

42 Media

Arndt Tzscherlich
David Heenemann
David Meinke
Sebastian Sebastian

Inhaltsverzeichnis

1	Einleitung	4
1.1	Aufgabenstellung.....	4
1.2	Ziel der Arbeit.....	4
1.3	Motivation	4
2	Projektorganisation	5
2.1	Vorgehensmodell	5
2.2	Rollendefinition	5
3	Anforderungen.....	5
3.1	Epic.....	5
3.2	User-Stories	7
3.2.1	Standardanwendung.....	7
3.2.2	Integration der Bibliotheksfunktion	7
3.2.3	Integration der Kindersicherung.....	7
3.2.4	Integration der Abspielfunktion	7
3.2.5	Integration der Empfehlungsfunktion	7
3.2.6	Integration der Katalogisierung von externen Medien.....	7
3.3	Allgemeine Qualitätsanforderungen und Qualitätsmanagement.....	8
4	Entwurf.....	9
4.1	Entwicklungsumgebung.....	9
4.1.1	IntelliJ IDEA.....	9
4.1.2	Groovy	9
4.1.3	Grails	9
4.1.4	Bootstrap CSS.....	9
4.2	GUI-Entwurf.....	10
4.3	Architektur.....	12
4.4	Datenmodell	13
5	Implementierung	14
5.1	Datenmodel.....	14
5.2	Kernfunktionen	15
5.2.1	Medienimport	15
5.2.2	Zusammenfassung Künstler & Alben.....	16
5.2.3	LastFM Rest Daten	18
5.3	Durchführung von Tests	20
5.4	Beispiele von Testroutinen	21

5.4.1	Test-SourceCode.....	21
5.4.2	Test-Ausgabe.....	23
6	Zusammenfassung und Ausblick.....	27
6.1	Zusammenfassung	27
6.2	Kritik.....	28
6.3	Ausblick.....	28
7	Anhang.....	29

Abbildungsverzeichnis

Abbildung 1:	Use-Case-Diagramm	6
Abbildung 2:	Vorgegebene Ordnerstruktur in Grails.....	9
Abbildung 3:	Erster GUI-Entwurf	10
Abbildung 4:	Gui-Entwurf, Detailseite.....	11
Abbildung 5:	GUI-Entwurf, weitere Funktionen	12
Abbildung 6:	Architekturmodell	13
Abbildung 7:	Datenmodell	14
Abbildung 8:	Domain in Grails	14
Abbildung 9:	Upload mittels Modalpage	15
Abbildung 10:	Dashboard nach dem Upload	16
Abbildung 11:	Musikübersicht.....	16
Abbildung 12:	Albenübersicht	17
Abbildung 13:	Beispiel SourceCode.....	18
Abbildung 14:	Detailansicht.....	19
Abbildung 15:	Methode um Parameter zu setzen	21
Abbildung 16:	Erste Testmethode.....	22
Abbildung 17:	Zweite Testmethode	22
Abbildung 18:	Falsche Parameterliste.....	23
Abbildung 19:	Kommandozeile erfolgreicher Test.....	23
Abbildung 20:	Erfolgreiche Testauswertung im Browser.....	24
Abbildung 21:	Detailansicht bei erfolgreichen Test	24
Abbildung 22:	Kommandozeile mit Fehlerausgabe.....	25
Abbildung 23:	Testauswertung im Browser mit Fehler	25
Abbildung 24:	Testauswertung im Browser mit Fehlerhinweis.....	26

1 Einleitung

Alle nachfolgenden Bilder, sind als original Datei im GitHub unter Dokumentation/Bilder einsehbar.

1.1 Aufgabenstellung

Im Rahmen des Moduls Softwaretechnik-Projekt ist eine softwaretechnische Problemstellung in Teamarbeit zu analysieren, mit bereits erlernten Methoden Lösungswege aufzuzeigen, sowie Teilaufgaben und deren jeweilige Schnittstellen zu definieren und anschließend zu implementieren. Dieses Dokument stellt die geforderte begleitende Projektdokumentation dar.

1.2 Ziel der Arbeit

Es soll ein qualitativ hochwertiges Media-Center entwickelt werden, das einen bisher nicht erreichten Funktionsumfang konsolidiert. Von besonderer Bedeutung sind dabei die beiden Aspekte:

- Implementierung der Anwendung als Client/Server Lösung,
- Effektive Nachnutzung bereits vorhandener Metadaten

1.3 Motivation

Inspiriert von Kodi, Emby und Plex zielt 42Media auf all die Media Enthusiasten ab, die über die Jahre hin Unmengen an DVDs, BluRays und CDs angesammelt haben. Wer möchte denn noch Abends vor dem Regal stehen und durch die Regale stöbern, welches Medium konsumiert werden soll? Wäre es nicht schön, die Sammlung könnte einfach mit Hilfe eines aktuellen Browsers durchstöbert werden? Egal ob nach Titel, nach Genre, nach Regisseur etc.. Wie oft hat man sich schon die Frage gestellt „Den Schauspieler kenne ich doch. Wo hat der noch mitgespielt?“ Das wollen wir lösen.

2 Projektorganisation

2.1 Vorgehensmodell

Das Vorgehensmodell des Projektes orientiert sich an Scrum, da zu Beginn des Projektes noch nicht ersichtlich ist, wie weit die Implementierung zum Projektende fortgeschritten ist. Die Sprintlänge beträgt zwei Wochen und zu Beginn jedes Sprints wird ein Sprint-Planning durchgeführt. Die Protokolle zu den einzelnen Sprints sind im Anhang zu finden.

2.2 Rollendefinition

Durch die überschaubare Teamgröße wurde auf die klassische Rolleneinteilung von Scrum verzichtet. David Heenemann nahm aufgrund seiner Vorkenntnisse des verwendeten Frameworks zusätzlich zu seiner Rolle als Entwickler auch Aufgaben des Scrum Master und Product Owner. So wurde von ihm zu Projektbeginn, aus unseren zusammengetragenen Anforderungen, das Product Backlog zusammengetragen, priorisiert und daraus Tasks abgeleitet. Ein Impediment Backlog wurde nicht geführt, trotzdem waren die anfänglichen Hindernisse hauptsächlich technischer Natur.

3 Anforderungen

3.1 Epic

42Media bietet Ihnen die Möglichkeit alle Schauspieler eines Filmes aufzulisten und darzustellen wo diese mitgewirkt haben. Mit einem eigenen Profil für Ihre Kinder können Sie bestimmen auf welche Inhalte diese zugreifen können. Anhand dessen, was zuletzt gesehen oder gehört wurde, sollen Empfehlungen unterbreitet werden. Man soll Benachrichtigungen erhalten, wenn für Sie interessante Filme oder Serien im TV ausgestrahlt werden oder bevorzugten Künstler in der Nähe zum Konzert aufspielen.

Ihre Medien liegen noch nicht digital vor? Ein einfacher Assistent unterstützt dabei, die Medien zu digitalisieren. Egal ob DVD, BluRay oder CD. Sie haben Medien bei Amazon Prime, iTunes oder anderen Plattformen erworben und wollen diese ebenfalls durchstöbern? 42Media ist es egal wo sich die Medien befinden: ob auf einem lokalen Laufwerk, auf einem NAS oder in der Cloud. Natürlich will man seine Medien nicht bloß durchstöbern, sondern auch wiedergeben: Auf der Leinwand zu Hause, auf dem Tablet in der Küche, auf dem Smartphone unterwegs.

Folgendes Use-Case-Diagramm fasst die wesentlichen Funktionen zusammen:

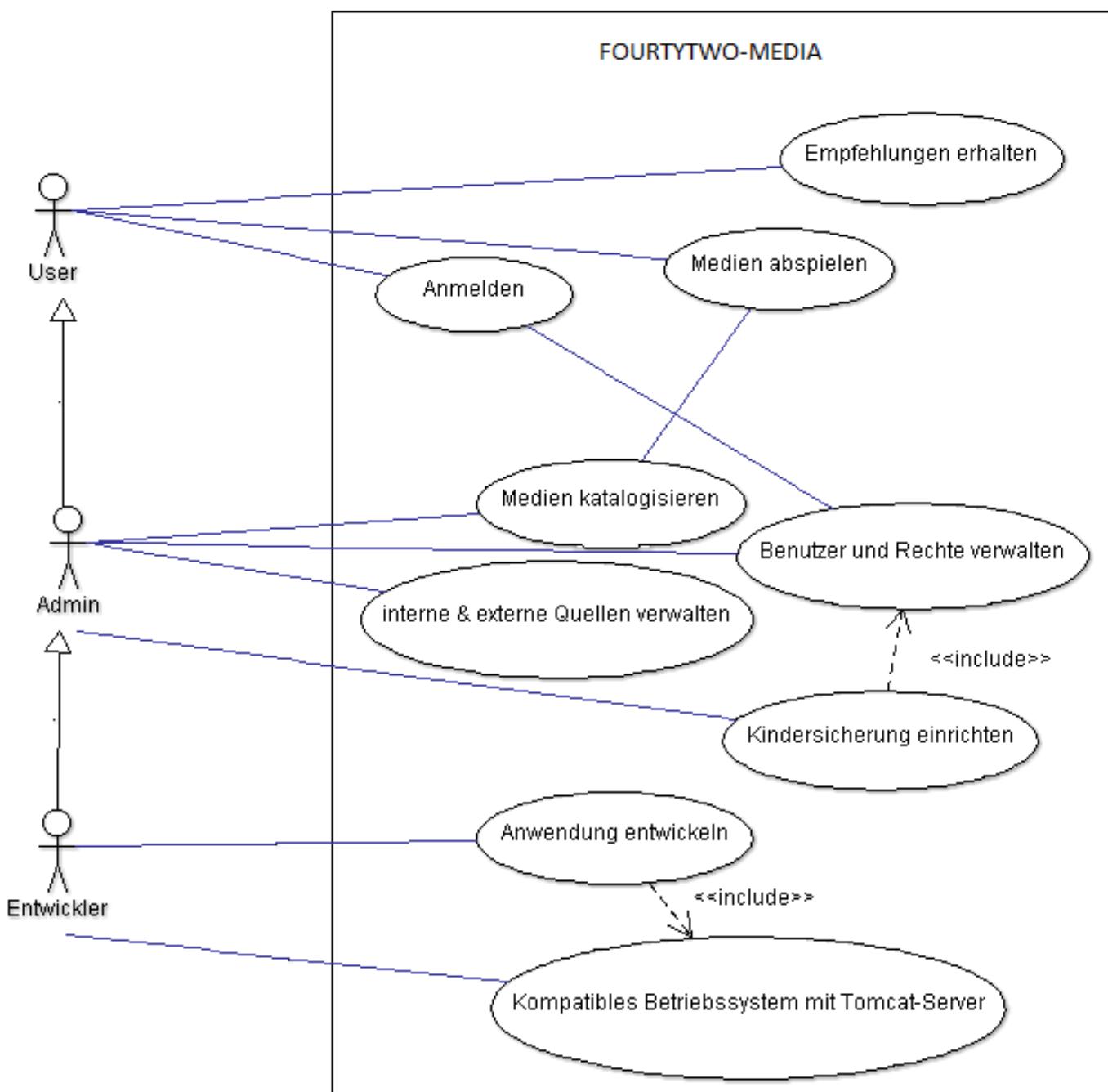


Abbildung 1: Use-Case-Diagramm

3.2 User-Stories

Folgende User-Stories sollen in absteigender Reihenfolge während des Projekts realisiert werden.

3.2.1 Standardanwendung

Art	Muss-Anforderung
Beschreibung	Die Entwickler sollen den Rahmen einer Webanwendung mit definierter Model-, View-, Controller- und Service-Modularität haben.

3.2.2 Integration der Bibliotheksfunktion

Art	Muss-Anforderung
Beschreibung	Der Anwender kann seine bereits vorhandenen Medienverzeichnisse durchsuchen lassen. Dabei werden vorhandene Metadaten für die spätere Nutzung in einer Datenbank abgelegt.

3.2.3 Integration der Kindersicherung

Art	Kann-Anforderung
Beschreibung	Minderjährigen Benutzern soll der Zugriff auf nicht altersgemäße Inhalte verwehrt bleiben. Dafür soll eine Benutzer- und Rechteverwaltung implementiert werden.

3.2.4 Integration der Abspielfunktion

Art	Kann-Anforderung
Beschreibung	Katalogisierte und importierte Medien sollen aus der Anwendung heraus vom Benutzer abspielbar sein.

3.2.5 Integration der Empfehlungsfunktion

Art	Kann-Anforderung
Beschreibung	Anhand dessen, was zuletzt gesehen oder gehört wurde, sollen dem Anwender Empfehlungen zum Thema „weitere Medien“ und „naheliegende Veranstaltungen“ unterbreitet werden.

3.2.6 Integration der Katalogisierung von externen Medien

Art	Kann-Anforderung
Beschreibung	Der Benutzer soll externe Medien und Bibliotheken befreundeter Nutzer in seine Anwendung einbinden können.

3.3 Allgemeine Qualitätsanforderungen und Qualitätsmanagement

Um die Qualität der Software messbar zu machen, sollen folgende Qualitätsanforderungen durch das Produkt erfüllt werden.

