OCTOTUNE

OCTOTUNE

EINE PLATTFORM, UM MUSIK ZU HÖREN

CEO: RENE STERNITZKY
BETREUER: CHRISTOF BEHNEN
KLASSE: ITA 23-1
PROJEKTZEITRAUM: 20.12 – 14.02

Inhalt

Aufgabenstellung

Einleitung: Was ist OctoTune?

„OctuTune“ ist eine Web-Anwendung, auf der Benutzer sich registrieren können. Anschließend erhalten die Benutzer Zugang zu verschiedenen Liedern. Außerdem können Playlisten erstellt werden, in denen man Lieder hinzufügen kann. Es soll im Endprodukt konkurrenzfähig zu anderen Streaminganbietern, wie z. B. Spotify oder Deezer, sein. Das Projekt verläuft im Rahmen meines 100-Stunden-Abschlussprojekts zum Abschluss des Informationstechnischen Assistenten und wird von Christof Behnen betreut.

Weshalb OctoTune?

Es gibt eine große Anzahl verschiedener Plattformen, auf denen man Zugang zu Musik hat. Jedoch sind diese oft kostenpflichtig, setzen auf ein Abo-Modell oder spielen Werbung zwischen den Liedern ab. Teilweise kann man nicht mal entscheiden, welches Lied man wirklich hört, um Nutzer in eine Abo-Falle zu locken. Deshalb habe ich mich dazu entschieden eine eigene Plattform zu entwickeln, auf der Nutzer genau das hören können, was sie möchten und nicht mit Werbung genervt werden.

Welche Funktionen muss OctoTune bieten?

1. Login und Registrierung
2. Lieder können abgespielt werden
3. Es können Playlisten erstellt und frei benannt werden
4. Lieder und Künstler können gesucht werden
5. Das Passwort wird verschlüsselt gesichert
6. Alle Daten sollen in einer relationalen Datenbank gesichert werden

Gibt’s Sachen, die nicht unbedingt implementiert werden müssen?

Ja.

1. Eine optisch ansprechende Weboberfläche, die auch funktional ist
2. Nutzer können zwischen verschiedenen Audioqualitäten wählen
3. Label-Accounts, die Künstler verwalten und Lieder hochladen können
4. Ein Programm, welches große Mengen an Liedern automatisch in die Datenbank einfügt
5. Playlists können privat gehalten oder veröffentlicht und geteilt werden

Nicht-so-funktionale Anforderungen

Zielgruppe:

Die Zielgruppe ist offensichtlich jeder, der gerne Musik hört und sich sein Geld nicht von dubiosen Anbietern - wie Spotify - aus der Tasche ziehen lassen wollen.

Entwicklungsumgebung:

- IDEs[1]: PhpStorm / WebStorm von JetBrains, VSCode
- Programmiersprachen: HTML[2], CSS[3], JavaScript, PHP[4], SQL[5]
- KIs[6]: Suno[7], VSCode Copilot Extension[8]

Risikoakzeptanz:

- Mögliche Angriffe auf die Datenbank
- „Erfahrene“ Webnutzer könnten Lieder herunterladen

Anwendungsbereich:

Die Webplattform wäre über dem Browser erreichbar, kann also an jedem Gerät mit Internetzugang über einen Internetbrowser genutzt werden.

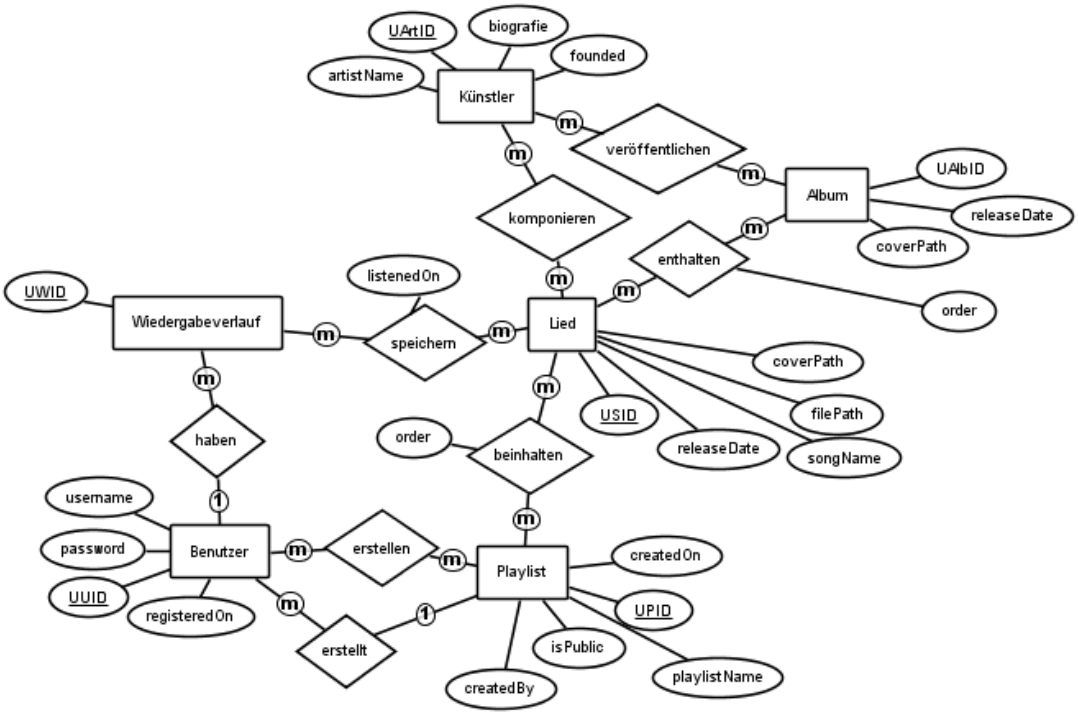
[1] Integrierte Entwicklungsumgebungen
[2] Hypertext Markup Language
[3] Cascading Style Sheets
[4] Personal Home Page Tools
[5] Structured Query Language
[6] Künstliche Intelligenzen
[7] Music-generative-AI
[8] Macht nette Codevorschläge, die sowieso immer angepasst werden müssen

