KLANG

5PEKTRUM

KLANG
5PEKTRUM

Bachelorarbeit an der Fakultät für
Gestaltung der Hochschule Augsburg

Interaktive Medien
Wintersemester 2016 / ´17

Erstgutachter: Prof. Dr. Michael Kipp
Zweitgutachter: Prof. Daniel Rothaug

Vorgelegt von: Michael Schwarz
Augsburg, 01. Februar 2017

ABSTRACT

Musikstreamingdienste wie *Spotify* bieten ihren Nutzern neben einer breiten Musikauswahl auch personalisierte Wiedergabelisten bzw. Playlisten und individuelle Musikvorschläge an. Diesen Service ermöglichen komplexe Algorithmen, welche nach aktuellen Trends die etablierten Kriterien der Musikauswahl ablösen.

Die Datenvisualisierung der vorliegenden Bachelorarbeit soll Spotifynutzern eine Analyse des eigenen Musikverhaltens auf der Basis von vereinfachten algorithmisch erstellten Songeigenschaften ermöglichen. Hierbei soll der Nutzer die abstrakten Werte der unterschiedlichen Eigenschaften mithilfe der bekannten Messgrößen *Songs* und *Genres* in Bezug setzen können. Der Betrachter wird bildlich gesprochen zum Algorithmus, um das eigene Musikprofil zu analysieren.

Für die Umsetzung dienen aktuelle Frameworks wie *Node.js* und Programmierparadigmen wie *REST* als Grundlage der serverseitigen Programmierung. Die Datenvisualisierung wird mit der JavaScript Bibliothek *P5.js* umgesetzt und bietet gute Interaktionsmöglichkeiten für den Nutzer. Die Visualisierung wird mit Verhaltensmethaphern zum Thema Musik untermalt und stärkt somit das Erlebnis des Nutzers.

Mithilfe eines Nutzertests wird die Verständlichkeit und die Bedienbarkeit der Webanwendung geprüft und verbessert. Des Weiteren hat die Webanwendung die Chance, Teil des offiziellen *Developer-Showcase* von *Spotify* zu werden, was die Nutzerzahlen ernorm steigern würde.

INHALTSVERZEICHNIS

1. EINLEITUNG

1.1 Motivation

Wir leben in einer zunehmend digitalisierten Welt, jedoch besitzen die meisten kaum Hintergrundwissen über digitale Vorgänge und Prozesse. Durch das digitale Zeitalter wachsen immer mehr Streamingdienste heran. Das sieht man im Bereich Film durch Größen wie *Amazon* und *Netflix*, sowie in der Musikindustrie durch beispielsweise *SoundCloud* und *Spotify*. Der anfängliche Service von *iTunes*, mit welchem Nutzer ihre Wunschmusik schnell und einfach digital erwerben und mit wenigen Klicks ein Album sortiert in deren Bibliothek abgelegen konnten, wurde überholt. Dienstleister wie *Spotify* bieten ihren Nutzern über Monatsverträge uneingeschränkten Zugang zu deren Musikarchiv und schlagen zusätzlich personalisierte Musik vor. Das Angebot erstreckt sich unter anderem über Radiosender und fertige Playlists, welche nach Stimmungen oder Genres sortiert sind.

Des Weiteren kann nach ähnlichen Künstlern gesucht und Musik mit ebenfalls angemeldeten Freunden geteilt werden. Streamingdienste wie *Spotify* haben die Musikindustrie und das Musikverhalten ihrer Nutzer verändert. Aktuelle Trends zeigen, dass sich das Musikverhalten des Einzelnen zunehmend von Kriterien wie *Genre* und *Künstler* abwendet. Die etablierten Größen der Musikauswahl scheinen von intelligenten, im Hintergrund arbeitenden Algorithmen abgelöst zu werden *(Schwarz, 2016)*.

Spotify ist für die meisten Nutzer allerdings eine sogenannte „Blackbox“, d.h. die Hintergrundprozesse der Software können von Nutzern ohne Fachwissen kaum nachvollzogen werden. Für einen Blick hinter die Kulissen ist ein großes Know-How an Informatikkenntnissen erforderlich, daher wundert es nicht, dass sich wenige Nutzer über diese Prozesse informieren bzw. dafür interessieren.

Aus Sicht eines Entwicklers sind jedoch gerade die dem Nutzer verborgenen Prozesse sehr spannend, welche große Mengen an Daten, sowie aufwändige Analysen beinhalten. Es sollte ein Interesse für diese „ausgeblendeten“ Prozesse unseres alltäglichen Lebens geweckt und ein Verständnis derer geschaffen werden.

1.2 Zielsetzung

Damit *Spotify* für seine Nutzer transparenter wird, soll eine Datenvisualisierung mithilfe von durch Hintergrundprozesse gewonnene Daten umgesetzt werden. Visualisierungen werden im Allgemeinen dazu genutzt große Menge an Daten, komplexe Prozesse und Zusammenhänge zu verdeutlichen.

Die vorliegende Bachelorarbeit wird sich in diesem Kontext mit einer unkonventionellen Herangehensweise beschäftigen, wie Spotifynutzern neues Wissen über Hintergrundprozesse spielerisch vermittelt werden kann. Die Erkenntnisse durch die Analyse und Interaktion mit der Anwendung sollen durch eine einmalige Benutzung gewonnen werden können und setzten keine erneute Nutzung der Anwendung voraus.

Um eine hohe Erreichbarkeit zu ermöglichen soll die Visualisierung als Webanwendung umgesetzt werden. Diese richtet sich an Spotifynutzer, welche Interesse an einen Einblick in die Datenstrukturen *Spotifys* haben.

Zur Überprüfung der Anwendung soll ein Nutzertest zeigen, ob die Probanden in der Lage sind, richtig mit dem Interface zu interagieren, die Visualisierung korrekt zu interpretieren und die Hintergrundprozesse zu verstehen.

2. RECHERCHE

2.1 Musik

Die Erklärung des Begriffs *Musik* beginnt in der Regel damit, dass Musik als organisierter Klang bezeichnet wird. Allerdings fällt unter diese Definition auch menschliche Sprache oder tierische Geräusche.

Das *Concise Oxford Dictionary* definiert Musik als „*die Kunst, stimmliche oder instrumentale Klänge (oder beides) zu kombinieren, um Schönheit von Form, Harmonie und Ausdruck von Emotionen zu erzeugen*“ *(Allen u. a., 1992).*

Jedoch werden in der heutigen Musikerstellung nicht nur harmonische Klänge sondern auch „krachige“ oder „freie“ Klänge in Genres wie zum Beispiel *Hardcore* oder *Jazz* verwendet. Ebenfalls wird Musik aktiv durch elektronische Geräte verzerrt und verfremdet.

Ein berühmtes Beispiel für das Dilemma der Definition von Musik ist die Komposition des Künstlers und Komponisten *John Cage: 4’33”*. Hierbei verdeutlicht ein Pianist lediglich in allen drei Sätzen durch Gesten, wann das Stück beginnt und wann dieses endet. Das Einzige, was die Gäste hören, sind zufällige Umgebungsgeräusche im Saal. Nach *John Cage* ist alles, was Menschen hören Musik und lehnt sich somit stark an die Definition von autonomer Musik an *(Gann, 2010).*

Um den abstrakten Begriff *Musik* einzugrenzen, wurden im ersten Schritt dieser Bachelorarbeit Mindmaps erstellt, welche den Begriff beschreiben. Nachfolgend werden die wichtigsten Begriffe und Assoziationen bezüglich der Thesis aufgelistet, woraus sich folgende Eigenschaften und Verhaltensweisen ableiten lassen:

Eigenschaften:
- Kleinteilig
- Komplex
- Organisch
- Menschlich
- Individuell
- Bunt
- Stimmungsvoll
- besteht aus Geräuschen / Schwingungen

Verhaltensweisen:
- Verformt sich
- Wächst
- Ruft eine Reaktion hervor
- Ist in Bewegung
- Kann man entdecken
- Umgibt uns
- Begleitet uns

2.2 Visualisierungsmethoden

Audiovisualisierungen stellen den Anspruch an sich, Musik visuell darzustellen. Dabei werden Grafiken und Animationen erzeugt auf Basis von Audiodateien, wie beispielsweise Sprache und Geräuschen. Es wird versucht, die technischen Informationen zu verbildlichen, wie auch künstlerisch die Musik zu untermalen *(Wikipedia, o. J.).*

„*Mit Visualisierung oder Veranschaulichung (Sichtbarmachen) meint man im Allgemeinen, abstrakte Daten und Zusammenhänge in eine graphische bzw. visuell erfassbare Form zu bringen.*“ *(Seifert, 2009).* Je nach Intention und gewünschtem Ergebnis gibt es unterschiedliche Visualisierungsarten bzw. -methoden. Die Folgenden sind für die vorliegende Bachelorarbeit besonders interessant.

Histogramm
Ein *Histogramm (vgl. Abb. 1, Histogramm)* ist eine graphische Darstellung der Häufigkeitsverteilung. Es erfordert die Einteilung der Daten in Klassen, die eine konstante oder variable Breite haben können. Die Höhe jeder Klasse stellt die Häufigkeitsdichte dar, also die Häufigkeit dividiert durch die Breite der entsprechenden Klasse.

Histogramme helfen konzentrierte Werte, Lücken oder ungewöhnliche Werte zu verdeutlichen. Sie sind auch nützlich, um eine grobe Aussage über die Wahrscheinlichkeitsverteilung zu treffen *(Severino, o. J.).*

Heatmap
„Eine Heatmap (vgl. Abb. 2, Heatmap) ist ein Diagramm zur Visualisierung von Daten, deren abhängige Werte einer zweidimensionalen Definitionsmenge als Farben repräsentiert werden. Sie dient dazu, in einer großen Datenmenge intuitiv und schnell besonders markante Werte zu erfassen.“ (Severino, o. J.).

Circular Treemap
Eine *Treemap (bzw. Baumdiagramm) (vgl. Abb. 3, Treemap)* dient der Visualisierung hierarchischer Strukturen, die durch ineinander verschachtelte Rechtecke dargestellt werden. Damit können Größenverhältnisse anschaulich dargestellt werden, indem die Fläche der Rechtecke proportional zur Größe der darzustellenden Dateneinheit gewählt wird. Die *Circular Treemap* ist eine Variation der klassischen Art und beinhaltet Kreise statt Rechtecke *(Severino, o. J.).*

Streudiagramm
Ein *Streudiagramm (vgl. Abb. 4, Streudiagramm)* ist die graphische Darstellung von beobachteten Wertepaaren zweier statistischer Merkmale. Diese Wertepaare werden in ein kartesisches Koordinatensystem eingetragen, wodurch sich eine Punktwolke ergibt. Man erhofft sich durch das Muster der Punkte im *Streudiagramm* Informationen über die Abhängigkeitsstruktur der beiden Merkmale zu erkennen, die durch die Koordinaten repräsentiert werden *(Severino, o. J.).*

