Technische Dokumentation

Tennismanager

Mona Bieschke, Tanja Kollarczik, Dominik Tappe, Dominik Weber

Inhaltsverzeichnis

1. Einführung

2. Spielablauf, Regeln und Mechaniken

- 2.1 Idee des Spiels
- 2.2 Übernahme von Daten der ATP
- 2.3 Spielbalance

3. Installationsanleitung

- 3.1 Voraussetzungen
- 3.2 Installation

4. Datenbank

- 4.1 DBMS
- 4.2 Datenbankentwurf
- 4.3 Datenbankmodell als Krähenfußdiagramm
- 4.4 Erläuterungen zu Aufbau und Inhalt der Tabellen
- 4.5 SQL-Skripte, Trigger und Funktionen

5. Softwareentwurf

- 5.1 Registrierung und Login
- 5.2 Spieler kaufen
- 5.3 Ranglisten einsehen
- 5.4 Saisons und Turniere anlegen
- 5.5 Turnier in Saison eintragen
- 5.6 Turnierergebnisse eintragen
- 5.7 Nutzerverwaltung

6. Design

7. Frameworks

- 7.1 Bootstrap
- 7.2 Less.js
- 7.3 JQuery

8. Gewählte Scriptsprachen zur Realisierung der Funktionalitäten

9. Datenbankkommunikation

9.1 Aufbau der Datenbankverbindung

9.2 Datenabfrage und -eingabe mit php mysqli

9.3 Unterteilung in Autor, Nutzer und Nutzer-Spieler Abfragen

9.4 Umleitung einer Seite durch Javascipt statt PHP

10. Austausch zwischen PHP und Javascript - Variablen durch JSON Kodierung

11. Nutzerbezogene Funktionalitäten

11.1 Registrierung, Login und Logout

11.2 Personalisierbare Seiten des Nutzers

11.3 Änderung der Nutzerangaben

12. Verwaltung der Turnier- und Spielerinformationen durch einen Autor

12.1 Saisoneingabe

12.2 Turniereingabe

12.3 Turnierergebnisse in Saison eingeben

12.4 Ergebniseingabe nach einem Turnier

12.5 Platzeingabe der Spieler

12.6 Spieler anlegen

12.7 Spielerdaten aktualisieren

13. Verwaltung der Nutzer durch einen Autor

14. Ranglisten anzeigen

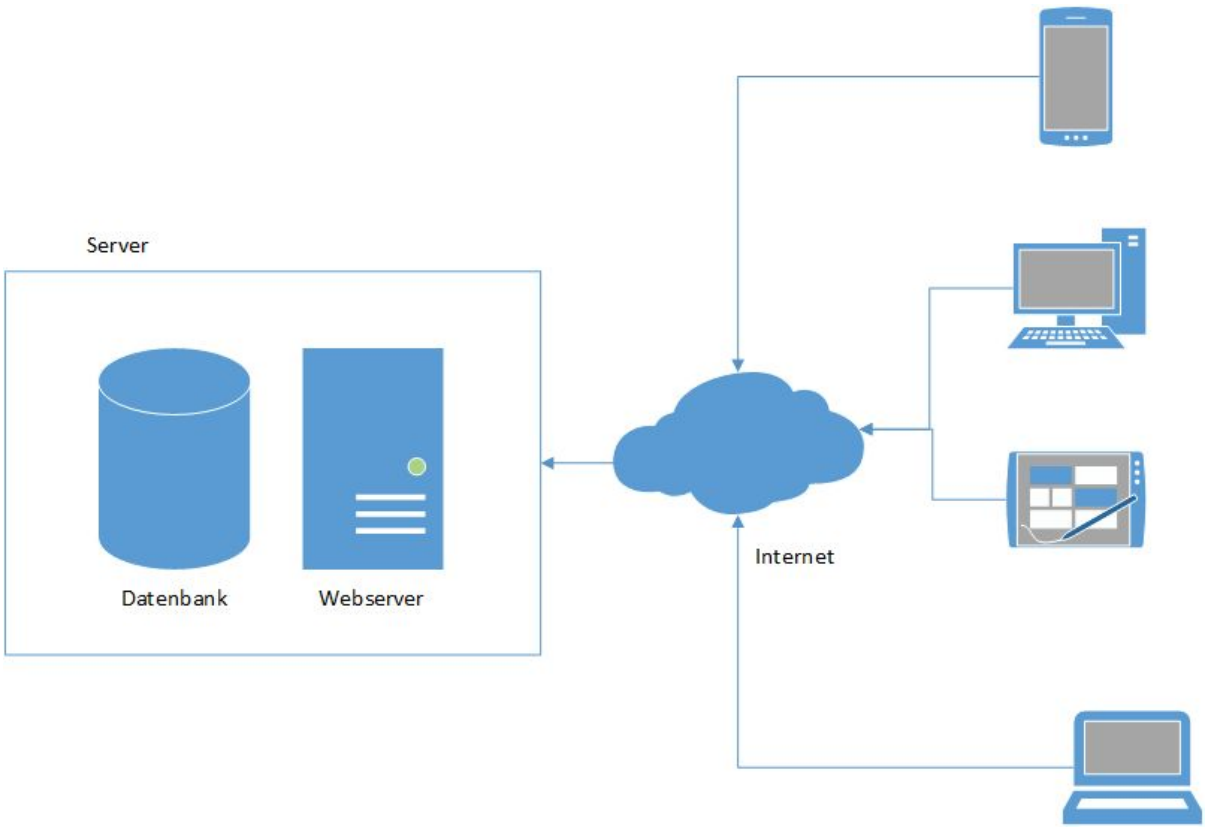
14.1 Turnier Ranglisten Spieler Anzeigen

14.2 Turnier Ranglisten Nutzer Anzeigen

14.3 Gesamt Ranglisten Nutzer Anzeigen

1. Einführung

Diese Anwendung ist als Studentenprojekt von Studenten der Hochschule Bochum entstanden. Es dient zur Erfahrungsammlung in Projekten und um die Eigenarbeit zu fördern. Wir haben uns dazu entschieden, einen Tennismanager, ähnlich zu dem Fußballmanager von Kicker, zu entwickeln.



Die Anwendung läuft auf einem Server, den sich die Datenbank und der Webserver teilen. Sie ist von allen Geräten, die über einen Browser verfügen, über das Internet erreichbar und voll optimiert für Smartphones, Tablets und PCs.

Anmerkung: In dieser Dokumentation werden zur Unterscheidung grundsätzlich die Begriffe Nutzer und Spieler verwendet, wobei Nutzer einen registrierten User bezeichnet und Spieler einen Tennisspieler. Spieler ist also in keinem Zusammenhang als Benutzer zu verstehen.

2. Spielablauf, Regeln und Mechaniken

2.1 Idee des Spiels

Die Grundidee des Spiels ist, dass ein Nutzer sich ein Team aus bis zu 6 Tennisspielern zusammenstellen, indem sie diese unter Vertrag nehmen, und Punkte für deren Erfolge in Turnieren bekommen. Platziert sich ein Spieler auf den vorderen Rängen eines Turniers, so erhält er einen Punktebetrag, der vom Rang und dem Preisgeld des Turniers abhängt. Alle Nutzer, die den Spieler zu diesem Zeitpunkt unter Vertrag haben, bekommen diese Punkte gutgeschrieben. Die Nutzer können sich in einzelnen Turnieren oder der ganzen Saison vergleichen. Der Nutzer mit den meisten Punkten in einem Turnier oder einer Saison ist Turnier- bzw Saisonsieger.
Verträge können jederzeit gekündigt und neu abgeschlossen werden. Die Idee ist, dass Nutzer versuchen, das beste Team für jedes Turnier zusammen zu stellen und ihr Wissen über die aktuellen Geschehnisse im Sport und ihre Einschätzung der Spieler nutzen.
Zudem können Nutzer ein Profil pflegen und die Profile anderer Nutzer einsehen. Auf der Home-Seite des Tennismanagers ist der ATP News Feed mit allen aktuellen Informationen einsehbar.

2.2 Übernahme von Daten der ATP

Eine Saison im Spiel geht genau wie bei der Association of Tennis Professionals, kurz ATP, ein Kalenderjahr. Die Spieler, die den Nutzern in einer Saison zur Verfügung stehen, sind die Top 100 der ATP-Wetrangliste des Vorjahres. Sie werden am Ende einer Saison und vor Beginn der nächsten mit ihrem Rang und ihren Punkten von der ATP-Weltrangliste eingetragen. Diese Werte sind die Grundlage für die nächste Saison im Spiel und werden während dieser nicht geändert. Die Ergebnisse aller ATP Grand-Slam, ATP World Tour 250, 500 und

Masters 1000, ATP World Tour Finals und Davis Cup Turniere werden eingetragen und Punkte für die Platzierungen vergeben.

2.3 Spielbalance

In der ATP-Weltrangliste zeigt sich eine sehr starke Dominanz des erstplatzierten Spielers, darunter nimmt die Differenz zwischen den Rängen immer weiter ab. Der erste Platz bei Turnieren sticht sowohl an Preisgeld als auch Punkten sehr hervor, auch hier werden die Unterschiede nach unten flacher. Diese Realitäten haben wir im Spiel übernommen, wollten aber dennoch erreichen, dass nicht nur der Kauf des führenden Spielers sinn- bzw reizvoll ist.
Konkret haben wir dies bei den Kaufpreisen der Spieler und der Berechnung der Punkte umgesetzt.
Die Kaufpreise werden zu Beginn der Saison berechnet und bleiben während der Saison unverändert. Sie werden durch Multiplikation der Weltranglistenpunkte eines Spielers mit dem Faktor 1.000 berechnet. Damit liegen die Preise zwischen 500.000 und 20.000.000 $, wobei der erstplatzierte Spieler am oberen Rand liegt, die nächstplatzierten im Mittelbereich, gefolgt von einem immer flacher werdenden Gefälle. Da man bis zu 6 Spieler kaufen kann und das verfügbare Guthaben 20.000.000 $ beträgt, werden so schon andere Spieler interessant.
Die Punkte, die ein Rang bei einem Turnier gibt, hängen von zwei Faktoren ab: dem Rang und der Preisgeld-Kategorie des Turniers. Die Turniere werden beim Anlegen in 21 Preisgeldklassen unterteilt, je höher das Preisgeld desto höher die Klasse. Der gleiche Rang bringt grundsätzlich bei jedem Turnier die gleichen Punkte, jedoch werden diese dann noch mit der Preisgeldklasse multipliziert, so dass wichtige Turniere mehr Punkte geben und eine gute Platzierung einflussreicher ist. Entscheidend für die Spielbalance ist jedoch die Berechnung der Punkte aus dem Rang. Hier wollten wir zum einen ein realistisches Gefälle erreichen, zugleich sollte man mit mehreren gut platzierten Spielern durchaus mehr Punkte erreichen können als mit nur dem Sieger des Turniers. Die Formel für Punkte abhänging vom Rang, die sich daraus am Ende ergab, lautet

$$\text{Punkte} = 1.600 * 0{,}5^{(\text{Rang}-1)} + 400$$

Das Prinzip von Halbfinale, Viertelfinale usw wird durch Vergabe gleicher Ränge abgebildet.

3. Installationsanleitung

3.1 Voraussetzungen

Um die Anwendung auf einem Server zu installieren sind ein paar vorhandene Anwendungen nötig. Die erste und wichtigste Anwendung ist ein Apache-Webserver, der so konfiguriert sein muss, dass .htaccess Dateien erkannt werden und deren Inhalte die serverseitige Konfiguration überschreiben. Eine MariaDB Datenbank der Version 5.5.41 oder höher und PHP5 müssen vorhanden sein.