Nr.	Beschreibung
1	Der Quellcode soll in maximal 150 Zeichen pro Zeile verfasst werden
2	Alle Klassen, Methoden & Parameter sollen Javadocs-konform kommentiert werden
3	Für den Code soll eine Testabdeckung von 80% erreicht werden
4	<p>Bei der Entwicklung soll sich an den offiziellen Code-Konventionen für die Java Programmiersprache orientiert werden. Insbesondere soll auf die Einhaltung folgender Vorgaben geachtet werden:</p> <ul style="list-style-type: none"> • Der Code für einen neuen Block soll um vier Leerzeichen eingerückt werden. • Es sind Variablen-Namen mit semantischer Bedeutung zu verwenden. • Variablen sollten in „Camel Case“ geschrieben werden. • Klassen sollten Substantive sein, wobei der erste Buchstabe jedes internen Wortes groß geschrieben wird • Methoden sollten Verben sein, die in Camel Case geschrieben sind. • Vor jeder öffnenden geschweiften Klammer soll ein Zeilenumbruch stehen.

Als Qualitätsmanagement-Maßnahme soll Quellcode von einem weiteren Team-Mitglied reviewed werden, bevor er freigegeben wird. Dabei soll sichergestellt werden, dass die Qualitätsanforderungen an den Quellcode erfüllt sind.

4 Entwurf

4.1 Entwicklungsumgebung

Nachfolgend werden die Komponenten vorgestellt, die für die Erstellung des Projektes von Nöten waren.

4.1.1 IntelliJ IDEA

Das Softwareunternehmen JetBrains hat IntelliJ IDEA als Entwicklungsumgebung für Java programmiert. Durch Plug-Ins kann das Programm erweitert werden. Diese Plug-Ins können auch selber geschrieben werden. Seit der Version 7.0 im Jahr 2007 werden auch Groovy und Grails unterstützt. Für Student besteht die Möglichkeit die Ultimate Edition kostenfrei zu nutzen.

4.1.2 Groovy

Groovy erschien 2003 und dient als Programmier- und Skriptsprache, die auf der Java Virtual Machine ausführbar ist. Somit kann eine weitest gehende Plattformunabhängigkeit gewährleistet werden. Groovy ist eine Kombination aus Java, Ruby und Python und gilt als beste integrierte Skriptsprache für die Java Virtual Machine.

4.1.3 Grails

Grails ist ein Web Application Framework basierend auf der Sprache Groovy. Die Webapplikation kann auf jedem Servlet-Container installiert werden. Grails erzeugt für ein Projekt immer die gleiche Verzeichnisstruktur mit den gleichen Namen der Elemente. Somit haben Entwickler eine gute Übersicht über die Projekte. Sobald man ein Projekt angelegt hat, kann man es nach dem Ausführen des Codes schon im Webbrowser betrachten. Änderungen am Code werden nach erneutem Aufrufen der veränderten Webseite direkt sichtbar. Unter dem Ordner „controllers“ werden alle benötigten Controller erstellt. Die Controller nehmen die Webanfrage an, lassen die Anfrage von den Services verarbeiten, und erzeugen daraus eine Ausgabe. Diese bezieht sich auf die dazugehörige „.gsp-Datei“. In dieser Datei steht der HTML- Code geschrieben. Somit folgt Grails dem MVC-Prinzip.

4.1.4 Bootstrap CSS

Twitter entwickelte zu 2011 das freie CSS-Framework Bootstrap. Es wurde für HTML5 und CSS3 entwickelt. Es enthält Vorlagen für Formulare, Button, Tabelle und weitere Elemente. Bootstrap lässt eine Webseite modern und angenehm aussehen, ohne dass es überladen wirkt. Viele Elemente haben abgerundete Ecken und man kann deren Größe und Farbe recht einfach festlegen. Die Verwendung von Bootstrap erspart Entwickler viel Zeit für die Erstellung eines ansprechendes Frontends.

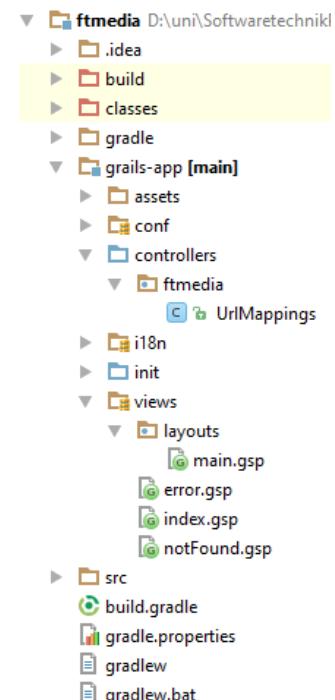


Abbildung 2: Vorgegebene Ordnerstruktur in Grails

4.2 GUI-Entwurf

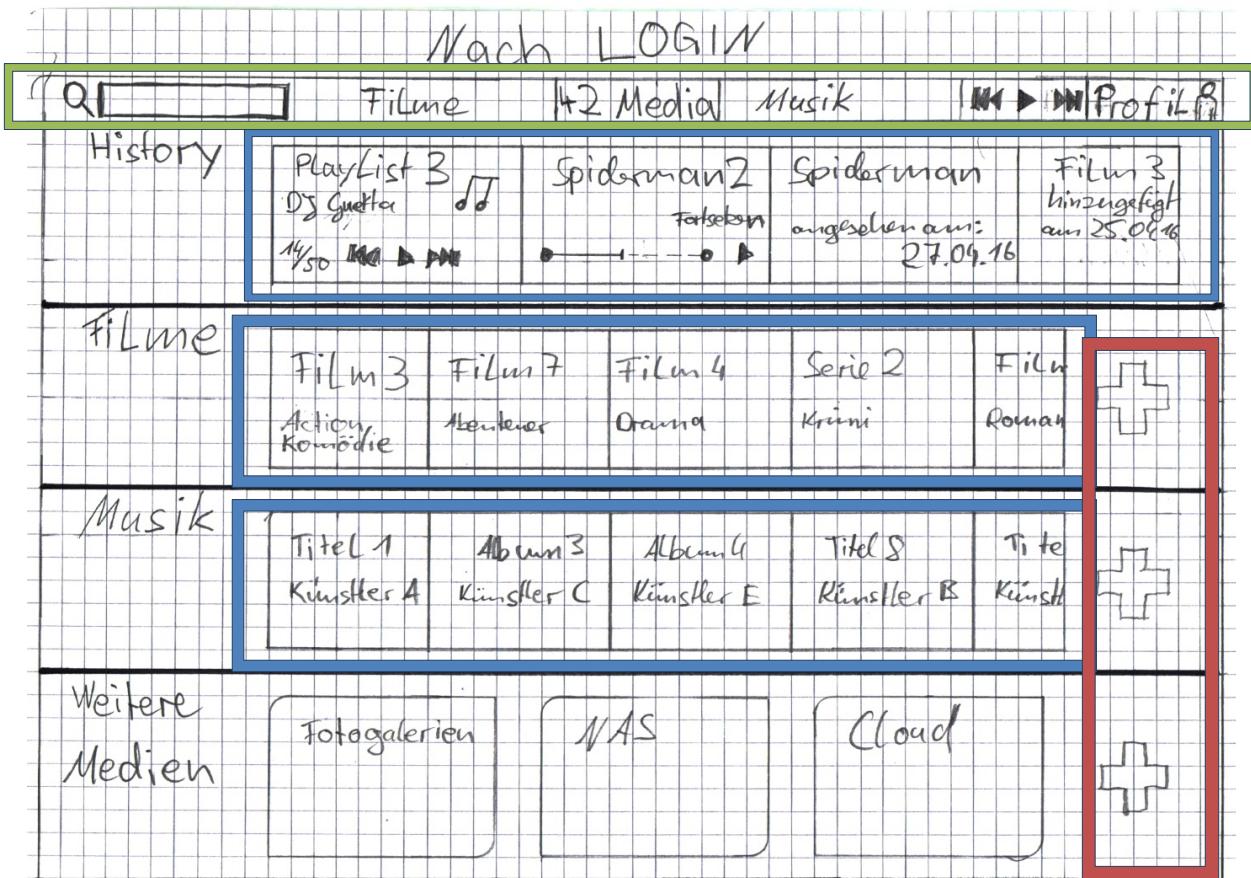


Abbildung 3: Erster GUI-Entwurf

Grüne Markierung: Kopfzeile der Webseite. Links Oben befinden sich die Suchfunktion und der „Filme“-Button. In der Mitte ist der Home-Button „42Media“, der einen wieder zu dieser Webseite zurückbringt. Anschließend folgen der „Musik“-Button und die Abspielfunktion der zuletzt gehörten Titel/Playlists. In der Ecke rechts oben befindet sich das Nutzerprofil.

Die Webseite ist unterteilt in Historie, Filme, Musik und Weitere Medien. Die Blau dargestellten Bereiche sind nach rechts scrollbar. Durch die rot hinterlegten „Plus“-Buttons kann der Nutzer direkt Medien in der gewünschten Kategorie hinzufügen.

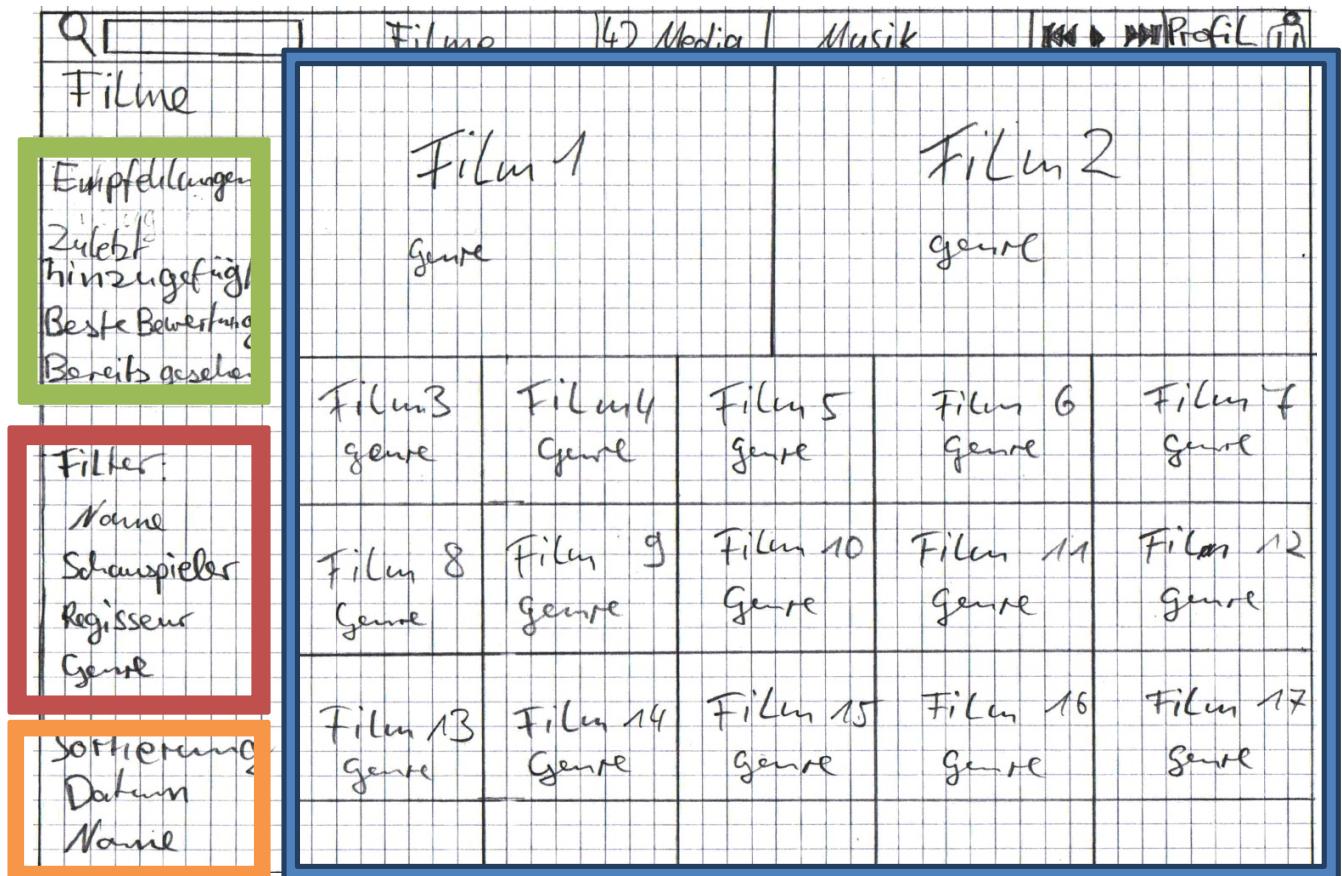


Abbildung 4: Gui-Entwurf, Detailseite

Im blauen Bereich finden sich alle verfügbaren Filme der gewählten Kategorie (grüner Bereich). Dazu kann man noch Filter (roter Bereich) setzen und das Ergebnis noch Sortiert darstellen (orangener Bereich). Der Filmbereich ist nach unten hin scrollbar.

