HTL - Perg
Höhere Abteilung für Informatik

Diplomarbeit

Face the Taste of Music

Webapplikation zur Visualisierung von Musikgeschmäckern

Projektteam:	Benedikt Alkin
	Simon Kroissmayr
Projektbetreuer:	Prof. Dipl.-Ing. Christian Aberger

In Zusammenarbeit mit der Johannes Kepler Universität Linz

Betreuer:	Herr Dr. Markus Schedl
Bearbeitungszeitraum:	01.10.2015 – 08.04.2016

1 Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Perg, 01.04.16 Unterschrift 

Simon Kroissmayr

Perg, 01.04.16 Unterschrift 

Benedikt Alkin

2 Danksagung

Wir möchten uns an dieser Stelle bei jenen Personen bedanken, die uns im Verlauf der Erstellung der Diplomarbeit in jeglicher Hinsicht unterstützt haben.

Primär gilt unser Dank Herrn Dr. Markus Schedl, der unsere Verbindungsperson zum Institut für Computational Perception an der Johannes Kepler Universität Linz ist. Er hat uns, besonders bei dem von uns am Institut absolvierten Praktikum, bei vielen Problemen weitergeholfen.

Darüber hinaus bedanken wir uns bei unserer Betreuungslehrkraft, Herrn Prof. Dipl.-Ing. Christian Aberger, der uns bei der Erstellung mit hilfreichen Tipps und Tricks zur Seite gestanden ist. Weiters möchten wir Herrn Prof. Ing. Werner Schöller erwähnen, der uns einen leistungsstarken Server zum Berechnen der Musikkarte zur Verfügung gestellt hat.

3 Impressum

Schule

- HTBLA für Informatik
- Machlandstraße 48
- 4320 Perg

Schuljahr

- 2015/2016

Klasse

- 5AHIF

Projekttitel

- Face the Taste of Music
- Webapplikation zur Visualisierung von Musikgeschmäckern

Projektteam

- Benedikt Alkin
- Simon Kroissmayr

Betreuungslehrkraft

- Prof. Dipl.-Ing. Christian Aberger

Auftraggeber

- Johannes Kepler Universität Linz

4 Inhaltsverzeichnis

1 Eidesstattliche Erklärung 1
2 Danksagung 2
3 Impressum 3
4 Inhaltsverzeichnis 4
5 Kurzbeschreibung 8
6 Abstract 9
7 Projektumfeld 10
7.1 Team 11
7.1.1 Benedikt Alkin 11
7.1.2 Simon Kroissmayr 11
7.2 Betreuungslehrkraft 12
7.2.1 Professor Dipl.-Ing. Christian Aberger 12
7.3 Ausbildungseinrichtung 12
7.3.1 HTL Perg, Höhere Abteilung für Informatik 12
7.4 Auftraggeber 13
7.4.1 Johannes Kepler Universität Linz 13
7.4.2 Herr Dr. Markus Schedl 14
7.4.3 last.fm 14
8 Einleitung 15
8.1 Motivation 15
8.2 Ziel der Arbeit 15
8.2.1 Technische Ziele 16
8.2.2 Allgemeine Ziele 16
9 Grundlagen 17
9.1 Verwendete Technologien 17
9.1.1 t-distributed Stochastic Neighbor Embedding (t-SNE) 17
9.1.1.1 The Crowding Problem 18
9.1.1.2 Alternativen zu t-SNE 18
9.1.1.2.1 Isomap 18
9.1.1.2.2 Sammon Mapping 18
9.1.1.2.3 Selbstorganisierende Karte 18
9.1.2 R 19
9.1.3 Java 19
9.1.4 Java Persistence API (JPA) 20

9.1.5 JavaServer Faces (JSF) 21
9.1.5.1 WildFly 21
9.1.6 Hypertext Markup Language (HTML) 21
9.1.7 Cascading Style Sheets (CSS) 22
9.1.8 JavaScript 22
9.1.9 Asynchronous JavaScript and XML (AJAX) 23
9.1.10 Structured Query Language (SQL) 23
9.1.11 Windows Server 2012 R2 23
9.2 Verwendete Entwicklungssysteme 24
9.2.1 NetBeans IDE 24
9.2.2 Eclipse IDE 25
9.2.3 Notepad++ 26
9.2.4 MySQL Workbench 27
9.2.5 RGui 28
9.2.6 Matlab 28
9.3 Verwendete Bibliotheken 29
9.3.1 D3.js 29
9.3.2 jQuery 29
9.4 Verwendete Architekturen 30
9.4.1 DAO (Data Access Object) 30
9.4.2 REST (Representational State Transfer) 30
9.5 Sonstige verwendete Software 31
9.5.1 Dropbox 31
9.5.2 Microsoft Office 2013 31
9.5.2.1 Microsoft Word 2013 31
9.5.2.2 Microsoft Excel 2013 31
9.5.2.3 Microsoft PowerPoint 2013 31
9.5.3 GNU Image Manipulation Program (GIMP) 32
10 Realisierung 33
10.1 Planung 33
10.1.1 Funktionale Idee 33
10.1.2 Use-Case-Diagramm 34
10.1.2.1 Gesamte Musikkarte anzeigen 35
10.1.2.2 Interpret Informationen anzeigen 35
10.1.2.3 Nach Benutzernamen suchen 35
10.1.2.4 Userspezifische Karte anzeigen 35

10.1.2.5 Musikkarte zoomen 35
10.1.3 Projektstrukturplan 36
10.1.4 Meilensteine 37
10.1.5 Datenmodell 38
10.1.5.1 Tabelle „User“ 39
10.1.5.2 Tabelle „Tag“ 39
10.1.5.3 Tabelle „Artist“ 39
10.1.5.4 Tabelle „Track“ 40
10.1.5.5 Tabelle „Artist_has_Tag“ 40
10.1.5.6 Tabelle „Listeningevent“ 40
10.2 Darstellung 41
10.2.1 Seitenaufbau 41
10.2.2 Seitendarstellung 42
10.3 Herausforderungen und Risiken 42
10.3.1 Datenmengen 42
10.3.1.1 Datenfilterung 43
10.3.1.2 Ressourcenauslastung 43
10.3.1.2.1 Lösungsansatz Xbox Cluster 43
10.3.1.2.2 Lösungsansatz Schulserver 44
10.3.2 Komplexität 45
10.3.2.1 Parameter perplexity 45
10.3.2.2 Parameter learning rate 48
10.3.2.3 Iterationen 48
10.4 Implementierung 51
10.4.1 Dimensionsreduktion 51
10.4.2 Vektorerstellung 52
10.4.2.1 Auslesen der Daten 52
10.4.2.2 Ausgabe des Vektors 53
10.4.3 t-SNE Algorithmus Versionen 54
10.4.3.1 JavaScript Version 54
10.4.3.2 Java Version 56
10.4.3.3 Matlab Version 58
10.4.3.4 R Version 60
10.4.3.4.1 Laden der benötigten Ressourcen 60
10.4.3.4.2 Erstellen einer Callback Methode zum Kontrollieren des Zwischenfortschrittes des Algorithmus 61

10.4.3.4.3 Ausführen des t-SNE Algorithmus 61
10.4.3.4.4 Exportieren des Ergebnisses in eine CSV Datei 62
10.4.4 Koordinatenaufbereitung 62
10.4.4.1 Ausgangsstruktur 62
10.4.4.2 Struktur nach der Aufbereitung 63
10.4.5 Erstellung des Java Servlets 64
10.4.5.1 Implementierung 64
10.4.5.1.1 Architektur 64
10.4.5.1.2 View 64
10.4.5.1.3 Allgemeine Karte 65
10.4.5.1.4 Benutzerspezifische Karte 66
10.4.5.1.5 Controller 67
10.4.5.1.6 Struktur eines Servlets 68
10.4.6 JavaScript Funktionen der Webapplikation 69
10.4.6.1 drawSVG 69
10.4.6.2 zoomMap 70
10.4.6.3 disableLoadingIndicator 70
10.4.6.4 drawDots 70
10.4.6.5 colorDotsInTagColors 71
10.4.6.6 calculateColor 72
10.4.6.6.1 HSL System 72
10.4.6.7 drawLegend 73
10.4.6.8 fillLatestSongs 74
10.4.6.9 highlightDots 74
11 Ergebnis 75
12 Resümee 79
13 Verteilung der Aufgabengebiete 80
13.1 Benedikt Alkin 80
13.2 Simon Kroissmayr 80
14 Glossar 81
15 Anhang 84
15.1 Literaturverzeichnis 84
15.2 Abbildungsverzeichnis 88
15.3 Tabellenverzeichnis 90

5 Kurzbeschreibung

Face the Taste of Music ist eine Diplomarbeit, welche im Jahr 2016 im Rahmen der Matura an der HTL Perg implementiert wurde.

Aufgabenstellung

Der Leitgedanke ist eine Karte aus musikalischen Beschreibungen zu erstellen, die Musikrichtungen darstellt. Aufgrund dieser Karte soll zusätzlich der Musikgeschmack eines einzelnen Benutzers visualisiert werden.

Realisierung

Den Ausgangspunkt der Diplomarbeit bilden die musikalischen Daten des Internetradios last.fm. Diese werden mithilfe des Clustering Algorithmus t-SNE, der besonders für hochdimensionale Daten geeignet ist, in eine zweidimensionale Karte transformiert. Diese Umwandlung passiert auf Basis der Zuordnungen und Gewichtungen eines Künstlers zu den verschiedenen Musikrichtungen. Die Koordinaten, die das Ergebnis dieser Berechnungen darstellen, werden weiter aufbereitet und in einer Datenbank zwischengespeichert. Anschließend werden die Daten über eine Webapplikation zugänglich gemacht und mit der JavaScript Bibliothek D3.js ansprechend visualisiert.

Ergebnis

Die Aufgabenstellung wurde mittels einer Webapplikation realisiert und in Kooperation mit der JKU Linz durchgeführt. Im folgenden Bild ist die Musikkarte in einem grafisch ansprechenden Design abgebildet.

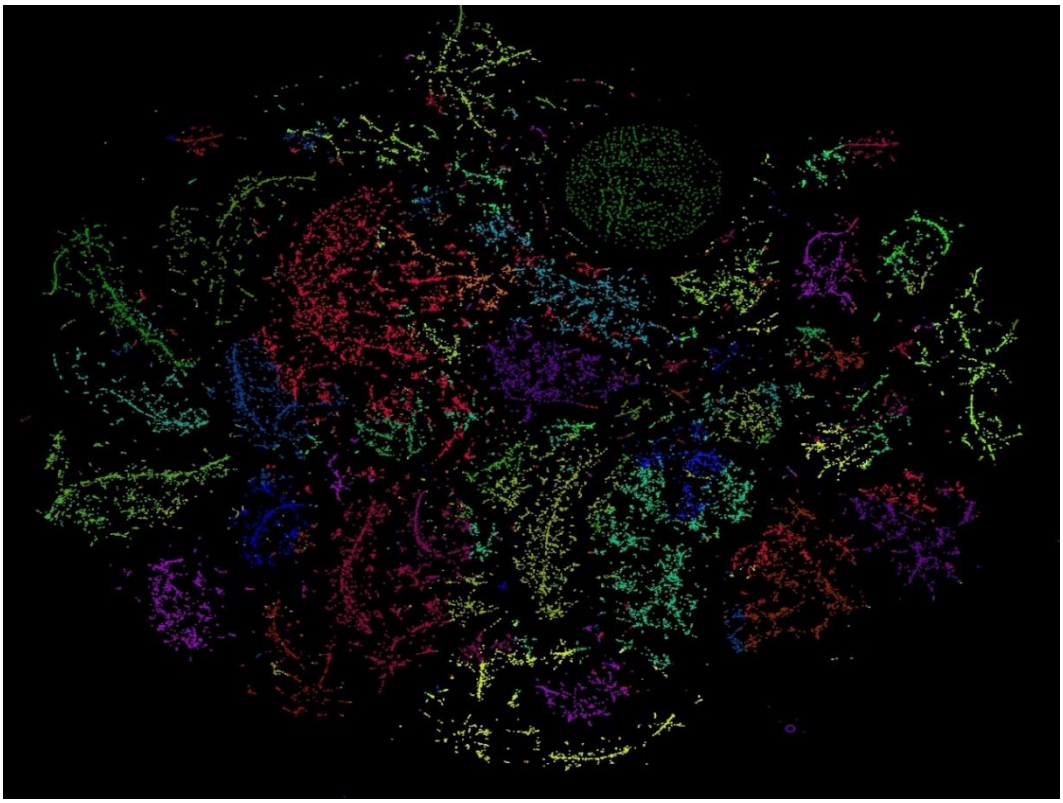

Abb. 1 Musikkarte

6 Abstract

Face the Taste of Music is a diploma project which has been implemented in 2016 as a part of the Higher School Certificate.

Definition of task
The central theme is the creation of a map, which consists of musical descriptions. The map shows all the different music genres. Based on that map, the music taste of one particular user is graphically visualized.

Implementation
The musical data of the online radio last.fm represent the basis of the diploma project. By using the t-SNE clustering algorithm, which is particularly appropriate for high-dimensional data, the data is transformed into a two-dimensional map. This transformation is based on the allocations and weightings between the artists and the different music genres. The result of these evaluations is further processed and saved in a database. Afterwards the data is accessible over a web application and visualised by the JavaScript library D3.js.

Result
The task was realized through a web application and executed in cooperation with the JKU Linz. In the following picture, the music map is pictured in a graphically appealing way.

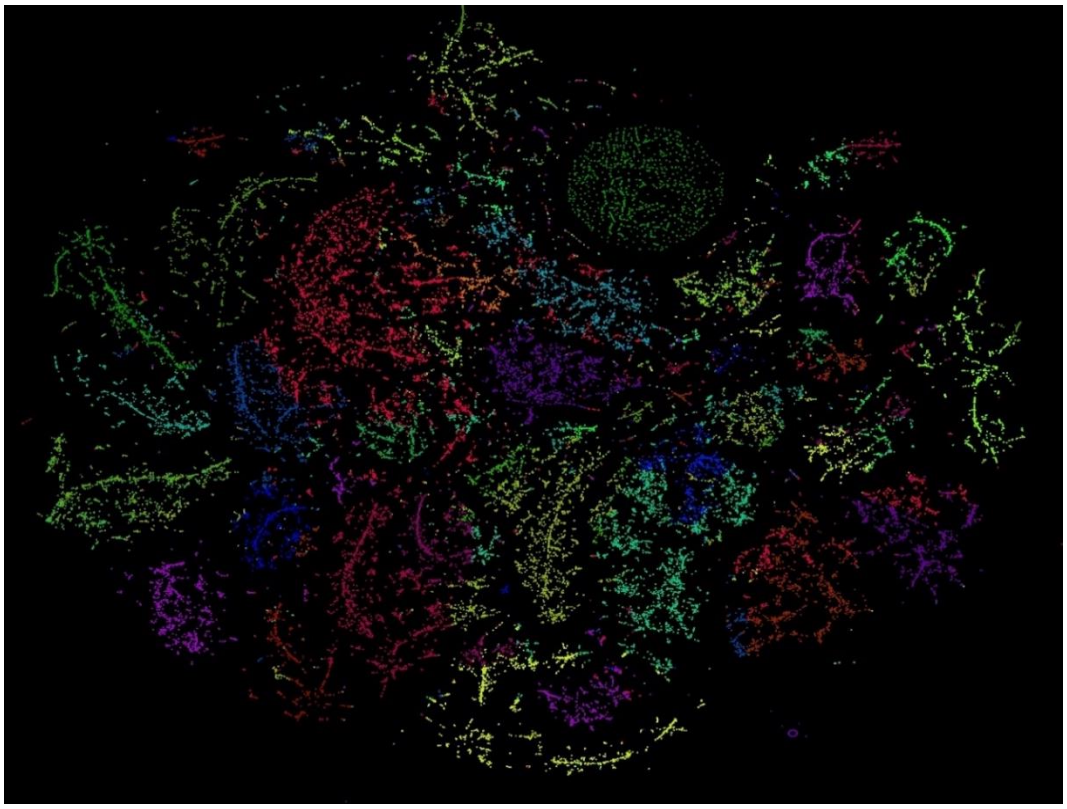

Abb. 2 music map

7 Projektumfeld

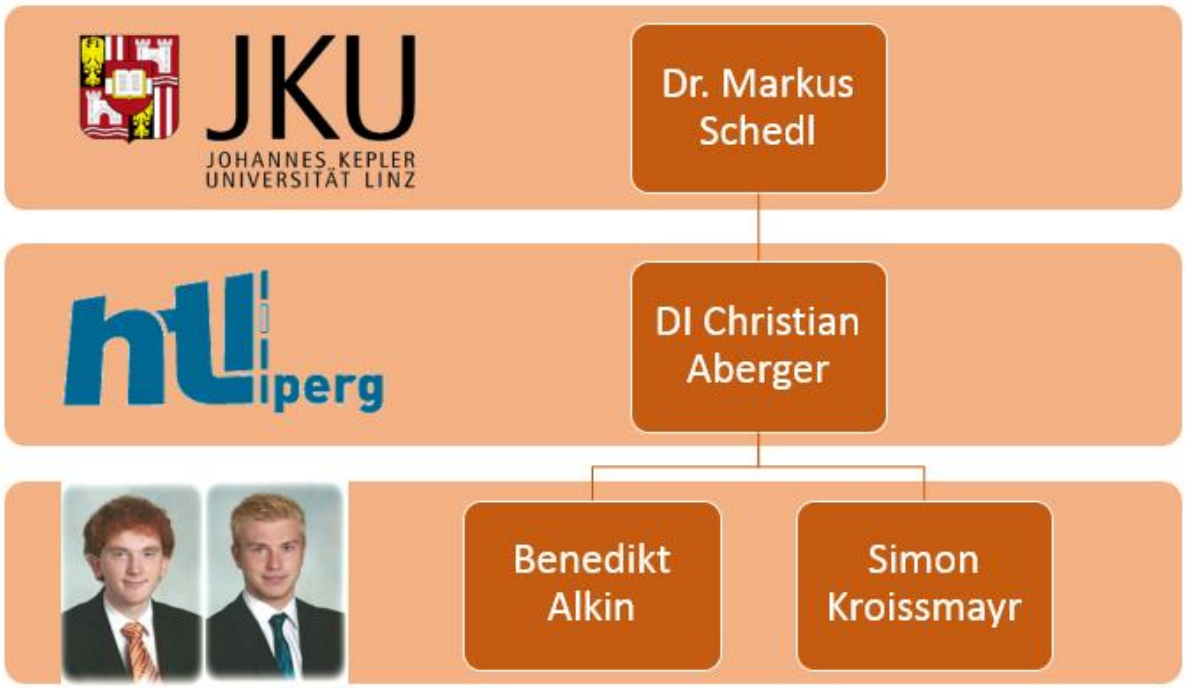

Abb. 3 Projektumfeld

Im November 2014 ist die Johannes Kepler Universität Linz an die HTL Perg herangetreten und hat der Schule das Angebot gemacht, zwei Schülerteams zu ernennen, die ihre Diplomarbeiten in Kooperation mit der Universität abwickeln.
Beiliegend sind die möglichen Themengebiete gewesen welche die JKU angeboten hat. Wir haben uns für das Themengebiet „Künstliche Intelligenz“ entschieden.

Nach Absprache mit unserer Ansprechperson an der JKU, Herrn Dr. Markus Schedl, haben wir uns auf ein konkretes Thema, die Visualisierung von Musik, geeinigt. Anhand dieses Themas ist nun die Diplomarbeit Face the Taste of Music entstanden.

Die beiden Diplomanden, die das Projekt durchgeführt und diese Arbeit verfasst haben, sind Benedikt Alkin und Simon Kroissmayr.

7.1 Team

7.1.1 Benedikt Alkin

Abb. 4 Benedikt Alkin

Persönliche Daten

Name:	Benedikt Alkin
Geburtsdatum:	11. April 1997
Wohnadresse:	Erlabachweg 3, 4303 St. Pantaleon

Referenzen

Ausbildung:	2003 – 2007	Volksschule St. Pantaleon
	2007 – 2011	Sportmittelschule St. Valentin
	2011 – 2016	HTL für Informationstechnologie in Perg
Praktika & Erfahrungen:	2013	Praktikum bei der Ärztekammer für OÖ
	2014	Praktikum bei der systema Steyr

Fähigkeiten

Sprachen	Deutsch (Muttersprache) Englisch (fließend in Sprache und Schrift)
EDV – Technologien	C#; Java; iOS; Android; HTML; PHP; JavaScript; MySQL; Oracle; SAP; MS Office

7.1.2 Simon Kroissmayr

Abb. 5 Simon Kroissmayr

Persönliche Daten

Name:	Simon Kroissmayr
Geburtsdatum:	14. Jänner 1997
Wohnadresse:	Werkstraße 52, 4300 St. Valentin

Referenzen

Ausbildung:	2003 – 2007	Volksschule St. Valentin
	2007 – 2011	Sportmittelschule St. Valentin
	2011 – 2016	HTL für Informationstechnologie in Perg
Praktika & Erfahrungen:	2013	Praktikum bei der Pensionsversicherungsanstalt Linz
	2014	Praktikum bei der Arbeiterkammer OÖ

Fähigkeiten

Sprachen	Deutsch (Muttersprache) Englisch (fließend in Sprache und Schrift)
EDV – Technologien	C#; Java; HTML; PHP; JavaScript; MySQL; Oracle; SAP; MS Office

7.2 Betreuungslehrkraft

7.2.1 Professor Dipl.-Ing. Christian Aberger

Kontakt

Aberger Software GmbH
Softwarepark 37
4232 Hagenberg
☎ +43 720 348 450
office@aberger.at
www.aberger.at

Abb. 6 Betreuungslehrkraft DI Christian Aberger [PERG, HTL, 2015]

Betreut wird unsere Diplomarbeit von Herrn Professor Dipl.-Ing. Christian Aberger, der seit der vierten Klasse unter anderem den Programmierunterricht im Bereich Java leitet. Da diese Diplomarbeit in die Richtung der Webprogrammierung geht, fällt sie in den Spezialbereich unseres Herrn Professors.

Neben seiner Arbeit als Professor an der HTL Perg, ist er auch Geschäftsführer der Aberger Software GmbH, welche im Softwarepark Hagenberg ansässig ist.

7.3 Ausbildungseinrichtung

7.3.1 HTL Perg, Höhere Abteilung für Informatik

Kontakt

HTBLA Perg
Machlandstraße 48
4320 Perg
☎ 0 72 62 539 26
office@htl-perg.ac.at
www.htl-perg.ac.at

Abb. 7 Ausbildungseinrichtung HTL Perg [FCP, 2015]

Der Ausbildungsschwerpunkt der Höheren Abteilung für Informatik liegt im Bereich der Softwareentwicklung. Ebenfalls besondere Beachtung wird den betriebswirtschaftlichen Perspektiven geschenkt.

Eine Besonderheit der Ausbildung ist der hohe Praxisbezug und die intensive Kooperation mit externen Firmen und anderen Projektpartnern. Diese Projekte werden während der Ausbildungszeit im Rahmen des Gegenstandes Projektentwicklung, oder als Abschluss im Zuge der Diplomarbeit, realisiert.

7.4 Auftraggeber

7.4.1 Johannes Kepler Universität Linz

Kontakt

Johannes Kepler Universität Linz
Alternberger Straße 69
4040 Linz
☎ +43 732 2468 0
info@jku.at
www.jku.at

Abb. 8 Logo der JKU Linz [UNI, 2015]

Die 1966 eröffnete Johannes Kepler Universität Linz gilt als größte wissenschaftliche Institution Oberösterreichs und ist in kürzester Zeit zu einem Impulszentrum für Wissenschaft, Wirtschaft und Gesellschaft geworden. Das Angebot von 60 Studienrichtungen nehmen über 19 000 Studierende in Anspruch. (vgl. [JKU, 2015])

Das Institut *Computational Perception* ist der Auftraggeber dieser Diplomarbeit.
Dieses Institut ist seit Oktober 2004 an der Johannes Kepler Universität ansässig. Dessen Aufgabe ist es, rechnerische Modelle und Algorithmen zu untersuchen und zu entwickeln, die es Computern erlauben die „äußere“ Welt wahrzunehmen und zu begreifen. Der Begriff Perception bezeichnet dabei den Vorgang der Extraktion von komplexen Informationen und Wissen aus simplen Daten. Unter simplen Daten werden Videos, Töne, Bilder, Texte, Datenbanken und das Internet bezeichnet.

Die Forschung spezialisiert sich auf Probleme wie pattern recognition, knowledge extraction, und data mining. Einige Methoden, die verwendet werden, sind classification und generell Computational Intelligence.
Der momentane Schwerpunkt des Institutes liegt auf dem Verarbeiten von Tönen und Musik und dem darausfolgenden Gewinnen von Informationen. (vgl. [JKU, 2015])

Abb. 9 Science Park JKU Linz [LINZ, JKU, 2015]

7.4.2 Herr Dr. Markus Schedl

Kontakt

JKU Linz
Altenberger Straße 69
4040 Linz
Science Park 3, Raum 442
markus.schedl@jku.at

Abb. 10 Ansprechpartner Dr. Markus Schedl [JKU, 2015]

Unser Ansprechpartner an der JKU, am Institut für *Computational Perception*, ist Herr Dr. Markus Schedl.
Er ist ein außerordentlicher Professor an der Johannes Kepler Universität und Autor vieler Tagungsberichte und namhafter Zeitungsartikel.
Sein Hauptaugenmerk liegt auf den Bereichen social media mining, information retrieval und music information research. (vgl. [SCHEDL, Dr. Markus, 2016])

7.4.3 last.fm

last.fm ist eine soziale Software zum Entdecken von Musik, die Nutzern, aufgrund deren Musikgeschmackes, Empfehlungen für neue Musik, Menschen mit ähnlichem Musikgeschmack und dazu passende Konzerte in der Nähe, liefert. Diese soziale Software ist die Quelle für alle in unserer Applikation verwendeten Musikdaten. (vgl. [SCHWIETERING, Heinrich, 2014])

Abb. 11 Logo Internetradio last.fm [SCHWIETERING, Heinrich, 2014]

Die in diesem Projekt verwendeten Daten wurden uns ausschließlich von der JKU Linz zur Verfügung gestellt.

8 Einleitung

8.1 Motivation

Diese Diplomarbeit kann dem Bereich der KI, also der künstlichen Intelligenz, zugeordnet werden.
Künstliche Intelligenz ist ein junges Unterkapitel der Informatik, jedoch gewinnt es immer mehr an Wichtigkeit und Aufmerksamkeit. In Form dieser Arbeit haben wir die Möglichkeit mit diesem Bereich der Informatik Bekanntschaft zu machen. Ebenfalls lässt sie uns begreifen, dass die künstliche Intelligenz weit mehr Einflussbereiche und Möglichkeiten hat, als vermutet. Weiters ist das Arbeiten mit großen Datenmengen eine interessante neue Herausforderung, da diese Datengrößen den Rahmen des Unterrichts weit überschreiten.
In Kombination mit einer der heutzutage wichtigsten Kunstgattungen, der Musik, ist es ein Projekt, das auch für Personen, die nicht so sehr mit dem Computer vertraut sind, ein interessantes und leicht verständliches, grafisches Ergebnis bietet.

8.2 Ziel der Arbeit

Das Ziel dieser Diplomarbeit ist es, eine Webapplikation zu implementieren, welche es dem Anwender ermöglicht, einen schnellen und klaren Überblick über alle Musikrichtungen in Form einer Karte zu erhalten, die in den Verzeichnissen des Internetradios last.fm vorkommen. Dabei werden die verschiedenen Musikrichtungen in eigene Bereiche auf der Karte gegliedert. Die Cluster werden dabei durch die, diesem Tag zugeordneten, Interpreten gebildet. Jeder Interpret wird durch einen Punkt im Cluster dargestellt. Wird mit der Maus auf einen dieser Punkte geklickt, so erscheinen neben der Musikkarte Informationen zu diesem Interpreten. Um die verschiedenen Tags visuell untereinander zu unterscheiden werden diese in verschiedenen Farben dargestellt. Um einen genaueren Blick auf die Datenpunkte zu werfen ist es möglich innerhalb der Karte zu zoomen und sich im gezoomten Bild zu bewegen.

Zusätzlich zu der allgemeinen Musikkarte, die oberhalb beschrieben ist, bietet die Applikation die Möglichkeit den individuellen Musikgeschmack eines last.fm Users darzustellen. Dabei werden auf Basis der Eingabe des Benutzernamens, aus der allgemeinen Musikkarte die Interpreten hervorgehoben, die der Benutzer am Liebsten hört. Die übrigen Datenpunkte in der Karte verändern sich nicht. Auf dieser Karte ist es ebenfalls möglich, Informationen über einen Interpreten anzeigen zu lassen, indem man mit der Maus auf den Punkt eines Interpreten klickt. Daraufhin erscheinen neben der Musikkarte Informationen zu diesem Interpreten. Ebenfalls auf der benutzerspezifischen Karte implementiert ist die Zoomfunktion, die analog wie bei der allgemeinen Karte funktioniert.

