Entwufsdokument 1 – Softwarepraktikum WS13/14

Gruppe 3B

Das folgende Entwufsdokument beschreibt eine unsere Social-Media Plattform namens dezibel™ .

Pascal Brackmann, Bastian Braun, Benjamin Knauer, Tristan Nicolas, Henner Niehaus, Richard Schulze, Alexander Trahe, Aristide Voufouo Mbogning, Tobias Wiedau

Inhaltsverzeichnis

1. Einleitung

Das folgende Entwurfsdokument beschreibt eine Software für eine Social-Media-Plattform. Hauptaugenmerk wird dabei auf die Verwaltung von Musik-Medien und der Kommunikation zwischen Benutzern, Künstlern und Label-Managern gelegt.
Dabei soll den Künstlern die Möglichkeit gegeben werden, mit ihren Fans in Kontakt zu treten, neue Musik über die Plattform zu teilen, oder verschiedene Labels zu kontaktieren. Für Label-Manager soll die Plattform als Verwaltung des Labels dienen. So können diese Musik im Namen ihrer Künstler veröffentlichen und News für diese schreiben. Fans soll es die Möglichkeit bieten neue Musik kennenzulernen und durch ihre Lieblingskünstler ähnliche Künstler zu finden. So können unter anderem veröffentlichte Musik angehört werden, oder Neuigkeiten kommentiert werden.
Durch diese diversen Funktionen erhalten Künstler und Label-Manager ein Feedback zu ihren Produkten und Benutzer erhalten kontinuierlich neue Informationen zu ihren favorisierten Künstlern.

2. Systementwurf

2.1 Entwurfsziele

2.1.1 Verlässlichkeitskriterien

Das System soll eine hohe Robustheit aufweisen, damit auch unerfahrene Benutzer das Programm nutzen können. Ungültige Benutzereingaben werden durch den Aufbau der GUI abgefangen. Sollte der Benutzer dennoch einen Fehler machen, wird er mittels einer Fehlermeldung informiert.
Des Weiteren soll das Programm sehr zuverlässig und stets verfügbar sein, um Benutzern einen Anreiz zu geben, das Programm zu benutzen.
Zudem weist das Programm eine bedingt gute Fehlertoleranz auf, da die einzelnen Funktionalitäten und Klassen ordentlich gekapselt wurden. Fällt ein Teil der Software weg, beispielsweise indem die Assoziationen zwischen einem Users und seinen Playlists verloren gehen oder falsch gesetzt werden, so kann er den restlichen Teil der Software uneingeschränkt weiternutzen. Sollte es jedoch passieren, dass ganze Teile der Datenbank wie z.B. alle Medien wegfallen, so ist mit unvorhergesehenen Konsequenzen zu rechnen, ein entsprechend geregelter Programmablauf aber u.U. noch möglich.
Der Schutz des Systems vor feindlichen Angriffen ist zu vernachlässigen, da es sich um eine Einzelplatzanwendung handelt. Ebenso ist die Sicherheit kein vorrangiges Ziel, da die Ausführung des Programms keine Auswirkung auf die reale Welt hat.

2.1.2 Wartungskriterien

Erweiterbarkeit, Modifizierbarkeit und Anpassungsfähigkeit spielen keine große Rolle bei der Umsetzung, da die Funktionalität zum Abgabetermin vorhanden sein muss. Eine nachträgliche Änderung des Codes ist nicht vorgesehen.
Aufgrund der Anforderungen soll das Programm auch auf weiteren Plattformen (wie z.B. Linux) lauffähig sein, dies ist durch die Umsetzung in Java in Grundzügen bereits gegeben.
Die Lesbarkeit und Rückverfolgbarkeit haben eine hohe Priorität, damit sich Außenstehende leicht in den Code einarbeiten können.

2.1.3 Leistungskriterien

Die Antwortzeit des Systems soll eine verzögerungsfreie Nutzung ermöglichen. Im Regelfall kann die Antwortzeit gering gehalten werden, da die Funktionen des Systems eine geringe bis mittlere Komplexität aufweisen.
Je nach Anzahl der verwalteten Medien steigt der Bedarf an Speicherplatz. Das Programm selbst nimmt nur einen geringen Anteil des Speichers in Anspruch.

2.1.4 Kostenkriterien

Die Kosten haben aufgrund der fiktiven Aufgabenstellung keine Priorität.

