Songify – Implementation Report – Project 1

Fontys University of Applied Sciences

By Jan Pinto Strohhäusl, Group DE-2, INF1-B

Rheurdt, 18.12.2021

Abstract

Der vorliegende Bericht handelt von der Implementation der Gruppe DE-2 an Songify. Der Bericht wurde geschrieben, um das Endergebnis der Gruppe an dem Projekt zu zeigen. Es werden verschiedene Funktionen der Seite aufgezeigt und erklärt. An einem bestimmten Use-Case wird auch der Code des Use-Cases gezeigt und im Detail erklärt.

Die Ergebnisse, die in diesem Bericht einfließen, entstanden aus verschiedensten Gruppenarbeits- und auch Einzelarbeitsphasen.

Der Bericht richtet sich an Personen, die den ersten Bericht gelesen haben und nun sich über das Endergebnis informieren wollen. Außerdem ist der Bericht für Webentwickler interessant.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungen

HTML5 / HTML: Hypertext Markup Language (5th version)

CSS: Cascading Style Sheets

PHP: PHP: Hypertext Preprocessor / Personal Home Page Tools

1 Einleitung

Der vorliegende Bericht handelt von der Arbeit unserer Gruppe, der Gruppe DE-2, an dem Projekt 1. Der Bericht soll erklären, wie wir an „Songify“ gearbeitet haben.

Der Bericht existiert, da nun das Ende des ersten Semesters naht und das Projekt sich dem Ende neigt. Damit endet auch die Implementationsphase des Projektes. Diese Phase wird nun in diesem Bericht zusammengefasst. Dieser Bericht wird auch in eine Präsentation einfließen, die noch gehalten wird.

Da der vorherige Bericht sich um die Ergebnisse der Artefacts der Analyse- und Design-Phase drehte, wird dieser Bericht von der Implementationsphase handeln. Die Website Songify haben wir auf Grundlage der Artefacts der Analyse- und Design-Phase und auch mit Hilfe der entstandenen Berichte aus dieser Phase implementiert. Da die Artefacts die Grundlage der Website darstellen, konnten wir letzten Endes aus diesen die eigentliche Website erstellen und die Website mit Leben füllen.

Die Erstellung der Website gestaltete sich als etwas schwieriger als erhofft. Wir mussten uns selbst die Sprachen HTML5, CSS und PHP beibringen, was am Anfang recht schwierig war, da dies viel auf einmal war und wir anfangs keine Ahnung hatten, wo wir anfangen sollten. Jedoch haben wir uns davon nicht zurückschrecken lassen. Um ein Gefühl für die Sprachen zu bekommen, haben wir uns abgestimmt, dass jeder einzeln eine Hauptseite aus reinem HTML5 und CSS-Code zusammenstellt. Da jeder eine andere Interpretation der Hauptseite und Songify im Allgemeinen hatte, war das die ideale Möglichkeit, eine Art Template für die Implementation zu erstellen und um zu gucken, wo wir Gemeinsamkeiten und Unterschiede hatten. Nachdem jeder seine Hauptseite fertig hatte, haben wir uns besprochen, welche Seite wir als Template nahmen und haben uns dann an dieser Seite orientiert.

Nach den ersten zwei Wochen mit HTML5 und CSS hatten wir diese Sprachen einigermaßen im Griff und konnten uns an PHP setzen. Da jede Sprache sich komplett zur anderen Sprache unterscheidet, hatten wir auch mit PHP Startschwierigkeiten. Da es jedoch Gemeinsamkeiten zwischen PHP und Java gibt, gab es hier Bereiche, die einfach waren, aber auch Bereiche, die nicht auf Anhieb funktionierten, da wir die Funktionen nicht richtig in den Code implementiert hatten. Das hat uns jedoch nicht davon abgehalten, uns PHP anzueignen. Danach haben wir uns darauf geeinigt, dass jeder ein Use-Case erarbeitet und es dann im Bericht und in der Präsentation thematisiert. Auch wenn jeder sein eigenes Use-Case hatte, haben wir trotzdem zusammengearbeitet. Am Ende hat alles funktioniert, so wie es soll, auch wenn es noch Verbesserungsmöglichkeiten gibt.

Am Anfang des Berichtes wird eine kurze Reflexion geschrieben. In dieser Reflexion wird nochmals kurz auf Elemente der Artefacts für die Seite eingegangen, die wir implementieren konnte und welche wir nicht implementieren konnten, da uns entweder die Zeit, das Wissen oder die Möglichkeiten fehlten. Danach wird die Seite im Allgemeinen erklärt und deren Funktion im Generellen. Daraufhin wird auf ein spezielles Use-Case eingegangen und dieses wird im näheren beleuchtet und beschrieben. Am Ende des Berichtes wird die Bearbeitung zusammengefasst und eine Bewertung meinerseits gefällt.

2 Reflexion

Dieser Abschnitt ist eine kurze Reflexion. Hier wird behandelt, was wir von unseren Erarbeitungen tatsächlich implementiert haben oder implementieren konnten und was nicht.

Da Songify unser erstes Projekt war, hatten wir viel vor und viele Ideen, von denen wir glaubten, dass diese sehr gut waren und auch gut passen würden. Jedoch in der Implementationsphase haben wir dann gemerkt, dass wir manches gar nicht implementieren konnten, da uns entweder das Wissen, die Zeit oder einfach die Möglichkeiten fehlten. Daher haben wir nur das implementiert, was auch ohne Probleme funktioniert und haben Restliches außen vor gelassen.

