets = new ArrayList<>();
311         ResultSet result = executeSqlSelect("SELECT * FROM bets");
312         System.out.println("--> getBets");
313         try {
314             while(result.next()) {
315
316                 // get relationships
317                 Team teamA = getTeamById(result.getInt("teamAId"));
318                 Team teamB = getTeamById(result.getInt("teamBId"));
319                 User user = getUserById(result.getInt("userId"));
320
321                 Bet bet = new Bet(
322                     result.getInt("idbets"),
323                     user,
324                     teamA,
325                     teamB,
326                     result.getInt("goalsTeamA"),
327                     result.getInt("goalsTeamB")
328                 );
329                 bets.add(bet);
330                 System.out.println("    -> Bet: " + bet.toString());
331             }
332         } catch (SQLException e) {
333             e.printStackTrace();
334         }
335         return bets;
336     }
337
338     @Override
339     public Integer addBet(Bet bet) throws RemoteException {
340         System.out.println("--> addBet");
341         System.out.println("    -> bet: " + bet.toString());
342         Integer newId = executeSqlUpdate(
343             "INSERT INTO bets VALUES("
344                 + " null, "
345                 + " " + bet.getUser().getId() + ", "
346                 + " " + bet.getTeamA().getId() + ", "
347                 + " " + bet.getTeamB().getId() + ", "
348                 + " " + bet.getGoalsTeamA() + ", "
349                 + " " + bet.getGoalsTeamB()

```

```

350         + ");");
351
352         bet.setId(newId);
353         notifyOnBetAdded(bet);
354         return bet.getId();
355     }
356
357
358     @Override
359     public void updateBet(Bet bet) throws RemoteException {
360         System.out.println("--> updateBet");
361         System.out.println("    -> Bet: " + bet.toString());
362
363
364         executeSqlUpdate(
365             "UPDATE bets SET"
366                 + " userId = " + bet.getUser().getId()
367                 + " teamAId = " + bet.getTeamA().getId()
368                 + " teamBId = " + bet.getTeamB().getId()
369                 + " goalsTeamA = " + bet.getGoalsTeamA()
370                 + " goalsTeamB = " + bet.getGoalsTeamB()
371                 + " WHERE idbets = " + bet.getId()
372                 + ";");
373
374         notifyOnBetUpdated(bet);
375     }
376
377
378     @Override
379     public void removeAllUsers() throws RemoteException {
380         executeSqlUpdate("DELETE FROM users WHERE idusers >= 0");
381         System.out.println("--> removeAllUsers");
382     }
383
384
385
386     @Override
387     public void removeAllGroups() throws RemoteException {
388         executeSqlUpdate("DELETE FROM groups WHERE idgroups >= 0");
389         System.out.println("--> removeAllGroups");
390     }
391
392
393
394     @Override
395     public void removeAllTeams() throws RemoteException {
396         executeSqlUpdate("DELETE FROM teams WHERE idteams >= 0");
397         System.out.println("--> removeAllTeams");

```

```
398     }
399
400
401
402     @Override
403     public void removeAllGames() throws RemoteException {
404         executeSqlUpdate("DELETE FROM games WHERE idgames >= 0");
405         System.out.println("--> removeAllGames");
406     }
407
408
409
410     private String getGroupNameById(Integer id) {
411         ResultSet groupResult = executeSqlSelect("SELECT * FROM groups WHERE
412             ↳ idgroups = " + id );
413         try {
414             while(groupResult.next()) {
415                 return groupResult.getString("name");
416             }
417         } catch (SQLException e) {
418             e.printStackTrace();
419         }
420         return "";
421     }
422
423     private Team getTeamById(Integer id) {
424         ResultSet result = executeSqlSelect("SELECT * FROM teams WHERE idteams
425             ↳ = " + id);
426         try {
427             while(result.next()) {