8.2.1 Technische Ziele

Als technische Ziele in dieser Diplomarbeit heben sich zwei Hauptpunkte hervor. Der erste ist die Bewältigung der enorm großen Datenmengen. Die Schwierigkeit dabei ist es, die optimale Datenmenge auszuwählen, um in akzeptabler Zeit ein zufriedenstellendes Ergebnis geliefert zu bekommen. Dabei sind die begrenzten Ressourcen ausschlaggebend für die Eingrenzung der Datengrößen.
Der zweite Hauptpunkt ist die Dateneinspeisung in den t-SNE Algorithmus, der die Clusteraufteilung aufgrund der Ähnlichkeiten vornimmt und die Positionen der Punkte in der Karte bestimmt. Dabei ist die Bestimmung der korrekten Eingangsparameter essentiell. Je mehr die Parameter vom optimalen Wert abweichen, desto mehr wird die Karte verfälscht. Somit ist es nicht mehr möglich die Musikrichtungen visuell eindeutig erkennbar zu machen.

8.2.2 Allgemeine Ziele

Das allgemeine Ziel der Diplomarbeit ist die Darstellung der großen Menge an Daten in einer möglichst übersichtlichen und ansprechenden Art.
Dafür ist das Design sehr einfach gewählt und bedient sich, zur Verdeutlichung der Darstellung, an einer Palette an Farben.

9 Grundlagen

Das Kapitel Grundlagen beschäftigt sich mit der Beschreibung der verwendeten Technologien, Entwicklungssysteme und Bibliotheken.

9.1 Verwendete Technologien

9.1.1 t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE ist ein Algorithmus, der von Laurens van der Maaten und Geoffrey Hinton entwickelt wurde. Er ist eine Verbesserung des Stochastic Neighbor Embedding (SNE) Algorithmus und wird dem Bereich des maschinellen Lernens zugeordnet. Maschinelles Lernen erkennt Muster und Gesetzmäßigkeiten in den Daten und ist ein Unterkapitel der künstlichen Intelligenz. Es wird verwendet, da die Rechenleistung mittlerweile verfügbar ist und ein Bedarf an autonomer maschineller Intelligenz besteht. (vgl. [NYSRET MUSLIU, Wolfgang Slany, 2016])
t-SNE ist besonders für die Überführung von hochdimensionalen Datensätze in zwei- oder dreidimensionale Daten geeignet. In niederdimensionalen Darstellungen werden ähnliche Objekte durch nahe beisammen liegende Datenpunkte und sich stark unterscheidende Objekte durch demensprechend weit verstreute Datenpunkte dargestellt. (vgl. [WIKIPEDIA, 2016])

Der Basisalgorithmus SNE beginnt damit, die hochdimensionalen euklidischen Abstände zwischen den Datenpunkten in bedingte Wahrscheinlichkeiten, die Ähnlichkeiten darstellen, umzuwandeln.
Obwohl es SNE ermöglicht, gute Visualisierungen zu erstellen wird es durch ein Phänomen behindert, das sich das Crowding Problem nennt. Das Crowding Problem ist im Kapitel 9.1.1.1 erklärt.
t-SNE unterscheidet sich von SNE in zwei Punkten. Erstens verwendet es eine symmetrisch gemachte Variante der Kostenfunktion mit einfacheren Steigungen. Zweitens verwendet es anstatt der Gaußschen Verteilung die Student-t Verteilung, um das Crowding Problem zu minimieren. (vgl. [LAURENS VAN DER MAATEN, Geoffrey Hinton, 2016])

Die Vorgehensweise von t-SNE kann in zwei Hauptschritte unterteilt werden. Der Algorithmus erstellt zuerst ein Wahrscheinlichkeitsmaß über Paare von hochdimensionalen Datensätzen, sodass ähnliche Paare eine hohe Ziehungswahrscheinlichkeit und unterschiedliche Paare eine winzige Ziehungswahrscheinlichkeit bekommen. Als nächsten Schritt erstellt t-SNE eine ähnliche Wahrscheinlichkeitsverteilung über die Punkte in der niederdimensionalen Karte und minimiert die Unterschiede der Wahrscheinlichkeiten. (vgl. [WIKIPEDIA, 2016])
Der Algorithmus t-SNE hat eine weite Einsatzbreite wie Computerwissenschaft, Musikanalyse und Bioinformatik.

t-SNE wird in dieser Diplomarbeit eingesetzt, um die hochdimensionalen Datensätze des Internetradios last.fm, die die Ähnlichkeiten der Artisten untereinander festhalten, in eine zweidimensionale Karte zu transformieren.

9.1.1.1 The Crowding Problem

Dazu ein Beispiel. Falls die Datenpunkte in einem zehndimensionalen Raum ungefähr einheitlich um einen bestimmten Bereich verteilt sind und die Abstände zu allen anderen Punkten bestimmt werden sollen, tritt das Crowding Problem auf. Der Bereich der zweidimensionalen Karte, der verfügbar ist, um entfernte Datenpunkte aufzunehmen ist im Vergleich zu dem Bereich, der verfügbar ist, um nahegelegene Punkte aufzunehmen, nicht annähernd groß genug. Weiter entfernte Datenpunkte wären also weit außerhalb der Karte platziert. Um diese Datenpunkte dennoch in der Karte darstellen zu können, zieht SNE die Datenpunkte, ähnlich einer Gravitationskraft, zur Mitte. Obwohl diese Kraft relativ gering ist, zerquetscht sie die Punkte in der Mitte und verhindert eine eindeutige Clusterbildung. (vgl. [LAURENS VAN DER MAATEN, Geoffrey Hinton, 2016])

9.1.1.2 Alternativen zu t-SNE

Alle, der im Folgenden aufgelisteten Technologien sind für die Dimensionsreduktion hochdimensionaler Daten geeignet. In dieser Diplomarbeit wird t-SNE verwendet, da es eine mehrfach ausgezeichnete Technik ist, die besonders für die Visualisierung von Daten geeignet ist, die aus der echten Welt extrahiert wurden. Das Verfahren ist zahlreich ausgetestet und weist, im Gegensatz zu anderen Verfahren, keine Einengung der Zuordnung, zwischen den Daten und der Visualisierung, auf. (vgl. [MAATEN, Laurens van der, 2016])

9.1.1.2.1 Isomap

Isomap ist eines der am Stärksten verwendeten niederdimensionalen Einbettungsverfahren. Der Algorithmus bietet eine einfache Methode zum Feststellen der Geometrie der Datenpunkte in einem Verteiler durch Schätzung der Nachbarn jedes einzelnen Datenpunktes. Isomap ist ein sehr effizientes Verfahren und ist auf unterschiedlichen Datenquellen und Dimensionen anwendbar. (vgl. [TENENBAUM, J.B., 2000])

9.1.1.2.2 Sammon Mapping

Sammon Mapping beschäftigt sich nicht direkt mit der Transformationsfunktion, sondern versucht aus dem hochdimensionalen Dataset ein Dataset mit weniger Dimensionen zu erzeugen, das jedoch die Struktur des Originals so weit als möglich beibehält. (vgl. [HENDERSON, Paul, 2016])

9.1.1.2.3 Selbstorganisierende Karte

Die Selbstorganisierende Karte ist ein Werkzeug des Data Mining. Die Knoten der Karte besitzen jeweils einen Gewichtsvektor. Auf Basis einer Aktivierungsfunktion werden die Gewichtsvektoren der Punkte angepasst, dass sie dem Eingabevektor ähnlicher werden. Anwendungen der selbstorganisierenden Karte findet man in der Computergrafik und in der Bioinformatik. (vgl. [RAUBER, Andreas, 1998])

9.1.2 R

R ist eine freie Programmiersprache, die für den Umgang mit großen Datenmengen und deren Analyse entwickelt wurde. Durch weitere, online abrufbare Pakete lassen sich Daten in verschiedensten Bereichen auswerten. (vgl. [PROJECT, R, 2016])

Abb. 12 Logo der Programmiersprache R [PROJECT, R, 2016]

In dieser Diplomarbeit wird der t-SNE Algorithmus in der Programmiersprache R angewandt.

9.1.3 Java

Java ist eine plattformunabhängige Programmiersprache, was bedeutet, dass Anwendungen, die in Java geschrieben sind, auf allen Plattformen genutzt werden können, auf denen Java installiert ist.
Java ist auch objektorientiert, das heißt, dass die Programme in Einheiten unterteilt sind. Diese Einheiten werden Objekte genannt. Nur die im Objekt vorhandenen Funktionen können dessen Daten verändern. Das Objekt befindet sich also in einem wohldefinierten, selbstkontrollierten Zustand. (vgl. [ITWISSEN, 2016])
Objekte können auf vielfältige Weise miteinander in Verbindung stehen.

Abb. 13 Logo der Programmiersprache Java [WIKIPEDIA, 2016]

Grob kann die Java Technologie in das Java Entwicklungswerkzeug, die JDK, und die Java Laufzeitumgebung, die JRE, unterteilt werden. (vgl. [UBUNTUUSERS, 2016])

Die Programmiersprache Java wird in dieser Diplomarbeit verwendet, um den Datenvektor zu generieren, sowie die Technologien JSF, siehe 9.1.5, und JPA, siehe 9.1.4, anzuwenden.

9.1.4 Java Persistence API (JPA)

Die Java Persistence API ist eine Sammlung von Klassen und Methoden, die dazu dienen, die Zuordnung und die Übertragung von Objekten zu Datenbankeinträgen zu vereinfachen. (vgl. [TUTORIALSPOINT, 2016])

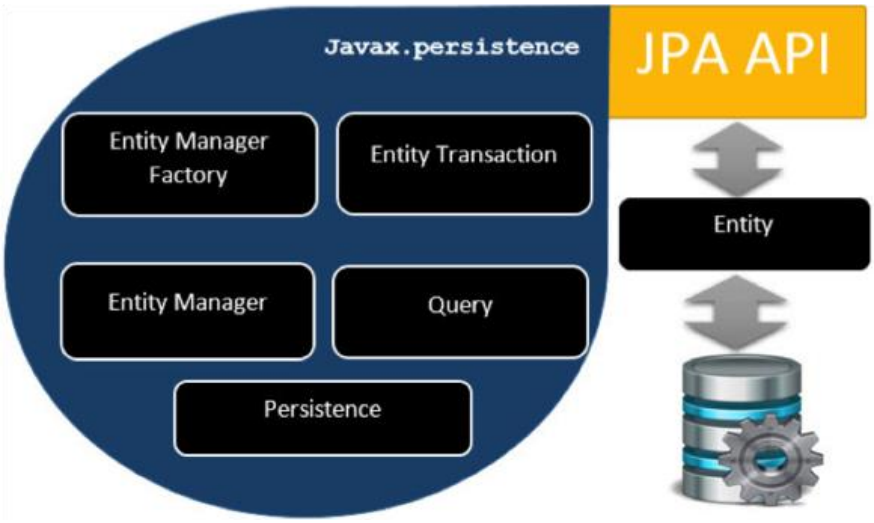

Abb. 14 JPA Funktionsweise [TUTORIALSPOINT, 2016]

Bestandteile	**Beschreibung**
EntityManagerFactory	Die Factory erstellt und verwaltet mehrere EntityManager Instanzen.
EntityManager	Der EntityManager ist ein Interface, das die Operationen auf die Datenobjekte verwaltet.
Entity	Entities sind die persistenten Objekte, die auf der Datenbank gespeichert sind und im Programm verwaltet werden.
EntityTransaction	Die EntityTransaction ist ein Interface, das die Transaktion eines EntityManagers verwaltet.
Persistence	Persistence ist eine Klasse, die dazu dient eine EntityManagerFactory Instanz zu erzeugen.
Query	Query stellt ein Interface dar, das es ermöglicht Datenbankabfragen durchzuführen.

Tabelle 1 JPA Bestandteile [TUTORIALSPOINT, 2016]

Die JPA Technologie wird in der Applikation verwendet, um die Verbindung zwischen den Objekten im Programm und der Datenbank herzustellen.

9.1.5 JavaServer Faces (JSF)

JavaServer Faces ist eine Sammlung von APIs für die Erstellung von grafischen Benutzeroberflächen. Das Framework basiert auf der Servlet- und JSP-Technik und gehört somit zu den Webtechnologien von Java. JSF ist eine flexible Technologie, die Entwickler in Bezug auf Auszeichnungssprache, Protokoll und Endgerät nicht einschränkt.
Vorrausetzung für die Entwicklung einer JSF Applikation sind Kenntnisse in der Programmiersprache Java sowie HTML, ein Applikationsserver und das JDK. (vgl. [IRIAN, 2016], [ORACLE, 2016])

9.1.5.1 WildFly

WildFly ist ein freier, plattformunabhängiger Anwendungsserver der in Java programmiert ist.

In dieser Diplomarbeit werden JSF und WildFly verwendet, um die Webapplikation zu schreiben, die die Musikkarte anzeigt.

Abb. 15 Logo WildFly [JBOSS, 2016]

9.1.6 Hypertext Markup Language (HTML)

HTML dient der Strukturierung und Gliederung digitaler Dokumente, nicht aber deren Formatierung. Sie ist also eine Auszeichnungssprache.
Inhalte dieser Dokumente können Links, Bilder und Tabellen aber natürlich auch Texte sein.
Diese digitalen Dokumente sind der Grundbaustein des World Wide Web und werden von den Webbrowsern gelesen und dementsprechend angezeigt. Die aktuelle Version ist HTML5. (vgl. [SELFHTML, 2016])

Abb. 16 Logo HTML5 [WIKIPEDIA, 2016]

HTML wird in dieser Diplomarbeit verwendet, um den Seitenaufbau der Applikation zu definieren.

9.1.7 Cascading Style Sheets (CSS)

CSS ist eine Formatierungssprache für HTML- und XML-Dokumente. Sie dient dazu, das Erscheinungsbild und die Formatierung von Dokumenten festzulegen.
Inhalte, die mit HTML (siehe 9.1.6) strukturiert werden, besitzen zwar schon vorgegebene Formatierung, diese lassen sich jedoch mit CSS auf vielfältige Art verändern und erweitern. (vgl. [SELFHTML, 2016])

Abb. 17 Logo CSS3 [OPENCODE, 2016]

Mit CSS ist es möglich, ähnlich wie in Microsoft Word mit Formatvorlagen, bestimmten Inhalten spezielle Formatierung zuzuweisen. Damit wird eine andauernde händische Formatierung nicht benötigt. Die aktuelle Version ist CSS3.

In dieser Diplomarbeit wird CSS angewandt, um die Webseite rund um die Musikkarte grafisch ansprechend zu gestalten.

9.1.8 JavaScript

JavaScript ist eine plattformunabhängige Skriptsprache, die entwickelt wurde, um die Möglichkeiten von HTML (siehe 9.1.6) und CSS (siehe 9.1.7) zu optimieren.
Mit JavaScript lassen sich dynamische Webseiten erstellen, die auf Eingaben des Benutzers reagieren. Weiters ist es möglich Inhalte der Webseite zu verändern und Inhalte nachzuladen.

Abb. 18 Logo JavaScript [DIFF, Visual, 2016]

Typische Verwendungszwecke von JavaScript sind:

- Dynamische Manipulation der Webseite
- Korrektheitsüberprüfung von Dateneingaben noch vor der Serverübertragung
- Datenaustausch, ohne, dass der Browser die Webseite neu laden muss, siehe Kapitel 9.1.9

(vgl. [SELFHTML, 2016])

JavaScript wird in dieser Diplomarbeit benötigt um das Herzstück der Webseite, also die Musikkarte entsprechend darzustellen.

9.1.9 Asynchronous JavaScript and XML (AJAX)

AJAX bezeichnet ein Konzept der asynchronen Datenübertragung zwischen dem Browser und dem Server.

Abb. 19 Logo AJAX [WIKIPEDIA, 2016]

Der Vorteil durch die Verwendung von AJAX ist, dass die Webseite nicht neu geladen werden muss um diese zu aktualisieren. Weiters können, auch nachdem die Seite fertig geladen hat, Daten zum Server geschickt werden und Daten vom Server empfangen werden.
Durch diese Funktionen erzeugt AJAX ein Gefühl der lebendigen Webseite. (vgl. [W3SCHOOLS, 2016])

AJAX wird bei dieser Diplomarbeit für den Datentransfer mit dem Server, auch nach dem Fertigladen der Webseite, verwendet.

9.1.10 Structured Query Language (SQL)

SQL ist eine Datenbanksprache zur Definition von Datenbankstrukturen und zum Abfragen von Daten. Sie hat das gleiche Grundprinzip auf allen Datenbanken, jedoch unterscheidet sich die Syntax bei verschiedenen Datenbankanbietern.

Abb. 20 Logo SQL [PETRIW, Zachary, 2016]

SQL kann in die drei Bereiche Data Manipulation Language (DML), Data Definition Language (DDL) und Data Control Language (DCL) unterteilt werden. (vgl. [FRITZL, Florian, 2015])

Die Abfragesprache SQL wird in dieser Diplomarbeit verwendet, um Daten aus der Datenbank auszulesen.

9.1.11 Windows Server 2012 R2

Windows Server 2012 R2 ist ein Betriebssystem von Microsoft, das im Oktober 2013 erschienen ist. (vgl. [MICROSOFT, 2016])

Das Betriebssystem ist auf dem Server der HTL Perg installiert, auf dem der t-SNE Algorithmus die Berechnungen durchgeführt hat.

9.2 Verwendete Entwicklungssysteme

9.2.1 NetBeans IDE

Die NetBeans IDE (Integrated Development Environment, dt. integrierte Entwicklungsumgebung) ist eine Open-Source Entwicklungsumgebung die komplett in der Programmiersprache Java geschrieben ist und auch hauptsächlich zum Programmieren dieser entwickelt wurde. Die IDE ist auch in deutscher Sprache verfügbar und ist mit jener von Eclipse (siehe 9.2.2) vergleichbar. (vgl. [NETBEANS, 2016])

NetBeans IDE wird in dieser Diplomarbeit für die Vektorgenerierung eingesetzt. Das Ergebnis wird anschließend in den t-SNE Algorithmus eingebracht.

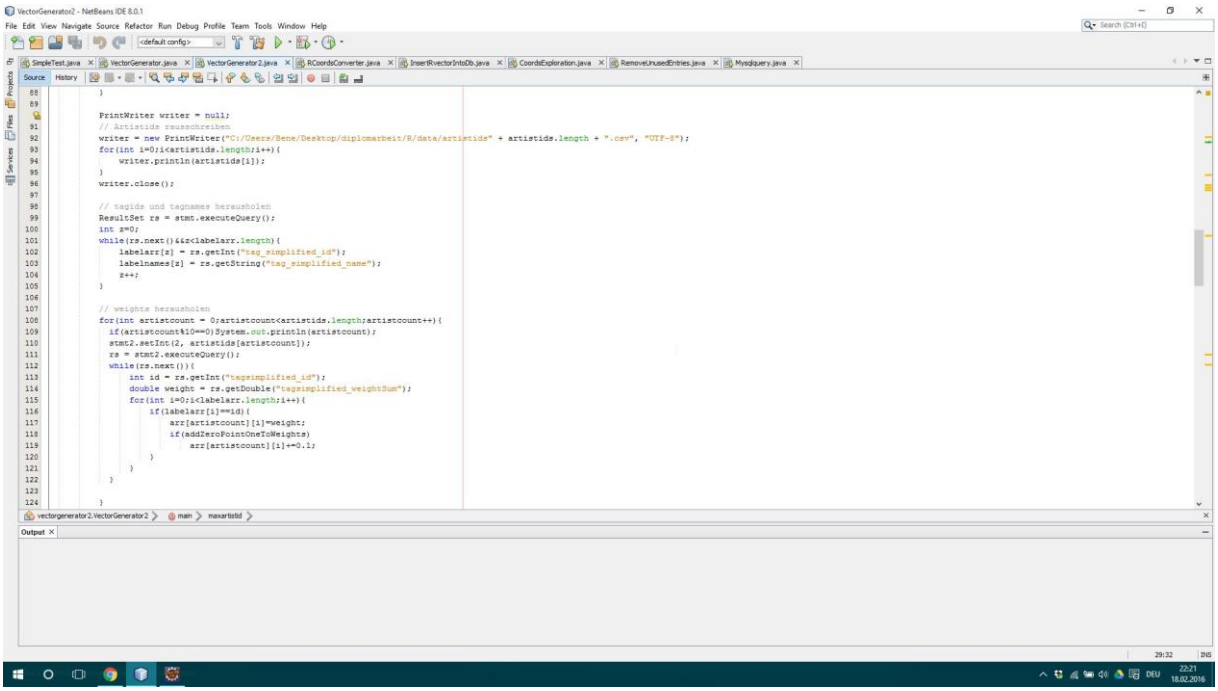

Abb. 21 Entwicklungsumgebung NetBeans IDE

9.2.2 Eclipse IDE

Die Eclipse IDE (Integrated Development Environment, dt. integrierte Entwicklungsumgebung) ist ein freies Programmierwerkzeug zum Entwickeln von Software. Das ursprünglich für die Programmiersprache Java genutzte Werkzeug, wird mittlerweile aufgrund seiner Erweiterbarkeit auch für andere Entwicklungsaufgaben eingesetzt. (vgl. [TECHTARGET, 2016])

Eclipse dient in dieser Diplomarbeit dazu die Webapplikation, inklusive Datenübertragung und Datenbankzugriff, zu implementieren.

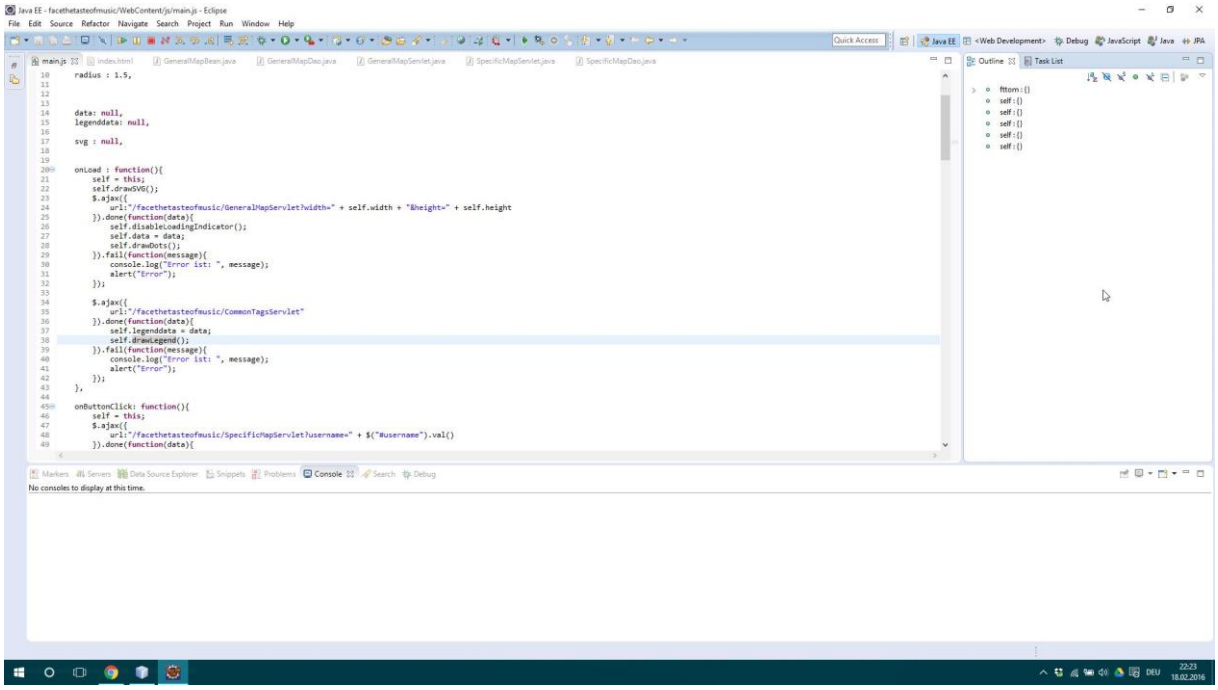

Abb. 22 Entwicklungsumgebung Eclipse IDE

9.2.3 Notepad++

Notepad++ ist ein frei verfügbarer Code Editor, der sich natürlich auch zum Verfassen einfacher Texte verwenden lässt. Seine wahre Stärke liegt jedoch im Umgang mit zahlreichen Programmiersprachen wie C, C++, Java, JavaScript und HTML. Durch farbige Hervorhebungen lässt er den Programmierer den Überblick behalten. Eine weitere Hilfe für den Programmierer ist die Autovervollständigung für häufig benutzte Ausdrücke. (vgl. [ONLINE, heise, 2016])

Notepad++ wird in dieser Diplomarbeit zum Anzeigen und Bearbeiten der Vektordateien verwendet.

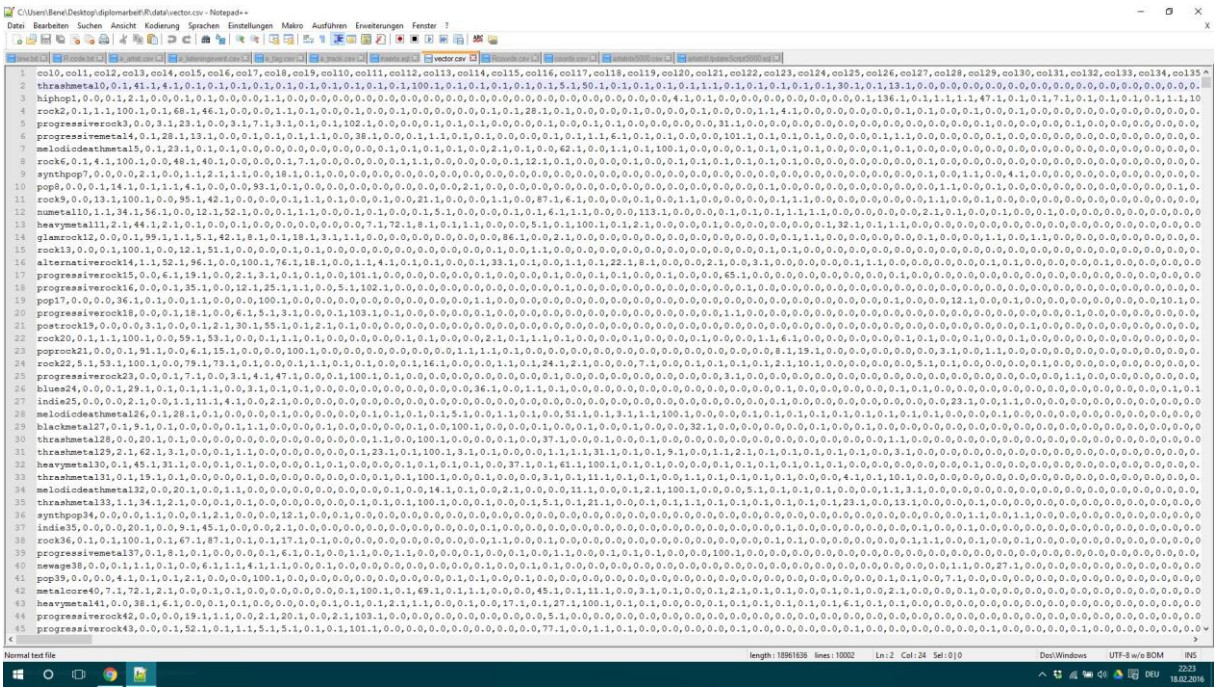

Abb. 23 Code Editor Notepad++

9.2.4 MySQL Workbench

Die MySQL Workbench ist ein Datenbank – Modellierungswerkzeug, das Datenmodellierung, SQL Entwicklung und viele weitere administrative Funktionen kombiniert. Die MySQL Workbench ist ausschließlich für MySQL Datenbanken anwendbar.
Eine sehr wichtige Funktion der Workbench ist das Erstellen von Datenmodellen und das daraus mögliche Erzeugen der Datenbank. (vgl. [MYSQL, 2016])

In dieser Diplomarbeit wird die MySQL Workbench verwendet, um die von der JKU zur Verfügung gestellte Datenbank einzubinden und auf diese Abfragen durchzuführen.