Beschreibung des Projektablaufs

Das ER-Modell[1] und das SQL-Script[2]

Zu Beginn habe ich erstmal das ER-Diagramm für die Datenbank erstellt. Dazu habe ich Xerdi verwendet und ich habe probiert den SQL-Code zur Generierung der Datenbank von Xerdi generieren zu lassen. Dies funktioniert recht gut, jedoch müssen Datentypen nachträglich im Code angepasst werden, da Xerdi alle Inhalte mit dem „CHAR(1)"-Datentyp erzeugt. Das ER-Modell, welches ich erstellt hatte, war recht fehlerfrei. Ein Fehler war zum Beispiel, dass „mehrere Benutzer mehrere Playlisten erstellen", „eine Playlist kann aber nur von einem Benutzer erstellt werden." Durch diesen Fehler wäre es nun möglich, dass mehrere Benutzer dieselbe Playlist bearbeiten könnten, was so eigentlich nicht vorgesehen war. Ich habe dies jedoch bewusst so drin gelassen, da dies eigentlich eine Wünschenswerte Funktion ist. Der Ersteller der Playlist bleibt dennoch identifizierbar und ich habe das ER-Modell entsprechend korrigiert.

Das finale ER-Modell sieht so aus:



Zwischen „Benutzer" und „Playlist" gibt es die Beziehung „erstellen", welche durch den oben genannten Fehler zustande kommt. Diese Beziehungstabelle beschreibt nun, wer eine Playlist bearbeiten darf. Die „erstellt" Beziehung bezeichnet den Ersteller der Playlist und ist unter „createdBy" in der „Playlist"-Tabelle hinterlegt.

[1] Entity-Relationship-Modell: zeigt Beziehungen innerhalb einer Datenbank
[2] Mithilfe des SQL-Scripts lässt sich die Datenbank erstellen

Welche Entitäten, Attribute und Beziehungen gibt es?

Zunächst muss geklärt werden, was diese Begriffe bedeuten:

- **Entität:** Ein Objekt in einer Datenbank, diese bekommen ihre eigenen Tabellen
- **Attribute:** Eigenschaften eines Objekts, wäre ein Objekt ein Benutzer, könnte ein Attribut der Benutzername sein
- **Beziehungen:** Diese beschreiben, wie die Objekte voneinander abhängig sind

Entitäten	Attribute	Beziehungen
Benutzer	• Benutzername • Passwort (als Hash) • UUID (Primärschlüssel) • Registrationsdatum	• Hat ein Wiedergabeverlauf • Können mehrere Playlisten erstellen • Können mehrere Playlisten bearbeiten
Playlist	• UPID (Primärschlüssel) • Playlistname • Erstelldatum • Erstellt von (Fremdschlüssel) • Ist sie öffentlich?	• Können von einem Benutzer erstellt werden • Können von mehreren Nutzern bearbeitet werden • Beinhalten mehrere Lieder (wobei die Reihenfolge gespeichert wird)
Wiedergabeverlauf	• UWID (Primärschlüssel • UUID (Fremdschlüssel)	• Wird von einem Benutzer besessen • Speichern welche Lieder gehört werden dazu wird die Uhrzeit und das Datum gespeichert
Lied	• USID (Primärschlüssel) • Veröffentlichungsdatum • Song Name • Pfad zur Datei des Liedes • Pfad zum Cover des Liedes	• Sind in mehreren Playlisten enthalten • Werden in mehreren Wiedergabelisten gespeichert • Werden von mehreren Künstlern komponiert • Sind in mehreren Alben enthalten (Reihenfolge der Lieder wird beachtet)
Album	• UAlbID (Primärschlüssel) • Pfad zum Cover des Albums • Albumname • Veröffentlichungsdatum	• Enthalten mehrere Lieder • Werden von mehreren Künstlern veröffentlicht
Künstler	• UArtID (Primärschlüssel) • Künstlername • Biografie • Gründungsjahr	• Veröffentlichen mehrere Alben • Komponieren mehrere Lieder

Wie werden Lieder und andere Daten in die Datenbank eingepflegt?