428                 return new Team(
429                     result.getInt("idteams"),
430                     result.getString("name"),
431                     getGroupNameById(result.getInt("groupId
432                         ↳ ")))
433             );
434         } catch (SQLException e) {
435             e.printStackTrace();
436         }
437         return null;
438     }
439
440     private User getUserById(Integer id) {
441         ResultSet result = executeSqlSelect("SELECT * FROM users WHERE idusers
442             ↳ = " + id);
443         try {
444             while(result.next()) {
445                 return new User(
446                     result.getInt("idusers"),
447                     result.getBoolean("active"),
448                     result.getString("name"),
449                     result.getString("password")
```

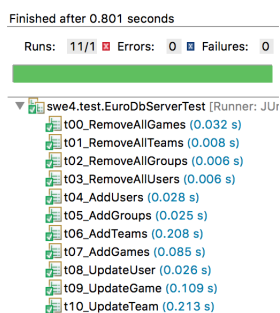
```
447         );
448     }
449     } catch (SQLException e) {
450         e.printStackTrace();
451     }
452     return null;
453 }
454
455
456
457
458
459
460 @Override
461 public void registerForNotification(EuroBetDbFactory n) throws RemoteException
462     ↪ {
463     System.out.println("registerForNotification: " +n);
464     clientList.add(n);
465 }
466
467 private void notifyOnUserUpdated(User user) throws RemoteException {
468
469     for(EuroBetDbFactory client : clientList) {
470         client.onUserUpdated(user);
471     }
472 }
473
474 private void notifyOnUserAdded(User user) throws RemoteException {
475     for(EuroBetDbFactory client : clientList) {
476         client.onUserAdded(user);
477     }
478 }
479
480
481 private void notifyOnTeamUpdated(Team team) throws RemoteException {
482
483     for(EuroBetDbFactory client : clientList) {
484         client.onTeamUpdated(team);
485     }
486 }
487
488 private void notifyOnTeamAdded(Team team) throws RemoteException {
489     for(EuroBetDbFactory client : clientList) {
490         client.onTeamAdded(team);
491     }
492 }
493
494
495 private void notifyOnGameUpdated(Game game) throws RemoteException {
496
497     for(EuroBetDbFactory client : clientList) {
498         client.onGameUpdated(game);
```

```
499         }
500     }
501
502     private void notifyOnGameAdded(Game game) throws RemoteException {
503         for(EuroBetDbFactory client : clientList) {
504             client.onGameAdded(game);
505         }
506     }
507
508     private void notifyOnBetUpdated(Bet bet) throws RemoteException {
509
510         for(EuroBetDbFactory client : clientList) {
511             client.onBetUpdated(bet);
512         }
513     }
514
515     private void notifyOnBetAdded(Bet bet) throws RemoteException {
516         for(EuroBetDbFactory client : clientList) {
517             client.onBetAdded(bet);
518         }
519     }
520
521
522
523     private Connection getConnection() throws SQLException {
524         return DriverManager.getConnection("jdbc:mysql://" + DB_URL + ":" +
525             ↪ DB_PORT + "/" + DB_NAME, DB_USER, DB_PASSWORD);
526     }
527
528     private ResultSet executeSqlSelect(String sql) {
529
530         try {
531             Connection connection = getConnection();
532             Statement statement = connection.createStatement();
533             return statement.executeQuery(sql);
534
535         } catch (Exception e) {
536             e.printStackTrace();
537             return null;
538         }
539     }
540
541     private Integer executeSqlUpdate(String sql) {
542
543         try {
544             Connection connection = getConnection();
545             Statement statement = connection.createStatement();
546             statement.executeUpdate(sql, Statement.RETURN_GENERATED_KEYS);
547             ResultSet rs = statement.getGeneratedKeys();
548             if (rs.next()) {
549                 int newId = rs.getInt(1);
550                 return newId;
551             }
552         }
553     }
554 }
```

```
551         return 0;
552
553     } catch (Exception e) {
554         e.printStackTrace();
555         return -1;