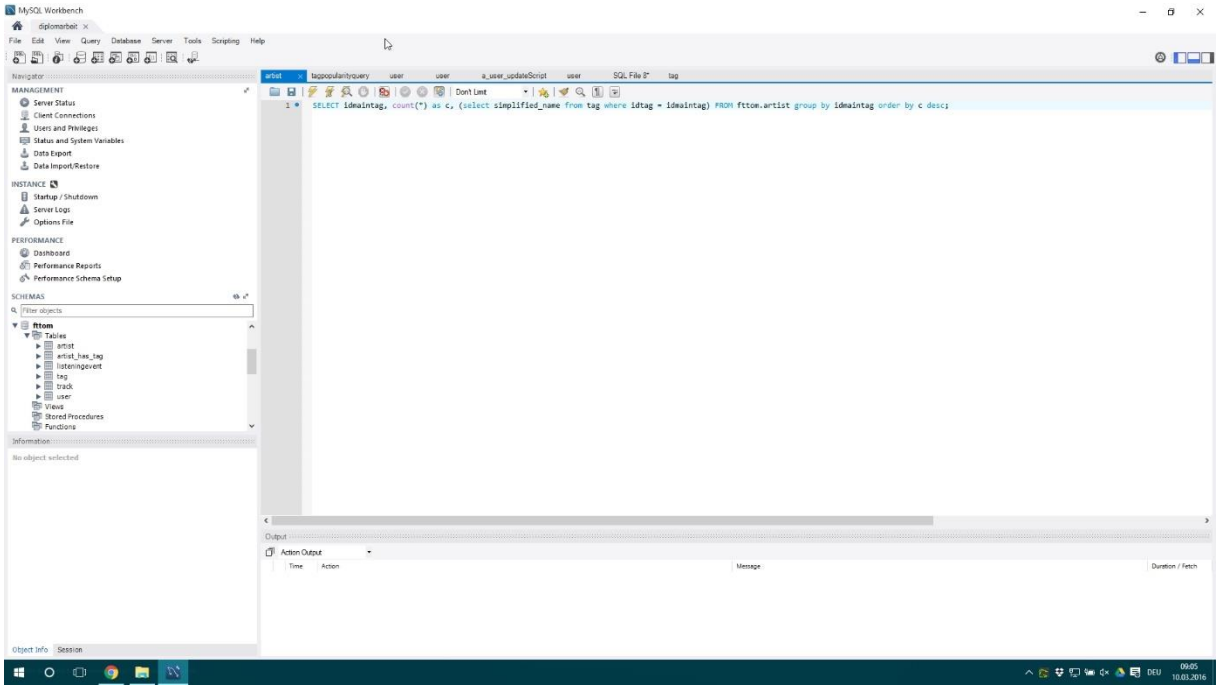

Abb. 24 MySQL Workbench

9.2.5 RGui

Die Entwicklungsumgebung RGui ist eine Kommandozeilenumgebung, die in der R-Installation enthalten ist. (vgl. [PROJECT, R, 2016])

RGui wird in dieser Diplomarbeit verwendet, um den, in der Programmiersprache R (siehe 9.1.2) bereits implementierten, t-SNE Algorithmus zu verwenden.

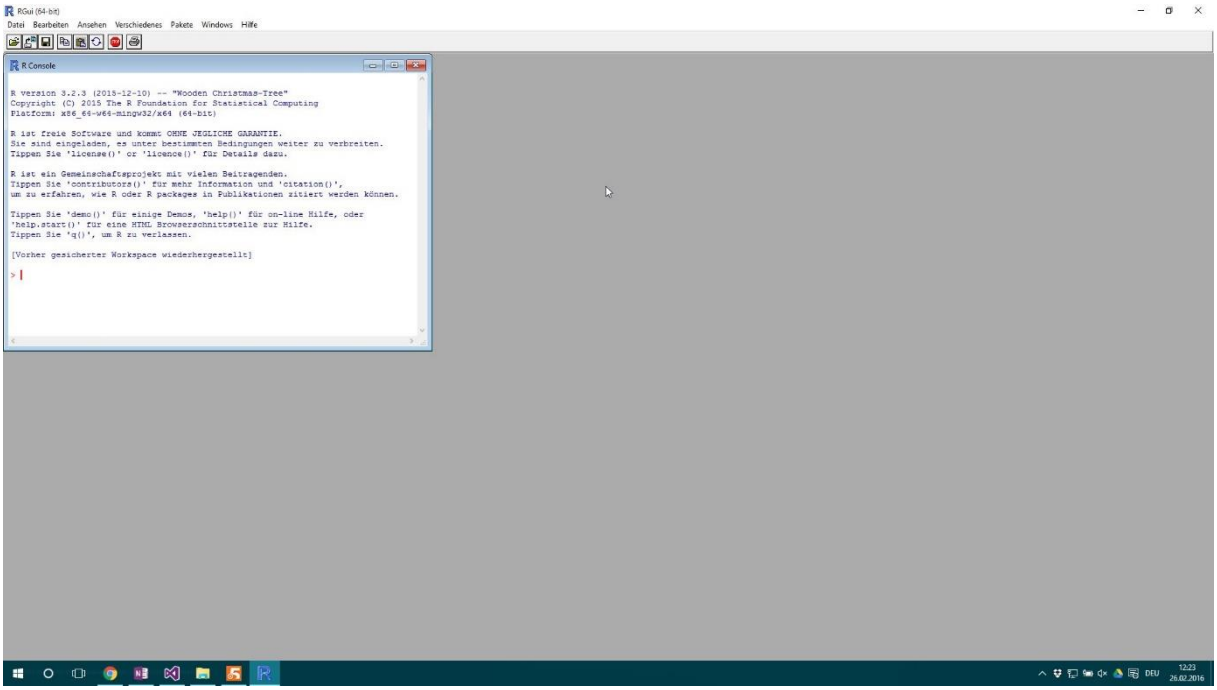

Abb. 25 Entwicklungsumgebung RGui

9.2.6 Matlab

Matlab ist eine Software zum Lösen mathematischer Probleme und zur grafischen Darstellung der Ergebnisse. Das Programm dient primär der numerischen Lösung von Problemen. Die Software wird in der Industrie und an Hochschulen für Simulationen, Datenerfassungen, Datenanalysen und Datenauswertungen eingesetzt. (vgl. [MATHWORKS, 2016])

Abb. 26 Logo Matlab [RAGUDO, Marielli Gem A., 2016]

In dieser Diplomarbeit wurde Matlab verwendet, um eine Version des t-SNE Algorithmus zu testen. Näheres zu dieser Version ist im Kapitel 10.4.3.3 erklärt.

9.3 Verwendete Bibliotheken

9.3.1 D3.js

Abb. 27 Logo D3.js [WIKIPEDIA, 2016]

Die Bezeichnung D3 steht für Data Driven Documents. Es ist eine JavaScript Bibliothek für die interaktive, dynamische und ansprechende Visualisierung von Daten in Webbrowsern und die Manipulation des Document Object Models. Dabei bedient es sich Techniken von HTML (siehe 9.1.6), CSS (siehe 9.1.7) und SVG. Scalable Vector Graphics ist eine Spezifikation zur Beschreibung von zweidimensionalen Vektorgrafiken. Dabei fordert D3 alle momentan zur Verfügung stehenden Möglichkeiten der Web Programmierung. D3 ermöglicht es ebenfalls Interaktionen und Animationen von sehr großen Datenmengen durchzuführen. (vgl. [BOSTOCK, Mike, 2016])

Die JavaScript Bibliothek D3.js wird in dieser Diplomarbeit verwendet, um die Datenpunkte in Form der Musikkarte darzustellen.

9.3.2 jQuery

Abb. 28 Logo jQuery [SAMEER, 2014]

jQuery ist eine freie JavaScript Bibliothek, mit der die Manipulation des DOM einfacher und schneller durchzuführen ist. Weiters ist es möglich Benutzerinteraktionen abzufangen und zu behandeln. Die Möglichkeiten der Darstellung und Veränderungen der Webseite werden ungemein erweitert. Auch die Verbindung mit AJAX (siehe 9.1.9) ist schnell hergestellt und durch viele Plugins werden viele weitere Funktionen zur Verfügung gestellt. (vgl. [HTML-SEMINAR])

jQuery wird in dieser Diplomarbeit verwendet, um dynamische Funktionen auf der Webseite auszuführen.

9.4 Verwendete Architekturen

9.4.1 DAO (Data Access Object)

Das Data Access Object Pattern wird verwendet um niedere von den höheren Geschäftsschichten zu trennen.

DAO besteht aus folgenden Bestandteilen:

DAO Klasse
Die Klasse ist verantwortlich, die Daten von einer Datenbank oder einer anderen Datenquelle zu empfangen.

Model
Das Model repräsentiert ein Objekt in der Datenbank und verwendet die ihm, in der DAO Klasse, zur Verfügung gestellten Methoden. (vgl. [COMMUNITY, tutorialspoint, 2016])

Das Data Access Object Pattern wird in dieser Diplomarbeit verwendet, um eine logische Trennung der Datenschichten vorzunehmen.

9.4.2 REST (Representational State Transfer)

REST bezeichnet ein Leitbild zur Erstellung von Webservices. Webseiten, Bilder und Servlets stellen Ressourcen dar. Diese Ressourcen werden nicht direkt manipuliert, denn der Zugriff erfolgt indirekt über die der Ressource zugeordneten URI.

Die meisten Anwendungsfälle können mit den HTTP Methoden GET, PUT, POST und DELETE abgedeckt werden, die jeder REST Ressource zur Verfügung stehen. (vgl. [BAYER, Thomas, 2002])

In dieser Diplomarbeit wird REST verwendet, um den Datenzugriff am Servlet zu realisieren.

9.5 Sonstige verwendete Software

9.5.1 Dropbox

Dropbox ist ein sogenannter Cloud Storage, also ein Online Speicher. Dies bedeutet, dass Dateien, die auf Dropbox hochgeladen wurden, auf jedem ans Internet angeschlossenen Gerät abrufen werden können. Der Zugriff auf Dropbox erfolgt entweder über den Browser oder mithilfe von Apps. (vgl. [SLACK, 2016])

Dropbox wird im Zuge dieser Diplomarbeit zur Datenspeicherung und zum Datenaustausch verwendet.

Abb. 29 Logo Dropbox [COMBS, Jason, 2013]

9.5.2 Microsoft Office 2013

Microsoft Office 2013 stellt ein umfangreiches Paket verschiedenster Programme dar, die im Büroalltag verwendet werden.
Die folgenden Programme werden im Zuge der Diplomarbeit verwendet.

Abb. 30 Logo Microsoft Office 2013 [COMMONS, Wikimedia, 2016]

9.5.2.1 Microsoft Word 2013

- Dokumentation
- Schriftliche Ausarbeitungen

Abb. 31 Logo MS Word 2013 [COMMONS, Wikimedia, 2016]

9.5.2.2 Microsoft Excel 2013

- Stundenlistenführung
- Aufgabenliste

Abb. 32 Logo MS Excel 2013 [COMMONS, Wikimedia, 2016]

9.5.2.3 Microsoft PowerPoint 2013

- Erstellung von Projektpräsentationen

Abb. 33 Logo MS PowerPoint 2013 [COMMONS, Wikimedia, 2016]

9.5.3 GNU Image Manipulation Program (GIMP)

GIMP ist ein kostenloses Bildbearbeitungsprogramm. Es enthält viele nützlich Tools und Funktionen zur Bildbearbeitung.

Abb. 34 Logo GIMP [COMMONS, Wikimedia, 2016]

Im Zuge dieser Diplomarbeit wird es zum Modifizieren und Gestalten von Grafiken, für die Erstellung des Logos und weitere Kleinarbeiten verwendet.

10 Realisierung

Das Kapitel Realisierung beschäftigt sich mit der Planung und Umsetzung der Webapplikation unter Verwendung der beschriebenen Technologien.

10.1 Planung

10.1.1 Funktionale Idee

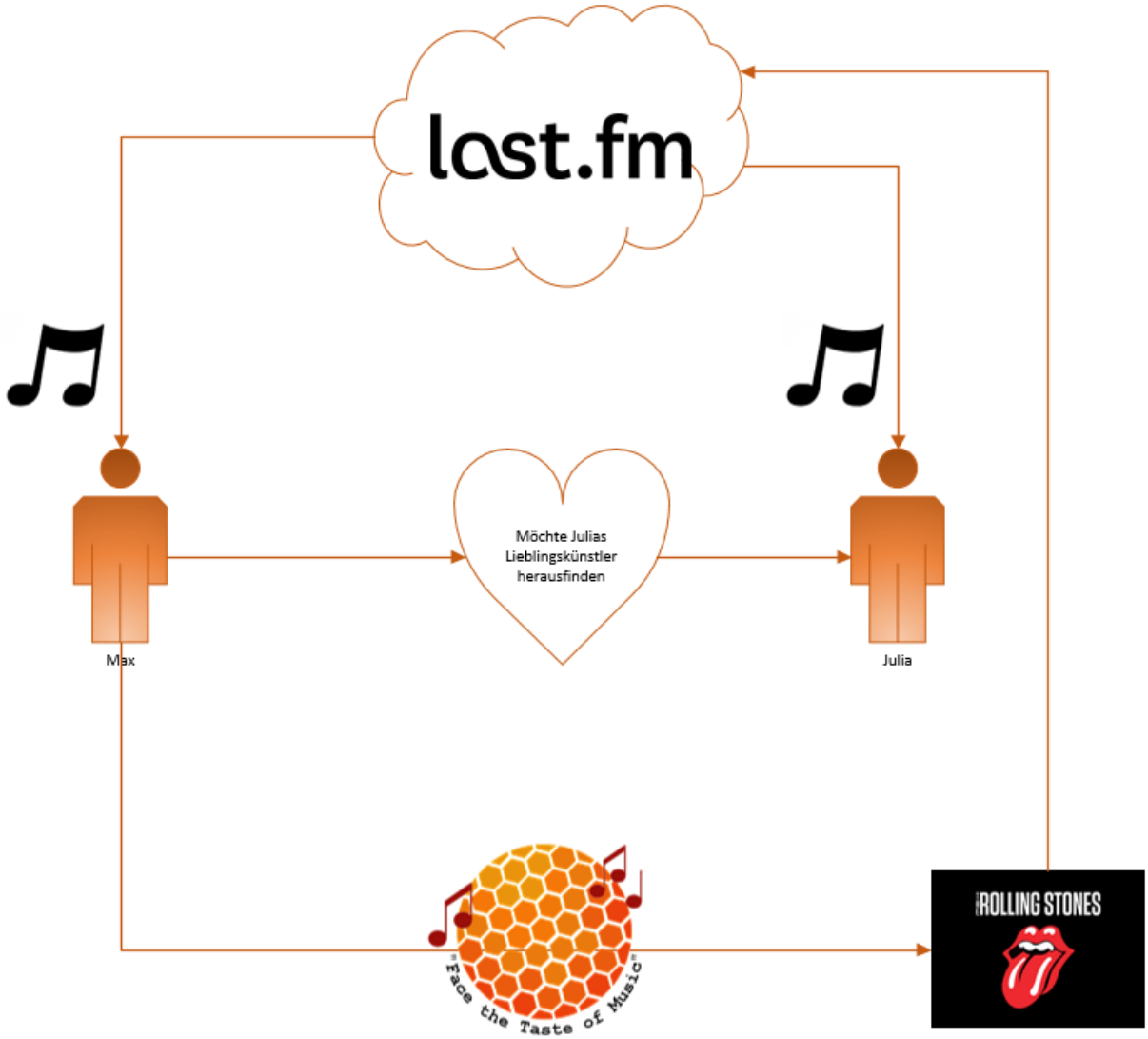

Abb. 35 Funktionale Idee

Diese Grafik stellt die funktionale Idee dieser Diplomarbeit grafisch dar. Sie zeigt also in einer, aus dem Leben gegriffenen Situation, die mögliche Verwendungsweise der Webapplikation Face the Taste of Music.

Die Situation ist folgende: Max hört gerne Musik. Er ist ein angemeldeter Benutzer beim Internetradio last.fm. Julia ist ebenfalls angemeldeter Benutzer bei last.fm. Max mag Julia sehr und möchte wissen, was nun Julias Lieblingsmusik ist um sie zu beeindrucken. Max hat eine Idee. Er gibt in der Webapplikation Face the Taste of Music Julias Benutzernamen ein und erhält sofort einen optisch aufbereiteten Überblick über Julias Musikgeschmack. Max findet heraus, dass Julia gerne die „Rolling Stones“ hört. Nun kann sich Max wiederum auf last.fm die Musik der Rolling Stones anhören.

10.1.2 Use-Case-Diagramm

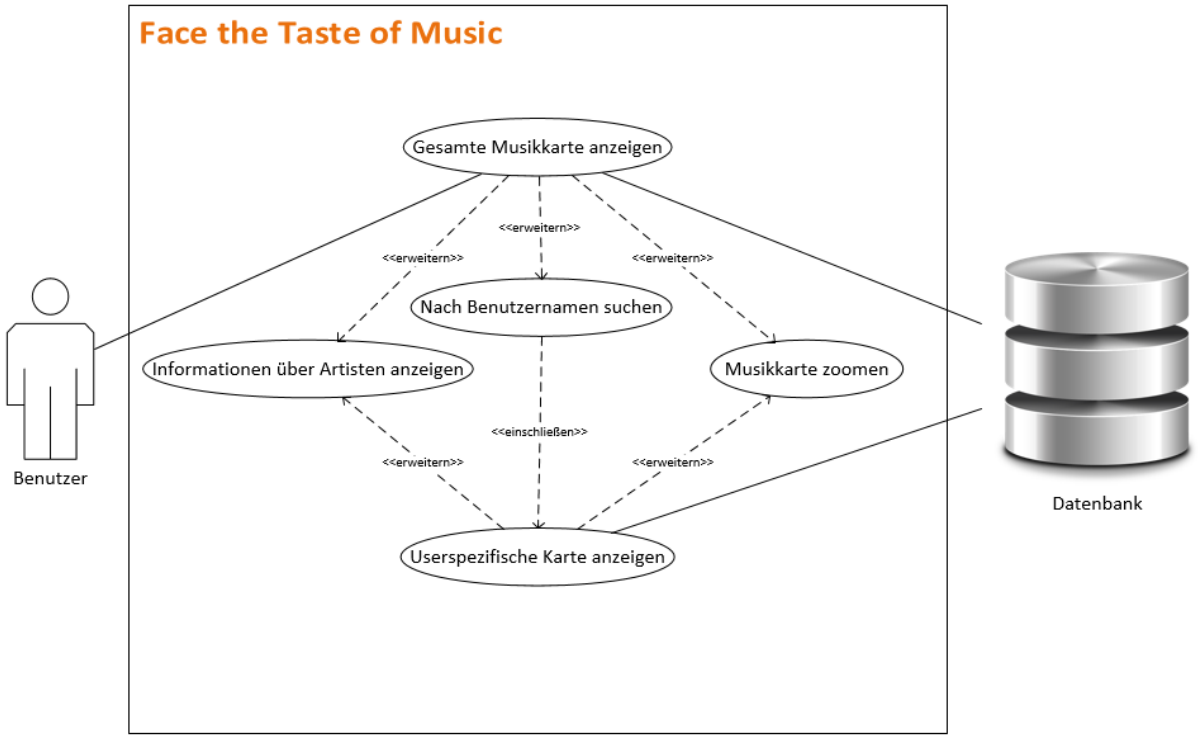

Abb. 36 Use-Case-Diagramm

Das Use-Case-Diagramm der Applikation Face the Taste of Music besteht aus fünf Use Cases:

- Gesamte Musikkarte anzeigen
- Interpret Informationen anzeigen
- Nach Benutzernamen suchen
- Userspezifische Karte anzeigen
- Musikkarte zoomen

10.1.2.1 Gesamte Musikkarte anzeigen

Nachdem der Benutzer die Webapplikation aufruft, erscheint die allgemeine Musikkarte als Startseite. Die für diese Karte benötigten Daten wurden aus der für diese Applikation zur Verfügung gestellten Datenbank herausgefiltert. Um dem Benutzer eine übersichtliche Darstellung zu bieten, werden die verschiedenen Tags farblich gekennzeichnet. Ebenfalls befindet sich rechts unten, neben der Karte, eine Legende, in der die Farben und die Namen der dazugehörigen Tags aufgelistet sind. Wenn sich der Mauszeiger über einem Tag in der Legende befindet, so werden alle Interpreten dieses Tags in der Karte hervorgehoben.

10.1.2.2 Interpret Informationen anzeigen

Der Benutzer hat die Möglichkeit auf der Karte schnell Informationen über einen beliebigen Interpreten zu erhalten. Er muss dazu lediglich mit der Maus auf einen der Datenpunkte, die Interpreten darstellen, klicken. Rechts oben, neben der Karte, erscheint daraufhin ein Informationsbereich. In diesem sind der Name des Interpreten, das Tag, zu dem der Interpret die höchste Zugehörigkeit hat, die MBID des Interpreten und ein Link zur last.fm Seite des Interpreten aufgelistet. Die MBID ist im Kapitel 10.1.5.3 erklärt.

10.1.2.3 Nach Benutzernamen suchen

Dem Benutzer wird die Funktion geboten, nach einem auf last.fm angemeldeten Benutzer zu suchen. Dafür sind oberhalb der Karte ein Textfeld und ein Button platziert. Gibt der Benutzer in das Textfeld einen ungültigen Benutzernamen ein und drückt auf den Button, so erscheint der Hinweis, dass dieser Benutzername nicht gefunden werden konnte.
Ist der Benutzername gültig, so verändert sich die allgemeine Karte dahingehend, dass die userspezifische Karte angezeigt wird.

10.1.2.4 Userspezifische Karte anzeigen

Auf der userspezifischen Karte wird dem Benutzer die Möglichkeit geboten, die, für den in dem vorherigen Use Case eingegebenen User, bevorzugten Interpreten farblich hervorzuheben. Dabei wird auch einkalkuliert wie oft der User einen bestimmten Interpreten gehört hat. Je nach Häufigkeit wird der Interpret in der Karte kleiner oder größer markiert.

10.1.2.5 Musikkarte zoomen

Sowohl auf der allgemeinen, als auch auf der benutzerspezifischen Karte ist es möglich, in der Karte zu zoomen und bestimmte Bereiche größer darzustellen und genauer zu betrachten. Das Herauszoomen ist natürlich ebenfalls möglich.