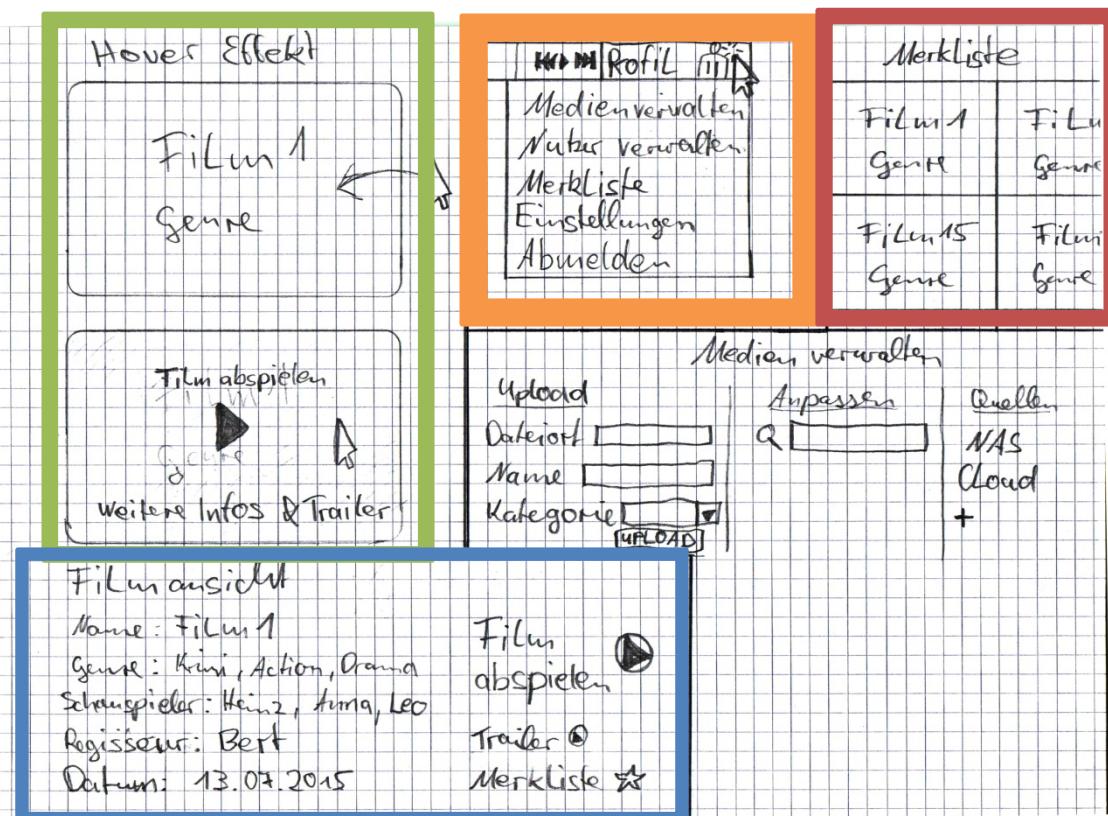


Abbildung 5: GUI-Entwurf, weitere Funktionen

Sobald man mit dem Mauszeiger über ein abspielbares Element herüberfährt (grüner Kasten) erscheinen der „Play“-Button und eine Schaltfläche für weitere Infos. Klickt man diese Schaltfläche landet man in der Detailansicht eines Elementes und erhält alle Informationen dazu (blauer Kasten). Zudem kann man Filme auf die Merkliste setzen, indem der Nutzer den Stern anklickt.

Im orangenen Kasten sieht man das ausgeklappte Profilmenu. Die Merkliste (roter Kasten) ist so aufgebaut wie die Filmeseite und dort finden sich alle gemerkten Filme wieder.

Bei Medien verwalten kann man neue Medien hinzufügen, anpassen oder auch weitere Quellen hinzufügen, die mit in die Weboberfläche eingebunden werden.

4.3 Architektur

Die Anwendung folgt dem Client-Server-Prinzip. Mittels beliebigen Webbrowsers, kann auf die Anwendung zugegriffen werden. Die Webapplikation wird vollständig modular entwickelt. Dazu wird die vom Grails-Framework zur Verfügung gestellte „Inline-Plugin“-Architektur verwendet. Dies bedeutet, dass jede Userstory als eigenständiges Plugin geschrieben wird, und mit den anderen Plugins kommuniziert. Dadurch kann gewährleistet werden, dass neue Funktionalität die bereits bestehende nicht beeinflusst. Damit kann das „loose coupling Prinzip“ eingehalten werden. Um das „DRY-KISS Prinzip“ einzuhalten, wird ein „Core-Plugin“ entwickelt, welches z.B. alle Klassen und Funktionalitäten beinhaltet, die von allen Plugins benötigt werden (z.B. Benutzer, Anmeldung, Registrierung).

Die nachfolgende Grafik beschreibt die vorgestellte Architektur und stellt die geplanten Komponenten dar.

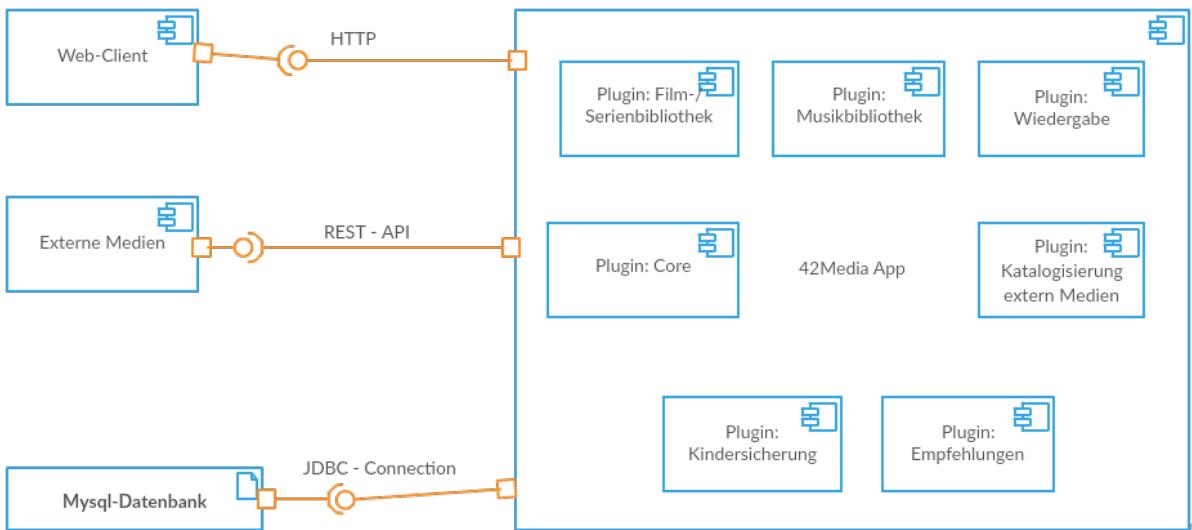


Abbildung 6: Architekturmödell

Wir sehen die Komponente „42Media App“ die auf einem Webserver betrieben wird. Innerhalb dieser App befindet sich die gesamte Anwendungslogik. Der Komponente „Web-Client“ werden über eine HTTP-Verbindung die verarbeiteten Daten zur Verfügung gestellt. Über eine JDBC-Verbindung werden die Model-Klassen auf eine Datenbank gemappt. Zusätzlich werden externe Daten von einschlägigen Informationsquellen einbezogen. Viele Media-Services im Internet stellen eine Rest-API zur Verfügung, mit der man u. a. Daten abfragen kann.

4.4 Datenmodell

Vor der Implementierung wurde anhand der Use-Cases der Anwendung ein Datenmodell entwickelt. Dabei wurde ein Top-Down-Ansatz verwendet. Es wurde ein Objekt-Orientiertes Datenmodell entwickeln, welche durch Mechanismen im Grails-Framework in eine relationale Datenbank übersetzt werden.

Die Model-Klassen sind:

MediaAsset	- Basisklasse aller Medien in der 42Media App
MediaType	-Bestimmt, ob MediaAsset Film, Musik, Serie oder Bild ist
ParentalRating	-Altersfreigabe nach US-Richtlinien
FileObject	-Konkrete Datei, die zu einem MediaAsset gehört
MusicAsset	-Von MediaAsset abgeleitete Musik Entität
VideoAsset	-Von MediaAsset abgeleitete Video Entität
CoverArt	-Cover-Bild eines MediaAsset
Role	- Vordefinierte Rollen für Benutzer

- User – Benutzer, der auf die App abhängig von seiner Rolle zugreifen darf
- UserRole – Konkrete Rolle zum Benutzer

Nachfolgend werden die Klassen des Datenmodells auf einer höheren Detaillierungsebene grafisch dargestellt.

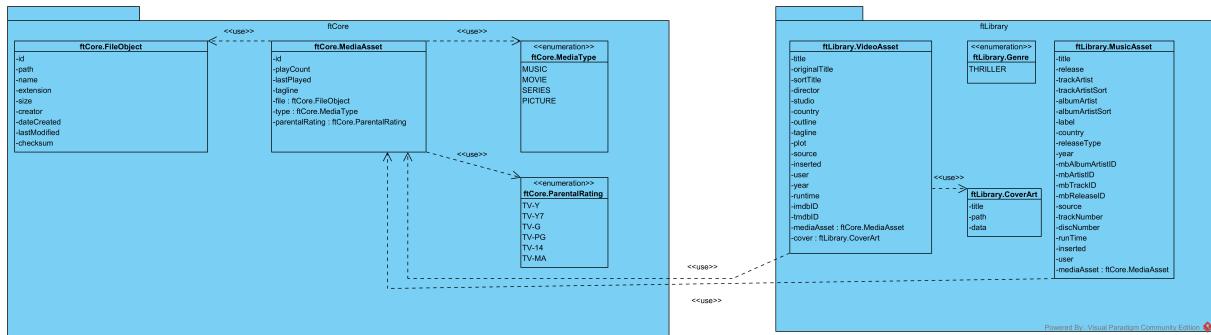


Abbildung 7: Datenmodell

5 Implementierung

5.1 Datenmodel

```

ais:ftmedia-plugins> ftPlay
in: flibrary> MusicAsset
src: /ArtistController.groovy / MusicService.groovy / show.gsp / MusicAsset.groovy
  package flibrary
  import ...
  /**
   * @author: stzschierlich
   * @version: 0.4
   * @date: 2016-06-30
   */
  class MusicAsset {
      String title
      release
      trackArtist
      trackArtistSort
      albumArtist
      albumArtistSort
      String label
      String country
      String releaseType
      String year
      String mbAlbumArtistID
      String mbArtistID
      String mbTrackID
      String mbReleaseID
      String source
      int trackNumber
      int discNumber
      int runTime
      Date inserted
      User user
      statichasMany = {
          mediaAsset:MediaAsset
      }
      static constraints = {
          trackArtist nullable: true, blank: true
          trackArtistSort nullable: true, blank: true, display: false
          albumArtist nullable: true, blank: true, display: false
          albumArtistSort nullable: true, blank: true, display: false
          label nullable: true, blank: true, display: false
          country nullable: true, blank: true, display: false
          releaseType nullable: true, blank: true, display: false
          trackNumber nullable: true, blank: true, display: false
          year nullable: true, blank: true, display: false
          discNumber nullable: true, blank: true, display: false
          runTime nullable: true, blank: true, display: false
          mbAlbumArtistID nullable: true, blank: true, display: false
          mbArtistID nullable: true, blank: true, display: false
          mbTrackID nullable: true, blank: true, display: false
          mbReleaseID nullable: true, blank: true, display: false
          source nullable: false, unique: true, display:false
      }
      String toString() {
          return title
      }
  }
  
```

Als erster Schritt war es zunächst notwendig das entworfene Datenmodell in Form von Domain Klassen in der Anwendung abzubilden. Eine exemplarische Umsetzung soll hier am Beispiel der Model Klasse „MusicAsset“ dargestellt werden:

Wie aus dem Screenshot hervorgeht, werden die Datentypen der Attribute mit Java bzw. Groovy Datentypen angegeben. Um die Umsetzung in datenbankspezifische Datentypen kümmert sich das auf Hibernate basierende „Grails Object Relational Mapping – GORM“. Für den Fall, dass das Standard Typecasting nicht erwartungsgemäß funktioniert, lässt sich dieses auch jederzeit überschreiben.

Mit dem Keyword „hasMany“ wird eine Relation zwischen den Domain-Klassen „MediaAsset“ und „MusicAsset“ definiert. In Abhängigkeit der Relation (1:1 - 1:n - n:m) stehen unterschiedliche Schlüsselworte zur Verfügung. Das Keyword „constraints“ leitet die Definition der Attributeinschränkungen ein.

Abbildung 8: Domain in Grails

Schlussendlich wird noch die „`toString()`“ Methode überschrieben. Obwohl in der Attributliste keine explizite ID Spalte angegeben wird, kann jedes Objekt über seine ID angesprochen werden. Diese wird beim Anlegen automatisch generiert.

