

# Prototyping Projektdokumentation

**Name:** Nicolas Fehr

**E-Mail:** fehrnic1@students.zhaw.ch

**URL der deployten Anwendung:** <https://ratingrecords.netlify.app/records>

**URL des Git-Repositories:** <https://github.com/fehrnic1/RatingRecords>

## 1. Einleitung

*[Kurze Beschreibung der Anwendung: Name der App, Idee, Grundfunktionen. Maximal eine halbe Seite.]*

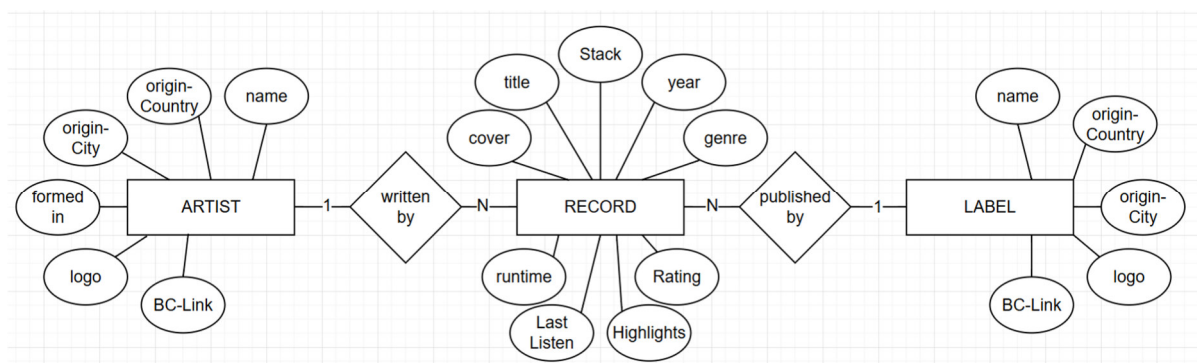
Auf «RatingRecords» kann der User seine Sammlung von physischen Musikmedien wie CDs oder Vinyl-Platten erfassen. Der User hat dabei die Möglichkeit das Album selbst, wie auch den Künstler und das Label, durch welche das Album veröffentlicht wurde, in die Datenbank aufzunehmen. Zudem kann dem Album ein Rating erteilt werden und die persönlichen Highlights des Users, wie auch das Datum, an welchem das Album zum letzten Mal gehört wurde, können hinterlegt werden.

Die Struktur des Anwendung ermöglicht es dem User seine Sammlung entweder über die Alben selbst, oder über die Künstler oder Labels einzusehen. So eröffnet sich die Möglichkeit eine Sammlung dynamisch zu strukturieren, also über die Entitäten Album, Künstler oder Label, was bei der physischen Sammlung selbst nur nach einer dieser Optionen möglich ist.

Zudem kann der User Alben zu seinem Stack hinzufügen, welcher Alben beinhaltet, die er sich als nächstes anhören will und Alben die er entweder schon lange nicht mehr gehört hat oder sehr gut bewertet hat werden ihm auf der Welcome-Page angezeigt.