Eine Kann-Kriterie ist ein Programm, welches Lieder, Künstler, Alben und mehr in die Datenbank einpflegt. Damit habe ich jedoch schon nach der Fertigstellung der Datenbank begonnen. Für dieses Programm habe ich die Programmiersprache Python verwendet. Das Programm beginnt erstmal damit alle Tabellen zu leeren. Anschließend werden verschiedene Funktionen ausgeführt:

Funktion fill_album_table():

- Stellt Verbindung zur Datenbank her
- Holt sich die Pfade über die get_files()-Funktion
- Alben werden über ein for-loop in ein Array gespeichert
- Das Array wird anschließend in die Album-Tabelle in der Datenbank hinzugefügt

Funktion push_songs(val=get_metadata()):

- Metadaten der Lieder von der get_metadata()-Funktion
- Stellt Verbindung zur Datenbank her
- Lädt Metadaten der Lieder in die Datenbank hoch

Funktion push_artists(artists=get_artist()):

- get_artist()-Funktion holt sich Künstler über Metadaten der Lieder, Duplikate werden hier gefiltert
- Stellt Verbindung zur Datenbank her
- Lädt Künstler in die Datenbank hoch

Funktion getRelations():

- Stellt Verbindung zur Datenbank her
- Alle Lieder und Künstler werden aus der Datenbank geholt
- For-loop überprüft, ob die Künstler eine Verbindung mit den Liedern haben, dies geschieht ebenfalls mithilfe der Metadaten
- Besteht eine Verbindung, wird sie in ein Array hinzugefügt, welches in die Datenbank hochgeladen wird

Funktion fill_song_album_relation():

- Stellt Verbindung zur Datenbank her
- Holt alle Lieder, sowie Alben aus der Datenbank
- Lieder werden in Alben gruppiert, ebenfalls mithilfe der Metadaten
- Beziehungen werden hergestellt und in die Datenbank hochgeladen

Funktion fill_artist_album_relation():

- Stellt Verbindung zur Datenbank her
- Künstler und Alben werden aus der Datenbank genommen
- Beziehungen werden ebenfalls über Metadaten hergestellt
- Beziehungen werden in die Datenbank hochgeladen

Für die Funktion ist es wichtig, dass die Dateien der Lieder alle Metadaten enthalten. Ist das nicht der Fall, kann es sein, dass Lieder nicht angezeigt werden.

Die Metadaten der Lieder werden über eine Python-Bibliothek „TinyTag“ ausgelesen. Außerdem werden noch die Bibliotheken „mysql.connector“, „os“ und „pathlib“ benötigt. Bei wird Funktionen wird ungefähr dieselbe Reihenfolge angewant. Zum Ende jeder Funktion, wird die Verbindung mit dem Datenbankserver geschlossen, weshalb zu beginn jeder Funktion eine neue aufgebaut wird.

Wie werden die Lieder komponiert?

Um jegliche Probleme im Bereich des Kopierschutzes zu meiden, generiere ich die genutzten und bereitgestellten Lieder selbst mithilfe einer Künstlichen Intelligenz „Suno AI“.

Auf der Webseite Suno AI[1] muss man sich zunächst ein Account erstellen. Ist dies erledigt wird man auf die Startseite geleitet. Auf dieser hat man links eine Leiste mit verschiedenen Reitern, darunter auch „Create“. Klickt man darauf, wird man auf die Seite geleitet, bei der man jegliche Bestandteile des Liedes generieren kann. Man kann den Liedtext selbst schreiben und mit diesem ein Lied generieren, jedoch kann man sich auch den Text generieren, was ich mir zunutze gemacht habe. Zusätzlich kann man dem Lied einem Stil hinzufügen. Dies funktioniert über eine Textbox, in der man verschiedene Genres eingeben kann, an die die KI sich richtet. Zum Schluss gibt man einen Titel ein, der auch KI-generiert werden kann und drückt auf „Create“. Das Generieren des finalen Liedes kostet 5 „Credits“, einer Währung auf der Seite, welche man durch ein abgeschlossenes Abonnement erhält. Jedoch gibt es einen kostenlosen „Basic Plan“, der einem täglich 50 Credits gutschreibt. Das Generieren eines Liedes kostet 5 Credits, wodurch man täglich insgesamt 10 Lieder generieren lassen kann.