5.2 Kernfunktionen

5.2.1 Medienimport

Über einen Dialog ist es bereits möglich, Medien aus Ordnern zu importieren; sowohl Filme wie auch Musik. Bei Filmen werden die Metadaten aus bereits vorhandenen „.nfo, Dateien“ ausgelesen. Dabei handelt es sich um eine XML Struktur, die von allen referenzierten Anwendungen verarbeitet wird. Dafür öffnet man einen Dialog, über den man einen lokalen Pfad oder einen UNC Pfad angeben kann.

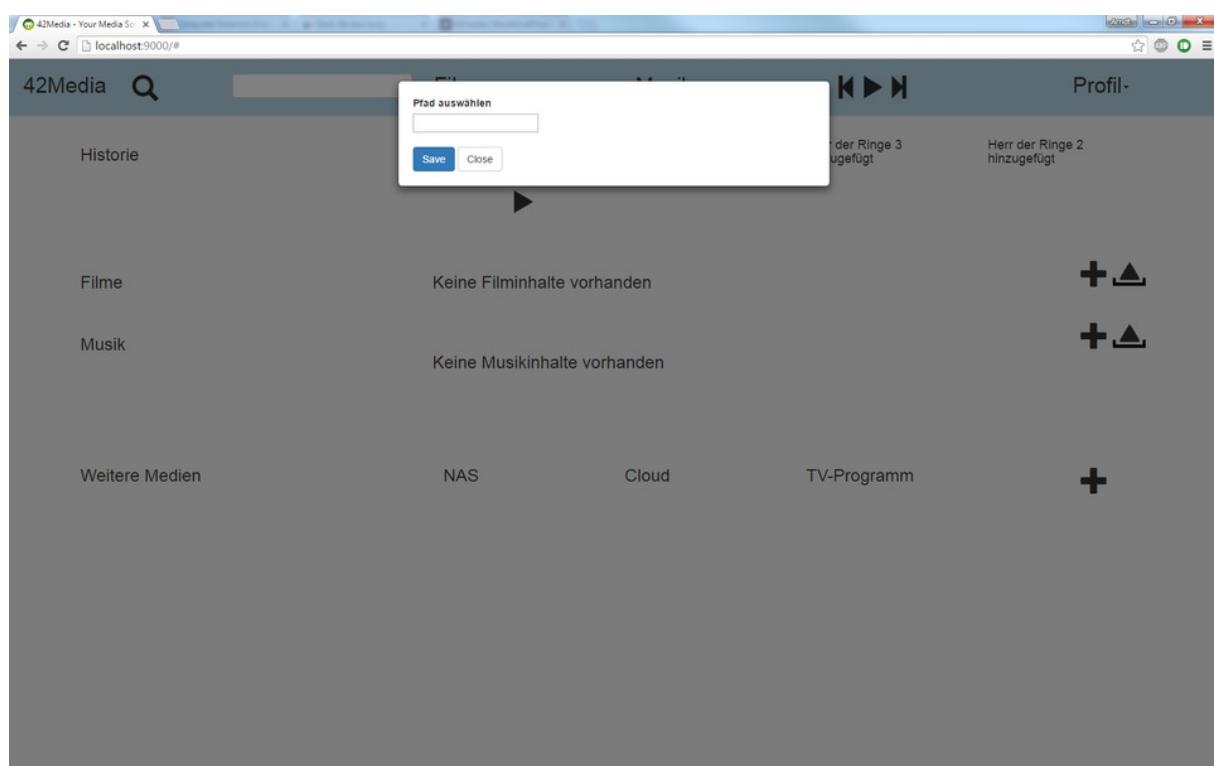


Abbildung 9: Upload mittels Modalpage

Hier haben wir Filme aus einer lokalen Ordnerhierarchie importiert. Die Filme und die dazu passenden Covers werden in der Standardansicht folgendermaßen dargestellt:

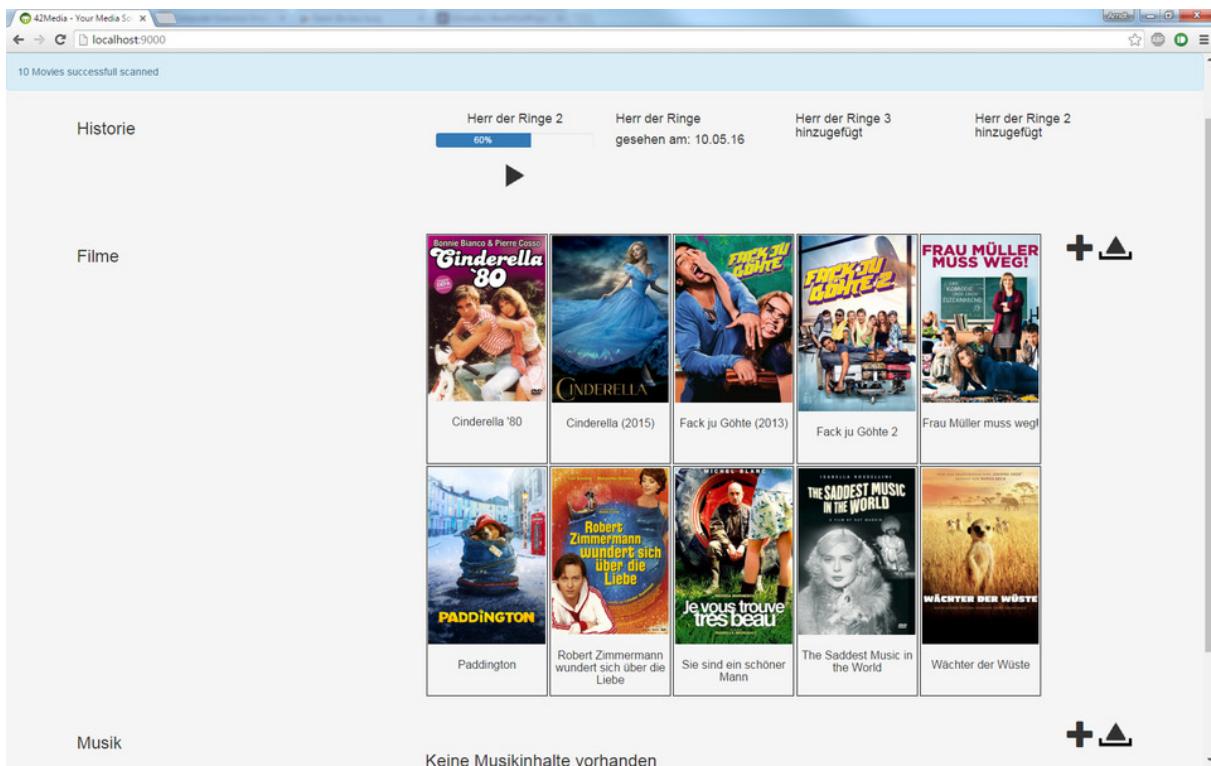


Abbildung 10: Dashboard nach dem Upload

Wie man sieht, werden die Cover und die Titel unter der Rubrik „Filme“ nach dem Import dargestellt.

5.2.2 Zusammenfassung Künstler & Alben

Ähnlich der Videoübersicht gestaltet sich auch die Ansicht der Künstler und Alben. Jeweils fünf Einträge nebeneinander, nach Möglichkeit mit Artwork, gespeist durch last.fm, versehen. Dabei öffnet ein Klick auf den Künstler die Detailseite zum Künstler; versehen mit Daten zum Künstler, sowie eine Liste all seiner Veröffentlichungen in der eigenen Sammlung.

The screenshot shows a music overview page. On the left, there are filter options for "Musik hinzufügen", "Zuletzt hinzufügt", "Beste Bewertung", and "Bereits gehört". Below these are dropdown menus for "Filter" (Album, Sänger, Produzent, Genre) and "Sortierung" (Name, Datum). The main area displays five artist profiles with their names and small profile pictures. From left to right, the artists are Adam Green, Modeselektor, Morcheeba, 50 Cent, and Alexisonfire.

Abbildung 11: Musikübersicht

Die Albenübersichtsseite bietet, ebenfalls mit fünf Einträgen nebeneinander, eine Übersicht über alle Alben in der eigenen Sammlung. Dabei ist die Auflistung standardmäßig nach Künstler\Veröffentlichungsjahr\Albumtitel sortiert.

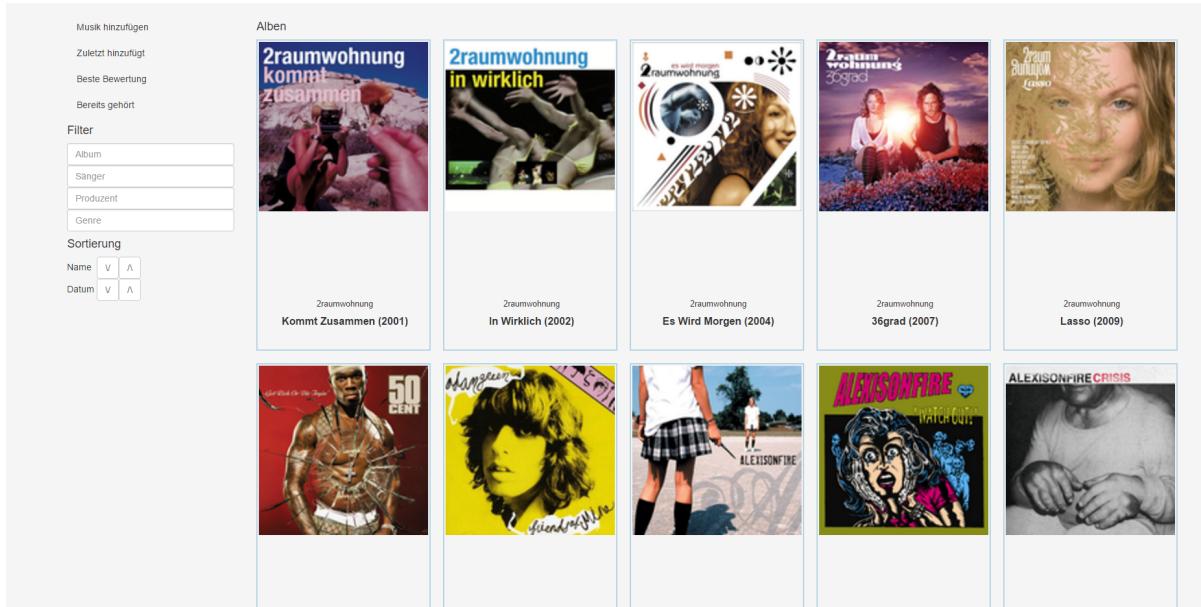


Abbildung 12: Albenübersicht

Beiden Ansichten gemein ist, dass sie nicht direkt aus Domain Objekten erzeugt werden, sondern aus Abfrageergebnissen der Datenbank. Während sich das zunächst in der Ansicht nicht kenntlich macht, ist dieser Umstand zum gegenwärtigen Zeitpunkt noch mit verschiedenen Einschränkungen versehen. Kann zum Beispiel bei der Darstellung von Domain Objekten für die Seitenweise Darstellung der Ergebnisse auf im Framework vorhandene Funktionalitäten zurückgegriffen werden, so muss dies für eine auf einer Abfrage basierten Ergebnisliste von Hand implementiert werden. Ein Punkt auf dem im Kapitel Zusammenfassung und Ausblick ausführlicher eingegangen wird.

5.2.3 LastFM Rest Daten

```

Map<String, String> getMetaByAlbum(Map<String, String> callParams) throws GenericSignatureFormatError
{
    HashMap params = new HashMap();
    Map metaMap = new HashMap();
    JSONObject json

    /*
    Setze default last.fm API parameter
    */
    params.put("api_key", apiKey)
    params.put("autocorrect", 1)
    params.put("format", "json")
    params.put("method", "album.getInfo")

    /*
    Setze Suchparameter
    mittels MusikbrainzID (mbid)
    oder mittels Künstler, Album (artist, album)

    wird keine gültige Parameterangabe verwendet, wird eine
    Exception geworfen
    */
    if(callParams.containsKey('mbid'))
    {
        params.put("mbid", callParams.mbid)
    }
    else if(callParams.containsKey("artist") && callParams.containsKey("release"))
    {
        params.put("artist", callParams.artist)
        params.put("album", callParams.release)
    }
    else
    {
        throw new GenericSignatureFormatError("getMetaByAlbum wurde falsch aufgerufen")
    }

    /*
    Aufruf der last.fm API
    schlägt der Aufruf fehl, wird eine Exception gefangen und der Methodenaufruf
    liefert NULL zurück
    */
    try
    {
        json = restService.getRequest(url ,params)
    }
    catch (Exception e)
    {
        println(e.getMessage())
        return null
    }

    /*
    Mapping der abgerufenen Metadaten auf das Rückgabebjekt
    */
    metaMap.put("AlbumKünstler", json.album.artist)
    metaMap.put("Albumtitel", json.album.name)
    metaMap.put("AlbumArt", json.album.image.last().get('#text'))
    metaMap.put("Aufrufe", json.album.playcount)
    metaMap.put("Zuhörer", json.album.listeners)
    def toptags = ""
    json.album.tags.tag.each{
        tag ->
        toptags <= tag.get("name")
        if(tag != json.album.tags.tag.last()){
            toptags <= ", "
        }
    }
    metaMap.put("Tags",toptags)
    metaMap.put("Link",json.album.url)
    metaMap.put("ReleaseDate", json.album.releasedate)
    metaMap
}

```

Abbildung 13: Beispiel SourceCode

Datenbank abgegrenzt, sodass der Nutzer jederzeit seine zusätzlichen Informationen auf einen Blick hat.

