# Einleitung

**Auftraggeber:** Prof. DI Gernot OBERLERCHER

**Scrummaster:** OStR Prof. DI Heidrun MÜLLER-STEGMÜLLER

**Programmiertechnische Beratung:** Prof. Mag. Gerald ORTNER

**Beginn:** 15.März 2017

**Ende:** 14.Juni.2017

**Teammitglieder:**

* Moser Raphael
* Santner Elias
* Sonnberger Martin
* Wilscher Mario

**Auftrag:** Der Fußballclub Moskitos benötigt eine Android-App, die es ermöglicht Spielergebnisse festzuhalten.

Jedem Spieler steht ein Benutzerkonto zur Verfügung, wobei diese sich in normale Benutzer und Admins unterteilen. In diesem werden neben Benutzername und Passwort, für die Anmeldung in der App, auch die bevorzugte Position am Spielfeld gespeichert.

In einem Spiel wird festgehalten, wann es stattgefunden hat, welcher Spieler in welchem Team gespielt hat, sowie dessen Position am Spielfeld und das Endergebnis. Die Teameinteilung und Positionszuweisung erfolgt bei der Spielerstellung entweder manuell vom Spielersteller oder automatisch durch eine Funktion in der App.

Die Statistik wird automatisch aus den Spielergebnissen generiert. Die Statistik besteht aus folgenden Punkten:

* Anzahl der teilgenommenen Spiele, die gespielte Position und ob dieses mit einem Sieg, Niederlage oder mit einem Unentschieden geendet hat
* Durchschnittliche Tordifferenz
* Geschossene Tore (Gesamt)
* Normale Tore
* Kopfballtore
* Kopfballtore bei Schnee
* Elfmetertore
* Bekommene Tore
* Geschossene Gurken

**Funktionsumfang:** Der Funktionsumfang ist davon abhängig ob es sich um einen normalen Benutzer oder Admin handelt, da dem Admin zusätzlich zu den normalen Funktionen, die Verwaltung der Spieler und Spiele zu Verfügung steht.

**Benutzer:**

* Wunschpositionen festlegen
* Spielergebnisse ansehen
* Spieler Statistiken ansehen

**Admin:**

* Neue Benutzer anlegen
* Bestehende Benutzer bearbeiten
* Bestehende Benutzer löschen
* Neue Spiele anlegen
* Bestehende Spiele bearbeiten
* Bestehende Spiele löschen

# C:\Users\Raphael\AppData\Local\Microsoft\Windows\INetCacheContent.Word\0001.jpgStorymap

### Product Backlog

## User

Diese Userstories sind allgemein für jeden User.

### Userstory: Anmelden

Ziel dieser Userstory ist es, dass sich der User anmelden kann, um auf die ihm zur Verfügung stehenden Funktionen Zugriff zu erhalten.

#### Akzeptanzkriterium:

Durch Eingabe der richtigen Benutzername-Passwort-Kombination wird der der User angemeldet und erhält so Zugriff auf die weiteren Funktionen.

Testfälle:

1. **Vorbedingung:** User „Admin“ mit Passwort „Test“ existiert.  
   **Aktion:** User gibt „Admin“ als Benutzername und „Test“ als Passwort ein und klickt auf Anmelden.

**Ergebnis:** User erhält Zugriff auf die App.

1. **Aktion:** User gibt keinen Benutzernamen oder kein Passwort ein und klickt auf Anmelden.

**Ergebnis:** User erhält eine Meldung, dass er Benutzername und Passwort eingeben muss, um sich anmelden zu können.

1. **Vorbedingung:** User „Ad“ mit Passwort „Test“ existiert nicht. **Aktion:** User gibt „Ad“ als Benutzernamen und „Test“ als Passwort ein und klickt auf Anmelden.  
   **Ergebnis:** User erhält eine Meldung, dass Benutzername oder Passwort ungültig ist.
2. **Vorbedingung:** User „Admin“ hat nicht das Passwort „xxx“. **Aktion:** User gibt „Admin“ als Benutzernamen und „xxx“ als Passwort ein und klickt auf Anmelden.  
   **Ergebnis:** User erhält eine Meldung, dass Benutzername oder Passwort ungültig ist.

### Userstory: Benutzerdaten ändern

Ziel dieserUserstory ist es, dass der User seine Benutzerdaten ändern kann, damit sein Profil richtig festgehalten wird.

#### Akzeptanzkriterium:

* User kann seinen Namen ändern, solange zwischen 3 und 25 Zeichen ist.
* User kann seinen Benutzernamen ändern, solange dieser zwischen 3 und 20 Zeichen ist, von keinem anderen User genutzt wird und nur aus Zahlen und Buchstaben besteht.
* User kann sein Passwort ändern, solange dieses länger als 5 Zeichen ist.
* User kann seine Positionen auf denen er spielen möchte auswählen, wobei er mindestens eine Position ausgewählt haben muss.

Testfälle:

1. **Vorbedingung:** User ist eingeloggt und kein Spieler hat den Username „Test“.  
   **Aktion:** User gibt „Test“ als Benutzername ein und drückt auf Speichern.