3.2 Installation

Sobald diese grundlegenden Vorraussetzungen gegeben sind, kann mit der Installation begonnen werden.
Um die Webseite an sich zu installieren, müssen lediglich die Seiten auf den Server kopiert werden. Danach muss die Datenbank angelegt werden. Hierzu wird ein Admin-Account auf MariaDB benötigt. Um die Datenbank auf den MariaDB Server aufzuspielen, müssen Sopra_Tennis_SQL.sql und Sopra_Tennis_SQL_Trigger.sql importiert werden. Danach sollten noch kleine Sicherheitsvorkehrungen in MariaDB getroffen werden. Ein eigener Datenbank-Benutzer, der als Einziger Zugriff auf die sopraTennis Datenbank hat, sollte angelegt werden.
Um dann die Webseite in Verbindung mit der Datenbank zu nutzen, sollten Sie die Datei connection.php bearbeiten und dort in die Zeile

```
$conn = new mysqli($servername, $username, $password, $dbname)
```

die MariaDB Server IP eintragen, den extra angelegten Benutzernamen mit seinem Passwort und den Namen der Datenbank, in diesem Fall sopraTennis. Ab diesem Punkt kann man sich auf der Webseite registrieren. Man sollte einen ersten Account registrieren und ihm die Rolle Admin zuweisen. Dazu wird eine SQL Abfrage benötigt:

```
UPDATE nutzer SET rolleID = 1 WHERE nutzername = "ihr nutzername";
```

Dabei ersetzen Sie “ihr nutzername” inklusive der Anführungszeigen durch Ihren Nutzernamen. Danach können weitere Accounts bequem über eine Funktion auf der Webseite in den Autoren Status gehoben werden.

Sobald diese Schritte alle durchlaufen wurden, kann die Webseite für neue Benutzer geöffnet werden.

4. Datenbank

4.1 DBMS

Das verwendete DBMS ist MariaDB. Da MariaDB auf MySQL basiert, gibt es hierfür sehr viele Hilfen und Nachschlagewerke, was einer unserer Hauptgründe war, MariaDB zu verwenden. Ein anderer sehr großer Punkt ist die Schnelligkeit und Erweiterbarkeit von MariaDB.

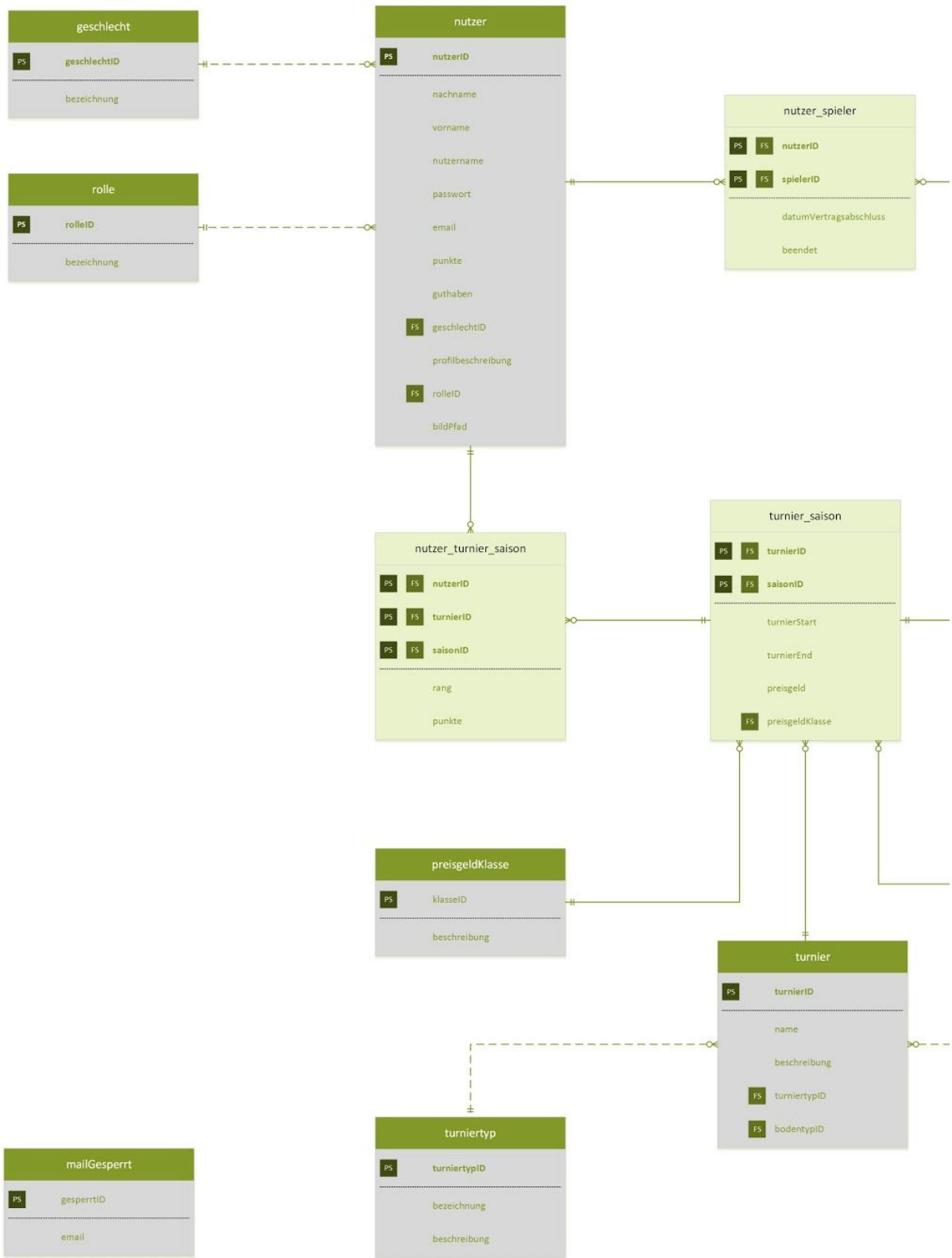
4.2 Datenbankentwurf

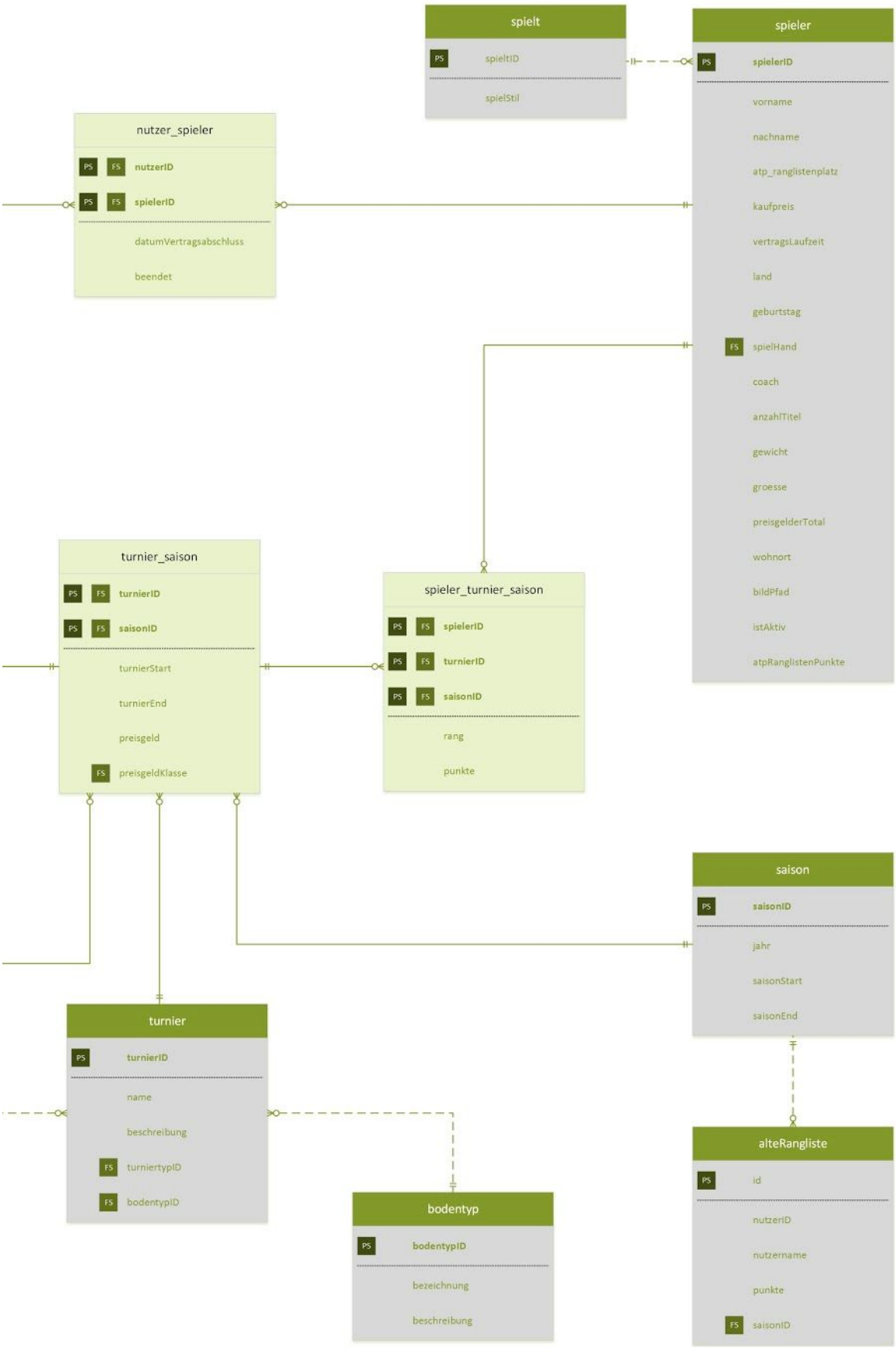
Der Datenbankentwurf hat ebenso wie der Softwareentwurf mit Microsoft Visio stattgefunden. Es wurde zunächst ein grobes Entity-Relationship-Modell entworfen, welches dann in ein Krähenfußdiagramm übertragen wurde, das auch die Hilfstabellen enthielt, die zur Umsetzung aller Beziehungen nötig waren. Dieses Modell wurde normalisiert und im Rahmen der Entwicklung angepasst und verfeinert. Bis in die späten Entwicklungsphasen fand eine Anpassung der Datenbankstruktur an die Anforderungen der Anwendung statt.

Das vollständige, aktuelle Krähenfußdiagramm, dass Aufbau und Beziehungen aller Tabellen zeigt, findet sich auf den übernächsten beiden Seiten. Primär- und Fremdschlüssel sind mit den grünen Kästen mit dem Inhalt “PS” bzw “FS” gekennzeichnet. Die Herkunft der Fremdschlüssel ist aus ihrem Namen ersichtlich. Die hellgrünen Tabellen stellen schwache Entitäten dar, deren Primärschlüssel eine Kombination aus Fremdschlüsseln ist. Sie setzen m:n-Beziehungen der beteiligten Entitäten um. Entsprechend haben sie starke Beziehungen (durchgezogene Linien). Die Umsetzung ist so gestaltet, dass zwischen allen Tabellen “MUSS 1 zu KANN n” Beziehungen gelten. Dies gilt als die beste Datenbank-Beziehung, da sie durch einen Fremdschlüssel einfach und vollständig umgesetzt werden kann. Somit müssen keine Beziehungen umständlich durch Trigger umgesetzt werden und Abfragen über beliebig viele Tabellen mit Joints können durch eine einfache

Verkettung von Schlüsselattributen umgesetzt werden. Redundanzen wurden möglichst vermieden und Attribute wie Geschlecht oder Rolle in eigene Tabellen ausgelagert, die mit Standardwerten befüllt sind. Das Datenbankmodell befindet sich in der dritten Normalform.