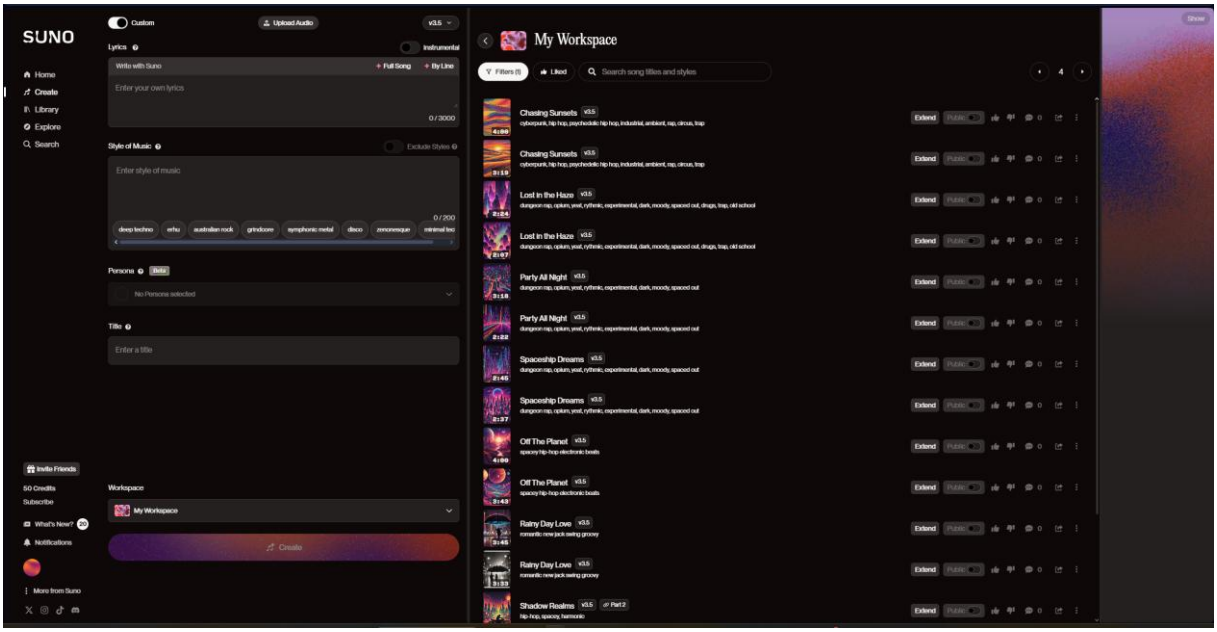


Abbildung 1: "Create"-Bereich von Suno AI

[1] https://suno.com/

Suno AI bietet anschließend die Möglichkeit diese Lieder herunterzuladen.

Auf der Startseite von Suno AI sind zudem andere Lieder von anderen Benutzern zu finden, welche man mit einem Trick ebenfalls herunterladen kann. Dazu muss man den „Netzwerk“-Bereich im Browser überprüfen. Spielt man ein Lied ab, wird dort eine Anfrage angezeigt. Doppelklickt man diese Anfrage, wird man zu der Datei auf dem Suno AI-CDN[1] geleitet, wo man sich das Lied einfach herunterladen kann. Dies wäre auch erlaubt, da KI-generierte Inhalte in der Regel nicht unter Kopierschutz fallen, welches hier[2] beschrieben wird.

Ist ein Lied heruntergeladen, müssen die Metadaten des Liedes angepasst werden. Dazu rechtsklickt man auf die Lieddatei und klickt auf „Eigenschaften“. Anschließend können die Metadaten im Reiter „Details“ angepasst werden.