**Ergebnis:** User bekommt Benachrichtigung, dass seine Benutzerdaten aktualisiert wurden.

1. **Vorbedingung:** User ist eingeloggt.  
   **Aktion:** User gibt „Test1“ als Passwort ein.

**Ergebnis:** User bekommt Benachrichtigung, dass seine Benutzerdaten aktualisiert wurden.

1. **Vorbedingung:** User ist eingeloggt und ein Spieler hat den Username „Admin“.  
   **Aktion:** User gibt „Admin“ als Benutzername ein und drückt auf Speichern.

**Ergebnis:** User bekommt Benachrichtigung, dass dieser Benutzername bereits in Nutzung ist.

1. **Vorbedingung:** User ist eingeloggt.  
   **Aktion:** User wählt keine Position auf der er spielen möchte aus und drückt auf Speichern.

**Ergebnis:** User bekommt Benachrichtigung, dass er mindestens eine Position auswählen muss.

1. **Vorbedingung:** User ist eingeloggt.  
   **Aktion:** User gibt „ad“ als Benutzername ein und drückt auf Speichern.

**Ergebnis:** User bekommt Benachrichtigung, dass sein Benutzername nicht kürzer als 3 Zeichen sein darf.

1. **Vorbedingung:** User ist eingeloggt.  
   **Aktion:** User gibt „ganzLangerBenutzerNameDerZiemlichLangIst“ als Name ein und drückt auf Speichern.

**Ergebnis:** User bekommt Benachrichtigung, dass sein Name nicht länger als 25 Zeichen sein darf.

1. **Vorbedingung:** User ist eingeloggt.  
   **Aktion:** User gibt „Test!?“ als Benutzername ein und drückt auf Speichern.

**Ergebnis:** User bekommt Benachrichtigung, dass sein Benutzername nur aus Buchstaben und Zahlen bestehen darf.

### Userstory: Statistiken ansehen

Ziel dieser Userstory ist es, dass der User Statistiken einsehen kann.

#### Akzeptanzkriterium:

* User kann Statistik eines Spielers einsehen.
* User kann Statistik eines Spiels einsehen.
* User kann sich auf dem Scoreboard mit anderen Spielern vergleichen.

Testfälle:

1. **Vorbedingung:** Es gibt mindestens einen Spieler.  
   **Aktion:** User wählt Spieler aus, dessen Statistik er ansehen will.

**Ergebnis:** User sieht Statistik des Spielers.

1. **Vorbedingung:** Es gibt mindestens einen Spieler.  
   **Aktion:** User öffnet das Scoreboard.

**Ergebnis:** User sieht die Spieler sortiert nach der Leistung.

1. **Vorbedingung:** Es gibt mindestens ein Spiel.  
   **Aktion:** User wählt Spiel aus, von dem er die Statistik sehen will.

**Ergebnis:** User sieht Statistik des Spiels.

## Admin

Diese Userstories sind nur für Admins zugänglich. Damit diese Userstories funktionieren besteht die Vorbedingung, dass der User Admin-Rechte hat.

### Userstory: Spieler anlegen

Ziel dieser Userstory ist es, dass ein Admin neue Benutzerkonten anlegen kann, damit auch neu hinzukommende Spieler die App benutzen können.

#### Akzeptanzkriterium:

* Admin kann einen Namen für den neuen User eintragen, solange dieser zwischen 25 und 3 Zeichen ist.
* Admin kann einen Benutzernamen für den neuen User eintragen, solange dieser zwischen 3 und 20 Zeichen ist, von keinem anderen User genutzt wird und nur aus Buchstaben und Zeichen besteht.
* Admin kann ein Passwort für den neuen User anlegen, solange dieses länger als 5 Zeichen ist.
* Admin kann dem neuen User Admin-Rechte erteilen.

Testfälle:

1. **Vorbedingung:** User ist eingeloggt mit Admin-Rechte.  
   **Aktion:** User gibt weder Name, Benutzername noch Passwort ein und drückt auf Speichern.

**Ergebnis:** User bekommt Benachrichtigung, dass er Name, Benutzername und Passwort eingeben muss.

1. **Vorbedingung:** User ist eingeloggt mit Admin-Rechte und ein Spieler hat bereits den Username „Test1“.  
   **Aktion:** User gibt „Test1“ als Benutzername, „Test“ als Name und „12345“ als Passwort ein und drückt auf Speichern.

**Ergebnis:** User bekommt Benachrichtigung, dass dieser Benutzername bereits vergeben ist.

1. **Vorbedingung:** User ist eingeloggt mit Admin-Rechte und kein Spieler hat den Username „Test1“.  
   **Aktion:** User gibt „Test1“ als Benutzername ein und drückt auf Speichern.

**Ergebnis:** User bekommt Benachrichtigung, dass ein neuer Spieler angelegt wurde.

### Userstory: Spieler bearbeiten

Ziel dieser Userstory ist es, dass ein Admin bestehende Benutzerkonten ändern kann, damit er den Spielern bei Problemen (z.B. Passwort vergessen) helfen kann.

