

Dokumentation der Projektarbeit

„Fußballtippspiel“

bei Prof. Dr. Gerald Timmer,

bearbeitet von:

Timo Flake Matrikelnummer: 712826

Annika Petry Matrikelnummer: 631907

## Inhaltsverzeichnis

1. Erläuterung des Projekts.....	3
2. Projektplanung .....	3
3. Projektdurchführung.....	6
3.1 Die Datenbank .....	6
3.2 Das Programm .....	8
3.2.1 Die Anmeldung.....	9
3.2.2 Das Benutzermenü .....	15
Das Tippmenü .....	16
Ergebnisse .....	20
3.2.3 Das Managermenü .....	24
Mannschaftsmenü.....	25
Turniermenü .....	29
Spielmenü.....	31
Punktestände abfragen.....	38

## 1. Erläuterung des Projekts

Im Nachfolgenden werden wir die von uns erarbeitete Lösung beschreiben. Uns wurden drei Entwicklungsumgebungen zur Auswahl gestellt, CORBA (als C++ oder Java), Java RMI oder webbasiert mit Java-Servlets.

Wir haben uns für eine Umsetzung mit Java-Servlets entschieden, dass wir uns für Servlets entschieden haben, hat mehrere Gründe.

Einerseits war die von uns bearbeitete Praktikumsaufgabe ein guter Start für das Projekt. Das Gästebuch, mit der Anmeldeseite war ein guter Ansatz, da wir auch bei unserem Tippsspiel eine Anmeldung benötigen. Wir hatten das Gefühl, dass wir mit der Praktikumsaufgabe bereits einen Start für das Projekt hatten.

Außerdem gestaltete sich das Arbeiten mit Java-Servlets für uns intuitiv einfacher, als mit CORBA oder Java RMI, dies entspricht aber eher einer persönlichen Vorliebe. Außerdem haben wir bei der Bearbeitung der Praktikumsaufgabe bemerkt, dass uns das Arbeiten mit Java-Servlets interessiert und wir wissen wollten, welche Möglichkeiten und Funktionen Java-Servlets noch bereit stellen.

Für die Speicherung der Daten im Tippsspiel wollen wir eine, eigens erstellte, Java-Datenbank nutzen. Dies kam uns sowohl im Bezug auf die Datenspeicherung als auch auf die Datenzugriffe sehr gelegen. Gerade weil wir bereits in zwei Modulen mit JDBC gearbeitet hatten.

## 2. Projektplanung

Im Zuge einer besseren Übersichtlichkeit im Programm haben wir uns dazu entschieden, eine strikte Menüstruktur mit Untermenüs für die Hauptpunkte der Aufgabenstellung zu verwenden. Hierbei haben wir uns dazu entschieden, dass jede Funktionalität, also jedes Menü, sowie jedes Untermenü, eine eigene Servlet-Klasse erhält.

Für die Zugriffe auf die Datenbank wollen wir eine eigene Klasse erstellen, sodass beispielsweise im Benutzermenü die Klassen Benutzermenü, Punktemenü und Tippmenü auf dieselbe, gemeinsame Datenbankenklasse zugreifen.

Eine Anmeldung als Manager verweist direkt in das Managermenü, eine als Benutzer in das Benutzermenü. Aus dem Managermenü heraus können die Untermenüs Mannschaftsmenü, Turniermenü und Spielmenü erreicht werden. Die Menüs der Mannschaften und der Turniere sind hierbei sehr ähnlich aufgebaut. Hier können Mannschaften bzw. Spiele erstellt und bearbeitet werden. Außerdem können bereits vorhandene gelöscht werden.

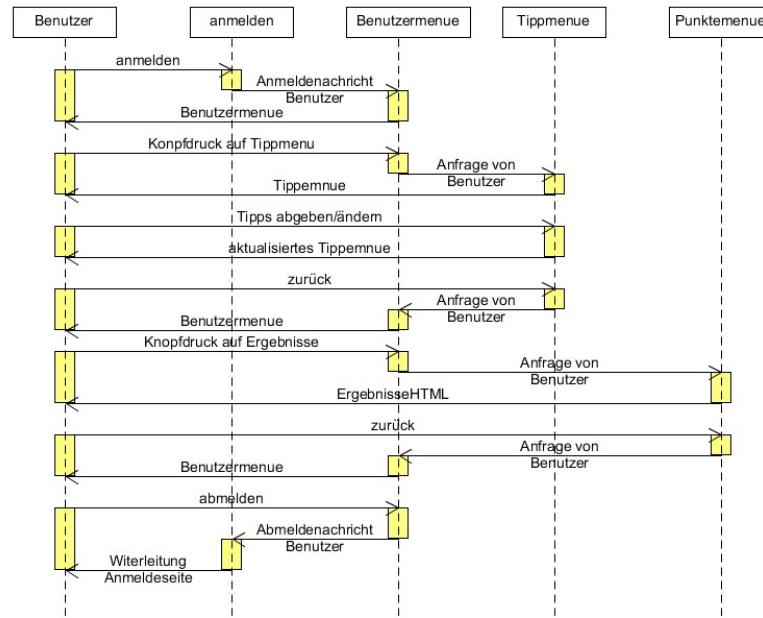
Im Spielmenü gibt es all diese Funktionalitäten auch, doch noch zusätzlich können offene Spiele gesperrt und Ergebnisse eingetragen werden. Ein weiterer Knopf „abgeschlossene Spiele“ verweist auf eine HTML-Seite, auf der alle abgeschlossenen Spiele, mit gespeichertem Ergebnis angezeigt werden. Eine Weitere Funktionalität des Managers ist eine Rangliste der Benutzer mit Punktestand abzufragen.

Aus dem Benutzermenü heraus können das Tippmenü und eine Ergebnisseite erreicht werden. Das Tippmenü bietet die Möglichkeit Spiele einzusehen, die noch offen sind, für die aber noch kein Tipp gespeichert wurde. Außerdem kann man seine bereits abgegebenen Tipps, ebenfalls für noch offene Spiele einsehen. Für noch nicht getippte Spiele kann ein Tipp abgegeben werden. Abgegebene Tipps bei offenen Spielen sind veränderbar.

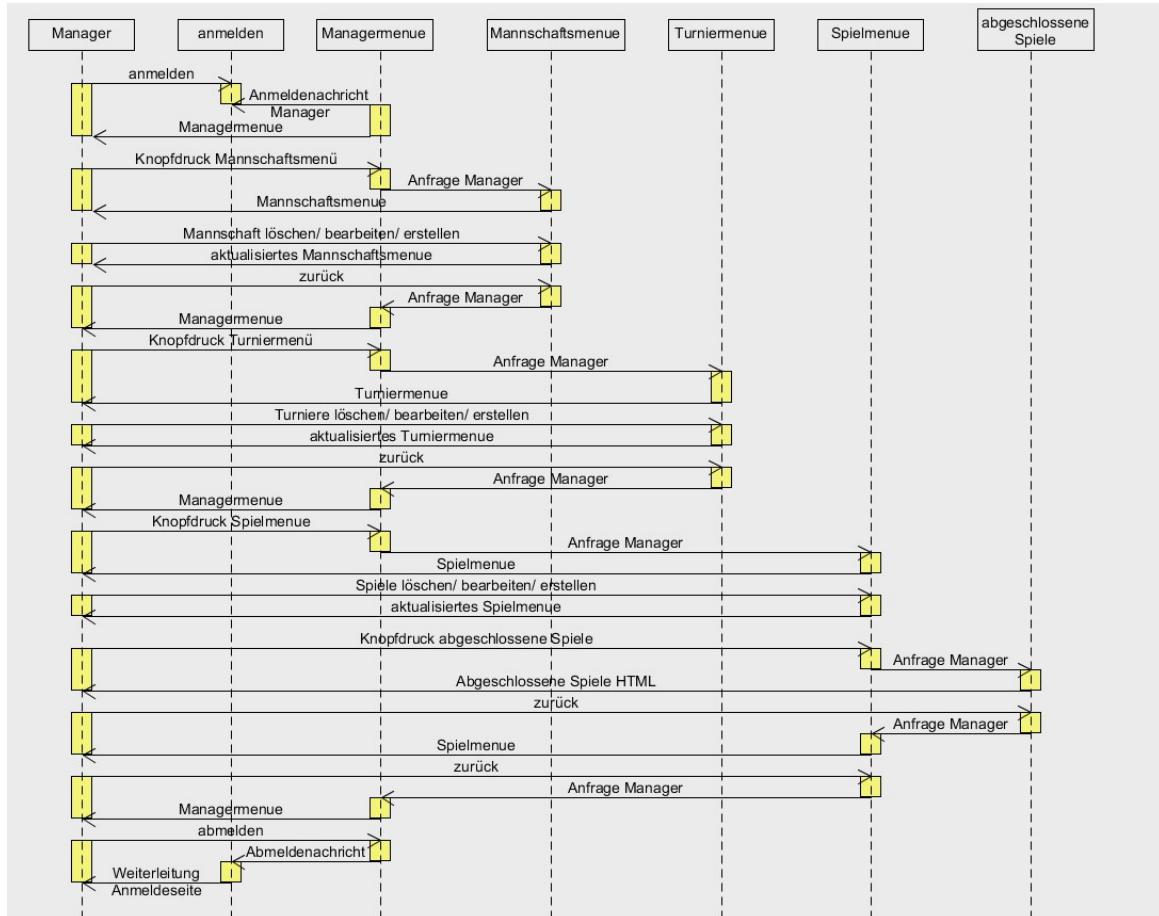
Auf der Ergebnisseite sieht der Benutzer eine Liste von Spielen, die gesperrt wurden. Hat er Tipps für diese Spiele abgegeben, werden ihm diese mitgeteilt. Außerdem kann der Benutzer seinen Punktestand und seinen Rang einsehen.

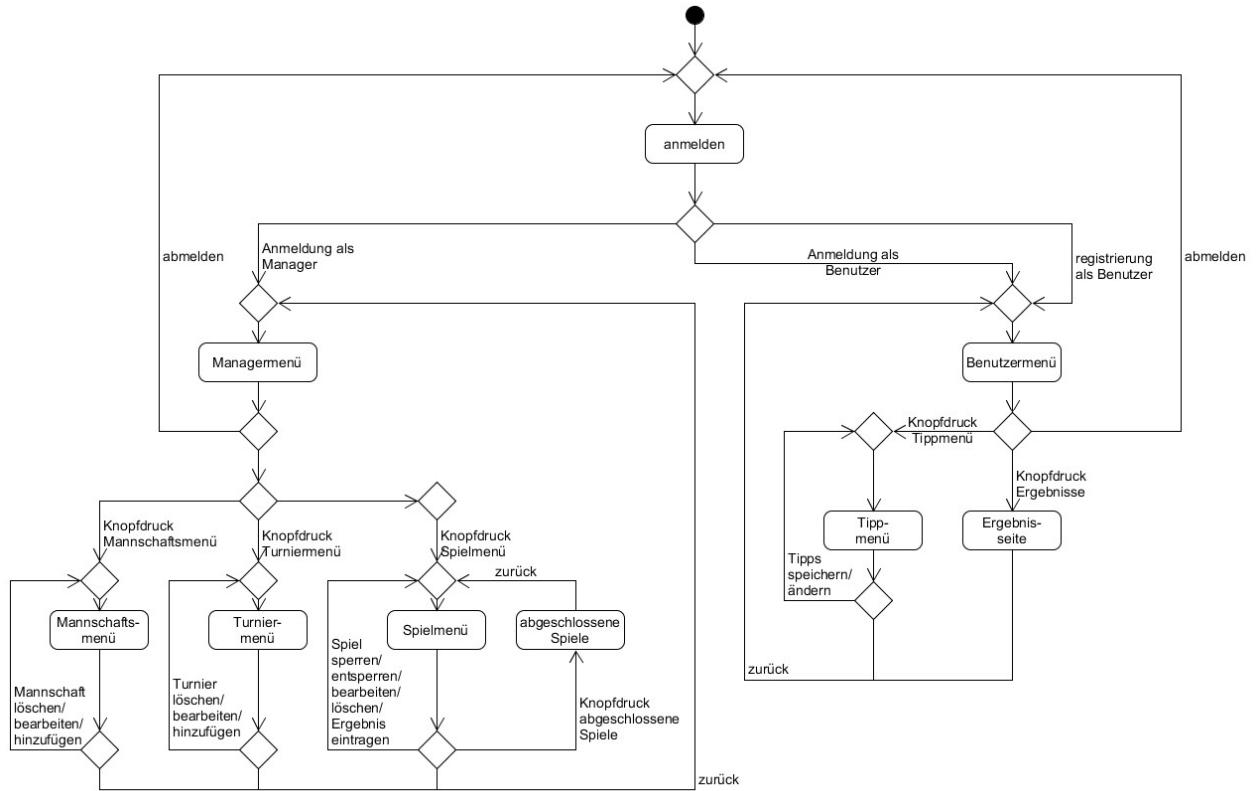
Eine Übersicht über den Ablauf des Programms zeigen Ihnen die Folgenden Sequenz- und Aktivitätsdiagramme.