Um möglichst viele Informationen zu den Musik-Dateien zu gewinnen wurde der Rest-Webservice implementiert. Die Rest Schnittstelle ist in der Anwendung exemplarisch an die API von last.fm angebunden.

Über diese hat man die Auswahl aus über 50 verschiedenen Methoden.

In unserer Web-anwendung werden drei davon benutzt: album.getInfo, artist.getInfo und track.getInfo. Die Abfrage der Informationen erfolgt über JSON, optional auch XML. Zur Abfrage von einem Künstler müssen die Paramater der Name, der Last.fm API-key, das Format und die Methode artist.getInfo übergeben werden. Nach der Abfrage wird nur ein Teil der gesamten Rest-Informationen für die Darstellung auf der Webseite verwendet.

Der Aufruf der Rest Daten erfolgt indem der Nutzer auf ein Album, einen Titel oder einen Künstler klickt. Bei einem Titel werden die Informationen zu dem Titel und zu dem Künstler gleichzeitig herausgesucht. Die Rest Daten sind optisch von den Daten aus der

U Not Like Me anzeigen

Eigene Daten

1. Track Artist	50 Cent
2. Track Artist Sort	50 Cent
3. Album Artist	50 Cent
4. Album Artist Sort	50 Cent
5. Label	Aftermath Entertainment
6. Country	US
7. Release Type	album
8. Track Number	18
9. Year	2003-02-04
10. Disc Number	1
11. Run Time	256
12. Mb Album Artist ID	8e68819d-71be-4e7d-b41d-f1df81b01d3f
13. Mb Artist ID	8e68819d-71be-4e7d-b41d-f1df81b01d3f
14. Mb Track ID	3914ae8e-3b64-3a5b-bc37-9ff5dddf367d
15. Mb Release ID	M:\Emby\50 Cent\Get Rich Or Die Tryin' - (2003) [ALBUM]\18 - 50 Cent - U Not Like Me.mp3
16. Source	05.07.2016 22:06:15 MESZ
17. Inserted	Get Rich Or Die Tryin'
18. Release	U Not Like Me
19. Title	admin
20. User	

REST-Daten

```

Interpret: 50 Cent
Genre-Tags: rap, Hip-Hop, Gangsta Rap, 50 Cent, G-Unit
Zuhörer: 42888
Titel: U Not Like Me
lastFM-Link: http://www.last.fm/music/50+Cent/_/U+Not+Like+Me
Album: Get Rich Or Die Tryin'
Aufrufe: 132722
AlbumArt: http://img2.ak.last.fm/i/u/34s/1fb1ff13f0d54b9299a2c189fe171217.png
Bild: http://img2.ak.last.fm/i/u/arQ/9afac043f684f8faeb91772c14246d3f.png
Ähnlich: G-Unit, Lloyd Banks, Tony Yayo, D12, The Game
Vita: Curtis James Jackson III (born July 6, 1975 in South Jamaica, Queens, New York), better known by his stage name 50 Cent, is a Grammy award nominated rapper, actor, singer, entrepreneur, author and founder of the hip hop group and label G-Unit. After leaving drug dealing to pursue a rap career, he released his debut album Guess Who's Back Again in 2002. He was discovered by Eminem and Dr. Dre and then signed to Interscope Records. He has since released Get Rich or Die Tryin' (2003) <a href="http://www.last.fm/music/50+Cent">Read more on Last.fm</a>
Hörer: 2488106

```

Abbildung 14: Detailansicht

Der Nutzer hat nun zusätzliche Informationen und unter anderem auch den Link zu dem Song auf der Last.fm Webseite.

5.3 Durchführung von Tests

Um den Code während der Entwicklung testen zu können, müssen verschiedene Tests geschrieben und ausgeführt werden. Das Framework „Grails“ bietet eine integrierte Test-Suite an, mit Hilfe derer man schnell und einfach verschiedene Arten von Tests schreiben und durchführen kann.

So ist es möglich unter anderem folgende Tests zu schreiben:

- Unit-Tests
- Integration-Tests
- Functional-Tests

Die Unit-Test können unter anderem folgende Testmöglichkeiten abdecken:

- Unit-Test um Controller zu testen
- Unit-Test um Domains (Klassen) zu testen
- Unit-Test um URL-Mappings zu testen
- Unit-Test um Tags für GSP-Seiten zu testen
- ...

Wenn über das Grails-Framework Methoden wie „CREATE“, „UPDATE“, „DELETE“, automatisch zu den Domains erzeugt werden, legt das Framework automatisch Controller- und Domain-Unit Tests an, die zum Teil gefüllt sind. So kann ohne großes zu tun, eine hohe Testabdeckung erreicht werden. Dennoch müssen und wurden eigene Testroutinen geschrieben werden.

Nachfolgend wird ein selbstgeschriebenes Beispiel gezeigt, dass eine Domain-Klasse testet. Dabei wird zum einen versucht, ein Modell zu erzeugen, ohne das Übergabeparameter verwendet werden, und einmal mit. Ohne Übergabeparameter, darf es nicht möglich sein, ein Model zu erzeugen. Mit Parametern muss ein gültiges Modell erzeugt worden sein, die Seite umgeleitet und dem Benutzer eine Nachricht angezeigt werden.

5.4 Beispiele von Testroutinen

5.4.1 Test-SourceCode

Es wurden unter anderem ein Domain-Test geschrieben, der aus mehreren Teilen besteht.

In dem ersten Teil, der aus der Methode „populateValidParams()“ besteht, werden die Parameter gesetzt, die benötigt werden, um später ein gültiges „MusicAsset-Modell“ zu erzeugen. Es werden mindestens die Parameter benötigt, die den Attributen in der Datenbank entsprechen, die auf „nullable=false“ gesetzt sind.

Das nachfolgende Bild zeigt die Parameter, die gesetzt wurden, um erfolgreich ein Modell erzeugen zu können.

```
package ftlibrary

import ftcore.security.User
import grails.test.mixin.*
import spock.lang.*
import ftlibrary.MusicAsset

@TestFor(MusicAssetController)
@Mock(MusicAsset)
class MusicAssetControllerSpec extends Specification {

    // Relevante Parameter setzen, die notwendig sind, um ein valides Modell von
    // MusicAsset zu erzeugen
    def populateValidParams(params) {
        assert params != null

        params.title      = 'Test Music'
        params.trackArtist = 'Test Music'
        params.release    = 2016
        params.inserted   = new Date()
        params.user       = new User(username: 'admin', password: 'password')
    }
}
```

Abbildung 15: Methode um Parameter zu setzen

Das nachfolgende Bild zeigt den SourceCode des ersten Tests, mit dem Namen „test saving an invalid MusicAsset“. Die Tests, die in Grails geschrieben werden, folgen immer der Logik, dass etwas unter einer bestimmten Bedingung getestet wird. In diesem Test sind die Bedingungen, dass ein POST-Request gesendet wird, der aber keine Parameter enthält. Es wird daraufhin versucht, ein neues Modell zu erzeugen und zu speichern.

```
// Es wird versucht ein Modell, ohne Übergabeparameter, zu erzeugen
// Verhält sich das Programm richtig, darf kein Modell erzeugt werden,
// und die Create View muss erneut eingeblendet werden
void "test saving an invalid MusicAsset"() {
    when:
        request.contentType = FORM_CONTENT_TYPE
        request.method = 'POST'

        def musicAsset = new MusicAsset()
        musicAsset.validate()

        controller.save(musicAsset)

    then:
        model.musicAsset!= null
        view == 'create'
}
```

Abbildung 16: Erste Testmethode

Der zweite Test überprüft, ob ein „MusicAsset-Modell“ angelegt wird, wenn die entsprechenden Parameter vorhanden sind. Dazu wird ein POST-Request simuliert, und die zuvor festgelegten Parameter übergeben. Das erstelle Modell wird daraufhin simuliert gespeichert. Der Test ist erfolgreich, wenn ein Modell vorhanden ist, eine Benachrichtigung an den Benutzer ausgegeben, und die Seite umgeleitet wurde.

```
// Es wird ein Modell mit gültigen Übergabeparametern erzeugt
// Verhält sich das Programm richtig, ist der Counter = 1, es wird auf die Startseite umgeleitet
// und eine Flash-Nachricht ausgegeben
void "test saving a valid musicAsset"() {
    when:
        request.contentType = FORM_CONTENT_TYPE
        request.method = 'POST'
        response.reset()

        populateValidParams(params)
        def musicAsset = new MusicAsset(params)

        controller.save(musicAsset)

    then:
        response.redirectedUrl == '/'
        controller.flash.message != null
        MusicAsset.count() == 1
}
```

Abbildung 17: Zweite Testmethode

Das nächste Bild zeigt die Parameter, die gesetzt wurden, um einen Fehler zu simulieren und eine Erstellung eines Modells zu verhindern. In diesem Fall muss der Test „Test saving a valid MusicAsset“ fehlschlagen, da nicht alle zwingend benötigten Parameter vorhanden sind.

```
// Relevante Parameter setzen, die notwendig sind, um ein valides Modell von
// MusicAsset zu erzeugen
def populateValidParams(params) {
    assert params != null

    params.title          = 'Test Music'
    params.trackArtist    = 'Test Music'
    params.release        = 2016
    //params.inserted      = new Date()
    //params.user          = new User(username: 'admin', password: 'password')
}
```

Abbildung 18: Falsche Parameterliste

5.4.2 Test-Ausgabe

Um eine Anwendung mit den zuvor definierten Tests zu testen, wird auf der Kommandozeile der Befehl „grails test-app“ ausgeführt. Alle hinterlegten Tests werden automatisch durchgeführt, und die Ergebnisse auf der Kommandozeile ausgegeben.

Eine Mögliche Ausgabe, die nach der Durchführung von allen Tests zu der Anwendung „42Media“ ausgegeben wurde, sieht, wenn alle Tests erfolgreich waren, folgendermaßen aus:

BUILD SUCCESSFUL

| Tests PASSED

Abbildung 19: Kommandozeile erfolgreicher Test

Neben der Kommandozeile, kann auch ein generierter Test-Report über einem Browser angezeigt werden. Dieser Report zeigt auf der Startseite eine Übersicht über die geläufigen Tests an, und verdeutlicht in %, wie viele Tests erfolgreich waren und wie viele nicht. Daneben gibt es eine Zeitangabe, die verdeutlicht, wie lange der komplette Test lief.

Test Summary



Packages

Classes

Package	Tests	Failures	Ignored	Duration	Success rate
ftlibrary	2	0	0	3.929s	100%

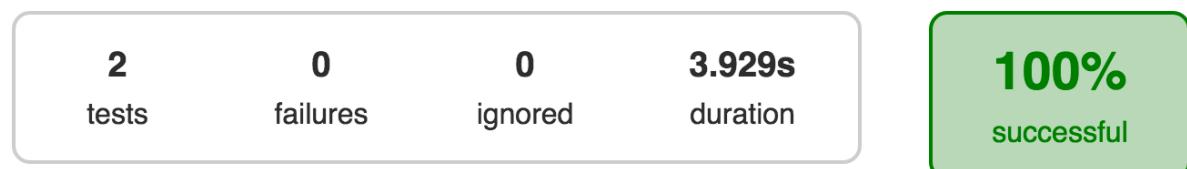
Generated by [Gradle 2.9](#) at 06.07.2016 17:35:57

Abbildung 20: Erfolgreiche Testauswertung im Browser

Wenn man auf den Link, der den Namen des Packages darstellt, klickt, können Details zu den einzelnen Tests angesehen werden.

Class ftlibrary.MusicAssetControllerSpec

[all](#) > [ftlibrary](#) > MusicAssetControllerSpec



Tests

Test	Duration	Result
test saving a valid musicAsset	0.339s	passed
test saving an invalid MusicAsset	3.590s	passed

Generated by [Gradle 2.9](#) at 06.07.2016 17:35:57

Abbildung 21: Detailansicht bei erfolgreichen Test

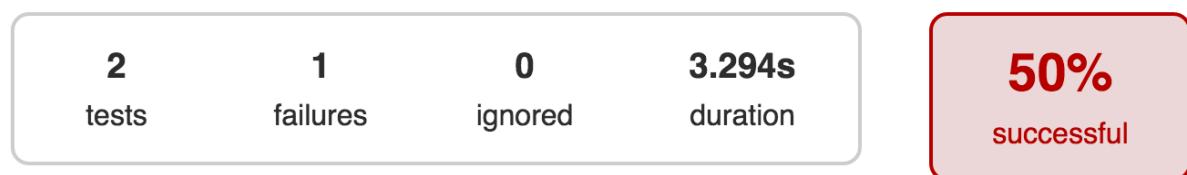
Im Fehlerfall wird der geworfene Fehler ebenfalls auf der Kommandozeile ausgegeben und sieht folgendermaßen aus:

```
ftlibrary.MusicAssetControllerSpec > test saving a valid musicAsset FAILED  
org.spockframework.runtime.SpockComparisonFailure at MusicAssetSpec.groovy:59  
  
2 tests completed, 1 failed  
:ftmedia:test FAILED  
  
FAILURE: Build failed with an exception.
```

Abbildung 22: Kommandozeile mit Fehlerausgabe

Auch im Fehlerfall können die Details zu den einzelnen Tests angezeigt werden. Dabei wird dem Entwickler zusätzlich dargestellt, an welcher Stelle das Programm sich nicht wie erwartet verhalten hat.