#### Akzeptanzkriterium:

* User kann Namen des Spielers ändern, solange zwischen 3 und 25 Zeichen ist.
* User kann Benutzernamen eines Spielers ändern, solange dieser zwischen 3 und 20 Zeichen ist, von keinem anderen User genutzt wird und nur aus Zahlen und Buchstaben besteht.
* User kann Passwort eines Spielers ändern, solange dieses länger als 5 Zeichen ist.
* User kann die Positionen des Spielers ändern.

Testfälle:

Siehe Benutzerdaten ändern (**Vorbedingung:** User hat Admin-Rechte).

## Userstory: Spieler löschen

Ziel dieser Userstory ist es, dass ein Admin Benutzerkonten, welche nicht mehr benutzt werden, löschen kann, damit alten Spielern der Zugriff auf die App verweigert wird.

## Akzeptanzkriterium:

* User kann die Benutzerkonten von Spielern löschen.
* User kann sein Benutzerkonto nicht löschen.

Testfälle:

1. **Vorbedingung:** User ist eingeloggt und hat Admin-Rechte.  
   **Aktion:** User wählt bei einem Spieler die Option löschen aus.

**Ergebnis:** Spieler wird gelöscht (Spieldaten bleiben erhalten).

1. **Vorbedingung:** User ist eingeloggt und hat Admin-Rechte.  
   **Aktion:** User wählt bei sich selbst die Option löschen aus.

**Ergebnis:** User wird benachrichtigt, dass er sich selbst nicht löschen kann.

## Userstory: Spiel anlegen

Ziel dieser Userstory ist es, dass ein Admin Spiele anlegen kann, damit Spiele festgehalten werden können und deren Ergebnisse.

### Akzeptanzkriterium:

Admin kann Spiele anlegen, Spieler dem Spiel zuweisen bzw. in Teams und das Ergebnis eintragen.

Testfälle:

1. **Vorbedingung:** User ist eingeloggt und hat Admin-Rechte.  
   **Aktion:** User weist keine Spieler dem Spiel zu.

**Ergebnis:** User bekommt Benachrichtigung, dass er die Mindestanzahl an Spielern nicht erfüllt um ein Spiel zu erstellen.

1. **Vorbedingung:** User ist eingeloggt und hat Admin-Rechte und hat Spieler dem Spiel zugewiesen.  
   **Aktion:** User weist einem Spieler kein Team zu.

**Ergebnis:** User bekommt Benachrichtigung, dass jeder Spieler einem Team zugewiesen sein muss.

1. **Vorbedingung:** User ist eingeloggt und hat Admin-Rechte und hat jeden Spieler ein Team zugewiesen.  
   **Aktion:** User trägt Ergebnis des Spiels ein.

**Ergebnis:** User bekommt Benachrichtigung, dass die Daten gespeichert wurden.

### Userstory: Spiel ändern

Ziel dieser Userstory ist es, dass ein Admin Spiele im Nachhinein bearbeiten kann, damit er mögliche Fehler korrigieren kann.

#### Akzeptanzkriterium:

Admin kann das Ergebnis von Spielen ändern.

Testfälle:

1. **Vorbedingung:** User ist eingeloggt und hat Admin-Rechte und es gibt mindestens ein Spiel.  
   **Aktion:** User wählt Spiel und ändert das Ergebnis.

**Ergebnis:** User bekommt Benachrichtigung, dass die Änderung gespeichert wurde.

### Userstory: Spiel löschen

Ziel dieser Userstory ist es, dass ein Admin Spiele löschen kann, damit möglich falsch erstellte Spiele entfernt werden können.

#### Akzeptanzkriterium:

Admin kann Spiele löschen.

Testfälle:

1. **Vorbedingung:** User ist eingeloggt und hat Admin-Rechte und es gibt mindestens ein Spiel.  
   **Aktion:** User wählt Spiel und löscht es.

**Ergebnis:** User bekommt Benachrichtigung, dass das Spiel gelöscht wurde.

## Supporting Stories

### Story: Einstellungen

Ziel dieser Story ist es den Benutzern Einstellungsmöglichkeiten zur Verfügung stellen. Dazu gehören:

* Sprache
* Benachrichtigungsart
* Layoutauswahl

### Story: Design

Ziel dieser Story ist es ein Einheitliches Design für die App zu erstellen.

### Story: Menü

Ziel dieser Story ist es das die App über ein Menü verfügt, welches Benutzerfreundlich und Übersichtlich ist.

### Story: Offline

Ziel dieser Story ist es das die App auch mit einer inaktiven Internetverbindung funktioniert, ohne dabei wichtige Funktionsfähigkeiten zu verlieren.