4.3 Datenbankmodell als Krähenfußdiagramm




spielt
PS spieltID
spielStil
spieler
PS spielerID
vorname
nachname
atp_ranglistenplatz
kaufpreis
vertragsLaufzeit
land
geburtstag
FS spielHand
coach
anzahlTitel
gewicht
groesse
preisgelderTotal
wohnort
bildPfad
istAktiv
atpRanglistenPunkte
nutzer_spieler
PS FS nutzerID
PS FS spielerID
datumVertragsabschluss
beendet
turnier_saison
PS FS turnierID
PS FS saisonID
turnierStart
turnierEnd
preisgeld
FS preisgeldKlasse
spieler_turnier_saison
PS FS spielerID
PS FS turnierID
PS FS saisonID
rang
punkte
saison
PS saisonID
jahr
saisonStart
saisonEnd
turnier
PS turnierID
name
beschreibung
FS turniertypID
FS bodentypID
bodentyp
PS bodentypID
bezeichnung
beschreibung
alteRangliste
PS id
nutzerID
nutzername
punkte
FS saisonID

4.4 Erläuterungen zu Aufbau und Inhalt der Tabellen

- `geschlecht` - enthält als Standartwerte 1 - “Maennlich”; 2 - “Weiblich” und 3 - “Keine Angabe”
- `rolle` - enthält die Standartwerte 1 - “Admin”; 2 - “Nutzer”, 3 - “Autor” [Erklärung der Rollen entweder hier oder Verweis]
- `nutzer` - enthält die Daten aller registrierten Nutzer und erlaubt das Erzeugen einer Nutzer-Rangliste nach Gesamtpunktzahl für die aktuelle Saison (Attribut `punkte`) und das Feststellen von Saisonsiegern
- `spielt` - enthält die Standartwerte 1 - “Rechtshaendig”; 2 - “Rechtshaendig, einhaendige Rueckhand”; 3 - “Rechtshaendig, zweihaendige Rueckhand”; 4 - “Linkshaendig”; 5 - “Linkshaendig, einhaendige Rueckhand”; 6 - “Linkshaendig, zweihaendige Rueckhand”
- `spieler` - enthält die Daten aller Spieler der aktuellen und vergangener Saisons. Indem einfach `istAktiv` auf `false` gesetzt wird, bleiben aktuell nicht benötigte Spieler in der Datenbank und können bei Bedarf wieder verwendet werden. `atp_ranglistenplatz` ist der Platz auf der ATP-Weltrangliste und `atpRanglistenPunkte` die ATP-Punkte am Ende der Vorsaison, die während der ganzen aktuellen Saison im Spiel gleich bleiben. `preisgelderTotal` enthält die Summe aller bisher gewonnenen Preisgelder des Spielers. Das Attribut `vertragsLaufzeit` wird aktuell nicht verwendet, es stammt aus einer Phase der Entwicklung, in der Vertragslaufzeiten geplant waren. Es wurde in der Datenbank gelassen, falls dies noch einmal implementiert werden sollte.
- `nutzer_spieler` - hier wird ein Datensatz angelegt, wenn ein Nutzer einen Spieler unter Vertrag nimmt. Bei Vertragskündigung wird der Boolean `beendet` auf `true` gesetzt. Das Datum, an dem der Vertrag abgeschlossen wurde, wird gespeichert und ist Teil des Primärschlüssels, so dass ein neuer Datensatz angelegt wird, wenn derselbe Nutzer denselben Spieler später noch einmal unter Vertrag nimmt. Somit existiert für jede “Vertragsphase” ein

Eintrag in der Datenbank. Über diese Tabelle wird entschieden, welche Nutzer Punkte für die Erfolge eines Spielers bekommen.

- `saison` - enthält die Saisons mit exaktem Start- und Endtag
- `turnier` - enthält die Daten der Turniere, die über die Saisons hinweg gleich bleiben, so dass diese nur einmalig gespeichert oder ggf geändert werden müssen
- `turnier_saison` - enthält die Daten der Turniere, die sich mit jeder Saison ändern können, für jedes Turnier in jeder Saison
- `spieler_turnier_saison` - enthält Rang und Punkte der betreffenden Spieler in einem bestimmten Turnier in einer bestimmten Saison und erlaubt somit die Erzeugung von turnierspezifischen Spieler-Ranglisten (daran können z.B. die Nutzer nach einem Turnier nachvollziehen, wie viele Punkte sie von welchem Spieler bekommen haben)
- `nutzer_turnier_saison` - enthält Rang und Punkte der Nutzer aus einem bestimmten Turnier in einer bestimmten Saison und erlaubt die Erzeugung von turnierspezifischen Nutzer-Ranglisten und das Feststellen von Turniersiegern
- `turniertyp` - enthält als Standardwerte 1 - “Grand-Slam”; 2 - “ATP World Tour Masters 1000”; 3 - “ATP World Tour 500”; 4 - “Davis Cup”; 5 - “ATP World Tour 250”; 6 - “ATP World Tour Finals”, jeweils mit Beschreibung
- `bodentyp` - enthält die Standardwerte 1 - “Sand”, “Outdoor”; 2 - “Hartplatz”, “Indoor”; 3 - “Rasen”, “Outdoor”; 4 - “Kunstrasen”, “Outdoor”; 5 - “Granulat”, “Indoor”; 6 - “Teppich”, “Indoor”
- `preisgeldKlasse` - enthält 21 Standardwerte von 1 - “'0 - 500k” bis 21 - “> 10m”, gestaffelt in 500k Schritten. Dieses dient der Unterteilung von Turnieren in die Kategorien entsprechend der Höhe ihres Preisgeldes in einer Saison. Turniere mit einer höheren Kategorie geben mehr Punkte für die gleichen Ränge als Turniere mit geringerem Preisgeld. So wird die unterschiedliche Wichtigkeit der Turniere berücksichtigt.
- `alteRanglisten` - dient dem Abspeichern alter Saison-Nutzer-Ranglisten
- `mailGesperrt` - dient dem Speichern gesperrter E-Mail-Adressen, als einzige nicht mit anderen Tabellen verknüpft

4.5 SQL-Skripte, Trigger und Funktionen

Das SQL-Skript zum Anlegen der Tabellenstruktur und zum Befüllen einiger Tabellen wie `turniertyp` oder `rolle` mit Standardwerten ist in Sopra_Tennis_SQL.sql zu finden. In Sopra_Tennis_SQL_Trigger.sql sind dem Namen entsprechend alle Trigger, aber auch Funktionen und Views enthalten, die zum reibungslosen Betrieb der Anwendung benötigt werden und einige Prozesse automatisieren.

Die Trigger, Funktionen und Views sind im Folgenden einzeln erklärt:

- `TRIGGER punktevergabe` - dieser Trigger reagiert auf das Einfügen eines Datensatzes in `spieler_turnier_saison` und verteilt die Punkte, die der Spieler erhalten hat, automatisch an alle Nutzer, die diesen Spieler aktuell unter Vertrag haben, indem er die Punkte auf den Gesamtpunktestand der Nutzer für diese Saison addiert. Zusätzlich führt er auch ein Update auf `nutzer_turnier_saison` aus und addiert auch hier den entsprechenden Nutzern die Punkte hinzu.
- `TRIGGER vertragBeginn` - dieser Trigger dient dem Abzug des Kaufpreises eines Spielers vom Guthaben eines Nutzers, wenn dieser ihn unter Vertrag nimmt. Er wird bei einem `INSERT` in `nutzer_spieler` ausgeführt, was dem Beginn eines neuen Vertrages entspricht.
- `TRIGGER vertragEnde` - dieser Trigger erstattet dem Nutzer den Kaufpreis analog zurück, wenn der Vertrag endet. Dies entspricht einem `UPDATE` in `nutzer_spieler`, bei welchem das Attribut `beendet` eines oder mehrerer Datensätze von `true` auf `false` geändert wird. Der Trigger überprüft dies für jeden geänderten Datensatz und handelt entsprechend.
- `TRIGGER preisEintragen` - dieser Trigger wird durch das Einfügen eines neuen Spielers ausgelöst und berechnet anhand des Attributs `atpRanglistenPunkte` den Preis des Spielers und weist ihn dem Spieler zu.

- `PROCEDURE zuruecksetzen()` - diese Funktion setzt das Guthaben aller Nutzer auf das Basisguthaben, die Punkte aller Nutzer auf 0 und beendet alle laufenden Verträge, indem beendet in den entsprechenden Datensätzen in `nutzer_spieler` auf `true` gesetzt wird. Sie wird vor Beginn einer neuen Saison verwendet, um alles auf die Startbedingungen zurück setzen. Das Basisguthaben für Nutzer ist in dieser Funktion als Variable festgelegt und kann dementsprechend dort geändert werden.
- `viewNutzerRangliste` - diese View dient als abgespeicherte Abfrage, die alle Nutzer mit ihrem Nutzernamen und Punkten ausgibt, absteigend geordnet nach `punkte`, also ihrer Gesamtpunktzahl in der laufenden Saison.

Bei der Entscheidung, welche Logik in Triggern und welche abseits der Datenbank stattfindet, ist die Natur von Triggern maßgebend. Da sie nur auf Datenbank-Events reagieren und auf der Datenbank arbeiten, eignen sie sich am besten für Vorgänge wie die Punkteverteilung, wo ein Wert von einer Tabelle in eine andere übertragen werden soll und Datenbank-Logik benötigt wird. Die Berechnung der Punkte selbst hingegen geschieht in PHP, welches auch als Sprache für Formeln wesentlich besser geeignet ist. Eine solche Berechnung bei jedem Durchlauf des Triggers wäre wesentlich zu umständlich und unperformant auf der Datenbank. So muss PHP-seitig aber nur das `INSERT` Statement in spieler_turnier_saison als Query gesendet werden und die langen `UPDATE` Befehle laufen automatisch direkt auf der Datenbank. Nur einfache Berechnungen wie die der Spielerpreise machen direkt auf der Datenbank Sinn.
Auch bei vertragBeginn und vertragEnde handelt es sich um einfache Datenbank-Operationen als Folge von einem `INSERT` oder `UPDATE`. Hier entsteht auch eine übersichtliche Trennung der Verantwortungen: PHP-seitig werden User-Aktivitäten (Vertrag abschließen oder beenden) in ihre Entsprechung auf der Datenbank übersetzt und weitergegeben. Weiter nötige Datenbankoperationen laufen dann automatisiert Datenbankintern ab. So ist an der PHP-Schnittstelle auch keine Kenntnis darüber nötig, ob die Spiellogik weitere Aktionen erfordert, und sie können nicht vergessen werden, da der Trigger die Umsetzung garantiert.