## 2. Datenmodell

*[ER-Diagramm (mit Erläuterungen, sofern nicht selbsterklärend)]*



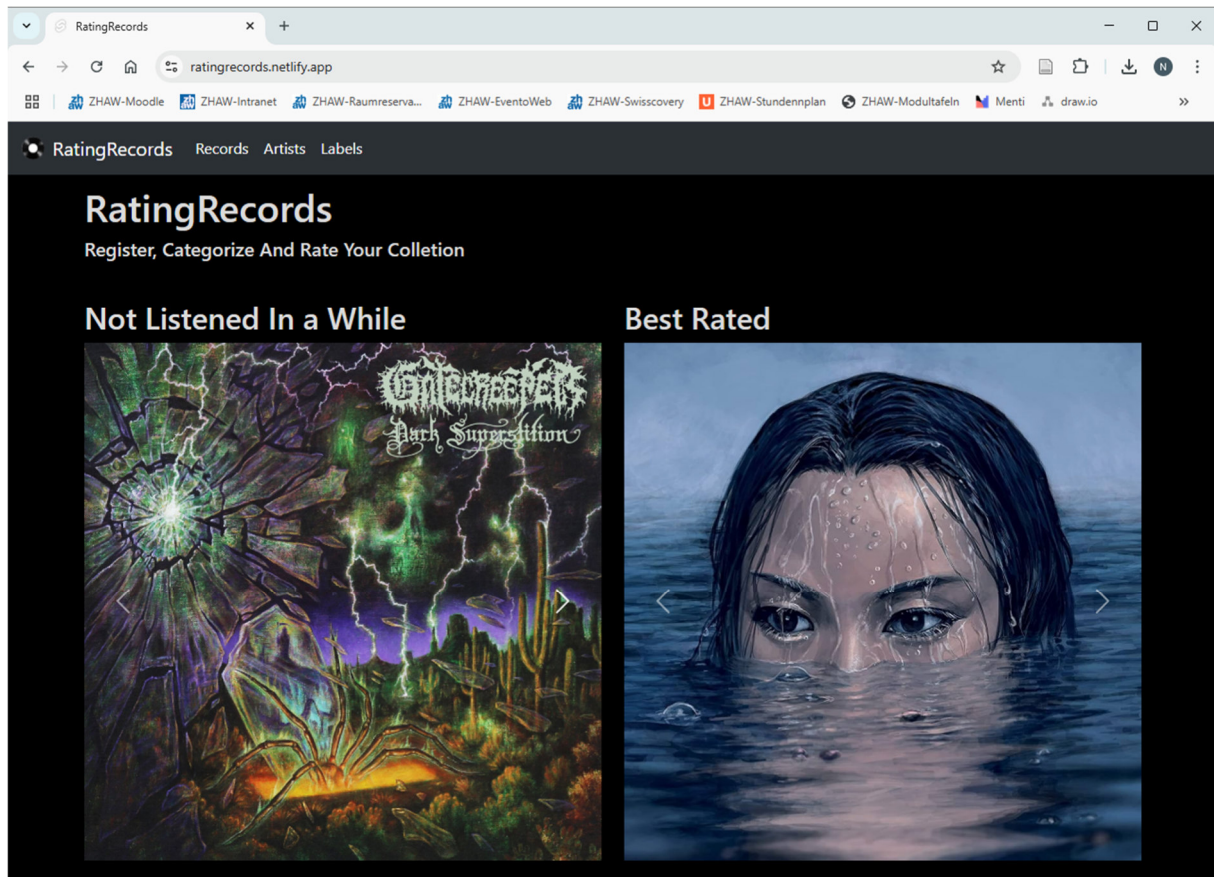
### 3. Beschreibung der Anwendung

[Beschreibung der einzelnen Pages und Funktionen inklusive Screenshots. Die Funktionalität und die Workflows müssen anhand der Screenshots und Textbeschreibungen nachvollziehbar sein. Der Code muss nicht beschrieben werden.]

**Disclaimer:** Die Dateien «styles.css», «app.html» und «+layout.svelte» finden in allen Funktionen Anwendung und werden somit nicht bei jeder der folgenden Kapitel aufgeführt.

#### 3.1. Welcome Page

**Route:** /routes



Auf der Welcome-Page (RatingRecords im NavBar) wird dem User über eine Carousel-Ansicht angezeigt, welche Alben er seit über einem Jahr nicht mehr gehört hat (Not Listened In A While) und welche Alben von ihm mit fünf Sternen Bewertet wurden (Best Rated).

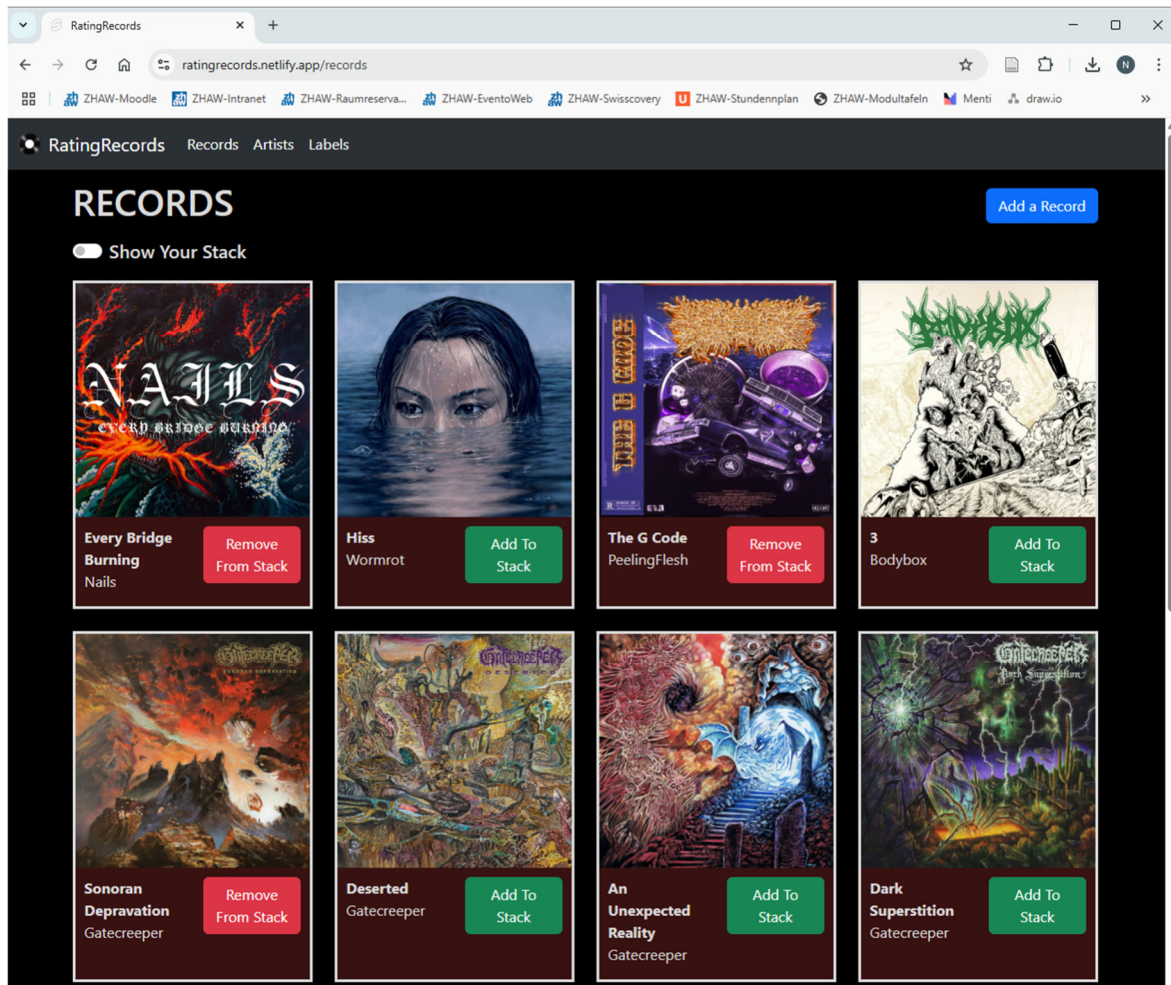
Die Covers können geklickt werden, womit der User zur Detailansicht des Albums gelangt (siehe Kapitel 3.3). Die genauer Funktion der Carousels wird in einem späteren Kapitel beschrieben (siehe Kapitel 4.1).

