Prototyping Projektdokumentation

Name: Joel Suhner

E-Mail: suhnejoe@students.zhaw.ch

URL der deployten Anwendung: <https://football-database.netlify.app/>

# Einleitung

Die entwickelte Webanwendung trägt den Namen "Football Player Database" und dient als interaktive Fussballspieler-Datenbank. Ziel der App ist es, Fussballinteressierten eine einfache Möglichkeit zu bieten, Spielerprofile, Vereinszugehörigkeiten und Ligainformationen zu durchsuchen und zu verwalten. Die Idee entstand aus dem Wunsch heraus, eine übersichtliche Plattform zu schaffen, auf der Nutzer Daten zu Spielern, Vereinen und Ligen einsehen, filtern und analysieren können – etwa zur Vorbereitung auf Fantasy-Football-Ligen, für Hobby-Scouting oder aus reinem Interesse.

Zu den Grundfunktionen gehören:

* **Darstellung von Spielerprofilen** mit Informationen wie Name, Alter, Nationalität, Position, Verein und Liga,
* **Filterfunktionen** nach Attributen wie Alter, Nationalität, Position, Verein
* **Verwaltung und Verknüpfung der Entitäten** (Spieler–Verein–Liga),
* **Erstellen von Spielern, Vereinen und Ligen** direkt über die Oberfläche inklusive
* **Optisch ansprechende Präsentation in Form von Spielerkarten**, wie man sie aus Spielen wie FIFA kennt

Abgrenzung: Es wurden keine Delete-Funktionen implementiert, da diese Datenbank auch als Spielerarchiv fungiert.

In einer zukünftigen Version könnten zusätzliche Funktionen ergänzt werden, um die Anwendung weiter aufzuwerten. Denkbar wären unter anderem eine Favoritenliste für Spieler, Statistikvergleiche, Live-Datenanbindung (z. B. aktuelle Leistungen oder Transfers), eine Benutzerverwaltung mit Login-Funktion, sowie Exportmöglichkeiten der Daten. Auch eine Verknüpfung mit weiteren externen APIs zur automatischen Datenaktualisierung ist eine denkbare Erweiterung.

# Datenmodell

Die Entitäten Club, League und Player sind selbsterklärend. Bei der Nationalität macht es aber Sinn, meinen Gedankengang wiederzugeben. In der MongoDB hat der Spieler ein Feld Nationalität mit einem Länder Text. Der Ländertext wird für das Joinen auf die Entität Nationalität verwendet, um die ISO-Codes der Länder abzurufen, die als Parameter für den Aufruf einer externen Flaggen-API verwendet werden. (<https://flagpedia.net/>)

Ein Bild, das Diagramm, Zeichnung enthält.

KI-generierte Inhalte können fehlerhaft sein.

# Beschreibung der Anwendung

## Startseite

Route: /

Dateien:

* lib/components/PlayerCard.svelte
* routes/+page.svelte
* routes/+layout.svelte
* routes/+page.server.js

Ein Bild, das Text, Screenshot, Software, Multimedia enthält.

KI-generierte Inhalte können fehlerhaft sein.

Auf der Startseite erfährt man Informationen zu Funktionalität und Zweck der Webanwendung. Es wurden dazu verschiedene Bootstrap Komponenten verwendet. Im unteren Bereich sind die zuletzt hinzugefügten Spieler im Card-Format zu sehen. Die Navigation und der Footer sind in der Datei +layout.svelte abgelegt. Es gibt für jede Entität einen Tab, der auf die Übersichtsseite des jeweiligen Entitätstypen führt.

## Ligenübersicht

Route: /leagues

Dateien:

* lib/components/LeagueCard.svelte
* routes/leagues/+page.svelte
* routes/leagues/+page.server.js

Ein Bild, das Text, Screenshot, Software, Marke enthält.

KI-generierte Inhalte können fehlerhaft sein.

Die Ligenübersicht zeigt alle erstellten Ligen an im Card-Format an. Diese werden über LeagueCard.svelte geladen. Oben links gibt es Möglichkeit, Ligen nach Land zu filtern, oder ob sie eine Top 5 Liga sind. Darunter gibt es einen Button, mit dem man zu /leagues/create weitergeleitet wird. Zusätzlich kann man auf den unterstrichenen Text in der Card klicken, um zu Liga Detailansicht zu gelangen (/leagues/[league\_id]).