5. Softwareentwurf

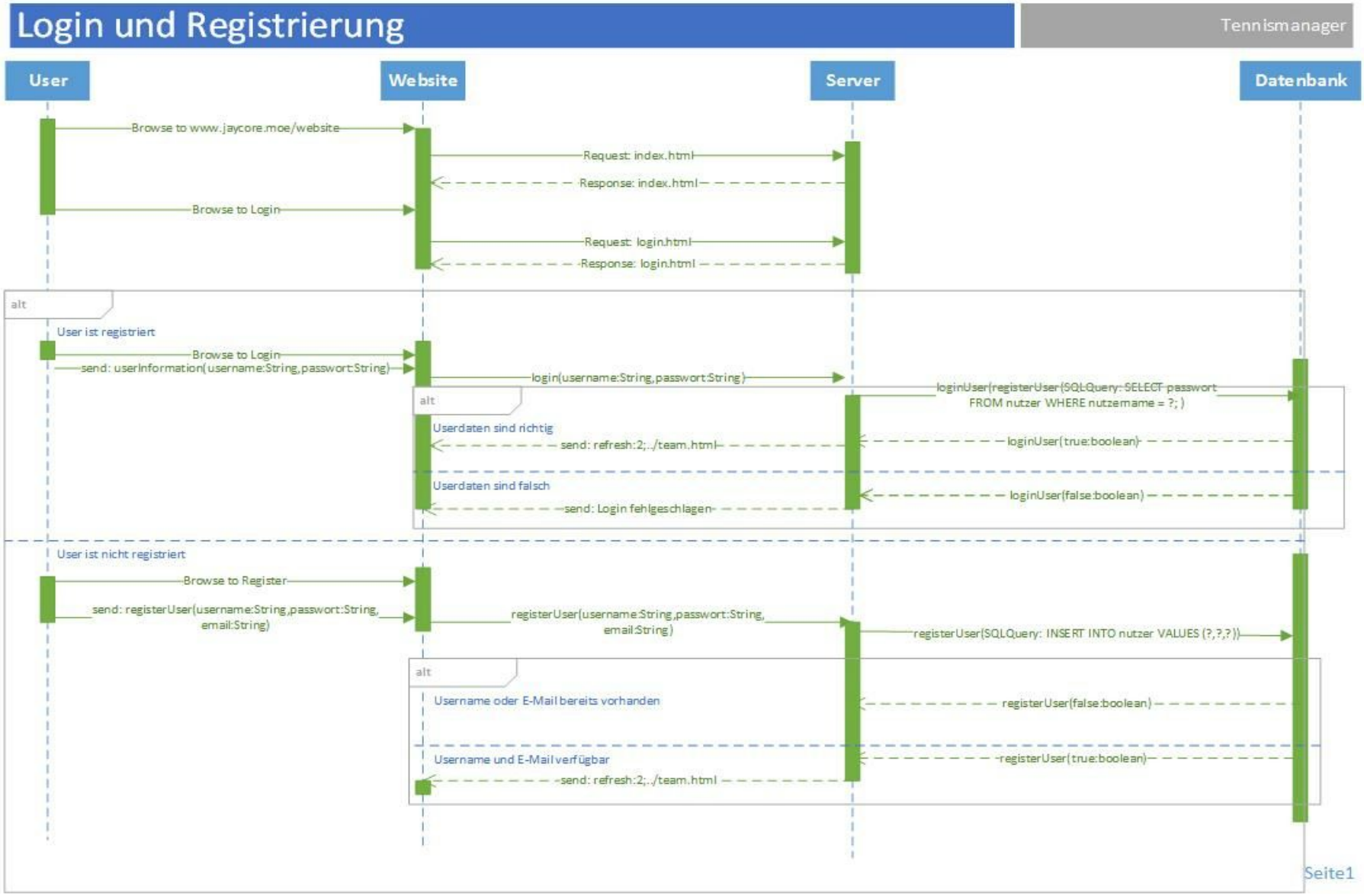
Der Softwareentwurf wurde mit Hilfe des Programmes Microsoft Visio 2013 entworfen unter Verwendung von UML-Sequenzdiagrammen.
Sequenzdiagramme bestehen aus verschiedenen Objekten und dienen dem Entwerfen und Verdeutlichen von Systeminternen Vorgängen.
Die blau hinterlegten Kästchen sind Akteure, hier User, Website, Server und Datenbank. Jeder dieser Akteure besitzt eine Lebenslinie und auf der jeweiligen Lebenslinie ein in grün eingefärbtes Rechteck. Dieses symbolisiert den Zeitraum, in dem der Akteur eine oder mehrere Operationen ausführt.
Die durchgezogenen Pfeile sind synchrone Nachrichten, die ein Akteur an einen neuen anderen Akteur sendet. Die gestrichelten Pfeile sind Antwortnachrichten auf eine vorhergegangene Nachricht.
Rahmen bzw. Blöcke die mit “Alt” markiert sind, symbolisieren if-else Blöcke, der Text in schwarz die Bedingung zur Ausführung dieses Blockes.

5.1 Registrierung und Login

Zur Registrierung beim Tennismanager wird ein Nutzername, Passwort sowie eine valide E-Mail Adresse benötigt. Keine Pflichtfelder sind Name, Vorname sowie Geschlecht.
Gibt ein Nutzer valide Daten an und sendet diese an den Server, wird die Methode `registerUser()` aufgerufen. Sie nimmt Nutzername, Passwort und E-Mail an und führt eine `INSERT INTO` SQLQuery auf die Datenbank durch. Ist die verwendete E-Mail oder der Nutzername bereits in der Datenbank vorhanden, erhält der Server ein `false` zurück und der Registrierungsversuch schlägt fehl. Der Nutzer wird aufgefordert, eine valide E-Mail bzw Nutzernamen zu wählen und erneut abzuschicken.
Erhält der Server ein `true` zurück, ist die Registrierung erfolgt.
Das Passwort wird automatisch gehasht und gesalted und der entstandene Hash wird als Passwort der Datenbank übergeben. Der Nutzer wird nun durch

"$_SESSION["berechtigter_user"]=1;" automatisch eingeloggt und per Refresh auf die team.html Seite weitergeleitet.

Der Login erfolgt durch Eingabe des Nutzernamens und des Passwortes auf login.html. Die Funktion `loginUser()` nimmt die eingegeben Werte entgegen, hashed das Passwort und führt eine `SELECT` SQLQuery auf die Datenbank aus. Stimmt der übergebene Hash mit dem in der Datenbank gespeichertem Hash überein, wird der Nutzer eingeloggt.
`$_SESSION["berechtigter_user"]` wird auf 1 gesetzt und `refresh:2;../team.html` wird ausgeführt.


Login und Registrierung
Tennismanager
User
Website
Server
Datenbank
Browse to www.jaycore.moe/website
Request: index.html
Response: index.html
Browse to Login
Request: login.html
Response: login.html
alt
User ist registriert
Browse to Login
send: userInformation(username:String,passwort:String)
login(username:String,passwort:String)
loginUser(registerUser(SQLQuery: SELECT passwort FROM nutzer WHERE nutzername = ?;)
alt
Userdaten sind richtig
send: refresh:2;../team.html
loginUser(true:boolean)
Userdaten sind falsch
send: Login fehlgeschlagen
loginUser(false:boolean)
User ist nicht registriert
Browse to Register
send: registerUser(username:String,passwort:String, email:String)
registerUser(username:String,passwort:String, email:String)
registerUser(SQLQuery: INSERT INTO nutzer VALUES (?,?,?))
alt
Username oder E-Mail bereits vorhanden
registerUser(false:boolean)
Username und E-Mail verfügbar
registerUser(true:boolean)
send: refresh:2;../team.html
Seite1

5.2 Spieler kaufen

Die teambuy.html Seite wird erstellt durch die Funktion `changeSpieler()`. `changeSpieler()` ruft per imbedded PHP die Methoden `getSpielerInformationUnterVertrag()`, `getNutzerInformation` und `testeVertraege()` auf. Diese Funktionen bekommen durch `$_SESSION["nutzername"]` den aktuellen Nutzer übergeben und führen SQL Queries aus. Die Ergebnisse der Queries werden per `json_encode()` in jSON umgewandelt und von `changeSpieler()` zu lesbarem HTML strukturiert und ausgegeben.

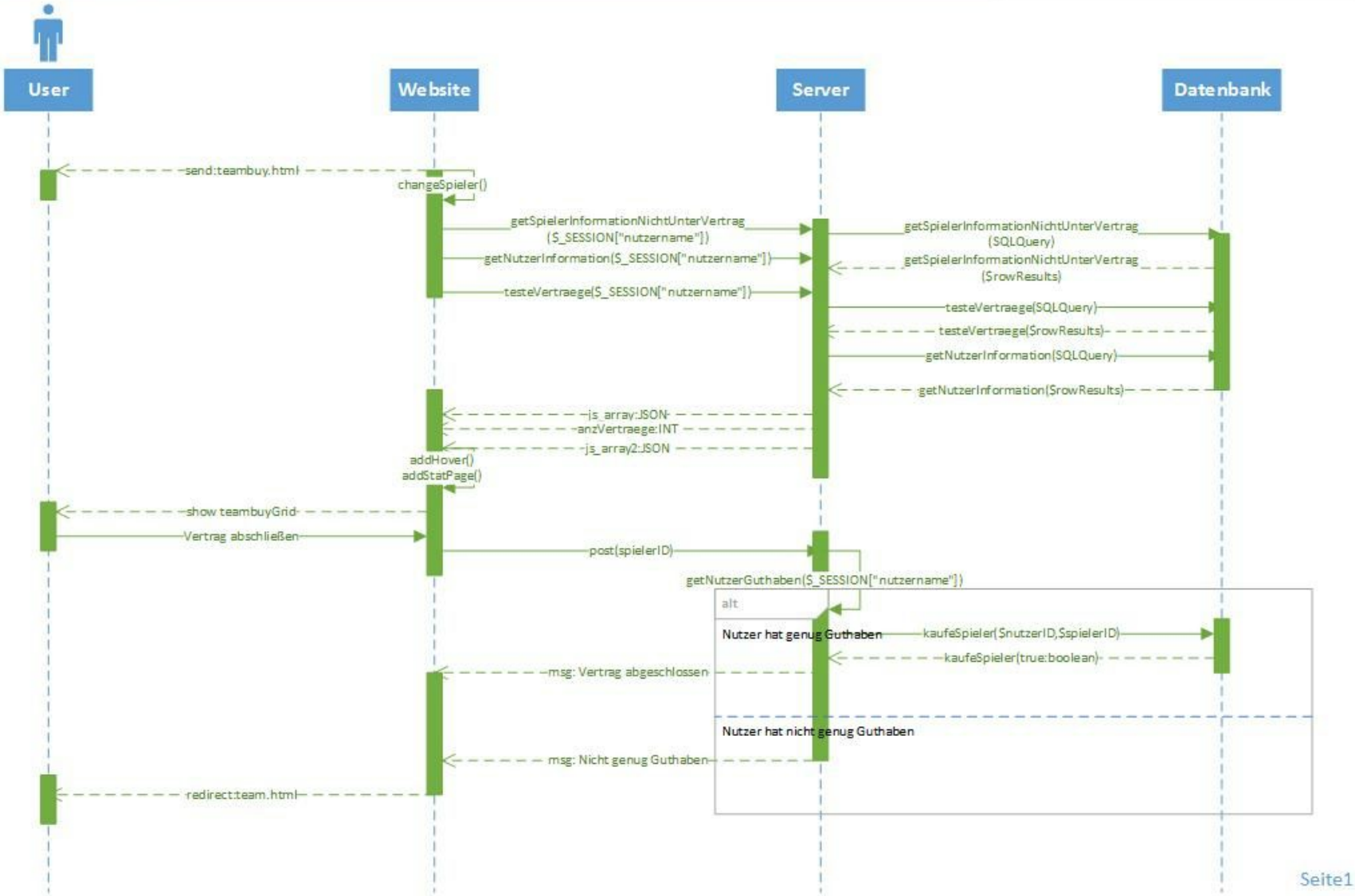
Die Methoden `addHover()` und `addStatPage()` dienen der Anzeige der in der Datenbank gespeicherten Spieler Informationen, diese werden aus dem jSON-Objekt ausgelesen.