10.1.3 Projektstrukturplan

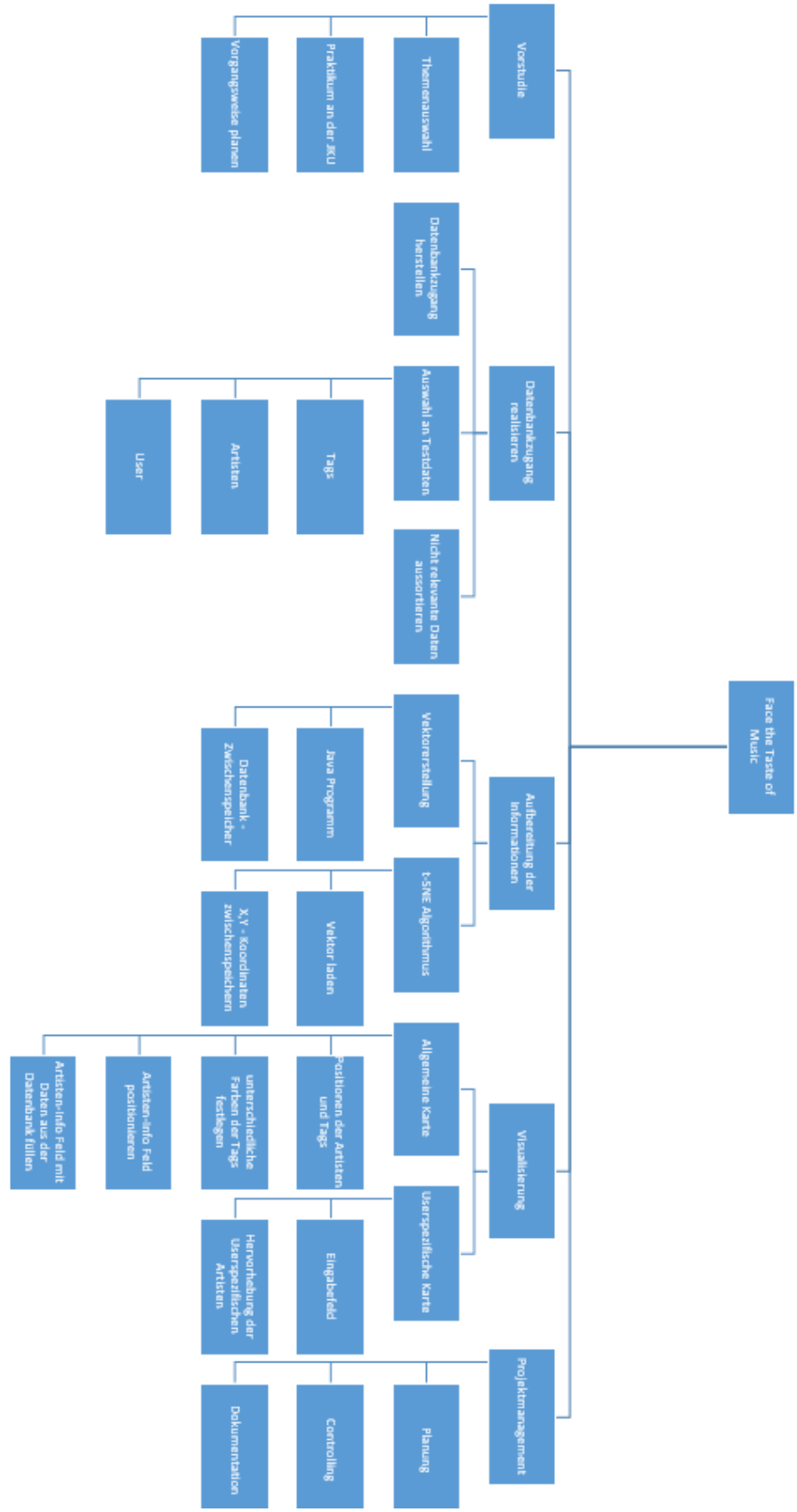

Abb. 37 Projektstrukturplan

10.1.4 Meilensteine

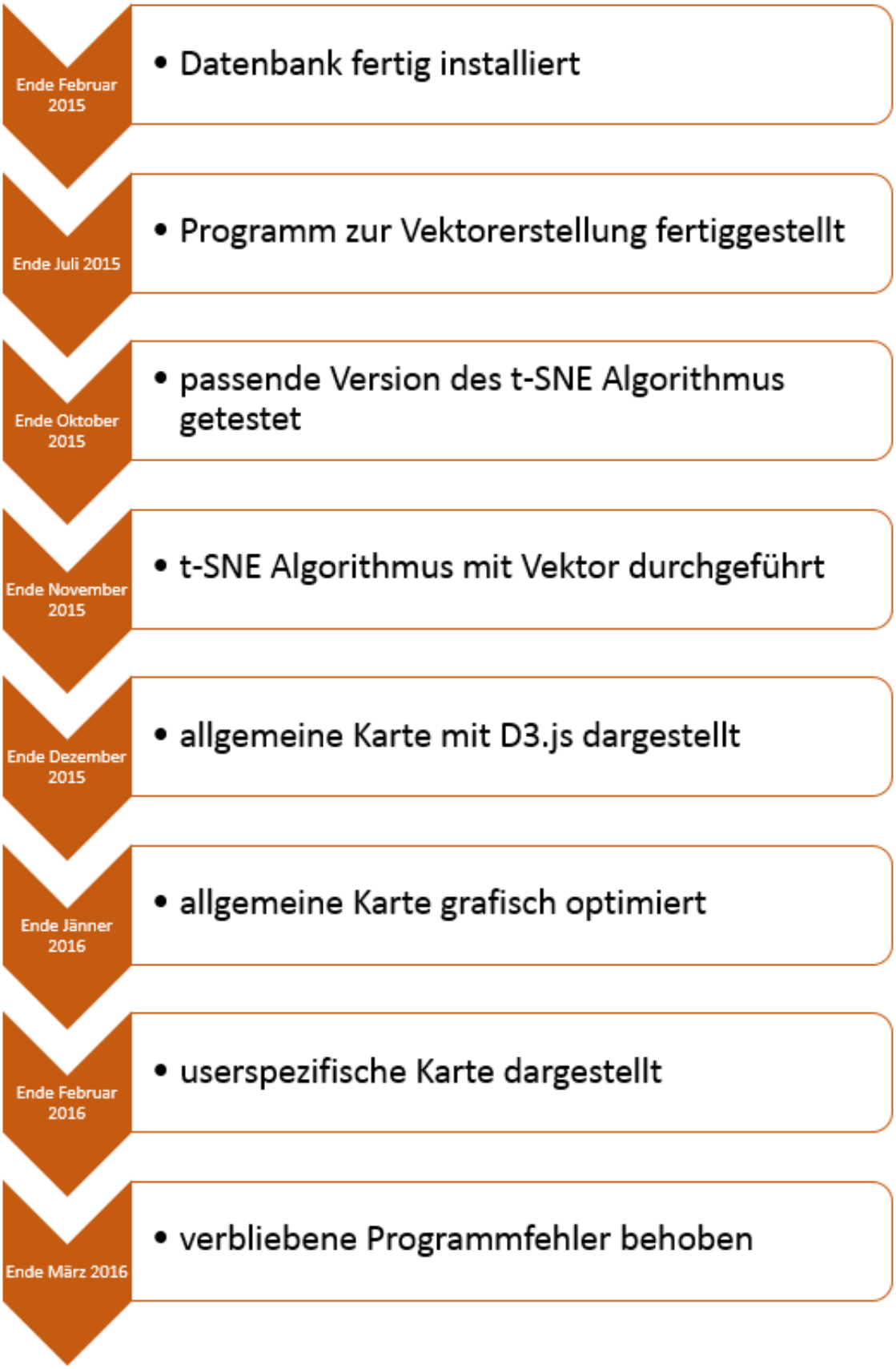

Abb. 38 Meilensteinliste

10.1.5 Datenmodell

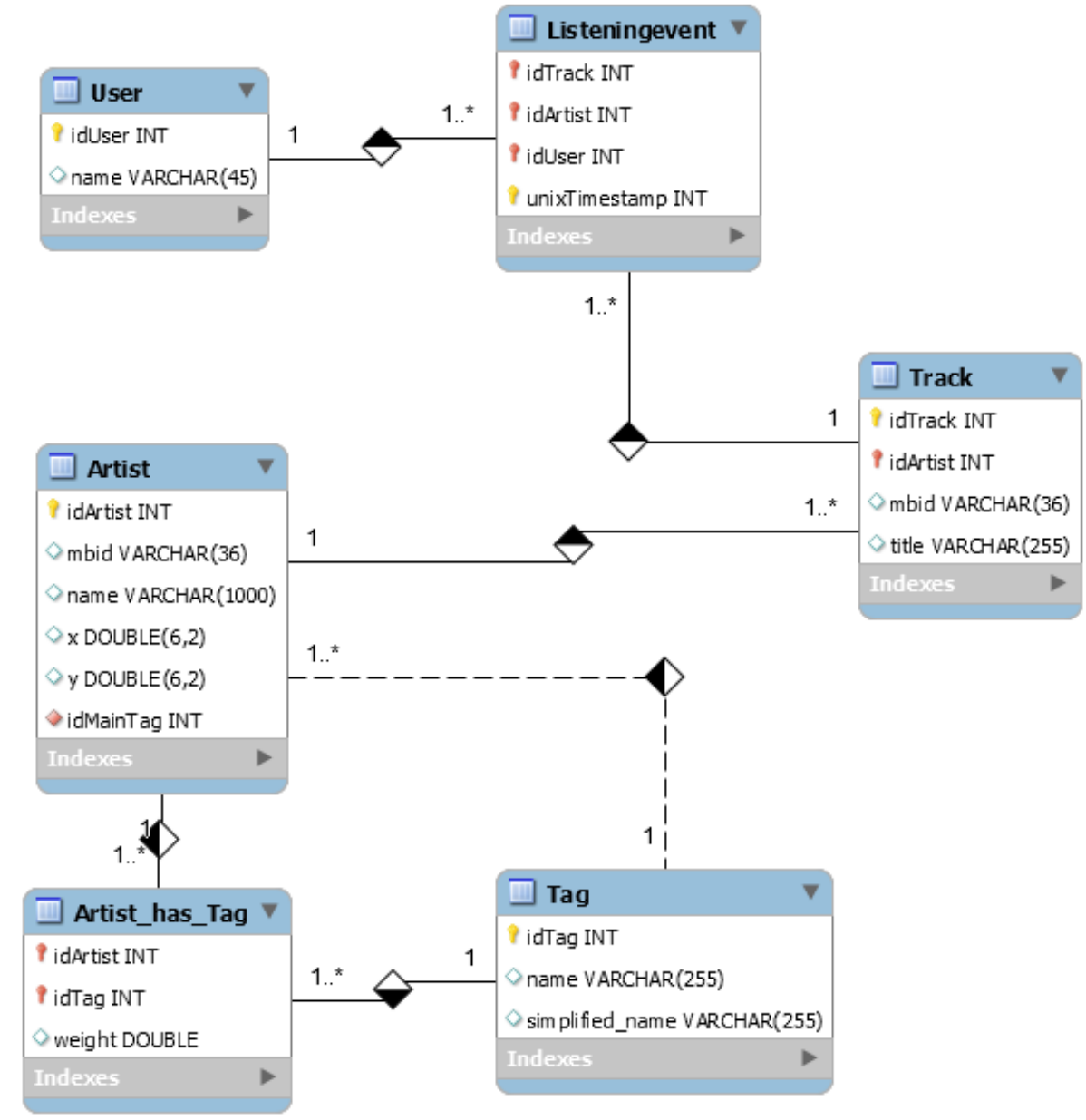

Abb. 39 Datenmodell

Die gesamte Applikation basiert auf diesem Datenmodell. Es wurde uns von der Johannes Kepler Universität zur Verfügung gestellt.

10.1.5.1 Tabelle „User“

	idUser	name
	384	Hans

Abb. 40 Tabelle User

Die Tabelle „User“ speichert jeden Benutzer des Internetradios last.fm. Der User wird durch das Attribut „idUser“ identifiziert und hat lediglich ein weiteres Attribut, seinen Namen. Dies ist der Name, nach dem bei der Erstellung der benutzerspezifischen Karte gesucht wird.

10.1.5.2 Tabelle „Tag“

	idTag	name	simplified_name
▶	28	-- electronic --	electronic

Abb. 41 Tabelle Tag

Die Tabelle „Tag“ speichert Musikgenres. Jedes Tag wird durch das Attribut „idTag“ identifiziert. Das Attribut „name“ speichert den vollständigen Tagnamen. Der „simplified_name“ stellt den vereinfachten Namen des Tags dar. Da jeder Benutzer von last.fm einem Künstler einen Tag zuweisen kann, wird der vereinfachte Name dazu benötigt, dass nicht zwei gleiche Tags unter einem anderen Namen zugeordnet werden. Gäbe es diese Vereinfachung nicht, könnte man z.B. einem Künstler die zwei Tags „electronic“ und „Electronic“ zuordnen, trotz der Tatsache, dass sie keine andere Bedeutung haben. Der vereinfachte Name wird in der Applikation in der Legende angezeigt.

10.1.5.3 Tabelle „Artist“

	idArtist	mbid	name	x	y	idMainTag
▶	1	a9044915-8be3-4c7e-b11f-9e2d2ea0a91e	Megadeth	0.42	36.42	2406

Abb. 42 Tabelle Artist

Die Tabelle „Artist“ speichert Interpreten. Jeder Interpret wird durch das Attribut „idArtist“ identifiziert. Weiters hat jeder Interpret einen Namen, der im Attribut „name“ gespeichert ist. Die „mbid“ ist eine aus 36 Zeichen bestehende, internationale Identifikationsnummer für Interpreten und Songs. Sie wird in dieser Applikation nicht verwendet. Das Attribut „idMainTag“ speichert das Tag, zu dem dieser Interpret die höchste Gewichtung und somit Zugehörigkeit hat. Die Attribute „X“ und „Y“ beschreiben die X- und Y- Koordinaten des Interpreten auf der Musikkarte. Mithilfe der Speicherung der Koordinaten ist die mehrfache Koordinatenberechnung nicht notwendig.

10.1.5.4 Tabelle „Track“

	idTrack	mbid	title	idArtist
▶	1	74f28c5a-f1e3-414e-92af-991aa1acbc1d	A Matter of Time	3

Abb. 43 Tabelle Track

Die Tabelle „Track“ speichert Songs. Jeder Song wird durch das Attribut „idTrack“ identifiziert. Jeder Song trägt ebenfalls einen Titel und speichert die Referenz auf den Interpreten, der den Song produziert hat. Die „mbid“ ist eine aus 36 Zeichen bestehende, internationale Identifikationsnummer für Interpreten und Songs. Sie wird in dieser Applikation nicht verwendet.

10.1.5.5 Tabelle „Artist_has_Tag“

	idArtist	idTag	weightSum
▶	1	2406	100

Abb. 44 Tabelle Artist_has_Tag

In der Tabelle „Artist_has_Tag“ werden den Interpreten ihre Tags zugeordnet. Die Zuordnung wird über Gewichtungen geregelt. Die Gewichtung wird im Attribut „weightSum“ gespeichert. Die maximale Gewichtung für eine Interpret – Tag Zuordnung ist 100. Der Wert 100 bedeutet, dass der Interpret diesem Tag vollkommen entspricht. Allgemein gesagt heißt das, dass das Gewicht, der Zugehörigkeit eines Interpreten zu einem Tag entspricht. Die Gewichtung ist auch später wichtiger Bestandteil des Algorithmus. Aufgrund seiner Gewichtungen und der daraus zu schließenden Zugehörigkeiten wird der Interpret in den Clustern positioniert.

10.1.5.6 Tabelle „Listeningevent“

	idTrack	idArtist	idUser	unixTimestamp
▶	135	47	384	1368675773

Abb. 45 Tabelle Listeningevent

In die Tabelle „Listeningevent“ wird ein Eintrag gemacht, wenn sich ein User einen Song von einem Interpreten anhört. Diese Schlüsselkombination wird noch um einen Zeitstempel ergänzt, der angibt, wann der Song gehört wurde.

10.2 Darstellung

Die Planung zur Darstellung der Webapplikation und der Musikkarte werden in folgendem Kapitel erklärt.

10.2.1 Seitenaufbau

Ein besonderes Augenmerk wird in dieser Diplomarbeit auf die Übersichtlichkeit und selbsterklärende Darstellung geworfen.
Der Benutzer soll sich bereits beim erstmaligen Aufrufen der Webseite sofort zurechtfinden. Um dies zu gewährleisten, besteht die Webanwendung lediglich aus einer Seite auf der alle Funktionen ausgeführt werden.

Der Seitenaufbau wurde ebenfalls in einem schlichten Design gehalten und ist in Abb. 46 zu sehen.

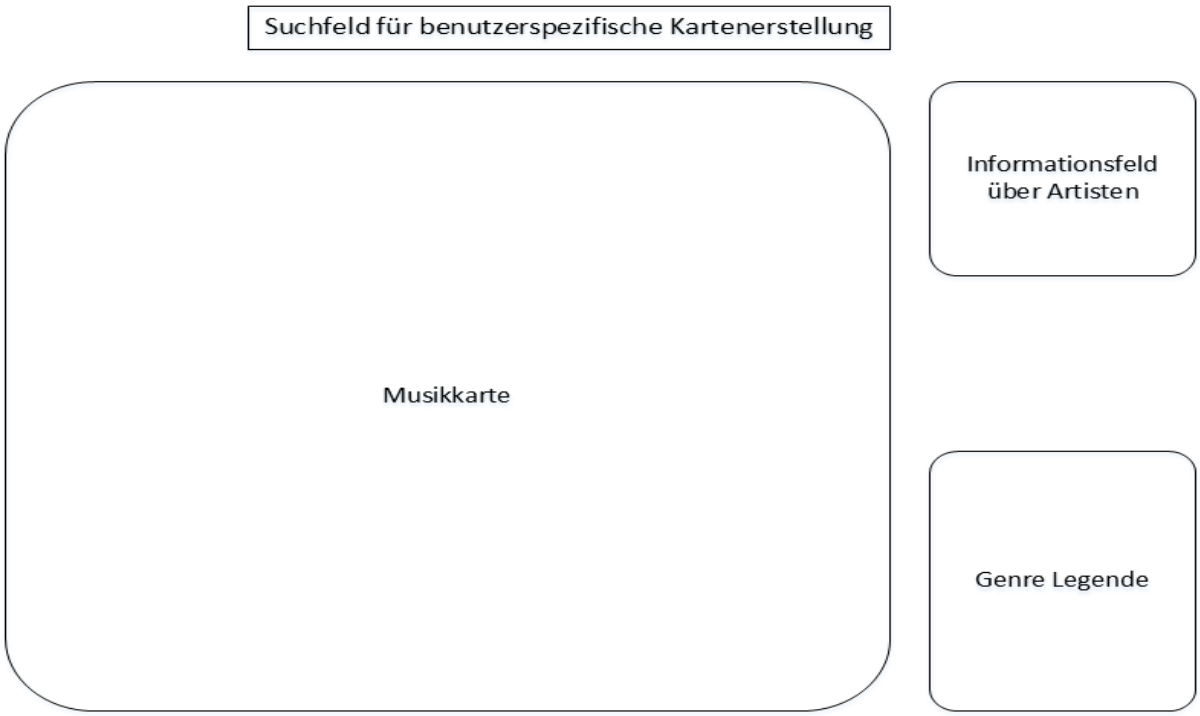

Abb. 46 Seitenaufbau

Im Zentrum der Seite befindet sich die eigentliche Musikkarte, also die Hauptanzeige, mit der vorwiegend gearbeitet wird. Der Karte wird auch die größte Fläche der Seite zugesprochen.
Oberhalb der Karte, und somit sofort bemerkbar, ist das Suchfeld für die benutzerspezifische Suche angebracht.
Zur Rechten der Karte befinden sich zwei Bereiche. Der unten platzierte stellt die Legende der Musikkarte dar. Die Legende entschlüsselt die in der Karte farbig markierten Tags mit der dazugehörigen Bezeichnung.
Der Bereich darüber wird nur angezeigt, wenn man mit der Maus auf einen, durch einen Punkt dargestellten, Artisten in der Musikkarte klickt. Daraufhin werden sämtliche Informationen über den Artisten in diesem Bereich angezeigt.

10.2.2 Seitendarstellung

Um die Anwendung für Benutzer attraktiv zu gestalten wird eine breite Palette an Farben verwendet. Währenddessen für den Hintergrund und die Informationsfelder helle Farben verwendet werden, müssen sich die Farben der einzelnen Tags in der Musikkarte eindeutig vom Hintergrund hervorheben, sich aber auch untereinander eindeutig unterscheiden, um falsche Zuordnungen zu verhindern.

Das Vergrößern der Karte soll es dem Benutzer ermöglichen, gewünschte Tags oder Bereiche der Karte genauer zu betrachten.

10.3 Herausforderungen und Risiken

Im Kapitel Risiken sind sämtliche Herausforderungen und Besonderheiten, mit denen wir uns im Zuge der Erstellung der Diplomarbeit Face the Taste of Music beschäftigen haben, aufgeführt.

10.3.1 Datenmengen

Eine Besonderheit dieser Arbeit war der Umfang der Grunddaten, aus denen die Musikkarte erstellt wird. Im Unterricht an der HTL Perg haben wir nicht mit solch einer Menge an Datensätzen gearbeitet. Demnach ist es uns nicht möglich gewesen, die Dauer der verschiedenen Berechnungen einzuschätzen.

Um den Datenumfang besser einschätzen zu können, sind hier die wichtigsten Daten zur Erstellung der Musikkarte und deren Anzahl an Eintragungen aufgeführt.

Bezeichnung	Anzahl an Eintragungen	zugehörige Tabelle in der Datenbank
Artisten	> 60 000	Artist
Tags	> 130 000	Tag
Zuordnungen	> 6 000 000	Artist_has_Tag

Tabelle 2 Datenmengen

Der Datenumfang wurde hauptsächlich bei der Vektorerstellung, siehe Kapitel 10.4.2, und der Ermittlung der Koordinaten durch den t-SNE Algorithmus, siehe Kapitel 10.4.3, zu einer wahren Herausforderung. Auch sind sämtliche Datenbankoperationen mit hohen Zeitaufwänden verbunden. Wartezeiten bis hin zu mehreren Stunden sind keine Seltenheit gewesen.

10.3.1.1 Datenfilterung

Um während der Entwicklung die Wartezeiten zu minimieren, wurde ein kleineres Dataset aus den Hauptdaten gefiltert. Dabei wurden die nicht relevanten Daten entfernt.
Aus den Grunddaten wurden alle Artisten verwendet, aber nicht alle Tags übernommen. Die Tags wurden dahingehend verringert, dass Tags, die nicht mindestens 0.07 Prozent aller Zuordnungen enthalten, ausgegliedert wurden. Damit beträgt die Anzahl der Mindestzuordnungen für ein Tag ca. 4 200 Artisten. Werden einem Tag beispielsweise lediglich 1 000 Artisten zugeordnet, so handelt es sich um ein sehr kleines Tag. Es liegt damit unter der 0.07% Grenze und wurde nicht in das Dataset übernommen.

10.3.1.2 Ressourcenauslastung

Die Filterung der nicht relevanten Daten ist hauptsächlich der Knappheit an Ressourcen zuzusprechen. Der Laptop, mit dem die Berechnungen anfänglich durchgeführt wurden, ist ein Qosmio X70-A-11Q. Dieses Model hat einen Intel Core i7 4700 MQ als Prozessor und 16 GB RAM. Die 16 Gigabyte Arbeitsspeicher sind jedoch nicht ausreichend. Deshalb sind Auslagerungsdateien erstellt worden. Eine Auslagerungsdatei ist ein Bereich auf der Festplatte, der so verwendet wird, als ob es sich um Arbeitsspeicher handelt.
Weiters wurde beim Starten der Entwicklungsumgebung RGui, der diesem Programm zur Verfügung stehende Arbeitsspeicher hinaufgesetzt, um überhaupt Ergebnisse mit geringen Datenmengen zu bekommen.

Leider haben auch diese Maßnahmen nicht genügend Ressourcen ermöglicht, um eine Musikkarte aller Artisten und Tags zu berechnen. Die größtmögliche Berechnung auf dem Laptop wurde mit 10 000 Artisten und 469 Tags durchgeführt.

Um eine größere Musikkarte zu erstellen wurden zwei Lösungsansätze herangezogen.

10.3.1.2.1 Lösungsansatz Xbox Cluster

Der Lösungsansatz Xbox Cluster sieht die Verwendung, der im Jahr 2015 abgeschlossenen Diplomarbeit „Sparky“, vor. Der Cluster wird durch einen Zusammenschluss von Spielkonsolen gebildet, die über ein Netzwerk kommunizieren und von einem Computer als Master gesteuert werden. Die Idee ist es, die Karte auf dem Cluster zu berechnen.
Dabei ist das Ziel, die Rechenzeit durch parallele Programmierung zu verringern.

Es ist nicht sicher gewesen, ob der t-SNE Algorithmus mit der parallelen Programmierung des Clusters zurechtkommen würde. Leider hat es der Zeitrahmen nicht zugelassen, diese komplexe Aufgabe auszutesten.

10.3.1.2.2 Lösungsansatz Schulserver

Der Lösungsansatz Schulserver sieht vor, den t-SNE Algorithmus auf einem Schulserver der HTL Perg auszuführen.
Da der Lösungsansatz Xbox Cluster in der verfügbaren Zeit nicht durchführbar gewesen ist, und eine Musikkarte mit den gesamten Musikdaten dennoch das Ziel der Diplomarbeit darstellt, hat die HTL Perg einen Server zur Verfügung gestellt um den t-SNE Algorithmus auszuführen.

Der benutzte Server verfügt über einen Arbeitsspeicher von 120 GB und einer Rechenleistung von 20 GHz. Trotz dieser großen Leistungswerte war es nicht möglich alle Daten zu analysieren. Die Erzeugung der größten möglichen Karte wurde mit einem Vektor bestehend aus 40 000 Künstlern und 32 Tags durchgeführt. Um ein anschauliches Bild zu erhalten wurden 500 Iterationen als Parameter für den Vorgang ausgewählt. Das Erstellen der Karte unter diesen Bedingungen dauerte selbst auf dem sehr leistungsstarken Server 88 Stunden und 45 Minuten.
Die innerhalb dieses Zeitraums aufgebrachte Arbeitsspeicher- und CPU-Nutzung sind in Abb. 47 und Abb. 48 zu sehen.

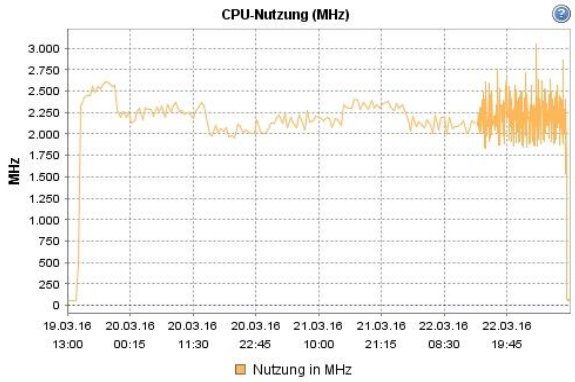

Abb. 47 CPU Auslastung

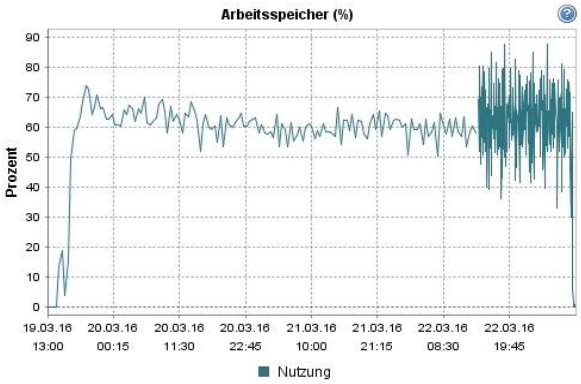

Abb. 48 Arbeitsspeicherauslastung

10.3.2 Komplexität

Unter dem Punkt Komplexität ist die Anwendung des t-SNE Algorithmus auf die bereits aufbereiteten Daten zu verstehen. Der Algorithmus steht in verschiedenen Programmiersprachen zum Downloaden zur Verfügung. Die Versionen, die im Zuge der Diplomarbeit verwendet wurden, sind im Kapitel 10.4.3 aufgeführt.
Hauptaugenmerk bei der Ausführung des Algorithmus liegt auf den korrekten Eingabeparametern. Dabei ist besonders der Parameter perplexity (siehe 10.3.2.1) zu beachten. Ein weiterer wichtiger Parameter ist die learning rate, die im Kapitel 10.3.2.2 erklärt ist. Aber auch die Iterationen des t-SNE Algorithmus haben einen großen Einfluss auf das Ergebnis (siehe 10.3.2.3).

10.3.2.1 Parameter perplexity

Laurens van der Maaten definiert die perplexity als ein Maß für die Anzahl der einflussreichen Nachbarn eines Datenpunktes. Der typische Wertebereich für die perplexity liegt zwischen den Werten fünf und 50. (vgl. [LAURENS VAN DER MAATEN, Geoffrey Hinton, 2016])

Die Anwendung des t-SNE Algorithmus auf den Musikdaten, mit dem vorgegeben Wertebereich der perplexity, hat keineswegs zu einem zufriedenstellenden Ergebnis geführt. Anstatt einer eindeutigen Clusteraufteilung der Tags, ist eine ballähnliche Figur entstanden.
Die Datenpunkte sind zentrisch um den Mittelpunkt angeordnet und die Abstände zwischen den Datenpunkten sind größer, desto weiter die Punkte vom Zentrum entfernt sind.

Abb. 49, Abb. 50 und Abb. 51 sind das Ergebnis des Java t-SNE Algorithmus (siehe 10.4.3.2) mit den Datengrößen 1 000 Artisten und 25 000 Tags.

In Abb. 49 wird der Algorithmus mit dem niedrigsten, typischen Wert, nämlich fünf durchgeführt. Eine Clusteraufteilung ist nicht erkennbar.

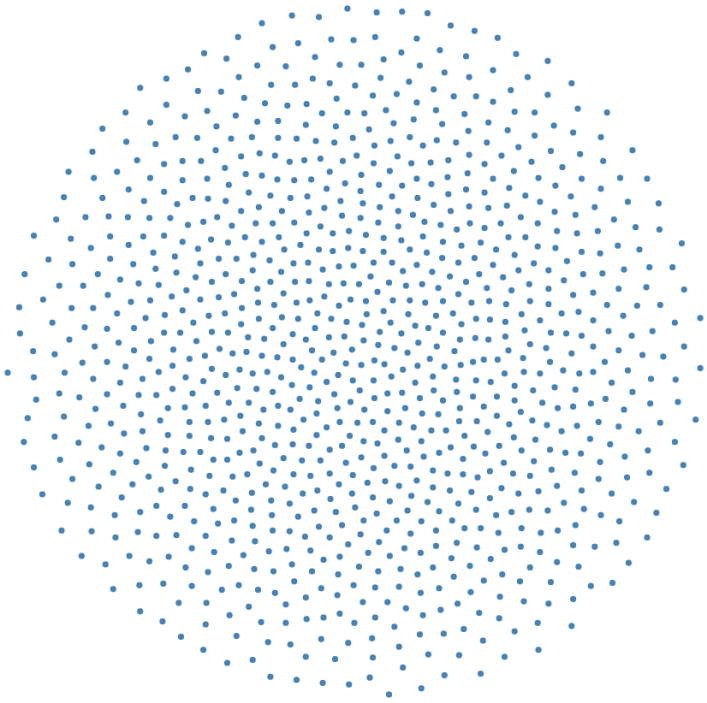

Abb. 49 t-SNE Visualisierung mit perplexity fünf

In Abb. 50 wird der Algorithmus mit dem höchsten, typischen Wert, nämlich 50 durchgeführt. Es ist weder eine Clusteraufteilung erkennbar noch gibt es merkliche Unterschiede zur Abb. 49.

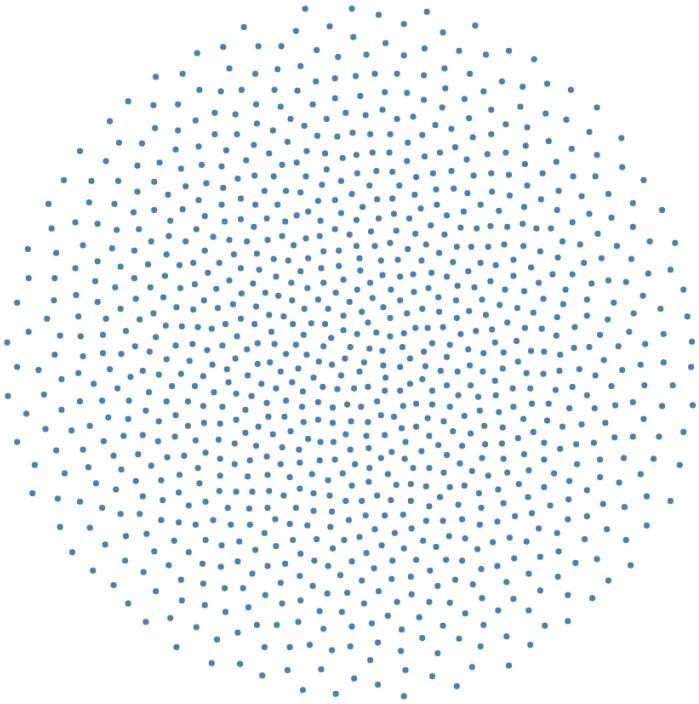

Abb. 50 t-SNE Visualisierung mit perplexity 50

In Abb. 51 ist das Ergebnis des t-SNE Algorithmus mit einer perplixity von 100 zu sehen. Die ballähnliche Form ist weiterhin deutlich zu erkennen, jedoch sind die Datenpunkte nicht mehr regelmäßig angeordnet. Ein frühes Stadium der Clusterbildung ist zu erahnen.

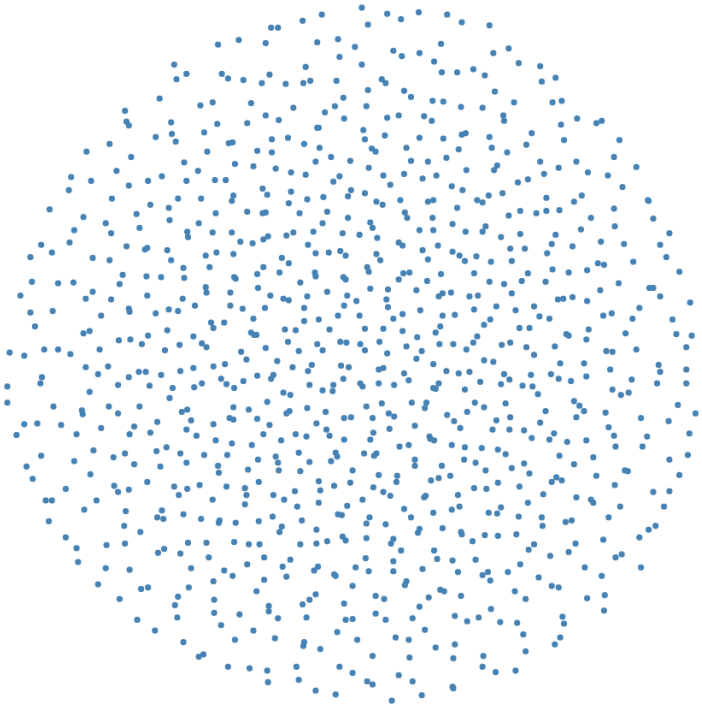

Abb. 51 t-SNE Visualisierung mit perplexity 100

Erste Clusterbereiche sind in Abb. 52, bei einer perplexity von 500, zu erkennen. Dies ist die erste Version, die nach der unter 10.3.1.1 beschriebenen Datenfilterung durchgeführt wurde. Der Wert der perplexity liegt bei diesem Beispiel weit außerhalb des vorgegebenen Wertebereichs.

Die in Abb. 52 verwendeten Datenmengen sind 2 000 Artisten und 32 Tags. Hier wurde die Matlab t-SNE Version (siehe 10.4.3.3) verwendet.