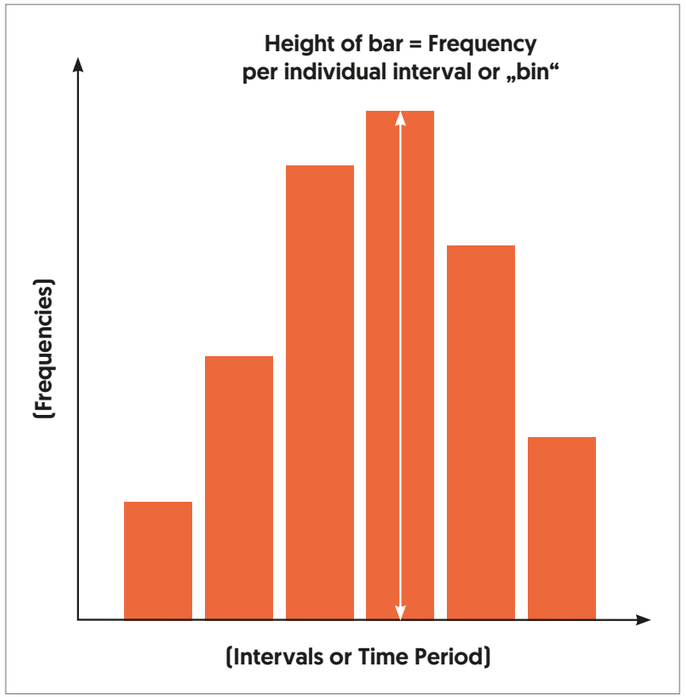


Abb. 1: *Histogramm*

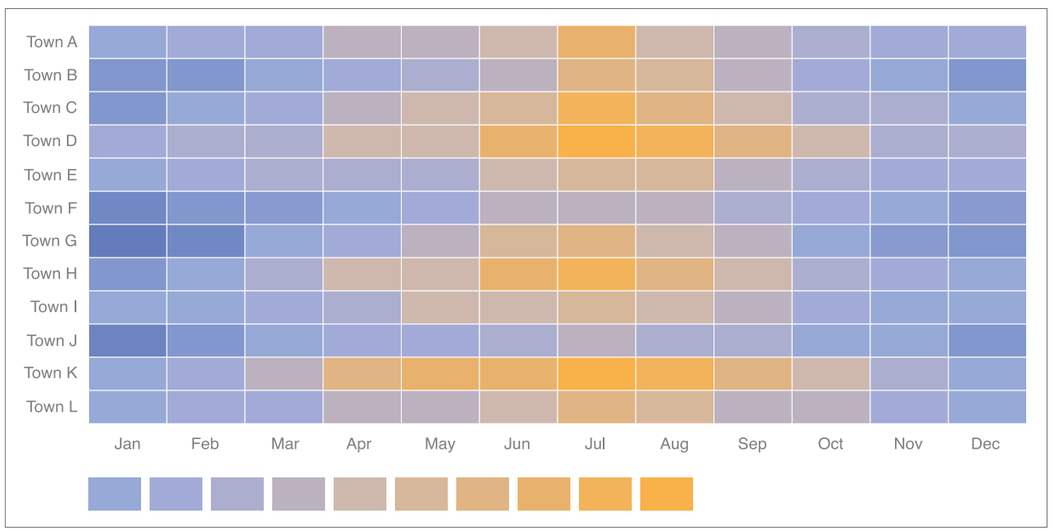


Abb. 2: *Heatmap*

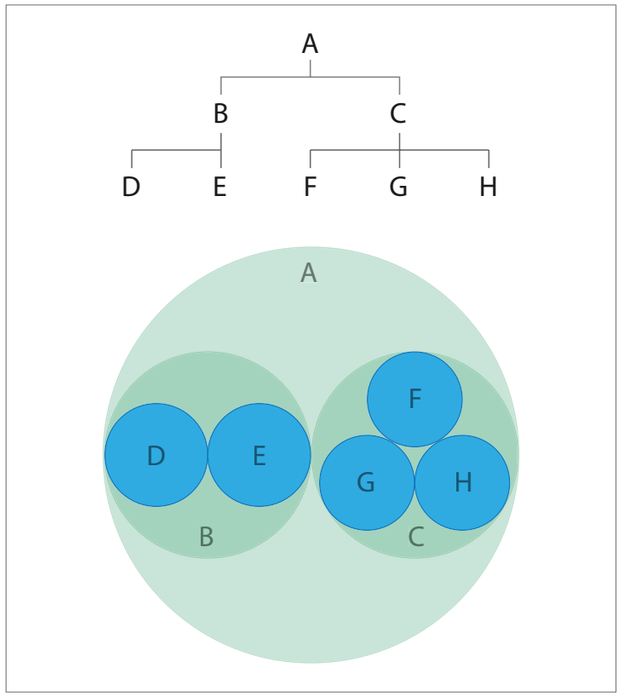


Abb. 3: *Treemap*

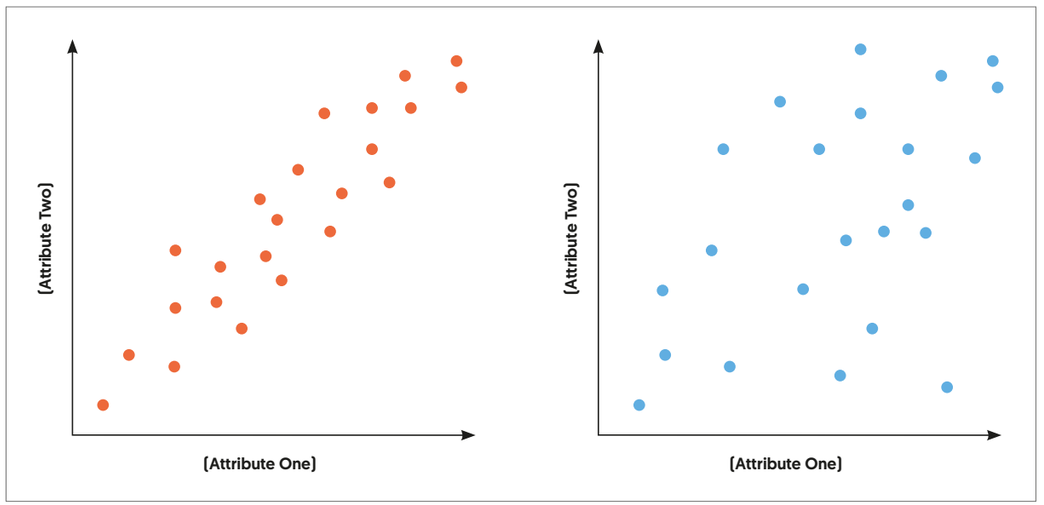


Abb. 4: *Streudiagramm*

2.3 Vergleichbare Arbeiten

ANYMAILS

Animails ist ein Projekt von *Carolin Horn* und wurde zusammen mit ihrem Betreuer *Prof. Brian Lucid* im Herbst 2006 und Frühjahr 2007 entwickelt. *Florian Jenett* programmierte den dazugehörigen Prototyp in *Processing*, einer Open-Source-Sprache, die auf der Programmiersprache *Java* basiert.

Die digitale Anwendung zeigt eine Visualisierung von *Carolin Horns* empfangenen Emails. Hierbei wurde untersucht, wie natürliche Metaphern benutzt werden können, um den Posteingang, dessen Struktur und Attribute zu visualisieren. Das Ziel der Anwendung ist dem Benutzer eine neue Erfahrung seiner Emailwelt zu bieten.

In dem Experiment werden Formen und Bewegungen verwendet um Informationen darzustellen. Wie kann durch Formen das Alter der Informationen angegeben werden? Wie sieht eine neue Email aus? Wie kann Bewegung die persönliche Bedeutung von Informationen verdeutlichen? Wie bewegen sich große oder kleine Informationen bezogen auf deren Dateigröße? *(Horn, 2007)*

Jede Email wird nach Kategorie, Alter und Lesestatus visualisiert. Dabei ergeben sich sechs Kategorien symbolisiert durch unterschiedliche Formen und Farben, sogenannte *Animals (vgl. Abb. 5, Kategorien Anymails)*.

Je nach Kategorie und Status bewegen sich die *Animals* unterschiedlich auf dem Display. Zunächst sieht man nur einen Schwarm von kleinen animierten „Tierchen“ und erkennt hierbei die Größenordung des Postfachs. Durch eine gezielte Auswahl verbinden sich die zusammengehörigen „Emailtiere“ zu einer Kette und helfen dem Betrachter Ordnung zu schaffen, um weitere Erkenntnisse zu erlangen. Ebenfalls kann man über eine Zeitleiste das Postfach an bestimmten Tagen untersuchen.

Um genauere Informationen über eine ausgewählte Email zu erhalten, hovert man über das gewünschte *Animal* und bekommt in einem Dialogfenster den Absender, Betreff und ähnliche Informationen angezeigt *(vgl. Abb. 6, Visualisierung Anymails)*.