556     }
557 }
558
559
560
561
562
563
564
565     public static void main(String[] args) {
566
567         try {
568             String hostPort = Util.getHostPortArg(args);
569             EuroBetDb b = (EuroBetDb) new EuroBetDbServer();
570             EuroBetDb buffSkel = (EuroBetDb) UnicastRemoteObject.
571                 ↪ exportObject(b, 0);
572             LocateRegistry.createRegistry(Util.getPort(hostPort));
573
574             Naming.rebind("rmi://" + hostPort + "/" + EuroBetDb.class.getSimpleName(), buffSkel);
575
576             System.out.println("EuroBetDb Server running, waiting for
577                 ↪ connections ...");
578
579             } catch (RemoteException e) {
580                 e.printStackTrace();
581             } catch (MalformedURLException e) {
582                 e.printStackTrace();
583             }
584         }
585     }
```

## 1.3 Testfälle

### EuroDbServerTest.java



```
1 package swe4.test;
2
3 import static org.junit.Assert.*;
4
5 import java.util.List;
6 import org.junit.runners.MethodSorters;
7 import org.junit.FixMethodOrder;
8 import org.junit.Test;
9
10 import swe4.models.Game;
11 import swe4.models.Team;
12 import swe4.models.User;
13 import swe4.rmi.client.EuroBetDbFactory;
14
15 @FixMethodOrder(MethodSorters.NAME_ASCENDING)
16 public class EuroDbServerTest {
17
18
19     EuroBetDbFactory server;
20     private Integer zero = 0;
21
22
23     @Test
24     public void t00_RemoveAllGames() {
25         Integer current;
26         EuroBetDbFactory.getInstance().removeAllGames();
27         current = EuroBetDbFactory.getInstance().getGames().size();
28         assertEquals(zero, current);
29     }
30
31     @Test
32     public void t01_RemoveAllTeams() {
33         Integer current;
34         EuroBetDbFactory.getInstance().removeAllTeams();
35         current = EuroBetDbFactory.getInstance().getTeams().size();
36         assertEquals(zero, current);
37     }
38
39
40     @Test
41     public void t02_RemoveAllGroups() {
42         Integer current;
43         EuroBetDbFactory.getInstance().removeAllGroups();
44         current = EuroBetDbFactory.getInstance().getGroups().size();
45         assertEquals(zero, current);
46     }
47
48     @Test
49     public void t03_RemoveAllUsers() {
50         Integer current;
51         EuroBetDbFactory.getInstance().removeAllUsers();
52         current = EuroBetDbFactory.getInstance().getUsers().size();
```



```
53         assertEquals(zero, current);
54     }
55
56
57     @Test
58     public void t04_AddUsers() {
59         User user = new User();
60
61         user.setName("Robert Wurm");
62         user.setActive(false);
63         user.setPassword("1234");
64         EuroBetDbFactory.getInstance().addUser(user);