Abb. 52 t-SNE Visualisierung mit perplexity 500

Eine Korrektheitsprüfung mittels Stichproben hat ergeben, dass die Artisten ohne Zusammenhang in Cluster aufgeteilt wurden. Die Artisten eines Tags sind über die gesamte Musikkarte verteilt. Somit ist die Information, die in der Musikkarte abgebildet ist, unbrauchbar. Die Stichproben sind ermittelt worden, indem zu jedem Datenpunkt in der Karte, der in der Datenbank abgespeicherte Name ausgegeben wurde. Nun ist es aufgrund des Namens möglich, die Tags herauszusuchen.

Herr Dr. Markus Schedl hat uns empfohlen, den t-SNE Algorithmus in einer anderen Programmiersprache zu verwenden, da er ebenfalls nicht herausgefunden hat, warum der Algorithmus aufgrund der Parametereingaben kein erfolgreiches Ergebnis liefert. Die Verwendung des t-SNE Algorithmus in einer anderen Programmiersprache hat schlussendlich auch zum Erfolg geführt.

10.3.2.2 Parameter learning rate

Die learning rate ist ein kritischer Parameter. Der t-SNE Algorithmus kann beschleunigt werden, indem das adaptierte Lernraten Schema verwendet wird. Dieses Schema erhöht die learning rate, bis sie stabil ist. Die empfohlene Wertangabe liegt zwischen 100 und 1 000. Falls die Kostenfunktion des Algorithmus während der Optimierung ansteigt, könnte die learning rate zu hoch angesetzt sein. Bleibt die Funktion in einem Bereich stecken, so könnte ein zu geringer Wert für die learning rate gewählt worden sein. (vgl. [LAURENS VAN DER MAATEN, Geoffrey Hinton, 2016], [SCIKIT-LEARN, 2016])

10.3.2.3 Iterationen

Die Iterationen geben die maximale Anzahl der Durchläufe des Algorithmus an. Der Wert soll mindestens bei 200 liegen. Laurens van der Maaten empfiehlt hier den Wert 500, der auch von anderen Quellen bestätigt wird. (vgl. [SCIKIT-LEARN, 2016])

Die folgenden Abbildungen sind Zwischenergebnisse des t-SNE Algorithmus. Dabei ist die Aufspaltung der Datenstränge und die Entstehung der Cluster deutlich zu beobachten. Diese Aufnahmen wurden bereits mit der endgültigen Datenmenge erzeugt, nämlich 40 000 Artisten und 32 Tags. Genaueres zur Datenfilterung ist im Kapitel 10.3.1.1 nachzulesen.

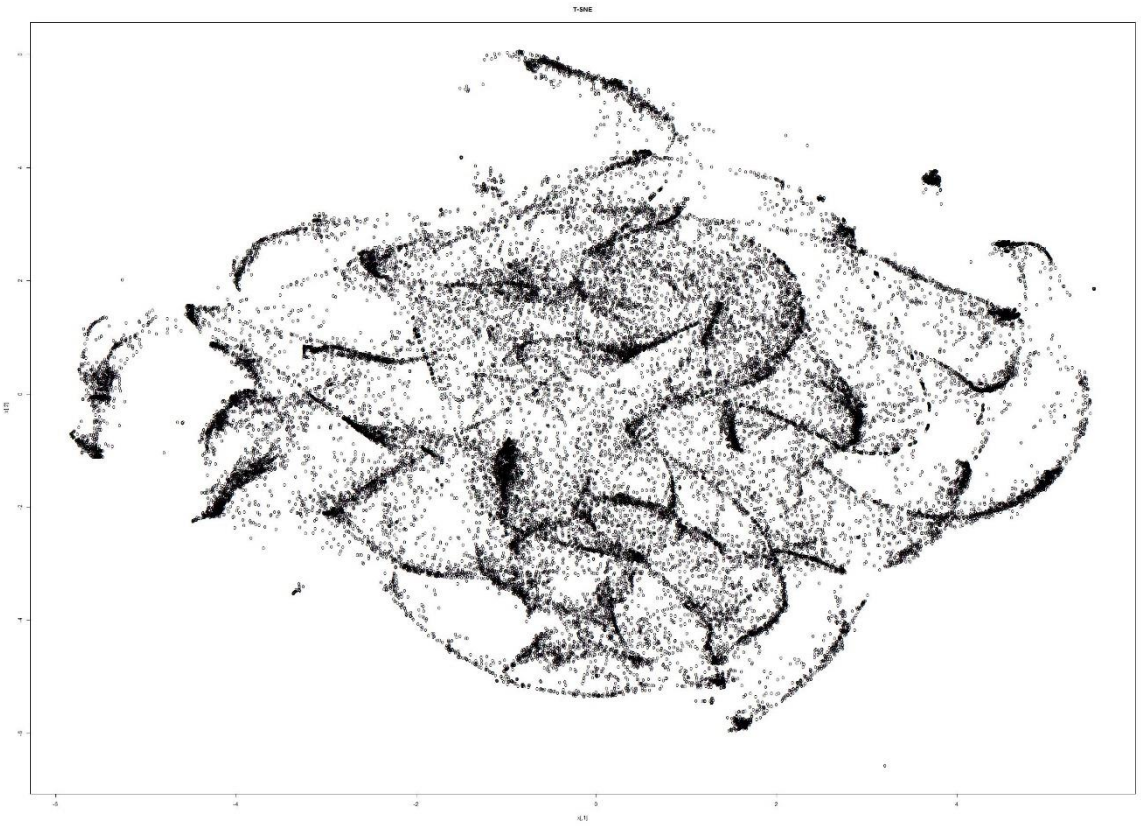

Abb. 53 t-SNE Algorithmus nach 100 Iterationen

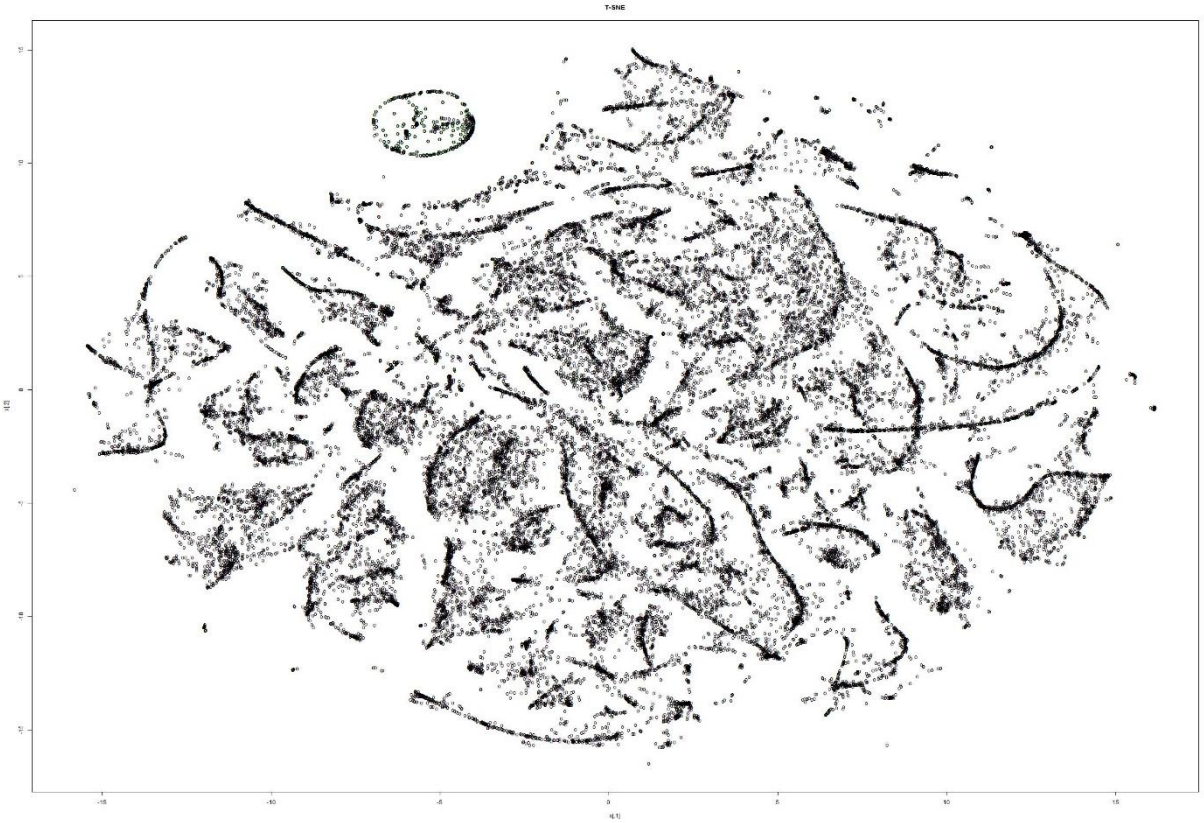

Abb. 54 t-SNE Algorithmus nach 200 Iterationen

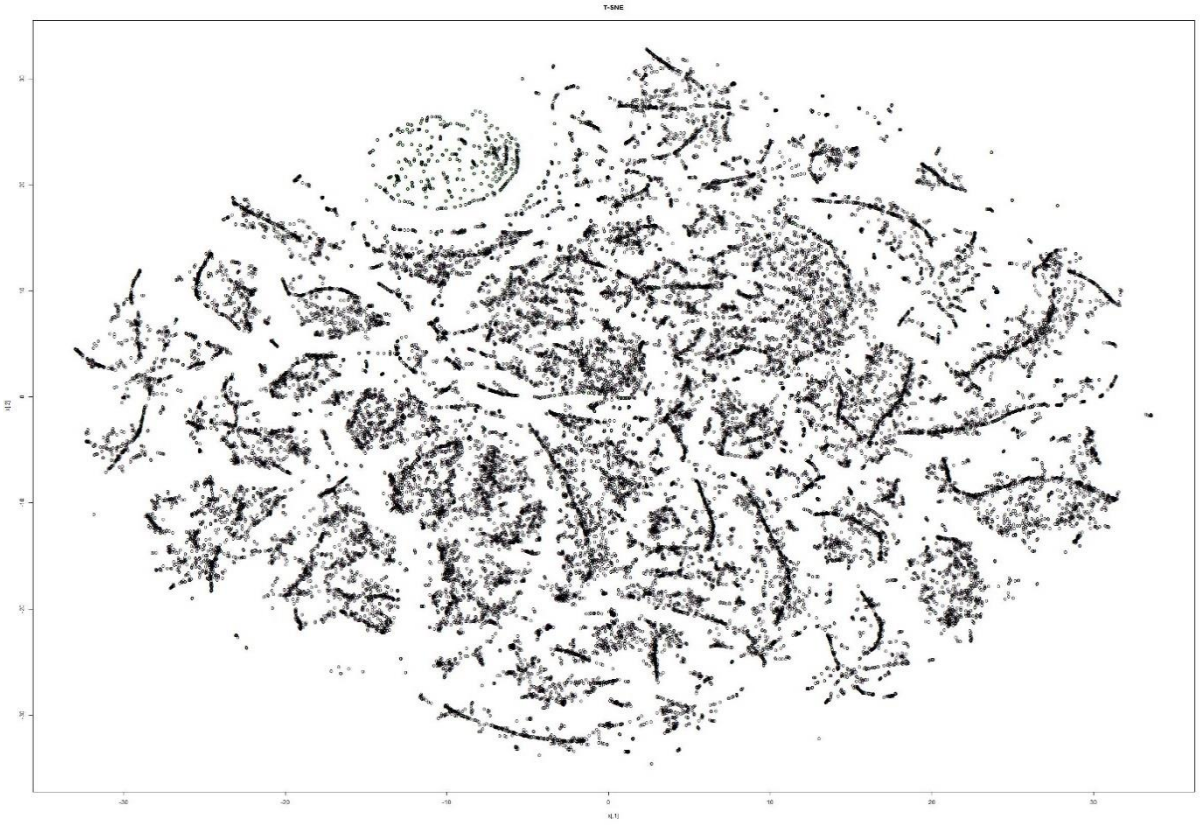

Abb. 55 t-SNE Algorithmus nach 300 Iterationen

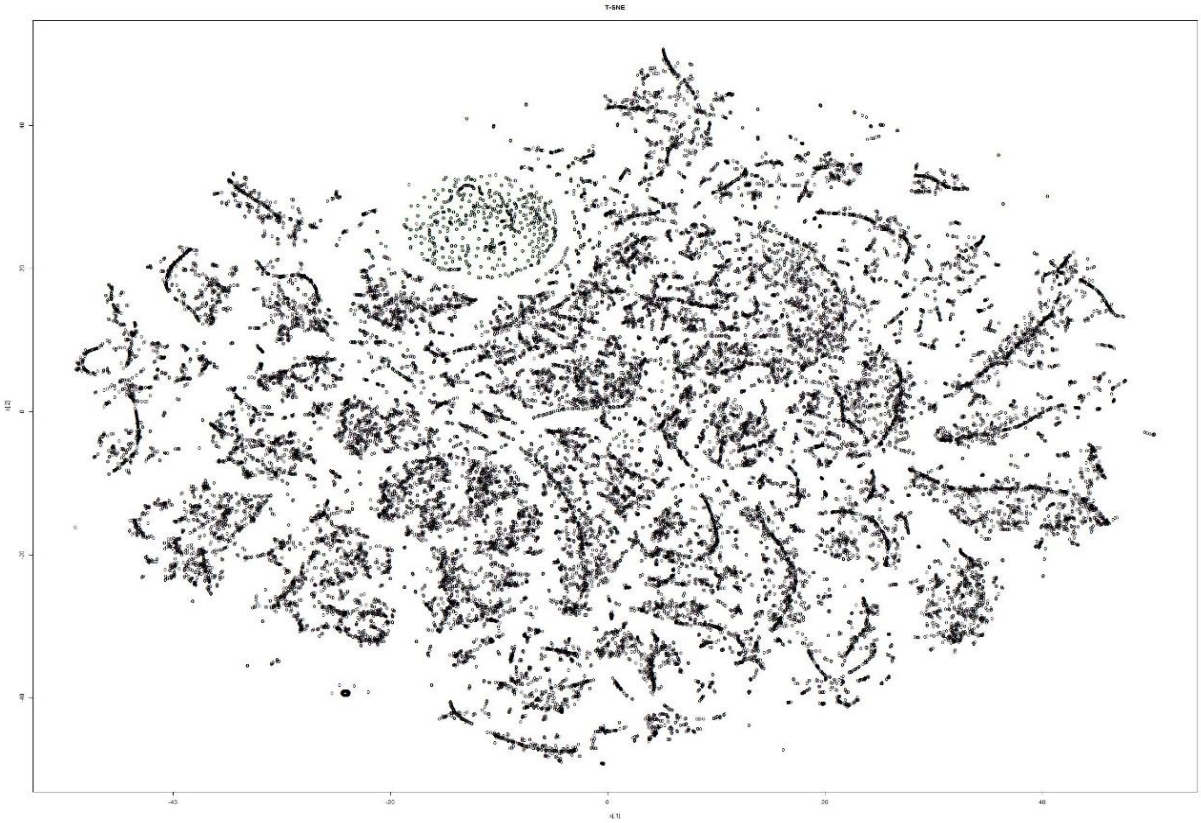

Abb. 57 t-SNE Algorithmus nach 400 Iterationen

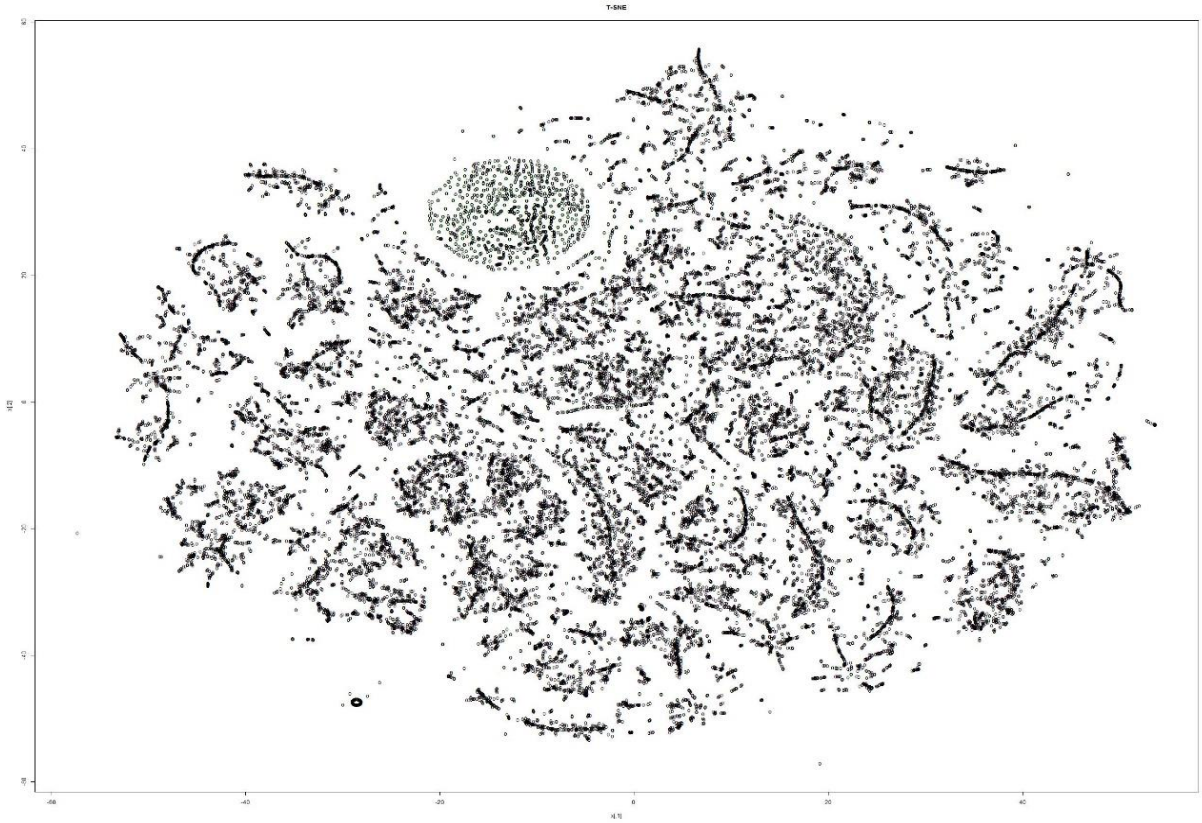

Abb. 56 t-SNE Algorithmus nach 500 Iterationen

10.4 Implementierung

Im Kapitel Implementierung wird die Programmierung der Diplomarbeit in Teilschritte zerlegt und chronologisch abgearbeitet. Ebenfalls werden Lösungsansätze beschrieben, die nicht zum Erfolg geführt haben.

10.4.1 Dimensionsreduktion

Wie bereits das Kapitel 10.3.1.1 beschreibt, wurden die Daten zum Erstellen der Karte nach einem Kriterium gefiltert.
Die Filterung der Daten verringert gleichzeitig die Anzahl der Dimensionen, da ein Tag eine Dimension darstellt.

Die Erklärung der Dimensionen wird an folgenden Beispiel durchgeführt.

```
SELECT * FROM fttom.artist_has_tag where idArtist =1 order by weightSum desc;
```

Abb. 58 Datenbankabfrage der Tabelle artist_has_tag

Ein Ausschnitt des Ergebnisses der oberhalb abgebildeten Abfrage auf die Tabelle „artist_has_tag“ unter dem Kriterium, dass die „idArtist“ „1“ ist und die Gewichtungen absteigend sortiert sind, ist in Abb. 59 abgebildet.

Jedes Tag stellt eine Dimension dar, da für jeden Artisten bestimmt werden muss, ob und wie exakt er diesem Tag zugeordnet ist.
Dank der Datenfilterung ist es möglich die Dimensionen von 130 000 auf 32 zu reduzieren.

idArtist	idTag	weightSum	weightMax
1	2406	100	100
1	99324	50	50
1	31	41	41
1	202984	30	30
1	219331	13	13
1	97018	5	5
1	32	4	4
1	169020	1	1
1	222802	1	1
1	99722	0	0
1	135802	0	0
1	136571	0	0
1	137539	0	0
1	153621	0	0
1	155469	0	0
1	135797	0	0
1	155648	0	0
1	171901	0	0
1	171903	0	0
1	171910	0	0
1	171911	0	0
1	172409	0	0
1	172443	0	0

Abb. 59 Ergebnis der Datenbankabfrage in Abb. 58

10.4.2 Vektorerstellung

Um den t-SNE Algorithmus ausführen zu können benötigt man zuerst einen sogenannten Vektor. Der Vektor beinhaltet alle Artisten mit den dazugehörigen Zuordnungen und Gewichtungen zu ihren Tags.
Der Vektor wird anschließend in einem passenden Format in eine CSV Datei hinausgeschrieben. Das hat mehrere Vorteile. Zum einen, dass der Vektor anschließend durch einen einfachen Texteditor inspiziert werden kann und eventuelle Fehler lokalisiert beziehungsweise ausgebessert werden können. Zum anderen, dass die Erstellung des Vektors einen in sich abgeschlossenen Arbeitsschritt darstellt, was äußerst hilfreich ist, wenn man bedenkt, dass dieser Vorgang ca. 24 Stunden dauert. Weiters wird das CSV Format gewählt, da die Programmiersprache R, in der der t-SNE Algorithmus ausgeführt wurde, eine Methode zum Einlesen von trennzeichengetrennten Dateien bietet.
Die Vektorerstellung wird in einem selbst entwickelten Java Programm durchgeführt.

Die Vorgehensweise dieses Programmes wird in den folgenden Kapiteln näher erläutert.

10.4.2.1 Auslesen der Daten

Als ersten Schritt werden alle Artisten und Tags aus der Datenbank ausgelesen und in einem Array zwischengespeichert.

```
ResultSet rs2 = stmt3.executeQuery();
int x =0;
while(rs2.next()){

    if(x%100==0)System.out.println(x);
    artistids[x]= rs2.getInt("artist_id");
    artistnames[x]=rs2.getString("artist_name");

    x++;
}
```

Abb. 60 Auslesen der Artisten aus der Datenbank

```
ResultSet rs = stmt.executeQuery();
int z=0;
while(rs.next()&&z<labelarr.length){
    labelarr[z] = rs.getInt("tag_simplified_id");
    labelnames[z] = rs.getString("tag_simplified_name");
    z++;
}
```

Abb. 61 Auslesen der Tags aus der Datenbank

Danach werden die Zuordnungen jedes Artisten ausgelesen und ebenfalls in einem Array zwischengespeichert. Dabei wird zu jeder Zuordnung, welche aus der Datenbank ausgelesen wird, der Wert 0.1 addiert. Das hat den Grund, dass eine Zuordnung welche in der Datenbank gespeichert ist, auch wenn sie das Gewicht 0 hat, eine stärkere Zugehörigkeit anzeigt als wenn keine Zuordnung existieren würde.

```
for(int artistcount = 0;artistcount<artistids.length;artistcount++){
  if(artistcount%10==0)System.out.println(artistcount);
  stmt2.setInt(2, artistids[artistcount]);
  rs = stmt2.executeQuery();
  while(rs.next()){
      int id = rs.getInt("tagsimplified_id");
      double weight = rs.getDouble("tagsimplified_weightSum");
      for(int i=0;i<labelarr.length;i++){
          if(labelarr[i]==id){
              arr[artistcount][i]=weight;
              if(addZeroPointOneToWeights)
                  arr[artistcount][i]+=0.1;
          }
      }
  }
```

Abb. 62 Auslesen der Zuordnungen und Gewichtungen

Zusätzlich wird der Tag mit dem höchsten Gewicht herausgesucht und auch in einem Array zwischengespeichert. Dieser Schritt ist für die Anzeige des höchst gewichtetsten Tags in der Webapplikation wichtig. Da dieser Schritt aufwendig ist und der wichtigste Tag des Künstlers in der Webapplikation angezeigt werden soll, wird dies nicht zur Laufzeit in der Webapplikation vorgenommen, sondern bereits vorher.

10.4.2.2 Ausgabe des Vektors

Um den Vektor in R einlesen zu können muss der Vektor eine bestimmte Form haben.

Beispiel eines Vektors mit drei Künstlern und neun Tags:

```
col0,col1,col2,col3,col4,col5,col6,col7,col8,col9
metal0,0.1,41.1,4.1,0.1,0.1,0.1,0.1,0.1,0.1
hiphop1,0.0,0.1,2.1,0.0,0.1,0.1,0.0,0.0,1.1
rock2,0.1,1.1,100.1,0.1,68.1,46.1,0.0,0.0,1.1
```

Abb. 63 Vektorformat für erfolgreiches Einlesen

Die erste Zeile der Datei zeigt die Anzahl der Tags an. Da die Namen der Tags nicht benötigt werden wird zur Vereinfachung die Spaltennummer hinausgeschrieben.
Die Zeilen bestehen aus dem wichtigsten Tag, zusammengesetzt mit der aktuellen Zeilennummer in der ersten Spalte, während die restlichen Spalten die Gewichtungen der Zuordnungen zu den jeweiligen Tags beinhalten.

10.4.3 t-SNE Algorithmus Versionen

Um die gewünschte und korrekte Anordnung der Datenpunkte zu erreichen wurden vier Versionen des t-SNE Algorithmus verwendet.

10.4.3.1 JavaScript Version

Da der Großteil der Webapplikation Face the Taste of Music in JavaScript verfasst ist, ist der erste Versuch gewesen, nach einer t-SNE Version zu suchen, die ebenfalls in JavaScript entwickelt ist.
Die gefundene Webseite ist eine voll einsatzfähige Version des t-SNE Algorithmus. Sie sieht vor, die zu analysierenden Daten in ein Textfeld zu kopieren. Nachdem der Button „Make t-SNE map“ gedrückt wird, wird die Karte auf Basis der kopierten Daten erstellt.

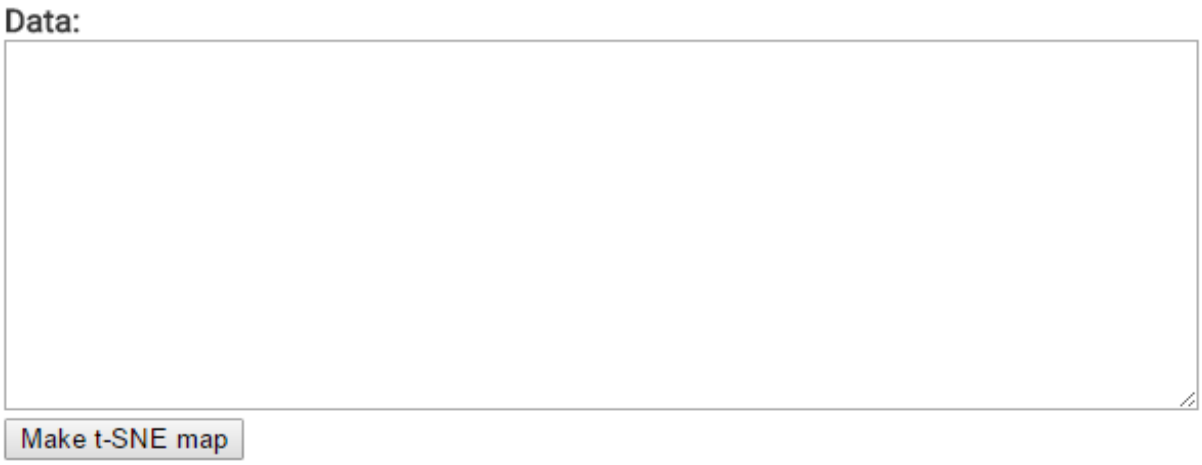

Abb. 64 t-SNE JavaScript Onlineversion [MAATEN, Laurens van der, 2016]

Diese Onlineversion von t-SNE kann jedoch mit den Datenmengen, die für Face the Taste of Music vorgesehen sind, nicht arbeiten, da es nur für kleine Datenmengen programmiert wurde.
Ebenfalls wurde ein Versuch mit der JavaScript Bibliothek tSNEJS durchgeführt. tSNEJS ist eine Implementierung des t-SNE Algorithmus in JavaScript. Die folgenden Abbildungen stellen den Ablauf des t-SNE Algorithmus in JavaScript schrittweise dar.

```
var opt = {}
opt.epsilon = 10;
opt.perplexity = 30;
opt.dim = 2;
```

Abb. 65 t-SNE JavaScript Parameter

Die ersten vier Zeilen spezifizieren die Parametereingaben *epsilon, perplexity* und *dim*, die in dem Objekt *opt* gespeichert werden.
Der Parameter *epsilon*, der die learning rate darstellt, ist im Kapitel 10.3.2.2 erklärt.
Die *perplexity* ist im Kapitel 10.3.2.1 erläutert.
Der Parameter *dim* gibt die Zieldimension an, die das Ergebnis haben soll.

```
var tsne = new tsnejs.tSNE(opt);
```

Abb. 66 t-SNE JavaScript Instanz anlegen

Es wird eine neue Instanz des Algorithmus mit den Parametereingaben erzeugt.

```
var dists = [[1.0, 0.1, 0.2], [0.1, 1.0, 0.3], [0.2, 0.1, 1.0]];
tsne.initDataDist(dists);
```

Abb. 67 t-SNE JavaScript Daten übergeben

Das Array *dists* ist in diesem Beispiel mit drei dreidimensionalen Datensätzen gefüllt. Durch die Funktion *initDataDists* werden die Datensätze dem Algorithmus zugeführt.

```
for(var k = 0; k < 500; k++) {
  tsne.step();
}
```

Abb. 68 t-SNE JavaScript Algorithmus ausführen

Der Algorithmus wird mit 500 Iterationen ausgeführt. Iterationen werden im Kapitel 10.3.2.3 erklärt.

```
var Y = tsne.getSolution();
```

Abb. 69 t-SNE JavaScript Lösung speichern

Das Ergebnis des t-SNE Algorithmus, die zweidimensionalen Koordinaten der Daten werden in einem Array *Y* gespeichert und können nun grafisch dargestellt werden.

