technische universität
dortmund

Endbericht PG 577:

Musikempfehlungen - Automatische Erstellung von Playlisten zur Entdeckung von Musik

musery
Let's change your mood

Betreuer:
Dr. Igor Vatolkin
Dr. Geoffray Bonnin
Prof. Dr. Dietmar Jannach

Inhaltsverzeichnis

1. Vorwort

Die Projektgruppe „PG 577: Musikempfehlungen - Automatische Erstellung von Wiedergabelisten zur Entdeckung von Musik“ fand als Lehrveranstaltung am Lehrstuhl 13 der Fakultät Informatik an der Technischen Universität Dortmund während des Wintersemesters 2013/2014 und Sommersemesters 2014 statt. Die Projektgruppe wurde von Dr. Igor Vatolkin und Dr. Geoffray Bonnin betreut und von Prof. Dr. Dietmar Jannach begleitet.

1.1. Allgemeine Hinweise zur Projektgruppe

Die Prüfungsordnung der Informatik in Dortmund sieht im Master und im Hauptstudium des Diplomstudiengangs die Teilnahme an einer Projektgruppe (PG) vor. Eine PG besteht aus acht bis zwölf Studierenden, die unter Anleitung ein Jahr lang ein Projekt durchführen. Das Ziel der Projektgruppe wird von den Veranstaltern im Projektgruppenantrag festgelegt.
Die Durchführung einer Projektgruppe beginnt mit einer Seminarphase. Jeder teilnehmende Student hält einen Vortrag über ein Teilgebiet des Projektgruppenthemas, damit alle einen Einstieg in die gemeinsame Arbeit und das Themengebiet finden. Im Anschluss arbeiten die Studierenden unter der Führung der Betreuer zwei Semester selbstorganisiert daran, das Projektgruppenziel zu erreichen.

1.2. Aufbau des Endberichtes

Der vorliegende Endbericht stellt die Vorgehensweise und Arbeitsergebnisse der „Projektgruppe 577: Musikempfehlungen - Automatische Erstellung von Wiedergabelisten zur Entdeckung von Musik“ im Wintersemester 2013/2014 und Sommersemester 2014 dar.
Zunächst erfolgt im ersten Kapitel ein Vorwort, in dem allgemeine Hinweise zur Projektgruppe erläutert und die teilnehmenden Projektgruppenmitglieder vorgestellt werden. Im zweiten Kapitel erfolgt eine Einführung, in der die Thematik im wissenschaftlichen Kontext erläutert wird. Darauffolgend werden die Ziele der Projektgruppe vorgestellt. Diese beinhalten sowohl die vorgegebenen Ziele, als auch die zu erreichenden Minimalziele. Der nächste Punkt befasst sich mit der Seminarphase der Projektgruppe. Es werden die Themen vorgestellt, die in Form von Vorträgen in einer zweitägigen Projektsitzung abgehalten wurden. Im vierten Kapitel wird die Arbeitsweise aus Sicht des Projektmanagements der Projektgruppe beleuchtet. Das fünfte und sechste Kapitel beinhalten Aufbau, Administration, Datenbank und Verwaltung des Projektes. Im siebten Kapitel werden die beiden Empfehlungssysteme erläutert, die nach Tags und nach Stimmungen eine Musikempfehlung geben. Das achte Kapitel behandelt Funktionalitäten und Entwicklungsprozess der graphischen Benutzeroberfläche. Das neunte Kapitel beinhaltet die zum Projekt durchgeführte Benutzer-Studie. Abschließend erfolgt im zehnten Kapitel eine Rückschau der Projektgruppenmitglieder, sowohl über auftretene Probleme als auch über positive Erlebnisse.

2. Einführung

Die nachfolgenden Texte sind aus dem Informationsheft zu den Projektgruppen der Fakultät Informatik mit Beginn im Wintersemester 2013/2014 entnommen.

2.1. Wissenschaftlicher Kontext

Die Anzahl der digital verfügbaren Musikstücke ist in den letzten Jahren rasant angestiegen. Es wird dadurch immer schwieriger, bestehende große Musiksammlungen zu verwalten oder neue interessante Musik zu entdecken. Musikempfehlungssysteme und Musikklassifikationssysteme können helfen, diese Aufgabe automatisch zu bewältigen [5, 20]. Es gibt mittlerweile viele etablierte Ansätze für Musikempfehlungssysteme [10]. Einige der am häufigsten angewendeten Methoden für Musikempfehlungen sind die Folgenden:

- Beim KOLLABORATIVEN FILTERN wird anhand des Benutzerverhaltens (explizites Bewerten von Musikstücken, Aufnahme eines Stücks in eine Wiedergabeliste, Kaufentscheidungen) vorhergesagt, ob einem konkreten Benutzer bestimmte Musik gefallen könnte.
- Da Musikstücke meist nicht einzeln, sondern in sogenannten Wiedergabelisten (Mixes) gehört werden, kann Musikempfehlung auch durch die automatische ERSTELLUNG VON PLAYLISTEN[1] gelöst werden [3].
- Wenn Benutzer selbst die Musikstücke auswählen, die ihnen gefallen, können die neuen Lieder mit Hilfe von KLASSIFIKATIONSMETHODEN kategorisiert werden. Die Musikstücke werden hierzu durch die automatisch extrahierbaren Charakteristika (Merkmale, Features) repräsentiert. Empfehlungen für weitere Musikstücke können danach auf Basis der Ähnlichkeit ihrer Merkmale (z. B. gleiches Genre, gleicher Künstler) durchgeführt werden, was einer inhaltsbasierten Empfehlungsstrategie entspricht.

Alle Methoden unterliegen Einschränkungen, die nach dem heutigen Stand der Forschung noch nicht vollständig gelöst sind:

- HOHE RESSOURCENANFORDERUNGEN FÜR DIE DATENEXTRAKTION: Dieses Problem betrifft die oben erwähnten Methoden in unterschiedlicher Art und Weise. Beim kollaborativen Filtern bedarf es expliziter oder impliziter Nutzerbewertungen, wobei bei weniger populären Liedern zu wenig Statistiken für zuverlässige Empfehlungen vorliegen. Man muss daher auf inhaltsbasierte Ansätze zurückgreifen, was jedoch üblicherweise eine automatische Merkmalsextraktion und -klassifikation erfordert. Basiert der Prozess auf Audiosignalmerkmalen, kann dies erheblichen Rechenaufwand mit sich bringen. Nutzt man hingegen partiturbasierte Merkmale, so können diese deutlich effizienter berechnet werden, wobei bei

[1] Im restlichen Teil dieses Dokuments wird größtenteils der Begriff "Wiedergabeliste" verwendet.

populärer Musik die Noten leider oft nicht vorhanden sind. Weiterhin bestehen digitale Musiksammlungen heute oft aus sehr großen Mengen von Musikstücken, die analysiert werden müssen. Gleichzeitig wird Musik heutzutage häufig auf mobilen Geräten gehört, wo die Hardware-Ressourcen eingeschränkt sind (Bildschirmgröße, Speicherplatz) [2].

- FRAGEN DER BENUTZERINTERAKTION: In jedem Empfehlungssystem ist die Interaktion mit dem Musikhörer unabdingbar, zumindest um die initiale Auswahl der Musikstücke, die einem Benutzer gefallen, zu erfassen. Für ein erfolgreiches Empfehlungssystem ist aber eine weitere intensive Miteinbeziehung der Nutzer oft vorteilhaft, z. B. in der Form von Feedback zu den vorgeschlagenen Empfehlungen. Dies benötigt jedoch Willen und Zeit des Nutzers solche Informationen aktiv bereitzustellen. Methoden der automatischen Klassifikation können helfen, die notwendigen Erfassungsaufwände für den Benutzer zu reduzieren, indem z. B. Musik vor der Empfehlung mit Hilfe von Clustering oder selbstorganisierenden Karten vorsortiert wird.

- DIE INTERPRETIERBARKEIT VON MODELLEN UND ERKLÄRUNGEN: Für den Nutzer eines Musikempfehlungsdienstes kann es von Vorteil sein, zu wissen und zu verstehen, warum ihm bestimmte Musik vorgeschlagen wird. Die Ergebnisse der Analyse von zahlreichen Benutzerstatistiken können allerdings die persönlichen Präferenzen nur bis zu einem gewissen Grad abbilden. Die Kategorisierung und Erklärung der Vorschläge anhand signalnaher Audiomerkmale ist ebenfalls wenig verständlich. Abb. 1 verdeutlicht diese Situation: hier wird zwischen drei Merkmalsquellen (horizontal) und Nähe zum Benutzer (vertikal) unterschieden. Während die meisten Klassifikationsszenarien größtenteils auf Signalmerkmalen operieren (unteres Rechteck), ist es dagegen eher der Wunsch des Anwenders, dass verständliche und persönliche Musikauswahlkriterien hergestellt werden (oberes Rechteck). Eine Möglichkeit, dieses Problem zu lösen, wäre die Integration von Merkmalen, die einen Bezug zu Musiktheorie haben und allgemein verständlich sind. Solche Charakteristika haben sich für Erkennung von Musikgenres und Musikstile in [21] bewährt und es gibt mittlerweile prototypische Web-Anwendungen, die in diese Richtung zeigen [14].

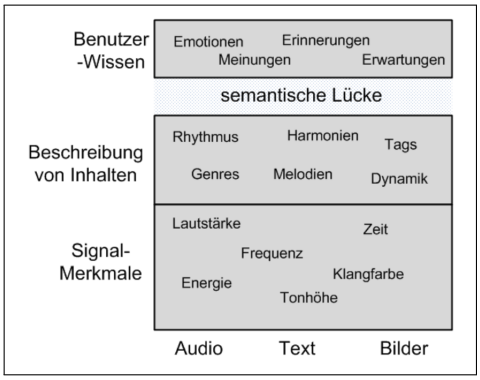

Abbildung 1: Drei Ebenen der Musikdaten nach [4]

- BEWERTUNG UND VERGLEICH VON ALGORITHMEN: Der Vergleich der Qualität von automatisch generierten Empfehlungen ist eine zentrale Fragestellung in der Forschung, u. a. auch nach benutzerorientierten Bewertungskriterien. Benutzerzufriedenheit lässt sich nicht immer formal beschreiben. Die Generierung von Wiedergabelisten erfordert daher spezifische Bewertungsmaße. Die übliche Vorgehensweise bei Wiedergabelisten-Bewertungen ist es, Trefferquoten anhand von vorhandenen Beispieldatensätzen zu berechnen. Da die Lieder einer Wiedergabeliste in der Regel nacheinander abgespielt werden, sollten z. B. die benachbarten Musikstücke nicht sehr unterschiedlich sein, beispielsweise in Bezug auf Emotion, Tempo oder Genre. Allerdings wird auch eine gewisse Diversität erwünscht (d.h. unterschiedliche Interpreten und Alben). Einige Eigenschaften guter Wiedergabelisten können letztlich nur mit Hilfe von Benutzerstudien bewertet werden.

2.2. Motivation und Ziele der Projektgruppe

Die Ziele der Projektgruppe werden durch die Vorgaben im Projektgruppenantrag bestimmt. Nachfolgend werden daher die Ziele aufgelistet, sowie die Ideen und Wünsche, die von der Projektgruppe selbst hinzugefügt wurden, in der Produktvision dargestellt. Im Laufe des vorliegenden Endberichtes kann so nachvollzogen werden, inwiefern die Planung des Projektes und die Ziele im ersten und zweiten Halbjahr erreicht und umgesetzt werden konnten.

2.2.1. Minimalziele

- Erstellung einer Web- oder wahlweise mobilen Anwendung, welche Musikstücke auf Basis vorgegebener Beispiele in Form einer Wiedergabeliste empfiehlt.
- Implementierung von mindestens zwei unterschiedlichen Ansätzen zur Musikempfehlung, z. B. inhaltsbasiert und auf Basis von kollaborativen Filtern.
- Durchführung einer Studie, bei der mehrere benutzerorientierte Kriterien verwendet werden, um das realisierte Empfehlungssystem zu bewerten.

- Dokumentation der Arbeit durch Erstellung eines Zwischenberichts (nach dem ersten Semester) und eines Endberichts (nach dem zweiten Semester).

2.2.2. Vorgaben für die Projektgruppe

Im Rahmen der Projektgruppe soll ein neuartiges Musikempfehlungssystem entwickelt werden, welches auf bestehende Softwarekomponenten der beteiligten Lehrstühle aufbaut, diese in innovativer Form zusammenführt und die oben skizzierten offenen Problemstellungen in angemessener Weise berücksichtigt. Im Ergebnis ist das zu erstellende System nicht nur in der Lage, geeignete Musikempfehlungen auf Basis verschiedener Daten (Inhalte, Statistiken) zu generieren, sondern auch die Möglichkeiten des AMUSE-Frameworks zur automatischen Ableitung von Inhaltsinformationen zu nutzen.

Abb. 2 stellt eine grobe Skizze für die zu erstellende Software dar. Zu den Mindestzielen der Projektgruppe zählt die Entwicklung einer webbasierten oder mobilen Anwendung, welche die Entdeckung neuer Musik, aber auch die Erstellung von Playlisten aus den vorhandenen Musikstücken ermöglicht. Alleine die Generierung von Playlisten kann hier auf unterschiedlichen Wegen gelöst werden: die Listen können automatisch, auf Basis von inhaltlichen Einflussfaktoren, generiert werden (Ähnlichkeit zu Vorgängern, Beschreibung durch Tags usw.), oder es wird interaktiv durchgeführt, ðder Benutzer wählt einigene Lieder aus den vorgeschlagenen aus und gibt somit seine Präferenzen explizit bekannt.

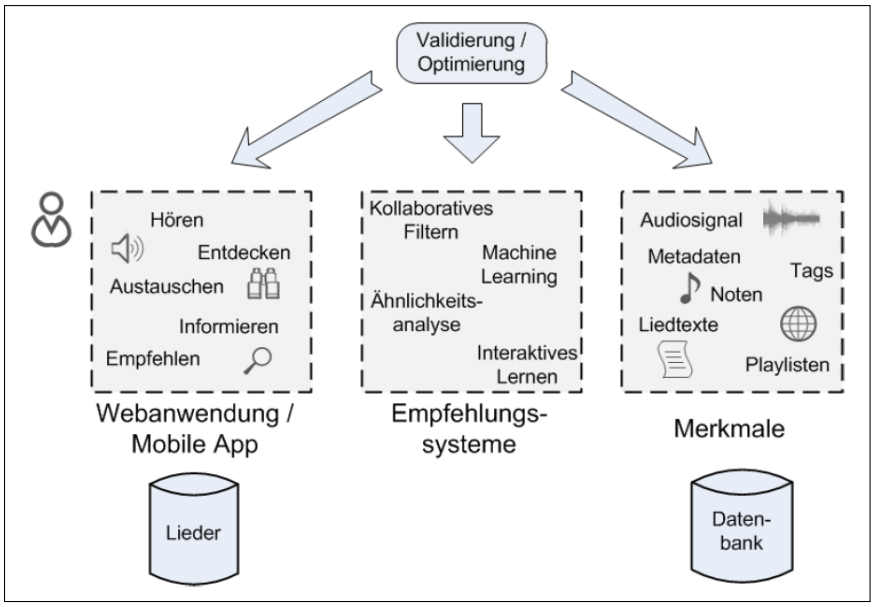

Abbildung 2: Skizze der Komponenten für eine Musikempfehlungsanwendung.

Der Schwerpunkt der Bewertung der Empfehlungssysteme wird auf benutzerzentrierte Kriterien gesetzt. Für die erzeugten Playlisten kann beispielsweise gemessen werden, wie lange die Lieder angehört wurden, wie viele übersprungen wurden oder, ob der Überraschungseffekt eine Rolle spielt. Aber auch der Interaktionsaufwand für den Nutzer

oder die Laufzeit- und Speicherplatzanforderungen der untersuchten Methoden sollen untersucht werden. Dazu sollen bestehende Kriterien aus einschlägigen Quellen [12, 6] verwendet, aber auch neue entwickelt werden. Es ist auch ausdrücklich erwünscht, eine Studie mit Hörerbeteiligung durchzuführen.

Da die Kombination von Merkmalen aus unterschiedlichen Quellen zu einer besseren Qualität der Genreerkennung in [11, 15] geführt hat, soll diese Herangehensweise zumindest prototypisch für Musikempfehlungen angewendet werden. Im Unterschied zu den oben erwähnten Studien kann der Einfluss der Merkmalskombination nicht nur anhand der Klassifikationserfolge / Trefferquoten gemessen, sondern auch mithilfe von benutzerzentrierten Maßen evaluiert werden. Insbesondere ist der Einbezug der interpretierbaren Merkmale vielversprechend, um die Frage zu beantworten, warum bestimmte Musik vorgeschlagen, bzw. in eine Playliste aufgenommen wird.

3. Seminarphase

Die Projektgruppe hat mit einer Seminarphase begonnen, zu der jeder Projektgruppenteilnehmer die Aufgabe hatte, sich in einen von den Betreuern ausgewählten Themenbereich einzuarbeiten und diesen vorzustellen. Ziel der Seminarphase war es, der Projektgruppe einen gemeinsamen Wissensstand der wichtigsten Themenfelder zu vermitteln. Die gehaltenen Vorträge erfolgten am 18. und 19. Oktober 2013, beschränkten sich auf jeweils eine halbe Stunde Redezeit mit anschließenden Fragen und sollten einen Überblick über das jeweilige Thema geben. Im Folgenden sind die Beschreibungen der Vorträge in chronologischer Reihenfolge aufgelistet.

3.1. Frontend-Entwicklung - Eine Einführung in JSF2

Für die Realisierung dieses Projekts wurde ein leistungsfähiges Framework für die Entwicklung der Benutzerschnittstelle benötigt. Im Bereich der Webentwicklung bietet *JavaServer Faces 2* eine vielversprechende Lösung an. Einerseits hat dieses Framework eine lange Entstehungsgeschichte hinter sich und ist als erwachsen anzusehen, andererseits bietet es auch die Unterstützung aktueller Technologien.
JavaServer Faces hat seit dem ersten Release im Jahre 2004 in der Version 1.0 einen langen Weg hinter sich. Das Ziel, dem Entwickler die Sorgen im Low-Level-Bereich der Webentwicklung (Umsetzung in HTML, JavaScript und Kommunikation mit dem Server usw.) zu nehmen, wurde schon in dieser Version erreicht, auch wenn es noch viele Unzulänglichkeiten gab. Seit Version 2.0 sind diese Kinderkrankheiten der Spezifikation überwunden und JavaServer Faces bietet ein komplettes und leistungsfähiges Framework, um komplexe Webapplikationen in allen Anforderungsbereichen zu realisieren.

Die Entwicklung von Webapplikationen ist in Zeiten des Internets ein omnipräsentes Thema. Gegenüber Desktop-Applikationen bieten Webapplikationen einige klare Vorteile. Allen voran sind sie von überall auf der Welt erreichbar. Die Weiterentwicklung und Fehlerbereinigung kann augenblicklich realisiert werden, da keinerlei kompilierter Code an den Benutzer bzw. Kunden ausgeliefert werden muss.

JavaServer Faces 2 (kurz JSF2) ist ein Framework für die Entwicklung von komplexen Webapplikationen, dessen Ziel ist es, dem Entwickler einen einfachen, aber leistungsfähigen und standardisierten „Werkzeugkasten“ an die Hand zu geben. Hierbei wurde großer Wert auf eine schnelle und effiziente Entwicklungsphase und die damit verbundene, anschließende Pflege gelegt.

Wie viele andere verfügbaren Frameworks im UI-Bereich verfolgt JSF2 das Model-View-Controller-Muster. Das Framework unterstützt das Erstellen von Formularen, die Validierung von Eingaben, die Verknüpfung und Ausführung von Business-Logik und die letztendliche Darstellung der Ergebnisse. Dabei beinhaltet die Spezifikation von JSF2 viele vorgefertigte Oberflächenkomponenten, mit denen es möglich ist, Ergebnisse entsprechend zu verarbeiten.

Es gibt unzählige Frameworks in verschiedenen Sprachen zur Entwicklung von Webapplikationen. Für die Projektgruppe wurde ein Framework benötigt, das einfach zu erlernen und in der Lage ist, die gestellten Anforderungen auf Ebene der Oberfläche zu realisieren. Für das gesamte Projekt wurde Java als primäre Sprache gewählt. Da die Einführung in Java zum festen Bestandsteil des Informatikstudiums an der TU Dortmund gehört, sind alle Mitglieder in der Gruppe damit vertraut. JSF2 ist Teil der *Java Platform Enterprise Edition*, welche im Bereich des Backends genutzt wurde. JSF2 garantiert hierbei eine nahtlose und einfache Integration in die Architektur.

3.2. Playlist Generation

Bei der Entdeckung neuer Musik (dem Hauptziel des Projektes) ist es aufgrund der unterschiedlichen persönlichen Neigungen der potentiellen Anwender anzuraten, ein möglichst großes Spektrum an Musik zur Empfehlung anzubieten. Daraus folgt eine Notwendigkeit, die Suche in dieser Musiksammlung einzugrenzen - zum einen um den Entdeckungsprozess zielgerichtet zu gestalten, zum anderen aus Performanzgründen.

Ein System zur Musikentdeckung sollte also aufgrund vorhandener Informationen, die die gewünschte Musik beschreiben, eine Liste von Musiktiteln generieren, die der Art der gewünschten, zu entdeckenden Musik, enstprechen. Eine solche Liste bezeichnen wir im folgenden als *Playlist* und ihr Erstellungsprozess *Playlist Generation*. Ein solcher Prozess kann sowohl *interaktiv* (also durch den Benutzer gesteuert) oder *automatisiert* ablaufen. Auch kann eine Playlist sich iterativ entwickeln, oder aber als ganzes generiert werden.

Um dem Anwender einen möglichst dynamischen Entdeckungsprozess zu bieten, bietet es sich für unser Projekt an den Nutzer iterativ aus den präsentierten Empfehlungen wählen zu lassen. So kann die Art der nächsten Empfehlung adaptiv an die aktuelle Situation angepasst werden: der Nutzer kann also die Art der folgenden Empfehlungen beeinflussen und ist somit nicht immer an dieselben Zielvorgaben gebunden.

Ein so gestalteter Prozess läßt sich auf den Themenbereich der Empfehlungssysteme zurückführen: die Playlist Generation erfolgt durch die wiederholten Selektionen von Musiktiteln aus den jeweiligen Vorschlagsmengen.

Deswegen ist es für die Generierung adäquater Vorschläge notwendig alle Musiktitel miteinander in sinnvolle Relationen zu setzten. Implementiert wird dies meist durch

eine Art *Distanzmaß* (oder aber das Komplement: ein *Ähnlichkeitsmaß*) zwischen den einzelnen Tracks, basierend auf *Metadaten* beliebiger Art und Struktur.

Somit beeinflusst sowohl die Implementierung der Scorefunktion, als auch die Wahl der jeweiligen Datenbasis (Inputs) die Qualität der Vorschläge. Diese unterteilen sich hauptsächlich in *collaborative filtering*-, Experten- und *Content*-basierte Daten.

3.3. User-Centered Evaluation

Die immer einfachere Verfügbarkeit von Musik und die steigende Masse der Lieder führen oftmals dazu, dass Nutzer von eben dieser Masse erdrückt werden. MIR-Entwicklungen (Music Information Retrieval), automatische Wiedergabelisten und die Ähnlichkeit von Liedern spielen dabei eine große Rolle und sollen dem Nutzer seine Arbeit erleichtern. Ist dieser auf der Suche nach einem einzelnen Lied oder Interpreten, so ist dies einfach und problemlos möglich. Wenn der Nutzer aber nun automatisch eine Playlist für sich erstellen möchte, zum Beispiel für einen bestimmten Kontext, so ist dies, je nach Größe der Sammlung, von Hand kaum bis nicht lösbar. Aus diesem Grund wurden inzwischen viele verschiedene Algorithmen zur automatischen Erstellung von Wiedergabelisten entwickelt. Bereiche die hier erforscht werden sind zum Beispiel die Stimmungserfassung von Liedern, oder die Ähnlichkeitsbestimmung der Musik. Problematisch ist, dass für viele dieser Algorithmen nur System-basierte Tests existieren. Dabei ist im Bereich der Musik der Nutzer selbst von besonderer Bedeutung, da Musik für viele ausschließlich von Gefühlen abhängt. Seit 2004 gibt es jedes Jahr die „Music Information Retrieval Evaluation eXchange“ (MIREX). Dies ist ein Community-basierter Rahmen für offizielle Tests von MIR Algorithmen und Systemen. Dort werden weltweit entwickelte MIR Systeme alle mit demselben vorgefertigten Datensatz getestet und ihre Ergebnisse miteinander verglichen. Es werden dazu verschiedene Arten von Aufgaben verfasst, auf die Bewältigung derer die Systeme getestet werden. Beispiele für diese Aufgaben sind:

- Audio Genre Classification
- Audio Melody Extraction
- Audio Mood Classification

Das Ziel der MIR Forschung ist eine möglichst ähnliche Entwicklung, wie bei textuellen Suchmaschinen. Dies ist jedoch ein weit schwerer zu lösendes Problem, da Musik viel komplexer ist als Text. Playlists, die mit Hilfe eines Algorithmus automatisch erstellt worden sind, werden häufig mit einer, oder mehreren, von drei unterschiedlichen Metriken bewertet. Diese sind:

- Semantischer Zusammenhang zwischen Liedern
- Studien über Nutzer Meinungen
- Vergleich mit vorgefertigten Listen

Bei dem semantischen Zusammenhang geht es darum, dass die Lieder einer Playlist gut zusammen passen. Zum Beispiel hinsichtlich des Genres oder des Stils. Bei der Bewertung von Nutzer-Meinungen müssen diese Wiedergabelisten anhören und bewerten. Aufgrund der abgegebenen Meinungen werden die Listen beurteilt. Diese Methode ist jedoch auch zeitaufwändig und teilweise schwer auszuwerten. Die dritte Metrik bezieht sich auf den Vergleich mit anderen Listen. Diese können zum Beispiel von Hand erstellt, oder aus einer Radio-Musik-Sammlung gewonnen werden. Innerhalb dieser Seminararbeit wird auf diese drei Metriken einzeln eingegangen. Sie werden genauer erläutert und es werden Arbeiten aus der aktuellen Forschung untersucht. Leider sind viele dieser benutzerbasierten Tests vom Umfang her zu aufwändig um sie innerhalb unserer Projektgruppe anzuwenden. Außerdem werden auch viele Verfahren nicht so weit erläutert, als das wir sie einfach anwenden könnten. Geeignet sind jedoch Tests, in denen die Benutzer unser System verwenden und anschließend dazu Fragen gestellt bekommen. Auch könnte die Zeit gemessen werden, die der Nutzer benötigt, um eine ihn zufriedenstellende Liste zu erhalten. Auf diese Weise können wir recht einfach eine allgemeine Meinung der Nutzer zu unserem System erhalten.

3.4. Recommender Systems

Empfehlungssysteme bieten in der Informatik die Möglichkeit, einem Nutzer Vorschläge zu unterbreiten, die auf unterschiedliche Art und Weise bestimmt werden. Vorschläge können hierbei Produkte sein, die ein Kunde möglicherweise kaufen möchte oder aber neue Lieder, die dem Hörer gefallen könnten.
In der Ausarbeitung wurden daher aktuelle Ansätze zur Erstellung bzw. technischen Umsetzung eines Empfehlungssystems eingeführt und dargestellt. Insbesondere werden dabei kollaborative, inhaltsbasierte, wissensbasierte und kontextbasierte Systeme thematisiert.
Für die in der Projektgruppe angedachte Anwendung ist ein gutes Empfehlungssystem extrem wichtig. Denn jeder Nutzer soll eigene Liedvorschläge basierend auf seinen Musikvorlieben und den Stimmungen der Lieder erhalten, wobei dem Nutzer auch neue, noch unbekannte Lieder angeboten werden sollen, die zu seinen Vorlieben passen. Aus diesen Gründen könnte sich eine Mischung aus einem inhalts-, wissens- und kontextbasierten System eignen. Denn mittels des inhaltsbasierten Systems könnten sich über Informationen, wie Genre, Album oder Produktionsort und -jahr ähnliche Lieder finden lassen, die möglicherweise auch das Entdecken neuer Lieder ermöglichen. Das wissenbasierte System könnte wiederum nützlich sein, um sehr spezifische Informationen - wie diverse Merkmale über die Tonart oder Stimmung - zu sammeln und damit dann stimmungsbasierte Vorschläge zu unterbreiten. Ein kontextbasiertes System könnte für die Verbesserung der Ergebnisse genutzt werden, da beispielsweise der aktuelle Aufenthaltsort oder das Datum die Vorlieben eines Nutzers beeinflussen könnten. So könnte in der Weihnachtszeit ein Weihnachtslied eher gefallen, als im Sommer. Das kollaborative System könnte ebenfalls für die Musikempfehlung genutzt werden, jedoch wären Ergebnisse erst bei hohen Nutzerzahlen zufriedenstellend. Außerdem ist die Prämisse, dass wenn ein Lied X einem Nutzer A gefällt, es auch dem Nutzer B gefallen wird, da die Vorlieben

in der Vergangenheit ähnlich waren, nicht immer richitg.
Es gibt also bereits bekannte Empfehlungssysteme, die sich für den Einsatz in der Anwendung eignen würden, um personalisierte Nutzervorschläge anbieten zu können. Es müssen jedoch noch wichtige Punkte geklärt werden, wie z.B. die gesammelten Informationen und welchen Einfluss die jeweiligen Informationen auf das Ergebnis haben. Beispielsweise hat das Tempo eines Liedes Einfluss auf die subjektive Wahrnehmung, jedoch ist die Extrahierung des Tempos nur schwierig umzusetzen [7].