Versucht ein Nutzer, einen Vertrag abzuschließen, wird per `getNutzerGuthaben()` sein aktuelles Guthaben ermittelt und mit dem Kaufpreis des gewählten Spielers verglichen. Ist sein Guthaben ausreichend, so wird eine `INSERT INTO` Query auf die Datenbanktabelle `nutzer_spieler` angewendet. Ist die Anweisung erfolgreich, wird `true` zurück gegeben und der Spieler wird auf team.html geleitet.

Hat der Nutzer nicht genügend Guthaben, erhält er eine Fehlermeldung.

Spieler Kaufen

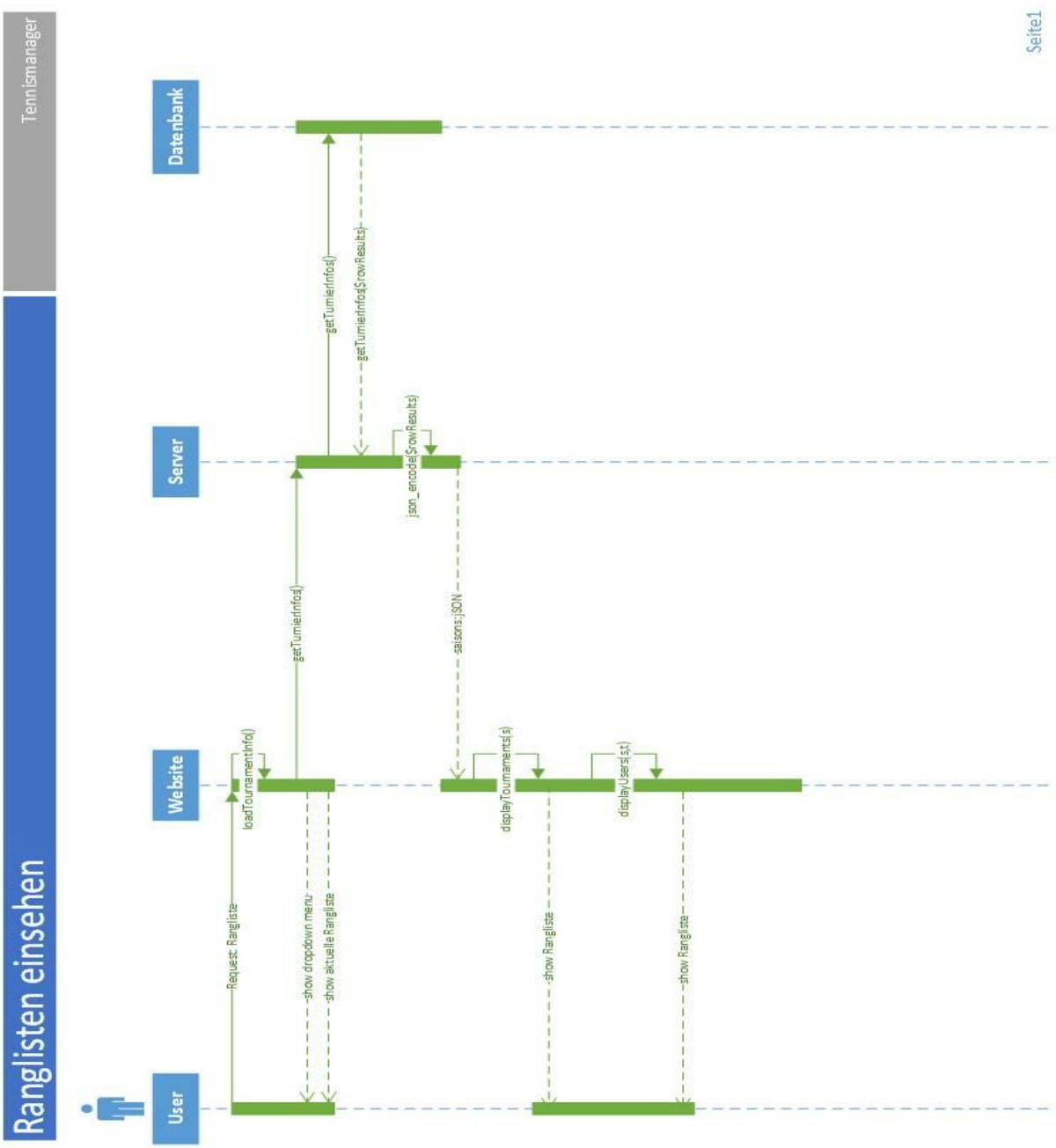
Tennismanager



Seite1

5.3 Ranglisten einsehen

Wird eine der Ranglisten Seiten aufgerufen, erscheint ein Dropdown-Menü mit zugehörigem Content. Über diese kann die gewünschte Saison und / oder das gewünschte Turnier ausgewählt und angezeigt werden. Per PHP Mehodenaufruf und SQLQuery werden die relevanten Daten ausgelesen und für den Nutzer aufbereitet.

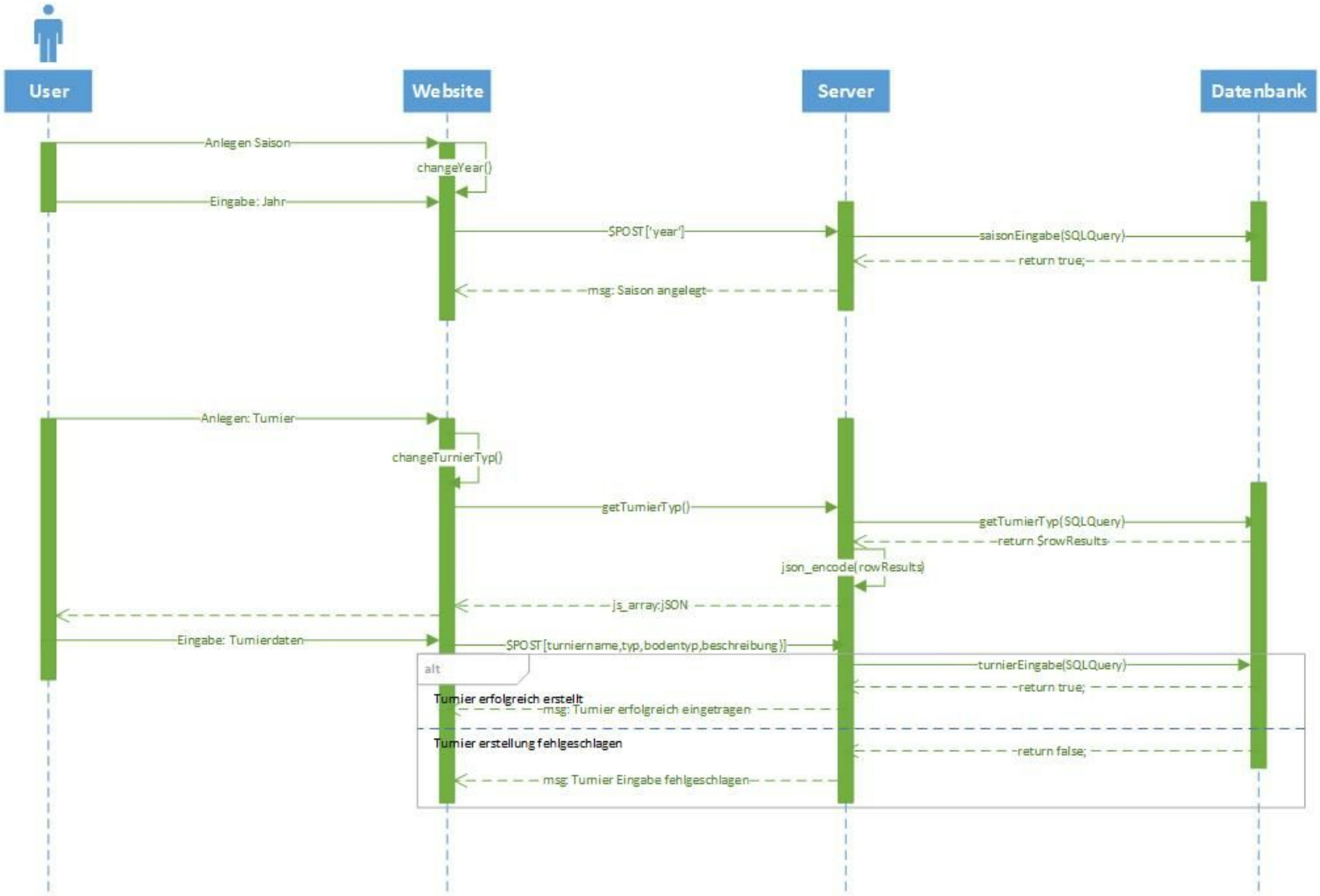


5.4 Saisons und Turniere anlegen

Das Anlegen einer neuen Saison oder eines neuen Turniers erfolgt durch das Aufrufen der jeweiligen Seiten. Beim Anlegen einer neuen Saison wird nur das gewünschte Jahr benötigt, dieses wird per Dropdown ausgewählt und per `$POST` an den Server übergeben. Per der Funktion `saisonEingabe()` wird die neue Saison in der Datenbank angelegt.

Möchte man ein neues Turnier eintragen, geschieht dies durch Eingabe der relevanten Daten. Der Turniertyp sowie der Bodentyp werden mit Hilfe von `getTurnierTyp()` aus der Datenbank geholt und in Dropdowns angezeigt. Nach Auswahl und Eintragung der gewünschten Daten werden diese per `$POST` an den Server übertragen und durch eine PHP SQLQuery in die Datenbank eingetragen.