65
66         user.setName("Hansi Hinterseer");
67         user.setActive(true);
68         user.setPassword("123456");
69         EuroBetDbFactory.getInstance().addUser(user);
70
71
72         user.setName("Frau Holle");
73         user.setActive(true);
74         user.setPassword("xxx");
75         EuroBetDbFactory.getInstance().addUser(user);
76
77
78
79         user.setName("Asterix");
80         user.setActive(true);
81         user.setPassword("xx56x");
82         EuroBetDbFactory.getInstance().addUser(user);
83
84
85
86         List<User> allUsers = EuroBetDbFactory.getInstance().getUsers();
87         assertNotNull(allUsers);
88         assertEquals(4, allUsers.size());
89
90         Boolean robertExists = false;
91         Boolean asterixExists = false;
92         for(User u : allUsers) {
93             if(u.getName().equals("Robert Wurm")) {
94                 assertEquals(u.getPassword(), "1234");
95                 assertEquals(u.getActive(), false);
96                 robertExists = true;
97             }
98
99             if(u.getName().equals("Asterix")) {
100                 assertEquals(u.getPassword(), "xx56x");
101                 assertEquals(u.getActive(), true);
102                 asterixExists = true;
103             }
104
105             if(u.getName().equals("Dummy !!!")) {
```

```
106         fail("username shouldn't exists!");
107     }
108 }
109 assertEquals(true, robertExists);
110 assertEquals(true, asterixExists);
111 }
112
113
114 @Test
115 public void t05_AddGroups() {
116
117     EuroBetDbFactory.getInstance().addGroup('A');
118     EuroBetDbFactory.getInstance().addGroup('B');
119     EuroBetDbFactory.getInstance().addGroup('C');
120     EuroBetDbFactory.getInstance().addGroup('D');
121     EuroBetDbFactory.getInstance().addGroup('E');
122     EuroBetDbFactory.getInstance().addGroup('F');
123
124     List<String> allGroups = EuroBetDbFactory.getInstance().getGroups();
125
126     assertNotNull(allGroups);
127     assertEquals(6, allGroups.size());
128
129     Boolean existsA = false;
130     Boolean existsD = false;
131     for(String g : allGroups) {
132         if(g.equals("A")) {
133             existsA = true;
134         }
135
136         if(g.equals("D")) {
137             existsD = true;
138         }
139         if(g.equals("Z")) {
140             fail("group shouldn't exists!");
141         }
142     }
143     assertEquals(true, existsA);
144     assertEquals(true, existsD);
145 }
146
147
148 @Test
149 public void t06_AddTeams() {
150     Team team = new Team();
151     Integer id;
152     Integer unableToCreate = -1;
153
154     team.setName("France");
155     team.setGroup("Z");
156     id = EuroBetDbFactory.getInstance().addTeam(team);