3.5. Metadata Features

In der Ära der Smartphones, mp3-Player und Musik-Streaming-Dienste findet man Musik immer öfter auf Festplatten oder im Internet gespeichert. Deshalb ist es von Bedeutung, digitale Musik mit Musikinformationen, den *Metadaten*, zu versehen. Sie bieten eine Möglichkeit, Musikkonsumenten mit verschiedenen Informationen (z. B. über den Interpreten, das Album, das Genre und vieles mehr) zu versorgen.

In der Ausarbeitung werden die Metadaten als Ansatz bei der inhaltsbasierten Musiksuche behandelt. Es werden web-basierte Dienste wie z. B. *Gracenote* (`gracenote.com`) und *MusicBrainz* (`musicbrainz.org`) betrachtet - beide Datenbanken bauen auf benutzerkontrollierten Einträgen auf Prinzip Peer-Review[2] und werden dadurch für die Allgemeinheit freigegeben. Diese Dienste bieten sowohl sachliche Metadaten, wie z. B. Zusatzinformationen zu einem Titel, als auch s. g. kulturelle Metadaten, die subjektive Konzepte beinhalten. Für ein Metadaten-System ist es wichtig, dass die Beschreibungen der Musik korrekt sind und die Bedeutung des Metadaten-Vokabulars von jedem verstanden wird.

Metadaten bzw. *Tags*[3] dienen dazu, Informationen im Textformat in den Audiodateien zu speichern. Der daraus resultierende Vorteil ist, dass man nicht alles in Datei- bzw. Ordnernamen unterbringen muss und so zu jeder Datei zusätzliche Angaben abspeichern kann. Für das Speichern von anderen Objekten, wie z. B. von Bildern, sind die meisten bestehenden Standards ungeeignet. Hierfür sind externe Dateien besser geeignet. Daher werden in dieser Ausarbeitung ein paar der bekanntesten und wichtigsten Tagging-Formate vorgestellt: ID3-Tag, Vorbis comment, APE-Tag.

Eine weitere Art von Musikinformation stellen die *Social Tags* dar. Diese sind Kommentare und Informationen, die von Nutzern in Form von Schlüsselwörtern der Musik hinzugefügt werden. Es gibt sehr viel nützliche Auskünfte, wie z. B. über das Genre, die Stimmung und die Qualität, jedoch finden sich hier ebenso auch irrelevante Informationen. Vorteile sind u. a. die Gruppierung von Liedern basierend auf Text oder die Ermittlung sozialer Gruppen mit gemeinsamen Interessen, z. B. Menschen die den selben Musikstücken Schlagwörter zuordnen. Trotz der Schwierigkeit, die sich bei der Kontrolle über die Informationsqualität ergibt, sind Social Tags eine Quelle des von Menschen

[2] Ein Verfahren zur Beurteilung wissenschaftlicher Arbeiten, insbesondere von Publikationen. Dabei werden unabhängige Gutachter aus dem gleichen Fachgebiet wie die Autoren herangezogen, um die Qualität zu beurteilen.

[3] Ein Tag (engl. Etikett, Schildchen, Auszeichner, Anhänger) ist eine Auszeichnung eines Datenbestandes mit zusätzlichen Informationen.

generierten, kontextbasierten Wissens über digitale Musik, welche eine wesentliche Rolle bei der Lösung vieler MIR[4]-Probleme einnehmen könnte.

Im Projekt wurde das Herauslesen von *Metadaten/ID3-Tags* aus Musikdateien selbst sowie die Abfrage der MusicBrainz-Datenbank bereits erfolgreich umgesetzt.

3.6. High-Level Features

Es gibt verschiedene Definitionen davon, was ein Feature zu einem High-Level Feature macht. Hier zeichnen sich High-Level Features dadurch aus, dass sie Musikstücke in einer abstrakten, von Menschen verständlichen Weise beschreiben. Somit unterscheiden sie sich von Low-Level Features, die Musikstücke technisch, signalorientiert repräsentieren. Sie können klassische musikalische Begriffe, wie Tonhöhe, Rhythmus und Tempo oder auch die Wahrnehmung von Musik durch Intensität, Stimmungen oder Gefühle beschreiben. Ein Empfehlungssystem kann somit von High-Level Features profitieren, da sie Wahrnehmung und Geschmack beschreiben können. Die Werte können auf verschiedene Weisen direkt aus dem Audiosignal oder mit Machine Learning Algorithmen aus Low-Level Features gewonnen gewonnen werden. Verschiedene Techniken werden vorgestellt, mit dem Ziel eine Einbindung in ein Musikempfehlungssystem zu ermöglichen.

3.7. Similarity Analysis

In der multivariaten Statistik gibt es Verfahren, die es ermöglichen, verschiedene Objekte miteinander zu vergleichen und bzgl. ihrer Ähnlichkeit zu bewerten. Anwendung finden diese Verfahren zur Ähnlichkeitsanalyse in vielen Bereichen, u. a. beim Vergleich von zwei unterschiedlichen Musikstücken. Das Ziel dieser Ausarbeitung ist einen Überblick über die Grundlagen der Ähnlichkeitsanalyse zu verschaffen, um aufbauend darauf die Methoden, die zur Analyse notwendig sind, näher beschreiben zu können. Beispiele für Anwendungen aus unterschiedlichen Bereichen, speziell im Bereich der Musikähnlichkeitsanalyse, sind ebenfalls Teil dieser Arbeit und sollen dem Leser einen schnellen Einstieg in das Themengebiet der Ähnlichkeitsanalyse ermöglichen. Zusätzlich kann diese Ausarbeitung als Vorbereitung auf weiterführende Literatur gesehen werden.
Der Leser wird zunächst in einige Grundbegriffe, wie dem Ähnlichkeitsmaß eingeführt und diese am Beispiel der Clusteranalyse näher erläutert. Folgend werden Gaussian Mixture Models grundlegend beschrieben, um die Verwendung der Kullback-Leibler Divergenz im Bereich der Musikähnlichkeitsanalyse erklären zu können. Die Kullback-Leibler Divergenz ist ein spezielles Ähnlichkeitsmaß, was den Unterschied zweier Verteilungen misst und so die Ähnlichkeit bzw. Unähnlichkeit der Verteilungen berechnet. Um dieses Verfahren in der Musikähnlichkeitsanalyse verwenden zu können, ist es notwendig die Verteilungen zweier Lieder zu berechnen - was in der Ausarbeitung nur kurz beschrieben wird. Für die Entwicklung des Empfehlungssystems in diesem Projekt, werden u.a. auf der Basis der vorgestellten Ähnlichkeitsmaße, eigene Ähnlichkeitsmaße entwickelt. Die Relevanz dieser Ausarbeitung für das Projekt ist also nicht nur theoretisch, sondern auch praktisch gegeben.

[4]MIR steht für Music Information Retrieval

3.8. Emotion Models From Music Theory

Diese Ausarbeitung beschäftigt sich mit den Auswirkungen der grundlegenden Musikelemente auf Emotionen. Die grundlegenden Musikelemente beinhalten Tonhöhe, Lautstärke, Konsonanz oder Dissonanz, Tonarten, Akkorde, Klangfarben, Rhythmus und Tempo. In dem ersten Teil der Ausarbeitung wird die Beziehung zwischen Lautstärke und Emotion beschrieben. Je lauter und dynamischer die Lautstärke ist, desto aktivierender wirkt das Lied. Auch der Zusammenhang der Emotionen und Tonhöhe wird in diesem Teil dargestellt: Eine sehr hohe Tonlage bringt elegante Stimmung und eine relative hohe Tonhöhe wirkt fröhlich, ruhig oder träumerisch, während die Musik sich mit relativ niedriger Tonhöhe lebhaft, erregend und würdevoll anhört.
Anschließend werden die harmonischen/disharmonischen Intervalle und ihre Auswirkungen auf Emotionen erläutert. Normalerweise erbringt die harmonische Melodie eine positive und ruhige Stimmung und disharmonische Melodie ruft eine unruhige und nervöse Stimmung hervor.
Tonarten und Akkorde werden in dem dritten Teil der Ausarbeitung behandelt. Oft wird Musik in Dur als heiter und fröhlich empfunden und die Musik, die in Moll geschrieben ist, ist häufig traurig und ernst.
Zuletzt wird es herausgefunden, was für Zusammenhang es zwischen Klangfarben/-Rhythmus/ Geschwindigkeit und Emotionen gibt. Musik mit hell strahlender Klangfarbe hört sich aktivierender an, während Musik mit weicher Klangfarbe beruhigend empfunden wird. Höhere Geschwindigkeit ist generell mit positiver und glücklicher Emotion verbunden, während eine niedrigere Geschwindigkeit mit trauriger Stimmung verbunden ist.
Die oben genannten Musikelemente und ihre Einflüsse auf die Stimmungen der Musik werden in diesem Projekt anhand Valence-Arousal-Model verwendet. Die Geschwindigkeit, Lautstärke und Tonarten sind die wichtigsten Kriterien für dieses Projekt (Siehe Kapitel 6.7.7.).

3.9. Scrum

Die Ausarbeitung beschäftigt sich mit Scrum (englisch für Gedränge), einer Softwareentwicklungsphilosophie, die sich seit Ende der 90er Jahren immer mehr in Unternehmen durchsetzt. Dabei basiert der Zusammenhang der Namenswahl auf dem Versuch der Rugbymannschaft, als Einheit zu agieren und dadurch Geschwindigkeit und Flexibilität zu erhöhen, wie es auch bei der agilen Softwareentwicklung der Fall ist. Zunächst erfolgt ein Rückblick auf vorherige Vorgehensmodelle (Stufenmodel, Wasserfallmodelll, Spiralmodell, V-Modell, RUP), um den Entwicklungsprozess nachvollziehen zu können. Die Prinzipien von Scrum werden anschließend genauer betrachtet, sowie der genaue Ablauf eines Produktentwicklungsprozesses mit Scrum. Hierzu wird auf die einzelnen Bestandteile eines Scrum Projektes näher eingegangen und jede Komponente (Rollen, Artefakte, Meetings) ausführlich erklärt. Abschließend wird eine Einschätzung abgegeben, ob und wie sich Scrum auf die Projektgruppe anwenden lässt.

3.10. MIR Frameworks

Ein System, welches aus einer Menge von Daten relevante Informationen extrahiert, bezeichnet man als *Information Retrieval System*. Die Ausarbeitung *MIR Frameworks* beschäftigt sich mit der speziellen Domäne der musikalischen Informationsgewinnung, mit dem Ziel verschiedene Frameworks zu evaluieren. Dabei wurden unter anderem die Frameworks *ChromaToolbox*, *MIRtoolbox* und *AMUSE* betrachtet. Im Speziellen werden die einzelnen Funktionen des *jMIR* Frameworks vorgestellt.

In den letzten Jahrzehnten ist vor allem durch das Internet und die immer größeren, digitalen Speichermöglichkeiten die Menge an Musikdaten, sowie die Menge an Metadaten, die diese Musik beschreiben, enorm gestiegen. Es existieren bereits verschiedene Standards, wie z. B. Metatags, um diese Daten zu strukturieren und zu beschreiben. Diese Informationen beschränken sich jedoch meist auf sachbezogene Informationen, wie z. B. Interpret, Genre oder Veröffentlichungsjahr. Nur selten lässt sich mit solchen Tags die Charakteristik von Musik im Detail becshreiben. Diese Tags ordnen Musik nur einer sehr groben Kategorie, wie z. B. dem Genre, zu. Es ist jedoch erwünscht, dass der Benutzer anhand von emotional geprägten Merkmalen auch neue Musik entdecken kann. Hierzu bedarf es komplexere Merkmale und Verfahren, wie gestütztes maschinelles Lernen, um einem Nutzer passende, unbekannte Musik vorzuschlagen.

4. Projektmanagement und Projektplanung

Die Arbeit in einer Projektgruppe, in einem Zeitraum von zwei Semestern, erfordert eine strukturierte und klar definierte Herangehensweise für einmalige und sich wiederholende Aufgaben. Dies betrifft sowohl die Mitglieder, als auch die Betreuer. Die Fakultät Informatik an der Technischen Universität Dortmund gibt Richtlinien zur Durchführung von Projektgruppen in ihrer Projektgruppenordnung vor, legt dabei aber nicht fest, wie die Gruppe sich intern organisieren muss.

4.1. Projektabgrenzung

Eine der größten Herausforderungen in Projekten besteht in der Entscheidung, was im geplanten Projekt geschehen soll, bzw. was nicht passieren soll. Erfolgsorientierte Planung ist sehr wichtig, dies wird konkret durch eine zeitliche, eine sachliche und eine soziale Projektabgrenzung beschrieben.

4.1.1. Zeitliche Projektabgrenzung

Erstes Semester. Zu Beginn der Arbeitsphase haben sich die Teilnehmer auf zwei feste Besprechungstermine im wöchentlichen Rhythmus geeinigt. Diese Termine wurden zusammen mit den Betreuern abgehalten. Sie dienten dazu, den jeweiligen Fortschritt von einzelnen Arbeitsgruppen festzuhalten, wichtige Entscheidungen bezüglich inhaltlicher Aspekte zu treffen und Arbeitsergebnisse zu präsentieren. Für arbeitsgruppenspezifische Aufgaben wurden zusätzliche Termine von den einzelnen Arbeitsgruppenmitgliedern in

Doodle-Umfragen vereinbart. Alle weiteren Arbeiten wurden in E-Mails und Telegram, einem Instant Messenger, besprochen.

Zweites Semester. Am Anfang dieses Semesters sollte zunächst eine Planung für das gesamte Semester festgelegt werden, dafür wurde die Hälfte der Projektgruppe als Projektplanungsgruppe ausgewählt. Die „Projektplaner“ mussten sich erst einmal einen Überblick über bestehende Ziele und Vorgehensweisen verschaffen. Diese deckten sich aber nicht hundertprozentig mit denen des Kunden (der Betreuer) und führten, wenn dies nicht rechtzeitig aufgeklärt wurde, im weiteren Projektverlauf zu „bösem Erwachen“ auf beiden Seiten. Um dieses Risiko zu minimieren und einen übersichtlichen Plan zu erschaffen, haben sich die Gruppenmitglieder für die Software namens *Project Libre*[5] entschieden.
Desweiteren wurden die wöchentlichen Treffen auf einen Termin in der Woche reduziert, die Projektgruppenmitglieder mussten aber dafür ergebnisorientiert arbeiten. Die Treffen fanden immer mittwochs statt und bestanden aus kurzen Präsentationen der Ergebnisse der vergangenen Woche. Der Projektplan wurde wöchentlich aktualisiert, ergänzt und gepflegt.
Als Starttermin für die ersten Aufgaben laut Project Libre wurde der 23. April 2014 festgelegt. Als Endtermin ist das Ende des laufenden Semesters gesetzt. Nach aktuellem Projektplan findet die endgültige Abgabe, inklusive Evaluation und Dokumentation, am 12. August 2014 statt.

4.1.2. Sachliche Projektabgrenzung

Die sachliche Projektabgrenzung legt Ziele und Nicht-Ziele (optional) des Projektes dar. Der Leistungsumfang und Besonderheiten des Projektes werden beschrieben.

Inhaltliche Ziele. Ziel des Projektes „Musikempfehlungen - Automatische Erstellung von Playlisten zur Entdeckung von Musik“ ist es, ein neuartiges Musikempfehlungssystem zu entwickeln, welches auf bestehende Softwarekomponenten aufbaut und diese in innovativer Form zusammenführt. Die Empfehlungen erfolgen entweder inhalts- oder stimmungsbasiert, worauf in den nächsten Kapiteln näher eingegangen wird. Die Minimalanforderung für einen erfolgreichen Abschluss des Projektes beinhaltet die Erstellung einer web-basierten Anwendung, welche Musikstücke auf Basis von Vorschlägen in Form einer Wiedergabeliste empfiehlt.

Organisatorische Ziele. Organisatorische Ziele sind zum einen die Durchführung einer Studie, um das realisierte Empfehlungssystem zu bewerten und zum anderen die Dokumentation der Arbeit durch die Erstellung eines Endberichts.
Aus den inhaltlichen und organisatorischen Zielen heraus ergeben sich die Projektphasen, welche wir in den beiden Semestern durchlaufen haben und die in der folgenden Tabelle beschrieben werden.

[5] `http://www.projectlibre.de/`

Tabelle 1: Die Projektphasen der PG577

Bezeichnung	Mittel/Methode	Resultate
Phase 1:Initialisierung	Kick-Off Meeting	
Phase 2:Definition	Projekthandbuch	Produktvision
Phase 3:Planung	Backlog, Balkendiagramm (ProjectLibre)	Pläne
Phase 4:Vorgehen	Vorgehensmodelle	Ergebnisberichte
Phase 5:Implementierung	Code schreiben	Anwendung
Phase 6:Kontrolle	Prüfplan (Quality Assurance)	Qualität
Phase 7:Evaluation	Online-Fragebogen	Feedback
Phase 8:Abschluss	Bericht	Endbericht

In der Initialisierungsphase (Phase 1) in der o. g. Tabelle wurde der Begriff „Kick-Off Meetings“ benutzt. Um sich unter diesem Begriff etwas Konkretes vorstellen zu können, werden an dieser Stelle einige Inhalte, die ein Kick-Off Meeting enthalten sollte, aufgezählt:

- Situationsanalyse
- Soll-Zustand formulieren (Systemziele)
- Projektnamen definieren
- Projektmanager vorschlagen/bestimmen
- Gesamtaufwand grob schätzen
- Risiken definieren
- Meilensteine und Termine ausarbeiten und aufführen (Eckdaten)

4.1.3. Soziale Projektabgrenzung

Die soziale Projektabgrenzung erfolgt durch die Definition von Rollen für folgende Bereiche:

- Projektauftraggeber: Dr. Igor Vatolkin, Dr. Geoffray Bonnin, Prof. Dr. Dietmar Jannach
- Projektmanager: Anfangs Martin Cmok, gefolgt von Michael Gajda und Lora Asenova
- Projektgruppenmitglieder

Die Aufgabenverteilung wurde folgendermaßen vorgenommen: Für jedes Arbeitspaket wurde ein Verantwortlicher (im Projektplan mit Sternchen versehen) und Mitarbeitende (s.g. Ressourcen) definiert.

4.2. Weitere Projekttools

Scrum. Ganz am Anfang des Projektes haben die Betreuer für die Wahl des Vorgehensmodells Scrum vorgeschlagen, was von den Teilnehmern einvernehmlich angenommen wurde. Bei der Etablierung von Scrum in die Projektgruppe wurden nur einige Elemente (Rollen und Artefakte) ausgewählt. Die strikte Anwendung von Scrum als ganzheitliches Vorgehensmodell wäre im Rahmen der Projektgruppe zu bindend und teilweise nicht anwendbar gewesen. Stattdessen wurden sinnvoll anwendbare Elemente ausgewählt bzw. abgeändert.
Als Beispiel lässt sich die Rolle des *Product Owners* nennen. Diese Rolle ließe sich im Kontext der Projektgruppe nicht umsetzen, da keine Zuordnung einer einzelnen Person zur strategischen Produktentwicklung möglich ist. Diese Rolle wird vielmehr von der gesamten Gruppe unter Berücksichtigung der Wünsche der Betreuer getragen. Außerdem gibt es keine täglichen Meetings, wie Scrum es vorsieht.
Für den Ablauf der Projektgruppe wurden Sprints mit bestimmten Aufgaben festgelegt, die sich über einen Zeitraum von vier bis acht Wochen erstreckten. Desweiteren wurden Arbeitsgruppen für die Aufgaben festgelegt und den Aufgaben zugeordnet.
Um die Nachvollziehbarkeit der Abläufe zu gewährleisten, wurde ein *Product Backlog* gepflegt, welcher alle relevanten Daten mit einer genauen Beschreibung beinhaltet. Der Product Backlog dient somit zum einem der Übersicht und Organisation der Aufgaben für die Projektgruppenteilnehmer und zum anderen zur Kontrolle und Einschätzung der Arbeit für die Projektgruppenbetreuer. Die Rolle eines *Scrum Masters* wurde zu jedem Sprint neu vergeben. Auch diese Rolle wurde nur eingeschränkt umgesetzt. Die Arbeit des Scrum Masters in der Projektgruppe beschränkte sich auf die Moderation und Organisation der wöchentlichen Meetings. Außerdem blieb der Scrum Master auch weiterhin Bestandteil der Arbeitsgruppen der Projektgruppe.
Um die Selbstorganisation der Projektgruppenmitglieder zu fördern, wurde zu jedem Meeting ein *Daily Scrum* abgehalten. Jedes Projektgruppenmitglied beschrieb seine bisherigen Fortschritte in der jeweiligen Arbeitsgruppe und gab einen Ausblick, mit welcher Arbeit er sich bis zum nächsten Meeting beschäftigt. Daily Scrum hat hier ein probates und unkompliziertes Mittel zur Einschätzung der aktuellen Lage einzelner Mitglieder und der Arbeitsgruppen, der geleisteten Arbeit und aufgetretener Probleme geboten. Die Ergebnisse der wöchentlichen Meetings wurden in Form von Protokollen festgehalten. Diese Reports beinhalteten auch die Daily Scrums, sodass sich auch im Krankheitsfall jedes Projektgruppenmitglied über getroffene Entscheidungen nachträglich informieren konnte.
Scrum wurde im Laufe des zweiten Projektsemesters fast komplett verworfen und durch wöchentliche Präsentation der Arbeitsgruppenergebnisse und *Project Libre* ersetzt. Mit dieser Maßnahme wurde innerhalb der Projektgruppe ergebnisorientierter gearbeitet. Zudem wurde die Planung dabei genauer und effizienter.

Redmine. *Redmine*[6] ist ein web-basiertes Projektmanagement-Tool und dient der Projektgruppe hauptsächlich zur Projektverwaltung, der Speicherung von relevanten

[6] `http://www.redmine.org/`

Dokumenten und der Nutzung eines Wikis. Desweiteren stellt *Redmine* ein Ticketsystem zur Verwaltung von einzelnen Teilaufgaben mit Zuweisung zu Personen, ein Gantt-Diagramm, einen Kalender und ein Forum zur Verfügung. Das Ticketsystem und das Forum wurden nur vereinzelt genutzt, da der Kommunikationsweg per E-Mail und direkt in den wöchentlichen Terminen vorgezogen worden ist. Im Wiki wurden die Fortschritte der Projektgruppe anhand von Reports festgehalten, alle wichtigen Daten zu Terminen und Kontaktdaten aufgelistet und einzelne relevante Dokumente zur Verfügung gestellt.

5. Aufbau des Projektes und der Infrastruktur

Aus den Anforderungen für das Softwareprojekt hat sich ein Rahmen für die Umsetzung ergeben. In diesem Abschnitt werden die damit verbundenen Entscheidungen und Ziele beschrieben. Zunächst wird mit dem Projektnamen und der Lizenz die formelle Seite behandelt. Danach werden die benötigten Funktionalitäten mit einem Interaktionsdiagramm analysiert. Dieses dient als Grundlage für die Umsetzung der Programmlogik. Es wird die Middleware JavaEE beschrieben, die es ermöglicht, eine Webanwendung umzusetzen. Dabei wird ein zentraler Server eingesetzt, der die Ausführungsumgebung der Software darstellt und eine Datenbank bereitstellt.

5.1. Projektarchitektur

Eine der wichtigsten Aufgaben im zweiten Semester der Projektgruppe war die Planung und Festlegung der Projektarchitektur. Dazu mussten die relevanten Projektkomponenten identifiziert, und derer Schnittstellen und Abhängigkeiten definiert werden. In den folgenden Abschnitten werden sie im einzelnen vorgestellt.

5.1.1. Architekturdiagramm

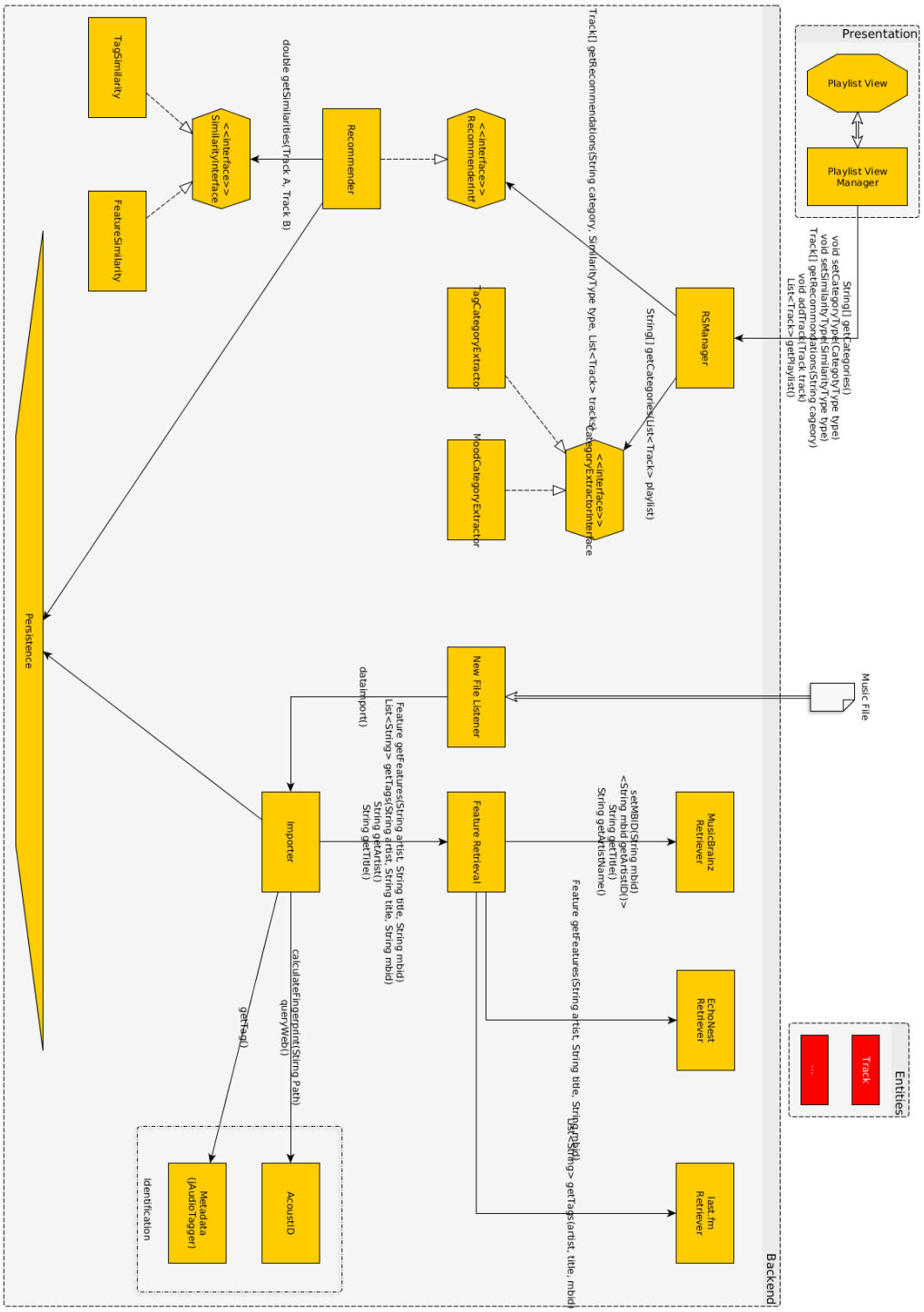

Abbildung 3: Übersicht des Architekturdiagramms

Das in Abbildung 3 dargestellte Diagramm beschreibt die realisierte Architektur. Wesentliche Bestandteile des dargestellten Diagramms sind Rechtecke und Sechsecke. Die Rechtecke beschreiben funktionelle Einheiten, die in Form von Klassen umgesetzt werden können. An Stellen, wo eine allgemeine Schnittstelle für unterschiedliche Implementierungen Sinn macht, beschreiben die Sechsecke Schnittstellen (Interfaces). Dazu kommen noch ein *Persistence*-Block für die Datenhaltung, ein Achteck für Seiten der Nutzeroberfläche und eine Musikdatei.

Die Verbindungen zwischen den Elementen haben unterschiedliche Semantiken. Der unterbrochene Pfeil zu einer Schnittstelle beschreibt wie in UML üblich, dass eine Klasse ein Interface implementiert. Die durchgängigen Pfeile bezeichnen eine „kennt"- bzw. „verwendet"-Beziehung. Sie sind an einigen Stellen mit einem oder mehreren vereinfachten Funktionsköpfen beschriftet. Dadurch wird der wesentliche Teil der Schnittstelle der Klasse und der Datenfluss definiert. Auf den Pfeilen stehen auch die Parameter für die Aufrufe der Funktionen.

Die aufgebaute Architektur unterteilt sich im Wesentlichen in drei Bestandteile: Die Nutzeroberfläche, das Empfehlungssystem und die Musikdaten-Aggregation. Es gibt aus verschiedenen teilbereichen Zugriffe auf die Datenhaltung. Diese beziehen sich im Wesentlichen auf *Track-Entities*, die ein Musikstück und die damit zusammenhängenden Daten beschreiben.

5.1.2. GUI

Eine vereinfachte Sicht auf die GUI ist in Abbildung 3 durch die Präsentationsschicht (*Presentation Layer*) gegeben, welche letztendlich durch eine Weboberfläche als Client realisiert wird. In der aktuellen Planung besteht sie aus einer einzigen *View* (*Playlist View*). Die Logik wird in der Komponente *Playlist View Manager* implementiert. Ihre primäre Aufgabe ist die Anzeige der aktuellen Wiedergabeliste sowie die Auswahl des nächsten Liedes. Hierfür wird zuvor die Art der Wiedergabelistengenerierung festgelegt, also tagbasiert oder stimmungsbasiert (*setRecommendationType*). Für die Darstellung zur Auswahl des nächsten Liedes werden bei jedem Auswahlvorgang die Kategorien und ihre Vorschläge benötigt, welche in der Backend-Schicht generiert werden (*getCategories*). Der Nutzer kann das nächste Lied für die Wiedergabeliste auswählen. Dazu übergibt der Playlist View Manager dem *Playlist Manager* eine Referenz auf das ausgewählte Lied (*setNextTrack*).