Neues Turnier/Neue Saison anlegen

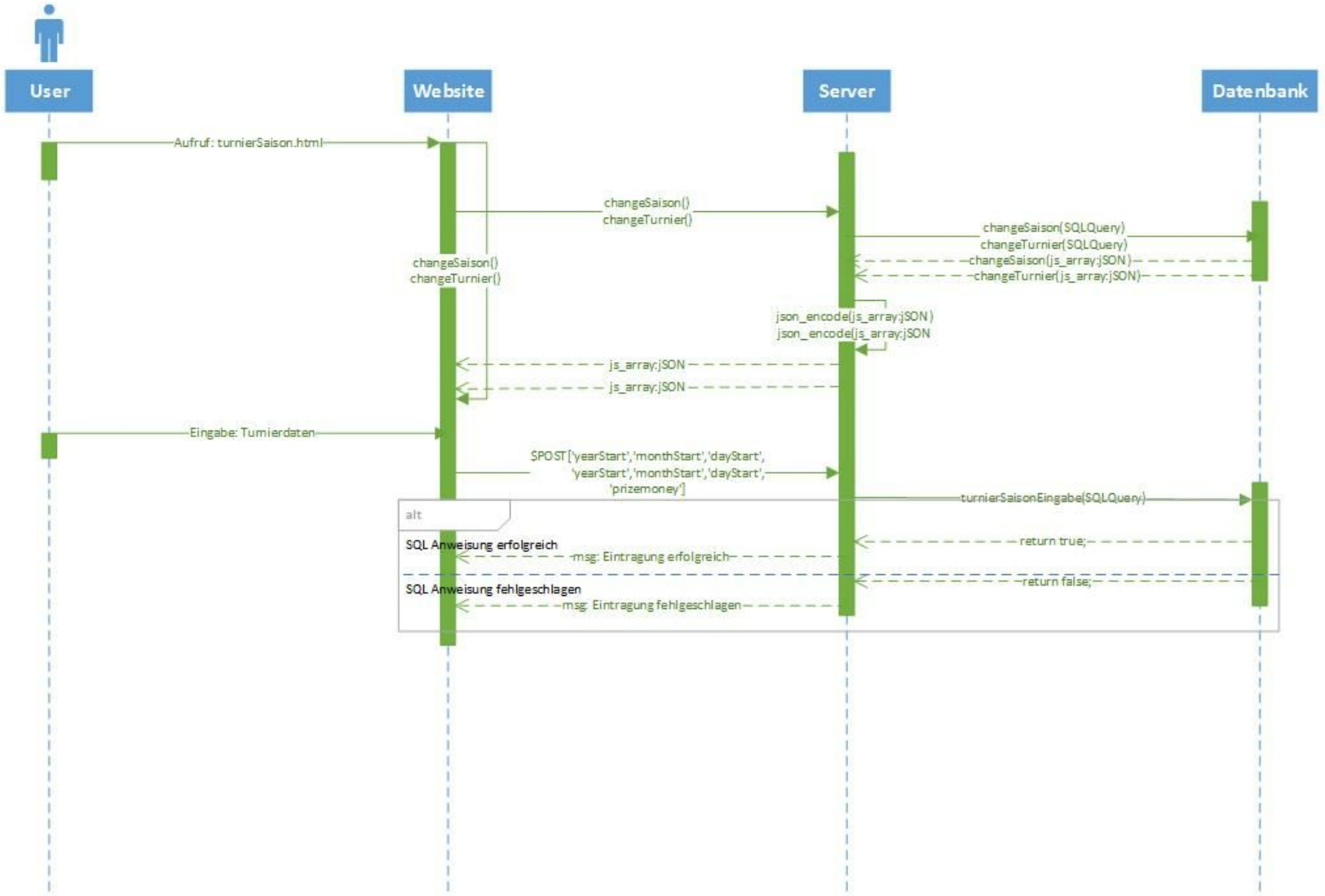


5.5 Turnier in Saison eintragen

Ein Turnier, welches im voherigen Schritt erstellt wurde, muss in eine Saison eingetragen werden, um verwendet werden zu können. Dies geschieht durch das Aufrufen der dafür vorgesehenen Seite. Die Funktionen `changeSaison()` und `changeTurnier()` werden onload geladen und sorgen dafür, dass die benötigten Daten in die Dropdown-Menüs geladen werden. Der Autor trägt nun das Start- und Enddatum sowie das Preisgeld des Turnieres ein und sendet das Formular ab. Wie in den obigen Schritten auch wird es per `$POST` gesendet und per PHP SQLQuery an die Datenbank geleitet, wo die Verknüpfung von Turnier und Saison erstellt wird.

Turnier in Saison eintragen

Tennismanager



5.6 Turnierergebnisse eintragen

Turnierergebnisse werden über das gleiche Verfahren eingetragen wie die Turniere auch. Hierzu dient eine in Rängen geschachtelte Sammlung von Dropdownmenüs. In diesem Menü werden die Namen der Spieler in dem entsprechenden Rang eingegeben und an den Server gesendet. Die weitere Verarbeitung erfolgt ähnlich wie in den vorherigen Punkten angegeben.

5.7 Nutzerverwaltung

Nutzerverwaltung ist analog zu den Eingabemasken der Saison und Turniere und technisch gleich. Die gewünschten Daten werden ausgelesen und ausgewählt, dann ergänzt und an den Server gesendet, welcher sie dann per `$POST` entgegennimmt und an die Datenbank sendet.

6. Design

Für das Design der Anwendung wurden zwei Mockups erstellt.

Das Erste (Abb. 1) ist relativ früh in der Entwicklung erstellt worden. Der rechte Teil der Seite in diesem Design war als News Feed und genereller Informations-Service gedacht.
Im zweiten Entwurf (Abb. 2) sind diese Funktionen jedoch auf die Home Seite gewandert, da sie nicht immer abrufbar sein müssen. Das Design ist durch den Wechsel schmaler geworden und ließ sich so einfacher auch für Smartphones und Tablets optimieren, ohne ein komplett neues Design zu entwerfen.

Abbildung 1: Tennismanager Mockup 1

Das Farbschema ist bei beiden Entwürfen dennoch gleich geblieben, genauso wie die Icons für die verschiedenen Menüpunkte. Diese Menüpunkte wurden nach ihren Themengebieten zusammengefasst. So gibt es “Home”, unter dem sich die

Hauptseite befindet, und “Profil”, unter dem sich das eigene Profil und diverse Bearbeitungsmöglichkeiten für eben dieses befinden sowie die eigenen Persönlichen Daten inkl Möglichkeit des Editierens. Unter dem Punkt “Team” gibt es sowohl das eigene Team zu sehen als auch die Möglichkeit, neue Spieler für sein Team unter Vertrag zu nehmen. ”Ranglisten” bietet Turnier Ranglisten, die die aktuellen Tennisspieler Ranglisten für die jeweiligen Turniere darstellen, Nutzer Gesamt Ranglisten, die die Gesamtpunkte der aktiven Nutzer in einer Saison darstellen und Nutzer Turnier Ranglisten, die die Punkte, die die Nutzer in einzelnen Turnieren bekommen haben, darstellen. Zuletzt gibt es “Hilfe”, worunter man die Hilfe erreichen kann.

Die Administrations Funktionen sind alle unter einem Menüpunkt “Administration” zusammen gefasst und nur sicht- und benutzbar, wenn man als Admin oder Autor eingeloggt ist.

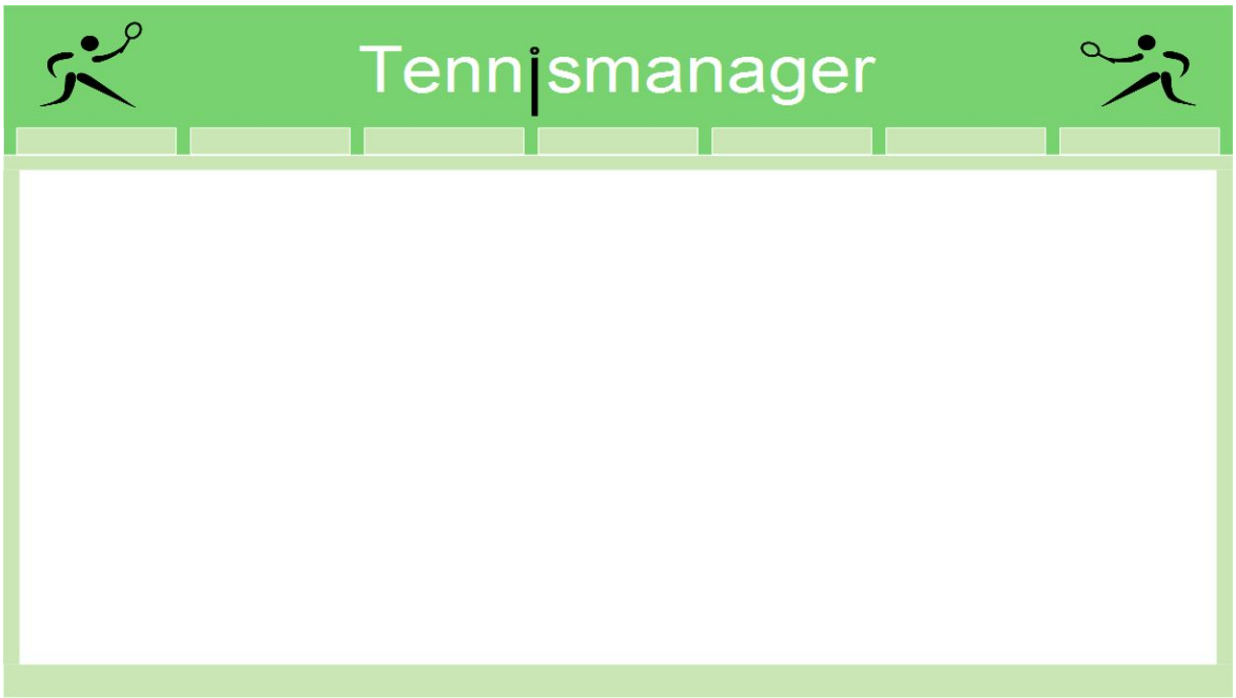


Abbildung 2: Tennismanager Mockup 2

7. Frameworks

7.1 Bootstrap

Das wichtigste Framework, das für die Webseite benutzt wurde, ist Bootstrap. Da Bootstrap von Haus aus ein Framework für Responsive Design ist und die Webseite immer erreichbar sein soll, war dies die beste Wahl. Desweiteren bietet Bootstrap viele Möglichkeiten für zukünftige Erweiterungen.

7.2 Less.js

Less ist ein CSS Framework und wird in Verbindung mit Bootstrap genutzt, um das Schreiben von CSS Skripten zu erleichtern und so ein besseres Endresultat zu erzielen. Die Less Dateien werden vorher mit Hilfe von Grunt, einem Javascript Task Runner, und diversen Skripten compiliert und in die Bootstrap Ordnerstruktur eingefügt. Less selbst lässt das Benutzen von Javascript zu, um verschiedene Design Elemente zu generieren und optimiert das resultierende CSS Konstrukt so, dass möglichst wenige Bytes übertragen werden.

7.3 JQuery

Obwohl viele der Funktionen in Standard Javascript geschrieben wurden, ist manchmal doch der Griff zu JQuery von Nöten gewesen, da diese Funktionen sonst viel zu lang und zu unübersichtlich geworden wären.

8. Gewählte Scriptsprachen zur Realisierung der Funktionalitäten

Zur Kommunikation zwischen dem Datenbankserver und einem entsprechenden Clientbrowser wurde sich für die Sprache php entschieden. Seiteninformationen werden mit Hilfe von Javascript manipuliert und Nutzerspezifisch angezeigt.

9. Datenbankkommunikation

9.1 Aufbau der Datenbankverbindung

Die Datenbankverbindung wird durch eine Klasse namens DatabaseConnection realisiert. Diese besteht aus einem privaten Attribut, Konstruktor, Destruktor und einem Getter zum Holen der Datenbankverbindung. Die Funktion mysqli() ermöglicht das Setzen der verbindungsspezifischen Parameter.

```
public function __construct()
{
   $this->connection = new mysqli("IP/Domain", "Benutzername", "Passwort", "Datenbank");
   if($this->connection->connect_error)
   {
      die("Connection failed: " .  $this->connection->connect_error);
   }
}
```

Aufbau des Datenbank Konstruktors

$connection ist das private Attribut was nur über den Konstruktor gesetzt werden kann. Der Getter getConnection() gibt die aktuelle Verbindung zurück.