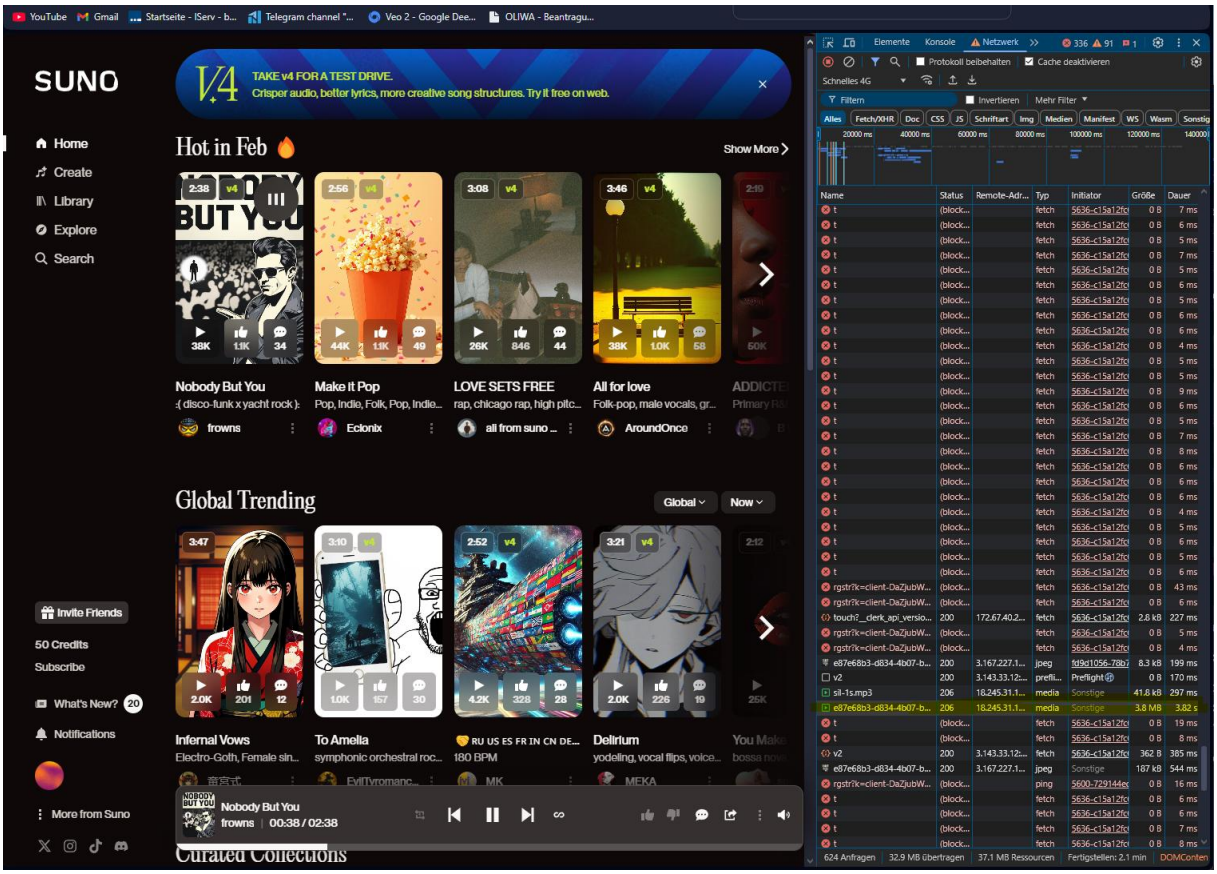


Abbildung 2: Suno AI-Startseite mit Netzwerkfenster und markierter Anfrage zur .mp3-Datei des Liedes

[1] „Content Delivery Network“: „versorgt“ den Nutzer mit Inhalten
[2] https://help.suno.com/en/articles/2746945

Die Login-Seite

Auf der Login Seite wird man „Welcome back to OctoTune!“ begrüßt. Im Zentrum der Seite ist ein Fenster mit der Überschrift „Login“. Darunter zwei Textfelder, eins für den Nutzernamen, eins für das Password. Dazu ein Login-Button und ein Text „Don't have an account? Register“, falls man noch kein Benutzer hat. Dies ist die Index-Seite. Unten auf der Seite steht noch das Jahr in dem OctoTune erschaffen wurde und der Name der Plattform.

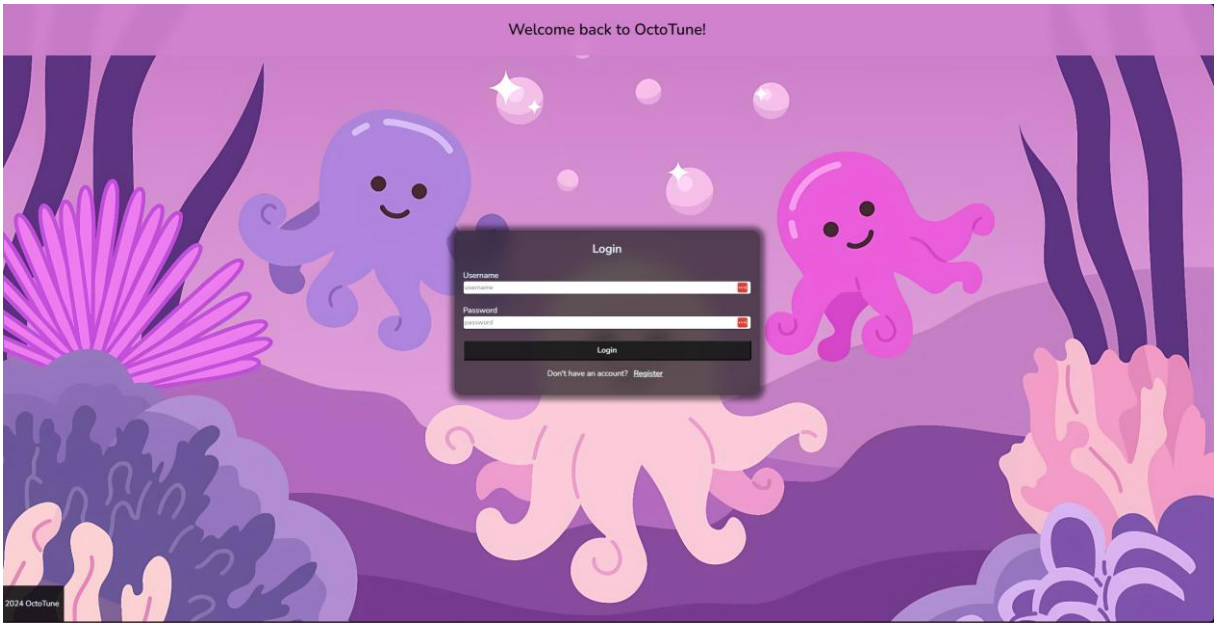


Abbildung 3 Login Seite

Wie funktioniert die Login Seite?

Beim Aufruf der Seite wird eine Verbindung zur Datenbank hergestellt. Dann wird überprüft, ob es bereits ein UUID[1]-Cookie[2] gibt, welcher den Nutzer identifiziert. Gibt es diesen und ist dieser in der Datenbank vorhanden, wird man direkt zur Webapp geleitet. Ist der Cookie nicht in der Datenbank, wird der Cookie entfernt und der Nutzer muss sich einloggen. Der Benutzer gibt im Feld „Username“ den Benutzername und im Feld „Password“ das Passwort ein. Das Passwort wird dabei nicht gezeigt. Wenn der Benutzer dann den Login-Button drückt, wird ein PHP-Teil ausgeführt, der den Benutzer einloggt. Das Password wird dabei mit dem „sha256“-Algorythmus verschlüsselt. Es wird überprüft, ob die eingegebenen Daten, mit denen in der Datenbank übereinstimmen. Wenn ja, wird der UUID-Cookie gesetzt und man wird zur Webapp weitergeleitet. Hat der Benutzer falsche Daten angegeben, wird ihm das mit dem Text „Invalid username or password!“ angezeigt.