5.1.3. Empfehlungssystem

In Abbildung 4 ist der Bereich der Projektarchitektur zu sehen, welcher für die Realisierung des Empfehlungssystems zuständig ist. Zentraler Punkt des Empfehlungssystems ist der *Recommendationsystem Manager*, welcher alle Aufgaben des Empfehlungssystems verwaltet. Dazu kann auf die Schnittstelle *Category Extractor* oder *Recommender* zugegriffen werden. Der Category Extractor besitzt die Funktion *getCategories(Track[] Playlist)*, um die Kategorien für die nächsten Empfehlungen zu bekommen. Wenn der *Tag Category Extractor* angesprochen wird, bestehen die Kategorien aus den am häufigs-

ten vorkommenden Tags des letzten Liedes in der Wiedergabeliste. Der *Mood Category Extractor* hingegen gibt immer die selben n Kategorien zurück, die für die stimmungsbasierte Empfehlung genutzt werden. Jedoch könnte eine Erweiterung mehr Funktionalitäten nötig machen.

Die Schnittstelle Recommender wird genutzt, um mittels *getRecommendations(Category, Playlist)* die Empfehlungen für die nächsten Vorschläge zu erhalten. Dazu müssen vorher die Kategorien festgelegt worden sein, damit Recommender mit diesen Informationen die in Frage kommenden Lieder aus der Datenbank abfragen kann. Sind alle Lieder extrahiert, müssen die besten Vorschläge berechnet werden. Dazu soll das musikalisch passendste Lied zum letzten Lied der Wiedergabeliste bestimmt werden, um eine homogene Wiedergabeliste zu ermöglichen. Die Berechnung geschieht mittels der Schnittstelle *Similarity Extractor*, welche die Ähnlichkeit für Tags oder für Audiomerkmale berechnet. Ein Vorteil dieser Modellierung ist, dass Recommender die Ähnlichkeitsmaße verbinden kann, um bessere Ergebnisse zu erzielen, falls ein Ähnlichkeitsmaß unbefriedigende Ergebnisse zurück liefert.

5.1.4. Musik Importer

Die Aufgabe der Importer-Komponente ist das Verwalten des Datenimports in die Persistenz-Schicht. Sie wird durch einen *New File Listener* aufgerufen. Diese Komponente überwacht einen Ordner auf der Festplatte und reagiert auf neue Audiodateien. Für jede neue Audiodatei wird die *Importer*-Komponente aufgerufen. Diese nutzt den *Audio Identifier* um das Lied anhand vorhandener Metadaten oder den akustischen Fingerabdruck zu identifizieren und erstellt ein entsprechendes *Track*-Entity.

Wird das Lied erkannt, so kann die Audiodatei in einen zentralen Speicherordner geschoben und z. B. ihr akustischer Fingerabdruck als eindeutiger Dateiname gegeben werden. Mit dem Interpreten, Titel und evtl. auch der MBID, der ID von MusicBrainz, kann die *Feature Retrieval*-Komponente verschiedene Retrieval-Komponenten nutzen, um Merkmalen des Liedes abzurufen. Die entsprechenden Informationen werden dann von der *Feature Retrieval*-Komponente im *Track*-Entity gespeichert. Diese Entities sind im persistenten Speicher und können vom Empfehlungssystem genutzt werden.

5.2. Projektname musery

Ziel des Projektes ist es, eine Anwendung zu entwickeln, die geeignete Musikempfehlungen auf Basis verschiedener Daten generiert und Wiedergabelisten anhand von vorhandenen Musikstücken ermöglicht, die zur Entdeckung von neuer Musik geeignet sind. Um diese Punkte hervorzuheben, wurde der Name musery als Kombination der Begriffe „music“ und „discovery“ von der Projektgruppe ausgewählt.

5.3. Projektlizenz

Um eine Wiederverwendung der entwickelten Software nach Ablauf der Projektgruppe zu gewährleisten, hat sich die Projektgruppe dazu entschlossen, die MIT-Lizenz zu nutzen. Sie erlaubt die Verwendung von *musery* sowohl für Software, deren Quelltext frei

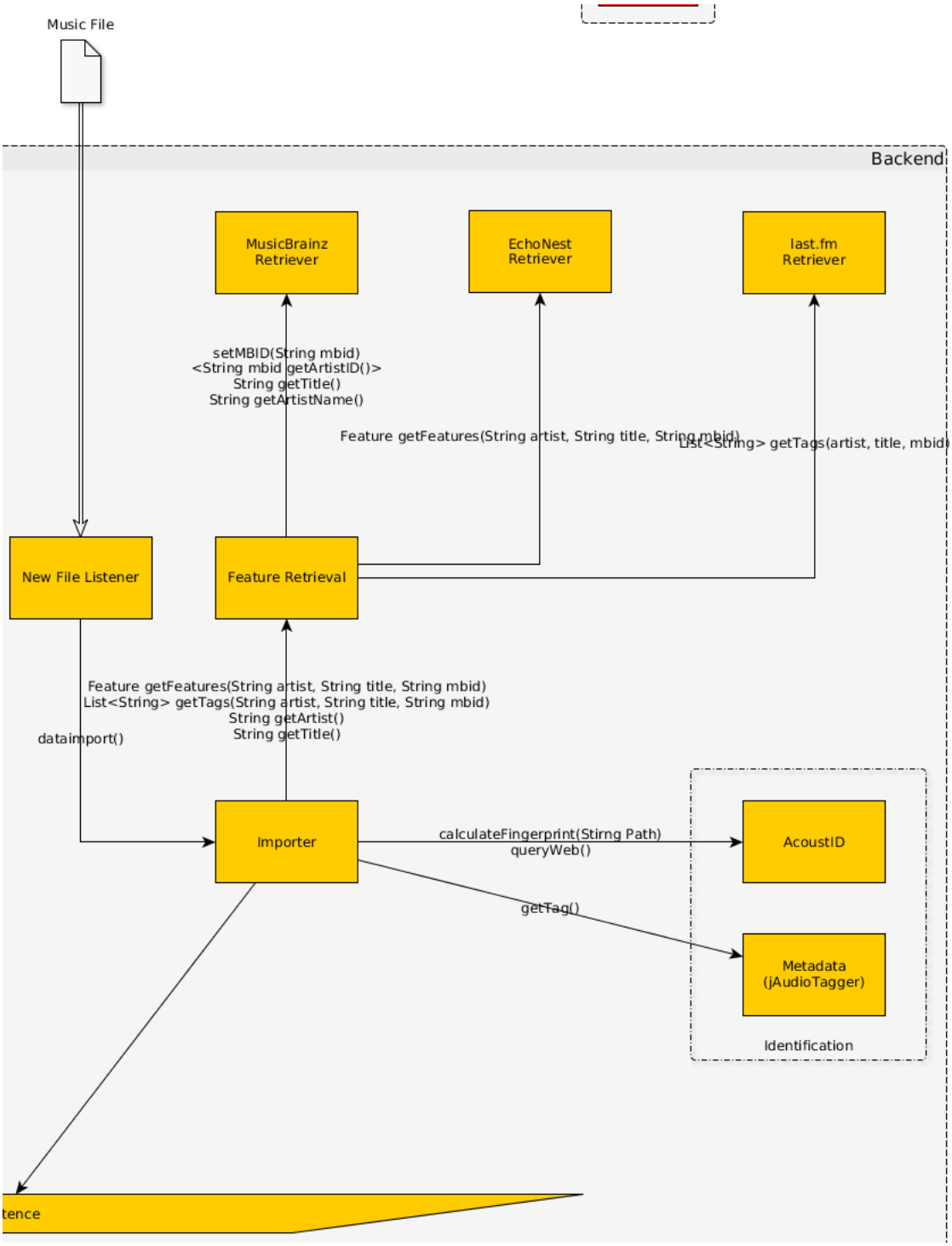

Abbildung 4: Komponenten des Import-Teils

einsehbar ist, als auch für Software, bei der dieser Quelltext nicht frei einsehbar ist. Mit der MIT-Lizenz kann musery für den eigenen Gebrauch modifiziert werden, weil sie die Wiederverwendung der unter ihr stehenden Software erlaubt. Sie erlaubt auch eine spätere Änderung der Lizenz. Die deutsche Übersetzung der MIT-Lizenz befindet sich im Anhang C.

5.4. Nutzerinteraktion

Zur Veranschaulichung der Interaktion des Nutzers mit der Anwendung wurde ein Diagramm erstellt (siehe Abbildung 5). So hat der Nutzer beim Aufrufen der Anwendung die Möglichkeit nach einem Lied zu suchen, welches das erste Lied seiner Wiedergabeliste werden soll. Basierend auf diesem Lied kann anschließend das Verfahren gewählt werden, womit weitere Vorschläge generiert werden. Zur Verfügung stehen ein tag-basiertes und ein stimmung-basiertes Empfehlungssystem. In beiden Empfehlungssystemen werden in mehreren Kategorien Lieder vorgeschlagen, aus denen der Nutzer eines auswählen und zur Wiedergabeliste hinzufügen kann. Anschließend kann die bislang erstellte Wiedergabeliste abgespielt, die Zusammenstellung beendet oder aber ein neues Lied aus den neu generierten Vorschlägen ausgewählt werden.

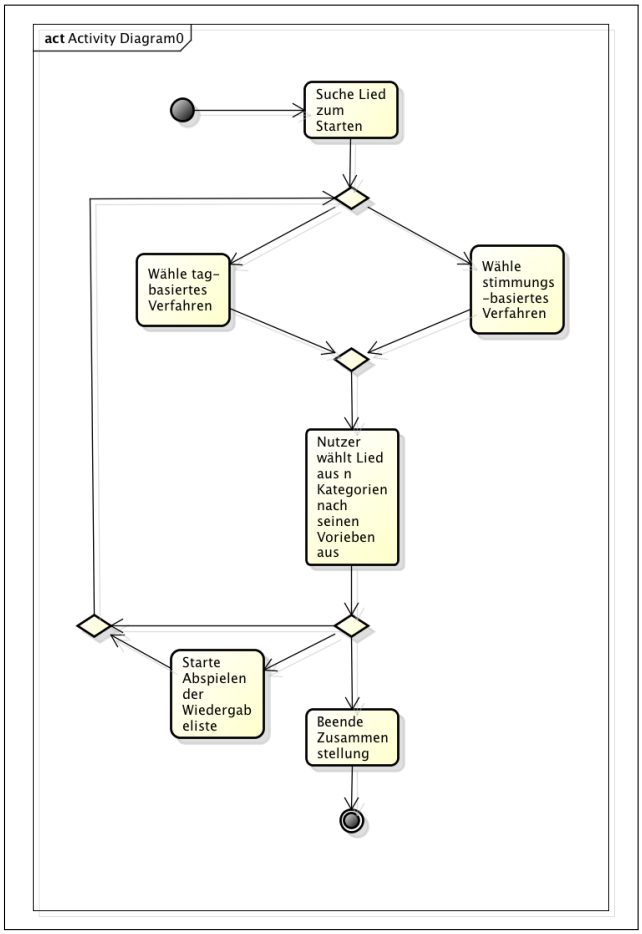

Abbildung 5: Diagramm der Interaktionen eines Nutzers mit der Anwendung

5.5. Java EE

In diesem Projekt wird *Java EE* (Abkürzung für „Java Platform, Enterprise Edition“) als Middleware eingesetzt. *Java EE* ist eine Sammlung von Spezifikationen, die ein Framework für verteilte, transaktionale und portable Applikationen bereitstellt. Dieses Framework bietet eine standardisierte Plattform für die Softwareentwicklung im großen Stil. Dabei werden unter anderem folgende zentrale Probleme und Aufgaben durch standardisierte Mechanismen unterstützt: Geschwindigkeit, Sicherheit, Ausfallsicherheit, Wartbarkeit, Nebenläufigkeit, Authentifizierung und Autorisierung, Transaktionen, Persistenz und Deployment.

Das *Java EE*-Framework wird in verschiedene logische Systeme, sogenannten *Container* unterteilt. Für dieses Projekt sind vor allem der *EJB*- und der *Web-Container* von zentraler Bedeutung.

Durch den *EJB-Container* wird eine Laufzeitumgebung für sog. *Enterprise JavaBeans* (kurz *EJB*s), oft auch *Session Beans* genannt, bereitgestellt. *EJB*s stellen nicht persistente, serverseitige Komponenten dar, welche die Logik eines Geschäftsprozesses

bzw. eines Anwendungsfalles oder eines Dienstes implementieren. Es gibt primär drei Ausprägungen von *EJB*s mit folgenden Eigenschaften:

- *Stateful Session Bean* - zustandsbehaftet
- *Stateless Session Bean* - zustandslos
- *Singleton Session Bean* - einmalig erzeugte, einzelne Instanz

Eine *Stateful Session Bean* bleibt dem Client, der sie angefordert hat, so lange zugeordnet, bis dieser sie wieder freigibt. Eine *Stateless Session Bean* bleibt dem Client lediglich für die Dauer eines einzelnen Aufrufs zugeordnet.

Neben dem *EJB-Container* wird in diesem Projekt der *Web-Container* genutzt. Dieser stellt u.a. eine Laufzeitumgebung für *Servlets* bereit. Hierdurch kann das *JavaServer Faces*-Framework eingebunden und für die Implementierung der Präsentationsschicht genutzt werden (Kapitel 3.1).

Das *Java EE*-Framework stellt für diese Projektgruppe ein leistungsstarkes Werkzeug für die Realisierung der Aufgaben dar und nimmt dabei viele architektonische Entscheidungen ab, die durch das Framework bereits durch Best-Practice-Ansätze gelöst sind. Durch die relativ strikten Vorgaben zur Implementierung gibt es insbesondere in mehrköpfigen Teams weniger Fehlerpotential und einen eindeutigeren Weg für die Integration von Funktionalitäten. Aus diesem Grund muss sich nicht jedes Mitgleid mit allen technischen Details auskennen. Es reicht wenn sich die einzelnen Mitglieder bzw. Unterteams auf die Schnittstellen festlegen und die Funktionalität ihrer Komponenten implementieren.

5.6. Zentraler Server

Um die Java EE Anwendung auszuführen, wird ein Anwendungsserver benötigt, der die entsprechenden Beans ausführt und in einen Kontext stellt. Dabei kann jeder Application Server verwendet werden, der die benötigten Java EE Standards implementiert. Es wurde sich darauf geeinigt, den *JBossAS 7* Server von *RedHat*[7] zu verwenden. Diesen wurde von der Projektgruppe primär getestet und bereits für unsere Entwicklung eingesetzt. Hierdurch ist sichergestellt, dass auch jeder Teilnehmer seinen aktuellen Code auch auf einer lokalen Instanz testen kann.

Es gibt aber auch Argumente, die für die Nutzung eines zentralen Servers sprechen. Ein über das Internet erreichbarer zentraler Server erlaubt es auch externen Personen einen Blick auf den aktuellen Fortschritt des Projektes zu werfen. Ebenfalls kann eine zentrale Instanz der *Hibernate*-Datenbank genutzt werden, um einen „Referenzdatensatz“ zu pflegen. Das Hibernate Framework bietet durch sogenanntes Object-Relational Mapping die Möglichkeit, den Zustand eines jeden Objektes in einer relationalen Datenbank zu speichern. Die aktuelle Musikdatenbank einer Instanz von musery kann so in einer SQL-Datenbank abgelegt werden. In der Projektgruppe wurde vereinbart, dass neue Methoden und Testdaten nur lokal getestet werden und auf dem zentralen Server

[7] `https://www.jboss.org`

nur ein fehlerfreier und definierter Satz an Liedern zur Verfügung steht. Diese Hibernate-Instanz kann in Form eines SQL-Dumps serialisiert werden, d. h. alle Daten als Folge von SQL-Befehlen per FTP heruntergeladen werden.[8]

5.7. Migration auf einen neuen Server im zweiten Semester

Bei dem von einem Projektgruppenteilnehmer gestellten Server handelte es sich um einen virtuellen Server mit einem 3,1GHz Intel Xeon E3-1220 Kern und 1GB RAM. Während intensiverer Arbeitsphasen im zweiten Semester wurde der Server von allen Projektgruppenteilnehmern verstärkt genutzt. Dies führte hin und wieder zu Leistungsengpässen, die ein kontinuierliches Testen des gerade entwickelten Codes erschwerten. Während des zweiten Semesters wurde aus diesem Grund ein neuer virtueller Server vom Lehrstuhl 13 unter der Adresse `http://cloudhost135.cs.tu-dortmund.de/` gestellt. Dieser verfügte über acht Kerne und 24GB RAM. Durch die vorhandenen Ressourcen konnte während der Entwicklung das Modell der kontinuierlichen Integration genutzt werden. Hierzu wurde die Kontinuierliche-Integrationssoftware Jenkins[9] verwendet, die mit jedem SVN-Commit musery neu gebaut und auf dem *JBoss* aufgespielt wurde. Bei Problemen mit dem Erstellen der Software wurde automatisch eine E-Mail an alle Teilnehmer verschickt und somit konnten diese sehr schnell behoben werden. Allerdings stellte sich heraus, dass es zu bestimmten Tageszeiten zu niedrigeren I/O-Leistungen des Servers kam. Selbst nach wiederholten Versuchen die Ursache des Leistungsproblems zu finden und zu beseitigen und dem Anheben des verfügbaren RAM auf 80GB, kam es immer noch zu erheblichen Verzögerungen bei der Nutzung von musery. Um letztendlich eine Durchführung der Evaluation zu ermöglichen, wurden kurz nach dem Ende der Entwicklungsphase der Server wieder auf den alten Server des Projektgruppenteilnehmers migriert.

5.8. Zentrale Konfiguration

Während der Entwicklung der ersten Prototypen wurden für den Zugang zu den verschiedenen Webdiensten API-Schlüssel und private Accounts der Teilnehmer verwendet. Um musery sinnvoll einzusetzen, ist es nötig, dass jede aktiv genutzte Instanz ihre eigenen API-Schlüssel hat. Nur so ist es möglich, die maximal erlaubten Abfragen pro Minute der verschiedenen Dienste umzusetzen. Eine wichtige Aufgabe bei der Integration der verschiedenen Prototypen war deshalb eine zentrale Konfigurationsmöglichkeit. Diese wurde nach dem Entwurfsmuster des Singletons implementiert. Es existiert immer nur eine Instanz der *Config*-Klasse[10], welche bei der ersten Nutzung zunächst vorhandene Werte aus der Datei *~/.musery/config.properties* lädt. Für alle Einstellungswerte, die nicht in dieser Datei überschrieben sind, werden Standardeinstellungen geladen. Jeder

[8] Da diese Aufgabe mit Hilfe eines privaten Servers eines Teilnehmers realisiert wurde, können die Zugangsdaten nicht öffentlich in diesem Endbericht dokumentiert werden.

[9] `http://jenkins-ci.org/`

[10] Package: de.musery.common

Einstellungswert hat einen Schlüssel, über den er von überall aus dem Projekt mithilfe der Config-Klasse geladen werden kann.

5.9. Zentrale Fehlerbehandlung

Die Fehlerbehandlung erfolgt durch eine *logger*-Klasse, die die ausgegebenen Meldungen je nach aufrufender Klasse und Dringlichkeit kategorisiert.

Zum einen bekommt jede Meldung einen *Marker*, der die Klasse enthält, in der die Fehlermeldung geworfen wurde. Zum anderen werden die Meldungen nach folgenden *Typen* gruppiert:

ERROR Kritische Fehlermeldungen

WARN Warnmeldungen

INFO einfache Informationen

DEBUG Meldungen, die der tieferen Analyse dienen

Über JBoss werden die Meldungen abgefangen und in der zentralen Log-Datei gespeichert. Auch die Selektion der ausgegebenen Meldungen kann über die JBoss Konfiguration angepasst werden.

5.10. Refactoring

Die Aufteilung des Projektes folgte zunächst kanonisch der angedachten Architektur. Dabei wurde musery wie folgt in sechs Module unterteilt:

- musery-ejb: die Hauptlogik des Projektes
- musery-ejb-persistence: Funktionen zur Persistierung der Entities
- musery-entity: die modellierten Entities
- musery-web: die Webinterface-Komponenten
- musery-common: sonstige Hilfsklassen
- musery-ear: Metaprojekt, welches lediglich alle anderen Komponenten zu einer Deployment-Datei zusammenfasst

Nach der *JavaEE-Spezifikation* wurden die wichtigsten Komponenten als Bean implementiert. Die Definition des lokalen *Business-Interface* jeder Bean erfolgte dabei parallel zu den eigentlichen Beans im jeweils selben Package. Allerdings haben sich aus dieser Aufteilung ein paar Probleme ergeben: In jedem Package befanden sich nun sehr viele Dateien. Neben den Beans existierten noch die dazugehörigen *Buisness-Interfaces* so wie eigene Interfaces. Ebenfalls bestand der Bedarf, einen *Remote-Client* zu implementieren, der nur die Interfaces kennen soll und nicht von allen anderen Komponenten des Projektes abhängen soll. Die Funktion des *Remote-Clients* wird genauer in Kapitel 5.11

beschrieben. Es wurde beschlossen, die Programmierarbeiten für einen Tag einzustellen und die Projektstruktur umfassend zu überarbeiten. Alle Inferfaces wurden in ein eigenes Untermodul namens *musery-ejb-interfaces* verschoben. In diesem befindet sich die selbe Package-Hierarchie wie in den anderen Module mit den zusätzlichen Packages *.ejb.interfaces* und *.ejb.interfaces.bi* für die Buisness-Interfaces. Ebenfalls wurde ein neues Modul für die Remote-Clients erstellt, welches nun nur noch die Interfaces und nicht den Rest des Projektes kennen muss. Daraus ergeben sich folgende acht Module des Projektes:

- musery-ejb: die Hauptlogik des Projektes
- musery-ejb-interfaces: Definition aller Schnittstellen
- musery-ejb-persistence: Funktionen zur Persistierung der Entities
- musery-ejb-remote-client: Remote-Client
- musery-entity: die modellierten Entities
- musery-web: die Webinterface-Komponenten
- musery-common: sonstige Hilfsklassen
- musery-ear: Metaprojekt, welches lediglich alle anderen Komponenten zu einer Deployment-Datei zusammenfasst

5.11. Remote Client JBoss

Neben dem Zugriffs über das Webinterface sollte auch eine weitere Möglichkeit des Aufrufs von musery implementiert werden. Dies ist speziell bei dem Datei-Import wichtig, da dieser je nach Anzahl der Lieder viele Minuten bis einige Stunden dauern kann. Ein *Remote Client* ist ein Kommandozeilenprogramm, welches einzelne Methoden einer Bean auf einem beliebigen Anwendungsserver aufrufen kann. Dies ermöglicht es den Import-Vorgang von Musik aus der Kommandozeile anzustoßen.

Listing 1: Aufruf des interaktiven Remote-Clients

```
java -jar musery-ejb-remote-client/target/musery-ejb-remote-
    client-jar-with-dependencies.jar FileListenerClient
    readDirectory
```

Listing 1 zeigt den Aufruf aus dem Skript *import.sh*, welches mit musery ausgeliefert wird, um den Importvorgang zu starten.

6. Datenbank und Verwaltung

Die persistente Haltung aller Daten geschieht als Entities mithilfe der Java-Persistance-API. Diese im JavaEE-Standard definierte Schnittelle ermöglicht es, die Metadaten der

importieren Lieder und die Klassifizierung der Stimmungen dauerhaft zu speichern. Im folgenden Kapitel werden verschiedene Teile der Datenwahl, -eingabe und -ausgabe beschrieben.

6.1. Import

musery importiert die benötigten Daten der Lieder, welche von den Empfehlungssystemen genutzt werden, in eine *mySQL*-Datenbank. Als Quelle werden ausschließlich Audio-Dateien der Typen „mp3“, „ogg“ und „flac“ unterstützt, welche lokal auf dem Server verfügbar sein müssen.

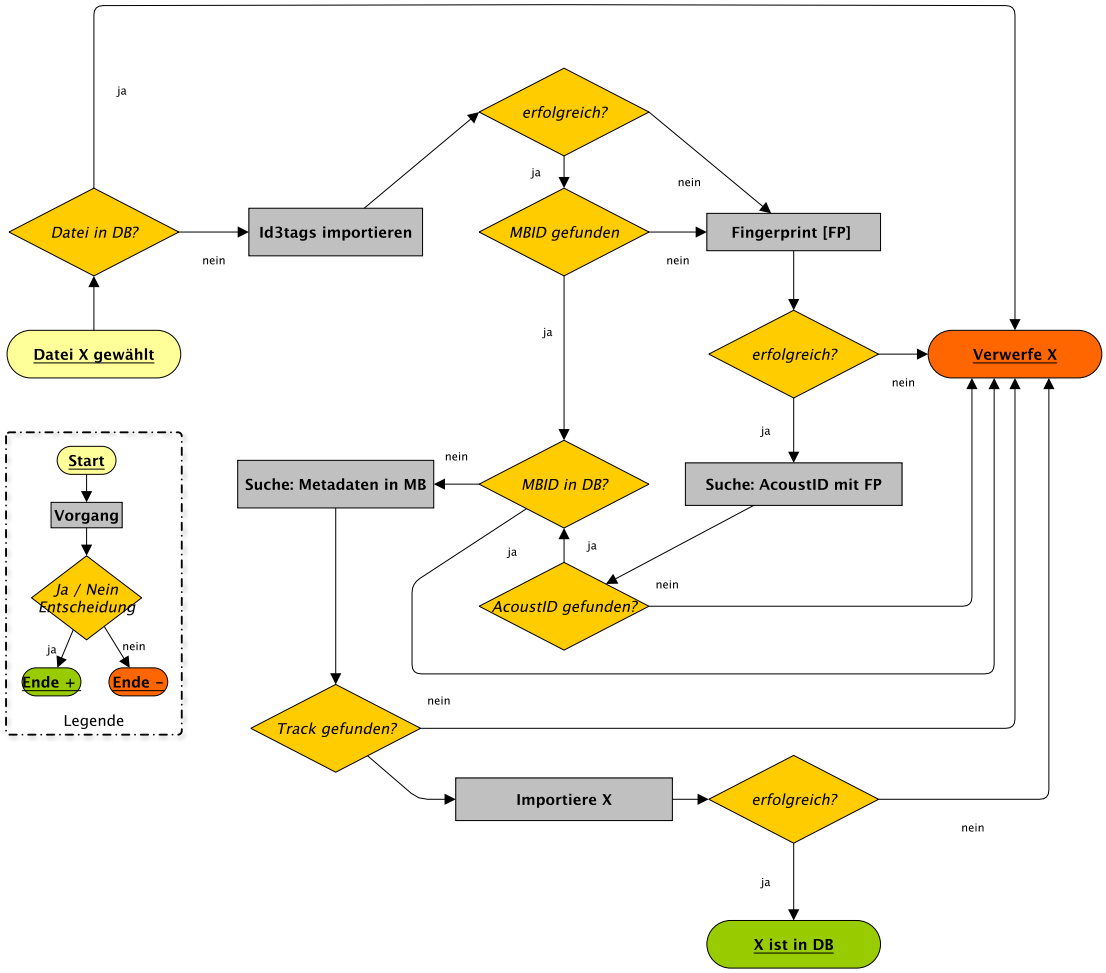

Abbildung 6: File Import / Identifizierung

6.1.1. File Import

Ein Importvorgang beginnt immer mit dem Auslesen von gegebenen Audiodateien. Der Lesevorgang wird durch das Kopieren der betreffenden Dateien in das Import-Verzeichniss `import`, unterhalb des musery-Ordners gestartet. Möglich wird dies durch eine kontinuierliche Überwachung des Verzeichnisses während einer laufenden musery-Instanz. Sobald sich der Inhalt des Import-Ordners ändert, durchsucht der dafür zuständige *FileListener* ihn rekursiv und fügt die gefundenen Audio-Dateien zu einer *Warteschlange* hinzu.

Diese wird dann abgearbeitet, bis alle Dateien importiert wurden oder gegebenenfalls bis der Import abgebrochen wurde, die Warteschlange sich also in jedem Fall wieder geleert hat.

Dabei werden bereits importierte Dateien automatisch anhand ihres, in der Datenbank gespeicherten, *Hashwertes* erkannt. Dadurch lassen sich redundante Imports von Dateien mit identischem Inhalt ausschließen und mehrfache Einträge in unsere Datenbank vermeiden.

Notwendig wird dies, da ein mehrfaches Auftreten von identischen Tracks die Konsistenz unserer Datenbank gefährden würde. In so einem Fall wären alle folgenden Empfehlungen und entsprechende Berechnungen von Zwischenergebnissen verfälscht und somit höchstwahrscheinlich unbrauchbar.

Da die *Hashwerte* sich jedoch durch Faktoren wie Dateityp, Bitrate, Qualität der Aufnahme u. v. m. unterscheiden, sind weitere Differenzierungen nötig, um wirklich erkennen zu können, ob zwei Dateien den gleichen Track repräsentieren oder nicht.

6.1.2. Identifizierung

Bereits im Zwischenbericht [13] wurde die zentrale Bedeutung der *MBID*[11] verdeutlicht. Jedes Musikstück in unserer Datenbank muss durch eine eindeutige MBID identfiziert werden können, so dass jedes Stück genau eine MBID hat und umgekehrt. Dementsprechend gilt ein Stück als vorhanden, wenn seine MBID bereits in unserer Datenbank existiert. In diesem Fall wird der Import abgebrochen.

Wir beziehen die MBIDs durch implizite *Metatags* oder über die Generierung und Abfrage von *Fingerprints*, der jeweiligen Dateien.

Dieser Vorgang ist in Abbildung 6 als Entscheidungs-Diagramm dargestellt. Im Falle von Misserfolgen, bei kritischen Online-Abfragen, welche MBID, Titel oder Interpreten betreffen, wird der Track als „nicht identifiziert“ betrachtet und der Import abgebrochen. Ebenso wird für andere kritische Fehler, z. B. in Persistenz und Multithreading, die nach der Identifizierung vorkommen können, der Importvorgang abgebrochen.