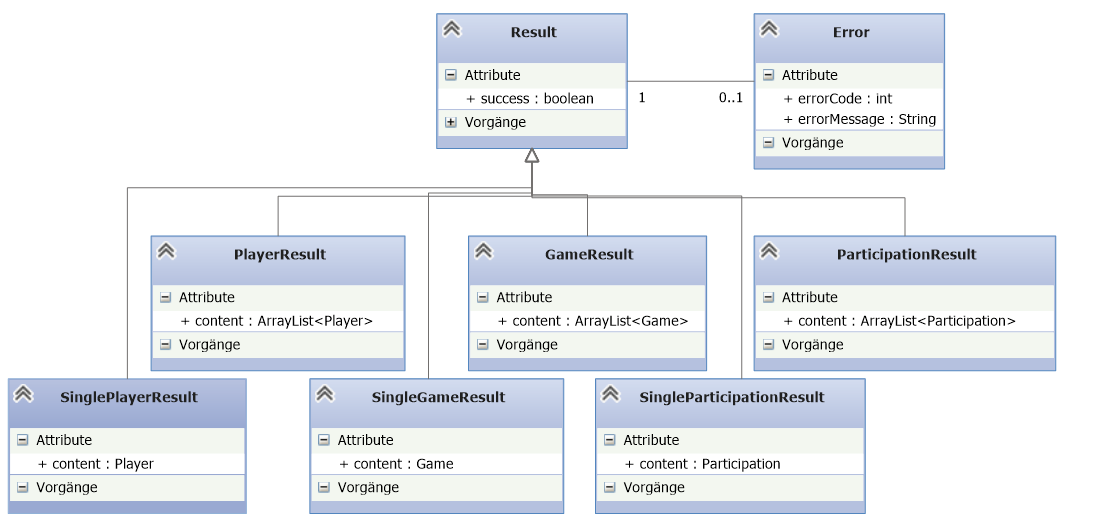
# C:\Users\Raphael\Documents\AndroidProjects\Soccer_app\doku\classDiagram.pngKlassendiagramm

# C:\Users\Raphael\Documents\AndroidProjects\Soccer_app\documents\db_erd.pngER-Diagramm

# Webservice

## General

### Data Model



|  |  |
| --- | --- |
| **Class** | **Description** |
| Error | Represents an error. Error Code is always 0, because there are no pre – defined errors in this API. |
| Result | Information to the result of a web service call. If it fails and success equals false, the error object shows the reason for the fail. If success equals true, there should be content in the response. The only reason for a succeeded call with an empty content object is that there wasn’t a single record matching the requests filter option. |

Es wird nur JSON supported.

## Methods

### GameResource

#### getAllGames

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| GET | url/game |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| **202** | { "type": "gameResult",  "success": true,  "content": [  {  "date": "2017-03-16",  "id": 1,  "remark": "Schneefall",  "scoreTeamA": 56,  "scoreTeamB": 0  },  {  "date": "2017-03-15 ",  "id": 2,  "remark": "Peter verletzt",  "scoreTeamA": 0,  "scoreTeamB": 0  },  {  "date": "2017-03-14 ",  "id": 3,  "remark": "",  "scoreTeamA": 0,  "scoreTeamB": 0  }  ]  } |
| **401** | {"type": "gameResult", "success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

### updateGame

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| PUT | url/game |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| Application/json | {  "date": "2017-03-16 ",  "id": 1,  "remark": "Schneefall",  "scoreTeamA": 56,  "scoreTeamB": 0  }, |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| **202** | {"success": true} |
| **202** | {"success": false} |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

Success true oder false je nachdem ob das Updaten erfolgreich war, oder nicht.

### getGamesByDate

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| GET | url/game/byDate/{date} |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| date | Path | Date for game | 2017-03-16 |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| **202** | {  "type": "gameResult",  "success": true,  "content": [ {  "date": "2017-03-16",  "id": 1,  "remark": "Schneefall",  "scoreTeamA": 56,  "scoreTeamB": 0  }]  } |
| **202** | {  "type": "gameResult",  "error": {  "errorCode": 0,  "errorMessage": "Unparseable date: \"fds\""  },  "success": false  } |
| **401** | {"type": "gameResult", "success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

### getGamesByPlayerId (and between 2 Dates)

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| GET | url/game/byPlayerId/{id} |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Param** | | **Type** | | **Description** | | **Example Value** |
| id | | Path | | Mandatory; Id for Player | | 1 |
| dateFrom | | Query | | Optional; Start Date | | 2017-03-15 |
| dateTo | | Query | | Optional; End Date | | 2017-03-18 |
| loginKey | Query | | Authentication Key | | 098f6bcd4621d373cade4e832627b4f6 | |

If no Date is specified, it will be ignored

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | {  "type": "gameResult",  "success": true,  "content": [ {  "date": "2017-03-16T00:00:00+01:00",  "id": 1,  "scoreTeamA": 56,  "scoreTeamB": 0  }]  } |
| **401** | {"type": "gameResult", "success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

### insertGame

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| POST | url/game |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| Application/json | {  "date": "2017-03-16",  "scoreTeamA": 56,  "scoreTeamB": 0,  "remark": 20  } |

Java.util.Date in Game was changed to Java.sql.Date, otherwise this wouldn’t be possible

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | {  "type": "singleGameResult",  "success": true,  "content": {  "date": "2017-03-16",  "id": 7,  "remark": "Schnee",  "scoreTeamA": 56,  "scoreTeamB": 0  }  } |
| **401** | {"type": "gameResult", "success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

### deleteGame

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| DELETE | url/game/{id} |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| id | Path | Id of Game that has to be deleted | 1 |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | {success: true} |
| 202 | {success: false} |
| **401** | {success: false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

Success true oder false je nachdem ob das Löschen erfolgreich war, oder nicht.