**Dateien:**

- /routes/+page.svelte
- /routes/+page.server.js

### 3.2. Records-Page

**Route:** /routes/records



Auf der Records-Page werden dem User alle seine Alben angezeigt mit Cover, Titel und Künstler. Zudem kann ein Album mit dem Stack-Button (Add/Remove) zu seinem Stack, also seiner Liste von Alben, die er als nächstes hören will, hinzufügen. Durch Aktivieren des Toggle-Switches «Show your Stack» werden Alben die nicht auf dem Stack sind ausgeblendet. Weiter Informationen zum Stackfilter werden in einem späteren Kapitel gegeben (siehe Kapitel 4.4).

Via Klick auf das Cover oder den Titel gelang der User zur Detail-Ansicht des Albums (siehe Kapitle 3.3).

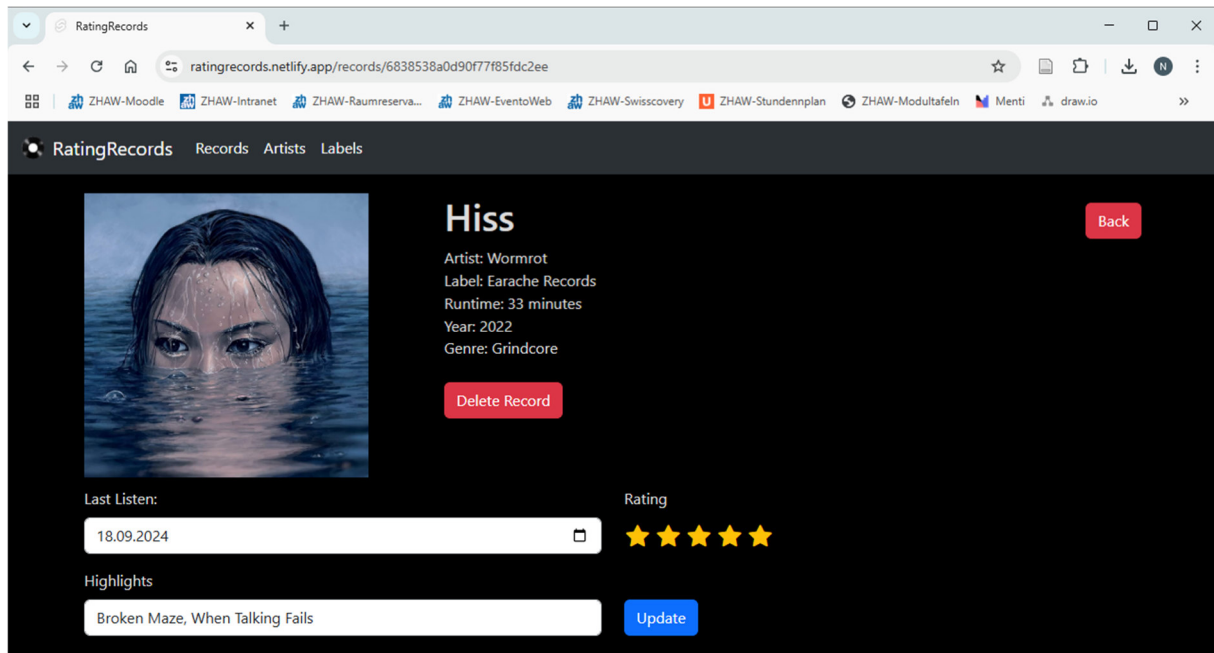
Mit dem Button «Add a Record» gelangt der User zur Anlage-Ansicht für neue Alben (siehe Kapitel 3.4).

#### Dateien:

- /routes/records/+page.svelte
- /routes/records/+page.server.js
- /lib/components/RecordCard.svelte

### 3.3. Record-Detailpage

**Route:** /records/[record\_id]



In der Detail-Ansicht eines Albums werden zusammen mit dem Cover genauere Informationen zum Album angezeigt. Diese können zwischen Stammdaten und Bewegungsdaten unterschieden werden. Im oberen Bereich werden die Stammdaten angezeigt wie zum Beispiel Titel, Künstler oder Genre. Zudem kann ein Album mit dem Button «Delete Record» aus der Sammlung entfernt werden, was über die Funktion `deleteRecord()` im File `/lib/db.js` ermöglicht wird. Bei Betätigung des Buttons wird der User zurück auf die Records-Page geleitet.

Bei den Bewegungsdaten handelt es sich um das Datum, wann das Album zuletzt gehört wurde, was die Highlights des Users sind und wie er das Album bewertet. Die Angaben können in diesem Formular verändert und mit dem Update-Button in der Datenbank angepasst werden. Ein entsprechender Hinweis bestätigt die Anpassung.

Mit dem Button «Delete Record» kann das Album aus der Datenbank entfernt werden, sollte der User seine physische Kopie davon verkaufen.

Über den Back-Button gelangt der User zurück zur Records-Übersicht (siehe Kapitel 3.2).

#### Dateien:

- `/routes/records/[record_id]/+page.svelte`
- `/routes/records/[record_id]/+page.server.js`
- `/lib/db.js`

### 3.4. Record-Createpage

**Route:** /records/create

The screenshot shows a web browser window with the URL `ratingrecords.netlify.app/records/create`. The browser's address bar and tabs are visible at the top. Below the browser window, the application's header shows the logo 'RatingRecords' and navigation links for 'Records', 'Artists', and 'Labels'. The main content area is titled 'Add a Record' and features a dark background with white text and input fields. The form is organized into two columns. The left column contains fields for 'Artist' (a dropdown menu with 'Choose Artist' selected), 'Title' (a text input), 'Runtime' (a text input), 'Last Listen' (a date input showing 'tt.mm.jjjj' with a calendar icon), and 'Rating' (a five-star rating system with all stars selected). The right column contains fields for 'Label' (a dropdown menu with 'Choose Label' selected), 'Year' (a text input), 'Genre' (a text input), and 'Highlights' (a text input). A red 'Back' button is located in the top right corner of the form area, and a blue 'Add Record' button is at the bottom right.