Anmerkung Im Unterschied zur Fassung im Zwischenbericht wurde in der momentanen Version eine alternative Suche in MusicBrainz über Titel und Interpreten nicht imple-

[11] MusicBrainz Identificator, von MusicBrainz generierte, alphanumerische Zeichensequenz, die ein Lied eindeutig identifiziert

mentiert. Es werden nur noch Dateien importiert, aus deren *id3Tags* oder Fingerprints eine MBID extrahiert werden konnte. Alle anderen werden somit ignoriert.

Begründet liegt diese Entscheidung darin, dass wir keinen unvollständigen id3Tags mit ungeklärter Herkunft und Schreibweise vertrauen. Da wir jedoch feststellen mussten, dass jede erfolgreiche Fingerprint-Abfrage einen MBID-Eintrag hat und wir MusicBrainz sehr wohl vertrauen können, kann auf die Nutzung von nicht-MBID relevanten Informationen in den dateiimpliziten Tags verzichtet werden.

Ein Teil der Metatag-Abfrage wird somit durch die Identifizierung abgedeckt, da Titel und Interpret minimal vorhanden sein müssen.

6.1.3. Metatags und Audio-Merkmale

Ein Track unserer Datenbank ist nur gepaart mit den entsprechenden Informationen von Nutzen.

Zum einen haben wir Coverbilder zu den jeweiligen Empfehlungen in die grafische Benutzeroberfläche implementiert. Für die Bereitstellung werden Informationen über das Album eines Tracks benötigt (siehe Unterunterabschnitt 6.1.4), solche werden aber durch die Abfrage bei MusicBrainz, während einer Identifizierung, auch direkt zurückgegeben. Daher können die zu einem Track gehörigen Alben im Anschluss an die Identifizierung direkt erzeugt und gespeichert werden.

Zum anderen brauchen wir für unsere Empfehlungssysteme erweiterte Informationen, um adäquate Vorschläge zu generieren. Diese beziehen wir von zwei weiteren Online-Quellen: Benutzer-Tags via *last.fm*[12] und eine Auswahl von Audio-Merkmalen („danceability“, „valence“, „energy“, „key“, „mode“, „tempo“, „loudness“) via *EchoNest*[13]. Die Abfragen werden zuerst über die MBID getätigt. Sollte dies jedoch nicht erfolgreich sein, wird ersatzweise über Interpret und Titel gesucht.

Für den reinen Importvorgang sind diese Informationen optional, d. h. sollte eine Abfrage fehlschlagen oder kein Ergebniss liefern, ist der Importvorgang des betreffenden Tracks davon unberührt.

Anmerkung Die abgerufenen Audio-Merkmale von EchoNest werden beim Import direkt weiterverarbeitet. Sie werden je nach deren Ausprägungen in eine von drei Emotionen kategorisiert („happy“, „sad“, „angry“).

6.1.4. Cover

Wie bereits in 6.1.3 beschrieben, erhält jedes Musikstück zur graphischen Präsentation im Empfehlungssystem ein Coverbild. Dazu wird die Datenbank von *coverartarchive.org*[14] verwendet, um ein Coverbild für das jeweilige Lied herunterzuladen. Die Suche wird über die MBID des Songs gestartet und liefert - falls vorhanden - ein Coverbild zurück, welches auf dem Server abgespeichert wird. Der Pfad zum Coverbild wird zudem

[12] `http://www.lastfm.com`
[13] `http://the.echonest.com`
[14] `http://www.coverartarchive.org`

in der Datenbank hinterlegt. Für jedes Coverbild wird außerdem noch eine verkleinerte Version (200x200px) - sogenanntes Thumbnail - erzeugt, um eine einheitliche Anzeige in der Benutzeroberfläche garantieren zu können. Falls kein geeignetes Coverbild auf *coverartarchive.org* gefunden werden sollte, wird ein Standardbild verwendet, welches statt des Coverbildes angezeigt wird.

6.1.5. Optimierung

Da sich der Import mehrerer Quellmedien (z. B. Festplatte, Internet) bedient, die unterschiedlich schnell angesprochen werden können, kann es bei einem Importvorgang vorkommen, dass auf ein Medium gewartet wird obwohl dafür theoretisch keine Notwendigkeit bestände (z. B. wegen Datenunabhängigkeit). Somit kommt es bei einer sequentiellen Verarbeitung der Tracks zu unnötigen Leerlaufzeiten. Um den Import zu beschleunigen, werden die Dateien parallel importiert. So können bereits andere Dateien importiert werden, während für die aktuelle auf z. B. Identifikation o. ä. (siehe Abbildung 6) gewartet wird.

Dazu wird ein thread-basiertes System genutzt, welches konkurrierende Zugriffe auf die Warteschlange (siehe Unterunterabschnitt 6.1.1) verwaltet.

Somit wird der (schnell reagierende) lokale Speicher weiter genutzt (indem Dateien importiert werden) und die langsamer laufenden Online-Abfragen blockieren diese nicht.

Ein Grund für die eben genannten Verzögerungen bei den Online-Abfragen ist die Bandbreite der Verbindungen.

Aber auch die jeweiligen Online-Dienste beschränken ihre Kapazitäten indem sie, bestimmte *Limits* an Abfragen über einen gewissen Zeitraum setzen.

Aus diesem Grund überwacht ein *Ratelimiter* die betreffenden Abfrage-Komponenten (AcoustId, MusicBrainz, EchoNest und Last.fm), um deren Frequenz zu kontrollieren. Dazu verwaltet dieser ein festes Limit für jede Komponente und lässt diese bei drohender Überschreitung warten.

Die jeweiligen *maximalen* Limits sind wie folgt von den jewiligen Anbietern gegeben:

AcoustId Unbekannt

MusicBrainz 50/s, jedoch nur max. 60/min pro ip

EchoNest 120/s

Last.fm 300/s

Per Konfigurationsdatei können diese Limits beliebig angepasst werden. Auch die jeweiligen Login-Informationen sind dort modifizierbar.

6.2. Datenbasis

Als Basis für unser Empfehlungssystem wurden aus einer zur Verfügung stehenden Anzahl von 20000 Tracks, 1000 zufällig ausgewählt. Dabei wurde auf eine möglichst vielfältige Musikauswahl geachtet, sowie auf die Vermeidung von Titeln deren Audio-Merkmale und Benutzer-Tags vermutlich (durch mangelnde Popularität o. ä.) nicht abrufbar wären.

Die ausgewählten Tracks wurden dann importiert indem sie in das auf unserem Server liegende, Importverzeichnis übertragen wurden.

6.3. Persistierung der Entities

In diesem Projekt wird die Datenbank nicht direkt über SQL-Befehle erstellt, sondern mit Hilfe der Java Persistence API anhand der Entities entworfen und aktualisiert. Die Struktur der Datenbank soll für die Entwickler verborgen bleiben. Hinzufügen, Aktualisieren, Löschen und weitere Aktionen, die auf den Daten ausgeführt werden können, lassen sich durch JPA-Queries realisieren. Deswegen wird nur die Struktur der Entities in diesem Kapitel ausführlich beschrieben.

Entities, über die man die Informationen speichern oder verarbeiten kann In diesem Projekt werden 17 Entities implementiert. JPA-Query ist die Abkürzung für Java Persistence API Query. Die Schnittstelle ist javax.persistence.Query. Mit dieser Schnittstelle kann man die Daten in der Datenbank über die Entities verarbeiten. Die Syntax des JPA-Query ist ähnlich der Syntax von SQL.

6.3.1. Überblick über die Entities

Die Beziehungen zwischen den Entities werden in Abbildung 7 dargestellt. Mit Hilfe der Entities werden Informationen über Lieder, Benutzer und Musikeigenschaften verarbeitet.

Um einen Überblick über die Entities zu bekommen, werden die Entities und ihre entsprechenden Inhalte im Folgenden aufgelistet:

- Benutzer und ihre Eigenschaften: User, ListeningHabit
- Lieder und Alben: Track, Artist, Genre, TagFrequency, Tag, Release
- Musikstimmung (Emotion): Feature, FeatureIntensity, EmotionIntensity, Emotion
- Dateien: File, FileTrack, FileCoverImage

Eine Entity kann mehrere Beziehungsattribute (Relation Attributes) und sonstige Attribute enthalten. Die Beziehungsattribute bedeuten, dass dieses Attribut einen Typ hat, welcher selbst ein Entity ist. Zwischen den beiden besteht immer eine Beziehung (1 zu 1, oder 1 zu n, oder 0 zu n, u. s. w.). Die Entities ohne Beziehungsattribute (z. B. Artist, Genre) werden in diesem Kapitel nicht explizit beschrieben. Die Informationen zu solchen Entities findet man in der entsprechenden Zelle der Tabelle, die zu Artist befinden sich zum Beispiel in Tabelle 6.3.2 unter dem Attribut „Artist“.

6.3.2. Details zu Entities

Entity Track
Entity Track speichert die Informationen eines Liedes. Siehe Entity Track in Abbildung 7 und Beziehungsattribute in Tabelle 2.

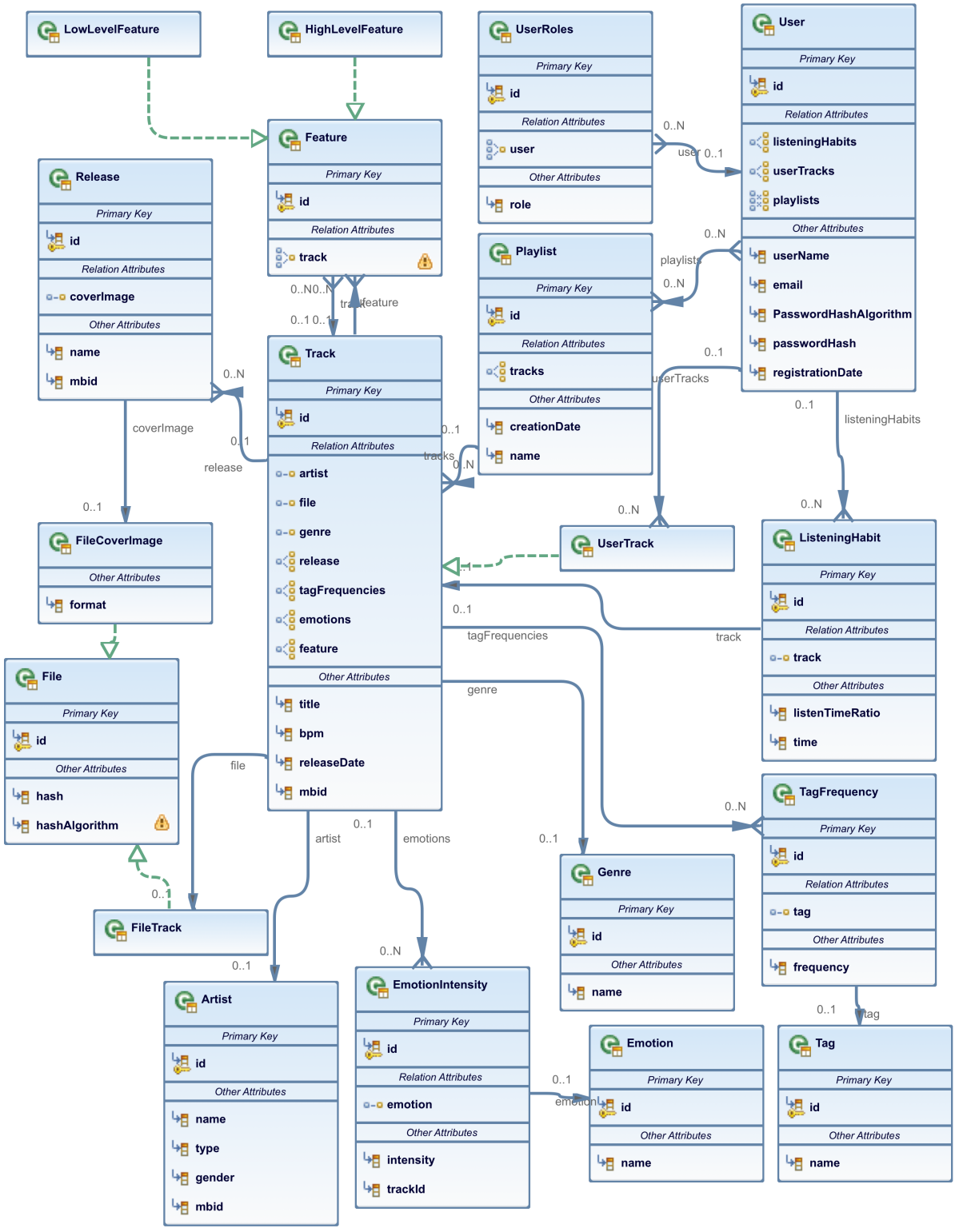

Abbildung 7: Überblick über die Beziehungen zwischen den Entities

Attributtyp	Beziehung	Funktion
Artist	1 : 1	Informationen über den Künstler des Liedes. Jeder Künstler hat Künstler-ID, Namen, Typ, Geschlecht und MBID.
File	1 : 1	Datei des Liedes auf dem System.
Genre	1 : 1	Jedes Lied gehört zu einem Genre. Jedes Genre enthält die Genre-ID und den Namen des Genres.
Release	1 : n	Ein Lied kann mehrere Releases haben, weil ein Lied mehrmals gemischt oder von den anderen gecovert werden kann.
TagFrequency	1 : n	Lieder können mehrere Tags enthalten. Dasselbe Tag kann von mehreren Benutzern gesetzt werden. Die Anzahl der Benutzer, die den Tag für das Lied gesetzt haben, wird in dem Attribut frequency (Häufigkeit) gespeichert.
Emotion	1 : n	Emotionen des Liedes.
Feature	1 : n	Features sind die Daten zum Analysieren des Liedes. Dazu gehören die High-Level-Features und Low-Level-Features.

Tabelle 2: Beziehungsattribute zu Entity Track

Außer Beziehungsattributen hat das Entity Track noch die folgenden Attribute (Siehe Tabelle 3):

Entity Release

Ein Track (Lied) kann je nach Release-Datum oder Release-ID mehrere Releases haben. Das Entity Release (Siehe Abbildung 7) hat ein Beziehungsattribut (Tabelle 4) und zwei sonstige Attribute (Tabelle 5).

Entity TagFrequency

Um die Häufigkeit eines Tags zu einem Lied zu speichern, wird das Entity TagFrequency gebraucht. Dieses hat ein Beziehungsattribut (Tabelle 6) und ein sonstiges Attribut (Tabelle 7), siehe auch Entity TagFrequency in Abbildung 7.

Attribut-Name	Typ	Funktion
title	String	Titel des Liedes.
bpm	double	Beats per Minute. Die Geschwindigkeit des Liedes wird in diesem Attribut gespeichert.
releaseDate	Date	Veröffentlichungsdatum des Liedes.
mbid	String	Die MBID ist der Fingerprint eines Liedes. MBID ist die Abkürzung für MusicBrainz Indentifiers.

Tabelle 3: Sonstige Attribute zu dem Entity Track

Typ	Beziehung	Funktion
FileCoverImage	1 : 1	Jedes Release kann ein Bild haben. Das Bild kann als Titelbild des Liedes angezeigt werden.

Tabelle 4: Beziehungsattribute zu Entity Release

Attribut-Name	Typ	Funktion
name	String	Name des Releases.
mbid	String	Fingerprint des Releases.

Tabelle 5: Sonstige Attribute zu Entity Release

Typ	Beziehung	Funktion
Tag	1 : 1	TagFrequency speichert das Tag und seine Häufigkeit. Die Informationen zu dem Tag werden in Entity Tag gespeichert. Das Tag hat das Attribut Name.

Tabelle 6: Beziehungsattributte zu Entity TagFrequency

Attribut-Name	Typ	Funktion
frequency	int	Die Häufigkeit des Tags.

Tabelle 7: Sonstige Attribute zu Entity TagFrequency

Entity EmotionIntensity
Das Entity EmotionIntensity speichert die Emotion und die Stärke der Emotion für ein Lied, siehe hierzu das Entity EmotionIntensity in Abbildung 7. Die Beziehungsattribute werden in der Tabelle 8 erklärt und sonstige Attribute werden in der Tabelle 9 erläutert.

Typ	Beziehung	Funktion
Emotion	1 : 1	TagFrequency speichert Tag und Häufigkeit. Die Informationen zu dem Tag werden in Entity Tag gespeichert. Dieser Tag hat das Attribut Name.

Tabelle 8: Beziehungsattribute zu Entity EmotionIntensity

6.4. Streaming

Damit dem Benutzer nicht nur Coverbilder, Tags und Moods zu den einzelnen Musikstücken präsentiert werden, besteht die Möglichkeit jedes zur Auswahl stehende Lied auch abspielen zu lassen. Dafür wurde für jedes Lied in der Datenbank die dazugehörige Audiodatei auf dem Server abgespeichert. Über die MBID des angeforderten Musikstücks wird der jeweilige Pfad zur Audiodatei abgefragt und an den Browser des Benutzers (Client) übertragen. Unmittelbar nach Beginn der Datenübertragung steht die Audiodatei dem Benutzer - noch während des Ladens - für die Wiedergabe zur Verfügung. Insbesondere das Vor- und Zurückspulen des Musikstücks ist somit möglich und verkürzt die Wartezeit erheblich. Diese Art der Dateiverfügbarkeit über das Internet nennt sich On-Demand-Streaming und eignet sich hervorragend für Audio-Visuelle Datenübertragung.

Attribut-Name	Typ	Funktion
intensity	double	Die Stärke der Emotion

Tabelle 9: Sonstige Attribute zu Entity EmotionIntensity

7. Empfehlungssysteme

Kernaufgabe der Projektgruppe war es, zwei Empfehlungssysteme zu implementieren, um dem Nutzer mittels zwei Ansätzen Musikvorschläge anzubieten. Ein Empfehlungssystem nutzt hierfür *Last.fm*-Tags und das andere Empfehlungssystem teilt Lieder in Stimmungen ein und berechnet basierend auf diesen neue Vorschläge. Im folgenden Kapitel soll nun die Entwicklung der zwei Empfehlungssysteme näher erläutert werden.

7.1. Architektur des Empfehlungssystems

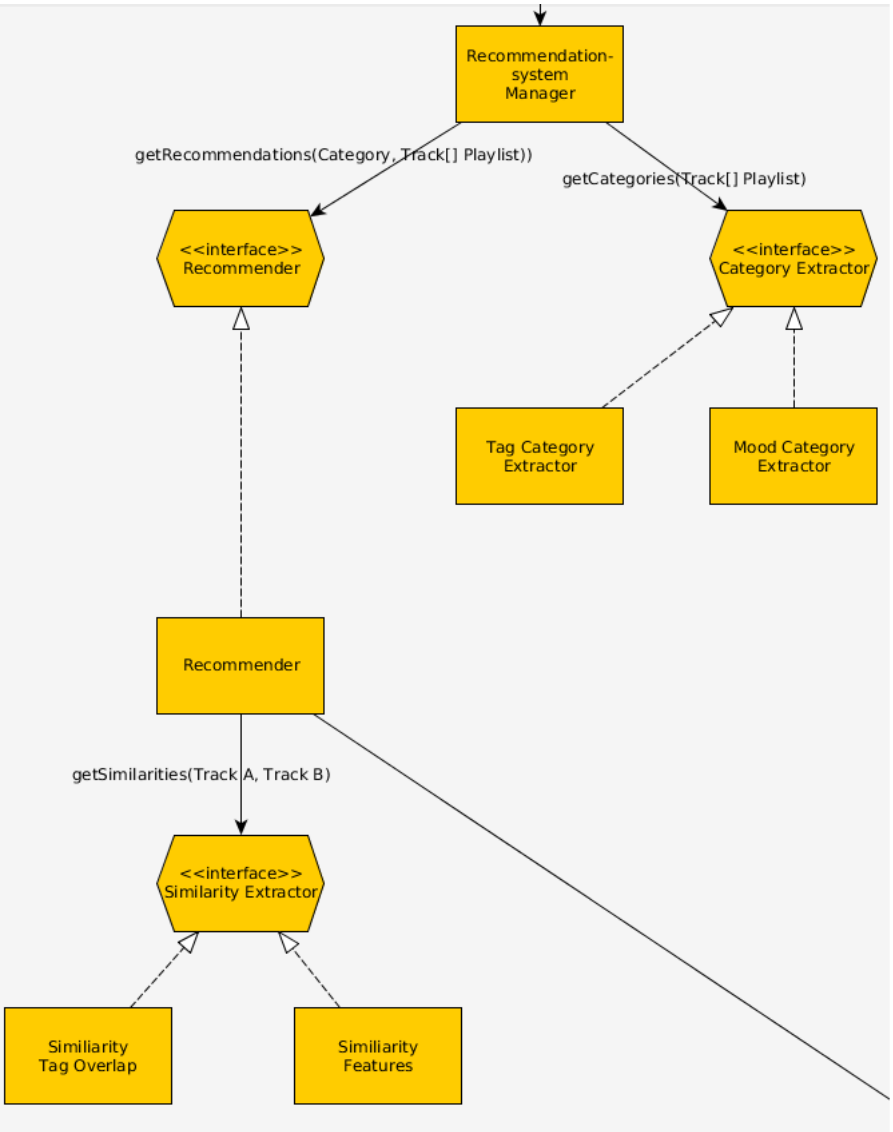

Abbildung 8: Komponenten des Empfehlungssystems

In Abbildung 8 ist der Bereich der Projektarchitektur zu sehen, welcher für die Realisierung des Empfehlungssystems zuständig ist. Zentraler Punkt des Empfehlungssystems ist der Recommendationsystem Manager, welcher alle Aufgaben des Empfehlungssystems verwaltet. Dazu kann auf die Schnittstelle *Category Extractor* oder *Recommender* zugegriffen werden. Der *Category Extractor* besitzt die Funktion *getCategories(Track[] Playlist)*, um die Kategorien für die nächsten Empfehlungen zu erhalten. Wenn der *Tag Category Extractor* angesprochen wird, bestehen die Kategorien aus den am häufigsten vorkommenden Tags des letzten Liedes in der Wiedergabeliste. Der *Mood Category*

Extractor hingegen gibt immer die selben n Kategorien zurück, die für die stimmungsbasierte Empfehlung genutzt werden.

Die eigens dafür definierte Schnittstelle *Recommender* wird genutzt, um mittels *getRecommendations(Category, Playlist)* die Empfehlungen für die nächsten Vorschläge zu erhalten. Dazu muss vorher die Kategorie-Art festgelegt worden sein, damit der *Recommender* mit diesen Informationen, die in Frage kommenden Lieder aus der Datenbank abfragen kann. Sind alle Lieder gefunden, müssen die besten Vorschläge berechnet werden. Dazu soll das musikalisch passendste Lied zum letzten Lied der Wiedergabeliste bestimmt werden, um eine homogene Wiedergabeliste zu ermöglichen. Die Berechnung geschieht mittels der Schnittstelle *Similarity Extractor*, welche die Ähnlichkeit, basierend auf Tags oder Audiomerkmalen, berechnen kann. Ein Vorteil dieser Modellierung ist, dass der *Recommender* die Ähnlichkeitsmaße verbinden kann, um bessere Ergebnisse zu erzielen, falls ein Ähnlichkeitsmaß unbefriedigende Ergebnisse zurückliefert. Eine Beschreibung der Ähnlichkeitsberechnung basierend auf Tags findet sich in Kapitel 7.2.3 und basierend auf Stimmungen in Kapitel 7.3.5.

7.2. Tag-basiertes Empfehlungssystem

Im Folgenden wird das Empfehlungssystem basierend auf den Tags von Last.fm vorgestellt. Dazu wurde die Eignung der Tags von Last.fm analysiert (siehe Kapitel 7.2.1), ein Prototyp für Empfehlungen mittels der Popularität von Tags entwickelt (siehe Kapitel 7.2.2), eine Ähnlichkeit von Liedern basierend auf Tag-Vektoren definiert (siehe Kapitel 7.2.3) und diese anschließend in einem modifizierten Prototypen implementiert (siehe Kapitel 7.2.4). Diese Vorarbeiten geschahen bereits im Laufe des ersten Semesters und wurden anschließend im zweiten Semester genutzt, um das System in das Projekt zu integrieren. Dabei musste die Implementierung angepasst werden und aufgrund einiger Geschwindigkeitsprobleme überarbeitet werden (siehe Kapitel 7.4).

7.2.1. Analyse von Last.fm-Tags für die Eignung als Kategorie-Bezeichner

Bevor entschieden wurde, welche möglichen Herangehensweisen zur Musikempfehlung verwendet werden könnten, mussten entsprechende Machbarkeitsanalysen vorgenommen werden. Da eine mögliche Option zur Ähnlichkeitsanalyse auf dem Vergleich von Liedtags basiert, und nicht von vornherein klar war, ob die Qualität der zur Verfügung stehenden Tags ausreicht, um eine Ähnlichkeitsanalyse vornehmen zu können, mussten diese vorher überprüft werden. Daher wurden stichprobenartig Lieder aus der last.fm-Datenbank[15] auf die Qualität ihrer Tags überprüft. Die Daten wurden in einer Tabelle festgehalten[16] und nach ihrer Häufigkeit sortiert. So erhielt man einen Überblick über die wichtigsten Tags für die ausgewählten Lieder und konnte evaluieren, ob diese zur Weiterverarbeitung geeignet sind. Das Ergebnis dieser Untersuchung wurde diskutiert und vorgestellt. Die Daten zeigen eine hohe Tag-Diversität pro Lied, d. h. dass eine große Anzahl an Tags pro Lied zur Verfügung steht, die aber durch die jeweiligen Tag-Bewertungen ihre Relevanz

[15] http://www.lastfm.de/api
[16] SVN-Repository /code/LastFM/CodeForDownloadingTags

zu den dazugehörigen Musikstücks ausdrücken. Je höher ein Tag bewertet wurde, desto relevanter ist es für den jeweiligen Lied. Deswegen wurde für die weitere Betrachtung eine Datenbereinigung vorgenommen, indem nur die zehn bestbewerteten Tags pro Lied herausgefiltert wurden. So haben wir eine bessere Tag-Beschreibung der Lieder erhalten und dadurch eine weitere Untersuchung ermöglicht. Diese bereinigte Version der Tags führte dazu, dass das Verfahren des Tag-basierten Empfehlungssystems als mögliche Herangehensweise weiterhin als Option bestehen blieb.

7.2.2. Empfehlung anhand der Popularität

Es wurde ein Prototyp[17] des Empfehlungssystems implementiert, der das Ziel hat, die geplante Empfehlungsstrategie basierend auf Tags zu evaluieren. Konkret soll einerseits die Eignung von Tags als Kategorien-Bezeichner getestet und andererseits festgestellt werden, ob die geplante interaktive Wiedergabelisten-Generierung für Benutzer ansprechend sein kann.

Der Prototyp setzt das Empfehlungssystem im Rahmen einer Konsolenanwendung um und arbeitet ausschließlich mit Daten von last.fm. Die last.fm-Daten sind sehr umfangreich und leicht abzurufen, daher können viele verschiedene Musikstücke in das System eingebunden werden. Die folgenden Punkte fassen die Funktionsweise kurz zusammen:

1. Benutzereingabe des ersten Musikstücks **m**
2. Auswahl von Kategorien aus Tags von **m**
3. Empfehlung mehrerer Musikstücke je Kategorie
4. Benutzer wählt neues **m** aus den Empfehlungen. Weiter bei 2.

Für die Kategorien der Empfehlungen werden die häufigsten Tags eines Musikstücks extrahiert. Um die Empfehlungen innerhalb der Kategorien zu geben, werden diejenigen Musikstücke bestimmt, auf die der Kategorie-Tag am häufigsten angewendet wurde. Diese werden dem Benutzer in der Form einer Tabelle aus Musiktiteln mit Buchstaben präsentiert. Durch die Eingabe eines Buchstabens wählt der Benutzer dann den nächsten Titel. Daraufhin werden neue Empfehlungen auf der Basis des gewählten Titels gemacht. In Abbildung 9 ist die Ausgabe des Programms für eine Ausführung dargestellt.

[17] SVN-Repository /code/tag-based-recommendation

```
Please enter song name as seed and press enter
nirvana

Nirvana - Smells Like Teen Spirit
Grunge      a. Nirvana - Smells Like T  b. Nirvana - Come as You A
rock        c. OneRepublic - Counting   d. Imagine Dragons - Radio
alternative e. AWOLNATION - Sail        f. Imagine Dragons - It's
Nirvana     g. Nirvana - Come as You    h. Nirvana - Smells Like T
90s         i. Spin Doctors - Two Pric  j. Semisonic - Closing Ti
Choose letter and press enter: i

Spin Doctors - Two Princes
90s         a. Spin Doctors - Two Pric  b. Semisonic - Closing Tim
rock        c. OneRepublic - Counting   d. Imagine Dragons - Radio
alternative e. AWOLNATION - Sail        f. Imagine Dragons - It's
alt. rock   g. 30 Seconds to Mars - Up  h. Muse - Supermassive Bla
pop         i. Miley Cyrus - Wrecking   j. Katy Perry - Roar
Choose letter and press enter:
```

Abbildung 9: Beispielhafte Ausführung des Prototypen mit Eingabe des Starttitels und zwei Empfehlungsschritten

Der Prototyp wurde innerhalb der Gruppe angewendet, um festzustellen, ob die gewünschten Ziele erreicht werden konnten. Es wurde klar, dass er helfen konnte, einen Überblick über die verfügbaren Daten zu erhalten. In Tabelle 10 ist der Verlauf der Kategorien in einer beispielhaften Wiedergabelisten-Generierung dargestellt. Es ist erkennbar, auf welche Weise Tags als Kategorien in dem Empfehlungssystem genutzt werden können. Neben Ort- und Zeit- Begriffen tauchen hauptsächlich Genre-Bezeichnungen als Kategorien auf. Diese bilden eine sinnvolle Unterteilung in Kategorien, da so die Verbindungen und die fließenden Übergänge zwischen Genres erkundet werden können. Künstlernamen, Titelnamen und Albumtitel sind auch enthalten und schränken die Empfehlungen auf einen sehr spezifischen Rahmen ein. Diese sind nur sinnvoll, wenn der Benutzer daran interessiert ist, das Repertoire eines bestimmten Künstlers genauer zu betrachten.