2.1.5 Endbenutzerkriterien

Da auch unerfahrene Benutzer das System benutzen sollen, ist eine einfache Nutzbarkeit notwendig. Das System soll außerdem nützliche Funktionen bereitstellen, damit es für den Benutzer einen Vorteil bietet.

2.2 Systemzerlegung

Das Ziel ist eine geschlossene Schichtenarchitektur zu haben und die Teilsysteme möglichst mit geringer Kopplung und hoher Kohäsion zu realisieren.

de.dezibel-music.gui:

grafische Oberfläche

Klassen: AdsPanel, AlbumMediaTableMode, AlbumPanel, AlbumTableModel, ApplicationDialog, ApplicationToArtistTableModel, ApplicationToLabelTableModel, ArtistTableModel, CommentDialog, CommentTableModel, ContextMenu, CreateAlbumDialog, DezibelColor, DezibelFont, DezibelPanel, DragablePanel, FavoritesTableModel, FollowerTableModel, LabelProfilPanel, LabelProfilModel, LoginPanel, MediaTableModel, MediumPanel, MenuItem, MyListsPanel, MyListsTableModel, NewsDialog, NewsPanel, NewsPanelTableModel, NewsSidePanel, NewsSideTableModel, PlayerPanel, PlaylistMediaTableModel, PlaylistPanel, ProfilPanel, RecommendationsTableModel, RegistrationPanel, SearchPanel, TextAreaCellRenderer, UploadDialog, UserTableModel

de.dezibel-music.player:

Paket für den Mediaplayer

Klassen: Player, PlayerObserver

de.dezibel-music.control:

enthält die Kontrollklassen (Logikschicht)

Klassen: AdminControl, AdsControl, AlbumControl, AlbumNameComparator, ApplicationControl, CommentControl, HashGenerator, LabelControl, LabelNameComparator, LoginControl, MediumNameComparator, MediumRatingComparator, MediumUploadDateComparator, NewsControl, NewsDateComparator, PlaylistControl, PlaylistObserver, ProfileControl, RegistrationControl, SaveControl, Search, UploadControl, UserNameComparator

de.dezibel-music.data:

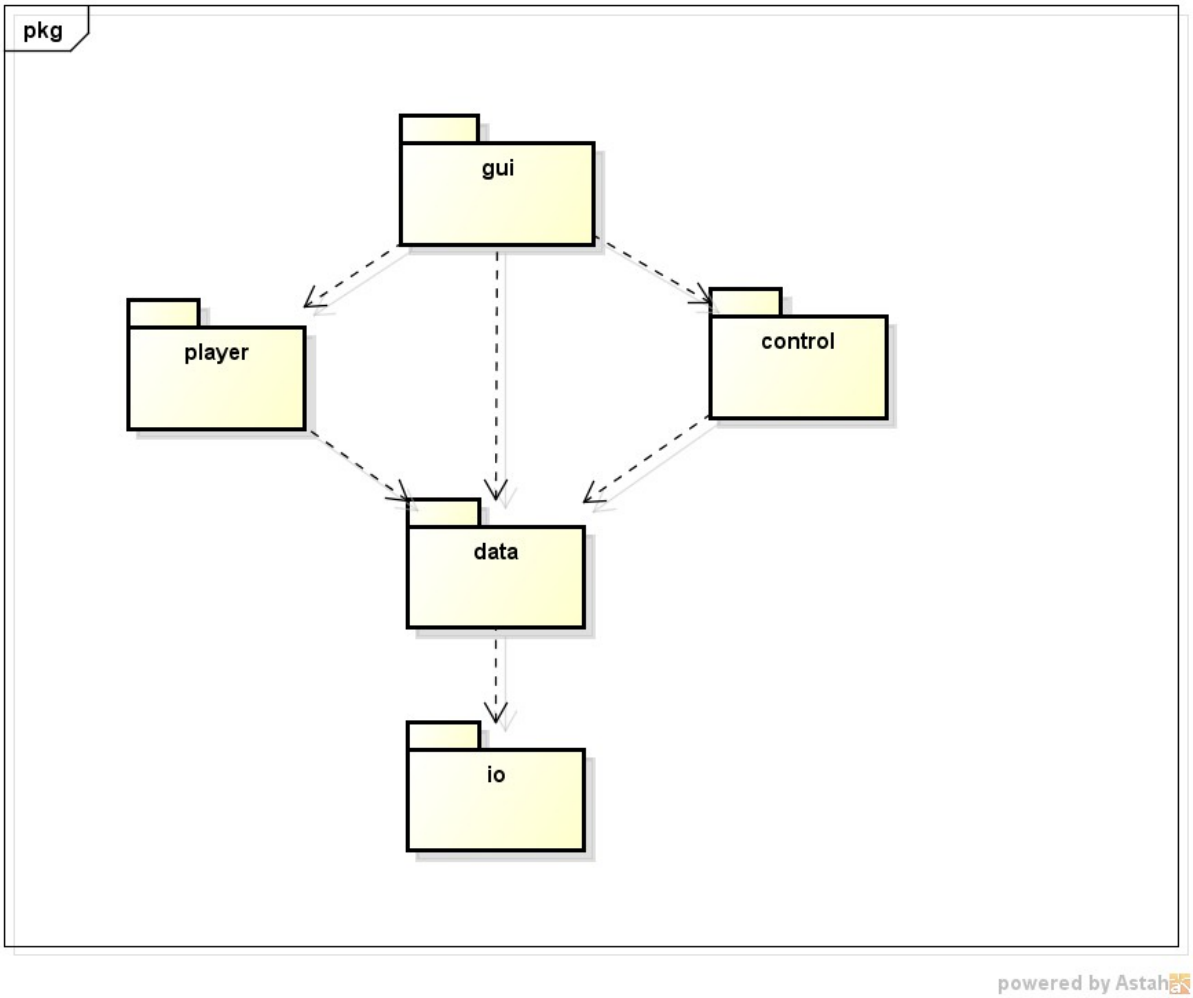
enthält die Entitätsklassen, also die Daten des Systems