Im Grunde hatten wir zwei Probleme, die manche Artefacts nicht implementierbar machten.

Zum Ersten war es, dass wir zu den jeweiligen Sprachen jeweils eine Vorlesung hatten, in denen uns die Sprachen vorgestellt wurden, die Grundlagen der Sprachen gezeigt wurde und was mit diesen gemacht werden kann. Den Rest mussten wir uns selbst erarbeiten, was gut für das Lernen war, aber auch manchmal frustrierend war, da manche Implementationen nicht so funktionierten, wie sie sollten. Das hat uns anfangs ausgebremst, aber zum Ende hin war es gut, da wir dann mit der Sprache vertraut waren.

Das zweite Problem war, dass wir kein JavaScript benutzen durften. Manche Artefacts, wie zum Beispiel die Personalisierung der Seite oder generell Artefacts, bei denen Veränderungen auf der Seite passieren, die den Code verändern oder die Seite verändern, ohne die Seite zu aktualisieren, sind ohne JavaScript nicht möglich. Daher sind manche Überlegungen, die wir vorher getroffen haben, nicht zu implementieren gewesen, da wir nicht die Möglichkeiten hatten.

Diese Probleme haben dazu geführt, dass wir Artefacts, bei denen es um Individualisierung ging, nicht durchführbar waren. Die Artefact „Personalisierung der Seite“, bei der der Nutzer Farben angeben kann und diese Farben dann auf jede Seite angewendet werden, oder das Artefact „Personalisierung nach Alter“ bei der geschaut wird, wie alt der Nutzer ist, um dann die Schrift der Seite zu vergrößern oder mehr Bilder einzufügen, waren nicht umsetzbar. Aus diesem Grund haben wir auch keine Einstellungen, auf die der Nutzer zugreifen kann.

Auch konnten wir keine „Skip“, „Shuffle“, „Loop“ und weitere Buttons für die Musikkontrolle einbauen, da wir nicht die Möglichkeiten hatten, diese zu implementieren. Stattdessen haben wir den Audioplayer von HTML5 benutzt, der zwar Musik spielen kann, aber auch sehr limitiert ist. Da es mit dem Audioplayer nicht möglich ist, Songs zu skippen, haben wir uns dazu entschieden, keine Playlisten zu erstellen. Trotzdem sind noch Fragmente von anfänglichen Überlegungen zu sehen, die jedoch keinen Nutzen haben und dort sind, damit es nicht leer ist.

Die Datenbank, die wir vor der Implementation angelegt hatten, haben wir mehrmals überarbeitet, da uns immer wieder Sachen aufgefallen sind, die keinen wirklichen Nutzen haben, und diese haben wir aus der Datenbank entfernt.

Trotz all dem haben wir es geschafft, eine einigermaßen akzeptable und funktionierende Seite zu erschaffen. Auf das Design und die Funktion der Seite wird im nächsten Kapitel eingegangen.

3 Songify im Allgemeinen

In diesem Kapitel wird das Design von Songify und die Funktion und Navigation auf der Hauptseite im Allgemeinen beschrieben. Es wird auch auf verschiedene Aspekte, zum Beispiel Code und Sicherheit eingegangen.

3.1 Hauptseite

Dieses Kapitel befasst sich mit der Hauptseite von Songify. Auch wird hier auf die Navigation zwischen den Seiten eingegangen. Genannt wird die Hauptseite „Index“, da dies der Standardname für Hauptseiten ist.

Da aus unserer Gruppe jeder einzeln vorher eine eigene Index-Seite erstellt hat, haben wir uns dann entschieden, eine dieser Index-Seiten als Template für spätere Seiten zu nutzen. Dabei haben wir uns für folgendes Template entschieden. In Abbildung 1 sehen Sie die Hauptseite von Songify aus der Sicht des Artists. Dies erkennen Sie an den „Upload“-Knopf oben rechts, denn bei normalen Nutzern ist dieser nicht zu sehen. Wir haben uns für dieses Template und diesem Design entschieden, da es modern und schlicht war. Wir hatten ja in der Analysephase festgelegt, dass wir unsere Seite so schlicht wie möglich gestalten, damit der Nutzer es so einfach wie möglich hat. Außerdem bot es uns auch die Möglichkeit, die benutzen Elemente, wie zum Beispiel die Navigationsleiste oder den Footer, mehrfach zu nutzen. Dies konnten wir, indem wir die einzelnen Elemente (Navigation, Aside, Footer) in einzelne Dateien eingefügt haben und diese in einer großen Datei zusammengefügt haben. Dadurch konnten wir uns viel Arbeit und Code sparen. Außerdem hat es bewirkt, dass die Navigationsleiste, der Aside und der Footer auf jeder Seite gleich ist. Das einzig Scrollbare auf der Hauptseite ist die Section, hier die Kasten mit den Bildern, denn die ist das einzig variable Element der Seite. Der Rest bleibt auf seinem Platz.

Abbildung 1: Index-Seite von Songify (Sicht des Artisten)

Das Moodboard aus der Design-Phase hat eine große Rolle in der Erstellung der Index gespielt. Wer noch das Moodboard in Erinnerung hat, erkennt, dass die dominierenden Farben des Moodboards in die Index-Seite geflossen sind. In Abbildung 2 erkennen Sie die Farbenpalette, die wir für die Index-Seite benutzt haben. Die Farben wurden etwas verdunkelt, da es sonst zu grell wäre, aber an dieser Farbenpalette haben wir uns orientiert.