In dieser Ansicht kann der User ein neues Album erfassen. Die Auswahl der Daten erfolgt über verschiedene Input-Methoden wie zum Beispiel das «Last Listen»-Datum via Kalender-Ansicht durch einen Klick auf das Kalender-Icon oder das Rating in Form eines Five-Star-Rating Systems. Title, Genre und Highlights verlangen nach einem Text-Input, Year und Runtime nach einem Zahlen-Input.

«Artist» und «Label» werden via Drop-Down Menü ausgewählt. Künstler und Labels, die zu diesem Zeitpunkt noch nicht erfasst und somit nicht im Drop-Down Menü angezeigt werden, müssten vor der Album-Anlage angelegt werden (siehe Kapitel 3.7 und 3.10).

Das Album kann mit dem «Add Record»-Button in die Datenbank übertragen werden. Ein entsprechender Hinweis bestätigt die Aufnahme in die Datenbank. Die Anlage erfolgt durch die Funktion `createRecord()` in der Datei `/lib/db.js`.

Über den Back-Button gelangt der User zurück zur Records-Übersicht (siehe Kapitel 3.2).

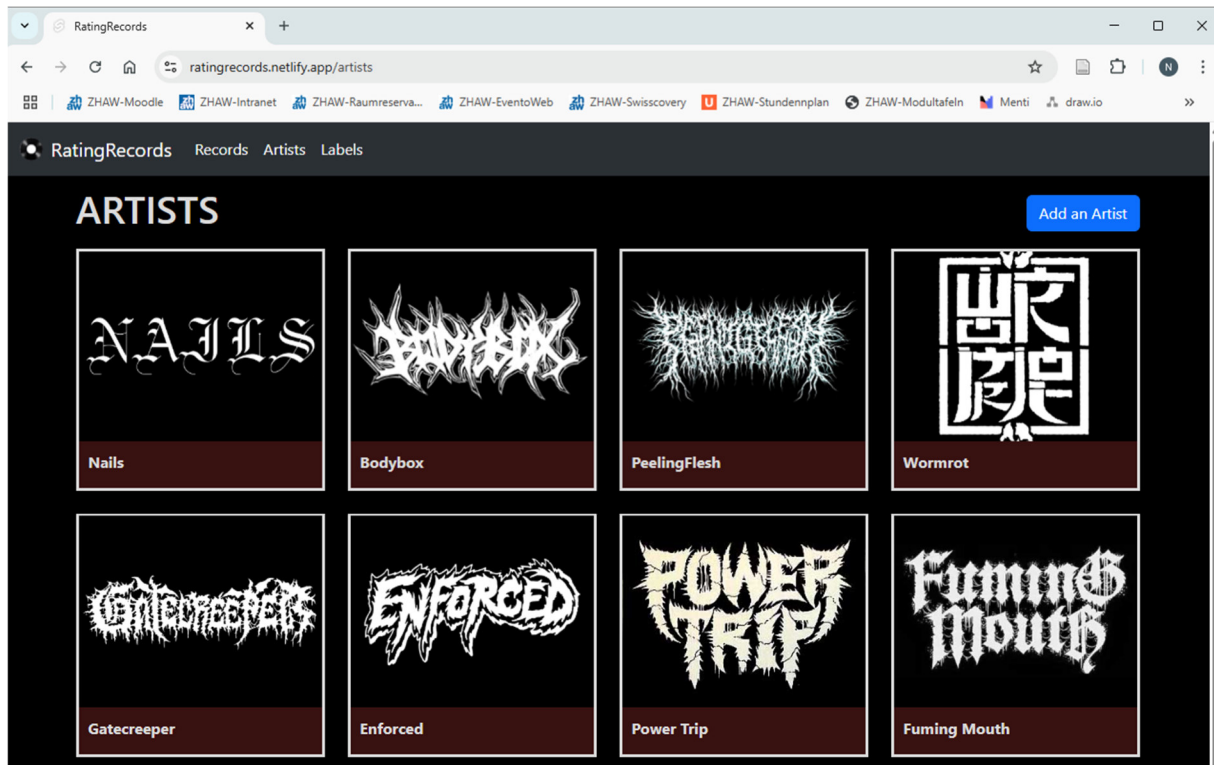
#### Dateien:

- `/routes/records/create/+page.svelte`
- `/routes/records/create/+page.server.js`
- `/lib/db.js`



### 3.5. Artists-Page

**Route:** /routes/artists



Auf der Artist-Page werden dem User alle Künstler angezeigt, von denen er ein Album erfasst hat mit Logo beziehungsweise Schriftzug und Name des Künstlers.

Via Klick auf das Logo oder den Namen gelang der User zur Detail-Ansicht des Künstlers (siehe Kapitel 3.6).