## PlayerResource

### getAllPlayers

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| GET | url/player |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| **202** | {  "type": "playerResult",  "success": true,  "content": [  {  "admin": true,  "id": 1,  "name": "Admin\uD83E\uDD18",  "positions": [  "ATTACK",  "GOAL",  "MIDFIELD"  ],  "statistics": {  "avgGoalDifference": 0,  "numDefeats": 0,  "numDraws": 1,  "numGamesPlayed": 1,  "numGoalsGot": 0,  "numGoalsShot": 0,  "numGoalsShotHead": 0,  "numGoalsShotHeadSnow": 0,  "numGoalsShotPenalty": 0,  "numNutmeg": 0,  "numPosAttack": 0,  "numPosDefense": 0,  "numPosGoal": 1,  "numPosMidfield": 0,  "numWins": 0  },  "username": "admin"  },  {  "admin": true,  "id": 2,  "name": "\u2584\uFE3B\u0337\u033F\u253B\u033F\u2550\u2501\u4E00 Moser",  "positions": [  "ATTACK",  "MIDFIELD"  ],  "statistics": {  "avgGoalDifference": -0.5,  "numDefeats": 1,  "numDraws": 1,  "numGamesPlayed": 2,  "numGoalsGot": 0,  "numGoalsShot": 0,  "numGoalsShotHead": 1,  "numGoalsShotHeadSnow": 0,  "numGoalsShotPenalty": 0,  "numNutmeg": 0,  "numPosAttack": 0,  "numPosDefense": 0,  "numPosGoal": 0,  "numPosMidfield": 2,  "numWins": 0  },  "username": "moe"  }]  } |
| **401** | {"type": "playerResult", "success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

### updatePlayer

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| PUT | url/player |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| Application/json | {  "admin": false,  "id": 1,  "name": "Martinii",  "username": "martin"  } |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| **202** | {"success": true} |
| **202** | {"success": false} |
| **401** | {"type": "singlePlayerResult", "success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

Success true oder false je nachdem ob das Updaten erfolgreich war, oder nicht.

### setPositions

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| PUT | url/player/positions/{playerid} |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| playerid | Query | Id of player to set positions | 5 |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| Application/json | {  "GOAL": false,  "MIDFIELD": true,  "ATTACK": false,  "DEFENSE": true  } |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| **202** | {"success": true} |
| **202** | {"success": false} |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

Success true oder false je nachdem ob das Updaten erfolgreich war, oder nicht.

### getPlayerByUsername

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| GET | url/player/{username} |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| username | Path | Username of the player you want to get | admin |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| **202** | {  "type": "singlePlayerResult",  "success": true,  "content": {  "admin": true,  "id": 1,  "name": "Admin\uD83E\uDD18",  "positions": [  "ATTACK",  "GOAL",  "MIDFIELD"  ],  "statistics": {  "avgGoalDifference": 0,  "numDefeats": 0,  "numDraws": 1,  "numGamesPlayed": 1,  "numGoalsGot": 0,  "numGoalsShot": 0,  "numGoalsShotHead": 0,  "numGoalsShotHeadSnow": 0,  "numGoalsShotPenalty": 0,  "numNutmeg": 0,  "numPosAttack": 0,  "numPosDefense": 0,  "numPosGoal": 1,  "numPosMidfield": 0,  "numWins": 0  },  "username": "admin"  }  } |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

### insertPlayer

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| POST | url/game |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| Application/json | {  "admin": false,  "name": "jerome",  "username": "guina"  } |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | {  "type": "singlePlayerResult",  "success": true,  "content": {  "admin": false,  "goalDifference": 0,  "id": 11,  "name": "jerome",  "numDefeats": 0,  "numDraws": 0,  "numWins": 0,  "username": "guina"  }  } |
| 202 | {  "type": "singlePlayerResult",  "error": {  "errorCode": 0,  "errorMessage": "com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolationException: Duplicate entry 'guina' for key 'USERNAME'"  },  "success": false,  "content": {  "admin": false,  "goalDifference": 0,  "id": 11,  "name": "jerome",  "numDefeats": 0,  "numDraws": 0,  "numWins": 0,  "username": "guina"  }  } |
| **401** | {"type": "singlePlayerResult", "success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

The second case (MySQLIntegrityConstraintViolationException) happens when the username is not available anymore because it’s already used by another player. In this case the existing player is returned.