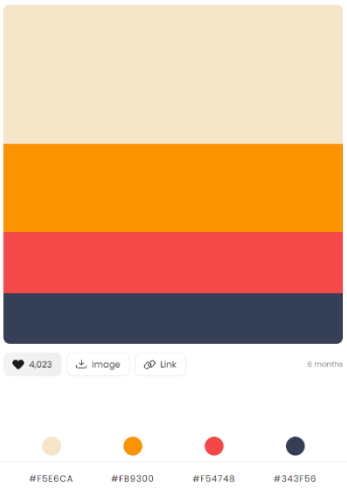

Abbildung 2: Farbenpalette (Internetquelle: https://colorhunt.co/palette/f5e6cafb9300f54748343f56)

Von der Index-Seite aus kann der Nutzer jede Seite erreichen. Wenn der Nutzer auf die Suchleiste klickt, etwas reinschreibt und „Enter“ drückt, wird der Nutzer auf die Suchseite mit den entsprechenden Suchergebnissen weitergeleitet. Da die Navigationsleiste auf jeder Seite zu sehen ist, kann der Nutzer auch von jeder Seite aus Songs suchen. Wenn der Nutzer oben links auf das Logo klickt, kommt der Nutzer von jeder Seite aus auf die Index-Seite zurück. Oben rechts kann

der Nutzer, wenn dieser mit der Maus über das Account-Profilbild geht, sich ausloggen. Wie eben erwähnt, kann der Artist auf den Upload-Knopf klicken und wird zur Upload-Seite weitergeleitet, normale Nutzer können nicht den Upload-Knopf sehen.

Links, die Aside, beinhaltet Fragmente von der Art, wie sie gerade in der Reflexion besprochen wurden. Die Knöpfe sollten ursprünglich zu Playlisten oder bestimmten Artisten führen und waren am Anfang implementiert worden, aber da es keine Playlisten oder Artistenprofile als Seite gibt, sind diese Knöpfe ohne Funktion. Die Knöpfe sind nur noch dort, damit die Aside nicht leer ist. Der untere Knopf „Legal Notice“ führt zu einer Impressum-Seite, auf der die Kontaktdaten jedes einzelnen aus unserer Gruppe stehen.

In dem Footer wird der aktuell ausgewählte Song mit Bild, Artisten und Titel angezeigt und kann auch dort abgespielt werden. Da aber auf der Index-Seite keine Songs ausgewählt sind, sind dort Platzhalter zu finden. Wenn der Nutzer auf eine Songseite kommt, werden die Platzhalter mit den jeweiligen Songdaten ausgetauscht und der Song kann abgespielt werden.

Die Section ist das einzig Scrollbare auf jeder Seite. Auch ist die Section das einzige Element, welches ausgetauscht wird. Der Rest der Elemente wird auf anderen Seiten wiederverwendet. Auf der Index-Seite wird in der Section Neuigkeiten der Website angezeigt, aber auch Verlinkungen zu Songs angezeigt. Wenn der Nutzer mit der Maus über einen Kasten fährt, erscheint ein Overlay mit weiteren Informationen zu dem Kasten. In Abbildung 3 wird ein Beispiel zu dem Overlay gezeigt: Bei Website-Nachrichten wird es so wie in Abbildung 3 angezeigt, bei Songs wird im Overlay die Beschreibung des Songs angezeigt.

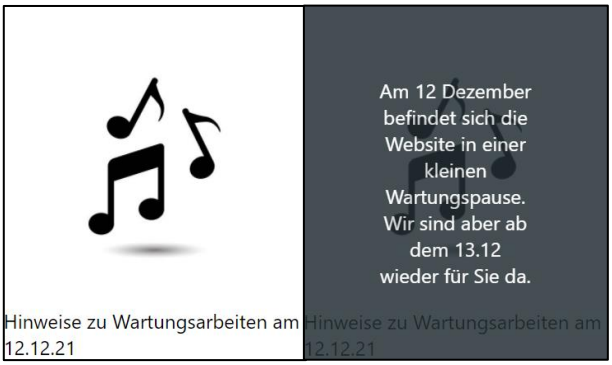

Abbildung 3: Links: normaler Kasten | Rechts: Overlay

Auf der Index-Seite wurden verschiedene Arten von Code verwendet, um diese so zu gestalten, wie sie jetzt ist. Im Grunde wurde jedes einzelne Element mit HTML initialisiert, mit CSS an die richtige Position verschoben und richtig designt und mit PHP die Verlinkung zwischen den Seiten oder andere Funktionen hergestellt. Unsere Codes haben wir auf GitHub geteilt. Dadurch, dass wir GitHub nutzten, konnten wir uns einfach und ständig austauschen und gegenseitig verbessern.

Auch werden nicht nur auf der Index-Seite, sondern generell auf Songify Sicherheitsaspekte beachtet. Beispielsweise kann die Index-Seite nur erreicht werden, wenn vorher ein Account angelegt wurde und sich der Nutzer in dieses eingeloggt hat. Falls dies nicht der Fall ist, wird der Nutzer auf die Login-Seite weitergeleitet. Dadurch wird sichergestellt, dass alle Nutzer von Songify vorher registriert sein müssen, bevor sie Zugang zur Seite haben. Dies ist nur ein Beispiel von manchen Sicherheitsaspekten, die auf Songify eingebaut wurden.

