Prototyping Projektdokumentation

Name: Andjelina Tosic

E-Mail: tosicand@students.zhaw.ch

URL der deployten Anwendung: https://greatestofnba.netlify.app

# Einleitung

NBA Legends ist eine moderne Webanwendung, die den Nutzer auf eine interaktive Reise durch die Welt der nordamerikanischen Basketballliga NBA mitnimmt. Die App bietet umfassende Informationen zu legendären Spielern, historischen Teams und anstehenden Spielen, alles übersichtlich und visuell ansprechend auf einer einzigen Plattform vereint.   
Im Mittelpunkt stehen die besten NBA-Spieler aller Zeiten. Nutzer können sich durch eine Vielzahl von Spielerprofilen klicken, die neben Bildern auch Informationen zu Team, Position, Alter und Status enthalten. Zusätzlich ist für viele Spieler eine detaillierte Biografie hinterlegt, die aus einer separaten MongoDB-Collection geladen wird. Sollte ein Lieblingsspieler in der Datenbank noch fehlen, kann man ihn ganz einfach über ein intuitives Formular selbst hinzufügen, inklusive Bild-URL und aller relevanten Daten. So lässt sich die Datenbank aktiv mitgestalten.

Die Teamübersicht präsentiert die wichtigsten Fakten über jedes Team – wie Arena, Ort, CEO und gewonnene Titel. Zudem kann man direkt ein Quiz zu jedem Team starten und so spielerisch sein NBA-Wissen testen.

Ein besonderes Highlight ist die Spielplan-Seite: Sie zeigt alle kommenden NBA-Spiele in Echtzeit – inklusive Heim- und Auswärtsteam, Ort, Datum und Uhrzeit. Damit bietet die App nicht nur historische Einblicke, sondern auch topaktuelle Informationen. Man muss keine andere Website mehr aufrufen – alle wichtigen Daten sind zentral abrufbar.

NBA Legends wurde vollständig mit SvelteKit im Runes-Modus entwickelt und nutzt MongoDB als Datenbasis. Das Projekt zeigt, wie eine datengetriebene Webanwendung sowohl informativ als auch unterhaltsam sein kann.

# Datenmodell

Ein Bild, das Screenshot, Kreis, Schrift, Schwarzweiß enthält.

KI-generierte Inhalte können fehlerhaft sein.

# Beschreibung der Anwendung

## Startseite

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

KI-generierte Inhalte können fehlerhaft sein.

Auf der Startseite wird die gesamte Anwendung kurz vorgestellt. Sie dient als zentraler Einstiegspunkt für die Nutzer. Die Seite besteht aus einem hervorgehobenen Titelbereich mit dem Namen der App „NBA Legends“, einem erklärenden Untertitel und farblich hervorgehobenen Buttons zu den vier Hauptbereichen:

* Spieler entdecken: Zeigt Biografien und Bilder berühmter NBA-Spieler
* Teams anzeigen: Präsentiert alle Teams mit Gründung, Arena, CEO und mehr
* Spielplan ansehen: Zeigt kommende NBA-Spiele über eine Live-API
* Quiz: Testet das Wissen über NBA-Teams anhand von Fragen

Darunter befindet sich ein Kachelbereich, der die vier Funktionen nochmals mit Icons und Beschreibung zusammenfasst. Die Seite nutzt die Navigationskomponente Navbar.svelte, die in der gesamten App verwendet wird.

**Dateien:**

* lib/components/Navbar.svelte
* routes/+page.svelte
* styles.css
* routes/+layout.svelte

## Spielerübersicht

Route: /player Ein Bild, das Text, Screenshot, Software, Person enthält.

KI-generierte Inhalte können fehlerhaft sein.

Auf der Seite Spielerübersicht /players werden alle verfügbaren NBA-Spieler in Kartenform angezeigt. Jede Karte zeigt grundlegende Informationen wie Name, Team, Position, Alter und Lebensstatus. Nutzer können die Liste mithilfe eines Suchfelds sowie Dropdown-Menüs nach bestimmten Kriterien wie Name, Position, Team oder Lebensstatus filtern. Zusätzlich steht ein Button zur Verfügung, um neue Spieler direkt zur Datenbank hinzuzufügen. Für die Darstellung der Spieler wird die Komponente PlayerCard.svelte verwendet. Die Daten werden serverseitig über +page.server.js geladen.