## Clubübersicht

Route: /clubs

Dateien:

* lib/components/ClubCard.svelte
* routes/clubs/+page.svelte
* routes/clubs/+page.server.js

Ein Bild, das Text, Screenshot, Logo, Symbol enthält.

KI-generierte Inhalte können fehlerhaft sein.

In der Clubübersicht sieht man alle erstellten Vereine im Card-Format an. Diese werden über ClubCard.svelte geladen. Oben links gibt es einen Button, mit dem man zu /clubs/create weitergeleitet wird. Auch hier kann man auf den unterstrichenen Text in der Card klicken, um zu der Club Detailansicht (/clubs/[club\_id]) weitergeleitet zu werden.

## Spielerübersicht

Route: /players

Dateien:

* lib/components/PlayerCard.svelte
* routes/players/+page.svelte
* routes/players /+page.server.js

Ein Bild, das Text, Screenshot, Person, Website enthält.

KI-generierte Inhalte können fehlerhaft sein.

Die Spielerübersicht zeichnet sich durch viele Filtermöglichkeiten aus, die auf alle Spieler in der Collection angewendet werden. Dies ist genauer im Kapitel 4.4 beschrieben. Zusätzlich zu der normalen Card Ansicht, gibt es eine Flagge, die die Nationalität des Spielers repräsentiert (Kapitel 4.2). Neue Spieler können über den Button unterhalb der Filteroptionen erstellt werden (/players/create).

## Liga Detailansicht

Route: /leagues/[league\_id]

Dateien:

* lib/components/ClubCard.svelte
* lib/components/PlayerCard.svelte
* routes/leagues/[league\_id]/+page.svelte
* routes/leagues/[league\_id]/+page.server.js

Ein Bild, das Text, Screenshot, Logo, Grafikdesign enthält.

KI-generierte Inhalte können fehlerhaft sein.

In der Detailansicht einer Liga werden zusätzliche Informationen zu der Liga angezeigt wie zum Beispiel das Land, oder ob es eine Top 5 Liga ist. Rechts daneben werden alle Clubs und Spieler der Liga angezeigt. Dabei werden die beiden Komponenten ClubCard und PlayerCard verwendet. Wenn man auf den unterstrichenen Text innerhalb der ClubCard klickt, gelangt man zu der Club Detailansicht (/clubs/[club\_id]).

## Club Detailansicht

Route: /clubs/[club\_id]

Dateien:

* lib/components/PlayerCard.svelte
* routes/clubs/[club\_id]/+page.svelte
* routes/clubs/[club\_id]/+page.server.js

Ein Bild, das Text, Screenshot, Person, Logo enthält.

KI-generierte Inhalte können fehlerhaft sein.

In der Detailansicht eines Clubs werden zusätzliche Informationen zu dem Club angezeigt wie zum Beispiel in welcher Liga der Klub ist und das Gründungsjahr. Rechts daneben werden alle Spieler des jeweiligen Vereines angezeigt. Hier wird nur noch die PlayerCard für das Anzeigen der Spieler benötigt.

## Liga erstellen

Route: /leagues/[create]

Dateien:

* lib/db.js
* routes/leagues/create/+page.svelte
* routes/leagues/create/+page.server.js

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite können neue Ligen erstellt werden. Dabei können der Name und das Land, in dem sich die Liga befindet, angegeben werden. Im db.js wird eine numerische ID generiert (Siehe 4.5) und ein Standardbild wird als Bild-URL verwendet. Nach erfolgreichem Erstellen einer Liga, gibt es ein visuelles Feedback. Mit dem Button «Back» gelangt man wieder zurück zur Ligenübersicht (/leagues).

## Club erstellen

Route: /clubs/[create]

Dateien:

* lib/db.js
* routes/clubs/create/+page.svelte
* routes/clubs/create/+page.server.js

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Hier kann ein neuer Klub erstellt werden. Die Attribute Clubname und Gründungsjahr sind klassische Input-Felder. Die Liga kann per Dropdown ausgewählt werden, da über getLeagues() alle bisherig verwendeten Ligen zur Populierung der Optionen verwendet wird. Genaueres dazu ist unter Kapitel 4.6 zu finden. Im db.js wird zusätzlich, wie bei den Ligen, ein Standardpfad für das Clublogo angegeben. Beim Klicken auf «Back» kehrt man auf die Seite «/clubs» zurück.