Test Summary



Failed tests

Packages

Classes

[MusicAssetControllerSpec. test saving a valid musicAsset](#)

Generated by [Gradle 2.9](#) at 06.07.2016 17:37:21

Abbildung 23: Testauswertung im Browser mit Fehler

Class **ftlibrary.MusicAssetControllerSpec**

[all](#) > [ftlibrary](#) > MusicAssetControllerSpec



Failed tests

Tests

test saving a valid musicAsset

Condition not satisfied:

```
MusicAsset.count() == 1
|           |
0            false
```

```
at ftlibrary.MusicAssetControllerSpec.test saving a valid musicAsset(MusicAssetSpec.groovy:59)
```

Abbildung 24: Testauswertung im Browser mit Fehlerhinweis

6 Zusammenfassung und Ausblick

6.1 Zusammenfassung

Zu Beginn des Moduls bestand die erste Herausforderung darin ein geeignetes Softwareprojekt zu benennen, welches im Rahmen dieses Moduls (teilweise) umgesetzt werden sollte. Die Teamzusammensetzung war zu weiten Teilen bereits im Vorfeld geklärt worden. Die Wahl wurde letzten Endes auf Grundlage einer persönlichen Interessenslage eines Teammitgliedes gemeinschaftlich getroffen. Die nächste wichtige Entscheidung bestand in der Wahl der Programmiersprache. Mit der schwachen Anforderung nach einer möglichst weitgehenden Plattformunabhängigkeit, stand dem Vorschlag die Lösung mit Grails/Groovy zu implementieren nichts im Weg. Obwohl nur ein Teammitglied bereits Erfahrung mit dieser Sprache sammeln konnte, waren alle Beteiligten offen, da es sich dabei um eine auf JAVA basierende Sprache handelt. Mit JAVA hatten alle Teammitglieder bereits Erfahrung sammeln können. Als IDE wurde IntelliJIDEA vorgeschlagen und eingesetzt. Gegenwärtig ist keine IDE mit einer besseren grails Unterstützung bekannt (Code Vervollständigung, ...).

Innerhalb kurzer Zeit wurde die Projektgrundstruktur auf github bereitgestellt. Weiterhin entstanden der GUI Prototyp sowie das grundlegende Datenmodell. Daraufhin wurden das Spring-Security-Plugin zur Nutzerauthentifizierung eingebunden, die Datenbankanbindung konfiguriert (zu Entwicklungszwecken wurde auf eine filebasierte H2 Datenbank zurückgegriffen), erste Domainklassen erstellt und darauf aufbauend mit Boardmitteln die grundlegende CRUD (Create, Update, Delete) Funktionalität realisiert und getestet. Im nächsten Schritt wurde zunächst ein generischer Service entwickelt, der die Möglichkeit bietet Webservice mittels REST Interface zu konsumieren. Eine konkrete Implementierung folgte in der Anbindung der last.fm API mit der Zielstellung Musik Assets (Tracks, Alben, Künstlern) um zusätzliche Informationen anzureichern. Dabei sind von besonderem Interesse: Artwork (zur Verbesserung der Useability / ‚schönere‘ Darstellung), Tags (für die geplante Vorschlagsfunktion), ähnliche Künstler (für die geplante Vorschlagsfunktion). Parallel dazu erfolgte die Implementierung der Importfunktionalität. Dabei wurde darauf geachtet zusammenhängende Funktionalitäten in einzelnen Services bereit zu stellen. Es entstand ein Service, der dateisystemnahe Funktionalitäten bereitstellt (alle Mediendateien eines entsprechenden Typs auflisten, .info Datei zu einem Film finden, Cover zu einem Film finden), ein Service, der die vorhandenen Metadaten extrahiert und dem entsprechenden Konstruktor passend aufbereitet zur Verfügung stellt, sowie ein Service, der die entsprechenden Datensätze in der Datenbank erzeugt.

Mit diesem Stand konnten zu jeder Zeit mit geringem Aufwand größere Mengen an Testdaten generiert werden. Diese sind nötig um darauf aufbauend mit der Anpassung der GUI, Verbesserung der Useability, Erweiterung des Funktionsumfangs sowie Performanceoptimierung beginnen zu können.

Als erste Maßnahmen wurden CSS Anpassungen vorgenommen um die Listenansicht der Startseiten Items in eine 5-spaltige Kachelansicht zu überführen (GUI), Sortier- sowie Filterfunktionen auf den Listenseiten implementiert

(Usability), Ansichten, die Musikstücke nach Künstler bzw. Künstler\Album gruppiert darstellen, erstellt (Funktionsumfang) sowie Einstellungen an der Webservice Implementierung vorgenommen um durch einen reduzierten ConnectionTimeout, die Anwendung flüssiger reagieren zu lassen.

6.2 Kritik

Rückblickend muss festgestellt werden, dass die Importfunktionalität und damit die Bereitstellung relevanter Testdaten(mengen) zu spät bereit stand. Als Ursache dafür muss davon ausgegangen werden, dass die Teammitglieder nicht im Detail mit der Funktionsweise des eingesetzten Frameworks (grails, hier im Speziellen GORM/Hibernate) vertraut waren. Weiterhin bremsten technische Schwierigkeiten im Umgang mit der DIE (kein auto-reload bei Änderung des Quelltextes; manueller Neustart der Anwendung nahm bis zu 2 Minuten in Anspruch) die Entwicklung stellenweise erheblich aus. Wie sich erst spät im Projektverlauf herausstellte, hat das anfänglich umgesetzte Datenmodel erhebliche Auswirkungen auf die gesamte Anwendung. Der Aufwand ein Refactoring am Datenmodel zu betreiben, wurde erheblich unterschätzt. Dies hat für den bis zum Projektende erreichten Stand keine Auswirkungen, jedoch für zukünftig angedachte Funktionalitäten.

6.3 Ausblick

Um weitere Evolutionen zu ermöglichen, ist eine Neuausrichtung des Datenmodels unvermeidbar. Beispielsweise bedarf es zwingend der Domainklassen Künstler und Album. Weiterhin wurden im Rahmen des Projektes zwar Filme betrachtet, jedoch keine Serien beziehungsweise Staffeln oder Episoden. Dies ist ebenfalls in einer weiteren Ausbaustufe zu realisieren. Die Nutzung der Metadaten von last.fm zeigen bereits einen richtigen Weg in Bezug auf die Einbindung webbasierter Dienste auf. Für Filme und Serien werden ähnliche Daten beispielsweise durch imdb.com bzw. themoviedb.org bereitgestellt. Diese Dienste müssen künftig zwingend mit eingebunden werden. Vorstellbar wäre ebenso die API von Amazon mit einzubinden. Zur Realisierung der Abspielfunktionalität muss ein Plugin konzipiert werden, welches lokale Dateien per http zur Verfügung stellen kann. Um die Anwendungsperformance zu verbessern, wird es künftig unvermeidlich sein die Abfrageergebnisse von Webservices in einem lokalen Cache zwischen zu speichern. Diesem Ziel kann weiterhin durch die konsequente Pagination der Suchergebnisse zugearbeitet werden. Werden im Musikbereich zur Zeit nur MP3 Dateien mit ID3v2 Tags unterstützt, so ist zukünftig die Unterstützung weiterer Audioformate voran zu treiben. Dabei sollten nicht mehr nur Texttags ausgelesen und gespeichert werden, sondern ebenfalls in Tags gespeicherte Cover Artworks.

7 Anhang

7.1 Protokolle

7.1.1 23.04.2016

Beuth Hochschule für Technik Berlin

Gesprächsprotokoll Agenda für KW 17/18

Betreuer: Herr Prof. Dr. Edlich

Abgabetermin: 23.04.2016

Gesprächsprotokoll

Datum: 23.04.2016

Ort: Präsenz

Protokollant: David Heenemann

Teilnehmer

Teilnehmer	Anwesenheit
Arndt	X
David	X
David	X
Sebastian	x

Ergebnisse

Thema	Inhalt	Hauptverantwortlicher
Dokumentation	Erstellung eines Grundgerüstes	Sebastian
SourceCode Struktur	Erstellung aller Plugins	David Heenemann
Gui-Prototype	Suche eines geeigneten GUI-Prototypen Tool	David Meinke
Datenmodell	Erstellung eines Datenbankmodells	Arndt

7.1.2 09.05.2016

Beuth Hochschule für Technik Berlin

Gesprächsprotokoll Agenda für KW 19/20

Betreuer: Herr Prof. Dr. Edlich

Abgabetermin: 09.05.2016

Gesprächsprotokoll

Datum: 09.05.2016

Ort: Slack

Protokollant: David Heenemann

Teilnehmer

Teilnehmer	Anwesenheit
Arndt	X
David	X
David	X
Sebastian	x

Ergebnisse

Thema	Inhalt	Hauptverantwortlicher
Interne Plugins	Aufbau des Core- und Filmbibliothekspugins	Alle
Externe Plugins	Integration des Spring-Security Plugins, zur Identifikation der Benutzer	David Heenemann
Klassen	Erstellung aller Klassen, die für die Benutzer und Filmbibliothek notwendig sind	Alle
CRUD-Funktionen	Erstellung von Create/ Update/ Delete - Funktionen, um schnell Daten einpflegen zu können	Alle

7.1.3 22.05.2016

Beuth Hochschule für Technik Berlin

42-Media

-

Feedback

Agenda für KW 21/22

Betreuer: Herr Prof. Dr. Edlich

Abgabetermin: 23.05.2016

Gesprächsprotokoll

Datum: 23.05.2016
Ort: Slack
Protokollant: David Heenemann

Teilnehmer

Teilnehmer	Anwesenheit
Arndt	X
David	X
David	X
Sebastian	X

Was ist passiert? (Sprint#1 09.05-22.05.2016)

- Zusammenfassung der Plugins „ftLibraryMusic“ und „ftLibraryMovie“ zu „ftLibrary“
- Implementierung des „Spring-Security“ Plugins
- Anpassung der Bootstrap-Datei um Standarduser zu erzeugen
- Erstellung eines Masterlayouts
- Erstellung der Core-Klassen
- Erstellung einer „H2“-Testdatenbank
- Erstellung von Default-Scaffolding für Standard-CRUD Funktionen
- Implementierung von Modal-Pages

→ Genaue Übersicht kann dem Bild (siehe Anlage) entnommen werden!

Was ist geplant? (Sprint#2 23.05-05.06.2016)

Thema	Inhalt	Hauptverantwortlicher
Interne Plugins	Erweiterung des Bibliothekspugins Erstellung des ExternalPlugins zur Importierung von Medien über eine Restschnittstelle	Alle

Projektinformationen

- Projektmanagementtool: <https://app.scrumdesk.com/#/projects/14369/work/>
- GitHub: github.com/42Media/

7.1.4 07.06.2016

Beuth Hochschule für Technik Berlin

42-Media

-

Feedback

Agenda für KW 23/24

Betreuer: Herr Prof. Dr. Edlich

Abgabetermin: 07.06.2016

Gesprächsprotokoll

Datum: 07.06.2016
Ort: Slack
Protokollant: David Heenemann

Teilnehmer

Teilnehmer	Anwesenheit
Arndt	X
David	X
David	X
Sebastian	X

Was ist passiert? (Sprint#2 23.05-05.06.2016)

- CSS-Anpassung der Detailseiten
 - Anpassung der bestehenden Seiten an das Layout
- Das Basis-Layout wurde angepasst und einheitlicher und übersichtlicher gestaltet
- Detail-Seiten wurden erstellt, um die Informationen von erstellten Filmen und Musikstücken nachträglich ansehen zu können
- Import von Dateien (Filme, Musik)
 - Film- und Musik-Dateien werden aus einem angegebenen Ordner importiert, die Meta-Daten werden dabei in die Klassen geschrieben und in die Datenbank gespeichert
- Implementierung eines generischen REST-Services
 - Service, der durch die Angabe von einer URL und den gewünschten Parametern, ein JSON-Objekt zurück liefert
- Service zur Nutzung der „LastFM“ REST-API
 - Spezieller Service, der die Meta-Daten von der Website von „LAST-FM“ als JSON zurückgibt

- Anlegen von weiteren Klassen
 - Image-Klasse wurde angelegt, um das DRY-Prinzip einzuhalten
 - Image-Klasse wird von anderen Klassen als Attribut verwendet
- Editieren von bereits existierenden Daten
 - Daten können nach der Erstellung manipuliert werden. Dazu wurden gesonderte Controller geschrieben
- Anpassung von „package“ Namen
 - Alle „package“ Namen wurden klein geschrieben, um den SourceCode zu vereinheitlichen

Was ist geplant? (Sprint#306.06-19.06.2016)