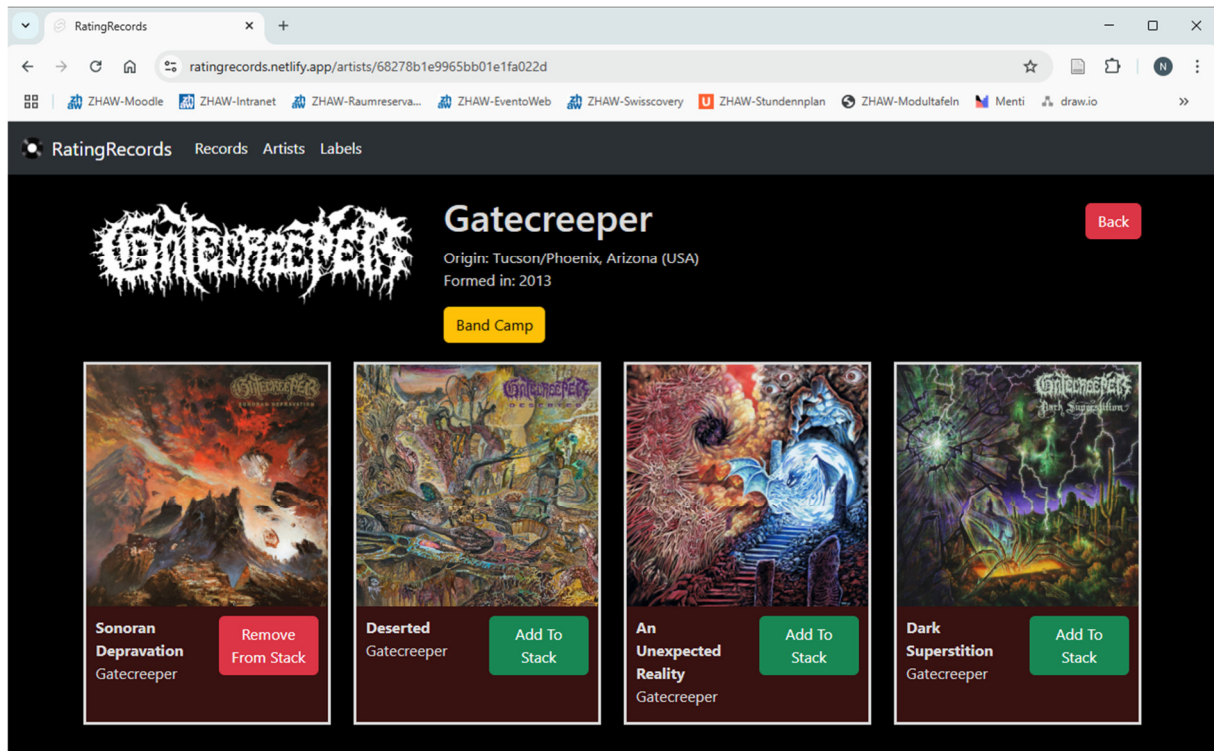
Mit dem Button «Add an Artist» gelangt der User zur Anlage-Ansicht für neue Künstler (siehe Kapitel 3.7).

#### Dateien:

- /routes/artists/+page.svelte
- /routes/artists/+page.server.js
- /lib/components/ArtistCard.svelte

### 3.6. Artist-Detailpage

**Route:** /routes/artists/[artist\_id]



In der Detail-Ansicht eines Künstlers werden zusammen mit dem Logo und Name zusätzliche angezeigt woher dieser stammt und seit wann dieser aktiv ist. Hierbei handelt es sich ausschliesslich um Stammdaten, die nicht verändert werden können. Zudem gelangt der User über den Button «Band Camp» auf die Band Camp Seite der Band, diese wird in einem neuen Tab geöffnet.

Unterhalb der Stammdaten werden dem User alle Alben des Künstlers angezeigt, die er besitzt, was durch die Funktion `getRecordsByArtist()` im File `/lib/db.js` erreicht wird (mehr dazu im Kapitel 4.3). Die Alben werden im selben Format angezeigt wie auf der Records-Page (siehe Kapitel 3.2). Somit können Album-Details auch in dieser Ansicht aufgerufen werden und Alben können zum Stack hinzugefügt werden.

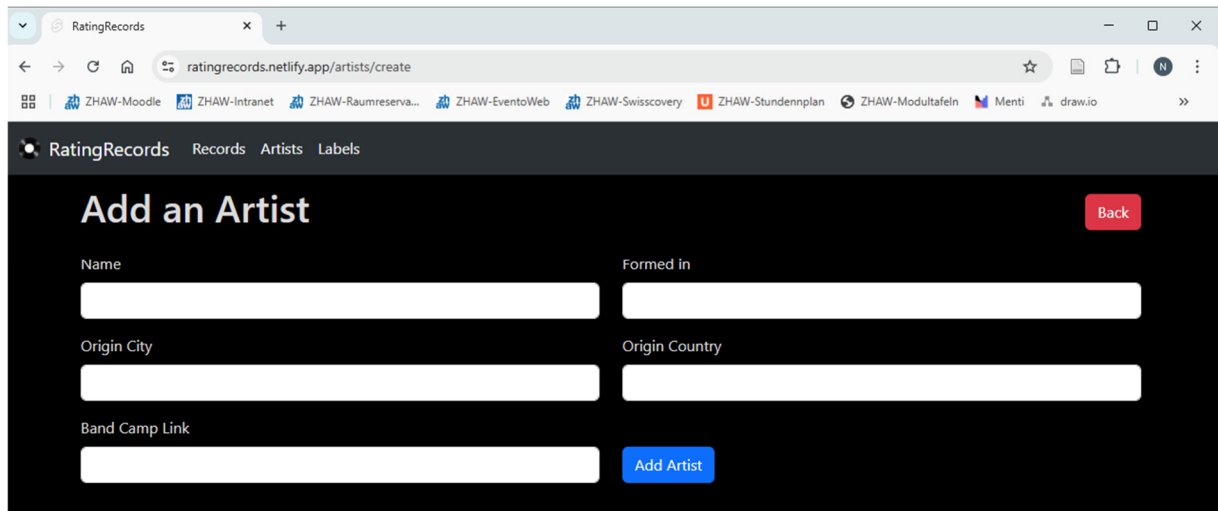
Über den Back-Button gelangt der User zurück zur Artists-Page (siehe Kapitel 3.5).

#### Dateien:

- /routes/artists/[artist\_id]/+page.svelte
- /routes/artists/[artist\_id]/+page.server.js
- /lib/components/RecordCard.svelte
- /lib/db.js

### 3.7. Artist-Createpage

**Route:** /routes/artists/create



The screenshot shows a web browser window with the URL `ratingrecords.netlify.app/artists/create`. The browser's address bar and tabs are visible at the top. Below the browser window, the application's navigation bar shows 'RatingRecords' and links to 'Records', 'Artists', and 'Labels'. The main content area is titled 'Add an Artist' and features a dark background with white text and input fields. The form includes five input fields: 'Name', 'Formed In', 'Origin City', 'Origin Country', and 'Band Camp Link'. A red 'Back' button is located in the top right corner, and a blue 'Add Artist' button is at the bottom right of the form.