Nun wurde die Seite im Allgemeinen beschrieben. Im nächsten Kapitel folgt ein explizites Use-Case: das Uploaden eines Songs. Dies geschieht, wenn der Artist auf den Upload-Knopf drückt. Wenn dieser gedrückt wird, wird der Artist zur Upload-Seite weitergeleitet. Näheres zum Hochladen eines Songs finden Sie im nächsten Kapitel.

4 Use-Case: Einen Song hochladen

Dieses Kapitel wird sich mit der Implementation und der Funktion befassen, wie ein Song auf Songify hochgeladen wird. Andere Use-Cases wie das Suchen eines Songs, einen Account anlegen usw. finden Sie in den Berichten meiner Kollegen.

4.1 HTML und CSS

Dieser Abschnitt wird sich mit der Implementation der Seite in HTML und CSS befassen. Die Funktion der Website durch PHP folgt später.

Vor der Implementation wurde eine Skizze erstellt, auf der alle wesentlichen Inhalte der Seite zu finden sind. Die Skizze wurde erstellt, um sicher zu gehen, dass keine wichtigen Elemente im Endprodukt fehlen. Die Skizze können Sie links in Abbildung 4 erkennen. Es wurden die Templates (Navigation als „Nav“ abgekürzt, Aside, Footer) und die restlichen Inhalte eingezeichnet. Oben wird ein Feld für die Songabgabe sein, darunter der Titel, Preis und Artist, darunter das Genre und das Bild des Songs und darunter Platz für eine Beschreibung. Rechts von der Beschreibung ist der „Submit“-Knopf, der dann, wenn alles eingegeben wurde, die Daten an den späteren PHP-Code weiterleitet.

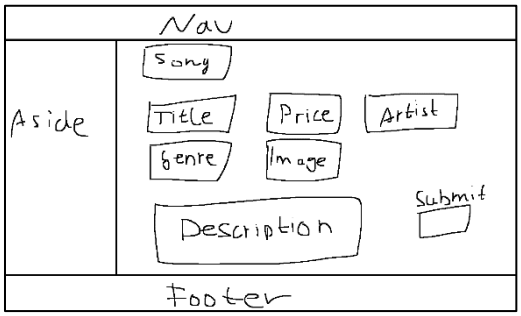

Abbildung 4: Skizze zur Upload-Seite

Um das Grundgerüst der Website zu erschaffen, wurden zuerst die Templates, die auf jeder Seite zu finden sind, implementiert (Navigation, Aside, Footer). Dadurch blieb noch der Rest, die Section, um das eigentliche Formular zu implementieren, um einen Song hochzuladen.

Neben den Templates wurden verschiedene Felder von HTML benutzt, um Eingabemöglichkeiten für den Nutzer zu bieten. Für Titel und Artist wurden Textfelder (HTML: `<input type=“text“…>`) genutzt, um dem Artist die Möglichkeit zu geben, die Titel und den Name des Artists selbst zu wählen. Die Beschreibung des Songs ist eine Textarea (HTML: `<textarea …> </textarea>`), da Textfelder wie von vorhin keinen Zeilenumbruch machen, und da wir dem Artist die bestmögliche Erfahrung beim Hochladen bieten wollen, muss hier die Textarea genutzt werden. Das Genre wird in einem Dropdownmenu aus voreingestellten Genres ausgewählt, sodass der Artist nichts eingeben muss. Der Preis des Songs wird in einem Nummernfeld (HTML: `<input type=“number“…>`) erfasst. Dadurch wird festgelegt, dass der Artist nur Zahlen als Preis angibt und keine Fehler machen kann. Auch wurde hier

```
<fieldset class="price_upload">
    Price:
    <input type="number" min="0.10" max="3.00" step="0.01" name="price" id="price" required>
</fieldset>
```

Abbildung 5: HTML-Code für Preis-Feld

festgelegt, dass der Song mindestens 10 Cent kostet und nicht über 3€ kosten darf. Den Beispiel-Code für das Preis-Feld sehen Sie hier in Abbildung 5. Der Song und das Bild vom Song werden als Datei (HTML: `<input type=“file“…>`) an den Server übermittelt. Dadurch kann das Bild und der Song später auf der Songseite abgerufen und abgespielt werden.

In der Skizze ist zu erkennen, dass es eine Art Reihenordnung von links nach rechts gibt (Title-Price-Artist / Genre-Image). Daher wurde hier in CSS dafür Flexbox (HTML: `<div`

`class="flex_container"> … </div>` | CSS: `.flex_container{ display: flex; …}`) genommen, um diese einfacher, einheitlicher und effektiver auszurichten, zu formatieren und zu gestalten. Auch konnte dadurch der Abstand zwischen den Feldern (CSS: `margin-top: 2%; | margin-left: 5%;`) unkompliziert und einheitlich gesetzt werden. Der Knopf, der den Upload ausführt, wurde auch mit CSS formatiert und dort gesetzt, wo er sich jetzt befindet. Wie bei allen Knöpfen wurde auch bei diesem Knopf erst die Größe eingestellt, danach die Schriftart und Schriftgröße, danach den Rand und die Farbe des Randes und des Knopfes. Auch wurde es so eingestellt, dass sich der Mauszeiger in einen „Pointer" umwandelt, also zu der Hand mit dem ausgestreckten Zeigefinger, um anzuzeigen, dass hier interagiert werden kann. Als letztes wurde ein „Hover"-Effekt eingefügt. Dies bedeutet, dass der Knopf in 0.2 Sekunden die Farbe wechselt, wenn die Maus über den Knopf geht. Den CSS-Code des Upload-Knopfes sehen Sie rechts in Abbildung 6.

```css
input[type=submit]{
    width: 150%;
    height: 100%;
    cursor: pointer;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 1.5vw;
    border-radius: 15px;
    border: 2px solid black;
    background-color: rgb(194, 104, 1);
    transition-duration: 0.2s;
}

#submit:hover{
    background-color: rgb(194, 62, 10);
}
```