Seqenzdiagramm Benutzer



Seqenzdiagramm Manager

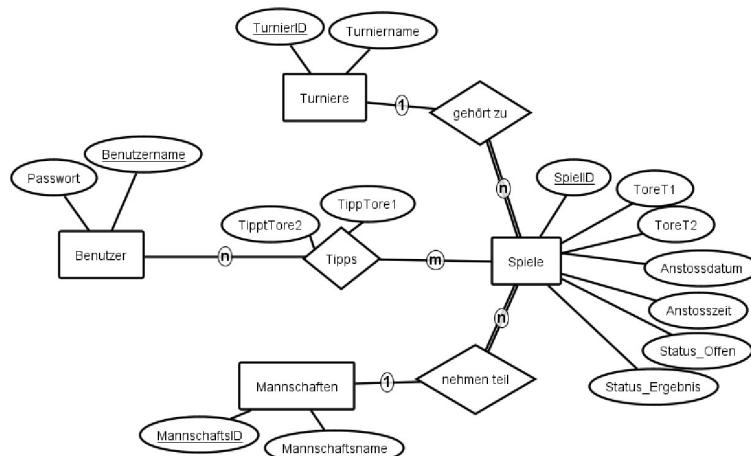




### **3. Projektdurchführung**

### 3.1 Die Datenbank

Bevor wir beginnen konnten, die Aufgabenstellung zu lösen haben wir uns Gedanken über die Datenhaltung gemacht. Da wir die Daten nicht in einem Textdokument sondern in einer Datenbank speichern wollten war sofort klar. Die Umsetzung, unserer Datenbankstruktur können Sie dem folgenden ER-Diagramm entnehmen.



Im Nachfolgenden sehen Sie die Definitionen der Datenbankrelationen der Datei „Datenbank\_Tippspiel.sql“.

```
CREATE TABLE Benutzer(
    Benutzername varchar(255) NOT NULL,
    Passwort varchar(255) NOT NULL,
    PRIMARY KEY(Benutzername)
);
```

Die Relation Benutzer besteht aus dem Schlüsselattribut, dem Benutzernamen und einem zugehörigen Passwort.

```
CREATE TABLE Manager(
    Benutzername varchar(255) NOT NULL,
    Passwort varchar(255) NOT NULL,
    PRIMARY KEY(Benutzername)
);
```

Die Relation Manager besteht aus ihrem Schlüsselattribut, dem Benutzernamen, und einem zugehörigen Passwort.

```
CREATE TABLE Mannschaften(
    MannschaftsID INT NOT NULL GENERATED ALWAYS AS IDENTITY(
        START WITH 1, INCREMENT BY 1),
    Mannschaftsname varchar(255) NOT NULL,
    PRIMARY KEY(MannschaftsID)
);
```

Die Relation Mannschaften besteht aus einem automatisch generierten Schlüsselattribut, der MannschaftsID. Außerdem enthält sie das Attribut Mannschaftsnamen.

```
CREATE TABLE Turniere(
    TurnierID INT NOT NULL GENERATED ALWAYS AS IDENTITY(
        START WITH 1, INCREMENT BY 1),
    Turniername varchar(255) NOT NULL,
    PRIMARY KEY(TurnierID)
);
```

Die Relation Turniere enthält ebenfalls eine automatisch generierte TurnierID als Schlüsselattribut. Außerdem enthält die Relation das Attribut Turniername

```
CREATE TABLE Spiele(
    SpielID INT NOT NULL GENERATED ALWAYS AS IDENTITY(
        START WITH 1, INCREMENT BY 1),
    Team1 INT NOT NULL,
    Team2 INT NOT NULL,
    ToreT1 int DEFAULT 0,
    ToreT2 int DEFAULT 0,
    Anstoßdatum Date NOT NULL,
    Anstoßzeit Time NOT NULL,
    Turnier INT NOT NULL,
    Status_Offen boolean NOT NULL DEFAULT true,
    Status_Ergebnis boolean NOT NULL DEFAULT false,
    PRIMARY KEY (SpielID),
    FOREIGN KEY (Turnier) REFERENCES Turniere(TurnierID),
    FOREIGN KEY (Team1) REFERENCES Mannschaften(MannschaftsID),
    FOREIGN KEY (Team2) REFERENCES Mannschaften(MannschaftsID)
);
```

Die Relation Spiele enthält ein Schlüsselattribut SpielID, das, wie bei Mannschaften und Turnieren, automatisch generiert wird. Aus den Relationen Turniere und Mannschaften erhalten wir die IDs als Fremdschlüsselattribute. Als weitere Attribute kommen das Anstossdatum, die Anstosszeit und die Bool-Werte Status\_Offen und Status\_Ergebnis. Außerdem werden die Attribute ToreT1 und ToreT2 um das Ergebnis zu speichern hinzugefügt.

```
CREATE TABLE Tipps (
    SpielID INT NOT NULL,
    Benutzer varchar(255) NOT NULL,
    TippTore1 int NOT NULL,
    TippTore2 int NOT NULL,
    PRIMARY KEY (SpielID,Benutzer),
    FOREIGN KEY (SpielID) REFERENCES Spiele(SpielID),
    FOREIGN KEY (Benutzer) REFERENCES Benutzer(Benutzername)
);
```

Die Relation Tipps besitzt einen Schlüssel, der sich aus den Fremdschlüsselattributen SpielID und Benutzername zusammen setzt. Die weiteren Attribute für die Relation Tipps sind die Attribute TippTore1 und TippTore2.

```
INSERT INTO Manager VALUES ('Admin', 'Admin');
INSERT INTO Manager VALUES ('Admin1', 'Admin1');
```

Da es nicht möglich sein soll ein Managerkonto während der Laufzeit des Programms zu registrieren, haben wir zwei Managerzugänge manuell hinzugefügt. Diese beiden haben die Benutzernamen „Admin“ und „Admin1“. Die Passwörter entsprechen den Benutzernamen.

## 3.2 Das Programm

Zur Initialisierung der Datenbankenverbindung haben wir eine Klasse „DBVerbindung“ erstellt. Als Klassenvariable besitzt diese Klasse ein static-Objekt von sich selbst, damit, während der Laufzeit des Programms, auf diese zugegriffen werden kann und nicht in jedem Menü ein neues Objekt erstellt werden muss.

```
public class DBVerbindung {
    public static DBVerbindung dbVerbindung = new DBVerbindung();
    private Connection connection;
    private Statement statement;
    private ResultSet resultSet;

    /** Methode fuer den Verbindungsauflauf mit der Datenbank ...4 lines */
    public DBVerbindung(){
        try{
            Class.forName("org.apache.derby.jdbc.ClientDriver");
            connection = DriverManager.getConnection("jdbc:derby://localhost:1527/Tippsspiel");
            statement = connection.createStatement();
        }
        catch(SQLException | ClassNotFoundException e){
            System.out.println(e);
        }
    }
}
```

Außer der Verbindungs methode besitzt diese Klasse nur getter und setter, damit wir auf die Datenbank zugreifen können beziehungsweise damit wir Daten aus der Datenbank auslesen können.

```
/** Getter Statement ...4 lines */
public Statement getStatement(){
    return statement;
}

/** Getter ResultSet ...4 lines */
public ResultSet getResultSet(){
    return resultSet;
}

/** Setter ResultSet ...4 lines */
public void setResultSet(ResultSet setter){
    resultSet = setter;
}
```

### 3.2.1 Die Anmeldung

Zu Beginn des Programms wird der Benutzer auf eine Anmeldeseite verwiesen. Diese haben wir auf der index.html definiert.

```
<html>
  <head>
    <title>Fussballtippspiel</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <form action="http://localhost:8080/Projektarbeit_V2/Anmelden" method="POST">
      <h2>Bitte anmelden:</h2>
      Benutzername:
      <input type="text" id="benutzername" name="benutzername" required autocomplete="off"><br>
      Passwort:
      <input type="password" id="passwort" name="passwort" required autocomplete="off"><br>
      <input type="submit" name="anmelden" value="anmelden">
      <input type="submit" name="registrieren" value="registrieren"><br>
    </form>
  </body>
</html>
```

#### Bitte anmelden:

Benutzername:

Passwort:

Hierbei gibt es drei Möglichkeiten, einmal sich als Bestehender Benutzer oder Manager anzumelden. Alternativ kann sich ein neuer Benutzer registrieren. Sowohl Benutzername als auch Passwort sind verpflichtend, wobei wir das Passwort als „Input type password“ angelegt haben, wodurch im Textfeld nur Punkte angezeigt werden.

Die Überprüfung zur Anmeldung findet dabei wie folgt statt:

```
protected void anmelden (HttpServletRequest request, HttpServletResponse response)
    throws SQLException, ServletException, IOException, ClassNotFoundException{

    String benutzername = request.getParameter("benutzername");
    String passwort = request.getParameter("passwort");

    HttpSession session = request.getSession(false);
    if(session!=null){
        session.invalidate();
    }
    session = request.getSession(true);

    if(verwaltung.anmeldungBenutzer(verbindung, benutzername, passwort)){
        session.setAttribute("user", benutzername);
        BenutzerSession benutzerDaten = (BenutzerSession) session.getAttribute("benutzerDaten");
        if (benutzerDaten == null) {
            benutzerDaten = new BenutzerSession();
            request.getSession().setAttribute("benutzerDaten", benutzerDaten);
        }

        Benutzermenue benutzer = new Benutzermenue();
        benutzer.benutzermenue(request, response);
    }
    else if(verwaltung.anmeldungManager(verbindung, benutzername, passwort)){
        session.setAttribute("user", benutzername);
        session.setAttribute("manager", verwaltung.listeManager(verbindung));
        BenutzerSession benutzerDaten = (BenutzerSession) session.getAttribute("benutzerDaten");
        if (benutzerDaten == null) {
            benutzerDaten = new BenutzerSession();
            request.getSession().setAttribute("benutzerDaten", benutzerDaten);
        }
        Managermenue manager = new Managermenue();
        manager.managerermenue( request, response);
    }
    else{
        fehlerAnmeldung(request, response);
    }
}
```

Hier werden zuerst die Daten aus Benutzername und Passwort der index.html ausgelesen.

Danach geschieht eine Session-Abfrage, in dieser wird geprüft, ob bereits eine Session mit der gleichen Session-ID existiert. Sollte das der Fall sein, wird diese geschlossen und eine neue Session wird erstellt.

Anschließend folgt eine Überprüfung, ob der Nutzer die richtige Kombination aus Benutzername und Passwort eingegeben hat. Diese sieht wie folgt aus:

```
public boolean anmeldungBenutzer(DBVerbindung dbVerbindung, String benutzername, String passwort)
    throws SQLException{

    dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
        "SELECT Benutzername, Passwort FROM Benutzer"));

    while(dbVerbindung.getResultSet().next()){
        if(benutzername.equals(dbVerbindung.getResultSet().getString("Benutzername"))){
            if(passwort.equals(dbVerbindung.getResultSet().getString("Passwort"))){
                return true;
            }
        }
    }
    return false;
}
```

```
public boolean anmeldungManager(DBVerbindung dbVerbindung, String benutzername, String passwort)
    throws SQLException {

    dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
        "SELECT Benutzername, Passwort FROM Manager"));

    while(dbVerbindung.getResultSet().next()) {
        if(benutzername.equals(dbVerbindung.getResultSet().getString("Benutzername"))){
            if(passwort.equals(dbVerbindung.getResultSet().getString("Passwort"))){
                return true;
            }
        }
    }

    return false;
}
```

Zunächst wird überprüft, ob der Nutzer in den Benutzerdaten steht, ist dies nicht der Fall, wird in den Managerdaten geprüft. Dies entscheidet, in welches Menü später weitergeleitet wird.

Soll ein Manager angemeldet werden, erstellen wir das Sessionattribut „manager“, in dem wir eine Liste mit den Benutzernamen der Manager speichern. Dies ist für die Prüfung im ValueBound wichtig, welche nachfolgend erklärt wird. Die Liste mit den Benutzernamen der Manager erhalten wir von der Funktion „listeManager“.

```
public ArrayList<String> listeManager(DBVerbindung dbVerbindung)
    throws SQLException{
    ArrayList<String> list;
    list = new ArrayList<>();

    dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
        "SELECT Benutzername FROM Manager"));

    while(dbVerbindung.getResultSet().next()){
        list.add(dbVerbindung.getResultSet().getString("Benutzername"));

    }
    return list;
}
```

Nach erfolgreichem Abgleich der Anmeldedaten mit den Daten der Datenbank erstellen wir ein BenutzerSession-Objekt und setzten dieses als Attribut in unsere aktuelle Session. Dadurch wird die valueBound-Methode unserer HttpSessionBindingListener-Klasse aufgerufen.

```
@Override
public void valueBound(HttpSessionBindingEvent event) {

    HttpSession session = event.getSession();
    benutzer=(String) session.getAttribute("user");
    manager=(ArrayList<String>) session.getAttribute("manager");

    if(manager!=null){
        for (BenutzerSession i : logins.keySet()) {
            for(int j=0; j<manager.size(); j++){
                System.out.println(manager.get(j));
                if(i.benutzer.equals(manager.get(j))){
                    System.out.println("Es ist bereits ein anderer Manager "
                            + "eingeloggt....wird ausgeloggert");
                    logins.get(i).invalidate();
                }
            }
        }
    }

    else{
        for (BenutzerSession i : logins.keySet()) {
            if(i.benutzer.equals(benutzer)){
                System.out.println("User bereits eingeloggt...."
                        + "wird ausgeloggert");
                logins.get(i).invalidate();
            }
        }
    }

    logins.put(this, event.getSession());
    System.out.println("Größe logins: " + logins.size());
    session.setMaxInactiveInterval(60);
    for (BenutzerSession k : logins.keySet()) {
        System.out.println("Eingeloggte User: " + k.benutzer );
    }
}
```

In der ValueBound-Methode wird für einen sich anmeldenden Benutzer geprüft, ob dieser bereits über eine andere aktive Session verfügt. Sollte dies der Fall sein, wird die bereits bestehende Session des Benutzers ungültig und die neue Session wird in die Map der aktiven Logins gespeichert.

Versucht hingegen ein Manager sich anzumelden, wird überprüft, ob irgendein anderer Manager in einer aktiven Session ist. Ist dies der Fall, wird die bereits bestehende Session abgemeldet und eine neue Session gestartet. So können wir verhindern, dass ein Manager, der zum Beispiel sein Browserfenster geschlossen hat, ohne sich abzumelden, nicht für immer die Anmeldeversuche anderer Manager, oder sich selbst, blockiert.