9.2 Datenabfrage und -eingabe mit php mysqli

Zur Vorbeugung von SQL Injections wurde sich für Prepared Statements entschieden, welche einen höheren Sicherheitsgrad als ungeschützte SQL Abfragen gewährleisten. Je nach Bedarf können Daten nur verändert werden oder entsprechende Rückgabedaten erhalten werden.

```php
function beispielfunktion($parameter){
   // Aufbau der Datenverbindung
     $conn = new DatabaseConnection;
     // entsprechendes SQL Statement
   $sql = "SELECT *
            FROM tabelle
            WHERE spaltenattribut=?;";
   // Aufbau der Datenbankverbindung und vorbereiten des SQL Statements
   $stmt = $conn→getConnection()→prepare($sql);

      // hier Bindung eines Parameters von Typ Integer
   $stmt->bind_param('i', $parameter);
   $stmt->execute();

     //Falls Daten zurückgegeben werden müssen
   $res = $stmt->get_result();
     // Aufbereitung der Daten als bennantes Array
   $row =$res->fetch_assoc();
     // alternativ fetch_all() für durchnummerierte Arrayrückgabe

   // Schließen der Datenverbidung und Rückgabe des Ergebnisses
   $stmt->close();
   return $row;
}
```

Beispiel Datenabruf durch PHP aus der Datenbank

Die Prepared Statements bestehen dabei aus der Vorbereitung prepared() und dem Binden eines Parameters des entsprechenden Typs. Weicht der Datentyp dabei ab, dann wird der SQL Befehl nicht ausgeführt.

Eine dynamische Erzeugung des ersten Parameters vom Typ String bei der Funktion bind_param() wurde aus nicht kalkulierbaren Sicherheitsbedenken unterlassen. Prinzipiell wäre ein Testen des übergebenen Datentyps möglich gewesen und eine Generierung des benötigten Strings realisierbar.

9.3 Unterteilung in Autor, Nutzer und Nutzer-Spieler Abfragen

Um die Übersichtlichkeit der einzelnen Datenbankabfragen zu erhöhen, wurde eine Gruppierung der Abfragen in SQL für Autoren, Nutzer und Nutzer-Spieler Abfragen vorgenommen. Hierbei inkludieren Autoren und Nutzer jeweils die Datenbankverbindung und, wegen der engen Verbindung zwischen den Nutzer und Spielerdaten, Nutzer-Spieler die Nutzer SQL Abfragen.

9.4 Umleitung einer Seite durch Javascipt statt PHP

Statt der kürzeren PHP Funktion header() wurde bei komplexeren PHP Seiten auf eine Javascript Variante gewechselt. Die erwähnte PHP Funktionen kann durch Erweiterung des PHP Codes zu unerwartetem Fehlverhalten führen und ist anfällig für nicht mehr funktionstüchtige Weiterleitungen[1]. Eine entsprechende Javascript Variante erwies sich als zuverlässiger.

```
//Der Funktion kann die zur Weiterleitung benötige url übergeben werden
function reload(url){
//Der zweite Parameter der anonymen Funktion gibt die Wartezeit in        Millisekunden
für die Umleitung an
   setTimeout(function() {
       window.location=url;
   }, 2000);
}
```

Ausgelagerte Umleitung in reload.js

[1] http://stackoverflow.com/questions/8028957/how-to-fix-headers-already-sent-error-in-php

10. Austausch zwischen PHP und Javascript - Variablen durch JSON Kodierung

Ein Zusammenspiel zwischen Javascript und PHP Variablen wird mit Hilfe der JSON Codierung ermöglicht. Beide Scriptsprachen besitzen entsprechende Decodierungs- und Encodierungsfunktionen.

```
<script>
<php?
//erzeugen des entsprechenden PHP Arrays
$erg = zugriffDatenbank($parameter);
//Kodieren des Arrays im JSON Format
$json_erg = json_encode($erg);
//Mit Hilfe von echo erzeugen der entsprechenden Javascript Variable
echo "var js_array = ". $json_erg . ";\n";
?>
</script>
```

Erzeugen eines JSON kodierten Arrays in PHP und speichern als Javascript Variabel

Wichtig ist dabei, dass die Umwandlung innerhalb eines Script Tags erfolgt. Innerhalb des HTML Bodys würde ansonsten eine Ausgabe durch den echo Befehl entstehen.

11. Nutzerbezogene Funktionalitäten

11.1 Registrierung, Login und Logout

Die Registrieung, der Login und der Logout werden über PHP Sessions realisiert. Bei der Registrierung wird das gewählte Passwort auf identische Eingabe überprüft und beim Abspeichern in die Datenbank über die PHP Funktion passwod_hash() und BCRYPT verschlüsselt und mit Salt&Pepper automatisch versehen. Das bedeutet, dass gehashte Passwörte, welche im Klartext denselben Text haben, dennoch unterschiedliche Hashcodes aufweisen. Verpflichtend ist die Eingabe eines Nutzernamens und der Email Adresse bei der Registrierung.

Login und Logout werden über die PHP Sessions verwaltet. Über einen Datenbankzugriff wird die Information über die Rolle des Nutzers gesetzt.

11.2 Personalisierbare Seiten des Nutzers

Die vom Nutzer veränderbaren Seiten unterteilen sich in zwei Bereiche. Zum einen in Informationen, die der Nutzer preisgeben möchte und in Nutzerdaten, die zum Beispiel das Angeben einer Email oder ändern des Passwortes für den Nutzer ermöglichen. Die Daten werden mit Hilfe eines Datenbankzugriffs geholt und durch einen onload Event Handler von Javascript mit diesem Array die Anzeige der Nutzerseiten personalisiert. Dabei ist die komplette Seite mit allen Feldern bereits ausprogrammiert und der entsprechende Inhalt wird direkt geändert.

```
document.getElementById("Feldname").innerHTML = js_array[i][j];
```

Direkte Änderung eines HTML Tags mit dem entsprechenden Feldnamen durch Javascript

11.3 Änderung der Nutzerangaben

Wie beim Login arbeiten hier eine HTML Seite mit den entsprechenden Feldern zum Ändern der Daten, eine PHP Seite zur kurzen Nutzerinformation über Erfolg oder Misserfolg der Datenänderung und dort der Aufruf der entsprechenden SQL Funktion aus einer inkludierter PHP Datei mit dem zu verwendenen Datenbankzugriff zusammen.

12. Verwaltung der Turnier- und Spielerinformationen durch einen Autor

Zur Eingabe der Turniere und Spielerergebnisse stehen 4 Eingabemasken zur Verfügung, die an die entsprechenden Tabellen des Datenbankmodells angelehnt sind.

Für die Spielereingabe und Aktualisierung stehen 3 Eingabemasken zur Verfügung.

Bei allen Masken wurde darauf geachtet, dass der Autor zum Ende seiner Änderungen anklicken muss, dass er sämtliche Daten kontrolliert hat und für korrekt befindet. Dieses dient zusätzlich dazu, dass versehentliches Drücken der Enter Taste nicht zu fehlerhaften Eingaben führt, die nur schwer zu korrigieren sind

12.1 Saisoneingabe

Besteht aus nur einem Eingabefeld. Zur erleichterten Eingabe wird automatisch aus der Datenbank nach der höchsten Jahreszahl gesucht. Kleinere Eingaben werden durch Manipulation des min Attributs des Eingabefeldes verboten und der Wert des Feldes automatisch um 1 inkrementiert gesetzt.

```
function changeYear() {
        <?php
            $erg = getMaxYear();
            $json_erg = json_encode($erg);
            echo "var js_array = ". $json_erg . ";\n";
        ?>

        document.getElementById('year').value=js_array[0][0]+1;
        document.getElementById('year').min=js_array[0][0]+1;
    }
```

Beispiel changeYear()

12.2 Turniereingabe

Zur Erweiterung der Turnierdaten steht eine Turniereingabe zur Verfügung. Um Tippfehler zu vermeiden wird zunächst ein Datenbankzugriff durchgeführt und die bereits eingepflegten Turniertypen auszulesen und mit Hilfe dieser Daten ein Select

Tag zu erzeugen in dem die entsprechenden Turniertypen ausgewählt werden können. Die entsprechende Turniertyp ID wird als Wert automatisch gesetzt.

```
function changeTurnierTyp(){
  <?php
  $erg = getTurnierTyp();
  $json_erg = json_encode($erg);
  echo "var js_array = ". $json_erg . ";\n";
  ?>
  var result="";
  for (var i = 0; i< js_array.length; i++) {
     result+="<option value=\""+js_array[i][0]+"\">" + js_array[i][1] + "</option>";
  }
  document.getElementById("inputTTyp").innerHTML= result;
}
```

Beispiel anhand changeTurnierTyp() zum Auslesen von in der Datenbank abgespeicherten Informationen und erzeugen der neuen HTML Elemente

12.3 Turnierergebnisse in Saison eingeben

Um Ergebnisse für ein Turnier eingeben zu können, muss dieses zunächst in die Saison aufgenommen werden. Wie bei changeTurnierTyp() werden Saison und Turniere automatisch aus der Datenbank gelesen und entsprechend erzeugt. Zur erleichterten Datumseingabe und unter der Annahme, dass Turniere zeitnah eingetragen werden, wird mit Hilfe von Javascript Funktionen das aktuelle Datum in Input Tags vom Typ number erzeugt. Bei dem Datenbankzugriff werden diese automatisch zusammen gesetzt.

```
function currentDate(){
  var array=[['dayStart','monthStart','yearStart'],["dayEnd","monthEnd","yearEnd"]];
  var date = new Date();
  for (var item1 in array){
     document.getElementById(array[item1][0]).value=date.getDate();
     document.getElementById(array[item1][1]).value=date.getMonth()+1;
     document.getElementById(array[item1][2]).value= date.getFullYear();
  }
}
```

Erzeugung des aktuellen Datums

```
$turnierStart=$_POST['yearStart']."-".$_POST['monthStart']."-".$_POST['dayStart'];
$turnierEnd=$_POST['yearEnd']."-".$_POST['monthEnd']."-".$_POST['dayEnd'];
```

Zusammensetzen des Datumsstrings

12.4 Ergebniseingabe nach einem Turnier

Zunächst wird mit Hilfe eines Datenbankzugriffs abgespeichert welche Saisondaten,Turniere und Spieler verfügbar sind. Mit Hilfe dieses Arrays werden entsprechende Select Tags erzeugt um die Gefahr von Tippfehlern vorzubeugen. Die Vorgehensweise ist dabei Identisch wie bei der Beispielfunktion changeTurnierTyp().