**Dateien:**

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

## Detailansicht Spieler

**Route:** /players/[player\_id]Ein Bild, das Text, Screenshot, Software, Website enthält.

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite werden alle verfügbaren Informationen zu einem einzelnen Spieler angezeigt. Dazu gehören Team, Position, Alter und Lebensstatus, die neben dem Spielerbild dargestellt werden. Unterhalb folgt eine ausführliche Biografie. Diese Biografie wird dynamisch aus einer separaten MongoDB-Collection «BioPlayers» geladen, anhand des Spielernamens. Die Suche erfolgt per Regex-Match auf den Namen des Spielers. Die Daten werden automatisch serverseitig beim Laden der Seite bereitgestellt.

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

KI-generierte Inhalte können fehlerhaft sein.Zusätzlich besteht die Möglichkeit, den Spieler aus der Datenbank zu löschen mit dem Button «Spieler löschen».

**Dateien:**

* lib/components/PlayerDetailCard.svelte
* routes/players/[player\_id]/+page.svelte
* routes/players/[player\_id]/+page.server.js
* lib/database.js

## Spieler erstellen

**Route**:/players/create

Ein Bild, das Text, Schrift, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.Ein Bild, das Text, Screenshot, Person, Website enthält.

KI-generierte Inhalte können fehlerhaft sein.Ein Bild, das Text, Software, Computersymbol, Betriebssystem enthält.

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite können Benutzer neue NBA-Spieler manuell zur Datenbank hinzufügen. Das Formular umfasst Felder für den Namen, das Alter, das Team, die Position, den Lebensstatus (lebend oder verstorben) sowie optional eine Bild-URL. Die Eingaben werden nach dem Absenden serverseitig verarbeitet und über eine definierte POST-Action in die MongoDB-Datenbank gespeichert. Die Altersangabe wird dabei automatisch als Zahl interpretiert. Nach erfolgreicher Erstellung wird der Benutzer zurück zur Übersichtsseite weitergeleitet, auf der der neue Spieler dann erscheint.

**Dateien:**

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

## Teamübersicht

**Route:** /teams

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

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite werden alle NBA-Teams aus der MongoDB-Datenbank visuell in Kartenform dargestellt. Jede Karte enthält wichtige Teaminformationen wie CEO, Arena, Ort und die Anzahl gewonnener Championships. Zusätzlich ist ein Button integriert, über den ein themenspezifisches Quiz zum jeweiligen Team gestartet werden kann. Die Teams werden durch die Server-Load-Funktion geladen und anschliessend mit der TeamCard-Komponente dargestellt. Ein Zurück-Button in der oberen rechten Ecke ermöglicht die einfache Navigation zur vorherigen Seite.

**Dateien:**

* src/lib/components/TeamCard.svelte
* src/routes/teams/+page.svelte
* src/routes/teams/+page.server.js

## Quiz

**Route:** /quiz/[teamname]

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

KI-generierte Inhalte können fehlerhaft sein.

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

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite wird ein interaktives Quiz für das jeweils ausgewählte NBA-Team angezeigt. Die Route basiert dynamisch auf dem Teamnamen in der URL, wie /quiz/los-angeles-lakers. Die Seite lädt beim Aufruf zunächst alle Teams aus der MongoDB und vergleicht den URL-Teamnamen mit den vorhandenen Teams. Anschliessend werden passende Quizfragen zu diesem Team aus der Collection quizfragen geladen. Der Benutzer kann die Fragen nacheinander beantworten. Nach dem Klick auf „Antwort prüfen“ wird visuell angezeigt, ob die Antwort richtig oder falsch war. Am Ende des Quiz wird die Gesamtpunktzahl angezeigt.

**Dateien:**

* → routes/quiz/[teamname]/+page.svelte
* → routes/quiz/[teamname]/+page.server.js
* → lib/database.js (getQuizByTeamname)

## Spielplanübersicht

**Route:** /schedule

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

KI-generierte Inhalte können fehlerhaft sein.

Auf dieser Seite werden automatisch alle anstehenden NBA-Spiele der nächsten Tage angezeigt. Die Daten stammen von der externen Balldontlie API und werden beim Laden der Seite über eine API-Route (/api/schedule) abgerufen. Es werden jeweils Heim- und Auswärtsteam, das Datum, die Uhrzeit sowie der Austragungsort der Partie dargestellt.

Zusätzlich wird oberhalb der Liste ein eingebettetes YouTube-Video angezeigt. Dies dient zur Einstimmung und verleiht der Seite zusätzliche Dynamik.