Außerdem legt die Funktion das Intervall fest, ab wann ein Nutzer als inaktiv gilt. In unserem Fall hier ist diese Zeit auf 60 Sekunden festgelegt.

Sollte während des Programmverlaufs eine Session auslaufen, durch Inaktivität oder Neuanmeldung, wird auf die folgende Fehlerseite verwiesen.

```
protected void fehlermeldungSession(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
response.setContentType("text/html; charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println(" <head>");
    out.println(" <title>Fehlermeldung Session</title>");
    out.println(" </head>");
    out.println(" <body>");
    out.println(" <form action=\"http://localhost:8080/Projektarbeit_V2/SessionFehler\" method=\"POST\">");
    out.println(" <h2>Fehler</h2><BR>");
    out.println("Leider ist Ihre Session ausgelaufen<BR>");
    out.println(" </form>");
    out.println(" </body>");
    out.println(" </html>");
}
}
```

## Fehler

Leider ist Ihre Session ausgelaufen

Nach dem Abgleich der Anmeldedaten mit der Datenbank und dem hinzufügen der neuen Session zu den aktiven Logins wird ein Benutzer an das Benutzermenü und ein Manager an das Managermenü weitergeleitet.

Will sich ein Nutzer jedoch registrieren, geschehen dieselben Überprüfungen bezüglich der Sessions, jedoch ist die Überprüfung der Anmeldedaten anders. Da ein Benutzer, der sich neu registrieren möchte einen eindeutigen Benutzernamen haben muss.

```
protected void registrieren (HttpServletRequest request, HttpServletResponse response)
    throws IOException, SQLException, ClassNotFoundException, ServletException {

    String benutzername = request.getParameter("benutzername");
    String password = request.getParameter("password");

    HttpSession session = request.getSession(false);
    if(session!=null){
        session.invalidate();
    }
    session = request.getSession(true);

    if(verwaltung.benutzernamePruefen(verbinding, benutzername)){
        verwaltung.benutzerHinzufuegen(verbinding, benutzername, password);

        session.setAttribute("user", benutzername);
        BenutzerSession userData = (BenutzerSession) session.getAttribute("userData");
        if (userData == null) {
            userData = new BenutzerSession();
            request.getSession().setAttribute("userData", userData);
        }

        Benutzermenue benutzer = new Benutzermenue();
        benutzer.benutzermenue(request, response);
    }
    else{
        fehlerRegistrierung(request, response);
    }
}
```

```
public boolean benutzernamePruefen(DBVerbindung dbVerbindung, String benutzername)
throws SQLException{

    dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
        "SELECT Benutzername FROM Benutzer"));

    while(dbVerbindung.getResultSet().next()){
        if(benutzername.equals(dbVerbindung.getResultSet().getString("Benutzername"))){
            return false;
        }
    }
    dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
        "SELECT Benutzername FROM Manager"));

    while(dbVerbindung.getResultSet().next()){
        if(benutzername.equals(dbVerbindung.getResultSet().getString("Benutzername"))){
            return false;
        }
    }
    return true;
}

public void benutzerHinzufuegen(DBVerbindung dbVerbindung, String benutzername, String passwort)
throws SQLException{
    dbVerbindung.getStatement().executeUpdate(
        "insert into benutzer values("
        + "'" + benutzername + "','" + passwort + "')");

}
}
```

Der neue Benutzername darf vorher weder bei einem anderen Benutzer noch bei einem Manager verwendet worden sein. Ist dies der Fall, wird der Benutzer mit seinem gewählten Passwort, mit der Methode „benutzerHinzufuegen“ in die Datenbank gespeichert. Danach wird er als angemeldeter Benutzer in das Benutzermenü weitergeleitet.

Versucht sich ein Benutzer zu registrieren, verwendet jedoch einen Namen, der bereits von einem Admin oder einem Benutzer verwendet wird, erscheint die folgende Fehlerseite:

```
protected void fehlerRegistrierung(HttpServletRequest request, HttpServletResponse response)
throws IOException{
    response.setContentType("text/html;charset=UTF-8");
    try(PrintWriter out = response.getWriter()){
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Fehler Registrierung</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit\_V2/Anmelden\" method = \"POST\">");

        out.println("<h2>Fehler</h2><br>");
        out.println("Fehler beim Registrieren.<br> Benutzername wird bereits verwendet.<br>");

        out.println("<input type=\"submit\" name=\"zurueck\" value=\"zurueck\" checked><br><br>");

        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

## Fehler

Fehler beim Registrieren.  
Benutzername wird bereits verwendet.  
[zurück](#)

Hat sich ein Nutzer bei der Anmeldung verschrieben, oder ein falsches Passwort angegeben wird er zu der folgenden Fehlerseite weitergeleitet.

```
protected void fehlerAnmeldung(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {

        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Fehler Anmeldung</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit_V2/Anmelden\" method = \"POST\">");

        out.println("<h2>Fehler</h2><BR>");
        out.println("Fehler beim Anmelden, Benutzername oder Passwort falsch<BR>");

        out.println("<input type=\"submit\" name=\"zurueck\" value=\"zurück\"><BR><BR>");

        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

## Fehler

Fehler beim Anmelden, Benutzername oder Passwort falsch

[zurück](#)

### 3.2.2 Das Benutzermenü

Die Startseite, die ein Benutzer direkt nach der Anmeldung sieht ist die folgende:

```
public void benutzermenue(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Benutzermenue</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit_V2/Benutzermenue\" method=\"POST\">");

        out.println("<h2> Aktivitäten auswählen</h2>");
        out.println("Tipps verwalten und hinzufügen: <BR>");

        out.println("<input type=\"submit\" name=\"tippmenue\" value=\"Tippsmenü\"><BR><BR>"); 
        out.println("Ergebnisse und Punktestände abfragen: <BR>");

        out.println("<input type=\"submit\" name=\"punktemenue\" value=\"Ergebnisse\"><BR><BR>"); 
        out.println("<input type=\"submit\" name=\"abmelden\" value=\"abmelden\"><BR><BR>");

        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

## Aktivität auswählen

Tipps verwalten und hinzufügen:  
[Tippsmenü](#)

Ergebnisse und Punktestände abfragen:  
[Ergebnisse](#)

[abmelden](#)

Der Aufbau hier ist eine einfache HTML-Seite, mit drei Auswahlmöglichkeiten. Von hier aus kann der Benutzer alle, für ihn möglichen Funktionalitäten erreichen. Die Funktionalitäten hinter den Buttons werden in den nachfolgenden Kapiteln erläutert.

Drückt man den Button „abmelden“ im Benutzermenü wird der Benutzer abgemeldet und die Session wird beendet. Dadurch wird die valueUnbound-Methode der HttpSessionBindingListener-Klasse aufgerufen und die Session des Benutzers aus der Map mit den aktiven Logins entfernt.

```
@Override
public void valueUnbound(HttpSessionBindingEvent event) {

    logins.remove(this);
    System.out.println("Größe logins: " + logins.size());
    for (BenutzerSession k : logins.keySet()) {
        System.out.println("Eingeloggte User: " + k.benutzer );
    }
}
```

Nach erfolgreicher Abmeldung wird die index.html erneut aufgerufen.

## Das Tippmenü

Durch das Betätigen des Buttons „Tippmenü“ gelangt der Benutzer in das Tippmenü. Im Tippmenü sind alle Spiele aufgeführt, die noch offen, also noch Tippbar sind.

### Spiele, auf die Sie noch nicht getippt haben:

<input type="checkbox"/> Hoffenheim : Moenchengladbach 2019-09-30 18:00:00	[0]	[+]	[0]	[+]
<input type="checkbox"/> Union Berlin : SGE 2019-09-30 18:00:00	[0]	[+]	[0]	[+]
<input type="checkbox"/> Fortuna Duesseldorf : Freiburg 2019-09-30 18:00:00	[0]	[+]	[0]	[+]
<input type="checkbox"/> 1.FC Koeln : Hertha 2019-09-30 18:00:00	[0]	[+]	[0]	[+]
<input type="checkbox"/> Augsburg : Leverkusen 2019-09-30 18:00:00	[0]	[+]	[0]	[+]
<input type="checkbox"/> Mainz 05 : Wolfsburg 2019-09-30 18:00:00	[0]	[+]	[0]	[+]

Tipps speichern

### Ihre noch änderbaren Tipps:

<input type="checkbox"/> RB Leipzig : Schalke 2019-09-30 18:00:00	[0]	[+]	[0]	[+]
<input type="checkbox"/> BVB : Werder Bremen 2019-09-30 18:00:00	[5]	[+]	[3]	[+]
<input type="checkbox"/> SC Paderborn : FC Bayern 2019-09-30 18:00:00	[1]	[+]	[3]	[+]

Tipps ändern

zurück

```

protected void tippmenu(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, SQLException, ParseException {
HttpSession session = request.getSession(false);
if(session!=null) {

Pruefmethoden pruefmethoden = new Pruefmethoden();
ArrayList<ArrayList<String>> list;
list = new ArrayList<>();
string id;
String benutzername;

benutzername= (String) session.getAttribute("user");
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>" + benutzername + "</title>");
out.println("</head>");
out.println("<body>");
out.println("<form action=\"http://localhost:8080/Projektarbeit_V2/Tippmenu\" method=\"POST\">");

out.println("<h2>Spiele, auf die Sie noch nicht getippt haben: </h2>");

list = benutzerDB.spieleNichtGetippt(verbindungDB, benutzername);
for (int i=0; i<list.size(); i++) {
if(pruefmethoden.anstosszeitPruefen(list.get(i).get(2), list.get(i).get(3))){
id=list.get(i).get(4);
out.println("<input type=\"checkbox\" name=\"nicht_getippt\" value=\"" + id + "\"> ");
out.println(list.get(i).get(0));
out.println(" " + list.get(i).get(1));
out.println(" " + list.get(i).get(2));
out.println(" " + list.get(i).get(3));
out.println("<input type=\"number\" id=\"tippl_1\" id+\"\" name=\"tippl_1\" id+\"\" min=\"0\" step=\"1\" value=\"0\"> ");
out.println(" ");
out.println("<input type=\"number\" id=\"tippl_2\" id+\"\" name=\"tippl_2\" id+\"\" min=\"0\" step=\"1\" value=\"0\"><br> ");
}
}

out.println("<input type=\"submit\" name=\"tipp_speichern\" value=\"Tipps speichern\">");

out.println("<h2>Ihre noch gesunderbaren Tipps: </h2>");

list = benutzerDB.spieleGetippt(verbindungDB, benutzername);
for (int i=0; i<list.size(); i++) {
if(pruefmethoden.anstosszeitPruefen(list.get(i).get(2), list.get(i).get(3))){
String tipptore1;
String tipptore2;
id=list.get(i).get(6);
out.println("<input type=\"checkbox\" name=\"neu_getippt\" value=\"" + id + "\"> ");
out.println(list.get(i).get(0));
out.println(" " + list.get(i).get(1));
out.println(" " + list.get(i).get(2));
out.println(" " + list.get(i).get(3));
tipptore1 = list.get(i).get(4);
tipptore2 = list.get(i).get(5);
out.println("<input type=\"number\" id=\"tippl_1\" id+\"\" name=\"tippl_1\" id+\"\" min=\"0\" step=\"1\" value=\"0\"> ");
out.println(" ");
out.println("<input type=\"number\" id=\"tippl_2\" id+\"\" name=\"tippl_2\" id+\"\" min=\"0\" step=\"1\" value=\"0\"> ");
}
}

out.println("<input type=\"submit\" name=\"Tipp_aendern\" value=\"Tipps ändern\"><br><br> ");
out.println("<input type=\"submit\" name=\"zurueck_Benutzermenue\" value=\"zurückschick\"><br> ");

out.println("</form>");
out.println("</body>");
out.println("</html>");

}
}

```

Hierbei werden sowohl die noch nicht getippten Spiele, als auch die Spiele, auf die bereits ein Tipp abgegeben wurde, mit dem abgegebenen Tipps angezeigt. Die dafür erforderliche Liste erhalten wir aus den Methoden „spieleNichtGetippt“ und „spieleGetippt“.