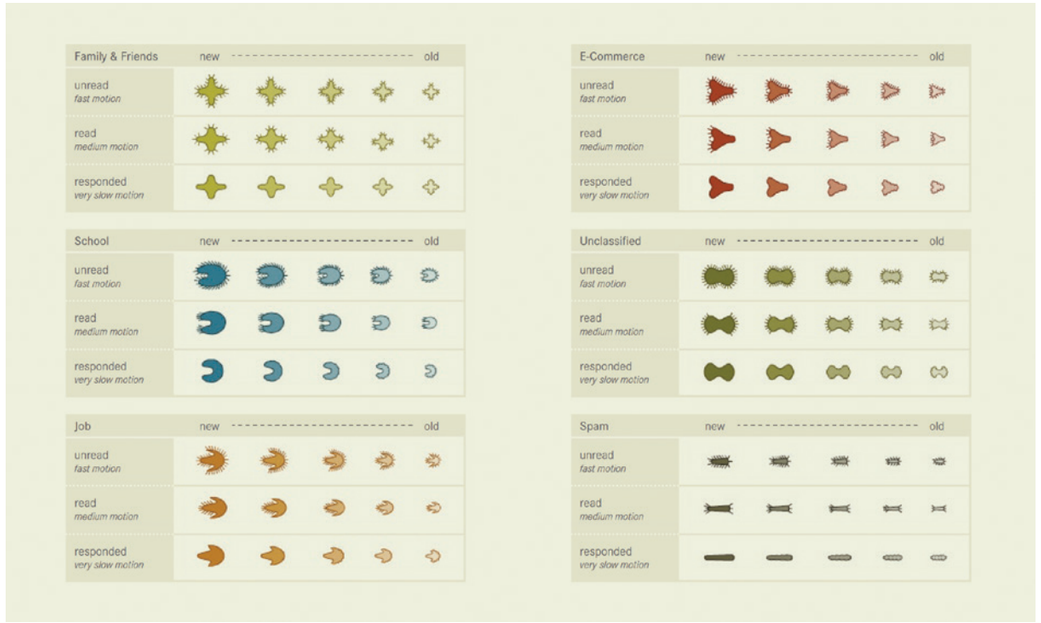


Abb. 5: Kategorien *Anymails*

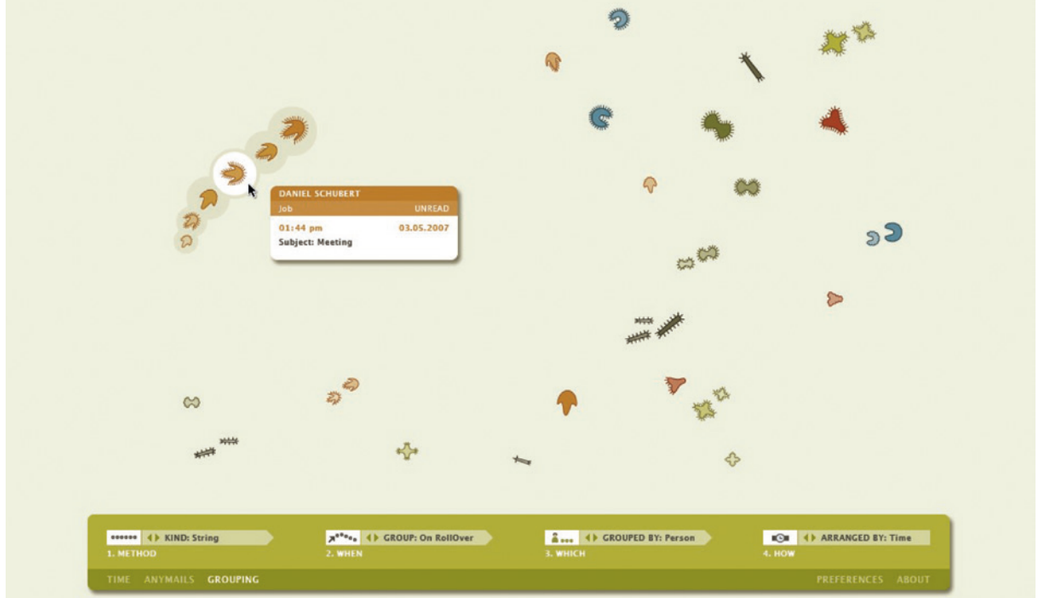


Abb. 6: Visualisierung *Anymails*

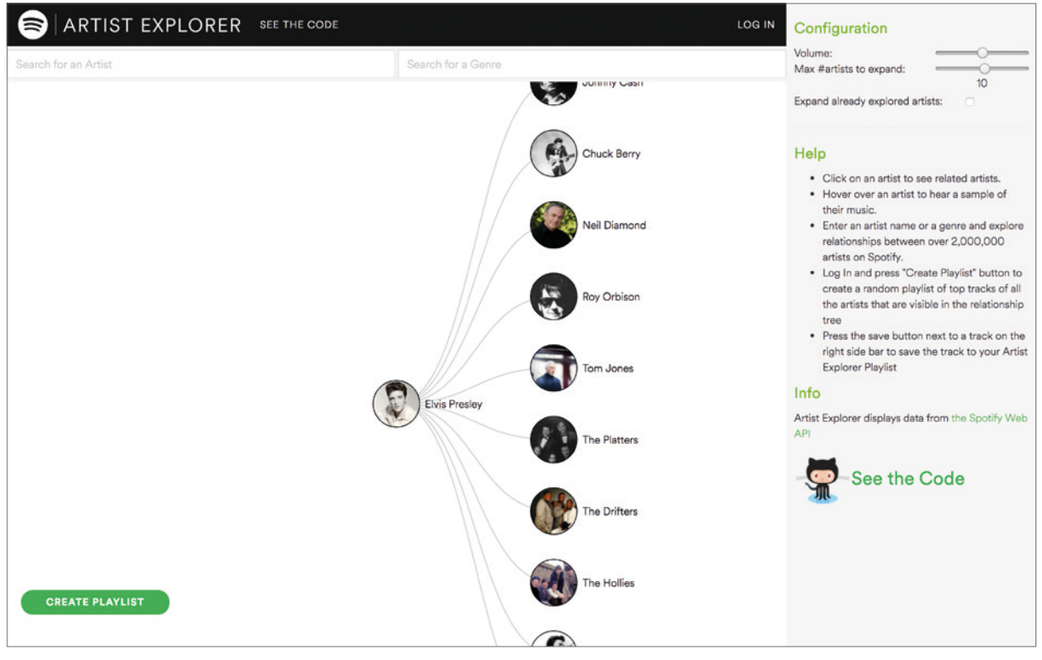


Abb. 7: Übersicht *Artist Explorer*

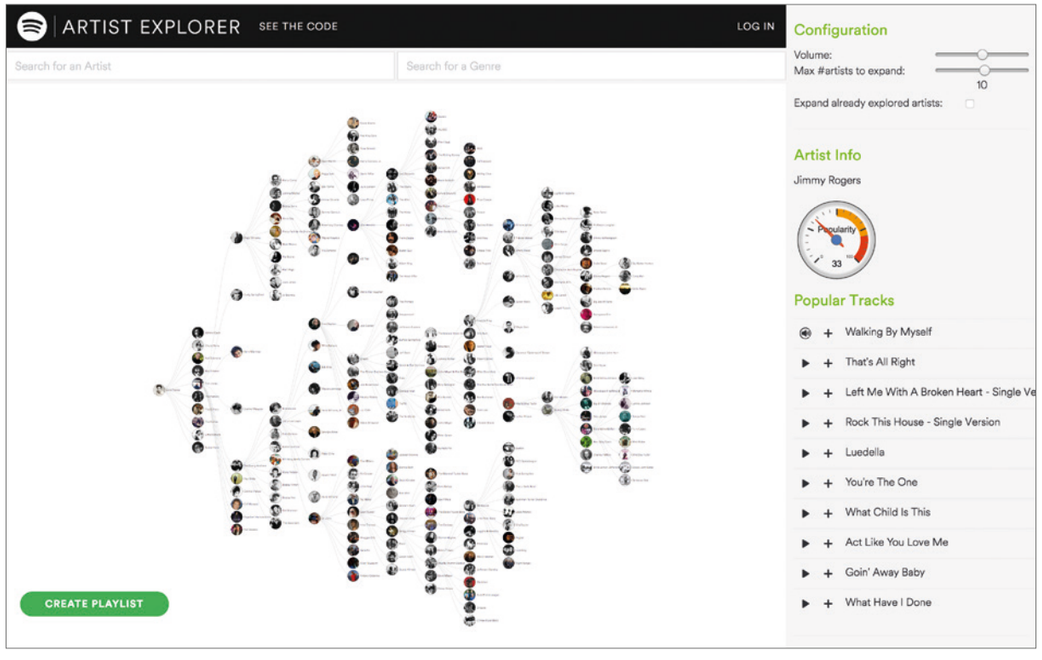


Abb. 8: Baumdiagramm *Artist Explorer*

ARTIST EXPLORER

Artist Explorer wurde von *Faruk Emre Sahin* entwickelt und hilft Nutzern die Beziehungen zwischen den über 2 000 000 Künstlern auf *Spotify* zu erkunden und ausgewählte Informationen aus der *Spotify* Web API und der *Echo Nest* API abzurufen. Eine genauere Begrifferläuterung zu *API* folgt unter dem Punkt *Konzeption* im Teil *Spotify Web API (Seite 28)* Die Visualisierung zeigt ein Baumdiagramm, dessen Aufbau bei wiederholter und längerer Benutzung der Anwendung stetig größer und komplexer wird. Durch die Struktur des Diagramms ist es jedoch dennoch möglich, schnell das Elter zu finden und den Künstler in Bezug zu anderen Künstlern zu setzen.

Um eine gezielte Suche zu ermöglichen, kann man über die Suchfelder einen Künstler oder ein Genre als Wurzelelement definieren und eine Suche starten. Es erscheint eine erste „Schicht“ mit zehn Künstlern, welche nur über ein Hoverevent genauer betrachtet werden können *(vgl. Abb. 7, Übersicht Artist Explorer).*

Zusätzlich dient eine Sidebar als Informationsgeber mit einer integrierten Playlist, welche die Songs des aktiv ausgewählten Künstlers beinhaltet und eine Popularitätsanzeige zum Einordnen des „Star-Grades“ des Musikers. Möchte man das Baumdiagramm um einem bestimmten Knoten erweitern, so klickt man auf den jeweiligen Künstler und es erscheinen weitere ähnliche Künstler *(vgl. Abb. 8, Baumdiagramm Artist Explorer).*

Die App nutzt die Datenvisualisierung zur Darstellung der Beziehungen zwischen den Künstlern und gleichzeitig als Navigation, was die Besonderheit der Anwendung kennzeichnet. Die unkonventionelle Art der Informationsbeschaffung über den „Künstlerbaum“ macht Spaß und ermöglicht einen schnellen Überblick. Ähnliche Künstler können direkt über das erste Elter gefunden werden. Sehr spannend wird es auch nach mehreren Knoten: Welcher Künstler steht mit wem über welche Ecken in Beziehung?

2.4 Fazit

Anymails beschäftigt sich intensiv mit Animation und spielerischer Interaktion. Animierte Charaktere dienen als Repräsentanten der ausgewählten Daten. *ArtistExplorer* nutzt dagegen das klassische Baumdiagramm für eine klare Ordnung der Daten, was zusätzlich als Navigation dient, um weitere Künstler entdecken zu können. Die Verknüpfung von Interaktion und Animation bieten verschiedenste Vorteile in Datenvisualisierungen, welche nun erläutert werden.

Verhaltensmetaphern
Durch Animationen können zusätzlich zu den grundlegenden Gestaltungselementen wie Form oder Farbe weitere Eigenschaften definiert und visualisiert werden, welche dem Betrachter eine einfachere Interpretation der visualisierten Daten ermöglichen. So könnte in einer Visualisierung die Geschwindigkeit einer Animation genutzt werden, um die Größe der Daten zu untermauern oder die Änderung von Formen zur Verdeutlichung verschiedener Zustände. Doch nicht nur inhaltlich erweitern Animationen die Visualisierung. Die Art und der Charakter eines Themas können visuell durch den Einsatz von Animationen gestärkt werden und so die Wahrnehmung der Informationen intuitiv in den richtigen Kontext setzen.

Benutzer einbeziehen
Durch Klick- und Hoverevents lassen sich zuvor unsichtbare Beziehungen und Informationen darstellen, welche die Visualisierungen dynamisch erweitern bzw. verändern. Der Betrachter bekommt die Aufgabe mit der Visualisierung zu interagieren, da nur so weitere Informationen offengelegt werden können. Zum Beispiel durch das Anklicken eines Objekts könnte eine Gruppierung mit anderen Inhalten farblich hervorgehoben werden. Der Benutzer lernt individuell spielerisch in kurzen Sequenzen die Inhalte kennen, welche ihn persönlich interessieren oder besonders auffallen.

Dies sind Vorteile, die klar für die Verwendung von Interaktion und Animation in Visualisierungen sprechen. Das Potenzial dieser positiven Eigenschaften ist groß und wird zunehmend in Datenvisualisierungen angewendet.

Ebenfalls soll im Konzept und der Umsetzung auf die entwickelten Musikeigenschaften und Techniken aus den verschiedenen Visualisierungstypen zurückgegriffen werden. Das Erzeugen von Mustern, wie im Streudiagramm oder die Gegenüberstellung von Werten durch die Größe, wie es bei einem Histogramm üblich ist, sind wichtige Erkenntnise für die angestrebte Datenvisualisierung.

3. WEBENTWICKLUNG

3.1 Übersicht der Frameworks

BACKEND FRAMEWORK
Node.js ist eine serverseitige Plattform in der Softwareentwicklung zur Ausführung von Netzwerkanwendungen. *Node.js* wird in der JavaScript-Laufzeitumgebung *V8* ausgeführt, die ursprünglich für *Google Chrome* entwickelt wurde und bietet daher eine ressourcensparende Architektur, die eine besonders große Anzahl gleichzeitig bestehender Netzwerkverbindungen ermöglicht. 2009 wurde das Projekt von *Ryan Dahl* ins Leben gerufen und hat seitdem stetig an Bekanntheit gewonnen *(SpringerGmbH, 2013)*.

„Als asynchrone, eventbasierte Laufzeitumgebung wurde Node.js speziell für die Entwicklung von skalierbaren Netzwerkanwendungen entworfen. Bei jeder neuen Anfrage wird die Callback-Funktion ausgeführt. Gibt es jedoch nichts zu tun, befindet sich Node im Ruhezustand. Dies steht im Gegensatz zu den heutzutage üblichen Modellen für Nebenläufigkeit, bei denen Threads des Betriebssystems genutzt werden. Threadbasiertes Networking is vergleichsweise ineffizent and sehr schwer umzusetzen. Zudem müssen sich Nodenutzer nicht um Deadlocks im Prozess sorgen, da es keine Blockierung gibt.“ (SpringerGmbH, 2013).

Für ein geeignetes und übersichtliches Routing ist *Express.js* empfehlenswert. *Express.js* ist ein einfaches und flexibles *Node.js*-Framework das zahlreiche leistungsfähige Features und Funktionen für Web- und mobile Anwendungen bereitstellt. Mithilfe unzähliger HTTP-Dienstprogrammmethoden und Middlewarefunktionen gestaltet sich das Erstellen einer leistungsfähigen API sehr schnell *(Gorski u. a., 2015)*.

DATENBANKANBINDUNG

MongoDB ist eine schemafreie, dokumentenorientierte NoSQL-Datenbank, die in der Programmiersprache C++ geschrieben ist. Da die Datenbank dokumentenorientiert ist, kann sie Sammlungen von JSON-ähnlichen Dokumenten verwalten, sogenannte BSON Dateien. Anwendungen können Daten auf natürlichere Weise modellieren, da die Daten zwar in komplexen Hierarchien verschachtelt werden können, dabei aber immer abfragbar, indizierbar und erweiterbar bleiben.

Die Entwicklung von *MongoDB* begann im Oktober 2007. 2012 war sie die am weitesten verbreitete NoSQL-Datenbank und ist zusätzlich Open Source *(NodeCode, 2013).*

Mongoose ist eine *JS*-Bibliothek und umfasst integrierte Funktionalitäten, um mit der Datenbank zu interagieren. Außerdem bietet sie eine einfache, schemabasierte Lösung für die Modellierung von Anwendungsdaten.

PROMISE KONTROLLIEREN

When.js ist eine Promises / A+ Implementierung, welche die nativen Promises von *ES6* sinnvoll erweitert. Arrays und Hashes von Promises werden sortiert und geordnet in Promise-Sequenzen gespeichert. Die geordneten Tasks werden parallel oder nacheinander ausgeführt und in richtiger Reihenfolge zurückgegeben. Node-style und andere callback-based APIs werden in promise-based APIs transformiert *(cujojs, 2013).*

REST ARCHITEKTUR

Representational State Transfer (abgekürzt *REST*) bezeichnet ein Programmierparadigma für Webservices. *REST* hat das Ziel, einen Architekturstil zu schaffen, der die Anforderungen des modernen Web besser darstellt. Dabei unterscheidet sich *REST* vor allem in der Forderung nach einer einheitlichen Schnittstelle von anderen Architekturstilen.

Der Zweck von *REST* liegt schwerpunktmäßig auf der Maschine-zu-Maschine-Kommunikation. Anders als bei vielen verwandten Architekturen codiert *REST* keine Methodeninformation in den URI. Der Vorteil von *REST* liegt darin, dass im *WWW* bereits ein Großteil der für *REST* nötigen Infrastruktur (z. B. Web- und Application-Server, HTTP-fähige Clients, HTML- und XML-Parser, Sicherheitsmechanismen) vorhanden ist und viele Webdienste per se *REST*-konform sind *(Fielding, 2000).*

STRUKTURIERUNG

Webpack ist ein Framework zur Strukturierung von Code. Dabei werden Code und statische Assets in Module aufgeteilt und in einem Modulbündel gespeichert. Dabei ist es möglich, alle Teile des Modulbündels anzupassen und externe Bibliotheken zu integrieren. Das Framework zeichnet sich zusätzlich durch eine kurze Ladezeit aus und ist für große Projekte geeignet.

3.2 Spezieller Einsatz in der App

Die *Spotify* API ist eine *REST*ful API. Grundsätzlich ist sie eine Schnittstelle, die Programme verwenden können, um Spotifydaten über das Internet abzurufen und zu verwalten. Die Web API verwendet dasselbe HTTP-Protokoll, das von jedem Internetbrowser verwendet wird. Jede Anwendung kann Daten von den *Spotify* Web API-Endpunkten anfordern.

Wenn die Anwendung jedoch Zugriff auf persönliche Daten eines Benutzers (Profil, Wiedergabelisten, usw.) sucht, muss sie registriert sein. Registrierte Anwendungen erhalten auch andere Vorteile, wie höhere Rate Grenzen *(vgl. Abb. 9, Kommunikation Spotify).*

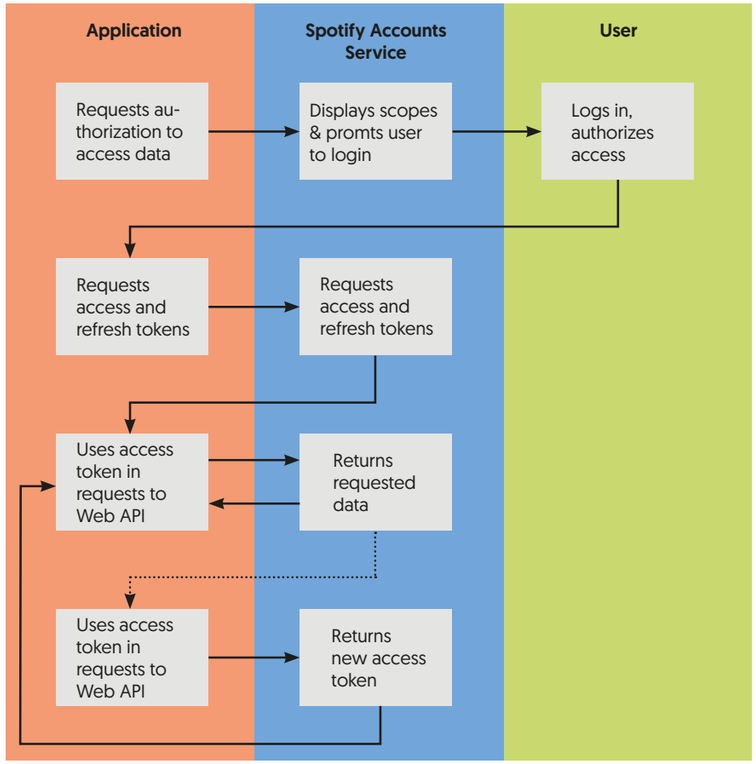


Abb. 9: Kommunikation *Spotify*

Um mit der API kommunizieren zu können, muss die Serverumgebung vorbereitet werden. Da ich bereits Kenntnisse im Backend mit Frameworks wie *Django* und *Web2Py* gesammelt habe, war es mir wichtig meinen Horizont mit *Node.js* zu erweitern. Ebenfalls ist die Community hinter *Node.js* sehr aktiv und mithilfe vom *Node Package Manager* lassen sich Module einfach und schnell installieren.