### deletePlayer

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| DELETE | url/player/{id} |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| id | Path | Id of Player that has to be deleted | 1 |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | {success: true} |
| 202 | {success: false} |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

Success true oder false je nachdem ob das Löschen erfolgreich war, oder nicht.

### login

|  |  |
| --- | --- |
| **Method** | **URL** |
| POST | url/player/security/login |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| Application/json | {  "username": "admin",  "password\_enc": "21232f297a57a5a743894a0e4a801fc3"  } |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | a6444a246547facadfbc56ed7a940b7f |
| 204 |  |
| **500** | {"error":"Something went wrong. Please try again later."} |

If LoginCredentials are valid, return a loginKey which is used for every further method call.

If they are invalid, nothing (NO\_CONTENT) is returned.

### setPassword

|  |  |
| --- | --- |
| **Method** | **URL** |
| PUT | url/player/security/{id} |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| id | Path | Id of the player you want to insert the password to | 1 |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| Application/json | {  "password\_enc": "21232f297a57a5a743894a0e4a801fc3"  } |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | {"success": true} |
| 202 | {"success": false} |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

Success true oder false je nachdem ob das Updaten erfolgreich war, oder nicht.

### setGeoLocation

|  |  |
| --- | --- |
| **Method** | **URL** |
| POST | url/player/geoloc/{id} |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| id | Path | Id of the player you want to set the geolocation | 1 |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| Application/json | {  "latitude": 46.601166,  "longitude": 13.843841  } |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | {"success": true} |
| 202 | {"success": false} |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

### getGeoLocation

|  |  |
| --- | --- |
| **Method** | **URL** |
| GET | url/player/geoloc/{id} |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| id | Path | Id of the player you want to set the geolocation | 1 |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | {  "type": "locationResult",  "success": true,  "content": {  "latitude": 46.617955,  "longitude": 13.848626  }  } |
| 202 | {"type": "locationResult", "success": false} |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

### getPlayersNearby

|  |  |
| --- | --- |
| **Method** | **URL** |
| GET | url/player/geoloc/nearby/{id} |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| id | Path | Id of the player you want to set the geolocation | 1 |
| radius | Query | Radius (in km) which is considered “nearby” | 2.5 |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | {  "type": "playerIDResult",  "success": true,  "content": [  1,  2,  3,  4  ]  } |
| 202 | {  "type": "playerIDResult",  "error": {  "errorCode": 0,  "errorMessage": "radius not set"  },  "success": false  } |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

ParticipationResource

### getAllParticipations

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| GET | url/participation |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| **202** | {  "type": "participationResult",  "success": true,  "content": [ {  "numGoalsGot": 0,  "numGoalsShotDefault": 0,  "numGoalsShotHead": 0,  "numGoalsShotHeadSnow": 0,  "numGoalsShotPenalty": 0,  "numNutmeg": 0,  "player": {  "admin": false,  "goalDifference": 56,  "id": 1,  "name": "Martinii",  "numDefeats": 0,  "numDraws": 0,  "numWins": 1,  "username": "martin"  },  "position": "ATTACK",  "team": "TEAM1"  }]  } |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

### getParticipationById

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| GET | url/participation/{id} |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| id | Path | Id of the participation you want to get | 5 |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| **202** | {  "type": "participationResult",  "success": true,  "content": [ {  "numGoalsGot": 0,  "numGoalsShotDefault": 0,  "numGoalsShotHead": 0,  "numGoalsShotHeadSnow": 0,  "numGoalsShotPenalty": 0,  "numNutmeg": 0,  "player": {  "admin": false,  "goalDifference": 56,  "id": 1,  "name": "Martinii",  "numDefeats": 0,  "numDraws": 0,  "numWins": 1,  "username": "martin"  },  "position": "ATTACK",  "team": "TEAM1"  }]  } |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

### getParticipationById

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| GET | url/participation/byGame/{id} |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| id | Path | Id of the game you want the participations for | 5 |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| **202** | {  "type": "participationResult",  "success": true,  "content": [ {  "numGoalsGot": 0,  "numGoalsShotDefault": 0,  "numGoalsShotHead": 0,  "numGoalsShotHeadSnow": 0,  "numGoalsShotPenalty": 0,  "numNutmeg": 0,  "player": {  "admin": false,  "goalDifference": 56,  "id": 1,  "name": "Martinii",  "numDefeats": 0,  "numDraws": 0,  "numWins": 1,  "username": "martin"  },  "position": "ATTACK",  "team": "TEAM1"  }]  } |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

### insertParticipation

|  |  |
| --- | --- |
| **Method** | **URL** |
| POST | url/participation |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| idGame | Query | Id of the game | 1 |
| idPlayer | Query | Id of the player | 2 |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| Application/json | {  "numGoalsGot": 0,  "numGoalsShotDefault": 0,  "numGoalsShotHead": 0,  "numGoalsShotHeadSnow": 0,  "numGoalsShotPenalty": 0,  "numNutmeg": 0,  "position": "ATTACK",  "team": "TEAM1"  } |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | {"success": true} |
| 202 | {"success": false} |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

Success true oder false je nachdem ob das Erstellen erfolgreich war, oder nicht. Es ist möglich, dass bei success false diverse SQL Errors mitgeschickt werden (z.B. Constraint verletzt o.Ä.)