Thema	Inhalt	Hauptverantwortlicher
Interne Plugins	<ul style="list-style-type: none"> - Erweiterung des Bibliothekspugins <ul style="list-style-type: none"> ○ Implementierung eines Berechtigungskonzeptes <ul style="list-style-type: none"> ▪ Es sollen nur die Inhalte Angezeigt werden, die einem Benutzer gehören ○ Erstellung einer Historie <ul style="list-style-type: none"> ▪ Die letzten Aktionen sollen dem User auf dem Dashboard angezeigt werden ▪ z.B. Zuletzt angesehen, zuletzt importiert, zuletzt gelöscht, etc. - Erweiterung der REST-API <ul style="list-style-type: none"> ○ Weitere Externe Datenbanken sollen angefragt 	Alle

	werden, um die Meta-Daten erhalten zu können	
		Alle

Projektinformationen

- Projektmanagementtool:
<https://app.scrumdesk.com/#/projects/14369/work/>
- GitHub: github.com/42Media/

Aktueller Stand

Backlog	Todo (0 - 0)		In progress (0 - 0)		Done	
3 items, 0 / 0 / 1 hours	4 tasks, 0 / 0 / 0 hours		5 tasks, 0 / 0 / 0 hours		19 tasks, 0 / 0 / 1 hours	
● Standardanwendung 110767	#181400 Erstellung einer Upload-Funktion ① 0/0/0 h	#189162 Metainformationen aus Datei lesen ① 0/0/0 h	#178717 Erstellung eines Berechnungskonzepts ① 0/0/0 h	#176671 Implementierung der Plugins ① 0/0/1 h	#189027 Aufräumen des Layouts ① 0/0/0 h	#178681 Erstellung eines Masterlayouts ① 0/0/0 h
				#180104 Umsetzen des Masterlayouts in die main.jsp ① 0/0/0 h	#179583 Implementierung des Spring-Security-Plugins ① 0/0/0 h	#186787 Erstellung der Core-Klassen ① 0/0/0 h
● Integration der Bibliotheksfunktion 110768			#186831 Erstellung von Modalpages für Detailansicht ① 0/0/0 h	#194268 Bibliothekseinträge aus bestehenden info Dateien erzeugen ① 0/0/0 h	#181403 Konfiguration der H2-File-Datenbank ① 0/0/0 h	#181136 Implementierung der REST-Schnittstelle ① 0/0/0 h
			#186830 Anpassung der Modalpages um Domains zu editieren ① 0/0/0 h	#189349 Domain Klasse für Bilder ① 0/0/0 h	#189039 Eigene Domain-Klassen für "Genre" und "Country" ① 0/0/0 h	
● Integration der Katalogisierung von externen Medien 110773	#178704 imdb: Filme/ Serien ① 0/0/0 h	#178705 Amazon: Filme/ Serien ① 0/0/0 h	#197005 Abfrage IMDB Rest ① 0/0/0 h	#178682 Erstellung der CRUD-Funktionen ① 0/0/0 h	#178675 Erstellung aller Klassen ① 0/0/0 h	#186829 Erstellung von Modalpages ① 0/0/0 h
				#179585 Erstellung von Detaileiten/-Anzeigen von Meta-Informationen ① 0/0/0 h	#179584 Erstellung der Bootstrap-Datei, zur Erstellung von Testdaten ① 0/0/0 h	#18405: Anpassung der Bootstrap-Datei, zur Erstellung von Testdaten ① 0/0/0 h

7.1.5 20.06.2016

Beuth Hochschule für Technik Berlin

42-Media

-

Feedback

Agenda für KW 25/26

Betreuer: Herr Prof. Dr. Edlich

Abgabetermin: 20.06.2016

Gesprächsprotokoll

Datum: 20.06.2016
Ort: Slack
Protokollant: David Heenemann

Teilnehmer

Teilnehmer	Anwesenheit
Arndt	X
David	X
David	X
Sebastian	X

Was ist passiert? (Sprint#2 23.05-05.06.2016)

- CSS-Anpassung der Modalpages
 - Anpassung der bestehenden Modalpages, dass mehr Informationen auf kleineren Raum passen, und dadurch die Übersichtlichkeit erhöht wurde
- Es wurde ein Bug behoben, der es verhinderte, dass der automatische Import und der manuelle Import zur selben Zeit, ohne Anpassungen am SourceCode, funktionierte
- Die Seiten, die die Musikstücke und Filme als Listen anzeigen, beziehen ihre Informationen nun aus der Datenbank
 - Über die Listen können die Einträge direkt bearbeitet werden
- Die Detailansicht der einzelnen Einträge wurde als Modalpage eingerichtet
- Die Detailansicht wurde in zwei Bereiche geteilt. Der obere Teil zeigt die Informationen eines Musikstücks oder Filmes an, die durch den manuellen oder automatischen Import des Benutzers angelegt wurde. Der untere Bereich zeigt die Informationen an, die von externen Bibliotheken via REST-Schnittstelle geladen werden konnten

- Es wurde ein Bug behoben, der verhinderte, dass alter Inhalt aus den Modalpages gelöscht wurde. Dadurch wurde immer derselbe Inhalt in den Modalpages angezeigt. Nun wird immer der richtige angeforderte Inhalt angezeigt

- Die Detail-Informationen von Musik und Video Einträgen, werden nun in einer ordentlichen Art und Weise in den Modalpages angezeigt

Was ist geplant? (Sprint#306.06-19.06.2016)

Thema	Inhalt	Hauptverantwortlicher
Interne Plugins	<ul style="list-style-type: none"> - Alle Plugins werden stabilisiert <ul style="list-style-type: none"> o Vorhandene Funktionalität wird fertig gestellt, und bekannte oder auftretende Bugs werden behoben o Alle offenen Tasks werden ebenfalls noch abgearbeitet - Die Dokumentation soll finalisiert werden 	Alle

Projektinformationen

- Projektmanagementtool: <https://app.scrumdesk.com/#/projects/14369/work/>
- GitHub: github.com/42Media/

Aktueller Stand

Backlog	Todo (0 - 0)			In progress (0 - 0)			Done			
3 items, 0 / 0 / 1 hours	5 tasks, 0 / 0 / 0 hours			3 tasks, 0 / 0 / 0 hours			30 tasks, 0 / 0 / 1 hours			
<p>Standardanwendung #110767</p> <p>0 0/0/1 h</p>	<p>#181400 Erstellung einer Upload-Funktion</p> <p>0 0/0/0 h</p>	<p>#189162 Metainformationen aus Datei lesen</p> <p>0 0/0/0 h</p>	<p>#199714 Modal-Dialog für Quelle der Dateien</p> <p>0 0/0/0 h</p>	<p>#178717 Erstellung eines Berechnungskonzepts</p> <p>0 0/0/0 h</p>	<p>#178671 Implementierung der Plugins</p> <p>0 0/0/0 h</p>	<p>#189027 Aufräumen des Layouts</p> <p>0 0/0/0 h</p>	<p>#178681 Erstellung eines Masterlayouts</p> <p>0 0/0/0 h</p>	<p>#180104 Umsetzen des Masterlayouts in die main.gsp</p> <p>0 0/0/0 h</p>		
<p>Integration der Bibliotheksfunktion #110769</p> <p>0 0/0/0 h</p>				<p>#186830 Anpassung der Modalpages um Domains zu editieren</p> <p>0 0/0/0 h</p>	<p>#186842 Erstellung der CRUD-Funktionen</p> <p>0 0/0/0 h</p>	<p>#178675 Erstellung aller Klassen</p> <p>0 0/0/0 h</p>	<p>#186829 Erstellung von Modalpages</p> <p>0 0/0/0 h</p>			
<p>Integration der Katalogisierung von externen Medien #110771</p> <p>0 0/0/0 h</p>	<p>#178704 imdb: Filme/ Serien</p> <p>0 0/0/0 h</p>	<p>#178705 Amazon: Filme/ Serien</p> <p>0 0/0/0 h</p>		<p>#178705 Abfrage IMDB Rest</p> <p>0 0/0/0 h</p>	<p>#178684 Erstellung von eigenen Bearbeitungsfunktionen</p> <p>0 0/0/0 h</p>	<p>#181406 Anpassung der Bootstrap-Datei, zur Erstellung von Postdaten</p> <p>0 0/0/0 h</p>	<p>#194268 Bibliothekseingabe bestehenden nfo-Daten erzeugen</p> <p>0 0/0/0 h</p>			
					<p>#178681 Erstellung von Detailseiten/ Anzeigen von Meta-Informationen</p> <p>0 0/0/0 h</p>	<p>#201705 Domain Model Refactoring evaluieren</p> <p>0 0/0/0 h</p>	<p>#201706 Cover Art einlesen und abspeichern</p> <p>0 0/0/0 h</p>	<p>#199720 Show-GSP für video und music anpassen</p> <p>0 0/0/0 h</p>		
					<p>#178681 Erstellung von Detailseiten/ Anzeigen von Meta-Informationen</p> <p>0 0/0/0 h</p>	<p>#203857 Listen beziehen Daten aus Datenbank</p> <p>0 0/0/0 h</p>	<p>#203858 Detailansicht in Modalpages</p> <p>0 0/0/0 h</p>	<p>#203855 CSS Anpassungen der Modalpages</p> <p>0 0/0/0 h</p>	<p>#203856 BugFix: Import</p> <p>0 0/0/0 h</p>	
					<p>#178710 last.FM: Musik</p> <p>0 0/0/0 h</p>	<p>#189161 Details/ Metainformationen von externen Diensten abrufen</p> <p>0 0/0/0 h</p>	<p>#203854 Rest-Informationen in Modalpages anzeigen</p> <p>0 0/0/0 h</p>			

7.1.6 04.07.2016

Beuth Hochschule für Technik Berlin

42-Media

-

Feedback

Abschlussbericht

Betreuer: Herr Prof. Dr. Edlich

Abgabetermin: 04.07.2016

Gesprächsprotokoll

Datum: 04.07.2016
Ort: Slack
Protokollant: David Heenemann

Teilnehmer

Teilnehmer	Anwesenheit
Arndt	X
David	X
David	X
Sebastian	X

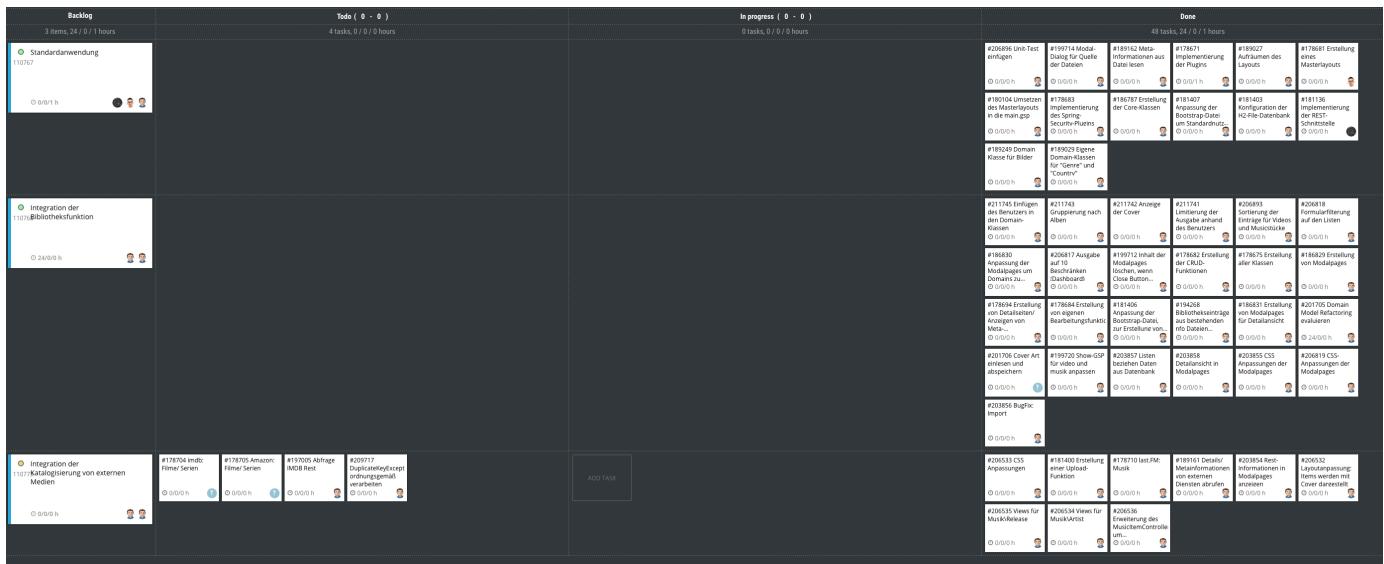
Was ist passiert? (Sprint#2 20.06-03.07.2016)

- CSS-Anpassung der Modalpages
 - Anpassung der Modalpages, um eine ordentliche Darstellung der Create-Modalpages zu ermöglichen
- Limitierung der Ausgabe auf der Startseite auf max. 10 Elemente
- Erstellung eines Modalpages, um Einträge bearbeiten zu können
 - Elemente können direkt aus der Modalansicht editiert werden
- Upload von Music und Videos mittel Pfadauswahl und Modalpages
- Erweitern der Klassen um „inserted“ Tag
- Erweitern der Klassen um „User“ Tag
- Auf den Listen-Seiten:
 - Filterung nach letzten 5 Einträge
 - Aufsteigende oder absteigende Filterung nach Namen oder Hinzufügedatum
 - Filterung der Music nach Titel oder Künstler oder Jahr
 - Filterung der Videos nach Titel oder Regisseur oder Jahr
- Es werden nun nur die Inhalte angezeigt, die der Benutzer erstellt hat, fremde Inhalte sieht er nicht
- Gruppierung der Musiktitel nach Alben in der Musikliste
- Es werden nun Cover-Bilder der Filme und Musikelemente angezeigt

Projektinformationen

- Projektmanagementtool: <https://app.scrumdesk.com/#/projects/14369/work/>
 - GitHub: github.com/42Media/

Letzter Stand



7.2 Scrum Übersicht

7.2.1 23.05.2016

Showing 2 item(s) out of 2 total.