Hinzu kommt, dass *Spotify* ein *Beginners Guide Tutorial* mit *Node.js* anbietet, was mir den Einstieg wesentlich erleichterte. Wie bereits beschrieben ist *Node.js* ein Framework für serverseitige Programmierung. Wie einfach *Node.js* funktioniert, zeigt die folgende Abbildung von einem vollfunktionstüchtigen Server *(vgl. Abb. 10, Node Server).*

user.js server_test.js

```
1
2  const http = require('http');
3
4  const hostname = '0.0.0.0';
5  const port = 4000;
6
7  const server = http.createServer((req, res) => {
8      res.statusCode = 200;
9      res.setHandler('Content-Type', 'text/plain');
10     res.end('Hello world\n');
11 });
12
13 server.listen(port, hostname, () => {
14     console.log(`Server running at: http://${hostname}:${port}/`);
15 });
```

Abb. 10: *Node* Server

Um eine leistungsfähige API entwickeln zu können, müssen die eingehenden Pfade (Routes) implementiert werden. Hierbei dient *Express.js* als Framework für den *Node*-Server. In der ersten Schicht werden drei Pfade für den Index, die Ansicht der Visualisierung und die API erstellt. Die API dient dabei als Schnittstelle für den Client zum Sever, um die Spotifydaten laden zu können. Die anderen beiden Pfade dienen für den Aufbau der HTML-Seiten.

Zusätzlich werden relevante Daten eines jeden Users in einer Datenbank gespeichert, damit weiterführende Requests vom Client an *Spotify* über den Server vermittelt werden können. Es ist wichtig, dass die Authentifizierungskeys der hier vorliegenden App nicht an den Client geschickt werden, um deren Missbrauch zu vermeiden.

Als Datenbank dient *MongoDB,* eine NoSQL (not only SQL) Datenbank, welche für moderne Webentwicklung ausgelegt und entwickelt wurde. Der große Vorteil einer *MongoDB* ist, dass Objekte (BSON) direkt gespeichert werden können. Diese können jederzeit verändert werden, ohne dass die Datenarchitektur rechenaufwendig umgestellt werden muss. Ebenfalls wird für BigData-Projekte NoSQL empfohlen und standardmäßig zusammen mit *Node.js* verwendet. Zur Implementierung von *MongoDB* dient das Framework *Mongoose.* Die folgende Abbildung zeigt *MongoDB* und *Mongoose* im Einsatz der Applikation *Klangspektrum.* Hiebei wird eine passwortgeschützte Verbindung zur Datenbank erzeugt und ein Model definiert *(vgl. Abb. 11, Mongo Model und Abb. 12, Mongo Connect).*

user.js

```
1  var mongoose = require('mongoose');
2  var Schema = mongoose.Schema;
3
4  var userSchema = new Schema({
5      created: {
6          type: Date,
7          default: Date.now
8      },
9      user: String,
10     access_token: String,
11     refresh_token: String,
12     name: String,
13     stracks: [],
14     ptracks: [],
15     alltracks: [],
16     image: [],
17     playlists: [],
18     ownPlaylist: []
19 });
20
21 mongoose.model('KlangUser', userSchema);
```

Abb. 11: *Mongo* Model

server.js

```
49 // Init database connection
50 mongoose.Promise = global.Promise;
51
52 // Build the connection string
53 //var dbURI = 'mongodb://localhost/KlangUser';
54 var dbURI = 'mongodb://0.0.0.0:27017';
55
56 // Create the database connection
57 mongoose.connect(dbURI, {
58     user: 'der-dirigent',
59     pwd: 'klangspektrum',
60     //mongos: true
61 });
62
63 // When successfully connected
64 mongoose.connection.on('connected', function () {
65     console.log('Mongoose default connection open to ' + dbURI);
66 });
67
68 // If the connection throws an error
69 mongoose.connection.on('error', function (err) {
70     console.log('Mongoose default connection error: ' + err);
71 });
72
73 // When the connection is disconnected
74 mongoose.connection.on('disconnected', function () {
75     console.log('Mongoose default connection disconnected');
76 });
77
78 // If the Node process ends, close the Mongoose connection
79 process.on('SIGINT', function () {
80     mongoose.connection.close(function () {
81         console.log('Mongoose default connection disconnected through app termination');
82         process.exit(0);
83     });
84 });
```

Abb. 12: *Mongo* Connect

Da die Umgebung nun fertig eingerichtet ist, kann die Umsetzung der *REST* API beginnen. Diesbezüglich wurde, wie schon erwähnt, eine Route „api“ erzeugt, welche für die Requests des Clients zur Verfügung steht. Die API beinhaltet 13 Pfade. Der Client kann sich über die API einloggen und bekommt über einen Callback wichtige Zugangsdaten gesendet. Im nächsten Schritt wird über „loadMe“ das Profil geladen. Mithilfe der weiteren Pfade in der API ist es möglich Songs, Playlists, Features und Artists abzufragen. Eine Übersicht der fertigen API zeigt die folgende Abbildung *(vgl. Abb. 13, REST API Klangspektrum).*

Damit die versendeten Anfragen an *Spotify* nicht zu einem Error und somit zu Blockierungen führen, wurde mit *When.js* festgelegt, wieviel Anfragen (Requests) parallel in Sequenzen ausgeführt werden dürfen. Ein großer Vorteil ist, dass alle Antworten (Response) der Anfragen geordnet im Array zurückgegeben werden.

Auf Seiten des Clients wird über window.onload() automatisch die Abfrage der Daten des Nutzers gestartet und mit einem Ladescreen verbunden. Für ein einfaches Handling der Requests auf der Seite des Clients wird *Fetch.js* verwendet. Nach Ablauf der Requests an den Server startet nach erfolgreichem Sortieren der *p5* Sketch.

user.js | api.js

```
8
9   router.get('/login/:user', login.get);
10  router.get('/callback', auth.get);
11  router.get('/loadMe/:access_token/:user', load.getMe);
12  router.get('/loadSavedTracks/:access_token/:user/:offset/:count/:first_time', load.getSavedTracks);
13  router.get('/loadPlaylists/:access_token/:user/:offset/:count/:first_time/:user_id', load.getPlayli
14  router.get('/loadPlaylistTracks/:access_token/:user/:user_id/:pindex', load.getPlaylistTracks);
15  router.get('/loadAllTracks/:user', load.createAllTracks);
16  router.get('/loadFeatures/:access_token/:user/:offset', load.getFeatures);
17  router.get('/loadArtists/:access_token/:user/:offset', load.getArtists);
18
19  router.get('/sortData', (request, response) => {
20      // sort data and send to client -> callfunction with client_id!!
21      // https://scotch.io/tutorials/build-a-restful-api-using-node-and-express-4
22      var data = { 'data': 477 }
23      setTimeout(function () {
24          response.json(data)
25      }, 10000);
26  });
27
28  router.get('/startApp', (request, response) => {
29      // call client function
30      setTimeout(function () {
31          response.json(data)
32      }, 10000);
33  });
34
35  router.get('/logout', (request, response) => {
36      // Auth. löschen und redirect auf Startseite
37      response.redirect('/');
38  });
39
40  router.get('/download', (request, response) => {
41      var file = __dirname + '/../download_folder/test.pdf';
42      response.download(file); // Set disposition and send it.
43  });
```

Abb. 13: *REST* API *Klangspektrum*

4. KONZEPTION

4.1. Spotify Web API

Für die Konzeption ist es enorm wichtig, welche Informationen die *Spotify* API bereithält, da sich hieraus Teile des Konzepts entwickeln, welche nun dokumentarisch erläutert werden. Die *Spotify* Web API ermöglicht es Entwicklern in deren Anwendungen Daten aus dem Musikkatalog *Spotifys* zu verwenden. Die Endpoints senden nach erfolgreicher Anfrage eine JSON-Datei zurück, die unter anderem Informationen über Künstler, Alben und Tracks enthält. Abhängig von der Berechtigung des jeweiligen Benutzers kann die API Entwicklern auch Zugriff zu nutzerbezogene Daten, wie zum Beispiel Wiedergabelisten, gespeicherten Songs und Alben von Benutzerbibliotheken bereitstellen.

Im Speziellen gibt es folgende Endpoints: Albums, Artists, Browse, Follow, Library, Personalization, Playlists, Profiles, Search und Tracks. Mithilfe der *Spotify* Console ist es möglich die Endpoints mittels Beispieldaten zu erforschen. Ebenfalls kann man sich in der Referenz genaue Informationen über jeden einzelnen Endpoint und dessen zugehöriges JSON-Objekt einholen.

Sehr interessant ist hierbei der Endpoint Audio-Features. Über die *Spotify*ID eines Songs erhält man ein JSON-Objekt mit verschiedenen Informationen zu Eigenschaften des ausgwählten Songs *(vgl. Abb. 14, AudioFeatures JSON)*.


```
{
  "danceability" : 0.587,
  "energy" : 0.965,
  "key" : 11,
  "loudness" : -4.106,
  "mode" : 1,
  "speechiness" : 0.101,
  "acousticness" : 0.0362,
  "instrumentalness" : 0,
  "liveness" : 0.138,
  "valence" : 0.818,
  "tempo" : 129.972,
  "type" : "audio_features",
  "id" : "1zHlj4dQ8ZAtrayhuDDmkY",
  "uri" : "spotify:track:1zHlj4dQ8ZAtrayhuDDmkY",
  "track_href" : "https://api.spotify.com/v1/tracks/1zHlj4dQ8ZAtrayhuDDmkY",
  "analysis_url" : "https://api.spotify.com/v1/audio-analysis/1zHlj4dQ8ZAtrayhuDDmkY",
  "duration_ms" : 204053,
  "time_signature" : 4
}
```


Abb. 14: AudioFeatures JSON

Im Bezug auf die Zielsetzung der Bachelorarbeit sollen diese Songeigenschaften als Ergebnisse aus komplexen Berechnungen und Algorithmen der Hintergrundprozesse interpretiert werden. Die Annahme ist nun, dass *Spotify* auf Basis dieser Werte individuelle Musikvorschläge und personalisierte Playlisten erzeugen kann, die nun durch die Datenvisualisierung eingesehen werden können.

Um die Werte später in einer Visualisierung gut vergleichen zu können, müssen die verschiedenen Songeigenschaften nach einem gemeinsamen Wertesystem funktionieren, was die folgenden Kategorien erfüllen. Ein Song wird somit durch Werte zwischen 0 und 1 in den verschiedenen Kategorien beschrieben: Danceability, Energy, Speechiness, Instrumentalness, Liveness, Acousticness und Mood.

Danceability beschreibt wortwörtlich übersetzt die Tanzbarkeit eines Songs.
Energy repräsentiert die Intensität und Aktivität des Songs.
Speechiness untersucht den Song auf gesprochene Worte und differenziert diese von Gesang.
Instrumentalness verzeichnet die Instrumentalanteile eines Songs, also die Anteile ohne Gesang oder gesprochene Worte.
Liveness gibt an, inwieweit ein Song live ist und erkennt beispielsweise Publikumsgeräusche in der Aufnahme.
Acousticness zeigt auf wie akustisch ein Song ist.
Mood beschreibt die Stimmung eines Songs in einem Spektrum von 0 bis 1, bzw. von traurig nach fröhlich.

Um die Kategorien und deren Werte besser nachvollziehen zu können, ist es notwendig den dazugehörigen Song anzuhören. Hierdurch erkennt man schnell, ob der Wert zum Song passt oder ob die Musikerkennung *Spotifys* eine andere Einstufung als erwartet getroffen hat. Bei manchen Liedern zeigt sich auch, dass die Musikerkennung in der heutigen Zeit noch nicht vollständig fehlerfrei ist und die Algorithmen stets verbessert werden müssen. Um Lieder eines Nutzers abspielen zu können bietet *Spotify* über die *REST* API mehrere Möglichkeiten an. Am einfachsten ist es hierbei über den Endpoint Tracks und die SongID die Preview-URL zu laden.