**Dateien:**

* → routes/schedule/+page.svelte
* → routes/api/schedule/+server.js
* → lib/database.js

# Erweiterungen

## Spielplan-Integration mit externer API und Video

Auf der Seite /schedule wird eine Liste kommender NBA-Spiele angezeigt, die über die externe API balldontlie.io geladen wird.  
Die Daten beinhalten Heim- und Auswärtsteam, Datum, Uhrzeit und Ort. Zusätzlich wurde ein YouTube-Video-Embed als visuelles Element ergänzt.

Im Hintergrund nutzt die API-Funktion getUpcomingGamesNBA() aus der zentralen Datei database.js, welche eine HTTP-Anfrage an die Balldontlie API sendet. Die abgerufenen Daten werden clientseitig verarbeitet und formatiert angezeigt.

**Dateien:**

* lib/database.js (Funktion getUpcomingGamesNBA)
* routes/api/schedule/+server.js (API-Handler)
* routes/schedule/+page.svelte (Anzeige & UI)

## Spielerbiografie aus MongoDB-Collection

Auf der Detailseite eines Spielers /players/[player\_id] wird neben den Basisdaten (Name, Team, Alter usw.) auch eine ausführliche Biografie angezeigt.  
Diese Biografie stammt aus einer separaten MongoDB-Collection «BioPlayers» und wird dynamisch per Spielernamen abgerufen, Regex-basierte Suche in getBioByName().

Die Biografiedaten sind in einer separaten Entität gespeichert, was die Datenstruktur flexibel und wartbar hält.  
Die Darstellung erfolgt in einer separaten Komponente PlayerDetailCard.svelte, die sowohl Spieler- als auch Biografiedaten verarbeitet.

**Dateien:**

* lib/database.js
* routes/players/[player\_id]/+page.server.js
* routes/players/[player\_id]/+page.svelte
* lib/components/PlayerDetailCard.svelte

## Teambezogenes Quiz

Auf der Seite /quiz/[teamname] wird ein interaktives Quiz über das gewählte NBA-Team angezeigt. Die Fragen stammen aus einer eigenen MongoDB-Collection «quizfragen».  
Die Seite ist dynamisch aufgebaut: Anhand des URL-Segments (z. B. los-angeles-lakers) wird das Team identifiziert und die passenden Fragen geladen.

Das Quiz enthält Multiple-Choice-Fragen und zeigt dem Nutzer nach jeder Antwort visuelles Feedback (richtig/falsch). Am Ende wird ein Score angezeigt.  
Die Zuordnung erfolgt über die Funktion getQuizByTeamname() aus der Datenbank.

**Dateien:**

* lib/database.js (Funktion getQuizByTeamname)
* routes/quiz/[teamname]/+page.server.js
* routes/quiz/[teamname]/+page.svelte

## Spielerbilder per externer URL

Die Bilder der Spieler auf der /players Seite werden über eine externe URL eingebunden, anstatt lokal im Projekt gespeichert zu werden. Dadurch können Nutzer beim Erstellen eines neuen Spielers jedes beliebige Bild aus dem Internet verwenden, ohne dass Bilddateien manuell in das Projektverzeichnis hochgeladen oder verwaltet werden müssen.

Wird keine Bild-URL angegeben, erscheint automatisch ein Fallback-Bild (/images/Ebene.png), um die Darstellung konsistent zu halten.

Diese Erweiterung reduziert Speicherbedarf und vereinfacht die Bildverwaltung für Nutzer erheblich.

**Dateien:**

* lib/components/PlayerCard.svelte –

## Spieler-Suche per Namenseingabe

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

KI-generierte Inhalte können fehlerhaft sein.

Auf der Seite /players wurde eine Suchfunktion implementiert, mit der Nutzer gezielt nach Spielernamen filtern können. Bereits während der Eingabe wird die Spielerübersicht in Echtzeit aktualisiert. Diese Funktion erleichtert die Navigation bei grossen Datenmengen und verbessert die Benutzerfreundlichkeit erheblich.

Die Suche erfolgt rein clientseitig über ein includes()-Matching gegen den Namen aller geladenen Spieler – unabhängig von Gross- oder Kleinschreibung.

**Dateien:**

* routes/players/+page.svelte   
  Eingabefeld mit bind:value={search} und dynamischem Filter in der each-Schleife