Um das Ausprogrammieren der einzelnen Form Elemente für die Platzeingabe zu vermeiden, kann die Anzahl der Plätze dynamisch erzeugt werden. Die Funktion reagiert auf einen onchange Event Handler.

```
function changePlatz(val){
  //Assoziatives Array zur Darstellung der Namen der entsprechenden Runde
  var turnierebene = ["Halbfinale",
                "Viertelfinale",
                "Achtelfinale",
                "Runde der letzen 16",
                "Runde der letzen 32",
                "Runde der letzen 64",
                "Runde der letzen 128",
                "Runde der letzen 256"];

//Festes Anlegen des 1. Platzes
     var  result  =  "<b>Finale:</b><br>Platz  1:  <br><select  name='platz[1][0]'>"  +
spielerListe +"</select><br>";
  anzahl = parseInt(val);

// Ausnutzen einer 2er Potenzabhängigkeit zur komfortablen Erzeugung der Platzanzahl für
die entsprechende Runde
  for (var i = 0; i< anzahl-1; i++){
     if(i>0 && i<turnierebene.length)
        result += "<b>"+(turnierebene[i])+":</b><br>";
     else
        result += "Platz " + (i+2) + ": <br>";
     for(var j=0; j< Math.pow(2,i) ; j++){
                result  +=  "<select  name='platz["+(i+2)+"]["+j+"]'>"  +  spielerListe
+"</select><br>";
     }
  }
  document.getElementById("reihen").innerHTML = result;
}
```

Funktion zur dynamischen Erzeugung der Platzeingabe

Um nicht für jeden einzelnen Platz einen erneuten Datenbankzugriff zu haben, wurde einmal beim Laden der Seite das Ergebnis der Spieler in der Datenbank abgespeichert und als Variable in der changePlatz() Funktion verwendet.

```
var spielerListe = changeSpieler();
```

Abspeichern der Spielernamen

12.5 Platzeingabe der Spieler

Zunächst werden die aktuellen Plätze der Spieler aus der Datenbank gelesen. Bei den Input Tags wird das placeholder Attribut mit den Plätzen der Spieler geändert, was zu einer blass grauen Anzeige führt. Werden Werte verändert, dann sind diese farblich schwarz dargestellt und so leichter von nicht geänderten Werten zu unterscheiden. Beim Eintragen der neuen Werte wird mit Hilfe von zwei for Schleifen überprüft, ob kein Rang doppelt vorhanden ist.

```
$test=false;
//Anzahl der geänderten Ränge wird bestimmt
for($i=0;$i<count($arrayRang); $i++) {
  for ($j = ($i+1); $j < count($arrayRang); $j++)
//Überprüfung ob kein Rang doppelt vorliegt
     if ($arrayRang[$i][0] == $arrayRang[$j][0]) {
                        echo   "<h3>FEHLER:   Zwei   Spieler   haben   denselben   Rang:
".$arrayRang[$i][0]."</h3>";
        $test=true;
        break;
     }
  if ($test)
     break;
}
```

Rangtest ob doppelte Rangplätze bei der Eingabe vorliegen

12.6 Spieler anlegen

Die wichtigsten Basisdaten müssen hier bei einem Spieler angegeben werden und sind mit Hilfe von dem Schlüsselwort require gegen fehlende Eingaben gesichert.

Automatisch wird aus der Datenbank die höchste Spieler ID ermittelt. Diese wird bei der Eingabe über ein verstecktes Feld mitgesendet.
Ein Select Feld wird für die Tabellenspalte welche Spielhand ein Spieler nutzt erzeugt. Die Pflichtdaten werden über ein SQL Insert eingefügt. Die zusätzlichen Daten werden über ein SQL Update ergänzt.

12.7 Spielerdaten aktualisieren

Zusätzlich zu den Feldern auf der Seite Spieler Anlegen wird zunächst ein Select Element generiert um eine Liste aller auswählbaren Spieler anzuzeigen.
Mit Hilfe des Programmierpattern AJAX wurde das Abrufen der Spielerdaten beim Ändern des Spielers realisiert.

```
document.getElementById('spielerID').onchange=function(){
   var spielerID = document.getElementById('spielerID').value;

   var js_array;
   var jason_array;

   var xmlhttp = new XMLHttpRequest();
   xmlhttp.onreadystatechange = function() {
       if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
           jason_array=xmlhttp.responseText;
           js_array = JSON.parse(jason_array);

           //Exemparische Änderung des placeholder Attributes eines Feldes
           document.getElementById("IDdesFeldes").placeholder =
           js_array['zugehoerigesAttribut'];
       }
   };

   xmlhttp.open("GET", "php/spielerData.php?spielerID="+spielerID, true);
   xmlhttp.send();
}
```

Holen der Spielerdaten durch AJAX und Ändern der Spielerdaten durch Setzen des placeholder Attributes

```
<?php

require 'sqlQueries/sqlQueryAutor.php';

$spielerID = $_REQUEST["spielerID"];
$erg = getSpielerData($spielerID);

$json_erg = json_encode($erg);
```

```
echo $json_erg;

?>
```

Zugehörige PHP Datei

Wie beim Aktualisieren der Plätze der Spieler ist es möglich modifizierte Daten von nicht modifizierten Daten anhand der Farbe zu unterscheiden.
Sämtliche Datenbankänderungen werden als SQL Update durchgeführt.

13. Verwaltung der Nutzer durch einen Autor

Zunächst wird beim Laden ein Select Element generiert, welches die vorhandenen Nutzer enthält. Als Wert wird die Nutzer ID zugewiesen. Bei der Auswahl eines Nutzers wird ein onchange Event Handler ausgelöst und ändert so die Felder Email und Nutzer ID. Um sicher zu gehen, dass nicht zufällig die ID oder die Email geändert werden, sind diese Input Felder mit dem Schlüsselwort readonly gekennzeichnet. Zum setzen neuer Werte wird dieses Attribut auf false und nach dem Ändern wieder auf true gesetzt

```
document.getElementById("nutzerID").readonly = false;
document.getElementById("nutzerID").value = '';
document.getElementById("nutzerID").readonly = true;
```

Beispiel Löschen der Nutzer ID durch ändern des readonly Wertes

Wird angeklickt, dass ein neues Passwort gesetzt werden soll, dann wird folgender Code zur Generierung eines Passwortes verwendet.

```
if(document.getElementById('passwortNeu').checked) {
    var length = 8,
    charset = "abcdefghijklnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789",
    passgen = "";
    for (var i = 0, n = charset.length; i < length; ++i) {
        passgen += charset.charAt(Math.floor(Math.random() * n));
}
```

Passwortgenerierung

Zusätzlich kann die Rolle des Nutzers geändert werden, der Nutzer gelöscht werden und, bei Bedarf, die E-Mail in eine Tabelle eingetragen werden um sie als Unerwünscht für eine erneute Registrierung zu kennzeichnen.
Dabei wurde darauf geachtet, dass ein Nutzer nicht ein neues Passwort bekommt, wenn er gelöscht wird und ein gebannter Nutzer auch automatisch als gelöscht markiert wird. Werden Unstimmigkeiten bei der Eingabe registriert, dann wurden individuelle Fehlermeldungen für den Fehlerfall geschrieben und durch das oninvalid Attribut in dem zugehörigen Form Element gesetzt.

```
oninvalid="this.setCustomValidity('Bitte bestätigen Sie, dass alle Angaben richtig sind');"
```

Beispiel Änderung der Fehlernachricht

Wurde der Fehlerfall bei einem Eingabefeld ausgelöst, muss anschließend ein setzen eines Leerstrings bei der setCustomValidity() Funktion folgen. Wird dieses versäumt, dann ist eine Eingabe korrekter Daten nicht möglich, da immer die Warnmeldung erscheint.

```
document.getElementById("IDdesFeldes").setCustomValidity('');
```

Beim Löschen des Nutzers wird zunächst überprüft wie viele gelöschte Nutzer bereits in der Datenbank vorhanden sind. Diese Zahl wird dem String “geloescht” angehängt und sämtliche Nutzerdaten werden genullt. Um sicher zu gehen, dass ein gelöschter Nutzer nicht seinen Namen heraus findet und sich mit dem alten Passwort anmeldet, wird ein neues Passwort mit der oben erwähnten Passwortgenerieung erzeugt und in der Datenbank gespeichert.

14. Ranglisten anzeigen

14.1 Turnier Ranglisten Spieler Anzeigen

Beim Aufruf der Seite werden zunächst 2 Objekte generiert: ein leeres seasons JSON Objekt und eine Variable, die das aktuelle Datum enthält.

```
var currentTime = new Date();
var seasons = {};
```

Danach wird beim Laden ein JSON Objekt generiert das alle vorhandenen Turniere und deren Saisons sowie die beteiligten Spieler beinhaltet. Dieses JSON Objekt wird dann nach `saisonID` und `turnierID` sortiert und in seasons gespeichert.

```
 for (i in saisons) {
        if (!seasons[saisons[i][7]]) {
                seasons[saisons[i][7]] = {};
        }
        if (!seasons[saisons[i][7]][saisons[i][6]]) {
                seasons[saisons[i][7]][saisons[i][6]] = [];
        }
        seasons[saisons[i][7]][saisons[i][6]].push(saisons[i]);
}
```

Nachdem dies geschehen ist, werden anhand der Saison Optionen für ein Select Element generiert. Diese werden sofort in dem Select Element `selSeason` dargestellt. Danach wird die `displayTournaments` Funktion aufgerufen die anhand einer bestimmten Saison passende Turnier Optionen für ein zweites Select `selTurnier` Element generiert.

```
function displayTournaments(s) {
        var tournaments="";
        for (j in seasons[s]){
        tournaments+="<option value=\""+j+"\">" + seasons[s][j][0][5] + "</option>";
```

```
        }
        document.getElementById("selTurnier").innerHTML=tournaments;
}
```

Die letzte Funktion, die aufgerufen wird, ist die `displayPlayer` Funktion, die anhand einer Saison und eines Turniers die beteiligten Spieler, ihren Platz und die erzielten Punkte darstellt.

```
function displayPlayer(s,t){
        var spieler="";
        for (p in seasons[s][t]){
                spieler+="<div  class=\"col-md-10 col-sm-10 col-xs-10\">";
                spieler +="<div class=\"row\" id=\"platz" + seasons[s][t][p][1] + "\"><p>Platz "
                + seasons[s][t][p][1] + "  " + seasons[s][t][p][3] + " " + seasons[s][t][p][4]
                +  " Erzielte Punkte " + seasons[s][t][p][2] +  "</p>";
                spieler += "</div>"
                spieler +="</div>";
        }
        document.getElementById("content").innerHTML=spieler;
}
```

Wenn im Select `selSeason` eine andere Saison ausgewählt wird, wird erneut die Funktion `displayTournaments` aufgerufen und danach die Funktion `displayPlayer`. Wenn im Select `selTurnier` ein anderes Turnier ausgewählt wird, wird nur die Funktion `displayPlayer` aufgerufen.

14.2 Turnier Ranglisten Nutzer Anzeigen

Die Darstellung der Nutzer Rangliste für einzelne Turniere erfolgt analog zur Darstellung der Spieler Rangliste.

14.3 Gesamt Ranglisten Nutzer Anzeigen

Beim Aufruf der Seite wird geprüft, ob ein `GET` Argument gesetzt ist. Da die aktuelle Rangliste und die vorherigen Ranglisten in unterschiedlichen Tabellen gespeichert werden, ist dies notwendig für die Umsetzung. Nach der Prüfung wird, wenn ein Argument vom Typ `saison` gesetzt ist, die passende Rangliste mit dieser

`saisonID` aus der Datenbank geladen und als JSON Objekt bereitgestellt. Sollte kein Argument vorhanden sein, wird die aktuelle Rangliste aus der Datenbank geladen und als JSON Objekt bereitgestellt.

```
if(isset($_GET["saison"])){
        $gg = $_GET["saison"];
        $erg = getRangliste($gg);
}
else{
        $erg=getRangliste("aktuelle");
        $gg=0;
}
```

Nachdem die Webseite geladen wurde, werden Option Elemente, anhand einer Abfrage ob dieses Turnier sich im aktuellen Jahr befindet, für ein Select Element generiert und dort eingefügt.

```
for (var i = saisons.length - 1; i >= 0; i--) {
        if(saisons[i][1]==currentTime.getFullYear()){
                alleSaisons+= "<option value=\"rangliste.html\">" + saisons[i][1] + "</option>";
        }
        else{
                alleSaisons+= "<option value=\"rangliste.html?saison="+saisons[i][0]+"\">"
                + saisons[i][1] + "</option>";
}
```

Danach wird anhand der `saisonID` die passendene Nutzer Rangliste angezeigt.

Wenn im Select Element eine andere Saison ausgewählt wird, wird die Webseite erneut geladen anhand der vorher eingefügten Values des Selects. So werden die `saisonID`s übergeben.