Klassen: Album, Application, Comment, Commentable, Database, Genre, Label, Lockable, Medium, News, Playlist, Rating, User

de.dezibel-music.io:

enthält Klassen zur Serialisierung und Deserialisierung der Entitätsklassen

Klassen: FileMover, ImageLoader, MailUtil, MediumLoader, XStreamAdapter



2.3 Verwendung existierender Software-Komponenten

Xstream:
Zur persistenten Datenhaltung, siehe hierfür auch 2.4 Persistente Daten und Persistenzmechanismus.
JavaFX:
Wir verwenden den MediaPlayer von JavaFX zur Musikwiedergabe.
Swing und Sanaware Java Docking:
Die Gui wird aus Swing Panels bestehen und mit Sanaware Java Docking (javadocking.com) bekommen wir die Möglichkeit, diese Panel frei zu verschieben und neu anzuordnen.
JavaMail API:
Wir nutzen die JavaMail API zum Versenden von Emails. Auf diese Weise können Benutzer z.B. über die Sperrung eines Mediums informiert werden.

2.4 Management persistenter Daten

Die Entitätsklassen müssen persistent gehalten werden. Zu diesen zählen:

Medium
Musikdatei, Genre, Titel, Interpret, Album, Hochladedatum
User
Vorname, Nachname, Geburtsdatum, Ort, Land, Geschlecht, Email, Passwort (verschlüsselt), Beschreibungstext, Flags, hochgeladene Playlists, hochgeladene Medien, hochgeladene Alben, favorisierte Medien, Follower, selbstgeschriebene Kommentare
News
Verfasser, Text, Titel, Verfassungsdatum

Applications
Absender, Empfänger, Text, Verfassungsdatum
Genres
Name, Obergenre, Sub-Genres
Playlist
Medien, Name, Ersteller
Album
Medien, Name, Ersteller
Label
Name, LabelManager, Künstler, Impressum, Follower
Rating
Medium, Bewerter, Punktzahl
Comment
Text, Ersteller, Verfassungsdatum

All diese Daten werden wir in einer Database-Klasse abspeichern, die dann als Fassade für das Datapaket dient.

Persistenzmechanismus:
XStream zum Speichern und Laden von persistenten Daten in .xml Dateien um Datenverlust beim Neustarten des Programms zu verhindern. Die Musikdateien werden in einem Unterordner abgelegt und beim Hochladen umbenannt.

3. Objektentwurf

3.1 Abwägungen des Objektentwurfs

Für den Objektentwurf haben die Anwednungsfälle eine große Rolle gespielt. Im Verlauf der Entwicklung haben wir unter Anderem sichergestellt, dass möglichst alle Anwednungsfälle durch die Entitätsklassen des Objektentwurfs implementiert werden können.
Priorität haben wir dabei auf eine robuste Assoziationverwaltung gelegt. Aus diesem Grund sind alle Entitätsklassen mit Methoden zur Assoziationspflege ausgestattet.