In dieser Ansicht kann der User einen neuen Künstler erfassen. Ausser das Feld «Formed in», welches einen Zahl-Input verlangt, werden in allen Feldern einen Text-Input benötigt. Diese einmalige Anlage eines Künstlers wird vorausgesetzt für die Anlage eines neuen Albums.

Der Künstler kann mit dem «Add Artist»-Button in die Datenbank übertragen werden, diese gelingt durch Aufruf der Funktion `createArtist()` im File `/lib/db.js`. Ein entsprechender Hinweis bestätigt die Aufnahme in die Datenbank.

Über den Back-Button gelangt der User zurück zur Artists-Übersicht (siehe Kapitel 3.5).

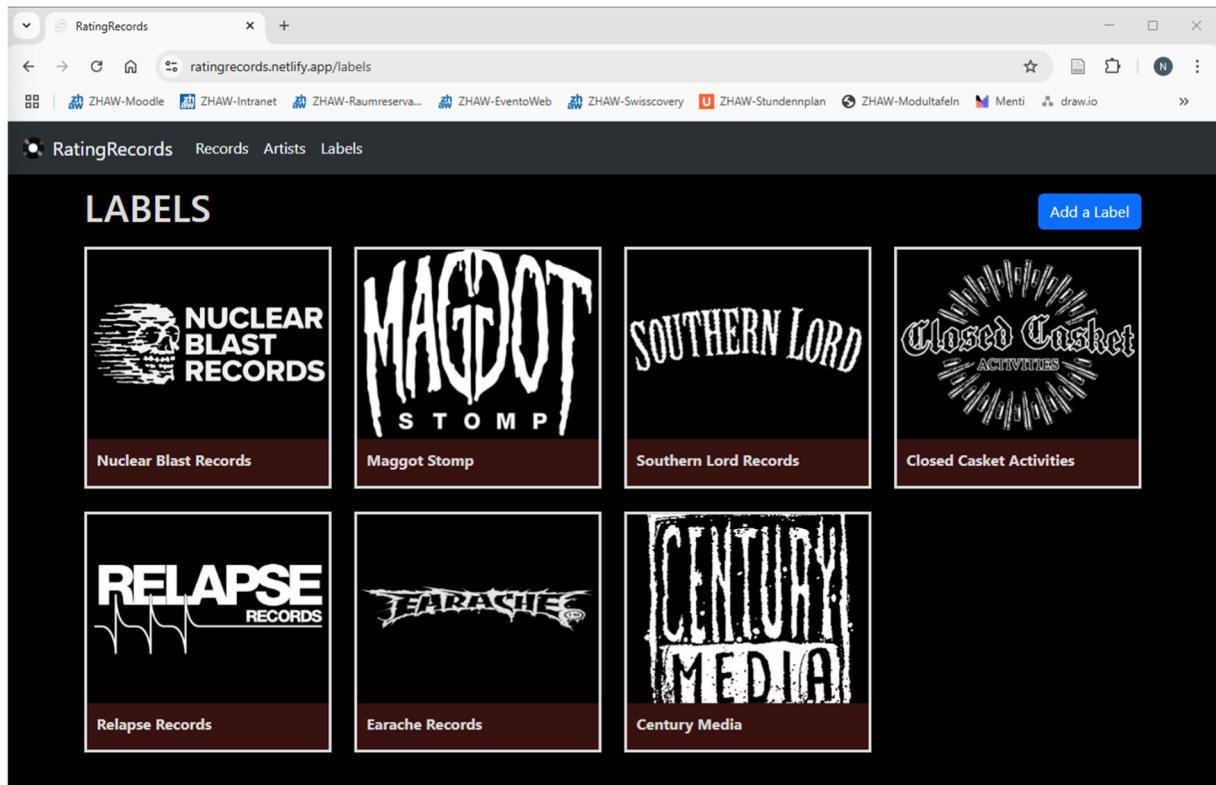
#### Dateien:

- `/routes/artists/create/+page.svelte`
- `/routes/artists/create/+page.server.js`
- `/lib/db.js`



### 3.8. Labels-Page

**Route:** /routes/labels



Auf der Label-Page werden dem User alle Labels angezeigt, von denen er ein Album erfasst hat mit Logo beziehungsweise Schriftzug und Name des Labels.

Via Klick auf das Logo oder den Namen gelangt der User zur Detail-Ansicht des Labels (siehe Kapitel 3.9).

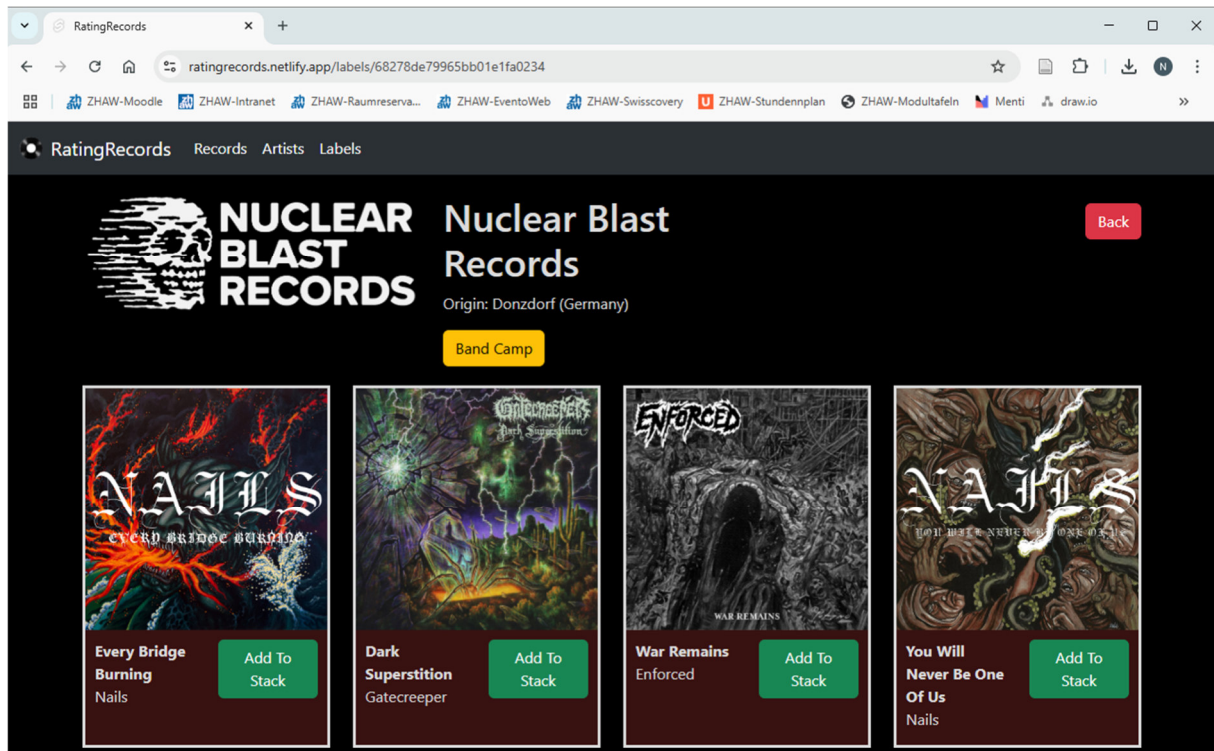
Mit dem Button «Add a Label» gelangt der User zur Anlage-Ansicht für neue Labels (siehe Kapitel 3.10).