Der Versuch, diese Version zu nutzen und die Datenmengen in 500er Schritten einzufügen, ist ebenfalls gescheitert, da der Algorithmus ab einer gewissen Datenmenge keinerlei Ergebnisse mehr geliefert hat.

10.4.3.2 Java Version

Der nächste Versuch die Karte zu erstellen wurde mit der Java-Version des t-SNE Algorithmus gestartet. Um den t-SNE Algorithmus ausführen zu können, müssen zuerst die benötigten Bibliotheken und Packages eingebunden werden.

Danach kann der bereits erstellte Vektor in ein double Array eingelesen werden und der t-SNE Algorithmus auf dieses Array angewendet werden. Dabei müssen die Dimensionen des Vektors, die *initial_dims,* und die *perplexity* angegeben werden. Das Ergebnis des Algorithmus wird in die *Y* Variable gespeichert und kann dann weiterverwendet werden.

```
int initial_dims = 469;
double perplexity = 500.0;

String filename = "C:/Users/Bene/Desktop/diplomarbeit/javasharedres/vector.txt";
double [][] X = MatrixUtils.simpleRead2DMatrix(new File(filename), ",");

TSne tsne = new SimpleTSne();

double [][] Y = tsne.tsne(X, 2, initial_dims, perplexity);
```

Abb. 70 t-SNE Algorithmus in Java

Da Java standardmäßig nur sehr geringe RAM Kapazitäten hat, müssen diese zunächst erhöht werden, um eine repräsentable Datenmenge analysieren zu können. Dazu werden in der Run Configuration des Java-Projektes die RAM Kapazitäten mittels der VM Optionen *-Xmx16g*, *-Xms16g* und *-Xss16g* auf 16 GigaByte erhöht.

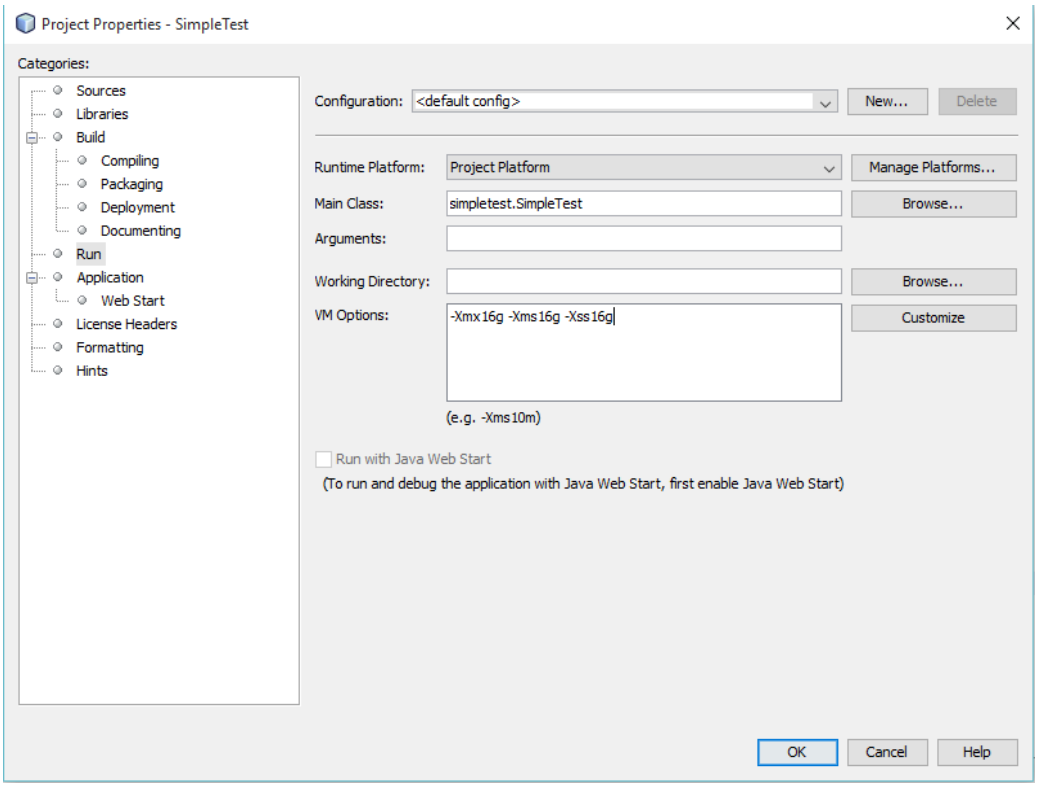

Abb. 71 Erhöhen der RAM Kapazitäten in NetBeans

Diese Version des t-SNE Algorithmus implementiert, im Gegensatz zur R-Version, siehe 10.4.3.4, keine Analyse zur Bestimmung der perplexity. Deshalb muss die perplexity manuell gesetzt werden. Das ist jedoch aufgrund unseres großen Datasets zu einem Problem geworden. Nähere Informationen dazu wurden bereits im Kapitel 10.3.2.1 angeführt.

10.4.3.3 Matlab Version

Ein weiterer Versuch die Musikkarte zu erstellen wurde mit dem Programm Matlab, siehe 9.2.6, durchgeführt.

Um den Algorithmus ausführen zu können, muss zuerst ein Ordner, mit den Matlab Dateien des t-SNE Algorithmus, als aktueller Ordner ausgewählt werden.

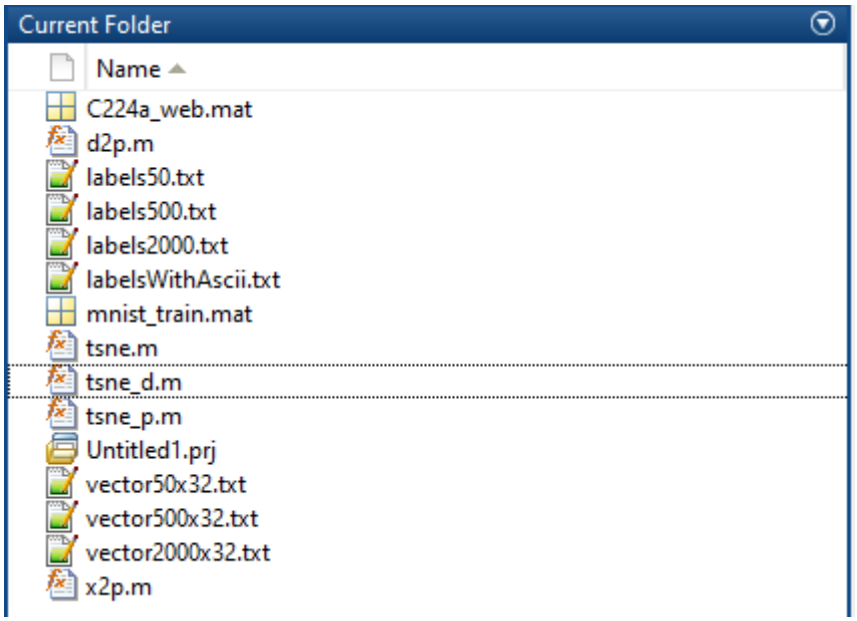

Abb. 72 Auswählen der Matlab Dateien

Danach benötigt man noch die Daten des Vektors in einem Matlab konformen Format. Das kann bequem über die *Import Data* Funktion des Programmes erreicht werden.

Nun muss man nur mehr die initialen Dimensionen und die *perplexity* für den Vektor definieren.

```
initial_dims = 32;
perplexity = 25;
```

Abb. 73 Parameterwerte setzen

Die definierten Variablen erscheinen dann im sogenannten Workspace-Fenster von Matlab.

Abb. 74 Workspace-Fenster Matlab

Nun wird der Algorithmus ausgeführt und das Ergebnis in eine Variable gespeichert um das Ergebnis weiterverwenden zu können.

```
mappedX = tsne(vector, [], no_dims, initial_dims, perplexity);
```

Abb. 75 Ausführen des t-SNE Algorithmus in Matlab

Ähnlich wie bei der Java Version, siehe 10.4.3.2, implementiert auch die Matlab Version des t-SNE Algorithmus keine Funktion, welche die perplexity anhand des Vektors bestimmen kann. Das bedeutet, dass die perplexity wieder manuell definiert werden muss, was abermals zu Problemen geführt hat. Nähere Informationen dazu sind im Kapitel 10.3.2.1 angeführt.
Außerdem war es nicht möglich eine eigene Anzahl an Iterationen zu setzen, denn diese Version sieht einen Standardwert von 2 000 Iterationen vor.

10.4.3.4 R Version

Die Version des t-SNE Algorithmus in der Programmiersprache R hat ein erfolgreiches Ergebnis geliefert und wird in der finalen Version von Face the Taste of Music verwendet.

10.4.3.4.1 Laden der benötigten Ressourcen

Zuerst wird die t-SNE Bibliothek für R geladen, welche die später benötigte Methode *tsne* zur Verfügung stellt.

```
# laden der t-SNE Bibliothek
library(tsne)
```

Abb. 76 Laden der t-SNE Bibliothek

Zusätzlich zur t-SNE Bibliothek wird noch der, in einer CSV Datei, zwischengespeicherte Vektor benötigt. Dazu wird er mit der Methode *read.table* eingelesen. Die Parameter geben hierbei den Pfad der CSV Datei, das Format der Datei und das Trennzeichen an. Mit Format ist hier gemeint, dass die erste Zeile die Spaltenüberschriften und die erste Spalte die Zeilennamen enthält.

```
# Daten in die R Entwlicklungsumgebung laden
data <- read.table("D:\\vector.csv", header=TRUE, sep=",")
```

Abb. 77 Laden des Vektors

10.4.3.4.2 Erstellen einer Callback Methode zum Kontrollieren des Zwischenfortschrittes des Algorithmus

```
x <- 0
epc <- function(x) {
    x <<- x + 1
    filename <- paste("D:\\plots\\plot", x, "jpg", sep=".")


    jpeg(filename, width=2400, height=1800)

    plot(x, t='n', main="T-SNE")
    text(x, labels=rownames(mydata))
    dev.off()
}
```

Abb. 78 Erstellung einer Grafik zur Kontrolle des Algorithmus

Um eine graphische Darstellung der Zwischenschritte, in welchen t-SNE die Anordnungen der Datenpunkte verändert, zu erhalten, wird eine Funktion *epc* definiert. Diese wird von dem im nächsten Schritt ausgeführten Befehl automatisch aufgerufen.
Um geordnete Dateinamen zu erhalten wird zuerst eine Zählervariable x definiert welche bei jedem Aufruf der Funktion erhöht wird.
Mithilfe dieser Variable und eines Pfades wird nun in der Funktion *paste* ein Dateipfad für das entstehende Bild zusammengebaut. Anschließend wird mit der Funktion *jpeg* das Bild erstellt und ein sogenanntes Device geöffnet, über den es beschrieben werden kann. Nun wird mit den Befehlen *plot* und *text* jeweils der Zeilenname an die Position des Datenpunktes gezeichnet. Abschließend wird das Device wieder geschlossen und das Bild wurde fertig erstellt. Diese Funktion dient lediglich zur Veranschaulichung und Kontrolle des Zwischenfortschrittes des Algorithmus.

10.4.3.4.3 Ausführen des t-SNE Algorithmus

```
tsne_data <- tsne(data, k=2, epoch_callback=epc, max_iter=500, epoch=100)
```

Abb. 79 Aufruf des t-SNE Algorithmus mit den benötigten Parametern

Hier wird der t-SNE Algorithmus mit den passenden Parametern ausgeführt und die Ergebnisse werden in eine Variable gespeichert.

Die Parameter sind:

- *data*: Der Vektor, der zuvor erstellt und eingelesen wurde.
- *k*: Die gewünschte Anzahl an Dimensionen welche das Ergebnis haben soll. Da die Musikkarte zweidimensional dargestellt werden soll, lautet der Parameter 2.
- *epoch_callback*: Die Funktion, welche in regelmäßigen Abständen aufgerufen werden soll.
- *max_iter*: Iteration werden im Kapitel 10.3.2.3 erklärt.

- *epoch*: In welchen Abständen die *epoch_callback* Funktion aufgerufen werden soll. Der Parameter lautet 100, so wird die Funktion fünfmal während der 500 Iterationen, die der t-SNE Algorithmus durchläuft, ausgeführt.

10.4.3.4.4 Exportieren des Ergebnisses in eine CSV Datei

```
write(tsne_data, file="D:\\Result.csv", sep = ",");
```

Abb. 80 Speicherung der tsne_data in einer CSV Datei

Das Ergebnis, das sich in der Variable *tsne_data* befindet, wird in eine CSV Datei mit dem angegebenen Trennzeichen exportiert.

10.4.4 Koordinatenaufbereitung

Da die Ausgabe eines R Arrays in eine Datei, ein für uns unpassendes Format hat, wird die von R erstellte CSV Datei umstrukturiert.
Die ursprüngliche Struktur wird in Zeilen zu je fünf Werten aufgeteilt, wobei zuerst alle x-Werte und danach alle y-Werte ausgegeben werden.

10.4.4.1 Ausgangsstruktur

In Abb. 81 ist ein Beispiel der Struktur, mit Variablen zur Veranschaulichung, angeführt.

```
x1,  x2,  x3,  x4,  x5
x6,  x7,  x8,  x9,  x10
x11, x12, x13, x14, x15
y1,  y2,  y3,  y4,  y5
y6,  y7,  y8,  y9,  y10
y11, y12, y13, y14, y15
```

Abb. 81 Struktur der CSV Datei vor der Aufbereitung

In Abb. 82 ist ein Beispiel der Struktur, mit realen Werten, angeführt.

```
0.4234801,  33.81956,   6.40158,    -36.34982,  -34.80663
-45.00055,  1.555323,   -19.55697,  -17.13683,  9.487734
4.908298,   1.455021,   -14.41506,  1.085006,   5.476553
-36.13729,  -37.24606,  13.03741,   -36.65516,  11.61482
1.903858,   7.100633,   5.594568,   -33.86157,  20.70431
12.6771,    -44.16635,  -36.78582,  -5.158426,  0.1133718
```

Abb. 82 Struktur der CSV Datei vor der Aufbereitung mit realen Werten

10.4.4.2 Struktur nach der Aufbereitung

Damit die Daten weiterverwendet werden können, wird eine Struktur benötigt, welche pro Zeile einen x und y Wert vorsieht. Deswegen ist ein Programm erstellt worden, welches die Koordinaten von R einliest und die Werte in das gewünschte Format bringt.

```
while(true){
    String line = br.readLine();
    String line2 = br.readLine();

    if(line==null||line2==null)break;

    String all = line + "," + line2;
    String[] split = all.split(",");
    for(String s : split){
        double d = Double.parseDouble(s);
        arr[arrcounter%arr.length][arrcounter/arr.length] = d;
        arrcounter++;
    }
}
```

Abb. 83 Codesegment des Strukturierungsprogramms

Das Ergebnis der Koordinatenaufbereitung wird ebenfalls in eine CSV Datei geschrieben. In der ersten Spalte befinden sich die X Koordinaten und in der zweiten Spalte befinden sich die Y Koordinaten.

```
0.4234801,   -36.13729
33.81956,    -37.24606
6.40158,     13.03741
-36.34982,   -36.65516
-34.80663,   11.61482
-45.00055,   1.903858
1.555323,    7.100633
-19.55697,   5.594568
-17.13683,   -33.86157
9.487734,    20.70431
4.908298,    12.6771
1.455021,    -44.16635
-14.41506,   -36.78582
1.085006,    -5.158426
5.476553,    0.1133718
```

Abb. 84 Ergebnis der Koordinatenaufbereitung

10.4.5 Erstellung des Java Servlets

Im folgenden Kapitel wird das Java Servlet behandelt. Das Servlet ist der Grundbaustein der Webapplikation Face the Taste of Music.

10.4.5.1 Implementierung

10.4.5.1.1 Architektur

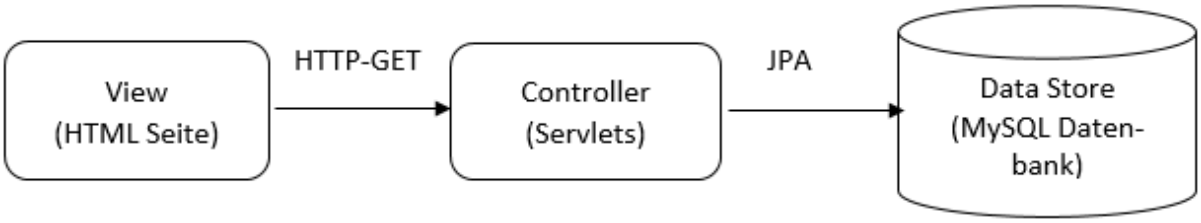

Abb. 85 Architektur des Java Servlets

Die Architektur der Web Applikation unterteilt sich in drei Bestandteile. Eine HTML Seite, die für die gesamte Darstellung verantwortlich ist. Der Data Store, welcher die gesamten Daten enthält, wurde in MySQL umgesetzt. Die Schnittstelle, welche der View die Daten des Data Stores zur Verfügung stellt wurde durch RESTful Java Servlets implementiert. Der Zugriff von der View auf den Controller erfolgt über asynchrone HTTP-GET Anfragen welche mit der Bibliothek jQuery, siehe 9.3.2, und AJAX, siehe 9.1.9, umgesetzt werden. Der Controller greift mithilfe von JPA, siehe 9.1.4, auf den Data Store zu und erhält so seine Daten.

10.4.5.1.2 View

Die View beinhaltet eine HTML Seite welche durch mehrere JavaScript Bibliotheken und ein Style Sheet erweitert wird. Da die Größe des Stylesheets überschaubar ist, wurde diese als Internal Style Sheet in die *index.html* implementiert. Die verwendeten JavaScript Bibliotheken D3.js, siehe 9.3.1, und jQuery wurden zur besseren Strukturierung in einem Ordner hinterlegt. Zusätzlich zu den Bibliotheken befindet sich in diesem Ordner noch die *main.js* Datei. Diese enthält alle Funktionen, die für die Darstellung und Datenbeschaffung benötigt werden.

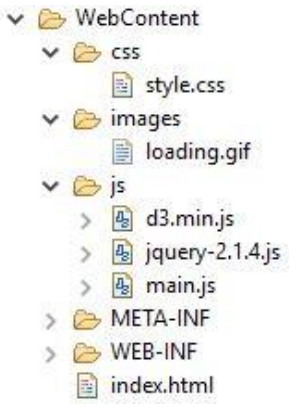

Abb. 86 Java Servlet View

10.4.5.1.3 Allgemeine Karte

Beim Aufruf der Website werden zuerst alle statischen Elemente der *index.html* geladen. Sobald diese geladen sind wird die Funktion *onLoad* aufgerufen, welche sich Daten von den Servlets besorgt und diese in statische Elemente speichert.

```
onLoad : function(){
    self = this;
    self.drawSVG();
    $.ajax({
        url:"/facethetasteofmusic/GeneralMapServlet"
    }).done(function(data){
        self.disableLoadingIndicator();
        self.data = data;
        self.drawDots();
    }).fail(function(message){
        console.log("Error ist: ", message);
        alert("Error");
    });

    $.ajax({
        url:"/facethetasteofmusic/CommonTagsServlet"
    }).done(function(data){
        self.legenddata = data;
        self.drawLegend();
    }).fail(function(message){
        console.log("Error ist: ", message);
        alert("Error");
    });
},
```

Abb. 87 JavaScript Codesegment, allgemeine Musikkarte

Zuerst wird mit der Funktion *self.drawSVG* ein Container erstellt, in dem später die einzelnen Datenpunkte gezeichnet werden können. Danach werden zwei asynchrone Anfragen an den Server gesendet. Eine für die Datenpunkte der Musikkarte und eine für die am meisten verwendeten Tags um die Legende befüllen zu können. Sobald die jeweiligen Daten geladen wurden, werden diese mit den Methoden *self.drawDots* und *self.drawLegend* in die zugehörigen statischen Elemente eingefügt. Im Fall eines Fehlers bei der Datenbeschaffung wird die Fehlermeldung auf die Konsole ausgegeben und der Benutzer wird informiert.

10.4.5.1.4 Benutzerspezifische Karte

Wenn der Benutzer in das vorgesehene Eingabefeld einen last.fm Benutzernamen eingibt und auf den Suchen-Button klickt, wird eine Anfrage an den Server gestellt, welche die Gültigkeit des Benutzernamen überprüft und sofern der Benutzername richtig ist, die benutzerspezifischen Daten zurückgibt. Diese werden dann in der Karte graphisch dargestellt.

```
onButtonClick: function(){
    self = this;
    $.ajax({
        url:"/facethetasteofmusic/SpecificMapServlet?username=" + $("#username").val()
    }).done(function(data){
        var errorfield = $("#userErrorMessage");
        if(data["userinvalid"]==0){
            errorfield.show();
        }else{
            errorfield.hide();
            self.drawDots();
            self.highlightDots(data["mapdata"]);
            self.fillLatestSongs(data["lastsongs"]);
        }

    }).fail(function(message){
        console.log("Error ist: ", message);
        alert("Error");
    });
},
```

Abb. 88 JavaScript Codesegment, benutzerspezifische Karte

Nachdem der Button gedrückt wurde, wird eine asynchrone Anfrage, mit dem eingegeben Benutzernamen als Parameter, an den Server geschickt. Das Servlet validiert den Benutzernamen und schickt bei Gültigkeit Daten zurück. Wenn ein ungültiger Benutzer eingegeben wurde, wird mit dem Befehl *errorfield.show* eine Fehlermeldung angezeigt. Ist das nicht der Fall werden die erhaltenen Daten in die Karte eingebaut. Auch hier wird im Fall eines auftretenden Fehlers die Fehlermeldung auf die Konsole geschrieben und der Benutzer wird informiert.

10.4.5.1.5 Controller

Um eine übersichtliche Unterteilung der Klassen zu erhalten wurden drei Packages erstellt.

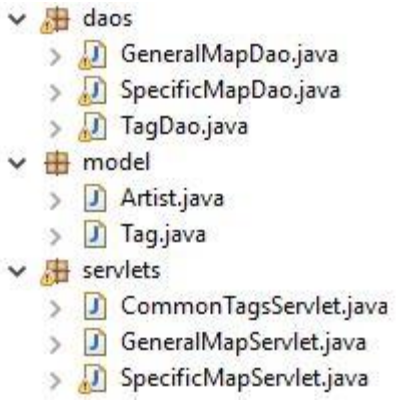

Abb. 89 Package Strukturierung

Im *daos* Package befinden sich alle Data Access Objects, siehe 9.4.1, die von den Servlets benötigt werden.
Das *model* Package enthält Klassen welche die Tabellen der Datenbank als Objekte abbilden. Dabei wurden nur diejenigen Klassen verwendet bei denen die Daten aus der Datenbank nahezu unverändert an die View weitergeleitet werden können. Bei den restlichen Tabellen ist es nicht sinnvoll gewesen, diese als Objekt abzubilden, da für die Darstellung in der View nicht direkt eine Datenzeile relevant ist, sondern die Zusammenfassung mehrerer Datensätze. Da Java nicht mengenorientiert arbeitet, wäre dieser Vorgang unter Java nur mit einem sehr hohen Ressourcenverbrauch realisierbar gewesen. Deshalb wurden diese Daten über native SQL Abfragen aus der Datenbank ausgelesen und über die entsprechenden Servlets an die View weitergeleitet. Dadurch, dass SQL mengenorientiert arbeitet, kann die Anwortzeit dramatisch verbessert werden.
Alle Servlets befinden sich im *servlets* Package und stellen Daten zur Verfügung die später über AJAX, siehe 9.1.9, angefordert werden.

Jedes Servlet beschafft sich seine Daten mit einem zugehörigen DAO. Dadurch erhält das Servlet entweder ein Array aus Objekten oder Objekte des Models zurück. Diese müssen dann mithilfe eines *JsonWriter* auf den *OutputStream* geschrieben werden.

10.4.5.1.6 Struktur eines Servlets

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("application/json");

    List<Artist> artists = dao.getArtistsWithScaledCoords();

    JsonArrayBuilder builder = Json.createArrayBuilder();
    for(int i=0;i<artists.size();i++){
        Artist a = artists.get(i);
        JsonArrayBuilder builder2 = Json.createArrayBuilder();

        String tagname = "null";
        try{
            tagname = a.getTag().getSimplifiedName();
        }catch(NullPointerException e){
            e.printStackTrace();
        }
        int tagid = 0;
        try{
            tagid = a.getTag().getIdTag();
        }catch(NullPointerException e){
            e.printStackTrace();
        }

        builder2.add(a.getIdArtist()).add(a.getX()).add(a.getY())
                .add(a.getName()).add(a.getMbid()).add(tagname).add(tagid);

        builder.add(builder2);
    }

    JsonWriter jsonWriter = Json.createWriter(response.getOutputStream());
    jsonWriter.writeArray(builder.build());
    jsonWriter.close();
}
```

Abb. 90 Servlet Strukturierung

Zuerst wird der ContentType entsprechend gesetzt. Anschließend holt sich das Servlet über sein DAO die benötigten Daten. Da wir Daten im JSON Format zurückliefern müssen, werden diese Daten mit zwei *JsonArrayBuilder* in ein zweidimensionales JSON Array umgewandelt und auf den *OutputStream* geschrieben.

10.4.6 JavaScript Funktionen der Webapplikation

Folgendes Kapitel führt alle JavaScript, siehe 9.1.8, Funktionen auf, die im Zuge der Webapplikation, verwendet werden.
Die beiden Basisfunktionen, die für die Erstellung der allgemeinen Karte und der benutzerspezifischen Karte zuständig sind, sind bereits in Kapitel 10.4.5.1.3 und Kapitel 10.4.5.1.4 erklärt. Die folgenden Funktionen sind in chronologischer Reihenfolge aufgeführt.

10.4.6.1 drawSVG

Die Funktion drawSVG definiert den Bereich, in dem die Datenpunkte entstehen und fügt die Zoomfunktion hinzu.

```
drawSVG : function(){

    this.svg = d3.select("#map")
        .append("svg")
        .attr("width", this.width)
        .attr("height", this.height)
        .append("g")
        .call(d3.behavior.zoom().scaleExtent([0.5,10]).on("zoom",self.zoomMap))
        .append("g");
```

Abb. 91 JavaScript Funktion drawSVG, Bereich definieren

Dem HTML Element *map* wird ein SVG Element angehängt. Die Funktion *append("g")* wird verwendet, um SVG Formen zu gruppieren. Falls in dem Bereich der *map* mit dem Mausrad gescrollt wird, so wird die Funktion *zoomMap* aufgerufen. Die Funktion *zoomMap* ist im Kapitel 10.4.6.2 beschrieben. Der *scaleExtent* gibt an, in welchem Bereich gezoomt werden kann. In dieser Applikation ist eine zehnfache Vergrößerung das Maximum.

```
var borderPath = this.svg.append("rect")
   .attr("rx", 12)
   .attr("ry", 12)
   .attr("x", 0)
   .attr("y", 0)
   .attr("height", this.height)
   .attr("width", this.width)
   .style("stroke", "grey")
   .style("fill", "white");
```

Abb. 92 JavaScript Funktion drawSVG, Rahmen definieren

Danach wird ein Rechteck generiert, das als Rahmen für die Musikkarte dient.

10.4.6.2 zoomMap

Die Funktion *zoomMap* wird ausgelöst, falls innerhalb der Musikkarte mit dem Mausrad gescrollt wird.

```
zoomMap : function(){
    self.svg.attr("transform","translate("+ d3.event.translate + ")scale(" + d3.event.scale + ")");
},
```

Abb. 93 JavaScript Funktion zoomMap, Musikkarte vergrößern

Dabei wird die Karte um den Grad der Vergrößerung oder Verkleinerung, *scale* transformiert.

10.4.6.3 disableLoadingIndicator

```
disableLoadingIndicator: function(){
    $("#loadingIndicator").hide();
},
```

Abb. 94 JavaScript Funktion disableLoadingIndicator, GIF ausblenden

Nachdem die Webseite aufgerufen wurde und während die Musikkarte lädt, wird eine Ladeanzeige in der Mitte des Kartenbereichs angezeigt. Die Funktion *disableLoadingIndicator* wird aufgerufen, nachdem die Karte fertig geladen ist. Daraufhin wird die Ladeanzeige mit jQuery, siehe 9.3.2, herausgesucht und mit der Funktion *hide* verborgen.

10.4.6.4 drawDots

Die Funktion *drawDots* ist verantwortlich für das Zeichnen der Datenpunkte in der Musikkarte.