## Spieler erstellen

Route: /players/[create]

Dateien:

* lib/db.js
* routes/players/create/+page.svelte
* routes/players/create/+page.server.js

Ein Bild, das Text, Screenshot, Zahl, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Um einen Spieler zu erstellen, muss man Eingaben bezüglich Namens, Geburtsdatum, Nationalität, Position und Clubzugehörigkeit tätigen. Die Liga kann im weiteren Verlauf von der Club ID abgeleitet werden. Bei der Nationalität ist es wichtig, dass nur der erste Buchstabe gross geschrieben ist und der Ländername in Englisch eingegeben wird. Die Auswahl des Klubs wird über ein populiertes Dropdown gemacht (Siehe 4.6). Für neue Spieler wird ein Standardprofilbild im Code von db.js (createPlayer()) zugewiesen. Mit «Back» kehrt man zurück zur Seite «/players».

# Erweiterungen

## Newest Created Players

Datei(en):

* lib/db.js
* routes/+page.svelte
* routes/+page.server.js

Auf der Startseite werden im unteren Bereich die drei Spieler angezeigt, die als letztes zur Datenbank hinzugefügt wurden. Dies wurde mit einem load auf die Spieler-Tabelle getPlayers() erreicht. Der return-Array wird dann auf die letzten drei Player-Objekte gesliced und angezeigt.

Ein Bild, das Text, Menschliches Gesicht, Person, Mann enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Externer API-Zugriff auf Flaggen-Bilder

Datei(en):

* lib/db.js
* lib/components/PlayerCard.svelte
* routes/players/+page.svelte
* routes/players/+page.server.js

Bei den Spieler-Cards gibt es auf der linken Seite eine Flagge. Diese Bilddatei wird über eine API in die Applikation geladen. Das Design ist in den Dateien «PlayerCard.svelte» und «stlye.css» abgelegt. Die Funktionen getPlayers(), getPlayersOfClub() und getPlayersOfLeague() enthalten diese Funktionalität. Der ISO-Code, der in der Collection «nationalities» abgelegt ist, wird verwendet, um auf die folgende URL zuzugreifen: [https://flagcdn.com/w320/<iso-code>.png](https://flagcdn.com/w320/%3ciso-code%3e.png). Dabei wird ein komplexes Query verwendet, welches die Stages lookup, unwind und project enthält. Bei der Erstellung von neuen Spielern muss die Nationalität (Land) in englischer Sprache eingegeben werden. Ausserdem muss der erste Buchstabe gross geschrieben werden.

Ein Bild, das Text, Kleidung, Screenshot, Menschliches Gesicht enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Liga Filter

Datei(en):

* lib/db.js
* lib/components/LeagueCard.svelte
* routes/leagues/+page.svelte
* routes/leagues/+page.server.js

### Länderfilter bei Ligen

Auf der Liga Seite kann man nach den Ländern filtern, in denen sich die Ligen befinden. Es kann also auch sein, dass ein Land mehrere Ligen hat, wie unten im Bild ersichtlich ist. Die Auswahl des Dropdowns wird anhand des Länder-Arrays, der von getLeagueCountries() returned wird, aufgebaut. So wird auch sichergestellt, dass jedes Land auch nur einmal vorkommt und der Dropdown alphabetisch sortiert ist.

Ein Bild, das Text, Screenshot, Logo, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

### Filter Top 5 Liga

Zusätzlich gibt es noch einen Checkbox-Filter, ob eine Liga zu den Top 5 Ligen gehört. Die Top 5 sind im unteren Bild zu sehen. In der league-collection gibt es ein Boolean Attribut namens «is\_top5», welches hier für die Filterung verwendet wird. Im Unterschied zu der vorherigen Funktion musste hier keine zusätzliche JS-Funktion geschrieben werden. Es wird hier lediglich auf das Attribut der league-Objekte zugegriffen. Beide Filter können gleichzeitig verwendet werden, da sie im Code mit logischen Operationen verbunden wurden.