Weiterhin konnte festgestellt werden, dass die Benutzung des Systems die Generierung ansprechender Wiedergabelisten ermöglicht. Dabei gibt es allerdings die Einschränkung, dass populäre Lieder stark bevorzugt werden. Dies mindert die Qualität der Empfehlungen, da häufig gleiche Songs auftauchen. Grund dafür ist die Auswahl der Titel in einer Kategorie. Da nur die populärsten Lieder, die den Kategorie-Tag haben, präsentiert werden, ist es nicht möglich, weniger bekannte Lieder in eine Wiedergabeliste einzufügen. Eine weitere Folge ist, dass die Empfehlungen in einer Kategorie immer gleich sind und

Titel	Tag 1	Tag 2	Tag 3	Tag 4
Smells Like Teen Spirit	Grunge	rock	alternative	Nirvana
Two Princes	90s	rock	alternative	alternative rock
Supermassive Black Hole	alternative rock	rock	alternative	Muse
Pompeii	indie	british	pop	alternative
It's Time	indie	alternative	indie rock	rock

Tabelle 10: Verlauf der Tag-Kategorien in den Empfehlungen für fünf aufeinanderfolgende Titel

nicht vom Vorgänger abhängen. Im folgenden Abschnitt wird eine Lösung für dieses Problem dargestellt und beschrieben, wie ein Bezug zum vorangehenden Titel hergestellt werden kann.

7.2.3. Ähnlichkeit auf Tag-Vektoren

Die Qualität von Empfehlungen bei der Wiedergabelisten-Generierung soll verbessert werden, indem ein stärkerer Bezug zum vorangehenden Titel hergestellt wird. Dies kann mit Hilfe von einer Ähnlichkeitsanalyse realisiert werden. Es wird die Überschneidung der zugewiesenen Tags berechnet und als Maß für die Ähnlichkeit interpretiert.

Dazu wurde ein Maß für das Projekt entwickelt, das sich an den Overlap-Koeffizienten $\frac{A \cap B}{\min |A|, |B|}$ anlehnt. Dabei ist A die Menge aller Tags eines Songs A und B die Menge aller Tags eines Songs B. Anstatt von Mengen wird allerdings mit Zahlenwerten in Form von Tag-Intensitäten gearbeitet und somit wird der Schnitt zum Minimum. Dem Effekt, dass Musikstücke mit größerer Popularität häufiger Tags erhalten, wird durch eine Normalisierung entgegen gewirkt. Es ergibt sich eine gleiche "Größe"der Tags beider Titel, wodurch der Nenner im Vergleich zum Overlap-Koeffizient weg fällt. Der Vergleich mit anderen Ähnlichkeitsmaßen ist noch offen, aber wurde bisher nicht priorisiert, da zufriedenstellende Ergebnisse erzielt werden konnten.

Sei S die Menge aller Musikstücke, dann sei für einen Titel $s \in S$ folgendes definiert:

T_s die Menge der Tags von s,

$f_s(t \in T_s) :=$ die Anzahl der Zuweisungen von Tag t für Titel s,

$c_s := \sum_{t \in T_s} f_s(t)$ die gesamte Anzahl an Tag-Zuweisungen.

Außerdem sei für zwei Musikstücke $a, b \in S$ die Menge aller vorkommenden Tag-Bezeichner $T := T_a \cup T_b$, dann ist die Überschneidung der zugewiesenen Tags die Funktion overlap: $S \times S \mapsto [0, 1]$

$$\text{overlap}(a, b) := \sum_{t \in T} \min \left(\frac{f_a(t)}{c_a}, \frac{f_b(t)}{c_b} \right).$$

Es gilt overlap$(a, a) = 1$ und für $T_a \cap T_b = \emptyset$ ist overlap$(a, b) = 0$. Bevor das Minimum als Überschneidung für einen einzelnen Tag bestimmt wird, wird die Anzahl eines Tags durch die gesamte Anzahl an Tag-Zuweisungen für einen Song geteilt. Dadurch wird eine Normalisierung erreicht, sodass Tag-Zuweisungen anteilsmäßig von einem Titel gewertet werden. Hiermit wird erreicht, dass Titel mit unterschiedlicher Anzahl an Tags gleichbedeutend für die Empfehlung sind.

7.2.4. Empfehlung mit Bezug auf die Ähnlichkeit

Mit dem Ziel, die Qualität der Empfehlungen zu verbessern, wurde eine neue Version[18] des Prototyps entwickelt. Die Funktionsweise ist grundsätzlich gleich, nur die Empfehlungen werden anders bestimmt.

Bei der Auswahl der Kategorien werden weiterhin die Top-Tags des Vorgänger-Titels benutzt. Bei diesen taucht häufig der Name des Interpreten auf. Dieser kann jetzt ausgelassen werden, da die Variabilität durch diese Kategorie stark eingeschränkt wird.

Bei der Auswahl der Musikstücke werden die Top-Songs eines Kategorie-Tags als Kandidaten für die Empfehlung in Betracht gezogen. Die Kandidaten können dann gefiltert werden, um weniger sinnvolle Empfehlungen auszuschließen. Dadurch wird verhindert, dass zu viele Titel des selben Künstlers oder ein Vorgänger aus der Wiedergabeliste empfohlen wird. Für alle Kandidaten wird die Ähnlichkeit der Tag-Vektoren zum Vorgänger-Titel berechnet. Die Musikstücke mit der höchsten Überlappung werden dann empfohlen.

Das Empfehlungssystem kann durch eine Menge an Parametern kontrolliert und gesteuert werden. In Abbildung 10 sind diese als Quelltextauszug dargestellt. Es kann dabei eingestellt werden, wie groß die Tiefe der betrachteten Tags und der Kandidaten ist. Dadurch kann der Aufwand der Ähnlichkeitsberechnung gegen die Vollständigkeit abgewogen werden. Weiterhin kann angegeben werden, nach welchen Kriterien Stücke oder Kategorie-Tags von der Empfehlung ausgeschlossen werden.

In Abbildung 11 wird die Ausgabe einer beispielhaften Ausführung des verbesserten Prototypen gezeigt. Die Konfiguration des Systems entspricht den Parameterwerten, die in Abbildung 10 dargestellt werden. Es wird neben jeder Empfehlung ein Prozentwert angegeben, der beschreibt, wie groß die Überlappung der Tags zum letzten Musikstück ist. Bei der Betrachtung der Empfehlungen wird deutlich, dass über die Tag-Ähnlichkeit ein tatsächlicher Zusammenhang zwischen den Stücken berechnet werden kann. Im Vergleich zu der Ausgabe in Abbildung 9 werden nicht nur populäre Songs mit dem Kategorie-Tag empfohlen, sondern auch Stücke, die eine gewisse Ähnlichkeit haben. Auch konnte im Vergleich zum vorigen Prototypen die Variabilität der Empfehlungen verbessert werden. Es werden in der selben Kategorie abhängig vom Vorgänger unterschiedliche Stücke empfohlen.

[18] SVN-Repository /code/tag-based-recommendation

```
//Stores a configuration for the recommendation system
public class SystemSetup {
  //Number of categories
  public int numCategories = 5;
  //Number of songs per category
  public int numRecommend = 3;
  //Number of included tags per song in similarity
  public int similarityTagDepth = 25;
  //Number of candidates in recommendation computation
  public int similaritySongDepth = 50;
  //Max number of songs from same artist in a category
  public int limitSameArtist = 1;
  //Allow artist of the last song as category
  public boolean artistAsCategory = true;
  //Allow songs by last artist (invalid for artist-category)
  public boolean includeLastArtist = false;
}
```

Abbildung 10: Code-Abschnitt zum Einstellen der Parameter des Empfehlungssystems

7.3. Stimmungsbasiertes Empfehlungssystem

Als Alternative zum Empfehlungssystem basierend auf Tags (siehe Kapitel 7.2) wurde ein zweites Empfehlungssystem entwickelt, welches Vorschläge mittels der Stimmungen von Liedern vorschlägt. Dazu wurden im ersten Semester ein einfacher Prototyp entwickelt (siehe Kapitel 7.3.1) und daraufhin eine Ontologie, sowie die Kriterien zur Kategorisierung festgelegt (siehe Kapitel 7.3.2). Im zweiten Semester wurde dann aufbauend auf diesen Ergebnissen ein Beispieldatensatz für die Klassifikation erstellt (siehe Kapitel 7.3.3). Mittels des Datensatzes wurde die Eignung unterschiedlicher Klassifikationsansätze für das Problem getestet (siehe Kapitel 7.3.4) und dann eine Klassifikation mit *Support Vector Machines* implementiert.

7.3.1. Einfache Empfehlung anhand des Tempos

Der merkmalsbasierte Prototyp nutzt die im Anhang A dargestellte Musikdatenbank, um dem Nutzer eine Reihe von Liedern für eine Wiedergabeliste vorzuschlagen. Mittels der *Beats per Minute* (bpm) eines Liedes sollte eine Zuweisung zu den Kategorien *sad* und *happy* geschehen. Dabei beschreiben die *Beats per Minute* das Tempo eines Liedes. Damit wird die Anzahl der Zählzeiten pro Minute angegeben. Eine Zählzeit entspricht in der Regel einer Viertelnote bei Takten wie 4/4 oder 3/4. Bei anderen Takten, wie 6/8, werden drei Achtelnoten bzw. eine punktierte Viertelnote gezählt. Der so errechnete Wert gibt dann das Tempo eines Musikstücks an.

```
Please enter song name as seed and press enter
nirvana

Nirvana - Smells Like Teen Spirit
Grunge      a. 75% Pearl Jam - Jerem b. 74% Soundgarden - Black H
rock        c. 54% Foo Fighters - My d. 51% Red Hot Chili Peppers
alternative e. 44% Beck - Loser      f. 43% Radiohead - High and
90s         g. 56% Blind Melon - No  h. 56% Soul Asylum - Runaway
alt. rock   i. 63% The Smashing Pump j. 54% Foo Fighters - My Her
Choose letter and press enter: c

Foo Fighters - My Hero
rock        a. 67% 3 Doors Down - Kr b. 61% Nickelback - How You
alt. rock   c. 68% The Smashing Pump d. 61% Red Hot Chili Peppers
alternative e. 49% Radiohead - Creep f. 46% Fall Out Boy - My Son
Grunge      g. 58% Soundgarden - Bla h. 56% Nirvana - Milk It
90s         i. 62% The Smashing Pump j. 56% R.E.M. - Losing My Re
Choose letter and press enter:
```

Abbildung 11: Ausgabe einer Ausführung des verbesserten Prototypen

Deshalb wurden mittels der Software *MixMeister BPM Analyzer*[19] von MixMeister die *Beats per Minute* für jedes einzelne Lied extrahiert und für den Prototypen gesichert. Mit den extrahierten *Beats per Minute* wurde dann für diesen Prototyp das arithmetische Mittel von 116 berechnet, welches als Schwellenwert genutzt wurde, um die Lieder in die Kategorien happy (> 116) und sad (< 116) einzusortieren. Da sich in unserer Testdatenbank die Anzahl der fröhlichen Musikstücke nicht wesentlich von der Anzahl der traurigen (eingeschätzt durch eine Meinungsumfrage innerhalb der Gruppe) unterschied, wurde in dem einfachen Prototyp das arithmetische Mittel genutzt.
Diese Daten wurden anschließend für eine Konsolenanwendung genutzt, womit dem Nutzer nach dem Start jeweils zufällig fünf Lieder aus den beiden Kategorien vorgeschlagen wurden. Nach der Wahl eines Liedes wurden dann jeweils fünf weiter Lieder pro Kategorie zufällig vorgeschlagen, wodurch sukzessiv eine Wiedergabeliste zusammengestellt werden konnte. Diese sollte im Anschluss auch ausgegeben werden, um sie manuell in eine andere Musiksoftware zu übertragen.
Bei der Evaluation des Ergebnisses wurde deutlich, dass das Zusammenstellen einer Wiedergabeliste anhand der *Beats per Minute* zwar ein sehr einfaches Modell ermöglicht, aber nur unbefriedigende Ergebnisse liefert. Dies liegt in erster Linie an der Extrahierung der *Beats per Minute*, welche nur sehr unzuverlässig funktioniert. Zwar wird immer ein Ergebnis ausgegeben, doch weichen die ermittelten *Beats per Minute* vor allem bei

[19] http://www.mixmeister.com/download-bpmanalyzer.php

eigentlich langsamen Songs stark vom korrekten Wert ab. So werden für diese extrem hohe *Beats per Minute* Werte errechnet, was ein allgemeines Problem in der Extrahierung dieses Wertes darstellt [7]. Einzig in Bereichen von ca. 100-140 *Beats per Minute* werden annähernd brauchbare Ergebnisse erzielt. Somit kann davon ausgegangen werden, dass in der finalen Applikation die *Beats per Minute* für die Ermittlung der Kategorien kaum nutzbar sind oder zumindest nicht alleine genutzt werden sollten.

7.3.2. Ontologie und Kriterien zur Kategorisierung

Um Songs in Stimmungs-Kategorien einteilen zu können, ist es notwendig, Kriterien zu finden, nach denen die Kategorisierung erfolgen soll. Songeigenschaften, die auf eine Stimmung schließen lassen, werden von der Musikplattform *EchoNest* berechnet und können durch die *EchoNest*-API abgerufen werden. Für die Kategorisierung wurden die folgenden Eigenschaften als relevant erachtet:

- Danceability
  Beschreibt den Grad der Tanzbarkeit. Ein hoher Wert verleitet eher zum tanzen, als ein niedriger.
- Valence
  Beschreibt den Grad der Emotionalität eines Songs. Je höher der Wert, desto intensiver die Emotion.
- Energy
  Beschreibt den Grad der Songenergie. Je höher der Wert, umso eher verleitet der Song zu impulsivem Verhalten.
- Mode
  Beschreibt das Tongeschlecht (Dur, Moll). Dabei steht 0 für Moll und 1 für Dur.
- Tempo
  Beschreibt die Geschwindigkeit des Songs in *Beats per Minute*. Je höher der Wert, desto höher die Songgeschwindigkeit.
- Loudness
  Beschreibt die Lautstärke über die gesamte Songlänge in Dezibel. Je höher der Wert, desto lauter ist der Song.

Als nächstes wurde eine geeignete Anzahl an Kategorien und deren Bezeichnungen ermittelt. Als Inspirationsquelle dafür galt das Valence-Arousal-Modell, welches in Abbildung 12 zu sehen ist. Ein Koordinatensystem dient als Grundlage, um Gefühlszustände zu beschreiben. Die Y-Achse wird als „Arousal“ bezeichnet und entstammt der Psychologie. Es beschreibt die Wachheit und Aufmerksamkeit, die ein Song erzeugen kann. Es ist somit keine emotionale Komponente. So hat man im Schlaf ein sehr niedriges Arousal-Level und ein sehr hohes bei Schmerzen oder ähnlich starken Erregungszuständen. Die X-Achse bezeichnet die „Valence“ und entstammt ebenfalls aus der Psychologie. Es beschreibt den emotionalen Wert, der durch einen Reiz erzeugt wird. So wird das Gefühl der

Zufriedenheit mit einem hohen Valencewert beschrieben und das Gefühl der Niedergeschlagenheit mit einem eher niedrigeren Wert. Die große Anzahl an möglichen Gefühlen, die relativ gut mit dem Valence-Arousal-Modell beschrieben werden können, würde eine Kategorisierung in dieser Form jedoch nicht vereinfachen, da zu viele Kategorien zur Auswahl stehen würden. Daher wurde das Modell vereinfacht und jeder Quadrant des Modells zu einer Kategorie zusammengefasst. So entstanden die vier Kategoriebezeichnungen: Happy, Sad, Angry, Relaxed/Chilled.

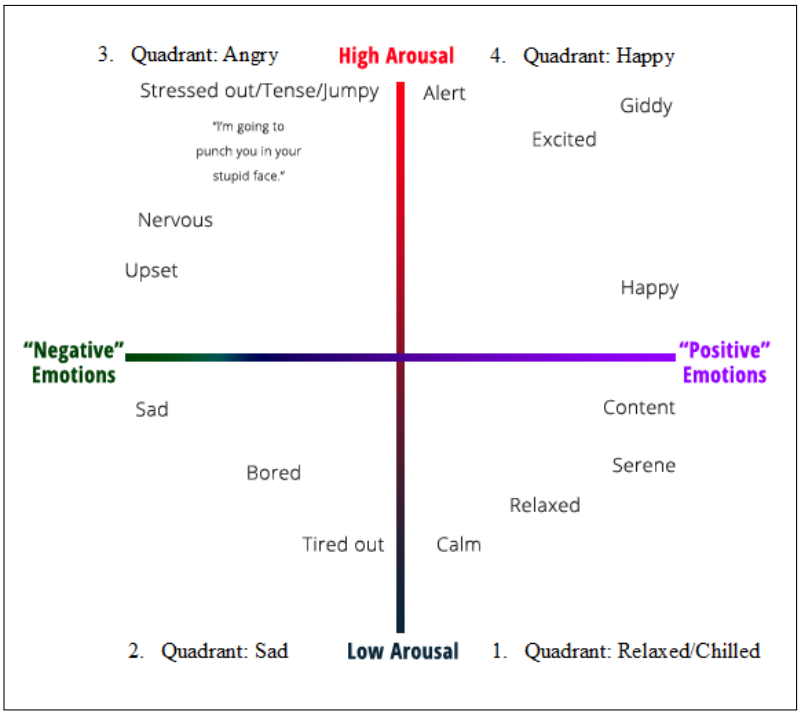

Abbildung 12: Valence-Arousal-Modell[17]

Auswertung	happy	sad	angry	relaxed
Anzahl	18	46	11	25
Anzahl	19471	23133	3838	1871

Tabelle 11: Verteilung der 100 (oben) und 48313 (unten) manuell zugeordneten Lieder zu den einzelnen Klassen

Die Abbildung 13 zeigt eine mögliche Zuordnung von verschiedenen Songeigenschaften zu den jeweiligen Stimmungs-Kategorien, die zunächst nach der persönlichen Einschätzung der zugehörigen Arbeitsgruppe durchgeführt wurde. Die Spaltenbezeichnungen werden durch die Anfangsbuchstaben der weiter oben beschriebenen Songeigenschaften dargestellt (D = Danceability, V = Valence, E = Energy, M = Mode, L = Loudness). Warum das Songtempo nicht in die Ähnlichkeitsberechnung mitaufgenommen wurde, wird am Ende des Abschnitts 7.3.1 erklärt. Die Pfeile repräsentieren die Ausprägungsart eines Merkmals. Dabei stellen nach oben zeigende Pfeile einen hohen Wert dar und die nach unten gerichteten einen niedrigen Wert. Ein Minus zeigt die Neutralität einer Eigenschaft für die jeweilige Kategorie. Es wurden zwar bereits Intervallwerte für die Eigenschaften innerhalb der Kategorien ermittelt, diese sind jedoch rein intuitiver Natur und müssen erst noch evaluiert werden.

Feature / Moods	D	V	E	M	L
Happy	↑	↑	–	1	↑
Sad	↓	↓	↓	0	↓
Angry	–	↓	↑	1/0	↑
Relaxed / Chilled	↓	↗	↓	1/0	↓

Abbildung 13: Mood-Kategorisierungsschema

7.3.3. Erstellung eines Beispieldatensatzes mittels EchoNest Daten

Das Problem einem Lied eine bestimmte Stimmung zuzuordnen, kann als klassisches Klassifikationsproblem angesehen werden. Dafür wird ein Beispieldatensatz benötigt, womit das Klassifikationssystem ein Model antrainieren kann, um neue Daten klassifizieren zu können. An solch einen Datensatz sind unterschiedliche Anforderungen gestellt. So müssen die Daten eine ausreichend genaue Repräsentation des Problems darstellen [8], um nicht nur auf dem Beispieldatensatz gute Klassifikationsergebnisse zu erzielen, sondern auch für noch unbekannte Lieder zufriedenstellende Ergebnisse liefert. Leider

gibt es für Musikstücke kaum Referenzmengen, wie beispielsweise in der Bildklassifikation mit Datensätzen, wie dem Caltech-Datensatz[20] oder diese sind nur schwer erreichbar. Deshalb wurde ein eigener Datensatz erstellt.

Dazu wurden 100 Lieder ausgewählt und per Hand mit einer Stimmung annotiert (siehe Anhang A). Diese 100 Lieder wurden jedoch so ausgewählt, dass sie die beliebtesten Genres gleichmäßig abdecken und dabei jeweils ca. die Hälfte der Lieder von Männern bzw. Frauen gesungen wird. Sie wurde für die Analyse der Tags genutzt und auch für diesen Anwendungsfall getestet. Die Annotation der Lieder geschah durch eine Person der Projektgruppe, um eine einheitliche Ausgangslage für die Annotation zu erhalten. Die Werte verteilten sich somit wie folgt auf die vier Klassen (siehe Tabelle 11).

Wie leicht zu sehen ist, bietet der Datensatz mit 100 Liedern eine ungleiche Verteilung der Daten. Die Klassen *sad* mit nahzu 50% der Annotationen und *relaxed* mit 25% der Annotationen dominieren. Die Klassen *happy* und *angry* sind mit 18% bzw. 11% dagegen deutlich unterrepräsentiert. Somit eignet sich der Datensatz als Beispielsatz nur unzureichend.

Die Erstellung eines solchen Beispieldatensatzes für Musikklassifikation lässt sich nur schwer umsetzen. Um eine wirklich repräsentative Menge zu erhalten, werden viele hundert oder tausende Beispiele benötigt. Diese per Hand herauszusuchen, ist jedoch ein Problem. Denn einerseits kennt niemand alle zur Zeit erhältlichen Lieder und außerdem ist nicht immer klar, welche Stimmung ein Lied vermitteln möchte. Oftmals werden dafür mehrere Hördurchgänge benötigt, die möglicherweise trotzdem Raum für Diskussionen lassen, da Musik von jedem anders empfunden wird. Diese Probleme konnten bereits bei einem kleinen Datensatz mit 40 Liedern festgestellt werden, welcher unter Anhang B betrachtet werden können. Die Größe des Datensatzes ist jedoch absolut unzureichend und eine Vergrößerung dessen würde zu viel Zeit beanspruchen, weswegen Alternativen gefunden wurden. Statt selber einen Datensatz per Hand zu erstellen, wurde ein automatisiertes Verfahren gewählt. Hierzu wurden Daten von Last.fm verwendet, welche die Lieder mit ihren Tags beinhalteten. Daraus wurden anschließend die Lieder ausgewählt, welche Tags annotiert hatten, die mit einer Stimmung übereinstimmten. Dazu wurde beispielsweise überprüft, welche Lieder Tags mit dem Namen *sad* enthielten. So konnten innerhalb kürzester Zeit ca. 48.000 Lieder [21] mit den vier Stimmungen ausgewählt werden. Die Lieder verteilen sich wie folgt auf die vier Stimmungen (siehe Tabelle 11).

Wie zu sehen ist, ähnelt die Verteilung dem ersten Datensatz aus 100 Liedern. Den Großteil machen die beiden Klassen *happy* und *sad* mit jeweils ca. 47% und 40%. Auf Grund der Größe des Datensatzes wurde dieser beibehalten. Anschließend wurden darauf, wie in Kapitel 7.3.4 vorgestellt, unterschiedliche Klassifikationsansätze evaluiert.

7.3.4. Auswertung unterschiedlicher Klassifikationsansätze

Um die bestmöglichen Klassifikationsergebnisse zu erzielen, wurden unterschiedliche Klassifikationsansätze auf demselben Datensatz getestet. Für vergleichbare Ergebnisse wurde für jeden Test eine zehnfache Kreuzvalidierung vorgenommen. Im Folgenden

[20] `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`
[21] SVN-Repository Resources/featureResults/featureResult.csv

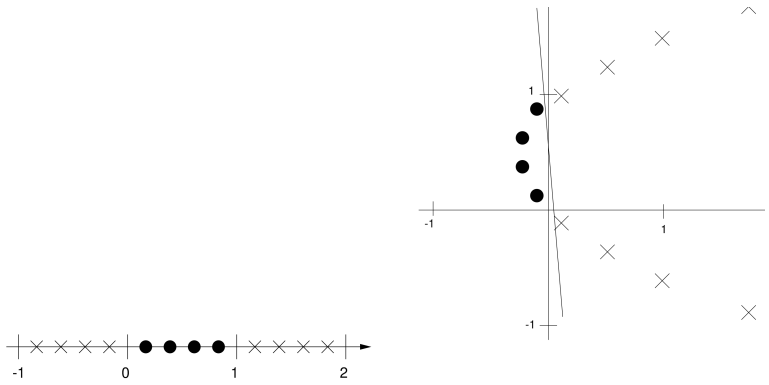

Abbildung 14: Links: Nicht linear separierbare Daten, Rechts: Transformation in höherdimensionalen Raum und linear separierbar (aus [8])

werden die Klassifikationsverfahren und die jeweiligen Ergebnisse vorgestellt.

k-Nearest Neighbour (kNN)
Zuerst wurde auf dem Datensatz ein einfacher kNN-Klassifikator [19] getestet. Dabei wurde für k der Wert 5 gewählt, womit ein Ergebnis von 36% erzielt werden konnte. Dies ist schon mehr als bei einer zufälligen Zuordnung der Lieder bei einer balancierten Menge (25%), aber letztlich zu wenig für eine gute Anwendung. Daher wurden Alternativen getestet.

Um eine geeignete Klassifikationsmethode zu finden, wurden alle nachfolgend beschriebenen Tests in der freien Programmiersprache für statistisches Rechnen und Grafiken R[22] programmiert. Die Methode mit den besten Ergebnissen wurde dann für die Anwendung verwendet.

Support Vector Machine (svm)
Bei der SVM [18] handelt es sich um eine Klassifikationsmethode, welches auch mit nicht-linear trennbaren Daten umgehen kann. Dabei verwendet es sogenannte Kernfunktionen, die vorher bestimmt werden müssen. Die Kernfunktion übernimmt dabei die Aufgabe die Daten vom ursprünglichen Raum in einen höherdimensionalen Raum zu transformieren (siehe Abbildung 14). Diese Transformation ermöglicht dann eine lineare Trennung im neuen Raum. Verschiedene weitere Einstellungsmöglichkeiten können ebenfalls vorgenommen werden, wie z.B. die Kosten bei einer Fehlklassifikation oder die Gewichtung der einzelnen Klassen bei ungleichverteilten Datensätzen. Es wurden etliche Konfigurationen getestet, jedoch werden hier nur einige Ansätze vorgestellt.

SVM mit unterschiedlichen Kernfunktionen
Wie bereits beschrieben, kann der SVM-Ansatz mit nicht-linear trennbaren Daten umgehen, indem der Kerneltrick verwendet wird. Für die vorhandenen Daten wurden ver-

[22] `http://www.r-project.org/`

schiedene Kernel angewandt, die sich hauptsächlich in der Laufzeit, aber nicht signifikant im Ergebnis unterschieden.

- Kernfunktion: Radial-Based
  Genauigkeit: 51%,
- Kernfunktion: Polynomial
  Genauigkeit: 50%
- Kernfunktion: Linear
  Genauigkeit: 50%

SVM mit Gewichten und Linear-Kernfunktion
Da die Klassen ungleich verteilt sind, wurden die Daten mit Gewichten versehen, um die Bedeutungen der jeweiligen Klassen hervorzuheben. Die Gewichte wurden relativ zur Größe der Klassen gewählt, so dass Klassen mit dem Label *angry* und *relaxed* ein höheres Gewicht erhalten haben und Klassenlabel *sad* und *happy* einen niedrigeren. Das Ergebnis dieser Konfiguration hat sich im Vergleich zu den vorherigen Einstellungen ohne Gewichten um knapp **2% verbessert**. Jedoch reichte diese Verbesserung nicht aus, um eine gute Empfehlung zu erhalten. Daher wurde ein neuer Klassifikationsansatz auf Basis einer SVM entwickelt.

Hierarchische SVM
Da die bisherigen Verfahren und Konfigurationen annähernd gleiche Klassifikationsraten aufwiesen, wurde ein neuer Ansatz entwickelt. Die hierarchische SVM führt erst eine Klassifikation auf zwei Klassen durch, um dann innerhalb der Klassen eine erneute Klassifikation durchzuführen. Daher wurden die Daten so präpariert, dass die Klassenbezeichnungen *sad* und *angry*, als auch *happy* und *relaxed* zusammengeführt wurden. Die dadurch neu entstandenen zwei Klassen *sad/angry* und *happy/relaxed* wurden dann mit einer SVM trainiert. Eine Klassifikationsrate in der ersten Hierarchiestufe von

- 64% bei *happy/relaxed* und
- 80% bei *sad/angry*

wurde durch die hierarchische SVM erreicht. Im nächsten Schritt wurden die zusammengeführten Klassen wieder getrennt und innerhalb der bereits klassifizierten Menge erneut trainiert. Das heißt es wurde jetzt eine Klassifikation zwischen *sad* und *angry* als auch zwischen *happy* und *relaxed* vorgenommen. Die Klassifikation zwischen *sad* und *angry* ergibt folgende Ergebnisse:

- sad: 71%,
- angry: 88,5%.

Bei der Klassifikation zwischen happy und relaxed liegen folgende Ergebnisse vor:

- happy: 77 %,
- relaxed: 67,5%.

Die Fehlklassifikation im ersten Schritt muss jetzt noch mit der Fehlklassifkation im zweiten Schritt multipliziert werden, sodass folgende Genauigkeiten entstehen:

- sad: 56,8%, angry: 70,8%,
- happy: 49,28%, relaxed: 43,2%

Im Durchschnitt ergibt sich eine Genauigkeit von **55,02%**, was im Gegensatz zu den anderen Verfahren eine Verbesserung darstellt, aber immer noch nicht ausreichend ist, um in einem Empfehlungssystem eingesetzt zu werden. Daher wurden alle bisherigen Verfahren verworfen.