#### Dateien:

- /routes/labels/+page.svelte
- /routes/labels/+page.server.js
- /lib/components/LabelCard.svelte

### 3.9. Label-Detailpage

**Route:** /routes/labels/[label\_id]



In der Detail-Ansicht eines Labels werden zusammen mit dem Logo und Name zusätzliche angezeigt woher dieses stammt. Hierbei handelt es sich ausschliesslich um Stammdaten, die nicht verändert werden können. Zudem gelangt der User über den Button «Band Camp» auf die Band Camp Seite des Labels, diese wird in einem neuen Tab geöffnet.

Unterhalb der Stammdaten werden dem User alle Alben des Labels angezeigt, die er besitzt, was durch die Funktion `getRecordsByLabel()` im File `/lib/db.js` erreicht wird (mehr dazu im Kapitel 4.3). Die Alben werden im selben Format angezeigt wie auf der Records-Page (siehe Kapitel 3.2). Somit können Album-Details auch in dieser Ansicht aufgerufen werden und Alben können zum Stack hinzugefügt werden.

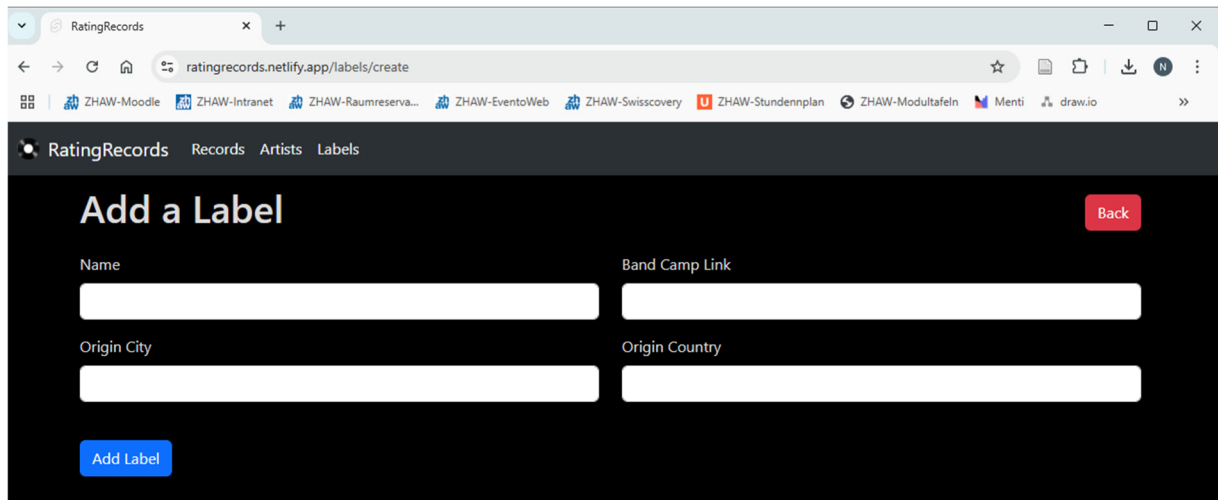
Über den Back-Button gelangt der User zurück zur Labels-Übersicht (siehe Kapitel 3.8).

#### Dateien:

- /routes/labels/[label\_id]/+page.svelte
- /routes/labels/[label\_id]/+page.server.js
- /lib/components/RecordCard.svelte
- /lib/db.js

### 3.10. Label-Createpage

**Route:** /routes/labels/create



The screenshot shows a web browser window with the URL `ratingrecords.netlify.app/labels/create`. The browser's address bar and tabs are visible at the top. Below the browser window, the application's navigation bar shows 'RatingRecords' as the active page, with 'Records', 'Artists', and 'Labels' as other options. The main content area is titled 'Add a Label' and features a dark background with white text and input fields. There are four input fields arranged in a 2x2 grid: 'Name', 'Band Camp Link', 'Origin City', and 'Origin Country'. A blue 'Add Label' button is located at the bottom left, and a red 'Back' button is at the top right.

In dieser Ansicht kann der User ein neues Label erfassen. In allen Feldern wird nach einem Text-Input verlangt. Diese einmalige Anlage eines Labels wird vorausgesetzt für die Anlage eines neuen Albums.

Das Label kann mit dem «Add Label»-Button in die Datenbank übertragen werden, diese gelingt durch Aufruf der Funktion `createLabel()` im File `/lib/db.js`. Ein entsprechender Hinweis bestätigt die Aufnahme in die Datenbank.