Abbildung 6: CSS-Code zur Formatierung des Upload-Knopfes

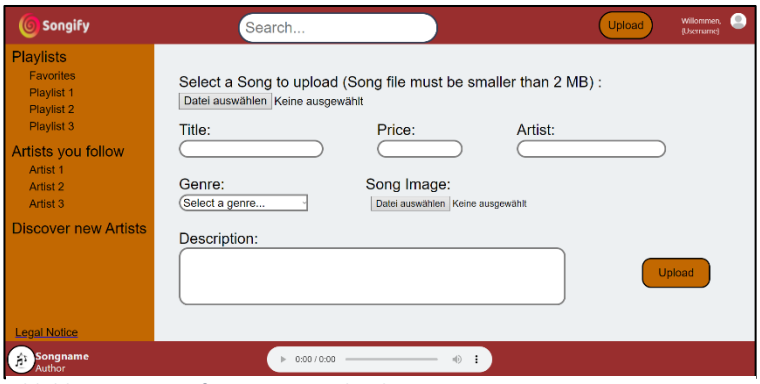

Abbildung 7: Fertig formatierte Upload-Seite

Außerdem wurden in alle Felder die Bedingung `required` hinzugefügt, damit der Artist auch alle Felder ausfüllen muss, bevor das Formular an den PHP-Code weitergegeben werden kann. Das Endergebnis der Seite sehen Sie hier in Abbildung 7. Sie können sehen, dass nah an der Skizze gearbeitet wurde. Stellenweise war es auch schwer, genau nach der Skizze zu programmieren, aber letzten Endes hat es funktioniert. Außerdem wird der Artist darauf hingewiesen, dass die Songdatei nicht größer als 2 Megabytes sein darf. Das liegt daran, weil PHP standardmäßig keine Dateien, die größer als 2 Megabytes sind, hochladen kann. Da wir auch keinen Weg gefunden haben, die Dateigröße vor dem Hochladen zu kontrollieren, wird der Künstler lediglich darauf hingewiesen.

Dies war der HTML und CSS-Code für die Upload-Seite. Weiter im nächsten Abschnitt geht es mit dem PHP und er Funktionalität der Upload-Seite.

4.2 PHP und SQL

Dieser Abschnitt handelt von der Funktionalität der Upload-Seite. Es wird hier auf die Verarbeitung der Daten und das Einfügen in die Datenbank eingegangen.

Wenn das Upload-Formular ausgefüllt wurde und der Artist auf „Upload" klickt, wird eine neue Seite geöffnet. Auf dieser Seite werden die Daten verarbeitet und schließlich in die Datenbank hochgeladen.

Den gesamten PHP-Code finden Sie im Appendix A. Dort ist der Code, weil dieser etwas zu groß für ein einziges Bild ist, aufgeteilt in 4 Abschnitte. Diese Abschnitte werden im Folgenden beschrieben.

Im Abschnitt 1 wird zuerst eine Verbindung zur Datenbank erstellt. Ohne diese Verbindung ist es nicht möglich, die Werte zu verarbeiten. Nachdem die Verbindung steht, werden die Werte aus dem Upload-Formular als Variablen abgespeichert. Dadurch können die Werte später in der Datenbank abgespeichert werden. Als letztes werden die Variablen, die „Titel“ und „Artist“ beinhalten, formatiert, bedeutet eventuelle Leerzeichen vor oder nach dem Wort werden entfernt. Außerdem werden die Wörter richtig geschrieben. Richtig geschrieben bedeutet, dass die Wörter mit einem Großbuchstaben beginnen und der Rest des Wortes kleingeschrieben ist (z.B. „ bRuNo MaRs “ → „Bruno Mars“). Dies wird getan, damit später die Kontrolle, ob das Lied in der Datenbank schon mal eingetragen wurde, einfacher und effizienter abläuft.

Der nächste Abschnitt, Abschnitt 2, beinhaltet die Kontrolle der Dateiformaten. Da nur JGP oder PGN-Dateien als Bilder und MP3-Dateien als Audiodateien erlaubt sind, wird hier kontrolliert, ob die Dateien, die im Upload-Formular abgegeben wurden, diese Bestimmungen erfüllen. Dies wird mit einem Boolean gemacht, der standardmäßig auf „`false`“ gesetzt ist. Zuerst wird kontrolliert, ob das Bild auch tatsächlich ein Bild ist. Da dies normalerweise der Fall ist, wird der Boolean auf „`true`“ gesetzt. Danach werden die Formate kontrolliert. Wenn ein Format nicht stimmt, wird eine Meldung abgegeben, dass das Format nicht stimmt, und der Boolean wird auf „`false`“ gesetzt. Dadurch wird gesichert, dass die richtigen Formate abgegeben werden. Der Grund, warum nur diese Dateiformate erlaubt sind, ist, dass diese Dateiformate die gängigsten Formate für Bild- / Audiodateien sind. Falls ein Künstler andere Formate abgibt, wird der Upload-Vorgang abgebrochen und der Künstler muss seine Dateien neu formatieren.