Klassifikation mit drei Klassen
Die unzureichenden Ergebnisse der vorgestellten Verfahren mit vier Klassen waren ausschlaggebend dafür, dass die Idee eine Klassifikation der Musikstücke auf vier Gefühlsklassen vornehmen zu wollen, ebenfalls verworfen und ein neuer Versuch auf nur drei Klassen gestartet wurde. Dabei entstanden die Klassen *sad*, *happy* und *angry*, wobei die sad-Klasse eine Zusammensetzung aller sad- und relaxed-Klassen ist. Schon beim ersten Versuch mit einer unveränderten SVM-Konfiguration, waren die Ergebnisse für eine zufriedenstellende Empfehlung - basierend auf den drei Gefühlsklassen - ausreichend. Weitere Konfigurationsmaßnahmen führten zu einem Ergebnis von ca. **70 %**. Zwar wurde das Problem so vereinfacht, jedoch kann eine ansprechende Interaktion des Nutzers nur dann entstehen, wenn die Klassifikation auch sinnvoll erscheint. Daher wurde entschieden das einfachere Problem vorzuziehen, um eine möglichst hohe Genauigkeit zu erzielen, damit der Nutzer ein besseres Erlebnis erhält.

Clustering: K-Means
Trotz der guten Ergebnisse der SVM wurde noch ein anderer Ansatz getestet. Statt die gewünschten Kategorien anzutrainieren, sollten mittels unüberwachtem Lernen neue Kategorien automatisch generiert werden. Die neuen, beliebig viele Kategorien sollten per Clustering bestimmt werden. Statt den Clustern Namen zu geben, sollten die darin enthaltenen Lieder eine eigene Repräsentation bzw. Beschreibung liefern. So könnten sich Nutzer anhand der am häufigsten vorkommenden Interpreten und einer Verteilung der vier Kategorien *happy*, *sad*, *angry* und *relaxed* im Cluster einen groben Überblick über diesen verschaffen. Für das Clustering wurde das k-means Clustering verwendet. Hierbei gab es aber bereits die ersten Probleme, da die Güte des Modells von den Startpunkten abhängt. Erst nach vielfachen Wiederholungen konnte eine zufriedenstellende Verteilung auf vier Cluster erreicht werden.

Leider ergab sich in jedem der Cluster wieder eine ähnliche Verteilung der vier Stimmungskategorien mit den dominierenden Kategorien *happy* und *sad* und auch die Interpreten waren ohne Muster verteilt, sodass basierend auf diesen Merkmalen dieser Ansatz keine zufriedenstellenden Ergebnisse lieferte.

Auswertung	C1	C2	C3	C4
Anzahl	11762	15745	14600	6206

Tabelle 12: Verteilung der 48313 Lieder mittels k-means Clustering

Daher wurde letztlich aufgrund der besten Klassifikationsergebnisse der SVM-Ansatz mit nur drei Klassen ausgewählt, da er für das beste Nutzererlebnis sorgt und sinnvolle Empfehlungen gibt.

7.3.5. Ähnlichkeit zwischen zwei Liedern mit Merkmalen

Um Kandidaten für Vorschläge aus einer jeweiligen Stimmungskategorie zu erhalten, wird eine Ähnlichkeit zwischen den einzelnen Liedern basierend auf den Merkmalen berechnet. Diese wird zwischen dem letzten Lied aus der Wiedergabeliste und allen Liedern aus einer Stimmungskategorie berechnet. Hiermit wird eine möglichst homogene Wiedergabeliste erreicht, obwohl Lieder aus unterschiedlichen Kategorien kommen können. Für die Berechnung der Ähnlichkeit zwischen zwei Liedern wird der euklidische Abstand genutzt.

$$sim(\vec{x}, \vec{y}) = \sqrt{(x_1 - y_1)^2 + ... + (x_n - y_n)^2}$$

Dabei beschreiben $\vec{x}$ und $\vec{y}$ die Merkmalsvektoren der beiden zu vergleichenden Lieder.

7.4. Verbesserung der Effizienz

Nach der Implementierung beider Systeme wurde festgestellt, dass die Berechnung neuer Vorschläge ungewöhnlich viel Zeit einnimmt; je größer die Datenbank der angebotenen Lieder war, desto länger dauerte die Berechnung. Grund hierfür war jedoch nicht Berechnung der Ähnlichkeit, sondern die Kommunikation mit der Datenbank (siehe Kapitel 6). Bislang wurden *JPA*[23] bzw. *JPQL*[24] verwendet, um Lieder aus der Datenbank auszuwählen. Diese nutzen das Prinzip der objektrelationalen Abbildung, um mit einer relationalen Datenbank wie mit einer objektorientierten Datenbank zu kommunizieren. Aufgrund komplexer Zusammenhänge zwischen den Entitäten in der Datenbank sorgte dies jedoch für große Performance-Einbrüche. Um dem entgegenzuwirken, wurden einige der Anfragen an die Datenbank durch native SQL Anfragen ersetzt.

Das stimmungs-basierte Empfehlungssystem benötigt für das Berechnen der Vorschläge nur die jeweiligen Lieder aus einer Stimmungskategorie. Dafür wurde eine neue native SQL Anfrage geschrieben, womit die entsprechenden Lied-Objekte erstellt werden konnten. Insgesamt werden pro Kategorie zwei Anfragen benötigt. Denn zuerst werden die jeweiligen Lieder gesucht und dann zu jedem Lied die jeweiligen Merkmale, um anschließend eine Ähnlichkeit zwischen den Liedern berechnen zu können. Offensichtlich steigt

[23] `http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html`
[24] `http://docs.oracle.com/javaee/6/tutorial/doc/bnbtg.html`

die Antwortzeit mit der Größe der Datenbank, wodurch irgendwann auch dieser Ansatz zu langen Wartezeiten führen würde. Daher wurde die SQL Anfrage dahingehend abgeändert, dass nicht mehr alle Lieder einer Kategorie ausgewählt werden, sondern nur noch eine gewisse Anzahl derer. Diese werden per Zufall ausgewählt und ermöglichen somit eine nahezu konstante Antwortzeit. Das Entfernen vieler möglicher Kandidaten ist in diesem Fall kein Nachteil. Denn die Distanzen zwischen den einzelnen Lieder sind aufgrund der Merkmale eher gering. Somit sind in dem System potentiell viele unterschiedliche Lieder ein guter Kandidat für eine Empfehlung und es entscheiden nur geringe Unterschiede. Daher kann auch nur ein Ausschnitt aus der Datenbank für die Empfehlung genutzt werden. Zudem erfüllt dieser Ansatz auch das Ziel der Musikentdeckung. Denn somit werden immer wieder neue, völlig unterschiedliche Vorschläge zu identischen Liedern generiert.

Das tag-basierte Empfehlungssystem hatte mit dem selben Problem zu kämpfen, wobei hier die komplexere Ähnlichkeitsberechnung ebenfalls problematisch war. Daher wurde bei diesem Ansatz bereits während der Kommunikation mit der Datenbank mittels nativen SQL Anfragen ein *JOIN*-Befehl auf den Tags des letzten Liedes und der Kandidaten berechnet, um nur die passenden Lieder zurückzuerhalten. Je nach Anzahl und Popularität der Tags kann jedoch auch dieses System für lange Wartezeiten sorgen, da der JOIN über viel mehr Tags und auch Lieder berechnet wird. Daher wurden nur die benötigten beliebtesten Tags für den JOIN genutzt und anschließend, wie bereits im stimmungsbasierten Empfehlungssystem, eine zufällige Auswahl der Kandidaten vorgenommen. Somit ließ sich auch hier eine annähernd konstante Antwortzeit gewährleisten.

8. Die graphische Benutzeroberfläche

Um die Musik problemfrei genießen zu können, bedarf es einer geeigneten Benutzeroberfläche zum Stöbern. Auf dem Weg dorthin gab es diverse Ideen und Konzepte, wodurch es insgesamt zu zwei GUI[25]-Versionen kam. Ihre Entwicklung und Funktionen werden in den folgenden Unterkapiteln beschrieben.

8.1. Enwicklungsprozess, Funktionalitäten und Gestaltungselemente der Benutzeroberfläche

In diesem Unterkapitel werden zunächst die einzelnen Entwicklungsphasen der graphischen Oberfläche beschrieben. Die Phasen beziehen sich dabei jeweils auf einen bestimmten Zeitraum, in dem an den beschriebenen Funktionen gearbeitet wurde. Das Problem bei der Planung der Benutzeroberfläche bestand darin, dass immer wieder Punkte angefallen sind, die noch ergänzt werden mussten. Außerdem hat die Projektgruppe gemeinsam entschieden, dass zunächst nur Aufgaben erfüllt werden sollten, die den Mindestanforderungen entsprechen und erst später weiterführende Punkte ergänzt werden sollten. So hatten wir zwar am Anfang einen Entwicklungsplan aufgestellt, haben diesen jedoch

[25] Abk. GUI vom Englischen *graphical user interface*. Im Deutschen als grafische Benutzeroberfläche oder auch grafische Benutzerschnittstelle bekannt.

immer wieder erweitert. Dadurch ergab sich mit der Zeit die Entwicklung der gesamten Oberfläche.

Phase 1: Erster Prototyp der Benutzerschnittstelle. Im ersten Semester wurde für die Empfehlungen ein einfaches Design entwickelt, das mit <p:layoutUnit> Objekten gearbeitet hat. Der Aufbau bestand aus dem Objekt „north“, welches den Header enthielt, dem Objekt „center“ für den Inhalt und „west“ und „east“, um die Seitenränder abzugrenzen. Der Inhalt wurde später weiter verwendet. In diesem befand sich ein <p:panelGrid>, das mit Hilfe von Zeilen und Spalten die „Mood-Line“ mit Dummy-Bildern angezeigt hat, wie in Abbildung 16 zu sehen ist.
Außerdem wurde im ersten Semester ein Template mit den drei Bereichen Header, Content und Player erstellt. Für technische Details und zusätzliche Informationen, siehe das Kapitel „Implementierung eines Templates“ im Zwischenbericht [13].
Im ersten Semester wurden auch JSF-Seiten zum Einloggen und Registrieren entwickelt. Diese wurden im weiteren Verlauf jedoch nicht erneut verwendet, da innerhalb der Gruppe beschlossen wurde, dass es kein Logging geben sollte.

Im zweiten Semester sollten nun die Content-Seiten für das Template angefertigt werden. Dazu wurde das erste einfache Design des vorherigen Semesters erneut verwendet. Es wurden alle Layout-Objekte entfernt und nur der Inhalt genutzt. Dieser konnte als eine neue Index-Seite gespeichert werden und mit Hilfe des Templates angezeigt werden. Dazu wurden die Dummy-Darstellungen entfernt und eine dynamische Funktion eingebunden, welche die Zeilen automatisch generiert hat. Dafür wurden Beispiel-Moods entwickelt und zum Testen angezeigt. Mit Hilfe eines neu erstellen Drop-Down-Menüs konnte auch bereits zwischen den Empfehlungssystemen gewechselt werden. Allerdings waren diese Daten anfänglich aus einer Dummy-Datenbank.

Abbildung 15: Ursprüngliches Template

Abbildung 16: Ansicht des ersten Designs für die stimmungs-basierte Musikempfehlung

Phase 2: Einbindung eines Players. Der zweite Schritt war, einen funktionsfähigen Player hinzuzufügen. Dieser konnte leicht mit Hilfe von *HTML5* eingebaut werden und ist in Abbildung 17 unten zu sehen. Zum Testen wurde zunächst ein Live-Streaming aus dem Internet verwendet. Das bedeutet, wir haben eine URL aus dem Internet mit der Endung „mp3“ verwendet. Der *HTML-audio-Tag* konnte diese dann ohne zusätzliche Funktionen automatisch abspielen.
Der HTML-audio-Tag verfügt über die folgenden Attribute:

- *src*: eine URL für den Inhalt,
- *loop*: wiederholt den Inhalt,
- *autoplay*: der Inhalt wird direkt abgespielt, sobald ausreichend Daten geladen sind,
- *autobuffer*: fängt direkt mit dem Laden an,
- *controls*: die Steuerelemente (Abspielen, Pause, Position und Lautstärke).

Parallel dazu wurde die Auswahl zwischen dem stimmungs- und tag-basierten Empfehlungssystem entwickelt. Um zwischen den beiden Varianten wählen zu können, wurde ein Drop-Down-Menü hinzugefügt, wie in Abbildung 17 zu sehen ist. Die eingeblendeten Lieder wurden testweise einer Dummy-Datenbank entnommen. Außerdem konnten mit Hilfe von Filtern die einzelnen Kategorien ein- und ausgeblendet werden.

Abbildung 17: Einbindung eines Players

Phase 3: Implementierung der Interaktion mit dem Benutzer. Der nächste Schritt war, dem Nutzer zu ermöglichen, eine abspielbare Wiederhabeliste zu erzeugen. Dazu wurde das „Abspielen“, „Hinzufügen“ und „Löschen“ implementiert. Wenn ein Lied zur Liste hinzugefügt wurde, haben sich die empfohlenen Lieder automatisch neu geladen. Diese wurden vorübergehend nur durch unterschiedliche Liednummern dargestellt, da noch keine festen Daten vorhanden waren. In dieser Phase wurde auch erstmalig das Design überarbeitet und angepasst. In Abbildung 18 sind die neuen Buttons, das allgemeine Design und die erste Version der Wiedergabeliste zu sehen.

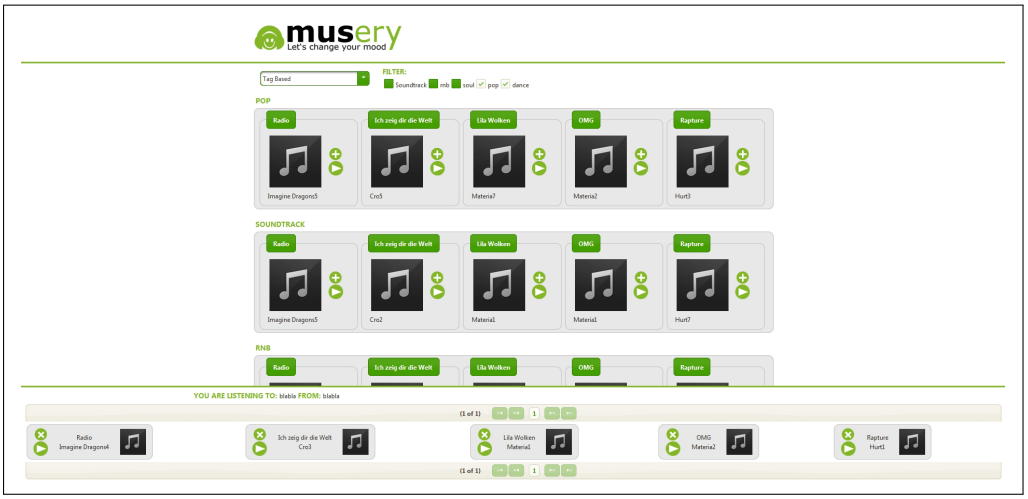

Abbildung 18: Das erste Design der überarbeiteten Seite

Phase 4: Benutzerfreundliche Navigation und Oberfläche. Im Folgenden wurden alle Funktionen der Navigation in den Header ausgelagert, um einen größeren Bereich für die Empfehlungen zu gestalten. Außerdem wurde ein neuer Filter eingefügt, der jedoch erst einmal ohne Funktion war, siehe Abbildung 19.
Eine weitere Neuerung war der Mouseover-Effekt der Buttons. Diese sollten nur zu sehen sein, wenn der Nutzer mit der Maus über das Bild des dazugehörigen Liedes führt.
Um den Player übersichtlicher zu gestalten, wurden Titel und Künstler des aktuell abgespielten Liedes eingefügt. Auch wurden die Lieder in der Wiedergabeliste durch eine neue Funktion, automatisch nacheinander abgespielt. Um an dieser Stelle platzsparender zu gestalten, wurde der Player zusätzlich um die Funktion „Ausblenden“ erweitert. Auf diese Weise konnte die Wiedergabeliste ausgeblendet werden, der Player selbst wurde aber noch angezeigt.
Eine weitere Neuerung war der Einbau der Fortschrittsanzeige (Abbildung 20).

Abbildung 19: Der überarbeitete Header

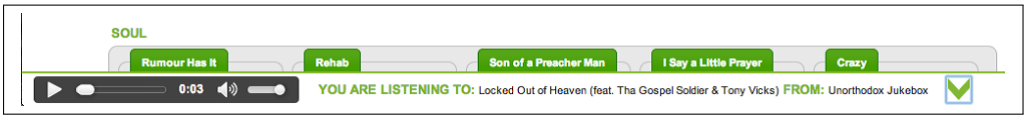

Abbildung 20: Der überarbeitete Player

Phase 5: Neue Startseite und intuitivere Bedienung. In der nächsten Phase wurde eine neue Startseite eingerichtet. Auf dieser befanden sich zwei Buttons, mit Hil-

fe derer benutzerfreundlicher auf die unterschiedlichen Empfehlungssysteme zugegriffen werden konnte (Abbildung 21). Außerdem wurde dadurch das Problem der unübersichtlichen Startseite behoben. Beim Klick auf einen dieser Buttons wurde jeweils das entsprechende Empfehlungssystem mit Kategorien angezeigt. Eine weitere Neuerung war, dass die Buttons der Musikempfehlungen nun auch über den Covern der Lieder angezeigt wurden, um die Bedingung intuitiver zu gestalten (Abbildung 22).

Abbildung 21: Die neuen Buttons der Startseite

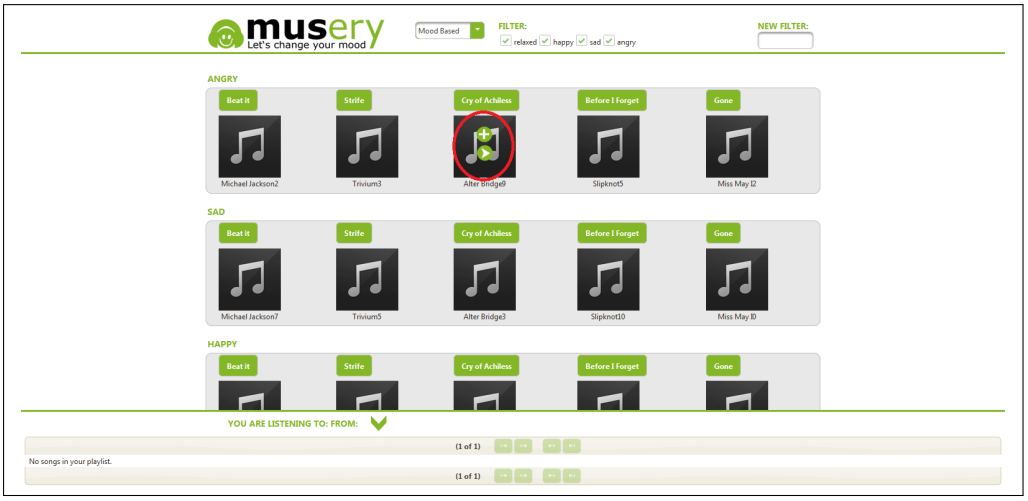

Abbildung 22: Die fertigen Mouseover-Effekte

Phase 6: Einbau einer Suchfunktion und Verbesserung des Designs. Hier wurde zunächst der Filter, der in Phase 2 eingebaut wurde, mit Funktionen versehen. Bei der Eingabe eines Wortes in das Suchfeld wurden nun Ergebnisse angezeigt, welche jedoch erst einmal aus einer Dummy-Datenbank stammten. Der Filter war auch in der Lage bei unsinnigen Eingaben eine Warnung auszugeben. Abschließend wurde ein „Suche“-Button zum Filter hinzugefügt, da viele Nutzer dieses intuitiv eher verwenden, als die Taste „Eingabe“ auf der Tastatur (Abbildung 23).
Außerdem wurde die Startseite überarbeitet und neu entworfen, um mit dem Corporate Design von musery übereinzustimmen (Abbildung 24). Eine weitere Aufgabe war, zwei neue Buttons im Player einzubauen. Diese sollten für die Funktionen „spiele ein Lied“ und „spiele alle Lieder“ genutzt werden (Abbildung 24 unten). Eine weitere Neuerung

waren Platzhalter, die immer dann eingeblendet werden sollten, sobald ein Lied kein Cover besitzt.
In dieser Phase wurde auch ein Verfahren zu *Drag-and-Drop* getestet. Leider hat sich dabei herausgestellt, dass dieses sehr kompliziert war. Innerhalb der Struktur unserer JSF-Seiten hätte sehr viel verändert werden müssen. Da dies nicht zu den Anforderungen gehörte, wurde es verworfen.

Abbildung 23: Das neue Suche-Feld

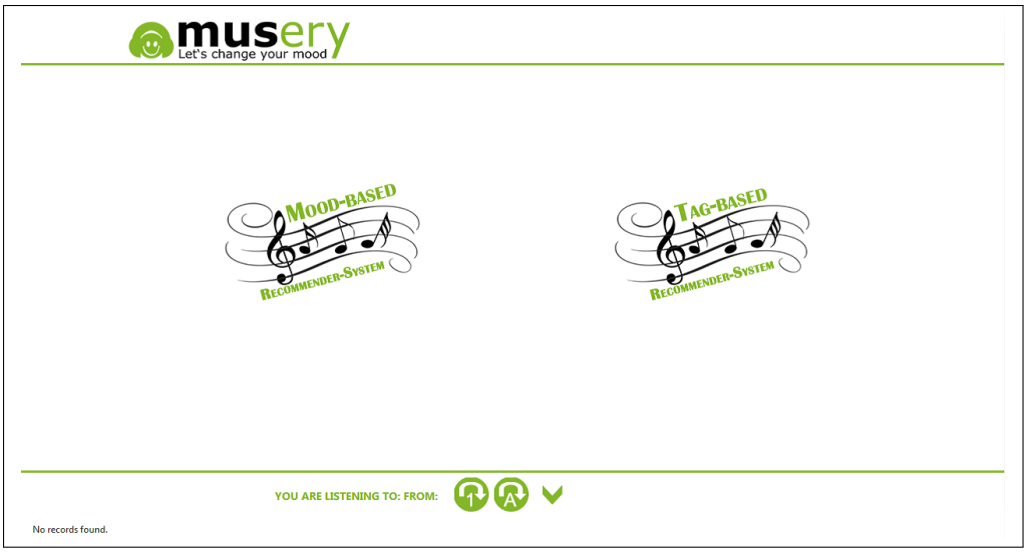

Abbildung 24: Design der Startseite

Phase 7: Überarbeitung der gesamten Struktur. In der letzten Phase wurde die gesamte Struktur der Seite überarbeitet. Dabei sollte die Startseite zwar erhalten bleiben, aber sie sollte nicht mehr direkt zu den Empfehlungssystemen führen, sondern zunächst die Auswahl eines Startsongs ermöglichen. Bei der stimmungs-basierten Empfehlung sollte an dieser Stelle die Filter-Suche der vorherigen Phase verwendet werden. Dieses wurde auch gleichzeitig an die Datenbank angeschlossen und konnte sofort mit richtigen Suchbegriffen arbeiten, siehe Abbildung 25.
Die tag-basierte Auswahl eines Startliedes sollte mit einer Tag-Wolke realisiert werden, wie sie in Abbildung 26 zu sehen ist. Dies wurde mit einem *Composite Component*[26]

[26]Kompositkomponenten sind eines der wichtigsten Features seit JSF 2.0. Entwickler erhalten durch die Verbindung von Facelets und Ressourcen die Möglichkeit, Komponenten aus beinahe beliebigen Seitenfragmenten aufzubauen. Eine Kompositkomponente ist im Grunde nichts anderes als ein XHTML-Dokument, das in einer Ressourcenbibliothek abgelegt ist und die Komponente deklariert.

gelöst, welches in der *PrimeFaces*-Bibliothek enthalten ist. Komponenten sind ein essenzieller Bestandteil von JSF und bilden einen zentralen Erweiterungspunkt. Eine *Composite Component* kann völlig unabhängig von der aufrufenden Seite Informationen über interne Zustände im JSF Komponentenbaum ablegen.
Bei den beiden o. g. Verfahren wurde, nach der Auswahl eines Liedes, das entsprechende Empfehlungssystem angezeigt und die empfohlenen Lieder wurden anhand des Start-Liedes errechnet.

Abbildung 25: Startlied für das Stimmungs-basierte Empfehlungssystem auswählen

Abbildung 26: Startlied für das tag-basierte Empfehlungssystem auswählen

8.2. Komponenten der neuen Entwicklung

In der ersten Version der GUI (musery GUI 1.0) wurden zwar die meisten Funktionen implementiert, aber die Wartbarkeit des Codes war nicht befriedigend. Zusätzlich entstanden am Ende des Projektes neue Anforderungen, einerseits sollte die Startseite umgebaut werden, andererseits das Streaming direkt aus einem eigenen Server geladen werden. Um die neuen Funktionen übersichtlicher zu entwickeln und die Softwarearchitektur und die Wartbarkeit der Codes zu verbessern, wurde musery neu entwickelt (musery GUI 2.0). In diesem Unterkapitel werden die neu entwickelten Komponenten beschrieben.

Design. Das ursprüngliche Design wurde hier wiederverwendet. Die Gestaltung basiert also auf der ersten Version, welche in der Produktvision erläutert wurde, siehe Abbildung 27. Dennoch gibt es ein paar Erweiterungen, die im Folgenden einzeln weiter beschrieben werden.

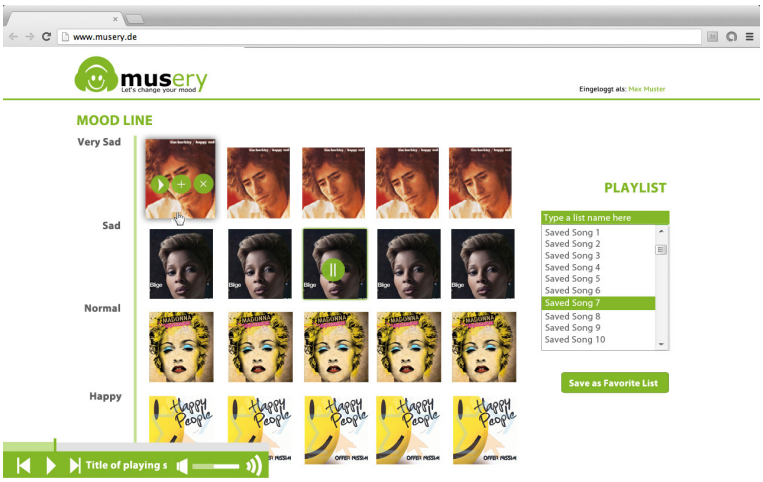

Abbildung 27: Gestaltung der Empfehlungsseite

Suchergebnisse. musery startet nun mit einer Suche und aus dem Suchergebnis wählt der Nutzer ein Lied als Startlied. Der Benutzer kann als Suchbegriff ein Wort oder auch mehrere Wörter eingeben. Wenn das Lied ein Coverbild hat, wird dies angezeigt, sonst wird ein Platzhalter angezeigt. Bei den Ergebnissen werden nicht nur Titel und Künstlername, sondern auch Moods und Tags zu dem jeweiligen Lied angezeigt. Der Benutzer muss auf „Mood“ oder „Tag“ klicken, um das stimmungs- oder tag-basierte Empfehlungssystem zu starten. Siehe hierzu Abbildung 28. Danach startet das System und es werden die berechneten Empfehlungen angezeigt.
Die Suche basiert auf Lucene[27]. Künstlername, Titel, Tags und Moods eines Liedes werden vor der Suche schon indiziert. Der Benutzer kann beliebige Begriffe in willkürlicher

[27] http://lucene.apache.org/core/

Kombination eingeben. Wenn die Begriffe in den angegebenen Suchfeldern wie Künstlername, Titel u. s. w. existieren, werden sie gefunden. Der Benutzer kann mit *AND*- oder *OR*-Logik suchen. Die Suchergebnisse werden auf die zehn Ergebnisse, die am Besten zu den Suchbegriffen passen, beschränkt. Suchergebnisse werden nach Relevanz sortiert und die ersten zehn werden angezeigt. Künstlername und Titel haben mehr Gewicht bei der Relevanzbewertung als Tags und Moods.

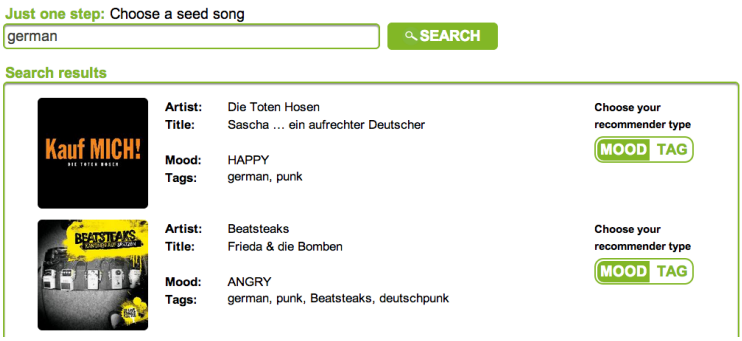

Abbildung 28: Suchergebnisse der GUI 2.0

Player. Der Player in der GUI 1.0 wurde automatisch aus dem HTML5 *audio*-Tag erzeugt. Da der audio-Tag jedoch von allen Browsern unterschiedlich gut unterstützt wird, sieht der Player auf verschiedenen Browsern anders aus. In der GUI 2.0 wird der Player daher aus einzelnen Symbolen zusammen gebaut und die Abspielfunktion in der Datei „audio.js“ durch Streaming implementiert.
Bei dem neuen Player kann man die Musik abspielen und pausieren, dabei wechselt das „Abspielen“-Symbol zu einem „Pausieren“-Symbol und wieder zurück. Wenn es mehrere Lieder in der Wiedergabeliste gibt, so kann man durch Klicken auf „Vorheriges“ oder „Nächstes“ zwischen den Liedern wechseln.
Ein kleines Cover-Bild des aktuell abgespielten Liedes wird links unten in der Ecke angezeigt. Rechts unten ist ein Button für das Ein- und Ausblenden der Wiedergabeliste, wie in Abbildung 29 zu sehen ist. Diese wird standardmäßig nicht angezeigt und kann bei Bedarf in einem Popup-Fenster angezeigt werden.