```

public ArrayList<ArrayList<String>> spieleNichtGetippt (DBVerbindung dbVerbindung, String benutzername)
throws SQLException{
ArrayList<ArrayList<String>> list;
list = new ArrayList<>();

dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
    "SELECT t1.Mannschaftsname, t2.Mannschaftsname, Spiele.Anstosssdatum, Spiele.Anstosszeit, Spiele.SpielID FROM Spiele "
    + "JOIN Mannschaften AS t1 ON t1.mannschaftsid = Spiele.Team1 "
    + "JOIN Mannschaften AS t2 ON t2.mannschaftsid = Spiele.Team2 "
    + "WHERE Spiele.Status_offen=true "
    + "AND Spiele.SpielID NOT IN( "
    + "SELECT SpielID FROM Tipps "
    + "WHERE Benutzer='"+benutzername+"')");
}

while(dbVerbindung.getResultSet().next()){
    ArrayList<String> temp = new ArrayList<>();
    temp.add(dbVerbindung.getResultSet().getString(1));
    temp.add(dbVerbindung.getResultSet().getString(2));
    temp.add(dbVerbindung.getResultSet().getString("Anstosssdatum"));
    temp.add(dbVerbindung.getResultSet().getString("Anstosszeit"));
    temp.add(dbVerbindung.getResultSet().getString("SpielID"));
    list.add(temp);
}
return list;
}

public ArrayList<ArrayList<String>> spieleGetippt (DBVerbindung dbVerbindung, String benutzername)
throws SQLException{
ArrayList<ArrayList<String>> list;
list = new ArrayList<>();

dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
    "SELECT t1.Mannschaftsname, t2.Mannschaftsname, Spiele.Anstosssdatum, Spiele.Anstosszeit, "
    + "Tipps.Tippore1, tipps.Tippore2, Spiele.SpielID FROM Spiele "
    + "JOIN Tipps ON Tipps.SpielID=Spiele.SpielID "
    + "JOIN Mannschaften AS t1 ON t1.mannschaftsid = Spiele.Team1 "
    + "JOIN Mannschaften AS t2 ON t2.mannschaftsid = Spiele.Team2 "
    + "WHERE Spiele.Status_Offen=true "
    + "AND Benutzer='"+benutzername+"')");
}

while(dbVerbindung.getResultSet().next()){
    ArrayList<String> temp = new ArrayList<>();
    temp.add(dbVerbindung.getResultSet().getString(1));
    temp.add(dbVerbindung.getResultSet().getString(2));
    temp.add(dbVerbindung.getResultSet().getString("Anstosssdatum"));
    temp.add(dbVerbindung.getResultSet().getString("Anstosszeit"));
    temp.add(dbVerbindung.getResultSet().getString("Tippore1"));
    temp.add(dbVerbindung.getResultSet().getString("Tippore2"));
    temp.add(dbVerbindung.getResultSet().getString("SpielID"));
    list.add(temp);
}
return list;
}

```

Hier können neue Tipps abgegeben, oder bestehende Tipps geändert werden. Es ist möglich einen oder mehrere Tipps gleichzeitig abzugeben oder zu ändern. Zuerst wird der Benutzername aus der Session ausgelesen, so wie wir ihn in der anmelden oder registrieren Methode gespeichert haben. Dies wird noch wichtig, beim Auslesen der Tipps. Die Funktionen „spieleNichtGetippt“ und „SpieleGetippt“ geben Mehrdimensionale ArrayListen zurück, jede Zeile steht hierbei für ein Spiel. Die Spalten sind, der Reihe nach: Mannschaftsname Mannschaft1, Mannschaftsname Mannschaft2, Anstosssdatum, Anstosszeit und SpielID. Sobald wir diese Ergebnislisten im Tippmenü zurückerhalten gehen wir sie durch und geben die Daten geordnet aus. Im Anschluss steht ein Nummernfeld, das so definiert ist, dass keine Werte unter Null darin stehen können, genauso wenig wie Kommazahlen. Dies erspart uns spätere Überprüfungen, die wir, bei einem reinen, undefinierten, Textfeld machen müssten. Um auszuschließen, dass der Benutzer auf Spiele tippen kann, bei denen bereits der Anstoß erfolgt ist, prüfen wir, vor der Ausgabe der Spiele, ob die Jetztzeit vor der Anstosszeit liegt. Dies geschieht in der Funktion „anstosszeitPruefen“ der Klasse „Pruefmethoden“.

```

public boolean anstosszeitPruefen(String anstossDatum, String anstossZeit)
throws IOException, ParseException {

    SimpleDateFormat datumFormat = new SimpleDateFormat("yyyy-MM-dd");
    String stringHeute = datumFormat.format(new Date());

    Date heute = datumFormat.parse(stringHeute);
    Date parseAnstossDatum = datumFormat.parse(anstossDatum);

    SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm:ss");
    String stringZeit = timeFormat.format(new Date());

    Date jetzt = timeFormat.parse(stringZeit);
    Date parseAnstossZeit = timeFormat.parse(anstossZeit);

    if(parseAnstossDatum.before(heute)) {
        return false;
    }
    else if(parseAnstossDatum.equals(heute) & parseAnstossZeit.before(jetzt)) {
        return false;
    }
    return true;
}

```

Das Spiel wird nicht gesperrt, da der Benutzer keine Rechte hierfür hat, es wird ihm jedoch nicht mehr angezeigt. Da es auch Spiele gibt, auf die der Benutzer bereits getippt hat, die aber noch nicht gesperrt wurden, kann der Benutzer seine Tipps hier noch ändern. Seine, bereits abgegebenen, Tipps werden als Default Werte in die Nummernfelder eingetragen und werden ihm so mitgeteilt. Auch hier geschieht eine Überprüfung nach der Jetztzeit, um Tippen auf bereits begonnene Spiele zu verhindern. Sollte ein Benutzer bereits längere Zeit auf der Seite verbracht haben, bevor er einen Tipp abgibt und in dieser Zeit ein Spiel begonnen haben, prüfen wir noch einmal im ProzessRequest, bevor ein Tipp gespeichert wird.

```

 HttpSession session = request.getSession(false);
 if( session == null ) fehlermeldungSession(request, response);

 if(request.getParameter("tipp_speichern") != null){
    String tippl;
    String tipp2;
    String[] temp = request.getParameterValues("nicht_getippt");
    String benutzername= (String) session.getAttribute("user");
    Pruefmethoden pruefmethoden = new Pruefmethoden();

    if(temp==null) fehlermeldung(request, response);
    for(int i=0; i<temp.length; i++){
        ArrayList<String> zeit=benutzerDB.getZeitID(verbindungDB, temp[i]);
        if(pruefmethoden.anstosszeitPruefen(zeit.get(0), zeit.get(1))){
            tippl = request.getParameter("tippl "+temp[i]);
            tipp2 = request.getParameter("tipp2 "+temp[i]);
            if(tippl.isEmpty()||tipp2.isEmpty()) fehlermeldung(request, response);
            benutzerDB.tippSpeichern(verbindungDB, temp[i], tippl, tipp2, benutzername);
        }
    }
    tippmenue(request, response);
}

else if(request.getParameter("tipp_aendern") != null){
    String tippl;
    String tipp2;
    String[] temp = request.getParameterValues("neu_getippt");
    String benutzername= (String) session.getAttribute("user");
    Pruefmethoden pruefmethoden = new Pruefmethoden();

    if(temp==null) fehlermeldung(request, response);
    for(int i=0; i<temp.length;i++){
        ArrayList<String> zeit=benutzerDB.getZeitID(verbindungDB, temp[i]);
        if(pruefmethoden.anstosszeitPruefen(zeit.get(0), zeit.get(1))){
            tippl = request.getParameter("tippl "+temp[i]);
            tipp2 = request.getParameter("tipp2 "+temp[i]);
            if(tippl.isEmpty()||tipp2.isEmpty()) fehlermeldung(request, response);
            benutzerDB.tippAendern(verbindungDB, temp[i], tippl, tipp2, benutzername);
        }
    }
    tippmenue(request, response);
}

```

Für diese Prüfung benötigen wir das Anstossdatum und die Anstosszeit der zu Prüfenden Spiele. Diese erhalten wir in der Methode „getZeitID“.

```
public ArrayList<String> getZeitID (DBVerbindung dbVerbindung, String id)
throws SQLException{
ArrayList<String> list;
list = new ArrayList<>();

dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
"SELECT Spiele.Anstossdatum, Spiele.Anstosszeit FROM Spiele "
+ "WHERE Spiele.SpielID='"+id+" '");

while(dbVerbindung.getResultSet().next()){
list.add(dbVerbindung.getResultSet().getString("Anstossdatum"));
list.add(dbVerbindung.getResultSet().getString("Anstosszeit"));
}
return list;
}
```

Dieselbe Prüfung findet auch beim Ändern eines Tipps statt.

```
/** Tipp in die Datenbank speichern ...10 lines */
public void tippSpeichern(DBVerbindung verb, String id, String torel, String tote2, String benutzername)
throws SQLException{

verb.getStat().executeUpdate(
"INSERT INTO Tipps "
+ "VALUES ("+id+", "
+ "'"+benutzername+"', "
+ "'"+torel+"', "
+ "'"+tote2+"')");

}

/** Methode zum ändern bereits gespeicherter Tipps ...10 lines */
public void tippAendern (DBVerbindung verb, String id, String torel, String tote2, String benutzername)
throws SQLException{
verb.getStat().executeUpdate(
"UPDATE Tipps SET Tipptore1='"+torel+"', "
+ "Tipptore2='"+tote2+" ' "
+ "WHERE SpielID='"+id+" ' "
+ "AND Benutzer ='"+benutzername+"'");
```

Ist die Prüfung erfolgreich wird entweder der neue Tipp gespeichert oder ein bereits vorhandener Tipp aktualisiert. Durch Klick auf den Button „zurück“ gelangt man wieder in das Benutzerobermenü.

## Ergebnisse

Drückt man aus dem Benutzerobermenü heraus den Knopf „Ergebnisse“, so gelangt man in das Punktemenü.

```
protected void punktmenu(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException, SQLException {

    ArrayList<ArrayList<String>> list;
    list = new ArrayList<>();

    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Ergebnisse</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit_V2/Punktemenu\" method=\"POST\">");

        HttpSession session = request.getSession(false);
        String benutzername= (String) session.getAttribute("user");

        out.println("<table>");
        out.println("<tr>");
        out.println("<th>Team1</th>");
        out.println("<th>Team2</th>");
        out.println("<th>Datum</th>");
        out.println("<th>Uhrzeit</th>");
        out.println("<th>Tore1</th>");
        out.println("<th>Tore2</th>");
        out.println("<th>Ihr Tipp1</th>");
        out.println("<th>Ihr Tipp2</th>");
        out.println("</tr>");

        out.println("<h1> gesperrte Spiele, ohne Ergebnis: </h1>");

        list = benutzerDB.spieleOhneErgebnis(verbindingDB, benutzername);
        for(int i=0; i<list.size(); i++){
            out.println("<tr>");
            out.println("<td>" + list.get(i).get(0) + "</td>");
            out.println("<td>" + list.get(i).get(1) + "</td>");
            out.println("<td>" + list.get(i).get(2) + "</td>");
            out.println("<td>" + list.get(i).get(3) + "</td>");
            out.println("<td>-</td>");
            out.println("<td>-</td>");
            if(list.get(i).get(4).equals("-1") && list.get(i).get(5).equals("-1")){
                out.println("<td>-</td>");
                out.println("<td>-</td>");
            }
            else{
                out.println("<td>" + list.get(i).get(4) + "</td>");
                out.println("<td>" + list.get(i).get(5) + "</td>");
            }
            out.println("</tr>");

        }

    }

}
```

```

    }
    out.println("</table><BR>");
    out.println("<h1> abgeschlossene Spiele: </h1>");
    out.println("<table>");
    out.println("<tr>");
    out.println("<th>Team1</th>");
    out.println("<th>Team2</th>");
    out.println("<th>Datum</th>");
    out.println("<th>Uhrzeit</th>");
    out.println("<th>Tore1</th>");
    out.println("<th>Tore2</th>");
    out.println("<th>Ihr Tipp1</th>");
    out.println("<th>Ihr Tipp2</th>");
    out.println("</tr>");

    list = benutzerDB.spieleAbgeschlossen(verbindungDB, benutzername);
    for(int i=0; i<list.size(); i++){
        out.println("<tr>");
        out.println("<td>" + list.get(i).get(0) + "</td>");
        out.println("<td>" + list.get(i).get(1) + "</td>");
        out.println("<td>" + list.get(i).get(2) + "</td>");
        out.println("<td>" + list.get(i).get(3) + "</td>");
        out.println("<td>" + list.get(i).get(4) + "</td>");
        out.println("<td>" + list.get(i).get(5) + "</td>");
        if(list.get(i).get(6).equals("-1") && list.get(i).get(7).equals("-1")){
            out.println("<td>-</td>");
            out.println("<td>-</td>");
        }
        else{
            out.println("<td>" + list.get(i).get(6) + "</td>");
            out.println("<td>" + list.get(i).get(7) + "</td>");
        }
        out.println("</tr>");

    }
    out.println("</table><BR>");

    list = benutzerDB.rangliste(verbindungDB);
    for(int i=0; i<list.size(); i++){
        if(list.get(i).get(0).equals(benutzername)){
            if(list.get(i).get(1) == null){
                out.println("Ihr Punktstand beträumt: 0<BR>");
            }
            else{
                out.println("Ihr Punktstand beträumt: " + list.get(i).get(1) + "<BR>");
            }
            out.println("Ihr Rang ist: " + (i+1) + " von " + (list.size()) + " Tippern <BR>");
        }
    }

    out.println("<input type=\"submit\" name=\"zurueck\" value=\"zuräum;ck\"><BR>");

    out.println("</form>");
    out.println("</body>");

    out.println("</html>");

}

```

### gesperrte Spiele, ohne Ergebnis:

Team1	Team2	Datum	Uhrzeit	Tore1	Tore2	Ihr Tipp1	Ihr Tipp2
Werder Bremen	Augsburg	2019-08-30	18:00:00	-	-	-	-
Union Berlin	BVB	2019-08-30	18:00:00	-	-	-	-
Schalke	Hertha	2019-08-30	18:00:00	-	-	-	-
Freiburg	1.FC Koeln	2019-08-30	18:00:00	-	-	-	-
Wolfsburg	SC Paderborn	2019-08-30	18:00:00	-	-	-	-
FC Bayern	Mainz 05	2019-08-30	18:00:00	-	-	-	-
RB Leipzig	Moenchengladbach	2019-08-30	18:00:00	-	-	-	-
Leverkusen	Hoffenheim	2019-08-31	18:00:00	-	-	-	-
FC Bayern	Mainz 05	2019-08-31	18:00:00	-	-	-	-
BVB	Werder Bremen	2019-09-30	18:00:00	-	2	2	2

### abgeschlossene Spiele:

Team1	Team2	Datum	Uhrzeit	Tore1	Tore2	Ihr Tipp1	Ihr Tipp2
FC Bayern	Hertha	2019-08-16	18:00:00	2	-	-	-
FC Bayern	Hertha	2019-08-16	18:00:00	2	2	2	2
Werder Bremen	Fortuna Duesseldorf	2019-08-17	18:00:00	1	3	-	-
Moenchengladbach	Schalke	2019-08-17	18:00:00	0	0	-	-
Leverkusen	SC Paderborn	2019-08-17	18:00:00	3	2	-	-
Wolfsburg	1.FC Koeln	2019-08-17	18:00:00	2	1	2	1
BVB	Augsburg	2019-08-17	18:00:00	5	1	5	1
Freiburg	Mainz 05	2019-08-17	18:00:00	3	0	3	0
Union Berlin	RB Leipzig	2019-08-18	18:00:00	0	4	-	-
SGE	Hoffenheim	2019-08-18	18:00:00	1	0	-	-
1.FC Koeln	BVB	2019-08-23	18:00:00	1	3	-	-
Mainz 05	Moenchengladbach	2019-08-24	18:00:00	1	3	-	-
SC Paderborn	Freiburg	2019-08-24	18:00:00	1	3	-	-
Hoffenheim	Werder Bremen	2019-08-24	18:00:00	3	2	-	-

Ihr Punktstand beträgt: 12

Ihr Rang ist: 1 von 3 Tippern

[zurück](#)

Hier können alle Spiele eingesehen werden, die geschlossen wurden. Das gilt für Spiele, für die bereits ein Ergebnis eingetragen wurde und für Spiele, bei denen dies noch nicht geschehen ist. Die Listen für diese Spiele bekommen wir durch die Funktionen „spieleOhneErgebnis“ und „spieleAbgeschlossen“.