Backlog	Todo (0 - 0)		In progress (0 - 0)		Done (0 - 0)			
2 items, 0 / 0 / 1 hours	5 tasks, 0 / 0 / 0 hours		3 tasks, 0 / 0 / 0 hours		10 tasks, 0 / 0 / 1 hours			
Standardanwendung 110767 Ø 0/0/1 h	#178717 Erstellung eines Berechtigungskonzepts Ø 0/0/0 h	#181400 Erstellung einer Upload-Funktion Ø 0/0/0 h	#181136 Implementierung der REST-Schnittstelle Ø 0/0/0 h	#178671 Implementierung der Plugins Ø 0/0/1 h	#178681 Erstellung eines Masterlayouts Ø 0/0/0 h	#180104 Umsetzen des Masterlayouts in die main.gsp Ø 0/0/0 h		
				#178683 Implementierung des Spring-Security-Plugins Ø 0/0/0 h	#186787 Erstellung der Core-Klassen Ø 0/0/0 h	#181403 Konfiguration der H2-File-Datenbank Ø 0/0/0 h		
Integration der Bibliotheksfunktion 110768 Ø 0/0/0 h	#178694 Erstellung von Detailseiten/Anzeigen von Meta-Informationen Ø 0/0/0 h	#181406 Anpassung der Bootstrap-Datei, zur Erstellung von Testdaten Ø 0/0/0 h	#186831 Erstellung von Modalpages für Detailansicht Ø 0/0/0 h	#186830 Anpassung der Modalpages um Domains zu editieren Ø 0/0/0 h	#178684 Erstellung von eigenen Bearbeitungsfunktion Ø 0/0/0 h	#178682 Erstellung der CRUD-Funktionen Ø 0/0/0 h	#178675 Erstellung aller Klassen Ø 0/0/0 h	#186829 Erstellung von Modalpages Ø 0/0/0 h

STORY MAP

- BACKLOG
- PLAN
- WORK
- RETRO
- REPORTS
- SETUP

PROJECTS

v4.16.1

S M L

7.2.2 06.06.2016

Backlog	Todo (0 - 0)		In progress (0 - 0)		Done			
3 items, 0 / 0 / 1 hours	4 tasks, 0 / 0 / 0 hours		5 tasks, 0 / 0 / 0 hours		19 tasks, 0 / 0 / 1 hours			
Standardanwendung 110767	#181400 Erstellung einer Upload-Funktion	#189162 Metainformationen aus Datei lesen	#178717 Erstellung eines Berechtigungskonzepts	#178671 Implementierung der Plugins	#189027 Aufräumen des Layouts	#178681 Erstellung eines Masterlayouts		
	Ø 0/0/0 h	Ø 0/0/0 h	Ø 0/0/0 h	Ø 0/0/1 h	Ø 0/0/0 h	Ø 0/0/0 h	Ø 0/0/0 h	
Integration der Bibliotheksfunktion 110768			#186831 Erstellung von Modalpages für Detailansicht	#194268 Bibliothekseinträge aus bestehenden nfo Dateien erzeugen	#186830 Anpassung der Modalpages um Domains zu editieren	#178682 Erstellung der CRUD-Funktionen	#178675 Erstellung aller Klassen	#186829 Erstellung von Modalpages
	Ø 0/0/0 h		Ø 0/0/0 h	Ø 0/0/0 h	Ø 0/0/0 h	Ø 0/0/0 h	Ø 0/0/0 h	Ø 0/0/0 h
Integration der Katalogisierung von externen Medien 110773	#178704 imdb: Filme/Serien	#178705 Amazon: Filme/ Serien	#197005 Abfrage IMDB Rest	#178710 last.FM: Musik	#189161 Details/ Metainformationen von externen Diensten abrufen			
	Ø 0/0/0 h	Ø 0/0/0 h	Ø 0/0/0 h	Ø 0/0/0 h	Ø 0/0/0 h			

7.2.3 20.06.2016

Backlog	Todo (0 - 0)			In progress (0 - 0)			Done			
3 items, 0 / 0 / 1 hours	5 tasks, 0 / 0 / 0 hours			3 tasks, 0 / 0 / 0 hours			30 tasks, 0 / 0 / 1 hours			
● Standardanwendung 110767	#181400 Erstellung einer Upload-Funktion ⌚ 0/0/0 h ?	#189162 Meta-informationen aus Datei lesen ⌚ 0/0/0 h ?	#199714 Modal-Dialog für Quelle der Dateien ⌚ 0/0/0 h ?	#178717 Erstellung eines Berechtigungskonzepts ⌚ 0/0/0 h ?	#178671 Implementierung der Plugins ⌚ 0/0/1 h ?	#189027 Aufräumen des Layouts ⌚ 0/0/0 h ?	#178681 Erstellung eines Masterlayouts ⌚ 0/0/0 h ?	#180104 Umsetzen des Masterlayouts in die main.gsp ⌚ 0/0/0 h ?		
● Integration der Bibliotheksfunktion 110768				#186830 Anpassung der Modalpages um Domains zu editieren ⌚ 0/0/0 h ?	#199712 Inhalt der Modalpages löschen, wenn Close Button betätigt wurde ⌚ 0/0/0 h ?	#178682 Erstellung der CRUD-Funktionen ⌚ 0/0/0 h ?	#178675 Erstellung aller Klassen ⌚ 0/0/0 h ?	#18629 Erstellung von Modalpages ⌚ 0/0/0 h ?		
● Integration der Katalogisierung von externen Medien 110773	#178704 imdb: Filme/ Serien ⌚ 0/0/0 h ?	#178705 Amazon: Filme/ Serien ⌚ 0/0/0 h ?		#197005 Abfrage IMDB Rest ⌚ 0/0/0 h ?	#178694 Erstellung von Detailseiten/ Anzeigen von Meta-Informationen ⌚ 0/0/0 h ?	#178684 Erstellung der Bootstrap-Datei, zur Erstellung von Testdaten ⌚ 0/0/0 h ?	#181406 Anpassung der Bootstrap-Datei, zur Erstellung von Testdaten ⌚ 0/0/0 h ?	#194268 Bibliothekseinträge aus bestehenden nfo Dateien erzeugen ⌚ 0/0/0 h ?		
					#186831 Erstellung von Modalpages für Detailansicht ⌚ 0/0/0 h ?	#201705 Domain Model Refactoring evaluieren ⌚ 0/0/0 h ?	#201706 Cover Art einlesen und abspeichern ⌚ 0/0/0 h ?	#199720 Show-GSP für video und musik anpassen ⌚ 0/0/0 h ?		
					#203857 Listen beziehen Daten aus Datenbank ⌚ 0/0/0 h ?	#203858 Detailansicht in Modalpages ⌚ 0/0/0 h ?	#203855 CSS Anpassungen der Modalpages ⌚ 0/0/0 h ?	#203856 BugFix: Import ⌚ 0/0/0 h ?		
					#178710 last.FM: Musik ⌚ 0/0/0 h ?	#189161 Details/ Metainformationen von externen Diensten abrufen ⌚ 0/0/0 h ?	#203854 Rest-Informationen in Modalpages anzeigen ⌚ 0/0/0 h ?			

7.2.4 04.07.2016

Backlog	Todo (0 · 0)	In progress (0 · 0)	Done
3 items, 24 / 0 / 1 hours	4 tasks, 0 / 0 / 0 hours	0 tasks, 0 / 0 / 0 hours	48 tasks, 24 / 0 / 1 hours
<ul style="list-style-type: none"> ● Standardanwendung #110767 <ul style="list-style-type: none"> ○ 0/0/1 h 			<ul style="list-style-type: none"> #206896 Unit-Test einfügen ○ 0/0/0 h #199714 Modal-Dialog für Quelle der Dateien ○ 0/0/0 h #189162 Metainformationen aus Dateien lesen ○ 0/0/0 h #178671 Implementierung der Plugins ○ 0/0/0 h #189027 Aufräumen des Layouts ○ 0/0/0 h #178681 Erstellung eines Masterlayouts ○ 0/0/0 h
<ul style="list-style-type: none"> ● Integration der Bibliotheksfunktion #110768 <ul style="list-style-type: none"> ○ 24/0/0 h 			<ul style="list-style-type: none"> #180104 Umsetzen des Masterlayouts in die main.gsp <ul style="list-style-type: none"> ○ 0/0/0 h #178683 Implementierung der "Logout"-Security-Page ○ 0/0/0 h #186787 Erstellung der Core-Klassen <ul style="list-style-type: none"> ○ 0/0/0 h #181407 Anpassung der main.gsp-Datei um Standardnutz... ○ 0/0/0 h #181403 Konfiguration der H2-File-Datenbank ○ 0/0/0 h #181136 Implementierung der REST-Schicht ○ 0/0/0 h
<ul style="list-style-type: none"> ● Integration der Katalogisierung von externen Medien #110770 <ul style="list-style-type: none"> ○ 0/0/0 h 	<ul style="list-style-type: none"> #178704 imdb: Filme/ Serien #178705 Amazon: Filme Serien #197005 Abfrage IMDB Rest <ul style="list-style-type: none"> ○ 0/0/0 h ○ 0/0/0 h ○ 0/0/0 h #209717 DuplicateKeyException ordnungsgemäß verarbeiten ○ 0/0/0 h 	<ul style="list-style-type: none"> #211745 Einfügen des Benutzers in den Domain-Klassen ○ 0/0/0 h #211743 Gruppierung nach Alben ○ 0/0/0 h #211742 Anzeige der Cover ○ 0/0/0 h #211741 Limitierung der Ausgabe anhand des Benutzers ○ 0/0/0 h #206893 Sortierung der Einträge für Videos und Musikstücke ○ 0/0/0 h #206818 Formularfilterung auf den Listen ○ 0/0/0 h 	<ul style="list-style-type: none"> #186830 Anpassung der Modalpages um Domains zu... ○ 0/0/0 h #206817 Ausgabe auf 10 Beschränkungen ○ 0/0/0 h #199712 Inhalt der Modalpages löschen, wenn Close Button... ○ 0/0/0 h #178682 Erstellung der CRUD-Funktionen ○ 0/0/0 h #178675 Erstellung aller Klassen ○ 0/0/0 h #186829 Erstellung von Modalpages ○ 0/0/0 h
	<ul style="list-style-type: none"> #178694 Erstellung von Detailsiten/ Anzeigen von Meta... ○ 0/0/0 h #178684 Erstellung von eigenen Bearbeitungsfunkti... ○ 0/0/0 h #181406 Bibliothekseinträge aus bestehenden Info Dateien... ○ 0/0/0 h #178681 Erstellung von Detailsichten für Modalpages ○ 0/0/0 h #201706 Cover Art einlesen und abspeichern ○ 0/0/0 h #199720 Show-GSP für video und musik anpassen ○ 0/0/0 h #203857 Listen beziehen Daten aus Datenbank ○ 0/0/0 h #203858 Detailsicht für Modalpage ○ 0/0/0 h #203855 CSS-Anpassungen der Modalpages ○ 0/0/0 h #206819 CSS-Anpassungen der Modalpages ○ 0/0/0 h 	<ul style="list-style-type: none"> #206533 CSS Anpassungen ○ 0/0/0 h #181409 Erstellung einer Upload-Funktion ○ 0/0/0 h #178710 last.FM: Musik ○ 0/0/0 h #199161 Details/ Metainformationen von externen Diensten abrufen ○ 0/0/0 h #203854 Rest-Informationen in Modalpages anzeigen ○ 0/0/0 h #206532 Layoutanpassung: Items werden mit Cover dargestellt ○ 0/0/0 h 	<ul style="list-style-type: none"> #206535 Views für MusikRelease ○ 0/0/0 h #206534 Views für MusikArtist ○ 0/0/0 h #206536 Erweiterung des MusicItemController um... ○ 0/0/0 h
		<ul style="list-style-type: none"> #206533 CSS Anpassungen ○ 0/0/0 h #181409 Erstellung einer Upload-Funktion ○ 0/0/0 h #178710 last.FM: Musik ○ 0/0/0 h #199161 Details/ Metainformationen von externen Diensten abrufen ○ 0/0/0 h #203854 Rest-Informationen in Modalpages anzeigen ○ 0/0/0 h #206532 Layoutanpassung: Items werden mit Cover dargestellt ○ 0/0/0 h 	

ADD TASK