```
drawDots : function(){

    this.svg.selectAll("circle").remove();
```

Abb. 95 JavaScript Funktion drawDots, Punkte entfernen

Zuerst werden alle Datenpunkte, falls vorhanden, gelöscht.

```
this.svg.selectAll("circle")
    .data(this.data)
    .enter()
    .append("circle")
    .attr("id", function(d) { return "circle" + d[0]; })
    .attr("tagid", function(d) { return d[6]; })
    .attr("cy", function(d) { return d[1]; })
    .attr("cx", function(d, i) { return d[2]; })
    .attr("r", this.radius)
    .style("fill", "steelblue")
```

Abb. 96 JavaScript Funktion drawDots, Punkte zeichnen

Danach wird für jeden Eintrag in *data* ein Kreis gezeichnet. Mittels *attr* werden dem Kreis Attribute hinzugefügt. Die Attribute, die mindestens für die Erstellung eines Kreises notwendig sind, sind die x-Koordinate *cx*, die y-Koordinate *cy* und der Radius *r*. Die weiteren Attribute werden in der Applikation verwendet, um später benötigte Daten zu diesem Datenpunkt zu speichern.

```
.on("click", function(d) {

    var html = "<table>";
    html += "<tr><td>Artistname</td><td>" + d[3] + "</td></tr>";
    html += "<tr><td>Mbid</td><td>" + d[4] + "</td></tr>";
    html += "<tr><td>last.fm link</td>" +
            "<td><a href='http://www.last.fm/music/"
            + d[3] + "'>" + d[3] + "</a></td></tr>";
    html += "<tr><td>Tag</td><td>" + d[5] + "</td></tr>";
    html += "</table>";

    var table = $(html);
    $("#artistinfo").html(table);
})
```

Abb. 97 JavaScript Funktion drawDots, auf Datenpunkt klicken

Wenn auf einen Datenpunkt geklickt wird, so werden die Attribute des Datenpunktes in das Artisteninformationsfeld eingesetzt.

```
    .on("mouseover", function(){
        $(this).css("stroke", "black");
        $(this).css("stroke-width", "2px");
    })

    .on("mouseout", function(){
        $(this).css("stroke-width", "0px");
    });

this.colorDotsInTagColors();
```

Abb. 98 JavaScript Funktion drawDots, auf Mauszeiger reagieren

Wird der Fokus des Mauszeigers auf einen Datenpunkt gesetzt, so wird der Datenpunkt mit einem schwarzen Rahmen etwas hervorgehoben.

Die Funktion *colorDotsInTagColors* ist im Kapitel 10.4.6.5 beschrieben.

10.4.6.5 colorDotsInTagColors

```
colorDotsInTagColors : function(){
    self = this;
    this.svg.selectAll("circle")
        .style("fill", function(d){ return self.calculateColor (d[5]);});
},
```

Abb. 99 JavaScript Funktion colorDotsInTagColors Punkte einfärben

Die Funktion *colorDotsInTagColors* ruft für jeden Kreis die Funktion *calculateColor*, siehe 10.4.6.6, mit dem zugehörigen Tagnamen auf.

10.4.6.6 calculateColor

```
calculateColor  : function(tagname){
    console.log(tagname);
    var val = 0;
    for(var i=0;i<tagname.length;i++){
        val+=tagname.charCodeAt(i);
    }
    var x = val%360;
    var y = val%50;
    var z = val*500/352%50;
    var a = val%2;

    if(a==1)
        return "hsl(" + x + "," + (25+y) + "%," + (25+z) + "%)";
    else
        return "hsl(" + x + "," + (75-y) + "%," + (75-z) + "%)";
},
```

Abb. 100 JavaScript Funktion calculateColor, Farben berechnen

Die Funktion *calculateColor* erstellt eine pseudozufällige Farbe für jedes Tag. Pseudozufällig deshalb, da in Wirklichkeit eine Berechnung mithilfe des Tagnamen durchgeführt wird. Im ersten Schritt der Berechnung wird die Summe der ASCII Zeichenwerte des Tagnamen gebildet. Die Berechnung der Tagfarbe wird mithilfe des HSL Systems, siehe 10.4.6.6.1, durchgeführt.

10.4.6.6.1 HSL System

Jede Farbe kann über die Eigenschaften Farbton, Sättigung und Farbhelligkeit beschrieben werden. Das HSL System baut auf diesen drei Grundmerkmalen auf. Der Farbton wird als Position auf einem Farbkreis mit Gradangaben zwischen 0° und 359° beschrieben. Die Sättigung ist verantwortlich dafür, ob eine Farbe satt und kräftig, oder schwach, erscheint. Sie wird über Prozentwerte zwischen 0% und 100% angegeben. Die Helligkeit beschreibt den Schwarz- oder Weißanteil eines Farbtons. Sie wird über Prozentwerte zwischen 0% und 100% angegeben. (vgl. [SCHULEN, Landesakademie für Fortbildung und Personalentwicklung an, 2016])

Zurück zur Funktion *calculateColor*. Für die Berechnung des Farbtons wird die Summe der ASCII Zeichen des Tagnamens, Modulo 360 gerechnet. Modulo ist eine Division, die keine Nachkommazahlen berechnet sondern den Rest speichert. Der Farbton wird in der Variable *x* gespeichert. Die Sättigung wird in der Variable *y* und die Helligkeit in der Variable *z* gespeichert. In der Variable *a* wird gespeichert, ob die Summe der ASCII Zeichen eine gerade oder ungerade Zahl ist. Falls die Summe der ASCII Zeichen gerade ist, so ist in *a* der Wert „0“ gespeichert und es wird eine andere Farbe in HSL berechnet. Dies ist notwendig, da es sonst sein kann, dass sich die Farben sehr ähnlich sind und eine eindeutige Unterscheidung nicht möglich ist. Die Werte für die Helligkeit und Sättigung werden lediglich Modulo 50 gerechnet, da Werte die kleiner als 25 und größer als 75 sind, keine ansprechenden Farbwerte enthalten.
Mithilfe dieser Funktion ist es möglich, 360 * 50 * 50 * 2 = 1 800 000 verschiedene Farben zu berechnen.

10.4.6.7 drawLegend

```
drawLegend : function(){
    self = this;
    var legend = $("#legend");
    legend.empty();
    $.each(this.legenddata, function(index, value){
        var row = $("<tr></tr>");
        row.attr("tagid", value[0]);
        var td = $("<td></td>");
        row.append(td);

        var colordiv = $("<div></div>");
        colordiv.addClass("colorField");
        colordiv.css("float", "left");

        colordiv.css( "background-color", self.calculateColor (value[1]));

        var parentdiv = $("<div></div>");
        parentdiv.css("float", "left");
        parentdiv.append(colordiv).append(value[1]);

        td.append(parentdiv);
```

Abb. 101 JavaScript Funktion drawLegend, Legende erstellen

Die Funktion *drawLegend* wird aufgerufen, nachdem die Webseite fertig geladen ist. Die vom Servlet zur Verfügung gestellten Daten werden in einer Tabelle dargestellt. Neben dem Tagnamen wird die Farbe zu diesem Tag abgebildet. Die Berechnung wird ebenfalls wieder mit der Funktion *calculateColor*, siehe 10.4.6.6, durchgeführt.

```
row.on("mouseover", function(d) {
    self.svg.selectAll("circle[tagid='" + row.attr("tagid") +"']")
         .style("stroke", "black")
         .style("stroke-width", "2px");
})
.on("mouseout", function(){
    self.svg.selectAll("circle[tagid='" + row.attr("tagid") +"']")
     .style("stroke", "black")
     .style("stroke-width", "0px");
});
legend.append(row);
```

Abb. 102 JavaScript Funktion drawLegend, auf Mauszeiger reagieren

Wird der Mauszeiger auf einem Tagnamen in der Legende platziert, so werden alle, diesem Tag zugehörigen, Interpreten in der Musikkarte hervorgeben. Dafür werden alle Kreise selektiert, die eben diese Tagid eingetragen haben.

10.4.6.8 fillLatestSongs

```
fillLatestSongs: function(data){
    self = this;
    var latestsongs = $("#latestsongs");
    latestsongs.empty();
    latestsongs.show();
    latestsongs.append($("<tr><th>Song</th><th>Artist</th></tr>"))
    $.each(data, function(index, value){
        var row = $("<tr></tr>");
        var td = $("<td>" + value[0] + "</td>");
        row.append(td);

        var td2 = $("<td><a href='http://www.last.fm/music/" + value[1] + "'>" + value[1] + "</a></td>");
        row.append(td2);

        latestsongs.append(row);
    });
},
```

Abb. 103 JavaScript Funktion fillLatestSongs, zuletzt gehörte Songs darstellen

Die Funktion *fillLatestSongs* wird ausgeführt, wenn nach einem Benutzernamen gesucht wird, der in der Datenbank existiert. Dabei werden lediglich die fünf zuletzt angehörten Songs in *data* vom Servlet übergeben. Die Daten werden mit der Funktion *$.each* iteriert. Die Songtitel und die Interpreten werden in einer Tabelle ausgegeben.

10.4.6.9 highlightDots