Im Abschnitt 3 wird dann der Song und das Bild zum Song hochgeladen und die Werte der Variablen in die Datenbank eingefügt. Zuerst wird mit dem Boolean kontrolliert, ob alle Dateiformate stimmen. Wenn diese stimmen, ist, wie gerade erwähnt, der Boolean auf „`true`“ gesetzt. Wenn der Boolean auf „`true`“ gesetzt ist, beginnt die Injektion in die Datenbank. Zuerst werden die Dateien hochgeladen. Da keine Dateien in die Datenbank eingefügt werden können, arbeiten wir hier mit den Dateipfaden. Dazu werden die Dateien in den Ordner hochgeladen, in den sich die anderen Dateien befinden. Wenn die Dateien in die Ordner verschoben wurden, werden die Dateipfade der Dateien als Variablen abgespeichert. Danach wird der SQL-Query vorbereitet und die Variablen dort eingefügt. Da die Datenbank auch „Download“, „Clicks“ und „Revenue“ beinhalten, werden die standardmäßig jeweils auf 0 gesetzt, da ein neuer Song normalerweise keine Clicks, keine Downloads und keinen Gewinn generiert hat. Wenn alle Variablen in dem SQL-Query eingefügt wurden, wird dieser ausgeführt. Da auf Datenbankebene ein Trigger existiert, der hier links in Abbildung 8 zu sehen ist, wird beim Eintrag in die Datenbank kontrolliert, ob der Song mit dem gleichen Titel und Artist schon in der Datenbank vorhanden ist. Falls dies der Fall ist, wird eine Exception ausgegeben, die im PHP-Code durch die Try-Catch-Anordnung abgefangen wird. Würde es die Try-

```
create or replace function checkSong()
    returns trigger as $$
begin
    if exists (select * from song where title = new.title and artist = new.artist) then
        raise exception 'Song already exists';
        return null;
    end if;
    return new;
end;
$$ language plpgsql;

create trigger checkSong
    before insert on song
    for each row execute procedure checkSong();
```

Abbildung 8: Trigger auf Datenbankebene

Catch-Anordnung nicht geben, würde der PHP-Code an dieser Stelle abbrechen und die Fehlermeldung würde auf der Seite stehen. Da dies aber nicht der Fall ist, wird bei dem gleichen Song eine Meldung ausgegeben, dass der Song bereits existiert und der Boolean wird auf „`false`“ gesetzt.

Im letzten Abschnitt, Abschnitt 4, wird ausgegeben, ob der Song hochgeladen wurde oder nicht. Anhand des Boolean kann auch hier gecheckt werden, ob es erfolgreich war, den Song hochzuladen oder nicht. Falls es nicht erfolgreich war, den Song hochzuladen, dann wird neben der Meldung, dass es nicht erfolgreich war, auch die Meldung stehen, warum es nicht erfolgreich war (ungültiges Dateiformat, Song existiert bereits). Außerdem hat der Artist dann die Möglichkeit, einen neuen Song hochzuladen oder zurück zur Index-Seite zu kommen. Dies steht zwar nicht in dem PHP-Code, aber gleich bei dem Beispiel wird es zu sehen sein.

Dies war der Mechanismus, wie ein Song hochgeladen werden kann. Im nächsten Abschnitt wird ein Beispiel-Upload gezeigt, um den gezeigten Code besser zu verstehen und auch die Funktion zu visualisieren.

4.3 Beispiel-Upload

Dieser Abschnitt beschäftigt sich mit einem Beispiel-Upload in die Datenbank. Auch werden hier bewusst Fehler-Einträge durchgeführt, um zu veranschaulichen, wie der Code mit Fehler-Einträgen umgeht.

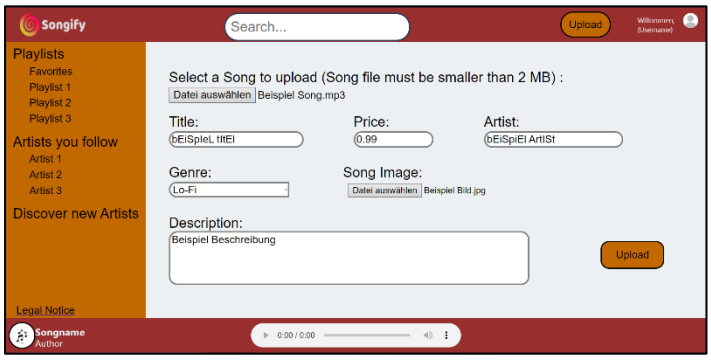

Abbildung 9: Ausgefülltes Upload-Formular

In Abbildung 9 sehen Sie einen Beispiel-Eintrag für einen Song, den ein Artist hochladen möchte. Es ist zu erkennen, dass der Titel und der Artist absichtlich falsch geschrieben wurden, um zu zeigen, wie diese anschließend beim Datenbankeintrag formatiert werden. Auch ist zu erkennen, dass der Name des Songs und des Bildes angezeigt werden. Die soll so sein, um dem Artist zu zeigen, welche Dateien er gerade am Hochladen ist, denn es kann vorkommen, dass sich dieser verklickt und aus Versehen eine Datei anklickt, die gar nicht hochgeladen werden soll. Wenn der Artist sich sicher ist, dass alle eingegebenen Werte stimmen, dann klickt er auf dem „Upload“-Knopf klicken, um den Vorgang des Hochladens zu beginnen. Falls der Artist etwas vergessen hat, dann wird dieser daran erinnert. In Abbildung 10 sehen Sie die Seite, auf der der Artist idealerweise weitergeleitet wird, wenn alles gut funktioniert hat. Hier werden nochmals der Titel und der Artist angezeigt. Es ist zu erkennen, dass der Titel und der Artist korrekt formatiert wurden. Diese Seite unterscheidet sich von allen anderen Seiten, da hier keine Templates genutzt wurden, um die Seite zu erstellen. Die