```

public ArrayList<ArrayList<String>> spieleOhneErgebnis (DBVerbindung dbVerbindung, String benutzername)
    throws SQLException{
    ArrayList<ArrayList<String>> list;
    list = new ArrayList<>();

    dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
        "(SELECT t1.Mannschaftsname, t2.Mannschaftsname, "
        + "Spiele.Anstoessdatum, Spiele.Anstoesszeit, tipps.Tipptore1, tipps.Tipptore2 FROM Spiele "
        + "JOIN Mannschaften AS t1 ON t1.Mannschaftsid = Spiele.Team1 "
        + "JOIN Mannschaften AS t2 ON t2.Mannschaftsid = Spiele.Team2 "
        + "JOIN Tipps ON tipps.spielid = Spiele.spielid "
        + "WHERE Spiele.status_ergebnis = false "
        + "AND Spiele.status_offen = false "
        + "AND Tipps.Benutzer = '"+benutzername+"'
        + "UNION ALL "

        + "SELECT t1.Mannschaftsname, t2.Mannschaftsname, "
        + "Spiele.Anstoessdatum, Spiele.Anstoesszeit, -1 as notippl, "
        + "-1 AS notipp2 FROM Spiele "
        + "JOIN Mannschaften AS t1 ON t1.Mannschaftsid = Spiele.Team1 "
        + "JOIN Mannschaften AS t2 ON t2.Mannschaftsid = Spiele.Team2 "
        + "WHERE Spiele.Status_Ergebnis = false "
        + "AND Spiele.Status_offen = false "
        + "AND Spiele.spielid NOT IN (SELECT spielid FROM tipps WHERE Benutzer = '"+benutzername+')) "
        + "ORDER BY Anstoessdatum, Anstoesszeit"));

    while(dbVerbindung.getResultSet().next()){
        ArrayList<String> temp = new ArrayList<>();
        temp.add(dbVerbindung.getResultSet().getString(1));
        temp.add(dbVerbindung.getResultSet().getString(2));
        temp.add(dbVerbindung.getResultSet().getString("Anstoessdatum"));
        temp.add(dbVerbindung.getResultSet().getString("Anstoesszeit"));
        temp.add(dbVerbindung.getResultSet().getString(5));
        temp.add(dbVerbindung.getResultSet().getString(6));
        list.add(temp);
    }

    return list;
}

public ArrayList<ArrayList<String>> spieleAbgeschlossen (DBVerbindung dbVerbindung, String benutzername)
    throws SQLException{
    ArrayList<ArrayList<String>> list;
    list = new ArrayList<>();

    dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
        "(SELECT t1.Mannschaftsname, t2.Mannschaftsname, Spiele.ToreT1, Spiele.ToreT2, "
        + "Spiele.Anstoessdatum, Spiele.Anstoesszeit, tipps.Tipptore1, tipps.Tipptore2 FROM Spiele "
        + "JOIN Mannschaften AS t1 ON t1.Mannschaftsid = Spiele.Team1 "
        + "JOIN Mannschaften AS t2 ON t2.Mannschaftsid = Spiele.Team2 "
        + "JOIN Tipps ON tipps.spielid = Spiele.spielid "
        + "WHERE Spiele.status_ergebnis = true "
        + "AND Tipps.Benutzer = '"+benutzername+"'
        + "UNION ALL "

        + "SELECT t1.Mannschaftsname, t2.Mannschaftsname, Spiele.ToreT1, "
        + "Spiele.ToreT2, Spiele.Anstoessdatum, Spiele.Anstoesszeit, -1 as notippl, "
        + "-1 AS notipp2 FROM Spiele "
        + "JOIN Mannschaften AS t1 ON t1.Mannschaftsid = Spiele.Team1 "
        + "JOIN Mannschaften AS t2 ON t2.Mannschaftsid = Spiele.Team2 "
        + "WHERE Spiele.Status_Ergebnis = true "
        + "AND Spiele.spielid NOT IN (SELECT spielid FROM tipps WHERE Benutzer = '"+benutzername+')) "
        + "ORDER BY Anstoessdatum, Anstoesszeit"));

    while(dbVerbindung.getResultSet().next()){
        ArrayList<String> temp = new ArrayList<>();
        temp.add(dbVerbindung.getResultSet().getString(1));
        temp.add(dbVerbindung.getResultSet().getString(2));
        temp.add(dbVerbindung.getResultSet().getString("Anstoessdatum"));
        temp.add(dbVerbindung.getResultSet().getString("Anstoesszeit"));
        temp.add(dbVerbindung.getResultSet().getString("ToreT1"));
        temp.add(dbVerbindung.getResultSet().getString("ToreT2"));
        temp.add(dbVerbindung.getResultSet().getString(7));
        temp.add(dbVerbindung.getResultSet().getString(8));
        list.add(temp);
    }

    return list;
}

```

Hat der Benutzer Tipps für die Spiele abgegeben, sieht er diese hier ebenfalls. Außerdem wird dem Benutzer sein Punktestand mitgeteilt und sein Rang unter den Tipfern. Wie sie im Verlauf der Hausarbeit sehen werden, haben wir auch eine Ranglistenfunktion im Manager. Hier wird dem Manager die Rangliste der Tipper, mit Benutzernamen und Punktestand angezeigt. Darauf haben wir beim Benutzer, als reinem Teilnehmer beim Tippspiel, aus Datenschutzgründen verzichtet. So bekommt ein Benutzer zwar mitgeteilt, auf welchem Platz er steht, jedoch nicht, wie die anderen Mitspieler platziert sind.

```
public ArrayList<ArrayList<String>> rangliste (DBVerbindung dbVerbindung)
    throws SQLException{
    ArrayList<ArrayList<String>> list;
    list = new ArrayList<> ();

    dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
        "SELECT Benutzer.benutzername, Sum(tmp.test) AS Punkte FROM Benutzer "
        + "LEFT JOIN "
        + "(SELECT Benutzer, 1 AS test FROM tipps "
        + "JOIN Spiele ON Spiele.SPIELID=Tipps.SPIELID "
        + "WHERE ((spiele.TORET1-Spiele.TORET2)>0 "
        + "AND (tipps.TIPPTORE1-Tipps.TIPPTORE2)>0) "
        + "OR ((spiele.TORET1-Spiele.TORET2)=0 "
        + "AND (tipps.TIPPTORE1-Tipps.TIPPTORE2)=0) "
        + "OR ((spiele.TORET1-Spiele.TORET2)<0 "
        + "AND (tipps.TIPPTORE1-Tipps.TIPPTORE2)<0) "
        + "AND Spiele.STATUS_ERGEBNIS=true "
        + "UNION ALL "
        + "SELECT Benutzer, 2 AS test FROM Tipps "
        + "JOIN Spiele ON Spiele.spielid = Tipps.spielid "
        + "WHERE Spiele.toret1 = tipps.tipptore1 "
        + "AND Spiele.toret2 = tipps.tipptore2 "
        + "AND Spiele.Status_Ergebnis=true) AS tmp ON tmp.Benutzer = Benutzer.benutzername "
        + "GROUP BY Benutzer.benutzername "
        + "ORDER BY Punkte DESC NULLS LAST"));

    while(dbVerbindung.getResultSet().next()){
        ArrayList<String> temp = new ArrayList<>();
        temp.add(dbVerbindung.getResultSet().getString("Benutzername"));
        temp.add(dbVerbindung.getResultSet().getString("Punkte"));
        list.add(temp);
    }
    return list;
}
```

Der Punktestand des Benutzers wird jedoch nicht statisch in der Datenbank als Wert gespeichert, sondern wird, auf Anfrage hin, berechnet. Die Berechnung findet in der SQL-Abfrage selbst statt.

Auch hier führt der Zurückknopf auf das Benutzerobermenü zurück.

### 3.2.3 Das Managermenü

Meldet man sich erfolgreich als Manager an, erscheint auch hier eine Menüstruktur.

```
public void managermenu(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException, SQLException {

    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Managermenü</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/projektarbeit_V2/Managermenu\" method=\"POST\">");
        out.println("<h2> Aktivitäten auswählen</h2>");

        out.println("Mannschaften hinzufügen, bearbeiten und löschen: <br>");
        out.println("<input type=\"submit\" name=\"mannschaftsmenu\" value=\"Mannschaftsmenü\"><br><br>");

        out.println("Turnier hinzufügen, bearbeiten und löschen: <br>");
        out.println("<input type=\"submit\" name=\"turniermenu\" value=\"Turniermenü\"><br><br>");

        out.println("Spiele hinzufügen, bearbeiten, löschen, sperren und Ergebnisse eintragen: <br>");
        out.println("<input type=\"submit\" name=\"spielmenu\" value=\"Spielmenü\"><br><br>");

        out.println("Punktestände der Benutzer abfragen: <br>");
        out.println("<input type=\"submit\" name=\"punktemenu\" value=\"Punktestände abfragen\"><br><br>");

        out.println("<input type=\"submit\" name=\"abmelden\" value=\"abmelden\"><br>");

        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

## Aktivität auswählen

Mannschaften hinzufügen, bearbeiten und löschen:  
[Mannschaftsmenü](#)

Turnier hinzufügen, bearbeiten und löschen:  
[Turniermenü](#)

Spiele hinzufügen, bearbeiten, löschen, sperren und Ergebnisse eintragen:  
[Spielmenü](#)

Punktestände der Benutzer abfragen:  
[Punktestände abfragen](#)

[abmelden](#)

Alle Funktionalitäten, die dem Manager zur Verfügung gestellt wurden, sind von hier aus erreichbar. Die, hinter den Buttons befindlichen Menüstrukturen werden in den folgenden Kapiteln einzeln erklärt.

Mit Klick auf den Button „Abmelden“ wird der Manager abgemeldet, die Session wird, genau so wie beim Abmelden des Benutzers beendet und die valueUnbound – Methode aufgerufen. Die index.html wird erneut aufgerufen und ein anderer Nutzer kann sich anmelden.

### Mannschaftsmenü

Durch einen Knopfdruck auf „Mannschaftsmenü“ gelangt ein angemeldeter Manager in das Mannschaftsmenü, in dem er Mannschaften verwalten kann. Es ist möglich Mannschaften zu erstellen und bereits vorhandene Mannschaften zu bearbeiten oder zu löschen.

```

protected void mannschaftsmenue(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, SQLException {
    ArrayList<ArrayList<String>> list;
    list = new ArrayList<>();
    list=managerDB.mannschaftsListe(verbindungDB);
    String id;

    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("  <head>");
        out.println("    <title>Mannschaftsmenue</title>");
        out.println("  </head>");
        out.println("  <body>");
        out.println("    <form action=\"http://localhost:8080/Projektarbeit_V2/Mannschaftsmenue\" method = \"POST\"");
        out.println("      <h2>Mannschaften</h2>");
        for(int i=0; i<list.size();i++) {
            id = list.get(i).get(0);
            out.println("<input type=\"radio\" name=\"mannschaftsauswahl\" value=\"" + id + "\">");
            out.println(list.get(i).get(1) + "<br>");
        }
        out.println("<br><input type=\"submit\" name=\"mannschaft_loeschen\" value=\"Mannschaft löschen \"/>");
        out.println("<input type=\"submit\" name=\"mannschaft_bearbeiten\" value=\"Mannschaft bearbeiten \"/>");
        out.println("<input type=\"submit\" name=\"mannschaft_hinzufuegen\" value=\"neue Mannschaft hinzufügen \"/>");
        out.println("<input type=\"submit\" name=\"zurueck_Managermenue\" value=\"zurück Managermenue \"/>");

        out.println("</form>");
        out.println("  </body>");
        out.println("  </html>");
    }
}

```

### Mannschaften

- RB Leipzig
- FC Bayern
- Wolfsburg
- Leverkusen
- BVB
- Freiburg
- SGE
- Moenchengladbach
- Hoffenheim
- Schalke
- Union Berlin
- Fortuna Duesseldorf
- Werder Bremen
- 1.FC Koeln
- SC Paderborn
- Augsburg
- Hertha
- Mainz 05

[Mannschaft löschen](#) [Mannschaft bearbeiten](#) [neue Mannschaft hinzufügen](#) [zurück](#)