Wie ist die Login Seite in HTML aufgebaut?

- Im <head>[3] wird der Seitentitel und Links zur CSS-Datei und zum Favicon [4]definiert

[1] Universally Unique Identifier: identifiziert einen Benutzer eindeutig
[2] Erlaubt die Zwischenspeicherung von Daten im Browser
[3] HTML-Element, dass die Metadaten der Datei enthält
[4] Bild, welches in der Tableiste im Browser angezeigt wird

- Im <body>-Element ist der gesamte Inhalt der Seite und hat ein Hintergrundbild
- Im <header>[1] wird der Willkommenssatz angezeigt und hat eine Hintergrundfarbe
- Im <main>-Element ist ein Formular für den Login
- Jedes <input>[2]-Element hat aus Style-Gründen eine eigene [3], in dem das <input>-Element und ein entsprechendes Label[4] vorhanden ist
- Im <footer>[5] ist das Datum in dem OctoTune entwickelt wurde und der Name des Projekts

Alle Elemente sind entsprechend mit Klassen versehen, um sie entsprechend gestalten zu können. Das Formular verwendet die POST-Methode.

Die Registrier-Seite

Im Grunde ist dies, die Login-Seite allerdings mit einigen kleinen visuellen und funktionellen Unterschieden. Man wird mit „Welcome to OctoTune!“ begrüßt. Das „back“ wurde weggelassen, da der Benutzer vermutlich zum ersten Mal OctoTune besucht. Auch hier ist im Zentrum ein Registrierfenster. Dies hat die Überschrift „Register“, drei Eingabefelder, ein zusätzliches, um das eingegebene Passwort zu bestätigen und ein Registrier-Button. Auch hierunter befindet sich ein Text „Already have an account? Login“, um den Nutzer die Möglichkeit zu geben, sich einzuloggen, falls er bereits ein Account hat. Allerdings wäre es auch möglich mehrere Accounts zu erstellen.

Wie funktioniert die Registrier-Seite?

Der Benutzer hat die Möglichkeit, einen Nutzernamen einzugeben, der später noch geändert werden kann. Zudem muss der Benutzer ein Passwort festlegen und um Rechtschreibfehler zu vermeiden, muss er dies auch bestätigen, also erneut eingeben. Klickt der Nutzer auf den Registrier-Button, wird ein PHP-Teil ausgeführt. Dieser verbindet sich mit der Datenbank, verschlüsselt das Passwort mit dem „sha256“-Algorythmus und generiert eine UUID mithilfe der [uniqid()]-Funktion und überprüft, ob es bereits ein Benutzer mit demselben Benutzernamen gibt. Ist das nicht der Fall, werden die Daten des Benutzers und das aktuelle Datum in die Datenbank eingefügt. Anschließend wird die generierte UUID in einem Cookie gespeichert und der Nutzer wird zur Webapp weitergeleitet. Sollte der gewählte Nutzername schon in Verwendung sein, wird dem Benutzer das mit dem Text „Username already taken!“ signalisiert.

[1] Anfang/Überschrift einer Webseite
[2] Eingabeelemente wie Textfelder, Absendetaste, etc
[3] Unterteilt die Webseite in einzelne Segmente
[4] Beschriftet das entsprechende Textfeld
[5] Ende einer Webseite

Wie ist die Registrier-Seite in HTML aufgebaut?

- Im <head> wird der Seitentitel und Links zur CSS-Datei und zum Favicon definiert
- Im <body>-Element ist der gesamte Inhalt der Seite und hat ein Hintergrundbild
- Im <header> wird der Willkommenssatz angezeigt und hat eine Hintergrundfarbe
- Im <main>-Element ist ein Formular für die Registration
- Jedes <input>-Element hat aus Style-Gründen eine eigene , in dem ein Eingabemodul und ein entsprechendes Label vorhanden ist
- Im <footer> ist das Datum in dem OctoTune entwickelt wurde und der Name des Projekts

Im Grunde ist die Registrier-Seite mit der Login Seite fast identisch, da ich sie kopiert habe. Deshalb sind in der Datei register.php auch Klassen zu finden, die auch auf der Login Seite zu finden sind. Das hat jedoch keinen negativen Einfluss auf die Funktionsweise, es erspart jedoch einige Arbeit, da ich die Seite nicht neu designen muss und stattdessen einfach dasselbe genommen wird.