### updateParticipation

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| PUT | url/participation |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Param** | | **Type** | | **Description** | | **Example Value** |
| idGame | Query | | Id of the game | | 1 | |
| idPlayer | Query | | Id of the player | | 2 | |
| loginKey | Query | | Authentication Key | | 098f6bcd4621d373cade4e832627b4f6 | |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| Application/json | {  "numGoalsGot": 0,  "numGoalsShotDefault": 0,  "numGoalsShotHead": 0,  "numGoalsShotHeadSnow": 0,  "numGoalsShotPenalty": 0,  "numNutmeg": 0,  "position": "ATTACK",  "team": "TEAM1"  } |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | {"success": true} |
| 202 | {"success": false} |
| **401** | {"success": false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

Success true oder false je nachdem ob das Löschen erfolgreich war, oder nicht.

### deleteParticipation

Request

|  |  |
| --- | --- |
| **Method** | **URL** |
| DELETE | url/participation/{id} |

|  |  |  |  |
| --- | --- | --- | --- |
| **Param** | **Type** | **Description** | **Example Value** |
| id | Path | Id of Participation that has to be deleted | 1 |
| loginKey | Query | Authentication Key | 098f6bcd4621d373cade4e832627b4f6 |

Response

|  |  |
| --- | --- |
| **Status** | **Response** |
| 202 | {success: true} |
| 202 | {success: false} |
| **500** | {"error":"Something went wrong. Please try again later."} |

Success true oder false je nachdem ob das Löschen erfolgreich war, oder nicht.

# Architektur – Programmierung

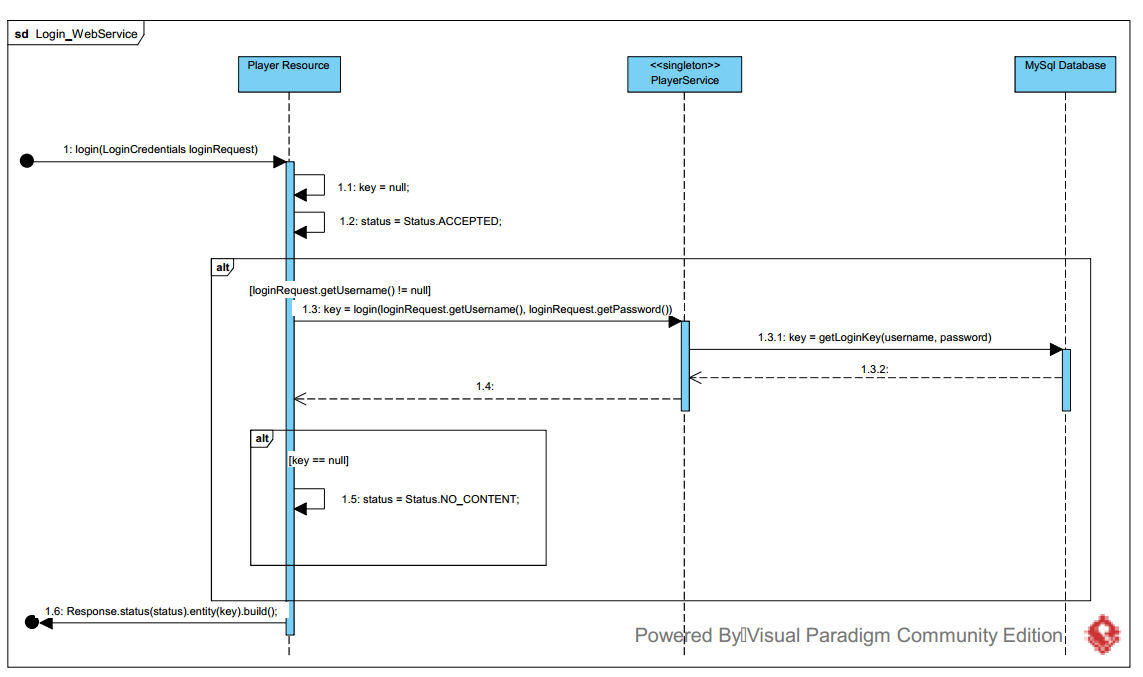
QR-Code Generator: <https://github.com/zxing/zxing>