Wir geben die Mannschaften mit Radio-Buttons an, damit der Manager wählen kann, welche, bereits bestehende Mannschaft er bearbeiten oder löschen möchte. Die Liste mit den Mannschaften bekommen wir aus der Methode „mannschaftsListe“.

```

public ArrayList<ArrayList<String>> mannschaftsListe (DBVerbindung dbVerbindung)
throws SQLException{
    ArrayList<ArrayList<String>> list;
    list = new ArrayList<>();

    dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
        "SELECT Mannschaftid, Mannschaftsname FROM Mannschaften"));

    while(dbVerbindung.getResultSet().next()){
        ArrayList<String> temp = new ArrayList<>();
        temp.add(dbVerbindung.getResultSet().getString(1));
        temp.add(dbVerbindung.getResultSet().getString(2));
        list.add(temp);
    }
    return list;
}

```

Wird keine Mannschaft ausgewählt, wenn Mannschaft löschen oder Mannschaft bearbeiten angeklickt wird, erscheint die allgemeine Fehlerseite, die Sie bereits kennen.

```

else if(request.getParameter("mannschaft_bearbeiten")!= null){
    String id = request.getParameter("mannschaftsauswahl");
    if(id==null)fehlermeldung(request, response);
    mannschaftBearbeitenHTML(request, response, id);
}

else if(request.getParameter("aendern_Mannschaft")!=null){
    String temp = request.getParameter("mannschaft");
    String id = request.getParameter("mannschaft_geaendert");
    managerDB.mannschaftBearbeiten(verbindungDB, temp, id);
    mannschaftsmenueHTML( request, response);
}

```

Da eine neue Mannschaft in unserer Datenbank ausschließlich mit einer automatisch generierten ID und einem Namen gespeichert wird, fragen wir beim Erstellen einer Mannschaft nur nach den Mannschaftsnamen ab.

```

protected void mannschaftHinzufuegen(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException, SQLException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Mannschaft hinzufügen</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit_V2/Mannschaftsmenue\" method = \"POST\"");
        out.println("h2>Mannschaft hinzufügen</h2>");
        out.println("<input type=\"text\" id=\"mannschaft\" name=\"mannschaft\" ><br>");
```

**Mannschaft hinzufügen**

Wurde ein neuer Mannschaftsname eingegeben, leitet ProcessRequest zuerst auf die Datenbankenfunktion „mannschaftHinzufuegen“ weiter und dann zurück auf das Mannschaftsmenü.

```

else if(request.getParameter("speichern_Mannschaft")!=null){
    String temp = request.getParameter("mannschaft");
    managerDB.mannschaftHinzufuegen(verbindungDB, temp);
    mannschaftsmenue( request, response);
}

```

```

public void mannschaftHinzufuegen(DBVerbindung verb, String name)
    throws SQLException, ClassNotFoundException{
    verb.getStat().executeUpdate(
        "INSERT INTO Mannschaften (Mannschaftsname) "
        + "VALUES ('"+name+"')");
}

```

Die Seite zum bearbeiten der Mannschaft ähnelt der Seite zum Mannschaft hinzufügen stark, da sich hier nur die SQL-Abfrage entsprechend ändert.

```

protected void mannschaftBearbeiten(HttpServletRequest request, HttpServletResponse response, String id)
    throws ServletException, IOException, SQLException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Mannschaft bearbeiten</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit_V2/Mannschaftsmenue\" method = \"POST\"");
        out.println("<input type=\"hidden\" name=\"mannschaft_geaendert\" value=\""+id+"\"");
        out.println("h2>Mannschaft bearbeiten</h2><br>");
```

**Mannschaft bearbeiten**

Die Aktualisierung der Datenbank geschieht in der Funktion „mannschaftBearbeiten“.

```
public void mannschaftBearbeiten(DBVerbindung verb, String name, String id)
    throws SQLException, ClassNotFoundException{
    verb.getStat().executeUpdate(
        "UPDATE Mannschaften SET Mannschaftename='"+name+"'
        + "WHERE Mannschaftsid='"+id+"');");
}
```

Soll eine Mannschaft gelöscht werden, muss erst überprüft werden, ob diese Mannschaft bereits in einem Spiel verwendet wird. Da die MannschaftsID ein Fremdschlüssel in der Tabelle Spiele ist.

```
else if(request.getParameter("mannschaft_loeschen")!= null){
    String id = request.getParameter("mannschaftsauswahl");
    if(id==null)fehlermeldung(request, response);
    if(managerDB.mannschaftLoeschenPruefen(verbindungDB, id)){
        managerDB.mannschaftLoeschen(verbindungDB, id);
        mannschaftsmenu( request, response);
    }
    else{
        fehlermeldungLoeschen(request, response);
    }
}
```

Dies geschieht mit Hilfe der Funktion „mannschaftLoeschenPruefen“

```
public boolean mannschaftLoeschenPruefen (DBVerbindung verb, String id)
    throws SQLException{
    verb.setRes(verb.getStat().executeQuery(
        "SELECT Spielid FROM Spiele "
        + "WHERE Team1 = "+id+
        + "OR Team2 = "+id+"));
    return !verb.getRes().next();
}
```

Diese Methode gibt true zurück, wenn die Mannschaft in keinem Spiel verwendet wird, die Mannschaft also gelöscht werden kann.

```
public void mannschaftLoeschen(DBVerbindung verb, String id)
    throws SQLException, ClassNotFoundException{
    verb.getStat().executeUpdate(
        "DELETE FROM Mannschaften "
        + "WHERE Mannschaftsid='"+id+"');");
}
```

Bekommen wir jedoch ein False zurück, gibt es also Spiele mit der zu löschen Mannschaft, erscheint die spezielle Fehlerseite „fehlermeldungLoeschen“.

```
protected void fehlermeldungLoeschen(HttpServletRequest request, HttpServletResponse response) throws IOException{
    response.setContentType("text/html;charset=UTF-8");
    try(PrintWriter out = response.getWriter()){
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Fehler beim Löschen der Mannschaft</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit_V2/Mannschaftsmenu\" method = \"POST\">");
        out.println("<h2>Fehler</h2><BR>");
        out.println("Die Mannschaft ist bereits in einem oder mehreren Spielen gespeichert<BR>");
        out.println("wenn Sie die Mannschaft trotzdem löschen wollen, bitte erst die Spiele löschen.<BR>"); Fehler
        out.println("<input type=\"submit\" name=\"zurueck\" value=\"zurueck\" type='button'><BR><BR><BR>");  

        out.println("</body>");
        out.println("</html>");  

    }
}
```

Die Mannschaft ist bereits in einem oder mehreren Spielen gespeichert  
wenn Sie die Mannschaft trotzdem löschen wollen, bitte erst die Spiele löschen.  
**zurück**

Treten andere Fehler auf, wie beispielsweise keine Mannschaft ausgewählt, die bearbeitet oder gelöscht werden soll, erscheint die, bereits bekannte, allgemeine Fehlerseite. Aus der

Fehlerseite heraus gelangt man über den „zurück“-Knopf zurück in das Mannschaftsmenü.  
Durch Knopfdruck auf den „zurück“-Knopf im Mannschaftsmenü gelangt man zurück in das Managermenü.

## Turniermenü

Wählt man im Managermenü die Option „Turniermenü“ gelangt man in das Turniermenü.

```
protected void turniermenu(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, SQLException {
    //Liste mit Turnieren, fuer die Ausgabe, Reihenfolge: TurnierID-Turniernamen
    ArrayList<ArrayList<String>> list;
    list = new ArrayList<>();
    list= managerDB.turnierListe(verbindungDB);
    String id;

    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Turniermenü</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit_V2/Turniermenu\" method = \"POST\">");

        out.println("<h2>Turniere</h2>");
        for(int i=0; i<list.size();i++) {
            id = list.get(i).get(0);
            out.println("<input type=\"radio\" name=\"turnierauswahl\" value=\"" + id + "\">");
            out.println(list.get(i).get(1) + "<br>");
        }

        out.println("<br><input type=\"submit\" name=\"turnier_loeschen\" value=\"Turnier löschen \"/>");
        out.println("<input type=\"submit\" name=\"turnier_bearbeiten\" value=\"Turnier bearbeiten \"/>");
        out.println("<input type=\"submit\" name=\"turnier_hinzufuegen\" value=\"neues Turnier hinzufügen \"/>");
        out.println("<input type=\"submit\" name=\"zurueck_managermenue\" value=\"zurückManagermenü\" /><br><br>");

        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
```

### Turniere

- 1. Bundesliga
- 2. Bundesliga
- 3. Bundesliga
- DFB Pokal

[Turnier löschen](#) [Turnier bearbeiten](#) [neues Turnier hinzufügen](#) [zurück](#)

Das Turniermenü in seiner Art und den Funktionen ist dem Mannschaftsmenü sehr ähnlich.

Da sich Turniere und Mannschaften auch in der Art, wie sie in der Datenbank gespeichert werden gleichen, jeweils mit einer ID und einem Namen.

```

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException, SQLException, ClassNotFoundException {

    HttpSession session = request.getSession(false);
    if( session == null ) fehlermeldungSession(request, response);

    if(request.getParameter("turnier_hinzufuegen")!= null){
        turnierHinzufuegen(request, response);
    }
    else if(request.getParameter("speichern_Turnier")!=null){
        String temp = request.getParameter("turnier");
        managerDB.turnierHinzufuegen(verbindungDB, temp);
        turniermenu( request, response);
    }
    else if(request.getParameter("turnier_bearbeiten")!= null){
        String id = request.getParameter("turnierauswahl");
        if(id==null) fehlermeldung(request, response);
        turnierBearbeiten(request, response, id);
    }
    else if(request.getParameter("aendern_Turnier")!=null){
        String temp = request.getParameter("turnier");
        String id = request.getParameter("turnier_geaendert");
        managerDB.turnierBearbeiten(verbindungDB, temp, id);
        turniermenu( request, response);
    }
    else if(request.getParameter("turnier_loeschen")!= null){
        String id = request.getParameter("turnierauswahl");
        if(id==null) fehlermeldung(request, response);
        if(managerDB.turnierLoeschenPruefen(verbindungDB, id)){
            managerDB.turnierLoeschen(verbindungDB, id);
            turniermenu( request, response);
        }
        else{
            fehlermeldungLoeschen(request, response);
        }
    }
    else if(request.getParameter("zurueck") != null){
        turniermenu(request, response);
    }
    else if(request.getParameter("zurueck_managermenu") != null){
        Managermenu manager = new Managermenu();
        manager.managermenu(request, response);
    }
}
}

```

Dies ist sowohl in den Funktionen, als auch im ProcessRequest sehr deutlich, denn alle Stellen, an denen Fehler abgefangen werden sind gleich.

```

/** Methode zum Erhalten einer Liste aller Turniere, mit ID und Name ...7 lines */
public ArrayList<String>>turnierListe (DBVerbindung verb)
    throws SQLException{
    ArrayList<ArrayList<String>> list;
    list = new ArrayList<>();

    verb.setRes(verb.getStat().executeQuery(
        "SELECT TurnierID, Turniername FROM Turniere"));

    while(verb.getRes().next()){
        ArrayList<String> temp = new ArrayList<>();
        temp.add(verb.getRes().getString(1));
        temp.add(verb.getRes().getString(2));
        list.add(temp);
    }
    return list;
}

/** Methode zu speichern neuer Turniere in der Tabelle Turniere ...8 lines */
public void turnierHinzufuegen(DBVerbindung verb, String name)
    throws SQLException, ClassNotFoundException{
    verb.getStat().executeUpdate(
        "INSERT INTO Turniere (Turniername) "
        + "VALUES ('"+name+"')");
}

/** Methode zum speichern der Aenderungen an einem Turnier ...9 lines */
public void turnierBearbeiten(DBVerbindung verb, String name, String id)
    throws SQLException, ClassNotFoundException{
    verb.getStat().executeUpdate(
        "UPDATE Turniere SET Turniername='"+name+"' "
        + "WHERE Turnierid='"+id+"'");
}

```

```
/** Methode zum Prüfen, ob ein Turnier bereits in einem oder mehreren Spielen ...10 lines */
public boolean turnierLoeschenPruefen(DBVerbindung dbVerbindung, String id)
throws SQLException{
    dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
        "SELECT Spielid FROM Spiele "
        + "WHERE turnier = "+id+""));

    return !dbVerbindung.getResultSet().next();
}

/** Methode zum Löschen eines Turniers ...8 lines */
public void turnierLoeschen(DBVerbindung dbVerbindung, String id)
throws SQLException, ClassNotFoundException{
    dbVerbindung.getStatement().executeUpdate(
        "DELETE FROM Turniere WHERE Turnierid="+id+"");
}
```

Die Datenbankfunktionen sind ebenfalls gleich, nur dass sie die Relation Turniere bearbeiten, nicht die Relation Mannschaften.

```
protected void turnierHinzufuegen(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, SQLException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Turnier hinzufügen</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit_V2/Turniermenue\" method = \"POST\"");
        out.println("<h2>Turnier hinzufügen</h2><BR>");
        out.println("<input type=\"text\" id=\"turnier\" name=\"turnier\" ><br>");
        out.println("<input type=\"submit\" name=\"speichern_Turnier\" value=\"speichern\"><BR><BR>");
        out.println("<input type=\"submit\" name=\"zurueck\" value=\"zurück\"><br><br>");
        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}

protected void turnierBearbeiten(HttpServletRequest request, HttpServletResponse response, String id)
throws ServletException, IOException, SQLException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Turnier bearbeiten</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit_V2/Turniermenue\" method = \"POST\"");
        out.println("<input type=\"hidden\" name=\"turnier_geaendert\" value=\""+id+"\">");
        out.println("<h2>Turnier bearbeiten</h2><BR>");
        out.println("<input type=\"text\" id=\"turnier\" name=\"turnier\" ><br>");
        out.println("<input type=\"submit\" name=\"aendern_Turnier\" value=\"&uuml;ndern\"><BR><BR>");
        out.println("<input type=\"submit\" name=\"zurueck\" value=\"zurück\"><br><br>");
        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

## Spielmenü

Im Spielmenü kann der Manager Spiele verwalten.