Das Design

Mein Projekt verwendet HTML, PHP und JS als Programmiersprachen, um die Webseite zu erstellen. HTML dient dabei als Struktur, PHP kommuniziert hauptsächlich nur mit der Datenbank, wird also als sogenanntes Back-End verwendet und JS, was für JavaScript steht, ist in der Lage die HTML-Struktur basierend auf Nutzereingaben zu manipulieren, also zu ändern. Außerdem verwende ich es, um Inhalte dynamisch anzuzeigen. Nur mit diesen Programmiersprachen lässt sich eine Webseite aber nur schlecht gestalten. Dafür wird CSS verwendet. Zuerst wollte ich hierbei auf ein Framework setzen. Ich habe mich jedoch dagegen entschieden, da ich die volle Freiheit von CSS ausnutzen wollte. Zusätzlich ist CSS selbst sehr ausführlich dokumentiert, wodurch ich bei Hindernissen schnell recherchieren kann. Als allgemeines Design habe ich mich für ein Oktopus entschieden, daher auch der Name OctoTune.

Wie funktioniert CSS?

CSS selbst ist relativ einfach, kann aber auch herausfordernd sein. Um ein HTML-Element zu designen, kann man in CSS ein HTML-Element direkt ansprechen oder man gibt den HTML-Elementen eine Klasse oder ID. IDs werden in CSS mit Zeichen „#[idname}“ angesprochen, Klassen mit „.[klassenname]“. Ich selbst habe HTML-Elementen Klassen gegeben, gelegentlich habe ich aber auch direkt die HTML-Elemente angesprochen. CSS bietet auch die Möglichkeit mehrere Elemente gleichzeitig anzusprechen oder auch die „Children“, also untergeordnete Elemente, anzusprechen.

CSS-Code Beispiele:

```css
.primaryButton:hover{
    background-color: rgba(190, 190, 190, 0.9);
    color: black;
}
```

Wenn man mit der Maus über einen Button der „primaryButton“-Klasse schwebt, wird dieser visuell angepasst, um den Nutzer zu zeigen, welchen Button man ausgewählt hat.

```css
.playerbuttons button img{
    height: 30px;
    margin: auto;
    display: flex;
    align-items: center;
    width: 30px;
    filter: invert(90%);
}
```

Hier wird ein Children eines Children bearbeitet, angesprochen werden Bilder, wie der „Play“-Button

Letztendlich ist meine CSS-Datei ~1000 Zeilen lang, weshalb ich sie nicht vollständig erklären werde. Bei dem Schreiben der Datei habe ich auch Copilot verwendet, eine VSCode-Erweiterung, die Codevorschläge und Hilfestellungen geben kann. Diese Vorschläge musste ich aber oft anpassen, teilweise waren sie komplett falsch und unbrauchbar. Sie hat aber letztendlich geholfen, die 1000 Zeilen zu erreichen, da oft redundante Eigenschaften zugewiesen wurden. Jedoch wäre es ein großer Aufwand, die CSS-Datei aufzuräumen, weshalb ich es nicht tun werden. Was ich aber tun kann, ist einige Eigenschaften, die ich häufig verwendet habe, zu erklären

Margin	Schafft Platz um ein Element herum
Padding	Schafft Platz innerhalb eines Elements
Display	Entscheidet in welcher Art der Anordnung Elemente dargestellt werden
Flex-Direction	• Column: Ordnet Elemente untereinander an • Row: Ordnet Elemente nebeneinander
Backdrop-filter	Lässt den hintergrund „verschwimmen“
Overflow	Falls Inhalte nicht in ein Element passen, kann dieser mit overflow gesteuert werden. Bei dem Wert Auto erscheint eine Scrollbar, sobald der Inhalt nicht mehr in das Element passt.
Border	Erstellt einen Rahmen um ein Element
Border-Radius	Rundet die Ecken der Rahmen ab
Box-Shadow	Gibt dem Element einen Schatten
Filter	Dadurch können Filter auf Bilder angewendet werden. Ich nutze es um die schwarzen Symbole weiß darzustellen
Cursor	Ändert den Cursor, wenn man mit der Maus über dem Element ist. Dadurch kann man z. B. ein -Element klickbar machen
Z-Index	Elemente können sich überlappen, Z-Index bestimmt welches Element über das andere ist
Width / Height	Bestimmt wie breit/hoch ein Element sein kann. Man kann dabei auch Mindest- und Maximalwerte festlegen
Background-Color	Hintergrundfarbe eines Elements
Color	Textfarbe in einem Element
Text-Size	Schriftgröße
Font-Weight	Schriftdicke
:hover	Eigenschaften werden nur angewendet, wenn man mit der Maus über dem Element ist
Align-Items	Positioniert Elemente innerhalb eines [display: flex]-Elements

Die App

Welche Funktionen bietet OctoTune?

Auf dieser Seite kann der Benutzer alle Funktionen von OctoTune nutzen. Er kann Lieder hören, Playlisten erstellen, sowie löschen und dabei den Namen frei wählen. Dementsprechend kann der Benutzer auch Lieder zu diesen Playlisten hinzufügen. Der Benutzer ist außerdem in der Lage, Lieder, Playlisten, Künstler und mehr in ein einer Suchleiste suchen. Es gibt auf der linken Seite eine Navigationsleiste, in der man zur Startseite und zum Wiedergabeverlauf kommen kann. Außerdem können hier auch die Playlisten erstellt werden und werden hier angezeigt. Der Benutzer kann die Lieder anhören, pausieren, kann das nächste Lied abspielen oder das vorherige. Außerdem kann man die Lautstärke und den Zeitstempel des Liedes über die Liedleiste unten auf der Seite ändern. Der Benutzer wird oben rechts neben der Suchleiste begrüßt. Wenn der Benutzer auf diese draufklickt, öffnet sich ein Fenster, in dem der Benutzer sich umbenennen und ausloggen kann. Möchte der Benutzer die Plattform nicht mehr nutzen kann er dort auch sein Benutzerkonto löschen.