Um das Musikverhalten eines Nutzer umfassend visualisieren zu können, benötigt man alle Songs, die der Nutzer hört und abspeichert, sowie den dazugehörigen zeitlichen Verlauf. Hierzu musste jedoch aus technischen Gründen auf das aktive Hörverhalten verzichtet werden und deshalb auf das Musikarchiv als Grundlage des Musikverhaltens zurückgegriffen werden. Betrachtet man ein Spotifyprofil, so hat jeder Nutzer die Möglichkeit seine Songs in Playlisten oder in einer persönlichen Musiksammlung abzulegen.

```
{
  "album" : {
    "album_type" : "single",
    "artists" : [ {
      "external_urls" : {
        "spotify" : "https://open.spotify.com/artist/0TnOYISbd1XYRBk9myaseg"
      },
      "href" : "https://api.spotify.com/v1/artists/0TnOYISbd1XYRBk9myaseg",
      "id" : "0TnOYISbd1XYRBk9myaseg",
      "name" : "Pitbull",
      "type" : "artist",
      "uri" : "spotify:artist:0TnOYISbd1XYRBk9myaseg"
    } ],
    "external_urls" : {
      "spotify" : "https://open.spotify.com/album/3X33e7UII5loqrEgauOKEC"
    },
    "href" : "https://api.spotify.com/v1/albums/3X33e7UII5loqrEgauOKEC",
    "id" : "3X33e7UII5loqrEgauOKEC",
    "images" : [ {
      "height" : 640,
      "url" : "https://i.scdn.co/image/2b8490f45660572fda9732df66adb83fde98a2a2",
      "width" : 640
    }, {
      "height" : 300,
      "url" : "https://i.scdn.co/image/e779257376593e7a3ba1f39e66660a3d938feda0",
      "width" : 300
    }, {
      "height" : 64,
      "url" : "https://i.scdn.co/image/2fbda0496761c99c8289a2a4bfa2e710af5ea2d1",
      "width" : 64
    } ],
    "name" : "Timber",
    "type" : "album",
    "uri" : "spotify:album:3X33e7UII5loqrEgauOKEC"
  },
  "artists" : [ {
    "external_urls" : {
      "spotify" : "https://open.spotify.com/artist/0TnOYISbd1XYRBk9myaseg"
    },
    "href" : "https://api.spotify.com/v1/artists/0TnOYISbd1XYRBk9myaseg",
    "id" : "0TnOYISbd1XYRBk9myaseg",
    "name" : "Pitbull",
    "type" : "artist",
    "uri" : "spotify:artist:0TnOYISbd1XYRBk9myaseg"
  }, {
    "external_urls" : {
      "spotify" : "https://open.spotify.com/artist/6LqNN22kT3074XbTVUrhzX"
    },
    "href" : "https://api.spotify.com/v1/artists/6LqNN22kT3074XbTVUrhzX",
    "id" : "6LqNN22kT3074XbTVUrhzX",
    "name" : "Kesha",
    "type" : "artist",
    "uri" : "spotify:artist:6LqNN22kT3074XbTVUrhzX"
  } ],
  "disc_number" : 1,
  "duration_ms" : 204053,
  "explicit" : false,
  "external_ids" : {
    "isrc" : "USRC11301695"
  },
  "external_urls" : {
    "spotify" : "https://open.spotify.com/track/1zHlj4dQ8ZAtrayhuDDmkY"
  },
  "href" : "https://api.spotify.com/v1/tracks/1zHlj4dQ8ZAtrayhuDDmkY",
  "id" : "1zHlj4dQ8ZAtrayhuDDmkY",
  "is_playable" : true,
  "name" : "Timber",
  "popularity" : 69,
  "preview_url" : "https://p.scdn.co/mp3-preview/ecb1639c58ceb6311919373fe56399e3ec
  "track_number" : 1,
  "type" : "track",
  "uri" : "spotify:track:1zHlj4dQ8ZAtrayhuDDmkY"
}
```


Abb. 15: Tracks JSON

Über die Endpoints *Playlist* und *Library* kann auf diese Musik zugegriffen werden und Informationen wie Titel, Datum und PreviewURL herausgefiltert werden *(vgl. Abb. 15, Tracks JSON)*.

Im Hinblick auf die Zielsetzung und der daraus geplanten interaktiven Datenvisualisierung ergibt sich aus der *Spotify* Web API die Möglichkeit, das Musikverhalten bezüglich des Musikarchivs anhand von Songeigenschaften zu visualisieren. Um die abstrakten Songeigenschaften und Werte besser zu verstehen, ist es zusätzlich möglich über eine Detailansicht die zugehörigen Songs abzuspielen und dessen Genres anzuzeigen.

4.2 Klangspektrum

Die Visualisierung von Prozessen und Daten ist aus mehreren Gründen signifikant: sie ermöglichen es, Informationen schnell zu erkennen, große Mengen an Daten zu verstehen, komplexe Prozesse und Zusammenhänge zu verdeutlichen, Trends und Muster in Datensystemen aufzudecken, sowie Daten nützlich zugänglich zu machen, damit sie interpretiert werden können *(Seifert, 2009)*.

Aus diesem Kontext heraus wird die interaktive Datenvisualisierung Spotifynutzern eine Analyse des eigenen Musikverhaltens basierend auf vereinfachten Songeigenschaften ermöglichen. Der Nutzer soll die abstrakten Werte der unterschiedlichen Kategorien mithilfe der bekannten Messgrößen Songs und Genres in Bezug setzen können. Der Betrachter wird bildlich gesprochen zum Algorithmus, um das eigene Musikprofil zu analysieren. Die Erkenntnisse durch die Analyse und Interaktion sollen durch eine einmalige Benutzung gewonnen werden können und setzen keine wiederholte Nutzung der App voraus.

Zur Erforschung der Entwicklung des eigenen Musikverhaltens soll es über eine Timeline ermöglicht werden, jeden Monat separat zu betrachten. Ebenfalls soll der Nutzer die Möglichkeit haben, über eine sortierte Playlist alle Songs abspielen zu können, welche in einer Songkategorie den gleichen Wert aufweisen.

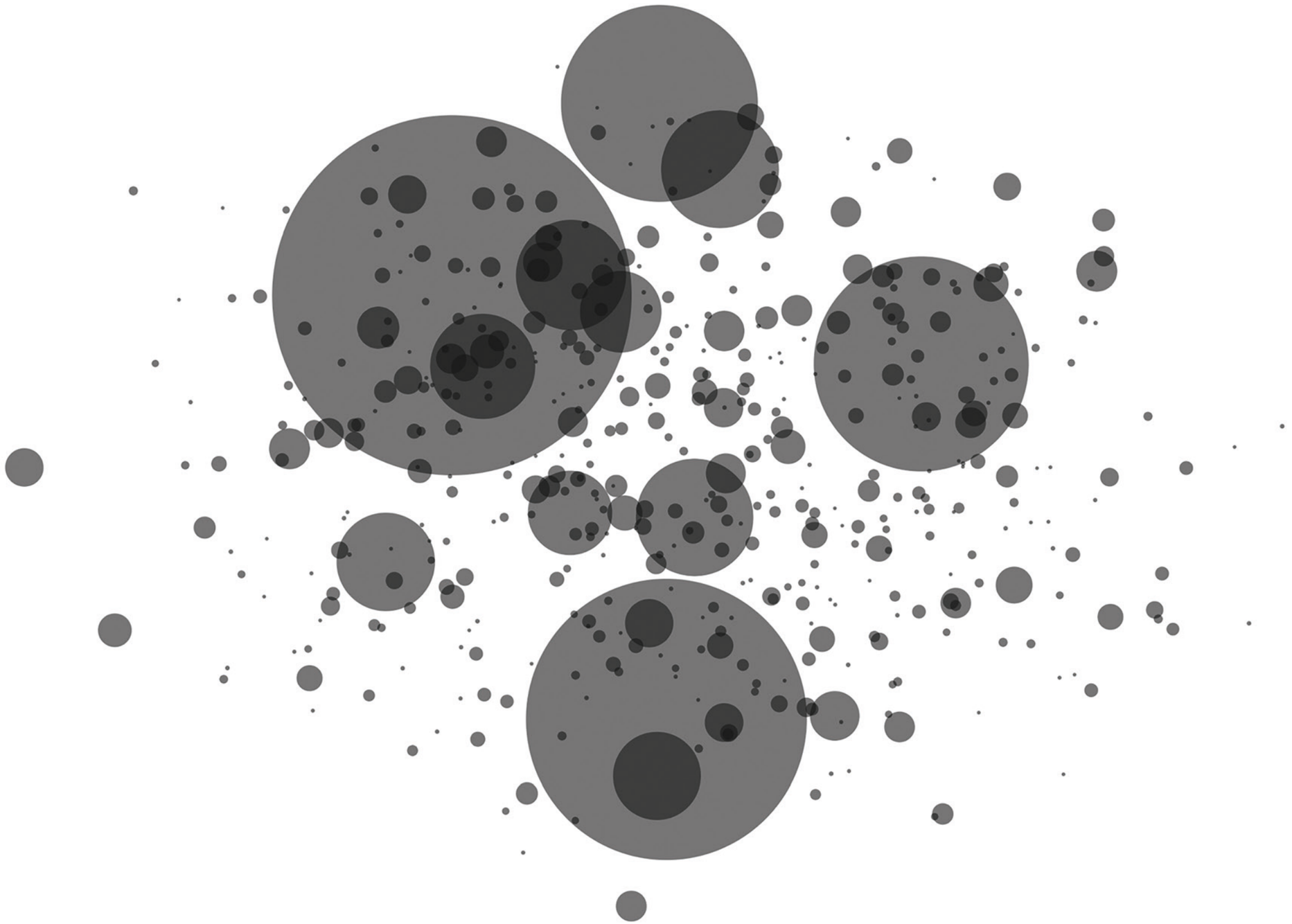
Der Nutzer kann so die Songeigenschaften und Werte in gewohnte reale Bezüge setzen und erlernt ein neues Verständnis zu den Hintergrundprozessen *Spotifys*.

Die Visualisierung wird neben den Informationen selbst auch deren natürliche Dynamik aufzeigen. Dazu zählen die Struktur, Navigation, Interaktivität, Visualisierung und Präsentation der Inhalte. Visuelle Verhaltensmetaphern hauchen den Informationen Leben ein und schaffen reichhaltige Erfahrungen für die Nutzer. Diesbezüglich sollen die Werte der Kategorien im „Schwarm“ analysiert werden können, um ein größtmögliches Interaktionsfeld zu bieten. Die Werte können so zum Beispiel spielerisch eingefangen und erforscht werden. Des Weiteren bietet sich die Form des Schwarms an, da alle Werte in direkter Relation stehen sollen, um Schwerpunkte und Muster schnell und einfach erkennbar zu machen. Zusätzlich sollen durch Animation und Audiosignale das Thema Musik und der Informationsgehalt verstärkt dargestellt werden.

Die Anwendung soll im Browser stattfinden, damit möglichst viele Menschen ihr Musikverhalten analysieren können.

KLANG
5PΣKTRUM

— 01
— 12
— 11
2016 — 10
— 09



x1
x2
x4
x8

sort

?

5. UMSETZUNG

5.1 Gestaltung

5.1.1 Visualisierung

GESTALTERISCHER BEZUG ZUM THEMA MUSIK – VERHALTENSMETAPHERN

Wie in dem Inhaltspunkt *Recherche* bereits erwähnt, besteht Musik aus einzelnen Tönen und Schwingungen und wird meist mit bunten Farben assoziiert. Diese Eigenschaften soll die Visualisierung aufgreifen. Kreise als harmonische Formen visualisieren die Werte der einzelnen Kategorien. Die Kategorien werden über einen Farbcode getrennt und in den Farben Grün, Blau, Cyan, Magenta, Lila, Orange und Gelb dargestellt. Da Musik aus der Komposition verschiedener Töne besteht, soll auch die Visualisierung jeden einzelnen Wert aufzeigen, woraus eine Art Schwarm entsteht.

Verhaltensmetaphern sollen die Verbindung zum Thema Musik durch Bewegung, Skalierung und Ton unterstreichen. Die Kreise sollen sich zufällig und frei im Raum bewegen. Jeder Kreis hat einen Puls. Dabei steht die Frequenz des Pulses im direkten Bezug zu dem durch den Kreis visualisierten Wert. Ein hoher Wert impliziert eine hohe Pulsfrequenz. Ebenfalls wird durch das Interagieren mit den Kreisen ein Ton erzeugt, welcher der (Puls-) Frequenz und somit dem Wert des Kreises angepasst ist. Die Bewegung und das Pulsieren der Kreise erzeugen eine natürliche Interpretation von scheinbar zufälligen Geräuschen. Durch einen Blur-Filter wird dieser Effekt zusätzlich verdeutlicht.

EIN PROFIL IM VERGLEICH

Damit der Betrachter das Musikverhalten über die Zeitabstände schnell einschätzen und vergleichen kann, zeigt die Visualisierung zunächst nur graue Kreise. Ähnlich wie bei einem Streudiagramm soll der Nutzer durch die unterschiedlichen aus Kreisen bestehenden Muster Informationen herauslesen können. Diesbezüglich kann erstmals das Musikspektrum interpretiert werden.

Die Größe eines Kreise steht für die Anzahl an Songs, die dessen spezielle Songeigenschaften teilen. Je größer der Kreis heranwächst, desto wichtiger ist die Eigenschaft im Bezug zum Musikverhalten des Nutzers, wodurch sich Schwerpunkte und Aktivität ablesen lassen. Dieses Prinzip folgt dem eines Histogramms, woraus ebenfalls durch die Ausprägung der Größe der Balken Trends und Erkenntnisse gewonnen werden können.

Zusammen mit den Kreisgrößen und der Musterbildung ergeben sich klare Verhaltensprofile. Sind zum Beispiel nur wenige Kreise zu sehen, so hat der Nutzer entweder ein sehr spezialisiertes Musikverhalten, was durch einzelne große Kreise symbolisiert wird oder nur sehr wenige Lieder abgespeichert, was sich durch wenige kleine Kreise ausdrückt. Ebenfalls können viele kleine Kreise auf ein experimentelles Musikverhalten hinweisen oder viele große Kreise auf einen breiten Musikhorizont *(vgl. Abb. 16 Profil Vergleich)*.