```

protected void spieldemenu(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException, SQLException, ParseException {
    ArrayList<ArrayList<String>> list;
    list = new ArrayList<>();
    String ID;

    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Spielmenue</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit\_V2/Spielmenue\" method=\"POST\">");

        out.println("<h2>offene Spiele:</h2>");
        list = managerDB.listeSpieleOffen(verbindingDB);
        Pruefmethododen pruefmethododen = new Pruefmethododen();
        for(int i=0; i<list.size(); i++){
            if(pruefmethododen.anstosszeitPruefen(list.get(i).get(2), list.get(i).get(3))) {
                ID = list.get(i).get(5);
                out.println("<input type=\"checkbox\" name=\"sperren\" value=\""+ID+"\">");
                out.println(list.get(i).get(0));
                out.println(" " + list.get(i).get(1));
                out.println(" " + list.get(i).get(2));
                out.println(" " + list.get(i).get(3));
                out.println(" Turnier: " + list.get(i).get(4) + "<br>");
            }
            else{
                managerDB.spielSperren(verbindingDB, list.get(i).get(5));
            }
        }
        out.println("<input type=\"submit\" name=\"spiel_sperren\" value=\"Spiel sperren\">");
        out.println("<input type=\"submit\" name=\"spiel_bearbeiten\" value=\"Spiel bearbeiten\">"); 
        out.println("<input type=\"submit\" name=\"spiel_loeschen\" value=\"Spiele löschen\"><br><br>");

        out.println("<h2>gesperrte Spiele:</h2>");
        list = managerDB.listeSpieleGesperrt(verbindingDB);
        for(int i=0; i<list.size(); i++){
            ID = list.get(i).get(5);
            out.println("<input type=\"checkbox\" name=\"ergebnis\" value=\""+ID+"\">");
            out.println(list.get(i).get(0));
            out.println(" " + list.get(i).get(1));
            out.println(" " + list.get(i).get(2));
            out.println(" " + list.get(i).get(3));
            out.println(" Turnier: " + list.get(i).get(4));
            out.println("<input type=\"number\" id=\"erg1_"+ID+"\" name=\"erg1_"+ID+"\" min=\"0\" step=\"1\" value=\"0\">"); 
            out.println("<input type=\"number\" id=\"erg2_"+ID+"\" name=\"erg2_"+ID+"\" min=\"0\" step=\"1\" value=\"0\" ><br>"); 

            out.println("<input type=\"submit\" name=\"ergebnis_speichern\" value=\"Ergebnisse speichern\">"); 
            out.println("<input type=\"submit\" name=\"spiel_entsperrnen\" value=\"Spiele entsperren\"><br><br>"); 
            out.println("<input type=\"submit\" name=\"neues_Spiel\" value=\"neues Spiel\">"); 
            out.println("<input type=\"submit\" name=\"abgeschlossene_Spiele\" value=\"abgeschlossene Spiele\">"); 
            out.println("<input type=\"submit\" name=\"zurueck_managermenue\" value=\"zurueck/ok\">");

            out.println("</form>"); 
            out.println("</body>"); 
            out.println("</html>"); 
        }
    }
}

```

### offene Spiele:

- Mainz 05 : Wolfsburg 2019-09-30 18:00:00 Turnier: 1. Bundesliga
- Augsburg : Leverkusen 2019-09-30 18:00:00 Turnier: 1. Bundesliga
- SC Paderborn : FC Bayern 2019-09-30 18:00:00 Turnier: 1. Bundesliga
- 1.FC Koeln : Hertha 2019-09-30 18:00:00 Turnier: 1. Bundesliga
- Fortuna Duesseldorf : Freiburg 2019-09-30 18:00:00 Turnier: 1. Bundesliga
- Union Berlin : SGE 2019-09-30 18:00:00 Turnier: 1. Bundesliga
- Hoffenheim : Moenchengladbach 2019-09-30 18:00:00 Turnier: 1. Bundesliga
- RB Leipzig : Schalke 2019-09-30 18:00:00 Turnier: 1. Bundesliga

[Spiele sperren](#) [Spiel bearbeiten](#) [Spiele löschen](#)

### gesperrte Spiele:

<input type="checkbox"/> Werder Bremen : Augsburg 2019-08-30 18:00:00 Turnier: 1. Bundesliga	<input type="button" value="0"/>	<input type="button" value="0"/>
<input type="checkbox"/> Union Berlin : BVB 2019-08-30 18:00:00 Turnier: 1. Bundesliga	<input type="button" value="0"/>	<input type="button" value="0"/>
<input type="checkbox"/> Schalke : Hertha 2019-08-30 18:00:00 Turnier: 1. Bundesliga	<input type="button" value="0"/>	<input type="button" value="0"/>
<input type="checkbox"/> Freiburg : 1.FC Koeln 2019-08-30 18:00:00 Turnier: 1. Bundesliga	<input type="button" value="0"/>	<input type="button" value="0"/>
<input type="checkbox"/> Wolfsburg : SC Paderborn 2019-08-30 18:00:00 Turnier: 1. Bundesliga	<input type="button" value="0"/>	<input type="button" value="0"/>
<input type="checkbox"/> FC Bayern : Mainz 05 2019-08-30 18:00:00 Turnier: 1. Bundesliga	<input type="button" value="0"/>	<input type="button" value="0"/>
<input type="checkbox"/> RB Leipzig : Moenchengladbach 2019-08-30 18:00:00 Turnier: 1. Bundesliga	<input type="button" value="0"/>	<input type="button" value="0"/>
<input type="checkbox"/> Leverkusen : Hoffenheim 2019-08-31 18:00:00 Turnier: 1. Bundesliga	<input type="button" value="0"/>	<input type="button" value="0"/>
<input type="checkbox"/> FC Bayern : Mainz 05 2019-08-31 18:00:00 Turnier: 1. Bundesliga	<input type="button" value="0"/>	<input type="button" value="0"/>
<input type="checkbox"/> BVB : Werder Bremen 2019-09-30 18:00:00 Turnier: 1. Bundesliga	<input type="button" value="0"/>	<input type="button" value="0"/>

[Ergebnisse speichern](#) [Spiele entsperren](#)

[neues Spiel](#) [abgeschlossene Spiele](#) [zurück](#)

Beim aufrufen und darstellen der Seite wird in der Methode „anstosszeitPruefen“, die wir bereits erläutert haben, geprüft, ob die Anstosszeit und das Anstossdatum eines offenen Spiels nach der Jetztzeit liegt. Ist das der Fall, wird das Spiel automatisch gesperrt und der Liste „gesperrte Spiele“ zugefügt. Versucht der Manager ein Spiel, dessen Anstossdatum und die Anstosszeit abgelaufen sind, zu entsperren, wird es automatisch beim neuen Aufbau der Seite wieder gesperrt. Spiele können einzeln, oder zu mehreren gesperrt oder entsperrt werden. Sollte ein Spiel versehentlich gesperrt worden sein, kann es wieder entsperrt werden.

```
else if(request.getParameter("spiel_sperren") != null){  
    String[] temp = request.getParameterValues("sperren");  
    if(temp==null) fehlermeldung(request, response);  
    for(int i=0; i<temp.length; i++){  
        managerDB.spielSperren(verbindungDB, temp[i]);  
    }  
    spielemenu(request, response);  
}  
  
else if(request.getParameter("spiel_entsperrnen") != null){  
    String[] id = request.getParameterValues("ergebnis");  
    if(id==null) fehlermeldung(request, response);  
    for(int i=0; i<id.length; i++){  
        managerDB.spielEntsperrnen(verbindungDB, id[i]);  
    }  
    spielemenu(request, response);  
}  
  
public void spelletsperren (DBVerbindung dbVerbindung, String id)  
throws SQLException{  
    dbVerbindung.getStatement().executeUpdate(  
        "UPDATE Spiele SET Status_Offen=false "  
        + "WHERE SpielID='"+id+"'");  
}  
  
public void spelletsperren (DBVerbindung dbVerbindung, String id)  
throws SQLException{  
    dbVerbindung.getStatement().executeUpdate(  
        "UPDATE Spiele SET Status_Offen=true "  
        + "WHERE SpielID='"+id+"'");  
}
```

Es können theoretisch alle Spiele, unabhängig ihrer Startzeit gesperrt werden, entscheidet sich also ein Manager ein Spiel für Tipps zu sperren, Stunden bevor der Anstoß erfolgt, so ist das erlaubt.

Beim bearbeiten eines Spiels ist es nicht zulässig mehrere Spiele auszuwählen, dies fangen wir im ProcessRequest ab.

```
else if(request.getParameter("spiel_bearbeiten") != null){  
    String id[] = request.getParameterValues("sperren");  
    if(id==null || id.length!=1) fehlermeldung(request, response);  
    spielBearbeiten(request, response, id[0]);  
}
```

```

protected void spielBearbeiten(HttpServletRequest request, HttpServletResponse response, String id)
    throws ServletException, IOException, SQLException {
    ArrayList<ArrayList<String>> list_Mannschaften = managerDB.mannschaftsListe(verbindungDB);
    ArrayList<ArrayList<String>> list_Turnier = managerDB.turnierListe(verbindungDB);

    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {

        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Spiel bearbeiten</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit_V2/Spielmenue\" method = \"POST\">");

        out.println("<input type=\"hidden\" name=\"spiel_geaendert\" value=\"" + id + "\">");
        out.println("<h2>Spiel bearbeiten</h2>");

        out.println("<form action = \"select.html\">");
        out.println("Mannschaften:");
        out.println("<select name=\"Mannschaften_DropDown\" size=\"1\">");
        for(int i=0; i<list_Mannschaften.size();i++){
            out.println("<option value=\"" + list_Mannschaften.get(i).get(0) + ">" + list_Mannschaften.get(i).get(1) + "</option>");
        }
        out.println("</select>");

        out.println("<form action = \"select.html\">");
        out.println("<select name=\"Turnier_DropDown2\" size=\"1\">");
        for(int i=0; i<list_Turnier.size();i++){
            out.println("<option value=\"" + list_Turnier.get(i).get(0) + ">" + list_Turnier.get(i).get(1) + "</option>");
        }
        out.println("</select><BR><BR>");

        out.println("Geben Sie die Anstoßzeit an:");

        out.println("<input type=\"date\" name=\"anstoss\"");
        out.println("<input type=\"time\" name=\"anstoss_uhrzeit\"><BR>");  

        out.println("<input type=\"submit\" name=\"aendern_Spiel\" value=\"&umld;ndern\">");  

        out.println("<input type=\"submit\" name=\"zurueck\" value=\"zur&uuml;ck\">");

        out.println("</form>");
        out.println("</body>");
        out.println("</html>");

    }
}

```

## Spiel bearbeiten

Mannschaften:	<input type="button" value="RB Leipzig"/>	<input type="button" value="RB Leipzig"/>
Turnier	<input type="button" value="1. Bundesliga"/>	
Geben Sie die Anstoßzeit an:	<input type="text" value="TT . MM . JJJJ"/>	<input type="text" value="-- : --"/>
	<input type="button" value="&amp;umld;ndern"/>	<input type="button" value="zurueck"/>

Beim bearbeiten eines Spiels können alle Aspekte bearbeitet werden, die teilnehmenden Mannschaften, das Turnier in dem gespielt wird, die Anstoßzeit und das Anstoßdatum.

Hat der Manager die Eingaben abgeschlossen, werden die neuen Daten im ProcessRequest ausgelesen und der Funktion „spielBearbeiten“ übergeben, die das gespeicherte Spiel aktualisiert.

```

else if(request.getParameter("senden_Spiel")!= null){
    String id=request.getParameter("spiel_geaendert");
    String team1 = request.getParameter("Mannschaften_DropDown1");
    String team2 = request.getParameter("Mannschaften_DropDown2");
    String turnier = request.getParameter("Turnier_DropDown");
    String anstossDatum = request.getParameter("anstoss");
    String anstossZeit = request.getParameter("anstoss_uhrzeit");
    Pruefmethoden pruefmethoden = new Pruefmethoden();

    if(pruefmethoden.spielSpeichernPruefen(team1, team2, anstossDatum, anstossZeit)){
        managerDB.spielbearbeiten(verbindungDB, team1, team2, turnier, anstossDatum, anstossZeit, id);
        spielemenu(request, response);
    }
    else{
        fehlermeldung(request, response);
    }
}

public boolean spielSpeichernPruefen(String team1, String team2, String anstossDatum, String anstossZeit)
throws IOException, ParseException{
if(team1==null || team2==null || anstossDatum==null || anstossZeit==null){
    return false;
}

SimpleDateFormat datumFormat = new SimpleDateFormat("yyyy-MM-dd");
String stringHeute = datumFormat.format(new Date());

Date heute = datumFormat.parse(stringHeute);
Date parseAnstossDatum = datumFormat.parse(anstossDatum);

SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm");
String stringZeit = timeFormat.format(new Date());

Date jetzt = timeFormat.parse(stringZeit);
Date parseAnstossZeit = timeFormat.parse(anstossZeit);

if(team1.equals(team2)){
    return false;
}
if(parseAnstossDatum.before(heute)){
    return false;
}
else if(parseAnstossDatum.equals(heute)
    && parseAnstossZeit.before(jetzt)){
    return false;
}
return true;
}

public void spelBearbeiten(DBVerbindung dbVerbindung, String team1, String team2,
String turnier, String anstossDatum, String anstossZeit, String id)
throws SQLException{
dbVerbindung.getStatement().executeUpdate(
"UPDATE Spiele SET Team1='"+team1+"', Team2='"+team2+"',
+ " Anstoßdatum='"+anstossDatum+"', Anstosszeit='"+anstossZeit+"'
+ " WHERE Spielid='"+id+"');
}
}

```

Als Fehler werden abgefangen, wenn ein Spiel mit denselben Mannschaften gespeichert werden soll. Soll zum Beispiel der BVB gegen den BVB spielen wird die allgemeine Fehlerseite des Spielmenüs aufgerufen. Dasselbe gilt für den Fall, dass eine Startzeit vor der Jetztzeit ausgewählt wird.

Sobald ein Spiel gesperrt wurde, kann dafür das Ergebnis gespeichert werden.