157     assertEquals(id, unableToCreate);
158     team.setGroup("A");
```

```
159         id = EuroBetDbFactory.getInstance().addTeam(team);
160         assertEquals(id, unableToCreate);
161
162
163         team.setName("Switzerland");
164         team.setGroup("A");
165         id = EuroBetDbFactory.getInstance().addTeam(team);
166         assertEquals(id, unableToCreate);
167
168
169         team.setName("Romania");
170         team.setGroup("A");
171         id = EuroBetDbFactory.getInstance().addTeam(team);
172         assertEquals(id, unableToCreate);
173
174
175         team.setName("Albania");
176         team.setGroup("A");
177         id = EuroBetDbFactory.getInstance().addTeam(team);
178         assertEquals(id, unableToCreate);
179
180
181         team.setName("Wales");
182         team.setGroup("B");
183         id = EuroBetDbFactory.getInstance().addTeam(team);
184         assertEquals(id, unableToCreate);
185
186
187         team.setName("Slovakia");
188         team.setGroup("B");
189         id = EuroBetDbFactory.getInstance().addTeam(team);
190         assertEquals(id, unableToCreate);
191
192
193         team.setName("England");
194         team.setGroup("B");
195         id = EuroBetDbFactory.getInstance().addTeam(team);
196         assertEquals(id, unableToCreate);
197
198
199         team.setName("Russia");
200         team.setGroup("B");
201         id = EuroBetDbFactory.getInstance().addTeam(team);
202         assertEquals(id, unableToCreate);
203
204
205         team.setName("Germany");
206         team.setGroup("C");
207         id = EuroBetDbFactory.getInstance().addTeam(team);
208         assertEquals(id, unableToCreate);
209
210
211         team.setName("Northern Ireland");
```

```
212         team.setGroup("C");
213         id = EuroBetDbFactory.getInstance().addTeam(team);
214         assertEquals(id, unableToCreate);
215
216         team.setName("Ukraine");
217         team.setGroup("C");
218         id = EuroBetDbFactory.getInstance().addTeam(team);
219         assertEquals(id, unableToCreate);
220
221         team.setName("Croatia");
222         team.setGroup("D");
223         id = EuroBetDbFactory.getInstance().addTeam(team);
224         assertEquals(id, unableToCreate);
225
226         team.setName("Spain");
227         team.setGroup("D");
228         id = EuroBetDbFactory.getInstance().addTeam(team);
229         assertEquals(id, unableToCreate);
230
231         team.setName("Czech Republic");
232         team.setGroup("D");
233         id = EuroBetDbFactory.getInstance().addTeam(team);
234         assertEquals(id, unableToCreate);
235
236         team.setName("Turkey");
237         team.setGroup("D");
238         id = EuroBetDbFactory.getInstance().addTeam(team);
