Prototyping Projektdokumentation

Name: Arbër Haliti

E-Mail: haljiarb@students.zhaw.ch

URL der deployten Anwendung: <https://haljiarb.netlify.app/>

# Einleitung

Die Anwendung My Techno & House Website ist eine spezialisierte Musik-App für Liebhaber:innen elektronischer Musik, insbesondere der Genres Techno und House. Ziel der App ist es, eine übersichtliche und interaktive Plattform zu bieten, auf der Nutzer:innen Songs und Artists entdecken, verwalten und pflegen können. Die App zeigt eine Sammlung von Songs an, inklusive Titel, Artist und weiterführender Informationen. Ebenso können Artists mit zusätzlichen Details wie Biografie oder Herkunft angezeigt werden. Nutzer:innen haben die Möglichkeit, eigene Songs zu erstellen oder zu löschen sowie Artists anzulegen oder zu bearbeiten. Ein zentrales Feature ist die Integration der Spotify API, durch die automatisch die aktuell beliebtesten Artists aus den Genres House und Dance eingebunden werden.

# Datenmodell

A diagram of a diagram

AI-generated content may be incorrect.

Das ER-Diagramm stellt die Datenstruktur der App „My Techno & House Website“ dar. Es besteht aus zwei Entitäten:

* Artist: Enthält Informationen zu den Künstler:innen wie artist\_name, genre und photo. Jeder Artist hat eine eindeutige ID (\_id).
* Songs: Enthält Informationen zu einzelnen Songs wie song\_name und photo. Jeder Song ist über das Attribut artist\_id einem bestimmten Artist zugeordnet.

# Beschreibung der Anwendung

*Route: /*

**Blau:** Die Header-Komponente mit dem Titel der Webseite, Navigationsmenü und einem Shuffle-Button. Der Shuffle-Button sorgt dafür, dass Songs und Artists erneut zufällig ausgegeben werden – also geshuffled werden. Technisch handelt es sich dabei lediglich um eine Reload-Funktion.

**Rot:** Eine zufällige Auswahl aus der MongoDB-Datenbank. Aus beiden Datensätzen wird jeweils ein Sample mit der Grösse 12 entnommen und angezeigt. Oben werden 12 zufällige Artists mit Foto und zugehörigem Artistnamen dargestellt. Der „Show All“-Button zeigt alle Songs an. Unten werden 12 verschiedene Artists mit Bild und Name angezeigt. Der „Show All“-Button zeigt alle Artists an.

**Pink:** Zwei kleinere Div-Container mit einer kurzen Beschreibung, um entweder einen Artist zu erstellen oder zu aktualisieren sowie einen Song zu erstellen oder zu löschen.

**Grün:** Es werden die derzeit beliebtesten Artists auf Spotify aus den Genres Dance und House angezeigt. Diese Daten werden über die Spotify API abgerufen.

A screenshot of a music application

AI-generated content may be incorrect.

Dateien:  
src/lib/components/Header.svelte  
src/lib/components/NavBar.svelte  
src/lib/components/HouseArtistList.svelte  
src/lib/server/spotify.ts  
src/app.html  
src/routes/+layout.server.ts  
src/routes/+layout.svelte  
src/routes/+page.server.ts  
src/routes/+page.svelte

## Alle Songs

Route: /songs

Eine Übersicht aller Songs, jeweils mit dem zugehörigen Artist, in einem Grid dargestellt.  
  
A screenshot of a music album

AI-generated content may be incorrect.

Dateien:

src/lib/components/Header.svelte  
src/lib/components/NavBar.svelte  
src/lib/components/HouseArtistList.svelte  
src/lib/server/spotify.ts  
src/app.html  
src/routes/+layout.server.ts  
src/routes/+layout.svelte  
src/routes/songs/+page.server.ts  
src/routes/songs/+page.svelte

## Song erstellen und löschen

Route: songs/create

Song erstellen: Auf dieser Seite kann ein neuer Song angelegt werden. Dazu müssen folgende Angaben gemacht werden:

* eine ID (diese muss eindeutig sein und darf noch nicht verwendet worden sein),
* der Songtitel,
* der zugehörige Artist,
* optional ein Bild.