3.2 Klassenmodell der Entitätsklassen

Siehe Klassendiagramm im Anhang (5.2).

3.3 Dokumentation weiterer interessanter Ausschnitte des Entwurfsklassenmodells

Die Klassen Database und Player sind aufgrund ihrer Semantik mit dem Singleton-Muster umgesetzt worden. Auf diese Weise ist sichergestellt, dass zu jedem Zeitpunkt nur eine Database bzw. nur ein Player verwaltet wird.

Die Klassen Player und PlayerPanel realisieren ihren Datenaustausch mithilfe eines Observer-Konstrukts. Diese Umsetzung begünstigt eine geringe Kopplung der beiden Komponenten. Änderungen können so mit geringem Aufwand vorgenommen werden.

Bei der Realisierung der Kommentarfunktion erschien uns die Einführung eines Interfaces 'Commentable' sinnvoll. Durch diese Umsetzung kann gewährleistet werden, dass jeder Kommentar immer nur einer kommentierbaren Entität zugeordnet sein kann.

Die Klasse 'Rating' ist im Klassendiagramm als Assoziationsklasse umgesetzt worden. Um diese Assoziation in Java zu implementieren, verwenden wir in der Klasse 'User' eine HashMap, die als Schlüssel das zu bewertende Medium und als Wert das Ratingobjekt speichert.
In der Klasse Medium wird entsprechend der bewertende 'User' als Schlüssel verwendet. Durch diese Umsetzung ist ein besonders performanter Zugriff auf einzelne Bewertungen möglich.

4. Glossar

Administrator: Ein User mit zusätzlichen Rechten, um die Plattform zu verwalten. Er kann Medien und Alben (ent)sperren und löschen und zusätzlich Profile sperren und entsperren.

Album: Eine Sammlung von Medien ähnlich der Playlist. Sie kann nur direkt von Künstlern bzw. Labels über deren Manager erstellt werden. Ein Album enthält keine doppelten Medien und kann ein Cover haben.

Application: Eine lange Nachricht (16.384 Zeichen) für die Bewerbung eines Artists bei einem Label oder eines Labels bei einem Artist durch den Label-Manager.
Stimmt der Empfänger zu, so gehört der Artist zu dem Label, wird auf dessen Profile angezeigt und der Label-Manager kann Medien im Namen dieses Artists für das Label hochladen.
Ein Artist kann zu mehreren Labels gehören.

Artist: Ein User, der seine eigenen Medien hochladen, News verfassen und mit Labels in Kontakt treten kann. Lädt ein User sein erstes Medium hoch, so wird er automatisch zum Artist.

Comment: Ein kurzer (256 Zeichen) Beitrag, den User zu Medien, News, Albums, Playlists verfassen können. Er wird dort immer öffentlich angezeigt. Zudem hat ein User Einsicht in all seine Comments in seinem Profil.

Database: Hier liegen Verweise auf sämtliche Instanzen der Entitätsklassen, um diese persistent zu halten. Zusätzlich verwaltet sie das Erstellen und Löschen der Instanzen über öffentliche Schnittstellen, so dass eine korrekte Assoziationspflege sichergestellt ist.

Dateiformat: Das Format der Musikdateien für dezibel™.

Empfehlung: Ein oder mehrere Medien, die das System für den Nutzer an Hand einiger Faktoren als potentiell interessant heraussucht und ihm anzeigt.

Fan: Siehe Follower.

Favorite: Ein User kann Artists, Labels, Albums und andere User favorisieren. Dadurch wird der Favorite im Profil des Users angezeigt und der User selbst erscheint als “Follower“ bzw. “Fan“ auf der Seite des Favorites.
Ein User erhält in seinem Profil stets automatisch News und Benachrichtigungen über die Aktivitäten (Medien hinzugefügt, Wiedergabeliste erstellt, neuer Artist beim Label, etc.) seiner Favorites.

Follower: Ein User ist Follower seines Favorites.

Genre: Ein Medium kann genau einem Genre zugewiesen werden, um es zu kategorisieren. Genres können beliebig tief verzweigte Sub-Genres haben, aber nur genau ein Ober-Genre. Zudem existiert zur internen Verarbeitung ein „Top-Genre“, welches, wenn ein Genre kein Ober-Genre haben soll, automatisch als Ober-Genre gesetzt wird.