```

else if(request.getParameter("ergebnis_speichern")!=null){
    String erg1;
    String erg2;
    String[] temp = request.getParameterValues("ergebnis");

    if(temp==null) fehlermeldung(request, response);
    for(int i=0; i<temp.length;i++){
        erg1 = request.getParameter("erg1_"+temp[i]);
        erg2 = request.getParameter("erg2_"+temp[i]);
        if(erg1.isEmpty() || erg2.isEmpty()) fehlermeldung(request, response);
        managerDB.ergebnisSpeichern(verbindungDB, temp[i], erg1, erg2);
    }
    spielemenu(request, response);
}

```

Hier wird das eingegebene Ergebnis ausgelesen, in der Funktion „ergebnisSpeichern“ wird dem Spiel dann ein Ergebnis zugewiesen.

```
public void ergebnisSpeichern(DBVerbindung dbVerbindung, String id, String erg1, String erg2)
    throws SQLException{
    dbVerbindung.getStatement().executeUpdate(
        "UPDATE Spiele SET ToreT1='"+erg1+"', "
        + "ToreT2='"+erg2+"', "
        + "Status_Ergebnis = true "
        + "WHERE SpielID='"+id+"');
}
```

Beim erstellen eines neuen Spiels ähnelt die HTML-Seite sehr der, wenn ein Spiel bearbeitet werden soll.

```
protected void spielHinzufuegen(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException, SQLException {

ArrayList<ArrayList<String>> list_Mannschaften = managerDB.mannschaftsListe(verbindungDB);
ArrayList<ArrayList<String>> list_Turnier = managerDB.turnierListe(verbindungDB);

response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Spiel hinzufügen</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<form action=\"http://localhost:8080/Projektarbeit_V2/Spielmenue\" method = \"POST\">");

    out.println("<h2>Spiel hinzufügen</h2>");

    out.println("<form action = \"select.html\">");
    out.println("Mannschaften:");
    out.println("<select name=\"Mannschaften_DropDown1\" size=\"1\">");
    for(int i=0; i<list_Mannschaften.size();i++){
        out.println("<option value=\""+list_Mannschaften.get(i).get(0)+">" +list_Mannschaften.get(i).get(1)+"</option>");
    }
    out.println("</select>");

    out.println("<form action = \"select.html\">");
    out.println("<select name=\"Mannschaften_DropDown2\" size=\"1\">");
    for(int i=0; i<list_Mannschaften.size();i++){
        out.println("<option value=\""+list_Mannschaften.get(i).get(0)+">" +list_Mannschaften.get(i).get(1)+"</option>");
    }
    out.println("</select><BR><BR>");

    out.println("Turnier:");

    out.println("<form action = \"select.html\">");
    out.println("<select name=\"Turnier_DropDown\" size=\"1\">");
    for(int i=0; i<list_Turnier.size();i++){
        out.println("<option value=\""+list_Turnier.get(i).get(0)+">" +list_Turnier.get(i).get(1)+"</option>");
    }
    out.println("</select><BR><BR>");

    out.println("Geben Sie die Anstoßzeit an:");

    out.println("<input type=\"date\" name=\"anstoess\">");
    out.println("<input type=\"time\" name=\"anstoess_uhrzeit\"><BR>");;
    out.println("<input type=\"submit\" name=\"speichern_Spiel\" value=\"speichern\">");;
    out.println("<input type=\"submit\" name=\"zurueck\" value=\"zurueck\">");

    out.println("</form>");

    out.println("</Body>");

    out.println("</html>");

}

if(request.getParameter("neues_Spiel")!=null){
    spielHinzufuegen(request, response);
}
else if(request.getParameter("speichern_Spiel")!=null){
    String team1 = request.getParameter("Mannschaften_DropDown1");
    String team2 = request.getParameter("Mannschaften_DropDown2");
    String turnier = request.getParameter("Turnier_DropDown");
    String anstoessDatum = request.getParameter("anstoess");
    String anstoessZeit = request.getParameter("anstoess_uhrzeit");
    Pruefmethoden pruefmethoden = new Pruefmethoden();

    if(pruefmethoden.spielSpeicherntruefen(team1, team2, anstoessDatum, anstoessZeit)){
        managerDB.spielHinzufuegen(verbindungDB, team1, team2, turnier, anstoessDatum, anstoessZeit);
        spilemenue(request, response);
    }
    else{
        fehlermeldung(request, response);
    }
}

public void spielHinzufuegen(DBVerbindung dbVerbindung, String team1, String team2,
                           String turnier, String anstoessDatum, String anstoessZeit)
    throws SQLException{

dbVerbindung.getStatement().executeUpdate(
    "INSERT INTO Spiele (Team1, Team2, ToreT1, ToreT2, Anstoessdatum, "
    + "Anstoesszeit, Turnier, Status_offen, Status_ergebnis)"
    + "Values ("+team1+","+team2+",0,0,'"+anstoessDatum+"',"
    + "'"+anstoessZeit+"','"+turnier "','TRUE','FALSE')");
}
```

Bevor ein neues Spiel gespeichert wird, wird es mit der Methode „SpielSpeichernPruefen“ überprüft, ob alle Speicherrelevanten Kriterien erfüllt sind. Hier gelten dieselben Speicherkriterien, wie bei Spiel bearbeiten. Mannschaften können, auch in derselben Liga, mehrfach gegeneinander spielen.

Mit einem Klick auf „abgeschlossene Spiele“ gelangt man auf eine HTML-Seite, auf der der Manager alle Spiele einsehen kann, für die bereits ein Ergebnis gespeichert wurde.

```

protected void abgeschlosseneSpiele(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException, SQLException {
    ArrayList<ArrayList<String>> list;
    list = new ArrayList<>();

    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>abgeschlossene Spiele</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/projektarbeit_v2/Spielmenue\" method=\"POST\">");
        out.println("<h1>abgeschlossene Spiele:</h1>");

        out.println("<table>");
        out.println("<tr>");
        out.println("<th>Team1</th>");
        out.println("<th>Team2</th>");
        out.println("<th>Tore1</th>");
        out.println("<th>Tore2</th>");
        out.println("<th>Datum</th>");
        out.println("<th>Uhrzeit</th>");
        out.println("</tr>");

        list = managerDB.listeSpieleAbgeschlossen(verbindungDB);
        for(int i=0; i<list.size();i++){
            out.println("<tr>");
            out.println("<td>" + list.get(i).get(0) + "</td>");
            out.println("<td>" + list.get(i).get(1) + "</td>");
            out.println("<td>" + list.get(i).get(2) + "</td>");
            out.println("<td>" + list.get(i).get(3) + "</td>");
            out.println("<td>" + list.get(i).get(4) + "</td>");
            out.println("<td>" + list.get(i).get(5) + "</td>");
            out.println("</tr>");
        }
        out.println("</table><br>");

        out.println("<input type=\"submit\" name=\"zurueck\" value=\"zurueck\">");

        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}
}

```

## abgeschlossene Spiele:

Team1	Team2	Tore1	Tore2	Datum	Uhrzeit
FC Bayern	Hertha	2	2	2019-08-16	18:00:00
FC Bayern	Hertha	2	2	2019-08-16	18:00:00
Werder Bremen	Fortuna Duesseldorf	1	3	2019-08-17	18:00:00
Moenchengladbach	Schalke	0	0	2019-08-17	18:00:00
Freiburg	Mainz 05	3	0	2019-08-17	18:00:00
BVB	Augsburg	5	1	2019-08-17	18:00:00
Leverkusen	SC Paderborn	3	2	2019-08-17	18:00:00
Wolfsburg	1.FC Koeln	2	1	2019-08-17	18:00:00
Union Berlin	RB Leipzig	0	4	2019-08-18	18:00:00
SGE	Hoffenheim	1	0	2019-08-18	18:00:00
1.FC Koeln	BVB	1	3	2019-08-23	18:00:00
Mainz 05	Moenchengladbach	1	3	2019-08-24	18:00:00
SC Paderborn	Freiburg	1	3	2019-08-24	18:00:00
Hoffenheim	Werder Bremen	3	2	2019-08-24	18:00:00

[zurück](#)

Die Liste mit abgeschlossenen Spielen erhalten wir von der Funktion „listSpieleAbgeschlossen“.

```
public ArrayList<ArrayList<String>> listSpieleAbgeschlossen (DBVerbindung dbVerbindung)
    throws SQLException{
    ArrayList<ArrayList<String>> list;
    list = new ArrayList<>();

    dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
        "SELECT t1.mannschaftsname, t2.mannschaftsname, Spiele.ToreT1, "
        + "Spiele.ToreT2, Spiele.Anstoessdatum, Spiele.Anstoesszeit FROM Spiele "
        + "JOIN Mannschaften AS t1 ON t1.MANNSCHAFTSID =Spiele.Team1 "
        + "JOIN Mannschaften AS t2 ON t2.MANNSCHAFTSID=Spiele.Team2 "
        + "JOIN Turniere ON Turniere.Turnierid=Spiele.Turnier "
        + "Where Spiele.Status_Ergebnis=true "
        + "AND Spiele.Status_Offen=false "
        + "ORDER BY Spiele.Anstoessdatum, Spiele.Anstoesszeit"));

    while(dbVerbindung.getResultSet().next()){
        ArrayList<String> temp = new ArrayList<>();
        temp.add(dbVerbindung.getResultSet().getString(1));
        temp.add(dbVerbindung.getResultSet().getString(2));
        temp.add(dbVerbindung.getResultSet().getString("ToreT1"));
        temp.add(dbVerbindung.getResultSet().getString("ToreT2"));
        temp.add(dbVerbindung.getResultSet().getString("Anstoessdatum"));
        temp.add(dbVerbindung.getResultSet().getString("Anstoesszeit"));
        list.add(temp);
    }
    return list;
}
```

Der Button „zurück“ auf der Seite Abgeschlossene Spiele führt auf das Spielmenü zurück. Bei einem Klick auf den Button „zurück“ im Spielmenü gelangt man in das Managermenü zurück.

### Punktestände abfragen

Die Funktionalität hinter dem Button „Punktestände abfragen“ ist, dass der Manager eine Rangliste der Benutzer, sortiert nach Punktestand ausgegeben bekommt.

```
protected void punkteliste (HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException, SQLException{
    ArrayList<ArrayList<String>> list;
    list = new ArrayList<>();

    response.setContentType("text/html;charset=UTF-8");
    try(PrintWriter out = response.getWriter()){
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Rangliste</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form action=\"http://localhost:8080/Projektarbeit\_V2/Managermenue\" method = \"POST\">");
        out.println("<h2>Rangliste der Tipper: </h2>");

        list = managerDB.rangliste(verbindungDB);
        out.println("<ol>");
        for(int i=0; i<list.size(); i++){
            if(list.get(i).get(1)==null){
                out.println("<li>" + list.get(i).get(0) + ": 0</li>");
            }
            else{
                out.println("<li>" + list.get(i).get(0) + ": " + list.get(i).get(1) + "</li>");
            }
        }
        out.println("</ol>");
        out.println("<input type=\"submit\" name=\"zurueck\" value=\"zurueck;ck\"><BR><BR>");

        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

### Rangliste der Tipper:

1. Benutzer1: 12
2. Benutzer2: 6
3. Benutzer3: 1

[zurück](#)

Hier wird, genau wie in der Ranglistenabfrage beim Benutzer die Punkteberechnung in der Datenbankenabfrage durchgeführt.

```
public ArrayList<ArrayList<String>> rangliste (DBVerbindung dbVerbindung)
throws SQLException{
ArrayList<ArrayList<String>> list;
list = new ArrayList<> ();

dbVerbindung.setResultSet(dbVerbindung.getStatement().executeQuery(
    "SELECT Benutzer.benutzername, Sum(tmp.test) AS Punkte FROM Benutzer "
    + "LEFT JOIN "
    + "(SELECT Benutzer, 1 AS test FROM tipps "
    + "JOIN Spiele ON Spiele.SPIELID=Tipps.spielid "
    + "WHERE ((spiele.TORET1-Spiele.TORET2)>0 "
    + "+AND (tipps.TIPPTORE1-Tipps.TIPPTORE2)>0) "
    + "+OR ((spiele.TORET1-Spiele.TORET2)=0 "
    + "+AND (tipps.TIPPTORE1-Tipps.TIPPTORE2)=0) "
    + "+OR ((spiele.TORET1-Spiele.TORET2)<0 "
    + "+AND (tipps.TIPPTORE1-Tipps.TIPPTORE2)<0) "
    + "+AND Spiele.STATUS_ERGEBNIS=true "
    + "UNION ALL "
    + "SELECT Benutzer, 2 AS test FROM Tipps "
    + "JOIN Spiele ON Spiele.SPIELID=Tipps.spielid "
    + "WHERE Spiele.TORET1=tipps.tipptore1 "
    + "AND Spiele.toret2 = tipps.tipptore2 "
    + "AND Spiele.Status_Ergebnis=true) AS tmp ON tmp.Benutzer = Benutzer.benutzername "
    + "GROUP BY Benutzer.benutzername "
    + "ORDER BY Punkte DESC NULLS LAST"));

while(dbVerbindung.getResultSet().next()){
    ArrayList<String> temp = new ArrayList<>();
    temp.add(dbVerbindung.getResultSet().getString("Benutzername"));
    temp.add(dbVerbindung.getResultSet().getString("Punkte"));
    list.add(temp);
}
return list;
}
```