```
highlightDots: function(data){
    for(var i in data){
        var s = "#circle" + data[i][1];
        d3.select(s)
            .attr("r", data[i][0]*this.radius*3)
            .style("fill", "black");
    }
},
```

Abb. 104 JavaScript Funktion highlightDots, Artisten hervorheben

Die Funktion *highlightDots* wird ausgeführt, wenn nach einem Benutzernamen gesucht wird, der in der Datenbank existiert. In *data* befinden sich alle Interpreten, deren Songs der User schon einmal gehört hat, sowie die Häufigkeit der Aufrufe in Prozent. Alle Häufigkeiten addiert ergeben den Wert 100. Um auch die Datenpunkte den Häufigkeiten entsprechend darzustellen, wird der Radius mit dieser Häufigkeit multipliziert. Der Faktor drei dient dazu, die Punkte visuell besser hervorzuheben.

11 Ergebnis

Das Ergebnis der Diplomarbeit Face the Taste of Music ist eine Webapplikation, die beim Aufruf folgendes Bild bietet.

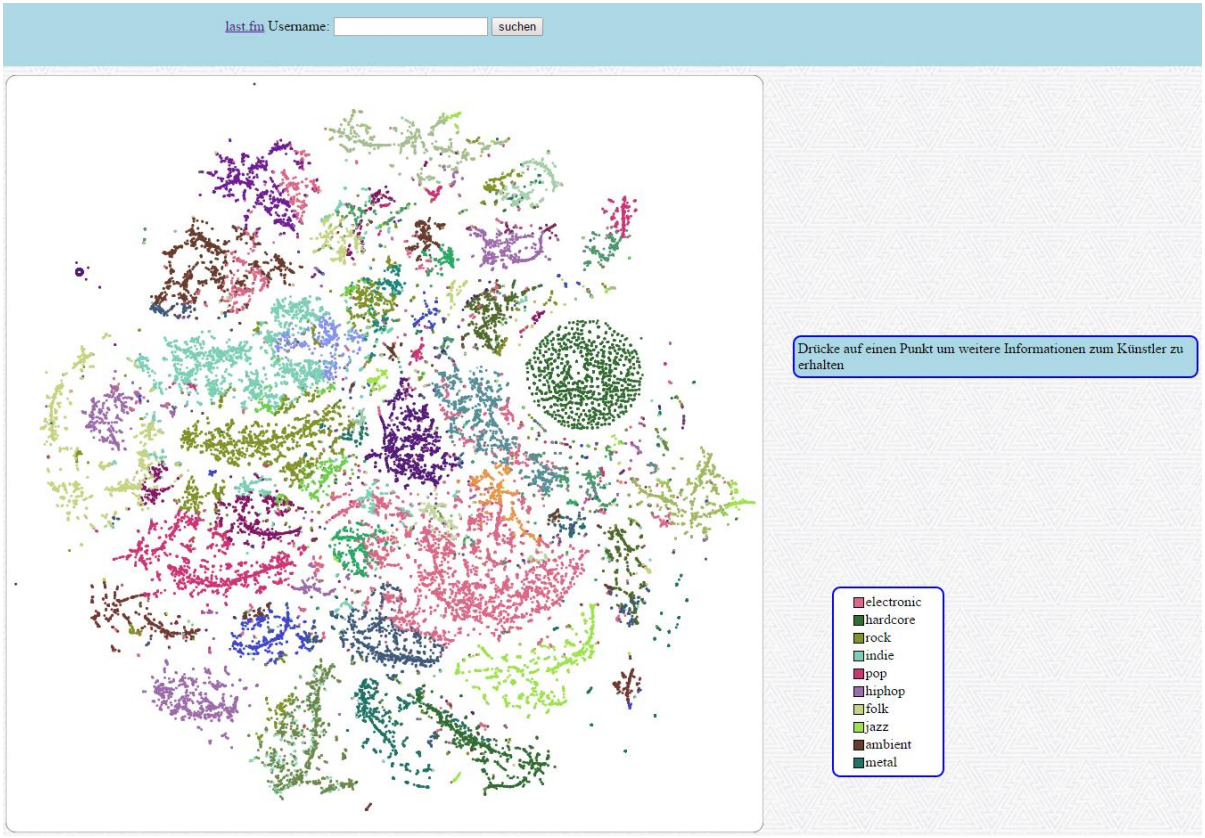

Abb. 105 Startseite Webapplikation

In der Mitte der Webseite ist die Musikkarte, das Herzstück der Applikation, abgebildet. Dies ist die Clusteraufteilung, die der t-SNE Algorithmus in über 88 Stunden Arbeitszeit gebildet hat. Die 32 Tags, die sich nach der Datenfilterung als die meist vorhandensten ausgezeichnet haben, sind in 32 verschiedenen, leicht unterscheidbaren Farben eingeteilt. Die zehn häufigsten Tags werden ebenfalls in der Legende rechts unten abgebildet.

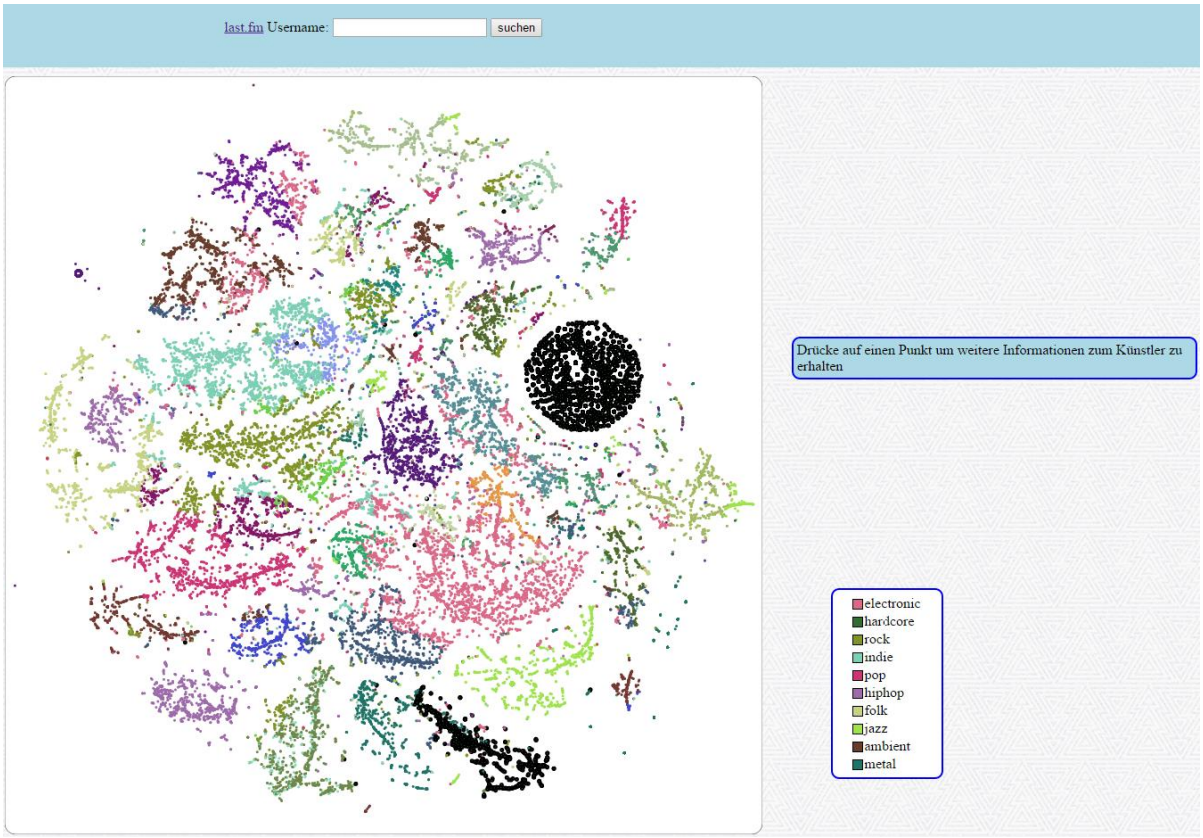

Abb. 106 Hervorheben der Artisten des Tags „hardcore“

Wird nun der Mauszeiger auf eines der Tags in der Legende gesetzt, so werden sofort die Artisten hervorgehoben, die diesem Tag entsprechen. In Abb. 106 werden alle Artisten des Tags „hardcore“ markiert.

Um nun die Artisten eines Tags genauer zu betrachten, ist es möglich in der Karte zu zoomen, wie in Abb. 107 dargestellt ist. Dafür ist es Voraussetzung, dass sich der Mauszeiger innerhalb der Musikkarte befindet. Zum Zoomen ist es lediglich nötig, das Mausrad zu drehen.

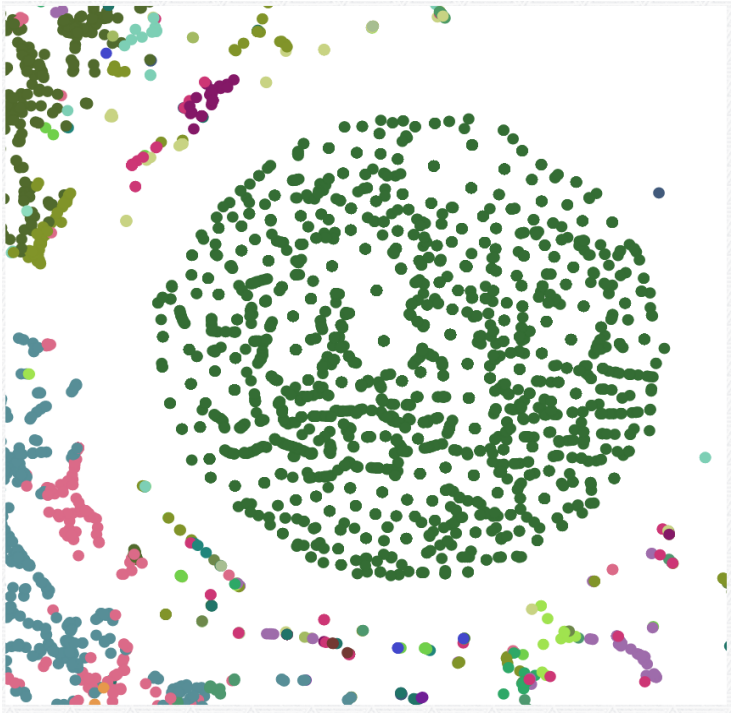

Abb. 107 Großaufnahme des Tags „hardcore"

Um nun über einen Artisten weitere Informationen zu bekommen, genügt ein Mausklick auf einen Datenpunkt. In dem Informationsfeld rechts neben der Musikkarte werden der Artistenname, die MBID, ein Link zur last.fm Seite des Künstlers und das zugehörige Tag angezeigt.

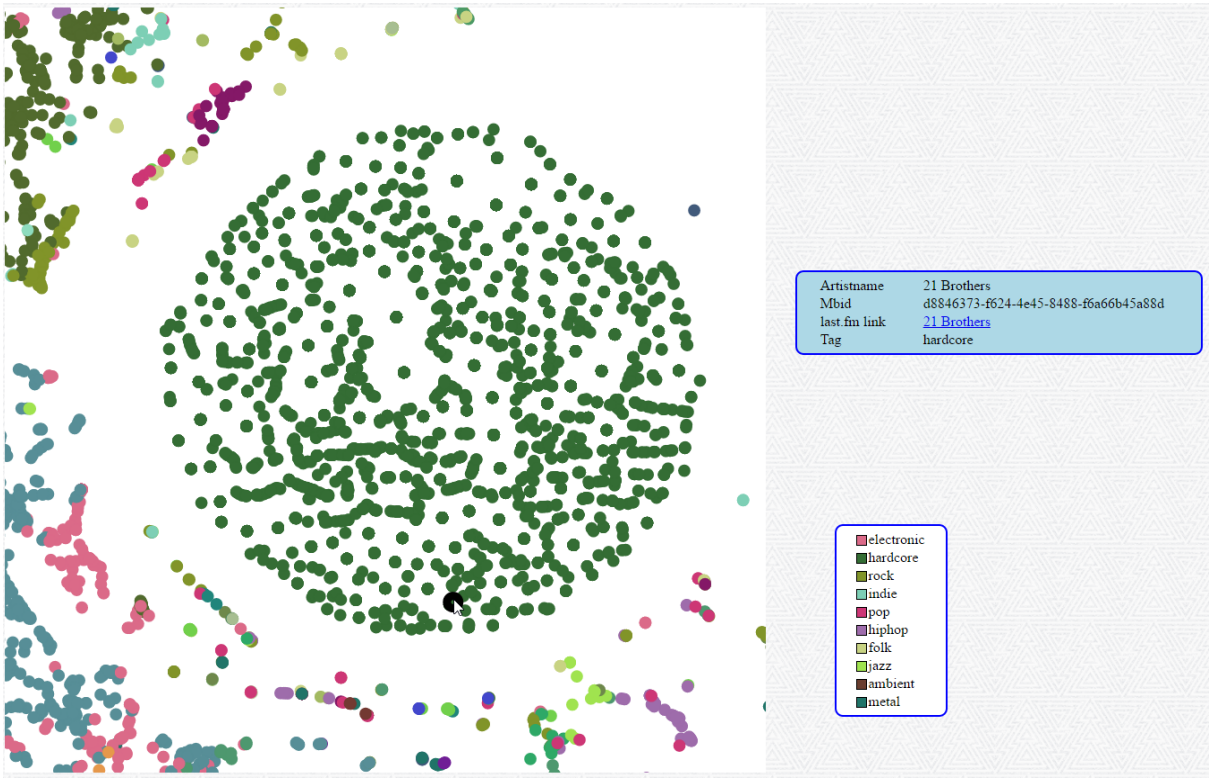

Abb. 108 Informationen über Artisten „21 Brothers“

Über der Musikkarte befindet sich ein Suchfeld. Wird in dieses Suchfeld ein vorhandener last.fm Benutzername eingegeben, so wird aus der allgemeinen Musikkarte die benutzerspezifische Musikkarte gebildet. Auf der Musikkarte werden alle Artisten hervorgehoben, die der Benutzer jemals gehört hat. Dabei spielt auch die Häufigkeit eine Rolle. Artisten, die der Benutzer oft hört, werden mehr hervorgehoben, als Artisten, die der Benutzer nur einmal gehört hat.

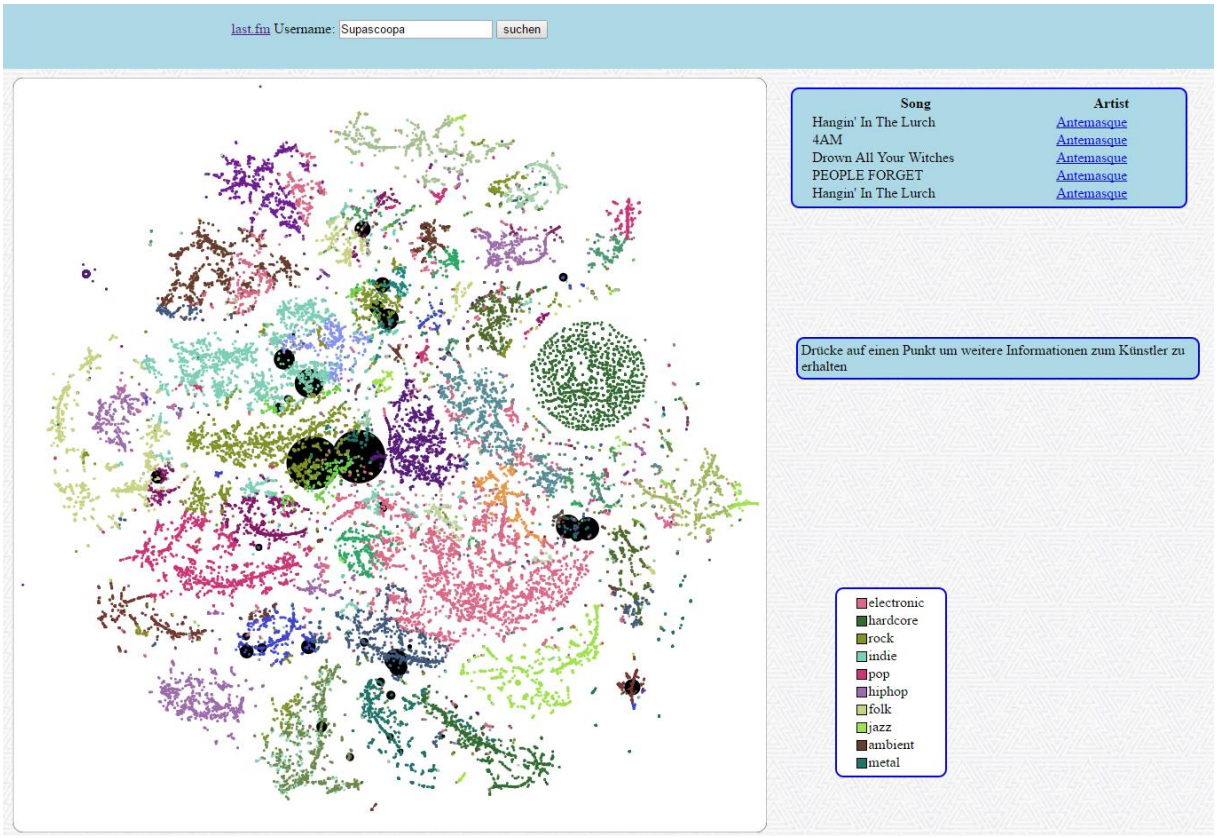

Abb. 109 Benutzerspezifische Musikkarte

12 Resümee

Durch die Diplomarbeit Face the Taste of Music ist es uns möglich gewesen, weitgehende Erfahrungen in der ausführlichen Selbsterarbeitung eines neuen Themengebietes zu sammeln. Eine wichtige Erfahrung ist, dass wir uns mit großen Datenmengen beschäftigt haben, mit denen es im Unterricht nicht möglich ist zu arbeiten, was einige spannende Erkenntnisse mit sich brachte. Beispielsweise, ist es uns bislang nie gelungen, den Arbeitsspeicher unserer Geräte auch nur annähernd auszulasten. Doch, wie wir bemerkt haben, genügt ein Algorithmus mit einigen Datensätzen, um die Geräte zum Limit zu treiben. Ohne die Unterstützung der HTL Perg wäre es nicht möglich gewesen, die Applikation mit dieser Menge an Daten zu befüllen. Die Diplomarbeit hat nicht nur aus Programmieren bestanden, sie war auch ein Schritt zum Wissenschaftlichen hin. Uns ist bewusst geworden, wie einen oft nur kleine Fehler, vom Erfolg trennen können.

Doch nicht nur der praktische Teil der Arbeit hat so manche Herausforderung dargestellt, sondern auch die Verfassung der Diplomschrift ist kein zu unterschätzender Arbeitsaufwand.

Obwohl eine Publizierung dieser Webapplikation nicht vorgesehen ist, haben wir durch diese Diplomarbeit die Möglichkeit gehabt, Erfahrungen in neuen Bereichen der Informatik zu sammeln.

13 Verteilung der Aufgabengebiete

13.1 Benedikt Alkin

- Kurzbeschreibung
- Abstract
- t-distributed Stochastic Neighbor Embedding (t-SNE)
- Datenmodell
- Komplexität
- Implementierung außer JavaScript Funktionen der Webapplikation
- Resümee
- Glossar

13.2 Simon Kroissmayr

- Kurzbeschreibung
- Abstract
- Projektumfeld
- Einleitung
- Grundlagen außer t-distributed Stochastic Neighbor Embedding (t-SNE)
- Planung außer Datenmodell
- Darstellung
- Herausforderungen und Risiken außer Komplexität
- JavaScript Funktionen der Webapplikation
- Ergebnis
- Resümee
- Glossar

14 Glossar

Algorithmus
Ein Algorithmus ist eine Handlungsvorschrift, welche eine bestimmte Eingabe in eine bestimmte Ausgabe überführt und stellt damit die Lösung eines Problems dar. (vgl. [STANGL, Lexikon, 2016])

API (Application Programming Interface)
Eine API oder auch Programmierschnittstelle bietet externen Programmen eine Schnittstelle zur Anbindung an das System. (vgl. [GRÜNDERSZENE, 2016])

Array
Ein Array ist ein Container, welcher eine fixe Anzahl an Werten eines bestimmten Datentyps enthält. Die Werte innerhalb eines Arrays werden als Elemente bezeichnet und der Zugriff erfolgt über einen numerischen Index. (vgl. [ORACLE, 2016])

ASCII (American Standard Code for Information Interchange)
ASCII ist eine standardisierte 7-Bit Zeichenkodierung welche sowohl Buchstaben als auch Zahlen und Steuerzeichen kodieren kann. (vgl. [HOPE, Computer, 2016])

classification
Unter Klassifikation versteht man die Zuordnung von Eingaben in zwei oder mehrere Klassen. Dabei müssen diese Klassen nicht generiert werden, sondern sind vorab bekannt. (vgl. [NYSRET MUSLIU, Wolfgang Slany, 2016])

Cluster
Als Cluster wird eine Gruppe von Objekten bezeichnet, die ähnliche Eigenschaften besitzen. Die Unterschiede zwischen den Clustern sollen möglichst groß sein. (vgl. [KROISSMAYR, Simon, 2015])

CPU (Central Processing Unit)
Die CPU ist das Kernstück eines Computers und führt alle Berechnungen durch. Sie ist ausschlaggebend für die Geschwindigkeit des Computers.

CSV (Comma Separated Values)
CSV bezeichnet eine Datei, welche mehrere mittels Trennzeichen getrennte Werte enthält. Als Trennzeichen wird häufig ein Semikolon verwendet.

Data Mining
Unter Data Mining versteht man das Extrahieren von Informationen aus großen Datenbeständen. Es sollen Regelmäßigkeiten und verborgene Zusammenhänge aufgedeckt werden. (vgl. [KROISSMAYR, Simon, 2015])

DOM (Document Object Model)
Eine Schnittstelle für den Zugriff auf HTML- oder XML-Dokumente wird mittels dem Document Object Model spezifiziert. (vgl. [ELLIOTTE RUSTY HAROLD, W. Scott Means, 2005])

euklidisch
Euklidisch bedeutet, dass etwas auf die Axiome von Euklid aufgebaut ist. (vgl. [DUDEN, 2016])

Framework
Ein Software Framework ist ein Entwicklungsrahmen, der die Software-Architektur eines Anwendungsprogramms definiert. (vgl. [ITWISSEN, 2012])

information retrieval
Information retrieval ist ein Fachgebiet, welches sich mit der Suchen nach komplexen Inhalten mithilfe des Computers beschäftigt. (vgl. [COMMUNITY, Wikipedia, 2016])

Interface
Ein Interface ist ein anderer Begriff für eine Schnittstelle zwischen Programmen, es ist also dasselbe wie ein API.

JDK (Java Development Kit)
Das Java Development Kit ist eines von Java Entwicklern meistgenutztes Software Development Kit. (vgl. [CORPORATION, Oracle, 2016])
Ein SDK ist eine Sammlung fertiger Programme, um ein neues Programm zu erstellen. (vgl. [GOLEM.DE, 2016])

JRE (Java Runtime Environment)
Die Laufzeitumgebung dient dazu, die geschriebenen Java Programme unabhängig vom installierten Betriebssystem auszuführen. (vgl. [PCMAGAZIN, 2016])

JSON (Java Script Object Notation)
JSON ist ein Datenformat welches zum Übertragen von Daten dient. Es wird vor allem bei Webanwendung zur Kommunikation mit einem Server benutzt.

JSP (JavaServer Pages)
JSP dient zur dynamischen Erzeugung von HTML- und XML- Ausgaben eines Webservers. [JSPTUTORIAL, 2016])

knowledge extraction
Das Erstellen von Wissen aus strukturierten und unstrukturierten Datenquellen, welches durch eine künstliche Intelligenz interpretiert werden kann, nennt man knowledge extraction. (vgl. [COMMUNITY, Wikipedia, 2016])

Künstliche Intelligenz / Computational Intelligence
Die künstliche Intelligenz bezeichnet den Versuch, die menschliche Intelligenz durch Algorithmen nachzubilden. Ein Computer soll also eigenständig Probleme bearbeiten können und aus seinen Erfahrungen lernen. (vgl. [GOLEM.DE, 2016])

music information research
Darunter versteht man das Modellieren und Entwickeln von Technologien oder Prozessen, die mit musikalischen Daten arbeiten. (vgl. [DIXON, Simon, 2012])

Pattern
Ein Pattern ist eine Regelmäßigkeit, Wiederholung, Ähnlichkeit oder Gesetzmäßigkeit in einer Datenmenge. (vgl. [COMMUNITY, Wikipedia, 2016])

pattern recognition
Darunter versteht man die Erkennung von Regelmäßigkeiten, Wiederholungen, Ähnlichkeiten oder Gesetzmäßigkeiten in einer Menge von Daten. Beispiele dafür sind Spracherkennung, Texterkennung und Gesichtserkennung. (vgl. [COMMUNITY, Wikipedia, 2016])

persistent
Eine Menge an Daten oder Objekten ist persistent, wenn sie auch nach Abbruch des Programmes noch zur Verfügung steht. (vgl. [ITWISSEN, 2016])

RAM (Random Access Memory)
Der Arbeitsspeicher eines Computers oder Servers wird als RAM bezeichnet.

social media mining
Social media mining ist das Analysieren, Extrahieren und Repräsentieren von verfolgbaren Mustern in Sozialen Medien. (vgl. [REZA ZAFARANI, Mohammad Ali Abbasi, Huan Liu, 2014])

Tag
Tag kommt aus dem Englischen und bedeutet Markierung. Auf last.fm kann man einen Artisten mit Markierungen versehen, die in diesem Fall einer Musikrichtung oder einem Musikgenre entsprechen.

URI (Uniform Resource Identifier)
Ein URI ist Zeichenfolge zur Identifizierung einer Ressource. Diese Identifier werden zur Benennung von Ressourcen, zum Aufruf von Webservices oder auch zum Identifizieren eines E-Mail Empfängers benutzt. (vgl. [COMMUNITY, Wikipedia, 2016])

Use-Case-Diagramm
Das Use-Case-Diagramm stellt Anwendungsfälle mit Akteuren und deren Beziehungen dar.

Vektor
Ein Vektor dient zur Zusammenfassung mehrerer Objekte gleichen Typs. (vgl. [HS-FULDA, 2016])

15 Anhang

15.1 Literaturverzeichnis

BAYER, Thomas. 2002. *OIO Rest Web Services*. [online]. [Accessed 8 März 2016]. Available from World Wide Web: <http://www.oio.de/public/xml/rest-webservices.htm>
BERNERS-LEE, Tim. 2005. *Uniform Resource Identifier (URI): Generic Syntax*. [online]. [Accessed 27 März 2016]. Available from World Wide Web: <https://tools.ietf.org/html/rfc3986>
BOSTOCK, Mike. 2016. *d3js*. [online]. [Accessed 8 März 2016]. Available from World Wide Web: <https://d3js.org/ >
COMBS, Jason. 2013. *Education App Fix – Dropbox*. [online]. [Accessed 8 März 2016]. Available from World Wide Web: <https://www.educationquest.org/blog/education-app-fix-dropbox/>
COMMONS, Wikimedia. 2016. *Wikimedia Commons*. [online]. [Accessed 8 März 2016]. Available from World Wide Web: <https://commons.wikimedia.org/wiki/File:Microsoft_Office_2013_logo_and_wordmark.svg >
COMMUNITY, tutorialspoint. 2016. *Data Access Object Pattern*. [online]. [Accessed 8 März 2016]. Available from World Wide Web: <http://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm>
COMMUNITY, Wikipedia. 2016. *Information Retrieval*. [online]. [Accessed 26 März 2016]. Available from World Wide Web: <https://de.wikipedia.org/wiki/Information_Retrieval>
COMMUNITY, Wikipedia. 2016. *Knowledge extraction*. [online]. [Accessed 26 März 2016]. Available from World Wide Web: <https://en.wikipedia.org/wiki/Knowledge_extraction>
COMMUNITY, Wikipedia. 2016. *Mustererkennung*. [online]. [Accessed 26 März 2016]. Available from World Wide Web: <https://de.wikipedia.org/wiki/Mustererkennung>
CORPORATION, Oracle. 2016. *Entwickeln mit dem JDK*. [online]. [Accessed 23 März 2016]. Available from World Wide Web: <https://www.java.com/de/download/faq/develop.xml>
DIFF, Visual. 2016. *Fuzzy Search*. [online]. [Accessed 4 März 2016]. Available from World Wide Web: <http://www.visualdiff.com/blog/2014/07/30/Fuzzy-Search.html>
DIXON, Simon. 2012. *Universität Wien*. [online]. [Accessed 26 März 2016]. Available from World Wide Web: <http://www.univie.ac.at/nuhag-php/dateien/talks/Dixon_2012-12_audiominer2012.pdf>
DUDEN. 2016. *Duden Rechtschreibung*. [online]. [Accessed 27 März 2016]. Available from World Wide Web: <http://www.duden.de/rechtschreibung/euklidisch>
ELLIOTTE RUSTY HAROLD, W. Scott Means. 2005. *data2type*. [online]. [Accessed 23 März 2016]. Available from World Wide Web: <https://www.data2type.de/xml-xslt-xslfo/xml/xml-in-a-nutshell/document-object-model/>
FCP. 2015. *FCP*. [online]. [Accessed 16 Dezember 2015]. Available from World Wide Web: <http://www.fcp.at/sites/default/files/images/projects/mainimages/03-010_htlperg.jpg>
FRITZL, Florian. 2015. *SQL*. Perg.
GOLEM.DE. 2016. *golem.de Künstliche Intelligenz*. [online]. [Accessed 23 März 2016]. Available from World Wide Web: <http://www.golem.de/specials/ki/>
GOLEM.DE. 2016. *golem.de Software Development Kit*. [online]. [Accessed 23 März 2016]. Available from World Wide Web: <http://www.golem.de/specials/sdk/>

GRÜNDERSZENE. 2016. *Application-Programming-Interface (API)*. [online]. [Accessed 23 März 2016]. Available from World Wide Web: <http://www.gruenderszene.de/lexikon/begriffe/application-programming-interface-api>
HENDERSON, Paul. 2016. *Sammon Mapping*. [online]. [Accessed 14 März 2016]. Available from World Wide Web: <http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0910/henderson.pdf>
HOPE, Computer. 2016. *ASCII*. [online]. [Accessed 23 März 2016]. Available from World Wide Web: <http://www.computerhope.com/jargon/a/ascii.htm>
HS-FULDA. 2016. *Zeiger und Vektoren*. [online]. [Accessed 23 März 2016]. Available from World Wide Web: <http://www2.hs-fulda.de/~klingebiel/c-vorlesung/teil7/ >
HTML-SEMINAR. *jQuery – das Framework um schmerzfrei mit JavaScript zu arbeiten*. [online]. [Accessed 7 März 2016]. Available from World Wide Web: <http://www.html-seminar.de/jquery-tutorial.htm>
IRIAN. 2016. *Einführung in JavaServer Faces*. [online]. [Accessed 5 März 2016]. Available from World Wide Web: <http://jsfatwork.irian.at/book_de/introduction.html>
ITWISSEN. 2012. *ITWissen Framework*. [online]. [Accessed 27 März 2016]. Available from World Wide Web: <http://www.itwissen.info/definition/lexikon/Framework-framework.html>
ITWISSEN. 2016. *ITWissen Persistenz*. [online]. [Accessed 27 März 2016]. Available from World Wide Web: <http://www.itwissen.info/definition/lexikon/Persistenz-persistence.html>
ITWISSEN. 2016. *Objektorientierte Programmierung*. [online]. [Accessed 3 März 2016]. Available from World Wide Web: <http://www.itwissen.info/definition/lexikon/Objektorientierte-Programmierung-OOP-object-oriented-programming.html>
JBOSS. 2016. *Visual Design for jboss.org*. [online]. [Accessed 3 März 2016]. Available from World Wide Web: <http://design.jboss.org/wildfly/>
JKU. 2015. *Computational Perception Background and Mission*. [online]. [Accessed 2015 Dezember 2015]. Available from World Wide Web: <http://www.cp.jku.at/mission.html>
JKU. 2015. *Computational Perception JKU*. [online]. [Accessed 20 Dezember 2015]. Available from World Wide Web: <http://www.cp.jku.at/people/schedl/#Current_Position>
JKU. 2015. *JKU Homepage*. [online]. [Accessed 22 Dezember 2015]. Available from World Wide Web: <http://www.jku.at/content/e213/>
JSPTUTORIAL. 2016. *jsptutorial*. [online]. [Accessed 27 März 2016]. Available from World Wide Web: <http://www.jsptutorial.org/content/introduction>
KROISSMAYR, Simon. 2015. *Data Mining*. Perg.
LAURENS VAN DER MAATEN, Geoffrey Hinton. 2016. *Visualizing Data using t-SNE*. [online]. [Accessed 13 März 2016]. Available from World Wide Web: <https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf>
LINZ, JKU. 2015. *Mechatronik Uni*. [online]. [Accessed 18 Dezember 2015]. Available from World Wide Web: <http://www.mechatronik.uni-linz.ac.at/verlinkte-bilder/Science%20Park_1184_exp_150.jpg>
MAATEN, Laurens van der. 2016. *Matlab Toolbox for Dimensionality Reduction*. [online]. [Accessed 29 März 2016]. Available from World Wide Web: <https://lvdmaaten.github.io/drtoolbox/>
MAATEN, Laurens van der. 2016. *t-SNE for CSV*. [online]. [Accessed 20 März 2016]. Available from World Wide Web: <http://homepage.tudelft.nl/19j49/tsnejs/index2.html>

MATHWORKS. 2016. *Matlab*. [online]. [Accessed 8 März 2016]. Available from World Wide Web: <http://de.mathworks.com/products/matlab/index.html?s_tid=gn_loc_drop>
MICROSOFT. 2016. *Windows Server 2012*. [online]. [Accessed 7 März 2016]. Available from World Wide Web: <https://www.microsoft.com/de-de/server-cloud/products/windows-server-2012-r2/overview.aspx>
MYSQL. 2016. *MySQL Workbench*. [online]. [Accessed 8 März 2016]. Available from World Wide Web: <https://www.mysql.de/products/workbench/>
NETBEANS. 2016. *NetBeans IDE*. [online]. [Accessed 7 März 2016]. Available from World Wide Web: <https://netbeans.org/features/index.html>
NYSRET MUSLIU, Wolfgang Slany. 2016. *Institut für Informationssysteme, TU-Wien*. [online]. [Accessed 13 März 2016]. Available from World Wide Web: <http://www.dbai.tuwien.ac.at/education/AIKonzepte/Folien/MaschinellesLernen.pdf>
ONLINE, heise. 2016. *Notepad++ 6.9*. [online]. [Accessed 7 März 2016]. Available from World Wide Web: <http://www.heise.de/download/notepad.html>
OPENCODE. 2016. *CSS*. [online]. [Accessed 4 März 2016]. Available from World Wide Web: <http://opencode.us/css/>
ORACLE. 2016. *Arrays*. [online]. [Accessed 23 März 2016]. Available from World Wide Web: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html>
ORACLE. 2016. *JavaServer Faces Technology Overview*. [online]. [Accessed 5 März 2016]. Available from World Wide Web: <http://www.oracle.com/technetwork/java/javaee/overview-140548.html>
PCMAGAZIN. 2016. *Java Runtime Environment (JRE 64-Bit)*. [online]. [Accessed 23 März 2016]. Available from World Wide Web: <http://www.pc-magazin.de/download/java-runtime-environment-jre-64-bit-1545207.html>
PERG, HTL. 2015. *Homepage HTL Perg*. [online]. [Accessed 16 Dezember 2015]. Available from World Wide Web: <https://www.htl-perg.ac.at/organisation/lehrer>
PETRIW, Zachary. 2016. *Selecting Random Row with Conditions in SQL*. [online]. [Accessed 6 März 2016]. Available from World Wide Web: <http://zpdev.net/?p=44>
PROJECT, R. 2016. *The R Project for Statistical Computing*. [online]. [Accessed 3 März 2016]. Available from World Wide Web: <https://www.r-project.org/>
RAGUDO, Marielli Gem A. 2016. *OpenIT Matlab*. [online]. [Accessed 7 März 2016]. Available from World Wide Web: <https://openit.com/tracking-matlab-license-usage-using-open-it/>
RAUBER, Andreas. 1998. *Die Selbstorganisierende Karte*. [online]. [Accessed 14 März 2016]. Available from World Wide Web: <http://www.ifs.tuwien.ac.at/ifs/research/pub_html/rau_oegai98/node2.html>
REZA ZAFARANI, Mohammad Ali Abbasi, Huan Liu. 2014. *Social media mining: An Introduction*. [online]. [Accessed 26 März 2016]. Available from World Wide Web: <http://dmml.asu.edu/smm/chapters/SMM-ch1.pdf>
SAMEER. 2014. *themespad*. [online]. [Accessed 7 März 2016]. Available from World Wide Web: <http://www.themespad.com/15-best-free-jquery-video-tutorials/>
SCHEDL, Dr. Markus. 2016. *Biographie*. Linz.
SCHULEN, Landesakademie für Fortbildung und Personalentwicklung an. 2016. *HSL-System*. [online]. [Accessed 21 März 2016]. Available from World Wide Web: <https://lehrerfortbildung-bw.de/kompetenzen/gestaltung/farbe/systeme/pc/hsl/ >
SCHWIETERING, Heinrich. 2014. *wiki.ubuntuusers*. [online]. [Accessed 2 Februar 2016]. Available from World Wide Web: <https://wiki.ubuntuusers.de/LastFM/>

SCIKIT-LEARN. 2016. *sklearn manifold TSNE*. [online]. [Accessed 16 März 2016]. Available from World Wide Web: <http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html>
SELFHTML. 2016. *CSS*. [online]. [Accessed 6 März 2016]. Available from World Wide Web: <https://wiki.selfhtml.org/wiki/CSS>
SELFHTML. 2016. *HTML*. [online]. [Accessed März 2016]. Available from World Wide Web: <https://wiki.selfhtml.org/wiki/HTML>
SELFHTML. 2016. *JavaScript*. [online]. [Accessed 6 März 2016]. Available from World Wide Web: <https://wiki.selfhtml.org/wiki/JavaScript>
SLACK. 2016. *Dropbox*. [online]. [Accessed 7 März 2016]. Available from World Wide Web: <https://slack.com/apps/A0F82E5R8-dropbox>
STANGL, Lexikon. 2016. *Lexikon Stangl Algorithmus*. [online]. [Accessed 23 März 2016]. Available from World Wide Web: <http://lexikon.stangl.eu/3027/algorithmus-algorythmus-algorhythmus/>
TECHTARGET. 2016. *What is Eclipse*. [online]. [Accessed 7 März 2016]. Available from World Wide Web: <http://searchsoa.techtarget.com/definition/Eclipse>
TENENBAUM, J.B. 2000. *A Global Geometric Framework for Nonlinear*. [online]. [Accessed 14 März 2016]. Available from World Wide Web: <http://isomap.stanford.edu/>
TUTORIALSPOINT. 2016. *JPA Tutorial*. [online]. [Accessed 4 März 2016]. Available from World Wide Web: <http://www.tutorialspoint.com/jpa/jpa_introduction.htm>
UBUNTUUSERS. 2016. *Java*. [online]. [Accessed 3 März 2016]. Available from World Wide Web: <https://wiki.ubuntuusers.de/Java/>
UNI. 2015. *UNI*. [online]. [Accessed 16 Dezember 2015]. Available from World Wide Web: <http://www.uni.at/uni/johannes-kepler-universitaet-linz-jku/>
W3SCHOOLS. 2016. *AJAX Tutorial*. [online]. [Accessed 6 März 2016]. Available from World Wide Web: <http://www.w3schools.com/ajax/>
WIKIPEDIA. 2016. *Ajax (programming)*. [online]. [Accessed 6 März 2016]. Available from World Wide Web: <https://en.wikipedia.org/wiki/Ajax_(programming)>
WIKIPEDIA. 2016. *Hypertext Markup Language*. [online]. [Accessed 4 März 2016]. Available from World Wide Web: <https://en.wikipedia.org/wiki/HTML>
WIKIPEDIA. 2016. *Java (programming language)*. [online]. [Accessed 3 März 2016]. Available from World Wide Web: <https://en.wikipedia.org/wiki/Java_(programming_language)>
WIKIPEDIA. 2016. *Logo d3*. [online]. [Accessed 7 März 2016]. Available from World Wide Web: <https://en.wikipedia.org/wiki/File:Logo_D3.svg>
WIKIPEDIA. 2016. *t-distributed stochastic neighbor embedding*. [online]. [Accessed 13 März 2016]. Available from World Wide Web: <https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding>

15.2 Abbildungsverzeichnis

Abb. 1 Musikkarte 8
Abb. 2 music map 9
Abb. 3 Projektumfeld 10
Abb. 4 Benedikt Alkin 11
Abb. 5 Simon Kroissmayr 11
Abb. 6 Betreuungslehrkraft DI Christian Aberger [PERG, HTL, 2015] 12
Abb. 7 Ausbildungseinrichtung HTL Perg [FCP, 2015] 12
Abb. 8 Logo der JKU Linz [UNI, 2015] 13
Abb. 9 Science Park JKU Linz [LINZ, JKU, 2015] 13
Abb. 10 Ansprechpartner Dr. Markus Schedl [JKU, 2015] 14
Abb. 11 Logo Internetradio last.fm [SCHWIETERING, Heinrich, 2014] 14
Abb. 12 Logo der Programmiersprache R [PROJECT, R, 2016] 19
Abb. 13 Logo der Programmiersprache Java [WIKIPEDIA, 2016] 19
Abb. 14 JPA Funktionsweise [TUTORIALSPOINT, 2016] 20
Abb. 15 Logo WildFly [JBOSS, 2016] 21
Abb. 16 Logo HTML5 [WIKIPEDIA, 2016] 21
Abb. 17 Logo CSS3 [OPENCODE, 2016] 22
Abb. 18 Logo JavaScript [DIFF, Visual, 2016] 22
Abb. 19 Logo AJAX [WIKIPEDIA, 2016] 23
Abb. 20 Logo SQL [PETRIW, Zachary, 2016] 23
Abb. 21 Entwicklungsumgebung NetBeans IDE 24
Abb. 22 Entwicklungsumgebung Eclipse IDE 25
Abb. 23 Code Editor Notepad++ 26
Abb. 24 MySQL Workbench 27
Abb. 25 Entwicklungsumgebung RGui 28
Abb. 26 Logo Matlab [RAGUDO, Marielli Gem A., 2016] 28
Abb. 27 Logo D3.js [WIKIPEDIA, 2016] 29
Abb. 28 Logo jQuery [SAMEER, 2014] 29
Abb. 29 Logo Dropbox [COMBS, Jason, 2013] 31
Abb. 30 Logo Microsoft Office 2013 [COMMONS, Wikimedia, 2016] 31
Abb. 31 Logo MS Word 2013 [COMMONS, Wikimedia, 2016] 31
Abb. 32 Logo MS Excel 2013 [COMMONS, Wikimedia, 2016] 31
Abb. 33 Logo MS PowerPoint 2013 [COMMONS, Wikimedia, 2016] 31
Abb. 34 Logo GIMP [COMMONS, Wikimedia, 2016] 32
Abb. 35 Funktionale Idee 33
Abb. 36 Use-Case-Diagramm 34
Abb. 37 Projektstrukturplan 36
Abb. 38 Meilensteinliste 37
Abb. 39 Datenmodell 38
Abb. 40 Tabelle User 39
Abb. 41 Tabelle Tag 39
Abb. 42 Tabelle Artist 39
Abb. 43 Tabelle Track 40
Abb. 44 Tabelle Artist_has_Tag 40
Abb. 45 Tabelle Listeningevent 40
Abb. 46 Seitenaufbau 41
Abb. 47 CPU Auslastung 44

Abb. 48 Arbeitsspeicherauslastung 44
Abb. 49 t-SNE Visualisierung mit perplexity fünf 45
Abb. 50 t-SNE Visualisierung mit perplexity 50 46
Abb. 51 t-SNE Visualisierung mit perplexity 100 46
Abb. 52 t-SNE Visualisierung mit perplexity 500 47
Abb. 53 t-SNE Algorithmus nach 100 Iterationen 48
Abb. 54 t-SNE Algorithmus nach 200 Iterationen 49
Abb. 55 t-SNE Algorithmus nach 300 Iterationen 49
Abb. 57 t-SNE Algorithmus nach 500 Iterationen 50
Abb. 56 t-SNE Algorithmus nach 400 Iterationen 50
Abb. 58 Datenbankabfrage der Tabelle artist_has_tag 51
Abb. 59 Ergebnis der Datenbankabfrage in Abb. 58 51
Abb. 60 Auslesen der Artisten aus der Datenbank 52
Abb. 61 Auslesen der Tags aus der Datenbank 52
Abb. 62 Auslesen der Zuordnungen und Gewichtungen 53
Abb. 63 Vektorformat für erfolgreiches Einlesen 53
Abb. 64 t-SNE JavaScript Onlineversion [MAATEN, Laurens van der, 2016] 54
Abb. 65 t-SNE JavaScript Parameter 54
Abb. 66 t-SNE JavaScript Instanz anlegen 55
Abb. 67 t-SNE JavaScript Daten übergeben 55
Abb. 68 t-SNE JavaScript Algorithmus ausführen 55
Abb. 69 t-SNE JavaScript Lösung speichern 55
Abb. 70 t-SNE Algorithmus in Java 56
Abb. 71 Erhöhen der RAM Kapazitäten in NetBeans 56
Abb. 72 Auswählen der Matlab Dateien 58
Abb. 73 Parameterwerte setzen 58
Abb. 74 Workspace-Fenster Matlab 59
Abb. 75 Ausführen des t-SNE Algorithmus in Matlab 59
Abb. 76 Laden der t-SNE Bibliothek 60
Abb. 77 Laden des Vektors 60
Abb. 78 Erstellung einer Grafik zur Kontrolle des Algorithmus 61
Abb. 79 Aufruf des t-SNE Algorithmus mit den benötigten Parametern 61
Abb. 80 Speicherung der tsne_data in einer CSV Datei 62
Abb. 81 Struktur der CSV Datei vor der Aufbereitung 62
Abb. 82 Struktur der CSV Datei vor der Aufbereitung mit realen Werten 62
Abb. 83 Codesegment des Strukturierungsprogramms 63
Abb. 84 Ergebnis der Koordinatenaufbereitung 63
Abb. 85 Architektur des Java Servlets 64
Abb. 86 Java Servlet View 64
Abb. 87 JavaScript Codesegment, allgemeine Musikkarte 65
Abb. 88 JavaScript Codesegment, benutzerspezifische Karte 66
Abb. 89 Package Strukturierung 67
Abb. 90 Servlet Strukturierung 68
Abb. 91 JavaScript Funktion drawSVG, Bereich definieren 69
Abb. 92 JavaScript Funktion drawSVG, Rahmen definieren 69
Abb. 93 JavaScript Funktion zoomMap, Musikkarte vergrößern 70
Abb. 94 JavaScript Funktion disableLoadingIndicator, GIF ausblenden 70
Abb. 95 JavaScript Funktion drawDots, Punkte entfernen 70

Abb. 96 JavaScript Funktion drawDots, Punkte zeichnen 70
Abb. 97 JavaScript Funktion drawDots, auf Datenpunkt klicken 71
Abb. 98 JavaScript Funktion drawDots, auf Mauszeiger reagieren 71
Abb. 99 JavaScript Funktion colorDotsInTagColors Punkte einfärben 71
Abb. 100 JavaScript Funktion calculateColor, Farben berechnen 72
Abb. 101 JavaScript Funktion drawLegend, Legende erstellen 73
Abb. 102 JavaScript Funktion drawLegend, auf Mauszeiger reagieren 73
Abb. 103 JavaScript Funktion fillLatestSongs, zuletzt gehörte Songs darstellen 74
Abb. 104 JavaScript Funktion highlightDots, Artisten hervorheben 74
Abb. 105 Startseite Webapplikation 75
Abb. 106 Hervorheben der Artisten des Tags „hardcore“ 76
Abb. 107 Großaufnahme des Tags „hardcore" 76
Abb. 108 Informationen über Artisten „21 Brothers“ 77
Abb. 109 Benutzerspezifische Musikkarte 78

15.3 Tabellenverzeichnis

Tabelle 1 JPA Bestandteile [TUTORIALSPOINT, 2016] 20
Tabelle 2 Datenmengen 42