Student: Lars Schneider

Email: [Lars.schneider@fh-erfurt.de](mailto:Lars.schneider@fh-erfurt.de)

Matrikelnummer: 10245459

Kurs: Praxisprojekt

im Sommersemester 2016

CPs: 4

Dozent: Dr. Steffen Avemarg

Abgabe am: 07.08.2016

Referee Availablity Planing - Dokumentation

Diese App ermöglicht es die Planung von Turnier Schiedsrichter Einsätzen zu planen und gibt den planenden Supervisern die Auskunft, wie ein die Schiedsrichter anreisen können und werden.  
Die zweite Funktion dieser App ist es, allen Schiedsrichtern die Möglichkeit zu geben, sich miteinander auszutauschen und Emails zu versenden.

Dokumentation Praxisprojekt

vollPro bReff

**Inhalt**

[1 Einleitung 1](#_Toc453002679)

[2 Model 1](#_Toc453002680)

[2.1 Datenbank 1](#_Toc453002681)

[2.2 Code Style 1](#_Toc453002682)

[2.3 Tabellendokumentation 2](#_Toc453002683)

[3 Server 3](#_Toc453002684)

[3.1 Server-Funktionen 3](#_Toc453002685)

[3.1.1 Datenimport 3](#_Toc453002686)

[3.1.2 Mailversand 3](#_Toc453002687)

[3.1.3 Login-Routine 3](#_Toc453002688)

[3.2 Java Server Pages 3](#_Toc453002689)

[3.2.1 Registrierung 3](#_Toc453002690)

[3.2.2 Einloggen 3](#_Toc453002691)

[3.2.3 Datenimport 3](#_Toc453002692)

[4 Schnittstellen 3](#_Toc453002693)

[4.1 Einfügen von Daten (Inserts) 3](#_Toc453002694)

[4.2 Einfügen von Daten (Updates) 3](#_Toc453002695)

[4.3 Löschen von Daten (Deletes) 3](#_Toc453002696)

# **Einleitung**

Das Projekt vollPro bReff ist aufgeteilt in die serverseitige Datenverarbeitung im Bachelor-Praxisprojekt sowie dem Erstellen einer Android-App im Wahlpflicht Fach Programmieren Mobiler Endgeräte. In dieser Dokumentation geht es nur um das Praxisprojekt. Im ersten Abschnitt wird das Datenmodell mit seinen Tabellen, der Schnittstelle zum Server über Views, sowie das Versionieren der Daten umfänglich erläutert. Im zweiten Teil geht es darum, den Server mit seinen Funktionen Datenimport und Mailversand anschaulich zu erklären. Dies alles mit dem Hintergrund, die Daten in einer App laden und sehen zu können.

# **Model**

## **Datenbank**

Als Datenbank wurde eine einfach MySQL Datenbank gewählt, in dem ein einziges Schema ausreicht. Dieses Schema hat einen Umfang von 15 Tabellen die jeweils mit folgenden Spalten komplettiert werden:

* Erstellungsuser
* Erstellungsdatum
* Änderungsuser
* Änderungsdatum
* Gültigkeit von
* Gültigkeit bis

Hiermit wird eine Versionierung ermöglicht.

## **Code Style**

Die Tabellennamen sind alle auf Deutsch geschrieben. Jede Tabelle, die keine N zu N Beziehung auflöst, hat eine eindeutige OID (*O*bjekt *ID*entifikation) als Primary Key. In Tabellen, die eine N zu N Beziehung auflösen, sind die beiden OIDs der aufzulösenden Tabellen als Primary- und Foreign Keys da.

Fremdschlüssel-Spalten haben immer als Name den logischen Wert der Spalte (z.B. ERSTELLT\_VON), sowie den Anhang \_ID (Beispiel: ERSTELLT\_VON\_ID). Der Fremdschlüssel ist folgendermaßen deklariert:

*FK\_[Tabelle, in der der FK deklariert wird]\_[Tabelle, auf die der Foreignkey zeigt]*

## **Tabellendokumentation**

Wie schon erwähnt ist das Datenmodell mit einem Umfang von 15 Tabellen sehr übersichtlich.



Dem Modell wurde nur eine Tabelle entzogen, die User-Tabelle. Außerdem fehlen in allen Tabellen die Spalten ERSTELLT\_VON\_ID (als Fremdschlüssel auf den User, der diese Tabellenzeile erstellt hat), ERSTELLT\_AM, GEAENDERT\_VON\_ID (als Fremdschlüssel auf den User, der die Tabellenzeile als letzten geändert hat), GEAENDERT\_AM, GUELTIG\_VON und GUELTIG\_BIS.

# **Server**

Als Webserver dient ein Apache-Tomcat-8.0.33. Dieses Open Source Projekt ermöglicht einem über Java-Server-Pages und einem Adapter auf die jeweiligen

## **Server-Funktionen**

Neben der Hauptfunktion des Servers, den Datenaustausch zwischen Datenbank und APP zu gewährleisten, soll der Server außerdem den Mailversand und den Datenimport von Superusern über den Browser realisieren.

### **Datenimport**

Um Daten importieren zu können muss man sich zunächst als User mit der Berechtigung alle Daten einsehen zu können, anmelden. Danach wird einem ein Excel Datenimport ermöglicht. Dies sieht folgendermaßen aus:

### **Login-Routine**

Das Passwort soll sowohl für niemanden in Klartext erkennbar sein. Deshalb wird es vom User in Klartext eingegeben und danach in der Datenbank MD5-Verschlüsslt gespeichert. Beim Überprüfen, ob das Passwort bei der Eingabe korrekt ist, kann dann der eingegebene Klartext wieder MD5-Verschlüsslt mit dem MD5-Wert aus der Datenbank abgeglichen werden. Die Funktion „getMD5(String inputString)“ liefert den MD5-verschlüsslten Wert.

## **Java Server Pages**

Die in der Skriptsprache JSP erstellte Schnittstelle für den Import von Daten hat neben der eben genannten Funktion auch eine Einlog- und Registrierroutine mit sich gebracht. Außerdem können Userdaten eingesehen sowie gelöscht werden.

### **Registrierung**

Beim Registrieren werden die Daten per Post-Methode an den Server übergeben und überprüft. Kommt es zu z.B. falsch eingegebenen Daten im Datumsfeld, wird mittels eines Ausgabe-String Arrays auf die falschen Daten hingewiesen.

### **Einloggen**

Eingeloggt wird sich mit einem Username und Passwort. Die Login Daten werden Datenbankseitig mit der Session ID aus dem Browser gespeichert. Als eingeloggter User hat man die Möglichkeit größere Mengen an Userdaten hochzuladen, eine Liste von Personen zu sehen und einzelne Personen zu löschen.

### **Datenimport**

Das Importieren von Daten wurde im vorherigen Absatz schon genannt. Hier wird ein Ordnerpfad einer Post-Methode übergeben und anschließend werden diese Daten ausgelesen. Der angegebene Pfad muss einer CSV-Datei entsprechen und wird anschließend in die Datenbank geladen.