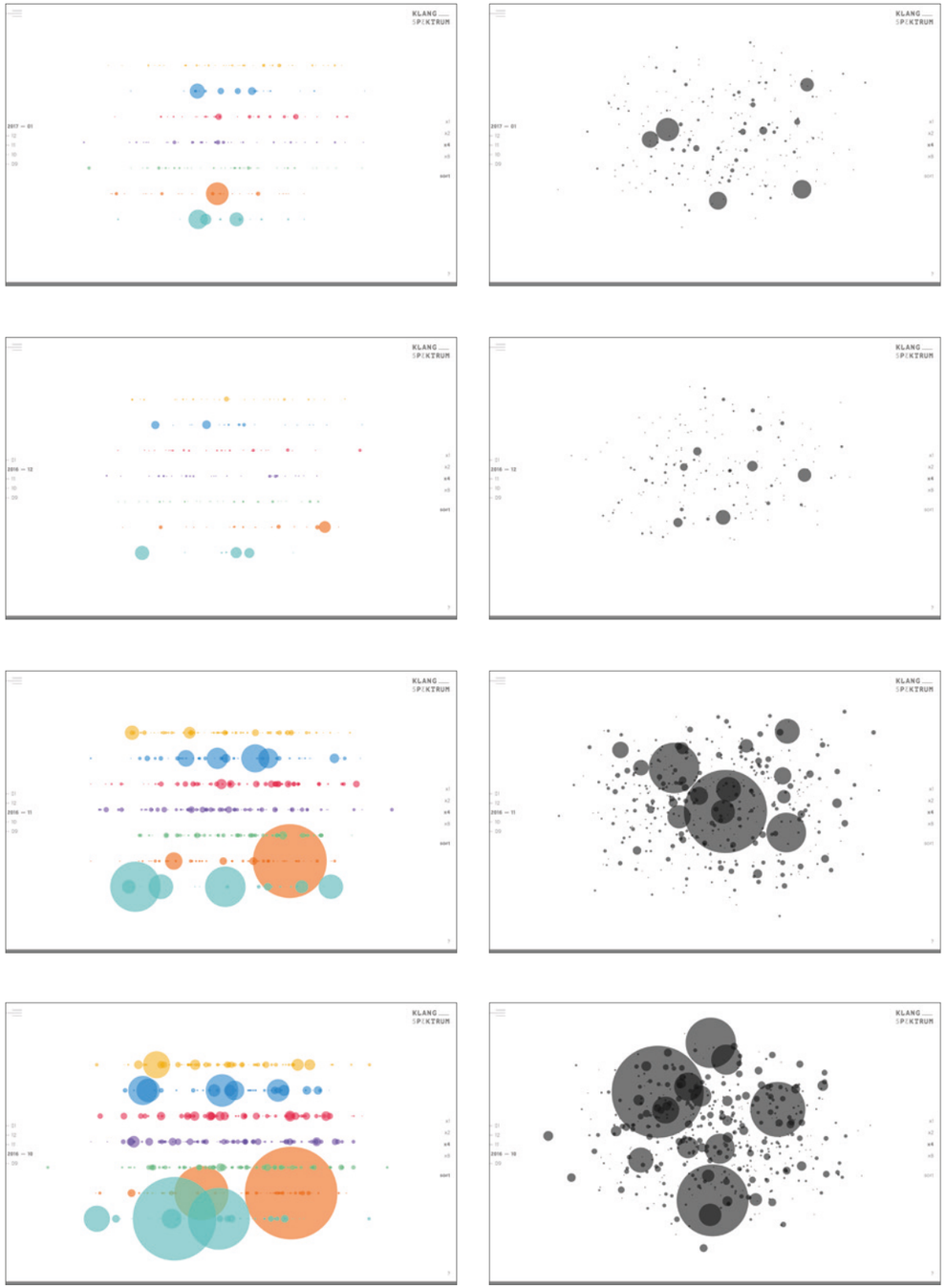


Abb. 16: Profil Vergleich

INTERAKTIONSMÖGLICHKEITEN

Damit der Betrachter weitere Informationen über einen ausgewählten Kreis erhält, ist es möglich über einen Kreis zu hovern bzw. diesen mit der Maus zu fangen. Hierbei hält der Kreis in seiner Bewegung inne, färbt sich in der jeweiligen Kategoriefarbe ein und dessen Wert erscheint mit zugehöriger Kategorie im oberen Bildschirmbereich *(vgl. Abb. 17, Aktiver Kreis)*.

Im nächsten Schritt kann der Nutzer nun die Kreiseigenschaft mit anderen Kreisen durch erneutes Hovern vergleichen oder den Kreis durch Klicken aktivieren und nur relevante Kreise in Bezug zum ausgewählten Kreis setzen. Hierbei färben sich alle zugehörigen Kreise in ihrer Kategoriefarbe, sie treten also in den Vordergrund und die restlichen Kreise treten passiv und hellgrau in den Hintergrund.

Ebenfalls sortieren sich die Kreise nach deren Kategorien und bewegen sich auf einer gemeinsamen X-Achse pro Kategorie. Der Nutzer bekommt somit einen Überblick über zugehörige Werte und Kategorien und kann auf schnelle und einfache Weise die Eigenschaften miteinander vergleichen. Zusätzlich zeigt sich durch die Anzahl der Kreise pro Kategorie wie spezialisiert das eigene Musikverhalten ist und ob sich darin Schwerpunkte ergeben *(vgl. Abb. 18, Sortierte Kreise)*.

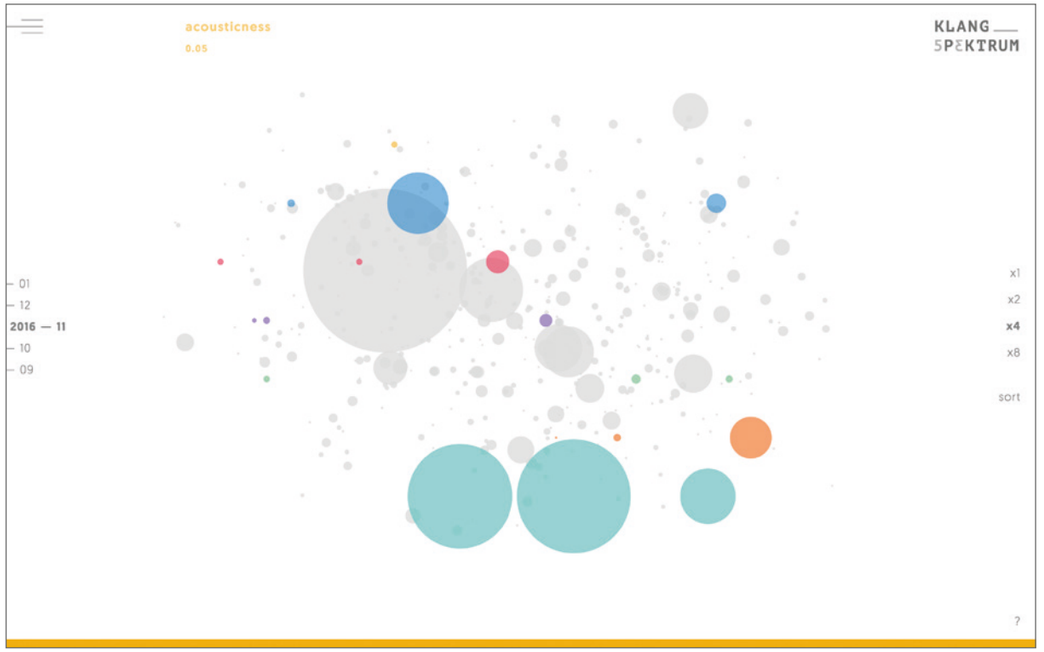


Abb. 17: Aktiver Kreis

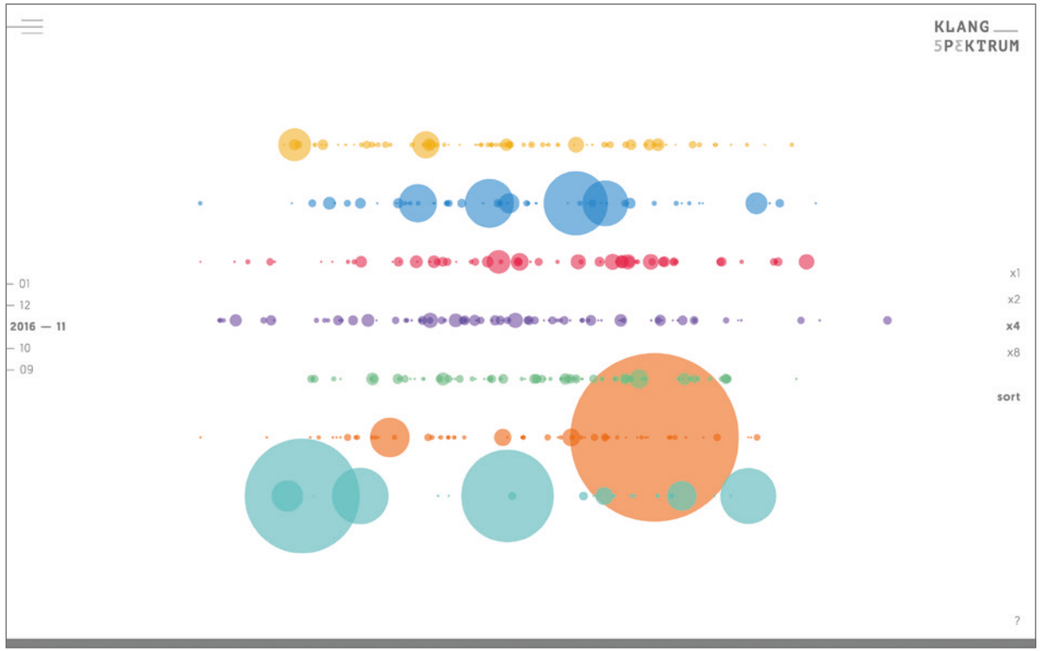


Abb. 18: Sortierte Kreise

Durch das Öffnen der Detailansicht über die Farbleiste am unteren Bildschirmrand werden die Songs und Genres bezüglich des aktiven Kreises aufgelistet. Der Nutzer kann über eine Playlist die Songs abspielen, wodurch er ein besseres Verständnis zu der Songeigenschaft entwickelt, was durch die Genreliste zusätzlich verstärkt wird. Ebenfalls weckt das Hören und Entdecken von Musik die Lust am weiteren Erforschen neuer Eigenschaften und Verhaltensmuster *(vgl. Abb. 20, Bezüge Klangspektrum)*.

GUTE VERSTÄNDLICHKEIT

Eine leicht verständliche Legende soll die korrekte Interpretation durch jeden Betrachter ermöglichen. Hieraus entsteht die Notwendigkeit, dass man zu jeder Zeit bei Fragen auf die Legende zugreifen kann. Diesbezüglich soll im Menü der Reiter *Legende* erscheinen und ein Fragezeichen in der Hauptansicht für Aufklärung sorgen. Die Legende beinhaltet Erklärungen zur Visualisierung, Gestaltung, den Kategorien und einen Vermerk zur Interaktion mit den Kreisen *(vgl. Abb. 19, Legende Klangspektrum)*.

Abb. 19: Legende *Klangspektrum*

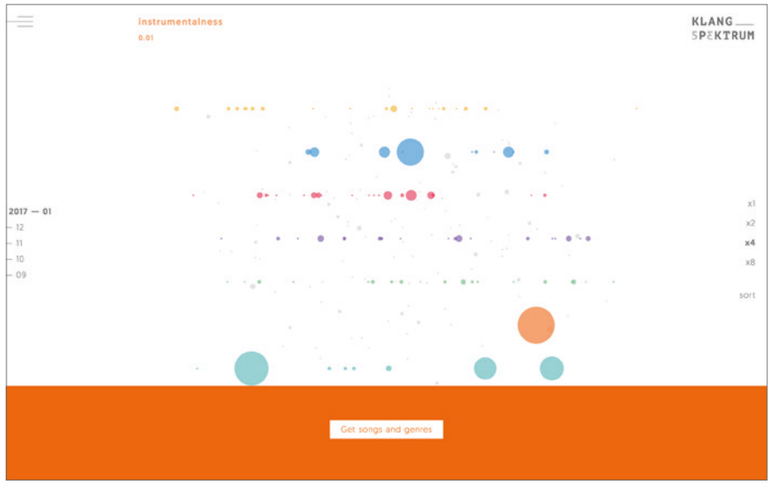


Abb. 20: Bezüge *Klangspektrum*

5.1.2 Webdesign und Logo

Da sich die App *Klangspektrum* mit Daten beschäftigt und diese visualisiert, ist das Webdesign an diesen Kontext angelehnt. Hierbei wurde ein schlichtes lineares Layout entworfen, das der Visualisierung viel Raum lässt und durch seine Reduziertheit im Kontrast zum organischen, musikalischen Charakter der Musikvisualisierung steht.

Die Grauwerte und die Wahl der Schriftfamilie *Geomanist*, sowie *PT Mono* sollen den Charakter von Daten und Maschinensprache unterstreichen. Als weiteres Stilelement wurde der Strich zur klaren Unterteilung wie beispielsweise bei den Menüpunkten eingeführt. Dieser lehnt sich an das Wertespektrum der Songeigenschaften und die Linearität von Daten an.

Das Logo trägt den Namen der Anwendung: *Klangspektrum*. Der Name ist sehr passend, da die Visualisierung Musik und Daten zu Songeigenschaften im Wertebereich zwischen 0 und 1 in Bezug setzt. Für das Logo wurde der Font *PT Mono* verwendet, welcher die Assoziation zur Computersprache heraushebt.