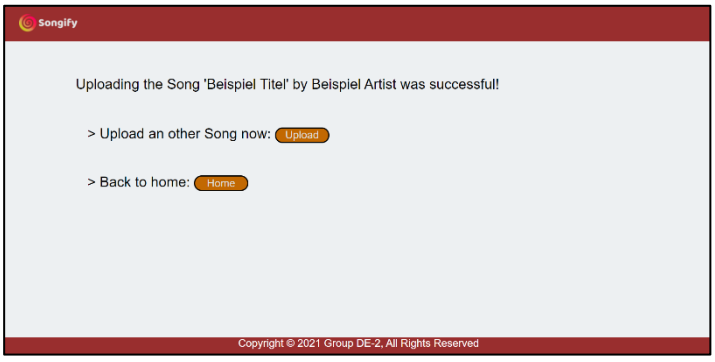

Abbildung 10: Hochladen erfolgreich

Navigationsleiste besteht hier nur aus dem Logo, welches im Gegensatz zu dem Template hier nicht interaktiv ist. Auch ist der Footer nur ein Copyright-Hinweis. Auch ist hier kein Aside zu sehen. Wir haben uns dazu entschieden, die Seite so zu gestalten, wie sie in Abbildung 10 zu sehen ist, da die Seite nur als Hinweis an den Artisten gemeint ist, ob der Song erfolgreich hochgeladen wurde oder nicht. Einen anderen Sinn hinter dieser Seite gab es nicht, deshalb gab es auch keinen Grund, diese weiter auszuschmücken und mit Templates zu gestalten. Das einzig interaktive sind die Knöpfe, die dort zu sehen sind. Der obere Knopf hat die gleiche Funktion wie der Knopf in der Navigationsleiste auf der Index-Seite und leitet zur Upload-Seite zurück, damit der Artist, falls er es möchte, einen weiteren Song hochladen kann oder, falls das Hochladen nicht funktioniert, einen anderen Song hochladen kann. Der untere Knopf leitet den Artist zur Index-Seite zurück, von der er aus wieder Songs suchen, nochmals Songs uploaden oder auch Songs hören kann.

Abbildung 11: Hochladen fehlgeschlagen - Song existiert bereits

Abbildung 11 zeigt den Fall, dass der Song bereits in der Datenbank eingetragen wurde. Das macht es unmöglich, den gleichen Song in die Datenbank einzutragen, da doppelte Songs vom gleichen Artisten nicht gut sind. Auch hier hat der Artist die Möglichkeit, wieder auf die Upload-Seite zurück zu gehen, um einen anderen hochzuladen oder zur Index-Seite zurückzukehren. Wie gerade gesagt, wird bei den Fällen, wenn ein Song nicht in die Datenbank eingetragen wird. In Abbildung 12 wurden die falschen Dateiformate angegeben, was dazu führte, dass der Song nicht hochgeladen wurde. In diesem Fall wurden beide Dateiformate falsch eingetragen. Falls nur das Bild oder nur die Audiodatei falsch ist, dann wird auch nur die Meldung zur falschen Datei angezeigt. Wenn alles funktioniert und der Song erfolgreich hochgeladen wurde, dann erscheint ein Datenbankeintrag in der Datenbank. Den Datenbankeintrag zum erfolgreich hochgeladenen Song aus dem Beispiel sehen Sie in Abbildung 13. Alle Daten wurden

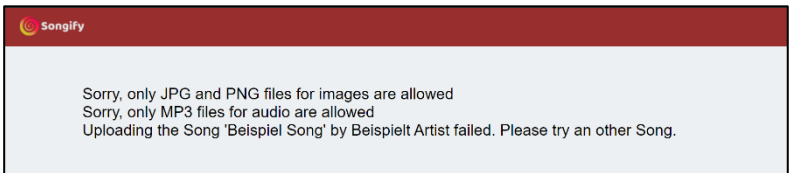

Abbildung 12: Hochladen fehlgeschlagen - falsche Dateiformate

song_id	title	artist	publ_date	price	description	genre	revenue	downloads	clicks
44	Beispiel Titel	Beispiel Artist	2021-12-18	0.99	Beispiel Beschreibung	Lo-Fi	0	0	0

Abbildung 13: Datenbankeintrag des erfolgreich hochgeladenen Songs

korrekt eingegeben und in die Datenbank eingefügt. Auch ist zu sehen, dass der Titel und der Artist erfolgreich formatiert wurden. Was nicht im Bild zu erkennen ist, sind die Dateipfade des Bildes und des Songs, denn diese werden auch in der Datenbank gespeichert. Diese Daten befinden sich rechts von „clicks“. Die ID des Songs wird automatisch generiert.

Dies war ein Beispiel zum Hochladen des Songs. Das letzte Kapitel beinhaltet die Zusammenfassung und eine Bewertung meinerseits.

5 Zusammenfassung

Zusammenfassend lässt sich sagen, dass Songify eine akzeptable Webapplikation geworden ist.