Abbildung 29: Player der GUI 2.0

Wiedergabeliste. Durch Klicken auf den „Sortieren“-Button kann die Reihenfolge der Lieder in der Wiedergabeliste beliebig verändert werden. Dabei wird ein Lied ausgewählt, das der Benutzer dann mit Hilfe von Drag-and-Drop an jede beliebige Stelle in der Liste schieben kann. Dasselbe Lied kann auch mehrmals in einer Wiedergabeliste existieren. Die Ansicht der Wiedergabeliste ist in Abbildung 30 zu sehen.

Die Wiedergabeliste wird zum großen Teil mit einem eigenem Javascript implementiert. Die Datei „playlist.js“ enthält das dazu entsprechende Objekt.

Abbildung 30: Wiedergabeliste der GUI 2.0

Empfehlungslisten. Die Vorschaubilder der Lieder in den Empfehlungslisten haben die Größe von 200 x 200 Pixel anstatt 100 x 100. Sie werden über Streaming direkt von einem eigenen Server geladen. Wenn kein Cover-Bild für das Lied auf dem Server vorhanden ist, wird ein Standard-Bild als Platzhalter geladen. Künstlername und Titel der Lieder werden direkt auf dem Coverbild angezeigt. Die Buttons zum Hinzufügen eines Liedes zu der Wiedergabeliste und zum Abspielen werden beim Mouseover angezeigt.
Die Empfehlungsliste hat zwei Pfeile, mit denen man die Liste nach links oder rechts „sliden“ kann, siehe Abbildung 31. Es können daher auch mehr als fünf Lieder zu einem Mood oder Tag empfohlen werden. Davon werden die ersten fünf direkt angezeigt, die anderen kann der Nutzer durch „sliden“ nach links oder rechts anschauen. Dieser Slider wurde mithilfe eines JQuery-Frameworks *SmoothDivScroll* implementiert.
Die Daten für die Empfehlungsergebnisse werden vom Server in JSON-Format übertragen und clientseitig mit Javascript geparst.

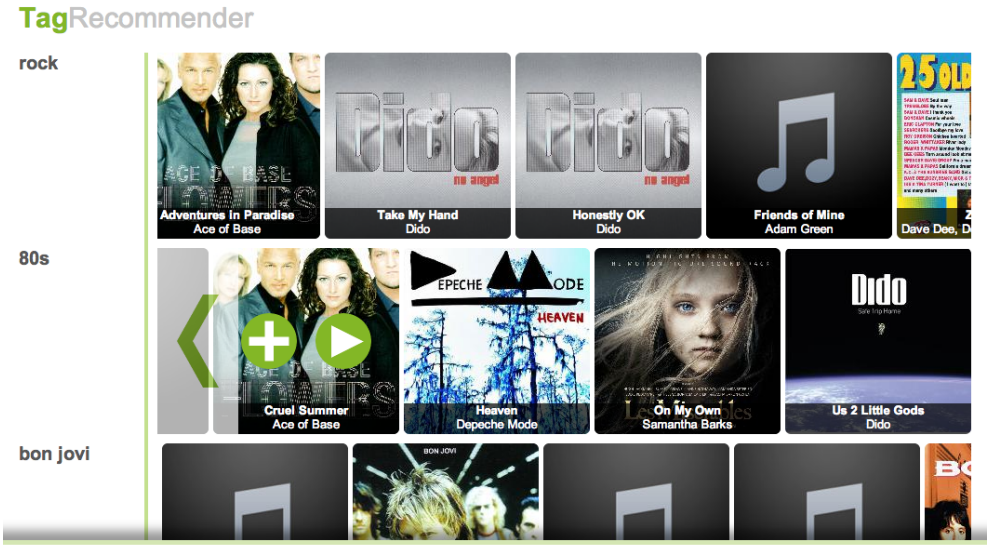

Abbildung 31: Empfehlungsseite der GUI 2.0

Fortschrittsanzeige. In der älteren Version der GUI wird die Fortschrittsanzeige mit dem Javascript-Plugin *loadie* implementiert. In der neuen Version ist die Fortschrittsanzeige eine GIF-Animation. Sie wird in einem Dialog platziert und wenn der AJAX-Aufruf stattfindet, wird dieser Dialog angezeigt bis der Aufruf zu Ende ist, siehe Abbildung 32.

Abbildung 32: Fortschrittsanzeige der GUI 2.0

9. Evaluation

Mit Hilfe einer Evaluation unserer Software wollen wir die Umsetzung der Benutzerfreundlichkeit und Nutzbarkeit von musery testen. Als Mittel zur Informationssammlung kommen dabei Fragebögen zum Einsatz. Diese werden ausgewertet, um verschiedene Aspekte des Projektes zu beurteilen. Ziele sind dabei die Qualitätssicherung, die Identifikation von Schwachstellen der Software und die Anpassung der Software an die Bedürfnisse des Benutzers. Im Folgenden wird die Erstellung und Analyse der Evaluation vorgestellt. Zunächst wurde die Zielgruppe für die Evaluation festgelegt (Kapitel 9.1). Anschließend wurde eine Analyse von Literatur für die Auswahl der Fragen und Kriterien

untersucht (Kapitel 9.2). Darauffolgend wurde der Inhalt des Fragebogens (Kapitel 9.3) und die hierzu verwendete Skala für die Bewertung der Fragen vorgestellt (Kapitel 9.4). Nach einer kurzen Erläuterung der Durchführung der Evaluation (Kapitel 9.5) wurden anschließend die Testdaten ausgewertet (Kapitel 9.6).

9.1. Auswahl der Testteilnehmer

Bei der Planung der Evaluation wurde diskutiert, welche Personengruppen für die Evaluation zur Verfügung stehen und wer sich eignet. Die Sammlung der potentiellen Teilnehmer ergab folgende Gruppen:

- Projektgruppenmitglieder,
- Freunde,
- Verwandte,
- Mitarbeiter der Universität und
- Onlinetester.

Von diesen Gruppen wurden Projektgruppenmitglieder und Onlinetester ausgeschlossen. Projektgruppenmitglieder wurden erwogen, da sie gut zu erreichen sind und eine sichere Beantwortung des Fragebogens liefern können. Wegen der mangelnden Neutralität gegenüber dem Projekt wurden sie aber von der Auswertung ausgenommen. Ein Onlinetest hat den Vorteil, dass eine große Anzahl an Befragungen ermöglicht wird. Die Umsetzung erfordert allerdings zusätzliche Befragungs- und Auswertungstechnologien. Da wir erwarteten, über Freunde, Verwandte und Mitarbeiter der Universität genügend Personen direkt zu erreichen, verzichteten wir auf die Onlinetester.

Damit die gesammelten Daten möglichst viele qualitative Informationen enthalten, sollen die Testteilnehmer einer potentiellen Zielgruppe der Software entsprechen. Deshalb haben wir uns dazu entschieden, Personen im Alter von 20 bis 35 Jahren zu wählen. Grund dafür ist das vermeintliche interesse an aktueller Musik und eine grundsätzliche Offenheit gegenüber technologischen Neuerungen.

9.2. Analyse der Literatur zur Auswahl der Fragen und Kriterien

Bei der Erstellung des Fragebogens wurde darauf geachtet, dass die Fragen und Kategorien mit einem wissenschaftlichen Bezug erarbeitet werden. Hierzu wurden Modelle untersucht, die bei der Technologienutzung von Informationssystemen zu den bekanntesten Theorien in diesem Forschungsgebiet gehören. Zum einen das *Technology Acceptance Model*, welches Aussagen darüber trifft, warum Personen eine Technologie nutzen oder nicht nutzen. Zum anderen das *User Satisfaction Model*, das sich mit der Wahrnehmung von IT-Systemen im Sinne der Nutzerfreundlichkeit beschäftigt. Das *Technology Acceptance Model* beruht auf sozialpsychologischen Modellen und wurde von Frank Davis entwickelt. Es versucht im Wesentlichen die Einflussfaktoren auf die Nutzerakzeptanz

zu verstehen und daraus Vorhersagen zu schaffen[9]. Kernidee des Modells ist, dass die Nutzung eines Systems durch eine Person von zwei Variablen abhängig ist[1]:

- Perceived Usefulness
  Beschreibt die wahrgenommene Nützlichkeit der Anwendung
- Perceived Ease of Use
  Beschreibt die wahrgenommene Einfachheit der Nutzung

Die Intention zur Nutzung des Systems ist dabei abhängig von der *Perceived Usefulness* und der *Attitude Toward Using*[9]. Im Forschungsgebiet der *User Satisfaction* gibt es viele verschiedene Studien zur Nutzerzufriedenheit. Das Messmodell umfasst einen Fragenkatalog von 39 Fragen, die für unseren Fragebogen aus praktischen Gründen auf 13 Fragen reduziert worden sind. Als wichtige Variablen für die Nutzerzufriedenheit wurden folgende festgehalten:

- Nützlichkeit
- Einfachheit der Nutzung
- Ergebnis
- Erwartung
- Systemqualität
- Servicequalität
- Informationsqualität

Die Bewertung der Evaluationen stützt sich oftmals auf die Genauigkeit der verwendeten Algorithmen[16]. Neueren Studien zufolge korreliert die subjektive Nutzerzufriedenheit nicht immer mit einer hohen Genauigkeit der verwendeten Algorithmen. Diese untersuchten zum einen, inwiefern systemzugehörige Aspekte Einfluss auf die Wahrnehmung und die Zufriedenheit der Person nehmen, d.h. welche Daten in das System einfließen und in welcher Art und Weise sie verarbeitet und ausgegeben werden. Zum anderen wurden *Recommendation Agents* getestet, die Interessen und Vorlieben der Nutzer im Vorfeld bereits ermitteln, was zu einer besseren Empfehlung führt aber im Umkehrschluss mit einem größeren Aufwand verbunden ist. Diesen Ansätzen fehlte allerdings die Betrachtung mit psychometrischen Methoden. Das SUMI (Software Usability Measurement Inventory) ist ein solches psychometrisches Evaluationsmodell, dass die Qualität der Software aus Sicht des Anwenders bewertet[16]. Das Modell besteht aus fünf Ebenen:

- Effizienz
- Emotion
- Hilfe

- Kontrolle
- Erlernbarkeit

Der Fragekatalog umfasst hierbei 50 Fragen für die Bewertung einer Software.

Eine Weiterentwicklung dieses Modelles ist ResQue, deren Modell auf fünzehn Ebenen erweitert wurde, aufgeteilt auf vier Schichten und aus insgesamt 43 Fragen besteht[16].

- Wahrgenommene Systemqualität
  - Qualität der Empfehlung
    inwieweit die Interessen und Vorlieben den Empfehlungen entsprechen

  Einer der wichtigsten Eigenschaften für die Bewertung eines Empfehlungssystemes. Hierunter fallen:

  - Genauigkeit
    wie genau die Empfehlungen getroffen wurden
  - Möglichkeit zu entdecken
    ob dem Anwender neue und interessante Empfehlungen auf Basis seiner ursprünglichen Suche möglich sind
  - Attraktivität der Wahrnehmung
    ob die Vorstellungskraft des Nutzers angesprochen wird
  - Vielfalt
    ob es keine wiederkehrenden Empfehlungen gibt
  - Layout
    ob eine ansprechende Gestaltung das Interesse des Nutzers positiv beeinflußt
  - Interaktionsmöglichkeiten
    wie relevante Informationen gesammelt werden bei Interaktionen des Nutzers
  - Informationsangebot
    ob das Angebot an Informationen für eine Entscheidung des Anwenders hilfreich sein können

- Vorstellungen
  - Nützlichkeit
    inwieweit das Empfehlungssystem nützlich ist
  - Benutzerfreundlichkeit
    ob der Nutzer seine Aufgaben effizient erledigen kann
  - Kontrolle
    ob Änderungen von Nutzereinstellungen und erhaltenen Empfehlungen möglich sind
  - Transparenz
    ob der Nutzer das Empfehlungssystem und seine Funktionsweise versteht

- Einstellung
  inwieweit der Nutzer Vetrauen in das Empfehlungssystem hat

- Intention
  ob das Empfehlungssystem in der Lage ist, die Entscheidungen des Nutzers zu beeinflußen

9.3. Inhalt des Fragebogens

Die Aspekte aus dem vorangehenden Teil wurden bei der Gestaltung des Fragebogens berücksichtigt und es ergaben sich sechs Fragekategorien, die die wichtigsten Ansätze abdecken:

- Angaben zur Person
- Design und Funktionen
- Systemqualität
- Benutzerfreundlichkeit
- Empfehlungen
- Allgemeine Zufriedenheit

Die einzelnen Fragen wurden im Laufe der Zeit überarbeitet. Die erste Version des Fragebogens beinhaltete 33 Fragen. Viele dieser Fragen korrelierten inhaltlich miteinander, so dass sie zusammengefasst wurden. Außerdem wurde der Umfang der Fragen auf 18 Fragen beschränkt. Die Kategorien setzen sich somit aus folgenden Fragen zusammen:

- Angaben zur Person
  - Bitte geben Sie ihr Alter an
  - Bitte geben Sie ihr Geschlecht an
- Design und Funktionen
  - Die Gestaltung von musery ist optisch ansprechend
  - Das Design unterstützt die Funktionen
  - Die einheitliche Gestaltung erleichert die Orientierung
  - musery verwendet gut verständliche Begriffe, Bezeichnungen, Abkürzungen und Symbole in den einzelnen Menüs
  - Mir gefallen die unterschiedlichen Ansätze zur Musikempfehlung, die nach Stimmungen ("moods") oder nach zugehörigen Schlüsselwörtern ("tags") suchen
- Systemqualität

  - Der Seitenaufbau erfolgt ohne Verzögerungen
  - Ungültige Eingaben werden erkannt und gemeldet
  - Wichtige Anweisungen, beispielsweise das Löschen einer Wiedergabeliste, werden vor der Ausführung zur Bestätigung angezeigt
- Benutzerfreundlichkeit
  - musery eignet sich gut, um Musik zusammenzustellen und zu hören
  - Fehlermeldungen sind einfach und präzise formuliert
  - Die Bedienung von musery ist einfach, verständlich und leicht zu erlernen
  - Eine Wiedergabeliste ist leicht zu erstellen
- Empfehlungen
  - Die Empfehlungen von musery haben eine hohe Qualität
  - Die Empfehlungen wiederholen sich nicht häufig
  - Man kann neue und interessante Musik entdecken
- Allgemeine Zufriedenheit
  - Wie bewerten Sie musery im Gesamteindruck?

9.4. Entwicklung einer Skala

Um die Information quantitativ messen zu können, wurden die Antwortvorgaben mit einem Zahlenwert von 1 bis 5 versehen, der zur Analyse der Datensätze genutzt wird. Als Antwortvorgaben wurden für die Kategorien Design und Funktionen, Systemqualität, Benutzerfreundlichkeit und Empfehlungen folgende ausgewählt:

- sehr zutreffend (1)
- zutreffend (2)
- mittel (3)
- unzutreffend (4)
- sehr unzutreffend (5)

Für die Fragekategorie allgemeine Zufriedenheit wurden folgende Antwortvorgaben ausgewählt:

- sehr gut
- gut
- mittel
- schlecht

- sehr schlecht

Zusätzlich war bei allen Kategorien, außer Angaben zur Person, die Möglichkeit gegeben einen Kommentar zu einer einzelnen Frage in einem Textfeld zu hinterlegen. Das gibt uns die Möglichkeit, weitere Einsichten in die Ursachen für auffällige Bewertungsmuster zu bekommen.

9.5. Durchführung der Befragung

In der Planungsphase war die Evaluation zunächst in schriftlicher Form angedacht. Hierzu wurde ein Fragebogen erstellt, der an Freunde und Verwandte der Projektgruppenmitglieder verteilt werden sollte. Zudem sollte ein Umfragestand in der OH14 an der TU Dortmund aufgebaut werden, um Mitarbeiter und Angehörige des Lehrstuhls 13 zu bitten an der Umfrage teilzunehmen. Aufgrund des hohen Zeitaufwandes für die Planung und Durchführung wurde die Umfrage Online erstellt. Als Umfragetool wurde *LimeSurvey* verwendet, einer Software zur Datenerhebung mittels Onlineumfragen, die vom Bereich E-Learning des ITMC betreut wird.

9.6. Auswertung der Daten

Im Folgenden werden die Testdaten zunächst anhand von verschiedenen Diagrammen erläutert (Kapitel 9.6.1). Danach erfolgt eine Analyse mit Hilfe von Korrelationen (Kapitel 9.6.2). Für die Auswertung werden die Fragestellungen wie folgt abgekürzt:

- **A** Die Gestaltung von musery ist optisch ansprechend
- **A1** Das Design unterstützt die Funktionen
- **A2** Die einheitliche Gestaltung erleichert die Orientierung
- **A3** musery verwendet gut verständliche Begriffe, Bezeichnungen, Abkürzungen und Symbole in den einzelnen Menüs
- **A4** Mir gefallen die unterschiedlichen Ansätze zur Musikempfehlung, die nach Stimmungen ("moods") oder nach zugehörigen Schlüsselwörtern ("tags") suchen
- **B** Der Seitenaufbau erfolgt ohne Verzögerungen
- **B1** Ungültige Eingaben werden erkannt und gemeldet
- **B2** Wichtige Anweisungen, beispielsweise das Löschen einer Wiedergabeliste, werden vor der Ausführung zur Bestätigung angezeigt
- **C** musery eignet sich gut, um Musik zusammenzustellen und zu hören
- **C1** Fehlermeldungen sind einfach und präzise formuliert
- **C2** Die Bedienung von musery ist einfach, verständlich und leicht zu erlernen

- **C3** Eine Wiedergabeliste ist leicht zu erstellen
- **D** Die Empfehlungen von musery haben eine hohe Qualität
- **D1** Die Empfehlungen wiederholen sich nicht häufig
- **D2** Man kann neue und interessante Musik entdecken
- **E** Wie bewerten Sie musery im Gesamteindruck?

9.6.1. Auswertung mit Diagrammen

Bei Abschluß der Evaluation haben insgesamt 80 Teilnehmer an der Umfrage teilgenommen, wobei nur 53 Fragebögen vollständig ausgefüllt wurden. Damit wurde die interne Zielvorgabe der Projektgruppe (50 Datensätze) erfüllt. Für die statistische Auswertung wurde *R* verwendet, einer Software zur statistischen Datenanalyse und zur grafischen Darstellung der Daten bzw. der Ergebnisse. Zunächst werden die Ergebnisse aller einzelnen Fragestellungen vorgestellt. Die erste Frage ermittelte das Alter der Teilnehmer an der Umfrage. Das Durchschnittsalter der Teilnehmer lag bei 26,89 Jahren, der jüngste Teilnehmer gab 12 Jahre als Alter an und der älteste Teilnehmer war 78 Jahre alt. Diese Altersangaben waren allerdings starke Ausreißer, was in der Abbildung 33 zu erkennen ist. Anhand des Boxplots lässt sich deutlich erkennen, dass 75 Prozent der Teilnehmer

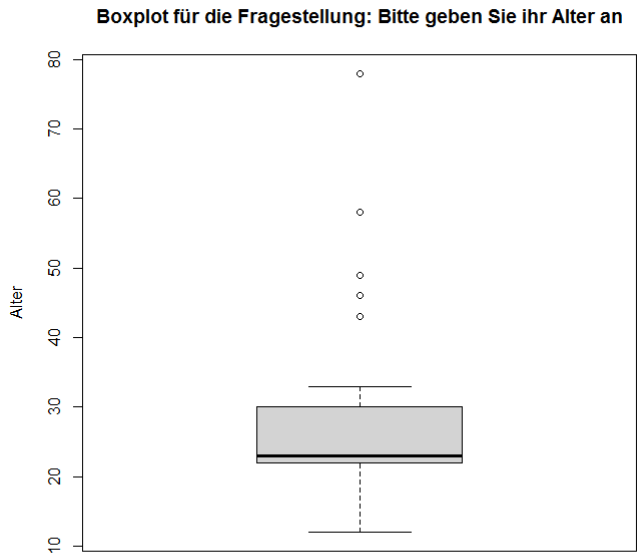

Abbildung 33: Boxplot zum Alter der Teilnehmer

ein Alter zwischen 22 und 30 Jahren haben. Somit wurde das angestrebte Alter der Zielgruppe mit der Umfrage erreicht. Aufgrund der geringen Anzahl an Ausreißern wurden diese nicht aus der Umfrage gefiltert. In der zweiten Frage wurde nach dem Geschlecht der Teilnehmer gefragt. Als Ergebnis kann man festhalten, dass der überwiegende Teil der Teilnehmer (64%) männlich war. In der ersten Fragekategorie Design und Funktionen wurden fünf Fragen gestellt. Die Ergebnisse der Befragung sind in der Abbildung 34 als Balkendiagramme zu sehen.

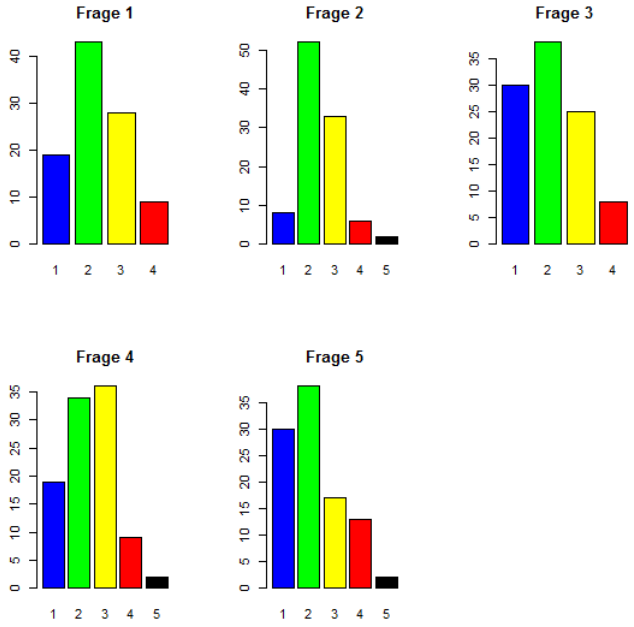

Abbildung 34: Bewertung von Design und Funktionen

Aus den Diagrammen geht hervor, dass das Design gut bewertet wurde. Insbesondere wurden die einheitliche Gestaltung und die unterschiedlichen Ansätze zur Musikempfehlung sehr gut bewertet. Hierzu wurden zwei Kommentare ausgewählt, die das Ergebnis widerspiegeln.

- „Nicht zu überladen und ansprechende Farben“
- „Positiv ist zu bemerken, dass nur die wichtigsten Funktionen angezeigt werden, bzw. nur das was zählt, kein überflüssiger Kram.“

Für die zweite Fragekategorie Systemqualität wurden drei Fragen gestellt, deren Ergebnisse in Abbildung 35 zu sehen sind.

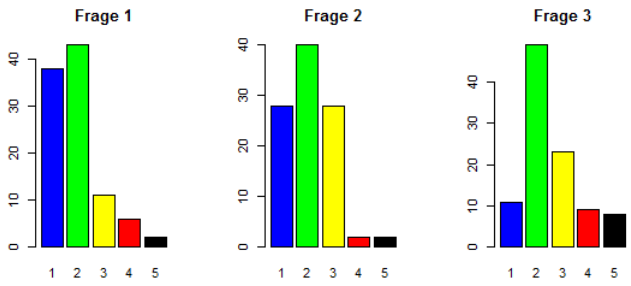

Abbildung 35: Bewertung der Systemqualität

Auch hier wurden allgemein gute Bewertungen erzielt. Auffällig sind dabei allerdings die schlechten Bewertungen bei der dritte Frage (Wichtige Anweisungen, beispielsweise das Löschen einer Wiedergabeliste, werden vor der Ausführung zur Bestätigung angezeigt). Hier wurde am häufigsten (fünf Abstimmungen) die schlechteste Bewertung abgegeben. Anhand der Kommentare lässt sich festhalten, dass in diesem Bereich einige Verbesserungen an musery möglich wären. Nachfolgend wurden die drei aussagekräftigsten Kommentare ausgewählt:

- „Lieder werden von der Playlist direkt gelöscht ohne Nachfrage.
  Wenn ich "back"besuche, verliere ich meine aktuellen Empfehlungen ohne Warnung.“
- „Die Titel sind bei jeder neuen Suche weg“
- „Bei Klick auf den Back-Button ausgehend von der Songauswahl wird leider keine Warnung ausgegeben, dass man seine Playlist verliert wenn man auf den Startbildschirm zurück geht“

Die nächste Fragekategorie befasst sich mit der Benutzerfreundlichkeit von musery und umfasst vier Fragen. Hier wurde besonders die Erstellung der Wiedergabeliste sehr gut bewertet, wie in Abbildung 36 zu sehen ist.

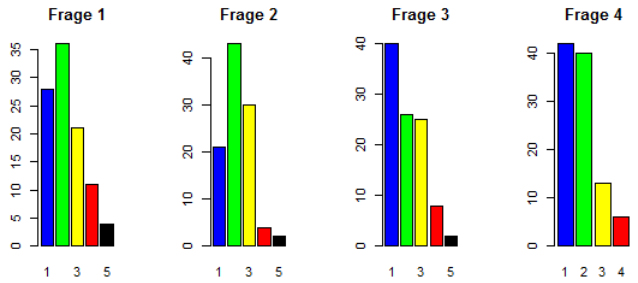

Abbildung 36: Bewertung Benutzerfreundlichkeit

In der Fragekategorie Empfehlungen wurden insgesamt drei Fragen gestellt. Die Ergebnisse sind in der Abbildung 37 zu sehen.

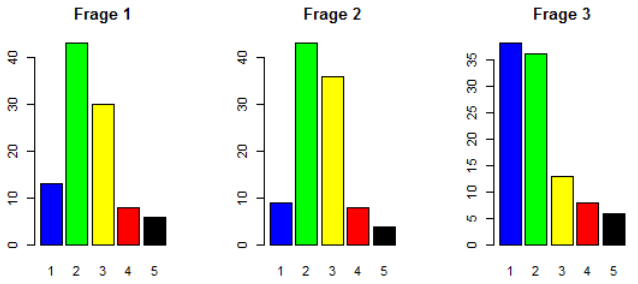

Abbildung 37: Bewertung der Empfehlungsqualität

Hier sind die Ergebnisse allgemein schlechter als in den anderen Kategorien. Dies liegt zum einen an der Zuordnung der moods, die in einzelnen Fällen bemängelt wurde und zum anderen an einem kurzzeitigen Serverproblem während der Umfragephase, auf das die Projektgruppe keinen Einfluß hatte. Hierzu wurden zwei Kommentare ausgesucht.

- „Viele Tags und Moods komplett unpassend. Bsp. Daft Punk - One More Time ist laut Anwendung angry.“
- „Einige Lieder sind meiner Meinung nach nicht unbedingt dem richtigen „mood“zugeordnet. Aber das ist wohl subjektiv ;-) Vielleicht könntet ihr mehr „moods“einführen.“

Die letzte Fragekategorie sollte die allgemeine Zufriedenheit von musery messen, so bestand diese auch nur aus der Frage, wie der Teilnehmer musery im Gesamteindruck bewerten würde. Die Abbildung 38 zeigt das Ergebnis der Frage an. Es ist deutlich zu

sehen, dass musery gut bewertet wurde. 57% haben musery mit gut bewertet, 70% haben mit sehr gut oder gut abgestimmt und nur 11% haben musery als schlecht oder sehr schlecht angesehen.

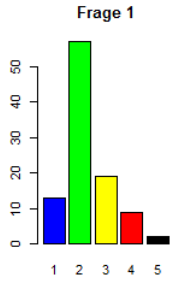

Abbildung 38: Bewertung der allgemeinen Nutzerzufriedenheit

Die Abbildung 39 zeigt eine Übersicht der absoluten Häufigkeiten, deren Grüntöne nach der Häufigkeit abgestuft sind.

Häufigkeitsvergleich der Antworten

	Frage 1	Frage 2	Frage 3	Frage 4	Frage 5	Frage 6	Frage 7	Frage 8	Frage 9	Frage 10	Frage 11	Frage 12	Frage 13	Frage 14	Frage 15	Frage 16
1	10	5	16	10	16	20	15	6	15	11	21	22	7	20	5	7
2	23	27	20	18	20	23	21	26	19	23	14	21	23	19	23	30
3	15	17	13	19	9	6	15	12	11	16	13	7	19	7	19	10
4	5	3	4	5	7	3	1	5	6	2	4	3	4	4	4	5
5	0	1	0	1	1	1	1	4	2	1	1	0	2	3	2	1

Abbildung 39: Heatmap der absoluten Häufigkeiten

Insgesamt kann man sagen, dass musery von den Teilnehmern positiv bewertet worden ist, was sich auch in der letzten Frage widerspiegelt.

9.6.2. Auswertung mit Korrelationen

Mit Korrelationen können Zusammenhänge zwischen Merkmalen quantifiziert werden. Der Korrelationskoeffizient kann Werte zwischen + 1 und − 1 annehmen. Bei einem Wert von + 1 bzw. − 1 besteht ein vollständig positiver bzw. negativer linearer Zusammenhang zwischen den betrachteten Merkmalen. Wenn der Korrelationskoeffizient den Wert 0 aufweist, hängen die beiden Merkmale nicht linear voneinander ab. Bei der Interpretation von Korrelationsstärken ist generell Vorsicht geboten, da eine Korrelation zunächst nur das gemeinsame Auftreten bestimmter Ausprägungen misst, was nicht zwangsläufig auch eine Kausalität bedeuten muss. Für die Umfrage wurde eine Korrelationsmatrix erstellt, die man in Abbildung 40 sehen kann. Die Ergebnisse wurden hierbei auf zwei Nachkommstellen gerundet. Man erkennt, dass jede Variable mit sich selbst eine Korrelation von 1 hat.