Über den Back-Button gelangt der User zurück zur Labels-Übersicht (siehe Kapitel 3.8).

#### Dateien:

- `/routes/labels/create/+page.svelte`
- `/routes/labels/create/+page.server.js`
- `/lib/db.js`

## 4. Erweiterungen

*[Liste der Erweiterungen, die über die Grundanforderungen hinausgehen, siehe Punkt «Erweiterungen» in der Aufgabenstellung. Achten Sie auf Vollständigkeit. **Es werden nur die hier beschriebenen Erweiterungen bewertet.** Der Code muss grundsätzlich nicht beschrieben werden, aber es sollten Hinweise gegeben werden, wo die Erweiterung implementiert wurde (Dateiname, Funktionsname, etc.). Screenshots sind nur nötig, falls die Erweiterung nicht unter «Beschreibung der Anwendung» beschrieben wurde.]*

### 4.1. Welcome-Page Carousels

Die Welcome-Page zeigt in Form von Carousels an, welche Alben der User schon lange nicht mehr gehört hat (Not Listened In a While) und welche Alben von ihm mit fünf Sternen bewertet wurden (Best Rated). Durch klicken des Albumcovers im Carousel gelang der User zur Detailansicht des Albums. Beide Carousels erhalten ihre Datengrundlage von Funktionen, welche zutreffende Alben via MongoDB-Query aus der Datenbank extrahieren. Die Funktionen sind Teil der Datei lib/db.js.

**Carousel:**

Not listened in a while  
Best rated

**Funktion:**

getOldRecords()  
getBestRecords()

**Datei(en):**

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

## 4.2. Star-Rating von Records

Ein Album kann über ein Star-Rating-Input bewertet werden. Die Bewertung kann bei der Anlage definiert und in der Detail-Ansicht des Albums angepasst werden. Durch die Auswahl eines Sterns wird der korrespondierende Wert als Attribut «rating» hinterlegt und über die Funktionen `createRecord()` respektive `updateRecord()` in der Datei `src.lib/db.js` in die Datenbankübertragen.

Pro Stern wird eine Radio-Checkbox verwendet, welcher versteckt wird. Angezeigt wird lediglich das Label des Radios in Form eines Sterns. Die Darstellung des Sterns wird über ein separates Stylesheet in der Datei `app.html` importiert, dass nicht teil des Bootstrap Stylesheets ist über die Klasse «`bi-star-fill`».

In der Datei `/routes/styles.css` werden zusätzlich folgende Klassen zur Darstellung der Sterne definiert:

Klasse	Funktion
<code>.star-rating{}</code>	Definiert in welche Richtung die Sterne befüllt werden Positionierung der Sterne Mauszeiger bei Interaktion
<code>.star-rating input{}</code>	(Wird über die Klasse « <code>.star-rating{}</code> » direkt für Inputs angewendet) Versteckt das eigentliche Input-Format, in diesem Fall der Radio
<code>.star-rating label{}</code>	(Wird über die Klasse « <code>.star-rating{}</code> » direkt für Labels angewendet) Definiert Basis-Farbe, Position und Mauszeiger bei Interaktion
<code>.star-rating label:hover,</code> <code>.star-rating label:hover~label,</code> <code>.star-rating input:checked~label</code> <code>{}</code>	(Wird über die Klasse « <code>.star-rating{}</code> » direkt angewendet bei Interaktion mit dem Mauszeiger) Definiert Farbe bei Auswahl und Interaktion mit dem Mauszeiger wenn Mauszeiger den Stern berührt oder auswählt. Zudem werden gemäss Reihenfolge vorhergehende Sterne gleich eingefärbt.

### Datei(en):

- `/routes/records/create/+page.svelte`
- `/routes/records/create/+page.server.js`
- `/routes/records/[record_id]/+page.svelte`
- `/routes/records/[record_id]/+page.server.js`
- `/routes/styles.css`
- `/lib/db.js`
- `/app.html`



#### 4.3. Verschachtelte Detail-Cards

Die Detailansicht des Künstlers und des Labels bietet eine Übersicht über alle Alben, die vom jeweiligen Künstler bzw. Label veröffentlicht wurden. Diese Record-Cards bieten dieselbe Funktionalität wie diejenigen in der Records-Übersicht. Die Alben des jeweiligen Künstlers beziehungsweise Labels werden über die Funktionen `getRecordsByArtist()` und `getRecordsByLabel()` durch ein MongoDB Query aus der Datenbank ausgelesen und über eine `each`-Funktion als `RecordCard` dargestellt.

##### **Datei(en):**

- `/routes/artists/[artist_id]/+page.svelte`
- `/routes/artists/[artist_id]/+page.server.js`
- `/routes/labels/[label_id]/+page.svelte`
- `/routes/labels/[label_id]/+page.server.js`
- `/routes/styles.css`
- `/lib/components/RecordCard.svelte`
- `/lib/db.js`
- `/app.html`

#### 4.4. Stack-Filter

Jedes Album verfügt in allen Übersichten, in denen es angezeigt werden kann über einen Button, durch dessen Betätigung der Stack-Status gewechselt werden kann und somit dem Stack hinzugefügt oder von diesem entfernt werden kann.

Auf der Seite `/records` können via Toggle-Switch Alben ausgeblendet werden, die nicht auf dem Stack liegen. Dies wurde mit einem Filter realisiert, der via `derived`-Funktion prüft, ob ein Album auf dem Stack liegt oder nicht und je nach Zustand des Toggle-Switches angezeigt werden soll oder nicht.

##### **Datei(en):**

- `/routes/records/+page.svelte`
- `/routes/records/+page.server.js`
- `/routes/artists/[artist_id]/+page.svelte`
- `/routes/artists/[artist_id]/+page.server.js`
- `/routes/labels/[label_id]/+page.svelte`
- `/routes/labels/[label_id]/+page.server.js`