Die Buchstaben S und E im Wort *Spektrum* wurden durch Zahlen ersetzt und farblich mit der Linie des Logos in Bezug gesetzt. Hier zeigt sich die Überschneidung von uns gewohnten Dingen, die uns tagtäglich umgeben (die lesbare Schrift) und die Implementierung des Digitalen in eben dieses alltägliche Leben. Dieser schleichende Implementierungsprozess fällt zunächst beim Lesen des Logos kaum auf, auf den zweiten Blick jedoch schon. Das Logo führt an die digitale Welt und deren Sprache (Programmiersprache, Daten, Buchstabenkombinationen) heran, ebenso, wie die Visualisierung an die digitalen Hintergründe des modernen Musikhörens und Musikverhaltens heranführt. Das Logo wirkt daher modern und passt in die heutige Zeit.

GEOMANIST

KLANGSPEKTRUM
Klangspektrum

KLANGSPEKTRUM
Klangspektrum
Klangspektrum

PT MONO

KLANGSPEKTRUM
Klangspektrum

KLANGSPEKTRUM
Klangspektrum

Corporate Design *Klangspektrum*

5.2 Technik

5.2.1 JavaScript Bibliothek

P5.js ist eine *JavaScript*-Bibliothek, die mit dem ursprünglichen Ziel von *Processing* beginnt, um das Programmieren für Künstler, Designer, Pädagogen und Einsteiger zugänglich zu machen und diese für das heutige Web neu zu interpretieren. Unter Verwendung der ursprünglichen Metapher eines Software-Skizzenbuchs, bietet *P5.js* sehr gute Zeichnenfunktionalitäten.

Diese Funktionen sind gerade für die Visualisierung von *Klangspektrum* sehr interessant und bilden das Fundament der animierten interaktiven Visualisierung. Addon-Bibliotheken machen es zusätzlich möglich, mit anderen *HTML5*-Objekten, einschließlich Text, Eingabe, Video, Webcam und Ton zu interagieren *[McCarthy, o. J.]*.

5.2.2. Sortieralogrithmus

Die Daten von *Spotify* müssen vor der Visualisierung zunächst nach Datum, Kategorie und Wert sortiert werden. Dabei ist wichtig, dass der Sortieralgorithmus O(n) < n^2 ist, da die Wartezeit bei mittleren Musikbibliotheken sonst zu lange dauern würde. *Herr Prof. Kowarschick* von der *Hochschule Augsburg* half mir bei der Optimierung meines Algorithmus sehr.

Im Folgenden sieht man eine vereinfachte Übersicht der Daten zu Beginn und nach dem Sortiervorgang *[vgl. Abb. 21, Raw data & Abb. 22, Sort data]*.


```
data.js
//SORT:

["2017-01-07":{
    "acousticness":{
        "0.60": [{
        "artist": "Grizzly Bear",
        "title": "Sleeping Ute (Nicolas Jaar Remix)"
        }],
        "0.65": [{
        "artist": "Her",
        "title": "Quite Like"
        }]
    },
    "danceability":{
        "0.85": [{
        "artist": "Grizzly Bear",
        "title": "Sleeping Ute (Nicolas Jaar Remix)"
        }],
        "0.65": [{
        "artist": "Her",
        "title": "Quite Like"
        }]
    },
    "energy":{
        "0.35": [{
            "artist": "Grizzly Bear",
            "title": "Sleeping Ute (Nicolas Jaar Remix)"
        }],
        "0.55": [{
            "artist": "Her",
            "title": "Quite Like"
        }]
    },
    "instrumentalness":{
        "0.70": [{
            "artist": "Grizzly Bear",
            "title": "Sleeping Ute (Nicolas Jaar Remix)"
        }],
        "0.05": [{
            "artist": "Her",
            "title": "Quite Like"
        }]
    },
    "liveness":{
        "0.15": [{
            "artist": "Grizzly Bear",
            "title": "Sleeping Ute (Nicolas Jaar Remix)"
        }],
        "0.10": [{
            "artist": "Her",
            "title": "Quite Like"
        }]
    },
    "speechiness":{
        "0.05": [{
            "artist": "Grizzly Bear",
            "title": "Sleeping Ute (Nicolas Jaar Remix)"
        }],
        "0.50": [{
            "artist": "Her",
            "title": "Quite Like"
        }]
    },
    "valence":[{
        "0.15": {
            "artist": "Grizzly Bear",
            "title": "Sleeping Ute (Nicolas Jaar Remix)"
        }],
        "0.25": [{
            "artist": "Her",
            "title": "Quite Like"
        }]
    }
}]
```

Abb. 21: Sort data

```
data.js
//RAW:

[
    {
        "added_at": "2017-01-07T20:55:58Z",
        "artist": "Grizzly Bear",
        "title": "Sleeping Ute (Nicolas Jaar Remix)",
        "features": {
            "acousticness": 0.599,
            "danceability": 0.824,
            "energy": 0.322,
            "instrumentalness": 0.69,
            "liveness": 0.111,
            "speechiness": 0.0445,
            "valence": 0.125
    },
    {
        "added_at": "2017-01-07T20:50:34Z",
        "artist": "Her",
        "title": "Quite Like",
        "features": {
            "acousticness": 0.642,
            "danceability": 0.625,
            "energy": 0.502,
            "instrumentalness": 0.0288,
            "liveness": 0.101,
            "speechiness": 0.469,
            "valence": 0.236
        }
    }
]
```

Abb. 22: Raw data

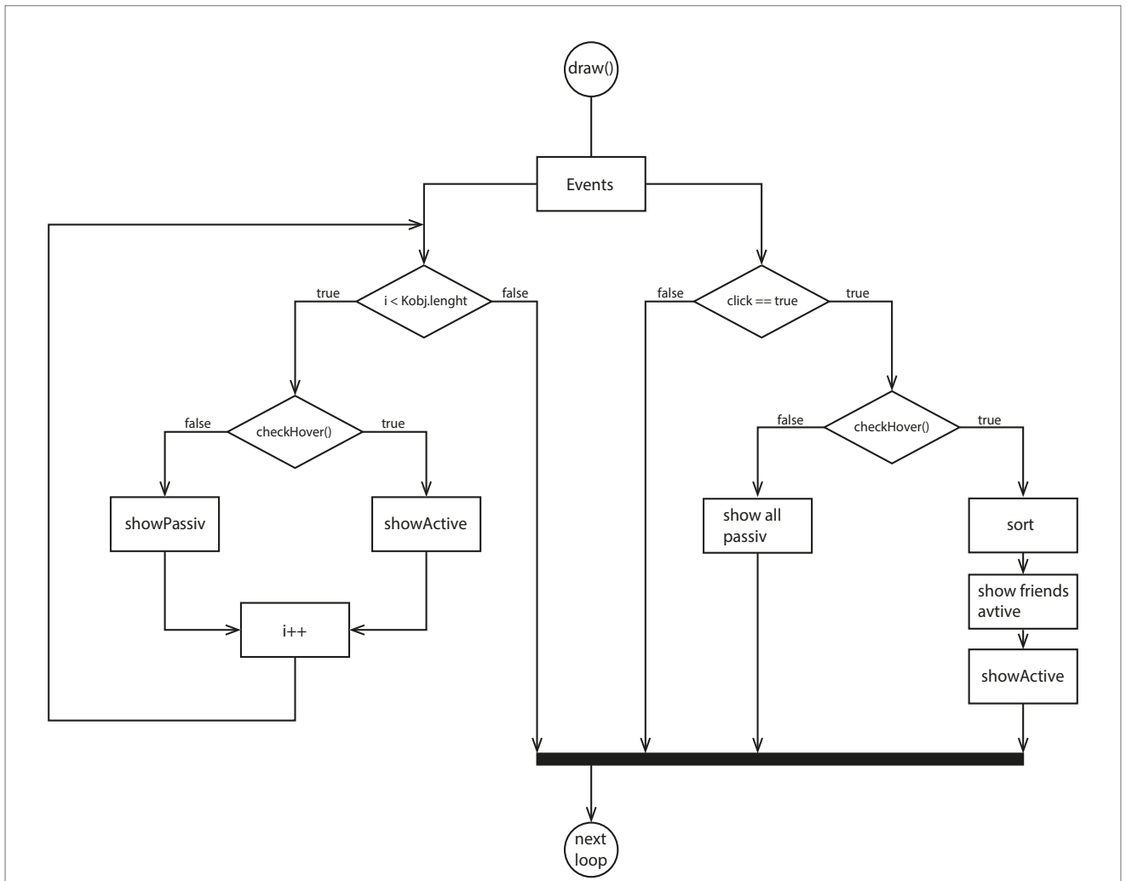


Abb. 23: Flussdiagramm

5.2.3 Softwarearchitektur

KLASSENDIAGRAMM

Auf der nachfolgende Seite ist ein stark vereinfachtes Klassendiagramm zu sehen. Das Programm wurde in einer Model-View-Controller Architektur erstellt *(vgl. Abb. 24, Klassendiagramm)*.

Model

Jedes Klangobjekt (Songeigenschaft) ist in der Visualisierung als Kreis dargestellt. Die Objekte besitzten alle notwendigen Attribute wie zum Beispiel die Position, States und drei Funktionen für die unterschiedlichen Bewegungsverhalten.

View

In der View werden alle wichtigen Updates für die DOM-Elemente, wie beispielsweise die Anzeige, durchgeführt. Außerdem werden alle Kreise nach deren aktuellen Eigenschaften gezeichnet.

Controller

Im Controller werden pro Frame alle Positionen der Kreise mit der Mausposition verglichen. Befindet sich die Maus dabei über einen Kreis, so wird dessen Status auf *active* gestellt. Zusätzlich werden durch das Klickevent weitere Zustandsveränderungen der Kreise initiiert.

FLUSSDIAGRAMM

Die Abbildung *(vgl. Abb. 23, Flussdiagramm)* zeigt ein stark vereinfachtes Flussdiagramm, welches die Logik der Zustandsveränderung der Kreise verdeutlicht. Pro Frame werden alle Kreise mit der Mausposition verglichen und ein möglicher aktiver Kreis ermittelt. Wird ein Klick ausgeführt, so muss ein aktiver Kreis existieren, damit eine Detailansicht sichtbar wird. Ist dies nicht der Fall, so werden die Kreise im passiven Zustand dargestellt.

Abb. 24: Klassendiagramm

Nach dem Laden und Sortieren der Daten werden diese in einem Array abgespeichert und stehen im Main-Sketch zur Verfügung.

Im Setup werden alle nötigen Variablen, Listener und Objekte erzeugt.

CreateObject erzeugt alle Klangobjekte anhand der Daten.

CreateDomElements erzeugt alle DOM-Elemente wie Listen, Buttons, Paragraphen oder das Canvas und verknüpft diese mit deren nötigen Eventlistenern.

SortEvent sorgt dafür, dass alle Klangobjekte ihren aktiven Farbzustand einnehmen und sich in der jeweiligen Kategorie auf einer X-Achse gruppieren.

ScaleEvent ändert die Skalierungsvariable nach der im Interface aktiven Größe, was sich in den zukünftigen Frames verdeutlicht.

PressedEvent ist *active*, wenn auf das Canvas geklickt wird. Hierbei wird die Visualisierung je nach Zustand aktualisiert.

TimeEvent prüft das aktuelle Zeitintervall und updated die Daten bei Änderungen.

In der Draw werden pro Frame alle Zustände geprüft und aktualisiert.

NodeHover überprüft alle aktuellen Kreispositionen und vergleicht diese mit der Mausposition. Befindet sich die Maus über einem Kreis, verändert sich der Zustand der Kreises.

UpdateDom aktualisiert die Anzeige des Interfaces.

DisplayNodes zeichnet alle Kreise bis auf die aktiven Kreise.

DisplayActiveNodes zeichnet alle aktiven Kreise zum Schluss, damit diese nicht von passiven Kreisen verdeckt werden.

5.2.4 Natürliche Bewegung

In dem Buch *The Nature of Code* von *Daniel Shiffman* wird ausführlich anhand konkreter Beispiele beschrieben, wie man natürliche Bewegungen erzeugen kann. *Perlin noise* ist ein zufälliger Sequenzgenerator, der eine natürlich geordnete harmonische Folge von Zahlen erzeugt; nicht zu verwechseln mit der random() Funktion, welche unabhängige zufällige Zahlen erstellt. Der Algorithmus wurde von *Ken Perlin* in den 1980er Jahren erfunden und wird seitdem in grafischen Anwendungen verwendet, um prozentuale Texturen, natürliche Bewegung, Formen oder Terrains zu produzieren, um einige Beispiele zu nennen *(Shiffman, 2012).*

In *P5.js* dient die Funktion noise() zur Rauschwerterzeugung nach *Perlin.* So bekommt jede Koordinate eines Kreises einen *Perlinwert* zugeschrieben, woraus eine natürliche und zufällige Bewegung entsteht.

6. NUTZERTEST

Ebenfalls war ein Teil der Arbeit, eine Testreihe mit der App durchzuführen, um Schwachstellen zu erkennen und Verbesserungen angehen zu können. Auch sollten in diesem Schritt Erkenntnisse gewonnen werden, ob die Zielsetzung erreicht wird.

6.1 Durchführung

Es gab eine Testreihe mit insgesamt sechs Personen, welche separat die Webanwendung testen sollten. Begonnen wurde mit der Aufnahme allgemeiner Personendaten, sowie der Erfahrung mit Datenvisualisierungen und Infografiken. Anschließend wurden die Testpersonen kurz gebrieft, um den Einstieg in das Thema der App zu vereinfachen. Daraufhin konnten sich die Personen einen ersten Überblick verschaffen und frei ohne konkrete Aufgabenstellung die App starten und benutzen. Daraufhin sollten die Testpersonen nun die Regeln der Visualisierung verstehen und versuchen, gezielt Informationen aus der Visualisierung herauszuziehen. Zum Schluss wurden allgemeine Verständnisfragen gestellt, um herauszufinden, ob die Person die abstrakten Werte der Musikeigenschaften verstanden hat und diese in den bekannten Bezug zu den Kriterien Genre und Songs setzen konnte.