Label: Ein Profile, welches keinen eigenen User darstellt, sondern durch mindestens einen Label-Manager verwaltet wird. Erstellt ein User ein Label-Profile, so wird er automatisch als dessen Label-Manager eingetragen.
Artists können sich bei Label-Profiles bewerben oder auch von den Label-Managern angeworben werden und werden dann auf dem Label-Profile aufgeführt.
Ein Label-Manager kann Medien der Artists des Labels für das Label-Profile hochladen, Albums aus diesen erstellen und News verfassen.

Label-Manager: Ein User, der seine zugeordneten Labels verwalten kann; also Applications, Medien, News etc.

Medienverwalter: Ein abstrakter User mit den Rechten Medien und Alben zu erstellen und zu verwalten.
Siehe Artist, Label-Manager.

Medium: Die Repräsentation eines Liedes oder einer Ankündigung für ein solches in dezibel™. Ein Medium beinhaltet ggf. die konkrete Musikdatei und Informationen über das Lied in Form der Metadaten. Titel und Interpret sind in jedem Fall zu setzen.
Erstellt ein Artist ein Medium, so gilt er automatisch als Interpret. Erstellt ein Label-Manager ein Medium für ein Label, so hat er aus der Liste der Artists des Labels einen zu wählen.
Fehlt die Musikdatei, so wird das Medium als Ankündigung betrachtet, kann also nicht angehört werden, aber bereits gesucht und Wiedergabelisten und Alben hinzugefügt werden, in denen es dann beim Abspielen übersprungen wird.

Metadaten: Informationen zu Medien an Hand derer sie in der Plattform gesucht werden können. Dazu zählen:
Artist, Album, Erscheinungsdatum, Label, Rating, Titel. Artist und Titel müssen jedem Medium zugewiesen sein.

News: Informationen über Medien, Artists, Veranstaltungen. Artists können sie erstellen und auf ihrem Profil veröffentlichen; Label-Manager auf ihren Label-Profiles.

Playlist: Eine nicht leere Sammlung von Medien, die von Usern erstellt und in deren Profil angezeigt wird, wenn dies gewünscht ist. Nachträgliche Änderungen sind möglich.
Wird eine Playlist angehört, werden die Lieder automatisch nacheinander abgespielt. Nicht verfügbare Medien werden dabei automatisch übersprungen.
Playlists können Comments erhalten, als Favorites gewählt und müssen benannt werden.
Profile: Der Auftritt eines Users oder Labels in dezibel™. Hier stehen Usern all ihre Medien, Wiedergabelisten und persönlichen Angaben stets zur Verfügung. Die News der Label und Artists werden hier auch veröffentlicht.
Profile-Inhalte sind individuell einstellbar öffentlich oder privat, so dass andere User bestimmte Informationen, wie z.B. E-Mail Adresse, Name oder Favorites, auf Wunsch nicht sehen können.

Rating: Die Bewertung eines Mediums.
User können Medien mit ein bis fünf Sternen auszeichnen, wobei fünf die Höchstanzahl ist und das Medium dabei automatisch als Favorite markiert wird.
Nachträglich kann nur die Anzahl der Sterne geändert werden, nicht aber das Rating für ein Medium ganz entfernt werden.
Das Gesamtrating eines Mediums ist das arithmetische Mittel aller seiner Ratings.

Sperrung: Ein Administrator kann Medien, Albums und Profiles sperren; letztere auch mit Begründung.
Ein gesperrtes Medium kann über die Suche noch gefunden werden, ist dort aber als gesperrt markiert und nicht mehr zum Anhören verfügbar. Ist ein gesperrtes Medium Teil einer Playlist, so zeigt der Player dies während der Wiedergabe dieses Mediums an und fährt nach einigen Sekunden automatisch mit dem nächsten Medium in der Liste fort.
Ist ein Album gesperrt, so sind die einzelnen Medien davon unbetroffen. Es kann lediglich nicht mehr direkt abgespielt werden.
Ein gesperrtes Profile kann noch gesucht und aufgerufen werden, auf dem Profile selbst erscheint jedoch ausschließlich die Benachrichtigung über die Sperrung. Ist das Profile eines Users gesperrt, so gilt der User selbst als gesperrt und wird beim Versuch sich anzumelden darüber informiert.

User: Der Endnutzer der Software. Ein User kann gleichzeitig auch beliebig Artist, Label-Manager und Administrator sein. Dies macht gegebenenfalls zusätzliche Funktionen verfügbar.

5. Anhang

5.1 UML Klassendiagramm der Entitätsklassen