239         assertEquals(id, unableToCreate);
240
241
242         team.setName("Italy");
243         team.setGroup("E");
244         id = EuroBetDbFactory.getInstance().addTeam(team);
245         assertEquals(id, unableToCreate);
246
247         team.setName("Ireland");
248         team.setGroup("E");
249         id = EuroBetDbFactory.getInstance().addTeam(team);
250         assertEquals(id, unableToCreate);
251
252         team.setName("Sweden");
253         team.setGroup("E");
254         id = EuroBetDbFactory.getInstance().addTeam(team);
255         assertEquals(id, unableToCreate);
256
257         team.setName("Belgium");
258         team.setGroup("E");
259         id = EuroBetDbFactory.getInstance().addTeam(team);
260         assertEquals(id, unableToCreate);
261
262         team.setName("Hungary");
263         team.setGroup("F");
264         id = EuroBetDbFactory.getInstance().addTeam(team);
```

```
265         assertEquals(id, unableToCreate);
266
267         team.setName("Iceland");
268         team.setGroup("F");
269         id = EuroBetDbFactory.getInstance().addTeam(team);
270         assertEquals(id, unableToCreate);
271
272         team.setName("Portugal");
273         team.setGroup("F");
274         id = EuroBetDbFactory.getInstance().addTeam(team);
275         assertEquals(id, unableToCreate);
276
277         team.setName("Austria");
278         team.setGroup("F");
279         id = EuroBetDbFactory.getInstance().addTeam(team);
280         assertEquals(id, unableToCreate);
281
282         List<Team> allTeams = EuroBetDbFactory.getInstance().getTeams();
283         assertNotNull(allTeams);
284         assertEquals(24, allTeams.size());
285
286         Boolean austriaExists = false;
287         Boolean walesExists = false;
288         for(Team t : allTeams) {
289             if(t.getName().equals("Austria")) {
290                 austriaExists = true;
291             }
292
293             if(t.getName().equals("Wales")) {
294                 walesExists = true;
295             }
296
297             if(t.getName().equals("Lichtenstein")) {
298                 fail("team shouldnt exists!");
299             }
300         }
301         assertEquals(true, austriaExists);
302         assertEquals(true, walesExists);
303     }
304
305
306     @Test
307     public void t07_AddGames() {
308         Game game = new Game();
309         Team austria = null;
310         Team iceland = null;
311         Team portugal = null;
312         Team hungary = null;
313
314         List<Team> allTeams = EuroBetDbFactory.getInstance().getTeams();
315
316         for(Team t : allTeams) {
317             if(t.getName().equals("Austria")) { austria = t; }
```

```
318         if(t.getName().equals("Iceland")) { iceland = t; }