Hinweis: Bei der ID-Eingabe ist darauf zu achten, dass keine bereits vergebene ID verwendet wird – sonst schlägt die Erstellung fehl. Dies liegt daran, dass die \_id-Felder ursprünglich manuell mit fortlaufenden Nummern (row numbers) erstellt wurden, anstatt die automatisch generierten IDs von MongoDB zu verwenden.

Falls das eingegebene Bild im Projektverzeichnis nicht vorhanden ist, wird es ebenfalls nicht angezeigt.

A screenshot of a computer

AI-generated content may be incorrect.

Dateien:

src/lib/components/Header.svelte  
src/lib/components/NavBar.svelte  
src/lib/components/HouseArtistList.svelte  
src/lib/server/spotify.ts  
src/app.html  
src/routes/+layout.server.ts  
src/routes/+layout.svelte  
src/routes/songs/create/+page.server.ts  
src/routes/songs/create/+page.svelte

## Alle Artists

Route: /artists

Eine Übersicht von allen Artists mit Namen. Wenn man auf den jeweiligen Artist draufdrückt kommt man zu seiner Detailseite.

A screenshot of a computer

AI-generated content may be incorrect.

Dateien:

src/lib/components/Header.svelte  
src/lib/components/NavBar.svelte  
src/lib/components/HouseArtistList.svelte  
src/lib/server/spotify.ts  
src/app.html  
src/routes/+layout.server.ts  
src/routes/+layout.svelte  
src/routes/artists/+page.server.ts  
src/routes/artists/+page.svelte

## Artists [ID] Detailseite

Route: artists/[id]

Eine Detailsite des jeweiligen Artists zeigt Namen und Genre an:

A group of men playing music

AI-generated content may be incorrect.

Dateien:

src/lib/components/Header.svelte  
src/lib/components/NavBar.svelte  
src/lib/components/HouseArtistList.svelte  
src/lib/server/spotify.ts  
src/app.html  
src/routes/+layout.server.ts  
src/routes/+layout.svelte  
src/routes/artists/[id]/+page.server.ts  
src/routes/artists/[id]/+page.svelte

## Artist erstellen oder updaten

route: artists/create

Artists erstellen:

Auf dieser Seite kann ein neuer Artist angelegt werden. Dazu müssen folgende Angaben gemacht werden:

* eine ID (diese muss eindeutig sein und darf noch nicht verwendet worden sein),
* der Artistname,
* optional das Genre
* optional ein Bild.

Artists updaten:

Auf dieser Seite kann ein neuer Artist geupdated werden. Dazu müssen folgende Angaben gemacht werden:

* Artist auswählen
* Neuer Name (optional)
* Neues Genre (optional)
* Neues Foto (optional)

Hinweis: Bei der ID-Eingabe ist darauf zu achten, dass keine bereits vergebene ID verwendet wird – sonst schlägt die Erstellung fehl. Dies liegt daran, dass die \_id-Felder ursprünglich manuell mit fortlaufenden Nummern (row numbers) erstellt wurden, anstatt die automatisch generierten IDs von MongoDB zu verwenden.

A screenshot of a computer

AI-generated content may be incorrect.

Dateien:

src/lib/components/Header.svelte  
src/lib/components/NavBar.svelte  
src/lib/components/HouseArtistList.svelte  
src/lib/server/spotify.ts  
src/app.html  
src/routes/+layout.server.ts  
src/routes/+layout.svelte  
src/routes/artists/create/+page.server.ts  
src/routes/artists/create/+page.svelte

# Erweiterungen

Für die Erweiterung habe ich mich für das CSS-Framework Tailwind und die Einbindung der Spotify API entschieden, um die drei beliebtesten House- und Dance-Künstler auf Spotify anzuzeigen.

## Tailwind Library

Verwendet in: allen Komponenten

Tailwind CSS ist ein Utility-First CSS Framework, das statt vordefinierter Komponenten viele kleine CSS-Klassen bereitstellt, mit denen direkt im HTML-Markup gestylt wird. Statt eigene CSS-Dateien zu schreiben, nutzt man Tailwind-Klassen wie z. B. bg-gray-100, text-center, p-4 oder rounded-lg, um Layout, Farben, Abstände, Schriftgrössen usw. zu definieren.