6.2 Ergebnisse

Umfeld: Bekanntenkreis
6 Testpersonen (3 männlich und 3 weiblich)
Durchschnittsalter: 24
Vorwissen: sehr wenig bis keines
Nervosität: sehr gering

Technische Probleme: Nein, Logout/Login benötigt Unterstützung

Fragen	Antworten
Was bedeuten die Farben?	Kategorie /Kategorie / Kategorie / Kategorie / Kategorie / Kategorie
Was bedeuten die Kreise?	Kategoriewerte / Kategoriewerte / Kategoriewerte / Kategoriewerte / Kategoriewerte / Kategoriewerte
Was bedeutet der Ton?	Wert / Wert / Wert / Wert / Wert / Wert / Wert
Was bedeutet der „Beat“ der Kreise?	Wert / Wert / Wert / Wert / Wert / Wert / Wert
Was bedeutet ein hoher Zahlenwert?	hohe Intensität / hohe Intensität / hohe Intensität / hohe Intensität / hohe Intensität / hohe Intensität
Was zeigt die Visualisierung, wenn alle Kreise grau sind?	Grobe Ansicht / Aktivität, Musikverhalten / Art der Musik, Häufigkeit, Profil pro Zeit / Musikverhalten, Hörverhalten, Menge / Playlisten und Songs, Aktivität / Analyse Musikverhalten, Ausprägung, Schwerpunkte
Was zeigt die Visualisierung, wenn ein Kreis ausgewählt ist?	Zugehörigkeit / Mehrere Kreise aus unterschiedlichen Kategorien, haben evtl. etwas miteinander zu tun / Zugehörigkeit / Zugehörigkeit / Muster, Zugehörigkeit / Beziehungen der Songs
Kannst du dein Musikverhalten unter der Betrachtung einer Kategorie in den letzten drei Monaten erklären?	Nimmt ab, leichte Schwankungen, mehr Varianz / sehr konstant / wenig Akustik, wenig Songs / variieren stark, Songanzahl steigt / Bezüge zu Kategoriewerten / Beziehung Songs
Wie definierst du die untersuchte Eigenschaft?	Akustik meiner Musik / Hoher Wert = viel Akustik / wenig Akustik ist elektronisch, viel Akustik was mit Gitarre / wie tanzbar die Musik ist / Akustik meiner Musik / starke Melodie
Haben die Songs und Genres geholfen, um ein besseres Verständnis für die Werte und Eigenschaften zu entwickeln?	Ja / besseres Verständnis / gut, aber auch etwas verwirrend / Ja / Ja / Ja, wird verständlicher
Was wäre deine erste Änderung an der Visualisierung?	Playlist besser sichtbar machen / mehr auf die Werte eingehen / Playlist / Playlist, Begriffe verständlicher oder in Legende besser Erklären / weniger Werte, Playlist / Playlist, Timelinehilfe / Playlist, Legende zu Beginn
Downloade und öffne die Thesis. (Zusatzfrage)	Störend im Menü / Störend im Menü / Probleme zurück zur App zu kommen / Ein Reiter „About“ mit möglichem Downloadlink wäre besser

Ergebnisse Nutzertest

6.3 Schlussfolgerung

Die Testpersonen konnten ohne die Legende nur wenig mit der Datenvisualisierung anfangen. Wichtig ist es deshalb gleich zu Beginn die Legende automatisch zu öffnen, damit der Nutzer die „Regeln“ lernt, um später die Visualisierung richtig interpretieren zu können. Nach dem Lesen der Legende konnten alle Testpersonen die Visualisierung korrekt lesen.

Während der Interaktion mit den Kreisen zeigte sich, dass die Playlisten mit den Songs und Genres nicht aufgerufen wurden. Keine Testperson hat erkannt, dass sich unter dem Balken am unteren Bildschirmrand eine Referenz zu den zugehörigen Liedern und Genres befindet. Nach kurzer Einweisung begannen die Nutzer jedoch Kreise/Werte und Songs/Genres in einen Bezug zu setzen, was für Spaß und interessante Erkenntnisse sorgte.

Das eigene Musikverhalten wurde ebenfalls intuitiv mit anderen Monaten verglichen. Hierbei wurde sofort klar, dass wenig kleine Kreise für wenig Aktivität stehen. Problematisch wurde es, wenn der Name der Kategorie nicht leicht zu verstehen war, wie beispielsweise „valence“ (jetzt „mood“). Die Kategorien sollten deshalb durch deren Namen und einer kurzen Erklärung in der Legende gut verständlich sein, um Verwirrungen oder falsche Interpretationen zu vermeiden.

Während des Tests stellte sich heraus, dass die Testpersonen sich teilweise weniger Werte wünschten. Die Rückschlüsse könnten auch mit ca. 50 Werten pro Kategorie gezogen werden. Es fiel weiter auf, dass die Thesis besser unter einem Menüpunkt „About“ zum Download bereit stehen sollte, da der direkte Download verwirrt.

7. SCHLUSSBETRACHTUNG

7.1 Fazit

Klangspektrum ist eine unkonventionelle Datenvisualisierung, die ihre Nutzer zur Interaktion einlädt, um Informationen und das eigene Musikverhalten zu erforschen. Das Prinzip von Verhaltensmetaphern wurde im Design umgesetzt, damit der Nutzer ein stärkeres Erlebnis und einen besseren Bezug zum Thema Musik erfährt. Abstrakte Musikeigenschaften können mit konventionellen Kriterien wie Songs und Genres in einen Zusammenhang gesetzt werden, um ein Verständnis zu den Hintergrundprozessen von *Spotify* zu entwickeln. Die Benutzer haben Spaß an der Bedienung des Interfaces und entdecken ihre Lieblingssongs unter einer ungeahnten Perspektive ganz neu. Oft wird nach der Betrachtung der Visualisierung über Themen gesprochen, beispielsweise wie stark technische Prozesse in das Leben der Menschen eingreifen und wie es wohl in Zukunft aussehen könnte.

Dennoch gibt es noch mögliche Verbesserungen, damit der User mehr Nutzen aus der Anwendung ziehen kann:

Filter zum Ändern des Zeitintervalls
Einige Nutzer haben nicht genügend Aktivität in ihrem Account aufzuweisen, damit eine monatliche Aufteilung Sinn macht. Um dem entgegenzuwirken sollte über einen Filter ein Zeitintervall eingestellt werden können, bis der Nutzer genügend Informationen auf dem Bildschirm sieht.

Filter für das Wertespektrum der Kategorien
Aktuell bilden 100 Wert zwischen 0 und 1 das komplette Spektrum einer Kategorie ab, welche den Song definieren. Ein Filter könnte hierbei zeitweise helfen, einen besseren Überblick und stärkere Unterschiede zwischen den Werten festzustellen, was zusätzlich das Verständnis für einzelne Werte verbessern würde.

Auswahl der zu visualisierenden Kategorien
Für eine gute Übersicht müssen alle Kategorien dargestellt werden. Bei einer Detailanalyse wäre es jedoch von Vorteil, wenn man Kategorien ausschließen könnte, um die Visualisierung zu spezialisieren.

Ein zusätzliches Gesamtprofil
Außerdem wäre es von Vorteil, wenn der Nutzer über eine weitere Grafik sein gesamtes Profil seiner Aktivitäten einsehen könnte, um schnell die persönliche Entwicklung und Topwerte zu erkennen.

7.2 Ausblick

Klangspektrum läuft in allen gängigen Browsern stabil und hat gute Chancen in den *Developer-Showcase* von *Spotify* integriert zu werden, was den Bekanntheitsgrad enorm steigern würde. Dadurch könnte das Projekt andere Entwickler inspirieren und zusätzlich den Umgang mit der *Spotify* API exemplarisch aufzeigen. Des Weiteren könnten zusätzliche Verhaltensanalysen erstellt und visualisiert werden, da von allen Nutzern der Anwendung *Klangspektrum* die wichtigsten Daten in einer Datenbank abgelegt werden.

Die Anwendung hat zusätzlich Potenzial in ihrer Dienstleistung zu wachsen und könnte zu einer Mehrfachanwendung erweitert werden. Zum Beispiel könnten Nutzer deren gewonnene Erkenntnisse konkret anwenden und Playlisten anhand spezieller Songeigenschaften erstellen. Ebenfalls könnten durch die Schwerpunkte im Musikverhalten auf Spotifynutzer verwiesen werden, welche dem Nutzer in seinem Musikverhalten ähnlich sind, um das eigene Netzwerk zu erweitern.

8. QUELLEN- UND LITERATURVERZEICHNIS

8.1 Internetquellen

Fielding, Roy Thomas (2000): „Architectural Styles and the Design of Network-based Software Architectures“. Ics.uci.edu. Abgerufen am 24. 01. 2017 von https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Horn, Carolin (2007): „CAROLIN HORN | PORTFOLIO | ANYMAILS“. Carohorn.de.
Abgerufen am 24. 01. 2017 von http://carohorn.de/portfolio/anymails/anymails.html.

McCarthy, Lauren (o. J.): „p5.js | home“. P5js.org.
Abgerufen am 24. 01. 2017 von https://p5js.org/.

McCarthy, Lauren (o. J.): „p5.js | reference“. P5js.org.
Abgerufen am 14. 01. 2017 von https://p5js.org/reference/#/p5/noise.

Severino, Ribecca (o. J.): „Histogram“. Datavizcatalogue.com.
Abgerufen am 05. 01. 2017 von http://datavizcatalogue.com/methods/histogram.html.

Severino, Ribecca (o. J.): „Scatterplot“. Datavizcatalogue.com.
Abgerufen am 10. 01. 2017 von http://datavizcatalogue.com/methods/scatterplot.html.

Severino, Ribecca (o. J.): „Heatmap“. Datavizcatalogue.com.
Abgerufen am 24. 01. 2017 von http://datavizcatalogue.com/methods/heatmap.html.

Severino, Ribecca (o. J.): „Treemap“. Datavizcatalogue.com.
Abgerufen am 24. 01. 2017 von http://datavizcatalogue.com/methods/treemap.html.

Springer, SebastianGmbH, yeebase (2013): „Node.js: Das JavaScript-Framework im Überblick“. t3n Magazin.
Abgerufen am 14. 01. 2017 von http://t3n.de/magazin/serverseitige-javascript-entwicklung-nodejs-einsatz-231152/.

Wikipedia (o. J.): „Audiovisualisierung“. De.wikipedia.org.
Abgerufen am 24. 01. 2017 von https://de.wikipedia.org/wiki/Audiovisualisierung.

(2013): „cujojs/when“. GitHub.
Abgerufen am 13. 01. 2017 von https://github.com/cujojs/when.

(2013): „MongoDB – Die dokumentorientierte JSON-Datenbank | NodeCode“. NodeCode.
Abgerufen am 15. 01. 2017 von http://nodecode.de/mongodb.

8.2 Literaturquellen

Allen, R. E; Fowler, H. W; Fowler, F. G (1992): The Concise Oxford Dictionary. 1. Aufl. Oxford: Clarendon Press.

Gann, Kyle (2010): No such thing as silence. 1. Aufl. New Haven [Conn.]: Yale University Press.

Gorski, Peter Leo; Lo Iacono, Luigi; Nguyen, Hoai Viet (2015): WebSockets. 1. Aufl. München: Hanser.

Schwarz, Michael (2016): „Umfrage zum Musikverhalten, damals und heute“.

Seifert, Josef W. : Visualisieren, Präsentieren, Moderieren, 26. Aufl., Offenbach 2009, S. 11–46

Shiffman, Daniel; Fry, Shannon; Marsh, Zannah (2012): The nature of code. 1. Aufl. Selbstveröffentlichung.

9. ABBILDUNGSVERZEICHNIS

10. ERKLÄRUNG ZUR ABSCHLUSSARBEIT

Hiermit versichere ich, die eingereichte Abschlussarbeit selbstständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt zu haben. Wörtlich oder inhaltlich verwendete Quellen wurden entsprechend den anerkannten Regeln wissenschaftlichen Arbeitens zitiert. Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht anderweitig als Abschlussarbeit eingereicht wurde. Das Merkblatt zum Täuschungsverbot im Prüfungsverfahren der Hochschule Augsburg habe ich gelesen und zur Kenntnis genommen. Ich versichere, dass die von mir abgegebene Arbeit keinerlei Plagiate, Texte oder Bilder umfasst, die durch von mir beauftragte Dritte erstellt wurden.

Ort, Datum

Unterschrift des/der Studierenden

11. DANKE...

Herrn Prof. Dr. Kipp für Ihre Unterstützung bei der Konzeption, die richtigen Worte zur richtigen Zeit und viel Geduld.

Herrn Prof. Rothaug für Ihre inspirierenden Gespräche in der Konzeptfindung und die guten Ratschläge im Webdesign.

Herrn Prof. Dr. Kowarschick für Ihre Hilfe beim Sortieralgorithmus und Ihre Flexibilität. Ohne Sie würden die Nutzer Stunden auf ihre Visualisierung warten müssen.

Meli, dass ich auf dich immer zählen kann. Backenschnurrer.

Basti, du bist einfach mit der Kamera geboren.

Iris, **Caro**, **Jo** und **Andi** für eure Zeit und Hilfe beim wichtigen UX-Test.

Michael Jaser für deine Ratschläge bei Webentwicklungsfragen.