319         if(t.getName().equals("Hungary")) { hungary = t; }
320         if(t.getName().equals("Portugal")) { portugal = t; }
321     }
322
323     assertNotNull(austria);
324     assertNotNull(iceland);
325     assertNotNull(portugal);
326     assertNotNull(hungary);
327
328
329
330     game.setTeamA(austria);
331     game.setTeamB(hungary);
332     game.setGameFinished(true);
333     game.setGoalsTeamA(0);
334     game.setGoalsTeamB(2);
335     EuroBetDbFactory.getInstance().addGame(game);
336
337
338     game.setTeamA(portugal);
339     game.setTeamB(iceland);
340     game.setGameFinished(true);
341     game.setGoalsTeamA(1);
342     game.setGoalsTeamB(1);
343     EuroBetDbFactory.getInstance().addGame(game);
344
345
346     game.setTeamA(portugal);
347     game.setTeamB(austria);
348     game.setGameFinished(false);
349     EuroBetDbFactory.getInstance().addGame(game);
350
351
352     game.setTeamA(hungary);
353     game.setTeamB(portugal);
354     EuroBetDbFactory.getInstance().addGame(game);
355
356     game.setTeamA(iceland);
357     game.setTeamB(hungary);
358     EuroBetDbFactory.getInstance().addGame(game);
359
360     game.setTeamA(austria);
361     game.setTeamB(iceland);
362     EuroBetDbFactory.getInstance().addGame(game);
363 }
364
365
366 @Test
367 public void t08_UpdateUser() {
368     List<User> allUsers = EuroBetDbFactory.getInstance().getUsers();
369     User robert = null;
370     User hanshi = null;
```

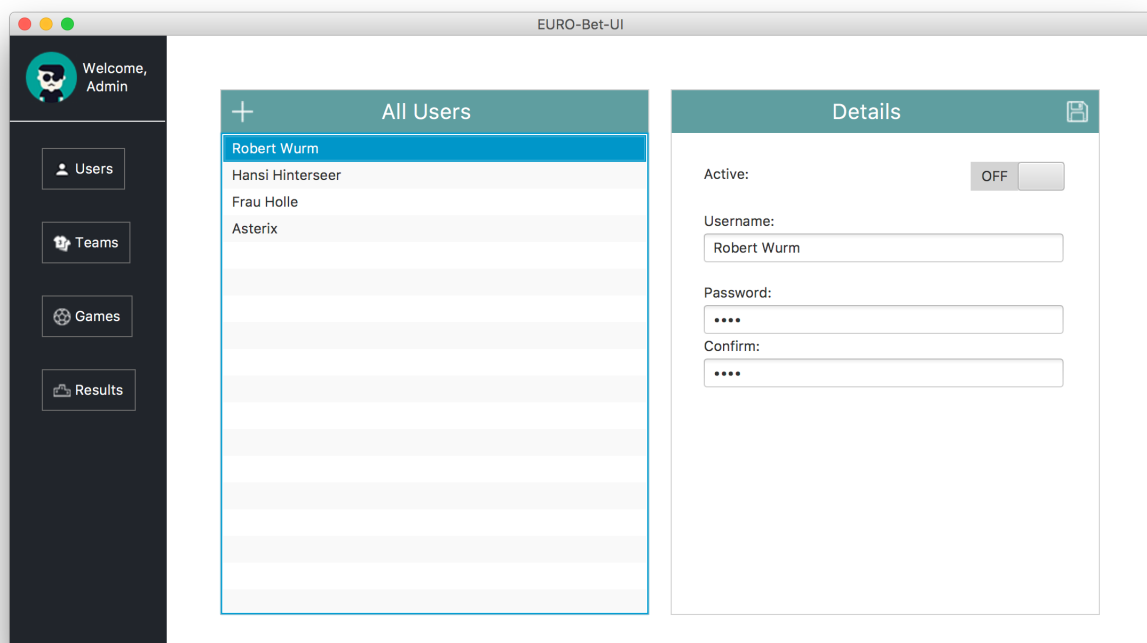
```
371         User hansy = null;
372         for(User u : allUsers) {
373             if(u.getName().equals("Robert Wurm")) { robert = u; }
374             if(u.getName().equals("Hansi Hinterseer")) { hansi = u; }
375             if(u.getName().equals("Hansy Hinterseer")) { hansy = u; }
376         }
377         assertNotNull(robert);
378         assertNotNull(hansi);
379         assertNull(hansy);
380
381
382         robert.setName("Updated Robert");
383         EuroBetDbFactory.getInstance().updateUser(robert);
384
385
386         allUsers = EuroBetDbFactory.getInstance().getUsers();
387         robert = null;
388         hansy = null;
389         for(User u : allUsers) {
390             if(u.getName().equals("Updated Robert")) { robert = u; }
391             if(u.getName().equals("Hansi Hinterseer")) { hansi = u; }
392             if(u.getName().equals("Hansy Hinterseer")) { hansy = u; }
393         }
394         assertNull(hansy);
395         assertNotNull(robert);
396         assertNotNull(hansi);
397
398         assertEquals(robert.getName(), "Updated Robert");
399         assertEquals(hansi.getName(), "Hansi Hinterseer");
400
401
402         robert.setName("Robert Wurm");
403         EuroBetDbFactory.getInstance().updateUser(robert);