***Funktionsweise:***

* Utility-Klassen: Jede Klasse beschreibt eine einzelne CSS-Eigenschaft (z. B. mt-4 für margin-top: 1rem).
* Direktes Styling im Markup: Designentscheidungen werden direkt im Code der Komponente getroffen, was die Trennung zwischen HTML/JSX und CSS reduziert.
* Responsives Design: Tailwind bietet Breakpoints wie sm:, md:, lg:, um Designs für verschiedene Bildschirmgrössen zu definieren.
* Customisierung über tailwind.config.js: Farben, Abstände und Schriftarten können projektweit angepasst werden.

***Vorteile der Erweiterung:***

* Schnelles Prototyping und konsistentes Design
* Responsives und modernes UI ohne eigene CSS-Dateien
* Verbesserte Wartbarkeit und Lesbarkeit durch zentrale Nutzung von Utility-Klassen

*Beispiel mit Tailwind:*

<button className="bg-blue-500 text-white px-4 py-2 rounded">

Klick mich

</button>

*Beispiel mit HTML und CSS:*

<button class="my-button">Klick mich</button>

.my-button {

background-color: #3b82f6; /\* entspricht blue-500 \*/

color: white;

padding: 0.5rem 1rem;

border-radius: 0.25rem;

}

**Dateien:**

Alle Components und Pages

## Spotify API

Diese Erweiterung ergänzt die Grundanforderungen, indem sie die offiziellen Spotify‑Live‑Daten nutzt, um stets aktuelle, populäre Artists der Genres House und Dance in der Anwendung anzuzeigen.

***Hauptdateien:***

* src/lib/server/spotify.ts – enthält die gesamte API‑Kommunikation (Token‑Handling, Abfragen).
* src/routes/+layout.server.ts – ruft die API serverseitig auf und stellt die Daten allen Unterseiten bereit.
* src/lib/components/HouseArtistList.svelte – rendert die Artist‑Liste im Frontend mit Tailwind CSS.

***Authentifizierung***

* In spotify.ts stellt die Funktion getSpotifyToken() den OAuth‑„Client Credentials Flow“ bereit. Dabei werden die Umgebungsvariablen SPOTIFY\_CLIENT\_ID und SPOTIFY\_CLIENT\_SECRET verwendet, um ein Access‑Token von https://accounts.spotify.com/api/token anzufordern.
* Das Token wird zusammen mit seinem Ablaufzeitpunkt in einer lokalen Variable zwischengespeichert und nur erneuert, wenn es abgelaufen ist. Dadurch werden unnötige Anfragen an Spotify vermieden.

***Datenabruf***

* Die Funktion getTopHouseArtists(limit) (ebenfalls in spotify.ts) ruft über das zuvor erhaltene Token die Endpoint‑URL https://api.spotify.com/v1/search auf.
* Es wird nach „genre:house, dance“ gefiltert und nach Popularität sortiert. Das Ergebnis ist ein Array mit den beliebtesten Artists (Name, Bilder, Popularität etc.).

***Server‑seitiges Laden***

* In +layout.server.ts wird im load()-Hook bei jedem ersten Seitenaufruf const houseArtists = await getTopHouseArtists() aufgerufen.
* Die Artists werden als Props an alle Unterseiten weitergegeben; dadurch werden sie direkt gerendert und Client‑Side‑Flashes entfallen.

***Frontend‑Darstellung***

* HouseArtistList.svelte erzeugt mit Tailwind‑Klassen (z. B. bg-[#202020] p-4 rounded-xl) ein responsives Grid aus Artist‑Cards.
* Jede Card zeigt das Artist‑Bild, die Platzierung und den Namen.

***Nutzen für den Benutzer***

* Stets aktuelle Inhalte: Die Listung aktualisiert sich automatisch über Spotify‑Live‑Daten.
* Zero‑Maintenance: Keine manuelle Pflege von Künstlerlisten nötig.
* Demonstriert fortgeschrittene API‑Integration, OAuth‑Flow und Server‑Rendering – klare Mehrwerte über die Grundanforderungen hinaus.