Wie ist die App in HTML aufgebaut?

- Im <head>-Teil werden Metadaten der Seite angegeben, darunter die CSS-Datei und die HowlerJS-Bibliothek, sowie etliche Definitionen von JavaScript-Funktionen
- Im <body>-Teil gibt es zunächst:
  - ❖ Eine Topleiste, in der das Logo, die Suchleiste und die Benutzerbegrüßung enthalten sind
  - ❖ Ein <main>-Teil, der die Seitenleiste und die Songliste enthält
  - ❖ Ein <footer>-Teil, in dem die Spielleiste enthalten ist.
  - ❖ Zwei Modals, eine für die Benutzereinstellungen, die andere um die Playlist auszuwählen, zu der man ein Lied hinzufügen möchte
- Die Suchleiste ist in weiteren Elementen unterteilt:
  - ❖ Eine Texteingabe zur Suche
  - ❖ eine Bildeingabe[1], um etwas zu suchen (Lieder werden jedoch direkt bei einer Eingabe gesucht)
- Die Seitenleiste ist unterteilt in:
  - ❖ Ein Button, um die Startseite anzuzeigen
  - ❖ Um den Wiedergabeverlauf anzuzeigen
  - ❖ Eine Playlist Unterteilung (-Element), die weiter unterteilt ist
    - Der Überschrift „Playlist“
    - Die Erstellung von Playlisten, wozu ein Textfeld und ein Button verwendet wird
    - Die Liste der erstellen Playlisten

[1] Ein Bild auf das man klicken kann, um eine Funktion auszuführen

- Die Songliste ist relativ simpel:
  - ❖ Sie ist unterteilt in -Elementen
  - ❖ Eine davon ist die der Contentheader, die Überschrift der Inhalte die gerade angezeigt werden
  - ❖ Eine Songliste
- Anschließend kommt auch schon der <footer>-Teil, welcher ebenfalls unterteilt ist:
  - ❖ Es gibt die Abspielsteuerungen
    - Vorheriges Lied abspielen
    - Pausieren
    - Nächstes Lied abspielen
  - ❖ Die Liedinfo
    - Songcover
    - Liedname
    - Künstlername
  - ❖ Lautstärkesteuerung
    - Bild, welches sich basierend auf die Lautstärke anpasst
      - Ist die Lautstärke = 0, wird es mit einem „Stumm“-Symbol ausgetauscht
      - Ist die Lautstärke nicht über 0, wird ein Lautsprechersymbol angezeigt
  - ❖ Zudem ein unsichtbares -Element, welches sich über den ganzen Footer streckt
    - Dort wird der Fortschritt des Liedes angezeigt
    - Durch einen Klick kann der aktuelle Zeitpunkt des Liedes angepasst werden

Diese Elemente und Funktionen sind zumeist und im Wesentlichen leer und funktionieren so nicht. Diese werden erst durch JavaScript-Funktionen gefüllt und belebt. Darunter zählen z. B. die Liederliste und die Playlisten in der Seitenleiste, sowie die Abspielfunktionen. Außerdem gibt es PHP-Funktionen, die ausgeführt werden, sobald die Seite lädt.

Von welchen PHP-Funktionen spreche ich?

PHP ist in der home.php-Datei kaum zu finden. Die meisten PHP-Funktionen sind ausgelagert und werden über JavaScript dynamisch angezeigt. Letztendlich wird PHP nur zur Kommunikation mit der Datenbank verwendet und da Inhalte dynamisch aktualisiert werden sollen, ohne die Seite zu aktualisieren, da sonst Lieder abgebrochen werden, ist dies die beste Methode, die mir eingefallen ist.

Die PHP-Funktionen, die vorhanden sind:

- bestätigen, dass der Benutzer eingeloggt ist
- generieren den Begrüßungstext, da dieser nie verändert wird

Die Funktionen sind zu Beginn der Datei definiert. Die Funktion, die bestätigt, dass der Benutzer eingeloggt ist, überprüft, ob die UUID in den Cookies in der Datenbank vorhanden ist und gibt einen Boolean-Wert zurück. Zum Ende des PHP-Teils gibt es eine if-Abfrage, die diese Funktion aufruft. Ist der Wert negativ, wird die Verbindung abgebrochen und der Benutzer, wird zur Index-

Seite, der Login Seite, geleitet. Die Funktion, die die Begrüßung generiert, holt sich basierend auf dem UUID-Cookie den Benutzername aus der Datenbank und zeigt ihn mit dem Text „Welcome [Benutzername]!“ an.

Welche JavaScript-Funktionen gibt es?