	A	A.1	A.2	A.3	A.4	B	B.1	B.2	C	C.1	C.2	C.3	D	D.1	D.2	E
A	1.00	0.67	0.37	0.42	0.51	-0.20	0.18	0.29	0.45	0.31	0.43	0.21	0.40	0.24	0.37	0.50
A.1	0.67	1.00	0.33	0.27	0.42	-0.17	0.03	0.22	0.37	0.14	0.53	0.31	0.34	0.20	0.31	0.44
A.2	0.37	0.33	1.00	0.43	0.52	-0.06	0.29	0.12	0.31	0.18	0.42	0.35	0.32	0.21	0.35	0.34
A.3	0.42	0.27	0.43	1.00	0.44	0.00	0.22	0.17	0.16	0.11	0.31	0.36	0.12	0.12	0.31	0.19
A.4	0.51	0.42	0.52	0.44	1.00	-0.06	0.32	0.31	0.67	0.28	0.49	0.28	0.55	0.33	0.44	0.40
B	-0.20	-0.17	-0.06	0.00	-0.06	1.00	0.21	0.03	0.12	0.35	-0.03	0.03	0.05	0.10	0.01	0.26
B.1	0.18	0.03	0.29	0.22	0.32	0.21	1.00	0.25	0.09	0.43	0.15	0.12	0.28	0.22	0.14	0.18
B.2	0.29	0.22	0.12	0.17	0.31	0.03	0.25	1.00	0.41	0.56	0.23	0.22	0.25	0.00	0.20	0.29
C	0.45	0.37	0.31	0.16	0.67	0.12	0.09	0.41	1.00	0.37	0.46	0.25	0.53	0.14	0.37	0.56
C.1	0.31	0.14	0.18	0.11	0.28	0.35	0.43	0.56	0.37	1.00	0.25	0.20	0.43	0.18	0.32	0.49
C.2	0.43	0.53	0.42	0.31	0.49	-0.03	0.15	0.23	0.46	0.25	1.00	0.36	0.44	0.13	0.34	0.55
C.3	0.21	0.31	0.35	0.36	0.28	0.03	0.12	0.22	0.25	0.20	0.36	1.00	0.21	0.16	0.32	0.29
D	0.40	0.34	0.32	0.12	0.55	0.05	0.28	0.25	0.53	0.43	0.44	0.21	1.00	0.28	0.44	0.67
D.1	0.24	0.20	0.21	0.12	0.33	0.10	0.22	0.00	0.14	0.18	0.13	0.16	0.28	1.00	0.56	0.34
D.2	0.37	0.31	0.35	0.31	0.44	0.01	0.14	0.20	0.37	0.32	0.34	0.32	0.44	0.56	1.00	0.65
E	0.50	0.44	0.34	0.19	0.40	0.26	0.18	0.29	0.56	0.49	0.55	0.29	0.67	0.34	0.65	1.00

Abbildung 40: Korrelationsmatrix

Für eine übersichtlichere Darstellung der Matrix wurden die Werte farblich, geordnet nach den jeweiligen Werten, ausgefüllt. In Abbildung 41 sieht man, dass je größer der positive lineare Zusammenhang ist, desto kräftiger der Blauton des Wertes eingefärbt wird.

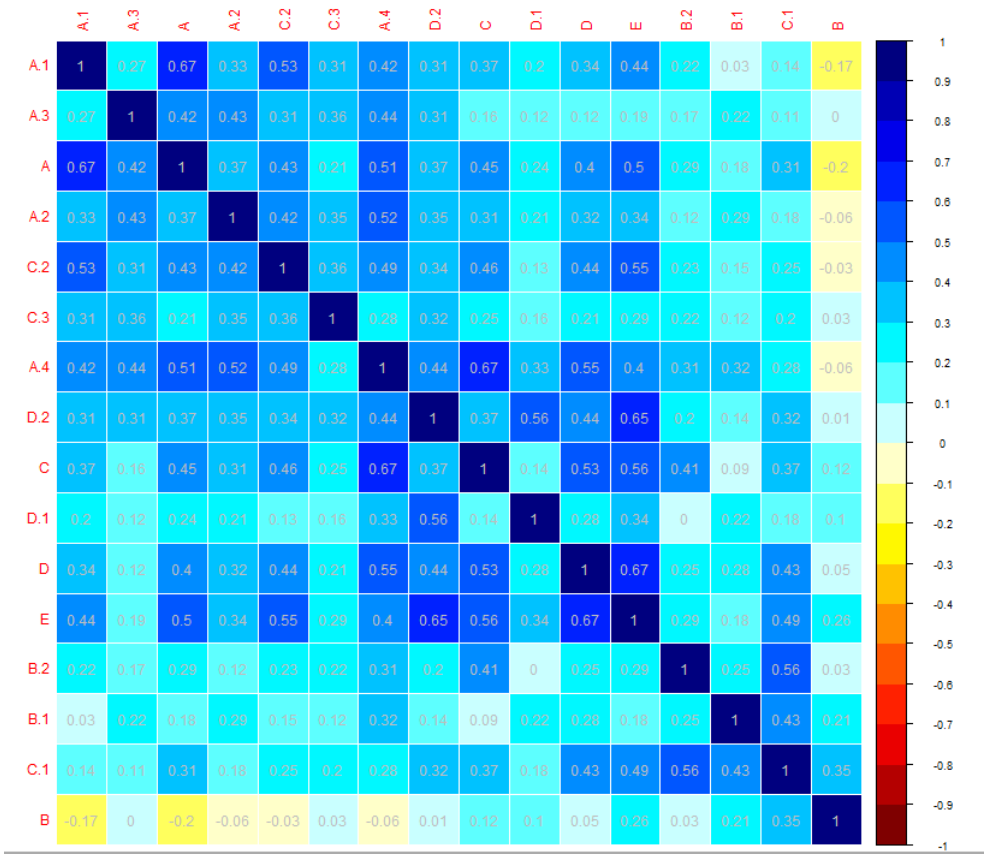

Abbildung 41: Korrelationsfarbmatrix

Für die visuelle Interpretation können auch Punktwolken verwendet werden, die die Richtung der linearen Abhängigkeit widerspiegelt. Korrelieren zwei Merkmale vollständig miteinander (Korrelationskoeffizient 1), dann liegen alle Messwerte auf einer Geraden. Bei einer positiven Korrelation steigt die Gerade, während sie bei einer negativen Korrelation sinkt. Je näher der Korrelationskoeffizient bei 0 liegt, desto kleiner wird der lineare Zusammenhang und kann nicht nehr durch eine Gerade dargestellt werden. Beispielsweise wenn die Messwerte symmetrisch um den Mittelpunkt verteilt sind. Die jeweiligen Punktwolken wurden zur besseren Betrachtung als Ellipsen dargestellt und sind in der Abbildung 42 zusammengefasst.

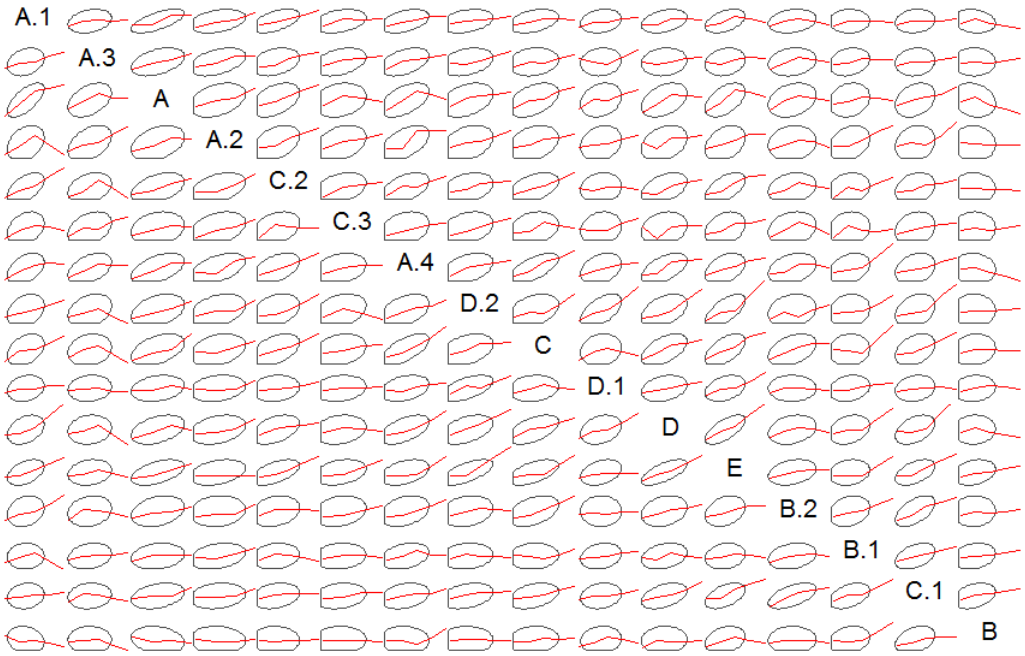

Abbildung 42: Korrelationsellipsen

Insgesamt betrachtet kann man bei der Auswertung eine leicht positive Ausprägung der Korrelation beobachten. Bei 44 Merkmalen (17,2%) konnte man einen starken positiven Zusammenhang zwischen den Merkmalen beobachten. Einen schwachen negativen Zusammenhang konnte man bei zehn Merkmalen (3,9%) beobachten. Der größten Anteil der Beobachtungen mit 202 Merkmalen hatte keinen oder nur einen schwachen Zusammenhang (78,9%). Auffällig war vor allem, dass die negativen Korrelationskoeffizienten ausschließlich bei der ersten Frage der Fragekategorie Systemqualität (Der Seitenauf-

bau erfolgt ohne Verzögerungen) aufgetreten sind. Bei dieser Frage konnte ein negativer Zusammenhang mit folgenden Fragen festgestellt werden:

- Die Gestaltung von musery ist optisch ansprechend
- Das Design unterstützt die Funktionen
- Die einheitliche Gestaltung erleichert die Orientierung
- Mir gefallen die unterschiedlichen Ansätze zur Musikempfehlung, die nach Stimmungen ("moods") oder nach zugehörigen Schlüsselwörtern ("tags") suchen

Allgemein hatten die erste Fragekategorie (Design und Funktionen) und die zweite Fragekategorie (Systemqualität) einen geringen bis keinen Zusammenhang, so dass sich daraus schließen lässt, dass beide Kategorien unabhängig voneinander bewertet wurden und man vermuten kann, dass beide Kategorien sinnvoll ausgewählt wurden. Betrachtet man allerdings die jeweiligen Fragen aus der ersten Kategorie untereinander, so fällt auf, dass vor allem die ersten beiden Fragen (Die Gestaltung von musery ist optisch ansprechend, Das Design unterstützt die Funktionen) einen starken Zusammenhang aufweisen. Dies lässt den Schluss zu, dass die beiden Fragen nicht sinnvoll gestellt wurden, weil sie zu stark miteinander korrelieren. Eine Kombination der Fragen (Die Gestaltung von musery ist optisch ansprechend und unterstützt die Funktionen) hätte in diesem Fall gereicht. In der dritten Fragekategorie (Benutzerfreundlichkeit) konnte man einige positive Zusammenhänge mit der ersten Fragekategorie (Design und Funktionen) feststellen. Folgende Fragen hatten einen schwachen bis starken positiven Zusammenhang:

- Die Gestaltung von musery ist optisch ansprechend
- musery eignet sich gut, um Musik zusammenzustellen und zu hören
- Mir gefallen die unterschiedlichen Ansätze zur Musikempfehlung, die nach Stimmungen ("moods") oder nach zugehörigen Schlüsselwörtern ("tags") suchen
- Die Bedienung von musery ist einfach, verständlich und leicht zu erlernen
- Eine Wiedergabeliste ist leicht zu erstellen

Auffällig sind hierbei die Zusammenhänge zwischen den unterschiedlichen Ansätzen zur Musikempfehlung, der Eignung Musik zusammenzustellen und der Erstellung einer Wiedergabeliste. Hier hätte man die Fragen differenzierter stellen oder andere Aspekte abfragen müssen. Auch der Zusammenhang zwischen der optischen Gestaltung und der Bedienung von musery ist hier auffällig und auch in diesem Fall hätte man über eine entsprechende Umgestaltung der Fragen nachdenken können. In der vierten Fragekategorie (Empfehlungen) konnte man einen starken Zusammenhang mit der fünften Fragekategorie (Allgemeine Zufriedenheit) erkennen. Folgende Fragen hatten einen Einfluss auf die letzte Frage der Umfrage (Wie bewerten Sie musery im Gesamteindruck?):

- Die Empfehlungen von musery haben eine hohe Qualität

- Man kann neue und interessante Musik entdecken

So könnte man die Vermutung aufstellen, dass die Qualität der Empfehlungen und damit die Möglichkeit interessante Musik zu entdecken, einen großen Anteil an der Bewertung von musery hatte.

10. Rückschau

Die PG577 ging über zwei Semester. Wie in vielen Projekten, kam es zu Höhen und Tiefen während der Projektarbeit. Die nächsten beiden Unterkapitel berichten darüber, wann es zu Problemen kam und an welchen Stellen alles reibungslos verlief.

10.1. Probleme in der Praxis

Warum scheitern die meisten Projekte? Diese Frage wird immer wieder gestellt und beantwortet, dennoch gibt es eine große Anzahl von Projekten, die scheitern. Viele Probleme, die dazu führen, sind auch bei unserer Projektgruppe aufgetreten. Oft liegt es an mangelndem Verständnis für die Entwicklung von zugrunde liegenden Prozessen und Problemen.

Grundsätzlich hat sich im Laufe der zwei Semester die Organisation des Projektes verbessert. Fehlte zu Beginn des ersten Semesters noch das nötige Handwerkszeug eines Projektmanagers, so wurde dies im zweiten Semester jedoch korrigiert. Zusätzlich kam hinzu, dass derjenige, der den Projektplan geschrieben hat, nicht gleichzeitig auch der zuständige Projektmanager war. Dies wurde erst sehr spät, gegen Ende des zweiten Semesters, korrigiert. Vielmehr wurde der Projektplan vorher von mehreren geschrieben, was das Ganze verkompliziert hat.

Um dieses Inkonsistenzproblem zu lösen, hätte einer der Betreuer die Position des Projektmanagers einnehmen sollen. Des weiteren hätte der Betreuer Vorgaben für den Projektplan geben und die Umsetzung mit strengen Disziplinarmaßnahmen kontrollieren müssen.

So kam es dazu, dass Aufgaben nicht rechtzeitig vollendet und Deadlines nicht eingehalten wurden, weil die Aufwandsabschätzung und Zeitpläne zu optimistisch waren. Dies lag unter anderem auch daran, dass die technologischen Herausforderungen und der damit verbundene Zeitaufwand falsch eingeschätzt wurden. Bei neuen Technologien, wie JSF, JPA u. s. w., mussten sich die meisten Gruppenmitglieder erst einarbeiten.

Auch die Kommunikation versagte teilweise, zum Beispiel wenn die Anforderungen zwischen den Auftragsnehmern (Gruppenmitgliedern) und Auftraggebern (Betreuern) erst während des Projektverlaufes geklärt wurden. Dies ist das s.g. „Macht-mal-Problem“: Der Umfang wurde vom Auftraggeber nicht eindeutig festgelegt und es wird „irgendetwas“ gemacht. Fehlende Spezifikationen bzw. ein Pflichtheft (inkl. Freigabe) wären hier hilfreich, sowie eine klare Vision, um Missverständnisse über Ziele und Änderungen zu vermeiden.

Die Kommunikation zwischen den Gruppenmitgliedern war teilweise ebenfalls schwierig, vor allem kurz vor der Fertigstellung des Projektes. Durch die fehlende bzw. falsche

Struktur bei der Projektplanung kam es immer wieder zu Unklarheiten und Schwierigkeiten bei der Aufgabenverteilung und Priorisierung von Aufgaben. Die Aufgaben wurden teilweise nicht nach Kompetenzen und Erfahrungen verteilt, was lange Einarbeitungszeiten mit sich zog und die Codequalität beeinträchtigt hat.

Das alles führte unweigerlich zu Konflikten innerhalb der Gruppe und beeinträchtigte die Teamfähigkeit. Hinzu kommt auch die Unzuverlässigkeit und mangelnde Disziplin von Mitgliedern, die eine zeitige Fertigstellung verhindern, sowie die Qualität des Ergebnisses beeinflussten.

10.2. Erfolgserlebnisse und Positives

Die Kompetenzen der einzelnen Mitglieder im Team waren unterschiedlicher Natur. So gab es einige, die sich bei den technischen Methoden sehr gut auskannten und wussten, welche Werkzeuge genutzt werden mussten, um bestimmte Ergebnisse zu erreichen. Auch führten die unterschiedlichen Kompetenzfelder zu einem Erfahrungsaustausch zwischen den Mitgliedern. Dieser wurde durch die regelmäßigen Meetings zusätzlich gefördert. Den Umgang mit neuen und komplexen Tools hat Einigen zwar viel Zeit gekostet, hat aber den gewünschten Lerneffekt erzielt. Ein Erfahrungsaustausch fand auch innerhalb der Aufgabengruppen durch die Kommunikation in den verschiedenen Kanälen statt. So wurden für einzelne Aufgaben Chats in Telegram eröffnet, in denen der Fortschritt besprochen und Probleme gelöst wurden. Auch musste man sich selbst teilweise zurücknehmen, um den Frieden im Projekt zu wahren.

Fortschrittsberichte innerhalb der Meetings waren, durch die Ergebnisorientierung im zweiten Semester, motivierend. Ebenfalls motivierend war das Teambuilding durch gemeinsame Aktivitäten außerhalb des Projekts. Letzteres fand nicht so oft statt, wie eigentlich gewünscht, was aber meistens am Zeitmangel lag und nicht weil die Gruppenmitglieder es nicht gewollt haben.

Für den Projektfortschritt war auch die Visualisierung der Aufgaben mit einem Gantt-Diagramm hilfreich. Es wurde auch ein Architekturdiagramm mit Legende genutzt, was geholfen hat, die Klassen besser umzusetzen. Zum Schluss muss man auch die insgesamt recht gute Bewertung des entwickelten Systems nach Auswertung der Evaluierung erwähnen, was für Betreuer und Teilnehmer ein positives Erlebnis war.

10.3. Perspektive und Weiterentwicklung des Projekts

Das gewünschte Anwendungsszenario wurde umgesetzt. Es bestehen allerdings viele Möglichkeiten, um dieses zu erweitern. Gute Beispiele dafür wären z. B. die Entwicklung einer mobilen App, die Vergrößerung des Datenbankbestandes oder die Erweiterung der stimmungsbasierten Empfehlungen um eine weitere Kategorie. Das letztere ist auch das Aufwendigste und ist aktueller Fokus für viele Forscher auf diesem Gebiet.
Fazit: Die Basis steht, Potenzial nach oben ist aber noch vorhanden.
Eine mobile App könnte so aussehen, dass diese u. a. auf eine bestehende Musiksammlung auf dem Gerät zugreift und diese zur Unterstützung der Musikempfehlungen nutzt. Dadurch kann auch der Datenbankbestand erweitert werden. Eine Verknüpfung mit sozialen

Netzwerken ist ebenfalls denkbar, wobei z. B. der dort angegebene Stimmungsstatus genutzt wird. Die Erweiterung um eine weitere Stimmungskategorie könnte kontextabhängig von der gehörten Musik generiert werden, so dass der Hörer passende Empfehlungen erhält. Bei der Verknüpfung mit sozialen Netzwerken generell, könnte die Favorisierung mancher Inhalte und Themen, wie z. B. Lieder, Filme usw., der Empfehlung von Musiktiteln dienlich sein. Es gibt viel Potenzial, das für die Erweiterung und Verbesserung der Musikempfehlungen genutzt werden kann.

Literatur

[1] Ahmad, Tunku Badariah Tunku, Kamal Basha Madarsha, Ahmad Marzuki Zainuddin, Nik Ahmad Hisham Ismail und Mohamad Sahari Nordin: *Facultys acceptance of computer based technology: Cross-validation of an extended model.* Australasian Journal of Educational Technology, 2010.

[2] Blume, Holger, Bernd Bischl, Martin Botteck, Christian Igel, Rainer Martin, Günther Roetter, Günter Rudolph, Wolfgang Theimer, Igor Vatolkin und Claus Weihs: *Huge music archives on mobile devices.* Signal Processing Magazine, IEEE, 28(4):24–39, 2011.

[3] Bonnin, Geoffray und Dietmar Jannach: *A Comparison of Playlist Generation Strategies for Music Recommendation and a New Baseline Scheme.* In: *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[4] Celma, Òscar: *Foafing the music: Bridging the semantic gap in music recommendation.* In: *The Semantic Web-ISWC 2006*, Seiten 927–934. Springer, 2006.

[5] Celma, Òscar: *Music Recommendation and Discovery.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[6] Celma, Oscar: *Music recommendation and discovery: The long tail, long fail, and long play in the digital music space.* Springer, 2010.

[7] Deinert, Thorsten, Igor Vatolkin und Günter Rudolph: *Regression-Based Tempo Recognition from Chroma and Energy Accents for Slow Audio Recordings.* In: *Audio Engineering Society Conference: 42nd International Conference: Semantic Audio*, Seiten 60–68, 2011.

[8] Fink, Gernot A.: *Mustererkennung.* Vorlesungsskript, 2013.

[9] H.Wixom, Barbara und Peter A. Todd: *A theoretical Integration of User Satisfaction and Technology Acceptance.* Information Systems Research Vol. 16, No. 1, pp. 85-102, 2005.

[10] Jannach, Zanker Markus, Alexander Felfernig und Gerhard Friedrich: *Recommender systems: an introduction.* Cambridge University Press, 2011.

[11] Lidy, Thomas, Andreas Rauber, Antonio Pertusa und José Manuel Iñesta Quereda: *Improving Genre Classification by Combination of Audio and Symbolic Descriptors Using a Transcription Systems.* In: *International Society for Music Information Retrieval*, Seiten 61–66. Citeseer, 2007.

[12] Liu, Jingjing und Xiao Hu: *User-centered music information retrieval evaluation.* In: *Proceedings of the joint conference on digital libraries (JCDL) workshop: music information retrieval for the masses*, 2010.

[13] Lora Asenova, Jaser Bratzadeh, Martin Cmok, N.N., Henning Funke, Michael G., Melissa Hennes, Jan Jansen, Martin Schwitalla und Xiaofei Shi: *PG577 Zwischenbericht.* Technischer Bericht, TU Dortmund, 2014.

[14] Mauch, M. und S. Over.: *Last.fm Driver's Seat Demo*, 2011. aufgerufen am: 06.05.2013.

[15] McKay, Cory: *Automatic music classification with jMIR.* Doktorarbeit, McGill University, 2010.

[16] Pu, Pearl, Li Chen und Rong Hu: *A User-Centric Evaluation Framework for Recommender Systems.* Proceedings of the ACM RecSys 2010 Workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces, 2010.

[17] Schanuel, Isla: *thinkingzygote.com.* website.

[18] Scholkopf, Bernhard und Alexander J. Smola: *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001.

[19] Shakhnarovich, Gregory, Trevor Darrell und Piotr Indyk: *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing).* The MIT Press, 2006.

[20] Song, Yading, Simon Dixon und Marcus Pearce: *A survey of music recommendation systems and future perspectives.* In: *9th international symposium on computer music modelling and retrieval (CMMR 2012)*, Seiten 395–410, 2012.

[21] Vatolkin, I.: *Improving Supervised Music Classification by Means of Multi-Objective Evolutionary Feature Selection.* Doktorarbeit, TU Dortmund, 2013.

A. Musikdatenbank mit 100 Liedern

Lana Del Ray - Video Games
a-ha - Summer Moved On
Toni Braxton - Unbreak my Heart
Limp Bizkit - Rollin‘
Eskimo Callboy - 5$ Bitchcore
Diana Krall - Jingle Bells
Georg Friedrich Händel - Zadok the Priest
Sacred Earth - Ascension
Michael Jackson - Dirty Diana
Bee Gees - Staying Alive
Elton John - I‘m still standing
Moby - Why does my Heart feal so bad
M83 - Midnight City
Calvin Harris - Feel so close
Stromae - Papaoutai
Avicii - Hey Brother
Moloko - The Time is Now
Mary J. Blige - Family Affair
Ray Charles - Hit the Road, Jack
Craig David - 7 Days
Alter Bridge - Blackbird
No Doubt - Don‘t speak
Rise Against - Savior
Papa Roach - Last Resort
Mastodon - Curl of the Burl
Iron Maiden - The Number of the Beast
iwrestledabearonce - tastes like kevin bacon
Stone Sour - Tired
Avenged Sevenfold - Nightmare
Machine Head - Locust
2Pac feat. Danny Boy - I Ain‘t Mad at cha
Eminem feat. Rihanna - The Monster
Gorillaz - Feel Good Inc
Nina Simone - Sinnerman
Chet Baker - My Funny Valentine
The Dave Brubeck Quartet - Blue Rondo a la Turk
Norah Jones - Don‘t know why
Ella Fitzgerald - Dream a little Dream of me
Henry Purcell - Symphony The Indian Queen
Madonna - La Isla Bonita
Nelly Furtado - Big Hoops(Bigger the Better)
Bruno Mars - Locked out of Heaven

Beyonce - Halo
James Blunt - High
Daft Punk - Lose Yourself to Dance
Snap! - Rhythm is a Dancer
Robyn - Dancing on my own
Cascada - Evacuate the Dancefloor
Eric Clapton - Tears in Heaven
R. Kelly - I Believe I can fly
Alicia Keys - Fallin‘
Al Green - Let‘s stay together
Trey Songz - Neighbors know my name
Red Hot Chili Peppers - Californication
Queen - Don‘t stop me now
Paramore - Misery Business
Evanescence - Bring me to life
Shinedown - Second Chance
Dream Theater - The Enemy Inside
Slipknot - Before I Forget
Cro - Einmal um die Welt
50 Cent - Candy Shop
Beastie Boys - Sabotage
Wu-Tang Clan - da mystery of chessboxin’
Macklemore - And we danced
Apathy - Shoot First
Fort Minor feat. Styles of Beyond - Remember the Name
Miles Davis - So what
Art Blakey - Moanin‘
The Dave Brubeck Quartet - Strange Meadow Lark
Michel Chapuis - Andante
William-Boyce - II.Moderato
Mozart - Rex Tremendae
Antonio Vivaldi - Four Seasons - Concerto In E Major
Craig Pruess - Buddham Sharanam
Dave Stringer - Samba Sadashiva
Macy Gray - I try
Guano Apes - Open your Eyes

B. Musikdatenbank mit 40 Liedern

Happy: Pharrell Williams - Happy
Rihanna - Don‘t stop the music
Ai se eu te lego - ai se eu te pego
Helene Fischer - Atemlos
Juli - Die perfekte Welle
The Jackson 5 - ABC
Village People - YMCA
Psy - Gangnam Style
Cro - Whatever
LMFAO - Party Rock Anthem
Angry: Skillet - Monster
Trivium - Built to Fall
Limp Bizkit - Shotgun
Slipknot - Before I Forget
Rage Against the Machine - Bulls on Parade
Stone Sour - Absolute Zero
Disturbed - Down with the sickness
Killswitch Engage - My Curse
System of a Down - Chop Suey!
Parkway Drive - Carrion
Sad: Limp Bizkit - Behind Blue Eyes
Ed Sheeran - I See Fire
Stone Sour - Bother
Xavier Naidoo - Dieser Weg
Adele - Someone like you Lana Del Rey - Video Games
Eric Clapton - Tears in Heaven The
Beatles - Let it be
Johnny Cash - Hurt
Metallica - Nothing Else Matters
Relaxed: Daft Punk - Harder Better Faster Stronger
Rise Against - Swing Life Away
Elaiza - Relaxed
Foo Fightes - Best of You
Clean Bandit - Rather Be
Revolverheld - Ich lass für dich das Licht an
Mr. Probz - Waves
Owl City - Fireflies
Basis - Wonderwall
Gorillaz - Feel Good Inc

C. Anhang MIT-Lizenz

Hiermit wird unentgeltlich jeder Person, die eine Kopie der Software und der zugehörigen Dokumentationen (die „Software“) erhält, die Erlaubnis erteilt, sie uneingeschränkt zu benutzen, inklusive und ohne Ausnahme dem Recht, sie zu verwenden, kopieren, ändern, fusionieren, verlegen, verbreiten, unterlizenzieren und/oder zu verkaufen, und Personen, die diese Software erhalten, diese Rechte zu geben, unter den folgenden Bedingungen: Der obige Urheberrechtsvermerk und dieser Erlaubnisvermerk sind in allen Kopien oder Teilkopien der Software beizulegen. DIE SOFTWARE WIRD OHNE JEDE AUSDRÜCKLICHE ODER IMPLIZIERTE GARANTIE BEREITGESTELLT, EINSCHLIESSLICH DER GARANTIE ZUR BENUTZUNG FÜR DEN VORGESEHENEN ODER EINEM BESTIMMTEN ZWECK SOWIE JEGLICHER RECHTSVERLETZUNG, JEDOCH NICHT DARAUF BESCHRÄNKT. IN KEINEM FALL SIND DIE AUTOREN ODER COPYRIGHTINHABER FÜR JEGLICHEN SCHADEN ODER SONSTIGE ANSPRÜCHE HAFTBAR ZU MACHEN, OB INFOLGE DER ERFÜLLUNG EINES VERTRAGES, EINES DELIKTES ODER ANDERS IM ZUSAMMENHANG MIT DER SOFTWARE ODER SONSTIGER VERWENDUNG DER SOFTWARE ENTSTANDEN.