404     }
405
406
407     @Test
408     public void t09_UpdateGame() {
409         Game game = null;
410         List<Game> allGames;
411
412         allGames = EuroBetDbFactory.getInstance().getGames();
413
414         for(Game g : allGames) {
415             if(g.getTeamA().getName().equals("Austria") &&
416                g.getTeamB().getName().equals("Iceland")) {
417                 game = g;
418             }
419         }
420
421         assertNotNull(game);
422
423         Integer newGoalsTeamA = 10;
```

```
424         Integer newGoalsTeamB = 0;
425         game.setGameFinished(true);
426         game.setGoalsTeamA(newGoalsTeamA);
427         game.setGoalsTeamB(newGoalsTeamB);
428
429         EuroBetDbFactory.getInstance().updateGame(game);
430
431         game = null;
432         allGames = EuroBetDbFactory.getInstance().getGames();
433
434         for(Game g : allGames) {
435             if(g.getTeamA().getName().equals("Austria") &&
436                g.getTeamB().getName().equals("Iceland")) {
437                 game = g;
438             }
439         }
440
441         assertNotNull(game);
442
443         assertEquals(game.getGoalsTeamA(), newGoalsTeamA);
444         assertNotEquals(game.getGoalsTeamB(), newGoalsTeamA);
445
446         game.setGoalsTeamA(0);
447         game.setGoalsTeamB(2);
448
449         EuroBetDbFactory.getInstance().updateGame(game);
450     }
451
452     @Test
453     public void t10_UpdateTeam() {
454         Team austria = null;
455         List<Team> allTeams;
456
457
458         allTeams = EuroBetDbFactory.getInstance().getTeams();
459         for(Team t:allTeams) {
460             if(t.getName().equals("Austria")) { austria = t; };
461         }
462         assertNotNull(austria);
463
464         austria.setName("Australia");
465         EuroBetDbFactory.getInstance().updateTeam(austria);
466
467         austria = null;
468         allTeams = EuroBetDbFactory.getInstance().getTeams();
469         for(Team t:allTeams) {
470             if(t.getName().equals("Australia")) { austria = t; };
471         }
472         assertNotNull(austria);
473
474         assertEquals(austria.getName(), "Australia");
475
476     }
```

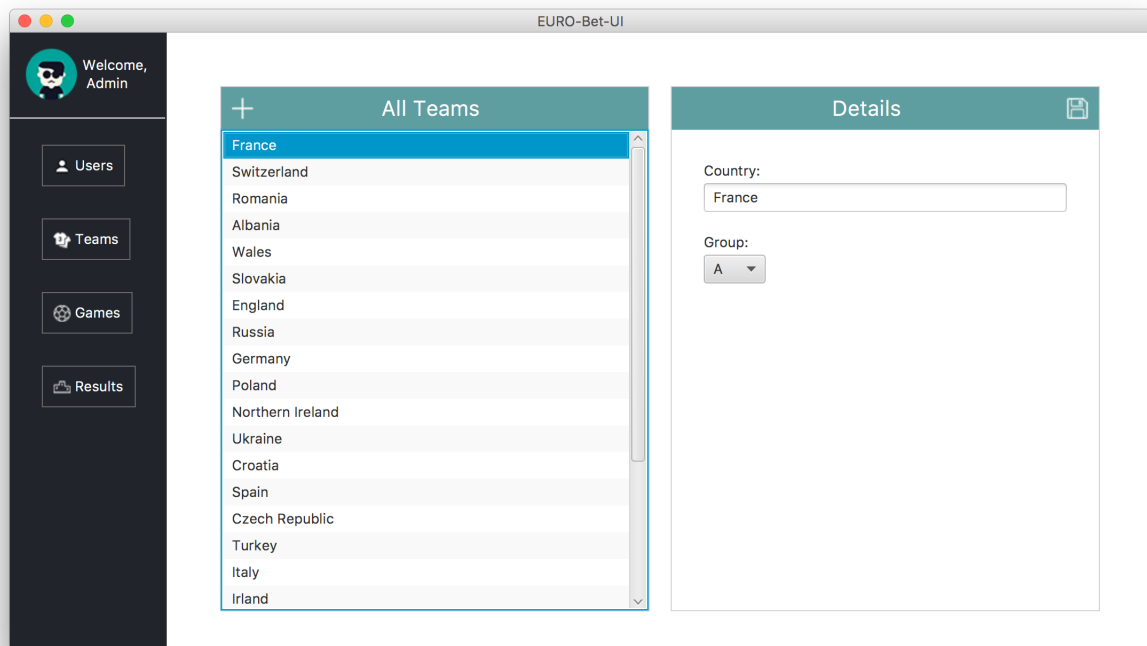


```
477
478     austria.setName("Austria");
479     EuroBetDbFactory.getInstance().updateTeam(austria);
480
481     austria = null;
482     allTeams = EuroBetDbFactory.getInstance().getTeams();
483     for (Team t : allTeams) {
484         if (t.getName().equals("Austria")) { austria = t; };
485     }
486     assertNotNull(austria);
487
488     assertEquals(austria.getName(), "Austria");
489 }
490 }
```

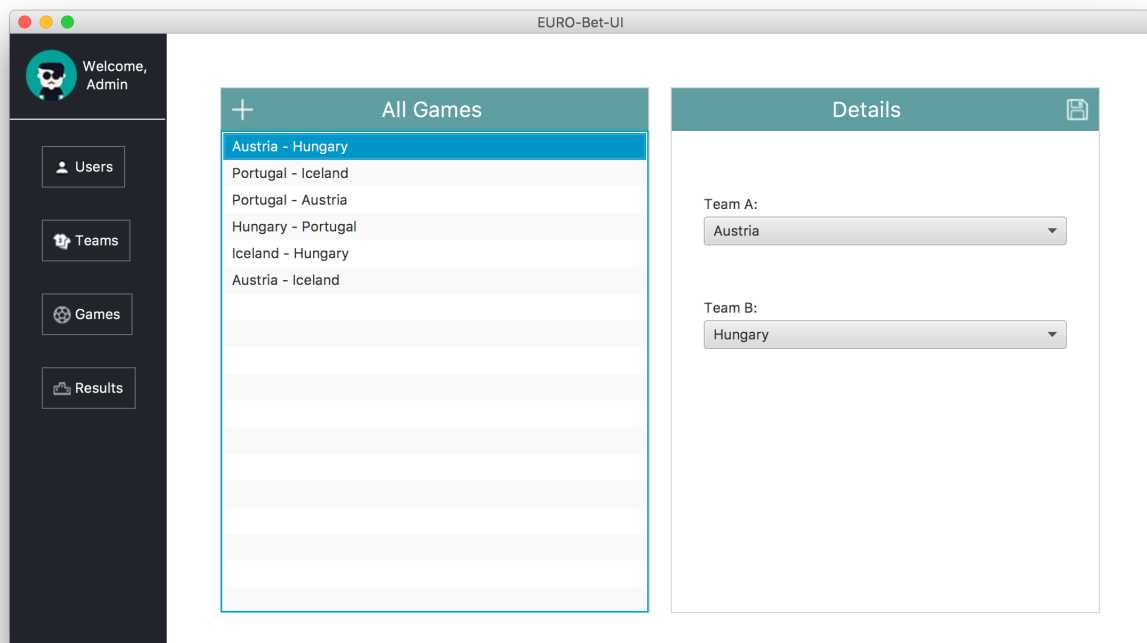
## Users



## Teams



## Games



## Results