Auch wenn wir nicht alles implementieren konnten, was wir erarbeitet haben, haben wir es trotzdem geschafft, eine gute und ansehnliche Website zu gestalten. Die Website, die mit HTML5, CSS und PHP erstellt wurde, hat viele Funktionen, die über das Einloggen auf der Website, ein Konto anlegen über einen Song suchen bis hin zu Musik hören und auch selbst Songs hochladen gehen. Die Kommunikation zwischen der Website und der Datenbank ist ständig bestehend und funktioniert gut. Wir konnten viele Inhalte, die wir in der Analyse- und Design-Phase erarbeitet haben, mit in die Website einfließen lassen.

Ich persönlich denke, dass wir als Team eine sehr gute Leistung erbracht haben. Auch wenn es manchmal frustrierend war, mit HTML5, CSS und PHP zu arbeiten, herrschte immer eine angenehme Arbeitsatmosphäre und es hat Spaß gemacht, in der Gruppe zu arbeiten. Wir konnten gemeinsam Probleme lösen, die der Code mit sich brachte, und wenn es andere Bedenken gab, konnten wir diese auch ohne Probleme ansprechen und lösen.

In den nächsten Wochen werden die Klausuren geschrieben. Nach den Klausuren endet das erste Semester. Außerdem gibt es am 7. Februar die „Informatics Expo Day“, auf welcher alle Songify und Tickify-Projekte vorgestellt werden.

Appendix A

PHP-Code

1

```php
<?php
//Verbindung zur Datenbank aufbauen
$host  = "prj1_postgres";
$port  = "5432";
$db = "postgres";
$user  = "postgres";
$pword = "mypassword";
$dsn = "pgsql:host=$host;port=$port;dbname=$db;user=$user;password=$pword";
try {
    $conn = new PDO($dsn);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    echo $e->getMessage();
}

//Variablen festlegen und formatieren
$title = $_REQUEST['title'];
$artist = $_REQUEST['artist'];
$price = $_REQUEST['price'];
$genre = $_REQUEST['genre'];
$desc = $_REQUEST['description'];

$publdate = date("Y-m-d");

$title = trim($title);
$artist = trim($artist);

$title = ucwords(strtolower($title));
$artist = ucwords(strtolower($artist));
```

2

```php
try {

    //Kontrollieren, ob die Formate der Dateien stimmen
    $checker = false;

    $image_dir = "/var/www/html/prj1-2021-groups-implementation-prj1-2021-de-2/Song_Data/Song_Picture/";
    $image_file = $image_dir . basename($_FILES["image"]["name"]);
    $imageFileType = strtolower(pathinfo($image_file, PATHINFO_EXTENSION));

    $check = getimagesize($_FILES["image"]["tmp_name"]);
    if ($check !== false) {
        $checker = true;
    }

    if ($imageFileType != "jpg" && $imageFileType != "png") {
        echo "Sorry, only JPG and PNG files for images are allowed";
        echo "<br>";
        $checker = false;
    }

    $song_dir = "/var/www/html/prj1-2021-groups-implementation-prj1-2021-de-2/Song_Data/Song_Files/";
    $song_file = $song_dir . basename($_FILES["song"]["name"]);
    $songFileType = strtolower(pathinfo($song_file, PATHINFO_EXTENSION));

    if ($songFileType != "mp3") {
        echo "Sorry, only MP3 files for audio are allowed";
        echo "<br>";
        $checker = false;
    }
```

3

```php
        //Wenn alles passt, dann werden die Dateien und Variablen hochgeladen
        if ($checker == true) {

            if (move_uploaded_file($_FILES["image"]["tmp_name"], $image_file)) {
            }

            if (move_uploaded_file($_FILES["song"]["tmp_name"], $song_file)) {
            }

            $songdata = "Song_Data/Song_Files/" . $_FILES["song"]["name"];
            $songpic = "Song_Data/Song_Picture/" . $_FILES["image"]["name"];

            $stmt = $conn->prepare("INSERT INTO song (title, artist, publ_date, price, description, genre, revenue, downl
            $stmt->bindValue(':title', $title, PDO::PARAM_STR);
            $stmt->bindValue(':artist', $artist, PDO::PARAM_STR);
            $stmt->bindValue(':publ_date', $publdate, PDO::PARAM_STR);
            $stmt->bindValue(':price', $price, PDO::PARAM_STR);
            $stmt->bindValue(':description', $desc, PDO::PARAM_STR);
            $stmt->bindValue(':genre', $genre, PDO::PARAM_STR);
            $stmt->bindValue(':revenue', 0, PDO::PARAM_STR);
            $stmt->bindValue(':downloads', 0, PDO::PARAM_STR);
            $stmt->bindValue(':clicks', 0, PDO::PARAM_STR);
            $stmt->bindValue(':song_data', $songdata, PDO::PARAM_STR);
            $stmt->bindValue(':song_profile_picture', $songpic, PDO::PARAM_STR);

            $stmt->execute();
        }
        //Wenn der Trigger eine Exception zurückgibt, wird die Exception hier abgefangen
    } catch (Exception $e) {
        echo "Sorry, this Song does already exists";
        echo "<br>";
        $checker = false;
    }
```

4

```php
//Endergebnis
if ($checker == true) {
    echo "Uploading the Song '" . $title . "' by " . $artist . " was successful!";
} else {
    echo "Uploading the Song '" . $title . "' by " . $artist . " failed. Please try an other Song.";
}
?>
```