QR-Code Scanner: <https://developers.google.com/vision/>

GSON: <https://github.com/google/gson>

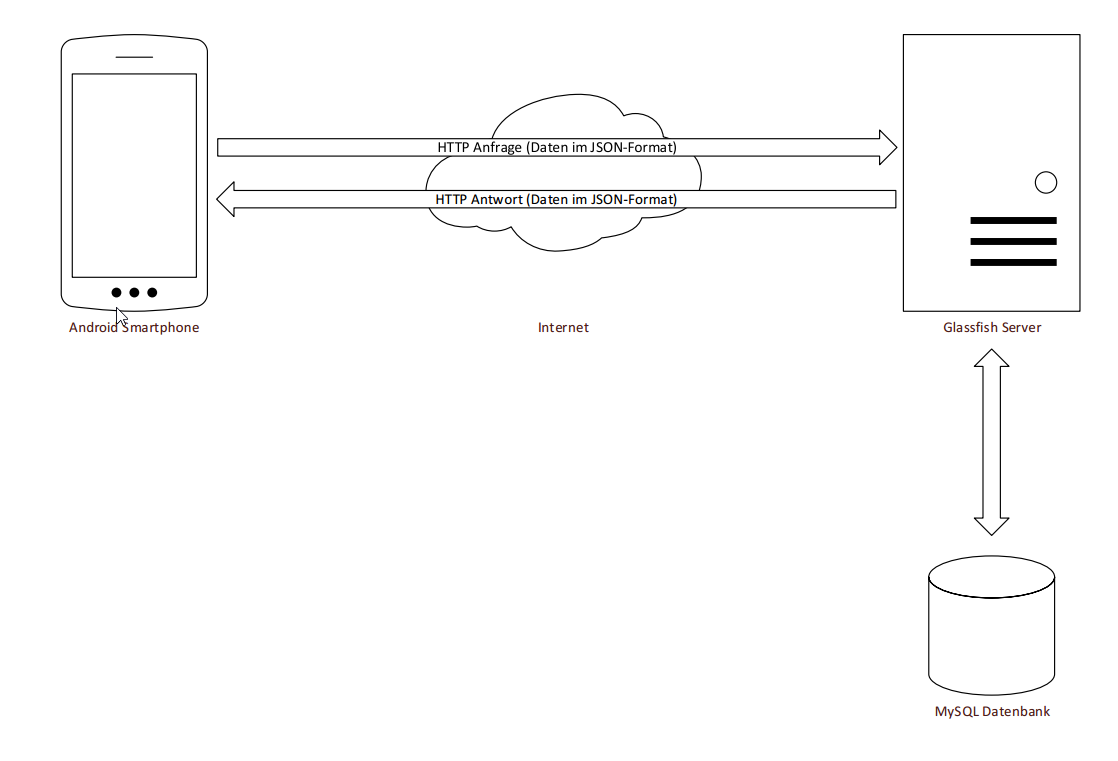
Apache Common Lang: https://commons.apache.org/proper/commons-lang/

## Login Webservice

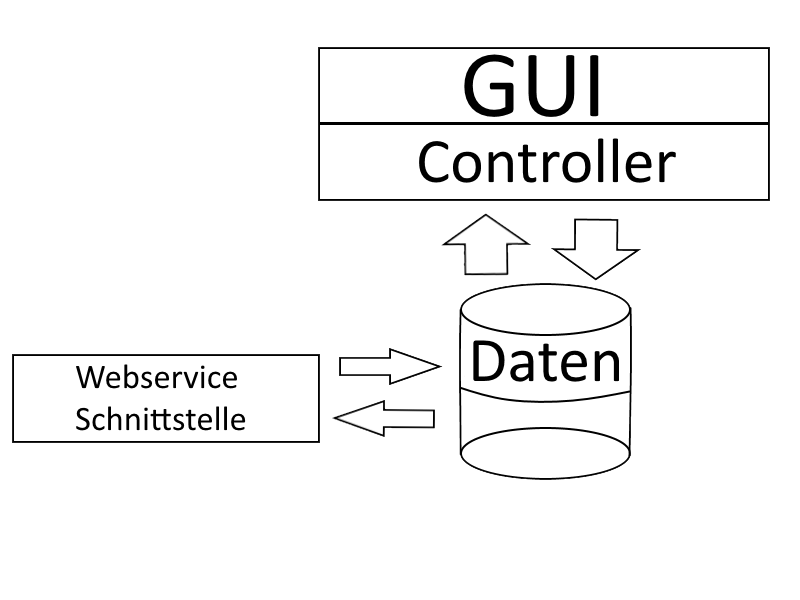


## C:\Users\Raphael\AppData\Local\Microsoft\Windows\INetCacheContent.Word\0001 (2).jpgLogin App

## Kommunikation (App und Webservice)



## Kommunikation in der App



# Arbeitsaufteilung

**Santner:**

* Datenbank
* Spiel löschen
* Spiel ändern
* Spiel anlegen
* Spieler löschen
* Spieler anpassen
* Spieler Statistiken
* Offline Modus
* Webservice
* Klassendiagramme
* Storymap
* Tester

**Wilscher:**

* Webservice Schnittstelle
* Einstellungen
* Datenbank
* Datenbank(Webservice)
* Klassendiagramme
* Impressum
* Tester

**Moser:**

* Spiel anlegen
* Spieler Statistiken (Scoreboard)
* Menü [Verantwortlich]
* Einstellungen
* Dokumentation
* Storymap
* Klassendiagramme
* Tester

**Sunny:**

* Layout
* Sicherheitsabfragen
* Spieler Auflistung
* Tester
* Spieler anlegen