Ein Bild, das Text, Screenshot, Logo, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Filter bei Spielern

Auch Spieler können gefiltert werden. Es gibt eine Reihe von Filtern, die gleichzeitig angewendet werden. Diese Filteroptionen werden im Folgenden genauer erläutert.

Datei(en):

* lib/db.js
* lib/components/PlayerCard.svelte
* routes/players/+page.svelte
* routes/players/+page.server.js

### Club Filter

Dieser Filter konnte mithilfe des club Objekts umgesetzt werden. Mithilfe von getClubs() gelangt man an die Clubnamen, welche dann automatisch durch Svelte alphabetisch sortiert werden.

### Nationalität- und Positionsfilter

Diese beiden Filter sind komplexer aufgebaut. Die Daten werden mit über die Funktion getPlayers() geliefert. Die Nationalität bzw. die Position könnte jetzt auch wie beim Club Filter implementiert werden. Jedoch gibt es das Problem, dass diese Optionen auch Duplikate enthalten und nicht alphabetisch sortiert sind. Darum wurde in der +page.svelte im script-Bereich eine Funktion geschrieben, die dieses Problem behebt.

### Alter Filter

Das Feld Alter gibt es gar nicht, lediglich das Geburtsdatum des Spielers. So musste eine Logik implementiert werden, die anhand des Geburtsdatums das Alter kalkuliert. Dies ist gar nicht so einfach, denn (aktuelles Jahr – Geburtsjahr) wäre falsch. Des Weiteren wurde aus dem generierten Alter-Array ein Maximal- und Minimalalter in eine Variable geschrieben. Diese werden für den Range Slider benötigt, denn es ist keine fixe Range, sondern diese ergibt sich aus dem tiefsten und höchsten Alter.

### «Reset filters»-Button

Zu guter Letzt wurde ein Button hinzugefügt, der alle gesetzten Filter auf dieser Seite zurücksetzt. Die Funktion resetFilters() (im script-Bereich von +svelte.page) wird ausgelöst von dem onclick-Event, das auf dem Button hinterlegt ist.

Ein Bild, das Text, Person, Mann, Menschliches Gesicht enthält.

KI-generierte Inhalte können fehlerhaft sein.

## «Auto Increment»-Funktion beim Erstellen von neuen Ligen, Clubs, Spielern

Datei(en):

* lib/db.js
* routes/players/create/+page.server.js
* routes/clubs/create/+page.server.js
* routes/leagues/create/+page.server.js

Wenn neue Documents in MongoDB erstellt werden, besteht die \_id aus einem zufällig generierten String. Dies ist unschön und erschwert das Selektieren von individuellen IDs. Bei dem Erstellen von Liga, Club oder Spieler wird der entsprechende create<Entity>() Befehl ausgeführt. Innerhalb dieses Befehls in der Datei db.js gibt es je eine Funktion count<Entities>(), die die aktuelle Anzahl an Entitäten übergibt. Diese wird um 1 erhöht und das neue Dokument hat somit eine um 1 höhere \_id als das vorherig höchste. So kann eine auto-inkrement Funktionalität sichergestellt werden. Ausserdem kann beispielweise nachvollzogen werden, welche Spieler/Clubs/Ligen als letztes hinzugefügt wurden.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Auto generierte Dropdowns für Erstellung von neuen Clubs und Spielern

Datei(en):

* lib/db.js
* routes/players/create/+page.svelte
* routes/players/create/+page.svelte
* routes/players/create/+page.server.js
* routes/clubs/create/+page.svelte
* routes/clubs/create/+page.server.js

Wenn man einen neuen Spieler erstellen will, muss man diesem einen Verein hinzufügen. Da dies bei einem Textfeld eine potenzielle Fehlerquelle für Dateninkonsistenzen ist, gibt es ein Dropdown, mit dem man nur bereits erstellte Vereine auswählen kann. Ein weiterer Vorteil daran ist, dass nicht ein Fremdschlüssel eingegeben werden muss, sondern die ID versteckt in dem «Option-Value» enthalten ist. Die Daten der Vereine werden über die Methode getClubs() geladen. Eine ähnliche Funktion gibt es auch bei dem Erstellen von Vereinen, wo eine bereits existierende Liga hinzugefügt werden muss.

Ein Bild, das Text, Screenshot, Zahl, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Software, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.